

**MIRAGE: A MODEL FOR
LATENCY IN COMMUNICATION**

Joseph Dean Touch

A DISSERTATION
IN
COMPUTER AND INFORMATION SCIENCE

Presented to the Faculties of the University of Pennsylvania in Partial Fulfillment
of the Requirements for the Degree of Doctor of Philosophy.

1992

David J. Farber
Supervisor of Dissertation

Mitchell P. Marcus
Graduate Group Chair

COPYRIGHT

Joseph Dean Touch

1992

DEDICATION

To my parents, Ralph B. and Filomena Touch, for whom education is always first.

ACKNOWLEDGMENTS

This work is supported by the Information Science and Technology Office of the Defense Advanced Research Projects Agency, under contract NAG-2-639, and by an AT&T Graduate Research Fellowship, grant #111349.

There are also many people who contributed to the development of this dissertation, and without whose help it would not have been possible.

My classmates at the University of Pennsylvania Department of Computer and Information Science provided active debates encouraged these ideas. Members of the Distributed Systems Lab, including John Shaffer, Anand Iyengar, Ming-Chit (Ivan) Tam, Brendan Traw, Jonathan Smith, and Amarnath Mukherjee discussed and debated these ideas. Daniel Kulp (Univ. Penn. Materials Science) provided fertile discussions on the physics analogs which provided the basis of Mirage (Appendix B). Other sections were directly helped by Jon Freeman (Appendix C), Christine Nakatani (Appendix G), Amarnath Mukherjee (branching-stream model equations, Chapter 2), Richard Gerber (communicability and Smyth's PowerDomains, Chapter 3), Phil Christie (Univ. Delaware) (violation of the Liouville theorem, Appendix D), and Stuart Frieberg (Univ. Wisconsin - Madison) (*isopotency* of Chapter 2).

This text was edited through the advice of many students and colleagues, including Diana D'Angelo, Ted Faber (Univ. Wisconsin - Madison), Brendan Traw, John Shaffer, and Nick Short, Jr. who provided extensive feedback. I owe Nick more than a few beers.

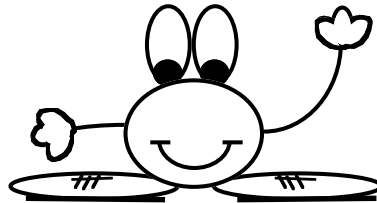
I also want to acknowledge a few of my mentors, most notably my first computer science teacher, John Mero, who was the first to introduce me to computers, and who was the first to give me the opportunity to study them at my own pace. I also would thank my undergraduate advisor Dr. Jack Beidler, who gave me the resources and opportunities to discover this science. Also, my first employer T. Russell Hsing, who is responsible for introducing me to this research, and my Ph.D. advisor, indirectly.

I also thank my dissertation committee members, for their participation, advice, and comments: David Sincoskie (Bell Communications Research), Ravi Sethi (AT&T Bell Labs), Jonathan Smith (Univ. Penn.), and the chair, Mitch Marcus (Univ. Penn.).

I want to thank my Ph.D. advisor, David Farber, for introducing me to the questions this dissertation investigates, for giving me the unique opportunity to investigate a new area of research on my own, and who supported this research to DARPA for a grant.

I also want to thank my friends for ‘moral’ support, including the Doctor’s Club members Samuel Kulp D.V.M. (& Ph.D. 1993), Daniel Kulp (Ph.D. 1992), Scott Miller M.D., David Bradin (Esq., 1994), and Stuart Miller. I thank my friends Janice Luteran, Paula Luteran, Susan Alexander, Fran Weiss, and Nick Short, Jr., especially for contributing to the financial development of telephone companies everywhere. I also thank Kenneth M. Rubinstein, my first professional colleague (cartooning, 1974), and my oldest friend.

Finally, I want to thank my family, whose help and support made this work, and all other I may do, possible. My parents Ralph B. and Filomena Touch (yeah, yeah, 10%, I know), my brother Ralph F. Touch, Esq., my sister Suzanne M. Touch (M.D. 1993), and my grandparents ‘Nonni’ and ‘Pop’ Cianfrani, and ‘Nonni’ and ‘Nonno’ Touch. My father helped edit this dissertation, which was prepared on a Mac IIsi provided by Nonni Cianfrani (more jimmies, Nonni!).



ABSTRACT

MIRAGE: A Model for Latency in Communication

Author: Joseph Dean Touch

Supervisor: David J. Farber

Mirage is an abstract model for the design and analysis of high speed wide area network (WAN) protocols. It examines the effects of latency on communication, and indicates that *information separation* is the distinguishing characteristic of gigabit WANs. Existing protocols will exhibit performance failures due to an inability to accommodate imprecision in the remote state. The name *Mirage* denotes the difficulty with latent communication, namely nodes never really “see” each other precisely; rather, they work with (and around) the *mirages* which high speed and fixed latency conjure before them.

This dissertation describes the Mirage abstract model as an extended finite state machine that accommodates imprecision through the use of multiple simultaneous states and state space volume transformations. We introduce *guarded messages*, to accommodate nondeterministic data streams, and *communicability*, the upper bound on communication, given fixed latency and state predictability. Mirage demonstrates how excess bandwidth can be used to accommodate latency, and shows the bounds on latency constrained communication. Supplemental discussions includes consider Mirage as an extension of Shannon’s communication theory, and compare it to physics analogs.

Mirage was applied to the Network Time Protocol (NTP), to demonstrate its use and exemplify its abstract components. We show the equivalence between variation in state space imprecision and variation in transmission latency. Several ‘optional’ components of the NTP specification are shown to be required, and layering is violated in permitting sender anticipation.

To show the model’s advantages, Mirage was applied to processor - memory interaction as a protocol, calling the result μ -Net (MicroNet). Using anticipation, we develop a novel interface which achieves a hit rate equivalent to that of a 50K byte cache, using 400 bytes of storage. μ -Net complements conventional cache techniques, especially where communication latency is the limiting factor in code execution, and where excess communication bandwidth is available. Dynamic traces measured the latency accommodation possible by the various implementation versions.

TABLE OF CONTENTS

Chapter 1 – Introduction.....	1
1.1. Background.....	2
1.2. What is a protocol?.....	5
1.2.1. Communication as state concurrence.....	7
1.3. The need for a new model	7
1.4. What has changed?.....	9
1.4.1. Change in the use of memory	14
1.5. Model goals	15
1.6. Preview	15
Chapter 2 – The Mirage model	17
2.1. Assumptions	18
2.1.1. Initial description of channel utilization	18
2.1.1.1. Phases	19
2.1.1.2. Channel utilization: linear case.....	20
2.1.1.3. Channel utilization: sender-based anticipation	21
2.1.1.3.1. An example for illustration- the turtle.....	21
2.1.1.3.2. Branching streams utilization	22
2.1.2. Conclusions of the branching model.....	27
2.1.3. Some reality checks	27
2.1.4. Some common sense.....	29
2.2. The Mirage model.....	30
2.2.1. More definitions.....	31
2.2.2. A description of the model.....	32
2.2.2.1. Time	34

2.2.2.2. Receive	35
2.2.2.3. Transmit.....	35
2.2.3. Implications of the model.....	36
2.2.3.1. Lag and stability.....	36
2.2.3.2. Communicability.....	37
2.2.3.3. Guarded messages.....	39
2.2.3.4. Isopotent sets	40
2.3. Discussion.....	41
2.3.1. A channel with imprecision.....	41
2.3.2. Looking into the structure of the stream	42
2.3.3. Implementations	43
2.3.3.1. Projections	43
2.3.3.2. Granularity.....	44
2.4. Insights	45
2.4.1. Kinds of information.....	45
2.4.2. Error and latency as conjugates	45
2.4.3. Entropy.....	46
2.4.4. Constraints.....	46
2.4.5. Contrasts & comparisons	46
Chapter 3 – Prior Work	48
3.1. Prior models of communication.....	49
3.1.1. The models	49
3.1.1.1. Shannon’s model of communication.....	50
3.1.1.2. Communicating finite state machines	51
3.1.1.3. Petri Nets	51
3.1.1.4. Estelle / LOTOS.....	51
3.1.2. Partitioning the state space.....	51
3.1.2.1. Partitions.....	52
3.1.2.2. Factoring.....	52
3.1.2.3. Projections / imaging.....	53
3.1.2.4. Powerdomains.....	53
3.2. Protocol optimizations.....	53
3.2.1. Universal Receiver Protocol.....	54
3.2.2. VMTP.....	55

- 3.2.3. XTP 55
- 3.2.4. TCP 56
- 3.2.5. NetBlt 56
- 3.2.6. ‘Cross Product Protocols’ 57
- 3.2.7. Delta-t 58
- 3.2.8. Virtual Clock 58
- 3.2.9. SNR (leaky bucket) 59
- 3.2.10. TP++ 60
- 3.3. Communicability 61
 - 3.3.1. Cybernetics and control theory 61
 - 3.3.2. Time 62
 - 3.3.2.1. Real time systems 63
 - 3.3.2.2. Aging variables 63
 - 3.3.2.3. Timers 63
 - 3.3.3. Constraints 64
- 3.4. Anticipation 64
 - 3.4.1. Operating systems 65
 - 3.4.1.1. Concurrent execution 65
 - 3.4.1.2. Time Warp 66
 - 3.4.2. Congestion control 66
 - 3.4.2.1. Anticipatory congestion control 66
 - 3.4.2.2. Timer-based congestion control 67
 - 3.4.2.3. Feedback congestion control 67
 - 3.4.2.4. Combination control 68
 - 3.4.3. Other forms of anticipation 68
 - 3.4.3.1. Client / server extensions 68
 - 3.4.3.2. Recent protocol discussions 69
- 3.5. Physics analogs 70
 - 3.5.1. Truth Maintenance Systems 70
 - 3.5.2. Physics in protocols 70

Chapter 4 – A Mirage of NTP	72
4.1. An overview of NTP	73
4.1.1. How NTP reads a clock.....	75
4.1.2. NTP background.....	77
4.2. Casting NTP into Mirage.....	78
4.3. Resolution of domain differences	79
4.3.1. Analysis of delay and offset measurements	82
4.4. Description of NTP in Mirage	88
4.4.1. State space	89
4.4.2. Transformations.....	90
4.4.2.1. Time	90
4.4.2.2. Send.....	92
4.4.2.3. Receive	93
4.4.3. Partitioning of the state space.....	96
4.4.4. NTP degenerates to a clock pulse.....	97
4.4.5. Constraints.....	98
4.5. Observations	98
4.5.1. Gains	99
4.5.2. Prior work.....	100
4.5.3. Conclusions	101
 Chapter 5 – μ-Net	 102
5.1. Preface to Chapters 5 and 6	102
5.2. Introduction.....	104
5.2.1. The Domain - the processor / memory interface	106
5.2.2. Description of current architectures.....	106
5.2.3. Effect of latency on existing architectures	111
5.3. μ -Net.....	114
5.3.1. Time transformations	115

5.3.2. Receive transformations.....	116
5.3.3. Send transformations.....	116
5.3.4. Guarded messages.....	118
5.3.5. Partitioning the state space (stability).....	119
5.3.6. Isopotent sets	119
5.4. μ -Net - Design	120
5.4.1. The Code Pump	120
5.4.2. The Filter Cache	121
5.4.3. Degrees of design.....	122
5.5. Elaboration of degrees of design.....	125
5.5.1. No opcodes anticipated (Null implementation).....	126
5.5.2. Unit Linear opcodes anticipated	127
5.5.3. Linear opcodes anticipated.....	129
5.5.4. Recursion and Linear anticipation	130
5.5.5. Branching and Linear anticipation.....	131
5.5.6. Combining Recursion and Branching anticipation.....	135
5.5.6.1. The TreeStack	136
5.5.6.2. The Total implementation of μ -Net.....	137
5.6. Implications	141
5.7. Conclusions.....	147
Chapter 6 – μ-Net under a μ-Scope	148
6.1. Performance gains.....	150
6.2. On the feasibility of implementations as architectures	157
6.3. Observations	166
6.3.1. Kinds of instructions	166
6.3.1.1. Regular opcodes (OTHER) and JUMPs.....	166
6.3.1.2. BRANCHES	167
6.3.1.3. INDIRECT.....	167
6.3.1.4. CALL and RETURN.....	168
6.3.2. Other observations	169

6.4. Relation to prior work	169
6.4.1. Instruction Issue Logic (Code Pumping)	170
6.4.1.1. Fairchild F8.....	170
6.4.1.2. Distributed Logic Instruction Issue.....	172
6.4.1.3. The Sustained Performance Architecture.....	174
6.4.2. Cache issues.....	176
6.4.2.1. Prediction/prefetching	176
6.4.2.2. Wide lines.....	177
6.4.2.3. Software.....	177
6.4.2.4. Prefetching vs. cacheing.....	178
6.4.2.5. Prefetch v.s pre-reply	179
6.4.2.6. Guarded messages.....	179
6.4.3. Other related architectures.....	180
6.4.3.1. IBM Stretch (7030) - one of the first prefetch.....	180
6.4.3.2. IBM 360/91 - dual prefetch	181
6.4.3.3. Rope multiple prefetch.....	182
6.4.3.4. Access / execute architectures	183
6.4.3.5. IBM RS/6000.....	184
6.4.4. Remote Evaluation.....	184
6.4.5. Multiple alternates	185
6.5. Conclusions.....	185
6.5.1. Notes for designers	186
6.5.2. Notes for researchers.....	187
6.5.2.1. Memory management.....	187
6.5.2.2. Opcode anticipation Amdahl's Law.....	188
6.5.2.3. Flynn's taxonomy	188
Chapter 7 – Conclusions.....	190
7.1. Review.....	190
7.1.1. New questions.....	191
7.1.2. The model.....	191
7.1.3. Existing protocols	192
7.1.4. New protocols.....	192
7.2. Evaluation.....	193
7.2.1. And the answer is.....	193
7.2.2. Is Mirage useful?	195

7.3. Future Directions.....	195
7.3.1. Abstract studies.....	195
7.3.2. Protocol studies (analysis).....	196
7.3.3. Implementation studies (design).....	196
Chapter 8 – Bibliography	197
Appendix A – Mirage & Shannon.....	207
A.1. The channel.....	207
A.2. State transformations.....	208
A.3. Levels of communication	209
A.3.1. Extensions for time	210
A.3.2. Time vs. error.....	211
A.4. Observations	212
Appendix B – Mirage & Physics	213
B.1. Origins of the analogy	213
B.1.1. Field interaction as communication	214
B.1.1.1. Four physical forces	215
B.1.1.2. Communication forces.....	215
B.1.2. Further references.....	215
B.2. Existing analogies from physics.....	216
B.2.1. Entropy	216
B.2.2. Uncertainty.....	217
B.2.3. Hamiltonian function.....	217
B.3. Quantum analogies	218
B.3.1. State sets / multiple worlds.....	218
B.3.2. State set collapse	219
B.3.3. Virtual pairs	220
B.3.4. Feynman path integrals.....	220

B.4. Observations from the analogy	221
B.4.1. Error and latency as conjugates	221
B.4.2. Stability.....	221
Appendix C – Upper Bound.....	224
Appendix D – The Liouville Theorem	226
Appendix E – Mirage in Set Notation.....	228
E.1. Definitions.....	228
E.2. Time Transform.....	229
E.3. Receive Transform.....	231
E.4. Send Transform	232
E.5. Observations.....	232
Appendix F – Mirage & Petri Nets	233
F.1. Petri Net Analogs.....	233
F.1.1 Communication channel	236
F.1.1.1 Basic block	237
F.1.1.2 Virtual tokens	238
F.1.2 Equivalences to Mirage transformations	240
F.1.2.1 Time	240
F.1.2.2 Send.....	241
F.1.2.3 Receive.....	242
F.2. Capacity in a PN	242

Appendix G – The TreeStack	244
G.1. Components	244
G.2. Operations.....	245
G.2.1. Push.....	245
G.2.2. Pop	246
G.2.3. Branch	246
G.2.4. Select subtree	247
G.2.5. Equivalence transforms - Twinning and UnTwinning.....	248
G.2.6. Canonical forms.....	248
G.2.6.1. Twinned TreeStack	249
G.2.6.2. UnTwinned TreeStack.....	249
G.2.7. The Graft transform	250
 Appendix H – μ-Scope Methods	 254
H.1. Existing Tools	254
H.2. The method	257
H.3. Observations	259
H.4. AWK scripts and C-language code listings	260
H.4.1. MSCOPE.h - C-language ‘include’ file	260
H.4.2. INSERT_SYMBOLS - AWK, inserts signals in SPARC assembler	260
H.4.3. INSERT_OPCODES - AWK, signals become SPARC assembler	264
H.4.4. MSCOPE.c code - output desired statistics.....	265
H.4.5. STATIC_COUNT - get static SPARC opcode distributions	267

LIST OF TABLES

Chapter 1 – Introduction.....	1
Table 1.1: Network size and speed equivalences	12
Table 1.2: Network and node characteristics.....	14
Chapter 2 – The Mirage model	17
Chapter 3 – Prior Work	48
Chapter 4 – A Mirage of NTP.....	72
Table 4.1: NTP versions and components.....	78
Table 4.2: Mirage interpretation of the state variables of NTP	89
Chapter 5 – μ-Net	102
Table 5.1: Opcode time transformations	117
Table 5.2: Degrees of implementation, and the implications of each.....	123
Table 5.3: Null Filter send actions	126
Table 5.4: Null Filter receive actions.....	126
Table 5.5: Null Converger actions	127
Table 5.6: Null Diverger actions.....	127
Table 5.7: Unit Linear Filter send actions.....	128
Table 5.8: Unit Linear Diverger actions.....	128
Table 5.9: Linear Filter send actions.....	129
Table 5.10: Linear Diverger actions	130
Table 5.11: Recursion Filter send actions	130
Table 5.12: Recursion Diverger actions.....	131
Table 5.13: Branching Filter send actions.....	133
Table 5.14: Branching Filter receive actions.....	133

Table 5.15: Branching Converger actions.....	134
Table 5.16: Branching Diverger actions	134
Table 5.17: Total Filter send actions.....	138
Table 5.18: Total Filter receive actions (same as Branching filter).....	139
Table 5.19: Total Converger actions.....	139
Table 5.20: Total Diverger actions	140
Table 5.21: CPI (EXEC) values of common CISC and RISC CPUs.....	146
Chapter 6 – μ-Net under a μ-Scope.....	148
Table 6.1: Approximate dynamic opcode distribution.....	152
Table 6.2: Approximate speedup in various degrees of implementation.....	153
Table 6.3: μ -Net implementations and cache equivalents (no branch assumption)	154
Table 6.4: μ -Net implementations & cache equivalents (equiprobable branching)	155
Table 6.5: Adjusted dynamic opcode distributions (approx. a cache miss stream).....	155
Table 6.6: Performance increases where latency dominates design parameters	156
Table 6.7: Mean and median limb lengths for μ -Nets implementing branching.....	162
Chapter 7 – Conclusions.....	190
Chapter 8 – Bibliography	197
Appendix A – Mirage & Shannon.....	207
Table A.1: Weaver’s 3 levels of communication	210
Table A.2: Mirage’s 2 levels of latency	212
Appendix B – Mirage & Physics	213
Table B.1: Physics analogs of protocol components	214
Appendix C – Upper Bound.....	224
Appendix D – The Liouville Theorem	226
Appendix E – Mirage in Set Notation.....	228

Appendix F – Mirage & Petri Nets	233
Appendix G – The TreeStack	244
Appendix H – μ-Scope Methods	254

LIST OF ILLUSTRATIONS

Chapter 1 – Introduction.....	1
Figure 1.1: Network rate	9
Figure 1.2: Increased rate as bit foreshortening.....	10
Figure 1.3: Figure 1.1, as it appears to the bits in transit	10
Figure 1.4: Network size and speed equivalences	12
Figure 1.5: Node buffer size (memory) vs. information separation (bit -latency)	13
Chapter 2 – The Mirage model	17
Figure 2.1: Kinds of protocol lookahead / branching	19
Figure 2.2: Utilization as latency increases (linear lookahead).....	20
Figure 2.3: Tree levels.....	24
Figure 2.4: Utilization of branching lookahead (P=5, D=2, L=4).....	25
Figure 2.5: Channel utilization (relative to linear) (P=5,D=2,L=4).....	26
Figure 2.6: Utilization vs. branch arm length (geometric).....	28
Figure 2.7: Utilization vs. branch degree (geometric)	28
Figure 2.8: Mirage communication channel.....	31
Figure 2.9: Shannon’s model.....	33
Figure 2.10: Visualization of state space volume transformations.....	34
Figure 2.11: Control space evolution.....	37
Figure 2.12: Entire space affected by unguarded message.....	40
Figure 2.13: Guarded messages affecting partitions only	40
Figure 2.14: Bit foreshortening and its effect on lookahead / utilization.....	42
Figure 2.15: Bit foreshortening and branching effecting utilization	43
Chapter 3 – Prior Work	48
Chapter 4 – A Mirage of NTP.....	72

Figure 4.1: NTP message format	74
Figure 4.2: NTP message exchange	75
Figure 4.3: Exchange slid forward (maximum offset)	77
Figure 4.4: Exchange slid backward (minimum offset)	77
Figure 4.5: Unidirectional delay as a Poisson pdf	80
Figure 4.6: Bidirectional delay is the convolution of unidirectional delays	80
Figure 4.7: Measured offset is unidirectional delay convolved with its reverse	80
Figure 4.8: Bidirectional delay as Erlangian (N=2)	81
Figure 4.9: Measured offset as pseudo-Gaussian.	81
Figure 4.10: Probability vs. [delay, offset] pairs, density plot.	81
Figure 4.11: Tuesday offset values (off-peak)	82
Figure 4.12: Friday offset values (peak)	82
Figure 4.13: Tuesday delay values (off-peak)	83
Figure 4.14: Friday delay values (peak)	83
Figure 4.15: Delay v.s offset, time-repetition density (off-peak)	84
Figure 4.16: Delay vs. offset, time-repetition density (peak)	84
Figure 4.17: Delay vs. offset vs. probability, time-series (off-peak)	85
Figure 4.18: Fixed (black) and variable (gray) delay in message exchange	86
Figure 4.19: Exchange maximum offset, variable delay	86
Figure 4.20: Exchange minimum offset, variable delay	86
Figure 4.21: Delay vs. offset, entire ensemble (all strata)	87
Figure 4.22: Stratum 1 ensemble	88
Figure 4.23: Stratum 2 ensemble	88
Figure 4.24: Stratum 3 ensemble	88
Figure 4.25: Stratum 4 ensemble	88
Figure 4.26: Clock value is a function of time	91
Figure 4.27: Clock interval as fixed expansion centered on time function	91
Figure 4.28: Transformation of a perception due to a sent NTP message	92
Figure 4.29: Transformation of a perception due to a received NTP message	94
Figure 4.30: Receive transformation, accommodating transit time effects	95
Figure 4.31: Representation of a Time Warp in the state space timeline	95
Figure 4.32: Partitioning of the state space results in variably ‘tight’ state collapse	96
Figure 4.33: Regular sender anticipation.	98
Chapter 5 – μ-Net	102
Figure 5.1: Mirage extends the channel model to include latency	105

Figure 5.2: Communication channel analog of processor/memory interaction.....	105
Figure 5.3: Explicit processor-memory protocol (voltage/time diagram).....	107
Figure 5.4: Explicit processor-memory protocol (as a protocol time line).....	107
Figure 5.5: Timer based processor-memory protocol (voltage/time diagram).....	108
Figure 5.6: Timer-based processor-memory protocol (as a protocol time line).....	108
Figure 5.7: Processor-memory interaction across a distance	109
Figure 5.8: Processor-memory interaction via a cache	109
Figure 5.9: μ -Net processor-memory interaction	110
Figure 5.10: Protocol timeline comparisons of processor-memory protocols	110
Figure 5.11: CPU state and Memory image	114
Figure 5.12: Detail of Filter Cache and Code Pump of μ -Net.....	121
Figure 5.13: Null Filter Cache design	127
Figure 5.14: Null data space	127
Figure 5.15: Unit Linear Filter Cache design	129
Figure 5.16: Recursion data space	131
Figure 5.17: Branching data space.....	133
Figure 5.18: Branching Filter Cache design.....	133
Figure 5.19: Total Anticipation data space.....	135
Figure 5.20: TreeStack structure.....	136
Figure 5.21: Total Filter Cache design.....	139

Chapter 6 – μ -Net under a μ -Scope.....148

Figure 6.1: Dynamic control opcode distributions	151
Figure 6.2: Percent of CALLs occurring at or above a given depth of recursion.....	157
Figure 6.3: Percent of CALLS not depth-traced (system calls).....	158
Figure 6.4: Average limb length (linearity).....	159
Figure 6.5: Percent increase in limb length (linearity), adding calls and returns	160
Figure 6.6: Dhrystone limb length distribution	161
Figure 6.7: GCC (weighted) limb length distribution.....	161
Figure 6.8: Linpack limb length distribution.....	161
Figure 6.9: T _E X limb length distribution.....	161
Figure 6.10: Mean limb length (L = 7) hit probability vs. BW vs. rtt.....	163
Figure 6.11: Mean limb length (L = 8) hit probability vs. BW vs. rtt.....	163
Figure 6.12: Mean limb length (L = 10) hit probability vs. BW vs. rtt.....	163
Figure 6.12: Mean limb length rtt. vs. utilization	164
Figure 6.13: Median limb length rtt. vs. utilization	164

Figure 6.14: Mean relative performance	164
Figure 6.15: Median relative performance	164
Figure 6.16: Short forward branch opcode sequence.....	165
Chapter 7 – Conclusions.....	190
Chapter 8 – Bibliography	197
Appendix A – Mirage & Shannon.....	207
Figure A.1: Shannon’s communication channel.....	208
Figure A.2: Mirage’s communication channel	208
Figure A.3: State space point transformation	209
Figure A.4: Visualization of state space volume transformations	209
Appendix B – Mirage & Physics	213
Figure B.1: Mirage’s relationship to other sciences	214
Appendix C – Upper Bound	224
Figure C.1: Error between upper bound and exact channel utilization.....	225
Appendix D – The Liouville Theorem	226
Appendix E – Mirage in Set Notation.....	228
Appendix F – Mirage & Petri Nets	233
Figure F.1: Petri Net (unmarked).....	234
Figure F.2: Petri Net markings	234
Figure F.3: Token machine of a Petri Net.....	235
Figure F.4: Meta-Petri Net of a Token Machine	235
Figure F.5: Network MPN partitioned into channels and nodes	236
Figure F.6: Basic block	237
Figure F.7: Token virtualization	238
Figure F.8: Token realization	238
Figure F.9: MPN subgraph (before transform).....	239

Figure F.10: MPN subgraph (after transform).....	240
Figure F.11: Sending introduces virtualization	241
Figure F.12: Reception causes realization.....	242

Appendix G – The TreeStack.....244

Figure G.1: TreeStack components.....	245
Figure G.2: TreeStack Push operation	245
Figure G.3: TreeStack Pop operation.....	246
Figure G.4: TreeStack Branch operation.....	246
Figure G.5: TreeStack Subtree Selection operation.....	247
Figure G.6: TreeStack Twinning and UnTwinning	248
Figure G.7: TreeStack canonical form - maximally Twinned.....	249
Figure G.8: TreeStack form - maximally UnTwinned.....	250
Figure G.9: Multiple Pops in max-Twinned TreeStack (before Pops)	251
Figure G.10: Multiple Pops after Twinning and Pops	251
Figure G.11: Multiple Pops after subsequent UnTwinning	251
Figure G.12: A Graft.....	252
Figure G.13: A Graft Subtree Selection.....	253

Appendix H – μ -Scope Methods254

PREFACE

The following is a section description of this dissertation. It includes the chapters, as well as the appendices. The appendices are auxilliary discussions or digressions, which are intended to indicate all discussions applicable to this work, regardless of development status.

Chapter 1 — Introduction

The introduction defines the problems that Mirage addresses. The original goal was to address anticipated protocol degradation in gigabit networks. We conclude that existing protocols would fail only in gigabit wide area networks, and that fixed latency is the real problem to be addressed. The protocol model developed addresses issues of latency in communication, where latency is fixed and known, and remote state evolves according to known state expansion functions.

Chapter 2 — The Mirage Model

The formal model based on state space subset transformations is presented. We introduce *guarded messages* and *communicability*. The model defines communicability in the presence of latency, and relates stability to communicability and the variability of a remote state. A notation of these transformations based on set notation is presented in Appendix E.

Chapter 3 — Prior Work

The discussion of prior work focuses on cybernetics and control theory, and abstract models of communication, most notably Shannon's theory of communication. Petri Nets, finite state transition models, and temporal logic are also discussed. Other prior work includes distributed systems and databases, especially common knowledge, quorum consensus, and client/server models. Similar protocol methods include VMTP, XTP, NetBlt, virtual clocks, flow protocols, Delta-t, URP, and TP++.

Chapter 4 — A Mirage of the Network Time Protocol

We apply the Mirage model to an existing protocol, the Network Time Protocol, to demonstrate the modeling method and exemplify Mirage's abstract components. This includes demonstrating the isomorphism between variability in state precision and variability in transmission latency, thus extending the domain where Mirage applies. We conclude that several 'optional' components of NTP are required for modeling, and thus should be required in the protocol, e.g., the logical clock and peer dispersion and data filter algorithms.

Chapter 5 — μ -Net

NTP was insufficient to model the unconventional aspects of Mirage, notably those which address latency compensation. We use the Mirage model to develop a processor-memory interface which anticipates opcode memory requests. This interface, called μ -Net (MicroNet), extends the conventional memory interface and is compatible with (and complementary to) a processor opcode cache. There are various degrees of implementation of μ -Net, whose complexity increases as anticipation handles larger subsets of opcodes. μ -Net reduces access latency across the interface, through the use of local storage and (in some cases) higher bandwidth requirements on memory.

Chapter 6 — μ -Net under a μ -Scope

Detailed measurements of opcode executions on a SPARC CPU indicate the effectiveness of the μ -Net designs. These measurements also specify the design parameters and describe the feasibility of the various implementations. For example, anticipating only fixed-jump and recursion opcodes achieves a predictive success rate equivalent to that of a 50K byte cache, with only 100 addresses (i.e., 400 bytes) of storage, with similar memory access load. The measurement tool, μ -Scope, is described in Appendix H.

Chapter 7 — Conclusions

Our goal was to design a model for latency in communication. One initial conclusion was that communication implies latency, and is defined as the sharing of state between temporally separated entities. Mirage has interesting properties in itself and elicits a novel view of existing protocols (NTP) and domains where telecommunication

protocols are not normally applied (μ -Net). Future work includes a more formal comparison to Shannon's work (Appendix A), elaboration of the TreeStack data structure (Appendix G), and a complete implementation of μ -Net.

Chapter 8 — Bibliography

List of references.

Appendix A — Mirage & Shannon

Mirage can also be considered a temporal extension to Shannon's communication theory. In Shannon's work, encoding trades error for latency. Mirage demonstrates a complement of this theory, trading latency for imprecision in state (error). This is a discussion of the ways in which Mirage is an extension to Shannon's theory, and the ways in which it is a complement to it.

Appendix B — Mirage & Physics

Mirage is based on principles from thermodynamics, statistical physics, General Relativity, and quantum physics. This is a discussion of some similarities noticed while designing the model.

Appendix C — Upper bound

In the Mirage model, both discrete and continuous equations for communicability were presented. This is a proof that the continuous equation is indeed an upper bound on the continuous equation, as claimed in Chapter 2.

Appendix D — The Liouville Theorem

The thermodynamic analogs in Mirage indicate an apparent violation of the Liouville Theorem, which restricts the extent to which state can vary over time. Here we explain how information and the open system of Mirage permit such an apparent contradiction.

Appendix E — Mirage in Set Notation

Mirage is described as an extension to state transition models, based on a notation of finite state subset transformations. Bounds are described, based on these equations and existing properties of communication.

Appendix F — Mirage & Petri Nets

Mirage is described as an extension to state transition models, but can be considered in terms of extensions to other models as well. We applied the Mirage principles to timed Petri Nets, and show the Petri Net transformations and equivalences of interesting components of our model.

Appendix G — The TreeStack

The TreeStack data structure is described in detail, as are the mechanisms for transformations required in the implementation of the Converger and Diverger components of the Code Pump of μ -Net (Chapter 5). The TreeStack manages a state space that permits multiple alternates and state space recursion simultaneously. It also reduces to a simple stack if recursion is prohibited, and to a simple tree if multiple alternates are prohibited.

Appendix H — μ -Scope Methods

Examining the feasibility of implementations and specification of the expected speedup required detailed dynamic opcode execution measurements, which were not possible using existing tools, such as PIXIE or SPIXTOOLS. μ -Scope (MicroScope) was developed to make the required measurements on existing compiled code, with an execution speed between 3x and 7x slower than unmeasured code.

In Retrospect

Part of the evaluation of the work should include a description of the changes that would have been considered, had the conclusions been known from the beginning. As a note of comparison, this dissertation does not substantially differ from the dissertation proposal.

We initially intended to examine flow protocols in addition to NTP and μ -Net, but decided that such an analysis would not assist in the description of the components of Mirage. Future research may focus on analyses of these protocols.

The μ -Net research was originally intended to be an investigation into the Mirage model implications on distributed shared memory. We chose processor-memory interaction instead, noting that processor I/O requirements are large enough to require gigabit bandwidths. Conversion of processors to optical pin-outs increases the available bandwidth, but also increases the access latency across the inter-chip boundary, due to the cascaded parallel-serial / serial-parallel conversions. Conventional caching collapses in high latency situations because the cache may be off chip and would incur the same conversion latencies as regular memory. Focusing on distributed shared code memory also permitted the description of temporal transformations sufficient to facilitate communicability. This focus indicates that Harvard architectures may be better suited to anticipation than arbitrary architectures, especially those that permit self-modifying code.

Finally, we note that the μ -Net research results have implications on distributed shared memory, in suggesting similar mechanisms for proactive distributed shared memory. Future investigations may also examine these implications.

CHAPTER 1

Introduction

Mirage is an abstract model for the design and analysis of protocols, for application in high latency domains. The model is an extension of a finite state machine that represents states as sets, rather than as single values. Each node of the network expresses local state by a single point in state space and remote state (the state of a remote node) as a set of possible states.

The model constrains the state space imprecision (i.e., set size), based on the amount of information in transit. It also specifies conditions of stability of a system (i.e., permitting controlled interaction) given the latency and communication bandwidth available.

This is a description of the abstract Mirage model using streams of information and an extended state transition model. The model is applied to the Network Time Protocol [Mi89a], to illustrate the abstract concepts of the model, and as an example of protocol analysis using the model. Mirage is also applied to protocol design, using the domain of processor-memory interaction as a protocol paradigm.

1.1. Background

This dissertation represents a departure from the conventional studies of high speed protocols. An explanation of its origins and evolution into an abstract model may thus be useful.

Various documents and individuals claim that [Gr87], [Ra87], [Po88], [Mi90a], [Pa90a], [Pa90b]:

Protocols will fail at gigabit speeds, requiring clean-sheet approach.

This statement implies that existing protocols will ‘fail’ due to improper design, and that a revolutionary approach will succeed, whereas evolutionary ones will not. Several questions arise from this pronouncement:

Why will they fail? What is so special about these speeds?

How will they fail? What does failure mean?

Failure here is a performance issue. Failure is usually considered a correctness issue, but if a protocol must achieve a certain level of throughput to be correct, then performance is a correctness criterion [St88]. The protocol *fails* to achieve a required level of performance, given the capability of the channel. There are other questions to consider:

When do protocols fail? At what speeds will this become a problem?

Assuming a failure exists, can it be avoided?

Latency, rather than speed, is the real issue. Latency remains constant as communication speeds increase. The result is an increase in the number of bits in transit, i.e., bit latency, between communicating entities; bit latency is information separation.

Nodes are not separated in space; they are ***separated in information***.

Two networks that have the same information separation, yet different operating speeds and spatial distances, can be considered equivalent. The following question remains:

What makes high speed protocols different from low speed protocols?

Nothing has changed except technology. An isomorphism exists between high speed LANs and low speed WANs; thus *newer* (faster LAN) networks can use *older* (slower WAN) protocols. Using this isomorphism, experimental networks without high bit latency can yield methodologies that will not apply to WANs. This has not yet been a problem, because the bit latency in WANs was not large if compared to node buffer size (we will address this later).

A protocol cannot differentiate the relative spatial scales of LANs and WANs, provided the corresponding transmission rates yield identical bit latencies. Alternately:

Is a protocol affected by the speed at which it runs?

No, except for implementation considerations¹. A protocol is affected only by the amount of data in transit; absolute speed has no other affect. Latency is the real issue, that is as bits in transit (bit-latency), rather than in units of time or space alone. Bit-latency is the unit measure of *information separation*.

This discourse leads to one final question:

How can a protocol be characterized to address these questions?

Protocols are often viewed as Petri Nets or finite state machines (FSMs). These models are awkward and inadequate if bit-latency is the determining characteristic. Mirage extends the FSM model to incorporate imprecision of state, thus modeling the effects of latency.

Furthermore, current paradigms (i.e., the ISO stack) model the design structure rather than the design space of protocols. Many protocols are designed by implementation alone. Considering the space of all protocols, only arbitrary points in the space (i.e., instances) are being examined. Each axis of this space is a continuum with tradeoffs.

This space needs to be characterized, but not just for testing existing protocols; the space could suggest new protocols, as new combinations of characteristics of existing protocols. The characteristics of this space of all protocols is inferred by existing instances, but the space has not been well defined. Before examining the characteristics of such a space, there are remaining questions:

¹Such implementation issues can be considerable, but are not substantial. In other words, such issues require attention, but not necessarily new methodologies.

Is there a space of all possible protocols?

Is there a way of examining part of this space in a useful way?

This dissertation provides a way to look at protocols, a model with which to test, design, and measure protocols, i.e., to examine this space in a more general fashion. We will show the following:

- 1) That existing protocols can exhibit low channel utilization in high bit-latency domains.
- 2) The advantages to modeling the endpoints of the link, rather than the channel itself.
- 3) Why the sender should anticipate the receiver.
- 4) How this results in a tradeoff between error and bit-latency.
- 5) Why achieving increased channel utilization necessitates avoiding layered protocols, i.e., why we need to look inside packets.
- 6) There is a limit to how well we can get around things, which is a function of:
 - a) variability in the receiver state
 - b) bit-latency
 - c) power of the sender to accommodate this variability
 - d) ability of the channel to accommodate this variability

We have made two other assumptions here¹, that all protocols exhibit performance failures with high information separation because they are similar, and new protocols can exist which do not fail. To understand the reasons for this claim, and to begin to answer the questions above, we need to start at the beginning:

What *is* a protocol?

¹The other assumptions to this point are that bit-latency is the central issue, and that LAN, MAN, and WAN networks are distinguished primarily by physical scale (i.e., not topology).

1.2. What is a protocol?

The term *communication* is not very easy to define with existing texts. Canonical course textbooks are not helpful; [Ta88] doesn't define it, nor does [Sh63] or [Be87]. A particularly bad example is:

“data transmission” [Ha88a]

This implies that communication is ‘sending bits across a distance.’ Although accurate, this doesn't explain much. Reference texts define communication as:

“exchange of information for the purpose of cooperative action” [St87]

This definition implies that computers have no internal communication. Furthermore, cooperative action is not strictly required; ‘Byzantine generals’ exhibit uncooperative communication [Pe80]. Another reference lists:

“transmission of information from one point to another” [Ja84]

This definition doesn't clarify communication either. English lexicography defines communication as:

“a process by which meanings are exchanged between individuals through a common system of symbols” [Go86]

“the imparting, conveying, or exchange of ideas, knowledge, information” [Si89]

These definitions are more descriptive, but not as general as we desire. Original texts in communication theory define communication as:

“all procedures by which one (entity) can affect another” [Sh63]

This is general, but the term ‘procedures’ is undefined.

We define communication here as:

COMMUNICATION: logically shared state among entities which do not physically share state [To90a]

Sharing of state provides all the earlier definition characteristics as consequences. The separation of the parties is specified by the distinction between *logical* and *physical* sharing of state. Logical separation is an abstraction, used in programming languages

(environment scope), whereas physical separation requires temporal separation, which cannot be abstracted away.

Now that we have arrived at our definition of communication, we are prepared to define a protocol. Again, a particularly bad example is:

“rules and conventions used in a layer-N to layer-N communication, or a set of rules governing the format and meaning of the frames, packets, or messages that are exchanged by the peer entities within a layer” [Ta88]

A better definition incorporates the notion of a protocol as a mechanism that facilitates communication:

“a set of rules formulated to control the exchange of data between two communicating parties” [Ha88a]

The more general definition is:

“agreement between two peer entities on the means of communication” [Sh63]

We prefer the following definition:

PROTOCOL: a method for maintaining shared state among *information separated* entities[To90a]

We define communication as shared state, so that two parties communicate only if they agree on some shared information. Conventionally this state is considered external to the entities (i.e., it is referred to as the state of the channel) , as in Shannon’s theory of communication [Sh63]. The shared state is a portion of the total state of each entity, so the receiver shares the communicated component of the sender’s state. A protocol is a mechanism for maintaining shared state. Because shared state is the basis for communication as we define it, a protocol is thus a method for providing communication.

We make no assumptions about the communicating parties; communication is the abstract process of two entities sharing state. We assume only that communicating entities are necessarily separated in time or space; actually, separation in time is our only criterion, because separation in space implies separation in time¹. If two entities are not separated in time, they are not distinct to the point of requiring communication.

Communication is required if a separation of time exists. Bandwidth is the capacity to transfer encodings across this separation [Sh63]. Latency measures the

¹Given a separation in space, we define the separation in time as the minimum time required for interaction, which is the time it takes information to traverse the spatial separation.

separation; if the latency is zero, there is no separation. Also, because physical separation implies logical separation, we can more generally define communication as:

COMMUNICATION: logically shared state among *information separated* entities

We make no assumptions about the state of an entity (i.e., a node), nor about the state that is shared. A state can be as little as the shared status of a protocol (i.e., current window number, current connection information, etc.), or as much as the contents of an entire file transferred. With as little as one bit of shared state maintained by a protocol, communication is possible (e.g., by an alternating bit protocol).

1.2.1. Communication as state concurrence

Our definition of ‘communication’ is referred to elsewhere in the literature as ‘connection management’, or the shared state which governs the transfer of data. We treat transferred data as the important component of the shared state, rather than “that which shared state facilitates.” A protocol is a mechanism or method for maintaining communication, so what we call a ‘protocol’ others call ‘communication’.

Algorithms and protocols are distinguished by information separation; an algorithm does not include interaction between agents separated by information distance¹. An algorithm that spans an information partition is conventionally called ‘distributed,’ and requires an underlying mechanism for communication among its components, thus distinguishing between the algorithm part and the communication part [Be87]. On an individual node, a protocol is implemented by an algorithm.

1.3. The need for a new model

One characteristic of gigabit wide-area networks that differentiates them from their slower or more proximal counterparts is the unprecedented amount of latent data (‘in the pipe’). We believe that fixed latency (forced by physics), combined with increasing data transmission rates, will result in network inefficiencies as bandwidth and network sizes scale, due to the performance failure of existing protocols [Mi90a], [Pa90b], [Pa90a]. The

¹An algorithm using procedure calls is not a protocol, unless the procedure calls are remote (separated in time from the algorithm). The global variables and passed arguments restrict the scope visible to a procedure, but do not imply a time lag.

Mirage model describes the conditions of this performance failure and helps determine the relevant issues in designing protocols for these new domains.

The name **Mirage** denotes the difficulty with high-speed, wide-area network protocols, in that by the time requested information arrives, it may no longer be accurate. Nodes in a high-speed network never really “see” each other precisely; rather, they work with (and around) the *mirages* which high speed and fixed latency conjure before them.

The predicted performance failure of current protocols in the gigabit wide-area domain has been used to justify the search for new protocol implementations. Most of these efforts focus on the complexity of existing implementations and executing these protocols at gigabit rates, seeking simpler protocols or more efficient implementations (e.g., XTP [Ch88a], NetBlt [Cl87], VMTP [Ch88b]). Instead, we seek to understand the distinguishing characteristic(s) of gigabit, wide-area networks, so protocols developed with this model will work in these domains by design, rather than accommodation.

This research is based on some analogies from physics. Communication theory already incorporates physics analogs, most notably that between information and negative entropy; here we investigate other analogies as well. The Mirage model, of state space volume transformations and guarded messages, is an attempt to incorporate the concept of imprecision evident in quantum models into communication protocol analysis.

We need to determine the salient feature of gigabit, wide-area networks that may prevent existing protocols from operating efficiently. The primary problem is that latency does not scale with speed increases, causing conventional protocols to decrease effective channel utilization, because many of these protocols were designed for file transfer based on sliding-window flow control (e.g., TCP [Po81a], NetBlt [Cl87]). We expect channel utilization to drop as latency increases because existing protocols will not be able to predict the amount of data sufficient to fill the round trip delay at these rates. We will suggest a model where data prediction permits indeterminism, where the round trip time is used to send sets of potentially useful data, rather than only data that was explicitly requested.

The Mirage model describes the effects of latency on communication, permitting the analysis of gigabit wide area network protocols, and showing how increased performance can be achieved. Mirage accounts for latency, but also includes conventional domains as degenerate cases where latency is assumed to be insignificant.

One cause for the deterioration of efficiency in existing protocols is that they use a point model of communication, based on Shannon’s communication theory [Sh63]. This theory accounts for channel error by sequence encoding; higher channel errors requiring encoding over longer sequences. The result is a tradeoff between error and the latency of

encoding. Mirage proposes a view where latency can be tolerated by accepting information imprecision (a measured form of error). Information about remote nodes, formerly precise points in state space, become imprecise volumes in state space. Mirage defines communication operations as transformations of these state space volumes, and incorporates the effects of time in these transformations.

1.4. What has changed?

In assessing the requirements of our new model, we first examine the distinguishing characteristics of the model's domain. Some suggest that existing protocols and protocol models will become inefficient as communication rates increase to gigabit rates [Mi90a], [Pa90a]. There are implementation challenges in scaling protocol processing rate and host bandwidth to accommodate the increased network speeds, but transmission latency does not scale, and cannot as directly be compensated.

In high speed protocols, an 'increase in latency' is usually named as the problem, although latency is a constant. For protocol operation, the important characteristic is information distance, or how many bits separate two communicating entities (bandwidth * delay product). We first show how a gigabit LAN is equivalent to a 400 Kilobit WAN, provided that we treat time as relative rather than absolute.

Changes in communication rates are equivalent to certain changes in scale. Increasing the bit rate foreshortens the bit length as it travels the wire (Figures 1.1, 1.2). Because bits travel at a constant speed, this allows more bits to be in transit at a given time.

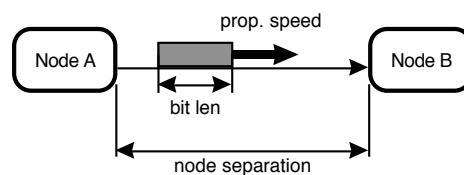


FIGURE 1.1
Network rate

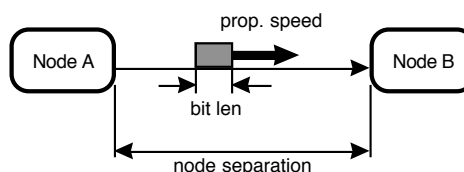


FIGURE 1.2
Increased rate as bit foreshortening

The foreshortening of bits is isomorphic to the original bits traveling faster, over a correspondingly longer distance (Figure 1.3). In this latter view, “*stepping on the gas*” moves the destination further away. Thus latency becomes a problem at high speeds; latency remains constant to external viewers, but distances (and latency) grow in the reference frame of the bit (i.e., ‘bit times’ to the destination)¹.

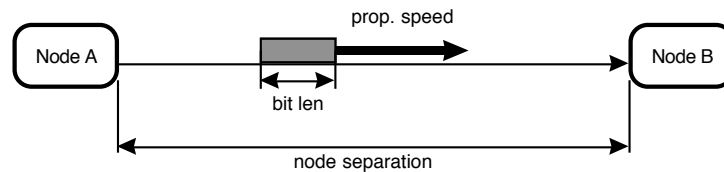


FIGURE 1.3
Figure 1.1, as it appears to the bits in transit

Finally, there is no absolute time reference in these networks, because existing protocols are independent of absolute time. There is no way to distinguish a protocol running on a network from that same protocol running on a network that is 10x larger, and whose data rate is 10x slower. A WAN (2500 miles) can be converted to a LAN (1 mile) by shrinking the network by 1/2500 and increasing the transmission rate equivalently, resulting in the same scale in information distance (bits in the pipe). We can treat a WAN running at 400 Kilobits as a LAN operating at a Gigabit, if we scale it appropriately.

The processing in gigabit LANs is accommodated by a combination of faster technology, increased parallelism, and more efficient algorithms². A 400 Kilobit WAN was supported by TCP by the early 1980’s (1.5 Megabits by 1985), when processor rates were 5 MHz, 16 bits wide. By porting TCP to a very small LAN (two back-to-back processors), on a CRAY Y-MP (167 MHz, 64 bits), technology provided an 130x processor speedup (increased word size and clock rates), and improved processing and limited implementations can support an additional factor of 15x speedup. The result is a nearly 2,000x speedup using these technology advances and optimizations alone, so that

¹This is a relativistic analogy. Information separation space dilation as transmission speed increases is analogous to time dilation as acceleration increases in General Relativity.

²These are order of magnitude arguments only.

the previous 400 Kilobit WAN protocols can be used in LANs at 800 Megabits (800 Megabit TCP has been implemented [Ni91])¹.

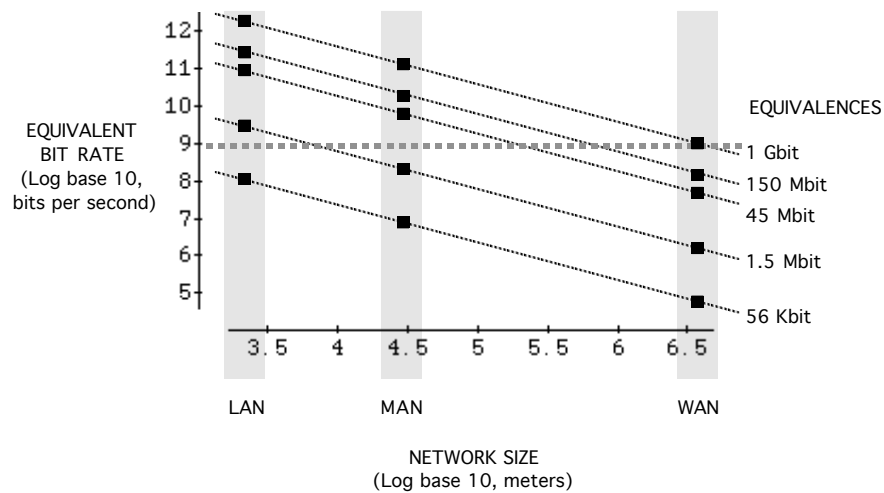
Building gigabit LANs therefore is an issue of scaling processing speed (if possible), not of changing protocol design, disregarding the difficult electronics problems. Nothing has changed in that case, except the time scale relative to the distance scale (propagation latency). Absolute transmission rates indicate little of the characteristic of a network; information separation is a better measure of differences that are not merely technological.

Assuming existing protocols work at current network speeds, MANs need to exceed 133 Gigabits per second, and LANs 2 Terabits per second, before either exhibits the behavior of WANs operating at 1 Gigabit per second (Figure 1.4). A gigabit LAN is equivalent in this sense to a 400 Kbps WAN, where NCP operated -- it is no surprise that, in this environment, very lightweight protocols based on single packet transfer suffice, as they are the modern analog of NCP [Ca70]. Similarly, a gigabit MAN is equivalent to a 10 Mbps WAN, where TCP operates. This assumes that, in each case, we increase the clock rate or processor width of the MAN or LAN to accommodate the change in scale. In these cases, nothing has changed.

Figure 1.4 compares the characteristics of LANs, MANs, and WANs. We assume that scale does not also imply topology. Network type is denoted by vertical gray areas, indicating approximate network scale. The dashed horizontal denotes the gigabit threshold. Actual rates are plotted as points (■), and equivalent rates are connected by gray sloped lines. Latency determines rate equivalence; the slope indicates the contour lines of equal latency, in information separation units (bit-latency, i.e., bandwidth * delay product).

WAN speeds can be compared to those of equivalently bit-latent MANs and LANs (Table 1.1).

¹Test cases were limited to 64K byte 'network' packets, software loopback mode (testing the TCP implementation only, with zero latency), 1.5M byte user packets.

**FIGURE 1.4**Network size and speed equivalences¹

Year	WAN (4,000 Km)	Equiv. MAN (30 Km)	Equiv. LAN (2 Km)
1970	56 Kbps	7.5 Mbps	112 Mbps
1986	1.5 Mbps	200 Mbps	3 Gbps
1990	45 Mbps	6 Gbps	90 Gbps
1995	~150 Mbps ³	~20 Gbps	~300 Gbps
2000	~1 Gbps	~133 Gbps	~2 Tbps

TABLE 1.1

Network size and speed equivalences

So, what has changed by going to 1 Gigabit per second? In the LAN and MAN cases, relatively nothing has changed. Only in the WAN case does speed cause a relative

¹This assumes that LAN, MAN, and WAN networks differ mainly by internodal distance. There may be additional topological implications to these classes; they are not considered here.

²This assumes that LAN, MAN, and WAN networks differ mainly by internodal distance. There may be additional topological implications to these classes; they are not considered here.

³'~' indicates projected estimate.

latency problem; there the bit latency exceeds that of any existing protocol domain, with the possible exception of satellite networks. Unfortunately, satellite protocol models may not be useful as WAN paradigms, because these assume topological constraints (central routing and control) which do not apply in WANs. The real change is the amount of information separation, and it is by this measure that protocols can be characterized, as in Figure 1.4.

The round trip data can be compared to the average size of the node computers in a network¹. Bit-latency had previously been 1-2 orders of magnitude smaller than the node memory, whereas proposed wide-area gigabit networks will cause this gap to narrow considerably (Figure 1.5, from estimates in Table 1.2).

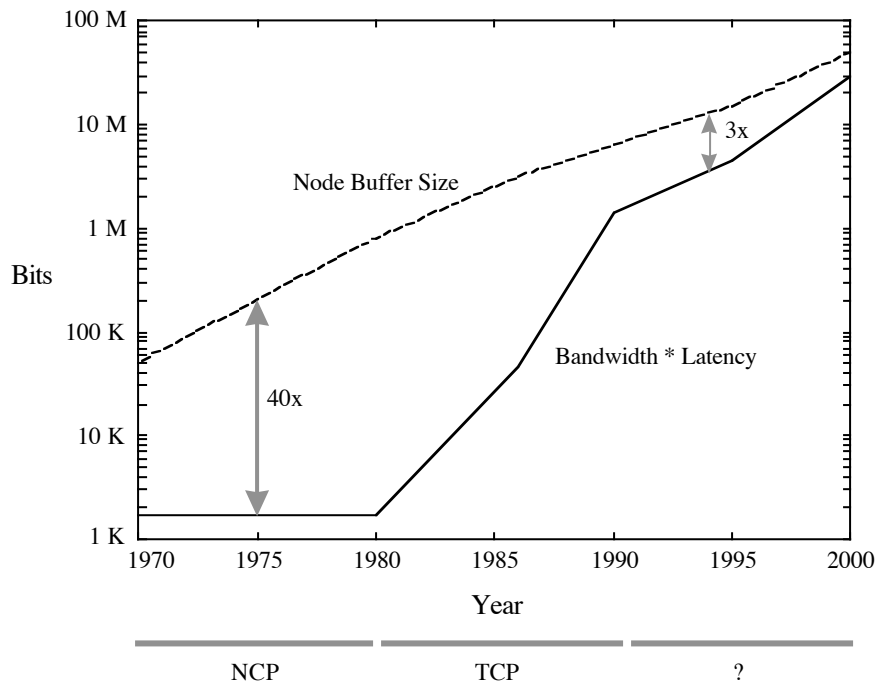


FIGURE 1.5
Node buffer size (memory) vs. information separation (bit -latency)

Year	Bandwidth	BW * Delay	Typical Node	Node Buffer Size	Required Buffer Size

¹Again, these are order of magnitude arguments.

1970	56 Kbit	1.7 Kbit	PDP 11	64 Kbyte	51 Kbit
1980	- ¹	-	VAX	1 Mbyte	800 Kbit
1986	1.5 Mbit	45 Kbit	-	-	-
1990	45 Mbit	1.4 Mbit	Sun 3/4	8 Mbyte	6.4 Mbit
1995	~150 Mbit	~4.5 Mbit	-	~20 Mbyte	~15 Mbit
2000	~1 Gbit	~30 Mbit	-	~64 Mbyte	~50 Mbit

TABLE 1.2Network and node characteristics²

1.4.1. Change in the use of memory

The use of buffer memory has also changed with the advent of high speed protocols. Buffer memory permits restoring altered message sequence within the network, and temporarily archives data for lost message retransmission. In high speed protocols, where latency is large compared to the transmitted information, buffers permit the protocol to both ‘run ahead’ and to amortize control effort over large chunks of data.

These uses of buffers by protocols assume that communication exists to facilitate file transfers. We are expecting a change in the use of the channel, where file transfer is superseded by interactive communication. In this realm, buffers cannot be used by the protocol to accommodate latency, because the protocol can no longer ‘run ahead’ in sending data. The data will no longer be a linear stream of packets, so the protocol cannot anticipate the data to fill the buffers, and the channel utilization plummets.

Large buffers need to be used for more than just linear sequences of data. We extend their use to accommodate sets of data, where only one item of the set is used. In

¹‘-’ indicates ‘no value’.

²Bandwidth*delay values are based on a 30ms speed-of-light propagation across the continental United States; buffer sizes are based on 1/10 * node buffer size, converted to bits.

³Bandwidth*delay values are based on a 30ms speed-of-light propagation across the continental United States; buffer sizes are based on 1/10 * node buffer size, converted to bits.

this way, branching in the data stream, i.e., imprecision in the protocol, can be used to increase the utilization of the channel.

The notion of a ‘channel with imprecision’ will be further elaborated when the Mirage model is described. At this point, it is sufficient to have shown the need for a new model, one that accommodates latency in a natural way, rather than as an extension of an existing model.

1.5. Model goals

We have several goals for this protocol model. Before discussing them, we should first outline our assumptions. We assume a network with no topological or routing restrictions, and where the bit-latency is high compared to the streams of data which are communicated. Current gigabit WANs (were one to exist) satisfy these conditions, where communication is dominated by tightly coupled interaction of the components of the network.

We introduce a model where the tradeoff between error and bit-latency can be expressed. We will trade imprecision for latency, so that constraints on the communication can be used to increase the utilization of the channel and more tightly couple the communicating parties. We will also show the uses of the increased bandwidth to compensate for the effects of latency, so that the detrimental effects of bit-latency which are increased by high-speed channels can be reduced by that same high-speed. We will also show the limitations of this compensation.

1.6. Preview

Mirage is an abstract model, which can be used to understand the issues in protocol design, or can measure the implications of a specific design decision. We will investigate these two roles of the model in two studies presented as part of this dissertation.

First, we look at the Network Time Protocol, and examine its operation using the Mirage model. We show how Mirage can indicate important aspects of the protocol, and new methods that are being proposed to augment the protocol.

We also apply the Mirage model to the domain of processor/memory interaction as a communication protocol. We show how Mirage provides a fresh view of this interaction, and indicates new solutions, to which the literature has only recently alluded.

Further, Mirage shows ways to measure the various implementations indicated, from the perspective of both channel utilization and design complexity.

We also include a description of the model as channel stream interactions, state space transformations, and Petri Net equivalences (for those who find this more helpful). The section on prior work has been restricted to discussions of predecessors to the abstract model. Similar discussions on prior work in time protocols and processor/memory interaction are included in their respective sections.