I4 Reading Group, March 18, 2016

Jerome H. Saltzer, David P. Reed, and David D. Clark
**End-To-End Arguments in System Design**
*ACM Transactions on Computer Systems (TOCS)*, Volume 2, Number 4, November 1984, Pages 277-288.

Timo Hönig
`https://www4.cs.fau.de/~thoenig`

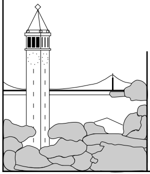
# End-To-End Arguments in System Design

- Scope
  - system design principles
  - distributed systems
  - post time-sharing era
- Authors
  - Jerome H. Saltzer (MIT): Multics, Kerberos
  - David P. Reed (MIT): TCP/IP, UDP
  - David D. Clark (MIT): Multics, Internet protocols
- Journal: ACM Transactions on Computer Systems (TOCS)
  - first issue: February 1983
  - current issue: TOCS Volume 33, Issue 4, January 2016
  - average citations per article: 45.14

# Context, Energy-Aware Computing

**Software Strategies for Portable Computer Energy Management**

*Jacob R. Lorch* *Alan Jay Smith*



**Report No. UCB/CSD-97-949**

May 1997

Computer Science Division (EECS)
University of California
Berkeley, California 94720

Software Strategies for Portable Computer Energy Management*

Jacob R. Lorch† Alan Jay Smith†

May 30, 1997


**Abstract**

Limiting the energy consumption of computers, especially portables, is becoming increasingly important. Thus, new energy-saving computer components and architectures have been and continue to be developed. Many architectural features have both high performance and low power modes, with the mode selection under software control. The problem is to minimize energy consumption while not significantly impacting the effective performance. We group the software control issues as follows: *transition*, *load-change*, and *adaptation*. The transition problem is deciding when to switch to low-power, reduced-functionality modes. The load-change problem is determining how to modify the load on a component so that it can make further use of its low-power modes. The adaptation problem is how to create software that allows components to be used in novel, power-saving ways. We survey implemented and proposed solutions to software energy management issues created by existing and suggested hardware innovations.


## 1 Introduction

Limiting energy consumption has become an important aspect of modern computing. The most important reason for this is the growing use of portable computers, which have limited battery capacities. Another reason is that high energy consumption by desktop computers translates into heat, fan noise, and expense. One way to reduce energy consumption is to simply use components that consume less power. Another way is to use components that can enter low-power modes by temporarily reducing their speed or functionality. This paper will discuss the software problems that arise from such hardware features, and what solutions have been proposed to deal with these issues. The aim of this paper is not to discuss hardware techniques for reducing power, but to discuss software techniques for taking advantage of low-power hardware that has already been designed.

| Category | Description |
|---|---|
| Transition | When should a component switch between modes? |
| Load-change | How can a component's functionality needs be modified so it can be put in low-power modes more often? |
| Adaptation | How can software permit novel, power-saving uses of components? |

Table 1: Categories of energy-related software problems

We classify the software problems created by power-saving hardware features into three categories: transition, load-change, and adaptation. The transition problem involves answering the question, "When should a component switch from one mode to another?" The load-change problem involves answering the question, "How can the functionality needed from a component be modified so that it can more often be put into low-power modes?" The adaptation problem involves answering the question, "How can software be modified to permit novel, power-saving uses of components?" Each of the software strategies we will consider addresses one or more of these problems.

Different components have different energy consumption and performance characteristics, so it is generally appropriate to have a separate energy management strategy for each such component. Thus in this paper we will generally consider each component separately. For each component, first we will discuss its particular hardware characteristics, then we will discuss what transition, load-change, and adaptation solutions have been proposed for that component. The components whose software power management problems are most significant are the secondary storage unit, the processing unit, the wireless communication unit, and the display unit, but we will also briefly discuss other components.


*This material is based upon work supported in part by the National Science Foundation under grants MIP-9116578 and CCR-9117028, by NASA under Grant NCC 2-550, by the State of California under the MICRO program, and by Apple Computer Corporation, Intel Corporation, Sun Microsystems, Fujitsu Microelectronics, Toshiba Corporation, and Sony Research Laboratories.

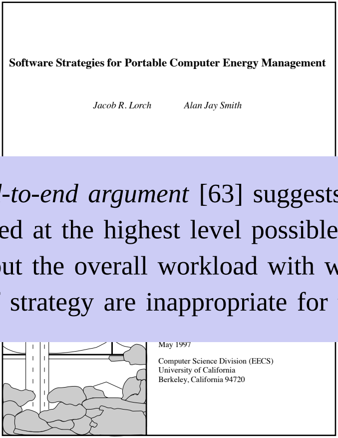


†Computer Science Division, EECS Department, University of California, Berkeley, California, 94720-1776




- Jacob R. Lorch and Alan J. Smith
  **Software Strategies for Portable Computer Energy Management**
  *IEEE Personal Communications*, Volume 5, Number 3, June 1998, Pages 60-73.

# Context, Energy-Aware Computing

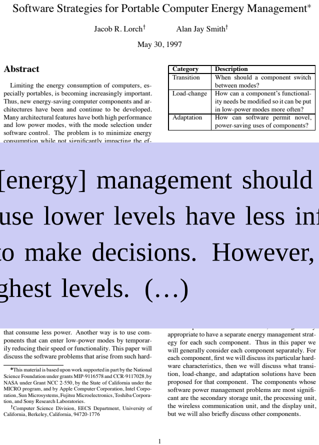


**Software Strategies for Portable Computer Energy Management**

*Jacob R. Lorch* *Alan Jay Smith*

May 1997

Computer Science Division (EECS)
University of California
Berkeley, California 94720



Software Strategies for Portable Computer Energy Management*

Jacob R. Lorch† Alan Jay Smith†

May 30, 1997

**Abstract**

Limiting the energy consumption of computers, especially portables, is becoming increasingly important. Thus, new energy-saving computer components and architectures have been and continue to be developed. Many architectural features have both high performance and low power modes, with the mode selection under software control. The problem is to minimize energy consumption while not significantly impacting the ef-

| Category | Description |
|---|---|
| Transition | When should a component switch between modes? |
| Load-change | How can a component's functionality needs be modified so it can be put in low-power modes more often? |
| Adaptation | How can software permit novel, power-saving uses of components? |

that consume less power. Another way is to use components that can enter low-power modes by temporarily reducing their speed or functionality. This paper will discuss the software problems that arise from such hard-

appropriate to have a separate energy management strategy for each such component. Thus in this paper we will generally consider each component separately. For each component, first we will discuss its particular hardware characteristics, then we will discuss what transition, load-change, and adaptation solutions have been proposed for that component. The components whose software power management problems are most significant are the secondary storage unit, the processing unit, the wireless communication unit, and the display unit, but we will also briefly discuss other components.

*This material is based upon work supported in part by the National Science Foundation under grants MIP-9116578 and CCR-9117028, by NASA under Grant NCC 2-550, by the State of California under the MICRO program, and by Apple Computer Corporation, Intel Corporation, Sun Microsystems, Fujitsu Microelectronics, Toshiba Corporation, and Sony Research Laboratories.

†Computer Science Division, EECS Department, University of California, Berkeley, California, 94720-1776



The *end-to-end argument* [63] suggests that [energy] management should be performed at the highest level possible, because lower levels have less information about the overall workload with which to make decisions. However, certain types of strategy are inappropriate for the highest levels. (…)

- Jacob R. Lorch and Alan J. Smith
  **Software Strategies for Portable Computer Energy Management**
  *IEEE Personal Communications*, Volume 5, Number 3, June 1998, Pages 60-73.

# End-To-End Arguments in System Design

- Problem Statement
  - *where* to implement functional units of a system design
  - *which* is the correct level of abstraction (low-level vs. high-level)
- Design Considerations
  - communication protocols
  - distributed system design
- Core Statement

> The function in question can completely and correctly be implemented *only* with the knowledge and help of the application standing at the endpoints of the communication system. (…) We call this line of reasoning against low-level function implementation the end-to-end argument.

# End-To-End Arguments in System Design — Discussion

## End-To-End Arguments in System Design

J. H. SALTZER, D. P. REED, and D. D. CLARK
Massachusetts Institute of Technology Laboratory for Computer Science

This paper presents a design principle that helps guide placement of functions among the modules of a distributed computer system. The principle, called the end-to-end argument, suggests that functions placed at low levels of a system may be redundant or of little value when compared with the cost of providing them at that low level. Examples discussed in the paper include bit-error recovery, security using encryption, duplicate message suppression, recovery from system crashes, and delivery acknowledgment. Low-level mechanisms to support these functions are justified only as performance enhancements.

CR Categories and Subject Descriptors: C.0 [**General**] Computer System Organization—*system architectures*; C.2.2 [**Computer-Communication Networks**]: Network Protocols—*protocol architecture*; C.2.4 [**Computer-Communication Networks**]: Distributed Systems; D.4.7 [**Operating Systems**]: Organization and Design—*distributed systems*

General Terms: Design

Additional Key Words and Phrases: Data communication, protocol design, design principles

### 1. INTRODUCTION

Choosing the proper boundaries between functions is perhaps the primary activity of the computer system designer. Design principles that provide guidance in this choice of function placement are among the most important tools of a system designer. This paper discusses one class of function placement argument that has been used for many years with neither explicit recognition nor much conviction. However, the emergence of the data communication network as a computer system component has sharpened this line of function placement argument by making more apparent the situations in which and the reasons why it applies. This paper articulates the argument explicitly, so as to examine its nature and to see how general it really is. The argument appeals to application requirements and provides a rationale for moving a function upward in a layered system closer to the application that uses the function. We begin by considering the communication network version of the argument.

This is a revised version of a paper adapted from End-to-End Arguments in System Design by J. H. Saltzer, D.P. Reed, and D.D. Clark from the 2nd International Conference on Distributed Systems (Paris, France, April 8–10) 1981, pp. 509–512. © IEEE 1981
This research was supported in part by the Advanced Research Projects Agency of the U.S. Department of Defense and monitored by the Office of Naval Research under contract N00014-75-C-0661.
Authors' address: J. H. Saltzer and D. D. Clark, M.I.T. Laboratory for Computer Science, 545 Technology Square, Cambridge, MA 02139. D. P. Reed, Software Arts, Inc., 27 Mica Lane, Wellesley, MA 02181.





In a system that includes communications, one usually draws a modular boundary around the communication subsystem and defines a firm interface between it and the rest of the system. When doing so, it becomes apparent that there is a list of functions each of which might be implemented in any of several ways: by the communication subsystem, by its client, as a joint venture, or perhaps redundantly, each doing its own version. In reasoning about this choice, the requirements of the application provide the basis for the following class of arguments:

*The function in question can completely and correctly be implemented only with the knowledge and help of the application standing at the endpoints of the communication system. Therefore, providing that questioned function as a feature of the communication system itself is not possible. (Sometimes an incomplete version of the function provided by the communication system may be useful as a performance enhancement.)*

We call this line of reasoning against low-level function implementation the *end-to-end argument.* The following sections examine the end-to-end argument in detail, first with a case study of a typical example in which it is used—the function in question is reliable data transmission—and then by exhibiting the range of functions to which the same argument can be applied. For the case of the data communication system, this range includes encryption, duplicate message detection, message sequencing, guaranteed message delivery, detecting host crashes, and delivery receipts. In a broader context, the argument seems to apply to many other functions of a computer operating system, including its file system. Examination of this broader context will be easier, however, if we first consider the more specific data communication context.

### 2. CAREFUL FILE TRANSFER

#### 2.1 End-to-End Caretaking

Consider the problem of *careful file transfer.* A file is stored by a file system in the disk storage of computer A. Computer A is linked by a data communication network with computer B, which also has a file system and a disk store. The object is to move the file from computer A's storage to computer B's storage without damage, keeping in mind that failures can occur at various points along the way. The application program in this case is the file transfer program, part of which runs at host A and part at host B. In order to discuss the possible threats to the file's integrity in this transaction, let us assume that the following specific steps are involved:

(1) At host A the file transfer program calls upon the file system to read the file from the disk, where it resides on several tracks, and the file system passes it to the file transfer program in fixed-size blocks chosen to be disk format independent.
(2) Also at host A, the file transfer program asks the data communication system to transmit the file using some communication protocol that involves splitting the data into packets. The packet size is typically different from the file block size and the disk track size.



- Jerome H. Saltzer, David P. Reed, and David D. Clark
**End-To-End Arguments in System Design**
*ACM Transactions on Computer Systems (TOCS)*, Vol. 2, Nr. 4, Nov. 1984, Pages 277-288.

# End-To-End Arguments in System Design — Remarks

- Influences beyond Academia
  - Architectural Principles of the Internet (RFC 1958, June 1996)
    `https://www.ietf.org/rfc/rfc1958.txt`

    Call for the implementation of end-to-end *protocols* for realising end-to-end *functions*.
- Coincidence?
  - The Twelve Networking Truths (RFC 1925, **1.** April 1996)
    `https://www.ietf.org/rfc/rfc1925.txt`

    Fundamental truths of networking for the Internet community.

```
(6) It is easier to move a problem around (for example, by
moving the problem to a different part of the overall network
architecture) than it is to solve it.

(6a) (corollary). It is always possible to add another level of
indirection.
```

# Further Reading


[1] Donald W. Davies, Keith A. Bartlett, Roger A. Scantlebury, and Peter T. Wilkinson
**A Digital Communication Network for Computers Giving Rapid Response at Remote Terminals**
*Proceedings of the 1$^{st}$ ACM Symposium on Operating System Principles (SOSP)*, 1967.

[2] Marjory S. Blumenthal and David D. Clark
**Rethinking the Design of the Internet: The End-to-End Arguments vs. the Brave New World**
*ACM Transactions on Internet Technology (TOIT), Volume 1, Number 1*, 2001.

[3] Tim Moors
**A Critical Review of "End-to-End Arguments in System Design"**
*Proceedings of the 12$^{th}$ IEEE International Conference on Communications (ICC)*, 2002.

[4] Kenneth L. Calvert, W. Keith Edwards, and Rebecca E. Grinter
**Moving Toward the Middle: The Case Against the End-to-End Argument in Home Networking**
*Proceedings of the 6$^{th}$ ACM Workshop on Hot Topics in Networks (HotNets)*, 2007.

[5] David P. Reed
**End-to-End Arguments: The Internet and Beyond**
*Proceedings of the 19$^{th}$ USENIX Security Symposium (Invited Talk)*, 2010.