Jerome H. Saltzer, David P. Reed, and David D. Clark
End-To-End Arguments in System Design

Timo Höning
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
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 temporar-ily reducing their speed or functionality. This paper will discuss the software problems that arise from such hard-

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 hard-

ware 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. We will also briefly discuss other components.

Table 1: Categories of energy-related software problems

<table>
<thead>
<tr>
<th>Category</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transition</td>
<td>When should a component switch between modes?</td>
</tr>
<tr>
<td>Load-change</td>
<td>How can the component’s functionality needs be modified so it can be put in low-power modes more often?</td>
</tr>
<tr>
<td>Adaptation</td>
<td>How can software permit novel, power-saving uses of components?</td>
</tr>
</tbody>
</table>

1. This material is based upon work supported in part by the National Science Foundation under grants CCR-9116578 and CCF-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.

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

This paper discusses novel, power-saving ways to use components that can enter low-power modes by temporarily reducing their speed or functionality.
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. (…)

Table 1: Categories of energy-related software problems

<table>
<thead>
<tr>
<th>Category</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transition</td>
<td>When should a component switch between modes?</td>
</tr>
<tr>
<td>Load-change</td>
<td>How can a component’s functionality needs be modified so it can be put in low-power modes more often?</td>
</tr>
<tr>
<td>Adaptation</td>
<td>How can software permit novel, power-saving uses of components?</td>
</tr>
</tbody>
</table>

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.

We classify the software problems created by power-saving hardware features into three categories: transition, load-change, and adaptation. Each category raises different issues created by existing and suggested hardware innovations. 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 it can be put in low-power, reduced-functionality modes?" The adaptation problem is how to 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 modify the load on a component so that it can make further use of its low-power modes. The load-change problem involves answering the question, "How can software permit novel, power-saving uses of components?"

Each component, first we will discuss its particular hardware characteristics, then we will discuss its particular hardware characteristics, then we will discuss what software power management problems are most significant for each component. This 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 software power management problems are most significant for each 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. However, certain types of strategy are inappropriate for the highest levels. (…)

Software Strategies for Portable Computer Energy Management

Jacob R. Lorch and Alan J. Smith

Abstract

Limiting the energy consumption of computers, especially portable computers, is becoming increasingly important. Thus, new energy-saving computer components and architectures have been and continue to be developed. However, certain software strategies 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 created by existing and suggested hardware innovations as follows:

- Effective performance.
- Consuming less power.
- Not significantly impacting the effective performance.

We will generally consider each component separately. For each component, first we will discuss its particular hardware characteristics, then we will discuss what software power management problems are most significant for each 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. However, certain types of strategy are inappropriate for the highest levels. (…)

Thomas A. Loughlin and Alan Jay 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.
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 with integrity, 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

End-To-End Arguments in System Design — Remarks

▶ Influences beyond Academia
  ▶ Architectural Principles of the Internet (RFC 1958, June 1996)

  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

[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”

Moving Toward the Middle: The Case Against the End-to-End Argument in Home Networking

[5] David P. Reed
End-to-End Arguments: The Internet and Beyond