End-to-End Reasoning

Layering RPC transport on TCP rather than UDP

- **does not simplify exactly-once implementation**
- **worsens performance in best case**
- **does simplify at-most once implementation**

This is an example of an *end-to-end argument*

- **structuring principle for distributed systems**
- **helps designer place functionality**

For given functionality

- ***correctness requires support of end points***
- ***support below end points cannot suffice***
- **at best, lower-level support may improve performance**

Optional Reading

(on course web site)

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

---

ACM Transactions on Computer Systems, Vol. 2, No. 4, November 1984

Statement of the Principle

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.)

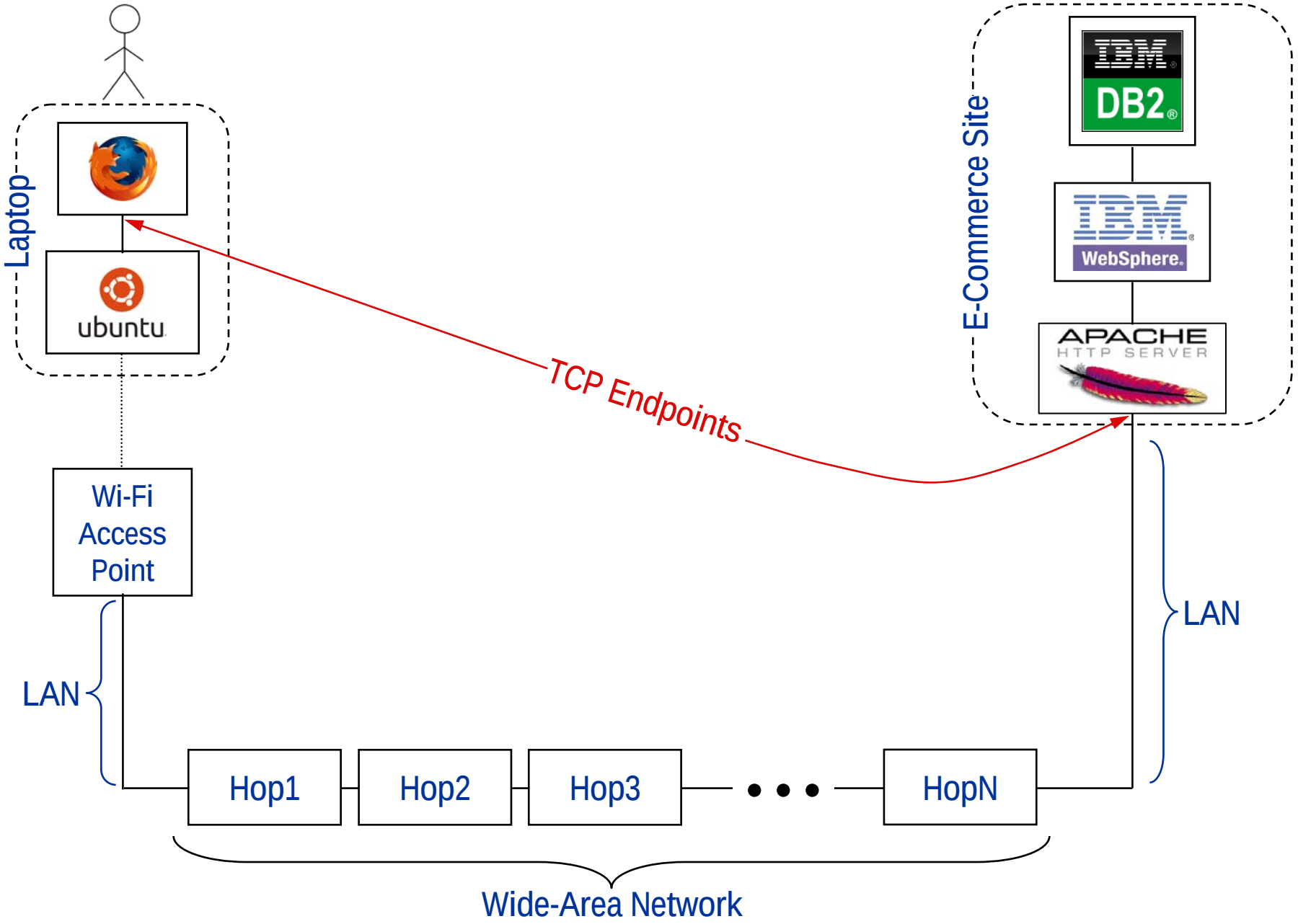
We call this line of reasoning against low-level function implementation the *end-to-end argument*. The following sections examine the end-to-end argument

Answers this critical question:

Where should I place a specific function in my distributed system?

AKA

Who should have responsibility?


Laptop
ubuntu
Wi-Fi Access Point
LAN
TCP Endpoints
E-Commerce Site
IBM DB2
IBM WebSphere
APACHE HTTP SERVER
LAN
Hop1
Hop2
Hop3
HopN
Wide-Area Network

Example: Large File Transfer

Consider transfer of a large file

- **network unreliable**
- **remote processor and software unreliable**

Method 1

- **ship entire file via TCP (which has ACKs in its implementation)**
- **write file to disk at remote site**
- **crash can occur during disk write**
- **hence TCP's guarantees count for little**
- **higher level confirmation needed**
- **ship back file (or checksum) and compare**

Method 2

- blast entire file via UDP
- write file to disk at remote site
- read back file, blast back file (or checksum) via UDP to first site
- compare original file with returned file (or compare checksums)
- match guarantees successful transmission!

Benefit of Method1

- *only missing parts of file need to be reshipped*
- results in better performance on flaky links
- in other words, *better worst-case performance*