End-to-end Arguments in System Design

Saltzer, Reed and Clark, ACM TOCS, Vol.2, No. 4, Nov. 1984

Ερώτημα

Πώς αποφασίζουμε σε ποιές οντότητες-δομοστοιχεία θα τοποθετήσουμε την λειτουργικότητα που θέλουμε να υπάρχει σε ένα σύστημα πελάτη-εξυπηρετητή;

Introduction

- Choosing the proper boundaries between functions is a primary activity of the computer system designer.
- In network-centric systems, the argument of function placement is sharpened, making more apparent the situations in which and the reasons why it applies:
  - Modular boundary around comm. subsystem
  - Firm interface between comm. subsystem and the rest of the system.
- 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.

Introduction

- Where do we implement the system functionality:
  - In the communication subsystem
  - In the clients of the comm. sub.
  - As a joint venture
  - Redundantly

End-to-end argument

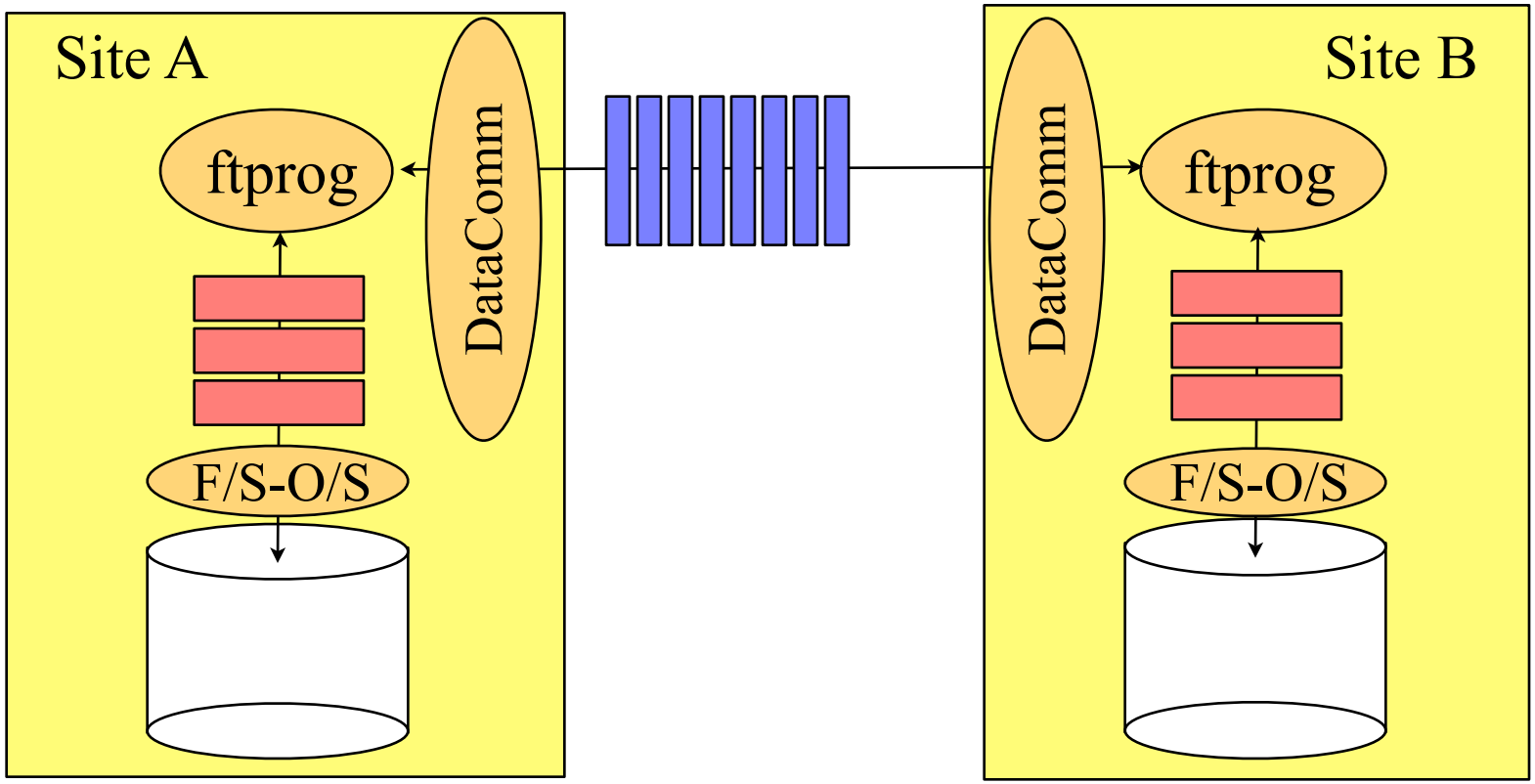
- The function in question can be completely and correctly be implemented only with the knowledge and help of the application standing at the endpoints of the communication system.
- Therefore, providing that questioned function as a feature of the communication subsystem itself is not possible.
- *Sometimes,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

Careful File Transfer

- Move file from computer A to computer B without damage.
- Steps taken:

1. At host A, the file transfer program calls the file system to read the file from disk. The f/s passes the file to the file trans. program in fixed-sized blocks chosen to be disk format independent.
2. At host A, the ftprog 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.
3. The data communication network moves packets from A to B.
4. At host B, the data communication program removes packets from the protocol and hands the contained data to a second part of the data transfer application operating on B.
5. At host B the file transf program asks the file system to write the received data on the disk of host B.

Careful File Transfer



Threats to transaction

1. Hardware faults in the disk storage result to *reading* incorrect data.
2. File system software or file transfer program or data communication system make a mistake in buffering and copying the data of the file either at A or B.
3. Hardware processor or local memory have transient error while doing buffering and copying at A or B.
4. Communication system drops or changes bits in a packet or deliver a packet more than once.
5. Either of the hosts may crash part way through the transaction after performing an unknown amount of the transaction.

Dealing with threats

- Reinforce each of the steps along the way using duplicate copies, time-out and retry, carefully located redundancy for error detection, crash recovery, etc.
- Systematic countering of threat (2) requires writing correct programs…
  - Not all programs are written by the file-transfer programmer.
- Using tri-modular redundancy for the whole process…

End-to-end Check and Retry

- Suppose that a checksum is stored with every file, reducing the possibility of error to an acceptably negligible value.
- Then, transfer file from A to B.
- FT Application at B reads the file back to its memory, computes the checksum and sends the value back to A for comparison.
- If comparison fails, retry…

What if?

- The communication subsystem provides internally a guarantee for reliable data transmission, through:
  - Selective redundancy in the form of packet checksums
  - Sequence number checking
  - Internal retry mechanisms
- We can lower the probability of dropped bits to a very small number, and eliminate threat (4).
- Henceforth, we achieve a reduction of the frequency of retries by the file transfer application.
- What is the effect on the correctness of the file-transfer outcome?
  - It may reduce the frequency of end-to-end retries
  - No effect on inevitability or correctness of the outcome

Conclusion

- To achieve careful file transfer, the application program that performs the transfer must supply a file-transfer-specific, end-to-end reliability guarantee.
- For the communication subsystem to go out of its way to be extraordinarily reliable does not reduce the burden on the application program to ensure reliability.

Performance Aspects

- So, should lower levels play no part in obtaining reliability?
- Consider a somewhat unreliable network, dropping a message in each hundred messages sent.
- What is the effect of this, as we transmit files of increasing size?
  - The probability that all packets of a file arrive correctly decreases exponentially with the file length (**prove this**).
  - So, the expected time to transmit the file grows **exponentially** with the file length.
- Performance of the file transfer application hurts!

Performance Trade offs

- The amount of effort to be put into reliability measures within the data communication system is an *engineering trade-off* (αντιζύγισμα) based on performance, rather than a requirement for correctness.
- If the communication subsystem is too unreliable, the file transfer application performance suffers.
- If the communication subsystem is beefed up with internal reliability measures, those measures have a performance cost:
  - Lost bandwidth to redundant data
  - Added delay from waiting for internal consistency checks to complete
  - And, after all, the end-to-end consistency check is still required, no matter how reliable the communication system becomes.

Performance Trade offs

- Using performance to justify placing functions in a low-level subsystem must be done carefully.
- Sometimes, the same or better performance can be achieved at the high level.
- Performing the function at the low level may:
  - be more efficient if the function can be performed with minimum perturbation of the machinery already included in the low-level.
  - cost more because:
    - Most applications using the low-level subsystem do not need the function.
    - The low-level subsystem does not have adequate information, like the higher levels, to do the job efficiently.

Other Examples of the e2e Argument

- Acknowledgment of message delivery:
  - The comm network can easily return an ack to the sender, whenever a message is delivered to a recipient.
  - This is not very helpful for applications, since:
    - The application wants to know if the target host *acted* on the message (receipt does not directly translate to action).
    - So, the appl. needs end-to-end ack.

Other Examples of the e2e Argument

- Secure transmission of data: if the data communication system performs encryption-decryption:
  - It will need to manage the keys.
  - Data will be vulnerable while passing from the communication system to the application.
  - Authenticity of the message must still be checked by the application. If the application performs the encryption, it can also do the authentication checks and keep its keys.
  - Is the encryption of data by the communication subsystem necessary?

Other Examples of the e2e Argument

- Transaction Manager (SWALLOW system)
  - Provides data storage servers (repositories) that can be used remotely to store and retrieve data.
  - Access to repository is done via messages specifying: object to be accessed, version, type of access (r/w), value to be written if access is a write.
- Does the communication subsystem need to suppress duplicate messages or provide acks?
  - The object identifier suffices to detect duplicate writes
  - The effect of duplicate read-requests is to generate duplicate responses
  - Acknowledgment needed for a write request is that data is stored safely; this ack cannot be provided by the communication subsystem

Identifying the ends

- The end-to-end argument is *not* an absolute rule but rather a guideline:
- Suppose we use a communication subsystem to send voice packets:
  - To support two people carrying a real-time conversation.
  - To transport a speech message.
- What are the choices we have?

Rethinking the Design of the Internet: The End-to-End Argument vs the Brave New World

Blumenthal and Clark, ACM TOIT, Vol.1, No. 1, Aug. 2001

Placement of Application-specific functions

- Moving app-specific functions out of the **core** of the network and providing only general-purpose system services there
- Why?
  - Complexity of the core is reduced - reducing cost, improving performance, facilitating future upgrades to the network
  - Generality in the network increases the chances that a new application can be added without having to change the core
  - Applications do not have to depend on the successful implementation and operation of application-specific services in the network (better reliability)

Moving away from End-to-End

- Operation in an untrustworthy world: original assumption was that end-points are in willing cooperation to achieve their goals
  - Attacks on the network, on individual end-points, undesirable forms of interaction, annoyances
    - Making the network more trustworthy while end-points cannot be trusted, seems to imply more mechanisms at the core to enforce "good" behavior
- More demanding applications:
  - "Best effort" delivery makes no guarantees about the throughput of any particular application
    - This is not enough in applications not tolerant to throughput fluctuations (streaming audio & video) - need to install intermediate storage sites positioning the content close to the recipient.

Moving away from End-to-End (ctd)

- ISP service differentiation
  - There is an acceleration in the deployment of applications based on intermediate servers that can be positioned within each ISP, and allow for differentiation of service quality inside islands of enhanced service
  - Check the **Net Neutrality** debate and recent FCC regulations
- The rise of third-party involvement
  - Demand by third parties to interpose themselves between communicating end-points, irrespective of the desires of the ends - officials of organizations, government, public safety
  - End-to-end arguments do not provide an obvious framework to reason about this situation. What do we do?

Moving away from End-to-End (ctd)

- Less sophisticated users
  - The implication that substantial software is present at the end-node is that end-to-end arguments become a source of complexity to the user (software must be installed, configured, maintained, upgraded etc)
  - This complexity can be addressed if configuration, protection, and control are moved away from the end-point to a common point, which can act as an agent for a pool of devices.
  - This common point would become a part of the application execution context
    - with this approach, there would be no longer **a single indivisible end-point** where the application runs

End-to-end arguments philosophy

- End-to-end arguments foster the "Internet philosophy":

freedom of actions, user empowerment, end-user responsibility for actions undertaken, and lack of controls "in" the Net that limit or regulate what users can do.

- End-to-end arguments foster that philosophy because they enabled the freedom to innovate, install new software at will, and run applications of the user's choice.

Requirements in todays' communications

- Users communicate but don't trust:
  - Two parties want to negotiate a binding contract: they may need symmetric proof of signing, protection from repudiation of contract etc
  - One party needs external confirmation of who the other party in the communication is
  - One party wants to preserve its anonymity
    - How can we track one's identity on the network?
- End-parties distrust their software and hardware
  - Web browsers store "cookies" and send them back to the same or different servers to provide a trail that links successive transactions
  - Processors may contain unique identifiers
  - Local network interfaces contain unique identifiers

Requirements in todays' communications

- The Ends vs. the Middle: 3-rd party rights
  - Government wiretapping certain communications inside their jurisdiction / spying outside their jurisdiction
  - Governments take the right to control the access of certain parties to certain material
  - Governments assert their right to participate in certain actions undertaken by their citizens for public policy reasons (enforcement of taxation on transactions)
  - Private ISPs assert their right to regulate traffic on their networks in the interests of managing load and segregating users with different intentions
  - Private parties assert their right to intervene in certain actions across the network to protect their rights (e.g. copyright) in the material being transferred.

Requirements in todays' communications

- One party forces interaction on another:
  - Application-level flooding with unwanted material (e.g., email spam)
  - Security attacks: penetration of computers with malicious intent (Trojan horses)
- Multiway communication
  - Either implemented through number of separate two-party communications at the network level
  - Through multicasting at the network level (teleconferencing)
  - What should be done when one of the parties seems to be failing or malicious?

Why is it complex?

- End-to-end arguments suggest that:
  - a willing sender can use any software he chooses to transfer material to willing receivers
  - each end-node is responsible for protecting itself from attacks by others (see antiviruses)
- However:
  - Holders of IPR may assert that they have the right to interpose themselves into that transfer to protect their rights (and ability to collect fees), which thus potentially becomes a network issue!
  - End-node responsibility may not be sufficient in today's world
- Thus: need mechanisms for:
  - Third parties to inject themselves into the communication
  - End parties to avoid intervention
- Third-party objectives trigger requirements for traffic analysis:
  - looking at IP addresses and other high-level identifying information describing communication

Technical Responses

- Different forms of End-to-End Arguments
  - End-to-end arguments applying to the **core** of the network (routers): providing basic data-forwarding service (elements that are “in” the network)
  - E-to-e arguments applying to elements **attached to** or **on** the network
    - From the perspective of the core, all devices and services attached to the network represent end-points, regardless of where they are (end-user site, ISP facilities)
- Modify the End-Node
  - E.g. filtering pornographic material
- Adding functions to the Core:
  - Firewalls, traffic filters, Network address translation boxes (NAT)

Design Issues: adding mechanisms to the Core

- Imposing a Control Element:
  - there is no single place in the Internet where a control point can be interposed in an unspecified flow
  - for a ***known*** *flow with a given source or destination* there is often an accessible location at which to insert a control point (where?)
- Revealing or Hiding the Content:
  - if a control point is installed and monitors the traffic passing through this, which aspects of the information are visible to the control device?
  - from totally visible to totally masked (encryption)
- Information Labeling (e.g. PICS standard)
  - A way of revealing some information about the content of a message

Design of Applications

- Two main trends in application design:
  - Different parties (end-users or network ops) wish to insert some sort of intermediary into the data path of an application not initially designed with this structure
  - Application requirements become more complex, leading away from end-to-end design
- Examples:
  - Anonymizing message forwarders
  - Helpful content filtering
  - Content caching
  - Using trusted third parties:
    - a mutually trusted third party, located somewhere on the network creates a context in which a 2-party transaction can be carried out successfully (e.g. Certification Authorities)

The Larger Context

- Non-technical solutions: the rule of law
  - Mechanisms outside the Net, such as law, regulation, or social pressure, restrain third parties that turn our to be untrustworthy, systems that do not protect one's identity as promised and so on.
  - Typically, legal mechanisms come into play after a violation; technical mechanisms result to a behavior that can be predictable a priori.
- The Internet is a system where technology rather than law is the force most immediately shaping behavior, and until the legal environment matures, there are comparatively fewer options for remedy after the fact in cyberspace than in real space.

Where we are today?

- Rise of New Players:
  - ISPs, who implement the core of the networks
- The Erosion of Trust
  - The simple model of the early Internet - a group of mutually trusting users attached to a transparent network - is gone forever
  - We need a more sophisticated view of trust and how it relates to other factors such as privacy, openness, and utility.
  - As trust erodes, both end-points and third parties may wish to interpose intermediate elements into a communication to achieve verification and control.
    - This leads to a tension between the need for devices to examine at least part of a data stream and the growing tendency of users & their software to encrypt communication streams for data integrity and privacy protection

Where we are today?

- Most critical tension between rights and responsibilities is that between anonymity and accountability.
- End-to-end arguments suggest that end-points can communicate as they please, without constraint from the network