End-To-End Arguments in System Design

J. H. Saltzer, D. P. Reed and D. D. Clark

Presented by: S. M. Nazrul Alam

Introduction

- Classic paper of 1981.
- A design principle to guide placement of functions in a distributed computer system
- Macro-level issues than specific protocols or implementation tricks
- Widely influential on placement of function in a layered system.

End-To-End Arguments

- In a modular system, functions can often be implemented in different sub systems
  - By network, client, joint venture, redundantly.
- Choosing the proper boundaries between functions is critical for computer system designer.
- Appropriate design principles are very valuable.

End-To-End Arguments

- Many functions require the knowledge only the applications have.
  - Impossible to implement in low level *completely and correctly.*
- End-to-end arguments
  - Reasoning against low-level function implementations.

Large System Design

- Black box
  - Many designers
  - Each designer's responsibility
- Two key issues
  - Correctness
  - Efficiency

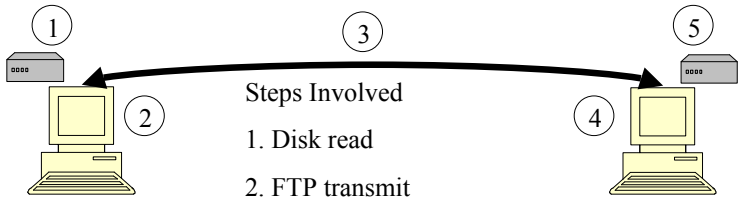
Key questions

- How to decompose the complex system functionality into protocol layers?
- Which functions placed *where* in network, at which layers?
- Can a function be placed at multiple levels ?

Internet End-to-End Argument

- "…functions placed at the lower levels may be *redundant* or of *little value* when compared to the cost of providing them at the lower level…"
- "…sometimes an *incomplete* version of the function provided by the communication system (lower levels) may be useful as a *performance enhancement*…"

Case Study: Careful File Transfer



Steps Involved

1. Disk read
2. FTP transmit
3. Network carries packets
4. FTP receive
5. Disk write

Careful File Transfer- Threats

1. Incorrect disk read
2. Software bug (mistake in buffering and copying)
3. Hardware glitch (while doing buffering and copying)
4. Dropped, duplicate or mutilated packets.
5. Host crash

Careful File Transfer- Coping Strategies

- Reinforce each of the steps
  - Duplicate copies
  - Time-out and retry
  - Carefully located redundancy in error detection, crash recovery etc.
- Difficult for threat (2) – Writing correct program!
- Brute force counter measure- doing everything three times!
  - Uneconomical

Careful File Transfer- Coping Strategies (cont.)

- End-to-End check (i.e., checksum) and retry
  - Useful if failures are fairly rare.
- Reliable data transmission by communication systems.
  - Threat (4) may be eliminated
  - Other threats still remains.

Does Reliable Communication Systems Help?

- Does it reduce the frequency of retries of the file transfer system?
  - YES!
- Does it ensure overall correctness?
  - NO!
- End to end reliability guarantee still required.

A Too-Real Example

- At MIT, a packet checksum was used on each hop from one gateway to the next.
  - Assumption: Primary threat to correct communication is bit error during transmission.
- Application programmers did not provide E2E reliability guarantee.
- One gateway computer developed a transient error while copying.
- Results: some files were corrupted.

Performance Aspects

- Low levels should play no role in obtaining reliability
  - Too simplistic!
- Example:
  - Assume high error rate in network
  - Reliable communication service at data link layer might help (why)?
  - Fast detection/recovery of errors

Trade-offs

- Some support at low level is very important.
- Lower level need not provide *perfect* reliability.
- Trade off based on *performance.*

'Proper trade-off requires careful thought'

Arguments Against Low Level Implementation

- Lower level subsystem is common to many application -> applications that do not need the function have to pay.
- Lower level subsystem may not have as much information as the higher levels -> inefficient

File Transfer – Early Retry

- Can be done by either
  - Communication subsystem
  - Application

Placing Early retry at Application Level

- Pros
  - Simplifies communication system
- Cons
  - Other applications must now provide its own reliability enhancement.

Placing Early retry at Communication System

- Pros
  - May be more efficient
  - Fast error detection/correction, since done on a hop-by-hop basis.
- Cons
  - Some applications that require less reliability may find cost (delay etc.) higher.

Choice must be made intelligently

Other End-To-End Examples

- Delivery Guarantees
- Secure Transmission of Data
- Duplicate Message Suppression
- Guaranteeing FIFO Message Delivery
- Transaction Management

Delivery Guarantees

- Acknowledgement at network level is not sufficient.
- E2E acknowledgement is required for most cases.

Secure Transmission of Data

- Network may not be trusted to securely manage the encryption keys.
- Data will be clear and vulnerable between target node and target application.
- *Authenticity* of the data must still be checked by the application.
- No need for automatic encryption of all traffic by network!

Secure Transmission of Data

- Automatic encryption of all data at network level may be useful to
  - Prevent information leakage from the system
- Encryption by application and network is complementary!

Duplicate Message Suppression

- Even if the network suppresses duplicates, the application still need to take care.
  - To prevent application level duplication.
  - e.g. remote system user puzzled by lack of response initiates a new login to a time-sharing system.

Guaranteeing FIFO Message Delivery

- Consider a distributed application
  - One node initiates request
  - The request initiates actions at several sites
  - Application level mechanism required to guarantee correct ordering of actions.

Transaction Management

- SWALLOW distributed data storage system
  - Repositories for remote storage and retrieval of data
    - To access it requires: message specifying
      - Object to be accessed
      - Version number
      - Type of access (read/write)
      - Value to be written (in case of write)

Transaction Management

- No low level suppression of duplicate message required
  - Object identifier and version suffice to detect duplicate write
  - Duplicate read is not harmful
- Result: simple low-level message communication protocol

Transaction Management

- No low level acknowledgement required
  - Acknowledgement for write can only be provided by the high levels of SWALLOW system
  - Acknowledgement for read is redundant. Response containing the value read is sufficient.
- Message reduction improves performance significantly.

Identifying the Ends

- Two people in real-time conversation vs. a speech message system
- Minimize delay vs. maximize accuracy.

Application to Other System Areas

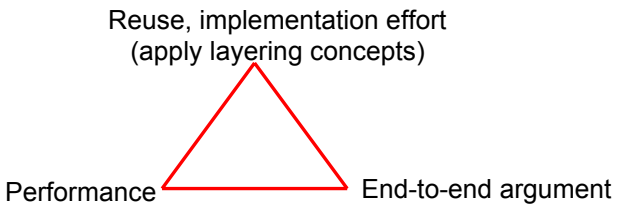
- Banking system
- Airline reservation system
- RISC
- Lampson's Open OS

Conclusion

- E2E arguments are a kind of *Occam's Razor* for systems design.
- Designers of lower level should remember the *KISS* principal.

Conclusions

Challenge of building a good (network) system: find the right balance between:



- No universal answer: the answer depends on the goals and assumptions!

The Design Philosophy of The DARPA Internet Protocols

David D. Clark

Introduction

- Why is TCP/IP as it is?
- Internet was designed to achieve some predetermined goals.
- Evolution of design philosophy.
- In the first proposal
  - Idea of datagram did not receive particular emphasis
  - No TCP and IP layers.

Fundamental Goal

- Multiplexed utilization of existing interconnected networks *effectively*.
- Integrating separately administered entities.
  - Packet switching is chosen over circuit switching.
  - Networks would be connected by a layer of Internet packet switches called Gateways.

Fundamental Structure of the Internet

"a packet switched communications facility in which a number of distinguishable networks are connected together using packet communications processors called gateways which implement a store and forward packet forwarding algorithm."

Second Level of Goals

In order of importance

- Survivability in the face of failure.
- Support multiple types of service.
- Accommodate varieties of networks.
- Distributed management of the resources
- Cost effective
- Permit host attachment with a low effort.
- Accountability.

Why This Particular Ordering?

- Military context.
  - Survivability first
  - Accountability last
- Commercial context would have given more importance to accountability.
- Not a *motherhood* list, but a set of priorities.

Goal # 1: Survivability

- Communication should continue after a failure without re-establishing or resetting high level state
- Synchronization would never be lost unless a total physical partition

Survivability: How to Achieve?

- State information of ongoing conversation must be protected.
- Two ways
  - Replication
  - Fate-sharing

Replication

- State is stored in the intermediate packet switching nodes of the network.
- To protect the information loss, must be replicated.
- Protect against a certain number of intermediate failures <= replicated copies
- Hard to engineer, robust algorithm is difficult to build.

Fate-Sharing

- Save the state at the end-points of the network, at the entities which are utilizing the service of the network
- Assumption
  - It is safe to lose state information when the entity itself is lost.

Adv. of Fate-Sharing

- Protects against any number of intermediate failures.
- Much easier to engineer than replication

E2E Arguments Prevail

- Fate-sharing was chosen.
- More trust is placed in the host machine than in an architecture where networks ensures the reliable delivery of data.
- Intermediate packet switching nodes, gateways, have no essential state information of connection.

Interconnection of Existing Networks- Topmost goal

- A more survivable technology is a single multimedia network design.
- Weak assumptions about the ability of a network to report failure->
  - Detection of network failures using Internet level mechanism.
  - Slower and less specific error detection.

Goal#2: Support For Multiple Types of Service

- Types of Service
  - based on speed, latency, reliability.
- *Virtual circuit* service
  - Bi-directional reliable delivery of data
  - First service provided in the Internet using TCP
  - Appropriate for remote login or file transfer.

TCP to Provide Services

- Initial concept: TCP could be general enough to support any service
  - Remote login requires low delay but accepts low bandwidth.
  - File transfer require high throughput but less concern about delay
  - TCP attempted to provide both types of services

Separation of TCP and IP

- TCP not suitable for services that has to work even when reliable services cannot be provided.
  - Example: XNET – cross internet debugger
  - Real time delivery of digitized speech.
- So TCP and IP were separated into two layers.
  - Evolution from first proposal.

TCP and IP

- TCP provides one particular type of service
  - The reliable sequenced data stream.
- IP provides a basic building block called *datagram* for a variety of services.
  - Reliability not guaranteed
  - But *best effort*
- UDP provides a application level interface to the basic datagram service of the Internet.

Challenges for Multiple Types of Service

- Difficult to achieve without explicit support from the underlying networks.
- Networks designed for one service may not be flexible enough to support other services.
  - X.25 designed for reliability is not much flexible

Goal #3: Varieties of Networks

- Incorporates and utilizes a wide variety of network technologies
  - Long haul nets (the ARPANET and various X.25 networks)
  - Local are nets (Ethernet, ringnet, etc.)
  - Broadcast satellite nets
    - DARPA Satellite Networks-64 Kbps and 3Mbps
  - Packet radio networks
    - DARPA, British and amateur packet radio network
  - A variety of serial Links
  - A variety of other ad hoc facilities

Goal #3: Varieties of Networks

- Achieves flexibility by makes a minimum set of assumptions about the networks
  - Network can transport a packet or datagram
  - Packet should be delivered with good but not perfect reliability.
  - Network should have some form of addressing.
  - Other services can be built on top of these at endpoints

Goal#4–Distributed Management : Achieved

Goal partially met

- Networks connected by gateways that are independently managed.
- 2 level routing hierarchy which permits gateways from different organizations to exchange routing tables.

Goal#4–Distributed Management : Required

- Lack of sufficient tools for distributed management
  - Routing decision need to be constrained by resource usage policies.
  - Can be done in a very limited way, requires manual settings of tables
- Development of a new generation of tools for resource management in the context of multiple administrations are very important.

Goal #5 – Cost Effectiveness

Needs more work

- Small data packets have large overheads.
- Inefficiency due to retransmission of lost packets end to end.
  - Recovery at the network level more cost effective.

Goal #5 – Cost Effectiveness

- Rough rule of thumb
  - 1 loss in 100 packets is reasonable
  - 1 loss in 10 packets suggests reliability enhancement is required for the network

Goal #6 – Easy Attachment of host

- Due to Fate Sharing approach,
  - All of the mechanisms to provide must be implemented in the host rather than in the network.
  - Cost of attaching a host is somewhat higher.
- Misbehaving hosts hurt robustness.

Goal #7 - Accountability

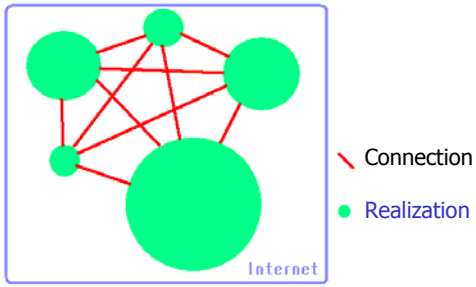
- Few tools for accounting for packet flows.
- Only currently being studied
  - As the scope of the architecture is being expanded to include non-military consumers.

Architecture & Implementation

- Set of services offered is determined by
  - Not the architecture
  - Actual engineering of software within particular hosts and gateways and related networks.

Realization

- Particular set of networks, gateways and hosts which have been connected in the context of Internet Architecture.



Realization

- Wide variability in *Realizations* by way of performance and characteristics.
- The Internet Architecture tolerates the variety by design.

Guidance for Designer of Realization

- The designer needs to know
  - What sort of bandwidth required to deliver a throughput at a certain rate?
  - Given a model of possible failures, what sort of redundancy is required?
- Available design aids can't help!

Protocol Verifiers

- Attended only to Logical Correctness, Ignored the issue of performance.
- No help for severe problems related to performance.
- Difficulty may arise in the OS!
- Good implementor guidance is needed.

Simulator

- Takes a particular realization and explores possible services under various loads.
- No simulator exists that considers wide variability of
  - gateway implementation
  - Host implementation
  - Network performance

Issue of Performance

- Difficult to formalize any aspect of performance constraint
  - Goal of the architecture was to permit variability, not to constrain performance
  - Tools for describing performance effectively don't exist.

Issue of Performance

- Military contractors do not meet any criteria not in the specification
- Hard to set performance criteria in the architecture.
- Individual performing the procurement must specify the performance constraints.

Datagrams

- Eliminate need for connection state within the intermediate switching nodes
  - Internet can be reconstituted after failure without concern about state.
- Basic building block which can implement various services
- minimum network service assumption -> allowed integration of network

Datagrams

- Myth:
  - Motivation for datagram is support for higher level services similar to datagram.
- Fact
  - Role of datagram is as a building block, not as a service in itself.

TCP

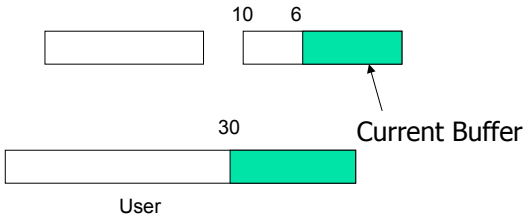
- Original ARPANET provided flow-control based on both bytes and packets
- Designers of TCP thought above approach was complex -> chose one form of regulation : bytes
  - For insertion of control information
  - To allow packets to be broken up into smaller packets
  - To allow gathering of small packets into one big packet (for retransmission)
- Correct design may have been to use both packets and bytes for the control, just as ARPANET did.

EOL Flag in TCP

- Original idea – break byte stream into records.
- Different records => different packets…incompatible with the idea of combining packets on retransmission.
- Hence semantics was changed - "one or more"
- Various application had to invent a way of delimiting records on top of the data stream.

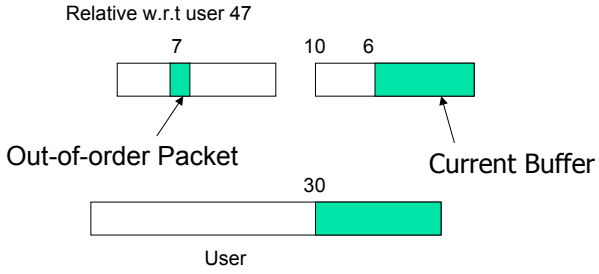
EOL-Intermediate Form (1/4)

- Host uses a fixed size buffer
- Return the buffer to user when it is full or EOL is received.



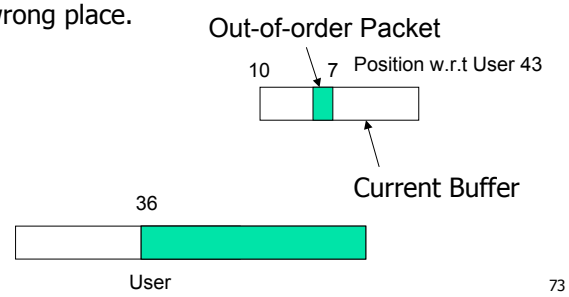
EOL-Intermediate Form (2/4)

- Arrival of out-of-order packet lie beyond current buffer.



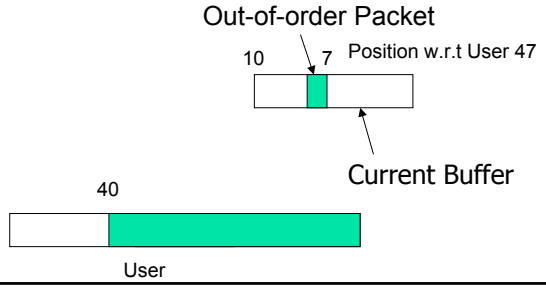
EOL-Intermediate Form (3/4)

- Then a packet with EOL arrives, partially full current buffer return to users.
- The out-of-order packet in the next buffer placed in the wrong place.



EOL-Intermediate Form (4/4)

- Solution: EOL should use up all the sequence space up to the next value which is zero mode the buffer size.



Observation

- Incorporate into TCP some means of relating the sequence space and the buffer management algorithm of the host might be a good idea.

Conclusion

- Internet Architecture has been very successful – widely used in both commercial and military environment
- More attention needed at
  - Accounting
  - Resource Management
  - Operations of regions with separate administrators
- Datagram not helpful for accounting and resource management.
  - Use building block *flow* and gateways keep *soft state* of flows.