Internet Design Principles

Protocol Design

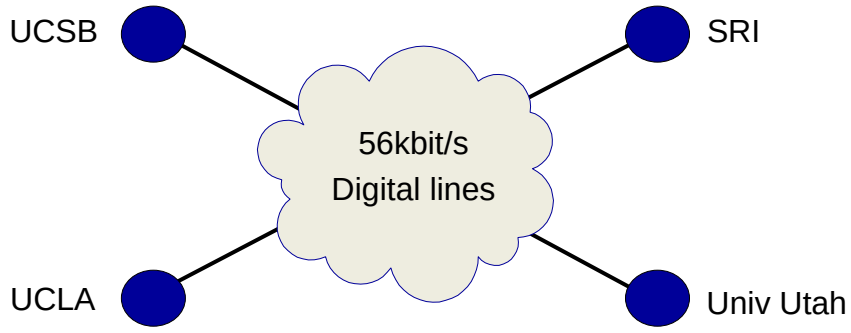
Origin of Packet Switching

- Concepts show up in the early 1960s
- Devised by multiple, largely independent groups
  - Paul Baran, Leonard Kleinrock, Donald Davies, Roger Scantlebury

ARPANET

- Development sponsored by DARPA
- Motivated by considerations of fail-safe networks
- Applications shall not depend on individual elements in the networks
  - Clearly based on quite some military thinking in general
  - But: NOT a particular response to a Soviet nuclear threat

Initial ARPANET (1969)



Common Goals for Early Internet Design

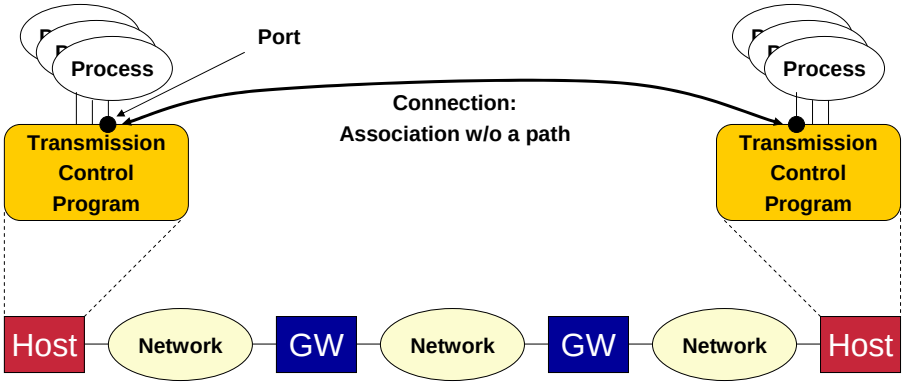
- Scientists interested in sharing and accessing (computing) resources
- Scientists and engineers interested in building an infrastructure to interconnect all computers in the world and make it work
- Designers, users, and operators: common vision and spirit

Some Aspects of the Internet Design Philosophy

"Has evolved considerably from the first proposal to the current standards…"

Dave Clark, 1988

Initial Terminology and Architecture (1974)



Design Goals

Fundamental goal:

- Develop an effective technique for multiplexed utilization of existing interconnected networks
  - Need to incorporate existing infrastructure
  - Alternative: design a new unified system for a variety of transmission media

Second level goals, in the order of importance (from Dave Clark, 1988)

1. Internet communication must continue despite loss of gateways
2. Internet must support multiple types of communication services
3. Internet architecture must accommodate a variety of networks
4. Must permit distributed management of resources
5. Internet architecture must be cost-effective
6. Host attachment must be possible with a low level of effort
7. The resources used in the Internet architecture must be accountable

Why Packets?

- Many existing applications naturally served by packet switching
  - Remote login, file transfer, …
- Networks to be integrated were packet-switching networks
  - Influence from the ARPANET development
- Packets can be used to provide different types of service
  - Circuit switching already implies certain services characteristics
  - (which may not be needed by all applications)
  - Datagrams (carried in packets) serve as elementary building block
- Packets contribute to survivability in the presence of failures

Why Minimal Service?

- Networks have different service characteristics
  - Requiring a good service would imply sophisticated mappings
  - Packets are easily carried over anything
  - May add functionality that is ultimately not needed
  - Ultimately, endpoints are responsible for more complex services

What is minimal functionality?

- Best effort packet delivery
- Fragmentation to adapt to variable packet sizes in different networks

Robustness in the Presence of Failures

- Communication shall continue even if networks/links or gateways/routers fail
- Applications must not be affected as far as possible
  - Network is required to deal with failures and repair them internally
  - Mask transient failures completely from endpoints and applications
  - Only visible error case: total partitioning of the network
  - (achieved today if applications would be sufficiently patient)
- Immediate implications
  - Essential state stored in the network must be protected against losses
  1. Replicate state information and provide means for recovery
  2. Do not store state in the network!
- Concept of Fate Sharing
  - Only applications on endpoints store the relevant state and data
  - If the endpoint or application fails, it does not matter any more that the state is lost

Separation of the Network and the Hosts

- Keep the network simple and “dumb”
  - Network is agnostic to the applications it carries
  - IP layer
- Provide the application functionality only in the endpoints
  - Applications do not know/care about the way the network operates
  - Everything above IP
- Allows the network to be changed independent of applications
- Allows new applications to be introduced w/o network involvment
- End-to-End Arguments (Saltzer, et al. 1984)

The End-to-End Arguments

- Reasoning against low-level implementation of functionality
  - Within the network / between network nodes in support of the application

> The function in question can completely and correctly be implemented only with the knowledge and help of the application standing at the end points 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.)
>
> Saltzer, et al. 1984

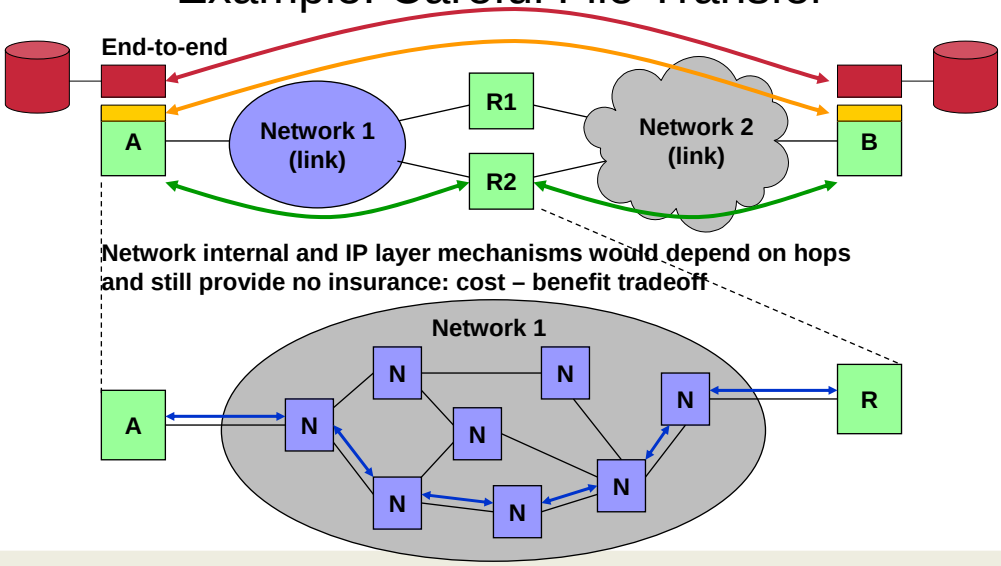
End-To-End Argument

- Internet infrastructure provides best-effort datagram forwarding service
  - No transport services
  - No quality of service
  - No security
- Motivation
  - Providing these services at lower levels conflicts with internetworking idea
    - Advanced functions require state in network elements
    - Not all functions can be efficiently implemented with all link layer technologies
  - Many functions have to be provided by the end-systems anyway
    - Reliable transport (verification)
    - Security

Example: Careful File Transfer

- Move a file from a disk attached to machine A to a disk connected to machine B via some network
- Ensure complete and identical availability of the file on B's disk afterwards
- Proper reception, processing, and storage can only be assured by the application itself
  - It is the only entity aware of the real requirements
  - Needs to implement proper validation mechanisms anyway
- Transport and lower layer protocols can help performance
- The proper tradeoff requires careful thought!

Example: Careful File Transfer



Low- vs. High-Level Implementation

- Lower layer implementation
  - May simplify applications or perform functions more efficiently
  - May be shared by numerous applications
  - But may be enforced on applications that do not need it
  - But may also operate on incomplete information (less efficient)
- Higher layer implementation
  - May be tailored to an application‘s needs
  - But may require the application (protocol) designer to deal with the issue
- Choice of several layers (network, transport, application)
- Trade-off is important!
  - Implies properly identifying “the ends”

Support “in the Network”

- May be the only way to achieve certain aspects
  - QoS-based forwarding / service differentiation
  - Efficient distribution of multicast packets
- May improve performance
  - (Link-layer) retransmissions for wireless links
  - Caching for web applications
- May cause side effects
  - Particularly if the applications are unaware of such support
  - Breaking end-to-end connectivity

End-to-End Examples

Pro

- Delivery guarantees
- Duplicate suppression
- Sequencing
- Transaction management
- Secure data transmission
- Authentication

Well?

- Authentication
- Congestion control

The Soft-State Principle

- Robustness requirement
  - Internet communication must continue despite loss of networks or routers
- End-to-End principle advocates stateless network elements
  - Routers forward packets independently, no circuit state
- Some exceptions
  - Routing:
    - Routing tables, spanning trees…
  - Integrated services (Qos)
    - Flow specifications, traffic shaping rules
- Observation
  - Complete stateful approach not feasible
    - Hosts and network elements can fail, Internet topology may change
- Solution: Soft-State
  - Required state information is periodically refreshed
    - Routing protocols, RSVP messages

The Internet Landscape Today

- Users
- Commercial ISPs
  - Working for profit
- Private sector network providers
- Governments
  - Want to care, need to care
- “Intellectual Property Rights” (IPR) holders
- Providers of content and higher level services
  - Streaming, telephony, media, ...
- Tensions between interests of the various parties
- “Support” for applications, users, etc.

Changes over time…

- From closed academic environment to global society
  - Trusted users ➔ non-trusted users (including criminals)
  - Users who know what they do ➔ users who don't want to (need to) know
- From research to commercial
- New stakeholders in the Internet
  - Internet Service Providers (ISPs)
  - Application Service Providers (ASPs), Hosters, Web site operators, …
  - Governments
- Third parties (to facilitate interactions)
  - Trusted entities, caches, proxies, ...
- ...

Moving away from End-to-End

- Operation in an untrustworthy world
  - Assumption that endpoints are willing to cooperate does not necessarily hold
- More demanding applications
  - Streaming media, large amounts of content
  - Two-stage delivery
- Rise of third-party involvement
  - Taxation, law enforcement, public safety, notary functions, …
- ISP service differentiation
  - Internal-use only services or quality guarantees
- Less sophisticated users
  - Responsible ISPs
  - Application service providers

Challenging the End-to-End Argument: Firewalls and NATs

- Firewalls: packet filters in network elements

The primary purpose of firewalls has always been to shield buggy code from bad guys. -- Steve Bellovin

  - Represent intelligence in the network
    - Delivery of packets is not controlled by end-systems
  - Firewalls require hard-state
    - Filter-rules, must be managed
  - New architectural considerations
    - DMZs…
  - Violate Internet transparency:
    - General packet forwarding service behaves differently
    - Some applications do not work anymore

Challenging the End-to-End Argument: Firewalls and NATs

- Network Address Translators: NATs
  - Network elements change IP header contents
  - Rewrite source and destination addresses for implementing private networks
  - Usage of port number space for multiplexing IP addresses
- General operation
  - Creation of port-address bindings
    - Typically automatically by inspecting TCP segments (and UDP messages)
  - Designed to be transparent to end-systems
- New behavior
  - Changing (typically constant) IP and TCP (UDP) header fields
    - Cannot assume authenticity of IP source addresses
    - End-to-end-security does not work
  - Many application do not work: VoIP, peer-to-peer communication

Challenging the End-to-End Argument: Active Networking

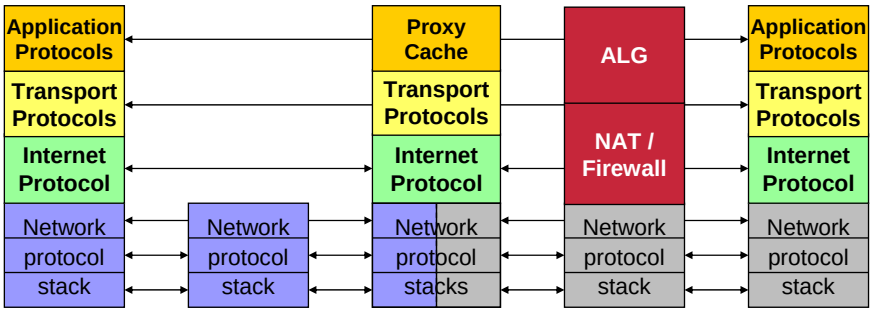
- Fundamental idea:
  - Programmable network elements
  - Hosts (applications) can install code at routers
- Motivation:
  - Enabling new services that require network support
    - E.g.: media transcoding, compression, multicast distribution
- Active Networking and the end-to-end argument
  - Programmability seems to contradict end-to-end argument
    - Complex and unpredictable interactions, loss of transparency
  - Counter-argument: programmability in lower layers as a means to defer design choices upwards in the layering
    - Applications (end-systems) can define concrete functions although functions are implemented in the network

Challenging the End-to-End Argument: Application Layer Gateways

- Web Proxies:
  - Non-transparent intermediaries that provide useful functions on the application layer
    - Caching, authorization
- General category: Application Layer Gateways (ALGs)
  - Do not interfere with basic forwarding and transport functions
  - Provide useful application functions
    - Performance optimizations, adaptations
  - Can represent applications overlay network
    - Different topology than actual Internet
      - suboptimal routing…

Application Protocols in Today's World

- Limited transparent end-to-end communications only
  - Deliberately or enforced
  - Supposed unnoticeable to the applications



Some recent views on design principles…

- Tensions between competing interests
  - Music sharing vs. IPR protection
  - Privacy vs. wiretapping
  - User freedom vs. ISP's desire for control (and accounting)
- Design for change
- Design for choice
- Design for variability in outcome
- Controlled transparency
- Isolation of conflicts of interest
- Some problems need to be solved in a non-technical fashion...!

The Twelve Networking Truths

RFC 1925, 1996-04-01

1

It Has To Work.

No matter how hard you push and no matter what the priority,
you can't increase the speed of light.

No matter how hard you try, you can't make a baby in much less than 9 months. Trying to speed this up *might* make it slower, but it won't make it happen any quicker.

With sufficient thrust,
pigs fly just fine.

However, this is not necessarily a good idea. It is hard to be sure where they are going to land, and it could be dangerous sitting under them as they fly overhead.

4

Some things in life can never be fully appreciated nor understood unless experienced firsthand.

Some things in networking can never be fully understood by someone who neither builds commercial networking equipment nor runs an operational network.

5

It is always possible to agglutinate multiple separate problems into a single complex interdependent solution.

In most cases this is a bad idea.

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.

It is always possible to add another level of indirection.

⑦

Good, Fast, Cheap: Pick any two.

⑧

It is more complicated than you think.

⑨

For all resources, whatever it is, you need more.

Every networking problem always takes longer to solve than it seems like it should.

10

One size never fits all.

11

Every old idea will be proposed again with a different name and a different presentation.

Regardless of whether it works.

In protocol design,
perfection has been reached
not when there is nothing left to add,
but when there is nothing left to take away.