Internet design philosophy

CSE 461, Spring 2021

Ratul Mahajan

Two foundational papers that *extract* principles from Internet design

- The Design Philosophy of DARPA Internet Protocols

  Clark, 1988

- End-to-end Arguments in System Design
  Saltzer, Reed, and Clark, 1984

These principles were not even articulated, let alone be explicit design goal, when the Internet was being engineered

Many applications, disparate requirements

File transfer: High throughput, reliability

YouTube: Min throughput, low jitter

Phone call: Low jitter, low latency

Zoom: Low jitter, minimum throughput, low latency

Web: Low latency

Your next idea

Internet design goals

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

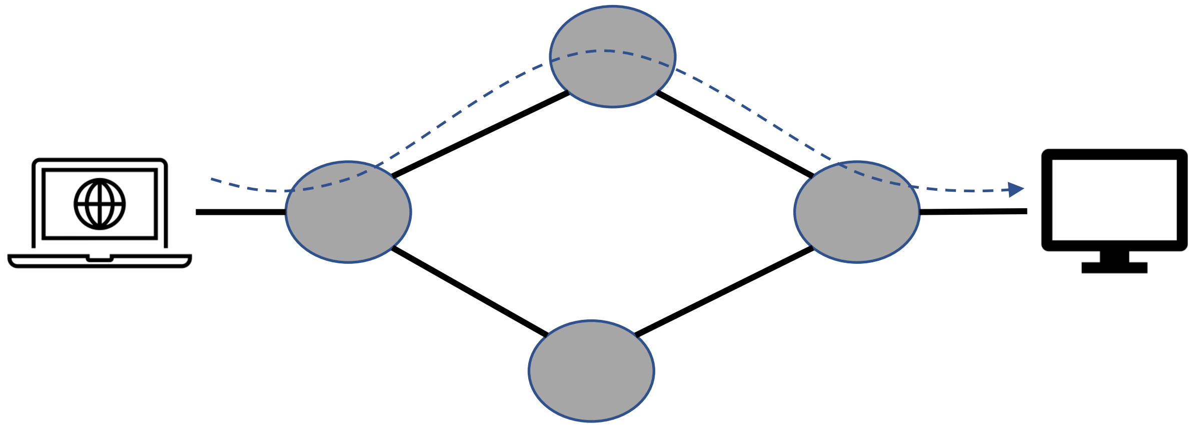
How to design a network that satisfies them all?

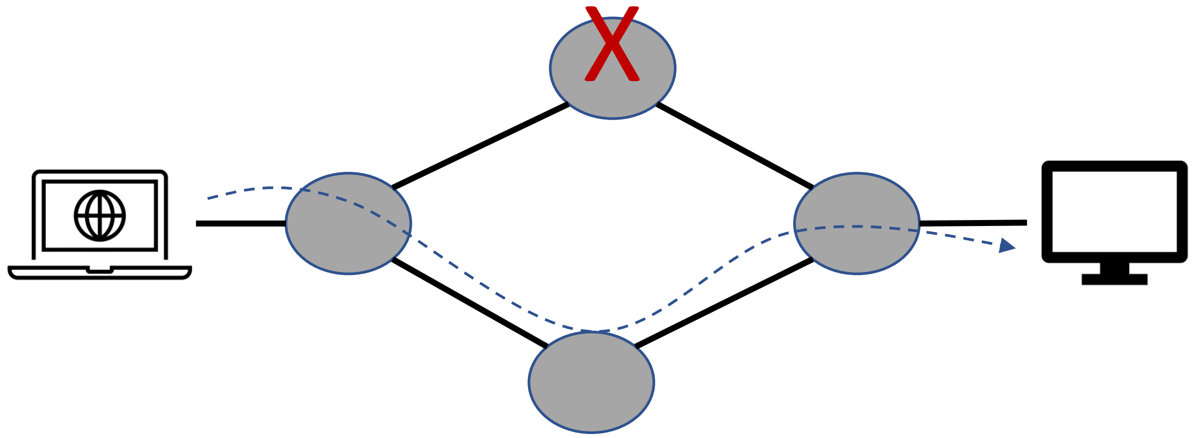
Two meta decisions

- Service model: Datagram or virtual circuits?
- What functionality to put in the network?
    - Reliable delivery
    - Delivery acknowledgement
    - Prevent duplication
    - Guarantee minimum throughput
    - Guarantee latency
    - FIFO
    - Encryption
    - Authentication
    - …

Goal1: Internet communication must continue despite loss of networks or gateways.

Masking failures with datagrams





Routers have no connection state, so datagrams can be easily re-routed

- Possible but difficult with VCs

Hosts have connection state

- “Fate sharing”

Moot point when there are single-points of failure in the network

What functionality to put in the network?

Internet: As little as possible

- Hard to think of a simpler network

Goal 2: The Internet must support multiple types of communication service.

Functionality in the network gets in the way of services that do not need it

Goal 3: The Internet architecture must accommodate a variety of networks.

Some networks may not be capable of providing expected functionality

End-to-end argument (coming soon)

Generally not possible for network to meet exact application requirements

These are different concerns

The cost of in-network functionality

Consider reliability and common ways of providing it

1. Cache the packet and resend if it is not acknowledged
    - Needs extra memory and some compute in routers
2. Send multiple times
    - High network overhead
3. Error coding
    - Simple example:
        - Send $A \oplus B$ in addition to sending A and B.
        - Recover A using B and $A \oplus B$
    - Needs extra computation at the sender and memory at the receiver

Were the other Internet goals met?

Goal 4: The Internet architecture must permit distributed management of its resources.

Goal 5: The Internet architecture must be cost effective.

Goal 6: The Internet architecture must permit host attachment with a low level of effort.

Goal 7: The resources used in the internet architecture must be accountable.

Limitations of Internet’s architecture

Security

- DoS attacks – you can send an arbitrary amount, anonymously
- Phishing – identity spoofing
- Prefix hijacking

Hard to support highly demanding applications

Suboptimal efficiency and performance

Privacy

Discussion

Were those the right design goals for that time?

What should be the design goals now?

End-to-End principle

End-to-end Principle

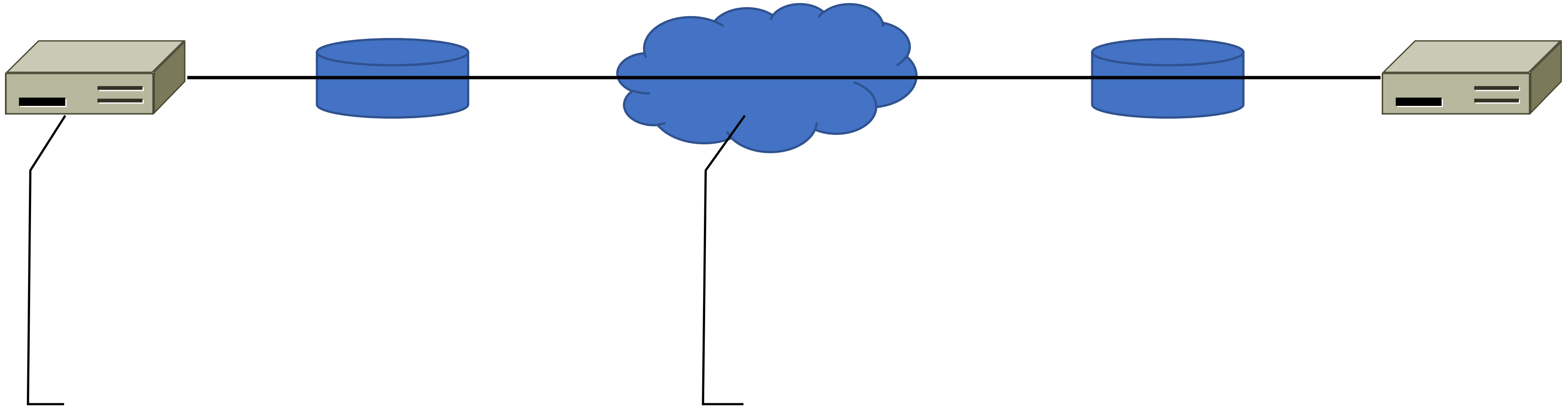
- Broad networking principle
  - First implementation in French CYCLADES network (after ARPA) (1970)
  - Articulated in its most recognizable form by Saltzer, Reed, Clark (1981) [paper]

- Guidance on placing functionality such as reliability, security, etc.—in network or at endpoints (hosts)?
  - Argues for endpoint placement

Multiple interpretations of the principle

- The network cannot be trusted. Do it yourself.
  - The network can suffer heavy damage
    - Nuclear attacks (but not DDoS attacks!)
  - Need end-to-end correctness anyway
- Diminishing returns from in-network functionality
  - Not everyone needs it
- Place functionality in the network only when necessary
  - E.g., for performance

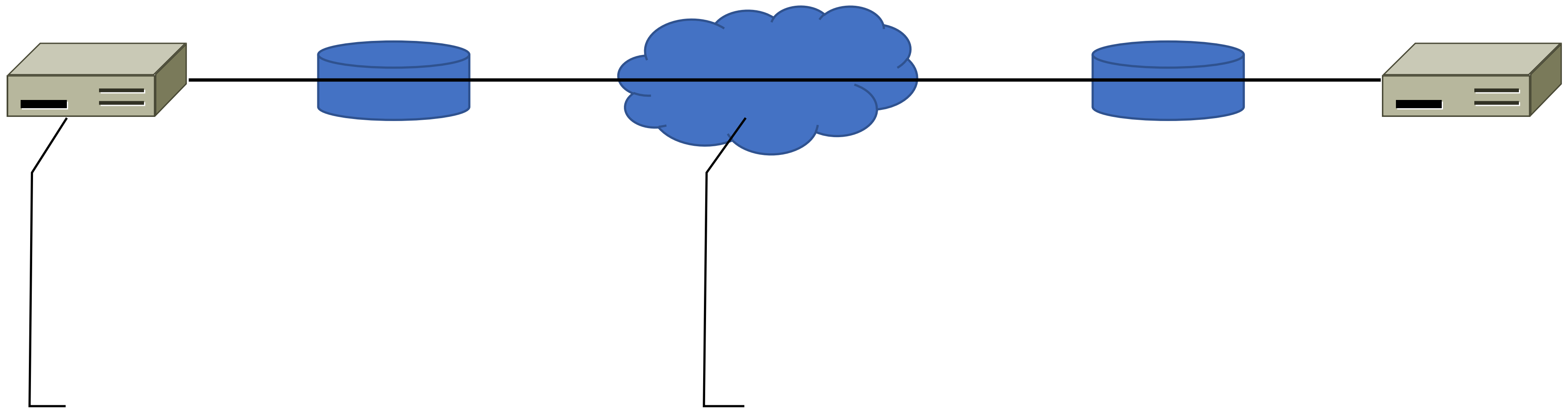
E2E Example: Error-correcting codes



IP:
Host detects errors

802.11:
Link detects errors

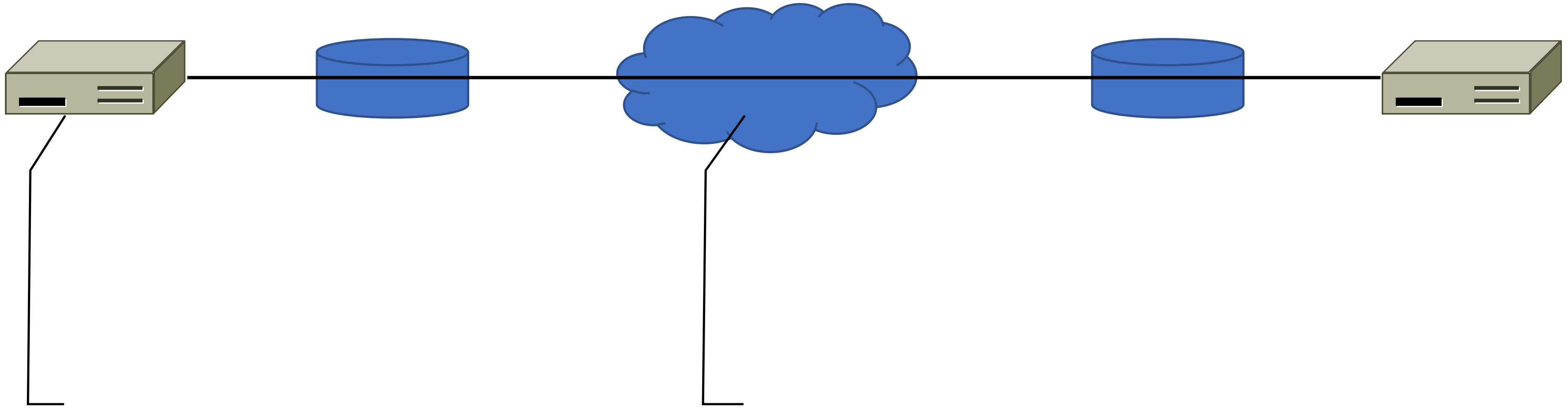
E2E Example: ARQ



TCP:
Host retransmits
on failure

802.11:
Link detects drops
and retransmits

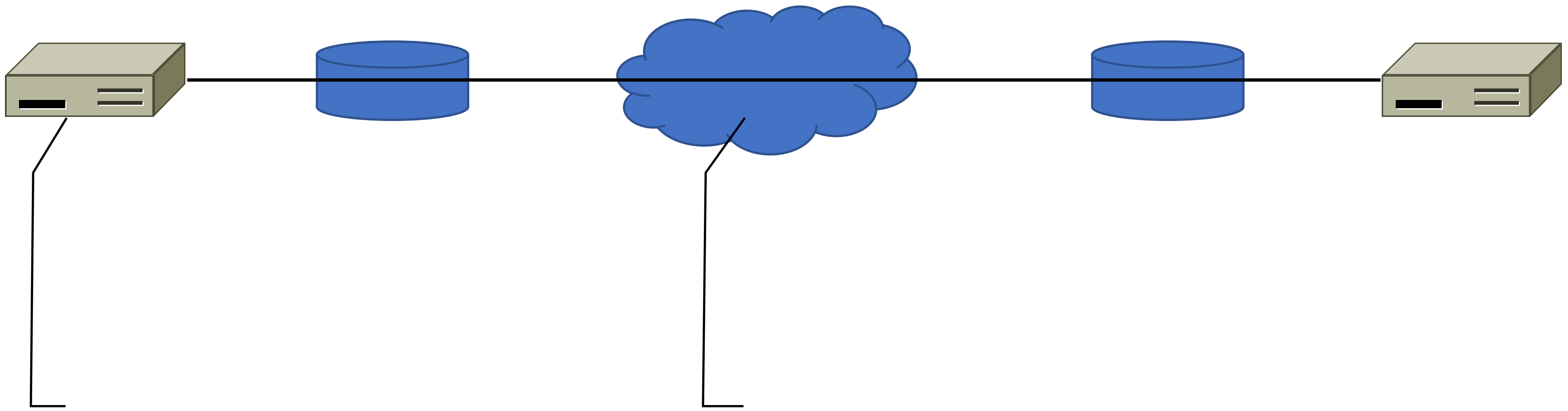
E2E Example: In-order delivery



TCP:
Host enforces in-order delivery

SS5:
Network enforces in-order delivery

E2E Example: Security



SSL:
Host encrypts content

GSM:
Network encrypts content

End-to-End limitations

- Some functionality cannot be implemented at endpoints
  - NATs, DoS protection, … the principle is silent on these

- Assumes a clear dividing line between network and endpoints
  - Reality of distributed applications (e.g., CDNs) is more complex

- No guidance on how much functionality can go in the network for performance

What is the opposite of e2e argument?

1. Network with rich functionality that covers most requirements
   - E.g., phone network
   - Practical for a data network?
2. Network with multiple ”lanes”?
   - CISC-like
   - Slight aside: Can applications be trusted?
3. Modular network
   - Applications mix-n-match what they need