

Part 3: Internet Design Principle

Goals:

- identify, study principles that can guide network architecture
- "bigger" issues than specific protocols or implementation wisdom,
- *synthesis*: the really big picture

Overview:

- Internet design principles
- rethinking the Internet design principles
- packet-switching versus circuit-switching revisited

Part 3 3-1

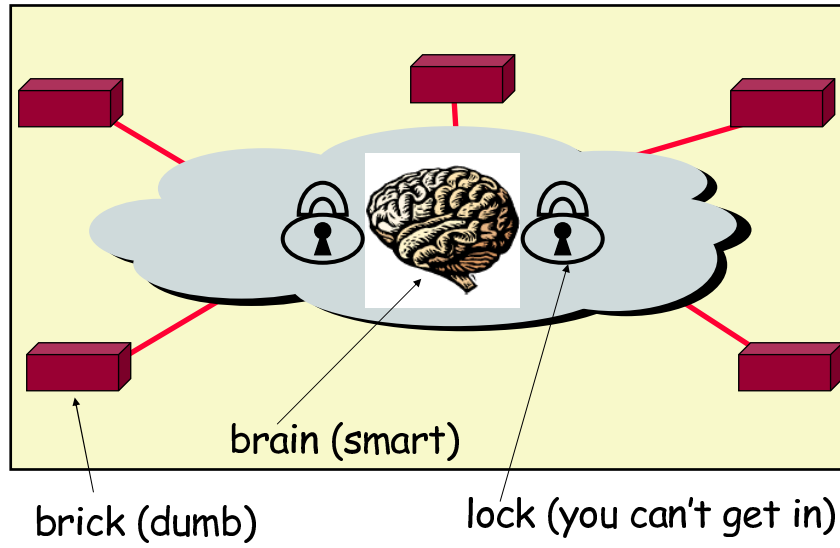
Key questions

- 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 ?

Answer these questions in context of Internet

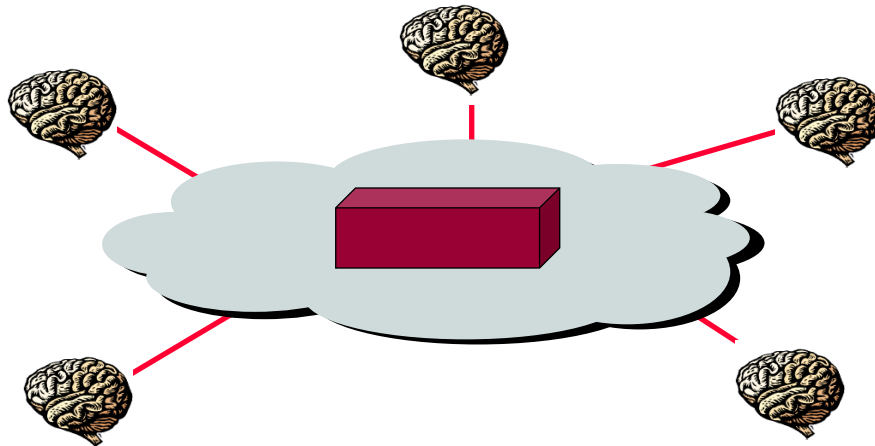
Part 3 3-2

Common View of the Telco Network



Part 3 3-3

Common View of the IP Network



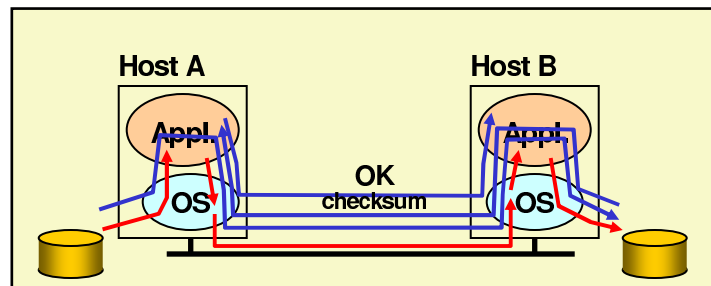
Part 3 3-4

Internet E2E Argument

- "...functions placed at the lower levels may be *redundant* or of *little value* when compared to the cost of providing them at the higher level..."
- "...sometimes an *incomplete* version of the function provided by the communication system (lower levels) may be useful as a *performance enhancement*..."
- This leads to a philosophy diametrically opposite to the telephone world of dumb end-systems (the telephone) and intelligent networks.

Part 3 3-5

Example: Reliable File Transfer



- Solution 1: make each step reliable, and then concatenate them
- Solution 2: each step unreliable: end-to-end check and retry

Part 3 3-6

Discussion

- Is solution 1 good enough?
 - No - what happens if components fail or misbehave (bugs)?
- Is reliable communication sufficient:
 - No - what happens if disk errors?
- so need application to make final correctness check
- Thus, full functionality can be entirely implemented at application layer; *no* need for reliability from lower layers

Part 3 3-7

Discussion

Q: Is there any reason to implement reliability at lower layers?

A: YES: "easier" (and more efficient) to check and recover from errors at each intermediate hop

- e.g: faster response to errors, localized retransmissions

Part 3 3-8

Tradeoffs

- ❑ application has more information about the data and semantics of required service (e.g., can check only at the end of each data unit)
- ❑ lower layer has more information about constraints in data transmission (e.g., packet size, error rate)
- ❑ *Note:* these trade-offs are a direct result of layering!

Part 3 3-9

End-to-end Argument: Examples

- ❑ Error control and reliable transmission
 - Audio conferencing
 - Digital music player
- ❑ Secure transmission of data
 - Per-hop vs. end-to-end encryption
 - Data integrity, privacy, and authentication
- ❑ RISC architectures
- ❑ Extensible operating systems

Part 3 3-10

Internet & E2E Argument

- network layer provides one simple service: best effort datagram (packet) delivery
- transport layer at network edge (TCP) provides end-end error control
 - performance enhancement used by many applications (which could provide their own error control)
- all other functionality ...
 - all application layer functionality
 - network services: DNS implemented at *application* level

Part 3 3-11

Internet & E2E Argument

Discussion: congestion control, flow control: why at transport, rather than link or application layers?

- congestion control needed for many application (assumes reliable application-to-tcp data passing)
- many applications "don't care" about congestion control - it's the network's concern
- consistency across applications- you **have** to use it if you use TCP (social contract)
- why do it at the application level
 - **Flow control** - application knows how/when it wants to consume data
 - **Congestion control** - application can do tcp-friendly cc

Part 3 3-12

Internet & E2E Argument

Why not at the link/network layer

- 1: not every application needs/want it
- 2: lots of state at each router (each connection needs to buffer, need back pressure) - it's hard

Part 3 3-13

E2E Argument: Interpretations

- One interpretation:
 - A function can *only* be completely and correctly implemented with the knowledge and help of the applications *standing at the communication endpoints*
- Another: (more precise...)
 - a system (or subsystem level) should consider only functions that can be *completely and correctly* implemented within it.
- Alternative interpretation: (also correct ...)
 - Think twice before implementing a functionality that you believe that is useful to an application at a lower layer
 - If the application can implement a functionality correctly, implement it a lower layer *only* as a performance enhancement

Part 3 3-14

End-to-End Argument: Critical Issues

- end-to-end principle emphasizes:
 - *function placement*
 - *correctness, completeness*
 - *overall system costs*

- Philosophy: if application can do it, don't do it at a lower layer - application best knows what it needs
 - add functionality in lower layers iff (1) used by and improves performances of many applications, (2) does not hurt other applications

- allows *cost-performance* tradeoff

Part 3 3-15

End-to-End Argument: Discussion

- end-end argument emphasizes correctness & completeness, does not emphasize:
 - *complexity*: is complexity at edges result in a "simpler" architecture?
 - *evolvability*: ease of introduction of new functionality: ability to evolve because easier/cheaper to add new edge applications than change routers?
 - *technology penetration*: simple network layer makes it "easier" for IP to spread everywhere

Part 3 3-16

Internet Design Philosophy (Clark'88)

In order of importance: *Different ordering of priorities would yield a different architecture!*

- 0 **Connect existing networks**
 - initially ARPANET, ARPA packet radio, packet satellite network
1. **Survivability**
 - ensure communication service even with network and router failures
2. **Support multiple types of services**
3. **Must accommodate a variety of networks**
4. Allow distributed management
5. Allow host attachment with a low level of effort
6. Be cost effective
7. Allow resource accountability

Part 3 3-17

1. Survivability

- Continue to operate even in the presence of network failures (e.g., link and router failures)
 - as long as network is not partitioned, two endpoints should be able to communicate
 - any other failure (excepting network partition) should be **transparent** to endpoints
- Decision: maintain e2e transport state only at end-points
 - eliminate the problem of handling state inconsistency and performing state restoration when router fails
- Internet: **stateless** network-layer architecture
 - No notion of a session/call at network layer
- Grade: A-
 - routing algorithm fail-over path is non-optimal, non-traffic sensitive. (Note: ISPs worry about this)

Part 3 3-18

2. Types of Services

- ❑ add UDP to TCP to better support other apps
 - e.g., "real-time" applications
- ❑ arguably main reason for separating TCP, IP
- ❑ datagram abstraction: lower common denominator on which other services can be built
 - service differentiation was considered (remember ToS?), but this has never happened on the large scale (Why?)
- ❑ Grade: B+
 - Need something (reliability) between TCP and UDP? Why not just build on top of UDP
 - Need time sensitivity for MM application
 - Need a quality of service notion: give me throughput X or give me a busy signal (this is what ATM is/was advocating)

Part 3 3-19

3. Variety of Networks

- ❑ Very successful (why?)
 - because the minimalist service; it requires from underlying network only to deliver a packet with a "reasonable" probability of success
- ❑ ...does not require:
 - reliability
 - in-order delivery
- ❑ The mantra: IP over everything
 - Then: ARPANET, X.25, DARPA satellite network..
 - Now: ATM, SONET, WDM...
- ❑ Grade: A
 - because it runs over everything

Part 3 3-20

Other Goals

- Allow **distributed management**
 - Administrative autonomy: IP interconnects networks
 - each network can be managed by a different organization
 - different organizations need to interact only at the boundaries
 - ... but this model complicates routing
 - Grade: B
 - Why: some stuff managed centrally: DNS, IP address allocation (but it's not that much)
 - Today's Distributed management makes it easy for misconfigurations or malicious users to corrupt infrastructure (e.g., AT&T routing black hole)
- **Cost effective**
 - sources of inefficiency
 - header overhead
 - retransmissions
 - routing
 - ...but "optimal" performance never been top priority
 - Grade: B+ (300 million people can't be wrong!)

Part 3 3-21

Other Goals (Cont)

- **Low cost of attaching a new host**
 - not a strong point → higher than other architecture because the **intelligence is in hosts** (e.g., telephone vs. computer)
 - bad implementations or malicious users can produce considerably harm (remember fate-sharing?)
 - Grade: B
 - Leverages low cost of end-system hardware (Ethernet NICs ~ \$20), DHCP makes self configuration easy
 - Very hard to debug problems
- **Accountability**
 - Grade: F

Part 3 3-22

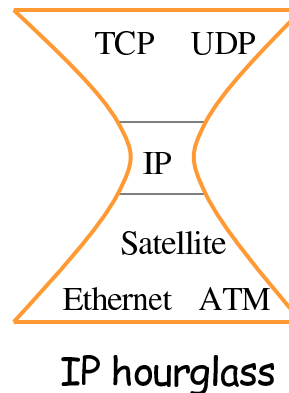
What About the Future

- ❑ Datagram not the best abstraction for:
 - resource management, accountability, QoS
- ❑ new abstraction: **flow** (see IPv6)
 - but no one knows what a flow *is*
- ❑ routers require to maintain per-flow state
- ❑ state management: recovering lost state is hard
- ❑ here (1988) we see the first proposal of "soft state"
 - **soft-state**: end-hosts responsible to maintain the state

Part 3 3-23

Summary: Internet Architecture

- ❑ packet-switched datagram network
- ❑ IP is the glue (network layer overlay)
- ❑ IP hourglass architecture
 - all hosts and routers run IP
- ❑ stateless architecture
 - no per flow state inside network



Part 3 3-24

Summary: Minimalist Approach

- **Dumb network**
 - IP provide minimal functionalities to support connectivity
 - addressing, forwarding, routing
- **Smart end-system**
 - transport layer or application performs more sophisticated functionalities
 - flow control, error control, congestion control
- **Advantages**
 - accommodate heterogeneous technologies (Ethernet, modem, satellite, wireless)
 - support diverse applications (telnet, ftp, Web, X windows)
 - decentralized network administration

Part 3 3-25

But that was **yesterday**

..... what about **tomorrow?**

Part 3 3-26