Internet Architecture:

Design Philosophy -Then and Now

Internet Philosophy and Design Principles

Architecture: the big picture

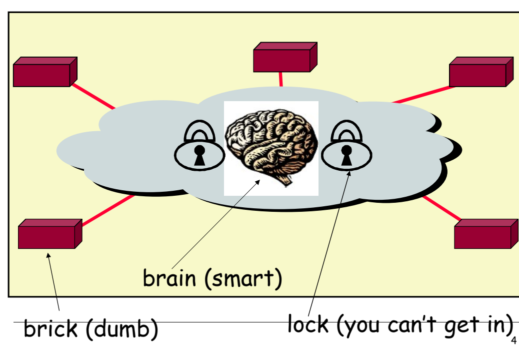
Goals:
- identify, study principles that can guide network architecture
- "bigger" issues than specific protocols or implementation tricks
- *synthesis:* the *really* big picture

Key questions

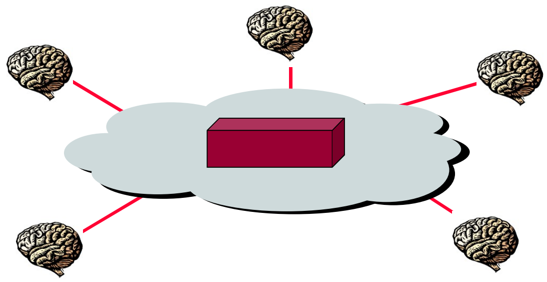
- How to decompose the complex system functionality into protocol layers?
- Which functions placed *where* in network, at which layers?
- Can a function be placed at multiple levels ?

Answer these questions in context of Internet, telephone network

Common View of the Telco Network



Common View of the IP Network



Readings: Saltzer84

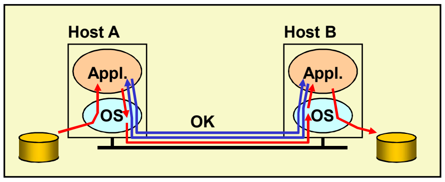
- End-to-end argument
  - Better to implement functions close to application
  - … except when performance requires otherwise
- Why?
  - …
- What should be the “end” for network “functionalities”, e.g., routing?
  - Router?
  - End host?
  - Enterprise edge?
  - Autonomous System?

Internet End-to-End Argument

According to [Saltzer84]:

- “…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*…”
- This leads to a philosophy diametrically opposite to the telephone world of dumb end-systems (the telephone) and intelligent networks.

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

E2E Example: File Transfer

- Even if network guaranteed reliable delivery
  - Need to provide end-to-end checks
  - E.g., network card may malfunction
  - The receiver has to do the check anyway!
- Full functionality can only be entirely implemented at application layer; no need for reliability from lower layers
- Does FTP look like E2E file transfer?
  - TCP provides reliability between kernels not disks

Discussion

- Solution 1 not good enough!
  - what happens if the sender or/and receiver misbehave?
- so receiver has to do check anyway!
- Thus, full functionality can be entirely implemented at application layer; *no* need for reliability from lower layers

10

Discussion

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

A: Yes, but only to improve performance

- Example:
  - assume high error rate in network
  - reliable communication service at data link layer might help (why)?
  - fast detection /recovery of errors

11

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

12

Internet & End-to-End 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 functionalities …
  - all application layer functionalities
  - network services: DNS

  implemented at application level

13

Internet & End-to-End Argument

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

- Claim: common functions should migrate down the stack
  - Everyone shares same implementation: no need to redo it (reduces bugs, less work, etc…)
  - Knowing everyone is doing the same thing, can help
- congestion control too important to leave up to application/user: true but hard to police
  - TCP is "outside" the network; compliance is "optional"
  - We do this for fairness (but realize that people could cheat)
- Why error control, flow control in TCP, not (just) in app

14

Trade-offs

- 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!

15

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

16

End-to-End Argument: Discussion

- end-end argument emphasizes correctness & completeness, but *not*
  - 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

Summary: End-to-End Arguments

- If the application can do it, don't do it at a lower layer -- anyway the application knows the best what it needs
  - add functionality in lower layers iff it is (1) used and improves performances of a large number of applications, and (2) does not hurt other applications
- Success story: Internet
  - But …

Readings: Clark88

- Basic story of Clark88
  - Enumerate (and prioritize) system goals
  - … and see what decisions that leads you to make
- Clark88 doesn't say much about routing, network trouble-shooting, security, etc., but
  - "Some of the most significant problems with the Internet today relate to lack of sufficient tools for distributed management, especially in the area of routing."
- What should be goals & priorities for routing, network trouble-shooting, security?
  - …

Internet Design Philosophy (Clark'88)

In order of importance:

0 Connect existing networks
   - initially ARPANET and ARPA packet radio 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

Different ordering of priorities would make a different architecture!

0. connect existing networks
Internet: virtualizing local networks

1974: multiple unconnected networks
- ARPAnet
- data-over-cable networks
- packet satellite network (Aloha)
- packet radio network

.. differing in:
- addressing conventions
- packet formats
- error recovery
- routing

21

Challenge 1: Address Formats

- Map one address format to another?
  - Bad idea → many translations needed
- Provide one common format
  - Map lower level addresses to common format

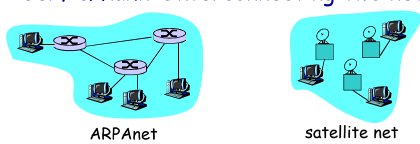
Challenge 2: Different Packet Sizes

- Define a maximum packet size over all networks?
  - Either inefficient or high threshold to support
- Implement fragmentation/re-assembly
  - Who is doing fragmentation?
  - Who is doing re-assembly?

Gateway Alternatives

- Translation
  - Difficulty in dealing with different features supported by networks
  - Scales poorly with number of network types (N^2 conversions)
- Standardization
  - "IP over everything" (**Design Principle 1**)
  - Minimal assumptions about network
  - Hourglass design

Cerf & Kahn: Interconnecting two networks

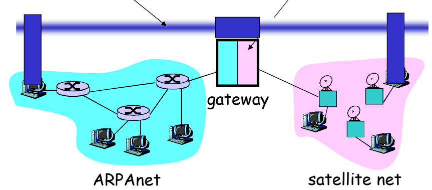


- "…interconnection must preserve intact the internal operation of each network."
- " ..the interface between networks must play a central role in the development of any network interconnection strategy. We give a special name to this interface that performs these functions and call it a GATEWAY."
- ".. prefer that the interface be as simple and reliable as possible, and deal primarily with passing data between networks that use different packet-switching strategies
- "…address formats is a problem between networks because the local network addresses of TCP's may vary substantially in format and size. A uniform internetwork TCP address space, understood by each GATEWAY and TCP, is essential to routing and delivery of internetwork packets."

Cerf & Kahn: Interconnecting two networks

Internetwork layer:
- addressing: internetwork appears as a single, uniform entity, despite underlying local network heterogeneity
- network of networks

Gateway:
- "embed internetwork packets in local packet format or extract them"
- route (at internetwork level) to next gateway



Historical Aside:
Proposed Internetwork packet in 1974:

local header	source address	dest. address	seq. #	byte count	flag field	text	checksum

network	TCP identifier
8	16

Cerf & Kahn's Internetwork Architecture

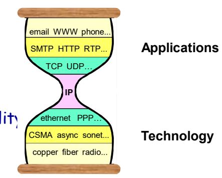
- two layers of addressing: internetowork and local network
- new layer makes everything homogeneous
- underlying local network technology (cable, satellite, 56K modem) is "invisible" at internetwork layer

IP Standardization

- Minimum set of assumptions for underlying net
  - Minimum packet size
  - Reasonable delivery odds, but not 100%
  - Some form of addressing unless point to point
- Important non-assumptions:
  - Perfect reliability
  - Broadcast, multicast
  - Priority handling of traffic
  - Internal knowledge of delays, speeds, failures, etc
- Also achieves Goal 3: Supporting Varieties of Networks

IP Hourglass

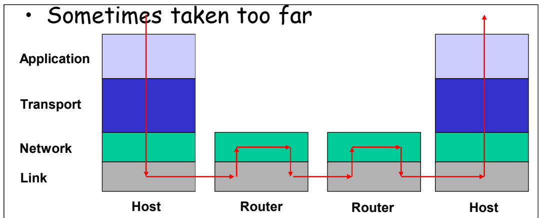
- Need to interconnect many existing networks
- Hide underlying technology from applications
- Decisions:
  - Network provides minimal functionality
  - "Narrow waist"



Tradeoff: **No assumptions, no guarantees.**

IP Layering (Principle 2)

- Relatively simple
- Sometimes taken too far



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 e-e transport state only at end-points
  - eliminate the problem of handling state inconsistency and performing state restoration when router fails
- Internet: stateless network architecture
  - No notion of a session/call at network layer
  - Grade: A-, because convergence times are relatively slow
  - BGP can take minutes to coverge
  - IS-IS OSPF take ~ 10 seconds

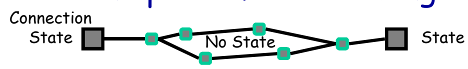
32

Survivability

- If network disrupted and reconfigured
  - Communicating entities should not care!
  - No higher-level state reconfiguration
- How to achieve such reliability?
  - Where can communication state be stored?

	Network	Host
Failure handing	Replication	“Fate sharing”
Net Engineering	Tough	Simple
Switches	Maintain state	Stateless
Host trust	Less	More

Principle 3: Fate Sharing



- Lose state information for an entity if and only if the entity itself is lost.
- Examples:
  - OK to lose TCP state if one endpoint crashes
    - NOT okay to lose if an intermediate router reboots
  - Is this still true in today’s network?
    - NATs and firewalls
- Survivability compromise: Heterogeneous network → less information available to end hosts and Internet level recovery mechanisms

Principle 4: Soft-state

- Soft-state
  - Announce state
  - Refresh state
  - Timeout state
- Penalty for timeout – poor performance
- Robust way to identify communication flows
  - Possible mechanism to provide non-best effort service
- Helps survivability

2. Types of Service

- **Principle 6**: network layer provides one simple service: best effort datagram (packet) delivery
  - All packets are treated the same
- Relatively simple core network elements
- Building block from which other services (such as reliable data stream) can be built
- Contributes to scalability of network
- No QoS support assumed from below
  - In fact, some underlying nets only supported reliable delivery
    - Made Internet datagram service less useful!
  - Hard to implement without network support
  - QoS is an ongoing debate…

ToS: TCP vs. UDP

- Original Internet model: "TCP/IP" one layer
- 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?)
- proven to allows lots of applications to be invented and flourish (except MM, but maybe that's not a transport service issue)

37

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…

A: can't name a link layer technology that IP doesn't run over (carrier pigeon RFC)

38

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
  - A for implementation, B for concept (disagreement)
- Cost effective
  - sources of inefficiency
    - header overhead
    - retransmissions
    - routing
  - …but "optimal" performance never been top priority
  - A

39

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?)
  - C: but things are improving with DHCP, auto-configurations. Looks like a higher grade in future
- Accountability
  - Internet gets an "F"

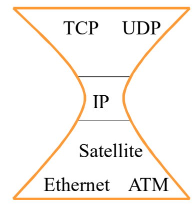
40

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

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



IP hourglass

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

But that was yesterday

……. what about tomorrow?

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 we see the first proposal of "soft state"!
  - soft-state: end-hosts responsible to maintain the state

45

Rethinking Internet Design

What's changed?

- operation in untrustworthy world
  - endpoints can be malicious
  - If endpoint not trustworthy, but want trustworthy network -> more mechanism in network core
  - Trust and security a big issue today!
- more demanding applications
  - end-end best effort service not enough
  - new service models in network (Intserv, Diffserv)?
  - new application-level service architecture built on top of network core (e.g., CDN, p2p)?
  - wireless and mobility

46

Rethinking Internet Design …

What's changed?

- ISP service differentiation
  - ISP doing more (than other ISPs) in core is competitive advantage
- rise of third party involvement
  - interposed between endpoints (even against will)
  - e.g., Chinese government, US recording industry
- new technologies (wireless, optical …)
- limited capability devices (e.g., PDA, smart phones, sensors, ……), or perhaps also less "sophisticated" users

All these changes motivate shift away from end-end!

47

What's at stake?

"At issue is the conventional understanding of the 'Internet philosophy'

- freedom of action
- user empowerment
- end-user responsibility for actions taken
- lack of control "in" the net that limit or regulate what users can do

The end-end argument fostered that philosophy because they enable the freedom to innovate, install new software at will, and run applications of the users choice"

[Blumenthal and Clark, 2001]

48

Technical response to changes

- Add functions to the network core ("middleboxes"):
  - filtering firewalls
  - application-level firewalls, web caches and proxies
  - NAT boxes
  - active networking
  - …
- Add "infrastructure services"
  - e.g., DNS,
  - (application-specific) content distribution networks (CDNs)

… All operate within network, making use of application-level information

- which addresses can do what at application level?
- If addresses have meaning to applications, NAT must "understand" that meaning

49

Next Week

- Review lecture notes and the required readings for this week:
  - [Saltzer84] and [Clark88]
    - also [Clark:Tussle] and [CerfKahn] if you have time
- Questions for you to think about:
  - What are the "architectural" advantages of Internet, and also its limitations?
  - If you can redesign it, how would you do it?

50