S3: Design Considerations

Internetworking: Architectural principles, names, addresses

Instructor:
Behnam Momeni

Fall 2018
Lecture: Design Considerations

• How to determine split of functionality
  • Across protocol layers
  • Across network nodes

• Assigned Reading
  • [SRC84] End-to-end Arguments in System Design
  • [Cla88] Design Philosophy of the DARPA Internet Protocols

• Optional Reading
  • [CT90] Architectural Considerations for a New Generation of Protocols
  • [Clark02] Tussle in Cyberspace: Defining Tomorrow’s Internet
Outline

• Design principles in internetworks

• IP design
Goals [Clark88]

0 Connect existing networks
   initially ARPANET and ARPA packet radio network

1. Survivability
   ensure communication service even in the presence of
   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
Goal 0: Connecting Networks

• How to internetwork various network technologies
  • ARPANET, X.25 networks, LANs, satellite networks, packet networks, serial links…

• Many differences between networks
  • Address formats
  • Performance – bandwidth/latency
  • Packet size
  • Loss rate/pattern/handling
  • Routing
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
**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
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?

<table>
<thead>
<tr>
<th></th>
<th>Network</th>
<th>Host</th>
</tr>
</thead>
<tbody>
<tr>
<td>Failure handing</td>
<td>Replication</td>
<td>“Fate sharing”</td>
</tr>
<tr>
<td>Net Engineering</td>
<td>Tough</td>
<td>Simple</td>
</tr>
<tr>
<td>Switches</td>
<td>Maintain state</td>
<td>Stateless</td>
</tr>
<tr>
<td>Host trust</td>
<td>Less</td>
<td>More</td>
</tr>
</tbody>
</table>
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
Principle 5: End-to-End Argument

- Deals with **where** to place functionality
  - Inside the network (in switching elements)
  - At the edges
- Argument
  - There are functions that can only be correctly implemented by the endpoints – do not try to completely implement these elsewhere
  - Guideline not a law
Example: Reliable File Transfer

- Solution 1: make each step reliable, and then concatenate them
- Solution 2: 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

• Is there any need to implement reliability at lower layers?
Discussion

• Yes, but only to improve performance
• If network is highly unreliable
  • Adding some level of reliability helps performance, not correctness
  • Don’t try to achieve perfect reliability!
  • Implementing a functionality at a lower level should have minimum performance impact on the applications that do not use the functionality
Examples

• What should be done at the end points, and what by the network?
  • Reliable/sequenced delivery?
  • Addressing/routing?
  • Security?
  • What about Ethernet collision detection?
  • Multicast?
  • Real-time guarantees?
Goal 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…
Types of Service

• TCP vs. UDP
  • Elastic apps that need reliability: remote login or email
  • Inelastic, loss-tolerant apps: real-time voice or video
  • Others in between, or with stronger requirements
  • Biggest cause of delay variation: reliable delivery
    • Today’s net: <100ms RTT
    • Reliable delivery can add seconds.

• Original Internet model: “TCP/IP” one layer
  • First app was remote login…
  • But then came debugging, voice, etc.
  • These differences caused the layer split, added UDP
Goal 4: Decentralization

- **Principle 7:** Each network owned and managed separately
  - Will see this in BGP routing especially

- **Principle 7’:** Be conservative in what you send and liberal in what you accept
  - Unwritten rule
  - Especially useful since many protocol specifications are ambiguous
  - E.g. TCP will accept and ignore bogus acknowledgements
The “Other” goals

5. Attaching a host
   • Host must implement hard part 😞 → transport services
     • Not too bad

6. Cost effectiveness
   • Packet overhead less important by the year
   • Packet loss rates low
   • Economies of scale won out
   • Internet cheaper than most dedicated networks

• But…
7. Accountability

- Huge problem

- Accounting
  - Billing? (mostly flat-rate. But phones have become that way also - people like it!)
  - Inter-ISPs payments

- Accountability and security
  - Without state in the network, how do we tell if hosts are behaving?
  - Huge problem.
  - Worms, viruses, etc.
    - Partly a host problem. But hosts very trusted.
  - Authentication
    - Purely optional. Many philosophical issues of privacy vs. security.
  - Greedy sources aren’t handled well
Other IP Design Weaknesses

- Weak administration and management tools
- Incremental deployment difficult at times
  - Result of no centralized control
  - No more “flag” days
  - Are active networks the solution?
Summary: Internet Architecture

- Packet-switched datagram network
- IP is the “compatibility layer”
  - Hourglass architecture
  - All hosts and routers run IP
- Stateless architecture
  - no per flow state inside network
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

• Successes: IP on everything!

• Drawbacks… but perhaps they’re totally worth it in the context of the original Internet. Might not have worked without them!

“This set of goals might seem to be nothing more than a checklist of all the desirable network features. It is important to understand that these goals are in order of importance, and an entirely different network architecture would result if the order were changed.”
Clark’s 2013 List of Principles

- Packet switching
  - Effective multiplexing of resources
  - Ability to operate over a range of networks.
  - Support for a wide range of applications.
- Gateways (what we call routers today)
  - Ability to exploit existing networks of many sorts—e.g. minimal assumptions about what the networks would do.
- Co-location of flow state with end-points of flows. (fate-sharing)
- Trust in the end-node
- No flow state in routers, which implies no flow setup, and thus the “pure” datagram model. Also implies strict separation of IP from TCP, with no knowledge of TCP in routers.
  - Availability in the face of failures
  - Minimal requirements for functions in gateways
- No mechanisms to report network failures to end-points.
- Minimal assumptions about service functions and performance.
Other Observations

- Longevity as a goal?
  - Variable vs. fixed length addressing
  - Performance vs. flexibility
- TTL
  - Helps isolate routing protocol decisions into domains
- Architecture vs. Realization
  - 2013 – Principles vs. mechanisms/implementation vs. deployment
- Recovery and hiding failures
  - No clear reporting/monitoring of failures
  - Impact on performance hidden
Changes Over Time

• Developed in simpler times
  • Common goals, consistent vision

• With success came multiple goals – examples:
  • ISPs must talk to provide connectivity but are fierce competitors
  • Privacy of users vs. government’s need to monitor
  • User’s desire to exchange files vs. copyright owners

• Must deal with the tussle between concerns in design
New Principles?

• Design for variation in outcome
  • Allow design to be flexible to different uses/results

• Isolate tussles
  • QoS designs uses separate ToS bits instead of overloading other parts of packet like port number
  • Separate QoS decisions from application/protocol design

• Provide choice → allow all parties to make choices on interactions
  • Creates competition
  • Fear between providers helps shape the tussle
Integrated Layer Processing (ILP)

- Layering is convenient for architecture but not for implementations
- Combining data manipulation operations across layers provides gains
  - E.g. copy and checksum combined provides 90Mbps vs. 60Mbps separated
- Protocol design must be done carefully to enable ILP
- Performance bottlenecks
  - In paper: presentation overhead, application-specific processing >> other processing
  - Today: memory bandwidth
Application Lever Framing (ALF)

• Objective: enable application to process data ASAP

• Application response to loss
  • Retransmit (TCP applications)
  • Ignore (UDP applications)
  • Recompute/send new data (clever application)

• Expose unit of application processing (ADU) to protocol stack
• ADUs can be processed in any order
• Naming of ADUs should help identify position in stream
• Size
  • Enough to process independently
  • Impact on loss recovery
  • What if size is too large?
NSF Programs

- Stagnation
  - 100x100 → Clean Slate Design
  - PlanetLab
  - Overcoming the Internet Impasse through Virtualization → GENI

- Internet architecture projects
  - Named Data Networking
  - MobilityFirst
  - eXpressive Internet Architecture
NSF FIND Project [Clark]

1988

1. Internet communication must continue despite loss of networks or gateways.
2. The Internet must support multiple types of communications 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.

2008

1. Security
2. Availability and resilience
3. Economic viability
4. Better management
5. Meet society’s needs
6. Longevity
7. Support for tomorrow’s computing
8. Exploit tomorrow’s networking
9. Support tomorrow’s applications
10. Fit for purpose (it works...)

38
NSF FIND Project [Clark]

2008

1. Security
2. Availability and resilience
3. Economic viability
4. Better management
5. Meet society’s needs
6. Longevity
7. Support for tomorrow’s computing
8. Exploit tomorrow’s networking
9. Support tomorrow’s applications
10. Fit for purpose (it works...)

Internet Support

1. End2end integrity/confidentiality with encryption
2. Routing and TTL – but little else
3. None
4. None
5. None
6. Has been effective – but parts such as addressing
7. Was developed in very heterogeneous settings. Hard to predict
8. Mobile and optical have raised issues. Link failure and network management interactions could be better.
9. Done well but are important applications just ignored?
10. It does work 😊
Named Data Networking

• In the beginning...
  – First applications strictly focused on host-to-host interprocess communication:
    • Remote login, file transfer, ...
  – Internet was built around this host-to-host model.
  – Architecture is well-suited for communication between pairs of stationary hosts.

• ... while today
  – Vast majority of Internet usage is data retrieval and service access.
  – Users care about the content and are oblivious to location. They are often oblivious as to delivery time:
    • Fetching headlines from CNN, videos from YouTube, TV from Tivo
    • Accessing a bank account at www.bank.com.
What does the network look like…
What should the network look like…
MobilityFirst Architecture

- **MobilityFirst key protocol features:**
  - Fast global naming service
  - Self-certifying public key names
  - Dynamic mapping of name to topological network address(es)
  - Routers support both flat name and hierarchical address routing
  - Storage-aware (generalized DTN) routing in access
  - Hop-by-hop (segmented) transport
  - Programmable computing layer
  - Support for content/context/location
  - Separate network mgmt plane

- **New components, very distinct from IP, intended to achieve key mobility and trust goals**
A History of Internet Evolution

- Hard to change IP
  - …especially after 1990
XIA: An Evolvable Internet Architecture

Network design

How do we design a network in which we can gracefully introduce new functionality?

End-point design

How should the end-points change when the networks provide new functionality?
XIA: Evolvable Set of Principals

- Identifying the intended communicating entities reduces complexity and overhead
  - No need to force all communication at a lower level (hosts), as in today’s Internet
- Allows the network to evolve

![Diagram showing interaction between host, content, services, and future entities]
Outline

• Design principles in internetworks

• IP design
IP Address Problem (1991)

- Address space depletion
  - In danger of running out of classes A and B
- Why?
  - Class C too small for most domains
  - Very few class A – IANA (Internet Assigned Numbers Authority) very careful about giving
  - Class B – greatest problem
    - Sparsely populated – but people refuse to give it back
IP Address Utilization (‘98)

http://www.caida.org/outreach/resources/learn/ipv4space/
IPv4 Routing Problems

• Core router forwarding tables were growing large
  • Class A: 128 networks, 16M hosts
  • Class B: 16K networks, 64K hosts
  • Class C: 2M networks, 256 hosts
• 32 bits does not give enough space encode network location information inside address – i.e., create a structured hierarchy
Solution 1 – CIDR

- Assign multiple class C addresses
- Assign consecutive blocks
- RFC1338 – Classless Inter-Domain Routing (CIDR)
Classless Inter-Domain Routing

- Do not use classes to determine network ID
- Assign any range of addresses to network
  - Use common part of address as network number
  - e.g., addresses 192.4.16 - 196.4.31 have the first 20 bits in common. Thus, we use this as the network number
  - netmask is /20, /xx is valid for almost any xx
- Enables more efficient usage of address space (and router tables)
Solution 2 - NAT

- Network Address Translation (NAT)
- Alternate solution to address space
  - Kludge (but useful)
- Sits between your network and the Internet
- Translates local network layer addresses to global IP addresses
- Has a pool of global IP addresses (less than number of hosts on your network)
• Operation: Source (S) wants to talk to Destination (D):
  • Create $S_g$-$S_p$ mapping
  • Replace $S_p$ with $S_g$ for outgoing packets
  • Replace $S_g$ with $S_p$ for incoming packets
• D & S can be just IP addresses or IP addresses + port #’s
Solution 3 - IPv6

- **Scale** – addresses are 128bit
  - Header size?
- **Simplification**
  - Removes infrequently used parts of header
  - 40byte fixed size vs. 20+ byte variable
- **IPv6 removes checksum**
  - Relies on upper layer protocols to provide integrity
- **IPv6 eliminates fragmentation**
  - Requires path MTU discovery
  - Requires 1280 byte MTU
IPv6 Changes

• TOS replaced with traffic class octet
• Flow
  • Help soft state systems
  • Maps well onto TCP connection or stream of UDP packets on host-port pair
• Easy configuration
  • Provides auto-configuration using hardware MAC address to provide unique base
• Additional requirements
  • Support for security
  • Support for mobility
IPv6 Changes

- Protocol field replaced by next header field
  - Support for protocol demultiplexing as well as option processing
- Option processing
  - Options are added using next header field
  - Options header does not need to be processed by every router
    - Large performance improvement
    - Makes options practical/useful
Summary: IP Design

• Relatively simple design
  • Some parts not so useful (TOS, options)
• Beginning to show age
  • Unclear what the solution will be
EXTRA SLIDES
Fragmentation

- IP packets can be 64KB
- Different link-layers have different MTUs
- Split IP packet into multiple fragments
  - IP header on each fragment
  - Various fields in header to help process
  - Intermediate router may fragment as needed

- Where to do reassembly?
  - End nodes – avoids unnecessary work
  - Dangerous to do at intermediate nodes
    - Buffer space
    - Multiple paths through network
Fragmentation is Harmful

- Uses resources poorly
  - Forwarding costs per packet
  - Best if we can send large chunks of data
  - Worst case: packet just bigger than MTU
- Poor end-to-end performance
  - Loss of a fragment
- Reassembly is hard
  - Buffering constraints
Path MTU Discovery

• Hosts dynamically discover minimum MTU of path

• Algorithm:
  • Initialize MTU to MTU for first hop
  • Send datagrams with Don’t Fragment bit set
  • If ICMP “pkt too big” msg, decrease MTU

• What happens if path changes?
  • Periodically (>5mins, or >1min after previous increase), increase MTU

• Some routers will return proper MTU
• MTU values cached in routing table
How is IP Design Standardized?

- **IETF**
  - Voluntary organization
  - Meeting every 4 months
  - Working groups and email discussions
  - “We reject kings, presidents, and voting; we believe in rough consensus and running code” (Dave Clark 1992)
    - Need 2 independent, interoperable implementations for standard

- **IRTF**
  - End2End
  - Reliable Multicast, etc..

<table>
<thead>
<tr>
<th>Field</th>
<th>Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version</td>
<td>0</td>
</tr>
<tr>
<td>IHL</td>
<td>4</td>
</tr>
<tr>
<td>Type of Service</td>
<td>8</td>
</tr>
<tr>
<td>Total Length</td>
<td>16</td>
</tr>
<tr>
<td>Identification</td>
<td>19</td>
</tr>
<tr>
<td>Flags</td>
<td>24</td>
</tr>
<tr>
<td>Fragment Offset</td>
<td>32</td>
</tr>
<tr>
<td>Time to Live</td>
<td>0</td>
</tr>
<tr>
<td>Protocol</td>
<td>4</td>
</tr>
<tr>
<td>Header Checksum</td>
<td>8</td>
</tr>
<tr>
<td>Source Address</td>
<td>16</td>
</tr>
<tr>
<td>Destination Address</td>
<td>24</td>
</tr>
<tr>
<td>Options</td>
<td>32</td>
</tr>
<tr>
<td>Padding</td>
<td>36</td>
</tr>
</tbody>
</table>
IP Type of Service

• Typically ignored
• Values
  • 3 bits of precedence
  • 1 bit of delay requirements
  • 1 bit of throughput requirements
  • 1 bit of reliability requirements
• Replaced by DiffServ
Fragmentation Related Fields

- Length
  - Length of IP fragment
- Identification
  - To match up with other fragments
- Flags
  - Don’t fragment flag
  - More fragments flag
- Fragment offset
  - Where this fragment lies in entire IP datagram
  - Measured in 8 octet units (11 bit field)
Other Fields

- Header length (in 32 bit words)
- Time to live
  - Ensure packets exit the network
- Protocol
  - Demultiplexing to higher layer protocols
- Header checksum
  - Ensures some degree of header integrity
  - Relatively weak – 16 bit
- Options
  - E.g. Source routing, record route, etc.
  - Performance issues
    - Poorly supported
Addressing in IP

- IP addresses are names of interfaces
- Domain Name System (DNS) names are names of hosts
- DNS binds host names to interfaces
- Routing binds interface names to paths
Addressing Considerations

• Fixed length or variable length?

• Issues:
  • Flexibility
  • Processing costs
  • Header size

• Engineering choice: IP uses fixed length addresses
Addressing Considerations

• Structured vs flat

• Issues
  • What information would routers need to route to Ethernet addresses?
    • Need structure for designing scalable binding from interface name to route!
  • How many levels? Fixed? Variable?
IP Addresses

• Fixed length: 32 bits
• Initial classful structure (1981)
• Total IP address size: 4 billion
  • Class A: 128 networks, 16M hosts
  • Class B: 16K networks, 64K hosts
  • Class C: 2M networks, 256 hosts

<table>
<thead>
<tr>
<th>High Order Bits</th>
<th>Format</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>7 bits of net, 24 bits of host</td>
<td>A</td>
</tr>
<tr>
<td>10</td>
<td>14 bits of net, 16 bits of host</td>
<td>B</td>
</tr>
<tr>
<td>110</td>
<td>21 bits of net, 8 bits of host</td>
<td>C</td>
</tr>
</tbody>
</table>
### IP Address Classes (Some are Obsolete)

<table>
<thead>
<tr>
<th>Class</th>
<th>Network ID</th>
<th>Host ID</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Class A</strong></td>
<td>0</td>
<td>Network ID</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Host ID</td>
</tr>
<tr>
<td><strong>Class B</strong></td>
<td>10</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Class C</strong></td>
<td>110</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Class D</strong></td>
<td>1110</td>
<td>Multicast Addresses</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Class E</strong></td>
<td>1111</td>
<td>Reserved for experiments</td>
</tr>
</tbody>
</table>

Some classes are obsolete:
Some Special IP Addresses

• 127.0.0.1: local host (a.k.a. the loopback address
• Host bits all set to 0: network address
• Host bits all set to 1: broadcast address

• For class A & B networks
• Very few LANs have close to 64K hosts
  • For electrical/LAN limitations, performance or administrative reasons
• Need simple way to get multiple “networks”
  • Use bridging, multiple IP networks or split up single network address ranges (subnet)
  • Must reduce the total number of network addresses that are assigned
• CMU case study in RFC
  • Chose not to adopt – concern that it would not be widely supported 😊
Subnetting

- Variable length subnet masks
- Could subnet a class B into several chunks
Subnetting Example

- Assume an organization was assigned address 150.100
- Assume < 100 hosts per subnet
- How many host bits do we need?
  - Seven
- What is the network mask?
  - 11111111 11111111 11111111 10000000
  - 255.255.255.128
Subnet Addressing Example

• Assume a packet arrives with address 150.100.12.176

• Step 1: AND address with subnet mask
IPv4 Problems

- Addressing
- Routing
### IPv6 Header

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>4</td>
<td>12</td>
<td>16</td>
<td>19</td>
</tr>
<tr>
<td>Version</td>
<td>Class</td>
<td>Flow Label</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Payload Length</td>
<td>Next Header</td>
<td>Hop Limit</td>
</tr>
</tbody>
</table>

**Source Address**

**Destination Address**
Principle 4

• Fate sharing
• Critical state only at endpoints
• Only endpoint failure disrupts communication
• Helps survivability
Internet & End-to-End Argument

• Only one higher level service implemented at transport layer: reliable data delivery (TCP)
  • Performance enhancement; used by a large variety of applications (Telnet, FTP, HTTP)
  • Does not impact other applications (can use UDP)
  • Original TCP & IP were integrated – Reed successfully argued for separation
• Everything else implemented at application level