L-2 Design Considerations

**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

---

**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

---

**Challenge**

- Many differences between networks
  - Address formats
  - Performance – bandwidth/latency
  - Packet size
  - Loss rate/pattern/handling
  - Routing
  - How to internetwork various network technologies

---

**Challenge 1: Address Formats**

- Map one address format to another. Why not?
- Provide one common format
  - Map lower level addresses to common format
Challenge 2: Different Packet Sizes

- Define a maximum packet size over all networks. Why not?
- 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

End-to-End Argument (Principle 2)

- 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 at them elsewhere
  - Caveat: can provide a partial form as performance enhancement
  - 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
- 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 application 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?

Internet & End-to-End Argument

• At network layer provides one simple service: best effort datagram (packet) delivery
• 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
• Does FTP look like E2E file transfer?
  • TCP provides reliability between kernels not disks

Principle 3

• Best effort 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

Principle 4

• Fate sharing
• Critical state only at endpoints
• Only endpoint failure disrupts communication
• Helps survivability

Principle 5

• 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 6

• Decentralization
• 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

IP Layering (Principle 8)

• Relatively simple
• Sometimes taken too far

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
• Presentation overhead, application-specific processing >> other processing
  • Target for ILP optimization

IP Design Weaknesses

• Greedy sources aren’t handled well
• Weak accounting and pricing tools
• Weak administration and management tools
• Incremental deployment difficult at times
  • Result of no centralized control
  • No more “flag” days
  • Are active networks the solution?

Outline

• Design principles in internetworks
  • IP design

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..

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
• IP packets can be 64KB
• Different link-layers have different MTUs
• Split IP packet into multiple fragments
  • IP header on each fragment
  • 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 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)

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
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

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)
- Network ID
  - Class A: 8 bits
  - Class B: 10 bits
  - Class C: 110 bits
  - Class D: 1110 bits
  - Class E: 1111 bits
- Host ID
  - Class A: 24 bits
  - Class B: 16 bits
  - Class C: 8 bits
  - Class D: Reserved for multicast addresses
  - Class E: Reserved for experiments
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 B & C 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 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?
  - 255.255.255.128

Subnet Addressing Example

- Assume a packet arrives with address 150.100.12.176
- Step 1: AND address with subnet mask

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
IPv4 Problems

• Addressing
  • Class B sparsely populated – but people refuse to give it back

• Routing
  • 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)

NAT Illustration

• 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
Problems with NAT

• What if we only have few (or just one) IP address?
  • Use NAPT (Network Address Port Translator)
  • NAPT translates:
    • Translates Paddr + flow info to Gaddr + new flow info
    • Uses TCP/UDP port numbers
    • Potentially thousands of simultaneous connections with one global IP address

• Hides the internal network structure
  • Some consider this an advantage
  • Multiple NAT hops must ensure consistent mappings
  • Some protocols carry addresses
    • e.g., FTP carries addresses in text
    • What is the problem?
    • Encryption
    • No inbound connections

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: 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: IP Design
• Relatively simple design
  • Some parts not so useful (TOS, options)
• Beginning to show age
  • Unclear what the solution will be → probably IPv6

Next Lecture: NS Tutorial
• NS – VINT project network simulator
• Assigned reading
  • Start HW 1 before lecture