15-446 Distributed Systems
Spring 2009

L-2 Internet Design Philosophy

Today’s Lecture
- Layers and protocols
- Design principles in internetworks

Lots of Functions Needed
- Link
- Multiplexing
- Routing
- Addressing/naming (locating peers)
- Reliability
- Flow control
- Fragmentation
- Etc....

What is Layering?
- Modular approach to network functionality
- Example:

  - Application
  - Application-to-application channels
  - Host-to-host connectivity
  - Link hardware
Protocols

- Module in layered structure
- An agreement between parties on how communication should take place
- Protocols define:
  - Interface to higher layers (API)
  - Interface to peer (syntax & semantics)
    - Actions taken on receipt of a messages
    - Format and order of messages
    - Error handling, termination, ordering of requests, etc.
- Example: Buying airline ticket

Layering

Layering: technique to simplify complex systems

Layering Characteristics

- Each layer relies on services from layer below and exports services to layer above
- Interface defines interaction
- Hides implementation - layers can change without disturbing other layers (black box)

The Internet Engineering Task Force

- Standardization is key to network interoperability
  - The hardware/software of communicating parties are often not built by the same vendor → yet they can communicate because they use the same protocol
- Internet Engineering Task Force
  - Based on working groups that focus on specific issues
- Request for Comments
  - Document that provides information or defines standard
  - Requests feedback from the community
  - Can be “promoted” to standard under certain conditions
  - Consensus in the committee
  - Interoperating implementations
- Project 1 will look at the Internet Relay Chat (IRC) RFC
E.g.: OSI Model: 7 Protocol Layers

- Physical: how to transmit bits
- Data link: how to transmit frames
- Network: how to route packets
- Transport: how to send packets end2end
- Session: how to tie flows together
- Presentation: byte ordering, security
- Application: everything else

TCP/IP has been amazingly successful, and it’s not based on a rigid OSI model. The OSI model has been very successful at shaping thought.

OSI Layers and Locations

The Internet Protocol Suite

The waist facilitates interoperability
Layer Encapsulation

Protocol Demultiplexing

Multiple choices at each layer

Multiplexing and Demultiplexing

Is Layering Harmful?

Layer N may duplicate lower level functionality (e.g., error recovery)
Layers may need same info (timestamp, MTU)
Strict adherence to layering may hurt performance
Some layers are not always cleanly separated.
  - Inter-layer dependencies in implementations for performance reasons
  - Some dependencies in the standards (header checksums)
Interfaces are not really standardized.
  - It would be hard to mix and match layers from independent implementations, e.g., windows network apps on unix (w/out compatibility library)
  - Many cross-layer assumptions, e.g. buffer management
Today’s Lecture

- Layers and protocols
- Design principles in internetworks

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

Priorities

- The effects of the order of items in that list are still felt today
  - E.g., resource accounting is a hard, current research topic
  - Let’s look at them in detail

0. Connecting Existing Networks

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

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

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”
    - Minimal assumptions about network
    - Hourglass design

1. 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>Switches</td>
<td>Maintain state</td>
<td>Stateless</td>
</tr>
<tr>
<td>Host trust</td>
<td>Less</td>
<td>More</td>
</tr>
</tbody>
</table>
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

Soft-State

- Basic behavior
  - 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

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

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

- If network guaranteed reliable delivery
  - The receiver has to do the check anyway!
  - E.g., network card may malfunction
- 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 applications that do not use the functionality

2. Types of Service

- 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
- No QoS support assumed from below
  - Accommodates more networks
  - 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
3. Varieties of Networks

- 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.

- Much engineering then only has to be done once

4. Management

- Each network owned and managed separately
- Will see this in BGP routing especially

5. Attaching a host

- Not awful; DHCP and related autoconfiguration technologies helping.

6. Cost effectiveness

- Economies of scale won out
- Internet cheaper than most dedicated networks
- Packet overhead less important by the year
- But...

7. Accountability

- Huge problem.
- Accounting
  - Billing? (mostly flat-rate. But phones are moving that way too - people like it!)
  - Inter-provider payments
- Accountability and security
  - 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

The “Other” goals

- 4. Management
  - Each network owned and managed separately
  - Will see this in BGP routing especially

- 5. Attaching a host
  - Not awful; DHCP and related autoconfiguration technologies helping.

- 6. Cost effectiveness
  - Economies of scale won out
  - Internet cheaper than most dedicated networks
  - Packet overhead less important by the year
  - But...

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
- Beginning to show age
  - Unclear what the solution will be → probably IPv6
- Discussion: what are the implications for distributed system design?