Today’s Lecture

• Layers and protocols

• Design principles in internetworks

• Applications

How to Design a Network?

• Has many users
• Offers diverse services
• Mixes very diverse technologies

• Components built by many companies
• Diverse ownership
• Can evolve over time

Protocol and Service Levels

Layering: modular approach to network functionality
A Layer Network Model

The Open Systems Interconnection (OSI) Model

- 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 is not based on a rigid OSI model. The OSI model has been very successful at shaping thought.

Layering Characteristics

- Each layer relies on services from layer below and exports services to layer above
- Interface defines interaction with peer on other hosts – called protocols
- Modules hide implementation - layers can change without disturbing other layers (black box)

What are Protocols?

- An agreement between parties on how communication should take place
- Module in layered structure
- Protocols define: 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
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 but 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

Other Relevant Standardization Bodies

  - government representatives (PTTs/State Department)
  - responsible for international "recommendations"

- T1 - telecom committee reporting to American National Standards Institute.
  - T1/ANSI formulate US positions
  - interpret/adapt ITU standards for US use, represents US in ISO

- IEEE - Institute of Electrical and Electronics Engineers.
  - responsible for many physical layer and datalink layer standards

- ISO - International Standards Organization.
  - covers a broad area

Life of a Packet

- Application
- Presentation
- Session
- Transport
- Network
- Data Link
- Physical

Host, Bridge/Switch, Router/Gateway, Host

Layer Encapsulation

User A
- Get index.html
- Connection ID
- Source/Destination
- Link Address

User B
Multiplexing and Demultiplexing

- There may be multiple implementations of each layer.
  - How does the receiver know what version of a layer to use?
- Each header includes a demultiplexing field that is used to identify the next layer.
  - Filled in by the sender
  - Used by the receiver
- Multiplexing occurs at multiple layers. E.g., IP, TCP, ...

<table>
<thead>
<tr>
<th>VHL</th>
<th>TOS</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>Flags/Offset</td>
<td></td>
</tr>
<tr>
<td>TTL</td>
<td>Prot</td>
<td>H. Checksum</td>
</tr>
<tr>
<td>Source IP address</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Destination IP address</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Options</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Protocol Demultiplexing

- Multiple choices at each layer

<table>
<thead>
<tr>
<th>FTP</th>
<th>HTTP</th>
<th>NV</th>
<th>TFTP</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCP</td>
<td>UDP</td>
<td>IPX</td>
<td></td>
</tr>
<tr>
<td>IP</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NET₁</td>
<td>NET₂</td>
<td>…</td>
<td>NETₙ</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Network</th>
<th>IP</th>
<th>TCP/UDP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type Field</td>
<td>Protocol Field</td>
<td>Port Number</td>
</tr>
</tbody>
</table>

The Internet Protocol Suite

<table>
<thead>
<tr>
<th>FTP</th>
<th>HTTP</th>
<th>NV</th>
<th>TFTP</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCP</td>
<td>UDP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IP</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NET₁</td>
<td>NET₂</td>
<td>…</td>
<td>NETₙ</td>
</tr>
</tbody>
</table>

Applications
- TCP
- UDP

Data Link
- Physical

The Hourglass Model

The waist facilitates interoperability

Is Layering Harmful?

- Strict adherence to layering may hurt performance
  - Layer N may duplicate lower level functionality (e.g., error recovery)
  - Layers may need same info (timestamp, MTU)
- 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
- Applications

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: Address Formats

- Map one address format to another?
  - Bad idea → many translations needed
- Provide one common format
  - Map lower level addresses to common format
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 that underlying networks must meet to be part of the Internet
  - 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.

Goal 1: Survivability

- If network is 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 handling</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
- Tradeoffs
  - Survivability: Heterogeneous network → less information available to end hosts and Internet level recovery mechanisms
  - Trust: must trust endpoints more

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
  - Not a law – more of a “best practices”

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 lower layers to implement full reliability
- 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 to Apps

- 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-ISP payments
- Accountability and security
  - Huge problem.
  - Worms, viruses, etc.
    - Party 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 lack of centralized control
  - No more “flag” days

Changes Over Time → New Principles?

- Developed in simpler times
  - Common goals, consistent vision
- With success came changes in 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
- Provide choice → allow all parties to make choices on interactions
  - Creates competition
  - Fear between providers helps shape the tussle

Today’s Lecture

- Layers and protocols
- Design principles in internetworks
- Applications

Applications and Application-Layer Protocols

- Application: communicating, distributed processes
  - Running in network hosts in “user space”
  - Exchange messages to implement app
  - e.g., email, file transfer, the Web
- Application-layer protocols
  - One “piece” of an app
  - Define messages exchanged by apps and actions taken
  - User services provided by lower layer protocols
Client-Server Paradigm

Typical network app has two pieces: client and server

Client:
- Initiates contact with server ("speaks first")
- Typically requests service from server,
- For Web, client is implemented in browser; for e-mail, in mail reader

Server:
- Provides requested service to client
- e.g., Web server sends requested Web page, mail server delivers e-mail

Server and Client

Server and Client exchange messages over the network through a common Socket API

Network Addressing Analogy

Telephone Call
- Professors at CMU
  - Extension
  - Telephone No
- Central Number
- Exchange
- Area Code
  - 15-441 Students

Network Programming
- Applications/Servers
  - Port No.
  - IP Address
  - Network No.
  - Host Number
- Clients

What Service Does an Application Need?

Data loss
- Some apps (e.g., audio) can tolerate some loss
- Other apps (e.g., file transfer, telnet) require 100% reliable data transfer

Timing
- Some apps (e.g., Internet telephony, interactive games) require low delay to be "effective"

Bandwidth
- Some apps (e.g., multimedia) require minimum amount of bandwidth to be "effective"
- Other apps (“elastic apps”) make use of whatever bandwidth they get
Transport Service Requirements of Common Apps

<table>
<thead>
<tr>
<th>Application</th>
<th>Data loss</th>
<th>Bandwidth</th>
<th>Time Sensitive</th>
</tr>
</thead>
<tbody>
<tr>
<td>file transfer</td>
<td>no loss</td>
<td>elastic</td>
<td>no</td>
</tr>
<tr>
<td>e-mail</td>
<td>no loss</td>
<td>elastic</td>
<td>no</td>
</tr>
<tr>
<td>web documents</td>
<td>no loss</td>
<td>elastic</td>
<td>no</td>
</tr>
<tr>
<td>real-time audio/</td>
<td>loss-tolerant</td>
<td></td>
<td>yes, 100’s msec</td>
</tr>
<tr>
<td>video</td>
<td>audio: 5Kb-1Mb</td>
<td></td>
<td>yes, few secs</td>
</tr>
<tr>
<td>stored audio/video</td>
<td>loss-tolerant</td>
<td>same as above</td>
<td>yes, few secs</td>
</tr>
<tr>
<td>interactive games</td>
<td>loss-tolerant</td>
<td>few Kbps</td>
<td>yes, 100’s msec</td>
</tr>
<tr>
<td>financial apps</td>
<td>no loss</td>
<td>elastic</td>
<td>yes and no</td>
</tr>
</tbody>
</table>

Other Requirements
- Network reliability
- Network service must always be available
- Security: privacy, denial of service, authentication, …
- Scalability.
- Scale to large numbers of users, traffic flows, …
- Manageability: monitoring, control, billing, …

User Datagram Protocol (UDP): An Analogy
- Single socket to receive messages
- No guarantee of delivery
- Not necessarily in-order delivery
- Datagram – independent packets
- Must address each packet

Postal Mail
- Single mailbox to receive letters
- Unreliable 😞
- Not necessarily in-order delivery
- Letters sent independently
- Must address each reply

Example UDP applications
Multimedia, voice over IP

Example TCP applications
Web, Email, Telnet

Transmission Control Protocol (TCP): An Analogy
- Reliable – guarantee delivery
- Byte stream – in-order delivery
- Connection-oriented – single socket per connection
- Setup connection followed by data transfer

Telephone Call
- Guaranteed delivery
- In-order delivery
- Connection-oriented
- Setup connection followed by conversation
Some Example Applications

- A basic early application – ftp
- The first popular app – web/http
- Today’s apps – lots ...

FTP: The File Transfer Protocol

- Transfer file to/from remote host
- Client/server model
  - Client: side that initiates transfer (either to/from remote)
  - Server: remote host
- ftp: RFC 959
- ftp server: port 21

Ftp: Separate Control, Data Connections

- Ftp client contacts ftp server at port 21, specifying TCP as transport protocol
- Two parallel TCP connections opened:
  - Control: exchange commands, responses between client, server. “out of band control”
  - Data: file data to/from server
- Ftp server maintains “state”: current directory, earlier authentication

Ftp Commands, Responses

- Sample Commands:
  - sent as ASCII text over control channel
  - USER username
  - PASS password
  - LIST return list of files in current directory
  - RETR filename retrieves (gets) file
  - STOR filename stores (puts) file onto remote host

- Sample Return Codes
  - status code and phrase
  - 331 Username OK, password required
  - 125 data connection already open; transfer starting
  - 425 Can’t open data connection
  - 452 Error writing file