Organizing Network Functionality

- Many kinds of networking functionality
  - e.g., encoding, framing, routing, addressing, reliability, etc.
- How should they be organized?
- How should they interact?
- Layering is one answer to these questions
Layering

- A technique to organize a network system into a **succession** of logically distinct entities, such that the service provided by one entity is **solely** based on the service provided by the previous (lower level) entity
  - no layering violations

- Advantages
  - Abstraction – lower layers can be changed **without** affecting the upper layers
  - Modularity – protocols easier to manage and maintain
  - Reuse – upper layers can reuse the functionality provided by lower layers

- Disadvantages
  - Information hiding – inefficient implementations
new application has to interface to all existing media
  - adding new application requires $O(m)$ work, $m = \text{number of media}$
new media requires all existing applications be modified
  - adding new media requires $O(a)$ work, $a = \text{number of applications}$
total work in system $O(ma) \rightarrow$ eventually too much work to add apps/media
3 Layers

- Solution: introduce an intermediate layer that provides a single abstraction for various network technologies
  - $O(1)$ work to add app/media
  - variation on “add another level of indirection”

![Diagram showing 3 layers: Application, Intermediate layer, Transmission Media with examples of technologies like SMTP, SSH, NFS, HTTP for Application layer, Coaxial cable, Fiber optic, Packet radio for Transmission Media layer]
ISO OSI Reference Model

- ISO – International Standard Organization
- OSI – Open System Interconnection
- Started to 1978; first standard 1979
  - ARPANET started in 1969; TCP/IP protocols ready by 1974
- Goal: a general open standard
  - allow vendors to enter the market by using their own implementation and protocols
ISO OSI Reference Model

- Seven layers
  - Lower three layers are peer-to-peer
  - Next four layers are end-to-end
Encapsulation

- A layer can use only the service provided by the layer immediate below it
- Each layer may change and add a header to data packet
  - higher layer’s header is treated as payload
OSI Model Concepts

- Service – says **what** a layer does
- Interface – says **how** to access the service
- Protocol – says **how** is the service implemented
  - a set of rules and formats that govern the communication between two peers
Physical Layer (1)

- Service: move the information between two systems connected by a physical link
- Interface: specifies how to send a bit
- Protocols: coding scheme used to represent a bit, voltage levels, duration of a bit

- Examples: coaxial cable, optical fiber links; transmitters, receivers
Datalink Layer (2)

- **Service:**
  - framing, i.e., attach frame separators
  - send data frames between peers
  - others:
    - arbitrate the access to common physical media
    - ensure reliable transmission
    - provide flow control

- **Interface:** send a data unit (packet) to a machine connected to the same physical media

- **Protocols:** layer addresses, implement Medium Access Control (MAC) (e.g., CSMA/CD)…
Network Layer (3)

- **Service:**
  - deliver a packet to specified destination
  - perform segmentation/reassemble
  - others:
    - packet scheduling
    - buffer management

- **Interface:** send a packet to a specified destination

- **Protocols:** define global unique addresses; construct routing tables
Transport Layer (4)

- Services:
  - provide an error-free and flow-controlled end-to-end connection
  - multiplex multiple transport connections to one network connection
  - split one transport connection in multiple network connections

- Interface: send a packet to specify destination
- Protocols: implement reliability and flow control
- Examples: TCP and UDP
Session Layer (5)

- **Service:**
  - full-duplex
  - access management, e.g., token control
  - synchronization, e.g., provide check points for long transfers

- **Interface:** depends on service

- **Protocols:** token management; insert checkpoints, implement roll-back functions
Presentation Layer (6)

- Service: convert data between various representations
- Interface: depends on service
- Protocol: define data formats, and rules to convert from one format to another
Application Layer (7)

- Service: any service provided to the end user
- Interface: depends on the application
- Protocol: depends on the application

- Examples: Kazaa, SMTP, ssh, NFS, WWW browser
OSI vs. TCP/IP

- OSI: conceptually define services, interfaces, protocols
  - more layers → less efficient
- Internet: provide a successful implementation
  - don’t need so many layers in practice
Layer Violations and Not

- **Types of violations**
  - Higher layer interacts with layer below previous layer
    - e.g., application has special behavior when running on wireless link
    - sometimes provides useful performance enhancement
  - Lower layer interacts with higher layer
    - e.g., router (network layer device) routes based on transport or application data in packet
    - usually a very bad idea (see end-to-end argument)

- **Not violations**
  - Reducing copying of packets
End-to-End Argument

- How do you divide functionality across layers?
  - One answer: the end-to-end argument
  - Push as much functionality to higher layers as possible while still maintaining correctness and performance

Example: Reliable File Transfer

- Problem: transfer a file composed of blocks
- Solution 1: transfer each block and retry if necessary, and then concatenate them
- Solution 2: transfer the blocks, verify the entire file and retry the entire file if necessary
Discussion

- Solution 1 not complete
  - e.g., disk fills up after block is verified, but before block is written to disk
- The receiver has to check the entire file anyway
- Thus, full functionality can be entirely implemented at application layer; no need for reliability from lower layers
- However, verifying and retrying the individual blocks improves performance
  - e.g., high probability of lost block → retrying the entire file on error is very inefficient
- End-to-end argument says implement reliability in both layers in this case
Intuitive Justification

- Application has more information about the data and the semantics of the service it requires (e.g., level of reliability)
- A lower layer has more information about constraints in data transmission (e.g., packet size, error rate)
- Implementing a functionality at a lower level should have minimum performance impact on applications that do not use the functionality
  - otherwise, network service model becomes cluttered with demands of unpopular applications
  - do add functionality to lower layers that improves the performance of many applications
Internet Embodies
End-to-End Argument

- Network layer provides simple service model: best effort datagram (packet) delivery
- Only one higher level service implemented at transport layer: reliable, in order data delivery (TCP)
  - performance enhancement; used by a large variety of applications (Gnutella, ssh, SMTP, HTTP)
  - does not impact other applications (can use UDP)
- Everything else implemented at application level
Properties

- **Advantages**
  - Service model can be implemented by a large variety of network technologies
  - No per flow state in network
    - Robust to failure
    - Scalable to large numbers of flows, nodes

- **Disadvantages**
  - no differentiation of quality of service
    - vulnerable to selfish users
    - vulnerable to malicious users
  - only unicast routing
    - no multicast, anycast
Summary

- Layering is way to organize communication functionality; provides
  - Abstraction
  - Modularity
  - Reuse
- End-to-end argument
  - Push as much functionality to higher layers as possible while still maintaining correctness and performance
- Internet embodies end-to-end argument