CS 640: Introduction to Computer Networks

Aditya Akella

Lecture 2
Layering, Protocol Stacks, and Standards

1

Today's Lecture

- Layers and Protocols
- Standards and standardization process
- Applications

2

Network Communication: Lots of Functions Needed

- Links
- Multiplexing
- Routing
- Addressing/naming (locating peers)
- Reliability
- Flow control
- Fragmentation

How do you implement these functions?
Key: Layering and protocols

3

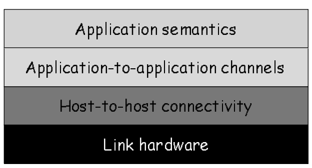
What is Layering?

- A way to deal with complexity
  - Add multiple levels of abstraction
  - Each level encapsulates some key functionality
  - And exports an interface to other components
  - Example?
- Layering: Modular approach to implementing network functionality by introducing abstractions
- Challenge: how to come up with the “right” abstractions?

4

Example of Layering

- Software and hardware for communication between two hosts

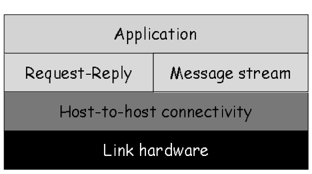


- Advantages:
  - Simplifies design and implementation
  - Easy to modify/evolve

5

What is a Protocol?

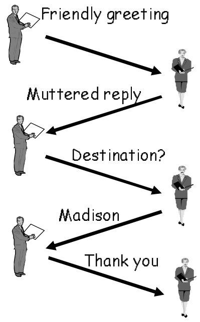
- Could be multiple abstractions at a given level
  - Build on the same lower level
  - But provide diferent service to higher layers
- Protocol: Abstract object or module in layered structure



6

Protocol

- Implements an agreement between parties on how communication should take place



7

1. Protocols Offer Interfaces

- Each protocol offers interfaces
  - One to higher-level protocols on the same end hosts
    - Expects one from the layers on which it builds
    - Interface characteristics, e.g. IP service model
  - A "peer interface" to a counterpart on destinations
    - Syntax and semantics of communications
    - (Assumptions about) data formats
- Protocols build upon each other
  - Adds value, improves functionality overall
    - E.g., a reliable protocol running on top of IP
  - Reuse, avoid re-writing
    - E.g., OS provides TCP, so apps don't have to rewrite

8

2. Protocols Necessary for Interoperability

- Protocols are the key to interoperability.
  - Networks are very heterogenous:

Ethernet: 3com, etc.	Hardware/link
Routers: cisco, juniper etc.	Network
App: Email, AIM, IE etc.	Application

  - The hardware/software of communicating parties are often not built by the same vendor
  - Yet they can communicate because they use the same protocol
    - Actually implementations could be different
    - But must adhere to same specification
- Protocols exist at many levels.
  - Application level protocols
  - Protocols at the hardware level

9

How do protocols/layers work?

- One or more protocols implement the functionality in a layer
  - Only horizontal (among peers) and vertical (in a host) communication
- Protocols/layers can be implemented and modified in isolation
- Each layer offers a service to the higher layer, using the services of the lower layer.
- "Peer" layers on different systems communicate via a protocol.
  - higher level protocols (e.g. TCP/IP, Appletalk) can run on multiple lower layers
  - multiple higher level protocols can share a single physical network

10

OSI Model

- One of the first standards for layering: OSI
- Breaks up network functionality into seven layers
- This is a "reference model"
  - For ease of thinking and implementation
- A different model, TCP/IP, used in practice

11

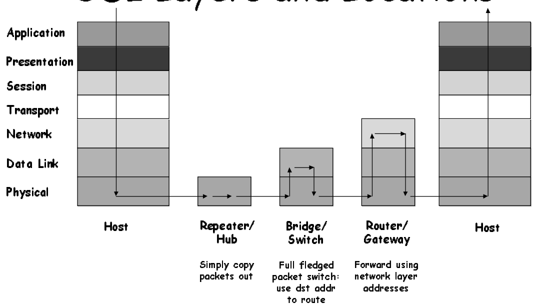
The OSI Standard: 7 Layers

1. **Physical**: transmit bits (link)
2. **Data link**: collect bits into frames and transmit frames (adaptor/device driver)
3. **Network**: route packets in a packet switched network
4. **Transport**: send messages across processes end2end
5. **Session**: tie related flows together
6. **Presentation**: format of app data (byte ordering, video format)
7. **Application**: application protocols (e.g. FTP)

- OSI very successful at shaping thought
- TCP/IP standard has been amazingly successful, and it's not based on a rigid OSI model

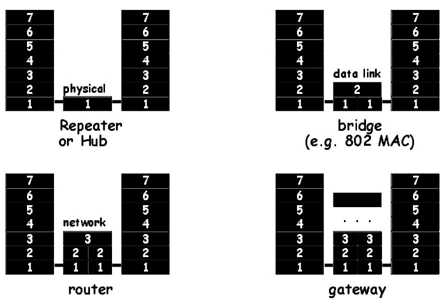
12

OSI Layers and Locations



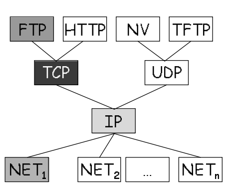
13

Internetworking Options



14

The Reality: TCP/IP Model



App protocols

Two transport protocols: provide logical channels to apps

Interconnection of n/w technologies into a single logical n/w

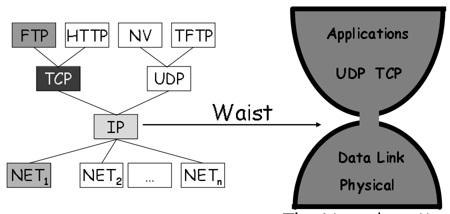
Network protocols implemented by a comb of hw and sw.

Note: No strict layering.

App writers can define apps that run on any lower level protocols.

15

The Thin Waist

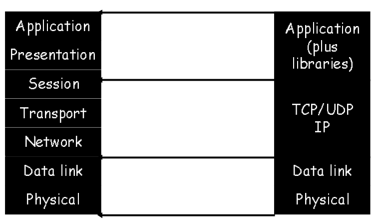


The Hourglass Model

The waist: minimal, carefully chosen functions.
Facilitates interoperability and rapid evolution

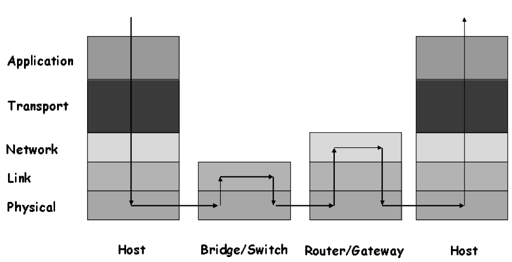
16

TCP/IP vs OSI



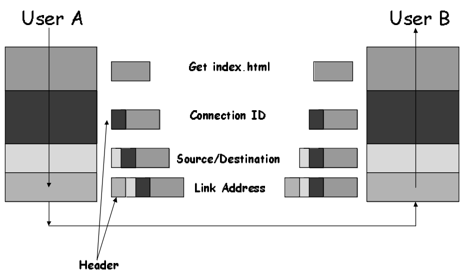
17

TCP/IP Layering



18

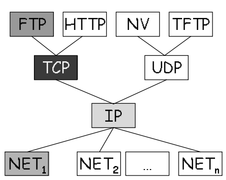
Layers & Encapsulation

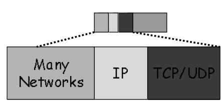


19

Protocol Demultiplexing

- Multiple choices at each layer
- How to know which one to pick?



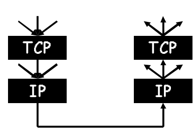


20

Multiplexing & Demultiplexing

- Multiple implementations of each layer
  - How does the receiver know what version/module of a layer to use?
- Packet header includes a demultiplexing field
  - Used to identify the right module for next layer
  - Filled in by the sender
  - Used by the receiver
- Multiplexing occurs at multiple layers. E.g., IP, TCP, …

V/HL	TOS	Length
ID		Flags/Offset
TTL	Prot.	H. Checksum
Source IP address		
Destination IP address		
Options..		



21

Layering vs Not

- Layer N may duplicate layer N-1 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
  - Many cross-layer assumptions, e.g. buffer management
- Layer 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/o compatibility library)

22

History of IP: The Early Days

- Early packet switching networks (61-72)
  - Definition of packet switching
  - Early ARPA net: up to tens of nodes (4 at the end of 1969; 15 at the end of 1972)
    - single network
    - Simple applications (first email program: 1972)
- Internetworking (72-80)
  - Several independent network implementations
  - Multiple networks with inter-networking: networks are independent, but need some rules for interoperability
  - Key concepts: best effort service, "stateless" routers, decentralized control (very different from telephones!)
  - Basis for Internet: TCP, IP, congestion control, DNS, …
  - Rapid growth: 10 to 100000 hosts in 10 years
    - NSFnet built a "backbone" to connect networks

23

Modern Times: Commercialization

- Industry interest in networking encourages first commercial network deployment.
  - In part also encouraged by NSFNET policies/backbone
- Introduction of the "Web" makes networks more accessible
  - Killer application
  - Good user interface that is accessible to anybody
  - Network access on every desktop and in every home
  - Shockingly recent - 1989, caught on in '92 or so
  - Spurred a lot of application
- Commercial success → multiple vendors
  - How to ensure inter-operability?

24

Standardization

- Crucial to network interoperability
  - An example we saw earlier: OSI model
- De facto standards
  - Standards are based on an existing system
  - Gives the company that developed the base system a big advantage
  - Often results in competing "standards" before the official standard is established
  - Popular in the early days
- A priori standards
  - Standards are defined first by a standards committee
  - Risk of defining standards that are untested or unnecessary
  - Standard may be available before there is serious use of the technology
  - There could still be competing standards
  - Most current standards

25

The Internet Engineering Task Force

- Internet Engineering Task Force.
  - decides what technology will be used in the Internet
  - based on working groups that focus on specific issues
  - encourages wide participation
- Request for Comments.
  - document that provides information or outlines standard
  - requests feedback from the community
  - can be "promoted" to standard under certain conditions
    - consensus in the committee
    - interoperating implementations
- "Rough consensus and working code"

26

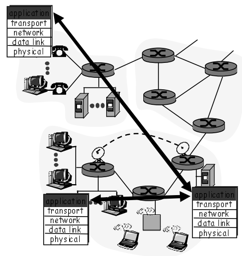
Higher Level Standards

- Many session/application level operations are relevant to networks
  - encoding: MPEG, encryption, ...
  - services: electronic mail, newsgroups, HTTP, ...
  - electronic commerce, ...
- Standards are as important as for "lower-level" networks: interoperability.
  - defined by some of the same bodies as the low-level standards, e.g. IETF

27

Applications and Application-Layer Protocols

- Application: communicating, distributed processes
  - Running in network hosts in "user space"
  - N/w functionality in kernel 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
  - Use services provided by lower layer protocols



28

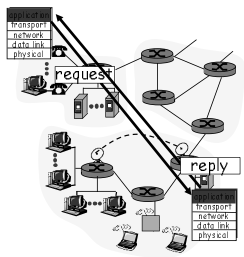
Client-Server Paradigm vs. P2P

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

- P2P is a very different model
  - *No* notion of client or server



29

Choosing the Transport Service

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

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

Bandwidth
- Some applications (e.g., multimedia) require a minimum amount of bandwidth to be "effective"
- Other applications ("elastic apps") will make use of whatever bandwidth they get

30

Transmission Control Protocol (TCP)

TCP
- Reliable – guarantee delivery
- Byte stream – in-order delivery
- Checksum for validity
- Setup connection followed by data transfer

Telephone Call
- Guaranteed delivery
- In-order delivery
- Setup connection followed by conversation

Example TCP applications
Web, Email, Telnet

31

User Datagram Protocol (UDP)

UDP
- No guarantee of delivery
- Not necessarily in-order delivery
- No validity guaranteed
- Must address each independent packet

Postal Mail
- Unreliable
- Not necessarily in-order delivery
- Must address each reply

Example UDP applications
Multimedia, voice over IP

32

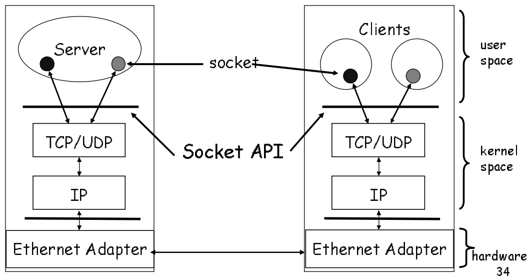
Transport Service Requirements of Common Applications

Application	Data loss	Bandwidth	Time Sensitive
file transfer	no loss	elastic	no
e-mail	no loss	elastic	no
web documents	no loss	elastic	no
real-time audio/video	loss-tolerant	audio: 5Kb-1Mb video:10Kb-5Mb	yes, 100's msec
stored audio/video	loss-tolerant	same as above	yes, few secs
interactive games	loss-tolerant	few Kbps	yes, 100's msec
financial apps	no loss	elastic	yes and no

33

Server and Client

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



34

Next Two Lectures

- Socket programming API (Ashutosh)
- Internet's design philosophy, more on applications and application performance

35