Chapter 1
Introduction

A note on the use of these Powerpoint slides:
We’re making these slides freely available to all (faculty, students, readers). They’re in PowerPoint form so you see the animations; and can add, modify, and delete slides (including this one) and slide content to suit your needs. They obviously represent a lot of work on our part. In return for use, we only ask the following:

- If you use these slides (e.g., in a class) that you mention their source (after all, we’d like people to use our book!)
- If you post any slides on a www site, that you note that they are adapted from (or perhaps identical to) our slides, and note our copyright of this material.

Thanks and enjoy! JFK/KWR

©All material copyright 1996-2016
J.F Kurose and K.W. Ross, All Rights Reserved
Chapter 1: roadmap

1.1 what is the Internet?
1.2 network edge
   ▪ end systems, access networks, links
1.3 network core
   ▪ packet switching, circuit switching, network structure
1.4 delay, loss, throughput in networks
1.5 protocol layers, service models
1.6 networks under attack: security
1.7 history
Protocol “layers”

Networks are complex, with many “pieces”:
- hosts
- routers
- links of various media
- applications
- protocols
- hardware, software

Question:
Is there any hope of organizing structure of network?

…. or at least our discussion of networks?
Organization of air travel

- ticket (purchase)
- baggage (check)
- gates (load)
- runway takeoff
- airplane routing

- ticket (complain)
- baggage (claim)
- gates (unload)
- runway landing
- airplane routing

- a series of steps
Layering of airline functionality

<table>
<thead>
<tr>
<th>Layer</th>
<th>Service</th>
<th>Layer</th>
<th>Service</th>
</tr>
</thead>
<tbody>
<tr>
<td>ticket (purchase)</td>
<td>ticket (complain)</td>
<td>ticket (purchase)</td>
<td>ticket</td>
</tr>
<tr>
<td>baggage (check)</td>
<td>baggage (claim)</td>
<td>baggage (claim)</td>
<td>baggage</td>
</tr>
<tr>
<td>gates (load)</td>
<td>gates (unload)</td>
<td>gates (unload)</td>
<td>gate</td>
</tr>
<tr>
<td>runway (takeoff)</td>
<td>runway (land)</td>
<td>runway (land)</td>
<td>takeoff/landing</td>
</tr>
<tr>
<td>airplane routing</td>
<td>airplane routing</td>
<td>airplane routing</td>
<td>airplane routing</td>
</tr>
</tbody>
</table>

**layers:** each layer implements a service

- via its own internal-layer actions
- relying on services provided by layer below
Why layering?

dealing with complex systems:

- explicit structure allows identification, relationship of complex system’s pieces
  - layered *reference model* for discussion

- modularization eases maintenance, updating of system
  - change of implementation of layer’s service transparent to rest of system
  - e.g., change in gate procedure doesn’t affect rest of system

- layering considered harmful?
Internet protocol stack

- **application**: supporting network applications
  - FTP, SMTP, HTTP
- **transport**: process-process data transfer
  - TCP, UDP
- **network**: routing of datagrams from source to destination
  - IP, routing protocols
- **link**: data transfer between neighboring network elements
  - Ethernet, 802.11 (WiFi), PPP
- **physical**: bits “on the wire”
ISO/OSI reference model

- **presentation**: allow applications to interpret meaning of data, e.g., encryption, compression, machine-specific conventions
- **session**: synchronization, checkpointing, recovery of data exchange
- Internet stack “missing” these layers!
  - these services, *if needed*, must be implemented in application
  - needed?
"Top-down" approach means we will first learn the application layer and then learn about lower layers.

*slide borrowed from Professor Koushik Kar in ECSE
Encapsulation

message
segment
datagram
frame

source

application
transport
network
link
physical

destination

application
transport
network
link
physical

Encapsulation

switch

router

network
link
physical

frame

message
segment
datagram
frame
Chapter 2
Application Layer

A note on the use of these Powerpoint slides:
We’re making these slides freely available to all (faculty, students, readers). They’re in PowerPoint form so you see the animations; and can add, modify, and delete slides (including this one) and slide content to suit your needs. They obviously represent a lot of work on our part. In return for use, we only ask the following:

- If you use these slides (e.g., in a class) that you mention their source (after all, we’d like people to use our book!)
- If you post any slides on a www site, that you note that they are adapted from (or perhaps identical to) our slides, and note our copyright of this material.

Thanks and enjoy! JFK/KWR

All material copyright 1996-2016 © J.F Kurose and K.W. Ross, All Rights Reserved
Chapter 2: outline

2.1 principles of network applications
2.2 Web and HTTP
2.3 electronic mail
   • SMTP, POP3, IMAP
2.4 DNS
2.5 P2P applications
2.6 video streaming and content distribution networks
2.7 socket programming with UDP and TCP
Chapter 2: application layer

our goals:
- conceptual, implementation aspects of network application protocols
  - transport-layer service models
  - client-server paradigm
  - peer-to-peer paradigm
  - content distribution networks

- learn about protocols by examining popular application-level protocols
  - HTTP
  - FTP
  - SMTP / POP3 / IMAP
  - DNS

- creating network applications
  - socket API
Some network apps

- e-mail
- web
- text messaging
- remote login
- P2P file sharing
- multi-user network games
- streaming stored video (YouTube, Hulu, Netflix)
- voice over IP (e.g., Skype)
- real-time video conferencing
- social networking
- search
- ...
- ...
Creating a network app

write programs that:
- run on (different) end systems
- communicate over network
- e.g., web server software communicates with browser software

no need to write software for network-core devices
- network-core devices do not run user applications
- applications on end systems allows for rapid app development, propagation
Application architectures

possible structure of applications:

- client-server
- peer-to-peer (P2P)
**Client-server architecture**

**server:**
- always-on host
- permanent IP address
- data centers for scaling

**clients:**
- communicate with server
- may be intermittently connected
- may have dynamic IP addresses
- do not communicate directly with each other
P2P architecture

- no always-on server
- arbitrary end systems directly communicate
- peers request service from other peers, provide service in return to other peers
  - self scalability – new peers bring new service capacity, as well as new service demands
- peers are intermittently connected and change IP addresses
  - complex management
Processes communicating

*process*: program running within a host

- within same host, two processes communicate using *inter-process communication* (defined by OS)
- processes in different hosts communicate by exchanging messages

*client process*: process that initiates communication

*server process*: process that waits to be contacted

*clients, servers*

aside: applications with P2P architectures have client processes & server processes
Sockets

- process sends/receives messages to/from its **socket**
- socket analogous to door
  - sending process shoves message out door
  - sending process relies on transport infrastructure on other side of door to deliver message to socket at receiving process
Addressing processes

- to receive messages, process must have *identifier*
- host device has unique 32-bit IP address
- **Q:** does IP address of host on which process runs suffice for identifying the process?
  - **A:** no, *many* processes can be running on same host
- *identifier* includes both IP address and port numbers associated with process on host.
- example port numbers:
  - HTTP server: 80
  - mail server: 25
- to send HTTP message to gaia.cs.umass.edu web server:
  - IP address: 128.119.245.12
  - port number: 80
- more shortly…

Application Layer 2-21
App-layer protocol defines

- **types of messages exchanged**, 
  - e.g., request, response

- **message syntax:** 
  - what fields in messages & how fields are delineated

- **message semantics** 
  - meaning of information in fields

- **rules** for when and how processes send & respond to messages

**open protocols:**
- defined in RFCs
- allows for interoperability
- e.g., HTTP, SMTP

**proprietary protocols:**
- e.g., Skype
Transport (TCP/UDP)
Network (IP)
Link (Ethernet)
Physical

application (www browser, email client)

packet capture (pcap)

packet analyzer
Things to do

- Download Wireshark from [www.wireshark.org](http://www.wireshark.org) and try capturing packets to/from your computer with it