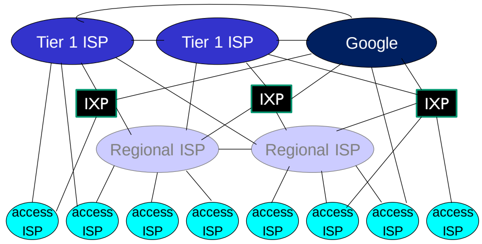
SOME HIGHLIGHTS FROM CHAPTER ONE

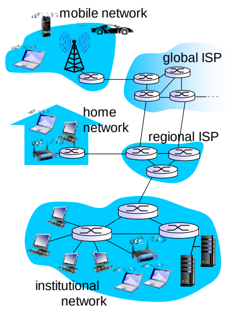
Internet structure: network of networks



- at center: small # of well-connected large networks
  - "tier-1" commercial ISPs (e.g., Level 3, Sprint, AT&T, NTT), national & international coverage
  - "internet exchange points" (IXPs): meeting points of multiple ISPs
  - content provider network (e.g, Google): private network that connects it data centers to Internet, often bypassing tier-1, regional ISPs

A closer look at network structure:

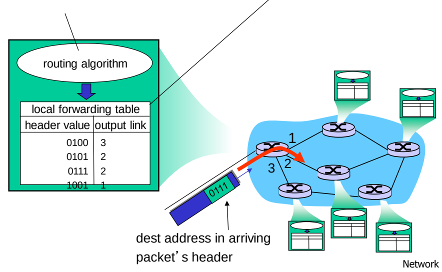
- *network edge:*
  - hosts: clients and servers
  - servers often in data centers
- *access networks, physical media:* wired, wireless communication links
- *network core:*
  - interconnected routers
  - network of networks



Two key network-core functions

routing: determines source-destination route taken by packets
- *routing algorithms*

forwarding: move packets from router's input to appropriate router output



"Real" Internet delays, routes

traceroute: gaia.cs.umass.edu to www.eurecom.fr

Do some traceroutes from exotic countries at www.traceroute.org

3 delay measurements from gaia.cs.umass.edu to cs-gw.cs.umass.edu

```
1  cs-gw (128.119.240.254)  1 ms  1 ms  2 ms
2  border1-rt-fa5-1-0.gw.umass.edu (128.119.3.145)  1 ms  1 ms  2 ms
3  cht-vbns.gw.umass.edu (128.119.3.130)  6 ms 5 ms 5 ms
4  jn1-at1-0-0-19.wor.vbns.net (204.147.132.129)  16 ms 11 ms 13 ms
5  jn1-so7-0-0-0.wae.vbns.net (204.147.136.136)  21 ms 18 ms 18 ms
6  abilene-vbns.abilene.ucaid.edu (198.32.11.9)  22 ms  18 ms  22 ms
7  nycm-wash.abilene.ucaid.edu (198.32.8.46)  22 ms  22 ms  22 ms
8  62.40.103.253 (62.40.103.253)  104 ms 109 ms 106 ms
9  de2-1.de1.de.geant.net (62.40.96.129)  109 ms 102 ms 104 ms
10  de.fr1.fr.geant.net (62.40.96.50)  113 ms 121 ms 114 ms
11  renater-gw.fr1.fr.geant.net (62.40.103.54)  112 ms  114 ms  112 ms
12  nio-n2.cssi.renater.fr (193.51.206.13)  111 ms  114 ms  116 ms
13  nice.cssi.renater.fr (195.220.98.102)  123 ms  125 ms  124 ms
14  r3t2-nice.cssi.renater.fr (195.220.98.110)  126 ms  126 ms  124 ms
15  eurecom-valbonne.r3t2.ft.net (193.48.50.54)  135 ms  128 ms  133 ms
16  194.214.211.25 (194.214.211.25)  126 ms  128 ms  126 ms
17  * * *
18  * * *
19  fantasia.eurecom.fr (193.55.113.142)  132 ms  128 ms  136 ms
```

trans-oceanic link (between hops 7 and 8)

* means no response (probe lost, router not replying)

Packet switching and circuit switching

- packet-switching: hosts break application-layer messages into *packets*
  - forward packets from one router to the next, across links on path from source to destination
  - each packet transmitted at full link capacity
- circuit-switching: end-end resources allocated to, reserved for "call" between source & destination:
  - Dedicated resources: no sharing
  - circuit-like (guaranteed) performance
  - Circuit segment idle if not used by call *(no sharing)*
  - Commonly used in traditional telephone networks
  - Virtual circuits may be used in modern communications

Network protocols

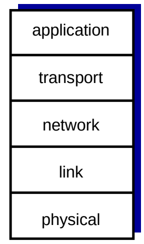
A network protocol is a set of rules governing the operations of the network.

Internet is in layered architecture, each layer has a set of protocols.

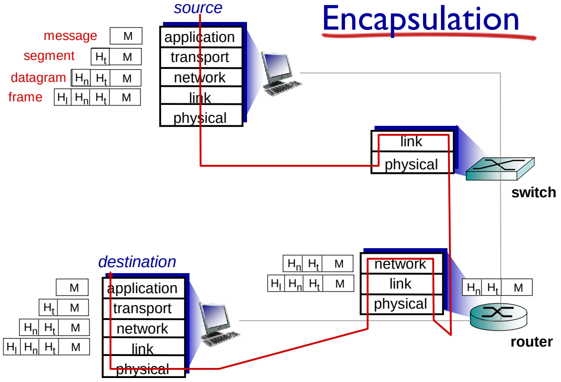
For example: at transport layer, TCP or UDP, at networking layer, IP, at data link layer, Ethernet or WiFi.

Internet protocol stack

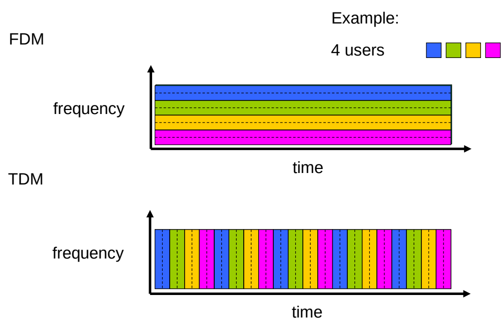
- *application:* supporting network applications
  - FTP, SMTP, HTTP
- *transport:* process-process data transfer
  - TCP, UDP
- *network:* routing of datagrams from source to destination
  - IP, routing protocols
- *link:* data transfer between neighboring network elements
  - Ethernet, 802.11 (WiFi)
- *physical:* bits "on the wire" or "in the air"



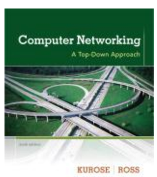
Encapsulation



FDM versus TDM



Chapter 2 Application Layer



Computer Networking: A Top Down Approach
6th edition
Jim Kurose, Keith Ross
Addison-Wesley
March 2012

A note on the use of these ppt 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

The course notes are adapted for Bucknell's CSCI 363
Xiannong Meng
Spring 2016

Chapter 2: outline

2.1 principles of network applications
  2.1.1 client-server model
2.6 P2P applications
2.2 Web and HTTP
2.3 FTP
2.4 electronic mail
- SMTP, POP3, IMAP

2.7 socket programming with UDP and TCP
2.5 DNS

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
- …
- …

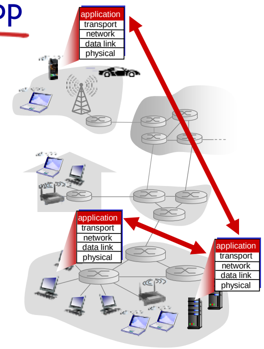
Creating a network app

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

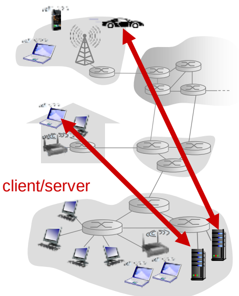


Application architectures

possible structure of applications:

- client-server
- peer-to-peer (P2P)

Client-server architecture



server:

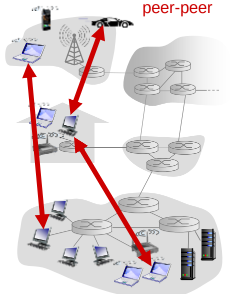
- always-on host
- wait for requests from clients
- permanent IP address
- server examples:
  - www.bucknell.edu, www.google.com

clients:

- client initiates the communication
- may be intermittently connected, dynamic (or static) IP
- do not communicate directly with each other

P2P architecture

- *no* always-on server
- arbitrary end systems directly communicate with each other
- 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
- example:
  - Skype, text message
- no server(s) at all?



Processes communicating

process: program running within a host

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

clients, servers

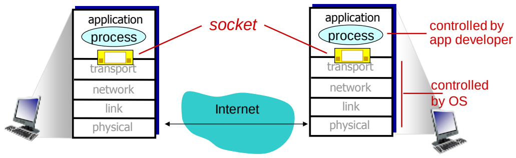
client process: process that initiates communication

server process: process that waits to be contacted

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

Sockets

- process sends/receives messages to/from its socket
- socket analogous to mailbox at your house or LC
  - sending process puts the message in the mailbox
  - sending process relies on transport infrastructure between the sending mailbox and receiving mailbox to deliver message to socket at receiving process



Some client-server examples

- Client-server in C
  - http://www.eg.bucknell.edu/~cs363/2016-spring/code/client-server-c/
- Client-server in Python
  - http://www.eg.bucknell.edu/~cs363/2016-spring/code/client-server-python/
- Web client-server in C
  - http://www.eg.bucknell.edu/~cs363/2016-spring/code/web-client-server-c/

What transport service does an app need?

data integrity

- some apps (e.g., file transfer, web transactions) require 100% reliable data transfer
- other apps (e.g., audio) can tolerate some loss

timing

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

throughput

- some apps (e.g., multimedia) require minimum amount of throughput to be "effective"
- other apps ("elastic apps") make use of whatever throughput they get

security

- encryption, data integrity, …

Securing TCP

TCP & UDP

- no encryption
- cleartext passwds sent into socket traverse Internet in cleartext

TLS and SSL

- provides encrypted TCP connection
- data integrity
- end-point authentication

SSL is at app layer

- Apps use SSL libraries, which "talk" to TCP

SSL socket API

- cleartext passwds sent into socket traverse Internet encrypted
- See Chapter 8