NETWORKING FUNDAMENTALS

George Porter
Jan 9, 2019

UC San Diego

1

ATTRIBUTION

- These slides incorporate material from:
    - Alex C. Snoeren, UC San Diego
    - Michael Freedman and Kyle Jamieson, Princeton University
    - Internet Society
    - Computer Networking: A Top Down Approach 6th edition

UC San Diego

2

ANNOUNCEMENTS

Peterson & Davie 1.3 (network architectures), 1.4, 1.5, 2.5, 3.2

Class Q&A forum (via Google) now available (link off course web page)

UC San Diego

3

Outline

1. Packet switching
2. Addressing
3. Performance
4. TCP and sockets
5. Reliable transmission

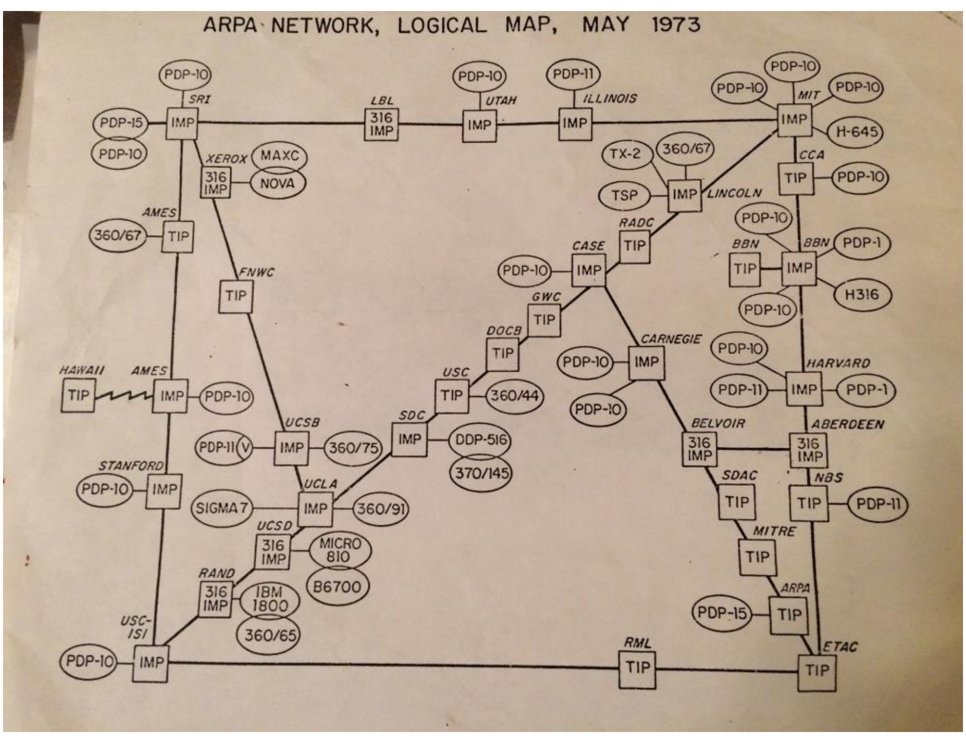
4

BRIEF HISTORY OF THE INTERNET

- 1968 - DARPA (Defense Advanced Research Projects Agency) contracts with BBN (Bolt, Beranek & Newman) to create ARPAnet
- 1970 - First five nodes:
  - UCLA
  - Stanford
  - UC Santa Barbara
  - U of Utah, and
  - BBN
- 1974 - TCP specification by Vint Cerf
- 1984 – On January 1, the Internet with its 1000 hosts converts en masse to using TCP/IP for its messaging

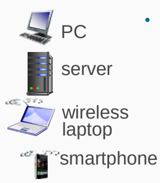
Data from the Internet Society

5

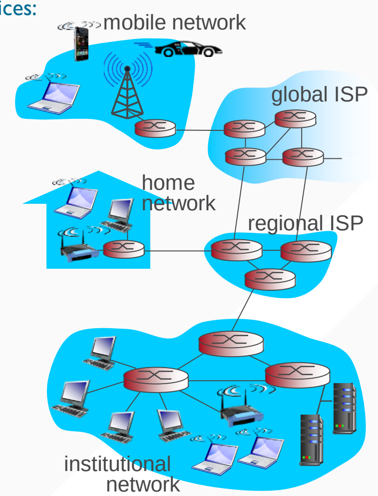


6

AN INTER-NETWORK



- billions of connected computing devices:
  - *hosts = end systems*
  - running *network apps*



- *communication links*
  - fiber, copper, radio, satellite
  - transmission rate: *bandwidth*

- *Packet switches:* forward packets (chunks of data)
  - *routers* and *switches*

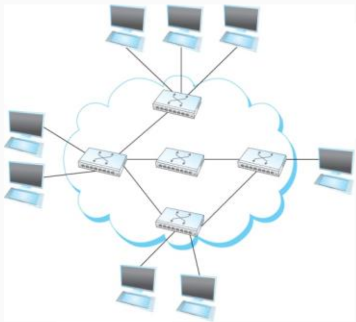
7

PACKET SWITCHING

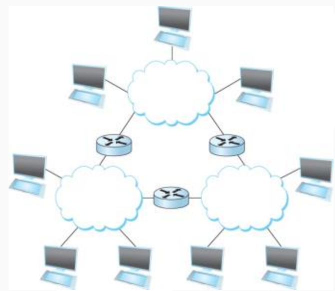
- Data transmitted from source to destination in discrete *packets*
  - Variable size, usually a maximum transmission unit (MTU) of ~1500 bytes
  - 1 GB file is approx 715,000 packets
- Each packet routed independently
  - Has its own source and destination *address* used to find the route to the ultimate destination

8

NETWORK TERMINOLOGY



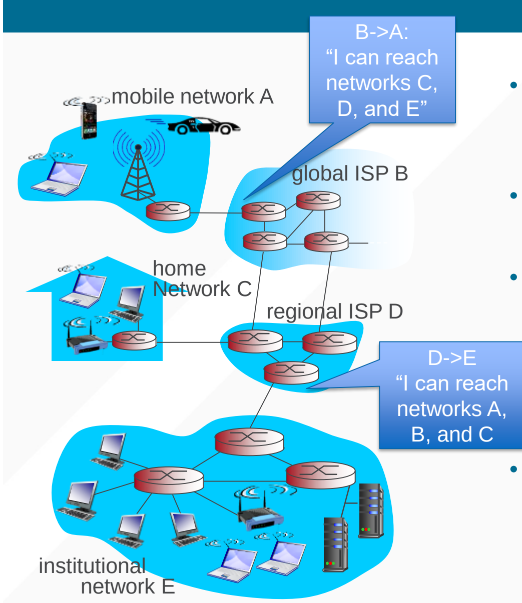
A single packet-switched network



Interconnection of networks

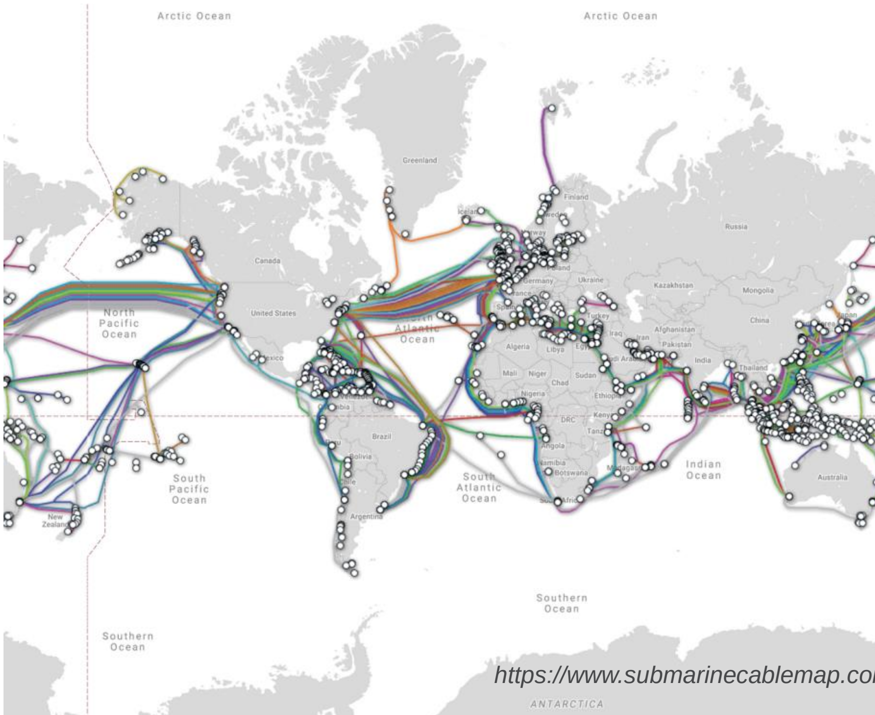
9

ROUTING PACKETS BETWEEN NETWORKS



- Networks use *Border Gateway Protocol (BGP)* to announce reachability
- Each network talks just with its neighbors
- Goal is to get a packet to the destination network
  - It is up to that destination network to get individual packets to their ultimate destination
- Back-to-back packets from the same "connection" might take different paths!
  - Might arrive out of order too

10

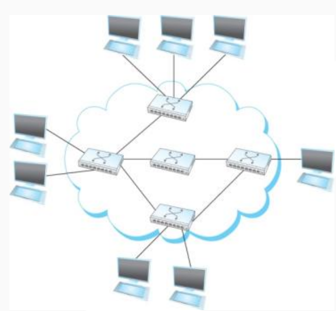


https://www.submarinecablemap.com/

11

ROUTING PACKETS WITHIN A SINGLE NETWORK

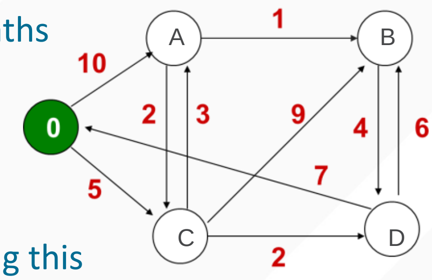
- E.g. UCSD, San Diego Comcast, a Google datacenter
- Packet switches communicate with each other using an *intra-domain* protocol called a Link-state protocol
  - Each packet switch sends a list of its neighbors to every other switch via a *broadcast*
  - As a result, each switch has the same "map" of what the network's *topology* looks like

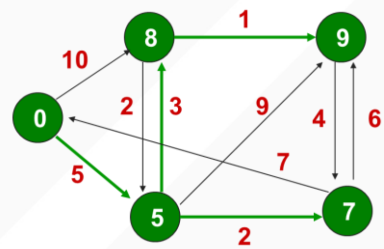


12

ROUTING PACKETS WITHIN A SINGLE NETWORK

- Each packet switch uses its local topology map to choose paths to each destination
- Typically using Dijkstra's shortest path algorithm
- Packets are forwarded along this shortest path *tree*

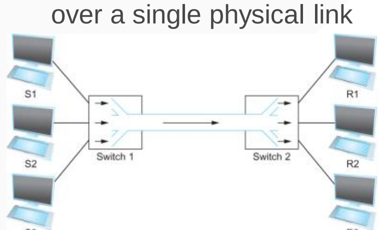


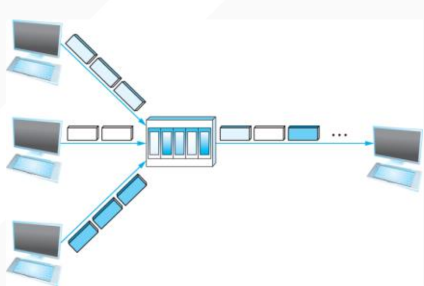


13

RESOURCE SHARING

Multiplexing multiple logical flows over a single physical link





- Resource: links and nodes
- How to share a link?
  - Multiplexing
  - De-multiplexing
  - Queueing

14

Outline

1. Packet switching
2. Addressing
3. Performance
4. TCP and sockets
5. Reliable transmission

15

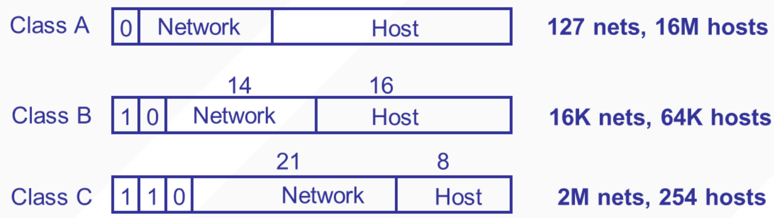
ADDRESSING CONSIDERATIONS

- Fixed length or variable length addresses?
- Issues:
  - Flexibility
  - Processing costs
  - Header size
- Engineering choice: IP uses fixed length addresses
- 32-bits in an IPv4 address
  - Dotted decimal format a.b.c.d
  - Each represent 8 bits of address
- Hierarchical: Network part and host part
  - E.g. IP address 128.54.70.238
  - 128.54 refers to the UCSD campus network
  - 70.238 refers to the host ieng6.ucsd.edu
- Which part is network *vs.* host?

16

CLASS-BASED ADDRESSING

- Most significant bits determines "class" of address



- Special addresses
  - Class D (1110) for multicast, Class E (1111) experimental
  - 127.0.0.1: local host (a.k.a. the loopback address)
  - Host bits all set to 0: network address
  - Host bits all set to 1: broadcast address

17

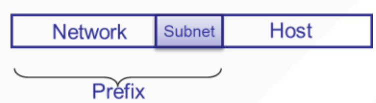
IP FORWARDING TABLES

- Router needs to know where to forward a packet
- Forwarding table contains:
  - List of network names and next hop routers
  - Local networks have entries specifying which interface
    - Link-local hosts can be delivered with Layer-2 forwarding
- E.g. www.ucsd.edu address is 132.239.180.101
  - Class B address – class + network is 132.239
  - Lookup 132.239 in forwarding table
  - Prefix – part of address that really matters for routing

18

SUBNETTING

- Individual networks may be composed of several LANs
  - Only want traffic destined to local hosts on physical network
  - Routers need a way to know which hosts on which LAN
- Networks can be arbitrarily decomposed into subnets
  - Each subnet is simply a prefix of the host address portion
  - Subnet prefix can be of any length, specified with netmask



19

SUBNET ADDRESSES

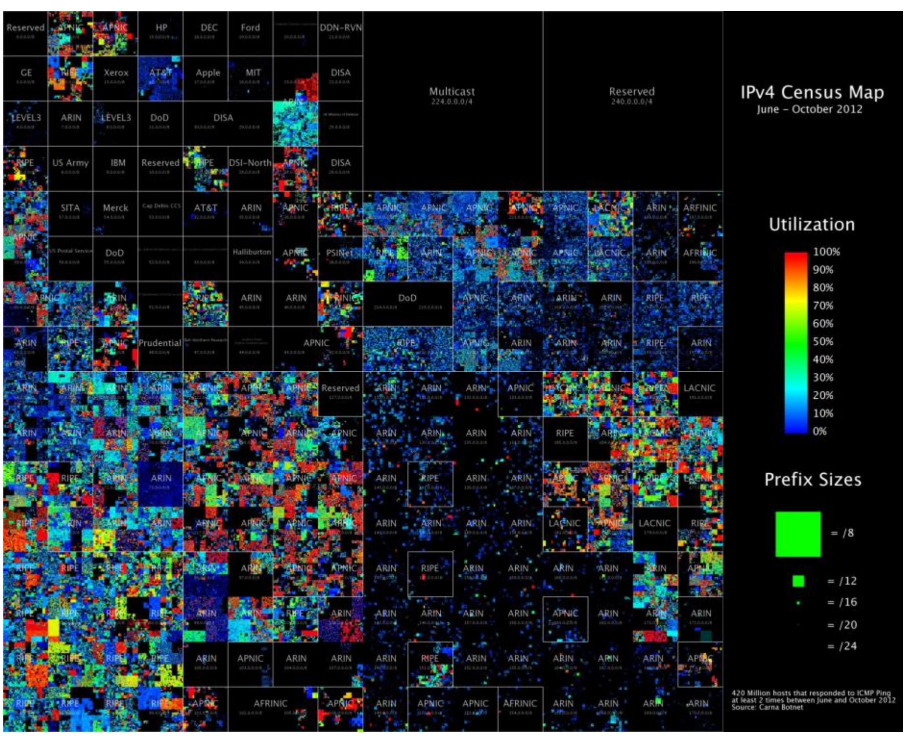
- Every (sub)network has an address and a netmask
  - Netmask tells which bits of the network address is important
  - Convention suggests it be a proper prefix
- Netmask written as an all-ones IP address
  - E.g., Class B netmask is 255.255.0.0
  - Sometimes expressed in terms of number of 1s, e.g., /16
- Need to size subnet appropriately for each LAN
  - Only have remaining bits to specify host addresses

20

IP ADDRESS PROBLEM (1991)

- Address space depletion
  - In danger of running out of classes A and B
- Why?
  - Class C too small for most organizations (only ~250 addresses)
  - Very few class A – very careful about giving them out (who has 16M hosts anyway?)
  - Class B – greatest problem

21



22

CIDR

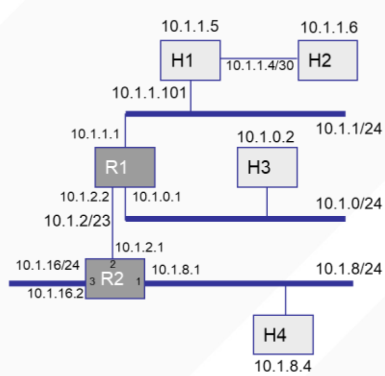
- Classless Inter-Domain Routing (1993)
  - Networks described by variable-length prefix and length
  - Allows arbitrary allocation between network and host address

  - e.g. 10.95.1.2 contained within 10.0.0.0/8:
    - 10.0.0.0 is network and remainder (95.1.2) is host
- Pro: Finer grained allocation; aggregation
- Con: More expensive lookup: longest prefix match

FORWARDING TABLE EXAMPLE (R2)

- Packet to 10.1.1.6
- Matches 10.1.0.0/23

Destination	Next Hop
127.0.0.1	loopback
Default or 0/0	10.1.16.1
10.1.8.0/24	interface1
10.1.2.0/23	interface2
10.1.0.0/23	10.1.2.2
10.1.16.0/24	interface3

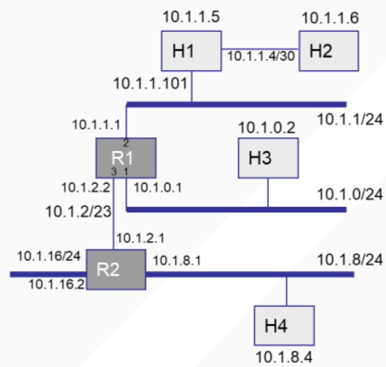


FORWARDING TABLE EXAMPLE 2

- Packet to 10.1.1.6
- Matches 10.1.1.4/30
  - Longest prefix match

Routing table at R1

Destination	Next Hop
127.0.0.1	loopback
Default or 0/0	10.1.2.1
10.1.0.0/24	interface1
10.1.1.0/24	interface2
10.1.2.0/23	interface3
10.1.1.4/30	10.1.1.101



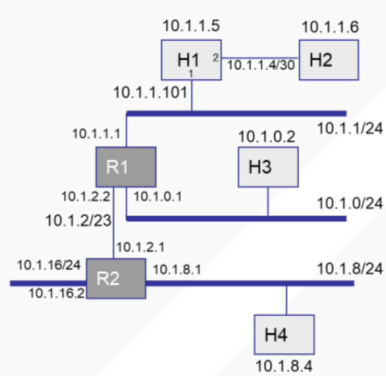
25

FORWARDING TABLE EXAMPLE 3

- Packet to 10.1.1.6
- Direct route
  - Longest prefix match

Routing table at H1

Destination	Next Hop
127.0.0.1	loopback
Default or 0/0	10.1.1.1
10.1.1.0/24	interface1
10.1.1.4/30	interface2



26

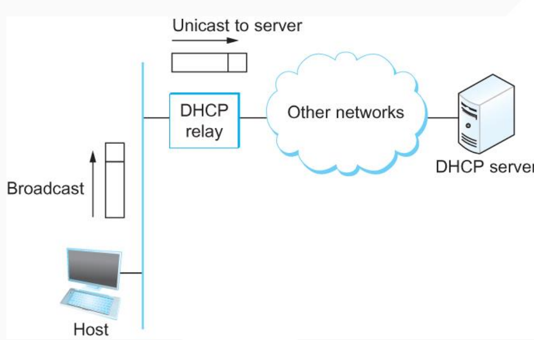
ASSIGNING ADDRESSES VIA DHCP

- DHCP server is responsible for providing configuration information to hosts
- There is at least one DHCP server for an administrative domain
- DHCP server maintains a pool of available addresses

27

DHCP IN ACTION

- Newly booted or attached host sends DHCPDISCOVER message to a special IP address (255.255.255.255)
- DHCP relay agent unicasts the message to DHCP server and waits for the response



28

DNS HOSTNAME VERSUS IP ADDRESS

- **DNS host name** (e.g. www.cs.ucsd.edu)
  - **Mnemonic** name appreciated by humans
  - **Variable length**, full alphabet of characters
  - Provides **little** (if any) information about **location**
- **IP address** (e.g. 128.112.136.35)
  - Numerical address appreciated by **routers**
  - **Fixed length**, decimal number
  - **Hierarchical** address space, related to host **location**

29

MAPPING NAMES TO ADDRESSES

```
GETADDRINFO(3)            Linux Programmer's Manual            GETADDRINFO(3)

NAME
       getaddrinfo,  freeaddrinfo,  gai_strerror - network address and service
       translation

SYNOPSIS
       #include <sys/types.h>
       #include <sys/socket.h>
       #include <netdb.h>

       int getaddrinfo(const char *node, const char *service,
                       const struct addrinfo *hints,
                       struct addrinfo **res);

       void freeaddrinfo(struct addrinfo *res);

       const char *gai_strerror(int errcode);
```

30

LINKED LIST OF 'ADDRINFO' STRUCTS

```
struct addrinfo {
    int              ai_flags;
    int              ai_family;
    int              ai_socktype;
    int              ai_protocol;
    socklen_t        ai_addrlen;
    struct sockaddr *ai_addr;
    char            *ai_canonname;
    struct addrinfo *ai_next;
};
```

- Q: Why a linked list?
- Q: Which of the multiple results should you use?

31

Outline

1. Packet switching
2. Addressing
3. Performance
4. TCP and sockets
5. Reliable transmission

32

PERFORMANCE METRICS

- Bandwidth: number of bits transmitted per unit of time
- Latency = Propagation + Transmit + Queue
  - Propagation = Distance/SpeedOfLight(*)
  - Transmit = 1 bit/Bandwidth
- Overhead
  - # secs for CPU to put message on wire
- Error rate
  - Probability P that message will not arrive intact

* In that particular medium

33

BANDWIDTH VS. LATENCY

1 Byte Object

	Latency: 1 ms	Latency: 100 ms
Bandwidth: 1 Mbps	1,008 μs	100,008 μs
Bandwidth: 100 Mbps	1,000 μs	100,000 μs

10 MB Object

	Latency: 1 ms	Latency: 100 ms
Bandwidth: 1 Mbps	80.001 s	80.1 s
Bandwidth: 100 Mbps	.801 s	.9 s

34

TERMINOLOGY STYLE

- Mega versus Mega, Kilo versus Kilo
  - Computer architecture: Mega → 2^20, Kilo → 2^10
  - Computer networks: Mega → 10^6, Kilo → 10^3
- Mbps versus MBps
  - Networks: typically megabits per second
  - Architecture: typically megabytes per second
- Bandwidth versus throughput
  - Bandwidth: available over link
  - Throughput: available to application

35

Outline

1. Packet switching
2. Addressing
3. Performance
4. TCP and sockets
5. Reliable transmission

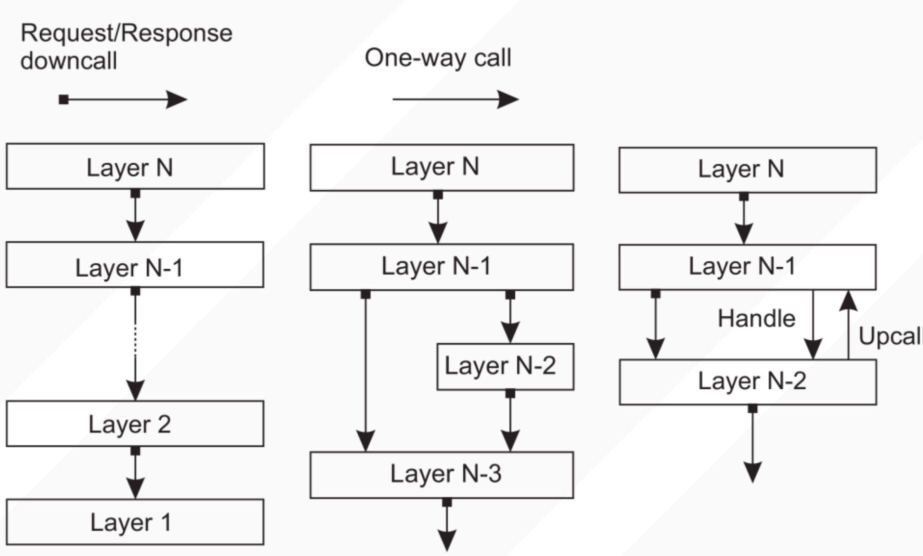
36

LAYERING: A MODULAR APPROACH TO COMM

- Sub-divide responsibilities
  - Each layer relies on services from layer below
  - Each layer exports services to layer above
- Interface between layers defines interaction
  - Hides implementation details (encapsulation)
  - Layers can change without disturbing other layers (modularity)
- Interface among peers in a layer is a protocol
  - If peers speak same protocol, they can interoperate

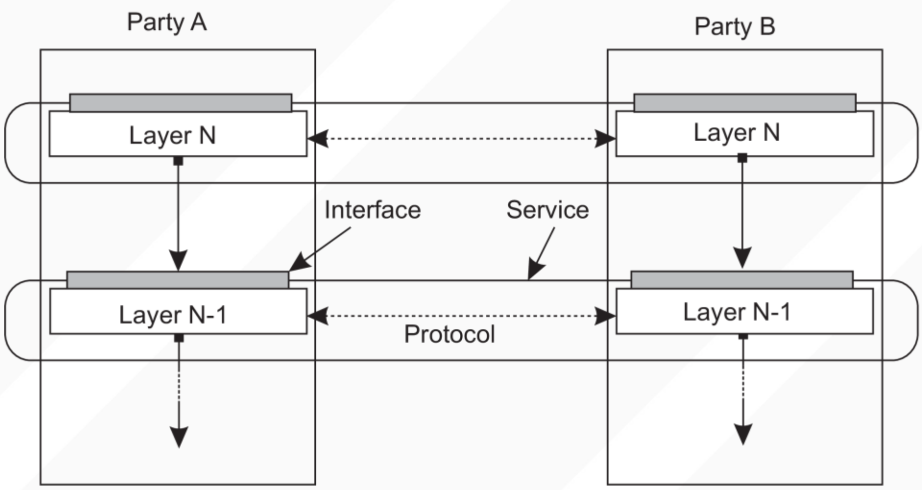
37

LAYERED ARCHITECTURES



38

SERVICE AND PROTOCOL INTERFACES



39

WHAT ARE PROTOCOLS?

Image from public domain

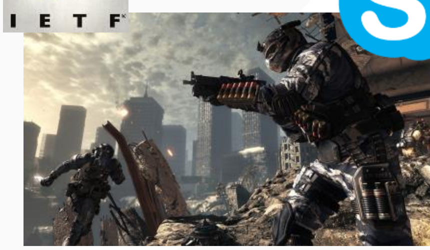
- Explicit and implicit conventions for how to communicate
  - Not for what is communicated

40

WHERE DO PROTOCOLS COME FROM?

- Standards bodies
  - IETF: Internet Engineering Task Force
  - ISO: International Standards Organization
- Community efforts
  - "Request for comments"
  - Bitcoin
- Corporations/industry
  - RealAudio™, Call of Duty multiplayer, Skype

International Organization for Standardization



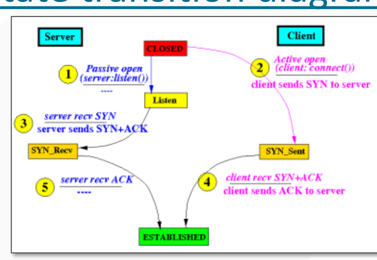
41

HOW ARE PROTOCOLS SPECIFIED?

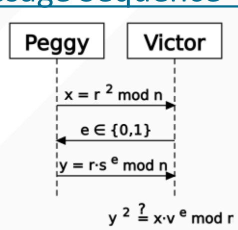
Prose/BNF

```
3.2.  HEADER FIELD DEFINITIONS

     These rules show a field meta-syntax, without regard for the
particular  type  or internal syntax.  Their purpose is to permit
detection of fields; also, they present to  higher-level  parsers
an image of each field as fitting on one line.

field       =  field-name ":" [ field-body ] CRLF

field-name  =  1*<any CHAR, excluding CTLs, SPACE, and ":">

field-body  =  field-body-contents
               [CRLF LWSP-char field-body]

field-body-contents =
              <the ASCII characters making up the field-body, as
               defined in the following sections, and consisting
               of combinations of atom, quoted-string, and
               specials tokens, or else consisting of texts>
```

State transition diagrams



Message Sequence Diagram



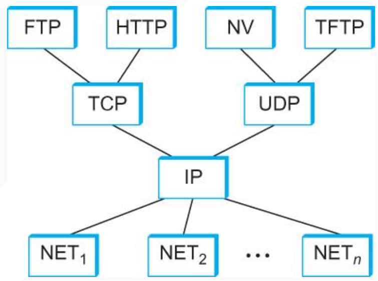
By Stefan Birkner, cc-by-sa-2.5,2.0,1.0

Packet formats

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Version|  IHL  |Type of Service|          Total Length         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Identification        |Flags|      Fragment Offset    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Time to Live |    Protocol   |         Header Checksum       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Source Address                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Destination Address                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Options                    |    Padding    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

42

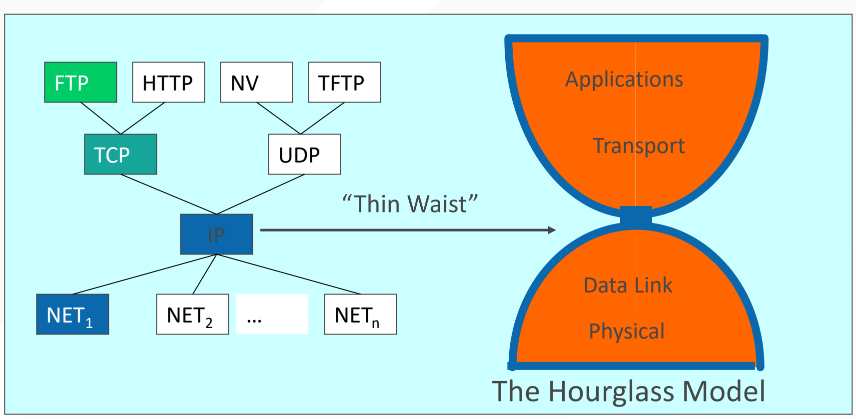
ROLE OF LAYERING IN PROTOCOLS



- Each layer offers useful semantics to layer above
  - IP gets packets to a destination host/server on the Internet (but is unreliable)
  - TCP uses IP to offer *reliable, in-order bytestream* abstraction
  - TCP useful for file transfer, as well as HTTP/web

43

INTERNET PROTOCOL SUITE



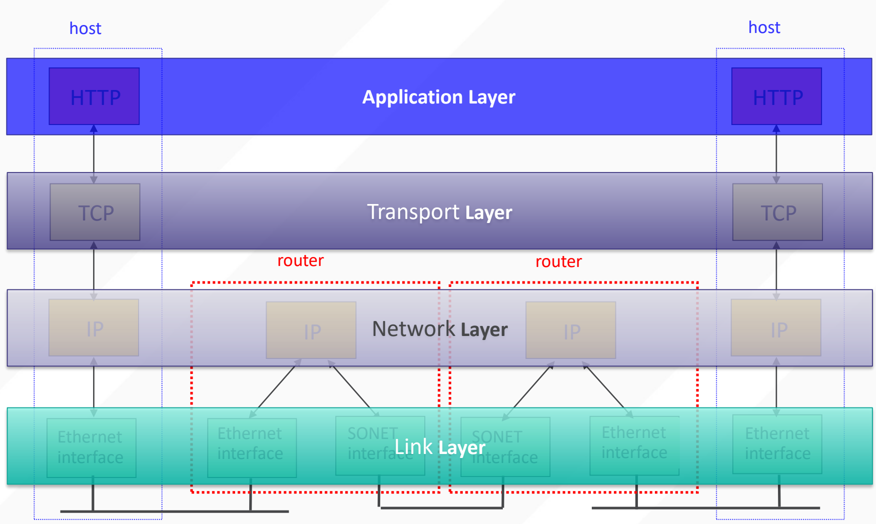
44

TRANSMISSION CONTROL PROTOCOL (TCP)

- Remember 1 GB ~ 715,000 packets?
  - Don't want to keep track of each packet, whether it got there, did it get lost? Did some get reordered??
- TCP offers *infinite bytestream* abstraction
  - If you put *N* bytes into TCP connection as sender, those *N* bytes will arrive to the destination in order, without loss, and without corruption
    - Compelling abstraction for higher-level applications such as web, video, gaming, …

45

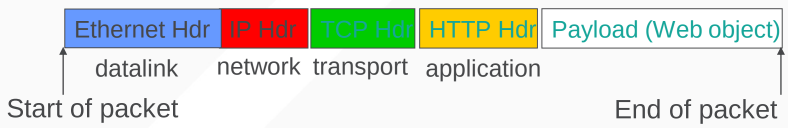
TCP/IP PROTOCOL STACK



46

ENCAPSULATION VIA HEADERS

- Typical web packet:



- Notice that layers add overhead
  - Space (headers), effective bandwidth
  - Time (processing headers), latency

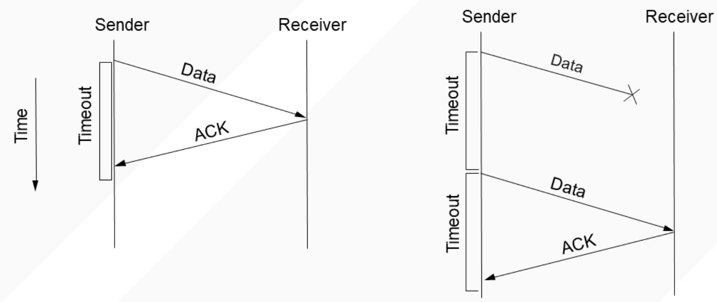
47

Outline

1. Packet switching
2. Addressing
3. Performance
4. TCP and sockets
5. Reliable transmission

48

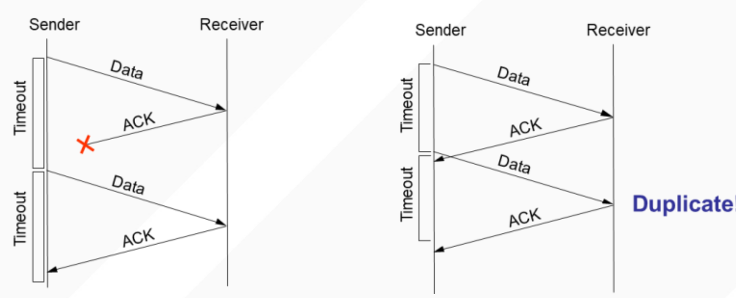
SIMPLE IDEA: ARQ



- Receiver sends acknowledgments (ACKs)
  - Sender "times out" and retransmits if it doesn't receive them
- Basic approach is generically referred to as Automatic Repeat Request (ARQ)

49

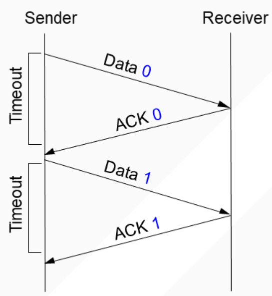
NOT SO FAST!

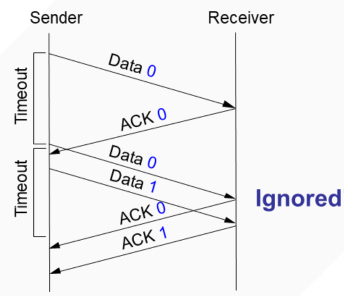


- Loss can occur on ACK channel as well
  - Sender cannot distinguish data loss from ACK loss
  - Sender will retransmit the data frame
- ACK loss—or early timeout—results in duplication
  - The receiver thinks the retransmission is new data

50

SEQUENCE NUMBERS

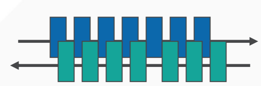




- Sequence numbers solve this problem
  - Receiver can simply ignore duplicate data
  - But must still send an ACK! (Why?)
- Simplest ARQ: Stop-and-wait
  - Only one outstanding frame at a time

51

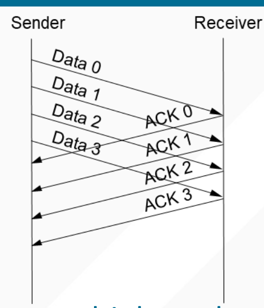
STOP AND WAIT PERFORMANCE

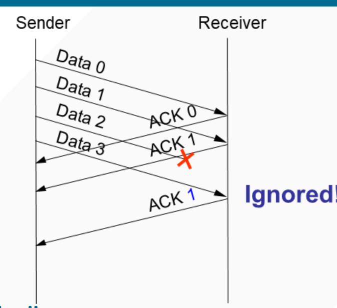


- Lousy performance if xmit 1 pkt << prop. delay
  - How bad?
- Want to utilize all available bandwidth
  - Need to keep more data “in flight”
  - How much? Called the bandwidth-delay product
- Also limited by quality of timeout (how long?)

52

PIPELINED TRANSMISSION

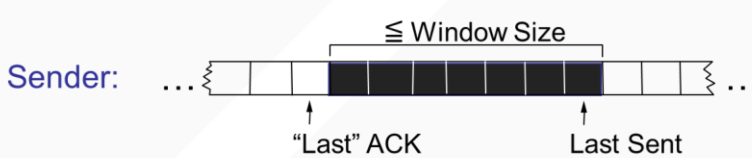




- Keep multiple packets "in flight"
  - Allows sender to make efficient use of the link
  - Sequence numbers ensure receiver can distinguish frames
- Sender buffers outstanding un-acked packets
  - Receiver ACKs the highest *consecutive* frame received
    - ACKs are cumulative (covers current frame and all previous)

53

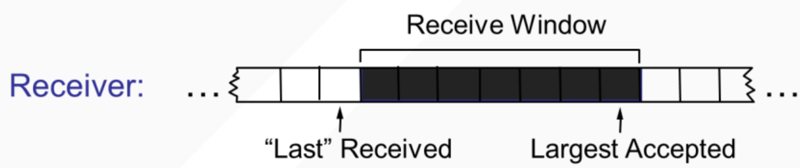
SLIDING WINDOW: SENDER



- Window bounds outstanding unACKed data
  - Implies need for buffering at sender
- "Last" ACK applies to in-order data
- What to do on a timeout?
  - Go-Back-N: resend all unacknowledged data on timeout
  - Selective Repeat: timer per packet, resend as needed

54

SLIDING WINDOW: RECEIVER



- Receiver buffers too:
  - data may arrive out-of-order
  - or faster than can be consumed
    - Flow control: tell sender how much buffer left at receiver
- Receiver ACK choices:
  - Cumulative, Selective (exempt missing frames), Negative

55

DECIDING WHEN TO RETRANSMIT

- How do you know when a packet has been lost?
  - Ultimately sender uses timers to decide when to retransmit
- But how long should the timer be?
  - Too long: inefficient (large delays, poor use of bandwidth)
  - Too short: may retransmit unnecessarily (causing extra traffic)
- Right timer is based on the round-trip time (RTT)
  - Which can vary greatly

56

UC San Diego

57