# 15-744: Computer Networking

L-1 Intro to Computer Networks

## Outline

- Administrivia
- Layering
- Design principles in internetworks
- IP design



## Who's Who?

- Professor: Srinivasan Seshan
  - http://www.cs.cmu.edu/~srini
  - srini@cmu.edu
  - Office hours: Friday 4:00-5:00
- TA: Vyas Sekar
  - vyass@cs.cmu.edu
  - Office hours: Tuesday 4:00-5:00
- Course info
  - http://www.cs.cmu.edu/~srini/15-744/F04/



## Objectives

- Understand the state-of-the-art in network protocols, architectures and applications
- Understand how networking research is done
  - Teach the typical constraints and thought processes used in networking research
- How is class different from undergraduate networking (15-441)
  - Training network programmers vs. training network researchers

## Web Page

- Check regularly!!
- Course schedule
- Reading list
- Lecture notes
- Announcements
- Assignments
- Project ideas
- Exams
- Student list



## Course Materials

- Research papers
  - Links to ps or pdf on Web page
  - Combination of classic and recent work
  - ~50 papers
  - Optional readings
- Recommended textbooks
  - For students not familiar with networking
  - Peterson & Davie or Kurose & Ross



## Grading

- Homework assignments
  - Problem sets & hands-on assignments (15%)
  - Hand-ins for readings (10%)
- Class participation (5%)
- 2 person project (30%)
- Midterm exam (20%)
- Final exam (20%)



## Waitlist & HW 0

- Class is heavily over-subscribed
  - Unlikely to take any more students
  - Position on waitlist irrelevant
- HW 0 – due next class
- If you are trying to add class
  - HW 0 due early – Wednesday to TA
  - I will email enrollment decisions as early as possible

## Class Coverage

- Little coverage of physical and data link layer
  - Students expected to know this
- Focus on network to application layer
- We will deal with:
  - Protocol rules and algorithms
  - Investigate protocol trade-offs
  - Why this way and not another?



## Lecture Topics

**Traditional**

- Layering
- Internet architecture
- Routing (IP)
- Transport (TCP)
- Queue management (FQ, RED)
- Naming (DNS)

**Recent Topics**

- Multicast
- Mobility/wireless
- Active networks
- QOS
- Security
- Network measurement
- Overlay networks
- P2P applications



## Outline

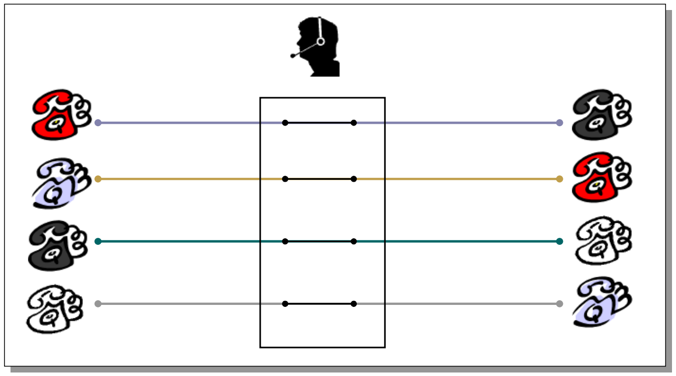
- Administrivia
- Layering
- Design principles in internetworks
- IP design



## What is the Objective of Networking?

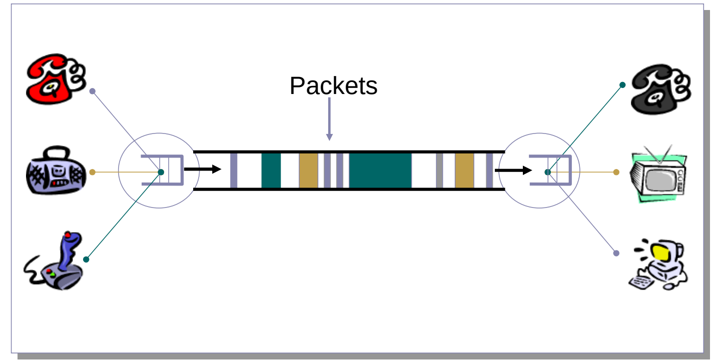
- Communication between applications on different computers
- Must understand application needs/demands
  - Traffic data rate
  - Traffic pattern (bursty or constant bit rate)
  - Traffic target (multipoint or single destination, mobile or fixed)
  - Delay sensitivity
  - Loss sensitivity

## Back in the Old Days…





## Packet Switching (Internet)






## Packet Switching

- Interleave packets from different sources
- Efficient: resources used on demand
  - Statistical multiplexing
- General
  - Multiple types of applications
- Accommodates bursty traffic
  - Addition of queues

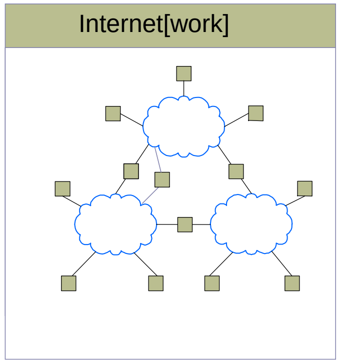


## Characteristics of Packet Switching

- Store and forward
  - Packets are self contained units
  - Can use alternate paths – reordering
- Contention
  - Congestion
  - Delay

## Internet[work]

- A collection of interconnected networks
- Host: network endpoints (computer, PDA, light switch, …)
- Router: node that connects networks
- Internet vs. internet



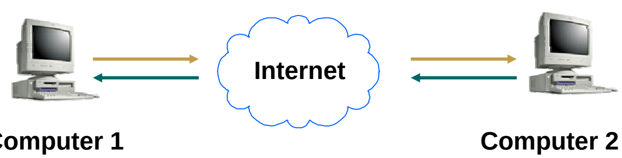



## Challenge

- Many differences between networks
  - Address formats
  - Performance – bandwidth/latency
  - Packet size
  - Loss rate/pattern/handling
  - Routing
- How to translate between various network technologies?



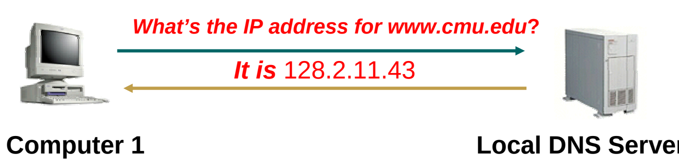
## How To Find Nodes?




Need naming and routing

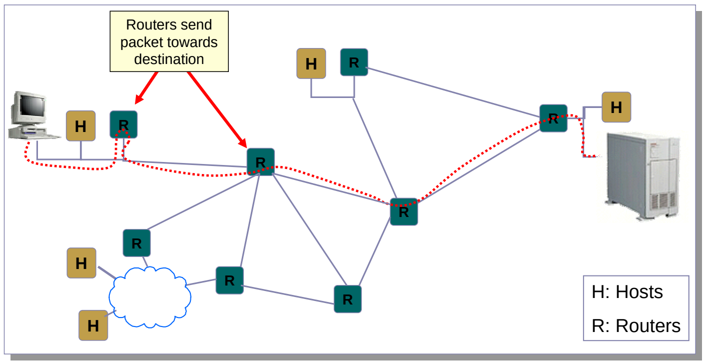


## Naming




Translates human readable names to logical endpoints

## Routing





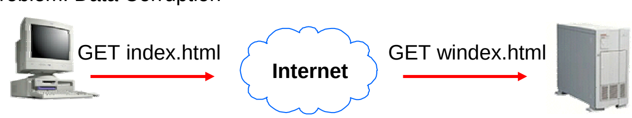

## Meeting Application Demands

- Reliability
  - Corruption
  - Lost packets
- Flow and congestion control
- Fragmentation
- In-order delivery
- Etc…

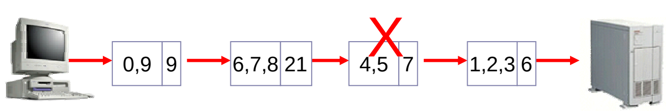


## What if the Data gets Corrupted?

Problem: Data Corruption



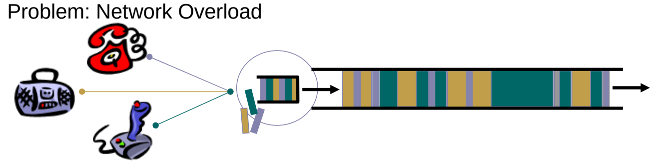

Solution: Add a *checksum*






## What if Network is Overloaded?

Problem: Network Overload



Solution: Buffering and Congestion Control

- Short bursts: buffer
- What if buffer overflows?
  - Packets dropped
  - Sender adjusts rate until load = resources → "congestion control"

## What if the Data gets Lost?

Problem: Lost Data


Solution: Timeout and Retransmit

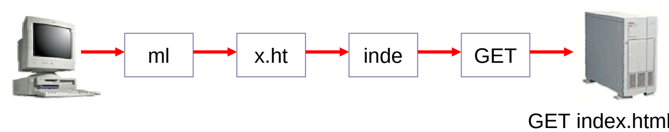



## What if the Data Doesn't Fit?

Problem: Packet size

- On Ethernet, max IP packet is 1.5kbytes
- Typical web page is 10kbytes

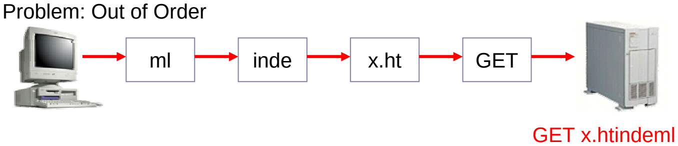
Solution: Fragment data across packets



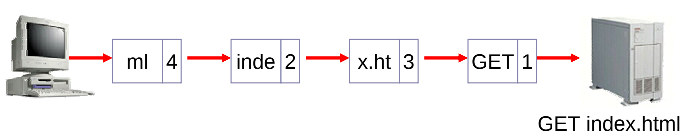



## What if the Data is Out of Order?

Problem: Out of Order




Solution: Add Sequence Numbers



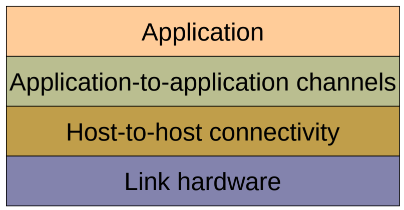



## Lots of Functions Needed

- Link
- Multiplexing
- Routing
- Addressing/naming (locating peers)
- Reliability
- Flow control
- Fragmentation
- Etc….

## What is Layering?

- Modular approach to network functionality
- Example:






## Protocols

- Module in layered structure
- Set of rules governing communication between network elements (applications, hosts, routers)
- Protocols define:
  - Interface to higher layers (API)
  - Interface to peer
    - Format and order of messages
    - Actions taken on receipt of a message

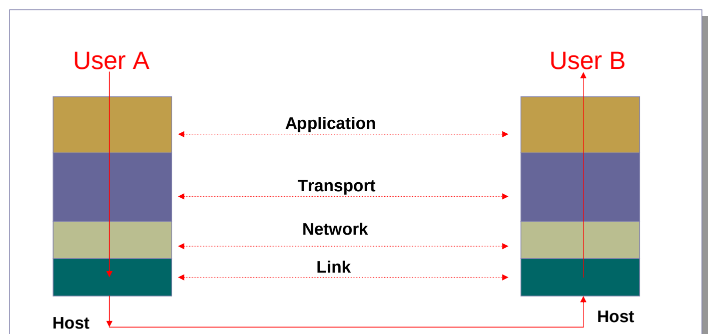


## Layering Characteristics

- Each layer relies on services from layer below and exports services to layer above
- Interface defines interaction
- Hides implementation - layers can change without disturbing other layers (black box)



## Layering



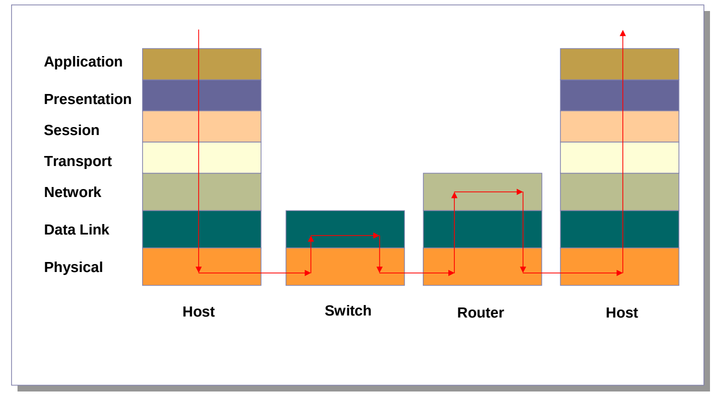

Layering: technique to simplify complex systems

## E.g.: OSI Model: 7 Protocol Layers

- Physical: how to transmit bits
- Data link: how to transmit frames
- Network: how to route packets
- Transport: how to send packets end2end
- Session: how to tie flows together
- Presentation: byte ordering, security
- Application: everything else

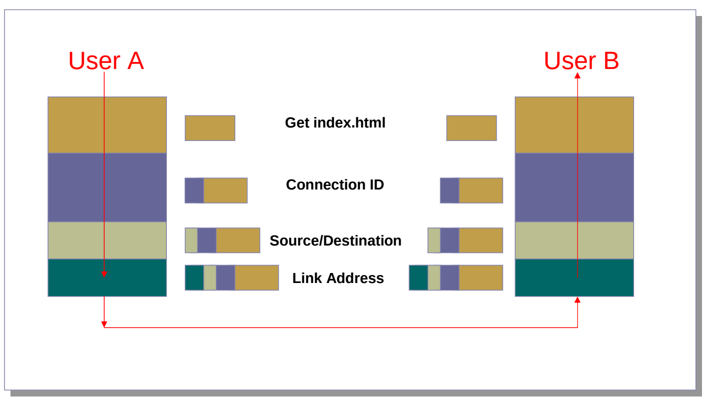


## OSI Layers and Locations





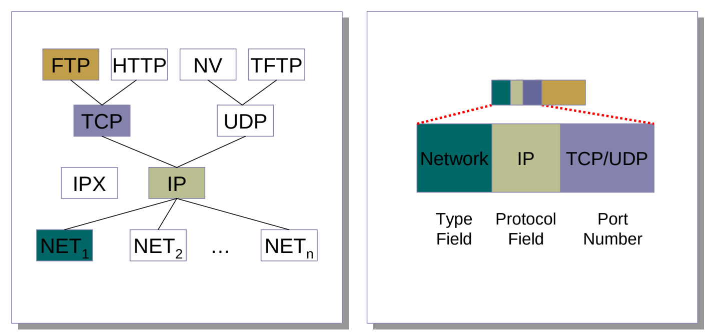

## Layer Encapsulation






## Protocol Demultiplexing

- Multiple choices at each layer



## Is Layering Harmful?

- Sometimes..
  - Layer N may duplicate lower level functionality (e.g., error recovery)
  - Layers may need same info (timestamp, MTU)
  - Strict adherence to layering may hurt performance



## Design Considerations

- How to determine split of functionality
  - Across protocol layers
  - Across network nodes
- Assigned Reading
  - [SRC84] End-to-end Arguments in System Design
  - [Cla88] Design Philosophy of the DARPA Internet Protocols
  - [Cla02] Tussle in Cyberspace: Defining Tomorrow's Internet



## Outline

- Administrivia
- Layering
- Design principles in internetworks
- IP design



## Goals [Clark88]

0 **Connect existing networks**

initially ARPANET and ARPA packet radio network

1. Survivability

   ensure communication service even in the presence of network and router failures

2. Support multiple types of services
3. Must accommodate a variety of networks
4. Allow distributed management
5. Allow host attachment with a low level of effort
6. Be cost effective
7. Allow resource accountability

## Challenge

- Many differences between networks
  - Address formats
  - Performance – bandwidth/latency
  - Packet size
  - Loss rate/pattern/handling
  - Routing
- How to internetwork various network technologies



## Challenge 1: Address Formats

- Map one address format to another?
  - Bad idea → many translations needed
- Provide one common format
  - Map lower level addresses to common format



## Challenge 2: Different Packet Sizes

- Define a maximum packet size over all networks?
  - Either inefficient or high threshold to support
- Implement fragmentation/re-assembly
  - Who is doing fragmentation?
  - Who is doing re-assembly?



## Gateway Alternatives

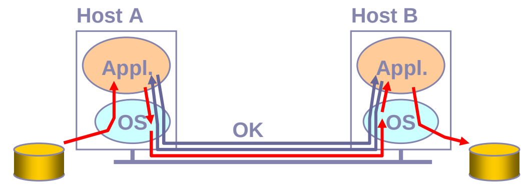
- Translation
  - Difficulty in dealing with different features supported by networks
  - Scales poorly with number of network types (N^2 conversions)
- Standardization
  - “IP over everything” (Design Principle 1)
  - Minimal assumptions about network
  - Hourglass design

## End-to-End Argument (Principle 2)

- Deals with where to place functionality
  - Inside the network (in switching elements)
  - At the edges
- Argument
  - There are functions that can only be correctly implemented by the endpoints – do not try to completely implement these elsewhere
  - Guideline not a law



## Example: Reliable File Transfer




- Solution 1: make each step reliable, and then concatenate them
- Solution 2: end-to-end check and retry



## E2E Example: File Transfer

- Even if network guaranteed reliable delivery
  - Need to provide end-to-end checks
  - E.g., network card may malfunction
  - The receiver has to do the check anyway!
- Full functionality can only be entirely implemented at application layer; no need for reliability from lower layers
- Is there any need to implement reliability at lower layers?



## Discussion

- Yes, but only to improve performance
- If network is highly unreliable
  - Adding some level of reliability helps performance, not correctness
  - Don't try to achieve perfect reliability!
  - Implementing a functionality at a lower level should have minimum performance impact on the applications that do not use the functionality

## Examples

- What should be done at the end points, and what by the network?
  - Reliable/sequenced delivery?
  - Addressing/routing?
  - Security?
  - What about Ethernet collision detection?
  - Multicast?
  - Real-time guarantees?



## Internet & End-to-End Argument

- Network layer provides one simple service: best effort datagram (packet) delivery
- Only one higher level service implemented at transport layer: reliable data delivery (TCP)
  - Performance enhancement; used by a large variety of applications (Telnet, FTP, HTTP)
  - Does not impact other applications (can use UDP)
  - Original TCP & IP were integrated – Reed successfully argued for separation
- Everything else implemented at application level
- Does FTP look like E2E file transfer?
  - TCP provides reliability between kernels not disks



## Principle 3

- Best effort delivery
- All packets are treated the same
- Relatively simple core network elements
- Building block from which other services (such as reliable data stream) can be built
- Contributes to scalability of network



## Principle 4

- Fate sharing
- Critical state only at endpoints
- Only endpoint failure disrupts communication
- Helps survivability

## Principle 5

- Soft-state
  - Announce state
  - Refresh state
  - Timeout state
- Penalty for timeout – poor performance
- Robust way to identify communication flows
  - Possible mechanism to provide non-best effort service
- Helps survivability



## Principle 6

- Decentralization
- Each network owned and managed separately
- Will see this in BGP routing especially



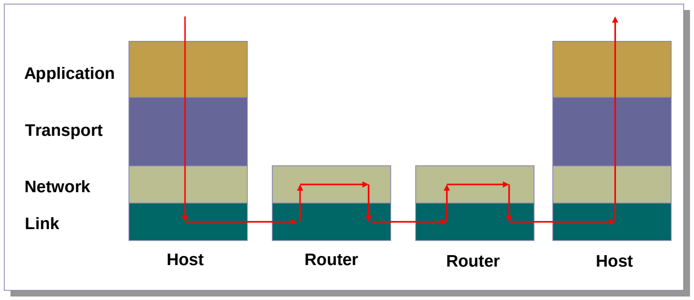
## Principle 7

- Be conservative in what you send and liberal in what you accept
  - Unwritten rule
- Especially useful since many protocol specifications are ambiguous
- E.g. TCP will accept and ignore bogus acknowledgements



## IP Layering (Principle 8)

- Relatively simple
- Sometimes taken too far



## IP Design Weaknesses

- Greedy sources aren't handled well
- Weak accounting and pricing tools
- Weak administration and management tools
- Incremental deployment difficult at times
  - Result of no centralized control
  - No more "flag" days
  - Are active networks the solution?



## Changes Over Time

- Developed in simpler times
  - Common goals, consistent vision
- With success came multiple goals – examples:
  - ISPs must talk to provide connectivity but are fierce competitors
  - Privacy of users vs. government's need to monitor
  - User's desire to exchange files vs. copyright owners
- Must deal with the tussle between concerns in design



## New Principles?

- Design for variation in outcome
  - Allow design to be flexible to different uses/results
- Isolate tussles
  - QoS designs uses separate ToS bits instead of overloading other parts of packet like port number
  - Separate QoS decisions from application/protocol design
- Provide choice → allow all parties to make choices on interactions
  - Creates competition
    - Fear between providers helps shape the tussle



## Outline

- Administrivia
- Layering
- Design principles in internetworks
- IP design

## How is IP Design Standardized?

- IETF
  - Voluntary organization
  - Meeting every 4 months
  - Working groups and email discussions
- “We reject kings, presidents, and voting; we believe in rough consensus and running code” (Dave Clark 1992)
  - Need 2 independent, interoperable implementations for standard
- IRTF
  - End2End
  - Reliable Multicast, etc..



## IPv4 Header – RFC791 (1981)

| 0 | 4 | 8 | 16 | 19 | 24 – 32 |
|---|---|---|---|---|---|
| Version | IHL | Type of Service | Total Length | | |
| Identification | | | Flags | Fragment Offset | |
| Time to Live | | Protocol | Header Checksum | | |
| Source Address | | | | | |
| Destination Address | | | | | |
| Options | | | | | Padding |



## IP Type of Service

- Typically ignored
- Values
  - 3 bits of precedence
  - 1 bit of delay requirements
  - 1 bit of throughput requirements
  - 1 bit of reliability requirements
- Replaced by DiffServ



## Fragmentation

- IP packets can be 64KB
- Different link-layers have different MTUs
- Split IP packet into multiple fragments
  - IP header on each fragment
  - Various fields in header to help process
  - Intermediate router may fragment as needed
- Where to do reassembly?
  - End nodes – avoids unnecessary work
  - Dangerous to do at intermediate nodes
    - Buffer space
    - Multiple paths through network

## Fragmentation Related Fields

- Length
  - Length of IP fragment
- Identification
  - To match up with other fragments
- Flags
  - Don't fragment flag
  - More fragments flag
- Fragment offset
  - Where this fragment lies in entire IP datagram
  - Measured in 8 octet units (11 bit field)



## Fragmentation is Harmful

- Uses resources poorly
  - Forwarding costs per packet
  - Best if we can send large chunks of data
  - Worst case: packet just bigger than MTU
- Poor end-to-end performance
  - Loss of a fragment
- Reassembly is hard
  - Buffering constraints



## Path MTU Discovery

- Hosts dynamically discover minimum MTU of path
- Algorithm:
  - Initialize MTU to MTU for first hop
  - Send datagrams with Don't Fragment bit set
  - If ICMP "pkt too big" msg, decrease MTU
- What happens if path changes?
  - Periodically (>5mins, or >1min after previous increase), increase MTU
- Some routers will return proper MTU
- MTU values cached in routing table



## Other Fields

- Header length (in 32 bit words)
- Time to live
  - Ensure packets exit the network
- Protocol
  - Demultiplexing to higher layer protocols
- Header checksum
  - Ensures some degree of header integrity
  - Relatively weak – 16 bit
- Options
  - E.g. Source routing, record route, etc.
  - Performance issues
    - Poorly supported

## Addressing in IP

- IP addresses are names of interfaces
- Domain Name System (DNS) names are names of hosts
- DNS binds host names to interfaces
- Routing binds interface names to paths



## Addressing Considerations

- Fixed length or variable length?
- Issues:
  - Flexibility
  - Processing costs
  - Header size
- Engineering choice: IP uses fixed length addresses



## Addressing Considerations

- Structured vs flat
- Issues
  - What information would routers need to route to Ethernet addresses?
    - Need structure for designing scalable binding from interface name to route!
  - How many levels? Fixed? Variable?

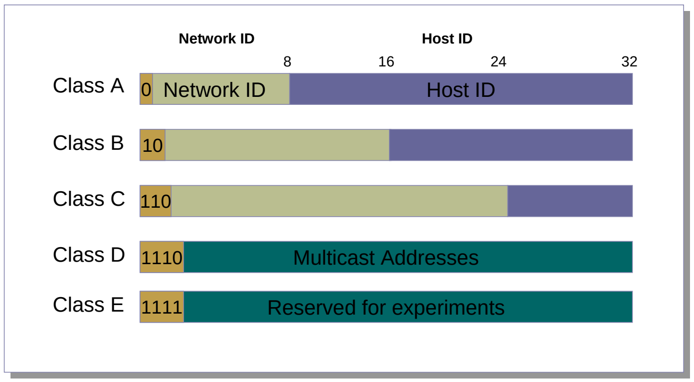


## IP Addresses

- Fixed length: 32 bits
- Initial classful structure (1981)
- Total IP address size: 4 billion
  - Class A: 128 networks, 16M hosts
  - Class B: 16K networks, 64K hosts
  - Class C: 2M networks, 256 hosts

| High Order Bits | Format | Class |
|---|---|---|
| 0 | 7 bits of net, 24 bits of host | A |
| 10 | 14 bits of net, 16 bits of host | B |
| 110 | 21 bits of net, 8 bits of host | C |

## IP Address Classes (Some are Obsolete)






## Some Special IP Addresses

- 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



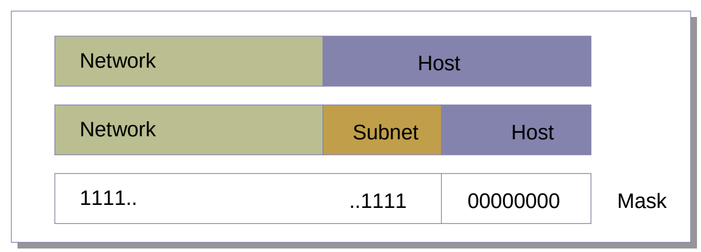
## Subnet Addressing – RFC917 (1984)

- For class A & B networks
- Very few LANs have close to 64K hosts
  - For electrical/LAN limitations, performance or administrative reasons
- Need simple way to get multiple “networks”
  - Use bridging, multiple IP networks or split up single network address ranges (subnet)
  - Must reduce the total number of network addresses that are assigned
- CMU case study in RFC
  - Chose not to adopt – concern that it would not be widely supported ☺



## Subnetting

- Variable length subnet masks
  - Could subnet a class B into several chunks



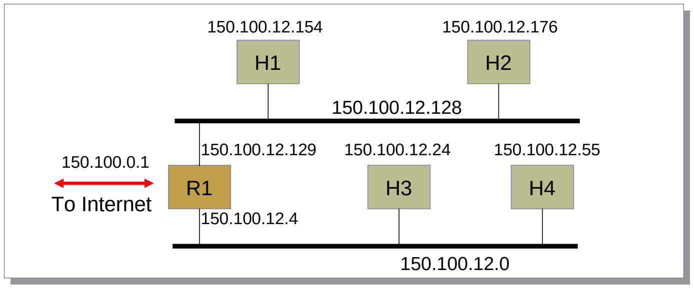
## Subnetting Example

- Assume an organization was assigned address 150.100
- Assume < 100 hosts per subnet
- How many host bits do we need?
  - Seven
- What is the network mask?
  - 11111111 11111111 11111111 10000000
  - 255.255.255.128



## Subnet Addressing Example

- Assume a packet arrives with address 150.100.12.176
- Step 1: AND address with subnet mask






## IPv4 Problems

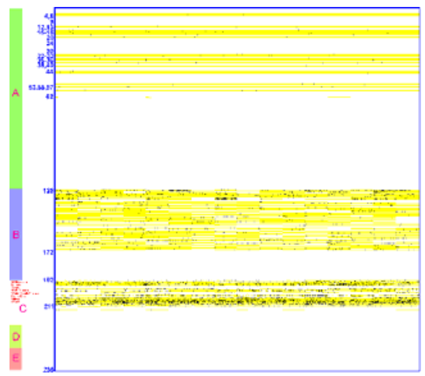
- Addressing
- Routing



## IP Address Problem (1991)

- Address space depletion
  - In danger of running out of classes A and B
- Why?
  - Class C too small for most domains
  - Very few class A – IANA (Internet Assigned Numbers Authority) very careful about giving
  - Class B – greatest problem
    - Sparsely populated – but people refuse to give it back

## IP Address Utilization ('98)



http://www.caida.org/outreach/resources/learn/ipv4space/



## IPv4 Routing Problems

- Core router forwarding tables were growing large
  - Class A: 128 networks, 16M hosts
  - Class B: 16K networks, 64K hosts
  - Class C: 2M networks, 256 hosts
- 32 bits does not give enough space encode network location information inside address – i.e., create a structured hierarchy



## Solution 1 – CIDR

- Assign multiple class C addresses
- Assign consecutive blocks
- RFC1338 – Classless Inter-Domain Routing (CIDR)



## Classless Inter-Domain Routing

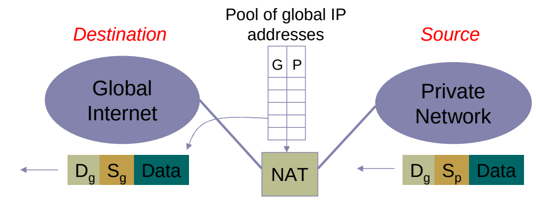
- Do not use classes to determine network ID
- Assign any range of addresses to network
  - Use common part of address as network number
  - e.g., addresses 192.4.16 - 196.4.31 have the first 20 bits in common. Thus, we use this as the network number
  - netmask is /20, /xx is valid for almost any xx
- Enables more efficient usage of address space (and router tables)

## Solution 2 - NAT

- Network Address Translation (NAT)
- Alternate solution to address space
  - Kludge (but useful)
- Sits between your network and the Internet
- Translates local network layer addresses to global IP addresses
- Has a pool of global IP addresses (less than number of hosts on your network)



## NAT Illustration




- **Operation: Source (S) wants to talk to Destination (D):**
  - Create $S_g$-$S_p$ mapping
  - Replace $S_p$ with $S_g$ for outgoing packets
  - Replace $S_g$ with $S_p$ for incoming packets
- D & S can be just IP addresses or IP addresses + port #'s

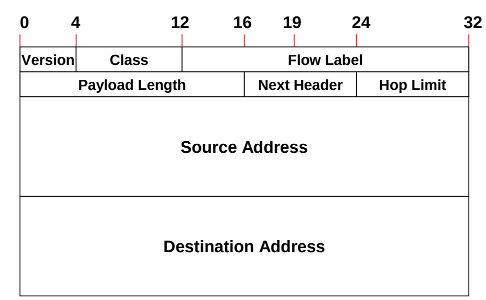


## Solution 3 - IPv6

- Scale – addresses are 128bit
  - Header size?
- Simplification
  - Removes infrequently used parts of header
  - 40byte fixed size vs. 20+ byte variable
- IPv6 removes checksum
  - Relies on upper layer protocols to provide integrity
- IPv6 eliminates fragmentation
  - Requires path MTU discovery
  - Requires 1280 byte MTU



## IPv6 Header



## IPv6 Changes

- TOS replaced with traffic class octet
- Flow
  - Help soft state systems
  - Maps well onto TCP connection or stream of UDP packets on host-port pair
- Easy configuration
  - Provides auto-configuration using hardware MAC address to provide unique base
- Additional requirements
  - Support for security
  - Support for mobility



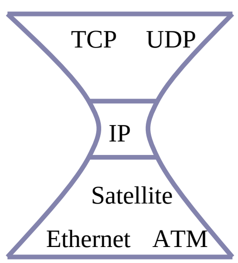
## IPv6 Changes

- Protocol field replaced by next header field
  - Support for protocol demultiplexing as well as option processing
- Option processing
  - Options are added using next header field
  - Options header does not need to be processed by every router
    - Large performance improvement
    - Makes options practical/useful



## Summary: Internet Architecture

- Packet-switched datagram network
- IP is the “compatibility layer”
  - Hourglass architecture
  - All hosts and routers run IP
- Stateless architecture
  - no per flow state inside network






## Summary: Minimalist Approach

- Dumb network
  - IP provide minimal functionalities to support connectivity
    - Addressing, forwarding, routing
- Smart end system
  - Transport layer or application performs more sophisticated functionalities
    - Flow control, error control, congestion control
- Advantages
  - Accommodate heterogeneous technologies (Ethernet, modem, satellite, wireless)
  - Support diverse applications (telnet, ftp, Web, X windows)
  - Decentralized network administration

## Summary: IP Design

- Relatively simple design
  - Some parts not so useful (TOS, options)
- Beginning to show age
  - Unclear what the solution will be → probably IPv6



## Next Lecture

- Forwarding
- IP lookup
- High-speed router architecture
- Intro to routing protocols
- Readings
  - [D+97] Small Forwarding Tables for Fast Routing Lookups
  - [BV01] Scalable Packet Classification
  - [McK97] A Fast Switched Backplane for a Gigabit Switched Router
  - [KCY03] Scaling Internet Routers Using Optics
  - Know RIP/OSPF