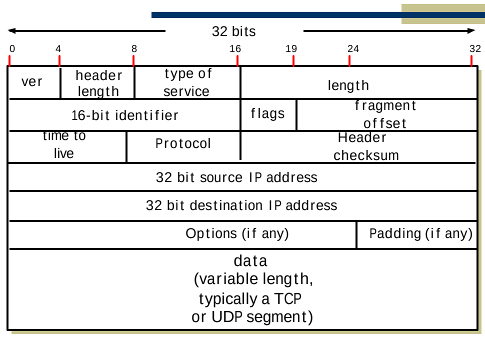
15-441 Computer Networking

Lecture 8 – IP Packets, Routers

Outline

- IP Packet Format
- NAT
- IPv6
- Router Internals
- Route Lookup

IPv4 Header – RFC791 (1981)



IP Header Fields

- Version → 4 for IPv4
- Header length (in 32 bit words)
  - Minimum value is 5 (header without any options)
- Length of entire IP packet in octets (including header)
- Identifier, flags, fragment offset → used primarily for fragmentation
- Time to live
  - Must be decremented at each router
  - Packets with TTL=0 are thrown away
  - Ensure packets exit the network

IP Header Fields

- Protocol
  - Demultiplexing to higher layer protocols
  - TCP = 6, ICMP = 1, UDP = 17…
- Header checksum
  - Ensures some degree of header integrity
  - Relatively weak – 16 bit
- Source/Dest address
- Options
  - E.g. Source routing, record route, etc.
  - Performance issues
    - Poorly supported

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

ICMP: Internet Control Message Protocol

- Used by hosts, routers, gateways to communication network-level information
  - Error reporting: unreachable host, network, port, protocol
  - Echo request/reply (used by ping)
- Network-layer "above" IP:
  - ICMP msgs carried in IP datagrams
- ICMP message: type, code plus first 8 bytes of IP datagram causing error

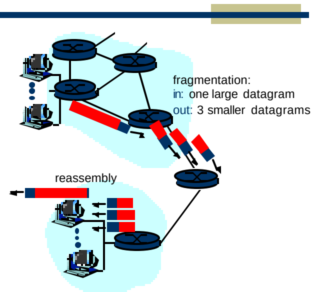
Type	Code	description
0	0	echo reply (ping)
3	0	dest. network unreachable
3	1	dest host unreachable
3	2	dest protocol unreachable
3	3	dest port unreachable
3	6	dest network unknown
3	7	dest host unknown
4	0	source quench (congestion control - not used)
8	0	echo request (ping)
9	0	route advertisement
10	0	router discovery
11	0	TTL expired
12	0	bad IP header

Fragmentation

- IP packets can be up to 64KB
- Different link-layers have different MTUs
- Split IP packet into multiple fragments
  - IP header on each fragment
  - Intermediate router may fragment as needed

IP Fragmentation & Reassembly

- Network links have MTU (max.transfer size) - largest possible link-level frame.
  - different link types, different MTUs
- Large IP datagram divided ("fragmented") within net
  - one datagram becomes several datagrams
  - IP header bits used to identify, order related fragments



Reassembly

- Where to do reassembly?
- End nodes
  - Avoids unnecessary work where large packets are fragmented multiple times
- Dangerous to do at intermediate nodes
  - How much buffer space required at routers?
  - What if routes in network change?
    - Multiple paths through network
    - All fragments only required to go through destination

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 (13 bit field)

IP Fragmentation and Reassembly

length =4000	ID =x	fragflag =0	offset =0

One large datagram becomes several smaller datagrams

length =1500	ID =x	fragflag =1	offset =0

length =1500	ID =x	fragflag =1	offset =1480

length =1040	ID =x	fragflag =0	offset =2960

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

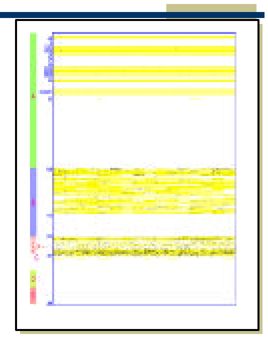
- Hosts dynamically discover minimum MTU of path
- Algorithm:
  - Initialize MTU to MTU for first hop
  - Send datagrams with Don't Fragment bit set
  - If ICMP "pkt too big" msg, decrease MTU
- What happens if path changes?
  - Periodically (>5mins, or >1min after previous increase), increase MTU
-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

Outline

- IP Packet Format
- NAT
- IPv6
- Router Internals
- Route Lookup

IP Address Utilization ('98)

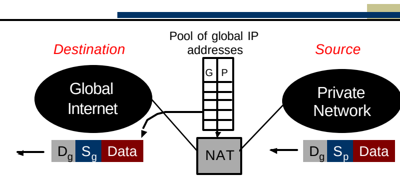
- Address space depletion
  - In danger of running out of classes A and B
- 32-bit address space completely allocated by 2008
- Two solutions
  - NAT
  - IPv6



Network Address Translation (NAT)

- Possible solution to address space exhaustion
  - 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)
- Uses special unallocated addresses (RFC 1597) locally
  - 10.0.0.0/8, 172.16.0.0/12, 192.168.0.0/16

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
- How many hosts can have active transfers at one time?

Problems with NAT

- What if we only have few (or just one) IP address?
  - Use Network Address & Port Translator (NAPT)
- NAPT translates:
  - Translates $addr_{private}$ + flow info to $addr_{global}$ + new flow info
    - Uses TCP/UDP port numbers
  - Potentially thousands of simultaneous connections with one global IP address

Problems with NAT

- Hides the internal network structure
  - Some consider this an advantage
- Some protocols carry addresses
  - E.g., FTP carries addresses in text
  - What is the problem?
- Must update transport protocol headers (port number & checksum)
- Encryption
- No inbound connections

Outline

- IP Packet Format
- NAT
- IPv6
- Router Internals
- Route Lookup

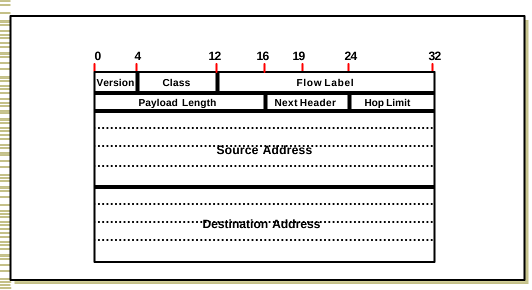
IPv6

- Primary objective bigger addresses
  - Addresses are 128bit → What about header size!!!
- Simplification
  - Header format helps speed processing/forwarding
  - Header changes to facilitate QoS
  - Removes infrequently used parts of header
  - 40byte fixed size vs. 20+ byte variable

IPv6 Changes

- 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 label
  - Identify datagrams in same “flow.” (concept of“flow” not well defined)
  - 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

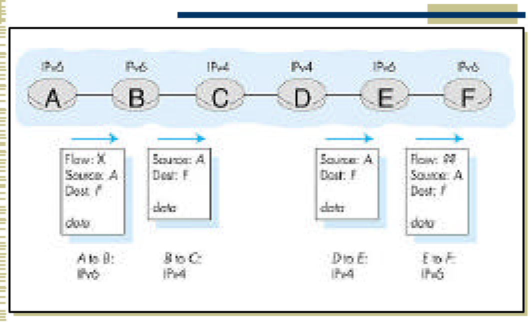
IPv6 Changes

- 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
- Additional requirements
  - Support for security
  - Support for mobility

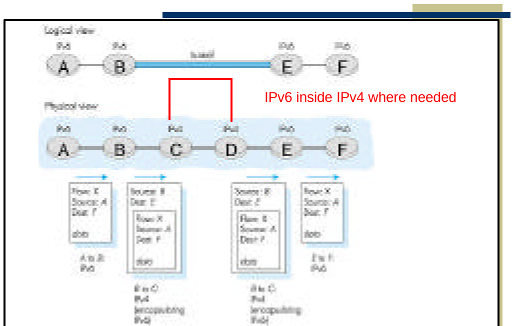
Transition From IPv4 To IPv6

- Not all routers can be upgraded simultaneous
  - No “flag days”
  - How will the network operate with mixed IPv4 and IPv6 routers?
- Two proposed approaches:
  - *Dual Stack*: some routers with dual stack (v6, v4) can “translate” between formats
  - *Tunneling:* IPv6 carried as payload n IPv4 datagram among IPv4 routers

Dual Stack Approach



Tunneling



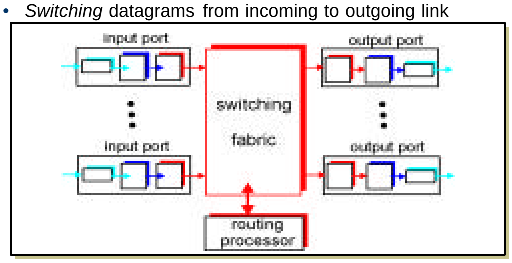
Outline

- IP Packet Format
- NAT
- IPv6
- Router Internals
- Route Lookup

Router Architecture Overview

Two key router functions:

- Run routing algorithms/protocol (RIP, OSPF, BGP)
- *Switching* datagrams from incoming to outgoing link



What Does a Router Look Like?

- Line cards
  - Network interface cards
- Forwarding engine
  - Fast path routing (hardware vs. software)
  - Usually on line card
- Backplane
  - Switch or bus interconnect
- Processor
  - Handles routing protocols, error conditions

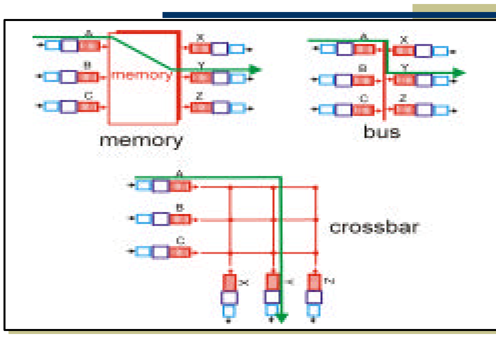
Router Processing

- Packet arrives arrives at inbound line card
- Header processed by forwarding engine
  - Forwarding engine determines output line card/destination
  - Checksum updated but not checked
- Packet copied to outbound line card
  - Odd situations sent to network processor

Network Processor

- Runs routing protocol and downloads forwarding table to forwarding engines
- Performs “slow” path processing
  - ICMP error messages
  - IP option processing
  - Fragmentation
  - Packets destined to router

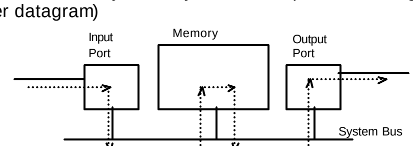
Three Types of Switching Fabrics



Switching Via Memory

First generation routers:

- Packet copied by system’s (single) CPU
- Speed limited by memory bandwidth (2 bus crossings per datagram)

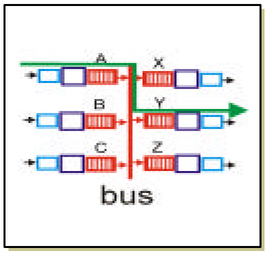


Modern routers:

- Input port processor performs lookup, copy into memory
- Cisco Catalyst 8500

Switching Via Bus

- Datagram from input port memory to output port memory via a shared bus
- Bus contention: switching speed limited by bus bandwidth
- 1 Gbps bus, Cisco 1900: sufficient speed for access and enterprise routers (not regional or backbone)



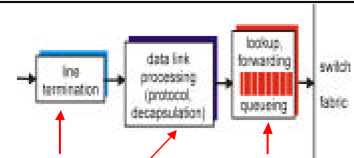
Switching Via An Interconnection Network

- Overcome bus bandwidth limitations
- Crossbar provides full NxN interconnect
  - Expensive
- Banyan networks, other interconnection nets initially developed to connect processors in multiprocessor
  - Typically less capable than complete crossbar
- Cisco 12000: switches Gbps through the interconnection network

Switch Design Issues

- Suppose we have N inputs and M outputs
  - Multiple packets for same output – output contention
  - Switch contention – switching fabric cannot support arbitrary set of transfers
    - I.e, not a full crossbar
- Solution – buffer packets when/where needed
- What happens when these buffers fill up?
  - Packets are THROWN AWAY!! This is where packet loss comes from

Input Port Functions



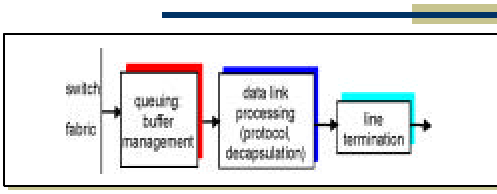
Physical layer: bit-level reception

Data link layer: e.g., Ethernet

Decentralized switching:

- Given datagram dest., lookup output port using routing table in input port memory
- Goal: complete input port processing at 'line speed'
- Needed if datagrams arrive faster than forwarding rate into switch fabric

Output Ports



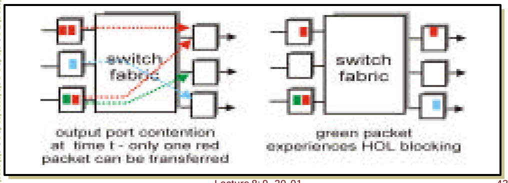
- Queuing required when datagrams arrive from fabric faster than the line transmission rate

Switch Buffering

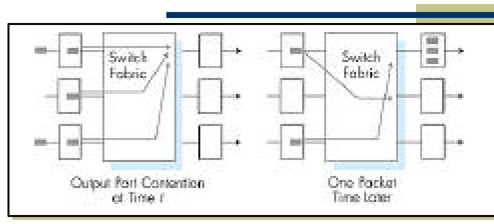
- 3 types of switch buffering
  - Input buffering
    - Fabric slower than input ports combined → queuing may occur at input queues
      - Can avoid any input queuing by making switch speed = N x link speed
  - Output buffering
    - Buffering when arrival rate via switch exceeds output line speed
  - Internal buffering
    - Can have buffering inside switch fabric to deal with limitations of fabric

Input Port Queuing

- Which inputs are processed each slot – schedule?
- Head-of-the-Line (HOL) blocking: datagram at front of queue prevents others in queue from moving forward



Output Port Queuing



- Scheduling discipline chooses among queued datagrams for transmission
  - Can be simple (e.g., first-come first-serve) or more clever (e.g., weighted round robin)

Virtual Output Queuing

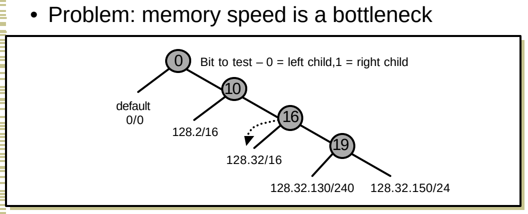
- Maintain per output buffer at input
- Solves head of line blocking problem
- Each of MxN input buffer places bid for output
- Challenge: map bids to schedule of interconnect transfers

Outline

- IP Packet Format
- NAT
- IPv6
- Router Internals
- Route Lookup

How To Do Variable Prefix Match

- Traditional method – Patricia Tree
  - Arrange route entries into a series of bit tests
- Worst case = 32 bit tests
  - Problem: memory speed is a bottleneck



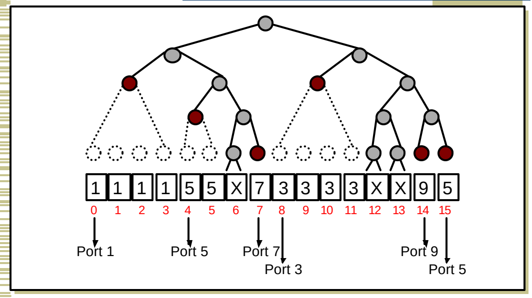
Speeding up Prefix Match - Alternatives

- Content addressable memory (CAM)
  - Hardware based route lookup
  - Input = tag, output = value associated with tag
  - Requires exact match with tag
    - Multiple cycles (1 per prefix searched) with single CAM
    - Multiple CAMs (1 per prefix) searched in parallel
  - Ternary CAM
    - 0,1,don't care values in tag match
    - Priority (I.e. longest prefix) by order of entries in CAM

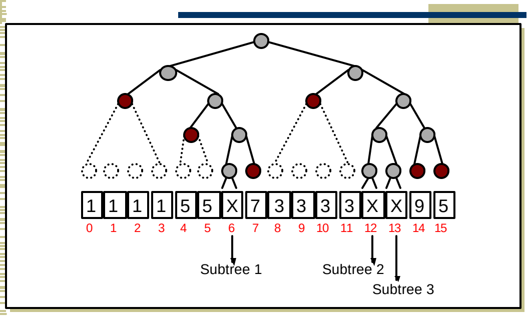
Speeding up Prefix Match

- Cut prefix tree at 16/24/32 bit depth
  - Fill in prefix tree entries by creating extra entries
    - Entries contain output interface for route
  - Add special value to indicate that there are deeper tree entries
    - Only keep 24/32 bit cuts as needed
- Example cut prefix tree at 16 bit depth
  - 64K entries!!
  - Use a variety of clever techniques to compress space taken

Prefix Tree



Prefix Tree



Speeding up Prefix Match

- Scaling issues
  - How would it handle IPv6
- Other possibilities
  - Why were the cuts done at 16/24/32 bits?

Speeding up Prefix Match - Alternatives

- Route caches
  - Packet trains → group of packets belonging to same flow
  - Temporal locality
  - Many packets to same destination
- Other algorithms
  - Bremler-Barr – Sigcomm 99
    - Clue = prefix length matched at previous hop
    - Why is this useful?