Addresses I

- Internet (IP) Addresses
  - 32-bit addresses unique for each network interface
  - each address divided into a *(netid, hostid)* pair
  - hierarchical nature makes routing simpler
    (but machines can't move around without reassignment)
- Primary Classes
  - Class A - over 64K hosts (7 bits net, 24 bits host)
  - Class B - 256 to 64K hosts (14 bits net, 16 bits host)
  - Class C - under 256 hosts (21 bits net, 8 bits host)
  - Class D - IP multicast addressing
  - Class E - reserved
  - Assigned by central Internet authority
    * Currently this is *Network Solutions, Inc*
    * Mail to `hostmaster@internic.net` for registrations
    * Phone is `(800) 444-4345`

Addresses II

- Dotted Decimal Representation (eg: *132.239.1.1*)
- net *127.0.0.0* is reserved for loopback net, host 1 is localhost
- Network Address
  - all 0's in the host field indicates the entire network
  - used by routing to indicate "this net" or default
- Broadcast Address
  - used to send single item to all hosts
  - used when when little knowledge of local net exists
  - 1's in the host field indicate directed broadcast address
  - 1's in both host and net field indicate local net broadcast

Subnet Addressing

- Extension to original IP addressing scheme
- Allows a site to use a single IP network number for multiple physical networks
- Now part of TCP/IP standard (RFC 950)
- only local gateways are aware of subnetting scheme
- rest of internet sees only Class A, B, or C network addresses
- local hosts divide address into *network* and *local* portions
- bits from the Class A, B, or C host numbers are stolen for network numbers
- *subnet mask* identifies which bits are network, which are host
- tradeoff between number of nets and number of hosts on each net
- accommodates growth by shielding internet from local details
- flexible and locally reconfigurable

Subnet Masks

- Each network has a (statically assigned) 32-bit *subnet mask*
- Mask bits set to 1 indicate corresponding IP address bit is from the network part
- Mask bits set to 0 indicate corresponding IP address bit is from the host part
- Example: `11111111 11111111 11111111 00000000`
  - alternatively written as `0xffffff00` or `255.255.255.0`
  - indicates 24 bits are used for network, 8 for host
  - assuming a Class B assignment, we have $2^8 = 256$ nets of $2^8 = 256$ hosts
- Example 2: `11111111 11111111 11111111 11000000`
  - alternatively written as `0xffffffc0` or `255.255.255.192`
  - indicates 26 bits used for network, 6 for host
  - assuming Class B assignment, $2^{(26-16)} = 1024$ nets of $2^6 = 64$ hosts

Using Subnet Masks

- Example: `132.239.0.0` with netmask `ffffffc0`
  - network : `10000100 11101111 00000000 00000000`
  - netmask : `11111111 11111111 11111111 11000000`
  - network number is first 26 bits, host is last 6 bits
  - 16 of the network bits are set for us, so 10 bits are left
  - we have 1024 (256x4) subnets of 62 (64-2) hosts each
  - Subnets are:

    1. 132.239.0.{0–63}
    2. 132.239.0.{64-127}
    3. 132.239.0.{128-191}
    4. 132.239.0.{192-255}
    5. 132.239.1.{0-63}
    6. ...

    1023. 132.239.255.{128-191}
    1024. 132.239.255.{192-255}
  - ending in (0,64,128,192) are whole nets (e.g. in tables)
  - ending in (63,127,191,255) are broadcasts

Subnet Propagation and Assignment

- Propagation:
  - Already mentioned ICMP *subnet mask request* message
  - Masks may also be recorded statically in the network interface
  - Also provided in modern boot protocols: BOOTP and DHCP
- Assignment:
  - Mask assignment (subnet partitioning) up to local network manager
  - Issues: number of hosts and nets, growth, etc.
  - Usually simple choice of masks is least painful

Subnet Broadcasts

- Standard broadcast scheme: all 1's in host part
- Still the same for subnets when viewed from outside
- Broadcast address must match with local subnet address
- Examples:
  - The *ifconfig* program:

```
cabot (1)> /usr/sbin/ifconfig elx0
elx0: flags=863<UP,BROADCAST,NOTRAILERS,RUNNING,MULTICAST> mtu 1500
inet 132.239.108.196 netmask ffffffc0 broadcast 132.239.108.255

cabot (2)> ping -s 132.239.108.255
PING 132.239.108.255: 56 data bytes
64 bytes from cabot (132.239.108.196): icmp_seq=0. time=10. ms
64 bytes from shepard (132.239.108.210): icmp_seq=0. time=70. ms
64 bytes from ucsdext (132.239.108.211): icmp_seq=0. time=330. ms
64 bytes from risky (132.239.108.212): icmp_seq=0. time=350. ms
64 bytes from cabrillo (132.239.108.197): icmp_seq=0. time=380. ms
64 bytes from clark (132.239.108.198): icmp_seq=0. time=400. ms
64 bytes from armstrong (132.239.108.194): icmp_seq=0. time=430. ms
64 bytes from muir (132.239.108.208): icmp_seq=0. time=460. ms
64 bytes from joliet (132.239.108.206): icmp_seq=0. time=480. ms
64 bytes from gagarin (132.239.108.203): icmp_seq=0. time=540. ms
64 bytes from glenn (132.239.108.204): icmp_seq=0. time=570. ms
64 bytes from columbus (132.239.108.199): icmp_seq=0. time=600. ms
64 bytes from lewis (132.239.108.207): icmp_seq=0. time=630. ms
64 bytes from hudson (132.239.108.205): icmp_seq=0. time=660. ms
64 bytes from ponce (132.239.108.202): icmp_seq=0. time=690. ms
...

----132.239.108.255 PING Statistics----
2 packets transmitted, 21 packets received, 10.50 times amplification
round-trip (ms)  min/avg/max = 10/350/690
```

IP Multicasting

- Support for multicast delivery at the network layer
- See RFC1112
- Class D addresses 224.0.0.0 thru 239.255.255.255
- Hosts join *groups* (IP addresses):
  - join (leave) with host–mrouter IGMP messages
  - group addresses only used as destinations
  - requires modifications to hosts and routers
- Mapping IP to Ethernet multicast addresses
  - ethernet supports *hardware multicast* addressing
  - the low-order 23 bits of the IP multicast address is placed into the low-order 23-bits of the Ethernet multicast address 01-00-5E-00-00-00 (hex)
  - this mapping is not unique, but the IP-layer software can drop non-matching packets
- The *MBONE*
  - an informal collection of multicast capable networks
  - IP "tunneling" through non-multicast-capable networks
  - existing since about 1992
  - used primarily to carry collaboration experiments
  - supported by *mrouted*, a multicast routing agent

Reserved Addresses

- The following addresses are reserved (RFC1700):

```
224.0.0.0  Base Address (Reserved)                   [RFC1112,JBP]
224.0.0.1  All Systems on this Subnet                [RFC1112,JBP]
224.0.0.2  All Routers on this Subnet                        [JBP]
224.0.0.3  Unassigned                                        [JBP]
224.0.0.4  DVMRP    Routers                          [RFC1075,JBP]
224.0.0.5  OSPFIGP  OSPFIGP All Routers             [RFC1583,JXM1]
224.0.0.6  OSPFIGP  OSPFIGP Designated Routers      [RFC1583,JXM1]
224.0.0.9  RIP2 Routers                                    [GSM11]
224.0.0.10 IGRP Routers                          [Dino Farinacci]
224.0.0.11 Mobile-Agents                            [Bill Simpson]
224.0.1.2  SGI-Dogfight                                      [AXC]
224.0.1.3  Rwhod                                             [SXD]
224.0.1.9  MTP Multicast Transport Protocol                  [SXA]
224.0.1.10 IETF-1-LOW-AUDIO                                  [SC3]
224.0.1.11 IETF-1-AUDIO                                      [SC3]
224.0.1.12 IETF-1-VIDEO                                      [SC3]
224.0.1.13 IETF-2-LOW-AUDIO                                  [SC3]
224.0.1.14 IETF-2-AUDIO                                      [SC3]
224.0.1.15 IETF-2-VIDEO                                      [SC3]
224.0.1.20 any private experiment                            [JBP]
```

Direct Delivery

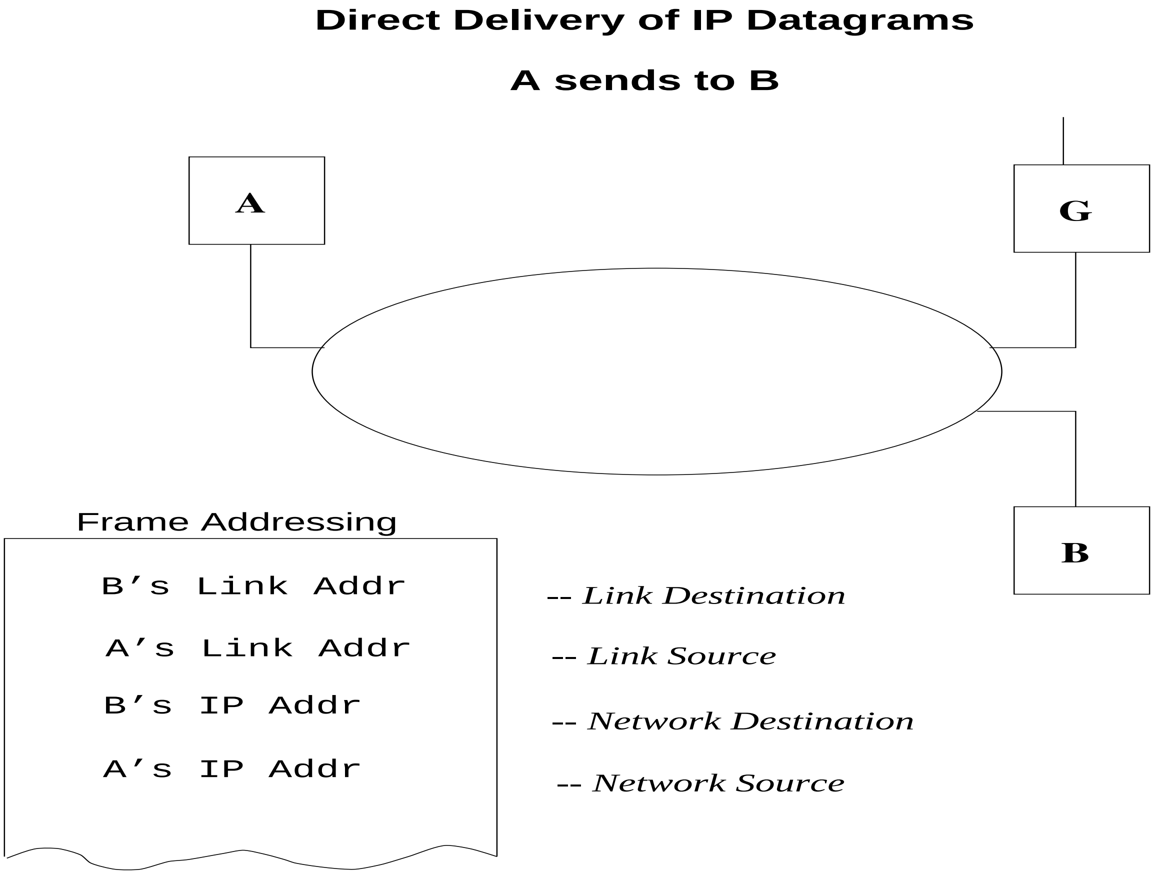

Figure 1: Direct IP Delivery

Indirect Delivery

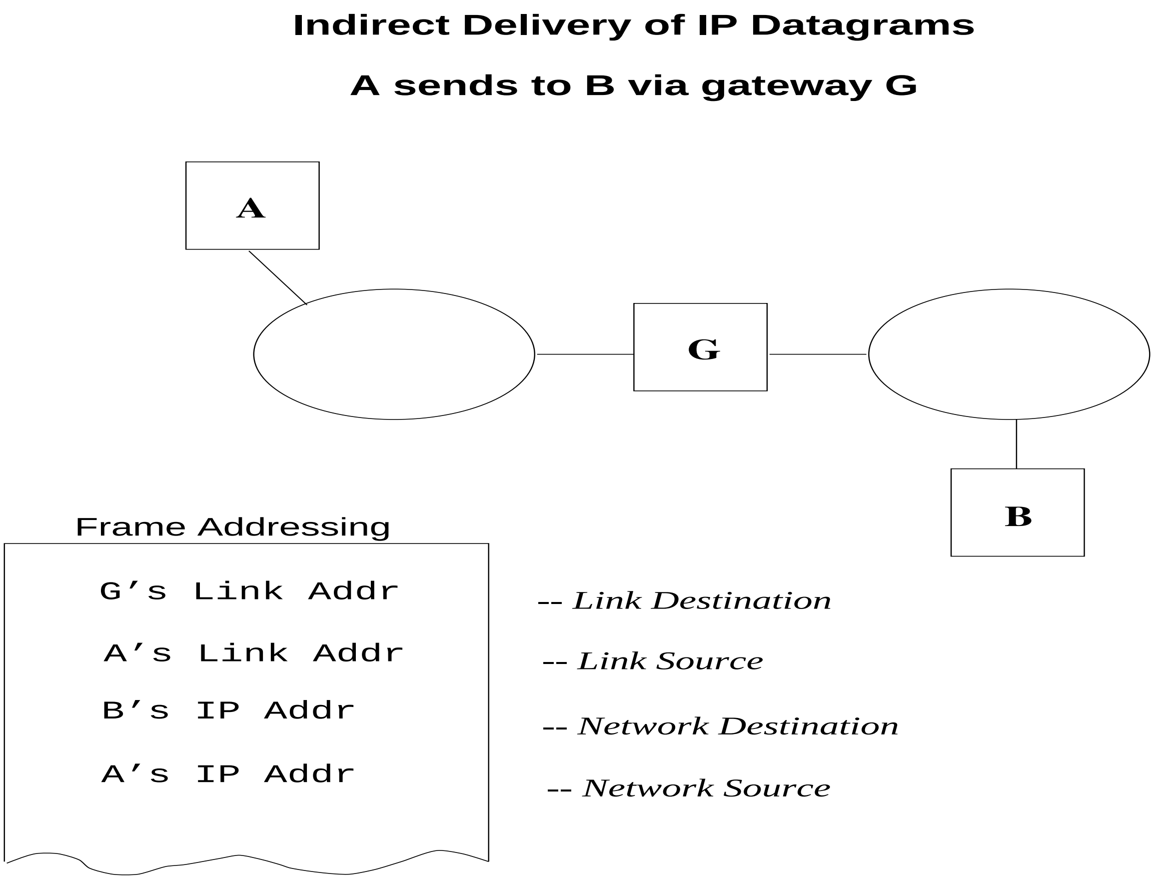

Figure 2: IP Indirect Delivery

Routing with Subnets

- Standard algorithm:
  - Routing table entries of the form:

    (network address, next-hop-address)
  - Algorithm knows implicitly net/host boundary locations
- Extended algorithm:
  - Routing table entries of the form:

    (subnet mask, network address, next-hop-address)
  - Algorithm must first apply subnet mask to destination address
- In practice, mask is static

IP Routing Algorithm with Subnets

1. Extract dest. IP address I_D from received datagram
2. For each k in $LNI_1 \ldots LNI_n$
   (a) Let dest network $I_N = I_D \wedge maskof(LNI_k)$
   (b) If $I_N = netof(LNI_k)$, perform direct delivery (done)
3. For each α in routing table:
   (a) Let $N = I_D \wedge (maskof(\alpha))$
   (b) If $N = (netof(\alpha))$,
   route to next hop (done)
4. If no matches found, declare routing error

- Observations:
  – Default route is all 0's mask field with 0 network address field
  – A host route is an all 1's mask field with the host address as network address field
  – To install a conventional Class B route, use `0xffff0000` mask field
  – Additional info in table implies fewer special cases in algorithm

Computer Naming

- Why Do We Name Machines?
  - Need a Way to Refer to Location of Services
  - Easier for Humans to Use than Numbers
- The Flat Name Space
  - every machine has its own unstructured name
  - sites must coordinate to not stomp on eachother
  - examples *ucbvax*, *sdcsvax*, *sri-nic*
  - stored in the centralized `HOSTS.TXT` file
  - *issue:* with the dawn of workstations, this file was now increasing at the rate of roughly users, rather than roughly organizations (mainframes)
  - didn't scale very well
- The Hierarchical Name Space
  - notion of a naming "domain"
  - much better scaling
  - decentralized control over naming assignment (not nec. geographical)
  - US examples: *ucsd.edu*, *nic.ddn.mil*, *ibm.com*, *usenix.org*
  - foreign examples: *cs.tut.fi*, *rena.dit.co.jp*, *dkuug.dk*

The Domain Name System

- Initial design of 1983
- `HOSTS.TXT` file insufficient
- DNS Design Criteria
  - provide at least as much info as text file
  - allow distributed database maintenance
  - no obvious limits to entry size, or associated info
  - inter-operate across DARPA Internet and others
  - have tolerable performance
  - "platform neutral"
  - "leanness"
- A "lean" Distributed Data-base:
  - no dynamic update (or atomicity)
  - no voting
  - no consideration for backup

DNS Architecture

- Two Primary Components
  - Name Servers
  - Resolvers
- Name Servers
  - repositories of information
  - agents which field mapping requests
- Resolvers
  - client interface to programs for DNS requests
  - often as a library

DNS Data Contents

- Keyed by *name*
- Data attached to each name:
  - Resource Record (RR)
  - each RR has a [type] and [class]
- Classes:
  - IN (Internet)
  - CH (CHOASnet, from MIT)
  - HS (Hesiod, from MIT)
- Some types (class IN):
  - A (host address)
  - NS (authoritative name server)
  - CNAME (canonical name for an alias)
  - SOA (start of an authority zone)
  - PTR (domain name pointer)
  - HINFO (host information)
  - MX (mail exchanger)
  - Resolvers

DNS Data Distribution

- Zones
  - complete description of contiguous section of total name space
  - created by obtaining delegation from a parent level
  - master file contains official zone contents
  - transferred to other servers via "zone transfer"
  - zone data is serialized, to know when to refresh
  - zone data is "authoritative"
- Caching
  - resolvers and combined servers/resolvers cache replies
  - caching is bounded by a TTL field on each RR
  - returns "non-authoritative" information
- Root Servers
  - 9 root servers ($\{A\text{-}I\}.ROOT\text{-}SERVERS.NET$)
  - other servers can offer *recursion*

DNS Lessons

- Surprises
  - semantics of data (e.g. how to order multiple A recs)
  - poor performance of underlying network (caching good)
  - negative caching
- Successes
  - variable depth hierarchy
  - topology/technology independence
  - datagram access (TCP was not as well-optimized then)
  - additional info processing
  - cache architecture (but can be misconfigured)
- Shortcomings
  - *class* and *types* grew smaller than expected
  - difficulty of upgrading applications
  - lack of expertise when given subdomain authority

An Example

- *nslookup*

```
cabot (1)> /usr/sbin/nslookup
Default Server:  ucsd.edu
Address:  132.239.1.1
> set debug
> nic.ddn.mil.
Server:  ucsd.edu
Address:  132.239.1.1
res_mkquery(0, nic.ddn.mil, 1, 1)
------------
Got answer:
    HEADER:
opcode = QUERY, id = 3, rcode = NOERROR
header flags:  response, want recursion, recursion avail.
questions = 1,  answers = 1,  authority records = 2,  additional = 2
    QUESTIONS:
nic.ddn.mil, type = A, class = IN
    ANSWERS:
    ->  nic.ddn.mil
internet address = 192.112.36.5
ttl = 518374 (5 days 23 hours 59 mins 34 secs)
    AUTHORITY RECORDS:
    ->  NIC.DDN.MIL
nameserver = NIC.DDN.MIL
ttl = 518374 (5 days 23 hours 59 mins 34 secs)
    ->  NIC.DDN.MIL
nameserver = DIIS-DEV.DDN.MIL
ttl = 518374 (5 days 23 hours 59 mins 34 secs)
    ADDITIONAL RECORDS:
    ->  NIC.DDN.MIL
internet address = 192.112.36.5
ttl = 518374 (5 days 23 hours 59 mins 34 secs)
    ->  DIIS-DEV.DDN.MIL
internet address = 192.112.38.89
ttl = 518374 (5 days 23 hours 59 mins 34 secs)
------------
Non-authoritative answer:
Name:    nic.ddn.mil
Address:  192.112.36.5
```

The Internet Protocol (IP)

Concepts

- Conceptually one fully connected network
- Underlying topology and architecture is hidden
- Abstraction is similar to link layer packet nets:
  - unreliable delivery
  - datagram abstraction
- Major Features
  - Single addressing format
  - Single maximum datagram size
  - Routing
- Transport layers (TCP and UDP) require IP underneath
- Actually includes both IP and ICMP (later)
- Version 4 evolving to IPng (version 6) (later)

IP Characteristics

- Unreliable Delivery
  - lost datagrams
  - duplicate datagrams
  - out of sequence datagrams
  - corrupted datagrams
  - IP does not detect any of these conditions
- Best–Effort Delivery
  - drops packets only when necessary
  - generally, most data delivered
- Key Components:
  - Packet format
  - Routing function
  - Rule set for datagram delivery

IP Header Format

- Header Format (from RFC791):

```
                    [1 tick = 1 bit offset]

 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    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
                Example Internet Datagram Header
```

- Version (4 bits) [currently 4, 5 is STII, 6 is IPv6]
- IHL – Internet Header Length (4 bits)
  - number of words in the header in 32-bit word units
  - minimum value is 5 (20-byte IP header, no options typ.)

IP Header Fields (summary)

- TOS – Type of Service (8 bits)
  - can specify abstract notion of TOS parameters
  - currently ignored by most implementations
- Total Length (16 bits)
  - number of bytes (octets) comprising the datagram
  - includes IP header and data fields
  - provides for datagram size of $(2^{16} - 1) = 65535$ octets
- Identification (16 bits)
  - identifying value assigned by sender
  - used to assemble datagram fragments
- Control Flags (3 bits)
  - Bit 0 – reserved, must be 0
  - Bit 1 – "Don't Fragment" ($0 =$ may, $1 =$ may not) fragment
  - Bit 2 – "Last Fragment" ($0 =$ last, $1 =$ more coming)

IP Header Fields (summary, cont)

- Fragment Offset (13 bits)
  - location this fragment belongs in overall datagram
  - units are in 64-bit quantities (8 octets)
- TTL – Time to Live (8 bits)
  - bounds maximum datagram lifetime
  - any node touching dgram decrements TTL field by one
  - if TTL becomes 0, datagram is dropped
- Protocol (8 bits)
  - Indicates to which transport protocol this dgram is destined
  - Look in "Assigned Numbers" RFC for current values
  - TCP is 6, UDP is 17
- Header Checksum (16 bits)
  - TTL decremented, Cksum field zeroed and cksum computed
  - computed cksum placed in IP packet, protects header
- Source and Destination IP Addresses (32 bits each)

IP Fragmentation

- *Fragmentation*
  - concept: chop up full IP datagram into *fragments*
  - any fragment must be less than link layer's MTU size
  - provides consistent view of IP datagram size at any endpoints
  - isolates datagram size from physical media constraints
- Fragmentation performed when datagram attempts to enter a net with too small an MTU size
- fragments may be fragmented
- fragment sizes limited to multiples of 8 bytes
- requirement: all hosts and gateways must deal with up to size 576 byte datagrams
- fragments and datagrams are handled identically when $<$ MTU size

IP Fragmentation

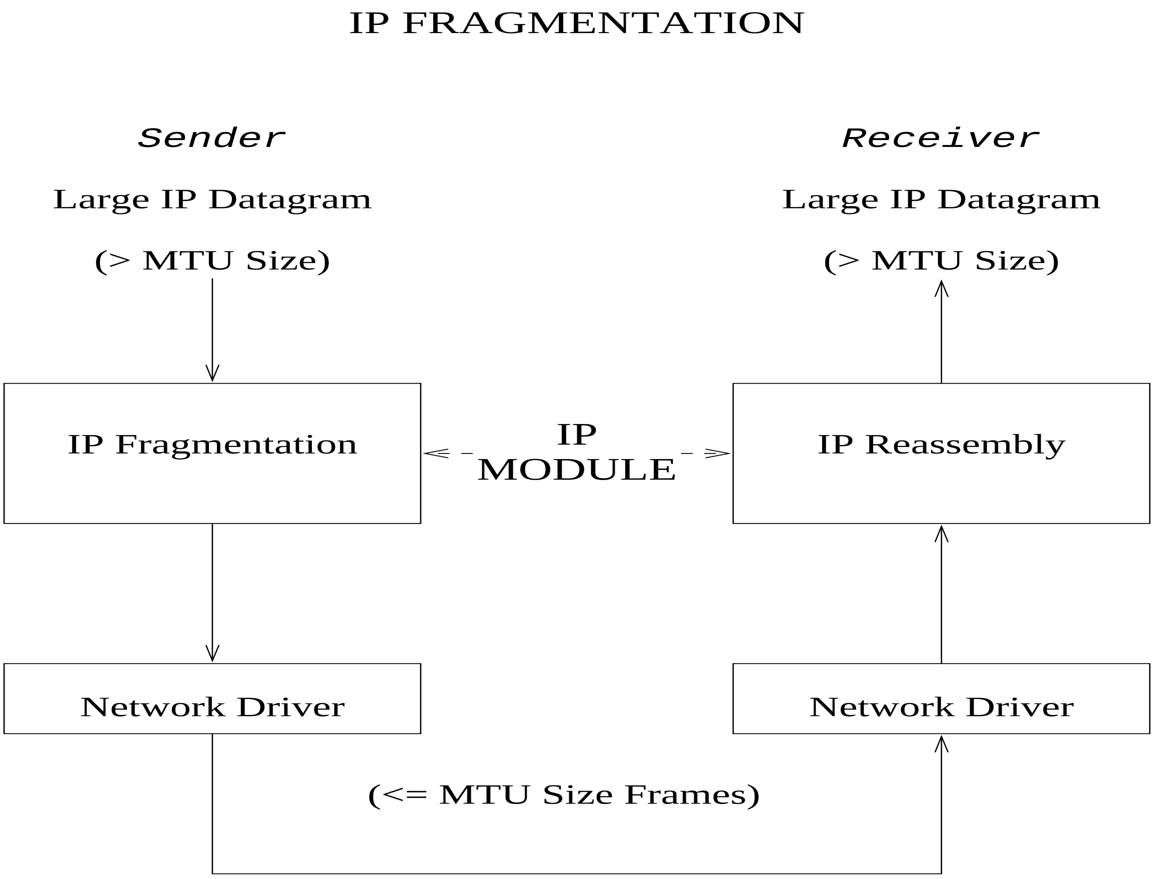

Figure 3: IP Fragmentation

IP Reassembly

- *Reassembly* – reconstruct IP datagram from fragments
- Performed at the ultimate destination:
  - small fragments must traverse network from point of fragmentation
  - loss of any single fragment implies entire datagram loss
  - probability of datagram loss grows with number of fragments
  - timeout of 60-120 secs causes flushing and ICMP Time Exceeded Message (RFC 1122)
- So, why do it this way?
  - simplifies gateway code (no reassembly queue)
  - allows for independent routing of fragments (may improve fault tolerance)

IP Options

- Options indicate special handling for particular IP datagrams
- All implementations must support options processing
- Option field is variable in length, up to a maximum
- Zero or more options present for any datagram
- Some Options:
  - Loose Source Routing (RFC791)
  - Strict Source Routing (RFC791)
  - Record Route (RFC791)
  - Probe and Reply MTU (RFC1063)
  - Security (RFC1108)

ICMP:
Internet Control Message Protocol

- Problem:
  - IP provides no direct way to find out the fate of a packet
  - need a mechanism to delivery error and control info to source
- ICMP Protocol (really part of IP module):
  - Specified by RFC 792, also see RFC 1122, RFC 1812
  - encapsulated in IP datagram like any other data
  - message destination is source's protocol module
  - generally, ICMP provides IP to IP module control communication
  - reports errors or conditions; doesn't prescribe cures
  - two classes of messages: *queries* and *errors*
  - ICMP message generation is limited to avoid *broadcast storms*

ICMP Restrictions

- ICMP Messages are *not* generated in response to (see RFC1812):
  - An ICMP error message (ok for queries)
  - A datagram with a bad or wrong-length IP header
  - A datagram destined for an IP multicast or broadcast address
  - A datagram sent using link-layer broadcast or multicast
  - An IP fragment other than the first
  - A datagram with a zero, loopback, broadcast or multicast source address

ICMP Message Format

- ICMP header appears as beginning of IP packet data portion
- All ICMP messages begin with same 3 fields (32 bits):

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Type      |     Code      |          Checksum             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

- The type field indicates 1 of 15 types of messages
- The code field indicates sub-types of the 15 major types
- The checksum covers the entire ICMP message and is required
- Errors traditionally contain offending IP header plus first 8 data bytes; according to RFC1812, it should contain all data such that datagram is less than 576 bytes

ICMP Message Types

- Values for the `type` field:
  - 0 = Echo Reply ("ping" response) [query]
  - 3 = Destination Unreachable [error]
  - 4 = Source Quench [error]
  - 5 = Redirect (change a route) [error]
  - 8 = Echo Request ("ping" request) [query]
  - 9 = Router Advertisement (RFC1256) [query]
  - 10 = Router Solicitation (RFC1256) [query]
  - 11 = Time Exceeded for a Datagram [error]
  - 12 = Parameter Problem for a Datagram [error]
  - 13 = Timestamp Request [query]
  - 14 = Timestamp Reply [query]
  - 15 = Information Request (obsolete) [query]
  - 16 = Information Reply (obsolete) [query]
  - 17 = Address Mask Request [query]
  - 18 = Address Mask Reply [query]

ICMP Dest Unreachable

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Type (3)    | Code (below)  |          Checksum             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                             unused                            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      Internet Header + 64 bits of Original Data Datagram      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

– `Code` indicates what is unreachable:
  * 0 = net unreachable
  * 1 = host unreachable
  * 2 = protocol unreachable
  * 3 = port unreachable
  * 4 = frag. needed but "don't fragment" bit was set
  * 5 = source route failed
  * 6 = destination network unknown (not used) (RFC 1122)
  * 7 = destination host unknown (RFC 1122)
  * 8 = source host isolated (obsolete) (RFC 1122)
  * 9 = communication with dest. network prohibited (RFC 1122)
  * 10 = communication with dest. host prohibited (RFC 1122)
  * 11 = network unreachable for TOS (RFC 1122)
  * 12 = host unreachable for TOS (RFC 1122)
  * 13 = communication administratively prohibited by filtering (RFC1812)
  * 14 = host precedence violation (RFC1812)
  * 15 = precedence cutoff in effect (RFC1812)

– Ports represent transport-level entities (later)

ICMP Dest Unreachable

- Host and Network Unreachable:
  - generated by gateways in response to a datagram it cannot forward because the destination is unreachable or down
  - network unreachable implies no next-hop entry for network, or all paths down
  - host unreachable imples the destination network is reachable, but the destination host is somehow not reachable
- Protocol Unreachable indicates no appropriate protocol implementation
- Port Unreachable indicates message cannot be fully delivered
- Host Precedence Violation:
  - Sent the first hop router to a host to indicate that a requested precedence is not permitted for the particular combination of source/destination host or
- Precedence Cutoff in Effect:
  - The network operators have imposed a minimum level of precedence required for operation, the datagram was sent with a precedence below this level

ICMP Time Exceeded

- Time Exceeded

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Type (11)   | Code (0 or 1) |          Checksum             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                             unused                            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      Internet Header + 64 bits of Original Data Datagram      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

  - `Code` field:
    - 0 = TTL exceeded in transit
    - 1 = fragment reassembly time exceeded
  - Gateways observing datagrams with TTL set to 0 must discard them
  - Gateways must also send a Time Exceeded message to the source
  - When reassembling datagrams, gateways act as hosts (i.e. when the gateway itself is the destination)

ICMP Redirect

- Redirect

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Type (5)    | Code (below)  |          Checksum             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                 Gateway Internet Address                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      Internet Header + 64 bits of Original Data Datagram      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

  - `Code` Field:
    * 0 = Redirect datagrams for the Network
    * 1 = Redirect datagrams for the Host
    * 2 = Redirect datagrams for the Type of Service and Network
    * 3 = Redirect datagrams for the Type of Service and Host
  - `Gateway IP Address` Field:
    * indicates IP address of gateway to use for destination network

ICMP Redirect

- Redirect (continued)
  - Description:
    - gateway G receives a datagram D from host H
    - G inspects its routing tables to determine next hop N for D
    - if N and H reside on the same network, G sends *redirect* to H
    - G also sends D on to N
    - H should update its routing table to use N versus G

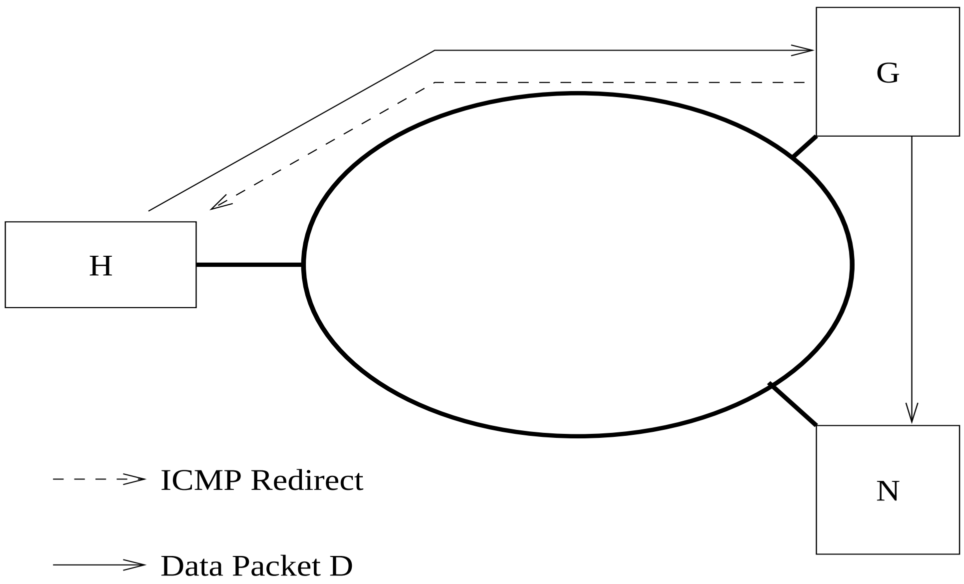

Figure 4: Operation of an ICMP Redirect

ICMP Address Mask

- Address Mask Request/Reply

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Type (17 or 18)|  Code (0)     |          Checksum             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Identifier (for matching)     |Sequence Number (for matching) |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         Address Mask                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

  – Address Mask:
    * subnet mask used on current network (subnetting discussed later)
  – Description
    * a host requesting the address mask may query a gateway directly or broadcast the query

ICMP Router Discovery

- Router Advertisement

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Type      |     Code      |           Checksum            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Num Addrs   |Addr Entry Size|           Lifetime            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Router Address[1]                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      Preference Level[1]                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Router Address[2]                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      Preference Level[2]                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                               .                               |
|                               .                               |
|                               .                               |
```

- Fields
  - Num Addrs - the number of addresses advertised in this message
  - Addr Entry Size - number of 32-bit words per each router address (2 here)
  - Lifetime - max number of seconds the addresses are valid

ICMP Router Discovery

- Fields (cont'd)
  - Router Address - the sending router's IP address(es) on the attached network
  - Preference Level - the preferability of each router address as a default as compared with others
- Routers may periodically multicast an advertisement from each of its interfaces
- Hosts discover the router simply by listening to these multicasts, or may send a *solicitation*
- Default advertising rate is once each 7-10mins, default life 30mins

ICMP Router Discovery

- Router Solicitation

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Type      |     Code      |           Checksum            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           Reserved                            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

- Idea is to multicast request to *all-routers* address (224.0.0.2) with ttl 1
- Alternative is to use network-limited broadcast (255.255.255.255)
- Fields
  - Reserved - sent as zero
- Hosts may solicitations (e.g. when booting) to acquire a first route
- A "small number" of retransmissions is provided for

IP Next Generation (IPng or IPv6)

- see RFC1883-1886
- Interesting Features:
  - larger addresses (128 bits)
  - "jumbo" datagrams (option for 32 bits length)
  - simplified header (IPv4 incompatible)
  - base + extension headers
  - authentication and privacy (see RFC1825-7)
  - autoconfiguration
  - extended function source routing
  - "flow" labels
  - fragmentation *only* at source

IPng Addressing

- Three General Classes of Addresses:
  - Unicast (one per interface)
  - Anycast (set of interfaces, delivered to 'nearest' intf)
  - Multicast (all interfaces given the address)
- Note: *no broadcast address*
- Text Representation of Addresses:
  - x:x:x:x:x:x:x:x:, x is 16 bits in hex
  - (e.g. FEDC:BA98:7654:3210:FEDC:BA98:7654:3210)
  - :: indicates multiple zeros
  - (e.g. FF01:0:0:0:0:0:0:43 is FF01::43)
- ::1 (loopback address), :: (unspecified address)
- Already-Allocated Pieces of the Address Space:
  - NSAP (.78%)
  - IPX (.78%)
  - Provider-Based Unicast (12.5%)
  - Geographic-Based Unicast (12.5%)
  - Link-Local (.098%)
  - Site-Local (.098%)
  - Multicast (.39%)

IPng Header

- IPv6 Header:

```
|<-------------------- 32 bits across ---------------------->|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Version| Prio. |              Flow Label                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Payload Length        |  Next Header  |   Hop Limit   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+                                                               +
|                                                               |
+                         Source Address                        +
|                                                               |
+                                                               +
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+                                                               +
|                                                               |
+                      Destination Address                      +
|                                                               |
+                                                               +
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

- Extension Headers:
  - Hop-by-Hop Option Header
  - IPv6 Header Options
  - Routing Header
  - Fragment Header
  - Authentication Header
  - Privacy Header
  - End-to-End Option Header

IPng Header

- Version (as in IPv4)
- Prio
  - source-specific priority ordering
  - 0–7: congestion-controlled traffic
    1. uncharacterized
    2. filler (e.g. netnews)
    3. unattended data transfer (e.g. email)
    4. (reserved)
    5. attended data transfer (e.g. FTP, NFS)
    6. (reserved)
    7. interactive traffic (e.g. telnet, X)
    8. interactive control (e.g. routing, SNMP)
  - 8–15: low-to-high non-congestion-controlled traffic
- Flow Label – random value at source to associate datagrams
- Payload Length – packet w/out header (0 indicates Jumbo option)
- Next Header – type of next header (IPv4 Proto Field)
- Hop Limit – max number of hops before discard
- Source Address – packet originator
- Destination Address – final destination *or next hop*

IPng Extension Headers

- IP layer info is encoded after base header, before transport headers
- With one exception, intermediate nodes do not interpret extension headers
- Example Layouts:

```
+---------------+------------------------
|  IPv6 header  | TCP header + data
| Next Header = |
|      TCP      |
+---------------+------------------------

+---------------+----------------+------------------------
|  IPv6 header  | Routing header | TCP header + data
| Next Header = |  Next Header = |
|    Routing    |      TCP       |
+---------------+----------------+------------------------

+---------------+----------------+-----------------+-----------------
|  IPv6 header  | Routing header | Fragment header | fragment of TCP
| Next Header = |  Next Header = |  Next Header =  |
|    Routing    |    Fragment    |       TCP       |
+---------------+----------------+-----------------+-----------------
```

- Ext headers are processed in order

Some Extension Headers

- Hop-by-Hop Extension Header
  - optional info examined by each node along delivery path
  - identified by `next hop` set to 0
  - can contain padding or "Jumbo Payload Option"
  - Jumbo Payload Option:
    * length of packet, including hop-by-hop option header
    * for sizes greater than 65,535 bytes
    * not used for fragments
- Fragment Header – support for fragmentation
- Destination Options – options relevant only to destination

Routing Extension Header

- Provides for IPv4-style source and record route
- Format:

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Next Header  |  Hdr Ext Len  |  Routing Type | Segments Left |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

- `routing type` field indicates particular routing header variants
- `segments left` lists number of addresses yet to be visited
- The Type 0 format (follows header above):

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Reserved    |             Strict/Loose Bit Map              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+                                                               +
|                                                               |
+                           Address[1]                          +
|                                                               |
+                                                               +
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                ...... more addresses ......                   |
```

- `strict/loose bit map` addresses listed are 1=strict, 0=loose

Routing Header Example

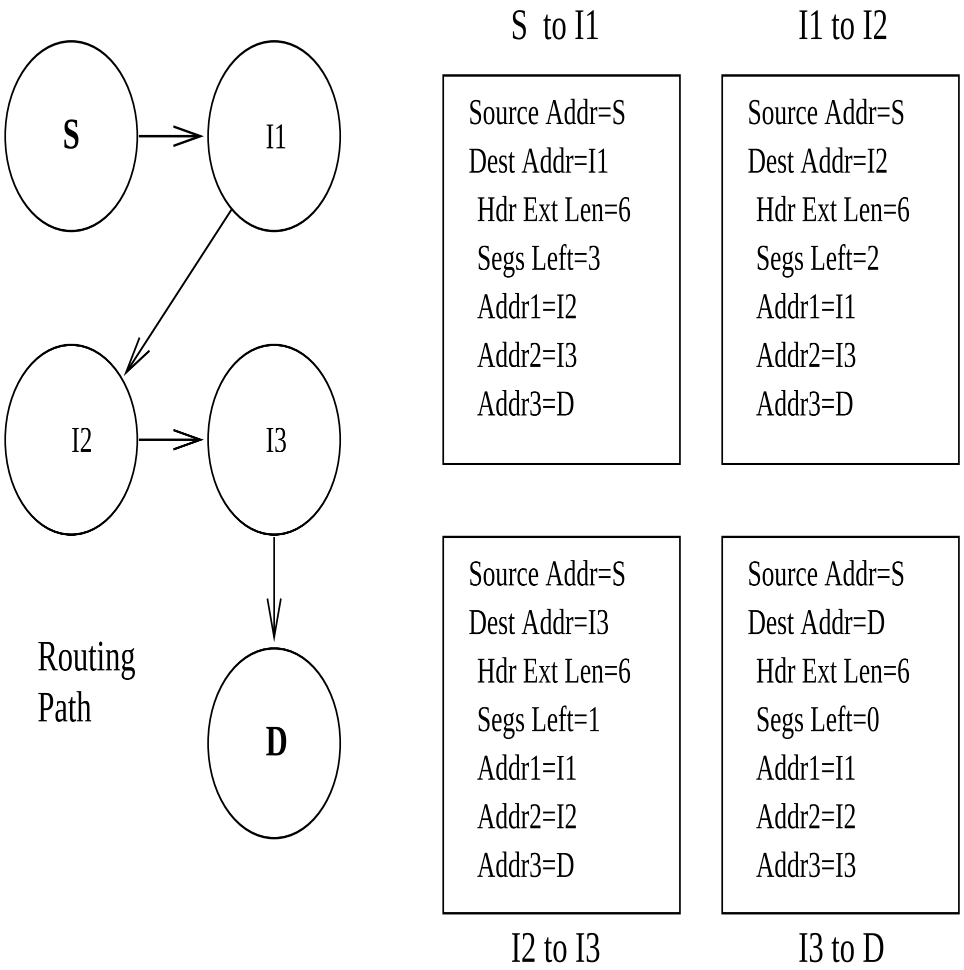

Figure 5: Using the Routing Header in IPng