The transport layer relies on the services of the network layer, which provides a communication service between hosts. In particular, the network layer moves transport-layer segments from one host to another. At the sending host, the transport-layer segment is passed to the network layer. It is then the job of the network layer to get the segment to the destination host and pass the segment up the protocol stack to the transport layer.
Network Layer

Chapter goals:
- understand principles behind network layer services:
  - routing
  - how a router works
  - advanced topics: IPv6, multicast
- instantiation and implementation in the Internet

Overview:
- network layer services
- routing principles
- IP addresses
- Internet routing protocols
- reliable transfer:
  - intra-domain
  - inter-domain
- ICMP
- Routers, bridges and switches
- IPv6
- multicast routing
Network layer functions

- transport packet from sending to receiving hosts
- network layer protocols in every host, router

three important functions:
- path determination: route taken by packets from source to dest. Routing algorithms
- switching: move packets from router’s input to appropriate router output
- call setup: some network architectures require router call setup along path before data flows

The network layer involves each and every host and router in the network. The role of the network layer in a sending host is to begin the packet on its journey to the receiving host. The role of the network layer is thus deceptively simple—to transport packets from a sending host to a receiving host. To do so, three important network-layer functions can be identified:

- **Path determination.** The network layer must determine the route or path taken by packets as they flow from a sender to a receiver. The algorithms that calculate these paths are referred to as **routing algorithms.**

- **Switching.** When a packet arrives at the input to a router, the router must move it to the appropriate output link.

- **Call setup.** With TCP, a three-way handshake is required before data actually flow from sender to receiver. This allowed the sender and receiver to set up the needed state information (for example, sequence number and initial flow-control window size). In an analogous manner, some network-layer architectures (for example, ATM) require that the routers along the chosen path from source to destination handshake with each other in order to setup state before data actually begins to flow. In the network layer, this process is referred to as **call setup.** The network layer of the Internet architecture does not perform any such call setup.
The *network-service model* defines the characteristics of end-to-end transport of data between one "edge" of the network and the other, that is, between sending and receiving end systems. The most important abstraction that the network layer provides to the transport layer is whether the network layer uses a *connection-oriented service* (virtual circuits) or a *connectionless service* (datagram) model.

With a **virtual circuits layer**, a circuit management (setup, data-transfer, teardown) and signaling are needed.

With a **datagram network layer**, each time an end system wants to send a packet, it stamps the packet with the address of the destination end system, and then pops the packet into the network.
Virtual circuits

“source-to-dest path behaves much like telephone circuit”
- performance-wise
- network actions along source-to-dest path

- call setup, teardown for each call before data can flow
- each packet carries VC identifier (not destination host ID)
- every router on source-dest path maintains "state" for each passing connection
  - transport-layer connection only involved two end systems
- link, router resources (bandwidth, buffers) may be allocated to VC
  - to get circuit-like performance

There are three identifiable phases in a virtual circuit:

- **VC setup.** During the setup phase, the sender contacts the network layer, specifies the receiver address, and waits for the network to set up the VC. The network layer determines the path between sender and receiver, that is, the series of links and packet switches through which all packets of the VC will travel. This typically involves updating tables in each of the packet switches in the path. During VC setup, the network layer may also reserve resources (for example, bandwidth) along the path of the VC.

- **Data transfer.** Once the VC has been established, data can begin to flow along the VC. Each packet knows the VC (carries the VC identifier), and every switch on the VC path is aware of the existence of the VC itself.

- **Virtual-circuit teardown.** This is initiated when the sender (or receiver) informs the network layer of its desire to terminate the VC. The network layer will then typically inform the end system on the other side of the network of the call termination and update the tables in each of the packet switches on the path to indicate that the VC no longer exists.
Virtual circuits: signaling protocols

- used to setup, maintain, teardown VC
- used in ATM, frame-relay, X.25
- not used in today’s Internet (at network layer)

The messages that the end systems send to the network to indicate the initiation or termination of a VC, and the messages passed between the switches to set up the VC (that is, to modify switch tables) are known as **signaling messages** and the protocols used to exchange these messages are often referred to as **signaling protocols**. In the Internet virtual circuits is not used at network layer, while ATM, frame relay and X.25, are three other networking technologies that use virtual circuit.
Datagram networks: the Internet model

- no call setup at network layer
- routers: no state about end-to-end connections
  - no network-level concept of "connection"
- packets typically routed using destination host ID
  - packets between same source-dest pair may take different paths

Datagram routing is similar to routing ordinary postal mail: packet switches route a packet toward its destination by examining the packet's destination address, indexing a routing table with the destination address, and forwarding the packet in the direction of the destination. There is no “connection management” as well as no tables or state are needed inside the network. Different packets to the same destination can be routed via a different route. The current Internet architecture provides only one service model, the datagram service, which is also known as "best-effort service." From the table, it might appear that best effort service is a euphemism for "no service at all." With best-effort service, timing between packets is not guaranteed to be preserved, packets are not guaranteed to be received in the order in which they were sent, nor is the eventual delivery of transmitted packets guaranteed. Given this definition, a network that delivered no packets to the destination would satisfy the definition of best-effort delivery service. However, there are sound reasons for such a minimalist network service model:

- it is easier to interconnect networks that used very different link-layer technologies
- it is easier to add a new service simply by attaching a host to the network and defining a new higher-layer protocol

The Internet's best-effort only service model is currently being extended to include so-called integrated services and differentiated service.
The Internet's network layer provides connectionless datagram service rather than virtual-circuit service. When the network layer at the sending host receives a segment from the transport layer, it encapsulates the segment within an IP datagram, writes the destination host address as well as other fields in the datagram, and sends the datagram to the first router on the path toward the destination host. The Internet’s network layer has three major components:

• The Internet Protocol, or more commonly, the **IP Protocol**, which defines network-layer addressing, the fields in the datagram (that is, the network-layer PDU), and the actions taken by routers and end systems on a datagram based on the values in these fields. There are two versions of the IP protocol in use today: **IPv4** [RFC 791] and **IPv6** [RFC 2071; RFC 2460], which has been proposed to replace IPv4 in upcoming years.

• The second major component of the network layer is the path determination component; it determines the route a datagram follows from source to destination. Examples of such components used in the Internet are RIP, OSPF, BGP.

• The Internet's network-layer error and information reporting protocol, **ICMP**, is a facility to report errors in datagrams and respond to requests for certain network-layer information.
Internet and intranet

- an intranet
  a collection of end and intermediate systems
  interconnected using the TCP/IP architecture
  normally inside one organization
- the Internet
  the global collection of all hosts and routers
  interconnected using the TCP/IP architecture
  coordinated allocation of addresses and implementation
  requirements by the Internet Society
- intranets are often connected to the Internet by
  firewalls
  hosts that act as application level relays

- an internet can use its own addresses
- Internet addresses are managed worldwide

There is no global Internet organization: like for telephony, the Internet service is provided by a collection of competing Internet Service Providers (ISPs)

Only addresses and standards are managed world-wide.
An IP address is technically associated with an interface, rather than with the host or router containing that interface. When IP in the host wants to send a datagram, it will do so over this link. The boundary between the host and the physical link is called an *interface*. A router's job is to receive a datagram on an "incoming" link and forward the datagram on some "outgoing" link, thus a router has multiple interfaces, one for each of its links.

Each IP address is 32 bits long (equivalently, four bytes), and there are thus a total of $2^{32}$ possible IP addresses. These addresses are typically written in so-called *dotted-decimal notation*, in which each byte of the address is written in its decimal form and is separated by a period ("dot") from other bytes in the address. For example, consider the IP address 193.32.216.9. The 193 is the decimal equivalent of the first eight bits of the address; the 32 is the decimal equivalent of the second eight bits of the address, and so on. Thus, the address 193.32.216.9 in binary notation is:

```
11000001 00100000 11011000 00001001.
```

Each interface on every host and router in the global Internet must have an IP address that is globally unique.
IP addresses cannot be chosen in a willy-nilly manner, however. In primis, an interface's IP address will be determined by the "network" to which it is connected. The three hosts in the upper-left portion are on the same “IP network” identified by the initial part of their address 128.178.1, and the router interface to which they are connected all have an IP address of the form 128.178.1.xxx. That is, they share a common leftmost 24 bits of their IP address. The 24 address bits that they share in common constitute the network portion of their IP address; the remaining eight bits are the host portion of the IP address. The network itself also has an address: 128.178.1.0/24, where the "/24" notation, sometimes known as a network mask, indicates that the leftmost 24 bits of the 32-bit quantity define the network. Sometimes, in the network mask the numbers are substituted with 255. Examples:

subnet mask at EPFL = **255.255.255.0**

What are the net:subnet and host parts of: **lrcsuns.lrc.epfl.ch**? The address is **128.178.156.24**

the prefix is **128.178.156.0**
At the origin, the prefix of an IP address was defined in a very rigid way. For class A addresses, the prefix was 8 bits. For class B, 16 bits. For class C, 24 bits. The interest of that scheme was that by simply analyzing the address you could find out what the prefix was.

The requirement that the network portion of an IP address be exactly one, two, or three bytes long turned out to be problematic for supporting the rapidly growing number of organizations with small and medium-sized networks. A class C (/24) network could only accommodate up to \(2^{24} - 2 = 254\) hosts (two of the \(2^{24} = 256\) addresses are reserved for special use)--too small for many organizations. However, a class B (/16) network, which supports up 65,634 hosts was too large.

Under classful addressing, an organization with, say, 2,000 hosts was typically allocated a class B (/16) network address. This led to a rapid depletion of the class B address space and poor utilization of the assigned address space. It was soon recognized that this form was too rigid. Then subnets were added. It was no longer possible to recognize from the address alone where the subnet prefix ends and where the host identifier starts. For example, the host part at EPFL is 8 bits; it is 6 bits at ETHZ. Therefore, an additional information, that is the subnet mask, is necessary.

Class C addresses were meant to be allocated one per network. Today they are allocated in contiguous blocks.

\[\text{Examples: } 128.178.x.x = \text{EPFL host}; 129.132.x.x = \text{ETHZ host}; 9.x.x.x = \text{IBM host}; 18.x.x.x = \text{MIT host}\]
With so-called CIDRized (CIDR: Classless Interdomain Routing) network addresses, the network part of an IP address can be any number of bits long, rather than being constrained to 8, 16, or 24 bits. A CIDRized network address has the dotted-decimal form `a.b.c.d/x`, where `x` indicates the number of leading bits in the 32-bit quantity that constitutes the network portion of the address. In our example above, the organization needing to support 2,000 hosts could be allocated a block of only 2,048 host addresses of the form `a.b.c.d/21`, allowing the approximately 63,000 addresses that would have been allocated and unused under classful addressing to be allocated to a different organization. In this case, the first 21 bits specify the organization's network address and are common in the IP addresses of all hosts in the organization. The remaining 11 bits then identify the specific hosts in the organization. In practice, the organization could further divide these 11 rightmost bits using a procedure known as **subnetting** to create its own internal networks within the `a.b.c.d/21` network.
IP Addresses (examples -1)

- subnet mask at ETHZ = \(255.255.255.192\) (that is \(11111111.11111111.11111111.11000000\))
- CIDR \(129.132.0.0/26\)
- question: net:subnet and host parts of spr13.tik.ee.ethz.ch = 129.132.119.77 ?
- answer: (77 = 01001101)
  - net:subnet = 129.132.119.64 (64 = 01000000)
  - host = 13 = 001101 (6 bits)
Sovkom has received IP addresses $194.167.0.0$ to $194.167.255.255$ total: $2^{16}$ addr., but .0 and .255 are not usable.

Java Business Solutions AG has received IP addresses $194.167.41.0$ to $194.167.41.255$ total: $2^8$ -2 addresses.

Tango SA has received IP addresses $194.167.42.0$ to $194.167.43.255$ total: $2^8(2^8-2)$ addresses.
Special Case IP Addresses

1. 0.0.0.0  this host
2. 0.hostId  specified host on this net
3. 255.255.255.255  limited broadcast (not forwarded by routers)
4. netId.all 1’s  broadcast on this net
5. netId.subnetId.all 1’s  broadcast on this subnet
6. 127.x.x.x  loopback
7. 10/8  reserved networks for internal use
172.16/12
192.168/16

Example: 128.178.255.255: broadcast to EPFL
128.178.156.255: broadcast to all LRC net
128.178.156.0: LRC net
129.132.119.64: tik-sprach
hostId = 0 designates the network

The following address blocks are reserved and cannot be used in the Internet. They are typically used in experimental or closed environments
10.0.0.0 - 10.255.255.255 (10/8)
172.16.0.0 - 172.31.255.255 (172.16/12)
192.168.0.0 - 192.168.255.255 (192.168/16)
We have seen the main principles of IP addressing: an IP address identifies an interface of an host (rather than the host itself); this address is unique in the Internet. In order to communicate, IP hosts needs to know the IP addresses. We can distinguish between routing inside a subnetwork and routing between subnetworks, where a subnetwork is a collection of hosts that can communicate directly without routers.
The IP packet forwarding algorithm is the core of the TCP/IP architecture. It defines what a system should do with a packet it has to send or to forward. The rule is simple:

- if the destination IP address has the same prefix as one of self’s interfaces, send directly to that interface
- otherwise send to a router as given by the IP routing table

Example of IP routing tables and Interface Tables:

The IP packet forwarding algorithm is the core of the TCP/IP architecture. It defines what a system should do with a packet it has to send or to forward. The rule is simple:

- if the destination IP address has the same prefix as one of self’s interfaces, send directly to that interface
- otherwise send to a router as given by the table

It uses the IP routing table; the table can be checked with a command such as “netstat” with Unix or “Route” with Windows NT
In reality there are exceptions to the rule. The complete algorithm is as above; the cases should be tested in that order (it is a nested if then else statement).

Remember that the above is the packet forwarding algorithm. The tables are written by the control method (the routing algorithms).
Every IP datagram has a source address field and a destination address field. The source host fills a datagram's source address field with its own 32-bit IP address. It fills the destination address field with the 32-bit IP address of the final destination host to which the datagram is being sent. The data field of the datagram is typically filled with a TCP or UDP segment. The IP datagram travels inside the network remaining unchanged. For routing purpose, the fields of main interest (e.g. the fields that are read and used) are the two addresses: source and destination. The way the network transports the datagram from the source to the destination depends on whether the source and destination reside on the same subnetwork.
### SOME INFO

#### DEADLINES
- Test1: 7 May - CO1
- Test2: 21 May - CO1
- Test3: 18 June - CO1
- Lab Session: 4 June - IN3
- Project due: 17 June

#### WEB NEWS
- TCP/UDP Java code
- Exercises
- new chapters
- course material
- if time: new chapter
Last week

transferred the apps data from S to D!

Transport Layer: Logical communication between processes

Reliable data transfer

- data received ordered & error-free
- Elements of procedure usually means the set of following functions
  - Error detection and correction (e.g. ARQ)
  - Flow Control
  - Connection Management

Retransmission Strategy

- Stop & Go
- Selective Repeat
- Go Back n

Internet:

- connectionless transport: UDP
  - checksum
  - pkt transmission
- connection-oriented transport: TCP
  - reliable service
  - flow and congestion control
  - TCP fairness
  - RTP

4: Network Layer 4-23
Last Week

- the network service model defines edge-to-edge channel
- transport pkt from sending to receiving hosts
- network layer protocols in every host, router
- the most important abstraction provided by network layer:
  - network-layer connection-oriented service: virtual circuit
  - network-layer connectionless service: datagram
- the network layer functions:
  - path determination
  - switching
  - call setup
- IP addressing:
  - network & host part
  - classes and CIDR
- IP principles:
  - homogeneous addressing
  - routing
  - routing to the same subnet
  - routing to another subnet
Host A wants to send an IP datagram to host B, which resides on the same network, 128.178.1.0/24, as A. This is accomplished as follows. IP in host A first consults its internal routing table, shown in Figure 4.22, and finds an entry, 128.178.1.0/24, whose network address matches the leading bits in the IP address of host B. The routing table shows that the number of hops to network 128.178.1.0 is 1, indicating that B is on the very same network to which A itself is attached. Host A thus knows that destination host B can be reached directly via A's outgoing interface, without the need for any intervening routers. Host A then passes the IP datagram to the link-layer protocol for the interface, which then has the responsibility of transporting the datagram to host B.
Host A wants to send a datagram to another host, say E, that is on a different network. Host A again consults its routing table and finds an entry, 128.178.2.0/24, whose network address matches the leading bits in the IP address of host E. Because the number of hops to the destination is 2, host A knows that the destination is on another network and thus an intervening router will necessarily be involved. The routing table also tells host A that in order to get the datagram to host E, host A should first send the datagram to IP address 128.178.1.4, the router interface to which A's own interface is directly connected. IP in host A then passes the datagram down to the link layer and indicates to the link layer that it should send the datagram to IP address 128.178.1.4. It's important to note here that although the datagram is being sent (via the link layer) to the router's interface, the destination address of the datagram remains that of the ultimate destination (host E.) not that of the intermediate router interface.
Arriving at 128.178.1.4, destined for 128.178.2.2
- look up network address of E
- E on same network as router's interface 128.178.2.9
  - router, E directly attached
- link layer sends datagram to 128.178.2.2 inside link-layer frame via interface 128.178.2.9
- datagram arrives at 128.178.2.2!!! (hooray!)

The datagram is now in the router, and it is the job of the router to move the datagram toward its ultimate destination. The router consults its own routing table and finds an entry, 128.178.2.0/24, whose network address matches the leading bits in the IP address of host E. The routing table indicates that the datagram should be forwarded on router interface 128.178.2.9. Since the number of hops to the destination is 1, the router knows that destination host E is on the same network as its own interface, 128.178.2.9. The router thus moves the datagram to this interface, which then transmits the datagram to host E.
The key fields in the IPv4 datagram are the following:

- **Version Number.** These four bits specify the IP protocol version of the datagram. By looking at the version number, the router can then determine how to interpret the remainder of the IP datagram. Different versions of IP use different datagram formats.

- **Header Length.** Because an IPv4 datagram can contain a variable number of options (that are included in the IPv4 datagram header) these four bits are needed to determine where in the IP datagram the data actually begins. Most IP datagrams do not contain options so the typical IP datagram has a 20-byte header.

- **TOS.** The type of service (TOS) bits were included in the IPv4 header to allow different "types" of IP datagrams to be distinguished from each other, presumably so that they could be handled differently in times of overload. When the network is overloaded, for example, it would be useful to be able to distinguish network-control datagrams from datagrams carrying data. It would also be useful to distinguish real-time datagrams from non-real-time traffic.

- **Datagram Length.** This is the total length of the IP datagram (header plus data) measured in bytes. Since this field is 16 bits long, the theoretical maximum size of the IP datagram is 65,535 bytes. However, datagrams are rarely greater than 1,500 bytes and are often limited in size to 576 bytes.

- **Identifier, Flags, Fragmentation Offset.** These three fields have to do with so-called IP fragmentation, a topic we will consider in depth shortly. Interestingly, the new version of IP, IPv6, does not allow for fragmentation at routers.

- **Time-to-live.** The time-to-live (TTL) field is included to ensure that datagrams do not circulate forever (due to, for example, a long-lived router loop) in the network. This field is decremented by one each time the datagram is processed by a router. If the TTL field reaches 0, the datagram must be dropped.

- **Protocol.** This field is used only when an IP datagram reaches its final destination. The value of this field indicates the transport-layer protocol at the destination to which the data portion of this IP datagram will be passed. For example, a value of 6 indicates that the data portion is passed to TCP, while a value of 17 indicates that the data is passed to UDP. For a listing of all possible numbers, see RFC 1700.

- **Header Checksum.** The header checksum aids a router in detecting bit errors in a received IP datagram. The header checksum is computed by treating each two bytes in the header as a number and summing these numbers using 1’s complement arithmetic.

- **Source and Destination IP Address.** These fields carry the 32-bit IP address of the source and final destination for this IP datagram. The use and importance of the destination address is clear.

- **Options.** The options fields allow an IP header to be extended.
The maximum amount of data that a link-layer packet can carry is called the **MTU (maximum transfer unit)**. Because each IP datagram is encapsulated within the link-layer packet for transport from one router to the next router, the MTU of the link-layer protocol places a hard limit on the length of an IP datagram. Having a hard limit on the size of an IP datagram is not much of a problem. What is a problem is that each of the links along the route between sender and destination can use different link-layer protocols, and each of these protocols can have different MTUs.

- **Modem link**: short MTU 1000 B at 9600 b/s = 530 ms too large for interactive traffic
- **large MTU** = higher throughput less overhead (TCP + IP = 40 bytes header overhead) no fragmentation loss avalanche effect
Suppose you receive an IP datagram from one link, you check your routing table to determine the outgoing link, and this outgoing link has an MTU that is smaller than the length of the IP datagram. Time to panic—how are you going to squeeze this oversized IP packet into the payload field of the link-layer packet? The solution to this problem is to "fragment" the data in the IP datagram among two or more smaller IP datagrams, and then send these smaller datagrams over the outgoing link. Each of these smaller datagrams is referred to as a fragment.

Fragments need to be reassembled before they reach the transport layer at the destination. Indeed, both TCP and UDP are expecting to receive complete, unfragmented segments from the network layer. However, Fragmentation and reassembly puts an additional burden on Internet routers and on the destination hosts. For this reason it is desirable to keep fragmentation to a minimum. This is often done by limiting the TCP and UDP segments to a relatively small size, so that fragmentation of the corresponding datagrams is unlikely.
A datagram of 4,000 bytes arrives at a router, and must be forwarded to a link with an MTU of 1,500 bytes. This implies that the 3,980 data bytes in the original datagram must be allocated to three separate fragments (each of which are also IP datagrams). Suppose that the original datagram is stamped with an identification number of $x$, all the fragmented datagrams have also ID = $x$. 
Routers answer with preference level, setup by admin.

ICMP protocol type = 9 (router advertisement) 10 (router solicitation)

sent over multicast addresses

advertisements randomized every 9 to 10 mn

host solicits 3 times 3 seconds apart

EGP is between stub/multihomed / transit

BGP is between transit nets -> supports policy routing

BGP lets all addresses of all nets be known to all BGP routers

router solicitation is for host to discover default router only
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

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

The most typical use of ICMP is for error reporting. ICMP is often considered part of IP, but architecturally lies just above IP, as ICMP messages are carried inside IP packets. That is, ICMP messages are carried as IP payload, just as TCP or UDP segments are carried as IP payload. ICMP messages have a type and a code field, and also contain the first eight bytes of the IP datagram that caused the ICMP message to be generated in the first place (so that the sender can determine the packet that caused the error). The well-known ping program sends an ICMP type 8 code 0 message to the specified host. The destination host, seeing the echo request, sends back a type 0 code 0 ICMP echo reply. Also Traceroute also uses ICMP messages.
The ICMP redirect is very useful when source and destination are directly connected. In this case, the source host receives, from the router it contacted for reaching the destination, an ICMP Redirect message that indicates that the destination is directly connected. In the TCP/IP architecture, hosts only transfer packets to connected hosts/routers. They do not have knowledge of the network and learn the minimal view of the network needed via ICMP. Routers, which performs *real* routing, need more extensive information.
ICMP Redirect Example

<table>
<thead>
<tr>
<th>dest IP addr</th>
<th>src IP addr</th>
<th>prot</th>
<th>data part</th>
</tr>
</thead>
<tbody>
<tr>
<td>128.178.29.9</td>
<td>128.178.156.24</td>
<td>udp</td>
<td>xxxxxxxx</td>
</tr>
<tr>
<td>128.178.29.9</td>
<td>128.178.156.24</td>
<td>udp</td>
<td>xxxxxxxx</td>
</tr>
<tr>
<td>128.178.156.1 128.178.156.100</td>
<td>icmp</td>
<td>type=redirect code=host cksum</td>
<td>28 bytes of 1</td>
</tr>
<tr>
<td>128.178.29.9</td>
<td>128.178.156.24</td>
<td>udp</td>
<td>..........</td>
</tr>
</tbody>
</table>
ICMP Redirect Example (cont’d)

lrcsuns:/export/home1/leboudec$ netstat -nr
Routing Table:

<table>
<thead>
<tr>
<th>Destination</th>
<th>Gateway</th>
<th>Flags</th>
<th>Ref</th>
<th>Use</th>
<th>Interface</th>
</tr>
</thead>
<tbody>
<tr>
<td>127.0.0.1</td>
<td>127.0.0.1</td>
<td>UH</td>
<td>0</td>
<td>11239</td>
<td>lo0</td>
</tr>
<tr>
<td>128.178.156.0</td>
<td>128.178.156.24</td>
<td>U</td>
<td>3</td>
<td>38896</td>
<td>le0</td>
</tr>
<tr>
<td>128.178.29.9</td>
<td>128.178.156.100</td>
<td>UGHD</td>
<td>0</td>
<td>19</td>
<td>le0</td>
</tr>
<tr>
<td>224.0.0.0</td>
<td>128.178.156.24</td>
<td>U</td>
<td>3</td>
<td>0</td>
<td>le0</td>
</tr>
<tr>
<td>default</td>
<td>128.178.156.1</td>
<td>UG</td>
<td>0</td>
<td>85883</td>
<td></td>
</tr>
</tbody>
</table>

BEFORE ICMP REDIRECT

AFTER ICMP REDIRECT

Note that ICMP adds route for a single host, not for a net!!!
For transferring packets from a sending host to the destination host a packet the network layer performs two functions: **routing** that, roughly speaking, determines the *path* or *route* that the packets are to follow and **packet forwarding**, the transmission of the packets to an address that can be reached directly. The former is generally performed between routers (by means of routing tables and data structures) and not in real-time; while packet forwarding, being a more simple action, is a real time action.

## Routing and Packet forwarding

- **Routing**
  - computation of routing tables or data structures for unicast and multicast
  - normally only between routers
  - non-real time: latency up to 2 minutes
  - uses protocols such as RIP, OSPF, EIGRP (Cisco) for unicast and DVMRP, M-OSPF, PIM for multicast

- **Packet Forwarding**
  - for every packet
  - real time
Routing

Routing protocol

Goal: determine "good" path (sequence of routers) thru network from source to dest.

Graph abstraction for routing algorithms:
- graph nodes are routers
- graph edges are physical links
  - link cost: delay, $ cost, or congestion level

"good" path:
- typically means minimum cost path
- other def's possible

Given a set of routers, with links connecting the routers, a **routing algorithm** finds a "good" path from source to destination. Typically, a "good" path is one that has "least cost." With the graph abstraction for routing algorithms, nodes in the graph represent routers--the points at which packet routing decisions are made--and the lines ("edges" in graph theory terminology) connecting these nodes represent the physical links between these routers. A link also has a value representing the "cost" of sending a packet across the link. The cost may reflect the level of congestion on that link or the physical distance traversed by that link.
Routing Algorithm classification

<table>
<thead>
<tr>
<th>Global or decentralized information?</th>
<th>Static or dynamic?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Global:</td>
<td>Static:</td>
</tr>
<tr>
<td>□ all routers have complete topology, link cost info</td>
<td>□ routes change slowly over time</td>
</tr>
<tr>
<td>□ &quot;link state&quot; algorithms</td>
<td>□ routes change more quickly</td>
</tr>
<tr>
<td>Decentralized:</td>
<td>□ periodic update</td>
</tr>
<tr>
<td>□ router knows physically-connected neighbors, link costs to neighbors</td>
<td>□ in response to link cost changes</td>
</tr>
<tr>
<td>□ iterative process of computation, exchange of info with neighbors</td>
<td></td>
</tr>
<tr>
<td>□ &quot;distance vector&quot; algorithms</td>
<td></td>
</tr>
</tbody>
</table>

A global routing algorithm computes the least-cost path between a source and destination using complete, global knowledge about the network. That is, the algorithm takes the connectivity between all nodes and all link costs as inputs. Example: link state algorithms.

In a decentralized routing algorithm, the calculation of the least-cost path is carried out in an iterative, distributed manner. No node has complete information about the costs of all network links. Instead, each node begins with only the knowledge of the costs of its own directly attached links. Then, through an iterative process of calculation and exchange of information with its neighboring nodes (that is, nodes that are at the "other end" of links to which it itself is attached), a node gradually calculates the least-cost path to a destination or set of destinations. Example: distance vector algorithms.

A second broad way to classify routing algorithms is according to whether they are static or dynamic. In static routing algorithms, routes change very slowly over time, often as a result of human intervention. Dynamic routing algorithms change the routing paths as the network traffic loads or topology change. A dynamic algorithm can be run either periodically or in direct response to topology or link cost changes. While dynamic algorithms are more responsive to network changes, they are also more susceptible to problems such as routing loops and oscillation in routes.

Only two types of routing algorithms are typically used in the Internet: a dynamic global link state algorithm, and a dynamic decentralized distance vector algorithm.
Routing in the Internet

- The Global Internet consists of Autonomous Systems (AS) interconnected with each other:
  - Stub AS: small corporation
  - Multihomed AS: large corporation (no transit)
  - Transit AS: provider

- Two-level routing:
  - Intra-AS: administrator is responsible for choice
  - Inter-AS: unique standard

Routers are aggregate into regions or autonomous systems (ASs). Routers within the same AS all run the same routing algorithm (for example, an LS or DV algorithm) and have information about each other—exactly as was the case in our idealized model in the previous section. The routing algorithm running within an autonomous system is called an intraautonomous system routing protocol. It will be necessary, of course, to connect ASs to each other, and thus one or more of the routers in an AS will have the added task of being responsible for routing packets to destinations outside the AS. Routers in an AS that have the responsibility of routing packets to destinations outside the AS are called gateway routers. In order for gateway routers to route packets from one AS to another (possibly passing through multiple other ASs before reaching the destination AS), the gateways must know how to route (that is, determine routing paths) among themselves. The routing algorithm that gateways use to route among the various ASs is known as an interautonomous system routing protocol.
An intra-AS routing protocol is used to configure and maintain the routing tables within an autonomous system (AS). Intra-AS routing protocols are also known as **interior gateway protocols**.
Intra-AS and inter-AS Routing

Intra-AS routing:
- Also known as Interior Gateway Protocols (IGP)
- Most common IGPs:
  - RIP: Routing Information Protocol
  - OSPF: Open Shortest Path First
  - EIGRP: Extended Interior Gateway Routing Protocol (Cisco propr.)

Inter-AS routing:
- Also known as Exterior Gateway Protocols (EGP)
- BGP (Border Gateway Protocol): the de facto standard

Why are there Different Inter-AS and Intra-AS Routing Protocols?
- Policy
- Scale
- Performance

Historically, three routing protocols have been used extensively for routing within an autonomous system in the Internet: RIP (the Routing Information Protocol), OSPF (Open Shortest Path First), and EIGRP (Cisco's propriety Enhanced Interior Gateway Routing Protocol).

The Border Gateway Protocol version 4, specified in RFC 1771 (see also RFC 1772; RFC 1773), is the de facto standard interdomain routing protocol in today's Internet. It is commonly referred to as BGP4 or simply as BGP. As an inter-autonomous system routing protocol, it provides for routing between autonomous systems (that is, administrative domains).

There are several reasons for having different Inter-AS and Intra-AS routing protocols:

• **Policy.** Among ASs, policy issues dominate. It may well be important that traffic originating in a given AS specifically not be able to pass through another specific AS. Similarly, a given AS may well want to control what transit traffic it carries between other ASs.

• **Scale.** The ability of a routing algorithm and its data structures to scale to handle routing to/among large numbers of networks is a critical issue in inter-AS routing. Within an AS, scalability is less of a concern.

• **Performance.** Because inter-AS routing is so policy-oriented, the quality (for example, performance) of the routes used is often of secondary concern. Indeed, among ASs, there is not even the notion of preference or costs associated with routes.
If you ever read commercial literature, you have to be aware of the difference between architecture names and implementation names. The word **router** (like most words) is unfortunately used in both contexts.

- From an architecture viewpoint, a router is any system which forwards packets based on layer 3 information. Router in that context is a function.

- The router function can be implemented by a piece of software on Unix or Windows NT, or by a complex dedicated machine (a Cisco, IBM, Bay Networks or Flextel box for example).

Most boxes called “routers” perform a set of additional functions that have nothing to do with packet forwarding using layer 3 addresses. For example, they can be used as bridges or application level relay.
Router Architecture Overview

Two key router functions:
- run routing algorithms/protocol (RIP, OSPF, BGP)
- switching datagrams from incoming to outgoing link

The routing algorithms control the routes taken by packets through the network. In the network layer, the real work is the forwarding of datagrams, at first, between an incoming link and an outgoing link. The switching function of a router is the transfer of datagrams from a router's incoming links to the appropriate outgoing links. The input port performs, among other functions, the physical layer functionality of terminating an incoming physical link to a router; and the data-link layer functionality needed to interoperate with the data link layer functionality on the other side of the incoming link. The switching fabric connects the router's input ports to its output ports. This switching fabric is completely contained within the router. The output port stores the packets that have been forwarded to it through the switching fabric, and then transmits the packets on the outgoing link. The output port thus performs the reverse data link and physical layer functionality as the input port. The routing processor executes the routing protocols, maintains the routing tables, and performs network management functions, within the router.
B (an old macintosh file server) runs only Appletalk. Only applications using the Appletalk protocols can be used (MacOS file sharing, printing). TCP/IP applications such as the web cannot be used on B.

C (a modern PC) runs only TCP/IP. All TCP/IP applications can be used, but not MacOS file sharing.

A (a windows NT server) runs both in parallel. It can talk to both C and B.

A bridge can be used to interconnect A, B and C; there is nothing special to do. If a router is used instead, it must run in parallel Appletalk and IP.

The protocol stacks shown are all implemented in software. They use the standard Ethernet adapters.
NetBIOS is an interface for distributed applications which is commonly used with IBM and Microsoft systems. Originally, NetBIOS used the LLC-2 protocol, a link layer protocol which does packet retransmissions, much as TCP does. Only MAC addresses are used. In addition, NetBIOS offers a naming service. This version of NetBIOS works only in a bridged environment.
IPv6

- The current IP is IPv4
- IPv4 address space is too small (32 bits)
  - will be exhausted some day
- IPv6 is the new version of IP
  - addresses are 128 bit longs
  - RFC 2460

- IPv6 is incompatible with IPv4

IPv6 is primarily IP with a larger address space. However, a number of details are different, in particular the IPv6 header is easier to process (but is also longer). An excellent online source of information about IPv6 is The IP Next Generation Homepage [Hinden 1999].

Many features which were originally designed for IPv6 are now part of IPv4 (security and mobility).

The most important changes introduced in IPv6 are evident in the packet format:

- *Expanded addressing capabilities.* IPv6 increases the size of the IP address from 32 to 128 bits. This ensures that the world won't run out of IP addresses. In addition to unicast and multicast addresses, a new type of address, called an **anycast address,** has also been introduced, which allows a packet addressed to an anycast address to be delivered to any one of a group of hosts.

- *A streamlined 40-byte header.* A number of IPv4 fields have been dropped or made optional. The resulting 40-byte fixed-length header allows for faster processing of the IP datagram. A new encoding of options allows for more flexible options processing.

- *Flow labeling and priority.* IPv6 has an elusive definition of a "flow." RFC 1752 and RFC 2460 state that this allows "labeling of packets belonging to particular flows for which the sender requests special handling, such as a non-default quality of service or real-time service."

IPv6 is incompatible with IPv4; this is to avoid the IBM’s SNA syndrom (a monster of complexity,, because the last version is compatible with all details of all previous versions). Interworking between the two will use the dual stack approach, as shown for interworking between Appletalk and IP.
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: routers with both v6, v4 "translate" between formats
Transition From IPv4 To IPv6

- Two proposed approaches:
  - Tunneling: IPv6 carried as payload in IPv4 datagram among IPv4 routers
Plug and Play and DHCP

- IPv6 address is allocated automatically by negotiation with routers
  - "stateless allocation"
- alternatively, Dynamic Host Configuration Protocol (DHCP) can be used
- DHCP can be used with IPv4 also
  - DHCP server on LAN has a list of IP addresses that can be allocated dynamically
  - MAC address used to identify a host to DHCP server
  - renumbering is possible
  - more complex to use than IPv6 stateless allocation

With IPv6 an host can negotiate and get its IP address directly from the router which is attached to. As alternative the Dynamic Host Configuration Protocol (DHCP) [RFC 2131], also available for IPv4, and used for MobileIP, can be used.

DHCP is sometimes referred to as Plug and Play. With DHCP, a DHCP server in a network (for example, in a LAN) receives DHCP requests from a client and, in the case of dynamic address allocation, allocates an IP address back to the requesting client. DHCP is used extensively in LANs and in residential Internet access.
A number of emerging network applications require the delivery of packets from one or more senders to a group of receivers. For each of these applications, an extremely useful abstraction is the notion of a multicast: the sending of a packet from one sender to multiple receivers with a single send operation. Clearly, this second approach toward multicast makes more efficient use of network bandwidth in that only a single copy of a datagram will ever traverse a link. On the other hand, considerable network layer support is needed to implement a multicast-aware network layer. Internet multicast is not a connectionless service—state information for a multicast connection must be established and maintained in routers that handle multicast packets sent among hosts in a so-called multicast group. This, in turn, will require a combination of signaling and routing protocols in order to set up, maintain, and tear down connection state in the routers. In the Internet architecture (and the ATM architecture as well), a multicast datagram is addressed using address indirection. That is, a single identifier is used for the group of receivers, and a copy of the datagram that is addressed to the group using this single identifier is delivered to all of the multicast receivers associated with that group. In the Internet, the single identifier that represents a group of receivers is a Class D multicast address. The group of receivers associated with a class D address is referred to as a multicast group. Multicast addresses are not allocated on a geographical basis. A global allocation scheme is under discussion at the IETF. Today, global scope addresses are allocated using the sd tool on Unix. Note that the unique IP unicast address of an host is completely independent of the address of the multicast group in which it is participating.
1 S sends data to multicast address m; there is no member, the data is simply lost at the router
2 A joins the multicast address m
3 R1 informs the rest of the network that m has a member at R1; the multicast routing protocol builds a tree. Data sent by S now reach A
4 B joins the multicast address m
5 R4 informs the rest of the network that m has a member at R4; the multicast routing protocol adds branches to the tree. Data sent by S now reach both A and B
The mapping IP to MAC for multicast addresses is not unique. Ethernet hosts must filter up to 32 IP addresses for one MAC multicast address.
IGMP: Internet Group Management Protocol

Purpose: manage group membership inside one subnet
- routers: know if group is present on an interface
  - know whether to forward locally or not
- hosts: know if a multicast address is already in use locally

1: IGMP query, TTL = 1, IGMP group @ = 0
   dest IP@ = 224.0.0.1; source IP@ = 128.178.156.1
2: IGMP report, TTL = 1, IGMP group @ = 224.2.166.207
   dest IP@ = 224.2.166.207; source IP@ = 128.178.156.24
3: IGMP report, TTL = 1, IGMP group @ = 224.2.127.255
   dest IP@ = 224.2.127.255; source IP@ = 128.178.156.24

The Internet Group Management Protocol, IGMP version 2 [RFC 2236], operates between a host and its directly attached router (e.g. S with R1, A with R5, etc.). It provides the means for a host to inform its attached router that an application running on the host wants to join a specific multicast group. TTL is in order to avoid broadcast propagation.
(1): a first response is sent spontaneously, a short timer (10s) set, then another response sent after expiration (because of possible loss)

(2): a random timer is chosen
The Multicast Backbone (MBONE) is a virtual network. It is layered on top of portions of the physical Internet to support routing of IP multicast packets since that function has not yet been integrated into many production routers. The network is composed of islands that can directly support IP multicast, such as multicast LANs like Ethernet, linked by virtual point-to-point links called "tunnels". The tunnel endpoints are typically workstation-class machines having operating system support for IP multicast and running the "mrouted" multicast routing daemon. IP multicast packets are encapsulated for transmission through tunnels, so that they look like normal unicast datagrams to intervening routers and subnets. A multicast router that wants to send a multicast packet across a tunnel will prepend another IP header, set the destination address in the new header to be the unicast address of the multicast router at the other end of the tunnel, and set the IP protocol field in the new header to be 4 (which means the next protocol is IP). The multicast router at the other end of the tunnel receives the packet, strips off the encapsulating IP header, and forwards the packet as appropriate. Each tunnel has an associated threshold against which the packet's IP time-to-live (TTL) value is compared. Currently, all MBONE applications use UDP.

Distance Vector Multicast Routing Protocol (DVMRP) allows each router to compute the outgoing link (next hop) that is on its shortest path back to each possible source. With the reverse path forwarding (RPF) algorithm, when a router receives a multicast packet with a given source address, it transmits the packet on all of its outgoing links (except the one on which it was received) only if the packet arrived on the link that is on its own shortest path back to the sender. Otherwise, the router simply discards the incoming packet without forwarding it on any of its outgoing links.
Suppose the host 128.178.156.24 wants to communicate with the Mbone multicast group. It sends (with multicast routing) the packet inside its local area (EPFL). When the multicast packet (UDP) reaches the Mbone-connected router (128.178.100.67) it is encapsulated within an IP packet and sent it within the Internet to the other IP multicast routers connected with tunnels to this MBone.multicast group. For example, it reaches the router 129.132.2.89, which has a participant to the MBone.multicast group (129.132.99.67).

At EPFL, IP multicast is supported as follows:(1) inside EPFL, CISCO routers support Multicast IP; (2) scoping is by use of TTL:

TTL < 8: inside LRC
TTL < 16: inside DI
TTL < 32 inside Ecublens
TTL < 64: inside EPFL
TTL = 127: world wide

The routing inside EPFL is done with PIM, outside via Mbone

-to know more about multicast ingeneral: http://www.ipmulticast.com
-at EPFL http://stiwww.epfl.ch/mbone/mbone.html
The multicast datagram socket class is useful for sending and receiving IP multicast packets. A MulticastSocket is a (UDP) DatagramSocket, with additional capabilities for joining "groups" of other multicast hosts on the internet.

One would join a multicast group by first creating a MulticastSocket with the desired port, 224.0.0.1 to 239.255.255.255, inclusive, then invoking the `joinGroup(InetAddress groupAddr)` method:

When one sends a message to a multicast group, all subscribing recipients to that host and port receive the message (within the time-to-live range of the packet). The socket need not be a member of the multicast group to send messages to it.

When a socket subscribes to a multicast group/port, it receives datagrams sent by other hosts to the group/port, as do all other members of the group and port. A socket relinquishes membership in a group by the `leaveGroup(InetAddress addr)` method. Multiple MulticastSocket's may subscribe to a multicast group and port concurrently, and they will all receive group datagrams.
Constructors and main methods

- **public MulticastSocket()** throws IOException
- **public MulticastSocket(int port)** throws IOException
- **public void joinGroup(InetAddress mcastaddr)** throws IOException
- **public void leaveGroup(InetAddress mcastaddr)** throws IOException
- **public void send(DatagramPacket p, byte ttl)** throws IOException
- **public void setTimeToLive(int ttl)** throws IOException

**CONSTRUCTORS**

**MulticastSocket()** throws IOException
Create a multicast socket.

**MulticastSocket(int port)** throws IOException
Create a multicast socket and bind it to a specific port.

**MAIN METHODS**

**public void joinGroup(InetAddress mcastaddr)** throws IOException
Joins a multicast group. Its behavior may be affected by setInterface. If there is a security manager, this method first calls its checkMulticast method with the mcastaddr argument as its argument.

**public void leaveGroup(InetAddress mcastaddr)** throws IOException
Leave a multicast group. Its behavior may be affected by setInterface. If there is a security manager, this method first calls its checkMulticast method with the mcastaddr argument as its argument.

**public void send(DatagramPacket p, byte ttl)** throws IOException
Sends a datagram packet to the destination, with a TTL (time-to-live) other than the default for the socket. This method need only be used in instances where a particular TTL is desired; otherwise it is preferable to set a TTL once on the socket, and use that default TTL for all packets.

**public void setTimeToLive(int ttl)** throws IOException
Set the default time-to-live for multicast packets sent out on this socket. The TTL sets the IP time-to-live for DatagramPackets sent to a MulticastGroup,
### Example

```java
......
byte[] msg = {'H', 'e', 'l', 'l', 'o'};
InetAddress group = InetAddress.getByName("228.5.6.7");
MulticastSocket s = new MulticastSocket(6789);
s.joinGroup(group);
DatagramPacket hi = new DatagramPacket(msg, msg.length, group, 6789);
s.send(hi);
// get their responses!
byte[] buf = new byte[1000];
DatagramPacket recv = new DatagramPacket(buf, buf.length);
s.receive(recv);
// OK, I'm done talking - leave the group...
s.leaveGroup(group);
......
```

In the example we see how to join a Multicast group and send the group salutations. The MulticastSocket `s`, that is a DatagramSocket, uses the DatagramPacket for building the datagram, which is then sent with the send() method.
Summary

- The network layer transports packets from a sending host to the receiver host.
- Main components:
  - addressing
  - routing
  - routers (and how a router works)
- advanced topics: IPv6, multicast
- the Internet network layer
  - Connectionless
  - Best-effort