### Network Layer

- It provides facilities for getting data from a source to a destination.
- This may require making many hops at intermediate routers along the way (routing).
  - Data link layer moves frames only from one end of the wire to the other.
- It must know the topology of the network and choose appropriate paths.
- Load-balancing among routes is required.
- Finally, it has to deal with network heterogeneity.

### Network Layer

- Again, distinguish between connection-less and connection-oriented services.
- Connection-oriented:
  - Modeled after the telephone system.
  - It acts like a tube: the sender pushes objects (bits) in at one end, and the receiver takes them out at the other end.
- Connection-less:
  - Modeled after the postal system.
  - Each message (letter) is routed through the system independent of all the others.

- There are two implementation techniques:
  - Virtual circuits are complete routes that are set up in advance.
  - Datagrams comprise individual packets of which the route is determined on the fly: they hop from router to router.

- **Note:** The services are independent of their implementation:

```
<table>
<thead>
<tr>
<th>Protocol</th>
<th>Implementation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCP over IP</td>
<td>Virtual circuit</td>
</tr>
<tr>
<td>UDP over IP</td>
<td>Datagram</td>
</tr>
</tbody>
</table>
```

### Virtual Circuits

**Example**

![Virtual Circuit Diagram]

<table>
<thead>
<tr>
<th>H1: 1</th>
<th>C1</th>
<th>A1</th>
<th>E1</th>
<th>C1</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>H2: 1</td>
<td>C2</td>
<td>A2</td>
<td>E2</td>
<td>C2</td>
<td>F2</td>
</tr>
<tr>
<td>In</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Datagrams

**Example**

```
<table>
<thead>
<tr>
<th>Protocol</th>
<th>Destination Address</th>
<th>Source Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
<td>C</td>
</tr>
<tr>
<td>D</td>
<td>E</td>
<td>F</td>
</tr>
</tbody>
</table>
```

---

**Comparison**

<table>
<thead>
<tr>
<th>Issue</th>
<th>Datagram</th>
<th>Virtual circuit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Circuit setup</td>
<td>Not needed</td>
<td>Required</td>
</tr>
<tr>
<td>Addressing</td>
<td>Each packet contains the full source and destination address</td>
<td>Each packet contains a short VC number</td>
</tr>
<tr>
<td>State info</td>
<td>Routers do not hold state information about connections</td>
<td>Each VC requires router table space per connection</td>
</tr>
<tr>
<td>Routing</td>
<td>Each packet is routed independently</td>
<td>Route chosen when VC is set up; all packets follow it</td>
</tr>
<tr>
<td>Effect of router failures</td>
<td>None, except for packets lost during the crash</td>
<td>All VCs that passed through the failed router are terminated</td>
</tr>
<tr>
<td>QoS</td>
<td>Difficult</td>
<td>Easy if enough resources can be allocated in advance for each VC</td>
</tr>
<tr>
<td>Congestion control</td>
<td>Difficult</td>
<td>Easy if enough resources can be allocated in advance for each VC</td>
</tr>
</tbody>
</table>

---

**Routing**

- Routers that constitute the network layer of a network, should cooperate to find the **best routes** between all pairs of stations
  - All optimal routes from station A to other stations in the network, jointly constitute a **sink tree**.

```
<table>
<thead>
<tr>
<th>Router 1</th>
<th>Router 2</th>
<th>Router 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
<td>C</td>
</tr>
<tr>
<td>D</td>
<td>E</td>
<td>F</td>
</tr>
</tbody>
</table>
```

- Routers have to collaborate to build the sink tree (or something that comes near to that) for each source station.

---

**Shortest Path Routing**

- **Dijkstra's Algorithm**
  - Labels on the arcs represent the cost (e.g., distance, delay)
  - During each step, select a newly reachable node at the lowest cost, and add the edge to that node, to the tree built so far.
  - Linear cost: \(O(n^2)\)

```
<table>
<thead>
<tr>
<th>Step 1</th>
<th>Step 2</th>
<th>Step 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
<td>C</td>
</tr>
<tr>
<td>D</td>
<td>E</td>
<td>F</td>
</tr>
</tbody>
</table>
```

---

Notes
Flooding

- Forward an incoming packet across every outgoing line, except the one it came in through.
- **Problem:** how to avoid “drowning by packets”?
  1. Use a hop counter:
     - after a packet has been forwarded across N routers, it is discarded
     - gotta find the right hop count, though.
  2. Be sure to forward a packet only once (i.e. avoid directed cycles).
     - requires sequence numbers per source router.
     - each router keeps track of the last sequence number per source router.
  3. Flood selectively: only in the direction that makes sense.
- Flooding always chooses the **shortest path** because all path are explored in parallel
- the overhead however grows significantly
- Flooding makes sense only when **robustness** is needed
- Also useful in **wireless networks**:
  - a message transmitted by a station is received by all other stations in range

Distance Vector Routing

- The previous protocols are **static**
  - they do not take the current network load into account
- Let’s now discuss dynamic protocols
- **Distance Vector Routing (DVR) (a.k.a. Bellman-Ford)**
  1. look at the costs that your direct neighbors are advertising to get a packet to the destination.
  2. select the neighbor whose advertised cost, added with the cost to get to that neighbor, is the lowest.
  3. advertise that new cost to the other neighbors.
- **Example:** given the following neighbors’ tables related to a given node A, which is the preferred neighbor to forward packets for A?
  
<table>
<thead>
<tr>
<th>Neighbor:</th>
<th>R1</th>
<th>R2</th>
<th>R3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Link cost:</td>
<td>12</td>
<td>8</td>
<td>5</td>
</tr>
<tr>
<td>Advertised:</td>
<td>28</td>
<td>25</td>
<td>39</td>
</tr>
<tr>
<td>Total:</td>
<td>40</td>
<td>33</td>
<td>44</td>
</tr>
</tbody>
</table>

- **R2** is the preferred neighbor
- **R3** will change its route as well!

```
⇒ routing through the current node and **R2** is cheaper (33 + 5 < 39)
```

**Notes**

- DVR is pretty expensive: it turns out to be $O(n^3)$
- too high for a fast convergence.

Distance Vector Routing

**Count-to-infinity**

(a) Good news propagate really fast
- A has been down for a while and now comes up
(b) Bad news are slow
- **Count-to-infinity Problem**
  - A suddenly goes down again
  - all routers increase their route up to infinity
  - usually set infinity to the longest path plus 1
  - more sophisticated solutions (e.g., poisoned reverse) are available too
Distance vector routing was used in the ARPANET until 1979, when it was replaced by Link State Routing.

Broadcast info on the entire network topology to all routers, and let each of them calculate a sink tree to the other routers.

What a router needs to do (next to just routing):
1. Find out who its neighbors are and get their network addresses.
2. Calculate the cost for getting a packet to a neighbor.
3. Construct a link state advertisement telling all it has just learned.
4. Send that packet to all other routers (not just neighbors).
5. Run Dijkstra locally.

You know your network interface(s), so just send an HELLO packet through each of them.

the router on the other end will send back a reply with its address.

What happens when you're on a broadcast LAN?

One way to model the LAN is to consider it as a node itself.

the fact that it is possible to go from A to C on the LAN is represented by the path ANC here.

Just send an ECHO packet through each interface, and measure the round-trip delay that will give you a reasonable estimate of the actual delay.

Problem: do we take the local load into account, or not (i.e., measure from the moment you queue the packet or insert it into the network)?

PROS:
- you can choose for momentarily better routes
- e.g., if it’s difficult to reach the highway from the inner city, trains will be justifiably chosen as the best route.

CONS:
- You may redirect traffic in such a way that the alternative route becomes overloaded
- e.g., advise everyone to take the train, causing the trains to become overloaded, while the access path to the highway is now underloaded (and thus forming a better alternative).

Every router uses flooding to sends its LSA to all other routers.

Once we have all the LSAs from every router, and therefore we have complete knowledge of the network, we run Dijkstra locally.

LSA_a = \{(a, b, 3), (a, e, 1), (a, d, 1)\}
LSA_b = \{(h, e, 1), (h, f, 4), (h, j, 14)\}
LSA_c = \{(d, a, 1), (d, g, 1), (d, e, 3)\}
LSA_d = \{(f, c, 1), (f, b, 1), (f, e, 3), (f, h, 4), (f, j, 2)\}
Comparison

<table>
<thead>
<tr>
<th></th>
<th>Distance Vector Routing</th>
<th>Link State Routing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Network Knowledge</td>
<td>local</td>
<td>global</td>
</tr>
<tr>
<td>Computation</td>
<td>global</td>
<td>local</td>
</tr>
<tr>
<td>Interaction</td>
<td>Asynchronous</td>
<td>Synchronous</td>
</tr>
</tbody>
</table>

Hierarchical Routing

- **Problem:** No routing algorithm discussed so far can scale all of them require each router to know about all others ⇒ too demanding with respect to memory and processing power.
- **Solution:** Go for suboptimal routes by introducing regions, and separate algorithms for intra-region and inter-region routing. two or three levels will generally do.

- **No optimal routes any more (e.g., 1A to 5C)**

Broadcast Routing

**Problem:** We want to send a message to (almost) every host on the network. In practice, this means we’re talking about (interconnected) LANs, or (relatively small) WANs.

1. Just send the message to each host individually
   - not really good.
2. Use flooding
   - acceptable provided that we can dam the flood.
3. Use multidestination routing, by means of a bitmap or list that is sent along with the packet
   - a router checks the destinations, and splits the list when forwarding it across different output lines.
   ⇒ the message must contain all the destinations
4. Build a sink tree at the source and use that as your multicast route.
   - the sink tree has to be a spanning tree (as in Dijkstra)
   ⇒ the routers need to know the trees
Broadcast Routing

Reverse Path Forwarding

- **Problem:** Suppose we don’t know every spanning tree. How can we construct one at low cost?
- **Solution:** We exploit information available at each router to build a spanning tree.

- **Reverse-path broadcast**
  - every router forwards a broadcast packet to every adjacent router, except the one where it received the packet router
  - a router \( u \) accepts a broadcast packet \( p \) originating at router \( s \) only if \( p \) arrives on the link that is on the direct (unicast) path from \( u \) to \( s \)

**Question**

- Could this protocol be used to disseminate Link-State Advertisements?
  - No, because it requires (unicast) routing information

Multicast Routing

- **Problem:** But suppose that we want to send a message to only a subset of all the nodes in a network.
- **Solution:** Construct a spanning tree (at each router) for the entire network.
  - use the group-id to prune paths to nodes that do not contain members for that group.

**Notes**

- What is the main issue with the just presented solution?
  - Scalability! We need to maintain a spanning tree per each source.

- An alternative design uses core-based trees
  - a single spanning tree per group is computed, with the root (the core) near the middle of the group
  - to send a multicast message, a host sends it to the core, which then does the multicast along the tree.

Routing for Mobile Hosts

- **Problem:** How can we forward messages to things that are constantly on the move, preferably in a wide-area context?
  - the issue is not moving hosts from LAN to LAN, but finding the mobile computers.

- **Solution:** use home and foreign agents
  1. the mobile hosts register with the foreign agents
  2. the foreign agent contacts the mobile host’s home agent and says: “One of your hosts is over here”
  3. when a packet is sent to a mobile host, it is routed to the host’s home LAN
  4. the home agent then forward the message to the foreign agent

- Similar protocol also used in GSM to track mobile users.

**Notes**
Routing for Mobile Hosts

Example

- **Tunneling**: sending an IP packet in an IP packet
  - that’s the way to keep the routers ignorant of the fact that they’re routing something else.
- **Adapt routers**: when you send the new address back to the source, intermediate routers can adapt their tables.

Mobile Ad Hoc Networks (MANET)

- **Mobile Ad Hoc Networks (MANET)**
  - not only are the hosts mobile; the routers are mobile as well
  - often necessary where there is no fixed infrastructure.
- **Example networks**:
  - A fleet of ships, military vehicles in a battlefield, etc.
  - Spontaneous networks of PDAs, notebooks, digital phones and other mobile devices
  - Wireless mesh networks created as an alternative to wired networks
- **Problem**:
  - we may be faced with a dynamically changing topology
  - nothing is fixed anymore.
- **Traditional protocols are out of question**
  - they would incur too much overhead
  => **Ad Hoc Distance Vector (AODV)**
  - on-demand protocols: routes are set up only when required

Mobile Ad Hoc Networks (MANET)

- **Ad Hoc Distance Vector**
  - Represent the network as a graph in which any two nodes are connected only if they can communicate directly with each other.
  - **A** wants to send a message to **I**
    1. it broadcasts a **route request** (which will reach only **B** and **D**) if **A** does not have a route to **I**
    2. if an intermediate node that doesn’t know how to get to **I**, increment the request’s hop counter, and broadcast again
    3. eventually, request will arrive at **I**
    4. Several route requests for **A** --> **I** may have arrived. The one with the lowest hop counter indicates the shortest path.
    5. Send a **route reply** back to the neighbor that had the lowest hop count

Mobile Ad Hoc Networks (MANET)

- **Route Maintenance**
  - The above protocol enables intermediates nodes to update their routes both to source and destination
  - Due to mobility, however, routes become obsolete and need to be refreshed
  - Example:
    - assume **G** leaves the network.
    - **D** will discover that the routes it had registered for **E**, **G**, and **I** are no longer valid
    - hence, it will inform **A** and **B** that they need to update their tables.
Congestion Control

- Problem: when too many packets have to be transmitted through the network, we can get into a serious performance problem:
  - Congestion can be caused by lack of bandwidth, but also by ill-configured or slow routers
  - Flow Control relates to the point-to-point traffic between sender and receiver
  - Open loop and closed loop solutions:
    - is feedback provided (closed) or not (open) ?
  - Prevention: avoid congestion by avoiding bursts:
    - shape your packet traffic, and let the network provider do the traffic policing (monitor the flow).

Flow Control relates to the point-to-point traffic between sender and receiver

Congestion Control involves all hosts

Open loop and closed loop solutions: is feedback provided (closed) or not (open) ?

Prevention: avoid congestion by avoiding bursts:
- shape your packet traffic, and let the network provider do the traffic policing (monitor the flow).

Paolo Costa

05 - Network Layer

Congestion 29 / 112

Virtual Circuits

When you set up a circuit, be sure that congestion can be avoided.
- Admission control: if it's too busy, just refuse to set up a virtual circuit. This is the same as refusing new users at an FTP site.
- Select alternative routes when a part in the network is getting overloaded (i.e., temporarily rebuild your view of the network):
  - Negotiate the quality of the circuit in advance
    - the network provider can reserve buffers and the like.
    - Resources are guaranteed to be there.

Paolo Costa

05 - Network Layer

Congestion 30 / 112

Datagrams

Choke packets:
- send info back to the source when the networks starts to perform bad
- a router checks the status of an output line:
  - if it's too occupied, it sends a choke packet to the source.
- the host is assumed to be cooperative, and that it will slow down
  - not always a good assumption
- Problem: The return path for a choke packet may be so long, that the destination is going to get into trouble anyway
  - there may simply be too much on the way already
  - then use a "push-back" approach

Paolo Costa

05 - Network Layer

Congestion 31 / 112

Choke Packets

Example

(a) A choke packets affecting only the source
(b) A choke packet affecting each hop

Paolo Costa

05 - Network Layer

Congestion 32 / 112

Notes

Notes
Quality of Service

The techniques we looked thus far are designed to reduce congestion and improve network performance. Now we focus on ways to provide a quality of service matched to application needs.

A stream of packets from source to destination is called a flow.

Quality of Service (QoS): The needs of each flow are determined by reliability, delay, jitter, and bandwidth.

<table>
<thead>
<tr>
<th>Application</th>
<th>Reliability</th>
<th>Delay</th>
<th>Jitter</th>
<th>Bandw.</th>
</tr>
</thead>
<tbody>
<tr>
<td>E-mail</td>
<td>high</td>
<td>low</td>
<td>low</td>
<td>low</td>
</tr>
<tr>
<td>File transfer</td>
<td>high</td>
<td>low</td>
<td>low</td>
<td>medium</td>
</tr>
<tr>
<td>Web access</td>
<td>high</td>
<td>medium</td>
<td>low</td>
<td>medium</td>
</tr>
<tr>
<td>Remote login</td>
<td>high</td>
<td>medium</td>
<td>medium</td>
<td>low</td>
</tr>
<tr>
<td>Audio on demand</td>
<td>low</td>
<td>low</td>
<td>high</td>
<td>medium</td>
</tr>
<tr>
<td>Video on demand</td>
<td>low</td>
<td>low</td>
<td>high</td>
<td>high</td>
</tr>
<tr>
<td>Telephony</td>
<td>low</td>
<td>high</td>
<td>high</td>
<td>low</td>
</tr>
<tr>
<td>Videoconferencing</td>
<td>low</td>
<td>high</td>
<td>high</td>
<td>high</td>
</tr>
</tbody>
</table>

Jitter

Jitter is the variation in the arrival times of packets.

Critical for multimedia applications

Having some packets taking 20 msec and others taking 30 msec to arrive will give an uneven quality to the sound or movie.

Techniques for Good QoS

Overprovisioning
- e.g., the telephone system (you always get a dial tone)
- the best approach but expensive

Resource reservation and Admission control
- can be applied only in virtual circuit networks

Buffering to reduce jitter

Traffic shaping and policing

Packet scheduling

QoS: Buffering

Just try to reduce jitter as much as possible by buffering incoming packets at the receiver before passing them to the application:
QoS: Traffic Shaping

Leaky Bucket

- We’ve eliminated bursts completely
  - packets are passed to the network when available, and all at the same rate
- This may be a bit overdone.
  - also, packets can get lost.

Token Bucket

- Tokens are added at a constant rate; as soon as enough tokens have been saved, one or more packets can be sent.
- To avoid still too much burstiness, put a leaky bucket behind a token bucket (with a larger rate).

QoS: Effects of Buckets

(a) Input to a leaky bucket
(b) Output from a leaky bucket
(c) Output from a token bucket with capacities of 250 KB
(d) Output from a token bucket with capacities of 500 KB
(e) Output from a token bucket with capacities of 750 KB
(f) Output from a 500KB token bucket feeding a 10-MB/sec leaky bucket

QoS: Packet Scheduling

- Problem: What happens when you need to support multiple flows and one of them is too resource-consuming?
  - To enforce cooperation, use weighted fair queuing:
    - routers have separate queues for each output line
    - when a line becomes idle, the router scans the queues round robin, taking the first packet on the next queue
    - with $n$ hosts competing for a given output line, each host gets to send one out of every $n$ packets

- To improve the fairness, byte-by-byte round-robin is used instead of packet-by-packet
  - the packets are then sorted in order of their finishing and sent in that order
Integrated Services

Problem: 

- In order to efficiently support streams, we cannot set up a single connection per stream, but need to integrate things.
- This is particularly the case with multicast applications.
- Large and frequently changing group of receivers.

Resource Reservation Protocol (RSVP)

- We set up multicast trees from sources to destinations, but this time take into account the bandwidth needed by the receivers.
- Bandwidth can be reserved on pre-constructed trees.
- If there is not enough bandwidth as required by the receiver, a failure is reported back.

RSVP Example

![Diagram of RSVP example]

Differentiated Services

- Problem: Integrated services require connection setup.
  - Scalability issues with thousands of flows
  - Large per-flow state on routers (vulnerable to crashes)
- Solution: Offer a means for local QoS decision-making.
  - Differentiated services (for routers in one administrative domain).
- For each class of applications, reserve resources.
  - E.g., let routers differentiate regular from expedited traffic.
  - Packets belonging to either class will be marked as such.

Differentiated Services Assured Forwarding

- Distinguish four priority classes and three discard probabilities in case you’re also shaping traffic.
  - This leads to 12 combinations.
- An 8-bit *Type of service* field is available in the IP header.
  - The shaper/dropper may delay some, e.g., using leaky or token buckets, or discard them.
Label Switching

Instead of having standard routers provide QoS on datagram routing, let them try to establish connections.

Add a connection-id to datagrams and let routers take that ID as index into a table with already determined routes, identified by outgoing interfaces.

It comes very closely to virtual circuits:
- in VCs, however, there is a setup phase for each connection;
- here, instead, routes are created on demand when a packet arrives (data-driven);
- alternatively, label are created by routers at boot time (control-driven).

When a router is booted, it checks to see for which routes it is the final destination (e.g., which hosts are on its LAN).

It then creates one or more FECs for them; allocates a label for each one, and passes the labels to its neighbors.

MLPS

While IETF was busy with integrated and differentiated services, router vendors were working on label switching approaches.

Finally, the IETF began to standardized this under the name of MultiProtocol Label Switching (MLPS).

The label is used per router to match incoming-to-outgoing interfaces. It may be changed when a packet leaves the router.

The QoS field indicates the class of service.

S indicates whether a hierarchy of labels is used.

Another difference between VCs and MPLS is that multiple flows can be aggregated.

Messages can be disambiguated through the IP address carried in the message.

Internetworking

Problem: Great! We have all these networks, with all different protocol stacks, and now we just to let them talk to each other.

Non-Solution: Enforce all networks to run the same protocol stack that’s asking for a lot of trouble, and effectively saying that we are not allowed to make any progress.

Solution: Construct all kinds of gateways that connect to different kinds of networks.

Gateways

Repeaters at the physical layer for boosting signals.

Bridges / Switches to make the interconnection at the data link layer.

Multiprotocol routers for forwarding, and possibly splitting up packets (bridges can’t do the latter).

Transport gateways for coupling byte streams in different networks.

Application gateways, e.g., for handling electronic mail between OSI and TCP/IP networks.
### The Differences

<table>
<thead>
<tr>
<th>Issue</th>
<th>Differences</th>
</tr>
</thead>
<tbody>
<tr>
<td>Service offered</td>
<td>Connection-oriented vs connectionless</td>
</tr>
<tr>
<td>Protocols</td>
<td>IP, IPX, CLNP, AppleTalk, etc.</td>
</tr>
<tr>
<td>Addressing</td>
<td>Flat (802) vs hierarchical</td>
</tr>
<tr>
<td>Multicasting</td>
<td>Present versus absent</td>
</tr>
<tr>
<td>Packet size</td>
<td>Network specific maximum</td>
</tr>
<tr>
<td>Quality of Service</td>
<td>None versus a lot versus who knows</td>
</tr>
<tr>
<td>Error handling</td>
<td>Reliable, (un)ordered delivery</td>
</tr>
<tr>
<td>Flow control</td>
<td>Sliding window, rate control, etc.</td>
</tr>
<tr>
<td>Congestion control</td>
<td>Leaky bucket, choke packets, etc.</td>
</tr>
<tr>
<td>Security</td>
<td>Privacy rules, encryption, etc.</td>
</tr>
<tr>
<td>Parameters</td>
<td>Timeouts, flow specs, etc.</td>
</tr>
<tr>
<td>Accounting</td>
<td>Connect time, packets, bytes, none</td>
</tr>
</tbody>
</table>

---

### Don’t worry

It’s impossible to resolve all differences. The solution is to just take a simple approach (like the Internet). We now only consider the red issues.

---

### Concatenated Virtual Circuits

- Assume the constituent networks support virtual circuits
  - in that case, the internet (note the lower case) can use virtual circuit technology through concatenation

---

### Using Datagrams

- Often, connection-oriented services are not even supported, e.g., in Internet (note the upper case)
  - the network layer offers only datagram services: unreliable, unordered packet flow.

---

### Tunneling

- We can solve a lot of the internetworking problems when we can assume that the source and destination are on the same type of network (e.g., Ethernet)
- In that case, we need only to tunnel packets through intermediate networks (e.g., ATM WAN)

---

Notes

---

Notes

---

Notes

---

Notes

---

Notes
Fragmentation

Problem: Different networks may impose different maximum packet sizes. ⇒ we may have to split a packet into smaller ones when forwarding it through an intermediate network (fragmentation)

Two options
1. let each router fragment / reassemble the packet (transparent)  
2. recombine fragments only at the final destination (non-transparent)

Question
Why couldn’t the same issue be solved by bridges? Because they just talk a single (data link layer) protocol

Solution:
- Define an elementary fragment size small enough that the elementary fragment can pass through every network
- The internet header must provide the original packet number and the number of the first elementary fragment contained in the packet
- There must also be a bit indicating that the last elementary fragment contained within the internet packet is the last one

The Internet

The Internet: view it as a collection of autonomous systems (ASes) connected together by a bunch of backbones:

Designed according to the principles in the RFC 1958:
- keep it simple
- exploit modularity
- look for a good design: it need not be perfect
- think about scalability

Requests for Comments (RFCs)
RFCs are documents published by the Internet Engineering Task Force (IETF) describing methods, research, or innovations applicable to the Internet

Internet Model

An application offers a data stream to the transport layer, using either connection-oriented or connectionless services.

The transport layer breaks up the data stream into datagrams, and passes these to the network layer.

The datagrams are routed through the internet, occasionally fragmented when needed.
- routers always pass the datagram to the underlying data link layer, generally LANs and (dial-up or leased) telephone lines
Internet Network Layer

- **IP**
  - addressing
  - datagram format
  - fragmentation and packet handling

- **ICMP**
  - error reporting
  - signaling

- **Routing**: defining paths and compiling forwarding tables
  - RIP
  - OSPF
  - BGP

---

### IP header

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>version</td>
<td>IP version (e.g., IPv4 or IPv6)</td>
</tr>
<tr>
<td>length</td>
<td>Length of the header</td>
</tr>
<tr>
<td>TOS/flags</td>
<td>3-bit priority plus 3-bit flag (delay, throughput, reliability)</td>
</tr>
<tr>
<td>TLen/offset</td>
<td>Length header + payload</td>
</tr>
<tr>
<td>ID/offset</td>
<td>Datagram id + fragmentation offset</td>
</tr>
<tr>
<td>TTL</td>
<td>Maximum number of hops allowed</td>
</tr>
<tr>
<td>Prot/Checksum</td>
<td>Globally defined ids for transport protocols. Checks the headers (has to be calculated at each hop)</td>
</tr>
</tbody>
</table>

**Flags**
- DF: Don't fragment
- MF: There are more fragments

**Source address**

**Destination address**

**Options (if any)**

---

### IP Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Security</td>
<td>Specifies how secret the datagram is</td>
</tr>
<tr>
<td>Strict source routing</td>
<td>Gives the complete path to be followed</td>
</tr>
<tr>
<td>Loose source routing</td>
<td>Gives a list of routers not to be missed</td>
</tr>
<tr>
<td>Record route</td>
<td>Makes each router append its IP address</td>
</tr>
<tr>
<td>Timestamp</td>
<td>Makes each router append its address and timestamp</td>
</tr>
</tbody>
</table>

**Note**
- Security is hardly used: specifying that a datagram is “really top secret” is not such a good idea.
- Source routing can be effectively used to force routes: debugging, politics, and mobile hosts.
- Record route and Timestamping are mainly used for debugging.

---

### IPv4 Addressing

- **32-bit addresses**
  - 4 billion possible IP addresses

**An IP address is associated with an interface, not a host**
- a host with more than one interface may have more than one IP address

**The assignment of addresses over an Internet topology is crucial to limit the complexity of routing and forwarding**

**The key idea is to assign addresses with the same prefix to interfaces that belong to the same organization**


For several decades, IP addresses were divided into 5 categories. This allocation has come to be called classful addressing (no longer used).

Each class allows up to a maximum number of hosts:

<table>
<thead>
<tr>
<th>Class</th>
<th>Max. Networks</th>
<th>Max. Hosts/Network</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>126</td>
<td>16,777,214</td>
</tr>
<tr>
<td>B</td>
<td>16,382</td>
<td>65,536</td>
</tr>
<tr>
<td>C</td>
<td>2,097,150</td>
<td>254</td>
</tr>
</tbody>
</table>

Network numbers are managed by a nonprofit corporation called ICANN (Internet Corporation for Assigned Names and Numbers) to avoid conflicts. In turn, ICANN has delegated parts of the address space to various regional authorities, which then dole out IP addresses to ISPs and other companies.

Network addresses, which are 32-bit numbers, are usually written in dotted decimal notation. Each of the 4 bytes is written in decimal, from 0 to 255. For example, the 32-bit hexadecimal address C0290614 is written as 192.41.6.20.

Subnet Masking

Problem: All hosts on the same network must have the same network number. This may cause a single organization to acquire several classes of addresses (e.g., each time it adds a LAN), that would subsequently need to be announced worldwide.

Solution: Use a single network address for the entire organization, and internally divide the host address space into a subnet address and a host ID.

This introduces a 3-level routing hierarchy:
1. External routers just look at the network address and forward the packet to one of the routers of the organization.
2. Subnet routers apply the subnet mask, and look-up whether the destination is on their subnet, or whether they should forward it to another subnet router.
3. When the host is identified, the router knows which interface to use to forward the packet.

Interconnection of Networks

Notes
All interfaces in the same subnet share the same *address prefix*
- e.g., in the previous example we have 123.1.1.—, 123.1.1.—, 101.0.1.—, and 111.3.3.—.

Network addresses prefix-length notation: *address/prefix-length*
- e.g., 123.1.1.0/24, 123.1.1.0/24, 101.0.1.0/24, and 111.3.3.0/24
- 123.1.1.0/24 means that all the addresses share the same leftmost 24 bits with address 123.1.1.0

This addressing scheme is not limited to entire bytes. For example, a network address might be 128.138.207.160/27

### Subnet Ranges

What is the range of addresses in 128.138.207.160/27?

```
subnet 10000000 10001010 11001111 101 00000

10000000 10001010 11001111 10100000
10000000 10001010 11001111 10100001
10000000 10001010 11001111 10100010
10000000 10001010 11001111 10100011
```

128.138.207.160–128.138.207.191

### Net Mask

- Network addresses, *mask* notation: *address/mask*
- A prefix of length \( p \) corresponds to a mask
  \[
  M = \underbrace{11 \cdots 1}_{p \text{ times}} \underbrace{00 \cdots 0}_{32- \text{prefix length} \text{ times}}
  \]
- e.g., 128.138.207.160/27 = 128.138.207.160/255.255.255.224
- 127.0.0.1/8 = 255.255.255.255
- 192.168.0.3/24 = 255.255.255.248
- 195.176.181.11/32 = 255.255.255.255

- In Java:

```java
boolean match(int address, int network, int mask) {
    return (address & mask) == (network & mask);
}
```

### Classless Interdomain Routing

- This *any-length prefix* scheme is also called *classless interdomain routing* (CIDR)
- as opposed to the original scheme which divided the address space in “classes”:

<table>
<thead>
<tr>
<th>address class</th>
<th>prefix length</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>8</td>
</tr>
<tr>
<td>B</td>
<td>16</td>
</tr>
<tr>
<td>C</td>
<td>24</td>
</tr>
</tbody>
</table>

- Although there are “enough” IP addresses, the use of classes is rapidly making them a scarce resource
- the main problem is that too many class B addresses are being used.

- Why is the idea of the common prefix so important?
- Routers outside a (sub)network can ignore the specifics of each address within the network
- there might be some 64 thousands hosts in 128.138.0.0/16, but they all appear as one address from the outside.
Allocation of Address Blocks

<table>
<thead>
<tr>
<th>ISP X</th>
<th>Internet</th>
</tr>
</thead>
<tbody>
<tr>
<td>123.4.0.0/16</td>
<td>thedude.org</td>
</tr>
<tr>
<td>123.4.1.0/24</td>
<td>maude.com</td>
</tr>
<tr>
<td>123.4.20.0/24</td>
<td>bowling.edu</td>
</tr>
<tr>
<td>98.7.1.0/24</td>
<td>margie.net</td>
</tr>
</tbody>
</table>

Notes

Longest-Prefix Matching

- In choosing where to forward a datagram, a router chooses the entry that matches the destination address with the longest prefix.

<table>
<thead>
<tr>
<th>forwarding table</th>
</tr>
</thead>
<tbody>
<tr>
<td>network</td>
</tr>
<tr>
<td>123.4.0.0/16</td>
</tr>
<tr>
<td>98.7.1.0/16</td>
</tr>
<tr>
<td>123.4.20.0/24</td>
</tr>
<tr>
<td>128.0.0.0/1</td>
</tr>
<tr>
<td>66.249.0.0/16</td>
</tr>
<tr>
<td>0.0.0.0/1</td>
</tr>
<tr>
<td>128.138.0.0/16</td>
</tr>
</tbody>
</table>

Notes

Network Address Translation

- NAT (Network Address Translation) partially solves the shortage of IP addresses by assigning each company a single IP address (RFC 3022).
  - within the company, every computer gets a unique IP address, which is used for routing intramural traffic.
  - however, when a packet exits the company and goes to the ISP, an address translation takes place.
- It also serves to share a home ADSL connection among multiple PCs and laptops.
- The NAT designers observed that most IP packets carry either TCP or UDP payloads:
  - both of these have headers containing a source port and a destination port.
  - ports identify different process running on the same host.
  - e.g., web server is running on port 80 and SSH on port 22.
- The following ranges have been declared as private:
  - 10.0.0.0 — 10.255.255.255/8 (16,777,216 hosts)
  - 172.16.0.0 — 172.31.255.255/12 (16,384 hosts)
  - 192.168.0.0 — 192.168.255.255/16 (65,536 hosts)

Notes

Network Address Translation Example

- When a connection is set up from address 10.0.0.1, port X, the router sends it off using source address 138.76.29.7 (its ISP-supplied address) on port Y, and registers the mapping X ↔ Y.
- When a reply comes in for port Y, it is sent back to 10.0.0.1 on port X.

Question

- How can you allow incoming connections? You can set up a static mapping on the NAT. E.g., redirect all traffic to port 80 to 10.0.0.3 on port 8080.
Network Address Translation

Criticism

Despite its recent widespread, many purists in the IETF community loudly object to NAT for several reasons (see RFC 2993):

(a) NAT violates the architectural model of IP, which states that every IP address uniquely identifies a single machine worldwide.
(b) NAT changes the Internet from a connectionless network to a kind of connection-oriented network.
   - The NAT box must maintain information (the mapping) for each connection passing through it.
(c) NAT violates the most fundamental rule of protocol layering: Layer k may not make any assumptions about what Layer k+1 does.
   - If TCP is later upgraded to TCP-2, NAT will fail.
(d) If a user on machine A decides to use some new transport protocol to talk to a user on machine B, NAT will fail.
(e) Many peer-to-peer protocols (e.g., Skype or BitTorrent) require full connectivity among hosts.
   - If a user is behind a NAT, it cannot be contacted by other peers.
   - Some techniques exist to circumvent the issue to some extent, e.g., using relays or NAT punching hole like STUN (more info on the website).
(f) IPv6 is the best solution to deal with the lack of IP addresses.

Universal Plug and Play (UPnP)

UPnP is a protocol to allow a user to discover and configure a nearby NAT.

With UPnP, an application can request a NAT mapping between its (private IP address, private port) and (public IP address, public port).

Example:

- Assuming a host with private address 10.0.0.1 is running BitTorrent on port 3345 and the NAT public address is 138.76.29.7.
- The BitTorrent application asks the NAT to create a hole that maps (10.0.0.1, 3345) to (138.76.29.7, 5001).
- The application advertises its public address to the tracker.
- The other peers can open connections to the host.

In summary, UPnP allows external hosts to initiate communication session to NATed hosts, using either TCP or UDP.

Internet Control Message Protocols (ICMP)

ICMP is used to inform hosts and routers when things go wrong, or, likewise, should be able to send queries to get status information.

- Distinguish several message types (destination unreachable, time exceeded, etc.)
- Each message type is further specified by a code.
  - Example: a destination can be unreachable (message type = 3).
  - Because of an unsupported protocol (code = 2).
  - Other classing example is ping: the querying node sends an ICMP type 8 code 0 message to the host which sends back a type 0 code 0 ICMP reply.
- Originally it was also used to throttle hosts sending too many packets.
  - Now discontinued as these messages tend to increase congestion.
- Congestion control in the Internet is largely done in the transport layer.

Question

How can we recognize ICMP messages when they are encapsulated in IP datagrams? Through the protocol identifier in the header of the IP datagram containing the ICMP message.

Network vs. Data Link Layer Addresses

- Addressing hosts using IP addresses is great, but these addresses are not recognized by the hardware of those hosts.
  - Example: a host on an Ethernet LAN will only read messages encapsulated in frames containing that host's hardware address.
- Problem: How do we find out the hardware (i.e. data link) address of a host, given its Internet address?
Address Resolution Protocol

- The protocol used to discover network addresses is called Address Resolution Protocol (ARP)
  1. Router: Ask each host on the LAN whether they have the requested IP address. This is done by encapsulating the query as an ARP message in a datalink frame, and broadcasting it.
  2. Host: Recognizes it is dealing with an ARP message, checks whether it has the requested address, and if so, sends a reply back with its datalink address.
  3. Router: Recognizes a reply ARP message, and (generally) caches the IP address with the datalink address. It can then forward IP datagrams to the correct host, encapsulating them in datalink frames.

- Usually the IP-to-Ethernet mapping of the router is also included in the request so all nodes can store it
- Several optimizations, e.g., caching recent replies or having each machine broadcast its address when booted, are also possible

Dynamic Host Configuration Protocol (DHCP)

- **Problem:** So how does a host know its own IP address?
- **Solution:** Dynamic Host Configuration Protocol (RFC 2131 and 2132)
  - A newly-booted machine broadcasts a DHCP DISCOVER packet
  - The DHCP server (or its relay) intercepts the packet and replies with the assigned IP address
  - The DHCP server can maintain static mappings or assign a different address each time the host connects

- To prevent a host to keep an address for ever (even after it disconnected), addresses must be periodically renewed (leasing)

Network Model

- So far we have studied routing over a “flat” network model

![Network Model Diagram]

- Also, our objective has been to find the least-cost paths between sources and destinations

More Realistic Topologies

![More Realistic Topologies Diagram]
Hierarchical Routing

- The network flat model is too simplistic for at least two important reasons:
  - **Scalability**
    - hundreds of millions of hosts in today's Internet
    - transmitting routing information (e.g., LSAs) would be too expensive
    - forwarding would also be too expensive
  - **Administrative autonomy**
    - one organization might want to run a distance-vector routing protocol,
      while another might want to run a link-state protocol
    - an organization might not want to expose its internal network structure

- Today's Internet is organized in **autonomous systems (ASs)**
  - independent administrative domains
  - **Gateway routers** connect an autonomous system with other autonomous systems

IP Routing

- Make a distinction between routing in an autonomous system, and between autonomous systems:
  - **An intra-autonomous system routing protocol** runs within an autonomous system
    - this protocol determines internal routes
      - internal router ↔ internal router
      - internal router ↔ gateway router
    - it should do this as best as possible (optimal routing)
      - e.g., RIP (distance vector) mainly used in low-tier ISPs and small enterprises and OSPF (link state) used by tier-1 ISP
  - **An inter-autonomous system routing protocol** determines routing at the autonomous-system level
    - Inter-AS routing has to deal with a lot of politics.
    - For example, some ASes should not be traversed at all, whereas some do not accept "foreign" packets.
    - **BGP** is the Internet standard
Intra-AS Routing

At AS3:
AS1 → AS1; AS2 → AS2; AS4 → AS1.

Inter-AS Routing

Hierarchical Routing

- All routers within an AS compute their intra-AS routing information
- Gateway routers figure out inter-AS routing information
  - inter-AS routing information is propagated within an AS
- Both inter-AS and intra-AS routing information is used to compile the forwarding tables
- Destinations within the same autonomous system are reached as usual
- What about a destination x outside the autonomous system?
  - inter-AS information is used to figure out that x is reachable through gateway G_x
  - intra-AS information is used to figure out how to reach G_x within the AS
  - what if x is reachable through multiple gateway routers G_x, G'_x, ...?
    - use intra-AS routing information to determine the costs of the (least-cost) paths to G_x, G'_x, ...
    - "hot-potato" routing: send it through the closest gateway

Interior Gateway Routing: OSPF

- Open Shortest Path First: link state routing protocol, replacement for (still widely used, but inadequate) RIP (routing information protocol). Requirements:
  - Openness: the algorithm should be made publicly available so that anyone could implement it.
  - Support for different distance metrics (hops, delays, costs, etc.)
  - Dynamic and efficient adaptability to changing topologies.
  - Support routing based on type of service (already specified in IP, but no one actually used it). Especially important for real-time (multimedia) traffic.
  - Support for load balancing: when a route is heavily used, another one should be selected.
  - Support hierarchical routing.
  - Offer security.
  - Support IP tunneling.
OSPF operates by abstracting the collection of actual networks, routers, and lines into a directed graph in which each arc is assigned a cost (distance, delay, etc.).
- A serial connection between two routers is represented by a pair of arcs, one in each direction.
- A multiaccess network is represented by a node for the network itself plus a node for each router.
- It then computes the shortest path from every router to every other router.

Many ASs on the Internet are themselves large and non-trivial to manage
- OSPF allows them to be divided into areas
- We can use the same algorithm for the areas and the backbone
  - Within each area, one or more area border routers are responsible for routing packets outside the area
  - The backbone area routes traffic between other areas in the AS
  - It must contain all border routers

The Border Gateway Protocol (BGP) is the inter-AS routing protocol in today's Internet
- Neighboring routers maintain a connection to simplify message reliability.
  - Provides reachability information from neighbor ASs
  - Transmits reachability information to all internal routers within an AS
- Determines good routes to all outside subnets
  - Based on reachability information
  - Based on policies
  - Routers do not automatically use the routes they find, but have to check whether it is allowed
- BGP is a path-vector protocol
  - Is based on distance vector routing, but paths rather than distances are announced
  - We do not have a count-to-infinity problem here, because a router decides on an entire path.
**Inter-AS Routing: BGP**

### Routes

- **BGP advertisement**: a router advertises routes to networks, much like an entry in a distance-vector
  - destinations are denoted by address prefixes
  - an AS may or may not forward an advertisement for a foreign network; doing so means being willing to carry traffic for that network
  - this is where a router may aggregate prefixes (a.k.a., "supernetting")
    - E.g.,
      - \[128.138.242.0/24\] \(\rightarrow\) \[128.138.242.0/23\]
      - \[191.224.128.0/22\] \(\rightarrow\) \[191.224.128.0/21\] \(\rightarrow\) \[191.224.132.0/22\]

### Message Structure

- **Autonomous system number (ASN)**: a unique identifier for each AS (with more than one gateway)
- **BGP attributes**: a route advertisement includes a number of attributes
  - **AS-PATH**: sequence of ASNs through which the advertisement has been sent
  - **NEXT-HOP**: specifies the interface (IP address) to use to forward packets towards the advertised destination
    - used to resolve ambiguous cases where an AS can be reached through multiple gateways (interfaces)
- **BGP import policy**: used to decide whether to accept or reject the route advertisement
  - e.g., a router may not want to send its traffic through one of the AS listed in AS-PATH

### Route Selection

- The count-to-infinity problem can be easily solved by inspecting path
  - e.g., assume \(G\) crashes and then \(F\) receives routes from its neighbors
  - it can immediately see that the two latter routes are pointless, since they pass through \(F\) itself, so it chooses \(FBCD\) as its new route
- **Router preference**: routes are ranked according to a
  1. preference value
    - depends on the policy (may be \(\infty\))
  2. shortest AS-PATH
  3. closest NEXT-HOP router

### YouTube Hijacking

- The Pakistani authorities wanted to prevent their citizens to access YouTube content
- Therefore, on Sunday, 24-02-2008, Pakistan Telecom started an unauthorized announcement of the prefix \(208.65.153.0/24\)
  - this way they aimed at redirect all the YouTube traffic to their servers
- Unfortunately, one of Pakistan Telecom’s upstream providers, PCCW Global forwarded this announcement to the rest of the Internet
  - hijacking of YouTube traffic on a global scale
Inter-AS Routing: BGP

Before, during and after Sunday, 24-02-2008: AS36561 (YouTube) announces 208.65.152.0/22.

24-02-2008, 18:47 (UTC): AS17557 (Pakistan Telecom) starts announcing 208.65.153.0/24. With two identical prefixes in the routing system, BGP policy rules, such as preferring the shortest AS path, determine which route is chosen.

This means that AS17557 (Pakistan Telecom) continues to attract some of YouTube’s traffic.


because of the longest prefix match rule, every router that receives these announcements will send the traffic to YouTube.


because of the longest prefix match rule, every router that receives these announcements will send the traffic to YouTube.

24-02-2008, 21:01 (UTC): AS3491 (PCCW Global) withdraws all prefixes originated by AS17557 (Pakistan Telecom), thus stopping the hijack of 208.65.153.0/24.

MobileIP

Problem Many users of the Internet have portable computers and want to stay connected to the Internet when they move

unfortunately, the IP addressing system make this hard to achieve

E.g., consider a machine with IP address 160.80.40.20/16

routers all over the world have routing tables telling which line to use to get to network 160.80

if the machine with that address is carted off to some distant site, the packets for it will continue to be routed to its home LAN

Have the routers use the complete IP addresses for routing, is out of question because this would increase the routing table size.

Solution: use the approach described earlier based on home and foreign agent

messages are routed to the home agent and then tunneled to the host’s new location

IPv6

Problem: The current version of IP can not support enough addresses, but also lacks the flexibility to act as the basis for a large variety of users. The goals:

Support billions of hosts.
Reduce the size of routing tables.
Simplify the protocol to enable faster routers.
Provide (better) security.
Better support for type of service.
Support scopes with multicasting.
Support roaming hosts without address changes.
Coexistence of the old and new protocol, and evolution of the new one.

IPv6 Datagram Format

The flow label is used to set up a pseudoconnection between source and destination.

it identifies a flow for which, e.g., bandwidth has been reserved.

Expanded addressing

128-bit addresses

anycast address (a single address representing a set of interfaces)

A simpler header is almost impossible

reduces processing cost and bandwidth usage

support for extensions and options is provided by next headers.
IPv6

What is Missing from IPv4?

- **Fragmentation**: IPv6 pushes fragmentation onto the end-systems
- **Header checksum**: IPv6 avoids recomputing the checksum by getting rid of it altogether
- **Options**: a fixed-length header is easier to process, better modularity for extensions and options

IPv6 Address Space

- IPv6 uses 16-byte addresses. This is really a lot: \(7 \times 10^{23}\) addresses per square meter, higher than the Avogadro's number.
- It allows us to be less efficient with address allocation: 72% is unassigned.
- **Hexadecimal notation**: eight group of four hexadecimal digits with colons between groups, e.g., 2001:610:110:4e2:217:42ff:fe1a:3ebf.
- IPv4 addresses are still supported: ::130.37.31.43.

IPv6 Extension Headers

- **Basic idea**: Keep the main header as simple as possible, and provide any further information in an (optional) extension header.
- **Fragmentation** is still supported, but only the source host can do it; routers never fragment datagrams anymore.

Higher-Level Protocol and Extensions
There was a lot of discussion on where and how to incorporate security in IPv6:

- If you are really concerned about security, would you trust anything else but end-to-end encryption?
- Having security in the network layer offers a generally useful service to many applications. Those that don’t want to use it, just ignore it.
- Network-layer protocols have to run in every country. Some countries disallow cryptosystems that the government can’t decrypt easily.
- Are the default crypto-algorithms good enough? For example, MD5 has been cracked.
- The main issue here, as with almost every protocol, is to decide in which layer we should put functionality.

There are many people who argue that only end-to-end solutions should be applied; the rest (i.e., general solutions) will never be good enough.

Problem: IPv6 is not going to take over in a split second.

- the two protocols will have to live side-by-side for a very long time.

There are three solutions:

(a) Dual-stack techniques
(b) Tunneling techniques
(c) Network Address Translation and Protocol Translation (NAT-PT)

The US office of Management and Budget (OMB) has required the transition to IPv6 by June 2008

- still 24 days to go...

The IPv6 header is converted into a IPv4 header

- some fields, e.g., flow, can be lost

Dual-stack nodes have a separate IPv4 and IPv6 address.

You can also have complete dual-stack networks: all routers run both protocols

- requires a lot of redundancy (e.g., routing tables).

Basic solution: Simply forward IPv6 packets as payload encapsulated in IPv4 packets

- fragmentation and such can simply be applied to the IPv4 packet.

Manual tunneling: Explicitly set up a tunnel between two dual-stack machines.

Automatic tunneling: Use IPv4-compatible IPv6 addresses and route these through a dual-stack network.
Automatic tunneling: 6to4

- Encode the IPv4 address of your network's gateway into every address of your local IPv6 network.
- For any 32-bit global IPv4 address, a 48-bit 6to4 IPv6 prefix can be constructed by prepending 2002 (hex) to the IPv4 address.

```
<table>
<thead>
<tr>
<th>0x0002 IPv4 gateway addr</th>
<th>network ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>3 13 32 16</td>
<td>001</td>
</tr>
</tbody>
</table>
```

E.g., supported by Apple Airport Extreme

NAT-PT

- Leverage off the same principle of IPv4 NAT
  1. When a packet coming from an IPv6 network directed to an IPv4 host arrives, the NAT picks up an IPv4 address from its pool and maps it to the IPv6 source address
  2. Then, it creates an IPv4 header by copying the IPv6 header using the newly created IPv4 address as source.
  3. When a reply is sent back to this IPv4 address, the NAT perform the opposite translation by converting the IPv4 message into an IPv6 message using the original IPv6 source address as destination address
- A similar process occurs if the communication is initiated by a IPv4 host

Mobile IPv6

**Essence**: Works the same as with Mobile IPv4, but some things can be done better when nodes speak IPv6:
- There is no need for a foreign agent: a mobile IPv6 host can get its own care-of address
- No lack of addresses
- The mobile hosts piggybacks its home address with every packet.
  - The IP layer pretends that this is the true source address (useful for firewalls).
  - A mobile IPv6 host locally maintains a home–to–care-of address binding, and replaces the former with the latter when sending packets.

Lab Session

- **Wireshark** (formerly known as Ethereal)
  - A packet sniffer computer application
  - It allows you to inspect the content of each network packet sent / received by the running host
  - Connect to a website (e.g., Google) and then inspect the packet
  - Look at Ethernet, IP and TCP header
  - What is the fragmentation offset?
  - How about the header and payload lengths?
  - What is the version of the protocol?

- **Ping**
  - A computer network tool used to test whether a particular host is reachable
  - It works by sending ICMP "echo request" packets to the target host and listening for ICMP "echo response" replies
  - It estimates the round-trip time (RTT) and records any packet loss
  - Try pinging the following sites and observe the different RTTs:
    - fluit.few.vu.nl
    - mit.edu
    - www.ust.hk
  - Inspect the packet contents using Wireshark

- **Traceroute**
  - A computer network tool used to determine the route taken by packets
  - It works by increasing the TTL value of each successive batch of packets sent
  - Check TTL fields using Wireshark
  - Check fragmentation fields when invoked without any arguments and when invoked as `traceroute mtu.edu:2000` (i.e., each packet has a size of 2000 bytes)