**CAPITULO 5**

The network layer is concerned with getting packets from the source all the way to the destination. Getting to the destination may require making many hops at intermediate routers along the way.

To achieve its goals, the network layer must know about the topology of the network (i.e., the set of all routers and links) and choose appropriate paths through it, even for large networks. It must also take care when choosing routes to avoid overloading some of the communication lines and routers while leaving others idle.

**5.1 NETWORK LAYER DESIGN ISSUES**

**5.1.2 Services Provided to the Transport Layer**

The network layer provides services to the transport layer at the network layer/transport layer interface. An important question is precisely what kind of services the network layer provides to the transport layer. The services need to be carefully designed with the following goals in mind:

1. The services should be independent of the router technology.

2. The transport layer should be shielded from the number, type, and topology of the routers present.

3. The network addresses made available to the transport layer should use a uniform numbering plan, even across LANs and WANs

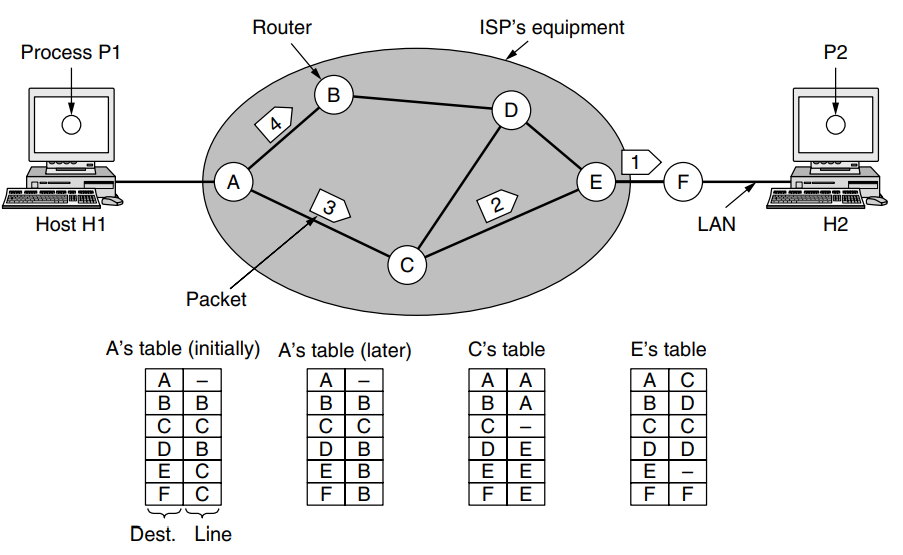
An ongoing discussion centers on whether the network layer should provide **connection-oriented service** or **connectionless service.**

One camp (represented by the Internet community) argues that the routers’ job is moving packets around and nothing else. In this view, the network is inherently unreliable, no matter how it is designed. Therefore, the hosts should accept this fact and do error control and flow control themselves. This viewpoint leads to the conclusion that the network service should be connectionless, with primitives SEND PACKET and RECEIVE PACKET and little else.  
  
The other camp (represented by the telephone companies) argues that the network should provide a reliable, connection-oriented service. In this view, quality of service is the dominant factor, and without connections in the network, quality of service is very difficult to achieve, especially for real-time traffic such as voice and video.

**5.1.3 Implementation of Connectionless service**

If connectionless service is offered, packets are injected into the network individually and routed independently of each other. No advance setup is needed. In this context, the packets are frequently called **datagrams** and the network is called a **datagram network**.

If connection-oriented service is used, a path from the source router all the way to the destination router must be established before any data packets can be sent. This connection is called a **VC (virtual circuit)**, in analogy with the physical circuits set up by the telephone system, and the network is called a **virtual-circuit network**.



At A, each packet is forwarded according to A’s table, onto the outgoing link to C within a new frame. Packet 1 is then forwarded to E and then to F. When it gets to F, it is sent within a frame over the LAN to H2. Packets 2 and 3 follow the same route.

However, something different happens to packet 4. When it gets to A it is sent to router B, even though it is also destined for F. For some reason, A decided to send packet 4 via a different route than that of the first three packets. Perhaps it has learned of a traffic jam somewhere along the ACE path and updated its routing table, as shown under the label ‘‘later.’’ The algorithm that manages the tables and makes the routing decisions is called the **routing algorithm**.

IP (Internet Protocol), which is the basis for the entire Internet, is the dominant example of a connectionless network service. Each packet carries a destination IP address that routers use to individually forward each packet. The addresses are 32 bits in IPv4 packets and 128 bits in IPv6 packets.

**5.1.3 Implementation of Connection-Oriented service**

For connection-oriented service, we need a virtual-circuit network. The idea behind virtual circuits is to avoid having to choose a new route for every packet sent.

When a connection is established, a route from the source machine to the destination machine is chosen as part of the connection setup and stored in tables inside the routers. That route is used for all traffic flowing over the connection.

Diagram

Description automatically generatedWhen the connection is released, the virtual circuit is also terminated. With connection-oriented service, each packet carries an identifier telling which virtual circuit it belongs to.

The first line of A’s table says that if a packet bearing connection identifier 1 comes in from H1, it is to be sent to router C and given connection identifier 1. Similarly, the first entry at C routes the packet to E, also with connection identifier 1.

Now let us consider what happens if H3 also wants to establish a connection to H2. It chooses connection identifier 1 and tells the network to establish the virtual circuit. This creates a conflict because although A can easily distinguish connection 1 packets from H1 from connection 1 packets from H3, C cannot do this.

For this reason, A assigns a different connection identifier to the outgoing traffic for the second connection. Avoiding conflicts of this kind is why routers need the ability to replace connection identifiers in outgoing packets.

In some contexts, this process is called **label switching**.

**5.1.3 Implementation of Connection-Oriented service**

Table

Description automatically generatedBoth virtual circuits and datagrams have their supporters and their detractors.

One trade-off is setup time versus address parsing time. Using virtual circuits requires a setup phase, which takes time and consumes resources. However, once this price is paid, the router just uses the circuit number to index into a table to find out where the packet goes. In a datagram network, no setup is needed but a more complicated lookup procedure is required to locate the entry for the destination.

A related issue is that the destination addresses used in datagram networks are longer than circuit numbers used in virtual-circuit networks because they have a global meaning. If the packets tend to be fairly short, including a full destination address in every packet may represent a significant amount of overhead, and hence a waste of bandwidth.

Another issue is the amount of table space required in router memory. A datagram network needs to have an entry for every possible destination, whereas a virtual-circuit network just needs an entry for each virtual circuit.

The loss of a communication line is fatal to virtual circuits using it, but can easily be compensated for if datagrams are used. Datagrams also allow the routers to balance the traffic throughout the network, since routes can be changed partway through a long sequence of packet transmissions.

**5.2 Routing Algorithms**

The main function of the network layer is routing packets from the source machine to the destination machine. In most networks, packets will require multiple hops to make the journey. The algorithms that choose the routes and the data structures that they use are a major area of network layer design.

The **routing algorithm** is that part of the network layer software responsible for deciding which output line an incoming packet should be transmitted on. If the network uses datagrams internally, this decision must be made anew for every arriving data packet since the best route may have changed since last time. If the network uses virtual circuits internally, routing decisions are made only when a new virtual circuit is being set up. Thereafter, data packets just follow the already established route. The latter case is sometimes called **session routing** because a route remains in force for an entire session.

It is sometimes useful to make a distinction between routing, which is making the decision which routes to use, and forwarding, which is what happens when a packet arrives. One can think of a router as having two processes inside it. One of them handles each packet as it arrives, looking up the outgoing line to use for it in the routing tables. This process is **forwarding**. The other process is responsible for filling in and updating the routing tables. That is where the routing algorithm comes into play.

**5.3 CONGESTION CONTROL ALGORITHMS**

Too many packets present in (a part of) the network causes packet delay and loss that degrades performance. This situation is called **congestion**. The network and transport layers share the responsibility for handling congestion. Since congestion occurs within the network, it is the network layer that directly experiences it and must ultimately determine what to do with the excess packets. However, the most effective way to control congestion is to reduce the load that the transport layer is placing on the network. This requires the network and transport layers to work together.

**5.6 The Network Layer in the Internet**

The glue that holds the whole Internet together is the network layer protocol, **IP (Internet Protocol)**. Unlike most older network layer protocols, IP was designed from the beginning with internetworking in mind. A good way to think of the network layer is this: its job is to provide a best-effort way to transport packets from source to destination, without regard to whether these machines are on the same network or whether there are other networks in between them.

Communication in the Internet works as follows. The transport layer takes data streams and breaks them up so that they may be sent as IP packets. IP routers forward each packet through the Internet, along a path from one router to the next, until the destination is reached. At the destination, the network layer hands the data to the transport layer, which gives it to the receiving process.

**5.6.1 The IP version 4 protocol**

Graphical user interface, text, application, email

Description automatically generatedNext, the format of the IP datagrams themselves. An IPv4 datagram consists of a header part and a body or payload part. The header has a 20-byte fixed part and a variable-length optional part. The bits are transmitted from left to right and top to bottom, with the high-order bit of the *Version* field going first.

The *Version* field keeps track of which version of the protocol the datagram belongs to. By including the version at the start of each datagram, it becomes possible to have a transition between versions over a long period of time.

Since the header length is not constant, a field in the header, *IHL*, is provided to tell how long the header is, in 32-bit words. The minimum value is 5, which applies when no options are present. The maximum value of this 4-bit field is 15, which limits the header to 60 bytes, and thus the *Options* field to 40 bytes.

The *Differentiated services* field is intended to distinguish between different classes of service. Various combinations of reliability and speed are possible. For digitized voice, fast delivery beats accurate delivery. For file transfer, error-free transmission is more important than fast transmission. The *Type of service* field provided 3 bits to signal priority and 3 bits to signal whether a host cared more about delay, throughput, or reliability.

The *Total length* includes everything in the datagram—both header and data. The maximum length is 65,535 bytes.

The *Identification* field is needed to allow the destination host to determine which packet a newly arrived fragment belongs to. All the fragments of a packet contain the same *Identification* value.

Next comes an unused bit, which is surprising, as available real estate in the IP header is extremely scarce.

Then come two 1-bit fields related to fragmentation. *DF* stands for Don’t Fragment. It is an order to the routers not to fragment the packet.

*MF* stands for *More Fragments*. All fragments except the last one have this bit set. It is needed to know when all fragments of a datagram have arrived.

The *Fragment offset* tells where in the current packet this fragment belongs. All fragments except the last one in a datagram must be a multiple of 8 bytes, the elementary fragment unit. Since 13 bits are provided, there is a maximum of 8192 fragments per datagram, supporting a maximum packet length up to the limit of the *Total length* field.

The *TtL (Time to live)* field is a counter used to limit packet lifetimes. It must be decremented on each hop and is supposed to be decremented multiple times when a packet is queued for a long time in a router. In practice, it just counts hops. When it hits zero, the packet is discarded and a warning packet is sent back to the source host. This feature prevents packets from wandering around forever, something that otherwise might happen if the routing tables ever become corrupted.

When the network layer has assembled a complete packet, it needs to know what to do with it. The *Protocol* field tells it which transport process to give the packet to. TCP is one possibility, but so are UDP and some others. The numbering of protocols is global across the entire Internet.

Since the header carries vital information such as addresses, it rates its own checksum for protection, the *Header checksum*. The algorithm is to add up all the 16-bit halfwords of the header as they arrive, using one’s complement arithmetic, and then take the one’s complement of the result. For purposes of this algorithm, the *Header checksum* is assumed to be zero upon arrival. Note that it must be recomputed at each hop because at least one field always changes.

The *Source address* and *Destination address* indicate the IP address of the source and destination network interfaces.

Graphical user interface, text

Description automatically generatedThe *Options* field was designed to provide an escape to allow subsequent versions of the protocol to include information not present in the original design, to permit experimenters to try out new ideas, and to avoid allocating header bits to information that is rarely needed.

**5.6.2 IP addresses**

A defining feature of IPv4 is its 32-bit addresses. Every host and router on the Internet has an IP address that can be used in the *Source address* and *Destination address* fields of IP packets.

It is important to note that an IP address does not actually refer to a host. It really refers to a network interface, so if a host is on two networks, it must have two IP addresses.

Routers have multiple interfaces and thus multiple IP addresses.

* **Prefixes**

A picture containing text

Description automatically generatedIP addresses are hierarchical, unlike Ethernet addresses. Each 32-bit address is comprised of a variable-length network portion in the top bits and a host portion in the bottom bits.

The network portion has the same value for all hosts on a single network.

This means that a network corresponds to a contiguous block of IP address space. This block is called a **prefix**.

IP addresses are written in **dotted decimal notation**. In this format, each of the 4 bytes is written in decimal, from 0 to 255. 80D00297 is written as 128.208.2.151.

Prefixes are written by giving the lowest IP address in the block and the size of the block.

The size is determined by the number of bits in the network portion. This means that the size must be a power of two. By convention, it is written after the prefix IP address as a slash followed by the length in bits of the network portion.

If the prefix (aka a network’s possible addresses) has 2⁸ addresses, then it has up to 2^8 hosts, meaning we need only 8 bits for the host section leaving 24 for the network.

Since the prefix length cannot be inferred from the IP address alone, routing protocols must carry the prefixes to routers. The length of the prefix corresponds to a binary mask of 1s in the network portion. When written out this way, it is called a **subnet mask**.

Hierarchical addresses have significant advantages and disadvantages.

The key advantage of prefixes is that routers can forward packets based on only the network portion of the address, as long as each of the networks has a unique address block. The host portion does not matter to the routers because all hosts on the same network will be sent in the same direction. It is only when the packets reach the network for which they are destined that they are forwarded to the correct host. This makes the routing tables much smaller than they would otherwise be.

While using a hierarchy lets Internet routing scale, it has two disadvantages. First, the IP address of a host depends on where it is located in the network. An Ethernet address can be used anywhere in the world, but every IP address belongs to a specific network, and routers will only be able to deliver packets destined to that address to the network.

The second disadvantage is that the hierarchy is wasteful of addresses unless it is carefully managed. If addresses are assigned to networks in (too) large blocks, there will be (many) addresses that are allocated but not in use

* **Subnets**

Network numbers are managed by a nonprofit corporation called **ICANN (Internet Corporation for Assigned Names and Numbers)**, to avoid conflicts.

The solution is to allow the block of addresses to be split into several parts for internal use as multiple networks, while still acting like a single network to the outside world. This is called **subnetting** and the networks (such as Ethernet LANs) that result from dividing up a larger network are called **subnets**.

When a packet arrives, the router looks at the destination address of the packet and checks which subnet it belongs to. The router can do this by ANDing the destination address with the mask for each subnet and checking to see if the result is the corresponding prefix.

* **CIDR—Classless InterDomain Routing**

Even if blocks of IP addresses are allocated so that the addresses are used efficiently, there is still a problem that remains: routing table explosion.

Routers in organizations at the edge of a network need to have an entry for each of their subnets, telling the router which line to use to get to that network. For routes to destinations outside of the organization, they can use the simple default rule of sending the packets on the line toward the ISP that connects the organization to the rest of the Internet.

Routers in ISPs and backbones in the middle of the Internet have no such luxury. They must know which way to go to get to every network and no simple default will work. These core routers are said to be in the default-free zone of the Internet.

In addition, routing algorithms require each router to exchange information about the addresses it can reach with other routers. The larger the tables, the more information needs to be communicated and processed. Greater range of communication increases the likelihood that some parts will get lost, at least temporarily.

Fortunately, there is something we can do to reduce routing table sizes. We can apply the same insight as subnetting: routers at different locations can know about a given IP address as belonging to prefixes of different sizes.

Here we combine multiple small prefixes into a single larger prefix. This process is called **route aggregation**. The resulting larger prefix is sometimes called a **supernet**, to contrast with subnets as the division of blocks of addresses. It is up to each router to have the corresponding prefix information. This design works with subnetting and is called **CIDR (Classless InterDomain Routing)**,which is pronounced ‘‘cider,’’ as in the drink.

* **NAT—Network Address Translation**

There are addresses that have special meanings.

The IP address 0.0.0.0, the lowest address, is used by hosts when they are being booted. It means ‘‘this network’’ or ‘‘this host.’’ IP addresses with 0 as the network number refer to the current network. These addresses allow machines to refer to their own network without knowing its number (but they have to know the network mask to know how many 0s to include).

The address consisting of all 1s, or 255.255.255.255—the highest address—is used to mean all hosts on the indicated network. It allows broadcasting on the local network. The addresses with a proper network number and all 1s in the host field allow machines to send broadcast packets to distant LANs anywhere in the Internet. However, many network administrators disable this feature as it is mostly a security hazard.

A picture containing text

Description automatically generatedFinally, all addresses of the form 127.xx.yy.zz are reserved for loopback testing.

------

IP addresses are scarce. This scarcity has led to techniques to use IP addresses sparingly. One approach is to dynamically assign an IP address to a computer when it is on and using the network, and to take the IP address back when the host becomes inactive. The IP address can then be assigned to another computer that becomes active.

The problem of running out of IP addresses is not a theoretical one that might occur at some point in the distant future. It is happening right here and right now. The long-term solution is for the whole Internet to migrate to IPv6, which has 128-bit addresses.

To get by in the meantime, a quick fix was needed. The quick fix that is widely used today came in the form of **NAT (Network Address Translation)**.

The basic idea behind NAT is for the ISP to assign each home or business a single IP address (or at most, a small number of them) for Internet traffic.

Within the customer network, every computer gets a unique IP address, which is used for routing intramural traffic. However, just before a packet exits the customer network and goes to the ISP, an address translation from the unique internal IP address to the shared public IP address takes place.

This translation makes use of three ranges of IP addresses that have been declared as private. Networks may use them internally as they wish. The only rule is that no packets containing these addresses may appear on the Internet itself. The three reserved ranges are:

Text

Description automatically generated

Within the customer premises, every machine has a unique address of the form 10.x.y.z. However, before a packet leaves the customer premises, it passes through a **NAT box** that converts the internal IP source address, 10.0.0.1 in the figure, to the customer’s true IP address.

When the reply comes back (e.g., from a Web server), it is naturally addressed to 198.60.42.12, so how does the NAT box know which internal address to replace it with? Herein lies the problem with NAT.

When a process wants to establish a TCP connection with a remote process, it attaches itself to an unused TCP port on its own machine. This is called the **source port** and tells the TCP code where to send incoming packets belonging to this connection. The process also supplies a **destination port** to tell who to give the packets to on the remote side.

Each outgoing TCP message contains both a source port and a destination port. Together, these ports serve to identify the processes using the connection on both ends.

Using the Source port field, we can solve our mapping problem. Whenever an outgoing packet enters the NAT box, the 10.x.y.z source address is replaced by the customer’s true IP address. In addition, the TCP Source port field is replaced by an index into the NAT box’s 65,536-entry translation table. This table entry contains the original IP address and the original source port. Finally, both the IP and TCP header checksums are recomputed and inserted into the packet.

Since the mapping in the NAT box is set up by outgoing packets, incoming packets cannot be accepted until after outgoing ones. In practice, this means that a home user with NAT can make TCP/IP connections to a remote Web server, but a remote user cannot make connections to a game server on the home network. Special configuration or **NAT traversal** techniques are needed to support this kind of situation.

NAT changes the Internet from a connectionless network to a peculiar kind of connection-oriented network.

Having the network maintain connection state is a property of connection-oriented networks, not connectionless ones. If the NAT box crashes and its mapping table is lost, all its TCP connections are destroyed. In the absence of NAT, a router can crash and restart with no long-term effect on TCP connections. The sending process just times out within a few seconds and retransmits all unacknowledged packets. With NAT, the Internet becomes as vulnerable as a circuit-switched network

Despite the issues, NAT is widely used in practice, especially for home and small business networks, as the only expedient technique to deal with the IP address shortage. It has become wrapped up with firewalls and privacy because it blocks unsolicited incoming packets by default. For this reason, it is unlikely to go away even when IPv6 is widely deployed.

**5.6.3 IP version 6**

**IPv6 (IP version 6)** is a replacement design for IPV4 that uses 128-bit addresses; a shortage of these addresses is not likely any time in the foreseeable future. However, IPv6 has proved very difficult to deploy. It is a different network layer protocol that does not really interwork with IPv4, despite many similarities.

IPv6 maintains the good features of IP, discards or deemphasizes the bad ones, and adds new ones where needed. In general, IPv6 is not compatible with IPv4, but it is compatible with the other auxiliary Internet protocols, including TCP, UDP, ICMP, IGMP, OSPF, BGP, and DNS, with small modifications being required to deal with longer addresses.

First and foremost, IPv6 has longer addresses than IPv4. They are 128 bits long. The second major improvement of IPv6 is the simplification of the header. It contains only seven fields (versus 13 in IPv4). This change allows routers to process packets faster and thus improves throughput and delay.

The third major improvement is better support for options. This change was essential with the new header because fields that previously were required are now optional (because they are not used so often).

IPv6 also represents a big advance is in security. Authentication and privacy are key features of the new IP.

* **The Main IPv6 Header**

The *Version* field is always 6 for IPv6 (and 4 for IPv4). During the transition period from IPv4 routers will be able to examine this field to tell what kind of packet they have.

The *Differentiated services* field is used to distinguish the class of service for packets with different real-time delivery requirements. Also, the low-order 2 bits are used to signal explicit congestion indications, again in the same way as with IPv4.

The *Flow label* field provides a way for a source and destination to mark groups of packets that have the same requirements and should be treated in the same way by the network, forming a pseudo-connection. In effect, flows are an attempt to have it both ways: the flexibility of a datagram network and the guarantees of a virtual-circuit network.

The *Payload length* field tells how many bytes follow the 40-byte header. The name was changed from the IPv4 Total length field because the meaning was changed slightly: the 40 header bytes are no longer counted as part of the length (as they used to be).

The *Next header field* lets the cat out of the bag. The reason the header could be simplified is that there can be additional (optional) extension headers. This field tells which of the (currently) six extension headers, if any, follow this one.

The *Hop limit field* is used to keep packets from living forever.

Next come the *Source address* and *Destination address* fields. The size of these are 16-bytes.

Graphical user interface, text, application, table

Description automatically generated

A new notation has been devised for writing 16-byte addresses. They are written as eight groups of four hexadecimal digits with colons between the groups:

8000:0000:0000:0000:0123:4567:89AB:CDEF

Since many addresses will have many zeros inside them, three optimizations have been authorized. First, leading zeros within a group can be omitted (0123 🡪 123). Second, one or more groups of 16 zero bits can be replaced by a pair of colons. Thus, the above address now becomes:

8000::123:4567:89AB:CDEF

Finally, IPv4 addresses can be written as a pair of colons and an old dotted decimal number:

::192.31.20.46

All the fields relating to fragmentation were removed because IPv6 takes a different approach to fragmentation. To start with, all IPv6-conformant hosts are expected to dynamically determine the packet size to use.

In brief, when a host sends an IPv6 packet that is too large, instead of fragmenting it, the router that is unable to forward it drops the packet and sends an error message back to the sending host. This message tells the host to break up all future packets to that destination. Having the host send packets that are the right size in the first place is ultimately much more efficient than having the routers fragment them on the fly.

* **Extension Headers**

Some of the missing IPv4 fields are occasionally still needed, so IPv6 introduces the concept of (optional) **extension headers**.

Table

Description automatically generatedSix kinds of extension headers are defined at present. Each one is optional, but if more than one is present they must appear directly after the fixed header, and preferably in the order listed.

**5.6.4 Internet Control Protocols**

In addition to IP, which is used for data transfer, the Internet has several companion control protocols that are used in the network layer. They include ICMP, ARP, and DHCP. ICMP and DHCP have similar versions for IPv6; the equivalent of ARP is called NDP (Neighbor Discovery Protocol) for IPv6.

* **IMCP—The Internet Control Message Protocol**

The operation of the Internet is monitored closely by the routers. When something unexpected occurs during packet processing at a router, the event is reported to the sender by the **ICMP (Internet Control Message Protocol)**. Each ICMP message type is carried encapsulated in an IP packet.

Table

Description automatically generated

* **ARP—The Address Resolution Protocol**

Although every machine on the Internet has one or more IP addresses, these addresses are not sufficient for sending packets. The question now arises, how do IP addresses get mapped onto data link layer addresses, such as Ethernet?

Let us start out by seeing how a user on host 1 sends a packet to a user on host 2. Let us assume the sender knows the name of the intended receiver. The first step is to find the IP address for host 2. This lookup is performed by DNS.

The upper layer software on host 1 now builds a packet with, for example, 192.32.65.5, in the Destination address field and gives it to the IP software to transmit. The IP software can look at the address and see that the destination is on its own network.

However, it still needs some way to find the destination’s Ethernet address to send the frame. One solution is to have a configuration file somewhere in the system that maps IP addresses onto Ethernet addresses.

A better solution is for host 1 to output a broadcast packet onto the Ethernet asking who owns IP address 192.32.65.5. The broadcast will arrive at every machine on the Ethernet, and each one will check its IP address. Host 2 alone will respond with its Ethernet address (E2). In this way host 1 learns that IP address 192.32.65.5 is on the host with Ethernet address E2. The protocol used for asking this question and getting the reply is called **ARP (Address Resolution Protocol)**. Almost every machine on the Internet runs it.

The advantage of using ARP over configuration files is the simplicity. The system manager does not have to do much except assign each machine an IP address and decide about subnet masks. ARP does the rest.

Various optimizations are possible to make ARP work more efficiently. To start with, once a machine has run ARP, it caches the result in case it needs to contact the same machine shortly. Next time it will find the mapping in its own cache, thus eliminating the need for a second broadcast.

* **DHCP—The Dynamic Host Configuration Protocol**

ARP (as well as other Internet protocols) assumes that hosts are configured with some basic information, such as their own IP addresses.

With **DHCP (Dynamic Host Configuration Protocol)** every network must have a DHCP server that is responsible for configuration. When a computer is started, it has a built-in Ethernet or other link layer address embedded in the NIC (NIC = network interface card, hardware responsible for the communication of a computer inside a network), but no IP address. Much like ARP, the computer broadcasts a request for an IP address on its network. It does this by using a DHCP DISCOVER packet. This packet must reach the DHCP server.

When the server receives the request, it allocates a free IP address and sends it to the host in a DHCP OFFER packet. To be able to do this work even when hosts do not have IP addresses, the server identifies a host using its Ethernet address.

An issue that arises with automatic assignment of IP addresses from a pool is for how long an IP address should be allocated. If a host leaves the network and does not return its IP address to the DHCP server, that address will be permanently lost. After a period, many addresses may be lost.

To prevent that from happening, IP address assignment may be for a fixed period of time, a technique called **leasing**. Just before the lease expires, the host must ask for a DHCP renewal. If it fails to make a request or the request is denied, the host may no longer use the IP address it was given earlier.