4

Basic Internet Architecture

Many design decisions and end-user experiences of multiplayer, networked games derive from the particular nature and characteristics of Internet Protocol (IP) networks. In this chapter we will cover the following core aspects of IP networking:

- ‘Best effort’ service
- IP addressing of hosts and other endpoints in the network
- Transport protocols – Transmission Control Protocol (TCP) and User Datagram Protocol (UDP)
- The difference between unicast, multicast, and broadcast communication
- Networks as meshes of routers and links
- Network hierarchies, address aggregation and shortest-path routing protocols
- Address management – Dynamic Host Configuration Protocol (DHCP), Network Address Translation (NAT) and the Domain Name System (DNS).

Feel free to skip this chapter if you already understand IP networking basics (such as IP addressing, subnets, prefixes, shortest-path routing, the role of routers and routing protocols). This chapter is primarily to refresh your memory and provide a backdrop for the interaction between IP network services and networked games.

We will illustrate IP networking principles with examples based on the current Internet’s core technology, known as IP version 4 (IPv4) [RFC791]. We will review how IP networks come in a variety of sizes, the rationale behind IP addressing, the differences between unicast and multicast packet delivery, the roles of the TCP and UDP transport layer protocols, hierarchies in network routing, and shortest-path routing protocols. (We will not discuss an emerging new version known as IP version 6 (IPv6). IPv6 has broadly similar architectural characteristics and is not covered in this book. Even the most optimistic estimates do not see IPv6 being widely relevant to consumer-based networked games until 2010 or beyond.)

Figure 4.1 attempts to illustrate how end-user applications (such as our favourite networked games) and support services (such as DNS or DHCP, which are rarely exposed to the end user) are layered on top of the basic data transport services provided by an IP network. The *Internet Protocol* is so named because it hides the many underlying technologies that can make up an IP network (such as optical fibre links, microwave links,

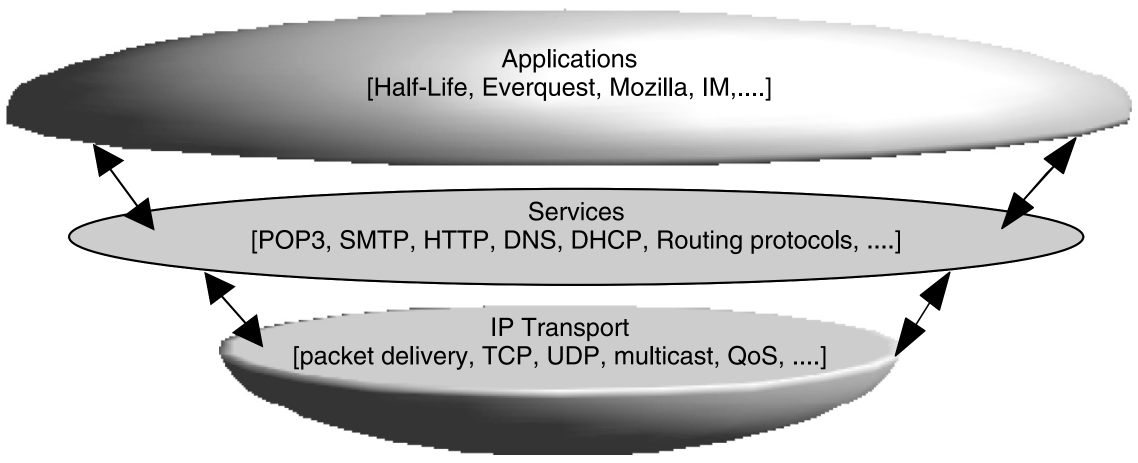
Applications
[Half-Life, Everquest, Mozilla, IM,....]
Services
[POP3, SMTP, HTTP, DNS, DHCP, Routing protocols, ...]
IP Transport
[packet delivery, TCP, UDP, multicast, QoS, ...]

Figure 4.1 End-user applications and services utilise packet transport services provided by the IP network

ethernet Local Area Networks (LANs), 802.11/WiFi networks, cable modems and so on). IP provides a single, global addressing scheme across all the underlying technologies.

4.1 IP Networks as seen from the Edge

From an end user's perspective, there is a reasonable analogy between the postal service and an IP network. With traditional postal service we place letters into envelopes, address them to the final destination, and place them into a local post box. After that, we simply trust the postal service to transport our envelope to its destination in some reasonable time. We neither know nor care how the envelope gets to the destination, the delivery time is measured in days, and we accept that envelopes are sometimes lost. And we implement our own end-to-end strategies to confirm delivery (such as a phone call to the recipient some days later, or reposting a copy of the original letter every few days until the recipient responds). The postal service is a network, and its edges are the post-boxes and letter-boxes where we drop off and pick up our mail.

A simple way to view an IP network is as an opaque cloud with devices attached around the edges. These edge devices may be any piece of hardware (or software) that transmits or receives digital information. The primary goal of an IP network is to provide connectivity, that is, deliver data (in packets) from sources (who transmit packets) to destinations (who receive packets). Devices at the edges of IP networks typically act as both sources and receivers. IP edge devices (or endpoints) are identified with IP addresses, and endpoints are required to look after their own needs for reliable delivery of data. And finally, the IP network is presumed to be totally agnostic towards the actual contents of the packets each endpoint is sending.

Traditionally, IP networks provide few guarantees of timeliness or certainty in packet delivery – usually referred to as *best effort* service (although it might be more aptly considered a 'no guarantees' services). Milliseconds, hundreds of milliseconds, or seconds may elapse between the time a source injects (transmits) a packet into the network cloud, and the destination edge receives that same packet. Sometimes packets simply get lost inside the network and never arrive at all.

The time it takes for a packet to reach its destination is often referred to as *latency*. Short-term variation in this latency from one packet to the next is referred to as *jitter*. Packet loss is often described in terms of a *packet loss rate* – the probability of an IP packet being lost across a certain part of the IP network.

Despite the simplicity of best effort service, IP networks can be quite complex internally. In all but the most trivial networks, there will be multiple internal paths a packet may take between any given source and destination. The network must continually make internal choices as to which path is most optimal at any given time. This is the task of *routing protocols*, which we will discuss later in the chapter.

4.1.1 Endpoints and Addressing

IPv4 endpoints are identified with numerical 32-bit (4 byte) values, conventionally written in *dotted-quad* form – four decimal numbers (representing each of the four bytes making up the IP address) separated by periods. For example, the 32-bit binary address 1000 0000 0101 0000 1100 0101 0000 0111 is written as 128.80.195.7 in dotted-quad form. In theory, this allows for 2^{32} IPv4 addresses. Later in this chapter we will discuss why significantly fewer than 2^{32} IPv4 addresses are available in practice.

Endpoints are also often referred to as *hosts*, although a host (whether a Personal Computer (PC), game console with network port, or any device capable of attaching to a IP network) may have multiple interfaces to an IP network. A host with multiple interfaces (commonly referred to as a *multihomed host*) will have unique IP addresses on each of its interfaces to the IP network. Servers (e.g. web sites or game servers) and clients (e.g. someone running a browser on their home computer) are both hosts on the IP network.

Figure 4.2 shows a simple case where an IP network has three hosts attached, with IP addresses 136.80.1.2, 21.80.1.32, and 142.8.20.8. When host 136.80.1.2 wants to send an IP packet to another host (for example, host 142.8.20.8), it specifies this destination address 142.8.20.8 in the packet and passes the packet to the IP network. The IP network itself worries about how to locate and reach 142.8.20.8.

As a consequence of IP networking's hierarchical routing scheme (described later in this chapter), an IP address is tied closely to where the host is attached in the network.

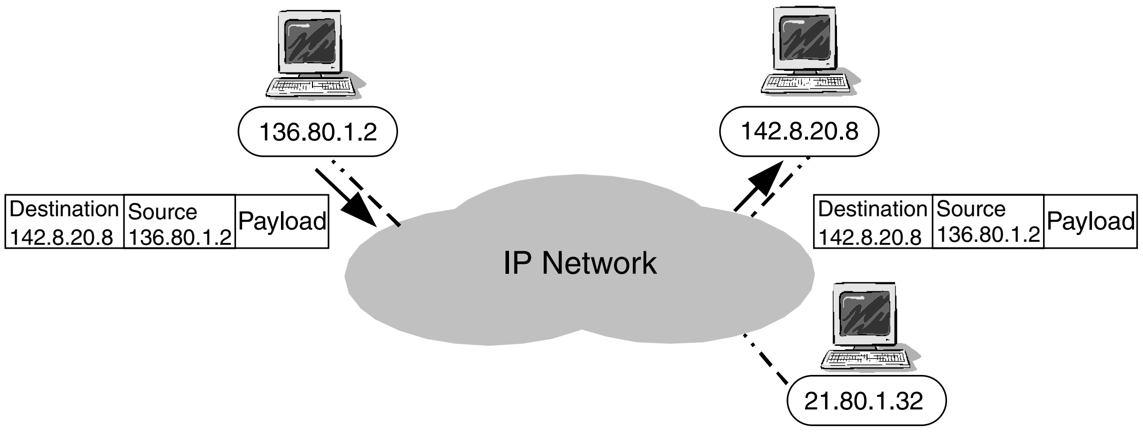

Figure 4.2 An IP Network – opaque cloud with attached devices – looks after getting packets to their destinations

In other words, an IP address represents both the identity of the attached host and the host's 'location' on the network. (This location is topological rather than geographical. It reflects where the host exists within the interconnections of IP networks and service providers that make up the Internet.)

IP addresses are closely related to, but not the same as, Fully Qualified Domain Names (FQDNs, or simply 'domain names'). Domain names (often imprecisely referred to as Internet addresses) are textual addresses of the form 'www.gamespy.com', 'www.freebsd.org' or 'www.bbc.co.uk'. Domain names must be resolved into IP addresses using the Domain Name System (DNS). Endpoint applications typically hide this translation step from the user, and use the resulting numeric IP address to establish communication with the intended destination. (We will discuss the DNS in greater detail later in this chapter.)

4.1.2 Layered Transport Services

Most game developers will utilise IP in conjunction with either the TCP [RFC793] or UDP [RFC768]. TCP and UDP are *transport* protocols, designed to provide another layer of abstraction on top of the IP layer's network service. Both TCP and UDP support the concurrent multiplexing of data from multiple applications onto a single stream of IP packets between two IP hosts. TCP additionally provides reliable delivery on top of the IP network's best effort service.

4.1.2.1 Transmission Control Protocol (TCP)

Early Internet applications – such as email, file transfer protocols and remote console login services – were sensitive to packet loss but relatively insensitive to timeliness (everything sent had to be received, but delays from tens of milliseconds to a few seconds were tolerable). The common end-to-end *transport* requirements of such applications (reliable ordered transfer of bytes from one endpoint to another) motivated development of TCP.

TCP sits immediately above the IP layer within a host (see Figure 4.3), and creates bidirectional paths (sometimes referred to as *TCP connections* or *TCP sessions*) between endpoints. An application's outbound data is broken up and transmitted inside TCP frames, which are themselves carried inside IP packets across the network to the destination. The

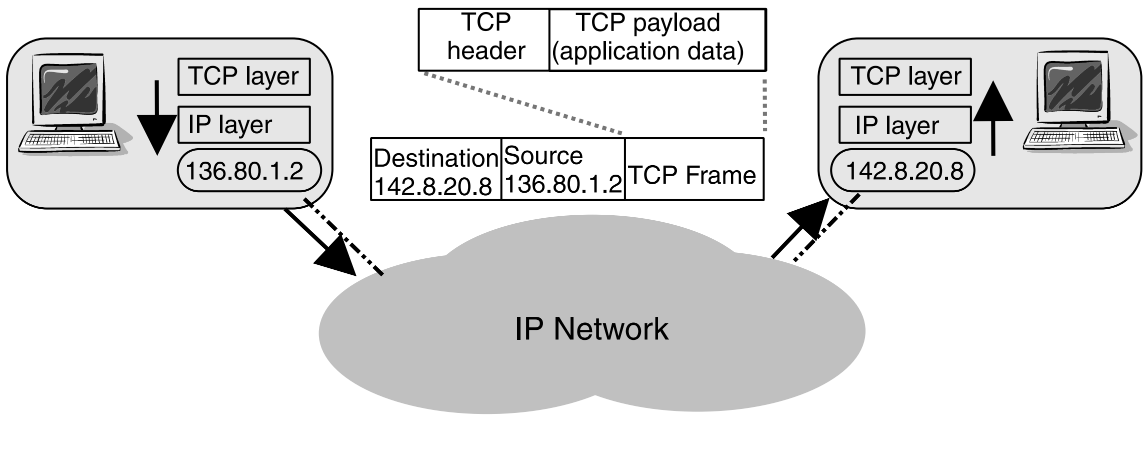

Figure 4.3 TCP runs transparently across the IP network

destination host's TCP layer explicitly acknowledges received TCP frames, enabling the transmitting TCP layer to detect when losses have occurred. Lost TCP frames are retransmitted until acknowledged by the destination, ultimately ensuring that the application's data is transferred with a high degree of reliability.

TCP uses *windowed flow control* to regulate how fast it transmits packets through the network. The current window size dictates how many unacknowledged packets may be in transit across the network at any given time. The source grows its window as packets are transmitted successfully, and shrinks its window when packet loss is detected (on the assumption that packets are only lost when the network is briefly overloaded). This regulates the bandwidth consumed by a TCP connection. Flow control and retransmission are handled independently in each direction.

Because TCP may keep retransmitting for many seconds when faced with repeated packet loss, the end application can experience unpredictable variations in latency (Figure 4.4). Thus, TCP is generally not suitable as the transport protocol for real-time messaging during game play of highly interactive networked games.

4.1.2.2 User Datagram Protocol (UDP)

UDP is a much simpler sibling of TCP, providing a connectionless, unreliable, datagram-oriented transport service for applications that do not require or desire the overhead of TCP's service. UDP imposes no flow control on packet transmission, and no packet loss detection or recovery. It is essentially a multiplexing layer sitting directly on top of IP's best effort service. As such an application using UDP will directly experience the latency, jitter and loss characteristics of the underlying IP network.

4.1.2.3 Multiplexing and Flows

Extending the postal service analogy a little further, while the IP address is analogous to a street address both TCP and UDP add the notion of *ports* – additional identification analogous to an apartment number or hotel room number. Each TCP or UDP frame carries two 16 bit *port numbers* to identify the source and destination of their frame within the context of a particular source or destination IP host. This allows multiplexing of different traffic streams between different applications residing on the same source and destination IP endpoints.

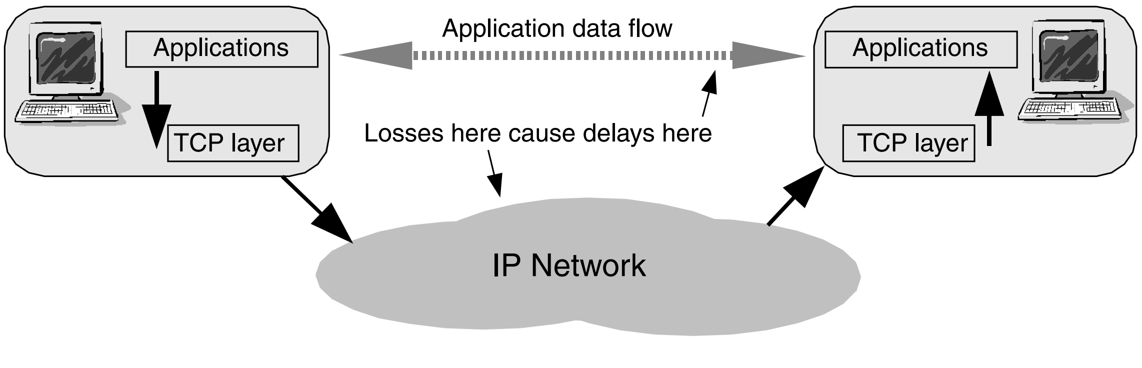

Figure 4.4 TCP converts IP layer packet loss into application layer delays

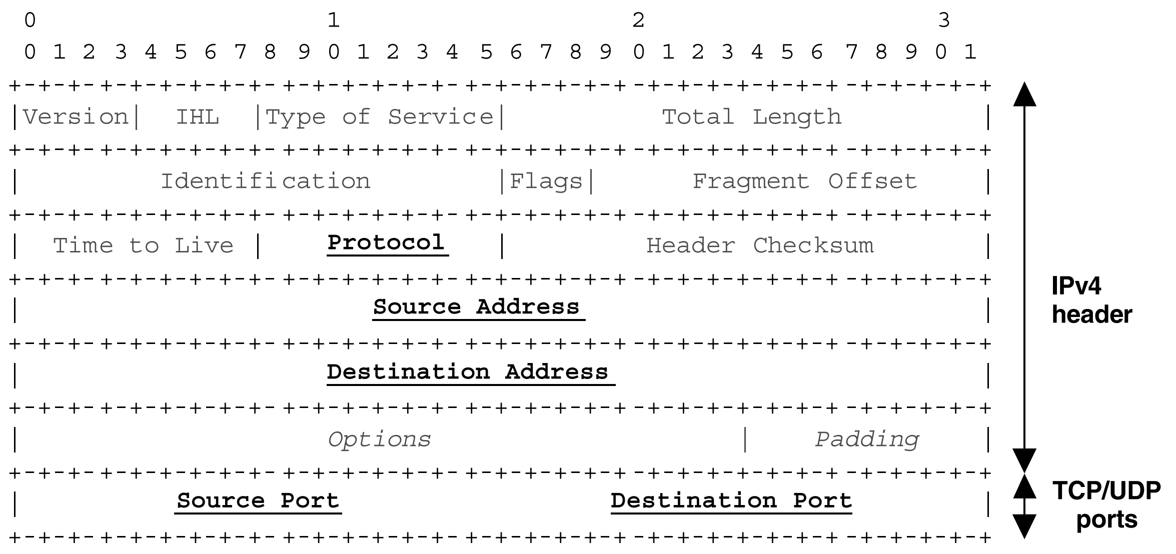

Figure 4.5 Header fields of interest in IPv4 packets

IP address and port number combinations are often written in the form '*ip-address:port*', with a ':' separating the address (either in dotted-quad or fully qualified domain name form) and the numerical port number.

Figure 4.5 shows the key fields of an IPv4 header and the first 32 bits of the TCP or UDP transport header. The protocol field specifies whether the IP packet carries TCP (protocol 6), UDP (protocol 17), or some other type of frame (discussed further in 'Directory of General Assigned Numbers' [IANAP]). The source and destination addresses identify a packet's source and destination host at the IP level.

Taken together, port numbers and IP addresses uniquely identify the source and destination applications that are generating and consuming the traffic. A sequence of packets exchanged between the same TCP or UDP ports on the same two endpoints is often referred to as an *application flow* (or just *flow*). Many applications use 'well-known' port numbers, often making it possible to infer the identity of an application from the source or destination port numbers. For example, the Simple Mail Transport Protocol (SMTP) typically uses TCP to port 25 on the mail server host [RFC2821], Quake III Arena servers default to using UDP port 27960, Half-Life 2 servers default to using UDP port 27015 and web servers typically respond to Hypertext Transport Protocol (HTTP) traffic on TCP port 80 [RFC2616].

Note that there are no rules preventing applications from using unconventional ports – we could, for example, just as easily run Quake III Arena on port 27015 and Half-Life 2 on port 27960, so long as everyone knows what is happening.

4.1.3 Unicast, Broadcast and Multicast

Sending a packet to a single destination is known as *unicast* transmission. Sending a packet to all destinations (within some specified region of the network) is known as *broadcast* transmission. Broadcasting may be implemented as multiple separate unicast transmissions, but this requires the source to actually know the IP addresses of all intended destinations. Usually the network supports broadcast natively – the source sends a single

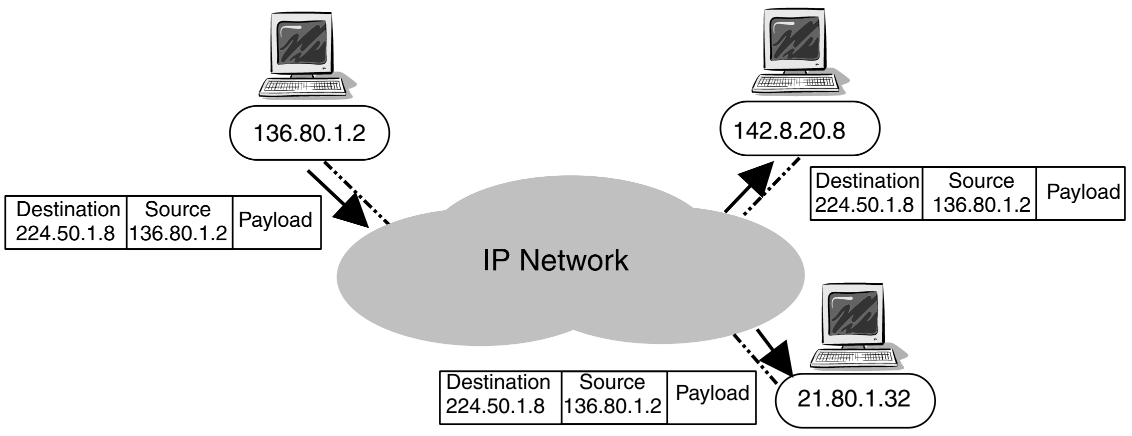

Figure 4.6 IP multicasting replicates a single packet to (potentially multiple) group members

packet into the network with a specific 'broadcast' destination address, and the network itself replicates the packet to all attached hosts within a restricted region.

A little-used alternative is *IP multicast* [RFC1112]. A source transmits one packet and the network itself delivers identical copies to multiple destinations (known as a *multicast group*, identified by special 'class D' IP destination addresses). Hosts explicitly inform the network when they wish to join or leave multicast groups. (Broadcast can be considered a special case of multicast, where every endpoint within a specific region of the network is automatically considered to be a group member.)

In IPv4, addresses in the range 224.0.0.0 to 239.255.255.255 are class D addresses, and represent multicast groups. Sources indirectly specify group members by using a class D address in their packet's destination address field.

Two attractive qualities of IP multicast are that a source does not need to track the multicast group members over time, and a source only sends one copy of each packet into the network. The network itself tracks group members and performs the necessary packet replications and deliveries.

Figure 4.6 shows a packet being sent to a group identified only by the destination address 224.50.1.8. Only the network is aware that the group includes endpoints 142.8.20.8 and 21.80.1.32. IP multicast is an 'any to many' service – a multicast group can have many members, and anyone can transmit to a multicast group from anywhere on the IP network (even if they are not a member of the group).

IP multicast holds some promise as a mechanism for efficiently delivering content, that is intended for concurrent delivery to multiple recipients. For example, replicating common game state across multiple clients or servers. Unicast requires a source to transmit its packets multiple times (once for each recipient), while multicast requires only one packet per update. However, because of the internal complexity required to support IP multicast there is little support in most public IP networks. This makes IP multicast difficult to use in networked games beyond specially constructed private networks.

4.2 Connectivity and Routing

From the game developer's perspective, it is often not necessary to understand the internal structure of IP networks. It is usually sufficient to comprehend the network's behaviour as seen from the edges. However, it is valuable to reflect on the internal details if you wish to more fully understand the origins of IP addressing schemes, latency, jitter, and packet loss.

An IP network is basically an arbitrary topology of interconnected links and routers. These terms are often thrown around casually, so we will define them here as follows:

- Links provide packet transport between routers.
- Routers are nodes in the topology, where packets may be forwarded from one link to another.

Upon receipt of an IP packet the router's primary job is to pick another link (the *next-hop* link) on which to forward (transmit) the packet, and then to do so as quickly as possible. Except for simple networks a router will usually have more than one possible choice of the next-hop link. Routers implement routing protocols to continuously exchange information with each other, subsequently learning the network's overall topology and agreeing on the appropriate next hops for all possible destinations.

This approach is known as *hop-by-hop forwarding*:

- An independent next-hop choice is made at each router.
- Each next-hop choice usually depends solely on the packet's destination address field.
- Routing protocols ensure that the network's routers agree on a coherent set of next-hop choices for all possible destinations.

Consider the network in Figure 4.7 where multiple paths exist between 136.80.1.2 and 21.80.1.32. Router R1 could send the packet to R2 or R3, both of which have the capability to forward the packet even closer to 21.80.1.32. In this example, R1 decides to use R2 as the next hop toward 21.80.1.32, and R2 has decided to use R5 as its next hop toward 21.80.1.32.

An IP network provides a *connectionless* service because it can transport IP packets from source to destination without any *a priori* end-user signalling. However, it is not *stateless*. The set of all source-to-destination paths currently considered optimal by the routing protocols is the state of the entire network.

In the rest of this section, we will look at how network hierarchies and address aggregation have been used to minimise the amount of state information that routing protocols need to handle. We will also touch on some routing protocols used in the Internet today.

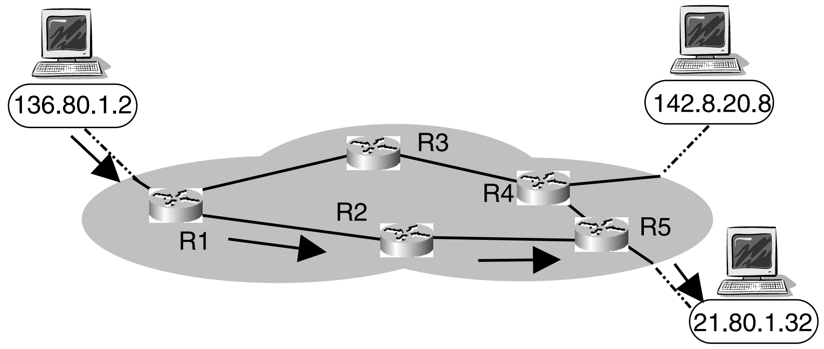

Figure 4.7 An arbitrary topology of routers may have multiple next hops

4.2.1 Hierarchy and Aggregation

The following issues are all closely related.

- IP address formats
- The association of IP addresses to endpoints
- How IP routing protocols establish appropriate paths?
- How routers make their next-hop forwarding decisions?

For small networks, it might seem reasonable for every router to simply know the identity and location of every endpoint. In practice, this approach is unworkable, as real networks may have thousands or tens of thousands of endpoints. Considering the many millions of hosts on the Internet itself, it is clearly impossible to expect routers (having only finite memory and processing capacity) to know all possible destinations.

The solution has been to introduce hierarchy into the IP address space – one that maps closely related IP addresses onto topologically localised sets of actual IP endpoints. This hierarchy allows routers to carry summarised information for regions of the network further away from them, and increasingly more detailed information for closer regions of the network. Hierarchy also creates sparseness of address allocation (consequently, far less than 2^{32} IP addresses can actually be allocated).

4.2.1.1 Class-Based Hierarchy

The IPv4 unicast address space was originally blocked into three classes – A, B and C (see Figure 4.8) [RFC791]. Specific combinations of an address' most significant 3 bits identified an addresses class. The next most significant 7, 14 or 21 bits of the IP address represented a *network number*. The Internet itself (at the time known as *ARPAnet*) was modelled as a backbone (a network of routers) with multiple independent networks directly attached. Each attached network was assigned a specific class A, B or C network number. Endpoints (hanging off each network) had their IP addresses constructed from their network's class bit(s), network number bits and a locally significant value for the remaining 24, 16 or 8 host bits. A router could easily determine which part of a packet's destination address represented the destination network, because the class of an IP address was encoded in the top 3 bits.

However, this class structure was particularly wasteful of address space. Many companies or institutions with more than 254 hosts had to obtain multiple class C networks (filling the backbones router tables) or a single class B (which would be barely utilised). In response, the Internet Engineering Task Force (IETF) developed Classless Inter Domain Routing (CIDR) in the early 1990s.

Class	Address Format in Binary	Networks	Hosts
A	0nnnnnnn.hhhhhhhh.hhhhhhhh.hhhhhhhh	2^{7} nets	2^{24}hosts
B	10nnnnnn.nnnnnnnn.hhhhhhhh.hhhhhhhh	2^{14} nets	2^{16} hosts
C	110nnnnn.nnnnnnnn.nnnnnnnn.hhhhhhhh	2^{21} nets	2^{8} hosts

Figure 4.8 Early IPv4 space divided into fixed-size classes

4.2.1.2 Classless Inter Domain Routing

CIDR replaced the previous A, B and C class rules (hence, *classless*) [RFC1519] with a flexible value/prefix-size pair scheme for identifying networks – the network number is encoded in the top bits of a 32-bit value, and the number of valid bits in the network number indicated by an integer prefix-size (Figure 4.9). Figure 4.9 shows that, in general, a prefix size of X results in a network that can theoretically contain up to $2^{(32-X)}$ endpoints.

A key benefit of CIDR was that variably sized networks could now be built from the old class C space. For example, 192.80.192/22 represents a single network with a 22-bit prefix and a network number of 192.80.192 – equivalent to four contiguous class C networks (192.80.192.*, 192.80.193.*, 192.80.194.* and 192.80.195.*, where '*' represents any number between 0 and 255). In other words, it represents a single '/22' network prefix in the backbone routers rather than four class C prefixes.

In the absence of CIDR, the last class B address would have been assigned in early 1994. CIDR significantly slowed the growth rate of the backbone routing tables, and increased the density with which IP addresses could be packed into a 32-bit field.

4.2.1.3 Subnetting

Creating a single network from multiple old class C networks is known as *supernetting*. The reverse, creating hierarchy within individual networks, is known as *subnetting*. Groups of endpoints may be aggregated into subnetworks (commonly referred to as *subnets*) if they are topologically localised within the scope of a larger network. Individual subnets contain endpoints whose addresses all fall under a common prefix (or *subnet mask*), a prefix that is itself a subset of the class or CIDR prefix assigned to the network of which they are a part. Subnets are networks within networks that can be described by a longer (that is, more precise) prefix or mask than the one that describes the network itself.

IP subnets are the lowest level of the IP routing and addressing hierarchy. Routing protocols do not concern themselves with local details within subnets. In all except the most simplistic network topologies, routers are needed in order to forward packets between subnets.

Layer 2 links between routers, such as Ethernet or similar LANs, are also often referred to as *subnets*. However, while multiple IP subnets may run over a single link, an IP subnet cannot (by definition) span more than one link without an intervening router.

Consider Figure 4.10, where Network 1 is made up of two internal subnets. The network's public identity (as advertised to the IP backbone's routers) is 128.80.0.0/16. Internally, Network 1 has two subnets – each with a longer, more precise 24-bit prefix (a subnet mask of 255.255.255.0). Subnet 1 covers all addresses in the range 128.80.1.0 to 128.80.1.255, whereas subnet 2 covers addresses in the range 128.80.9.0 to 128.80.9.255.

Subnet 1 and 2 may be geographically separate from each other yet owned by a common administrative entity (for example, a large company). Router R1 only advertises a

```
 Address Format (binary)                         Networks and Hosts
 nnnnnnnn.nnnnnnnn.nnnhhhhh.hhhhhhhh
|< - - - X - - - - >|                            2X nets, 2(32-X)
   hosts
```

Figure 4.9 CIDR relaxes the Network Prefix Lengths

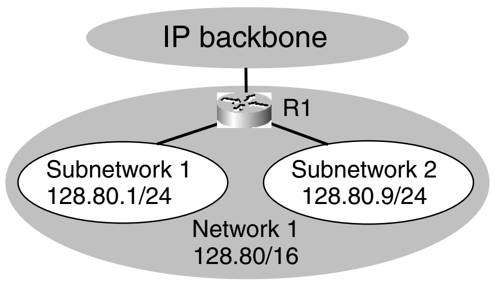

Figure 4.10 Subnetting allows aggregation within a network

single prefix (128.80.0.0/16) to the outside world, and takes care of forwarding packets to whatever subnets have been internally carved from the 128.80.0.0/16 address space.

Subnets may themselves be internally subnetted, with increasingly longer prefixes. Taken to an extreme, a subnet may map directly to a single link and have only two members (the IP interfaces at either end of the link).

The IPv4 address 255.255.255.255 is a special address meaning 'broadcast to all hosts on the local subnet'. Packets to 255.255.255.255 are never forwarded beyond the IP subnet on which they originate. A more general form, known as the *directed broadcast* address, is constructed by setting the host part of an IP address to ones. For example, you could transmit a packet to members of subnet 128.80.1.0/24 by using a destination address of 128.80.1.255. Because of the potential for remotely triggered mischief, routers are often set to filter out directed broadcast packets.

4.2.2 Routing Protocols

Network topologies change frequently, may be due to human interventions or the usual unpredictable failures that bedevil any large-scale system. Routing protocols must perform a number of tasks such as the following in a timely manner:

- Dynamically discover a network's topology, and track the topology changes that occur from time to time.
- Build shortest-path forwarding trees.
- Handle summarised information about external networks, possibly using different metrics to those used in the local network.

The Internet uses distributed routing protocols, which push topology discovery and route calculation processes out into every router. Since the processing load is shared across all routers, sections of the network can continue to adapt locally to changing conditions even if they become isolated from the rest of their network.

Figure 4.11 illustrates how every router participates both in forwarding packets (on the basis of previously calculated rules) and in performing distributed routing calculations (updating the forwarding rules as necessary).

The detailed art of IP routing is beyond the scope of this book, so we will only briefly summarise a few routing protocols used in the Internet.

4.2.2.1 Shortest-Path Routing

When multiple paths exist between a source and a destination, IP networks use *shortest-path* routing to pick one particular path. The 'length' of a path is typically measured in

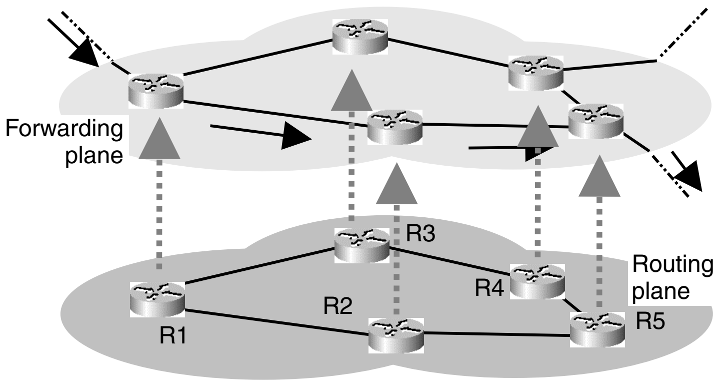

Figure 4.11 IP routing conceptually consists of separate forwarding and routing functions within each router

terms of hops (the number of routers or links through which the packet passes) but may be defined using any metric desired by the network operator. The path with the lowest 'sum of metrics over the entire path' is the shortest path. (for example, in Figure 4.7 the path through R1, R2 and R5 is the shortest path from 136.80.1.2 to 21.80.1.32 when measured by number of router hops.)

Metrics may reflect physical characteristics such as available bandwidth (lower weighting typically given to links with more bandwidth), link delays (higher weighting typically given to links with higher delay) and link costs; or weights may simply represent the administrator's relative preference for traffic to be on a particular link.

The results of a router's shortest-path calculations are stored as a set of forwarding rules in a *forwarding table*, sometimes also referred to as a F*orwarding Information Base* (FIB). Forwarding rules specify the appropriate next-hop destinations for packets matching various combinations of network/prefix pairs.

To ensure that routers always utilise the most precisely specified path, they are required to implement a *longest prefix match* when forwarding packets. In essence, the forwarding table's entries must be searched for the entry with the longest prefix that matches a packet's destination. The entry thus discovered is the correct next hop.

The routing protocol may also choose to use (or be required to account for) two extreme network/prefix pairs – *default routes* and *host routes*. Default routes are represented by the network/prefix 0.0.0.0/0 – a guaranteed match to any IP address. Because the prefix length is zero, this route is the last entry in a router's forwarding table.

Default routes are the ultimate in aggregation – if there is only one next-hop link out of the local network, a default route entry can point to that link (instead of having explicit forwarding rules for all the network/prefix pairs that can be reached in the world outside the local network). For example, in Figure 4.7 router R1 would have specific routes pointing into Network 1 for destinations under 128.80/16, and a default route entry pointing out toward the IP Backbone.

Host routes are represented by the network/prefix w.x.y.z/32 – a rule that only matches packets specifically destined for endpoint w.x.y.z. Host routes are discouraged because they are very difficult to aggregate and therefore can consume disproportionate amounts of memory resources in routers throughout the network.

Each destination prefix (whether a network, subnet, or actual host) known to the local network's routing protocol is said to be the root of its own particular *shortest-path tree*.

The tree has branches passing through every router in the network, although not all links in the network are branches on every shortest-path tree. No matter where a packet appears within the network, the packet will find itself on a branch of a shortest-path tree leading toward its desired destination.

The challenge for a dynamic IP routing protocol is to keep these shortest-path trees current in the presence of router failures, link failures or deliberate modifications to the network's topology – link failures usually require recalculating the shortest-path trees for many destination prefixes.

4.2.2.2 Autonomous Systems (ASs)

IP networks contain one additional component to their hierarchy – the Autonomous System (AS). An AS is defined loosely as a self-contained, independently administered network or internally connected set of networks. Larger networks, including the Internet itself, can be viewed as an arbitrary topology of interconnected ASs.

The AS exists to provide a bounded scope over which any given routing protocol must track internal topology. Within an AS, routing is managed by Interior Gateway Protocols (IGP, gateway being an old name for routers). Routing of traffic between ASs is managed by an Exterior Gateway Protocol (EGP). The IGP can focus on specific and detailed routes and destinations within the AS, while the EGP deals with summarised information about the AS and networks that can be reached through the AS.

4.2.2.3 Interior Gateway Protocols

Routing protocols that operate within ASs are called *IGPs*. *Distance Vector* (DV) and *Link State* (LS) algorithms have both been used as the basis of IGPs. DV algorithms tend to be simpler to implement, while LS algorithms are more robust when faced with regular topology changes within a network.

DV algorithms require each router to advertise to its neighbours information about the relative distance to each network the router knows (a vector of distances). A router may receive multiple advertisements for the same network X, each from a different neighbour, in which case the router remembers the advertisement with the lowest distance. The neighbour advertising the lowest distance toward X becomes the next hop for packets heading to any destination within network X. The advertising process (typically with intervals in the tens to hundreds of seconds) ensures that information about new networks, or new distances to existing networks, ripples out across the local network whenever changes occur.

LS algorithms distribute maps of the local network's entire topology (along with the state and metrics of all the links in the topology). The maps are distributed by flooding LS advertisements, whereby each router informs its neighbours about sections of the network topology that the local router knows about. When a state change occurs (for example, a link goes up or down, or a new route is associated with an existing link), the new LS information is flooded across the local network to ensure that all router's have up-to-date LS maps.

Each router then uses LS maps to locally calculate shortest-path trees to all listed destination networks, and hence determine the appropriate next hops out of the router itself. Because the next-hop calculations are based on complete knowledge of the network's state, every router can be expected to agree on the shortest-path trees.

Although DV protocols are simple to describe and implement, a network's shortest-path trees can get tangled up in transient loops while DV algorithms converge after topology changes. (Slow convergence is a fundamental limitation of any scheme in which the local router has only second hand, interpreted information about the nature of the network beyond the router's local interfaces.)

LS protocols are more complex than DV protocols. They contain two separate functions – maintenance of a distributed LS database and stand-alone shortest-tree calculation. The shortest-path trees can be assured to be loop free almost immediately after any LS changes occur and the information is flooded throughout the network. However, the network-wide flooding of state changes also limits the scalability of LS-based networks.

Two examples of DV IGPs are the Routing Information Protocol (RIPv2) [RFC2453] and IGRP (a proprietary IGP from Cisco Systems, which became Enhanced IGRP, EIGRP). Two examples of LS IGPs are Open Shortest Path First (OSPF) [RFC2328] and Intermediate System to Intermediate System (IS–IS).

4.2.2.4 Exterior Gateway Protocols

Border Gateway Protocol (BGP) version 4 is the standard EGP used in the Internet today [RFC1771]. BGP-4's primary role is to distribute information between ASs indicating where all the constituent networks are located. Every AS has one or more routers that interface to a peer AS – these are the *border routers* for the AS. Each border router runs an instance of BGP-4, enabling them to distribute to their neighbouring ASs information about the reachable networks within the local AS.

BGP-4 is a *path vector* protocol, which borrows a number of key DV concepts. In path vector, each border router advertises not only the existence of a path to particular networks (*reachability*) but also the list of ASs through which the path passes. Any given border router can confirm that an advertisement for a given network is loop free if the border router's own AS number does not already appear in the path vector. After an advertisement is accepted, the local border router inserts its own AS number into the path vector before readvertising the reachability information to its neighbours.

It is beyond the scope of this book to describe BGP's mechanisms to control the scope of reachability advertisements, support relative priorities between different inter-AS paths, and support policies that may restrict the ASs through which certain traffic can be routed.

4.2.2.5 Backbones and Routing Policies

There is no single backbone in today's Internet. Instead, we have a number of peer backbones. Top-level backbones typically interconnect only at a few geographically diverse points – Network Access Points (NAPs) or Internet exchanges (IX) – to ensure that any point on the Internet can connect to any other. Backbones interconnect at multiple points, providing redundancy against failure and potentially shortening many end-to-end paths.

However, political, geographical and/or commercial reasons mean that not all backbones wish to directly interconnect, even if physically possible. BGP-4 allows operators to constrain the advertisement of AS reachability in accordance with *routing policies* that reflect each operator's political or business agreements. As a result, IP paths may be convoluted simply because the source and destination are connected to different backbones that have no agreement to directly exchange traffic. Two geographically close sites (for example, London and Paris, Sydney and Melbourne, or Los Angeles and San Francisco) might find themselves communicating over paths that loop through New York, Tokyo, or Amsterdam depending on their choice of backbone provider and where the backbone providers interconnect.

4.2.3 Per-hop Packet Transport

This section reviews how an individual IP interface (whether on a router or an endpoint) uses the services of an underlying link to get IP packets to the appropriate next hop.

4.2.3.1 Link Layer Networks

For most of this chapter links have been treated as simple, point-to-point paths with only two interfaces attached. In reality, links are often networks in their own right. LAN (such as Ethernet) and wide area networks (such as frame relay) are examples of the link layers that support multiple attached devices. Devices attached to a link layer network may support IP, some other services, or a mixture of both.

Two addresses are associated with an IP interface attached to a given link layer network:

- The interface's IP address (representing the interface's identity in the IP topology).
- The interface's link layer address (representing the interface's specific identity in the context of the underlying link layer network).

In general, the link layer network is unaware of the IP address assigned to any given IP interface attached to the link. An IP packet's next hop (expressed as an IP address in a router's forwarding rules) must be translated to a link layer address before packet transmission can occur across the link.

Consider Figure 4.12, where a single Ethernet LAN [8023] has three attached interfaces, belonging either to routers or hosts on the 136.80.1/24 subnet. At the IP level, Interface 1 is known as 136.80.1.2, Interface 2 is known as 136.80.1.5, and Interface 3 is known as 136.80.1.9. Yet the Ethernet LAN only knows these interfaces by their 48-bit (6 byte) 'Media Access Control' (MAC) addresses (in this example MAC.1, MAC.2, and MAC.3, respectively).

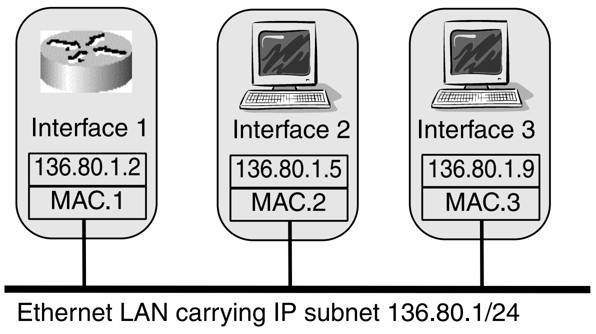

Figure 4.12 Each interface on an Ethernet LAN has both IP and Ethernet addresses

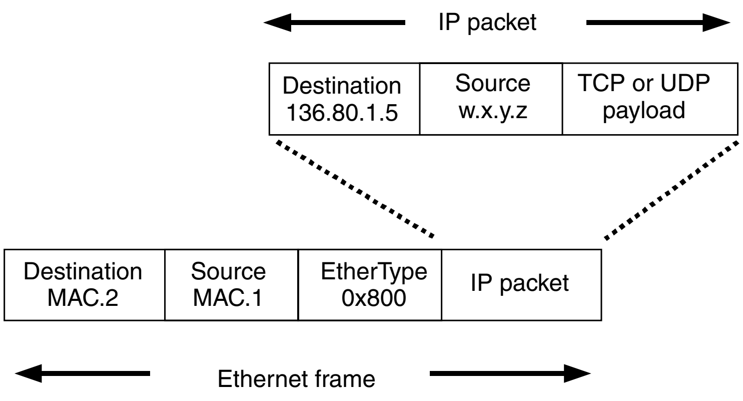

Figure 4.13 IP packets from the router at 136.80.1.2 to the host at 136.80.1.5 are encapsulated for transmission between Ethernet addresses MAC.1 and MAC.2

For example, if Interface 1 belonged to a router whose forwarding rules had just decided 'send this packet to the interface identified as 136.80.1.5', the following two steps would be executed

- A mapping would be established from 136.80.1.5 to MAC.2, the link layer address of Interface 2.
- The IP packet would be sent from MAC.1 to MAC.2 inside a suitably constructed Ethernet frame.

Figure 4.13 shows a highly simplified picture of how an IP packet from outside the 136.186.1/24 subnet would be encapsulated inside an Ethernet frame to be sent from Interface 1 to Interface 2. Ethernet frames carry an ethernet protocol type code – EtherType – of 0x800 to identify the payload as an IP packet (or more precisely, an IPv4 packet).

4.2.3.2 Address Resolution

Next-hop IP addresses are mapped to link layer addresses in a process referred to as *address resolution*. Address resolution may occur using information that is manually configured or is dynamically discover on-demand.

Manual configuration is unwieldy in all but the simplest of static network configurations. Most routers and hosts implement a dynamic Address Resolution Protocol (ARP) to identify what link layer address is associated with a particular IP address. ARP allows IP interfaces to move from one link layer interface to another without manually reconfiguring all the other interfaces attached to the link. This can be useful when, for example, an Ethernet card is replaced on a host or router – the Ethernet address changes, and the dynamic ARP process will ensure other interfaces on the link soon learn the new mapping. Figure 4.14 attempts to capture how ARP is both a peer of, and a service for, the IP layer.

Interfaces keep current address mappings in a local cache, an *ARP table*, which is searched each time a packet is transmitted. If a mapping exists, it is used. Otherwise the ARP is executed to discover the desired mapping. To ensure that old or incorrect mappings are regularly refreshed, cached ARP table entries are deleted after a period of time.

Each link layer technology has its own ARP mechanism. Some examples are ARP for IP over FDDI [RFC1390] and IP over ATM [RFC2225]. Perhaps the longest serving example

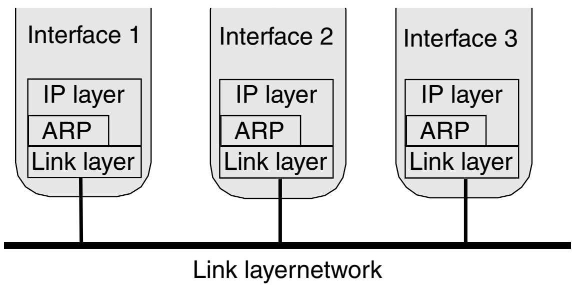

Figure 4.14 ARP is both peer of, and service for, the IP layer. Both sit over the Link Layer

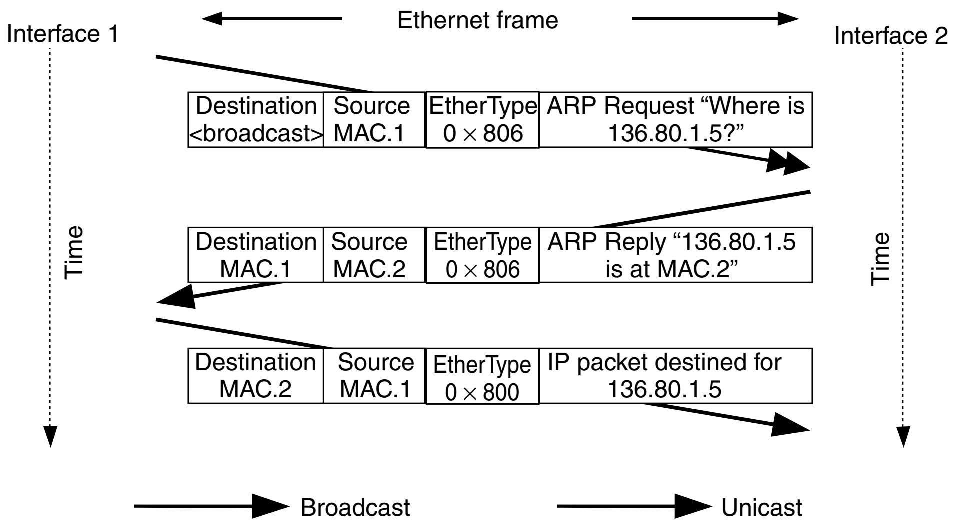

Figure 4.15 Frame sequence when initially sending an IP packet from Interface 1 to Interface 2

is ARP for Ethernet [RFC826], usually just referred to as *ARP*. When an IP address cannot be located in the local interface's ARP table, the interface issues a broadcast *ARP Request* on the LAN, essentially asking anyone else if they know the mapping. (Transmitting the ARP Request to the Ethernet broadcast address ensures that all attached devices are reached without the local interface needing to know who is, or is not, attached to the LAN at any given time.) Usually, the target (the interface whose IP address is being queried) will respond with a unicast *ARP Reply* containing the requested IP to Ethernet address mapping.

Figure 4.15 shows the packet exchange that would have occurred in the example of Figure 4.12 if Interface 1's local ARP table did not have a mapping for 136.80.1.5. (An EtherType 0x806 indicates an ARP Request or Reply, while EtherType of 0×800 indicates an IPv4 packet.) The sequence would be as follows:

1. Interface 1 transmits a broadcast ARP Request for 136.80.1.5.
2. Interface 2 unicasts back an ARP Reply (it knows the Ethernet address of Interface 1 from the initial ARP Request)
3. Interface 1 unicasts the IP packet to Interface 2.

For the curious reader: If you are running Windows XP or similar, entering *arp -a* in a console window will show the current local ARP cache entries. Under many versions of Unix (for example, FreeBSD or Linux) the command *arp -an* will show the current local ARP cache entries.

4.2.3.3 Time to Live

Transient errors in router forwarding tables can sometimes create loops in the IP network, known as *routing loops*. Routing loops tend to occur shortly after topology changes, while the network's routing protocol converges on a new set of shortest-path trees. Routing loops often act like black holes in the network – packets head into the region of the routing loop, and then get stuck, consuming bandwidth as they circulate. Under extreme circumstances, the routing loop can disrupt the routing protocol itself, by saturating links carrying routing protocol update messages.

To prevent endless looping, IP packets carry an 8-bit *Time to Live* (TTL) field (see Figure 4.5). In practice, the TTL represents 'hops to live' – a limit on the maximum number of router hops a packet can traverse before it expires in transit. A packet's TTL field is set to a nonzero value by the source, and is decremented by one every time the packet passes through a router. The packet is discarded when its TTL field is decremented to zero (whether or not it has reached its final destination).

If the source sets the TTL too low, some distant regions of the Internet may become unreachable. (Indeed, there were examples of this occurring with a popular PC operating system's default TTL in the early 1990s as the Internet became more topologically convoluted.) Setting the TTL too high increases the potential disruption a source's packets can cause during routing loops (by increasing the length of time the packets stay in transit around the loop). Many operating systems today set their initial TTL to 64 or some multiple of 32 above that (more by historical quirk than any particular mandatory requirement).

4.2.3.4 Maximum Transmission Units and IP Fragmentation

Although in principle, IP packets may be as large as 64 K bytes, most link layer technologies impose a substantially smaller limit on link level frame size. For example, Ethernet imposes a limit of 1500 bytes on the size of IP packets that can be carried in an Ethernet frame.

The underlying link layer's frame size limit is reflected at the IP layer by a parameter known as the *Maximum Transmission Unit* (MTU). When forwarding an IP packet larger than the link's MTU, IP interfaces must perform *IP fragmentation* – chopping (fragmenting) the IP packet up into a sequence of smaller IP packets that all fit under the MTU limit. IP fragmentation occurs underneath the TCP or UDP layer, which allows the source UDP- or TCP-based applications to be unaware of the actual MTUs of links along the path to the destination. The ultimate destination is responsible for reassembling the fragments into the original packet, and then treating the reassembled IP packet as though it had arrived in one piece.

IP fragmentation tends to occur when a packet's path originates on a link with a large MTU, and then at some point along the route passes across a link with a smaller MTU. It is not considered a good thing, as it creates less efficient data transfer along the path [RFC1191][RFC1981].

For the curious reader: Under many versions of Unix (for example, FreeBSD or Linux) the command *ifconfig* shows a variety of details about a host's currently attached link layers, including the currently assigned IP address(es), subnet mask(s), and MTU(s). If you are running Windows XP or something similar, the command *ipconfig /all* at a console window will also print out a variety of details about the host's current configuration.

4.2.3.5 First and Last Hops

Forwarding tables are not just for routers. Hosts also have a limited forwarding table that tells them the initial next hop (often referred to as the *first hop*) for outbound packets. The first hop will either be to a router (for traffic destined beyond the local subnet) or directly to a neighbour on the same subnet. In cases where the next hop goes directly to a neighbour on the same subnet it is referred to as the *last hop*. (The first and last hops may be one and the same in the case of communication between two hosts on the same subnet.)

Forwarding tables have special rules for subnets that are directly attached to one of the host's or router's IP interfaces. Rather than having a specific entry for every IP interface reachable on a local subnet, the 'next-hop' IP address is copied directly from the packet's destination address field. The ARP cache is then scanned for a match to this next-hop IP address, and the packet transmitted to the link layer destination found in the ARP cache.

Hosts typically only have a few entries in their forwarding table, for example, one entry for the directly attached subnet, and a default route (network/prefix 0.0.0.0/0) pointing to a router on the local subnet that provides access to the rest of the network. If a host has link interfaces to multiple IP subnets, it will have forwarding table entries for each directly attached subnet, and possibly multiple forwarding entries for nonlocal traffic.

For the curious reader: If you are running Windows XP or something similar, entering *route print* in a console window will show the current forwarding rules. Under many versions of Unix (for example, FreeBSD or Linux) the command *netstat -rn* will show the host's current forwarding rules.

4.2.3.6 Tunnels as Links

Links are simply mechanisms for getting an IP packet from one router to another. A link may even be an IP network in its own right. Transmission of IP packets within other IP packets is known as *IP tunnelling*, and the link is known as an *IP tunnel*. From the perspective of the outer IP packet, the packet being tunnelled is just another payload (as uninteresting as a TCP or a UDP frame). From the perspective of the tunnelled packet, the tunnel looks like another link layer. From an implementation perspective, an IP tunnel is a link layer where source and destination addresses also happen to be IP addresses.

The tunnel's endpoint is the IP interface identified by the destination IP address in the outer packet's header. When the outer IP packet reaches its destination, the original (inner) IP packet is extracted and processed as though it had arrived over a regular interface. The outer packet's IP Protocol Type identifies the payload as a tunnelled packet, for example, protocol type 4 indicates that the payload is an IPv4 packet [RFC2003]. Because a tunnel represents a single hop from the perspective of the tunnelled packet, its TTL is decremented by one (rather than the number of hops between the tunnel endpoints).

Tunnelling over an IPv4 network imposes an additional 20 bytes of overhead (the header of the outer IP packet) [RFC2003]. The MTU of the tunnel's virtual link is 20 bytes smaller than the smallest MTU along the outer packet's path. (RFC 2004 suggests a more efficient encapsulation mechanism incurring only 8 or 12 bytes of overhead but with some loss of generality, for example, fragmented IPv4 packets cannot be tunnelled [RFC2004]. For RFC2004-based tunnelling, the protocol type in the encapsulating/tunnelling header is 55.) When tunnelling over IPv6 networks, the MTU drops by at least 40 bytes (the size of the encapsulating IPv6 header) [RFC2473].

4.3 Address Management

Every IP interface needs an IP address, which raises some very real administrative issues when building big networks. There are two key aspects to address management:

- Establishing an IP subnet from which you can assign IP addresses
- Actually assigning individual IP addresses to interfaces.

In this section, we will review how address blocks are assigned to customer networks from blocks delegated to Internet Service Providers (ISPs); how NAT can be used to overcome limitations in ISP address assignments; how the DHCP simplifies address assignment to individual devices inside your networks and how the domain name service attempted to decouple the naming of endpoints from the addressing of endpoints.

4.3.1 Address Delegation and Assignment

To be part of the wider Internet you cannot pick a subnetwork number and prefix at random. You need IP addresses that are globally unique and routable on the Internet. Such addresses are typically obtained from your ISP, which assigns you addresses from larger blocks allocated to the ISP by regional *registries* around the world [RFC2050].

For example, ARIN (the American Registry for Internet Numbers) manages space under 204/8 (204.0.0.0/8) and a number of other large blocks of IPv4 address space, and APNIC (the Asia-Pacific Network Information Center) manages space under 218/8 and a number of other large blocks. An ISP who asks for space from APNIC will receive an allocation under 218/8 or one of APNIC's other address blocks. Regional registries develop their own policies for subdividing the address blocks they manage. Up-to-date information on registries and assignment policies can be found in the Internet Assigned Numbers Authority web site, http://www.iana.org.

Problems arise when you decide to change ISPs. You will usually be forced to adopt a new IP address space assigned by your new ISP (called *renumbering*). Renumbering of your network is usually required by ISPs because routing would become more convoluted if the address hierarchy was allowed to arbitrarily diverge from the hierarchy of actual connectivity among the service providers.

A number of IPv4 address blocks (10/8, 172.16/12 and 192.168/16) have been reserved for use in *private internets* [RFC1918]. These are useful when building IP networks that will never be connected to the Internet, or will be connected only in a very limited fashion. In principle, such IP networks could be built using any prefixes. However, using designated private IP address spaces helps administrators distinguish between internal and external hosts in cases where their network contains a mixture of internal and external connectivity.

Interesting problems arise when a previously private network wishes to connect to the Internet. These will be discussed in the following section, along with a currently popular solution – NAT.

4.3.2 Network Address Translation

NAT is used at the boundaries between IP networks, most often between private networks and the public Internet [RFC3022]. Fundamentally, it solves the problem of a private network whose internal IP address space does not map cleanly into an unused, publicly routable IP address space.

4.3.2.1 Pure NAT

Pure NAT is best explained with an example. Consider the situation in Figure 4.16. A company has a private network of 100 hosts, using addresses in the private range 192.168.0.1 to 192.168.0.100. At some point in time, the company wishes for all hosts on its internal IP network to access the Internet. The company contacts an ISP and is allocated a CIDR block of 256 addresses, perhaps 128.80.6/24. If there were only a router between the private network and the ISP, every host would need to be renumbered to a unique address in the 128.80.6/24 range.

However, NAT provides an alternative to renumbering. Basically, NAT dynamically modifies the source and destination addresses in packets as they are forwarded between the private network and the ISP.

For packets being transmitted out to the Internet, the steps are as follows:

- Source hosts use their own private address in the packet's source address field.
- Internal routing forwards packets to the NAT-enabled router linking the private network to the ISP.
- The NAT-enabled router swaps the source address in each packet with a source address taken from the publicly routable address space 128.80.6/24, and then forwards the packet to the ISP.

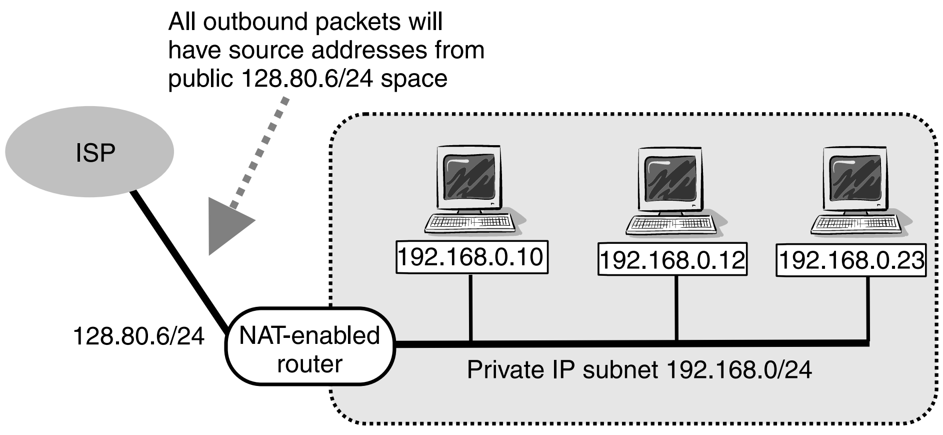
All outbound packets will have source addresses from public 128.80.6/24 space
ISP
192.168.0.10
192.168.0.12
192.168.0.23
128.80.6/24
NAT-enabled router
Private IP subnet 192.168.0/24

Figure 4.16 NAT helps map a private address space into the public address space

- The router remembers the address mapping it used, so it can reverse the process for inbound packets coming from the ISP.

For packets coming back in from the Internet, the reverse steps are as follows:

- A packet arrives at the NAT-enabled router, with a destination address in the 128.80.6/24 range.
- The NAT-enabled router looks up the mapping between 128.80.6/24 addresses and internal 192.168.0.*, and replaces the packet's destination address with the private IP address of the intended destination host.
- Internal routing (within the private network) forwards the modified IP packet to the correct destination.

A NAT-enabled router is generally free to use whatever address mapping schemes it chooses. For example, in Figure 4.16 the NAT-enabled router might choose to map internal address 192.168.0.10 to public address 128.80.6.20, or indeed any address in the 128.80.6/24 range. Mappings may be statically assigned, or dynamically generated on-demand. The only requirement is that mappings are unique – multiple private host addresses should never map to the same public IP address, and vice versa.

4.3.2.2 Network Address Port Translation

A common scenario for home networks is where the ISP (whether regular modem dial-up or a broadband service) charges additional monthly fees for a second or third public IP address. The solution is an extended version of NAT called *Network Address Port Translation* (NAPT) [RFC3022]. An NAPT-enabled router transparently makes multiple hosts on the private network appear to be a single host from the perspective of the public Internet.

NAPT extends NAT by additionally manipulating the port numbers of TCP and UDP traffic going in and out of the private network. Consider the scenario of Figure 4.17, where two hosts on a private LAN (192.168.0.12 and 192.168.0.13) are sharing a single public IP address (128.80.6.200).

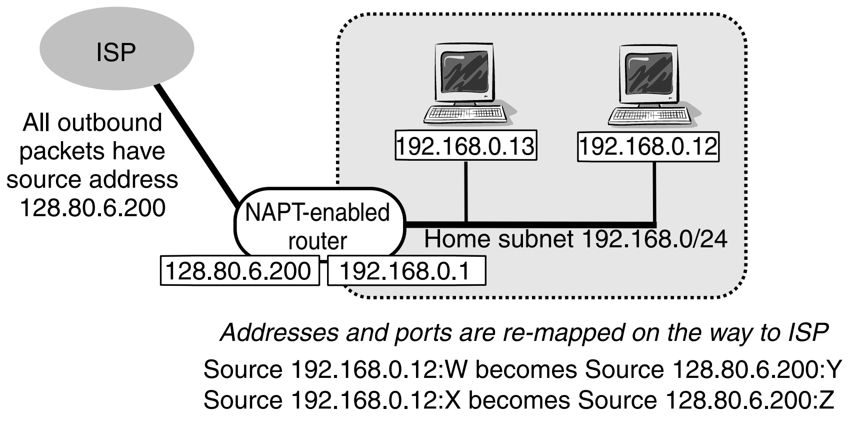
ISP
All outbound packets have source address 128.80.6.200
192.168.0.13
192.168.0.12
NAPT-enabled router
Home subnet 192.168.0/24
128.80.6.200
192.168.0.1
Addresses and ports are re-mapped on the way to ISP
Source 192.168.0.12:W becomes Source 128.80.6.200:Y
Source 192.168.0.12:X becomes Source 128.80.6.200:Z

Figure 4.17 NAPT maps both addresses and TCP/UDP ports to share public IP addresses across multiple private hosts

Imagine that both hosts are engaged in separate TCP connections with other, unrelated hosts on the Internet. The host TCP connections originate from ⟨addr=192.168.0.12, port=W⟩ and ⟨addr=192.168.0.13, port=X⟩ respectively. Because there is no coordination between hosts when they choose their TCP source ports, there is no guarantee that W and X are different.

NAPT remaps both the source port and source address fields to ensure that the individual TCP connections appear unique on the public side of the router. In this example, outbound packets from ⟨addr=192.168.0.12, port=W⟩ are modified to appear as packets from ⟨addr=128.80.6.200, port=Y⟩ and forwarded to the ISP. Likewise, packets from ⟨addr=192.168.0.13, port=X⟩ are modified to appear as packets from ⟨addr=128.80.6.200, port=Z⟩, where Z and Y are guaranteed to be different. Packets coming back in from the Internet are modified with the reverse mappings before being forwarded onto the private network.

4.3.2.3 Convenience and Limitations

Both NAT and NAPT provide independence from the need to renumber when your public IP address(es) change. Only the NAT-enabled router needs to be aware of any change in the range of public IP addresses assigned to the company's network – the hosts remain unchanged. This makes it easy for small companies to change ISPs with minimal disruption of internal network operations. NAPT also allows multiple hosts on a home LAN to access the Internet while avoiding additional charges for more than one IP address.

Naturally, all this convenience comes with caveats [RFC2993]. NAT and NAPT break the transparency of TCP and UDP communication between hosts, and require special-case coding to handle other protocols. While hosts on the private network may initiate communication with anyone else on the Internet, the reverse is far more complex. Additional functionality is required in your NAT/NAPT router to enable hosts inside the private network to support 'well-known' servers visible to the rest of the Internet.

For example, imagine you have a small corporate site with 200 hosts and three of them want to run publicly accessible web servers. The default 'http://www.companyname.com' web address format actually implies that the web server is listening for HTTP traffic on TCP port 80. However, if you only have one public IP address, the NAPT router can only map inbound ⟨dst addr, port=80⟩ traffic to *one* of your internal hosts, not three. The second and third would-be web servers will either need to give up on their plans, merge with first machine, or configure the NAPT router to utilise nonstandard mappings (for example, mapping ports 8080 and 8081 to the second and third internal machines respectively, and giving external web addresses of the form 'http://www.companyname.com:8080' and 'http://www.companyname.com:8081' respectively).

Running game servers behind NAT/NAPT is similarly problematic. Many games require the server to register its IP address and port number with a master server (through which potential players find available game servers). But when sitting behind NAT/NAPT, inbound connections (e.g. from new players) are typically only allowed by the NAT/NAPT router if they correspond to a recently initiated outbound connection. But since players initiate contact with the game server, not the other way round, we have a dilemma. (For example, consider Figure 4.17 with a Quake III Arena server running on host 192.168.0.13 at port 27960. Further, assume the NAPT router is manually configured to map 128.80.6.200:28000 to 192.168.0.13:27960. The master server 'sees' the Quake III Arena server at 128.80.6.200:28000. However, without special configuration the NAT/NAPT router will not allow new players to actually connect through 128.80.6.200:28000 to the game server itself. We discuss this again in Chapter 12.)

NAT/NAPT has its admirers and detractors. Nevertheless, it does serve a purpose for private networks that cannot afford lots of public IP addresses or wish to avoid renumbering of their internal networks on a regular basis.

Consumer home routers/gateways invariably support some form of NAT/NAPT functionality. Demand is driven by the deployment of broadband IP access over Asymmetric Digital Subscriber Line (ADSL) or cable modem services, and the fact that many homes have multiple computers. Typically the home router has one Ethernet port to the ADSL modem or cable modem, and one or more Ethernet ports for the internal, home network. (To assist in address management of a small home network, many home routers also support the dynamic host configuration protocol described in the following section.)

4.3.3 Dynamic Host Configuration Protocol

The DHCP [RFC2131] automates the configuration of various fundamental parameters hosts need to know before they can become functional members of an IP network. For example, every host minimally needs to know the following:

- The host's own IP address
- The subnet mask for the subnet on which it sits
- The IP address of at least one router to be used as the default route for all traffic destined outside the local subnet.

Without these pieces of information, a host cannot properly set the source IP address of its outbound packets, cannot know if it is the destination of inbound unicast packets, and cannot build a basic forwarding table that differentiates between on-link and off-link next hops.

DHCP allows hosts to automatically establish the preceding information, and provides two key benefits:

- The need for manual intervention is minimised when installing and turning on new hosts.
- IP addresses can be *leased* for configurable periods of time to temporary hosts.

Minimising administrative burdens clearly saves money and time, and increases overall convenience. The benefits of dynamic address leasing become apparent in networks where not all hosts are attached and operational at the same time.

4.3.3.1 Configuring a Host

DHCP is a client–server protocol. Each host has a *DHCP client* embedded in it, and the local network has one or more nodes running *DHCP servers*. DHCP runs on top of UDP, which at first glance suggests a Catch-22 situation with the unconfigured IP interface. However, DHCP only requires that an unconfigured IP interface can transmit a broadcast packet (IP destination address ‘255.255.255.255’) to all other IP interfaces on the local link.

A DHCP client solicits configuration information through a multistep process:

- First the client broadcasts a DHCP Discover message in a UDP packet to port 67, to identify its own link layer address and to elicit responses from any DHCP servers on the local network.
- One or more DHCP servers reply with DHCP Offer messages, containing an IP address, subnet mask, default router, and other optional information that the client may use to configure itself [RFC2132].
- The DHCP client then selects one of the servers, and negotiates confirmation of the configuration by sending back a DHCP Request to the selected server.
- The selected DHCP server replies with a DHCP Ack message, and the host begins operating as a functional member of the subnet to which it has been assigned.

Address administration is thus centralised to one or more DHCP servers.

4.3.3.2 Leasing Addresses

DHCP servers may be configured to allocate IP addresses in a number of ways:

- Static IP mappings based on *a priori* knowledge of the link layer addresses of hosts supposed to be on the managed subnet.
- Permanent mappings that are generated on-demand (the server learns client link layer addresses as clients announce themselves).
- Short-term leases, where the DHCP client is assigned an IP address for a fixed period of time after which the lease must be renewed or the address returned to the server’s available address pool.

The third option is most useful where network access must be provided to a large group of transient hosts using a smaller pool of IP addresses. For example, consider a public access terminal centre at a university with 50 Ethernet ports into which students can plug their laptop computers. Many hundreds or thousands of students might use the centre over a week or a month. Thousands of IP addresses would be required for a static IP address assignment scheme, one for each student’s laptop. On the contrary, leasing addresses for short periods of time means that a much smaller pool of IP addresses can serve the terminal centre’s needs.

A DHCP server can specify lease times in the order of hours, days, or weeks with a minimum lease time of one hour. DHCP clients are informed of the lease time when they first receive their address assignment. As the lease nears expiration, DHCP clients are expected to repeat the Request/Ack sequence to renew their lease. The DHCP server usually allows clients to continue with their leases at renewal.

DHCP clients are also allowed to store their assigned IP address in long-term storage (battery-backed memory, or local disk drive) and request the same address again when it next starts up. If the address has not subsequently been issued to another client, DHCP servers typically allow a lease to be renewed after clients go through a complete restart.

4.3.4 Domain Name System

As noted earlier in this chapter, IP addresses are not the same as FQDNs (often referred to simply as *domain names*). Domain names are human-readable, text-form names that indirectly represent IP addresses.

The DNS is a distributed, automated, hierarchical look-up and address mapping service [RFC1591]. People typically use domain names to inform an application of a remote Internet destination, and the applications then use the DNS to perform on-demand mappings of domain names to IP addresses. This level of indirection allows consistent use of well-known domain names to identify hosts, while allowing a host's IP address to change over time (for whatever reason).

Two forms of hierarchy exist in the DNS – hierarchy in the structure of names themselves and a matching hierarchy in the distributed look-up mechanism.

4.3.4.1 Domain Name Hierarchy

Domain names are minimally of the form ⟨*name*⟩.⟨*tld*⟩ where ⟨tld⟩ specifies one of a handful of *Top Level Domains* (TLDs) and ⟨name⟩ is an identifier registered under the specified top-level domain. Examples of generic three letter TLDs (gTLDs) include com, edu, net, org, int, gov, and mil. Country code TLDs (ccTLDs) are constructed from standard ISO-3166 two letter 'country codes' (e.g. au, uk, fr, and so on) [ISO3166].

The nested hierarchy is read from right to left, and ⟨name⟩ may itself be broken up into multiple levels of subdomains. Some TLDs are relatively flat (for example, the 'com' TLD), with companies and originations around the world able to register second-level domains immediately under 'com'. Country code TLDs have varied underlying structures, sometimes replicating a few of the existing three letter TLDs as second-level domains (for example, Australia registers domain names under a range of second-level domains including 'com.au', 'edu.au', and so on.)

The hierarchical structure reflects the administrative hierarchy of authority associated with assigning names to IP addresses. For example, consider an address like 'mail.accounting.bigcorp.com'. The managers for 'com' have delegated all naming under 'bigcorp.com' to a second party (most likely the owners of 'BigCorp, Inc.'). BigCorp no doubt has various internal departments, including the Accounting department. Someone in the accounting department has been delegated authority for naming under 'accounting.bigcorp.com', and they have assigned a name for the mail server in the accounting department. Figure 4.18 represents the relationships between the subdomains discussed so far.

A domain name hierarchy is independent of the hierarchy of IP addresses and subnets discussed earlier in this chapter. For example, onemachine.bigcorp.com might well be on an entirely different IP subnet (indeed, even a different country) from othermachine.bigcorp.com.

Domain name registration has become a commercial business in its own right, and multiple *registrars* jointly manage different sections of the DNS. Up-to-date information on registrars and domain assignment policies can be found in the Internet Assigned Numbers Authority web site, http://www.iana.org.

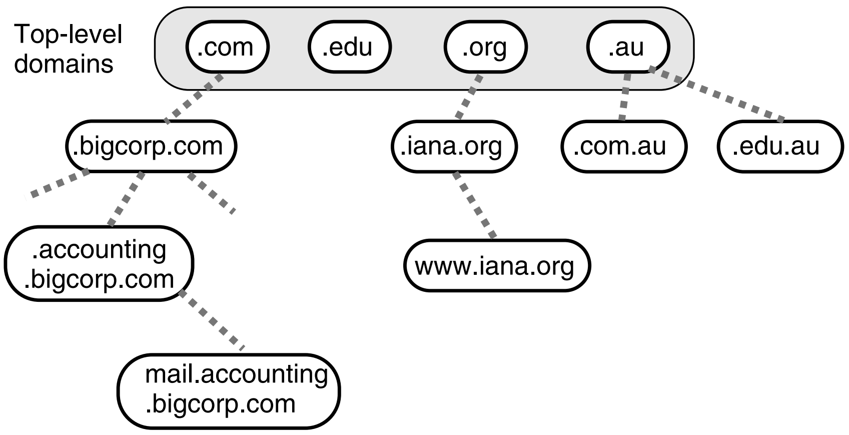

Figure 4.18 Hierarchy within domain name structure reflects a hierarchy of delegation authority

4.3.4.2 DNS Hierarchy

The hierarchy in a domain name essentially describes a search path across the distributed collection of *name servers* that together make up the Internet's DNS. Name servers are queried whenever a domain name needs to be resolved to an IP address. Certain name servers are responsible for being authoritative sources of information for particular domains or subdomains. The name servers ultimately responsible for each TLD are known as *root name servers*.

Before hosts can use the DNS they must be configured with the IP address of a local name server – the host's entry point into the DNS. The ISP or whoever supports your network typically provides the local name server. The name server's IP address is either manually configured into each host, or can be automatically configured (for example, DHCP provides an option for configuring the local name server's address [RFC2132]).

Local name servers are manually configured to know the IP address of at least one root name server, and possibly another name server further up the domain name tree. Name servers either answer queries with local knowledge, or seek out another name server who is responsible for mappings higher up the domain name hierarchy. Local knowledge is often held in a cache built from recent queries from other hosts – the cache allows rapid answers for frequently resolved domain names.

For the curious reader: Many recent versions of Windows, and Unix-link operating systems such as Linux and FreeBSD have a tool called *nslookup*. Often installed as a command line application, nslookup allows you to manually perform DNS queries and explore your local network's DNS configuration. Similar tools may be found under names like *dig* or *host*.

References

[ISO3166] http://www.iso.org/iso/en/prods-services/iso3166ma/02iso-3166-code-lists/list-en1-semic.txt.

[RFC768] J. Postel, Ed, "User Datagram Protocol", RFC 768. August 1980.

[RFC791] J. Postel, Ed, "Internet Protocol Darpa Internet Program Protocol Specification", RFC 791. September 1981.

[RFC793] J. Postel, Ed, "Transmission Control Protocol", RFC 793. September 1981.

[RFC826] D.C. Plummer, "An Ethernet Address Resolution Protocol", RFC 826. November 1982.

[RFC1112] S. Deering, "Host Extensions for IP Multicasting", RFC 1112. August 1989.

[RFC1191] J. Mogul, S. Deering, "Path MTU Discovery", RFC 1191. November 1990.

[RFC1390] D. Katz, "Transmission of IP and ARP over FDDI Networks", RFC 1390. January 1993.

[RFC1519] V. Fuller, T. Li, J. Yu, K. Varadhan, "Classless Inter-Domain Routing (CIDR): an Address Assignment and Aggregation Strategy", RFC 1519. September 1993.

[RFC1591] J. Postel, "Domain Name System Structure and Delegation", RFC 1591. March 1994.

[RFC1771] Y. Rekhter, T. Li, "A Border Gateway Protocol 4 (BGP-4)", RFC 1771. March 1995.

[RFC1918] Y. Rekhter, B. Moskowitz, D. Karrenberg, J. de Groot, E. Lear, "Address Allocation for Private Internets", RFC 1918. February 1996.

[RFC1981] J. McCann, S. Deering, J. Mogul, "Path MTU Discovery for IP version 6", RFC 1981. August 1996.

[RFC2003] C. Perkins, "IP Encapsulation within IP", RFC 2003. October 1996.

[RFC2004] C. Perkins, "Minimal Encapsulation within IP", RFC 2004. October 1996.

[RFC2050] K. Hubbard, M. Kosters, D. Conrad, D. Karrenberg, J. Postel, "Internet Registry IP Allocation Guidelines", RFC 2050. November 1996.

[RFC2131] R. Droms, "Dynamic Host Configuration Protocol", RFC 2131. March 1997.

[RFC2132] S. Alexander, R. Droms, "DHCP Options and BOOTP Vendor Extensions", RFC 2132. March 1997.

[RFC2225] M. Laubach and J. Halpern, "Classical IP and ARP over ATM", Internet Request for Comment 2225. April 1998.

[RFC2328] J. Moy, "OSPF Version 2", RFC 2328. April 1998.

[RFC2453] G. Malkin, "RIP Version 2", RFC 2453. November 1998.

[RFC2473] A. Conta, S. Deering, "Generic Packet Tunneling in IPv6 Specification", RFC 2473, December 1998.

[RFC2616] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, T. Berners-Lee, "Hypertext Transfer Protocol – HTTP/1.1", RFC 2616. June 1999.

[RFC2821] J. Klensin, Ed, "Simple Mail Transfer Protocol", RFC 2821. April 2001.

[RFC2993] T. Hain, "Architectural Implications of NAT", RFC 2993. November 2000.

[RFC3022] P. Srisuresh, K. Egevang, "Traditional IP Network Address Translator (Traditional NAT)", RFC 3022. January 2001.

[IANAP] Internet Assigned Numbers Authority, "Directory of General Assigned Numbers (last viewed January 2006)", http://www.iana.org/numbers.html.

[8023] IEEE Std 802.3. "IEEE Standards for Local and Metropolitan Area Networks: Specific Requirements. Part 3: Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications", 1998.