7 DIRECTORIES

In a conversation with Richard Cambridge in 1755, Samuel Johnson made the now famous remark that "Knowledge is of two kinds. We know a subject ourselves, or we know where we can find information upon it" (Boswell 1791). In a large network, it is neither possible nor desirable for every network element[1] to know everything about every other network element—impossible because there is too much to know, and undesirable because much of the information is constantly changing, and the more widely distributed it is, the more difficult it is to maintain synchronization among all the many places in which it is replicated.

Since we cannot invest every network element with complete knowledge, we must provide a system whereby they "can find information upon it." This is the role of the network directory, which—as anyone who has ever used a telephone directory might easily surmise—is simply a place to store lots of information about the elements of a network. By doing so, the network directory not only solves the problems of "There's too much information to store everywhere" and "The information changes too often to be kept current everywhere" but also permits references to network elements to be made indirectly, through the directory, by name, rather than by some other attribute (such as network address, geographic location, or organizational affiliation) that might not be as permanently or reliably attached to the element as its name.

1. Intuitively, we think of network elements as the computers—PCs, workstations, bridges, routers—that provide network and host services. In the context of directories, however, the notion is extended to applications, to users of the network, and even to more fundamental pieces of such network elements including the information stored in these elements; in fact, very nearly every object we *name* within a network may be characterized and accessed by the use of directories.

Given such a general definition of *directory*, it is easy to imagine the network directory as a general-purpose database system—and in principle, it could be. In practice, however, the directory services that have been developed for open systems networking have been designed for the more specialized purpose of relating a particular class of names to a particular set of attributes associated with them—for instance, mapping Internet host names (such as "`nic.switch.ch`") to IP addresses (such as "`130.59.1.40`"). In this chapter, we will concentrate on the open systems directories whose core service is name-to-address mapping (although they provide other services as well): OSI's X.500 Directory and the Internet's Domain Name System. However, we will also take a look at a number of "directorylike" utilities and introduce the field of *networked information retrieval*, which began with a directory model but has evolved far beyond it.

The basic function of a directory service is deceptively simple, but the way in which directory services are used in open systems networks is not. The telephony model of a directory is a good place to begin our discussion of the design criteria for a network directory service, precisely because it is *not* a good model for such a service.

The Telephony Model

For many people, the term *directory* suggests the local telephone company's printed telephone directories and dial-up "directory assistance."[2] The telephone directory system is simple and straightforward: given the name and address of a telephone subscriber, it returns the subscriber's telephone number (unless the subscriber has paid a surcharge to the telephone company for an "unlisted number"—in which case, the information is available only to duly authorized law–enforcement personnel). This service is possible because the telephone companies jointly and exclusively administer a common pool of telephone numbers, for which the boundaries of local jurisdiction are (with rare exceptions) well-established and universally accepted.

This directory service has three limitations that argue strongly against its applicability to a worldwide open systems Internet. First, it is

2. Directory assistance was widely known as "information" until the telephone companies, for which providing "information" is an expensive gratuity to customers who are "too lazy to look it up in the book," successfully stamped out the familiar term in favor of one that subtly suggests infirmity on the part of the user.

able to provide a telephone number only when given the subscriber's name and address. Lacking either a complete name or a sufficiently detailed set of attributes to disambiguate entries for the same name in the same locality, no other information that characterizes or distinguishes an individual can be used to facilitate a query. (Some printed directories, such as the yellow pages, may contain additional information provided by the subscriber; this is not always accessible "on-line" through directory assistance. For example, it is possible to ask for "a florist in Horsham" and obtain the name and telephone number of at least one in that city, but it is unlikely that an operator will be able to tell you whether the florist is a member of FTD or Teleflora, or which credit cards the florist accepts.) Second, it is not uniformly accessible. In order to obtain directory assistance, one must first know the appropriate country, city, and/or area codes, as well as the number for directory assistance itself, and construct one's query according to the appropriate local conventions—for example, in the appropriate local language. Third, the telephone directories—particularly the printed versions—are typically distributed no more frequently than once a year and thus inevitably contain a significant amount of incorrect (outdated) information.

The telephone directory system benefits enormously from a key feature of the telephone network: the fact that there is a direct relationship between the hierarchical structure of telephone numbers (with their country, city, area, and exchange components) and the geographic location hierarchy within which the telephone subscribers live. Data networks typically do not share this characteristic, despite being organized hierarchically. The corporate network of a large multinational company, for example, is likely to be organized according to the company's operating hierarchy (divisions, departments, cost centers, etc.), which may place the London and Hong Kong sales offices close together and the sales and personnel departments geographically colocated in Hong Kong far apart. Further, the affiliations and service relationships of data networks are (to date) neither as uniform nor as tightly regulated as telephony networks.

Directory System Principles

Given that the simple telephony model is not appropriate for a large-scale network directory, what *are* the characteristics that such a directory should have?

- Both the directory database and the mechanisms for operating on it must be distributed; the size of the database and the frequency with which directory information must be updated preclude centralization.

- However, the directory must appear to its users to be a single, consistent database. The same query, originating from any point in the network, should return the same information.[3]
- The directory must be organized hierarchically, so that the responsibility for managing the information in the directory can be delegated to different organizations as self-contained "subtrees." The hierarchy must be extensible, since the way in which organizations allocate and apportion responsibility among and within themselves in a large open systems network is certain to change over time.
- The directory must be organized in such a way that it is possible to formulate unambiguous queries; that is, it must not be the case that two different information elements contained in the directory cannot be distinguished by the directory's query mechanism.
- Because the directory for a large-scale network is necessarily itself a large-scale distributed system, it must not be application-specific—that is, it must be capable of storing information about objects for many (ideally, all) of the applications for which the network is used. This argues strongly for a directory information model based on the definition of *object classes* (families of objects sharing certain characteristics) and *attributes* (information about an object that either describes the object or distinguishes it from other objects), so that two different applications that refer to the same type of object do not require that the directory store two separate (application-specific) sets of information about the same objects.

Open System Directories

The world of open systems networking has produced two large-scale, open directory systems. *The Domain Name System* (DNS) is the established

3. Actually, the consistency requirements for a network directory are not as stringent as they typically are for a distributed database system. It is reasonable and acceptable, after an update, for the directory to exhibit local inconsistencies for some period of time until the change has propagated throughout the system. A directory query is almost never an end in itself but is followed by an attempt to use the information obtained, with a strong likelihood that the inconsistency will be exposed. Since the directory query and the subsequent network access based on it are not synchronized, it matters little whether the information changed before or after the directory query if it changes before the information is actually used.

Internet directory, and the *OSI Directory*, which is also known as "X.500" (after the first of the set of CCITT Recommendations that defines it) is both the standard for OSI networks and a candidate for use within the Internet (although not necessarily as a replacement for the Domain Name System).

The OSI Directory is deliberately comprehensive; it is intended to capture the relationship between an arbitrary name and an arbitrary list of attributes for virtually any network application. The Domain Name System is—particularly in practice—more narrowly focused: it associates the names of two specific resources (electronic mailboxes and Internet hosts) with two specific corresponding pieces of information about them (mail server addresses and IP addresses, respectively), although its design permits extension to other uses. Although the generality of the OSI Directory invites its application to other problems—it is used, for example, to store *universal document identifiers* for some of the networked information retrieval projects described at the end of this chapter—we are concerned here primarily with its deployment as a traditional "name lookup" service in both OSI networks and the multiprotocol Internet.

The Domain Name System

In the beginning, there were just four nodes in the only Internet around (the ARPANET), and maintaining a table of mappings from host name to network address was not a problem. In the early years of the ARPANET, growth was modest, and the host-name–to–address mappings were maintained by the Network Information Center (NIC) in a single file (`hosts.txt`), which was periodically retrieved (by using electronic file transfer or, in extreme cases, by requesting a magnetic or even paper tape) by each host or site administrator and loaded into each host attached to the network. Each host would then search through the file whenever it needed to find the network address for a named host.

This system worked well while the ARPANET was small, but as it grew, and as the composition of the network changed, the bandwidth consumed by the periodic and increasingly frequent electronic file transfers to retrieve the `hosts.txt` file from the NIC, and the disconnect between site administrators' management of their local names and addresses and the appearance of changes in the NIC's definitive `hosts.txt` file, made it clear that the centralized scheme was impractical and that an alternative would have to be found.

The Domain Name System began as a class project at the University of California at Berkeley and has, in recent years, been released by the Berkeley UNIX group as part of the Berkeley Software Distribution. Paul Mockapetris—first at Berkeley and later at the Information Sciences Institute of the University of Southern California—conceived an alternative to `hosts.txt` based on (1) a distributed database containing generalized *resource records* and (2) a naming scheme based on hierarchically structured *domain names*. His original design was published in November 1983 in RFCs 882 and 883; after experience with several implementations, the Domain Name System (DNS) was formally specified in RFCs 1034 and 1035 in November 1987.

The deployment of DNS in the Internet is, to say the least, nonuniform. Different versions of BIND—the *Berkeley Interactive Name Demon*, the UNIX software for name resolution—are bundled with different releases of UNIX software, and host-name lookups are done in a variety of different ways, depending on the vendor and on individual site philosophy. Some networks—including, for example, the entire U.S. federal MILNET—have decided not to use DNS and to rely instead on the old `hosts.txt` file for host-name–to–IP-address translation for the simple reason that there is no DNS security model to support the authentication of users and providers of DNS services. Nevertheless, DNS is one of the most important services in the Internet, and most of the strategic plans for the evolution of the Internet and the TCP/IP architecture depend on the near-universal deployment of DNS or a successor.

The DNS is a distributed system, with distribution based on the concept of *delegated authority* for the administration of individual domains and subdomains. This means that local system administrators maintain files containing information about their local hosts and networks, and this information is made available to users of the DNS through *name servers* that are locally configured and maintained.

Domain Names

The DNS name space is hierarchical, consisting of a set of nested domains, each of which represents an administratively related set of Internet hosts. Directly below the root of the hierarchy is a set of *top-level domains*, which originally referred to logical parts of the U.S. ARPANET:

`arpa:`	for the U.S. Department of Defense Advanced Research Projects Agency itself
`com:`	for commercial organizations
`edu:`	for educational institutions
`gov:`	for U.S. nonmilitary government agencies
`mil:`	for U.S. military agencies

`net:`	for organizations directly involved in the provision and support of ARPANET and its services
`org:`	for "other" organizations

With the expansion of the modern Internet outside the United States, additional top-level domains have been added, corresponding to individual countries; for example, `au, ca, us, uk, se`.

These country-code domain names are normally chosen from the two-letter codes registered in ISO 3166, *Codes for the Representation of Names of Countries* (ISO 3166: 1988).

The introduction of country-code domain names came about not because the DNS architects believed that it was the right way to register domains in countries outside of the United States but because they could not convince most of the non-U.S. network and site administrators that the established `com, edu,` *and other domains were not exclusively for U.S.-based organizations. The introduction of country-code top-level domains in parallel with the old nongeographic domains has led to some interesting anomalies. RARE,[4] for example, administers the domain* `rare.nl` *under the top-level domain for the Netherlands; RIPE,[5] which is organizationally affiliated with RARE, administers* `ripe.net.` *In the United States, the Corporation for National Research Initiatives (which operates the IETF secretariat, among other things) operates* `nri.reston.va.us`*; Bolt Beranek and Newman (which built the first IMPs and routers for the ARPANET, among other things) operates* `bbn.com.` *Observing the benign chaos with which the original top-level domain scheme has been infected by the introduction of country-code domains, Paul Mockapetris introduced the* `int` *top-level domain—for which Paul is the administrator—specifically for "people who don't understand that* `org` *isn't just for U.S. organizations."*

Below the top-level domains, names are constructed hierarchically by identifying subdomains, sub-subdomains, and so on to (in principle) any desired depth, until the final name (at a leaf of the tree; see Figure 7.1) completes the identification of an individual Internet host. Thus, within the top-level domain `uk`, the subdomain `ac.uk` is administered on behalf of U.K. academic institutions; within `ac.uk`, the subdomain `ucl.ac.uk` is administered by University College, London; and within

4. *Réseaux Associés pour la Recherche Europóene.* RARE is the principal sponsor and coordinator of academic and research networking in Europe.
5. *Réseaux IP Européens* (literally, "Research IP for Europe").

the subdomain `ucl.ac.uk`, the host name `murphy.ucl.ac.uk` identifies a workstation sitting on the desk of Professor Suzanne Chapin in her office in Whitehead Hall at University College, London.

The rules for the formation of names are straightforward: a name may be no longer than 63 characters; it must start with a letter; it must end with either a letter or a digit; the rest of the name may consist of letters, digits, and hyphens; and both upper- and lowercase letters may be used (although name lookups in the DNS are defined to be case-insensitive).

How the DNS Works

The DNS is implemented as two distinct components: DNS *servers* (usually called *name servers*), which contain information about one or more *zones;* and DNS *clients* (usually called *resolvers*), which interrogate name servers on behalf of local host processes.

The DNS Server Each DNS server provides name-to-address mappings for one or more *zones*. A zone is a set of contiguous domains beginning at some point in the DNS naming hierarchy and comprising that point and all the subtrees below it, as far, in each subtree, as either the leaves of the subtree or the point at which another (subordinate) zone is defined. In practice, an actual name server may be responsible for serving more

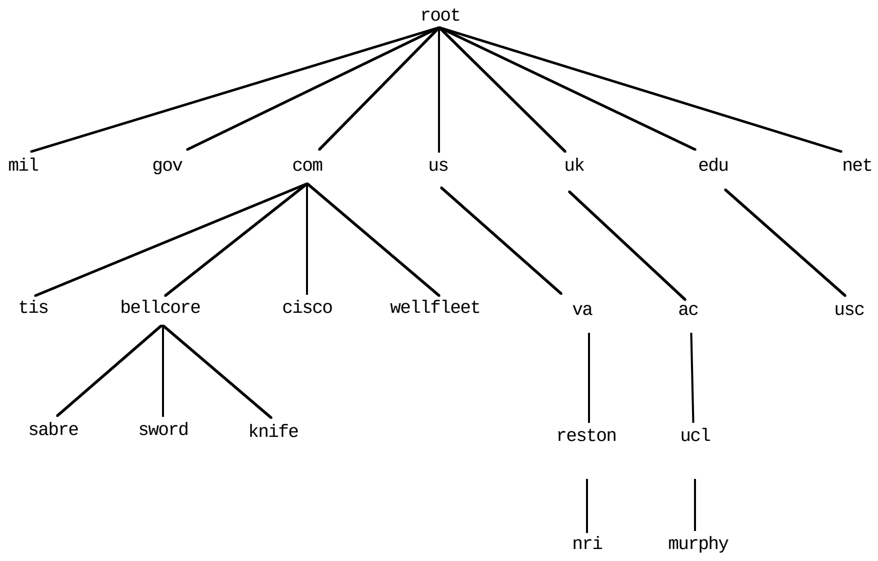

Figure 7.1 A Partial DNS Tree

Note: The word `root` is used merely to illustrate where, conceptually, the root of the name tree begins; domain names do not have the explicit identifier `root` as a component.

than one zone and will generally also serve the reverse `in-addr.arpa` zones (see "'Reverse' Lookups," later in the chapter) that correspond to its local domains.

Secondary name servers can be identified for each zone, so that name resolution for that zone is not cut off when the primary name server, or the links to it, are lost.

The DNS Resolver A DNS name server not only stores and manages information about domains; it also responds to queries concerning that information from client processes or *resolvers*. A resolver is typically a set of library routines that are invoked by an application program when it needs to resolve a name reference. In BSD UNIX, for example, the routines are `gethostbyname` and `gethostbyaddr`. The resolver takes the name and uses it to query either a local table[6] (if the reference is to a locally maintained name or to a name that the resolver has maintained in a local cache) or one or more DNS servers for corresponding *resource records*.

The resolver's most significant task is to formulate a proper query. Since the DNS works with fully qualified names, this may involve a bit of interpretation on the part of the resolver, which may be presented by its user with a name that is not fully qualified. Some resolvers abdicate this responsibility entirely and are capable of looking up only the name string exactly as provided by the user. Others are able to recognize a less than fully qualified name and supply the missing high-order domain specification automatically (defaulting, for example, to the domain in which the user's own host resides).

Clients and servers use a common format for DNS queries and replies. Basically, a client provides a unique *identifier* (so it can later match responses to queries) and poses a number of *questions* the server should attempt to resolve. In the case of name-to-address lookups, each question is an entry of the form {query domain name, type, class}; in this example, the domain name to be resolved is provided, the query type is set to a value indicating that the client is looking for an IP address, and the query class is set to a value indicating which object class (Internet domain names) is to be interrogated. The server returns a similarly formatted

6. Many resolvers have sophisticated methods of caching domain names, to reduce delay in resolving DNS queries. A frequent practice is to request and copy an entire DNS server's name information, then periodically send consistency-check inquiries to the DNS server. Two benefits are introduced when hosts practice caching: first, the processing of a DNS inquiry is faster, because there is no DNS client/server (protocol) interaction; second, if a server becomes unreachable, the local resolver can continue to satisfy lookups using the cache.

message with *answers* in a general form (a *resource record*). The resource record again contains a domain name, type, and class and, additionally, provides a suggested *time to live* for the information contained in this resource record and a variable-length *resource data* field, preceded by a length indicator field (see Figure 7.2).

Although the principal users of a DNS resolver are host applications, most host operating systems allow human users to present a query

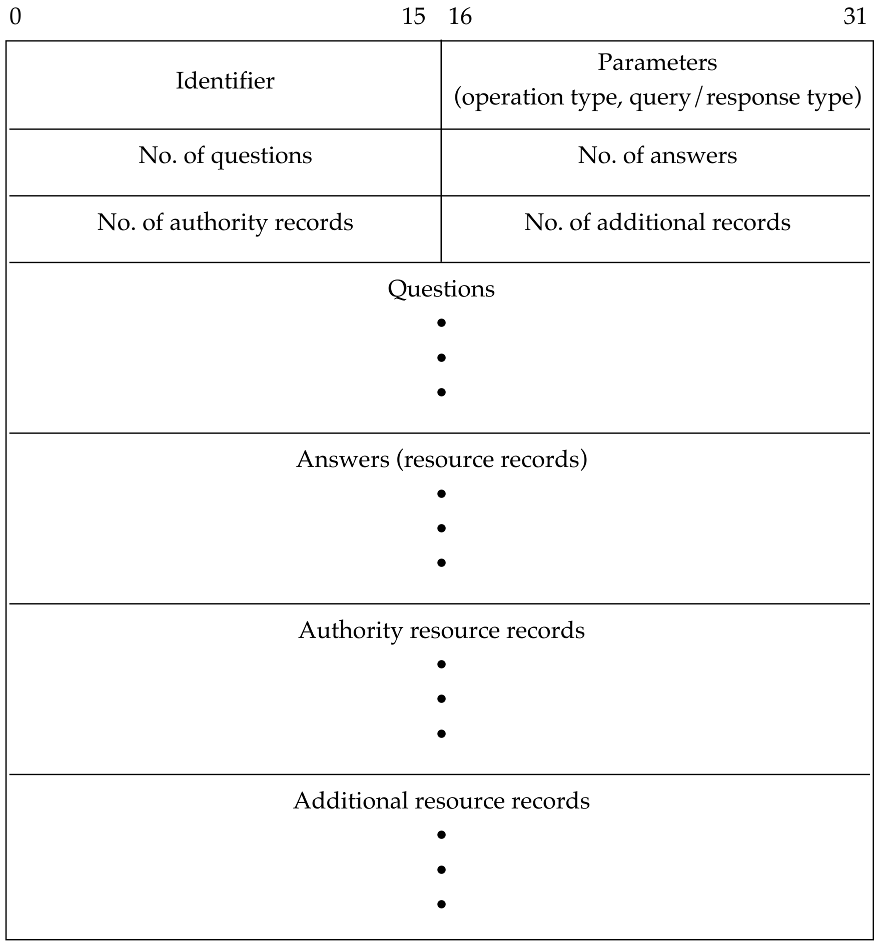
0
15
16
31
Identifier
Parameters (operation type, query/response type)
No. of questions
No. of answers
No. of authority records
No. of additional records
Questions
Answers (resource records)
Authority resource records
Additional resource records

FIGURE 7.2 DNS Message Format

directly, using, for example, a command such as the UNIX *nslookup*, which, in the SunOS version, looks like this:

```
% /usr/etc/nslookup
Default Server: nameserver.bbn.com
Address: 128.89.1.2

> nic.ddn.mil
Server: nameserver.bbn.com
Address: 128.89.1.2

Name: nic.ddn.mil
Address: 192.112.36.5


>exit
```

Most Internet applications, including common ones like FTP and TELNET, accept a domain name from the user and call a resolver internally, so it is rarely necessary for a user to query the DNS directly (except out of curiosity and perhaps to trouble-shoot networking problems).

"Reverse" Lookups

A special domain within the domain name space, `in-addr.arpa`, provides a mechanism for performing "reverse" lookups—that is, finding the name associated with a given address. The entries in the `in-addr. arpa` domain are constructed by reversing the order of the components of a network number (the first 1, 2, or 3 bytes of an IP address, depending on whether the address is a class-A, class-B, or class-C address; see Chapter 13), and appending the `in-addr.arpa` domain name. Thus, for example, the entry for BBN's class-B network number, 128.89.0.0, would be `89.128.in-addr.arpa`; a lookup on this entry would return the domain name `bbn.com`. Individual host systems, such as the Macintosh on the desk in Lyman Chapin's office at BBN, appear as subdomains of the network-number domain; for example, `224.16.89.128.in-addr.arpa`.

Mail Exchange

The DNS also plays an important role in electronic-mail service (see Chapter 8). A host acting as a mail transfer agent must know the host name and IP address of the host to which a mail message is to be delivered before it attempts the delivery. It extracts the domain name part from the destination mail address (again, see Chapter 8), and creates a query with a *question* of query type *MX* (for "mail exchange"). The DNS server returns an answer with one or more MX resource records. Each MX record identifies a host to which mail may be forwarded, with each host domain name accompanied by an indication of how desirable it is to use this host for forwarding mail relative to others in the list (a *preference field*).

The OSI Directory

The OSI Directory is both a logical database of information about a set of objects in the real world and a system of agents and protocols that manage the information and support a variety of queries and searches by directory users. It is not intended to be a general-purpose database system, although an actual implementation of the Directory might be built upon such a system. It is an expressed architectural goal of the OSI Directory that it be capable, in principle, of accommodating the information-storage and -distribution needs of every network and every host throughout the world of open systems networking:

ISO/IEC 9594 [the ISO equivalent of CCITT Recommendation X.500] refers to The Directory in the singular, and reflects the intention to create, through a single, unified name space, one logical directory composed of many systems and serving many applications. Whether or not these systems choose to interwork will depend on the needs of the applications they support. Applications dealing with nonintersecting worlds of objects may have no such need. The single name space facilitates later interworking should the needs change. (ISO/IEC 9594-1: 1990)

The X.500-Series Standards

In 1988, the first jointly-developed ISO/IEC/CCITT standards for a worldwide directory system, generally known as "X.500," were published. Although the term *X.500* is commonly used to refer to the directory standards, they in fact consist of eight separate specifications, which are listed in Table 7.1. (The ISO/IEC 9594 series of standards are referenced in subsequent subsections; refer to this table if you wish to find the

TABLE 7.1 The X.500-Series Directory Standards

CCITT Recommendation	*ISO/IEC Standard*	*Title*
X.500	9594-1	Overview of Concepts, Models, and Services
X.501	9594-2	The Models
X.511	9594-3	Abstract Service Definition
X.518	9594-4	Procedures for Distributed Operation
X.519	9594-5	Protocol Specifications
X.520	9594-6	Selected Attribute Types
X.521	9594-7	Selected Object Classes
X.509	9594-8	Authentication Framework

corresponding subject in an X.500-series Recommendation.)

Work on the directory standards continues as a joint enterprise of ISO/IEC and CCITT. By the end of 1992, several critical areas of study were completed, among them:

- A model for replication of parts of the directory information base and, in particular, definition of a standard replication protocol. Hitherto, replication protocols used between directory system agents were proprietary.
- A list-based access-control mechanism. The X.500-1988 and ISO/IEC 9594: 1990 versions of the OSI Directory have well-defined authentication mechanisms but no standard means of restricting access to specific parts of the directory information base; thus, anyone who is currently authorized to use the directory is authorized to look at *everything* in the database.

Architecture

The principal architectural features of the OSI Directory are:

- *Decentralized maintenance:* Each system providing an OSI Directory service is responsible for the maintenance and integrity of only its own local part of the directory information base; wherefore updates and other management operations can be carried out independently by "keepers of the directory information," formally known as directory system agents.
- *Structured information model:* The OSI Directory defines an object-oriented model and database schema that applies uniformly to all the information stored in the directory.
- *Hierarchical global name space:* The hierarchy of distinguished names depends uniformly from a single, global root, providing a homogeneous name space for directory users.
- *Extensive search and retrieval capability:* Directory users may construct arbitrarily complex queries and perform highly complex interactive searches of the directory information base.

The X.500 standards define the directory in terms of "models":

- The *information model* specifies the contents of directory entries, how they are identified, and the way in which they are organized to form the directory information base.
- The *directory model* describes the directory and its users, the functional model for directory operation, and the organization of the Directory.
- The *security model* specifies the way in which the contents of the

directory are protected from unauthorized access and updates to the directory information base are authenticated.

The Directory Information Model

The information contained in the OSI Directory is organized as a set of *entries*; the set of all such entries constitutes the *directory information base* (DIB). The entries in the DIB are arranged hierarchically and can be represented in the form of a tree (the *directory information tree*, or DIT). The acronyms *DIB* and *DIT* are often used interchangeably;[7] a useful way to think about the relationship between the "database" and the "tree" is to consider that every entry in the database occupies a position in the tree—the tree therefore expresses the hierarchical relationship that the OSI Directory defines for the entries in its database, as illustrated in Figure 7.3.

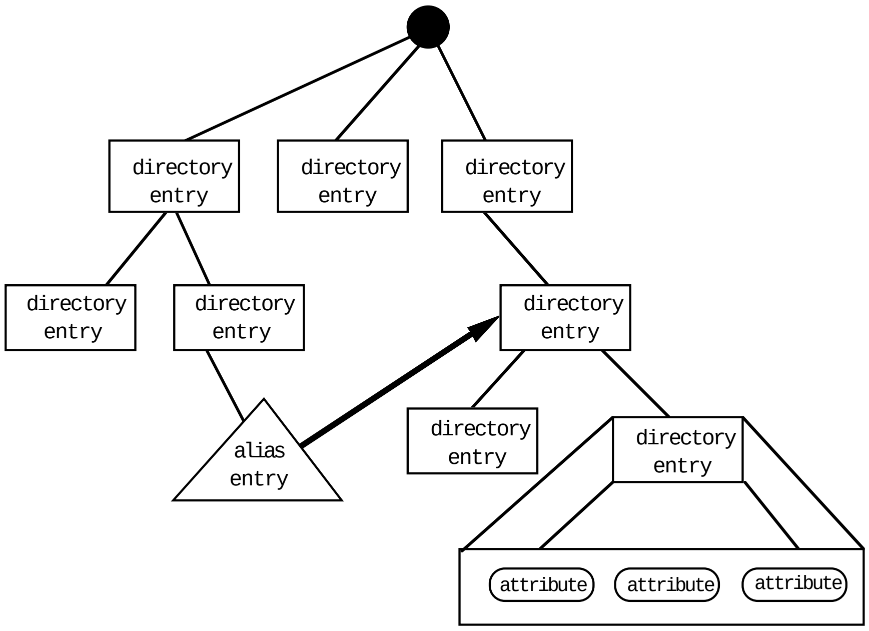

FIGURE 7.3 Structure of Directory Information and of Entries

7. The X.500 standards always refer to the complete set of entries in the directory as the DIB, reserving the equivalent term *DIT* for those circumstances in which it is important to emphasize the hierarchical tree structure of the database.

Directory Entries and Attributes The ASN.1 encoding of a directory entry is:

```
Attribute ::=
   SEQUENCE {
      type      AttributeType,
      values    SET OF AttributeValue }

AttributeType ::= OBJECT IDENTIFIER

AttributeValue ::= ANY
```

Each directory entry consists of a set of *attributes*, each of which consists of an *attribute type*, which identifies the class of information given by the attribute, and one or more corresponding *attribute values*, which are particular instances of that class of information (see Figure 7.4). An entry may not contain more than one attribute of a given type.

What sort of things are attributes? Virtually anything that describes an object for which a directory entry is created. Some attribute types are

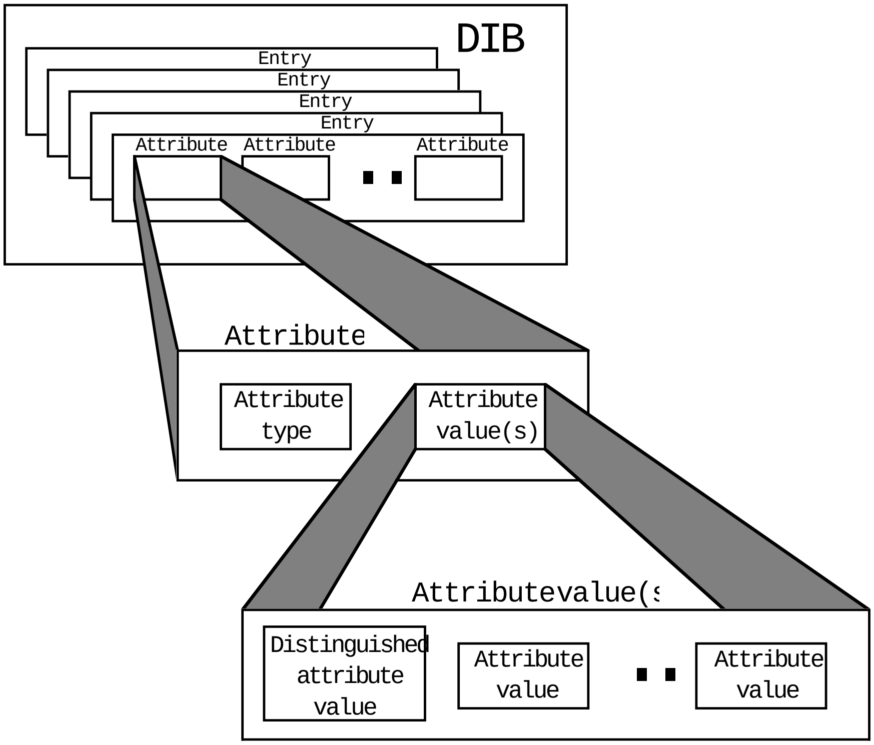

FIGURE 7.4 Directory Entries and Attributes

internationally standardized. A set of "selected attribute types," considered by the standards committees to be widely applicable, is provided in ISO/IEC 9594-6: 1990; they include:

- *Labeling attribute types:* Common Name, Surname, Serial Number
- *Geographic attribute types:* Country Name, Locality Name, State or Province Name, Street Address
- *Organizational attribute types:* Organization Name, Organizational Unit, Title
- *Postal-addressing attribute types:* Postal Address, Postal Code, Post Office Box, Physical Delivery Office Name
- *Telecommunications-addressing attribute types*: Telephone Number, Telex Number, Teletex Terminal Identifier, Facsimile Telephone Number, X.121 address, ISDN Number, Registered Address
- *OSI application attribute types:* Presentation Address, Supported Application Context

This is but a sampling. Other attributes are defined by national administrative authorities or private organizations. For example, it is perfectly appropriate to imagine a set of attributes for the medical profession, including residency or attending physician attributes (where? how many years? under whom? area[s] of specialization? publications, honors received, recommendations, evaluations, hours of practice?) and malpractice attributes (insurance carrier, annual premium, suits pending, suits settled). In other words, you can assume that organizations or individuals can create an attribute type for anything they wish to use to distinguish an individual or an object from another of its class.

In order to ensure that attribute types are assigned in such a way that each is distinct from all other assigned types, they are identified by an object identifier. The syntax (and hence, the data type) of attribute values for a particular attribute type is specified when the attribute type is defined.

An *attribute value assertion* (AVA) is a proposition—which may be true, false, or undefined—concerning the value(s) (or in some cases, only the distinguished values; see the following paragraphs) of an entry; it is usually expressed as a sequence of one or more statements of the form `AttributeType = AttributeValue`.

Directory Names At most, one of the values of an attribute may be designated as a *distinguished value*—in which case, the value appears in the *relative distinguished name* (RDN) of the entry. Every entry has a unique relative distinguished name, which consists of a set of attribute value assertions (each of which is true) concerning the distinguished values of attributes of the entry. The set contains exactly one assertion about each distinguished value in the entry. By far the most common case is that in which an entry has just one distinguished value, and the relative distinguished name therefore consists of a single attribute value assertion; however, this need not always be the case.

The *distinguished name* (DN) of a given object is defined as the sequence of relative distinguished names of (1) the entry in the directory information base that represents the object and (2) All of the entries superior to it in the directory information tree, in descending order. A distinguished name can be used as the primitive name (see Chapter 5) of the object it identifies. Object identifiers can be transformed in a simple way into distinguished names for access to an X.500-based directory service, either by a direct mapping (in which the object identifier value corresponds directly to a distinguished name of which the components are values of a directory attribute of type `object-identifier-component-value`) or by ensuring that object identifier component values are allocated together with corresponding relative distinguished name values.

Figure 7.5 illustrates the relationship between relative distin-

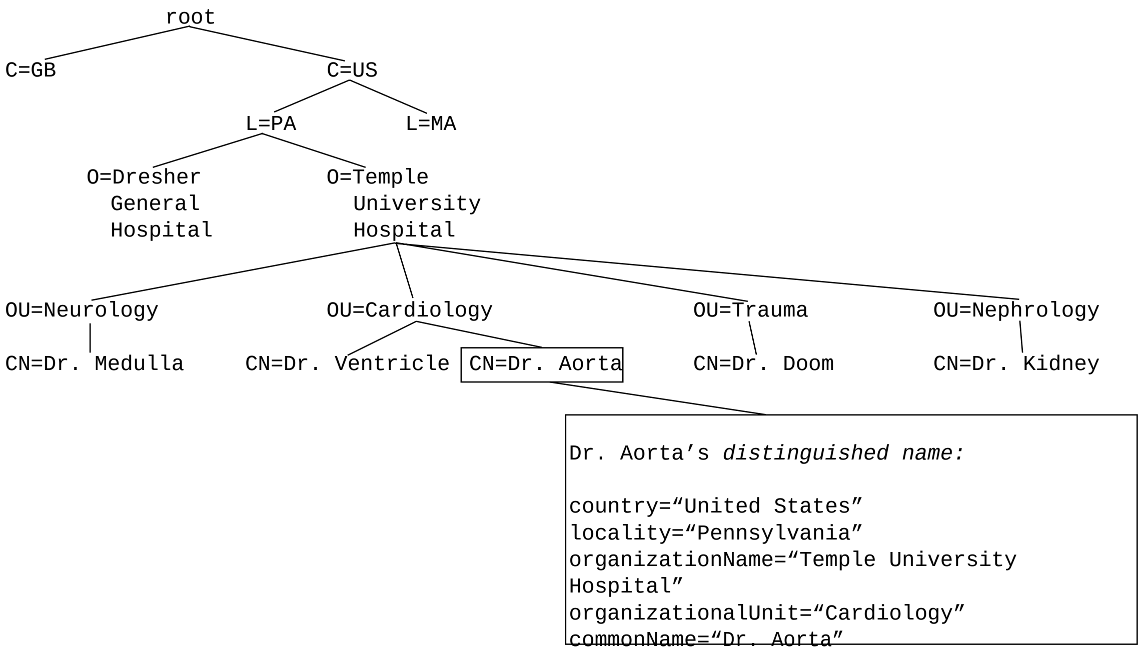

FIGURE 7.5 The Directory Information Tree

guished names and distinguished names. Entries at leaves of the tree are sometimes *alias* entries. They are pointers to object entries and provide alternative names for the objects to which they point.

AHA *It is important to recognize that directory distinguished names are not necessarily user-friendly; user-friendly naming is a property of the directory service, not of distinguished names per se. Thus, "user-friendly naming" occurs not as a natural result of the use of directory distinguished names but as a consequence of directory user agents making judicious use of the directory system to ensure that those distinguished names for which the property of "user-friendliness" is important have it.*

Object Classes Conceptually, a family of objects that have in common a well-defined set of attributes constitutes an *object class*. Object class definitions provide (detailed) "characterizations" so that individual objects (object instances) may be associated with a particular object class. ISO/IEC 9594-7, *Selected Object Classes*, provides an initial set of object classes for use in the Directory. As an example, most people can be associated with a `residentialPerson` object class, which is itself a subclass of *person*. A directory entry for a `residentialPerson` must contain a locality name attribute and may optionally contain any of the following attributes: a locale attribute set (locality name, state or province name, street address); a postal attribute set (physical delivery office name, postal address, postal code, post office box, street address); a preferred delivery method; a telecommunications attribute set (fax, ISDN, telex number, etc.); and a business category. Similarly, many people can be associated with the `organizationalPerson` object class; a directory entry for an `organizationalPerson` may contain a locale attribute set, an organizational unit name, a postal attribute set, a telecommunications attribute set, and a title (position within the organization).

Role of the Directory Like the Domain Name System, the Directory plays an important role in name-to-address resolution; in particular, the Directory may be used to determine the addressing information required for applications to communicate. Two object class definitions—`applicationProcess` and `applicationEntity` (see Figure 7.6)—allow directory providers a means to create entries in the Directory to enable application "lookups" (the usefulness of this service, and the importance of the attributes of these object class definitions, becomes more apparent in Chapters 10 and 11).

```
applicationProcess OBJECT-CLASS
     SUBCLASS of top                 -- all object classes are subclasses of “top”
     MUST CONTAIN {
        commonName }
     MAY CONTAIN {
        description,                 -- a textual description of the application
        localityName,                -- geographic/physical location of application
        organizationalUnitName,      -- unit with which application is affiliated
        seeAlso }                    -- name(s) of other directory objects that
                                     -- describe this application

applicationEntity OBJECT-CLASS
     SUBCLASS OF top
     MUST CONTAIN {
        commonName,
        presentationAddress }        -- see Chapter 5 and Chapter 11
     MAY CONTAIN {
        description,                 -- a textual description of the application
        localityName,                -- geographic/physical location of application
        organizationName,            -- organization with which application is affiliated
        organizationalUnitName,      -- unit with which application is affiliated
        see Also,
        supportedApplicationContext } -- see Chapter 10
```

FIGURE 7.6 `applicationProcess` and `applicationEntity` Object Class Definitions

The Directory Model

The directory information base is distributed throughout the worldwide collection of directory system agents that form the OSI Directory. In the general model, queries are forwarded from a directory user agent, which acts as the agent for a real user, to a directory system agent, which attempts to satisfy the request. In many instances, a local directory system agent can do so by using information it maintains in its own local piece of the directory information base; in cases where the query refers to a part of the directory database for which the local agent has no information, it passes the request to an agent that does (see Figure 7.7).

The organizational mapping and administration of the OSI Directory follow the model applied to the X.400 Message Handling System (discussed in Chapter 8). Directory system agents may operate singly or together as a group to provide a directory service to one or more directory user agents, under a single administration called a *directory management domain* (DMD). If the management domain is operated by a public telecommunications provider, it is referred to as an *administrative directory management domain*, and if the management domain is operated by a company or noncommercial organization, it is referred to as a *private*

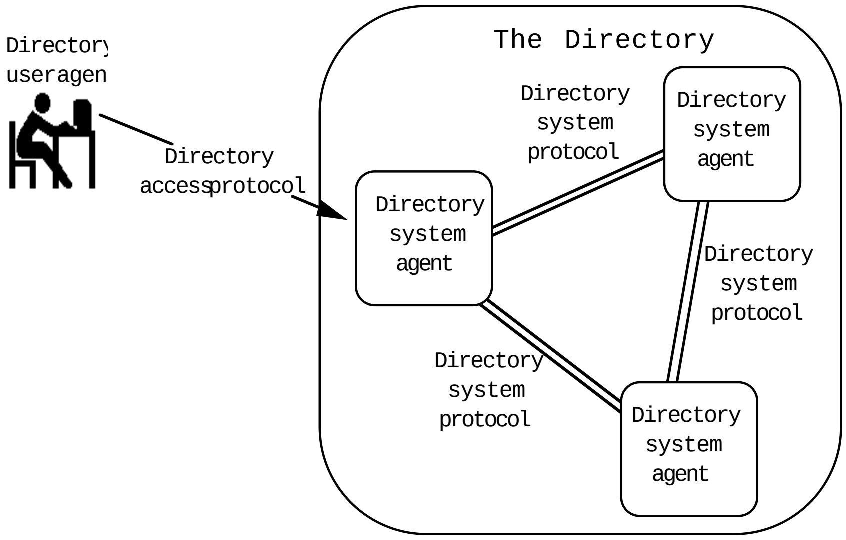

FIGURE 7.7 Directory Model

directory management domain. The organization that administers a directory management domain is responsible for overseeing the creation and modification of directory entries in its DSAs, allocating names to entries, and ensuring the integrity (and privacy, if applicable) of the entry information.

AHA *ISO/IEC 9594-2: 1990 and the X.500 Recommendations say little about the rules governing the interconnection of private and administrative directory management domains. One might assume that the politics and economics that determined the "correct" methods for interconnection of publicly and privately administered Message Handling Systems apply to directory systems as well. Not so. Having learned just how constricting the guidelines were during the early deployment of X.400, those responsible for the definition of the X.500 Directory backed off considerably, and generally speaking, providers of OSI Directory services are encouraged to interconnect in whatever fashion is most appropriate (and legal within the constraints of particular national regulations!).*

The Directory Service

The directory service is provided to users through access points called *ports*. Different types of ports exist for different directory services. A *readPort* is available for reading attributes of a directory entry, comparing an attribute of a directory, and canceling a previous directory inquiry. A *searchPort* may be used to obtain a list of subordinates of a specified, named directory and to search the directory information base for the set of entries that satisfy some filter. Like a "yellow pages" service, this facility enables a user to get a list of entries that all have a common attribute—for example, all organizational persons having the same organizational unit name within an organization or all residential persons sharing the same postal code. Both list and search capabilities of the searchPort facilitate browsing through the database. The *modifyPort* provides the means to add and remove a "leaf" entry from the Directory Information Base, as well as the means to add, delete, or replace attributes of an existing directory entry and to modify the relative distinguished name of a leaf entry. The ASN.1 macros for these ports are shown in Figure 7.8.

```
directory
     OBJECT
          PORTS {
          readPort[S],        -- the directory is treated as an object
          searchPort[S],      -- it is a supplier [S] of services through
          modifyPort[S] }     -- these ports
::= id-ot-directory

dua
     OBJECT
          PORTS {
          readPort[C],        -- the directory user agent is treated as an object
          searchPort[C],      -- it is a consumer [C] of the services provided
          modifyPort[C] }     -- through ports by the directory
::= id-ot-dua

readPort
     PORT {
          CONSUMER INVOKES {
                      Read, Compare, Abandon }
::= id-pt-read

searchPort
     PORT {
          CONSUMER INVOKES {
                      List, Search }
::= id-pt-search

modifyPort
     PORT {
          CONSUMER INVOKES {
                      AddEntry, RemoveEntry,
                      ModifyEntry, ModifyRDN }}
::= id-pt-modify
```

FIGURE 7.8 ASN.1 Macros for the `readPort`, `searchPort`, and `modifyPort`

To access the Directory, a directory user agent *binds* to one of the types of ports offered on behalf of an end user. The BIND ASN.1 macro is used to create an *association* (see Chapter 10) between the Directory and the local directory user agent. To execute a bind operation, an end user indicates the type of port needed and may be required to provide *credentials*; these may be as simple as the user name, or the directory service may require that the user provide stronger credentials, which can be employed to authenticate the user (see "Directory Security Model," later in this chapter). If the bind operation is successful, the user may perform any of the remote operations available through the port type requested.

Using the *read* operation, for example, the user may specify as an argument to the remote operation an object name from which information is requested (e.g., the name of a residential person) and a selection of information that is associated with that name (e.g., the name of the locality in which the person resides). The result expected from the read is the selected entry information; unexpected results (e.g., errors resulting from the incorrect specification of attributes or violation of an access control in the read request) are also accommodated via the read remote operation.

Once a user no longer requires the services of a port, he or she uses the *unbind* operation to release the association between the Directory and the directory user agent that provided access to the Directory.

Directory System Agent Interaction

ISO/IEC 9594-4, *Procedures for Distributed Operation*, describes a framework within which directory system agents may work cooperatively to provide wide distribution of information maintained in the directory database. (Here, each DSA that works in cooperation with other DSAs to provide a directory service is modeled as a single object, and the Directory is modeled as a set of objects.) In circumstances where a directory system agent does not have the information requested by an end user locally available, the agent may use *chained service ports* to communicate with other DSAs and "pass a request" to another DSA.[8] The chained service ports offer DSAs the opportunity to use *chaining*, in which the local DSA, in effect, puts the user "on hold" while it communicates with another DSA to find the requested information, which it then passes back to the user; or *referral*,

8. Although the directory standards admit to the need to replicate information among DSAs, the current ISO and CCITT X.500 standards do not define protocols to support replication, nor do they describe the methods for keeping replicated information current; these are expected in the 1992 extensions (see also Radicati [1992]). During the interim, OSI directory implementations have resorted to proprietary means of propagating information as well as managing how to distribute information or "knowledge" about how directory information has been distributed/replicated.

in which the DSA responds to the user's request immediately with a message to the effect that "I can't answer your question, but here is the name of a DSA that can—go talk to that DSA yourself." A third form of DSA-DSA interaction, *multicasting*, is effectively an extension of chaining; it allows a DSA to issue the same request to multiple DSAs, either simultaneously or sequentially. The ASN.1 macros for the *refinement* of the DSA directory object to include these ports are shown in Figure 7.9.

The `chainedRead`, `chainedSearch`, and `chainedModify` ports complement the read, search, and modify ports in DSAs but are only

```
DirectoryRefinement ::= REFINE directory AS
  dsa  RECURRING
          readPort[S], VISIBLE   -- the DSA provides this port to
DUAs
          searchPort[S],        VISIBLE          --the DSA pro-
vides this port to DUAs
          modifyPort[S]  VISIBLE --the DSA provides this port to
DUAs
          chainedReadPort   PAIRED WITH dsa   -- provided to DSAs
only
          chainedSearchPort PAIRED WITH dsa   -- provided to DSAs
only
          chainedModifyPort PAIRED WITH dsa   -- provided to DSAs
only

dsa
  OBJECT
      PORTS {
          readPort[S],
          searchPort[S],
          modifyPort[S],
          chainedReadPort,
          chainedSearchPort,
          chainedModifyPort }
::= id-ot-dsa

chainedReadPort
  PORT {
      ABSTRACT OPERATIONS {
                          ChainedRead, ChainedCompare, ChainedAbandon
}}
::= id-pt-chained-read

chainedSearchPort
  PORT {
      CONSUMER INVOKES {
                          ChainedList, ChainedSearch }}
::= id-pt-chained-search
```

FIGURE 7.9 ASN.1 `DirectoryRefinement` Macros

supplied to other DSAs (and not to directory user agents). The distributed aspects of the directory model are illustrated in Figure 7.10.

The interaction between DSAs is similar to the DUA-DSA interaction. DSAs use a `DSABind` operation to establish an association with other DSAs, then make use of the services provided through the chained port type—chained read, search, modify—to which they have been bound. When a DSA has completed use of a chained port, it releases the association using a `DSAUnbind`.

Directory Protocols

The OSI Directory standards define a protocol for communication between a directory user agent and a DSA (the directory access protocol [DAP])[9] and a protocol for communication among peer DSAs (the directory system protocol [DSP]); these are defined in ISO/IEC 9594-5, *Protocol Specifications*. The protocols are described in terms of the remote operations the user agent and system agent may perform. Specifically, the directory access protocol is defined in terms of three application service elements—the `readASE`, `searchASE`, and `modifyASE`—which are consumers of the corresponding operations shown in Figure 7.8. The directory service protocol is defined in terms of three other application service elements—`chainedReadASE`, `chainedSearchASE`, and `chainedModifyASE`—which are consumers and suppliers of the corresponding operations shown in Figure 7.9. Figure 7.11 illustrates the way in which these elements are related by the directory protocol model.

Directory Security Model

The directory security model provides for both authentication services and access control. *Authentication services* are provided to verify the ini-

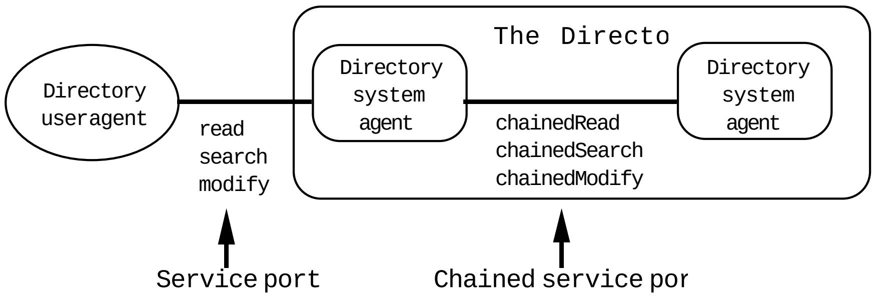

FIGURE 7.10 Distributed Directory Model

9. Note that in cases where a directory user agent and directory service agent are located in the same system, use of the DAP may not be necessary.

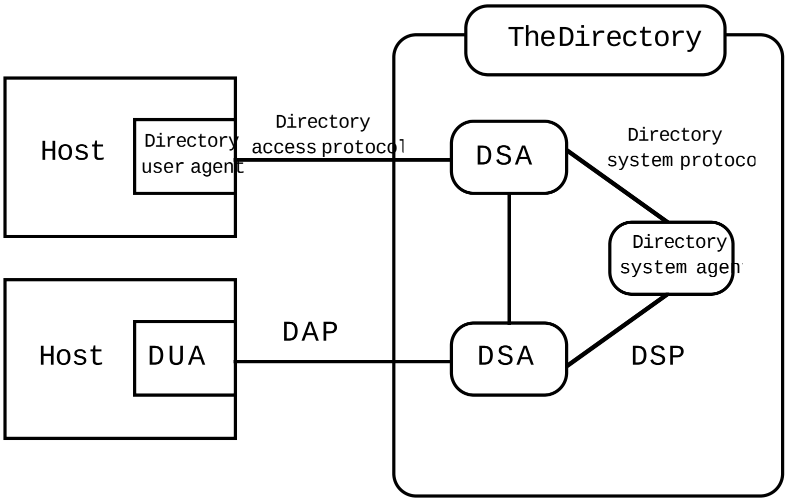

FIGURE 7.11 The Directory Protocol Model

tiator of a directory request; *access controls* are provided to keep parts of the directory information base private from directory users who have not been given the privilege of accessing that information. ISO/IEC 9594-8: 1990, *Authentication Framework*, provides for two levels of authentication. A *simple authentication* uses a password mechanism to verify the identity of a directory user, whereas a *strong authentication* uses cryptographic mechanisms based on a public-key encryption cryptosystem (Diffie and Hellman 1976). The strong authentication specified for the Directory may be used in both the directory-access and system protocols to authenticate the initiator of a request as well as the responder to a request. This is useful in protecting against identity interception, masquerading, and replay. A basic access control—one in which access to directory entries, attributes, or attribute values of an entry can be controlled—is specified in the 1992 version of X.500 and post-1990 revisions to ISO/IEC 9594.

The Relationship Between the OSI Directory and Message Handling Services

The most immediate consumer of the services of a global OSI Directory is the X.400 Message Handling System (see Chapter 8). The Message Handling System requirement for a directory service, which was implied but not stated in the 1984 version of the X.400-series Recommendations, is explicit in the 1988 and later versions. In the 1984 version, the *originator/recipient* (O/R) name used to identify the sources and destinations of mail messages could be only one thing: an O/R address. In the 1988 and later versions of the standard, an O/R name can be either an O/R address (as before) or a directory name (that is, a distinguished name in the context of the X.500 Directory). The 1988 and later versions require that a directory service be either directly or indirectly accessible to the message transfer systems of a message handling system.

A message handling system is likely to make use of a directory system for at least the following services:

- *User-friendly O/R names:* If the O/R names for mail users may only be O/R addresses, users must deal directly (and frequently!) with lengthy, cumbersome, and pedantic strings such as `/c=us/admd=mcimail/prmd=nermc/o=bbn/s=Chapin/g=Lyman.` If, on the other hand, a relative distinguished name in the Directory may be used as an O/R name, then mail users may deal directly with much friendlier (for one thing, shorter!) strings, such as "Lyman Chapin" (a clear improvement). In this example (and assuming that the Directory has been properly configured to place the "Lyman Chapin" relative distinguished name at the appropriate place in the directory information tree), a distinguished name entry would appear in the directory for `c=us@o=nermc@ou=bbn@commonName=Lyman Chapin`, which can be transformed in a straightforward (and standardized) way into an O/R address.
- *Expansion of distribution lists:* When multiple recipients of electronic mail are combined in a distribution list, the name of the list can appear as the O/R name in a mail message. The Directory can be employed to support the use of a user-friendly directory name, rather than an O/R address, for a distribution list, in the same way as it supports the use of directory names for individual message recipients. It can also be used to support the expansion of a distribution list by the message transfer agents responsible for it, since the Directory can store all the O/R addresses for the message recipients included in a distribution list under the single relative distinguished name of the distribution list.
- *Message Handling System user searches:* A Message Handling System user who lacks sufficient information about a message re-cipient to properly address a message may—outside of the Message Handling System itself—use the search capabilities of the Directory to find the missing information (for example, the recipient's full "common

name") based on whatever partial information the user may have (such as, perhaps, the recipient's first name and phone number).

The components of the X.400 Message Handling System use the Directory in the same way as any other Directory user; there are no special protocols linking X.400 components to X.500 DUAs or DSAs. The directory service required by the 1988 and later X.400 standards need not, therefore, be provided by X.500, nor indeed by any "global" directory. Each Message Handling System component deals with a local DUA or DUA equivalent, and no formal coupling between the Message Handling System and the Directory is required as a matter of conformance to the standards.

The OSI Directory in the Internet

The X.500 Directory is potentially far more powerful and comprehensive than the patchwork combination of `hosts.txt` files, Domain Name System, and other information services and locators that are currently used to perform directorylike functions in the Internet. Since the approval of the 1988 X.500 Directory Recommendations, there has been considerable interest in the use of the OSI Directory in the TCP/IP Internet.[10] A very large number of "pilot" projects—experiments in the use of the OSI Directory service in the Internet—are today operational and interconnected. They share a common global `root` directory maintained at the University of London Computer Centre by the PARADISE project.

PARADISE

PARADISE, the COSINE X.500 Directory service pilot, was launched in November 1990 to coordinate an international directory service for the European research and development community. PARADISE provides some services itself, such as a user interface to the directory service, and as of November 1991, also serves as a link between national pilots in the 18 countries that are participating in COSINE. PARADISE is managed by University College, London, and involves the University of London Computer Centre, X-Tel Services, and a group of public service providers, including PTT Telecom in the Netherlands, PTT Switzerland, and Telecom Finland.

10. The use of the term *TCP/IP Internet* here is intentional; it refers to the use of the OSI Directory to maintain information about the traditional TCP/IP protocol suite in the Internet, in addition to its use to maintain information about OSI protocols that may be supported by a multiprotocol Internet.

One of the PARADISE services is a public-access interface to the Internet OSI Directory system. The interface, called "`de`" (for "directory enquiries"), is distributed with ISODE (the ISO Development Environment) releases 7.0 and later. Users of "`de`" can find information about people and organizations that are listed in the directory, using a query model that supports approximate matching and a variety of "wild cards." Users can also list entries in the Directory—people within a department, departments within an organization; organizations within a country, or countries represented in the Directory.

The public-access interface is easy to use. Users with access to the Internet can use TELNET to connect to the host `paradise.ulcc.ac.uk.` A sample interactive session is shown in Figure 7.12. (Note that user-entered text is in bold typeface.)

```
SunOS UNIX (found.paradise.ulcc.ac.uk)

login: dua
Last login: Thu May 28 17:46:52 from 134.246.150.51
SunOS Release 4.1.1 (DUA) #4: Tue Apr 21 11:37:06 BST 1992

        Welcome to PARADISE - the COSINE Directory Service

Connecting to the Directory - wait just a moment please ...
You can use this directory service to look up telephone numbers and electronic mail
addresses of people and organisations participating in the Pilot Directory Service.
You will be prompted to type in:

:- the NAME of the person for whom you are seeking information
:- their DEPARTMENT (optional),
:- the ORGANISATION they work for, and
:- the COUNTRY in which the organisation is based.

On-line HELP is available to explain in more detail how to use the Directory Service.
Please type ?INTRO (or ?intro) if you are not familiar with the Directory Service.

?                     for HELP with the current question you are being asked
??                    for HELP on HELP
q                     to quit the Directory Service (confirmation asked unless at the
request
                       for a person's name)
Control-C             abandon current query or entry of current query

Person's name, q to quit, * to browse, ? for help
:- p kirstein
Department name, * to browse, <CR> to search all depts, ?
                            for help
:-
Organisation name, * to browse, ? for help
:- univ london
Country name, * to browse, ? for help
:- uk
United Kingdom
```

```
Got the following approximate matches. Please select one from the list by typing the
number
  corresponding to the entry you want.

United Kingdom
  1 Brunel University
  2 University College London
  3 University of London Computer Centre
Organisation name, * to browse, ? for help
:- 2
United Kingdom
                            University College London
                              Computer Science
                                Peter Kirstein
                                  description         Head of department
                                  telephoneNumber     +44 71-380-7286
                                  electronic mail     P.Kirstein@cs.ucl.ac.uk
                                  favouriteDrink      not while on duty
                                  roomNumber          G01
                                  X.400 mail address  /I=P/S=Kirstein/OU=cs/O=ucl/
                                                        PRMD=UK.AC/ADMD=GOLD 4
00/C=GB/

Person’s name, q to quit, <CR> for ‘p kirstein’, * to browse, ? for help
:-
```

FIGURE 7.12 Sample PARADISE Interactive Session

X.500 Implementations

Interoperability among X.500-based directory systems is primarily a matter of lineage; the many different systems deployed in the Internet today are the direct descendants of just a handful of original implementations. The most widely used implementation of the OSI Directory is called QUIPU (Hardcastle-Kille 1992), originated from University College, London, a product of the Integrated Network Communication Architecture (INCA) project. A rival X.500 implementation, developed at the Institut National de la Recherche en Informatique et Automatique (INRIA) in France under the auspices of the ESPRIT project Thorn, is called, not coincidentally, “Pizarro.”

A list of currently available implementations of X.500-based directory systems, with particular emphasis on implementations that are designed to operate in the Internet TCP/IP environment, is contained in RFC 1292, *A Catalog of Available X.500 Implementations.*

Other Internet Directory Utilities

Only two of the formal directory services and their application in the Internet have been discussed thus far. These services are relatively new. In the rich history of the Internet, two earlier directory applications—WHOIS and FINGER—are noteworthy, as they represent earlier attempts at providing "name-to-attribute" services.

WHOIS

The WHOIS service supports a limited form of name-to-attribute mapping for IP networks and IP network administrators in the Internet. It is primarily used by system postmasters or other administrators to find the "point of contact" for an Internet site. A centralized WHOIS database is maintained by the Internet's Network Information Center at host `nic.ddn.mil`. WHOIS servers are also distributed throughout the Internet wherever individual sites choose to run the BSD "whois" program for access to local and/or remote WHOIS databases. WHOIS (formally, "NICNAME/WHOIS") is specified by RFC 954 (1985).

X5WHOIS, developed as part of the FOX project,[11] is a variant of the WHOIS server program that provides access to information in the `@o=Internet@ou=WHOIS` subtree of the X.500 Directory. (One of the activities associated with the deployment of X.500 pilots in the Internet has been to load the contents of the NIC's master WHOIS database into a subtree of the Internet X.500 directory.) SRI International, in Menlo Park, California, provides a public-access X5WHOIS server that may be reached in one of two ways:

1. Through the regular BSD `whois` program:
   ```
   % whois -h inic.nisc.sri.com <search string>
   ```
2. Through TELNET to `inic.nisc.sri.com`:
   log in as `x5whois` (no password) and type one-line `<search string>s`

The X5WHOIS server looks like a normal directory user agent to the X.500 Directory and uses the standard X.500 directory access protocol to query X.500 DSAs with the user-supplied `<search string>`.

FINGER

The FINGER protocol (formally, "NAME/FINGER") is specified in RFC 1288, updated in December 1991. The FINGER protocol has been implemented for UNIX systems (i.e., the `fingerd` daemon), and for a small number of non-UNIX systems, to provide an informal (and highly idio-

11. FOX—Field Operational X.500—is a pilot X.500 directory project funded jointly by the Department of Energy (DOE), the National Aeronautics and Science Agency (NASA), the National Science Foundation (NSF), and DARPA; coordinated by USC/ISI; and operated jointly with Performance Systems, Inc., Merit, Inc., and SRI International.

syncratic) mechanism for discovering information about a user logged in on a local or remote Internet host. When invoked on one host—with, for example, the command line `finger dave@mail.bellcore.com`—it returns information about "Dave Piscitello" obtained from the remote host's `mail` operating system and (optionally) from that user's `.plan` and `.project` files:

```
[Mail]
Login name: dave        In real life: Dave Piscitello

Office: 1C322, x2286    Home phone: n/a

Directory: /u/dave      Shell: /bin/ksh

Last login: Tue Apr 7 10:12 on type from thumper.bellcore.com

Project: all manner of fast packet technologies

Plan: to make public networks a "safe space" for datagrams
```

The usefulness of this service is severely limited by the fact that one must already know a person's user name and host name in order to obtain information from FINGER; by the fact that few non-UNIX systems support the service; by security concerns, which cause many site administrators to disable it; and by the very restricted query model, which supports FINGERing only a specific user on a specific host.

Resource Location

The proliferation of information that is stored "somewhere in the Internet" has promoted a familiar problem to the top of many current networking research agendas: how does one locate the specific information that one needs? A directory service can help to identify the potential sources of information, but it is impractical to construct a directory that is both efficient in the performance of its principal task (that of mapping various identifiers, such as mail user names, to a list of attributes, such as the address of the mail transfer agent to which mail for that user name should be forwarded) *and* capable of processing complex, incompletely specified queries such as "Where can I find information about research on low-temperature fusion in Great Britain since 1991?"

As the library science community discovered the convenience and boundless opportunities associated with the networking of libraries, the new field of *networked information retrieval*—the term generally used for the problem of locating and retrieving information resources that are accessible by means of a network—was born. The authors can only scratch the surface here, but the following projects are particularly interesting examples of the way in which systems designed to discover "information about information" far beyond the capabilities of traditional directories have been designed.

Archie

Archie (a play on the word *archive*) began as a project at McGill University to address the problem of how to quickly and easily scan the offerings of the already bewildering and rapidly growing number of anonymous FTP sites scattered around the Internet. The current system—which is accessible through an interactive TELNET session,[12] by electronic mail, and through command-line and X-window clients—accepts queries such as "Where can I find the following file . . . ?" and returns a list of anonymous FTP archives that contain the named file; for example, the request

```
% archie rfc-index.txt
```

returns the host names and directory locations where the `rfc-index.txt` file resides, i.e.,

```
Host nic.ddn.mil
  Location: /rfc
    FILE -rw-r-r-    20166 May 28 1992 rfc-index.txt
Host nnsc.nsf.net
  Location: /rfc
    FILE -rw-r-r-    20224 May 28 1992 rfc-index.txt
```

and so on. (There are a fairly large number of Internet sites that maintain a copy of the `rfc-index.txt` file!)

The shortcomings of such a service are obvious: it searches only for specific file names (and you have to know the exact name of the file—"fuzzy" matches are not supported), and it searches only Internet anonymous FTP archive servers. It is, however, a dramatic improvement over nothing at all.

Wide Area Information Service

The goal of the Wide Area Information Servers (WAIS) project is to facili-

12. There are public-access Archie servers all over the Internet, including:
`archie.mcgill.ca` (the first Archie server, at McGill University in Montreal)
`archie.funet.fi` (in Finland)
`archie.ans.net` (in New York)
`archie.au` (in Australia)
`archie.doc.ic.ac.uk` (in the United Kingdom)
`archie.rutgers.edu` (at Rutgers University in New Jersey)

tate the growth of a distributed system of information servers and clients based on the ANSI standard bibliographic search and retrieval protocol (ANSI Z39.50-1988). WAIS is a much more ambitious undertaking than Archie; it is a general-purpose search and retrieval system with two significant characteristics:

1. It uses the standard Z39.50 protocol to search documents and document indexes stored in a wide variety of repositories (not just Internet anonymous FTP archives).
2. It supports a unique, user-oriented search model that closely matches the searching strategy with which people are already familiar: (a) start with a few key words or phrases; (b) see what WAIS retrieves; (c) tell WAIS which of the retrieved articles, or sections of articles, are most relevant to the subject of your search and ask it to search again using your selections as models; and (d) repeat the process until you've found what you want.

The "search from a good example" strategy makes WAIS a very powerful tool, since it not only provides appropriate feedback to the user during the search but also permits the scope or even the original purpose of the search to be changed, iteratively and interactively, as the search proceeds. WAIS also has a built-in accounting system: the client search screens include an explicit "cost" field, which presents both a statement of what it costs (or would cost) to make a particular query and a running decremented count of "how much money you have left."

WAIS uses Z39.50 over TCP/IP, modem, OSI, and other networks; the motivation for using a standard protocol is to eventually be able to work with a wide variety of standard bibliographic search and retrieval systems that are being developed by the library science community. It was originally developed at Thinking Machines Corp. in Cambridge, Massachusetts, by Brewster Kahle, who turned it over to the WAIS Clearinghouse at the nonprofit Center for Communications Research at Research Triangle Park near Charlotte, North Carolina.

A simple WAIS public-access interface is supported by the NSFnet Network Services Center; connect using TELNET to `nnsc.nsf.net` and log in as `wais`, with no password.

WorldWideWeb

The "WorldWideWeb" project is an ambitious attempt to make all online information readily accessible to users as a "web" of documents and links among them. It was originally developed in 1989 by Tim Berners-Lee, Robert Cailliau, and Jean-François Groff at CERN for use by the high-energy physics community but has expanded far beyond its original target audience.

The goal of WWW is to merge the techniques of "hypertext," in which links between pieces of text (or other information, such as video frames) emulate human associations among related ideas, and text retrieval, which allows associations to be formed based on the content of text or other information media.

WWW "browsers," as the client interfaces are called, are designed to make it easy for users to pursue the links that are displayed from current information to other information, as this example from a simple line-mode browser illustrates:

```
WORLD WIDE WEB

    The WorldWideWeb (W3) is a wide-area hypermedia[1]
    information retrieval initiative aiming to give univer-
    sal access to a large universe of documents.

General Project Information

    See also: an executive summary[2] of the project,
    Mailing lists[3] you can join, Policy[4] , latest W3
    news[5] , Frequently Asked Questions[6] .

  Project Status[7]      A list of project components and
                         their current state. (e.g. Line
                         Mode[8] , X11 Viola[9] , X11
                         Erwise[10] , NeXTStep[11] ,
                         Daemon[12] )

  People[13]             A list of some people involved in
                         the project.

  Bibliography[14]       Paper documentation on W3 and ref-
                         erences.

  History[15]            A summary of the history of the
                         project.

  How can I help[16] ?   If you would like to support the
                         web.

1-29, Back, <RETURN> for more, Quit, or Help:
```

Wherever a link exists to related information (shown, in this example, as square-bracketed numbers), the user can follow the link by typing the number at the prompt; other browsers use a "point and click" interface, which would show links by highlighting the words or phrases to which they were attached (rather than by assigning a number to them, as in the terminal-oriented interface used for this example).

Conclusion

The Domain Name System is used in the Internet today principally to perform a mapping from some form of name to an Internet (IP) address. As the Internet evolves to a multiprotocol environment, it is desirable that the DNS and/or other directory systems evolve to a system in which the information returned by a name-based query includes multiprotocol (not just IP) addresses, other names, security parameters, protocol-stack information, and other "attributes." The desire to generalize DNS is quickly tempered by the realization that the basic "name-to-address" translation is a fundamental and real-time operation, that it lies in the critical path for real-time applications, and that it cannot be generalized at the expense of efficiency. By contrast, the "name-to-arbitrary-object/attribute-list" translation is not as time-critical, and the OSI Directory accommodates this aspect of directory services nicely, with the potential for providing a platform for many if not all of the information and resource locators mentioned earlier. One of the issues the Internet community faces as the Internet becomes ever more multiprotocol in nature is how to evolve DNS alongside X.500 Directory services. DNS simply can't—and won't—go away: the fact that the X.500 architecture does not currently break out the two different classes of "directory" capability nicely makes it difficult to "simply use X.500" in situations in which a name must be matched with an Internet address very quickly. The DNS, however, is expanding to accommodate OSI network addressing, and although it is easy to speculate how X.500 might evolve so that it serves two different purposes—real-time name-to-address translation (the function of the DNS) and non–real-time white pages and yellow pages services (both to human users and to distributed-system applications)—it appears that both directory applications will play important roles in the Internet.