CHAPTER 2

An Overview of Directories and the Internet

A DIRECTORY FOR THE DOGS

As an example of a typical directory request that illustrates property-based information retrieval, consider a directory that contains information about dogs. Imagine that a directory client wants to know about breeds of dogs that are black and weigh less than 20 pounds. In this situation, the directory client would present two facts that it knows about objects that are in the directory:

- Weight < 20 pounds
- Color = Black

The directory server that the client contacts is expected to return information about all dog breeds of which it is aware, that have a weight property that is smaller than twenty, and a color property equal to black. The precise information that is returned depends upon the type of directory service, and what information is available, but would include other properties of the people that matched the request, such as breed name, height, weight, etc. Figure 2.1 shows this interaction between the directory client and the directory server.

There are numerous kinds of directories that have been defined by the Internet. Most directory services are identified by the protocol that is used to communicate between directory clients and directory servers,

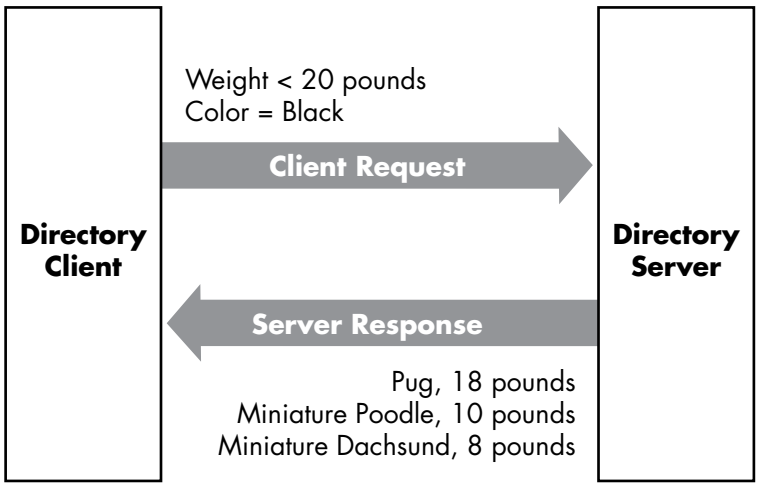

Fig. 2.1 Directory/client server interaction

even though there may be other protocols that are defined by this service that allow for communication among the directory servers. These server-to-server communication protocols are often termed *back end* protocols.

Compare this notion of directories to that of the widely used Internet search engines. The directory allows the client to supply some properties and proposed values, and the server returns matching entries and actual values from those entries. The search engine is specifically designed to go to pages on web sites that want to be searchable and reads them, using hypertext links on each page to discover and read a site's other pages. Then, using this information, the search engine has a program that creates a huge index (sometimes called a "catalog") from the pages that have been read. This index or catalog can then be searched by users to find web sites of interest. The typical search engine then allows the user to supply some words and returns some web pages that it has searched that have some type of match with the supplied words. Notice that in the directory the client request has a definite property-value structure, while the search engine request is relatively unstructured.

A Security Primer

This section introduces network security. Network security provides safeguards against various threats that may be targeted against computer networks. As directories are a network application service, they could be prime targets for these threats. Typical attacks against a directory

involve the theft of information that is stored in the directory and the denial of service to a directory client. If a rogue user gains access to the directory, then that user may be able to change the directory information and cause the directory to provide incorrect answers to the clients' queries.

In order to provide a complete directory service some level of network security must be provided. But what parts of security are relevant to directories? Users need to prove who they are since some data that the directory stores is marked with access control information indicating who is allowed to read and write it. In some systems, proof of users' identities is accepted by just trusting the IP address from which the request comes, or by sending a password across the network, neither of which is secure. A better way is through cryptography, in which you provide knowledge of a secret without divulging it. Using cryptographic techniques allows a directory client to prove to the directory server that it knows the password without ever sending the password across the network. Thus, even if there is an eavesdropper on the connection between the client and the server, the user's password is still safe.

Authentication is the process of determining whether someone or something is, in fact, who or what it claims to be. It is often the case that a directory client must be authenticated to the directory server before being granted access to the directory information. Secure forms of authentication involve the process in which you provide knowledge of a secret without divulging it. There are two types of secrets: the first type is where both sides (i.e., the client and the server) know the same secret. The other type is known as public key schemes, in which each user has a pair of keys, one public and one private, which are mathematically related such that you can verify if someone knows the private key by using the public key. Authentication is the primary use of security in directory systems.

Various security mechanisms that are used by directories are introduced here. A complete discussion of network applications can be found in Kaufman, Perlman, and Speciner's book, *Network Security: Private Communication in a Public World*.[1] The most basic tool used in network security is *encryption*. Encryption allows data to be modified into a form that allows it to be hidden from unauthorized people. The encrypted form of the data is known as the cipher data. Decryption is the process that these authorized people use to transform the cipher data back into the original data. Authorized people encrypt and decrypt data by making use of *keys*

[1] Published by Prentice Hall PTR, 1995.

that are used by the encryption or decryption process. A key is a short piece of data that is known only to the authorized people. Keeping the key secret allows the encryption algorithms to be published. The typical use of encryption in directories is to keep certain data private as it is transferred across the network between the directory client and server. There are a large number of encryption algorithms that can be used to provide this privacy; the most widely used algorithms fall into two categories:

- *Public-key* algorithms use a pair of keys that are created in a special process that allows one key to be used in encryption, and the other key to be used in decryption. One of the keys is kept secret (the *private* key), and the other key is made widely available (the *public* key).
- *Secret-key* algorithms allow only a single key to be used for both encryption and decryption.

A few notes about these two types of encryption algorithms:

- Public-key algorithms are substantially slower than secret-key algorithms. In fact, public-key algorithms are so much slower that they are rarely used for encrypting data that is larger than a few kilobytes.
- The keys used in public-key algorithms are much larger than the keys used in secret-key algorithms. The key used in the popular RSA public-key algorithm defined in RFC 2437 is normally 256 bytes to provide adequate security. The key used in the popular RC5 secret-key algorithm defined in RFC 2040 is normally only 8 or 16 bytes.
- Great care must be taken in transmitting the key used in the secret-key algorithm among authorized users. If any unauthorized users gain access to the key, then any data encrypted using that key has been compromised.

Secret-Key Encryption

The most common secret-key encryption algorithms operate on a fixed length segment of data at a time, usually 8 or 16 bytes. The algorithms take this data segment in combination with the secret key as input data in order to produce the encrypted data. The encrypted data is almost always the same size as the input, plain text data. When the data to be encrypted is longer than the segment that the algorithm is designed to accept, the plain text data is broken into several blocks which are

encrypted one at a time. For example, if a document to be encrypted is 100 bytes long, and the encryption algorithm operates on data blocks that are 8 bytes long, then the document would be divided into 13 blocks. Note that the last block of data would not be 8 full bytes, but only 4 bytes long. In order to decrypt the data, the algorithm uses the encrypted data along with the same secret key that was used in the encryption.

Some algorithms are defined to encrypt each block independent of all the other blocks of data. Alternatively, the algorithm can use the results of a previous block in the input to the encryption of the next block. The details of any particular encryption algorithm are beyond the scope of this book. However, it can be assumed that if a document has been encrypted with a strong secret-key encryption algorithm, then the encrypted data may be safely transmitted across the Internet. As long as only the originator and the intended recipient know the key used in the encryption process, any malicious intruders that may intercept the document cannot decrypt the document.

Public-Key Encryption

Two keys are required, in public-key encryption algorithms. One key is used in the encryption process, and another key is used in the decryption process. In public-key technology, the two keys must have some sort of special relationship to each other, and be generated by a special mathematical process at the same time. Each pair of keys belongs to a user. The user will publish one of the keys (i.e., the public key) in order to make it available to other users. The second key (i.e., the private key) is kept confidential and not made available to anyone else. For example, if Alice wants to send Bob a secret message, she would retrieve Bob's public key (perhaps from a known directory) and use it to encrypt the message. Once the message has been encrypted, only Bob can decrypt it using his private key. Thus, even though Alice knows the public key, the plain text data, and the encrypted data, she still cannot derive Bob's private key. This is due to the special mathematical relationship between the two keys. In one popular encryption algorithm, the attempt to derive the private key would require the potential attacker to factor a large number. This large number is in the range of the size of the key. If the key is 1024 bits, then the attacker would have to factor a number in the range of 2^{1024} power, which is a number with more than a hundred decimal digits and, therefore, nearly imposssible to guess or derive.

Public keys should be widely published. If Alice published her public key so that it is widely available, anybody who needs to send her encrypted data can easily retrieve Alice's key and securely send her information. A

good way of publishing public keys is by storing them in a directory. A directory client can provide Alice's e-mail address, and the directory server will be able to perform the lookup to find the entry in the directory, which contains Alice's public key, and return it to the end user.

Message Digests, Digital Signatures, and Authentication

Another security algorithm of special interest is the message digest algorithm. A message digest algorithm takes any size document as input (i.e., the message) and produces a fixed size data block as output. This fixed size data block is called the *message digest*. For example, the popular MD5 message digest algorithm is described in RFC 1321: The MD5 Message Digest Algorithm**.** MD5 produces a 16-byte message digest of its input. The message digest is also called a fingerprint because of the analogy to a person's fingerprint. Just as it is extremely difficult to find two people with the same fingerprint, it is also extremely difficult to find two documents that produce the same MD5 message digest. A good message digest algorithm has the property that it is computationally infeasible to produce two messages having the same message digest. Similarly, it is also computationally infeasible to produce any message having a given prespecified target message digest.

A message digest algorithm must have these properties to be useful in the creation of digital signatures. If Alice wants to create a digital signature for a document, she must first create the message digest of the plain text document. Then Alice will create the digital signature using her private key. She can then send the plain text document along with the document's signature to Bob. Bob can verify the digital signature by first creating the message digest of the plain text document. Then he will verify the digital signature using Alice's public key. If the decrypted digital signature and the message digest that Bob created are identical, then Bob has *verified* Alice's signature. In verifying the digital signature, Bob is guaranteed of two facts:

- ☞ The document that Bob received is precisely the document that Alice sent, and it has not been altered en route.
- ☞ The document was actually sent by Alice and no one else. This is due to the fact that no other person could have created the digital signature since it required the use of Alice's private key, and only Alice has access to her private key.

Digital signatures are especially useful in directories for the purpose of authentication. The digital signature process can be used in this scenario. Once the client connects to the server, the server provides the client with a piece of data that the client must sign. Once the server verifies the signed data, the server is assured of the identity of the client, and the client can continue operating on the directory. Additionally, digital signatures are also useful so that the information in the directory can be signed. This allows the directory clients to trust the information in the directory (especially if the information has been signed by a trusted agent).

THE INTERNET

Now that the basic concepts of directories have been introduced, the concepts of the Internet that are important to directories can be discussed. The Internet refers specifically to the original network that was funded by the United States Department of Defense Advanced Research Projects Agency (DARPA) known as ARPANET. Since its inception in the late 60s the ARPANET has evolved to a network that connects millions of hosts across the world. These hosts are all connected by protocols that are known as the TCP/IP suite of protocols. When an Internet-like network is contained inside an enterprise, it can also be called an intranet.

This section will not attempt to explain the entire Internet suite or protocol stack. That will take the perspective of viewing the Internet stack from the perspective of a directory, and discussing what parts of the Internet Suite directories use. For example, the services offered by the Internet Protocol (IP) and the Internet Core Message Protocol (ICMP) are not directly used by directory services, and so won't be discussed. However, an understanding of the Transport Control Protocol (TCP), Transport Layer Security (TLS, also known as Secure Sockets Layer or SSL), and the User Datagram Protocol (UDP) are directly used by directory services, and thus will be introduced. However, before discussing these services specifically, it is important to understand how TCP/IP works so that the features that are available to the Directory are known.

The suite of protocols used by the Internet is known as the TCP/IP suite because the principal protocols used by Internet applications to communicate are the Transport Control Protocol (TCP) and the Internet Protocol (IP). The TCP/IP suite of protocols is typically viewed as a stack, in which one layer is piled upon another. This is an indication that the layers at the top of the stack make use of services at the layers toward the bottom of the stack. A view of a portion of the TCP/IP stack is shown in Figure 2.2.

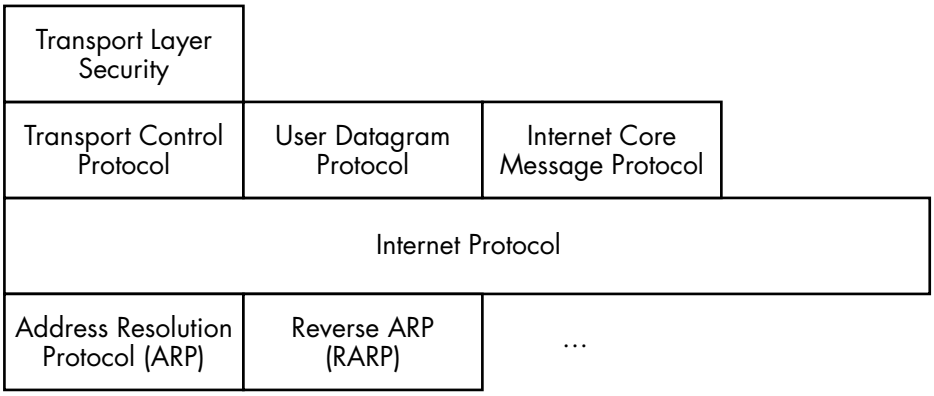

Fig. 2.2 Upper layers of the Internet Protocol Stack.

In this figure, each layer in the stack provides a set of services that are available to the layers that reside logically on top of it. TLS directly makes use of the services provided by TCP, but does not use any of the services that are offered by UDP or ICMP. Furthermore, TLS is generally ignorant of the services that are offered by the IP layer, and does not directly make use of those services. Internet applications, such as web browsers, connect to Internet servers, such as web servers, by creating a connection known as a *socket* between the application and the server. A socket can be viewed as a pipeline between the application and the server through which data may be exchanged once it is created.

The creation of a socket is very analogous to the dialing of a telephone call. Once the telephone number is entered, the destination line rings, and when the receiving party answers the ringing telephone on their end, a telephone connection is established. In order to create a socket connection between two entities on the Internet, the calling party (known as the client) enters an Internet address (the format of which will be discussed shortly). This Internet address is made by some process running on the client's machine which attempts to make a connection to some process that is running on the machine named by the given Internet address. If there is any process listening for connections on the destination machine, then the connection can be established. All of the Internet transports that are used in Internet directories (i.e., TCP, UDP, and TLS) make use of sockets for communication between clients and servers, but use different types of sockets. However, the sockets for all of the transport types have similar behavior. Thus, when they are used for the simple transport of data between the client and the server, the different socket types can be used by the directory entities in virtually the same way.

During the attempt to create a socket, the client and server go through a process known as *handshaking*. During the handshaking process, the client and server each exchange some information before the socket can be created. If either side is not satisfied with the information that is provided by the other side (known as its peer), then the attempt to create the socket is broken off, and no socket connection is created. For example, a socket server may be configured in such a way that it only allows sockets to be opened by clients from a specified set of hosts. During the handshaking process the client and server exchange their address information. If the client's address is not one of those that the server is configured to accept, then the server will reject the client's attempt to create the socket.

During any handshaking process at the Internet Transport Layer, the client and server exchange information in order that (among other things) they may be able to identify each other. In the real world, people identify each other by any number of means—names, telephone numbers, electronic mail addresses, etc. In the Internet, peer entities are able to identify each other by several different means, but by far the two most common mechanisms are Internet addresses and Internet host names. The following section presents an overview of the upper layers of the TCP/IP stack, from the top of the stack first, since those layers are directly used by the directory.

THE TLS LAYER

A very simplified view of the handshaking that occurs in the TLS layer is shown in Figure 2.3.

In this view of the handshaking that occurs during the attempt by a TLS client to open a socket with a TLS server, both the client and the server send two pieces of information across the network prior to a successful TLS socket creation. The client initiates the handshaking when it sends a special message defined by TLS, known as a *client hello*. This message contains various parameters that define those kinds of TLS services that are being requested. For example, the TLS client can request that the connection be encrypted by any of several different means. It can also request that any data being passed across the socket is to be compressed. The client hello message also includes some randomly generated data that aids in the creation of the encrypted connection. The server responds to a client hello with another special message that is defined by TLS, known as a *server hello*. The main piece of information that is included in the server hello message is information that is unique

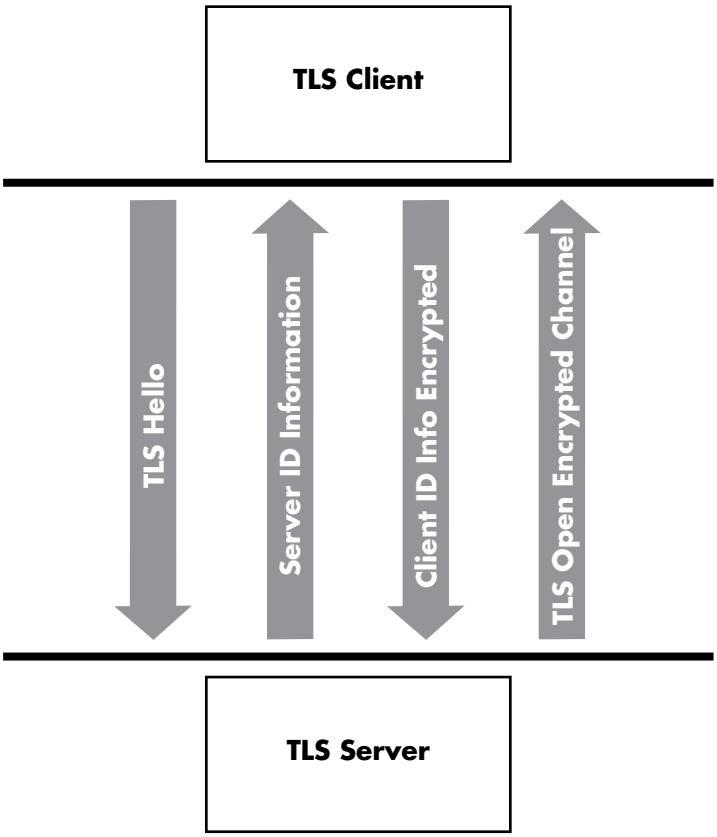

Fig. 2.3 TLS handshaking

to the server, known as a *certificate*. The server presents its certificate in such a way that the client can verify that it really does belong to the server. This verification is known as *authentication*. The precise details of how this authentication works are not particularly important to the functions of Internet directories, but a short overview of the authentication process will be discussed in Chapter 4, "Directory Management." The important feature of this first stage in the handshaking process is that it has allowed the TLS client to verify that the TLS server to which it has connected is indeed the one that it intended to contact.

Once the client has authenticated the server, the second phase of the TLS handshaking can begin. The objectives of the second phase are to allow the server to authenticate the client, and for the creation of an encryption key that allows all data passed across the TLS session to be kept confidential. If the server requested client authentication in its server

hello message, then the client is obligated to provide its certificate information in the subsequent message. At this time the client uses the random data that it provided in the client hello message along with information that is in the server certificate to generate the encryption key. Simultaneous to this, the server is performing the same process, thus guaranteeing that the client and server have generated the same key, often known as a *shared secret*. If the server is satisfied with the client certificate information that the client has provided, then it provides a response that indicates that the connection has been opened successfully. (Keep in mind that this explanation of a TLS connection creation has been greatly simplified, and, while accurate, many details have been omitted.)

THE TCP LAYER

TCP is the Transport Control Protocol. It takes care of providing a reliable connection between two Internet nodes (e.g., hosts). For example, TCP nodes synchronize with each other and number each packet that is sent between them. For each packet that is sent from one host to the other, an acknowledgment is returned. If a host does not receive an acknowledgment for a sent packet, that packet is presumed to have been lost and is retransmitted. The TCP handshaking and socket setup process is much simpler than that of TLS. When a TCP client wishes to open with a TCP server, the client transmits a special message that indicates it wishes to *synchronize* sequence numbers with the server. This message, known as a SYN message, is the first step in the TCP handshaking. The steps in the TCP handshaking are illustrated in Figure 2.4 (taken from RFC 793, which defines TCP).

RFC 793 describes this process, known as the *3 way handshake,* as follows: The TCP client begins by sending a SYN segment indicating that it will use sequence numbers starting with 100. Next, the TCP server sends a SYN and acknowledges the SYN it received from the TCP client. Note that the acknowledgment field indicates that the TCP server is now expecting to hear sequence 101, acknowledging the SYN, which occupied sequence 100. Finally, the TCP client responds with an empty segment containing an ACK for the TCP server's SYN. With this third message, the TCP client and server have successfully negotiated a socket connection.

The synchronizing of the TCP sequence numbers is particularly important to the reliability of the TCP layer. As RFC 793 indicates, the TCP must recover from data that is damaged, lost, duplicated, or delivered out of order by the IP layer. This is achieved by assigning a sequence number to each octet transmitted, and requiring a positive acknowledg-

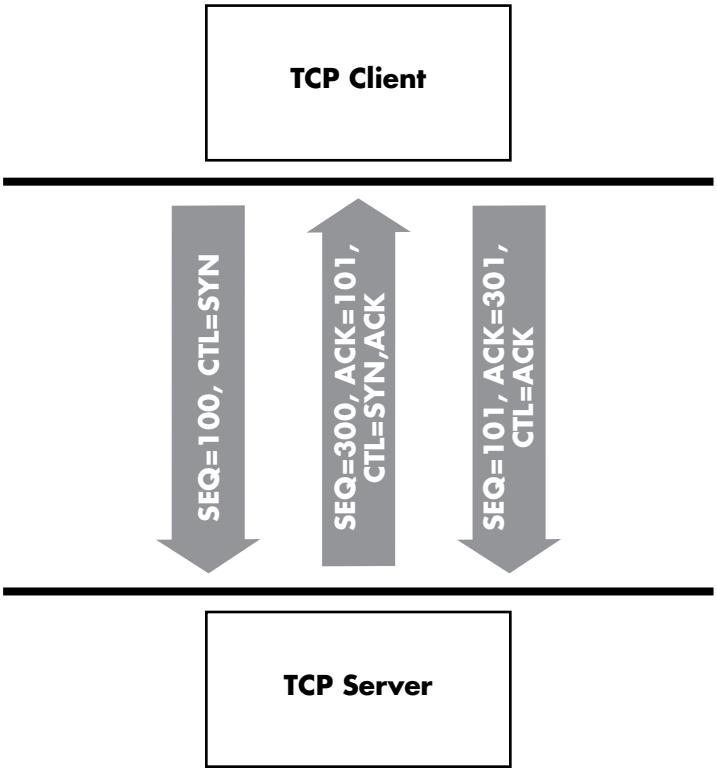

Fig. 2.4 TCP handshaking

ment (ACK) from the receiving TCP. If the ACK is not received within a timeout interval, the data is retransmitted. At the receiver, the sequence numbers are used to correctly order segments that may be received out of order and to eliminate duplicates. Damage is handled by adding a checksum to each segment transmitted, checking it at the receiver, and discarding damaged segments.

THE UDP LAYER

TCP is an inherently reliable protocol (using the above notion of reliability). This is due to the fact that packets are sent again when lost or somehow damaged in transit. Its companion protocol, User Datagram Protocol (UDP), is considered to be an inherently unreliable protocol. This unreliability is due to the fact that the guarantees of TCP are not made by UDP. UDP does not recover from data that is damaged, lost, duplicated, or delivered out of order by the IP layer. UDP does not have a handshak-

ing process that takes place. UDP is designed for situations in which all that is needed is a single message and a single response. Thus, no ongoing connection between a UDP client and server is maintained. UDP is defined in RFC 768, which indicates that UDP provides a procedure for application programs to send messages to other programs with a minimum of protocol mechanisms. Applications requiring ordered reliable delivery of streams of data should use TCP. UDP operates by the UDP client opening a socket to a UDP server and sending some data across the socket. The UDP server replies to this request with whatever data is appropriate, and the UDP socket is then closed.

TYING THE LAYERS TOGETHER

Most hosts that are reachable on the Internet have been assigned one or more Internet Protocol (IP) addresses, and usually have been assigned a host name as well.[2] The IP is the numeric form, while the host name is the textual form. For convenience, the numeric addresses are normally represented textually in the *dotted notation* form, e.g., 127.0.0.1, rather than as the raw 32-bit or 64-bit number. Humans prefer the host name, as the names generally have some intrinsic meaning. For example, the web site for the Prentice Hall publishing company is located on the Internet host *www.prenhall.com*. Host names are the text strings that appear on the right side of the "@" in electronic mail address. For example, the author can be reached via e-mail at: bgreenblatt@directory-applications.com.

Version 4 of IP, which is the current ubiquitously deployed version, allows for the IP address to be 32 bits, which is large enough to hold approximately four billion different addresses. The recently approved version 6 of IP allows for 64 bit addresses, which is large enough to hold over four quintillion (i.e., 4.6×10^{18}) addresses.

In addition to knowing the name or address of the server on which it wants to open a socket, the client must also know the number of the port to which the server is connected. To allow for many processes within a single host to use TCP communication facilities simultaneously, the TCP provides a set of addresses or *ports* within each host. Concatenated with the network and host addresses from the IP layer, this forms a socket. This definition, taken directly from RFC 793, indicates that the purpose of the TCP port is to allow for many servers to operate on the

[2] Modern techniques allow hosts to participate on the Internet without having an IP address (or host name) assigned to them.

same Internet host at the same time. A TCP port is identified by an integer. In order to make this happen, each server must listen to a different incoming port. If a server attempts to listen on a port to which another server is already listening, TCP will return an error to the second server that indicates that the port is busy. Both UDP and TLS support the same implementation of port numbers, in that only one server is allowed to listen on a port at a time. Historically, Internet application services have reserved ports when they were defined. The Internet Assigned Numbers Authority (IANA) keeps track of all port numbers that have been assigned. Some of the more notable assignments for service contact ports are named in Table 2.1.

Table 2.1 Port Number Assignments for Some Internet Protocols

Protocol Name	TCP Port	UDP Port	TLS Port
Telnet	23	23	992
SMTP	25	25	465
DNS	53	53	
Whois	43	43	
Whois++	63	63	
Finger	79	79	
Http	80	80	443
Ldap	389	389	636
Rwhois	4321	4321	

Note in the above table that not all Internet application protocols have been assigned TLS ports. Port numbers are divided into three separate ranges:

- Well Known Ports—numbered from 0 through 1023
- Registered Ports—numbered from 1024 through 49151
- Dynamic or Private Ports—numbered from 49152 through 65535

Thus, DNS uses a well-known port while Rwhois uses a registered port. The distinction between these two port types is minor, in that typically only processes or programs that are run by the most privileged users can listen on a well-known port number. If a service has a well-known or registered port assignment, then clients of that service can assume that the default configuration of the server has the server listening on the assigned port. Thus, finger clients normally assume that there is a finger server listening on port number 79 on most Internet hosts, and that there

is never a server that doesn't understand the finger protocol listening on that port. This means that when the finger client attempts to open a socket to a finger server, it will try port 79, and it will both succeed and talk to a finger server, or there will not be a finger server active on that server. It will never be the case that the finger client will attempt to open the socket, and that there is a server listening on port 79 that does not understand the finger protocol (for example, a web server).

More information on the definition of the Internet may be found in the Internet document, FYI 20, entitled "What is the Internet?" as well as any number of other published references.

INTERNET DIRECTORIES

Recall from our earlier discussion that a directory is an application service that primarily performs property-based information retrieval. Directories store objects of various types. Each object that is stored has properties. For example, dog objects have properties, such as color, breed, height, weight, age, etc. A directory that contains objects of this type would allow clients to retrieve information about dogs based on the properties that have been defined.

As they relate to the Internet, directories perform various necessary and useful functions. For example:

- They allow for the resolution of host names to underlying IP address.
- They allow for the creation of an Internet Public Key Infrastructure (PKI) in order to allow for the secure exchange of information across an insecure network.
- They allow for controlled access to resources across a distributed network.
- They allow for location of a server based upon the type of the service (e.g., electronic mail) rather than upon the name of the server.
- They allow for the exchange of index information among themselves in order to facilitate the routing of queries to the appropriate server. This function allows each server to have some knowledge about the data that is contained on many other servers.

Directories provide many useful services for the Internet. But the Internet also provides many useful functions for directories. TCP provides a reliable means of transporting data from client to server. TLS

allows for the directory to provide a secure means of transporting data from client to server. TLS also allows the directory peers to provide a strong means of identifying each other, rather than simply passing user IDs and passwords across the network.

The previous section provided an overview of the Internet, and the notion of Directories as a service that allows for property-based information retrieval has been touched upon. Putting the Internet and directories together yields the concept of the Internet directory**.** An Internet directory is a service that has *property-based information retrieval* as its primary function, and uses one or more of the *Internet transports* (TLS, TCP, or UDP) as its native means for communication between the client and server. The two most prominent Internet directories are the Domain Name System (DNS) and the Lightweight Directory Access Protocol (LDAP).

DNS

DNS is an Internet standard that is defined in RFCs 1034 and 1035. The primary goal of the DNS directory service is to provide for the mapping of Internet host names to IP addresses. In terms of the property-based information retrieval concept, the objects that are stored in the DNS directory are Internet hosts. The properties that are available for retrieval are host names and IP addresses. DNS clients, known as *resolvers,* send requests to DNS servers. In the DNS, resolvers typically provide a server with a host name, and the server provides the resolver with the IP address that corresponds to the provided host name. The growth in the Internet was the impetus for the development of DNS. As the number of hosts attached to the Internet grew beyond several hundred in the early 1980s, scalability problems with the previous mechanism for providing the name to address mapping were exposed.

As described in RFC 1035, prior to the implementation of DNS, host name to address mappings were maintained by the Network Information Center (NIC) in a single file (HOSTS.TXT) which was copied by all hosts [RFC-952, RFC-953]. The total network bandwidth consumed in distributing a new version by this scheme was proportional to the square of the number of hosts in the network; the outgoing FTP load on the NIC host was considerable. Explosive growth in the number of hosts didn't bode well for the future. Furthermore, the network population was also changing in character. The timeshared hosts that made up the original ARPANET were being replaced with local networks of workstations. Local organizations were administering their own names and addresses,

but had to wait for the NIC to change HOSTS.TXT to make alterations visible to the Internet at large. The proposals for the replacement of this mechanism varied, but a common thread was the idea of a hierarchical name space, with the hierarchy roughly corresponding to organizational structure, and names using "." as the character to mark the boundary between hierarchy levels. The implementation of DNS met the requirements with a distributed database of information rather than a centrally administered hosts file. The distributed administration of directory administration devised by DNS was to become a hallmark for virtually all of the Internet Directory services that were to follow. The implementation and protocols involved in DNS will be discussed in significant detail in a subsequent chapter.

LDAP

The Lightweight Directory Access Protocol (LDAP) was defined as a result of the desire to pursue implementation of the X.500 series of recommendations of the International Telecommunications Union (X.500) on the part of Defense Advanced Research Projects Agency (DARPA). X.500 defines several different models and protocols that are used in the implementation of directories. The most notable protocol defined by X.500 is the Directory Access Protocol (DAP). DARPA wanted to deploy directories based on the X.500 series, but their implementation was slow in coming. DARPA decided to fund a research project at the University of Michigan that would result in the definition of a different version of DAP that would be significantly easier to implement, but would still retain the core features of the X.500 model. The end product of this research project was LDAP.

An early definition of LDAP was experimental in nature, and the first widely implemented definition of LDAP was LDAP version 2, as defined in RFC 1777. Due to deficiencies in the areas of security, internationalization, and extensibility, a third version of LDAP was defined in RFC 2251. RFC 2251 indicates key aspects of this version of LDAP:

- ☞ All features of LDAPv2 (RFC 1777) are supported. The protocol is carried directly over TCP or other transport, bypassing much of the session/presentation overhead of X.500 DAP (which is defined on top of the OSI protocol stack).
- ☞ Most of the data that is passed between LDAP clients and servers can be encoded as ordinary strings (X.500 uses various binary data types to encode its information).

- Referrals to other servers may be returned when the server initially contacted by the LDAP client does not have enough information in order to completely fulfill the client request.
- Any mechanism may be used with LDAP to provide security services that can be used in the authentication step between the client and the server.
- Attribute values and distinguished names have been internationalized to allow for any character (e.g., in the Chinese character set) to be used in LDAP strings.
- The protocol can be extended to support new operations, and controls may be used to extend existing operations.
- Clients publish schemas in the directory for use so that the types of information that are available for retrieval are available as part and parcel of the normal information that is published by LDAP servers.

The last point is especially interesting. In the context of Internet directories, the schema defines the types of objects and the properties of those objects that are available for retrieval by clients. Since LDAP clients are the ones that are capable of publishing the information that appears in directories, it is only natural that the clients are allowed to publish (and retrieve) the schema for that information. While appearing natural, this innovation is new in version 3 of LDAP, and allows clients to be able to find out about new types of objects that are stored in the directory, and to determine the precise syntax that is used in the properties of these new types of objects. This notion of dynamic schema discovery is a marked difference between LDAP and earlier directories. In DNS, resolvers are expected to have full knowledge of the various types of records that are maintained by DNS servers. There is no defined way in the DNS scheme of things for resolvers to understand new record types on the fly.

INTERNET DIRECTORY REQUIREMENTS

In the previous sections of this chapter, various functions of Internet directories have been discussed, but the fundamental requirements for a directory service has only been touched on. In traditional software development, the requirements gathering phase is the first part of developing software and it defines the external behavior of the software system to be built. In terms of Internet directories, the requirements indicate the features of the clients that are made available to their users. Note that not

all directory services implement all of these requirements. For example, as we mentioned previously, DNS does not implement schema discovery. This section is meant to describe, in a general way, the types of features that are available in many Internet directory services, in order to distinguish them from other types of application services.

The foremost requirement of an Internet directory service is to allow for property-based information retrieval. Regardless of the type of information that is stored in the directory, each object that is stored has various properties, and the directory service must allow for clients to retrieve this information based on these properties.

Data Storage

Internet directories store their data in such a way that the properties and protocol that are used fit in naturally with the rest of the Internet. For example, multimedia objects are typically represented in the Internet by making use of the MIME (Multipurpose Internet Mail Extensions) structure. MIME was first defined as a way to transport various types of binary data across the Internet in electronic mail messages. However, it has come to be used in numerous protocols that need to define ways to transport multipart, possibly binary data. For example, MIME is used not only in electronic mail, it is also used in LDAP, the Common Index Protocol (CIP), the Hypertext Transport Protocol (HTTP) used in the World Wide Web, and many other Internet application protocols.

In the original days of the Internet, data was assumed to be stored in the United States (U.S.) version of the ASCII character set. U.S. ASCII represents each character as a single byte, the high order bit of which is always zero. The resulting 128 different characters that are defined by U.S. ASCII include the 26 upper and lowercase letters, the 10 digits, and various other punctuation and control characters. Of course, information that is transported across the Internet needs to include characters from many different cultures outside the US. For example, European and Oriental characters are not represented in U.S. ASCII. The French word "çiel" can't be represented using U.S. ASCII, since the character 'ç' is not one of the 128 characters defined by ASCII. In order to represent such kinds of information, the Universal Character Set (UCS) repertoire was devised. UCS character sequences are normally represented on the Internet by using the specification known as UTF-8. UTF-8 is defined in RFC 2279, which is titled, "UTF-8, a transformation format of ISO 10646." The International Organization for Standardization (ISO) standard numbered 10646 defines the multibyte character set known as UCS.

UCS characters are either two bytes long or four bytes long, and they use the full range of possible two or four byte values.

The point of UTF-8 is that it encodes UCS characters as a sequence of one or more 8-bit ASCII characters. UTF-8 defines an encoding mechanism that has the characteristic of preserving the full U.S. ASCII range. It also provides a mechanism for encoding characters outside of this range in more than one byte. Table 2.2 (taken directly from RFC 2279) summarizes the format of these different octet types. The letter "x" indicates bits available for encoding bits of the UCS-4 character value.

Table 2.2 UTF-8 Character Encoding

UCS-4 range (hex.)	UTF-8 octet sequence (binary)	Number of 8 bit characters needed
0000 0000-0000 007F	0xxxxxxx	1
0000 0080-0000 07FF	110xxxxx 10xxxxxx	2
0000 0800-0000 FFFF	1110xxxx 10xxxxxx 10xxxxxx	3
0001 0000-001F FFFF	11110xxx 10xxxxxx 10xxxxxx 10xxxxxx	4
0020 0000-03FF FFFF	111110xx 10xxxxxx 10xxxxxx 10xxxxxx 10xxxxxx	5
0400 0000-7FFF FFFF	1111110x 10xxxxxx ... 10xxxxxx	6

Thus, the UTF-8 encoding of the word "Directory" is precisely the same as the original ASCII encoding of that same word. Some other examples from RFC 2279 of character encoding are:

- The UCS-2 sequence "A<NOT IDENTICAL TO><ALPHA>." (0041, 2262, 0391, 002E) may be encoded in UTF-8 as follows:

```
41 E2 89 A2 CE 91 2E
```

- The UCS-2 sequence representing the Hangul characters for the Korean word "hangugo" (D55C, AD6D, C5B4) may be encoded as follows:

```
ED 95 9C EA B5 AD EC 96 B4
```

- The UCS-2 sequence representing the Han characters for the Japanese word "nihongo" (65E5, 672C, 8A9E) may be encoded as follows:

```
E6 97 A5 E6 9C AC E8 AA 9E
```

☞ Since UTF-8 encoding is the Internet Standards track definition for international character representations, when representing data outside of the US-ASCII character set, Internet directories should make use of UTF-8 encoding for that data, as specified in RFC 2279.

Protocol Usage

The protocols that are used in Internet directories should be carried directly on top of an Internet transport, i.e., TCP, UDP, or TLS. Native integration of applications with the TCP/IP stack makes integration with future enhancements to this stack more likely to be smooth. For example, TLS has been designed in such a way that creation and deletion of TLS sockets is done in virtually the same way as the creation and deletion of TCP sockets. For example, even though TLS had not yet been invented at the time that LDAP v2 was released, LDAP v2 clients have no problem in changing from the use of TCP to access directory servers to using TLS to access those same directory servers. LDAP v2 clients thereby gained the advantage of encrypted sessions, server authentication, and other TLS provided services without any change at all in the LDAP protocol definition.

Distributed Operation

Internet directories should operate in a distributed manner in such a way as to allow consistent access to their information throughout the Internet. Not all Internet hosts are uniformly available from any site on the Internet. This is due to a wide variety of factors, not the least of which are geographic considerations and other bandwidth-related concerns. However, access to directory information should not suffer from these problems. It must be possible to allow multiple directory servers to provide services for the same set of objects. This allows directory clients to access whichever directory server is most conveniently located. In order to illustrate this requirement, consider Figure 2.5.

The point here is that any of the clients can present their query ("What breeds of black dogs weigh under 20 pounds?") to any of the servers and expect to get the same answer back. Due to geographical and other bandwidth considerations, the timing will be different for each of the clients working with each of the servers. Therefore, Internet directory servers are required to cooperate in order to present a uniform view of the data they manage to Internet directory clients. Historically, directory servers that do support the notion of information sharing or replication have defined their own information sharing protocols that are

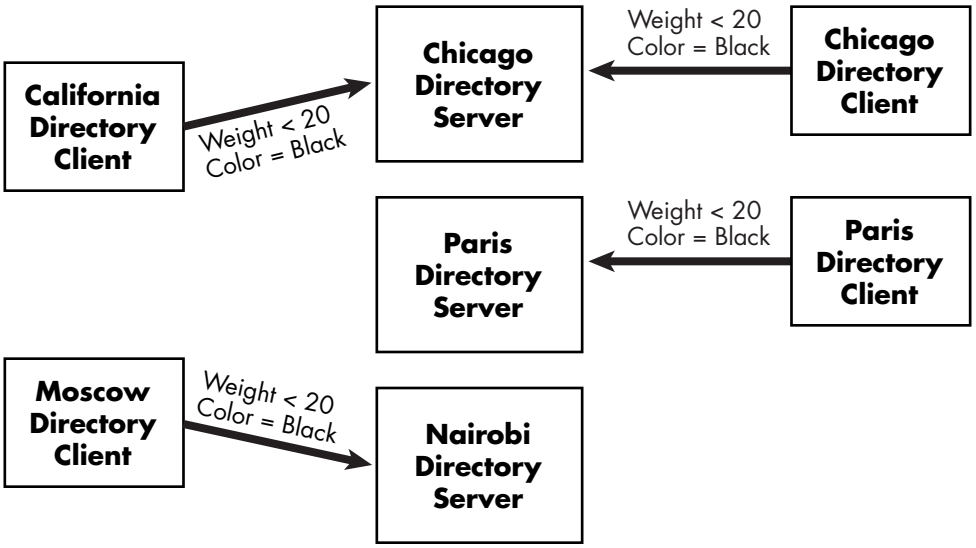

Fig. 2.5 Directory replication

specific to one type of directory (e.g., DNS, X.500, etc.). RFC 2651 defines the Common Indexing Protocol (CIP). CIP allows directory servers to share and publish much of the information that they contain, and CIP is not limited to any particular protocol.

Because directory servers cooperate to provide a service, they will often not be able to fulfill a client request. In this situation, the directory server that is being queried may be able to refer the client to another directory server that can be consulted with this query. This notion of *referrals* is illustrated in Figure 2.6.

In this figure the directory client first submits its query about small black dogs to the Chicago directory server. The Chicago directory server is unable to fulfill this request, and refers the directory client to the Paris directory server. The directory client *chases* the referral that is provided, and presents the same query to the Paris directory server. In this instance, the Paris directory server is able to fulfill the client's request, and furnishes an appropriate response. Thus, a referral is an indication by a directory server that it does not have the information needed to return that desired result. The referral contains that information needed by the directory client for it to be able to contact some other directory server. Directories can gather knowledge of the information that is held by other directory servers. This knowledge can be manually gathered by the directory administrator, or it can be automated by the use of back end directory protocols.

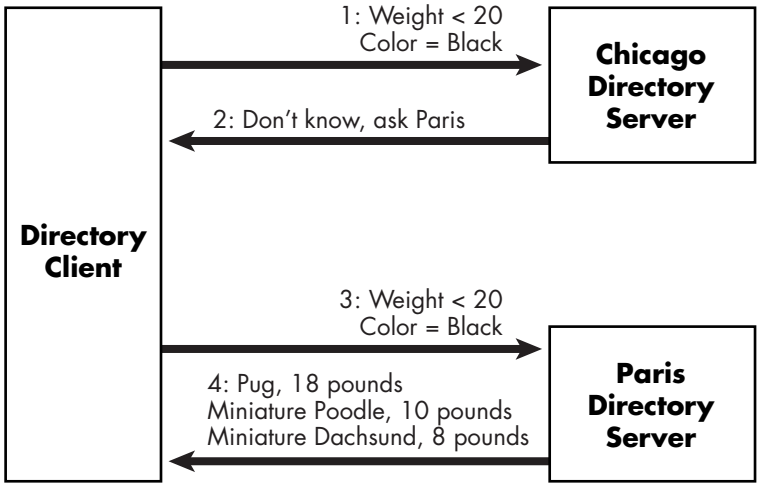

Fig. 2.6 Directory Referrals

WHITE PAGES SERVICE

One of the main uses for a directory on the Internet is for the provision of an Internet White Pages Service (IWPS) in which the client furnishes some properties of a user that it knows about, and the service responds with various types of address information about this user. In the context of the telephone system, a white pages book, in which a list of names are arranged alphabetically, is often used. If the white pages client knows the name of a listed telephone subscriber, then the white pages service will supply the client with a telephone number and possibly a street address. An IWPS, as a special type of white pages service that is located on the Internet, is intended to provide Internet-related addressing information. In the IWPS, the principal addressing information that is to be returned by the IWPS server is the electronic mail address of a user. Other types of related information can also be returned. For instance, RFC 2148 "Deployment of the Internet White Pages Service" describes in great detail the requirements for an IWPS, and RFC 2218 "A Common Schema for the Internet White Pages Service" defines the data that is to be maintained by an IWPS server. A small subset of the user properties that are defined by RFC 2218 is:

E-mail
Certificate
Home Page

Given Name
Surname
Organization
Country
Personal Phone
Personal Fax

From the list above, it can be seen that some of the properties are those that are generally provided by the IWPS client in a request (e.g., Given Name, Surname, Country), while others are generally provided by the IWPS server in its response (e.g., E-mail, Certificate, Personal Phone). For example, an IWPS client in a search for someone's electronic mail address could provide:

Given Name = "John"
Surname = "Smith"
Organization = "Prentice Hall"

while the IWPS server would respond with *john.smith@prenhall.com*. An IWPS was historically connected with electronic mail packages, and often called an address book service. An IWPS has been confused by many with the directory service itself. This important distinction is made between a service and an application of that service. The directory service is a general-purpose property-based information retrieval service, while an IWPS is a special purpose service optimized for retrieval of user's addressing properties.

A Simple Directory

Consider as an example of a directory service a system that converts IP addresses into host names. This directory fits the definition that is being used for a directory (a property-based information retrieval service). The service will operate as a client server application protocol. The client, when attempting to retrieve a host name for a known IP address, will open a socket to the Internet host that corresponds to that IP address. The service operates equally well on TCP, UDP, or TLS. Once the socket has been successfully opened, the client uses the protocol to request the host name for the server that accepted the incoming socket. The server responds to this request by sending its host name and closing the socket connection.

The property that the client knows about is the IP address of the server, and the property that it is attempting to retrieve is the host name of the server. The question for the protocol designers was: "what information does the client need to present to the server in order to relay the information request?" It turns out that once the client opens the socket no further information needs to be transmitted by the client. So, the client only needs to indicate that it is ready for the server to transmit the data. This can be represented in the protocol as the sequence of ASCII characters, carriage return, and line feed. The hexadecimal representation of this in the protocol that is transmitted over the socket is "0D0A." Once the server receives this data, it is expected to respond with its host name. While this example may seem a bit contrived, it turns out (as will be seen later) that this protocol really exists as a subset of the Internet finger protocol. Try using a telnet application to connect to port 79 on an Internet host (the finger port), and hitting the enter key. If a finger server is listening, it should tell you the host name.

CHAPTER SUMMARY

This chapter provided a general purpose definition of a directory as a *property-based information retrieval system*. This definition will be used throughout the book. A simple example of a directory was given. This chapter defined computer and network security as they are used by directories. Additionally, the various upper layers of the Internet protocol stack were defined. Finally, a quick overview of some of the important directories was given.

With this foundation of computer networking and security, as well as a quick overview of some directories and their applications, we can progress into more detailed definitions in the chapters to come. The following chapters will give detailed descriptions of DNS, LDAP, and many of their applications, as well as shorter overviews of some other Internet directories.