Chapter 2

Introduction to Identity-Management Models

Introduction

The elements of security in computing begin with an identity. An *identity* is a computer representation of an active entity that can be physical (such as a human, a host system, or a network device) or can be a programming agent. Such an agent can be assigned a well-known system function (such as a running daemon) or a program delivering a business function on behalf of some entity. Modern systems adopt a fine level of identification sustainable even at the basic computing tasks and execution threads of an address space and may cross the boundaries of single computing systems with the advent of network and distributed computing.

The evolution of computing to automate more and more of the aspects of human interactions such as in business transactions led to the need of identity representation in computing that reflects that of real-life entities such as human beings. An identity therefore evolved from being simply an assigned identifier to an identifier that points to various attributes and entitlements, collectively referred to as a *profile*. *Identity management* has therefore emerged to address the issues surrounding the proliferation of identity profiles among various computing platforms within the boundaries of an enterprise and cross-enterprises and organizations to even the Internet. Foremost of these issues is the cross-referencing among profiles that represent the same identity as well as the synchronization of attributes among these profiles.

We begin by providing a taxonomy of identity models that is based on the scope of an identity, the naming space in which it is uniquely known and used. We discuss the local identity scope, followed by the network and then the global scope. For each we present the benefits as well as the limitations. The global identity model is exemplified by the XNS approach, a novel method that holds the promise of an elegant Web identity-management model. Lastly, we discuss the emerging model of enterprise-level identity management as exemplified by the latest technologies. Without some level of assurance, an identity cannot stand by its own. After all, it is merely a representation of

some active entity. In Chapter 3 we cover the foundations of identity trust and discuss various mechanisms that are currently available.

Identity-management paradigms in computing have taken a natural course that is analogous to real-life practices to a great extent. An individual person initially has direct knowledge of some people that he or she can identify with. That individual further builds knowledge of other persons by directly coming in contact with them or by way of introductions performed by existing acquaintances. The scope of individual identities varies from one person to another. An individual may be known only to his or her family, immediate neighbors, or a workplace; another person can be known throughout his or her locality or a much bigger geography; while some are known all over the globe. The scope of an identity in computing follows in a similar fashion. An identity can be known locally, known over a network of computing devices, or perhaps universally known. Knowledge of some entities can be direct, by way of a registration, or can be indirect, through some other brokering entity.

An individual person can be associated with multiple digital identities in the same manner he or she can be known to other people through multiple nicknames. Regardless of the number of identities one might be associated with, there is an increasing need in computing that all should unambiguously point to the same individual. Each such individual is uniquely identified by a set of attributes, commonly referred to as a *profile* and more recently a *wallet*. We divide the space of identity management along the scope in which an identity is known. We distinguish four classes of identity management that we list in the order of increasing scope as follows:

- Local identity,
- Network identity,
- Federated identity, and
- Global Web identity.

Local Identity

This paradigm evolved with centralized computing. A host system maintains and manages a local registry of identities (users). Computational units are all identifiable with identities locally known to the system. An external entity that wishes to use the system is required to acquire an identity for use with that system. The adopted namespace of identities is flat and is in reference to the local system. A newly added identity is expected to be unique with respect to the names already in the registry. Addition and revocation or removal of identities are discrete operations that do not side-effect other identities. Managed entitlements are associated with the privileges one might have over the local system resources. This model offers the advantage of simplicity. *Capacity scaling* and the *flat name space* are issues that it faces. Figure 2.1 represents a high-level view of the local identity model. In A each system

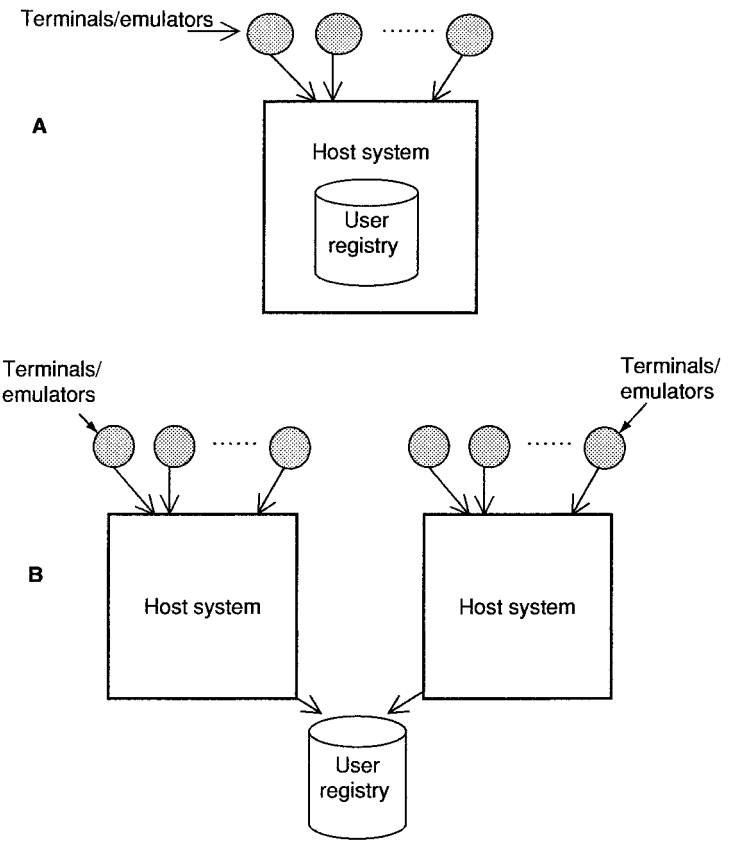

FIGURE 2.1 The local identity model

maintains a separate user registry, while in B the registry is shared across multiple systems. Sharing of user registries is an attempt by the local identity model to alleviate the overhead of the host-centric identity management by registering users only once and allowing them to have access to multiple systems.

Advantages of the Local-Identity Model

Simplicity

The simplicity of this model is mainly the result of the local scope of managed identities and the flat naming space that is generally adopted. Establishing an identity is a simple local process that compares the credential presented by an entity to that stored in the host registry for the same entity.

The flat naming model lends itself to the adoption of flat data constructs with relatively simple structures. Identities are managed as discrete entities except for when they interleave through group memberships. The centralized identity attributes are easy to administer but remain meaningful only within the scope of the host system.

Scalability

Scalability has two dimensions: one is capacity and the other is performance. In the local identity model, the issue of capacity becomes apparent as the population of users and subsystems using or running on the system grows. The system has to store and manage identity information for every such entity. The paradigm is that of directly "knowing everyone." This pushes the limits and capability of the user registry and may result in a performance downgrade. User groups are not considered a remedy to the issue of capacity scaling as identities need to be discretely defined and managed irrespective of group memberships.

Flat Name Space

The flat name space generally adopted in the local identity-management model sets a limitation on the scope of an identity and results in name collisions. The scope of an identity is confined to the host system in which it is defined. Name collisions will occur sooner or later as users select names that already represent other users in the system. The resolution to that usually comes in the form of names that are suggested by the system and that may not reflect the nature of a friendly name chosen by the user. Because an identity is known in reference to the system where it is defined, an identity can be used on multiple hosts without having to be associated with the same entity.

Management Issues in the Local-Identity Model

Each system is associated with its own local identity registry. Users, applications, and subsystem components need to maintain the credentials required for them to establish identity on each of the operating systems used. Passwords, the most prevalent method by which identities are established, are inherently associated with a number of issues. These issues are more apparent and prevalent in the local-identity model. We discuss some of them below.

Password and Attribute Synchronization

The proliferation of passwords on various systems and applications naturally makes it difficult to keep track of them. *Password synchronization* is an alternative solution that mitigates this problem by having each user adopt a single password for all systems. A synchronization mechanism automatically communicates a password change or reset to the participating systems. Unlike single-sign-on (SSO) solutions, however, the user still has to explicitly use the

password for each system or application that requires it. Password synchronization is a much easier approach than single sign-on and does not require drastic changes to an organization's existing infrastructure as might be the case with SSO mechanisms. Similarly keeping various user attributes synchronized is a challenge in the local-identity model. Ultimately, synchronizing user attributes in this case tends to be a manual process which increases overhead and can be error prone. A communications means across registries of different systems is required to automatically synchronize passwords and user attributes across multiple systems in this case.

One solution to this problem is for multiple systems to share a single user registry. This method dispels concerns over synchronizing user passwords and attributes. It may, however, lead to a performance problem due to the registry becoming a bottleneck. To alleviate this, the single registry can be replicated locally across the participating systems.

Single Sign-On

SSO further advances the state of art as represented by password synchronization in that it lets a user establish his or her identity once. Thereafter, access to other applications and systems networked together becomes seamless as it alleviates the use from the burden of reauthenticating. Various SSO implementations have been developed. In homogeneous environments where a single authentication technology is used such as the case with Kerberos, SSO is automatically achieved. In the local-identity model with a stand-alone user registry, SSO is meaningful only across subsystems and applications deployed on the system such as database and transactional systems. The user authenticates once to the system; thereafter a security context is established and passed to different systems by the system runtime functions.

Identity Provisioning

This relates to the processes and procedures in use for the creation, revocation, and deletion as well as the maintenance of user accounts. This is an aspect common to all identity-management schemes, but it presents more overhead in the case of the local-identity model. This is because the effort of provisioning identities is proportionate to the number of systems used by an organization. Furthermore, related issues such as password reset and update tend to increase the cost of identity management. Centralized enterprisewide identity-provisioning tools are becoming the solution of choice to these issues. We discuss these later in the chapter.

Example: IBM Resource Access-Control Facility

The IBM Resource Access-Control Facility (RACF) providing security for the IBM MVS operating system family (recently evolved into z/OS) defines a user by way of creating a profile in its registry [IBMC02]. Information stored

TABLE 2.1 The main elements of the base segment in a RACF user profile.

Attribute	Description
USERID	Identifies the user
NAME	User's name
OWNER	Identity of the owner of this profile
DFLTGRP	User's default group
AUTHORITY	User's authority in the default group
PASSWORD	User's password information (one-way encrypted)
REVOKE	Date on which RACF prevents the user from accessing the system
RESUME	Date on which RACF lets the user regain access to the system
WHEN	Days of the week and hours of the day in which the user is allowed into the system
SECLEVEL	Security level of the user (used for mandatory access policy)
SECLABEL	Default security label associated with the user (used with for mandatory security policy)
SPECIAL	Gives the user the systemwide SPECIAL attribute
AUDITOR	Gives the user the systemwide AUDITOR attribute
OPERATIONS	Gives the user the systemwide OPERATIONS attribute
CERTNAME	Names of the profiles containing this user's certificates
CERTLABL	The labels for the certificate associated with this user
CERTPUBKY	The encoded public key of this user
CERTSJDN	User's distinguished name

in each RACF user profile is organized in two blocks. The first is called the base segment, present in all such profiles, and contains the key security definitions for the user such as its identity, its credential (e.g., a password), as well as the level of the RACF authority assigned to the user in his or her default group. Table 2.1 illustrates the base segment in the RACF user profile.

The second class of RACF user-profile information is optional and consists of a set of segments, each containing fields that define various attributes that can be associated with the user. These attributes have mostly evolved with the need of other subsystem components to maintain their own attributes about the user. This feature has allowed RACF to evolve over the years and adapt to the security requirements of newly developed subsystems and applications. Table 2.2 shows the segment of a user profile intended for use by the IBM's Customer Information Control System (CICS) terminal operators.

TABLE 2.2 Elements of the RACF CICS segment in a user profile.

Attribute	Description
OPCLASS	Classes assigned to this operator to which basic mapping support (BMS) messages are to be routed
OPIDENT	Identification of the operator for use by BMS
OPPRTY	Priority of the operator
TIMEOUT	Time that the operator is allowed to be idle before being signed off
XRFSOFF	Indicates whether the operator is to be signed off by CICS when XRF takeover occurs

A new user profile is defined by using the ADDUSER command. Thereafter attributes are added, removed, or updated using the ALTUSER command.

Network Identity

The advent of distributed computing has led to the emergence of the network-identity concept. The idea is simple but has far-reaching implications. An identity is authenticated to a network of computing nodes rather than to a single hosting system. Once an identity is established in this fashion, it navigates through the participating network nodes requesting services and accessing resources without having to explicitly engage in further identity establishment. The scope of an identity is no longer confined to a single system; instead, it is bounded by the network in which it is defined. To achieve this extended scope, identity services have evolved into network components.

The extent of the network in which an identity is defined generally remains limited to a single enterprise. Advances in network identity, however, have led to the ability of establishing cross-enterprise network identities. In some cases, this has resulted in tightly coupled interenterprise links (such as with cross-domain Kerberos implementations), while in other cases, interdomain identities are established via loosely connected enterprises (such as with cross-certification provided by public key infrastructures). We discuss these topics in further detail later in this chapter. The characterizing factor of network identity remains its confinement in scope regardless of the number of participating domains. Figure 2.2 represents a high-level view of a network identity. In A the identity is confined to a single domain, while in B an identity is used throughout two domains.

Federated Identity

Foundations of Federated Identity

The term *federation* has been used in the literature with varying semantics. Indeed, it conveys a generic sense of flexibility and perhaps speaks of the activities of a loosely coupled set of cooperating entities. In the Internet domain name services (DNS), for instance, the federation reflects the delegation of authorities among a hierarchical tree of name servers. The effect of such delegation is the decentralization of name-to-address resolution, the core function of DNS. In the electronic business, a federation can represent a relationship between two or more organizations where each has its own computing infrastructure. The federation manifests itself at the identity level by the mechanisms used to allow one participant organization to directly provide services to entities registered at another organization member of the

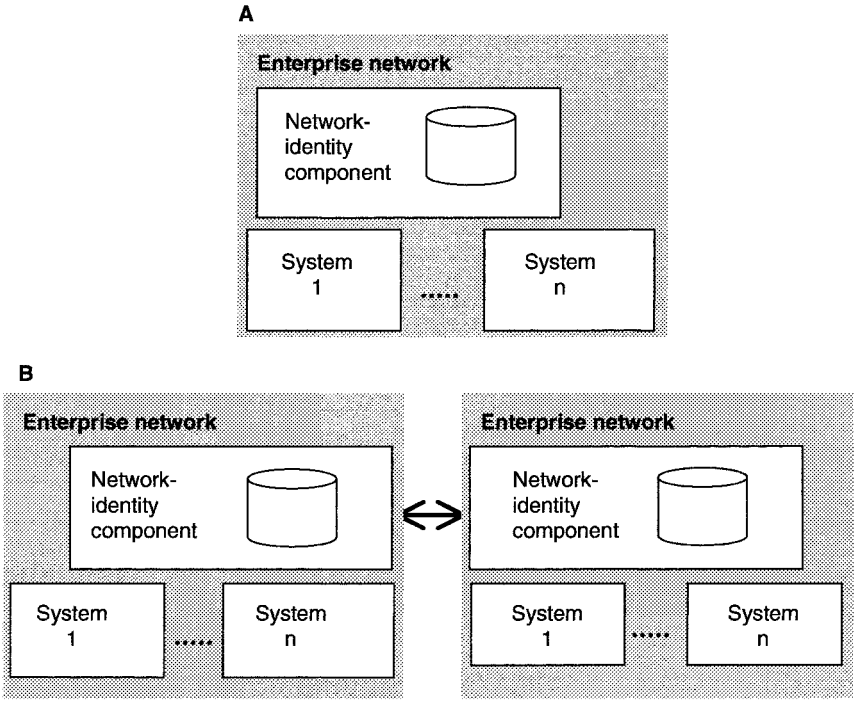

FIGURE 2.2 A High level view of network identity

underlying federation. The result is that each of the participants in a federation will have achieved an extension of the space of identities operating under its premises without having to manage the entirety of this space of identities.

Cross-organizational trust is the foundation of federated identity. The federation is accomplished by the means through which an organization is capable of acquiring the necessary information about a foreign entity that wishes to access one of its services. Furthermore, identity information about a foreign user is acquired from the home organization in a secure and trusted fashion. This process is achieved with full transparency to the users and applications crossing organizational boundaries. The end user remains unaware of such cross-domain activities taking place.

An end user does not need to register with foreign organizations, nor does he or she need to directly engage in an authentication process with an entity other than the home organization. Under the covers of the federation, attributes of an entity that is established at its home organization are communicated to foreign organizations. User attributes exchanged over a federation may ultimately be required to adhere to a common representation syntax and

semantics. This requirement represents one of the major hurdles addressed in forming federations. Furthermore, the lack of a universal set of attributes that can be associated with an identity and be consistently interpreted by every organization is a hindrance to accomplishing federated identity.

While a user profile registered to his or her home organization may be contain all attributes necessary to request services from that organization, other attributes may be missing from that profile when services are requested from another organization. One approach that can be used to address this problem is to confine the definition of various profile attributes to third-party organizations that are the source of those attributes. For instance, the definition for credit information can be the responsibility of banking and financial institutions, while the definition for attributes that are universally common to every entity (such as identification name, address, and contact information) can be agreed on by a much wider forum that is open to participation from every organization. The model adopted here is to leave data definitions to the concerned organizations only. The use of XML as a means of defining such data elements can ease interoperability and lead to a speedy acceptance of those definitions across organizations.

The security mechanisms by which trust can be established and maintained across organizations are at the core of an identity federation. Although these mechanisms may differ in the way in which trust is computed and verified, standard mechanisms implemented at the higher level are key to joining various trust models under a unified federated scheme. In that respect, the advent XML-based component technologies such as the security-assertion markup language (SAML) is expected to raise identity federation to an unprecedented level.

A pure identity federation allows an entity to be profiled and registered only once, generally at its home organization. The scope of that identity, however, ends up spanning multiple domains participating in the federation. A generalized and a more practical federation approach allows a user to register at multiple organizations, yet accomplishes a single logical view of all such registrations if so desired by that user, the owner of the identity. Such is the case with the XNS infrastructure that we discuss below. With XNS identity, *federation* is defined as the distributed resolution of names and IDs across a decentralized network of identity servers and clients. The novel concept of addressable identities in XNS forms the foundation on which federation is based. Identity cross-referencing and linking in XNS enables users to participate in a logical federated web that is defined and controlled at the identity level. Synchronization of attributes in this federation is transparent and automatic. Control in XNS federations is brought to the level of an entity rather than the traditional confinement of such controls to participating organizations.

Figure 2.3 illustrates the high-level concept of identity federation. The different shapes representing organizations illustrate the fact that each participant organization manages its own model of identity that may or may not be

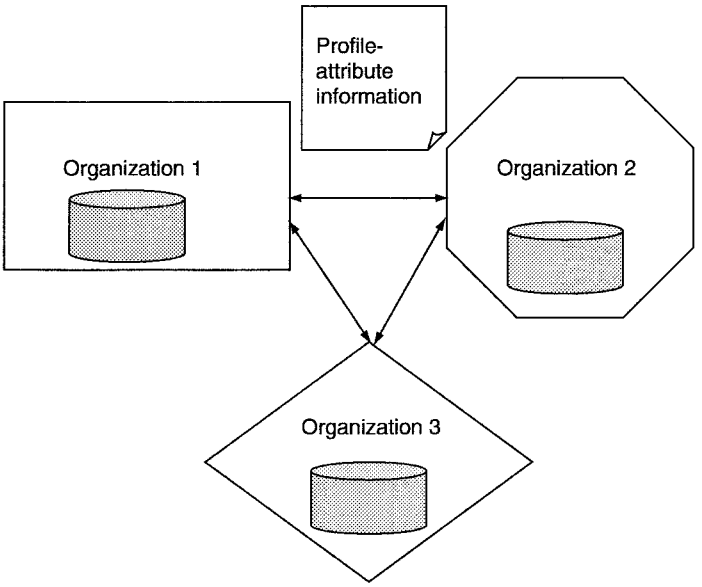

FIGURE 2.3 A high-level illustration of the concept of federated identity

the same model used in other members of the federation. The links between each two organizations represent an established trust that is securely verifiable.

Federation Topologies

Federated identity can be accomplished through various ways. Recall that the characterizing aspect of a federation is the fact that end entities undergo a single registration process. In the event that such registration is performed more than once (i.e., at different participating organizations), complete redundancy of profile attributes for the underlying entity should be avoided. Otherwise, the semantics of the federation become questionable. The differences among various federation topologies can be related to many factors. Most important is the way trust among the federation members is established and the model used to store, maintain, and manage profile attributes. One other differentiating factor is the level of scalability that the topology affords. After all, there is an implicit thinking that any federated identity scheme automatically implies the requirement for a reasonable level of scale.

In the following, we discuss a few possible federation topologies that we categorize based on the method of by which entity profiles are registered and managed. In all these cases, the concept of the home organization of an entity is maintained.

Local Profiling

In this scheme, each end entity is registered within the identity infrastructure of its home organization. Profile attributes of an entity are fully maintained and managed by the local organization. Attributes can expand and contract based on the privileges, roles, and entitlements of the end entity. All other member organizations are unaware of such registration except for when a service request crosses organization boundaries, at which time the underlying identity attributes are exchanged underneath the trust relationship defined by the federation. As we already have mentioned, this model becomes better suited for implementation when data elements for profile attributes are well defined and understood by the member organizations. Parties that are most concerned with the underlying attributes are the best candidates for defining standard attributes.

Distributed Profiling

In this topology, an end entity begins with a registration within its home organization. As the need arises, the entity may further expand and hence acquires new profiles at other member organizations. One reason for having additional registrations is the need for new attributes that are specific to a particular organization. In a sense, the definitions for an entity's profile become distributed across multiple organizations. As a consequence, definitions for the same profile attributes may be duplicated, and thus attribute synchronization may become an issue. This scheme offers the advantage of flexibility and somewhat leads to separation of concerns when it comes to managing user attributes among organizations.

Profiling by a Third Party

In this scheme, a designated third party within the established federation is tasked with brokering the management of end entity profiles. Member organizations are thus entirely alleviated from this task. The third party may distinguish among profile information that is common to all or to a subset of the member organizations as well as those that are pertinent to specific ones. This scheme offers the advantage of having to manage trust establishment with the third party only. Attribute synchronization problem will be limited to the confines of the single third party where specific organizational information may be duplicated for two or more target organizations. One disadvantage can be the issue of scalability as more and more member organizations may contend over the single third party for the retrieval and update of profile information. The replication of the third party may be needed to relieve such a problem. When that happens, the replicas are required to be kept synchronized.

Global Web Identity

The need for a global identity seems to be driven in large part by the emergence and the viability of the World Wide Web as a computing platform. A *Web identity* is one that is uniquely known throughout the Internet Web. Like an Internet resource that is identifiable via its universal resource identifier (URI) [BERN98], a Web identity exists in the global context of the Internet. Every Web identity stands alone to represent the entity that owns it in the same way a Web URI represents the physical resource behind it. Unlike Internet identifiable resources that represent objects that remain locally managed by an enterprise's computing domain, Web identity information is capable of being uniquely resolved to one entity and being recognized and used locally as well as by other Web nodes.

Identity Mapping and Synchronization

The ushering of the Web computing era is increasingly accepted due in large part to the fact that it builds on existing computing infrastructures. The advent of global Web identity mechanisms should not represent an exception. It needs to exploit the identity-management services that have been in deployment and existed for so long. These services are generally based either on local or network identity registries. For that to happen, a unified Web identity requires a mapping to various identity registries in which it exists. The single Web identity would allow navigating the myriad of Web services that ultimately may be deployed over the World Wide Web in a seamless fashion and a great deal of transparency to end users. A number of identity-management technologies that provide this seamless navigation experience exist today. Among them are metadirectories and affiliate networks.

MetaDirectories

The metadirectory approach bridges disparate domains by exposing the user's identity to a higher level while retaining its relationship to various participating enterprise networks in which the identity is known. The relationships of the global identity to the corresponding enterprise-level identities are formed by the links binding metadirectory information to the directories of the participating organizations. Common user attributes are maintained by the metadirectory. Updating these attributes is centrally done, and synchronization is performed automatically. For example, a large organization that maintains information about its users in multiple directories (each is perhaps being used by a different application) can join them via a single metadirectory, thus enabling seamless sharing and maintenance of identity information. Figure 2.4 represents the operation of joining multiple directories using a single metadirectory. The metadirectory on the left joins multiple directories of the same

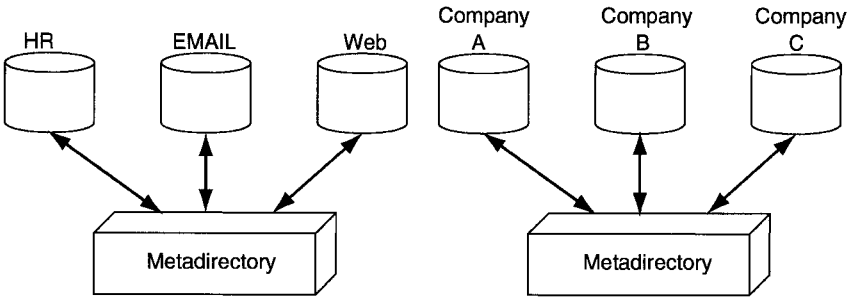

FIGURE 2.4 Joining multiple directories via a metadirectory

organization, while the one on the right joins multiple directories across different organizations.

The key drawback of this approach is that it cannot scale to the extent to which it can accommodate a potentially large number of worldwide identity domains. Figure 2.5 illustrates the concept of identity mapping from global to local using the metadirectory approach.

Affiliate Networks (Virtual Directories)

Affiliate networks, also called *virtual directories*, participate in a tightly coupled structure by directly mapping an identity defined in one directory onto

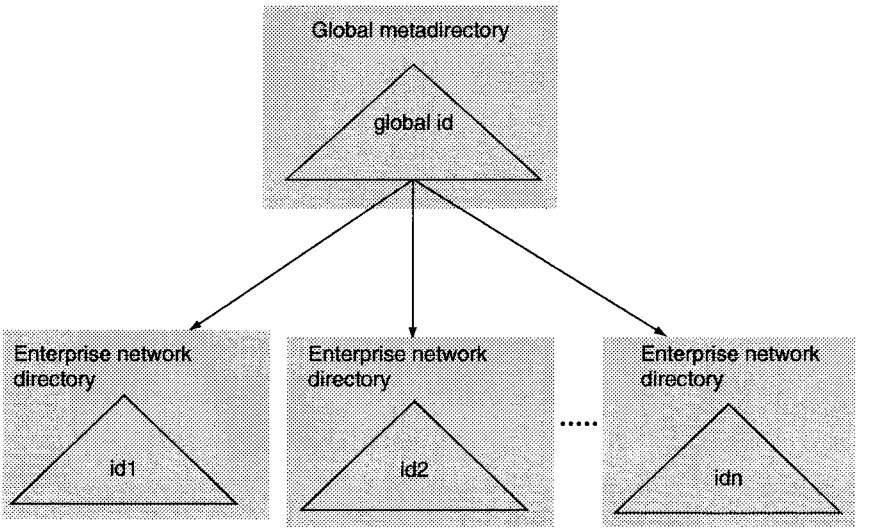

FIGURE 2.5 Mapping entities via a metadirectory

a corresponding identity in another enterprise directory. The main difference between this mapping approach and that enabled by metadirectories is that here the mapping is achieved without actually having to create an additional "join" in directory. This approach has a better scalability property over metadirectories in that the mappings are discretely distributed over the participating directories. Mapping users across all directories, however, creates management complexities associated with the *n*-wise mapping problem. Updating user-identity information requires updating *n* directories. Figure 2.6 depicts the three-way identity-mapping problem presented by the affiliate networks architecture.

Mapping an identity is not simply about associating names from one name space to another. Most important, the mapping applies to the attributes associated with an identity. Updates to such attributes in one directory may require synchronization across multiple directories. Synchronization, if not completely automated, increases administrative complexity, requires establishing cryptographically secure channels, and can be prone to errors. Directory-attribute synchronization is supported through extensions to the lightweight directory-access protocol (LDAP) [HOWE03] as well as the

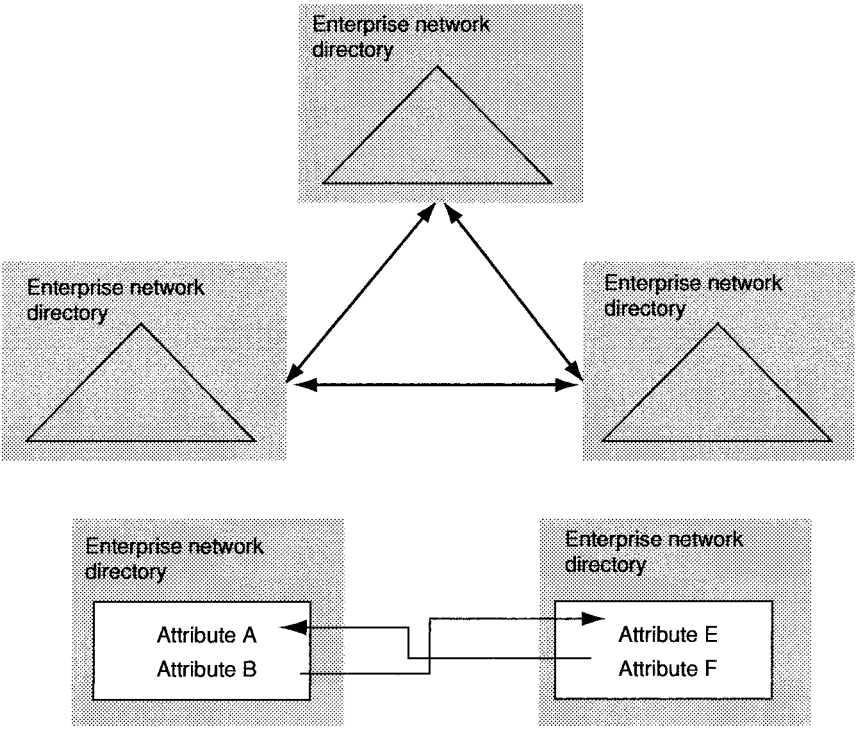

FIGURE 2.6 Joining multiple directories via affiliate networks

XML-based directory-interchange standards, such as the directory-services markup language (DSML).

Dynamic Scoping of a Security Context

A global identity that is navigating the Web should be encapsulated behind a security context that is reliably established and verified and cannot be forged. The security context carries with it attributes of the identity it represents, generally containing a subset of the user's profile. Exposing a user's attributes over the Web requires stringent security measures. A host of issues are relevant here, at the top of which are privacy concerns such as the Web transactional pattern or the medical attributes of an individual, identity impersonation, and theft of sensitive attributes such as a credit-card number or bank accounts. The user should be provided with the power to disseminate his or her digital profile information on a discretionary basis. This allows the user to maintain control over the propagation of his or her attributes to visited Web services. The Web security context, therefore, should allow for dynamic changes under the controls of the user and should be capable of expanding and contracting. Confinement or simply preventing information leakage of the user's attributes at the serving Web sites remains a major security concern that is compounded by the nature of the Web and the unlimited number of services that can all seamlessly cooperate in delivering a single end-user service request. The paths involved in such a request can be unbounded.

The XNS Approach to the Global Web Identity

Current technologies used to solve the issues surrounding Web identity as we noted are not addressing the problem from the basic infrastructure perspective. They are, instead, component solutions that do not form an integrated infrastructure. Existing identity-management components in many ways are being retrofitted to solve a new problem—that of the global Web identity. Development in Web-identity infrastructure is considered yet at its infancy. A promising novel approach is one being undertaken by the XNS Public Trust Organization (XNSORG), which is developing an infrastructure specification referred to as the extensible name service (XNS) protocol for a Web identity [XNSO02].

XNS is an XML-based protocol for identifying and linking together identities that participate in a Web transaction. It is intended by its designers to provide a flexible and interoperable method for establishing and maintaining persistent digital identities and the relationships between them. The protocol provides services for registering and resolving identities in a way similar to resolving addresses. It defines the elements of managing identity documents, conducting and protecting identity transactions, and linking and synchronizing identity attributes. XNS adopts XML-based technologies such as the

XML schema [W3CO01a, W3CO01b, W3CO99] and the Web services [W3CO02a] in defining its constructs and services. As such, it is designed to be platform-independent and extensible. XNS also builds on emerging XML security standards such as XML signatures [W3CO02b], XML encryption [W3CO02c] and the security assertion markup language (SAML) to protect identity documents and assert credentials and entitlements exchanged during Web transactions [OASI02].

The approach followed in the architecture of XNS is based on abstracting the user identity to a new logical level, that of the Web identity with a global scope. The architecture of XNS is inspired to a great extent from the Web architecture itself and in particular the design of the Internet domain-name service (DNS). The novel aspect of the World Wide Web as we know it is its elevation of enterprise data to a logical representation layer that can be accessed via a universal client tool (the Browser), using a ubiquitous protocol (HTTP), and formatted in a standard markup language (HTML). Most important, this logical layer forms a global Web that links related content with an unprecedented level of location transparency, ease of use, and seamless navigation experience. The designers of XNS have developed a parallel to that with respect to identity. Figure 2.7 illustrates the analogy between the Web architecture and the approach undertaken by XNS.

Two elements are key contributors to the level reached by the Internet Web today:

- The domain name service (DNS) that weaves interconnected systems together and enables the seamless navigation of Internet hosts and computing devices, and
- The mechanisms by which documents are linked through references to a universal addressing scheme.

Indeed, the XNS design appears to be entirely inspired from these two aspects of the Internet. We begin by first taking a quick tour of DNS which in itself provides an unprecedented global naming scheme that is hierarchical in structure.

Elements of DNS

DNS, defined in RFC 1034 [MOCK87a] and RFC 1035 [MOCK87b], has grown to become one of the most successful distributed systems for naming Internet hosts and resources and performing name resolution to corresponding Internet protocol (IP) addresses. DNS components define a hierarchy of services structured in an inverted tree. Each node in the tree is concerned with a particular naming subspace also referred to as a domain name. The latter consists of an ordered set of labels (symbolic names); each is associated with a subordinate node. This ordered set begins at a leaf node and follows up through a path leading to the root node (one with a null label). Labels are delimited using the dot character (.). By convention, the labels that compose

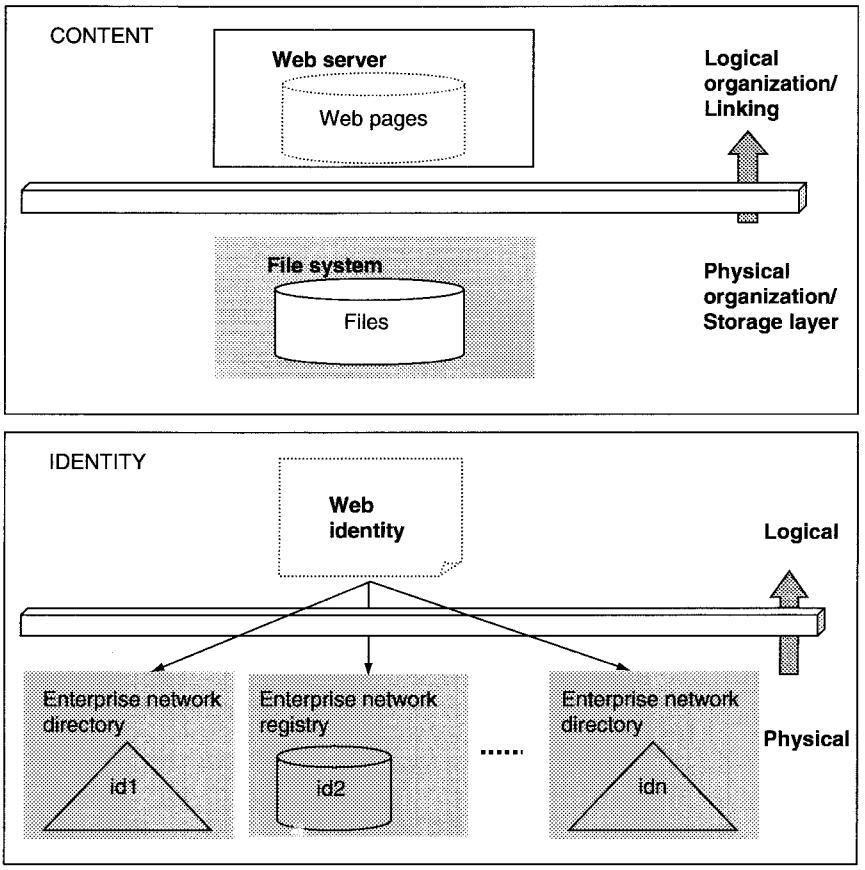

FIGURE 2.7 The XNS approach to Web identity (analogy to the World Wide Web infrastructure)

a *domain name* are printed or read left to right, from the most specific (farthest from the root) to the least specific (closest to the root). In the example shown in Figure 2.8, the root domain has three subdomains—EDU, MIL, and ORG. The RPI.EDU domain has one immediate subdomain called CS.RPI.EDU.

DNS makes use of two key components:

- *Name servers* Maintain the mapping information about an entire domain tree or a particular subtree representing a subset of a domain naming space. In the latter case, a name server also maintains pointers to other name servers that can lead to resolving domain mapping information from any part of the domain tree. A name server is said to be the authority over the subspace it maintains. Authoritative information

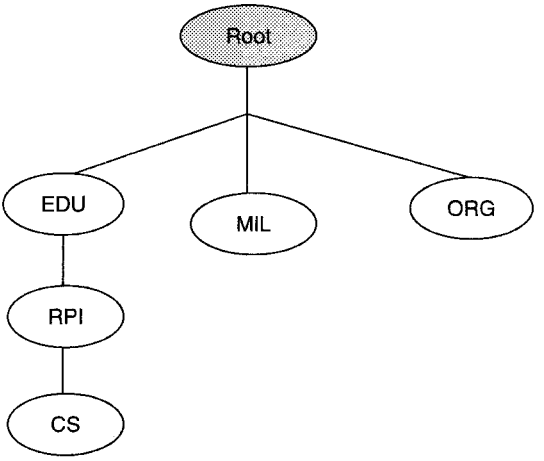

FIGURE 2.8 An instance of the DNS naming space

is organized into units called *zones*.

- *Resolvers* These are agents residing at the edges of the network and are directly invoked by application programs. They represent the client side of DNS. The purpose is to initiate the process of resolving a symbolic domain name into its IP address. Resolvers are configured to access at least one name server and use that name server's information to answer a mapping query directly or further pursue the query using referrals to other federated name servers authoritative over the entire name or a portion of it until the name is finally resolved.

Figure 2.9 depicts the layered structure represented by DNS. For an end user, a name resolution consists of an interaction with the local resolver, while to a resolver the interaction may lead to one or more remote name servers. Each name server is an authority over its own particular zone. The database of names operated by each server is basically a flat-file data store in which the primary key is the domain name and the main values maintained are the IP addresses forming the mapping from Internet domain and host names to corresponding IP addresses. The power of DNS stems from the federation formed by the participating name servers worldwide, each operating on its own local data store. As we know, the sum of these basic elements gave rise to one of the most reliable computing infrastructures known to date. We take it for granted every time we navigate the Internet, send an email, or browse the Web.

Three concepts are worth pointing out at this juncture: First, the uniqueness of an absolute IP address in representing a physical host or a network device at some location; second, the presence of a hosted resource, such as a

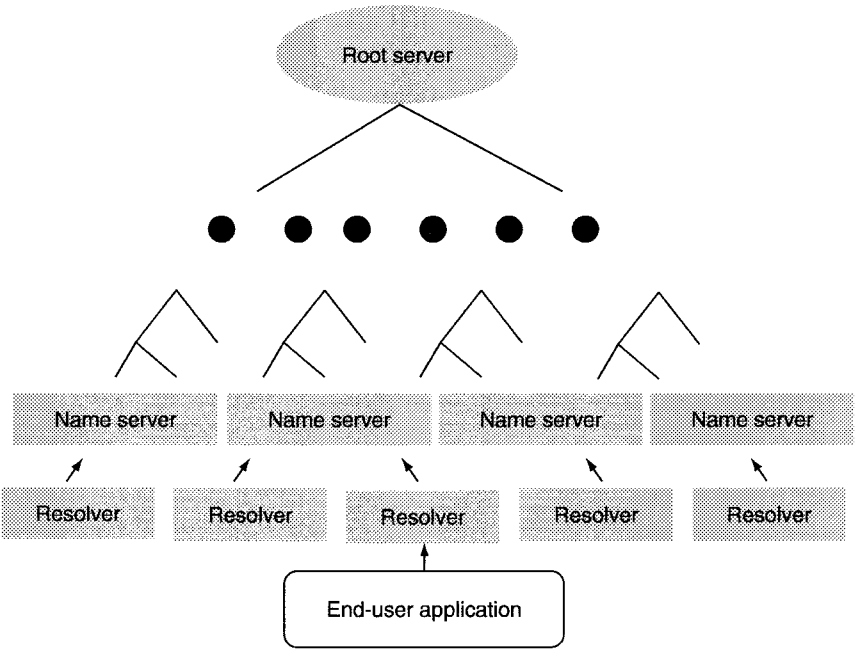

FIGURE 2.9 A view of the DNS federation

file or a service that can be reference relative to its globally addressable hosting system; and third, the user level addressing of hosts and network resources with semantic names in the form of domain and host names. XNS draws from these elements of DNS and the globally addressable Web resources (URIs) to bring identities to an unprecedented level of globally addressable entities.

The invention of the TCP/IP protocol suite as we know it led to the abstraction of disparate networks into a logically single global network, the Internet. DNS, although not an absolute necessity for the Internet to function, presents an immense value to the Internet-based protocols such as Telnet, SMTP, and HTTP. It enabled programmers to use human-friendly names to identify Internet endpoints, rather than the physical addresses as represented by the IP numbers. Figure 2.10 shows a higher level of abstracting IP addresses when DNS is present between the TCP/IP layer and applications. DNS provides the following benefits:

- Network endpoints are abstracted into location-transparent names. Addressing network entities in distributed applications therefore remains unaffected by changes in the physical address of an endpoint.
- Multiple names can be used to identify the same network endpoint if

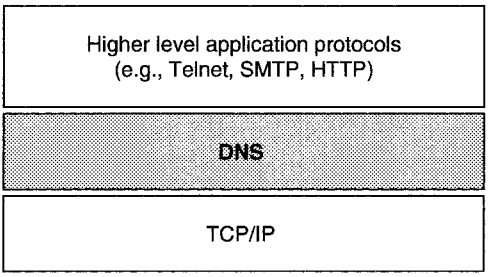

FIGURE 2.10 DNS brokering of network endpoints addresses

so desired. Such names ultimately will all resolve to the same target endpoint without ambiguity.

- Semantic names represented as domain or host names appear to be locally scoped, yet become global when translated through DNS.

Elements of XNS

From a higher-level perspective, the network architecture of XNS appears to be similar to DNS. Like DNS, XNS data-store is distributed across globally federated identity servers. Unlike DNS, however, the paradigm of interactions among XNS entities is peer-to-peer. In DNS the flow of execution is unidirectional in that at the lowest level an application invokes a resolver, which invokes its authoritative name server. The name servers are, in turn, federated in a way that requests are initiated by lower authoritative servers to higher ones. The separation between clients and servers is clearly defined in DNS. The peer-to-peer nature of XNS draws no distinction in the interaction between identity clients referred to as identity agents and identity servers. In XNS all requests are answered by identity agents that run on either a client or a server machine. The peer-to-peer aspect of XNS is a key defining characteristic of its Web identity architecture. Figure 2.11 illustrates the peer-to-peer relationships among XNS entities. The architecture of XNS is characterized by the following elements:

- Identity is the addressable unit or resource. This may be considered the key contribution from the XNS designers. The innovative aspect of XNS evolves around the view of an identity as an addressable entity like any other network resource. Identities are profiled and represented by *identity documents*, which are XML documents containing instances of XNS defined data types describing attributes associated with an identity.
- Peer-to-peer relationships exist across identity agents and servers. The liberating nature of peer-to-peer computing is brought to the Web identity, thereby increasing the level of flexibility, independence and reliability. Identity agents are the entities operating on identity docu-

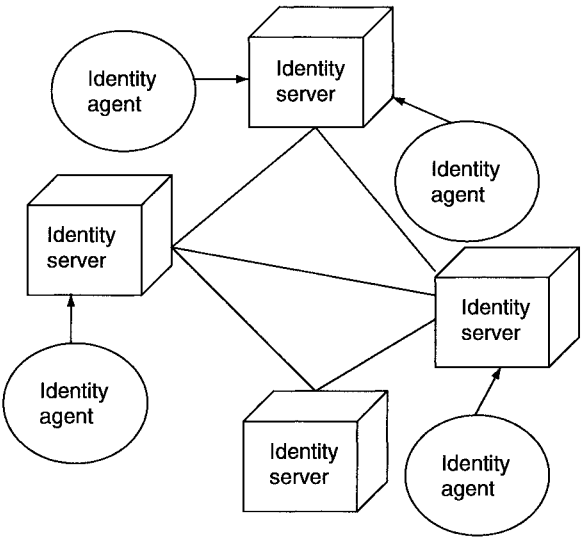

FIGURE 2.11 The peer-to-peer aspect of XNS

ments via a set of Web services defined by the XNS infrastructure. Each identity document is associated with an identity agent responsible for it.

- The presence of a discovery service allows agents to dynamically discover and invoke the services available from each other. By way of adopting XML as a mechanism for describing its constructs, XNS is self-defining. XNS service specifications are published as XNS identity documents capable of being discovered, versioned, published, subscribed to, and linked in the same way identity documents are.

The advent of TCP/IP followed by the high-level common application protocols such as SMTP and FTP is analogous to the newly emerging layer of the Internet infrastructure as represented by the exchange of XML-structured data objects via the simple-object access protocol (SOAP) [W3CO00]. This new abstraction layer promises to bring the composition of service elements to the same level reached by composing functional elements as we came to be familiar with in modern programming languages. The depth of such compositions can be unbounded and involving a large number of logical endpoints referred to as *actors*. Concern over the security of seamless combinations and compositions of actors involved in SOAP interactions has resulted in an ever greater need for the secure attachments of identities to various service elements. We refer to this as the Web global security context. With a striking resemblance to DNS, XNS is presented by its designers as the global identity layer of the

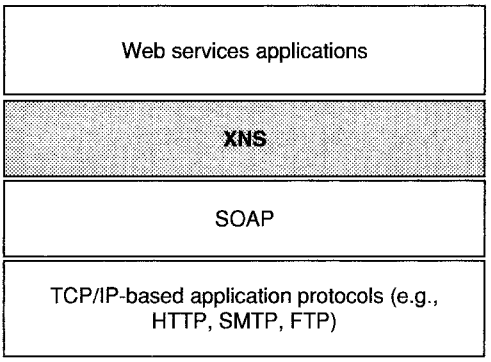

FIGURE 2.12 The XNS layer for the enablement of the XML/SOAP computing model

emerging Web services computing model. Figure 2.12 illustrates the positioning of the XNS layer with respect to Web services.

XNS Identity Types

XNS recognizes three types of physical entities that can be associated with identities. These entities are referred to as *identity controllers* in XNS; sometimes they are also called *identity owners.*

- *Persons* Identities assigned to individuals (*personal identity*).
- *Organizations* Also called *business identities.*
- *The general public* This extends the space of entities in XNS beyond just persons and organizations. Objects such as planets and various Web resources can also be assigned XNS identities. *General identities* are controlled not by persons or organizations, but rather by linguistic, cultural, or scientific conventions and remain under the auspices of XNSORG. This is somewhat a departure from the traditional meaning of an identity in computing. XNS identities extend beyond the realm of active entities such as end users and programmable agents.

Each of these three entities can be represented by one or more XNS identities. An XNS identity is not one-to-one with its controller or in general terms the entity with which it is associated. Nevertheless, XNS is capable of maintaining the relationships across multiple identities of the same principal in a way that results in a single logical identity. We discuss this in further detail below.

The XNS Identity Document

Identity information traditionally referred to as a *user account* is encapsulated by an *identity document* that maintains various elements profiling an entity including a set of associated attributes. These attributes or, in generic

TABLE 2.3 Abstraction of an identity in XNS.

Data Element	Description
IdentityType	Determines the classification of an identity which can be a person, organization, or general.
Memberships	A list of XNS groups to which this identity belongs.
PublicKey	The certified public key bound to this identity.
Types	A list of various XNS-typed objects containing attributes associated with this identity, links to other identities, contracts and so forth.

terms, XNS objects are expressed using data types that are defined in XML. XNS as such operates on a distributed database of identity documents. Each document is a highly structured object that contains the abstracted XNS data types described in Table 2.3, a generic instance of which is illustrated in Figure 2.13.

IDs and Names in XNS

An *XNS identity* (ID) is a logical abstraction of a semantic identity referred to as an XNS name and also called a *named URI*. An ID is invariant as opposed to the attributes of XNS names with which it may be associated. Once an ID is generated, it remains unchanged, persists, and is globally unique, while a name generally has a fixed lifetime, a fixed scope, and

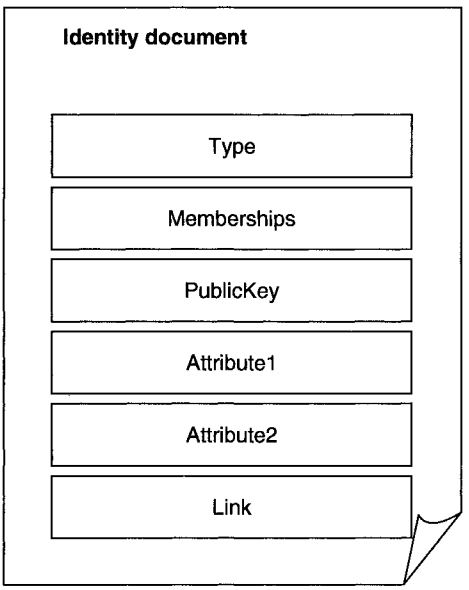

FIGURE 2.13 Abstract view of an identity document

context. XNS names are mutable semantic identifiers that are unique only within a particular name space. IDs are assigned once and never reissued. When a named entity is terminated, its ID is retired. The requirement for an XNS ID to persist is satisfied by the ID service generating and handling a globally addressable construct in the form of a uniform resource name (URN) [MOAT97].

Links across identities are based on XNS IDs and not named URIs. XNS supports moving identities to new hosting environments without breaking the links. Figure 2.14 illustrates the concept of identity abstraction in XNS. Names are handled by the name server, while IDs are handled by the ID server of XNS. A name is linked to an existing ID when it is first registered with the name server. Releasing a name results in removing the link to the corresponding ID.

XNS Resolvers

As we have noted the association between XNS IDs and names is one-to-many. The ID service and the name service of XNS are capable of resolving an ID to a name and a name to its identity address, respectively. An ID is resolved to the named URIs with which it is associated. These URIs are used to channel communication to the identity hosted at a network endpoint. It is worth noting that the addressability of an XNS identity is what brings identity management in XNS to a logical layer analogous to content in the World Wide Web. A hosting endpoint provides XNS hosting service to other identities defined at the same network endpoint. A hosting endpoint is associated with a host identity document that specifies among other things a list of transport protocols over which the host accepts XNS communications. The host forms the backbone of the community that it serves. Identity URIs are scoped by the identity of the system in which they are hosted. Figure 2.15 depicts the layering of XNS components involved in resolving identities.

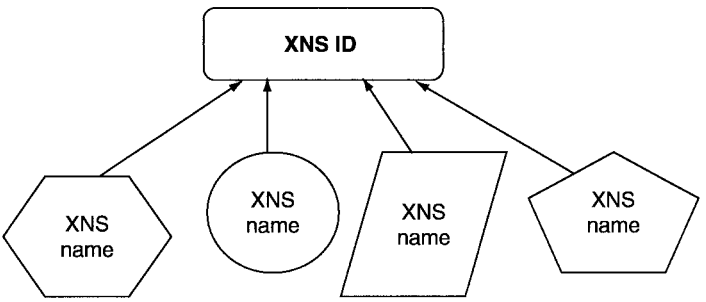

FIGURE 2.14 Abstracting semantic identities in XNS

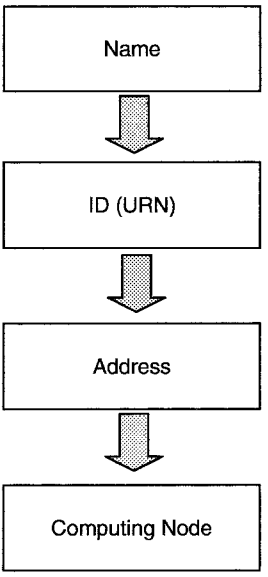

FIGURE 2.15 Resolving identities in XNS

Cross-Referencing XNS Identities

An entity such as a person may be associated with multiple XNS identities; each identifies the person to a particular domain of operations such as an organization, a community or a particular business. The proliferation of multiple identities per physical entity such as an individual person, although comes with all the complexities of identity management, it has become a common practice in computing. XNS builds on such existing identity paradigms and practices only to further enhance them. Multiple-identity documents owned by the same entity logically represent a single entity and thus generally contain common profiling information such as a person's name, home address, telephone number, and physical attributes. XNS allows identity documents controlled by an individual entity to be cross-referenced so that a logical equivalence is established across such documents. Any XNS object in an identity document can be cross-referenced with another XNS object in a different identity document anywhere in the XNS network, including an entire identity document. Shared attributes can thus be recognized across multiple hosting communities and can be seamlessly synchronized. This behavior is provided subject to the discretion of the identity controller. A person, for instance, may prefer to maintain separation across multiple profiles he or she owns, thereby remaining anonymous or pseudonymous. XNS cross-referencing is expected to dramatically simplify user profile management, and authentication and leads to a reliable capability of SSO in particular. Figure 2.16 illustrates identity cross-referencing in XNS.

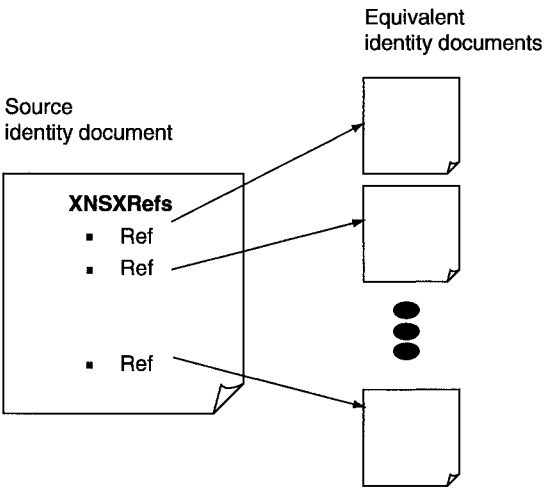

FIGURE 2.16 Cross-referencing XNS identities

Forming Trust Relationships in XNS

Access to identity attributes can be exposed to the public in general or can be constrained based on a policy adopted by the holder of the identity. The flow of identity attributes is a key enabling aspect of electronic commerce and transactions over the Web. Concern over privacy is a major issue that arises with the dissemination of user profile information. XNS takes a novel step in exposing identities over the Web. Support for privacy and protection of identity attributes transacted over the Web is fundamental to XNS. Transacting over such attributes is performed under the mutual consent and agreement of the parties involved using a *negotiation* service that is currently being specified in the XNS protocol.

XNS defines the trust relationships among its managed identities via contract links that can be embedded within identity documents. A contract is a uniquely identified construct that governs the exchange of attributes with some other addressable identity on the XNS Web. It specifies what data is to be exchanged, the protection mechanisms to be used for the exchange, and any policies that govern the automatic propagation of those attributes for synchronization purposes.

Although confinement of data to the trusted entities remains an issue that in the end simply falls in real-world trust among entities. Trust relationships that can be defined in an XNS Web are unbounded. The ability for expanding such relationships and their peer-to-peer aspect is a powerful concept underlying XNS. Figure 2.17 is a representation of the discrete dissemination of identity attributes in XNS.

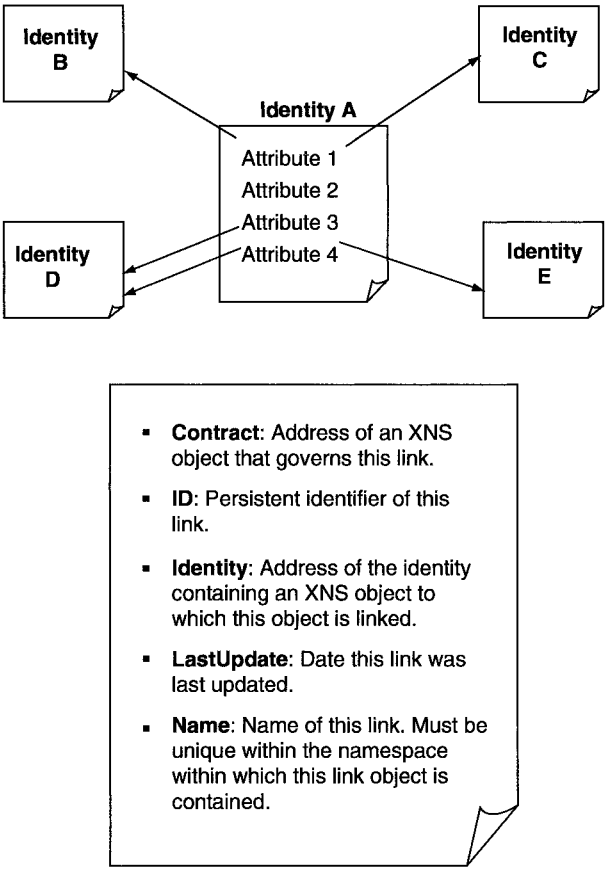

FIGURE 2.17 Identity linking and attribute dissemination in XNS networks

XNS Services

The XNS 1.0 infrastructure specifies a set of component services designed according to the paradigm of self-describing Web services. These services are organized along four major functions:

- *URN services* The URN services are at the core of XNS. They represent the novel concept of addressable identities and weave the identity web comprised of network actors in the same way DNS weaves network endpoints together. The major aspect here is the separation of semantic identifiers, (names) from persistent abstract constructs (IDs) [MOAT97].
- *Attribute-management services* This service manages entity profiles as represented by collections of attributes expressed in terms of various XNS basic data types.

- *Exchange and linking services* These services allow the secure dissemination of attributes across identity controllers. Currently a negotiation service is specified for XNS entities to establish identity transaction contracts. An *introduction* service is expected to be developed also. This service permits an identity linked to two other identities to introduce those two entities to one another and thus result in a new direct linking relationship.
- *Credential management services* These services allow identity establishment, secure communication of credentials, and the management of secure associations (sessions).

Centralized Enterprise-Level Identity Management

Administration of identity-management processes is an important factor in controlling the cost of computing in large enterprises. Typically, the computing infrastructure of such an enterprise is composed of various resources distributed over a local or a wide-area network. These resources may include nodes of different operating-system platforms, a large number of application subsystems such as data-base systems and human resources repositories, Web application servers, directories, and possibly business applications that require managing user subscriptions. Such might be the case in a utility computing infrastructure providing services on demand. Each of these subsystems typically has its own identity registry. Managing each such registry separately inhibits scalability as it can easily introduce errors, and inconsistencies and may become very costly. Over a period of time, the growth of the computing resources will undoubtedly increase the complexity of managing the enterprise identity systems and may lead to loss of control when a large user population and a myriad of systems are in use.

Centralized identity management is an appealing solution to large enterprises. It is likely to reduce management costs and most important will enforce an element of control within the enterprise. It enables a single view of the multitude of systems in the enterprise, provides a consistent interface to all these systems, and unifies identity-management processes. The emerging model of centralized identity management defines a centralized layer that sits on top of existing systems, thereby enabling a common perspective to all managed systems. Figure 2.18 shows a high-level illustration of the centralized identity-management model. The different shapes showing managed systems represent the heterogeneity of systems that can be managed. The organizational structure of an enterprise is defined in the identity manager and managed objects such as suborganizations and end users are all defined at this layer. User access to a target managed service is represented by an account for that service. An end entity such as a person is in a one-to-many relationship with the set of available accounts. Attribute synchronization across various accounts of a single entity may be performed automatically if so desired.

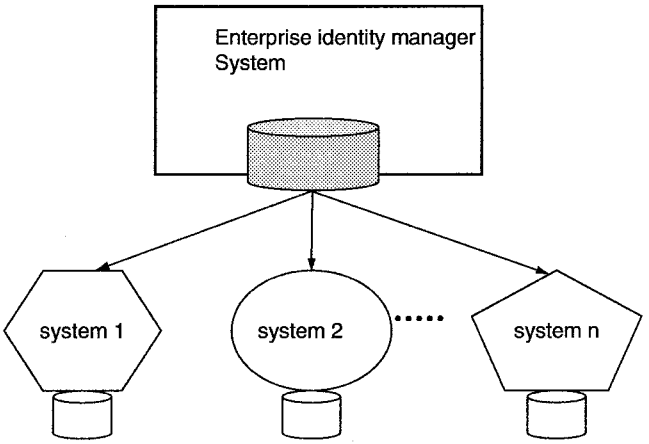

FIGURE 2.18 A high-level view of centralized identity management in the enterprise

Based on the data model adopted, we distinguish among two schemes of centralized identity management systems. We refer to the first one as the unified representation of identity and the second one as the decoupled identity representation scheme. But first we elaborate on two major benefits of a centralized identity-management system.

Synchronizing Identity Attributes

The side effect from updating attributes of a given entity at the level of the central identity manager may result in propagating the change to all or a subset of accounts associated with that identity on the managed systems. An example would be the synchronization of a security credential such as a password or a public key certificate on all systems and services in which the entity possesses an account. Attribute synchronization can be subject to various policies that may govern the underlying attribute. A password policy, for instance, may have different variations on each of the managed systems. In the event an attribute obeys different rules, it is treated differently on each of the managed systems, even when semantically it represents the same construct.

While a centralized identity manager may be mostly concerned with attributes being updated centrally and then pushed down to the managed services, updates that are initiated at the target services need to be accommodated as well. For instance, an individual that performs a password change while directly interacting with a managed UNIX system may result in the update propagated to other managed systems, including the central identity manager.

Policy-Based Identity Provisioning

Automation of account provisioning on the managed services and systems is an important element of reducing cost in enterprisewide identity management. Once an entity such as a user is defined to the central identity manager, it is likely that the same entity will require creation of accounts on one or more of the managed services and systems. Policy-based account provisioning refers to setting up provisioning policies to perform this automation process. Such policies can be based on various conditions such as role, position within the organization, or possession of a particular attribute. They should be easy to develop, be flexible enough, and allow for coarse and fine granularity. For example, a coarse policy may state that all users in a particular organization will automatically have accounts on a designated managed service. A finer policy may state that such accounts be created only to individuals with a particular job function.

Unified Identity-Representation Scheme

In this scheme, the centralized identity manager defines and maintains a superset of attributes that can be assigned to a managed entity such as an end user (Figure 2.19). Managed target services contribute to this overall superset of attributes by introducing attributes of their own. A managed service therefore may be aware of only a subset of the overall attributes. A record with the full set of attributes is maintained for each managed entity by the central identity manager. Some attributes in this record may not necessarily have values assigned to them. For example, a user that does not have an account on a particular service will not require values for any of the attributes that are specific to that service. A mapping may be needed to relate an attribute defined by the central identity manager to the corresponding attribute on

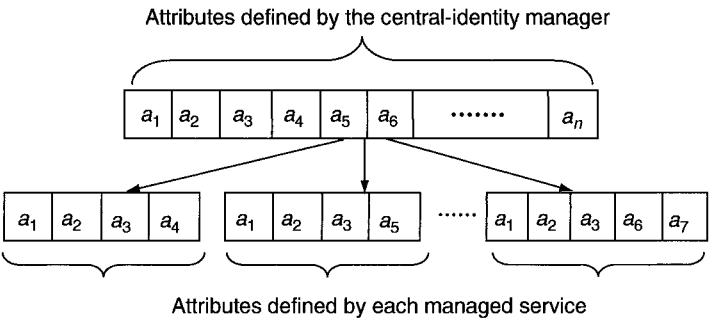

FIGURE 2.19 Attribute relationships between the central identity manager and the managed services in the case of the unified identity-representation scheme

a managed service. This definition would take place during the process of defining the managed service to the identity manager. Multivalued attributes are used to maintain the fact that the same attribute is assigned different values depending on the target service in which the entity has an account. For example, due to conflicting identity policies, a user identifier (uid) may be required to have different values on each target service where an entity maintains an account.

This scheme offers the benefit of maintaining all identity data in one central repository in addition to the fact that data is replicated piecewise across the managed services. Attribute retrieval operations therefore can be processed at the identity manger layer and do not require involving the managed services.

The drawback of the unified-identity-representation scheme is that it does not easily allow for dynamic changes to the schema representing the unified identity. Such changes can be easily introduced when a managed service defines attributes of its own and they are not already known to the identity manager layer. The change in the identity schema as such may require reconfiguring the identity-management system. Furthermore, one cannot expect to indefinitely keep defining new attributes that are sparsely common to the managed services.

Dynamic Definition of Identity Attributes

If we think of a representation of an identity as being a set of attributes and associated values, the first of the issues addressed in such a unified identity-representation model is the size of attributes that can possibly be assigned to an identity. Each of the target-managed services may contribute its own set of attributes that may or may not be common with other services. The unified identity that is visible at the centralized identity-manager level may require dynamic redefinition and potentially will be associated with more and more attributes. These dynamic changes may require periodic redefinitions in the data model used by the central-identity manager. Implementation examples include a change in the schema used by an underlying LDAP repository or that of a relational database system. Due to the impact of redefining the set of unified attributes that an entity may possibly have at any of the managed systems, careful thought needs to be given to the set of attributes to use early in the deployment stage of a centralized enterprise-identity manager.

Decoupled Identity-Representation Scheme

In this scheme, the central-identity manager maintains the values of a fixed set of attributes for every managed entity. Data relating to service specific attributes is kept at the target service. The identity manager remains aware of the schema for the attributes of the managed service, however. The key benefit here is the flexibility by which a service can be added to the identity-manager pool of managed services without impacting the overall data

schema of the identity manager. Any operations that apply to attributes that are service specific will require the interaction with the underlying managed services. Availability of these services, therefore, is necessary, whereas in the unified-identity-representation scheme such operations can take place in the identity manager and be scheduled to side-effect the managed service later when the services are available. Figure 2.20 illustrates the attributes relationship for this scheme. Attributes b_i are specific to the managed services.

Example: IBM Identity Manager

The IBM-Tivoli identity-manager (TIM) product adopts the unified identity-representation scheme that we previously defined and represents the latest in enterprise identity-management technologies. TIM maintains identity information about the entities that it manages in a central LDAP repository where an organization is modeled as a hierarchical structure that is horizontally scalable. A large number of related or independent organizations can coexist below a single root organization. TIM is a Web-based application that executes within a Web application-server (WAS) environment. Its design is highly modular and is composed of various independently developed components, each of which addresses a separate concern. Examples include workflow management, policy management, identity and password policy management, as well as reporting. But most important perhaps is the remote-services component that enables distributed systems and application subsystems that may exist in an enterprise to become TIM-managed resources. As a demonstration of its modularity, a special such managed resource is the TIM service in itself. Managed services and systems can be incrementally added as needed. The interaction of TIM with a managed service is accomplished through the deployment of a service agent, also referred to as an *adaptor* or a *connector*. A service agent acts as the intermediary to the managed service, and thus from one side it adheres to the protocol interactions with TIM that are common to

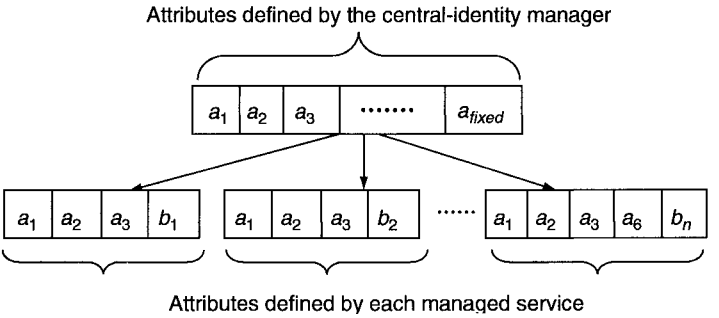

FIGURE 2.20 Attribute relationships between the central-identity manager and the managed services in the case of the decoupled attributes representation scheme

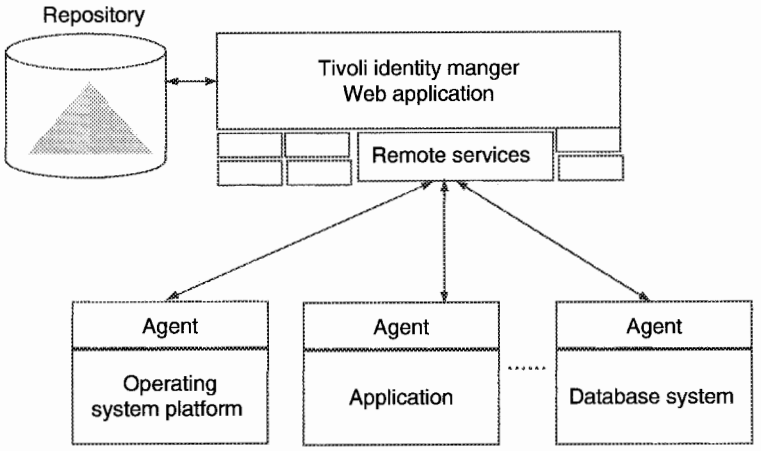

FIGURE 2.21 A high level view of the IBM-Tivoli identity manager structure

all agents, while on the other side it interacts with the target service using the service's native protocol interface.

The modeling of an enterprise in TIM begins with the definition of the hierarchical organizational structure. Each user of the enterprise is represented as a person entity. Such entity becomes an active user of any of the managed services, including TIM, by way of acquiring an account for that service. In which case, the user is said to be provisioned on the target service. Each service may contribute its own subset of identity attributes. Various policy-based rules can be used to automate identity provisioning within an organization structure. Synchronization of identity attributes across multiple managed services can also be achieved. Furthermore, reconciliation of existing identity registries with the TIM central repository can be performed.

TIM access-control mechanism enables flexible controls over the managed entities and objects residing in its repository, which is further enhanced through delegated administration support. Controls can apply at a coarse level (such as an organization) or at a much finer level (such as an identity attribute). TIM adopts a role-based model in its provisioning policies as well as in the controls it asserts over the managed constructs. Figure 2.21 represents a high-level view of the logical structure of TIM.