A Secure Platform for Peer-to-Peer Computing in the Internet

Wooyoung Kim, Sven Graupner, Akhil Sahai
HP Laboratories Palo Alto

HPL-2001-324
December 10th , 2001*

E-mail: wooyoung_kim@hp.com, {sven_graupner | akhil_sahai} @hp.com

peer-to-peer, security, discovery, vocabulary, virtualization, advertising service

Peer-to-peer computing (P2P) draws growing interest as a new distributed computing paradigm for its potential to harness "edge" computers (e.g., PCs) and make their under-utilized resources available to each other. P2P based e-commerce on the Internet is of particular interest because of P2P's cost effectiveness and redundancy-induced dependability. Beneath the promising benefits lie daunting challenges of supporting security, reliability, resilience, and scalability. In particular, scalable discovery and secure transaction are of paramount importance due to the sheer size and the laissez faire nature of the Internet. E-Speak is an e-services infrastructure where services advertise, discover, and interoperate each other in a dynamic and secure way. The E-Speak security adopts a multi-layered approach and builds a range of protection mechanisms on top of the Public Key Infrastructure. The E-Speak advertising services have a dynamic pluggable architecture and implement a scalable wide-area discovery based on distributed queries. We argue that E-Speak may be used as the common secure, scalable infrastructure for different multiple P2P applications.

A Secure Platform for Peer-to-Peer Computing in the Internet

Wooyoung Kim
Web Services Operation, Hewlett Packard
10450 Ridgeview Court ms 49EL-FR,
Cupertino, CA 95014, USA
wooyoung_kim@hp.com

Sven Graupner and Akhil Sahai
Hewlett-Packard Laboratories
1501 Page Mill Road,
Palo Alto, CA 94394, USA
{sven_graupner|akhil_sahai}@hp.com

Abstract

Peer-to-peer computing (P2P) draws growing interest as a new distributed computing paradigm for its potential to harness "edge" computers (e.g., PCs) and make their under-utilized resources available to each other. P2P based e-commerce on the Internet is of particular interest because of P2P's cost effectiveness and redundancy-induced dependability. Beneath the promising benefits lie daunting challenges of supporting security, reliability, resilience, and scalability. In particular, scalable discovery and secure transaction are of paramount importance due to the sheer size and the laissez faire nature of the Internet. E-Speak is an e-services infrastructure where services advertise, discover, and interoperate each other in a dynamic and secure way. The E-Speak security adopts a multi-layered approach and builds a range of protection mechanisms on top of the Public Key Infrastructure. The E-Speak advertising services have a dynamic pluggable architecture and implement a scalable wide-area discovery based on distributed queries. We argue that E-Speak may be used as the common secure, scalable infrastructure for different multiple P2P applications.

1. Introduction

Technological innovations on microprocessors and storage devices have furnished regular PCs, let alone workstations, enormous computing power and storage capacity. These computers usually come with more capability than is needed, resulting in subpar utilization. Peer-to-peer computing (P2P), whose name was coined after the scheme that stores and delivers contents using peer users' computers, draws growing interest as a new distributed computing paradigm for its potential to harness "edge" computers, such as PCs and handheld devices, and make their under-utilized resources available to each other.

The exact definition of term "P2P" varies widely depending on the context it is used in. Some define P2P as a class of applications operating in an environment of unstable connectivity [29]. Others view pure P2P as applications without centralized servers [7, 11]. Some even use it as an umbrella term for technologies that increase utilization of information, bandwidth, and computing resources in the Internet [31]. We adopt a general definition and define P2P as computing on a network of computational entities where the role of an entity (such as client and server) is determined per transaction basis. An entity in P2P which behaves as a client in a transaction can be involved concurrently in another transaction as a server.

P2P replicates and distributes resources and services and allows users to access them from "leaves" in the network. Replication implies high availability. Distribution yields balanced and efficient utilization of resources such as compute cycles, storage capacity, and network bandwidth, thereby avoiding hot spots and reducing access time. The redundancy-induced dependability and the potential cost-effectiveness make P2P an attractive platform for e-commerce on the Internet. Beneath the promising benefits however lie daunting challenges of supporting security, reliability, resilience, and scalability. In particular, scalable discovery and secure transaction are of paramount importance due to the sheer size and the *laissez faire* nature of the Internet.

Most existing P2P platforms, whether intended or not, are designed exclusively for sharing only one or two types of resources [25, 22]. Security for these systems is not as critical as for enterprises which weigh adopting P2P for collaboration across their organizations as well as in the Internet. As a consequence they do not address security issues to the extent that is required by the enterprises. Other areas that we identify today's P2P platforms fall short in support on include structured resource description and scalable discovery, identity verification, distributed authorization and access control, secure end-to-end communication, and coordination among services. These are evidently crucial requirements for successful implementation of secure and reliable e-business applications on P2P platforms.

E-Speak is an e-services platform that allows e-services to advertise, discover, and interoperate each other in a dynamic and secure way. It hides details on underlying hardware, operating systems, network topology, and other machine-specifics, and creates an abstract and uniform view of a network of resources and services. Resources and services are described and advertised in one or more vocabularies and are dynamically discovered with structured queries. Interaction between services is based on asynchronous message passing and end-to-end security between services is guaranteed with the E-Speak session layer security [16]. Advertising services in E-Speak form a loosely coupled network to provide scalable service discovery in wide area. Its event distribution services are used for distributed collaboration among services.

We argue that E-Speak can serve as a common secure platform for multiple P2P applications. In particular we present secure collaboration and scalable wide-area discovery in E-Speak, the two most challenging problems in P2P-based e-commerce in the Internet. First, we introduce the computation model and the system architecture of E-Speak. The next section explains E-Speak's layered security mechanisms in detail, which is followed by discussion on private shared name spaces. The E-Speak security is built on top of the Public Key Infrastructure (PKI) and offers a range of protection mechanisms including authentication, content integrity, visibility control, and capability-based access control. E-Speak advertising service is described in Section 5 which implements a flexible and scalable solution to wide-area discovery. Its search constraints are qualified with vocabularies to partition search spaces and enable juxtaposition of multiple P2P applications. Finally, we briefly make comparisons of E-Speak to other research and development efforts and conclude the paper.

2. Background

2.1. E-Speak Computation Model

The flavor of P2P inherent in E-Speak comes from the abstraction of *resource* and the interaction model built on asynchronous message passing. [1] A *resource* is representation of an entity's metadata in E-Speak [16]; all entities including services and clients are represented and managed as *resources* and referenced with *resource handles*. Services and clients interact with each other through asynchronous message passing. A *resource* is associated with a message box to which E-Speak delivers requests and replies. E-Speak ecosystem, a network of E-Speak engines, provides connectivity and reachability among *resources* (Figure 1). Ecosystems grow or shrink dynamically as service engines join and leave the systems or other ecosystems are introduced.

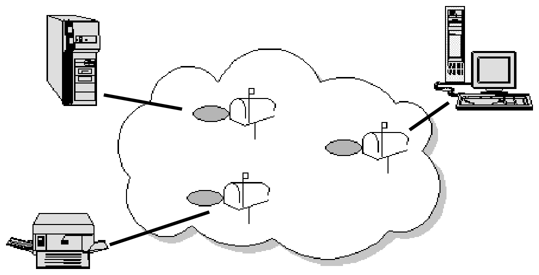

Figure 1. E-Speak Computation Model. Each gray oval represents a *resource*.

Separating metadata from services allows secure resource sharing. The metadata are advertised to other engines while the actual service is kept locally so that accesses to the service are mediated. The owner may trust the local E-Speak engine and rely on access mediation by E-Speak. Or, she may elect to enforce access control policy on the service for herself. Another benefit of the separation is *resources* representing metadata create uniform views to otherwise heterogeneous services. Consequently, to accommodate heterogeneous services in E-speak engines becomes simpler, and engine implementation does lighter. Because services are external to a service engine, they can be dynamically relocated and redeployed without surprising clients.

2.2. Specification, Descriptions, and Vocabularies

E-Speak services and resources are registered with a specification and descriptions. Description represented by a set of *attributes* (i.e., name-value pairs) is about how the entity is presented to users while specification contains information on how to access the entity. Service specification is composed of interfaces describing how clients interact with the service, security information used to enforce Public Key Infrastructure (PKI)-based security, and a filter constraint which specifies those who may discover the service. Upon request descriptions may be advertised to other E-Speak engines, but some of specification may not. The dichotomy of the resource representation is the basis of the the flexible yet secure service discovery in E-Speak [12].

Descriptions are described in a *vocabulary*. A vocabulary defines valid attribute names and their types, composing a discrete namespace. Descriptions are validated against the specified vocabulary. In this sense, vocabularies is for

[1] E-Speak does not distinguish active services from passive resources. It is concerned only with representation of their metadata. Thus, we use resources and services interchangeably. However, we reserve the italic *resource* to refer to the E-Speak abstraction exclusively.

descriptions what types are for values in programming languages. The use of vocabulary is two fold. One is to facilitate the namespace sharing and the other is to avoid the potential name collision problem in multiple descriptions. Since vocabularies naturally partition the search space of descriptions, they may evolve over time independently of other vocabularies. Vocabulary is one of means to allow multiple peer-to-peer communities to co-exist in a single E-Speak infrastructure.

Vocabulary itself is an E-Speak *resource* and is described in some vocabularies which require yet others and so on. This seemingly infinite recursion on vocabularies is ended by the use of the base vocabulary. The base vocabulary is not described in any vocabulary and is uniquely defined across all E-Speak engines with attributes `Name`, `Type`, and `Version`. Figure 2 shows an XML document which represents a vocabulary registration request. The newly created vocabulary is described in the base vocabulary and named `Used-Car`.

Vocabulary being resource also means that anyone can create a vocabulary and register it with E-Speak. Thus, it is likely two identical vocabularies exist in an E-Speak ecosystem and cause vocabulary conflicts in service lookup. One solution is to use structural equivalence which declares equivalent two vocabularies registered by the same user with the same definitions and the same descriptions. Vocabulary equivalence is beyond the scope of the paper and should be dealt separately.

```
<?xml version="1.0"?>
<resource xmlns=
   "http://www.e-speak.net/Schema/E-Speak.register.xsd">
 <resourceDes>
  <vocabulary>
   http://www.e-speak.net/Schema/E-Speak.base.xsd
  </vocabulary>
  <attr name="Name"><value>Used-Car</value></attr>
  <attr name="Type"><value>Vocabulary</value></attr>
 </resourceDes>
 <attrGroup name="used-car" xmlns=
     "http://www.e-speak.net/Schema/E-Speak.vocab.xsd">
  <attrDecl name="make" required="true">
   <datatypeRef name="string"/></attrDecl>
  <attrDecl name="model" required="true">
   <datatypeRef name="string"/></attrDecl>
  <attrDecl name="ask-price" required="true">
   <datatypeRef name="float"><default>0.0</default>
    <minInclusive>0.0</minInclusive></datatypeRef>
  </attrDecl>
    ...
 </attrGroup>
</resource>
```

Figure 2. A vocabulary creation request.

2.3. Queries

Users discover services by constructing queries and looking up the E-speak ecosystem. Queries may be evaluated in the local E-Speak engine or they may be sent to E-Speak advertising services. A query contains a constraint, zero or more preferences, and an arbitration policy. It may have vocabulary declarations if attributes used in its constraint or preferences are from multiple vocabularies. The constraint specifies a condition that services of interest must satisfy. The engine applies the preferences collectively to order the results. Arbitration policy specifies how many results are returned. Figure 3 shows an example query where a user is trying to find those who advertise both a used Honda Prelude and a second-hand dragon book for sale. Two vocabularies, `Used-Car` and `Used-Book` are referenced. The preference tells the user prefers offers with the smaller asking price for the car.

```
<?xml version="1.0"?>
<esquery xmlns=
   "http://www.e-speak.net/Schema/E-Speak.query.xsd" >
 <from src="http://www.john-doe.com/" />
 <vocabulary name="car" src="Used-Car" />
 <vocabulary name="book" src="Used-Book" />
 <result>$serviceInfo</result>
 <where>
  <condition>car:make="Honda" and car:model="Prelude"
    and book:author="Ravi Sethi" and book:title=
    "Compilers: Principles, Techniques, and Tools"
  </condition>
 </where>
 <preference><operator>min</operator>
            <expr>car:ask-price</expr></preference>
 <arbitration><cardinality>all</cardinality></arbitration>
</esquery>
```

Figure 3. A lookup request

2.4. Communication

Messages are sent with target service's resource handle and the E-Speak ecosystem of service engines delivers them to the target's message box. Service engines hide differences on hardware platforms from clients and services and forge a network to present them the logical view of the system. Each engine routes messages using only locally available information.

A message box is a pair of *inbox* and *outbox*; the former for inbound communication and the latter for outbound communication. When the sender and the target of a message reside in the same engine, the message is sent to the sender's outbox to the receiver's inbox. From the inbox the message may be pushed to the receiver (*push*) or the receiver may pick the message up (*pull*). Message delivery involving more than two engines is a simple extension to the single engine case (Figure 4) with neighboring engines being managed as *resources*. A message is sent to the sender's outbox to the inbox for the remote engine. There it is sent to (or, picked up onto) the outbox for the local engine in the remote engine. The remote engine will deposit the message in the receiver's inbox if it can locate the receiver locally. Otherwise, the same delivery process is repeated until the message is delivered.

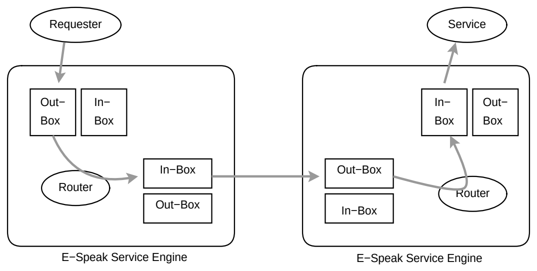

Figure 4. Logical view of communication involving two service engines.

3. E-Speak Security

The E-Speak security assumes a very broad threat model and is very frugal in making assumptions on trust. It assumes crackers snoop communication, infiltrate systems, and retrieve confidential information. Some may even impersonate businesses and spoof and defraud others. It also assumes internal employees or people with access permissions for the network may be involved in unauthorized (or even illegal) activities such as eavesdropping. The E-Speak security is designed to ward off attacks ranging from traffic analysis to eavesdropping to message tampering to deletion to identity theft.

In addition to the threat model we made three deployment assumptions. First, no central security administration is available. The security system should operate in a decentralized environment. Second, the security infrastructure should scale up to millions of machines. Third, message confidentiality and message authentication should not be taken for granted. Coincidentally, these are defining characteristics of peer-to-peer systems in the Internet.

The threat model together with the assumptions led us to the cryptography-based security (in particular, PKI) with unrestricted certificate issuance, the split trust model (similar to Pretty Good Privacy [36]), and end-to-end message encryption and authentication. In E-Speak issuing certificates is no longer a privileged operation; anyone can issue certificates. Whether or not a certificate grants access to any *resource* depends on if the *resource* owner trusts the issuer of the certificate. An entity need not trust all issuers equally (thus, split trust). It may trust issuers only to the extent to issue certificates granting access to a subset of its operations [16, 36].

E-Speak entities have varying authentication requirements. Service engines need to authenticate other engines

field	description
issuer	public key of the certificate issuer
subject	public key, key hash, or object hash identifying an entity to which the certificate is issued; use of object hash allows capability certificates to be issued to web pages or programs
tag	capabilities transferred from the issuer to the subject
delegation	boolean flag stating whether the subject is allowed to delegate the capabilities
validity	a set of expressions that must be evaluated to `true` for the certificate to be valid; it includes an expression stating the valid period of the certificate
signature	digital signature of the issuer

Table 1. Fields in an E-Speak capability certificate.

for their identity and properties, users for their identity, profile information, and privileges, and *resources* for their identity and metadata (i.e., descriptions and specification). A user needs to authenticate a service engine she logs in as well as any *resources* for their identity and metadata which she discovers through the service engine. A service provider (or service) needs to authenticate a service engine it connects to and capabilities presented by a client to access the service.

These authentication data are represented in terms of capability certificates. The term "capability" denotes a named access right or a named property. Capabilities are issued to subjects in the form of capability certificate signed by the issuer. A capability certificate states that any entity which is able to demonstrate knowledge of the corresponding private key has been transferred the rights listed in the certificate by the issuer. Digital signatures are used to protect capability certificates.

Informally, capability certificates can be represented as an ordered 6-tuple, <*issuer*,*subject*,*tag*,*delegation*,*validity*, *signature*>. Short description on each field is given in Table 1. Capability certificates containing access rights are called authorization certificates. Identities and properties are certified with name certificates whose tag field is empty. Access rights are expressed in the tag field. A tag is a list of lists, with each list delimited by a pair of parentheses. For example,

```
(tag (files (* prefix //ftp.e-speak.hp.com/pub/)
            (* set read write)))
```

may grant read and write access to any file in the `pub` directory including subdirectories. The tag-prefix form `(* prefix ...)` is used to specify a set of objects whose name begins with the specified prefix while the tag-set form `(* set ...)` is used to specify a group of permissions. The E-Speak security defines only the syntax of tags. Services who wish to enforce access control should define proper interpretation of their tags; two services may interpret the same tag differently.

Using the capability certificates E-Speak defines a three-pronged security system. The underlying secure communication is implemented based on provable identity and cryptographic encryption using Simple Public Key Infrastructure (SPKI) [10]. On top of it lie visibility controls on *resources* and capability-based access control.

3.1. Secure Communication

All features related to secure communication in E-Speak are implemented in the Session Layer Security (SLS) which extends Secure Socket Layers (SSL) [9]. The two communicating parties use SLS to perform a handshake including Diffie-Hellman key exchange to create a shared secret and establish a secure session. In particular, the E-Speak security uses a Diffie-Hellman based on elliptic curve cryptography [5]. The handshake may actually happen over multiple service engines through the firewall. SLS's tunnelling support which nests a secure session inside another one, possibly with different end points, is used to establish the end-to-end security [2]; no intermediary engines may see messages in the clear text.

During the handshake, the two end points negotiate capability certificates, secure channel identifiers, and a cipher suite list. Capability certificates are exchanged to verify capabilities. Secure channel identifiers are used to specify which keys to use for encryption or decryption especially when multiple secure channels are created at an end point. They are randomly generated to avoid a denial-of-service attack [16]. A cipher suite is a full set of cryptographic algorithms for secure channel. Both end points should agree on which cipher suite to use.

After the handshake is completed, all traffic between the two end points is encrypted and authenticated using the agreed cipher suite so that no attackers may tamper the communication without being detected. Symmetric encryption is used for speed after the initial authentication and key exchange. Messages contain session information and a sequence number to protect communicating parties from insertion, deletion, and reply attacks.

3.2. Visibility Control

You cannot attack unless you know what to attack. Visibility control means clients find no more services than permitted by their trust level, access rights, and profile information, ultimately protecting services from possible identity-related attacks. E-Speak provides service providers with a range of visibility control mechanisms.

The basis of visibility control in E-Speak is name virtualization. When requested by a client, E-Speak virtualizes names that identify services. With name virtualization neither service providers nor clients need reveal their true identities in order to interact with each other. The hosting engine keeps the mapping from virtual to actual resource handles. Together with dynamic discovery, name virtualization may be used to implement dynamic fail-over, seamless run-time upgrades, transparent service relocation, and load balancing with service replication. Name virtualization is quite a useful abstraction and deserves detailed discussion of its own (Section 4).

E-Speak vocabulary offers another visibility control mechanism. Since attribute names in a query need be qualified with a vocabulary reference (see Figure 3), those who do not know about a vocabulary in which a service is described may not construct a query that fetches the service. A service provider protects her services from being discovered by unwanted intruders by creating her own vocabulary and making it visible only to her preferred clients. She may even attach access control policies to the vocabulary. A service engine does not return any results unless a requester presents valid capabilities for vocabularies used in her query.

The most conspicuous form of visibility control is to allow only a certain group of users to discover a certain set of services. A mortgage broker may want clients with good credit history to find mortgage programs with preferred interest rate. A chip design company may want only its chip designers to find high-resolution plotters and printers. This kind of visibility control is specified by using *filters*. Filter constraint is a predicate over service attributes and user profile information. Users are represented as *resources* and their profile information is described in vocabularies. The filter constraint of a service is evaluated when the service's description matches a user's query; the integrity of user profile information should be authenticated before the evaluation. Any negative evaluation removes the service from the result set to be returned.

3.3. Capability-based Access Control

In E-Speak a client should present a valid capability certificate to access a service. The service authenticates the capability certificate by verifying the client's knowledge of the private key corresponding to the given public key. The results of authentication are cached so that the same authentication need not be repeated on every access. Different access rights to a service can be granted depending on authenticated authorization certificates.

However, presenting a valid capability certificate does not necessarily guarantee access to the service. Before granting the access the service must be able to establish the authority of the issuer of the certificate for the capabilities in the certificate. Services have their own list of trusted cer-

[2]Supporting end-to-end security using SSL is very difficult.

tificate issuers to whom they confer the authority. Only capabilities that are contained in valid certificates issued by trusted issuers are authorized. Only when the service verifies that the validated certificates actually grant the access, the service is rendered and the request is processed.

3.4. Certificate Delegation and Revocation

E-Speak supports SPKI delegation to allow entities to delegate capabilities. Whoever possesses a capability certificate with the `delegate` field `true` may construct and distribute delegate certificates to become an issuer. However, it cannot issue more capabilities for the entity than the delegate certificate allows, unless it is trusted by an entity verifying the certificate. This is enforced by intersecting the authorizations specified by all tags in the delegation chain and taking the smallest validity period [16].

Every capability certificate expires after their validity period. E-Speak primarily relies on this certificate expiration to revoke certificates. Clients that want to continue accessing a service must renew the corresponding certificate periodically. In addition, service engines and services may maintain certificate revocation lists (CRLs) at their will. A particular capability certificate is revoked by placing its hash on a CRL. All delegate certificates of a revoked certificate are implicitly revoked because they eventually produce the revoked certificate and thus can no longer be verified. Certificate revocation is one of few gray areas in E-Speak and warrants further investigation and development.

4. Private Shared Name Spaces

Collaboration and resource sharing in distributed environments require names such as addresses, URLs, and resource handles, be agreed on; a name used in an interaction should denote the same thing for all entities involved. Peer-to-peer computing as seen today in the Internet relies on the global persistent name space (GPNS) of the Internet to resolve the name agreement problem. Napster-like systems register descriptions of resources (e.g., music titles) with addresses from which the resources can be obtained.

GPNS in the Internet has proven quite effective in sharing public names. On the other hand, it is difficult in GPNS to protect names or selectively share names to the extent demanded in e-commerce in the Internet. Many market participants want to remain anonymous. Service providers want to reveal to customers only names that are needed for successful completion of a transaction. They also want the names to be invalidated after the transaction is completed, keeping customers from sharing the names without proper authorization.

Besides the inability to protect names GPNS has its own drawbacks. Global names are intended and optimized for long-lived entities. Experiences with globally unique object identifiers (OIDs) in middleware systems [33] have shown that the frequent allocation and deallocation of global names is costly. Also, recycling those names may surprise users who cache them for repeated access.

Private shared name spaces (PSNS) in E-Speak complements GPNS by providing privacy and identity protection much needed in e-commerce applications. It is an effective means against identity theft in the Internet. PSNS is session-based; it is established between two service engines at the beginning of a session and reclaimed when the session is ended. It is private because the name space is visible only to the two service engines. SLS's handshake process verifies the identities of the two engines.

4.1. Virtual Names

E-Speak implements PSNS using name virtualization. Virtual names hide actual identities of services and clients to fend off security attacks using traffic analysis. A PSNS defines bindings between virtual names and actual resource handles. Multiple PSNS's may co-exist independently of each other; no central coordination is needed among PSNS's. Virtual names are unique with respect to their PSNS and refer to one and only one entity within the PSNS. In contrast, a resource handle may be bound to multiple virtual names within a PSNS. Also, entities may be referenced from more than one PSNS.

Successful establishment of a secure session between two service engines initializes a binding table for each engine. When a resource marked as *virtualize* is to be exported to a remote engine, three events occur in sequence. A virtual name is allocated, the binding from the name to the resource handle is added to the binding table, and the virtual name is exported. The virtual name is used as the resource handle to the resource in the remote engine. The name is only visible and valid within the name space. Figure 5 shows three PSNS's between three service engines. `Myservice` in `PSNS(1)` refers to a service different from `myservice` in `PSNS(2)` if any. `Service X` is exported to two remote engines and thus is referred to in the two name spaces.

4.2. Message Routing

Given messages whose recipients are specified with resource handles in the default Internet/Web name space, a service engine will pass the resource handles (represented as URLs) to the underlying transport along with messages. No name translation is required for sending the messages. In contrast, for messages whose recipients are specified with virtual names a service engine translates the virtual names to get the connection to the target engine and sends the messages over the connection. At the target engine the

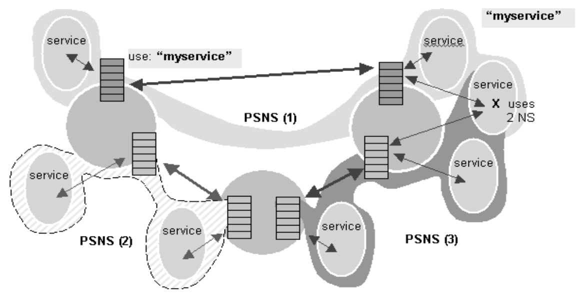

Figure 5. Three private shared name spaces with interacting entities.

binding table is accessed to get the actual resource handles and deliver the messages. Virtual names themselves may be virtualized and communicated in messages to other engines. In this case, pairwise name virtualization will create a routing chain of service engines.

Less obvious is use of virtual names against denial-of-service attacks. By creating virtual names with an address randomly selected from a pool of gateway addresses, we let requests from different origins follow different paths to reach a service. Even if one or two gateways are under attack, the majority of users can still access the service. Moreover, the attacks are spread to multiple gateways and attenuated, mitigating the damage to the system.

5. Advertising Service and Ad Hoc Discovery

Advertising service allows services to discover other services and resources that are otherwise disconnected and unknown. It is particularly challenging to implement the advertising service both efficiently in time and scalably in space in a dynamic wide-area network. E-Speak, in particular, advertising service was designed with flexibility as the a main design goal. Nevertheless, we obtained encouraging performance results as shown in Table 2 [3].

Security	Register	Un-register	Find	Service Invocation	Advertise
ON	8.9	12.2	114.8	437.6	9.8
OFF	9.1	11.7	117.2	574.7	19.5

Table 2. Throughput of a single E-Speak service engine (operations per second).

[3]For the measurement a service engine, an advertising service using multicast, 10 client applications, and a test controller were executed on a single machine with a 700 MHz Pentium III CPU and 256 MB main memory running Microsoft Windows NT 4.0. All the java programs were compiled and run with Sun JDK 1.3.0.

Advertising service is implemented as an external service to an E-Speak service engine (SE), a collection of them forming a logical advertising network. Advertising services external to E-Speak service engine allow flexibility in designing the E-Speak system. First, we can have multiple advertising services implementing different protocols, for instance one participating in the Napster community, another participating in the Gnutella community, etc. As a new exchange community sprouts, one can easily join the community by adding a new advertising service. Obsolete advertising services are taken away dynamically. Second, an advertising service and its hosting engine may reside in different machines. The advertising service may even be shared by other service engines. Should a failure occur on the advertising service, one can still access other services mediated by the engine.

When a service provider advertises its service, it sends appropriate advertising services the advertisement request. The advertising services announce the availability of the service to their respective community on behalf of the service provider. When an advertising service (AS) receives a discovery request (step 1 in Figure 6), it sends the request to the advertising service network and receives as results a list of connection objects for remote advertising services that might have matching services (step 2). Connection object contains a piece of information that is understood by a service engine (SE) to connect to a remote one. A typical connection object consists of a protocol name, a host address and a port number. The local AS uses the connection objects to connect to each remote SE through its local SE and sends the query to the corresponding remote AS. The remote AS returns a list of matching resource handles (step 3) which are in turn returned to the client (step 4). Note that the remote SE may contain *resources* that the owner does not want to share with others without proper authentication.

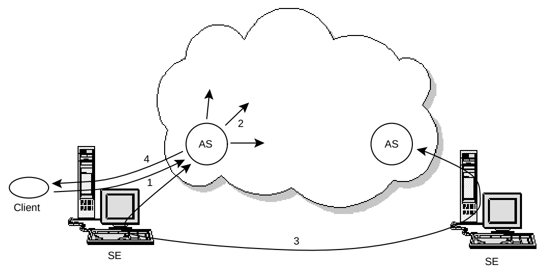

Figure 6. Discovery of advertised resources using advertising services

E-Speak relies on the domain name service (DNS) to resolve host names contained in connection objects. The implicit dependence on DNS manifests itself when engines are connected to the network with dynamic IP addresses (either Internet or intranet). If an advertising network needs to support engines with dynamic IP addresses, the advertising services in the network can be configured to use connection objects containing an IP address. When such an advertising service goes off line, it may - preferably - revoke all its advertisements. When it backs up, it can re-advertise those advertisements. Or, the advertisements remain in the advertising network until their time-to-live expires (these are called *timed advertisements*). The advertising service may come back before the expiration and renew the advertisements. At the worst case, an advertising service makes connection to a remote SE using a connection object only to find it is not available, which means only a little longer lookup time to a client. Timed advertisement is for scalable distributed garbage collection and similar to the advertisement refresh in SLPv2 [13] and the distributed leasing in Jini [30].

5.1. Implementation

The current release of E-Speak supports two kinds of advertising services, one using multicast and the other using a central repository. Advertising services configured to use multicast employ a discovery mechanism which resembles Service Location Protocol (SLP) without directory agents [13]. Each advertising service maintains a list of peer advertising services. (How to discover peer advertising services in the first place is discussed in Section 5.2.) When advertisement requests arrive an advertising service may multicast them to the others or simply keep them locally. For discovery, advertising services multicast the request to the others, collect the results, and return them to the client. The configuration with multicast is suitable for systems that span a local area network and manage a small to modest number of shared resources (cf. Jini [30]).

Advertising services of the second kind are built around a backend repository. They are designed to exploit the native multi-thread support of a backend repository. Use of a backend repository is only visible to advertising services. Advertising services form an opaque layer so that existence of a backend repository is invisible to end clients and services. The opaqueness allows for on-the-fly upgrade of backend repository to, for example, one running on a cluster of computers or a set of replicated/distributed repositories (e.g., UDDI [3]). The current E-Speak implementation uses an LDAP server as the backed repository. LDAP [20] has native support for multi-value attributes as well as rich queries of its own. The advertising services perform necessary schema and query translation.

On top of the advertising services E-Speak provides scalable wide area discovery using *community list*. A community is a set of service engines which share an advertising network. Community list is a list of representative advertising services for different communities which may be spread across the Internet. A user passes a community list along with a query when she looks up multiple communities. The E-Speak client library [17] unicasts the query to each advertising service contained in the list.

The aforementioned discovery mechanisms are of limited scalability though they are sufficiently scalable for today's existing e-commerce applications. To further improve the scalability we are looking into ways to embed multiple randomized trees [22, 27, 21, 14] into a set of communities in a hierarchical structure.

5.2. Initial Discovery and Join

For a service engine to join a community it needs to discover proximate advertising services in the community. The process of initial discovery and join allows an advertising service to obtain configuration parameters specific to the community, such as the URL of the backend repository, and adapt itself to the community. The dynamic initial discovery is particularly important when host machines are allowed to roam or they regularly disconnect from the network (e.g., laptops).

E-Speak advertising service (AS) provides a simple initial discovery mechanism using multicast. Without any hardware multicast support advertising services need be configured manually. A joining AS multicasts a `DSCV_RQST` message containing its own *resource* handle (called ESURL) on a predefined port. It may include in the message the name of a community it wants to join. The name is to uniquely identify the community when multiple communities are hosted in the same physical local network. The community name should be communicated among advertising services out of band before it is used.

ESURL contains connection information for a hosting service engine. Listening AS's that belong to the specified community save the ESURL and reply with a `DSCV_RPLY` message containing their ESURL. Those who do not belong to the community simply discard the request. The joining AS receives `DSCV_RPLY` messages and garners information about neighboring AS's in the community. In the case of advertising networks with a backend repository, existing AS's respond with configuration information on the backend repository instead. Note that the very first AS must bootstrap itself to the initialization.

6. Related Work

E-Speak is started as an attempt to bring to reality the vision of client utility [4] that e-services are a form of utility and should be readily accessible just like water and electricity. Many research projects [35, 18, 15, 23, 34] share the vision. Unlike those that developed everything from an infrastructure platform to participating devices, E-Speak has taken an open, horizontal approach and built a scalable, flexible platform with strong security support to host other vertical systems.

E-Speak's computation model resembles that of Actors [1]. Both support message driven interaction where requests are delivered to the message boxes (mail queues in Actors) of the targets in the form of asynchronous messages. The notable differences are two folds. First, metadata is separated from an actual resource in E-Speak; second, *resources* are discoverable. In contrast, actors have no metadata and become known to others only by communicating their mail addresses in messages. Resource discovery through advertising services relates E-Speak remotely to Linda [6] which uses templates to match tuples. E-Speak provides a flexible discovery framework with structured queries.

E-Speak security's split trust model is similar to the "web of trust" of Pretty Good Privacy (PGP) [36], a freeware electronic-mail security program; both assume no key certification authorities. A couple of differences are worth noting. First, E-Speak uses PKI not only for protecting message content but also for capability-based access control. Second, users may even remain anonymous in interaction with others by virtualizing their names whereas receiver's key ID is exposed unprotected in PGP. Session layer security (SLS) of E-Speak is closely related to Transport Layer Security (TLS) [9]. In fact, SLS extends TLS to provide transport independence, end-to-end security via tunnelling support, and attribute certificates using SPKI.

The design of the E-Speak advertising service has benefited from Service Location Protocol (SLP) [13]. For instance, the notion of scope was extended to support community-based wide area discovery. Related to the advertising service is Ninja's secure Service Discovery Service (SDS) [8] which supports a tree-based wide area discovery. Although the use of Bloom filtering mitigates space requirements near the root node, SDS suffers from the problem of the root node being a bottleneck. Systems in [22, 27, 21, 14] embedded multiple logical tree structures in a physical network to avoid the bottleneck problem. However, they came short in supporting efficient re-construction of embedded trees when nodes are allowed to join and leave the network dynamically.

A number of P2P applications have sprouted in the recent years [2, 32, 7, 11, 19, 24, 25, 28]. Particularly pertinent to E-Speak are those that support security for end-users [7, 22]. E-Speak has a flexible architecture which allows different P2P applications to co-exist in a single common platform. The JXTA project [31] by Sun Microsystems also aims to "juxtapose" different P2P applications on a single shared platform though it does not fully integrate security with the platform and provides limited security support in a separate layer. JXTA is in its infancy and further developments are yet to be seen. Peer-to-Peer Trusted Library (PtPTL) [26] allows software developers to add the element of "trust" to their P2P applications. However, their efforts are limited to providing P2P applications with library-level security support.

7. Conclusion

Growing interests in P2P as a scalable e-commerce platform have manifested many requirements that have been neglected intentionally or unintentionally in many P2P systems. In the paper, we discussed two important requirements, secure collaboration and scalable discovery in wide area, in the context of E-Speak. The E-Speak security is based on Public Key Infrastructure (PKI) and offers a range of protection mechanisms including authentication, content integrity, fine- and coarse-grained visibility control, and capability-based access control. E-Speak advertising service has an adaptable architecture to accommodate multiple P2P applications. It provides a flexible querying mechanism with the notion of vocabularies and user-controlled distributed queries for service discovery in wide area. The popularity of P2P has spawned many P2P systems with specific intent of usage and many more will arise. It is not foreseeable that any single system will dominantly subsume the space. We envision that users select and "juxtapose" best-of-breed P2P systems using E-Speak as a common secure platform.

8. Acknowledgements

We thank Alan Karp for his insight and inspiration on virtual naming. We acknowledge Nigel Edward, Mike Wray, and the E-Speak Bristol team for their work on E-Speak Session Layer Security and capability-based access control. We thank Thiruvalluvan Swaminathan at HP ESDV at Bangalore, India, for the E-Speak performance measurement.

References

[1] G. Agha, S. Frølund, W. Kim, R. Panwar, A. Patterson, and D. Sturman. Abstraction and Modularity Mechanisms for Concurrent Computing. *IEEE Parallel and Distributed Technology: Systems and Applications*, 1(2):3–14, May 1993.

[2] Aimster. http://www.aimster.com/.

[3] R. Atkinson and J. Munter. UDDI Version 2.0 Replication Specification, May 2001. UDDI Candidate Draft Specification.

[4] J. Birnbaum. IEEE Media Briefing: How the Coming Digital Utility May Reshape Computing and Telecommunications, October 1996. http://www.hpl.hp.com/speeches/ieee.html.

[5] I. Blake, G. Seroussi, and N. Smart. *Elliptic Curves in Cryptography*. Cambridge University Press, 1999.

[6] N. Carriero and D. Gelernter. Linda in Context. *Communications of the ACM*, 32(4):444–458, April 1989. http://www.cs.yale.edu/Linda/linda.html.

[7] I. Clarke, O. Sandberg, B. Wiley, and T. Hong. Freenet: A Distributed Anonymous Information Storage and Retrieval System. In *Designing Privacy Enhancing Technologies: International Workshop on Design Issues in Anonymity and Unobservability, Proceedings*, 2001. LNCS 2009, http://freenet.sourceforge.net.

[8] S. Czerwinski, B. Zhao, T. Hodes, A. Joseph, and R. Katz. An Architecture for a Secure Service Discovery Service. In *Proceedings of MobiCom '99*, Seattle, WA, August 1999.

[9] T. Dierks and C. Allen. The TLS Protocol Version 1.0, January 1999. Network Working Group RFC 2246.

[10] C. Ellison, B. Frantz, B. Lampson, R. Rivest, B. Thomas, and T. Ylonen. SPKI Certificate Theory, September 1999. Network Working Group RFC 2693. ftp://ftp.isi.edu/in-notes/rfc2693.txt.

[11] Gnutella. http://gnutella.wego.com/, http://www.gnutella.co.uk/.

[12] S. Graupner, W. Kim, D. Lenkov, and A. Sahai. E-speak: an Enabling Infrastructure for Web-based E-services. In *Proceedings of SSGRR 2000*, l'Aquilla, Italy, August 2000. http://www.ssgrr.it/en/ssgrr2000/papers/134.pdf.

[13] E. Guttman, C. Perkins, J. Veizades, and M. Day. Service Location Protocol, Version 2, June 1999. IETF Network Working Group RFC 2608. http://www.ietf.org/rfc/rfc2608.txt.

[14] J. D. Guyton and M. F. Schwartz. Locating Nearby Copies of Replicated Internet Servers. In *Proceedings of ACM SIGCOMM*, pages 288–298, 1995. http://www.acm.org/sigcomm/sigcomm95/papers/guyton.pdf.

[15] Hewlett-Packard Company. The Cooltown Project. http://cooltown.hpl.hp.com/.

[16] Hewlett-Packard Company. E-Speak Architectural Specification. Release A.0, January 2001. http://www.e-speak.hp.com/media/a0/architecturea0.pdf.

[17] Hewlett-Packard Company. E-Speak Programmer's Guide. Release A.0, January 2001. http://www.e-speak.hp.com/media/a0/programmers_guidea0.pdf.

[18] IBM Corporation. The IBM T-Spaces Project. http://www.almaden.ibm.com/cs/TSpaces/index.html.

[19] ICQ. http://www.icq.com.

[20] IETF. Lightweight Directory Access Protocol (v3), February 2001. Internet Draft. http://www.ietf.org/html.charters/ldapbis-charter.html.

[21] D. Karger, E. Lehman, F. Leighton, M. Levin, D. Lewin, and R. Panigraphy. Consistent Hashing and Random Trees: Distributed Caching Protocols for Relieving Hot Spots on the World Wide Web. In *Proceedings of STOC '97*, pages 654–663, May 1997.

[22] J. Kubiatowicz, D. Bindel, Y. Chen, S. Czerwinski, P. Eaton, D. Geels, R. Gummadi, S. Rhea, H. Weatherspoon, W. Weimer, C. Wells, and B. Zhao. OceanStore: An Architecture for Global-Scale Persistent Storage. In *Proceedings of ASPLOS 2000*, November 2000.

[23] MIT. The MIT Oxygen Project. http://www.oxygen.lcs.mit.edu/.

[24] Mojo Nation. http://www.mojonation.net/.

[25] Napster. http://www.napster.com.

[26] Peer-to-Peer Working Group. Peer-to-Peer Trust Library (PtPTL). http://www.peer-to-peerwg.org/tech/index.html.

[27] C. G. Plaxton, R. Rajaraman, and A. W. Richa. Accessing Nearby Copies of Replicated Objects in a Distributed Environment. In *Proceedings of SPAA '97*, pages 311–320, 1997. http://www.cs.utexas.edu/users/plaxton/ps/1997/spaa.psg.

[28] SETI@HOME. http://setiathome.ssl.berkeley.edu/.

[29] C. Shirky. What is P2P... And What Isn't, November 2000. The O'Reilly Network. http://www.openp2p.com/pub/a/p2p/2000/11/24/shirky1-whatisp2p.html.

[30] SUN Microsystems. JINI^{TM} Network Technology. http://www.sun.com/jini/.

[31] SUN Microsystems. Project JXTA. http://www.sun.com/jxta/.

[32] The ALPINE project. http://www.cubicmetercrystal.com/-alpine/.

[33] The Object Management Group (OMG). The Common Object Request Broker Architecture. http://www.corba.org.

[34] UC Berkeley. The Ninja Project. http://ninja.cs.berkeley.edu.

[35] M. Weiser. Some computer science issues in ubiquitous computing. *Communications of the ACM*, 36(7):75–84, July 1993.

[36] P. Zimmermann. *The Official PGP User's Guide*. MIT Press, 1995.