Peer-to-Peer Architectures and the Magi™ Open-Source Infrastructure

December 6, 2000

Gregory Alan Bolcer, Michael Gorlick, Arthur S. Hitomi, Peter Kammer,
Brian Morrow, Peyman Oreizy, Richard N. Taylor

Endeavors Technology, Inc.
19700 Fairchild Avenue, Suite 200
Irvine, California 92612
http://www.endtech.com/

ABSTRACT. Peer computing is the next stage in the evolution of the Internet and promises to be a versatile underpinning for an enormous variety of novel, entertaining, and useful applications. However, peer-to-peer applications pose many difficult challenges for developers. We examine a set of requirements for robust peer computing, propose a small set of architectural principles designed to meet those requirements, and discuss the systematic exploitation of those principles in the context of *Magi*, a general-purpose, peering infrastructure for both embedded and enterprise applications.

Table of Contents

1. Why Peer Computing Now?

Peer computing has been available in various forms for well over a decade. Why is it an important issue now? Until recently any two peers were roughly comparable—hosts with large memories, fast processors, and high bandwidth network connections. The revolution of peer computing (as understood today) replaces physical equivalence with an astonishing physical disparity. Devices literally no larger than a matchhead now interact on an equal footing with hosts whose capacity, by every metric imaginable, exceeds that of their lilliputian peer by a factor of 10,000 to 1. Thus, for the first time, personal digital assistants, wireless telephones, and kitchen appliances can peer with hosts anywhere within the network.

It is common practice to describe as "peer-to-peer" any relationship in which multiple, autonomous, hosts interact as equals. An autonomous host is useful in its own right even in the absence of others. The peering relationship implies that additional functions are available to the peers *collectively* as a consequence of their collaborations with other hosts. Known as the *network effect*, the value and extent of these added powers increases dramatically as the number and variety of peers grows.

The difference between peer computing then and now is the confluence of three profound and pervasive trends:

- The ease of interconnection
- The expansion of bandwidth
- The wealth of cycles

In other words, in a world where (network access) interconnection is universal, (network) bandwidth is plentiful, and (processor) cycles are inexpensive, peering among physical unequals is both natural and desirable. Within the new world of peer computing, standard Internet protocols are the universal level playing field. So long as a device obeys the network protocols required for peering, its size, shape, form, and location are irrelevant.

In contrast, the present form of the Web is based upon a client/server relationship. The dialogue between a client (a Web browser) and a Web server is always initiated by the client. The server is unable to take the initiative because, if the server contacts the client first, there is no software resident on the client that understands the server or can respond in kind. Peer computing replaces the asymmetric client/server relationship with a symmetric one in which all peers are simultaneously client *and* server requesting service of, and providing service to, their network peers.

For example, peering permits multiple hosts to share files seamlessly irrespective of their location on the network. Thus an office worker telecommuting from home can, with the appropriate peering infrastructure, transparently access files and information that reside

(perhaps on numerous hosts) within the office network. In fact, peer computing erases the distinction between the "home" network and the "office" network.

Peering has applications that extend far beyond file sharing. A home alarm, in its role as a network peer and in cooperation with other peers, can track you down through your Internet-enabled wireless phone or palm device, alert you, and supply snapshots of the home interior and grounds as events unfold (while simultaneously notifying the police). The peering computers of a geographically distributed development team can automatically cross-index and update project files. Team members will always have the latest data at hand (with appropriate security approvals) irrespective of where that data resides in the peering network. Automated cross-indexing and update is but one example of the role of automated software agents in a peering infrastructure. Such agents can search or monitor peer communities continuously, and when they find an item of interest, trigger complex workflows or transactions that span multiple peers and communities.

Peer computing encourages and empowers users to form durable, ad-hoc groups devoted to an interest or task. Group life-spans may range from hours (a single meeting) to years (an ongoing project). Whenever there are shared interests, awareness of the state of the group becomes itself a matter of interest. As peers enter and exit a group, other group members (themselves peers) may desire notification of their comings and goings. For example, a peer may be awaiting the arrival of a fellow peer in order to complete a transaction. Known as *presence*, it is but one of many forms of *event notification* in which peers generate events on items of note for the benefit of fellow peers. Event notification is a core service that aids the creation, evolution, and dissolution of collaborative peer groups.

Beyond the Basics

Peer computing, though attractive on its own, benefits greatly from integration with other emerging Internet technologies:

- *Internet-enabled wireless telephones* fit the Internet into your pocket. Peer computing allows your personal WAP (Wireless Access Protocol) phone to peer with all of your other computing and information devices at both the home and office, forming a virtual private network that integrates all of your disparate devices into a single, cohesive whole. By permitting another peer to host a surrogate or proxy for your WAP phone, one can exercise remote control over any device or computational process from anywhere at anytime

- *Software agents* are workflow programs that span a community of peers, and when triggered by peer-generated events, can initiate, monitor, and control a complex chain of actions spanning multiple communities. For example, Magi Dispatch, built atop the Magi peer computing infrastructure, manages the dispatch, tracking, and closure of field service repairs for service dispatch centers. Armed with no more than a WAP phone a service technician can be alerted to a service call, obtain driving directions, place part orders, and file status reports.

Why Peer Computing Matters

The evidence is incontrovertible: networks create value for all businesses. Consequently any enhancement of the value, flexibility, pervasiveness, or reach of digital networks has profound economic consequences. We briefly explore why peer computing matters by examining its multiplying effect on the value and extent of network services.

In any natural ecology, the small greatly outnumber the large. The world of computing seems no different. Approximately four billion embedded processors are manufactured every year to appear in everything from doorknobs to power plants while the number of desktop and server processors, though large in absolute terms, is but a small fraction of the number of embedded processors.

However, the technical fact of peering large numbers of disparate devices does not explain the promise or excitement of peer computing. For that understanding we turn to three laws of networks. With n being the total number of members (peering devices) in the network, then according to:

- *Sarnoff's Law*, the value of the network is proportional to n [Reed 1999a]
- *Metcalfe's Law*, the value of the network is proportional to n^2 [Reed 1999a]
- *Reed's Law*, the value of the network is proportional to 2^n [Reed 1999b]

Sarnoff's Law regards the network as a broadcast medium with few transmitters and many receivers (that is, only a scant few among the n members of the network are transmitters; the vast majority are receivers). Here the network represents the linear value of services targeted toward individual devices. Metcalfe's Law regards the network as a medium to facilitate intercommunication in which any single device may communicate with as many as $n-1$ other devices, thus allowing as many as $n(n-1) \approx n^2$ simultaneous transactions among pairs of devices. Here the network represents the quadratic value obtained by facilitating transactions. Finally, Reed's Law regards the network as a grouping medium in which as many as $2^n - n - 1 \approx 2^n$ nontrivial interest groups may coalesce. From this perspective the network represents the *exponential* value of group affiliations.

Peer computing increases network value in all respects: for Sarnoff by linearly increasing the number of transmitters and receivers of information, for Metcalfe by quadratically increasing the number of transaction participants, and for Reed by exponentially increasing the number and diversity of groupings within the network.

The value of peer computing will increase in two distinct stages: the *restructuring of the desktop* and the *dominance of embedded peer computing*. First, consider the ratio today (December, 2000) of Web servers to Web clients. There are millions of Web servers but *hundreds of millions* of Web clients with the vast majority of these clients running on desktop platforms with large memories, fast processors, and high bandwidth network connections. The first stage of the transition to peer computing transforms the enormous

numbers of clients into servers making their computational, storage, and information resources available for use by other peers. This stage alone is a tremendous boon because it increases the Metcalfe value of the network by order $(10^8)^2 = 10^{16}$ and the Reed value of the network by a jaw-dropping 2^{10^8}. Doing nothing but introducing desktop hosts as peers will have an enormous impact on the total collective value of the network. However, note that in this scenario the peering is taking place largely among physical equals; to the first order all of these desktop peers are roughly identical.

The value of the second stage in the evolution of peer computing swamps the value of the first stage by encouraging embedded devices to join the community of peers. While there will be on the order of 10^8 desktop and server peers there will be on the order of 10^{11} embedded peers over the next decade. We predict that the greatest opportunity for peer computing will be in the arena of embedded devices and that access to information on demand will become an essential aspect of our personal and corporate lives. The need for mobile and instantaneous access will be driven by:

- Cost-driven demands for ever-increasing efficiencies in resource allocation and consumption
- Business-driven demands for ever more responsive and well-informed decision-making
- Revenue opportunities from entertainment- and education-content distribution

These drivers will force manufactures to add increasing amounts of "information" (that is, embedded digital devices) to all manner of common appliances and manufactured goods and will lead over the course of the next decade to an aggregated network "value" in excess of $2^{10^8} \bullet 2^{10^{11}}$ units!

The peering infosphere of the next decade will be as rich and diverse as any natural ecology. However, the engineering and technical challenges are as great as the potential values suggested by the Sarnoff, Metcalfe, and Reed Laws. In the following sections we examine some of the challenges that await us.

2. Dominant Characteristics of Peer Computing

Information and services, once tightly held at the core of the network on expensive, centralized servers, is rapidly migrating to the periphery of the network. The network, once dominated by large resource-rich processors, is now populated by a variety of smaller devices ranging from laptops to personal digital assistants to cell phones to embedded controllers. However, despite their network presence, the information residing on these "edge devices" is largely inaccessible. It is an odd contradiction for the edge devices now contain and generate tremendous quantities of useful information but until recently lacked the means to directly share it with others. Instead of having to move (copy) this information to a central, shared server, peer computing moves the server to each of these devices. For the first time in the history of computing, personal and

embedded devices are powerful enough to run servers and network connectivity is ubiquitous enough to make it feasible and practical to do so.

The Internet has undergone radical change twice before in its brief history, first when it transitioned from the ARPAnet to the Internet and became a true network of networks, and again when the World Wide Web was introduced. Peer computing is the next phase change in that history and completes the transition of computing from the machine room and office to the home and the world at large. Peer computing grants individual access to, and ownership of, content and services. Much like the ownership of physical property the individual ownership of content and information services confers direct personal access, freedom of movement and use, and the granting of subsidiary rights to trusted partners. Peer computing empowers users to:

- **Place** content and services on any device insofar as it is practical to do so
- **Maintain** the privacy, confidentiality, and integrity of personal and proprietary information
- **Share** with family, friends, and colleagues
- **Govern**, administer, and create content and services
- **Access** content and services irrespective of location, time of day, source, or end device
- **Exercise** direct control anytime, anywhere, from any device
- **Specialize** presentation, content, and services
- **Steward** the delivery of services and content

However, the promise and powers of peer computing are not without their challenges. Collectively the powers enumerated above promise a revolution in ease of use, access, and control. The implementation of that promise requires technical advances in any number of dimensions including distribution, heterogeneity, mobility, sharing, scaling, user interfaces, and security. We discuss these powers below and their technical consequences.

Placement

Content, in the absence of arbitrary constraints, will naturally migrate to where it is most needed and accessed. It is the responsibility of a peer computing infrastructure to vigorously and systematically remove obstacles that impede the free and seamless transfer of content and services from one peer to another. Thus it must be possible to (re)name content and services; transfer them in a manner that respects their "type," encoding, metadata, and resource requirements; and locate them using search engines, both general and specific.

Content and service migration may also be automatic, that is, a side-effect of frequent or repeated access. Thus a robust peer infrastructure must accomodate a rich constellation of caching mechanisms. It is not important (nor even desirable) that the infrastructure favor one caching mechanism over another. Instead the architecture must be broad enough to support the transparent introduction of "intermediaries," peers whose role is to cache or migrate content and service from the origin to the point(s) of use.

Security

With great powers come great risks and the ability of peer infrastructures to casually slosh about large quantities of sensitive information is rightly regarded with suspicion and alarm. Any viable peer computing infrastructure is obliged to respect and accomodate the quadrumvirate of security:

- Authentication
- Permission
- Confidentiality
- Data integrity

It must be possible to authenticate the identity of users (and perhaps that of devices as well); allow users and agents to grant, retract, or deny permission to inspect content or engage services; guarantee the confidentiality of transfers through technical means; and, ensure the integrity of content at the end point. Establishing a policy-neutral, mechanism-rich, and robust security architecture in a peer computing infrastructure is a daunting prospect. Nonetheless, peers must respect the security concerns of their neighbors—acting in good faith to maintain mutually negotiated security policies using accepted, standard mechanisms.

Sharing

Sharing content and services is at the core of peer computing. However, sharing is at the discretion of the content or service owner who may choose to share with specific individuals, the members of designated groups, the public at large, or only users and peers who satisfy domain-, content-, or service-specific criteria. At a minimum, four distinct forms of sharing must be assisted by the infrastructure:

- *Sharing computation and data storage.* The shared goals of the peers is effective use of the aggregate computing and storage power of the hosts in the network. Processing and content is moved to peers that would otherwise be underutilized for an interval.

- *Sharing content.* Here the significance and value of metadata shines. Invoking their power to administer content users create their own annotations on data (such as keywords or opinions describing a document) thereby shaping its future use, establishing its value, or forming the basis for new services. The context in which the data is created or last touched may constitute, in itself, useful metadata (identifying, for example, platform-specific information). Metadata may also function as a

surrogate for the data, for example, abstracts of technical papers, may serve in lieu of the papers themselves for some purposes.

- *Sharing relationships.* While some shared content or service may be considered to exist on its own, without any relationship to other content or services, most information and service lives within a complex web of relationships. Links must be separable from the content in which they appear or the content that they reference and the peer infrastructure should recognize and support the exchange and sharing of links (relationships) just as it does content and service.

- *Sharing activities.* Teams of people engage in complex cooperative interactions. Sharing in this context means the sharing of actions and service, not just content. Fundamental to this form of sharing is the concept of event and notification services in which agents (individual users, computational processes, or assemblages of both) register interest in events and are notified in a timely manner when the event occurs along with its particulars.

Governance

The ownership of content or service implies governance—the control of who may use what, when they may use it, and in what manner. This requires the ability to describe such activities, effect them, and support their likely change over time. Simple governance may require nothing more than support for distributed authoring, by providing remote document locking, access control, and versioning. More complicated governance requires notification services ("I've finished working with this artifact"), as well as support for mobile task descriptions ("Whoever is available and has sufficient authority should act on this request"). The diversity of potential useful mechanisms is mind-boggling, ranging from the simple user/group/world access permissions of POSIX to elaborate digital rights management languages.

Access

The power of access, no matter what the location or device, requires that any peer computing infrastructure embrace, as a fundamental principle, the broad range and diversity of source and end devices. Peer computing does not imply that the peers participating in the exchange of information or service are commensurate in processing power, memory capacity, bandwidth, or any other like metric. A resource-constrained peer, such as a pager, may find itself participating as a "co-equal" with a large mainframe or a Web tablet.

Access demands that peers acknowledge the underlying differences of platform and negotiate with one another at a more abstract level—that of protocol and service. Heterogeneity is the rule rather than the exception in peer computing. Devices will offer unique services, but that diversity is supported by adherence to a common core set of principles. Consider briefly the wide variety of devices present on a corporate network, each device capable of contributing to peer-to-peer applications. Such devices span servers, desktops, laptops, Palm Pilots, printers, cell phones and fax machines. These devices differ, perhaps by orders of magnitude, in many respects including

communication bandwidth, power reserves, available memory, and the persistence of their network connections. A Palm Pilot, for instance, connected to the network via a cellular modem has very different characteristics than a desktop workstation connected via a 100 megabit-per-second Ethernet. Embracing these differences and accommodating them in a systematic and uniform fashion is a principal challenge for an inclusive peer computing infrastructure.

The power of access forces developers to favor protocols over platforms and adaptation over failure. In other words, peerage is the important quality, not implementation or hosting, and all peers are obligated to accommodate, to the extent permitted by their host platform and the computing resources at their disposal, the differences in the abilities of their fellow peers. This implies that the infrastructure define levels of "ability" and allow each peer to choose the level that best suits its platform. Each peered exchange may involve a brief negotiation as the peers determine a common level of service. We may expect larger, resource-rich peers to routinely accommodate smaller, resource-constrained peers by reducing their service expectations, transcoding content, or acting as proxies for service requests that exceed the capabilities of their less capable brethren.

Control

The power of control guarantees control at a distance—the ability to regulate and command any peer from any other peer (modulo access, permission, and the limitations of the end devices). In other words, control is effected by any end device that is capable of establishing an appropriate peering relationship with the controller (itself a peer device). Thus a cell phone may be pressed into service to adjust a home entertainment system and a wireless PDA employed to query and test a remote pumping station. Peer computing infrastructure bears the burden of erasing (through translation and transcoding) artificial or proprietary barriers between the device or process being controlled and the device or process effecting the control and providing feedback to the agent or user.

Specialization

Accompanying the power of universal control and access is the power of specialization. This power has two faces. From the user's perspective it is having information and services presented and accessed in a manner suited to one's needs and tastes (personalization). From the peer's perspective it is the power to specialize, that is, to offer peer-specific content and services that differ (specialization), in perhaps substantial ways, from the content and services offered by like peers.

Personalization acknowledges the rights of individuals to shape and view the infospace in a manner respectful of their gifts and limitations. This implies that personalization, like any other content or service in the peer infrastructure, follows the user about ensuring (perhaps through a complex web of intermediaries and services) that the preferences of the user (or agent) are enforced in a context-appropriate manner.

Specialization allows a peer to narrow its offerings to just that set of content and services appropriate to the device. Both a cell phone and an MP3 player can be peers but offer

content and services suited to their functions. Thus a cell phone may provide a call history as one of its content elements while for an MP3 player a play list is an appropriate choice. Beyond a simple core set peer computing encourages peers to differentiate and specialize—all the better to increase the diversity and variety of peer content.

The dual power of personalization/specialization has another subtle, but profound, consequence—freedom from the tyranny of user interfaces. Fully embracing peer computing requires that users be allowed to determine the look, feel, and modality of their interfaces. In other words, interfaces are but another form of (specialized) peer, interacting with other service and content peers in a manner and form shared by peers throughout the infosphere. When the interface is relegated to its proper status as "just another peer" then it too enjoys all of the powers, and bears all of the responsibilities, of peerage. The user in turn, enjoys the power of choice, and is now free to select the "interface peer" that provides just the form of interaction that is desired on the device selected by the user.

Stewardship

Stewardship encourages peers to seek assistance from other peers in the network. By stewardship we mean the power to proxy content and service on behalf of other more specialized or less capable peers. In the extreme case a small peer, for example a lowly thermostat, may directly support the narrowest of content (reporting the present temperature) and services (resetting the thermostat) and for all content or service requests proxy those requests to a larger and more capable peer. Stewardship relieves peers of the burden of providing all services to all peers, thereby permitting large classes of peers to specialize and simplify. Thus stewardship greatly enriches the ecology and numbers of peers by encouraging both extreme specialization and large numbers of inexpensive devices.

Summary

The powers of peer computing—placement, security, sharing, governance, access, control, and stewardship—are sweeping and demand a principled architectural foundation. In the following two sections we outline the design choices that shape the Magi architecture and illustrate how the seven powers are supported.

3. Powers as Architecture: The Magi Approach

Peer computing compels and challenges us. A peer-to-peer infrastructure that simultaneously grants all of the powers discussed in Section 2 is a formidable undertaking. We believe that four, key, judicious architectural decisions enable Magi to uphold those powers. This section reveals these four key architectural decisions. Section 4 then presents Magi's architecture by describing its major components and how they fit together to provide a general-purpose infrastructure for peer computing.

Decision #1: Build atop the Web's Infrastructure

The World Wide Web (*Web* for short) is a superb example of an Internet-scale network-based application, of which peer applications are a variant. The Web's architecture has

achieved remarkable utility, scalability, extensibility, performance, and adoption—many of the same qualities that now challenge peer technology. These similarities strongly suggest the use of the Web's architecture as a foundation for peer computing.

The Web's architecture is embodied in four key standards: HTTP, WebDAV, URIs, and MIME. These standards represent a careful balancing of concerns for network performance, scalability, and extension [FT2000]. Magi embraces these standards as its architectural cornerstones. Specifically, Magi uses:

- Hypertext Transfer Protocol (HTTP) version 1.1 [RFC 2616] as its default and dominant communication protocol among peers;

- Web Distributed Authoring and Versioning (WebDAV) [RFC 2518] to support collaboration and annotation;

- Uniform Resource Identifiers and Locators (URIs and URLs, respectively) [RFC 2396 and 1738] to name and locate all network resources; and,

- Multipurpose Internet Mail Extensions (MIME) [RFC 2045] to identify and encapsulate the representation of resources, whether HTML, WML, XML, or otherwise.

Basing Magi's architecture upon these four widely adopted Web standards and extending them to support true peer-to-peer communication brings us a long way towards granting the powers of peer computing. Below, we describe how each standard contributes to Magi's goals.

Hypertext Transfer Protocol (HTTP). The technical characteristics of HTTP that make it so successful as the core protocol of the Web are the same characteristics that make it an exceptional choice as the basis for communication between peers. Three qualities make HTTP a superior choice for peer computing:

1. HTTP is an open, documented protocol. Open protocols free peers to use any programming language, operating system, or programming model. In fact, the only restriction on peers is that they adhere to the semantics and the on-the-wire syntax of the protocol. With closed, proprietary protocols, in contrast, peers *must* use the library module provided with the peer infrastructure, which confines peers to the available implementations of the library. This not only limits the choice of operating system and programming language, but, more importantly, a library module presents the same interface to all peers, preventing a peer in a special context from optimizing its use of the network.

2. HTTP has standard semantics, i.e., the *meaning* of a client's request (whether *GET, POST,* or *PUT*) is well known. This allows network intermediaries, such as proxies, firewalls, and caches, to precisely interpret each request and respond accordingly. Many other open protocols, such as Remote Procedure Calls (RPC),

Sun's Remote Method Invocation (RMI), CORBA's Internet Inter-ORB Protocol (IIOP), and Microsoft's Component Object Model (COM), lack this essential capability. With these protocols, intermediaries can only interpret the *syntax* of a request, *not its semantics*; only the ultimate target of a request can interpret its meaning. Hence, they are incapable of providing value-added services such as caching and proxying, which are essential to achieving Internet-scale.

3. HTTP has been widely adopted. The adoption, ubiquity, and mindshare attained by HTTP strengthens its use as a foundation for peer computing. Ubiquity aids the goal of effective dissemination and installation of peer technology on a wide variety of devices. Mindshare enables developers to quickly grasp the central issues as well as to effectively integrate their own proprietary technologies. Corporate IT departments *already* possess the expertise necessary to roll out HTTP-based peer technology.

In today's Web, each network node acts as either a client or a server. Hence, all communication is *one-way*: from a Web client (typically a browser) that initiates a connection to a Web server, issues a request, and receives a response. Magi, in contrast, operates each node as client *and* server, allowing either party to initiate a connection. In effect, Magi "runs the Web in both directions," transforming it into the *two-way* Web.

The semantics of HTTP, and specifically its support for intermediaries such as proxies and security, helps us grant the powers of access, placement, security, governance, sharing, and stewardship.

Web Distributed Authoring and Versioning (WebDAV). A relatively recent extension to HTTP, WebDAV provides facilities for remote resource manipulation, locking, versioning, and annotation. By changing the Web into a *writeable* medium, WebDAV helps grant the power of sharing. Major software vendors, including Microsoft with its Office 2000 product line, have implemented WebDAV in their products. Since remote resource manipulation (such as file sharing, swapping, and printing) is a popular aspect of many peer-to-peer applications, it makes good sense to take advantage of this open, standardized, and increasingly popular protocol. In addition to its immediate benefits, our adoption of WebDAV paves the way for integrating future extensions to the WebDAV standard. One such extension aims to incorporate versioning and configuration management of Web resources. Such capabilities are needed in many peer computing applications.

Uniform Resource Identifiers and Locators (URIs and URLs). Magi uses URLs to refer to *all* network resources. A resource may be a shared artifact, such as an MP3 file on a peer's workstation, or a shared service, such as a CD burning service provided by another peer. This offers substantial flexibility and interoperability and allows Magi to reference *any* resource on the World Wide Web, whether it be tomorrow's weather forecast or a complex database query. The converse is also true: URLs embedded in a Web page can just as easily refer to Magi resources as those on the Web. This is a powerful capability, as it enables Magi to manipulate the Web's resources and to

seamlessly interoperate with the systems and services offered therein. By providing the capability to name and locate resources on a peer network, URLs help grant the powers of access, placement, and control.

It is important to realize that a URL can represent a *service* just as easily as *content*. Consider, for example, a peer that offers a Magi service for accessing a shared read/write CD device at the URL http://john-pc.endtech.com/cd-burner. Using Magi, friends of this peer can remotely operate this device as if it were directly connected to their PC.

Multipurpose Internet Mail Extensions (MIME). Magi uses MIME in much the same way as HTTP—as a means of identifying and encapsulating the representation of resources. This choice lets Magi cope with the wide variety of file formats which permeate the Web, spanning HTML, the WAP Markup Language (WML), XML, and proprietary formats such as Microsoft Word. Magi peers declare their preferences (and support) for particular resource formats whenever issuing requests, thereby enabling a peer to respond using a representation that is acceptable to the requesting peer.

Decision # 2: Exploit an Asynchronous, Event-Based, Component Architecture

Magi's architecture is component- and event-based. Component-based architectures offer benefits for customization, reuse, adaptation, and extension. These benefits are so strong and obvious that component-based systems are "conventional wisdom". Magi uses component architectures in two ways. First, Magi's structural elements are composable, allowing the easy configuration of customized solutions for particular markets. Second, several of Magi's components can be used in other applications, independent of Magi. Event-based architectures have emerged over the past decade as the preferred choice for applications in which independent entities interact at unpredictable times, based upon independent actions local to the peers. Event-based architectures present many benefits for wide-area coordination, timely notification, independent evolution, and service extension.

Magi's flexible architecture lets independent third-party developers extend or contract the infrastructure to suit their particular needs. This could include, for example, removing non-essential components in order to port Magi to an embedded device such as a cell phone, and replacing Magi's security model with a one that reflects the capabilities of cell phones. As a consequence, third parties have tremendous flexibility in adapting the powers provided by Magi's infrastructure.

Decision # 3: Provide a Platform for Others

A successful infrastructure must not only support the set of applications envisioned by its infrastructure developers, but also enable other developers to integrate their applications and to port the infrastructure to new platforms. The requirement on the developers of the infrastructure is thus to avoid making design choices which limit the types of applications which can be supported. We explicitly provide a platform for others by exposing interfaces, using open standards, and making many components open source. Here are certain choices that we have made:

No User Interface constraint. Magi explicitly separates user interface decisions and application software from the infrastructure. Thus, Magi avoids being platform dependent, since no UI library is wired in. Equally important, however, Magi does not require interactions with it to occur through a "frame" or "portal". Magi's open architecture promotes the integration of custom applications with exactly and only those Magi services required.

Open standards. As discussed earlier, Magi makes extensive use of the HTTP, URL, WebDAV, and MIME open standards and eschews the use of proprietary middleware. This positions Magi to take advantage of future enhancements to those standards. Magi is also contributing to the development of certain other relevant open standards, including the Simple Workflow Access Protocol (SWAP) [SWAP 1998].

Open source components. As the Apache Web server demonstrates, developing infrastructure components in an open source fashion can have enormous benefits for the user community. Magi has adopted this approach for its key infrastructure elements, ensuring that its evolution can be overseen by a community of users, with its long-term success not dependent on a single company or organization. Magi's dependence on both open standards and open source reduces risks for adopters of this common P2P platform.

Platform-independent architecture with multiple implementations. Whether for convenience or simply out of ignorance, many otherwise good technologies are of limited impact because they rest, unnecessarily, on proprietary substructure, closed "standards", or not-easily-portable languages. Worse still are technologies whose *architecture* is intrinsically platform-specific. We have sought to avoid this through the use of a platform-independent architecture and the open standards described above. This architecture can then be realized in multiple implementations, specialized as necessary for particular platforms. Additionally we are exploring the use of emerging standards such as Simple Object Access Protocol (SOAP) [SOAP 2000], which can be used for remote service invocation atop other protocols such as HTTP, RPC, and Simple Mail Transport Protocol (SMTP).

While platform independence may come at a cost in development time or in performance, it is the right approach for an *enabling* architecture. Platform-specific optimizations can then be created as necessary, such as for embedded devices. Starting with a platform-specific approach, in our view, seldom leads to an architecture that has credible potential for moving to other software or hardware platforms, let alone working across heterogeneous platforms.

Decision # 4: Promote "the Independence of Peers"

In most, if not all (by definition), peering networks, resources originate at the fringes of the network. That is, an individual peer is the original source for content or service. The philosophy of Magi is that P2P application designers, and *not the infrastructure*, should decide whether that resource is duplicated among some select set of other peers, or is replicated throughout the network. These are design choices and what is most

appropriate for one situation may be wholly inappropriate for another. Application designers – not Magi's designers – thus determine the overhead one pays for using Magi.

4. Architectural Overview of Magi

The design principles discussed in the last section are reflected in the Magi architecture. Each Magi peer provides a basic set of core services. These services provide the foundation for extension modules that tailor each peer based on the services it will provide and the capabilities of the platform on which it is installed. Using the HTTP protocol, peers link together in networks to share information and services.

We will first discuss the internal architecture of individual peers and what services they provide. We will then examine the network architecture of Magi, how the peers connect, and approaches to overcoming the obstacles presented by some network configurations. After the discussion of Magi networks, security and authorization in peer-to-peer interactions is given a more thorough coverage to explain how peers can identify themselves to each other and protect their communication. Finally, we will look at some of the typical configurations of Magi and the specialization of peers based on their capabilities and roles within the network.

Architecture of a Canonical Peer

Each Magi peer provides a simple core set of services that handles interaction with the network, manages information about the location, status, and access privileges of "buddies," and supports the dynamic extension of Magi. Extension modules may depend on these capabilities being available in any Magi peer into which the module is loaded. Figure 1 (see next page) shows the architecture of an individual peer. The core services form the main part of the diagram, with additional modules shown plugged-in at the top. By adopting this simple base architecture and set of services, the Magi system provides a flexible foundation that can be configured with collections of components.

The principal functions of Magi are layered one on top of another, providing APIs that allow the loaded modules to access services at each level. The extension modules enrich this functionality with added capabilities. We will examine each of the basic Magi capabilities in turn, starting from the lowest level, working upwards, and then discussing how the extension modules may use them.

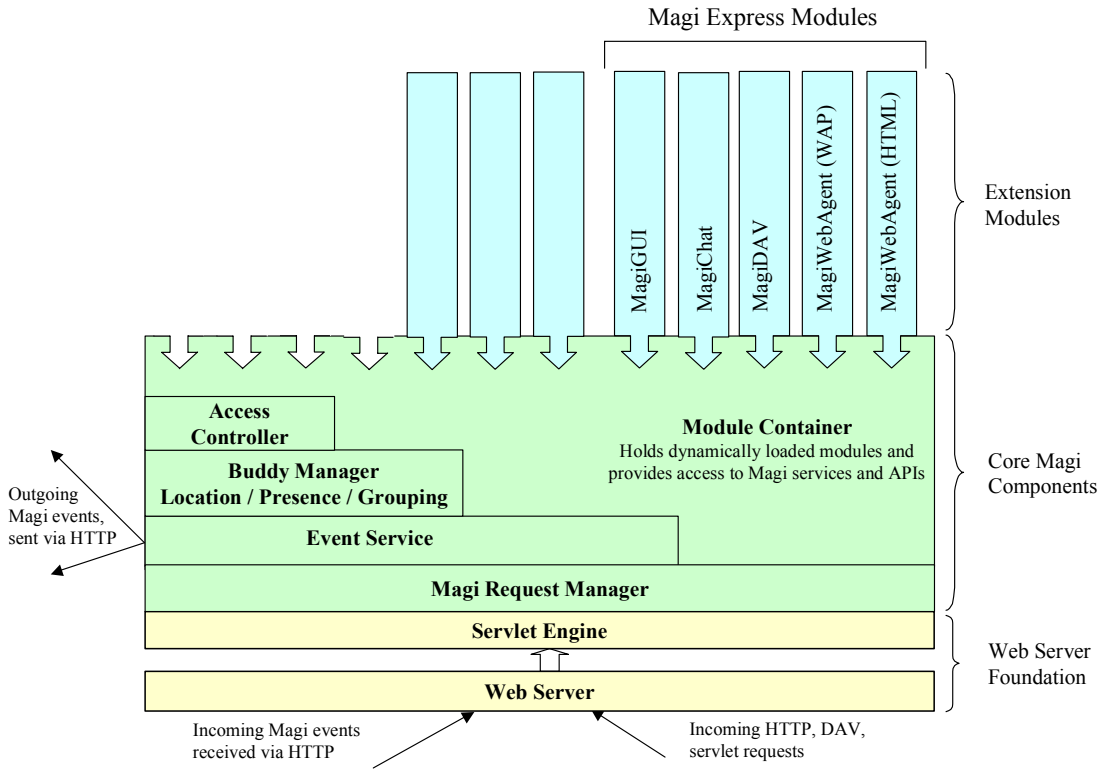

Figure 1. Internal architecture of a Magi peer

Building on top of the network architecture

Magi is itself based on top of a traditional Web server and servlet engine which trigger Magi functionality based on incoming HTTP requests. This basic network infrastructure may vary by platform and provides the binding to the Web services available on that platform. These two components (sometimes combined in a single application) provide simple parsing of the HTTP requests, passing the content on to the platform independent Magi request manager to be processed.

Magi Request Manager

The request manager receives the parsed HTTP request from the underlying server and, based on the content, invokes services within the peer to handle the request. Loaded modules may register for requests based on the URL. Once the request manager passes a request to a module, the module can provide the response (if any) to be returned, or pass the request along to another module for additional processing. This approach, modeled on Java servlets, provides a simple way to integrate traditional mechanisms for custom Web content into the Magi peer.

Event Service

Magi events, passed over HTTP, carry the bulk of communication between Magi peers. The event service receives the events (packaged in an HTTP request from the request manager) and invokes the services that have registered interest in those events. In

addition, the event service provides the mechanisms to allow components to send events to other peers. To increase the efficiency and utility of the event processing, the event service also provides other capabilities. For example, it maintains priority event queues where appropriate and manages network transactions for efficient communication of events. Together with other Magi peers, it attempts to detect and accommodate users who, by using firewalls or proxies to connect to the network, limit their communication options (we will examine these issues later in this section).

Events are described in simple XML constructs and have a type that identifies the purpose of the event. Modules and core Magi components register for events of interest based on this type. Also included within each event is information about its creation: the address of the creator, a local timestamp, a count indicating the local ordering of events since that Magi peer was started, and a globally unique identifier string (based, for example, on the other creation properties). If the event is in response to another (e.g. a request for information), it may also include the identifier of the event to which it is responding. The content, or payload, is included in the event as arbitrary text or structured XML. The type and content of events is extensible, so modules may create their own event types to communicate with related modules, including those on other Magi peers or other services implementing the Magi event service.

Buddy Manager

Buddy names form the basis for representing the identity of individual Magi users and controlling the access relationships between Magi peers. "Buddies" are familiar to many users from other applications such as instant-messaging tools, and provide an appealing and flexible mechanism for defining such things as who may access resources and about whom the user would like information.

In Magi, users are identified by a name, and their individual Magi peers are identified by that name paired with a location. By convention, this pairing is shown using the pair separated by an apostrophe, or tick-mark (’). For example, a user, *Greg*, might have Magi installed on several devices, his office desktop machine, his laptop, and his palm device. These installations may be referred to as *Greg’Office*, *Greg’Laptop*, and *Greg’Palm* respectively.

Each device has a location on the network that might be consistent (e.g. a desktop computer that is always connected at the office,) or might change frequently (e.g. a laptop that is carried to meetings, dials in from a hotel room, etc.) Magi manages this issue by tracking users’ locations and making them available if modules require them. This is done in cooperation with directory services (provided by other peers) that help locate connected users. In addition to tracking the last known location of a user, by communicating with other Magi peers, the buddy manager tracks presence information about other users. That is, it provides information about when they are on-line or off-line and when these transitions occur. This permits operations, such as transferring files, to be triggered at appropriate times.

We have seen that Greg has a way of easily identifying all the devices that belong to him and can use that to set access controls (we will discuss this a bit more in the next section). In addition to this universal mechanism for identifying people and their devices, individual users may associate their buddies together in groups. For example, if Greg is working on a proposal document with Michael, Joyce, and Joe, he can associate them together in a group, called "proposal." He can then configure operations, such as setting access controls, based on this group identifier.

Access Controller

The access controller uses the notions of buddies, devices, and groups to control access to resources on a Magi peer. Information provided in the incoming event or HTTP request serves to identify users as being permitted defined levels of access. This might be, for example, access to public resources only, read only access to shared resources as a member of a group, or read/write access as the owner (e.g. if Greg is accessing his desktop machine from his laptop).

The user may permit or deny access to services. Further, once another user requesting a service is permitted access by these primary controls, the extension modules may determine, based on information from the controller, a more refined level of access control to their services. Access control interplays with a number of issues that impact the network architecture. We will discuss these topics more thoroughly in a later section.

Module Container

Magi's core functionality is extended by dynamically loaded extension modules that provide new functionality for specific applications. The modules may be incorporated dynamically to allow custom configuration of Magi peers, in most cases even while the peer is running.

Once loaded, these extension modules are provided access to the core APIs described earlier in this section. They may be mapped to handle incoming HTTP requests, register for events, send events to other Magi peers, and access information about buddies and access controls. Modules can communicate in a controlled fashion with other modules within the same peer using a shared context through which they can exchange data or publish their interfaces. In addition, they may send events internally through the event service.

A P2P Network

Building on the architecture of the individual peer, in this section, we will describe the networked architecture of a community of Magi peers. We will first look at the simple case of Magi peers on a public network and then look at special cases where peer communication is restricted by some common networking situations.

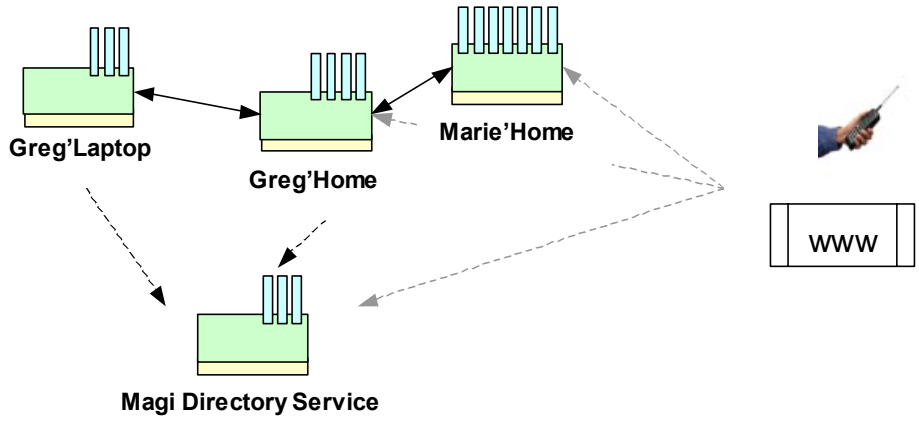

Figure 2. Magi Network Architecture, simple case

Basic Magi network architecture

Discovery: Who is out there and what can they do?

Figure 2 shows the basic Magi network architecture. The dashed gray lines indicate relatively short-term interactions; the solid lines ongoing frequent interactions. When the user connects to the network, the Magi peer first attempts to connect to a directory server (a specialized Magi peer) to locate each of the user's buddies and to register the location of the user that is now coming on-line. If a directory service is not available or if information is not available for a particular buddy, the peer first falls back on cached knowledge about the locations of buddies, derived from either previous contact with a directory service, previous contact with those buddies, or information entered by the user. If all these efforts fail to locate other Magi users, then a broadcast mechanism may be used to find other peers on the same local network (thereby supporting the construction of ad-hoc interconnection of peers). Broadcast supports such common scenarios as isolated private networks within an organization or one-time gatherings of devices connected to a single detached network hub.

Once peers are located, they can provide information about their capabilities and the services they provide. This supports the dynamic specialization of peers (which we will describe a bit more in a later section), and appropriate use of peers based on their capabilities avoiding, for example, an attempt to download an extremely large file over a slow modem connection.

Presence: Opening the lines of communication

Once it determines their locations, the peer then attempts to establish contact with all of the buddies that it has found. Established contacts may be reflected in a user interface, providing information to the user about what other users are on-line. This information is also available to extension modules through the buddy management APIs described earlier.

Once online, the peer can be accessed through a number of interfaces provided by the modules. Interacting through the functionality provided by the MagiWebAgents

(described further on), each peer might also be accessed using a WAP phone or a Web browser, once again tailored and controlled based on their capabilities and to whom they permit access.

Firewalls

For the purposes of our discussion, a firewall represents a connection to the network that only permits outgoing requests. This allows users to browse Web pages on external servers, for example, but does not allow connections back to the browsing machine. Some firewalls also limit the types of outgoing connections. Magi requires just that the firewall permit standard outgoing HTTP connections (perhaps through an HTTP proxy) to support its event communication. Figure 3 illustrates this arrangement. The arrows illustrate the direction in which communication is initiated (i.e. from the firewall outward).

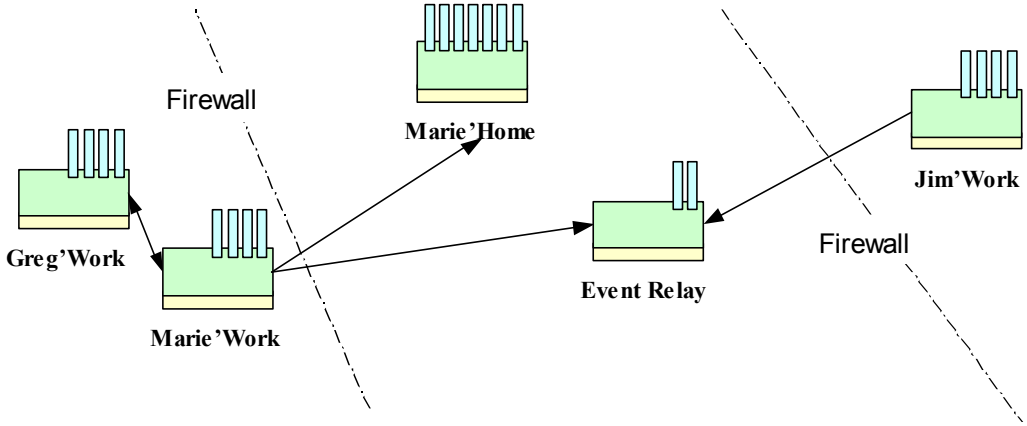

Figure 3. Using Magi with firewalls

There are several variations on this scenario. When both peers are behind the same firewall, direct communication in both directions is possible and users can browse each other's Magi Web servers using Web browsers subject to the normal access controls. This is illustrated by the relationship between *Greg'Work* and *Marie'Work* in Figure 3.
In the case where two peers wish to communicate and only one is behind a firewall, the publicly accessible peer can cache events destined for the peer behind the firewall. At regular intervals, the peer behind the firewall contacts its buddies and requests any cached events. This is the arrangement between *Marie'Home* and *Marie'Work* in our diagram.

If both peers are behind separate firewalls, communication is further restricted. Since neither user may be reached from the outside, a relay must be set up to store events from one peer and forward them to the other peer when it makes a request. This approach is illustrated by *Marie'Work* and *Jim'Work*. The relay represents another specialization of a Magi peer to support the Magi network.

Respecting the firewall prohibitions, it is not possible to browse Magi peers from outside the firewall using a Web browser or a WAP phones. Similarly, it is not possible to send files to Magi peers behind a firewall. Magi users behind the firewall may, of course, browse and retrieve files from other servers (including outside Magi peers).

Network Address Translators (NATs)

Network address translators (commonly referred to as NATs) provide a mechanism for multiple machines to use a single public Internet address. A NAT creates a private network for the local machines and handles the translation of addresses between a machine on the private network and the Internet at large. Figure 4 shows this arrangement.

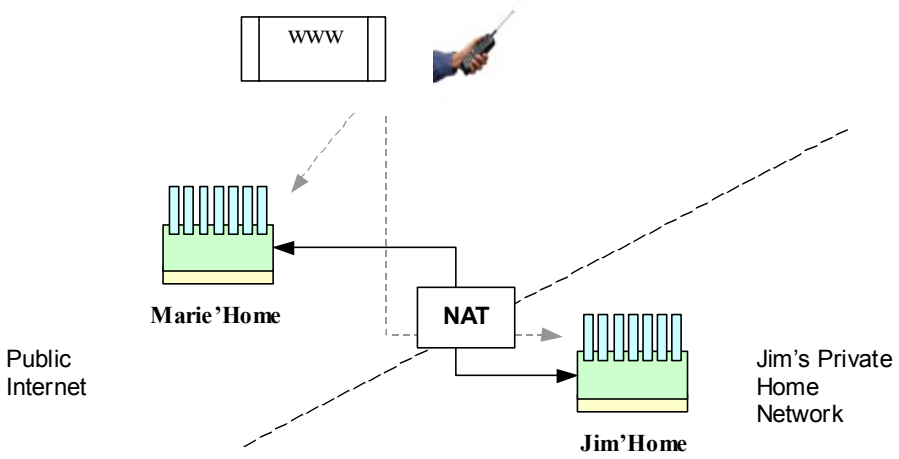

Figure 4. Using Magi with a NAT

Obtaining all the functions of Magi in this configuration depends on the ability of the address translator to map a unique identifier either automatically or through configuration options. This can be done based on either a specific port or specific URL assigned by the NAT to the machine on which the Magi peer is running.

If the NAT has this capability, all the Magi peer's functionality can be made available, including bi-directional communication with other peers, and the ability to access Magi through a Web browser or WAP-phone. If the NAT does not provide this functionality, the Magi peer is effectively hidden behind the NAT. In this case, the configuration reverts to the firewall configuration described above.

Interacting with resource constrained devices

Some peers within a Magi network may be restricted in their capabilities, making it impractical or impossible for them to offer a wide range of services on their own. In this case, they may instead, elect other, more capable, devices to provide services on their behalf. This notion of "proxied services" is a fundamental element of Magi as it allows small devices to present themselves as full featured by proxying expensive services to a larger, more capable host.

Consider, for example, a tiny Magi server embedded within a thermostat. Due to its limited memory, the device can elect a more capable and trusted Magi server to provide basic services, such as authentication and authorization, on its behalf. The thermostat redirects requests to this proxy, which processes them in place of the thermostat.

Security

Magi adopts much of the Web's security mechanisms and adapts them for peer computing. As discussed previously, Magi peers communicate using HTTP. Each HTTP request, whether it attempts to download an MP3 file or control a read-writable CD device, uses a URL to designate the resource of interest. For example, a peer could share one if its MP3 files using the URL http://peter.endtech.com/public/song.mp3 and share its CD device using the URL http://peter.endtech.com/devices/cd. Peers use these URLs to identify the resource of interest when making HTTP requests.

Each and every HTTP request received by a Magi peer passes through three security checkpoints:

1. *Authentication,* which determines the identity of the sender,

2. *Authorization,* which determines whether or not the sender has the necessary permissions to make the request, and

3. *Encryption,* which cryptographically scrambles the communication between the two peers so that no one else can decipher "what" is being said.

Each of these checkpoints is managed by the *Access Controller* component, one of Magi's core components (see Figure 1). We describe each of these checkpoints in the following three sections.

Authentication

Magi comes pre-configured with support for four standard authentication policies common to the Web and the Internet: *Basic, MD5, Token,* and *SSL certificates*.

- **Basic**. The *Basic* authentication policy is defined in the HTTP 1.1 standard [RFC 2616]. With this policy, every request is accompanied by a username and password. *Basic* authentication has the advantage that all Web browsers implement it. Unfortunately, it provides a rather weak form of authentication, making it suitable only in situations where no other mechanism can be used, such as in an ad-hoc peer network with the most basic of HTTP clients.

- **MD5**. The MD5 authentication policy is defined as a part of the WebDAV standard [RFC 2518]. It provides a substantially more secure challenge/response mechanism than the *Basic* policy, making it significantly more difficult for a third party to intercept and decode a request's username and password.

- **Tokens**. The most commonly used authentication mechanism in Magi is a Kerberos derived security system called *Token Authentication*. Like Kerberos, *Token Authentication* is a centrally managed authentication mechanism that authenticates buddies using Magi's MDNS. After Magi peers authenticate with the MDNS, communication between two Magi peers who wish to communicate directly with one another must obtain a session key from the MDNS. The session key is thus shared

between the Magi peers and the MDNS for the duration of the session (see Figure 5).

Like Kerberos, the biggest advantage to this policy is that access is centrally maintained. This is especially attractive for environments where access and resource control change frequently and must be enforced instantaneously. However, this scheme also shares a major disadvantage of Kerberos. Centrally maintained access mechanisms implies that systems will have difficultly scaling beyond workgroups and that dependencies on the central authority will inhibit some of the natural advantages of peer computing.

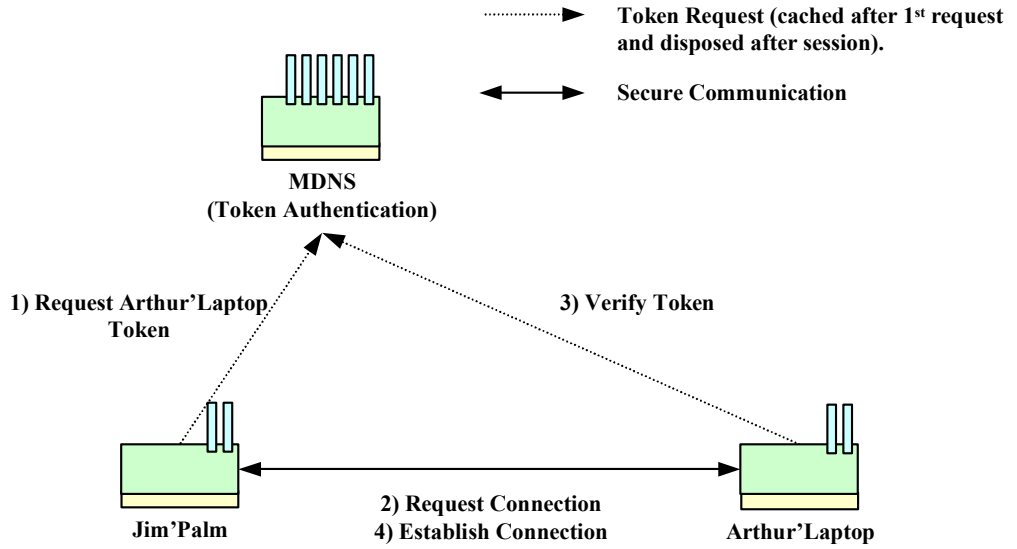

Figure 5. Token Authentication among two Magi peers with the aid of the MDNS.

- **SSL certificates.** SSL certificates are defined as a part of the HTTPS standard [RFC 2818]. With this policy, Magi's Dynamic Name Service (MDNS) acts as a trusted third-party and certificate authority, issuing a unique, tamperproof certificate to each Magi peer. A Magi peer sends its unique certificate when it issues a request to another Magi peer. The Magi peer receiving the request verifies the authenticity of the certificate, and hence the identity of the peer, using the MDNS service (see Figure 6). For efficiency, Magi peers cache certificates in a private key ring to avoid making repeated requests to the MDNS service. A private key ring also has the advantage that it can be consulted when the MDNS is unreachable, such as when two peers are using a private network without access to the Internet.

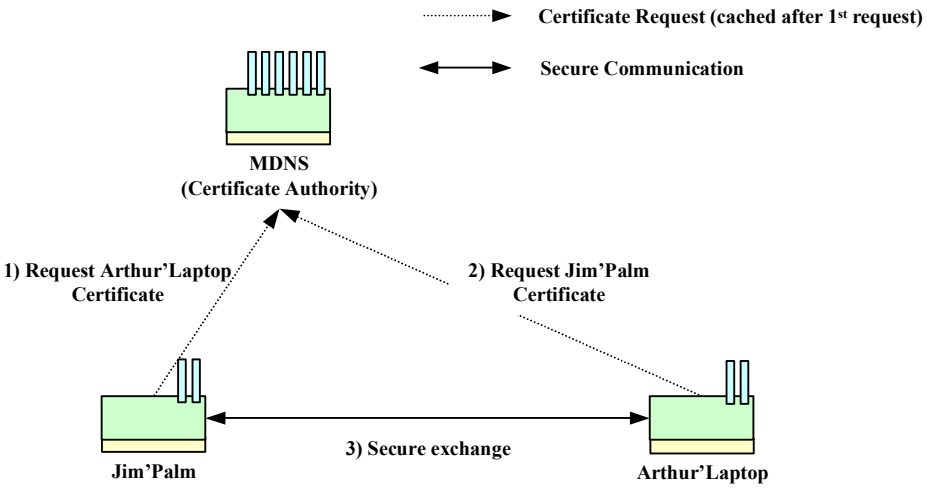

Figure 6. Authentication using SSL Certificates

Magi peers prefer *SSL certificates* to all other authentication policies. Both the *Basic* and *MD-5* policies are fast but prone to password discovery and therefore not recommended for use where secure transactions are required. Basic and MD5 authentication are the only mechanisms that allow non-buddies to connect to a Magi peer.

Magi's *Access Controller* supports an extensible set of authentication policies. Magi's pre-installed policies are implemented as plug-ins to the *Access Controller* component, so third parties can augment or replace these policies at will. The *Access Controller* loads and configures all installed authentication plug-ins when Magi is started.

Authorization

Access to each and every Magi resource can be controlled. In Magi, the URL used to refer to a resource serves as the namespace used to govern its access[1]. Magi's *Access XML* component maps these namespaces to a corresponding set of permissions using a hierarchical tree structure represented in XML. Every Magi URL is represented as a node in this tree, and each node is annotated with a description of the resource, its attributes, and the authorization policy used to govern access to it. All the components within a Magi peer can query the Access XML component.

After a Magi peer determines *who* sent it a request (the role of the authentication checkpoint), it queries the *Access XML* component to determine whether the sender has the necessary permissions to access the given resource. The *Access XML* component considers the identity of the sender, the URL of the resource, and the HTTP method requested before returning a response. The Access XML component consults each of the loaded security modules in succession. Permission is granted if any one of the loaded security modules authorizes the request, otherwise permission is denied.

[1] For more information on namespaces, consult the URL and WebDAV standards.

If permission is granted, the request is processed and handled by the particular Magi module responsible for the namespace identified by the URL.

Encryption

Magi uses the Secure Sockets Layer (SSL) for encryption. SSL is the standard for secure Internet communication, trusted by financial and government institutions worldwide, and supported by most popular Internet Web browsers.

Enhancing and Specializing Magi Peers

While the Magi core provides a basic set of functionality on which to build peer-to-peer applications, Magi's real potential is realized in the extension modules that enhance its functionality and provide the mechanism to deploy services to the places they are the most appropriate and useful.

Not every peer is created equal. A palm-top with a cell-modem will not be providing the same services as a desktop machine on a broadband network connection. Further, these constraints may change over time, even for an individual peer. A laptop in an office docking station will be ready to do more than that same laptop dialing in from a hotel room. In this section, we will examine this from two angles. First, we will describe the starting point for Magi on a desktop system. Then we will discuss specialization of capable Magi peers to provide services to networks of other peers.

Magi Express

As shown in Figure 1**,** the principal plug-in modules that are incorporated in Magi Express are grouped together. These implement a base set of services that extend the core Magi services to support WAP, for Magi access to WAP phones, and WebDAV for team collaboration and authoring over the Web. WebDAV in particular is a key element for hosting complex peer-to-peer applications on the Magi infrastructure.

Web agents

MagiWebAgents provide the access to Web pages and directories that one would expect from a personal Web server, using customizable Magi presentation themes. Versions of this module are included for both HTML browsers and WAP phones, and are extensible to support other devices or mark-up languages.

For the owner, the agents provide additional capabilities. In addition to browsing resources on the system, they allow sending (via Magi), e-mailing, faxing, and printing documents. Further, they provide control over much of the configuration of the peer (all subject, of course, to authentication by the access controller). The owner may even access the instant messaging services of the peer if that component is installed.

Distributed Authoring and Versioning

The MagiDAV extension module implements the WebDAV protocol extensions to HTTP, supporting distributed authoring and versioning of resources (explained earlier in Section 3). The request manager forwards DAV-specific requests to this module for handling. Distributed authoring and versioning is a critical element for sharing resources

and collaborating over the Web, forming the foundation of many applications to support Web-based collaboration.

Informal Communication

Magi Express supports two forms of informal communication. Short messages provide persistent notes sent from one peer to another. If the user is not currently on-line, the message is held until the destination next becomes available. In addition, peers can host a chat channel where multiple users can join and participate in discussion. In this case, the conversation is ephemeral and is not saved.

Independent user interface

The Magi core itself provides no user interface; the design of Magi deliberately decouples the UI to provide the greatest flexibility in interface design. MagiGUI represents extensions necessary for the graphical user interface (GUI). The user interface, may be, for example, a platform-independent Java user interface or a interface customized for the platform on which Magi is being run. For Magi Express, the GUI provides access to both the core features and other Magi Express modules. It also allows users to control their on-line presence, create and manage buddy lists, see who is online, exchange files and messages, and join chat channels.

Customized Magi peers may run with no user interaction and consequently offer their services sans GUI. Alternatively, the UI may, in fact, be a peer on its own, operating independently and communicating with the principal Magi peer (or several Magi peers) through the event service.

Specializing peers to support the network

The services a peer provides may be specialized for the networks or groups in which it participates. While Magi provides a peer-to-peer infrastructure, it does not dictate the model of distribution for the services provided by the peers. Applications built on top of this infrastructure will exhibit varying degrees of decentralization based on the application's needs. Specialization of individual peers allows them to provide services to support the network of other Magi peers. Reasons to specialize peers to perform these services include:

- Capability of the host machine (performance, access to resources, hardware capacity, network topology),
- Trust (verification of identity, provision of critical information, roles),
- Stable point of contact (it is important that a peer be able to find the service)
- Cost (the service is expensive to replicate broadly).

We briefly examine three of the services offered by specialized Magi peers to their compatriots. It is important to note that none of these represents a globally centralized resource, that is, they may be replicated, and that none are essential to operating a Magi network in most circumstances.

Directory Service

Directory service peers provide location and presence information about Magi peers. As mobile users change their network locations, they can inform directory services of these changes so that their buddies may easily locate them. Typically, a peer will register with a limited and relatively consistent set of directory services in order to minimize the cost of locating it. Individual peers may cache the directory information to avoid the expense of a directory lookup or as a backup in case of directory failure or network outage.

Certificate Authority

Public certificates are a mechanism to verify the identity of users and authorize them to access resources on a peer. Once obtained, the individual peers cache these certificates until they expire. Certificates must initially come from an authoritative source. Certificate authorities—themselves verified by information in the Magi installation—provide the initial seed of trust. In the absence of such an authority, certificates must be transmitted by some verifiable "out-of-band" communication. Interaction with certificate authorities is relatively infrequent, however, and not necessary for most peer-to-peer interactions.

Cache and Searching

The tasks of broadly caching and searching resources are usually limited to designated peers. In Magi, caching improves performance and availability of resources to peers by providing access to information from a convenient source. Caching also provides the illusion of continuous connectivity by providing access to cached resources even when the origin peer is unavailable.

5. Summary

Peer computing is the third and latest evolutionary phase in the history of the Internet. It simultaneously:

- Grants extraordinary powers to individuals to organize their computing universe as they see fit
- Encourages the formation of groups of individuals, devices, and hosts on demand in support of tasks and projects
- Eliminates network barriers to the free exchange of content
- Increases network value by integrating both desktop workstations and embedded devices into the infosphere
- Expands the role of networks in daily personal and business affairs

The key to the success of the Magi peer computing infrastructure are four simple design principles:

- Build atop existing Web infrastructure by relying on open, broadly adopted Internet protocols

- Exploit an asynchronous, event-based, component architecture to build the core peer computing infrastructure
- Provide a platform for others that is easily adapted for specialized applications
- Promote the independence of peers and let application designers dictate the placement of resources

The systematic application of these principles has led to a flexible and robust peer computing infrastructure that not only empowers its users but provides a rich underpinning for peer computing applications.

6. Glossary of P2P-related Terms

Adaptability – The ability of a peer to recognize unmet needs within its community of peers and to adapt itself to meet those needs.

Agent – A software routine that waits in the background and performs an action when a specified event occurs. For example, agents could transmit a summary file on the first day of the month or monitor incoming data and alert the user when a certain transaction has arrived. Agents are also called "intelligent agents," "personal agents" and "bots."

Agent Factory – An agent that will make copies of itself. Usually these are mobile agents.

Apache Server – (A "patchy" server) A widely used public domain Web server from the Apache Group (www.apache.org). It is based on, and is a plug-in replacement for, NCSA's HTTPd server Version 1.3. The name came from a body of existing code and many "patch files."

API – An Application Programming Interface that can be called upon by other software programs and asked to perform certain functions or to provide certain information.

Authentication – The process of verifying the identity of a user that is logging onto a computer system or verifying the integrity of a transmitted message.

Client – A node in relation to an edge when the node referred to as the client is directing the information exchange. Also, initiates communications but does not accept communications initiated elsewhere

Client/Server – A two-node relationship characterized by fixed capabilities, where the client and server are confined to their roles for technical reasons.

Collaboration – To work collegially with others or together especially in an intellectual endeavor

Discovery – The act of exploring a network with the goal of mapping out networked services and limitations such as firewalls, proxies, name servers, bridges, routers, and other interconnected software or devices. Also see Resource Discovery.

Distributed Computing – The direct sharing, purchase, or exchange of resources including data or services between computers. Distributed computing is typically used to cumulatively harness idle resources such as processing or storage,

Distributed Networking – The direct sharing, purchase, or exchange of network access or bandwidth between computing devices. Distributed networking typically refers to mobile or wireless devices which either borrow access from other already connected devices or lend to one in need. See Distributed Computing.

Distribution – The act of physically or conceptually dispersing a set of objects.

DNS – Domain Name System (or Service), an Internet service that translates domain names into IP addresses.

DNS – Distributed Network Server, a specialized peer elected or configured to perform group functions within a P2P community, such as authentication.

Domain – A name that identifies one or more IP addresses.

Drag and Drop – A graphical user interface (GUI) capability that lets you perform operations by moving the icon of an object with the mouse into another window or onto another icon.

Dynamic DNS – A resolution mechanism that can map a name to any number of Internet addresses depending on the location of the device to which the name refers. See DNS.

Elegant – In information theory, a construct is elegant if no smaller or less costly construct can produce the same result in the same amount of time.

Embedded Program – A program placed into a non-volatile memory such as ROM or flash memory and to be run on a processor adjacent to such ROM.

Embedded System – A specialized computer used to control devices such as automobiles, home and office appliances, handheld units of all kinds as well as machines as sophisticated as space vehicles. Operating system and application functions may be combined in the same program.

Encryption – To encode data for security purposes. The process of scrambling data content so that it can be descrambled, and thereby understood by a recipient who has and uses the descrambling (encryption) key.

Event Service – An application that responds to input from the user or other application at unregulated times. It's driven by choices that the user makes (select menu, press button, etc.).

Firewall – A method for keeping a network secure by limiting incoming messages to only responses prior to outgoing messages.

Heterogeneity – The quality of being diverse, or in the case of software, the ability for two or more non-comparable peers to interact.

Homogeneity – The quality of being the same, or in the case of software, the ability for only one type of comparable peers to interact.

Host – Something that provides a hospitable environment within which agents reside. This can be a specific part of the operating system or it can be recursive and provided by another agent.

HTTP – Hyper Text Transfer Protocol, the basic communication protocol used by the World Wide Web to run over the Internet.

HTTP Event Service – A matching and notification mechanism built on top of the Web for subscribing to items of interest or publishing items that may be of interest to others.

Identity – The means by which a peer can establish identification with another; Identify may include a name as well as security and networking information.

Infrastructure – The fundamental underlying structure of a system or organization. The basic, fundamental architecture of any system (electronic, mechanical, social, political, etc.) determines how it functions and how flexible it is to meet future requirements.

IP – Internet Protocol, specifies the format of packets, also called datagrams, and the addressing scheme.

IP Address – A series of four 3-digit numbers which uniquely identify a computing device or NAT connected to the Internet.

Load Balancing – A technique for allocating work to machines, software, or people that are least busy or have the shortest waiting job queue.

Magi technology – P2P middleware based on thin server Web technology. A suite of open source and freeware toolkits for building mobile and disconnected applications using common Web technologies in a peer-to-peer manner. Magi technology makes the Web two-way and write-able.

Micro-Web browser – Browsers with small file sizes that can accommodate the low memory constraints of handheld devices and the low-bandwidth constraints of a wireless-handheld network.

Micro-Web server – A small implementation of an HTTP-based Web server designed to run on memory constrained platforms.

Mobile Access – Ability to access and change information from a non-fixed location—typically using wireless protocols.

Mobile Agent - An active entity that can move its residence from one host to another while maintaining a persistent identity.

Namespace – A name or group of names that are defined according to some naming convention. A flat namespace uses a single, unique name for every device. The Internet uses a hierarchical namespace that partitions the names into categories known as top level domains such as .com, .edu and .gov, etc., which are at the top of the hierarchy. Also, the span of an HTTP Event Service.

NAT - Network Address Translation, an Internet standard that enables a local-area network (LAN) to use one set of IP addresses for internal traffic and a second set of addresses for external traffic.

Native Ports – Compiling a program for use on a specific hardware or software platform using the platform's specific features or interfaces. Native ports are typically done to take advantage of speed or usability features of a specific platform at the cost of portability, i.e. being able to easily port the finished product to other platforms.

Network - A system that transmits any combination of voice, video and/or data between users. It includes the cables and all supporting hardware such as bridges, routers and switches. In wireless systems, antennas and towers are also part of the network.

Network Adapter - A printed circuit board that plugs into both the clients (personal computers or workstations) and servers and controls the exchange of data between them. The network adapter provides services at the data link level of the network, which is also known as the "access method." Also known as a Network Interface Card (NIC).

Open Architecture – A system in which the specifications are made public in order to encourage third-party vendors to develop add-on products. Much of Apple's early success was due to the Apple II's open architecture. PC hardware is open architecture.

Open Source – Free source code of a program, which is made available to the development community at large. The rationale is that a broader group of programmers will ultimately produce a more useful and more bug-free product for everyone, especially because more people will be reviewing the code. Peer review is a natural byproduct of open source projects.

Open Standards – Interoperability between hardware and software that is defined by the industry at large and not one or two vendors.

Peer – A unit (host) of communications hardware or software that is on the same protocol layer of a network as another.

Peer-to-Peer – Any relationship in which multiple, autonomous hosts interact as equals. Peer-to-peer communication refers to these real or virtual connections between corresponding systems.

Peer-to-Peer Architecture – A structure for interoperability between computing devices in which all such devices are seen as peers.

Platform – A hardware or software architecture upon which applications can be designed, developed and deployed.

Plug Ins – An auxiliary program that works with a major software package to enhance its capability. For example, plug-ins are widely used in image editing programs such as Photoshop to add a filter for some special effect. Plug-ins are added to Web browsers to

enable them to support new types of content (audio, video, etc.). The term is widely used for software, but could also be used to refer to a plug-in module for hardware.

Presence – The ability for a peer to project information about its network status at any point in time.

Protocol - Rules governing transmitting and receiving of data.

Proxy Server - Also called a "proxy" or "application level gateway," it is an application that breaks the connection between sender and receiver. All input is forwarded out a different port, closing a straight path between two networks and preventing a hacker from obtaining internal addresses and details of a private network. An HTTP proxy is used for Web access and an SMTP proxy is used for e-mail. Proxies generally employ network address translation (NAT), which presents one organization-wide IP address to the Internet. Proxies may also cache Web pages, so that the next request can be obtained locally.

Resource Constraint – A limitation of available resources such as memory, disk space, power, bandwidth, or computations per second that may prevent software or hardware from completing its tasks.

Resource Discovery – The process of investigating and thereby learning the resource capabilities and limitations of a peer. Also see Discovery

Resource Sharing – Allows access to memory, disk space, power, bandwidth, computations per second, or data to one or more peers.

Scalability – Refers to how much a system can be expanded. The term by itself implies a positive capability. For example, "the device is known for its scalability" means that it can be made to serve a larger number of users without breaking down or requiring major changes in procedure.

Secure Channels of Access – Network channels that are safe, such as by use of encryption.

Server – A computer that processes requests for HTML and other documents that are components of Web pages

Service - an API that allows for an agent to accomplish specific types of tasks. Usually there is security associated with access to specific classes of agent services. Hosting is simply a specific type of service. Agents can offer services to local or remote agents as well.

Sharing – The ability of multiple peers to have collegial access to some resources.

SMS - Short Message Service, a text message service that enables short messages of generally no more than 140-160 characters in length to be sent and transmitted from a cell phone. SMS is supported by GSM and other mobile communications systems. Unlike

paging, short messages are stored and forwarded in SMS centers. In the GSM system, short messages ride on a separate signaling path so they are transmitted simultaneously with voice, data and fax.

TCP/IP – Transmission Control Protocol / Internet Protocol, the low level hardware/software protocol to move content around the Internet and on most corporate intranets.

Thin Server – An HTTP Web server (e.g. Magi) that has been specifically adapted for sizing and performance to function as Internet middleware between collaborating computing devices.

TLD/gTLD – Top Level Domain, or TLD, typically refers to the three letter ending of a domain name such as *.com*, *.gov*, *.mil*, *.edu*. A global TLD represents the country code and can be used in place of or in conjunction with a TLD, e.g. *.uk* = United Kingdom.

Ubiquity – Having a presence everywhere or in many places, including simultaneously

User Interface - The combination of menus, screen design, keyboard commands, command language and online help, which creates the way a user interacts with a computer.

Web - A system of Internet servers that support specially formatted documents. The documents are formatted in a language called HTML (HyperText Markup Language) that supports links to other documents, as well as graphics, audio, and video files. This means you can jump from one document to another simply by clicking on hot spots. The Web is the most popular and active subset of the Internet.

WebDAV – Web Distributed Authoring and Versioning protocol. WebDAV defines standard extensions to HTTP that allow reading, and writing of documents and data to a Web server in a safe manner.

Web Server – A computer that delivers (serves up) Web pages. Every Web server has an IP address and possibly a domain name. For example, if you enter the URL http://www.endtech.com/index.html in your browser, this sends a request to the server whose domain name is *endtech.com*. The server then fetches the page named index.html and sends it to your browser.

WAP - The Wireless Application Protocol is a secure specification that allows users to access information instantly via handheld wireless devices such as mobile phones, pagers, two-way radios, smart phones and communicators.

Wireless Markup Language (WML) – A compact HTML subset used with WAP to display Web pages on devices with limited bandwidth, screen real-estate, and computing power.

7. References

[Fielding 2000] Roy T. Fielding, "Architectural Styles and the Design of Network-based Software Architectures", Ph.D. dissertation, Information and Computer Science, University of California, Irvine, 2000.

[FT2000] Roy T. Fielding and Richard N. Taylor. "Principled design of the Modern Web Architecture". Proceedings of the 2000 International Conference on Software Engineering (ICSE 2000). Limerick, Ireland, pp. 407-416, June 2000.

[Reed 1999a] David P. Reed, "Weapon of Math Destruction", Context Magazine, Spring 1999.

[Reed 1999b] David P. Reed, "Reed's Law: That Sneaky Exponential-Beyond Metcalfe's Law to the Power of Community Building", Context Magazine, Spring 1999.

[RFC 821] J. B. Postel. "Simple Mail Transfer Protocol (SMTP)". Internet RFC 821. August 1982.

[RFC 1738] T. Berners-Lee, L. Masinter, and M. McCahill. "Uniform Resource Locators (URL)". Internet RFC 1738, Dec. 1994.

[RFC 2045] N. Freed and N. Borenstein. "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies". Internet RFC 2045, Nov. 1996.

[RFC 2396] T. Berners-Lee, R. T. Fielding, and L. Masinter. "Uniform Resource Identifiers (URI): Generic syntax". Internet RFC 2396, Aug. 1998.

[RFC 2518] Y. Goland, E. J. Whitehead, Jr., A. Faizi, S. Carter, and D. Jensen. "HTTP Extensions for Distributed Authoring — WEBDAV". Internet RFC 2518, Feb. 1999.

[RFC 2616] R. T. Fielding, J. Gettys, J. C. Mogul, H. F. Nielsen, L. Masinter, P. Leach, and T. Berners-Lee. "Hypertext Transfer Protocol—HTTP/1.1". Internet RFC 2616, June 1999.

[RFC 2818] E. Rescorla. "HTTP Over TLS". Internet RFC 2818, May 2000

[SWAP 1998] K. Swenson, "Simple Workflow Access Protocol (SWAP)", Strawman document, August, 1998. http://www.ietf.org/inter-net-drafts/draft-swenson-swap-prot-00.txt

[SOAP 2000] D. Box, D. Ehnebuske, G. Kakivaya, A. Layman, N. Mendelsohn, H. F. Nielsen, S. Thatte, D. Winer. “Simple Object Access Protocol (SOAP) 1.1”, W3C Note 08 May 2000. http://www.w3.org/TR/SOAP/

[WISEN 1998] Workshop on Internet-Scale Event Notification. July 13-14, 1998. University of California, Irvine, USA.

