Towards a real-world wide web

Tim Kindberg, John Barton

Internet and Mobile Systems Laboratory
Hewlett-Packard Laboratories
1501 Page Mill Road
Palo Alto, CA 94304, USA
timothy@hpl.hp.com

1 Introduction

As computer hardware shrinks and lightens and as communications becomes cheaper and more widely available, we have new opportunities for software systems to support *nomadic* users of *ubiquitous* or *pervasive* computing services [16, 24]. We are interested in systems that support interaction with the physical world wherever users happen to be. More and more things in the physical world, such as our cars and domestic appliances, are becoming "smart". Users need a convenient framework in which they can benefit from the smart artifacts around them. Users can also benefit from services built for non-electronic entities that they encounter in their everyday lives, such as goods on the shelves of stores, paintings in galleries and items in the office. We think the physical world and the virtual world would both be richer if they were more closely linked.

The approaching pervasiveness of web technology, new kinds of wireless networks, and cheaper portable devices provide an opportunity to construct a virtual bridge between mobile users and physical entities. At HP Labs we have been exploring this opportunity through an infrastructure to support "web presence" for people, places and things [14], and the "CoolTown" demonstrator applications that run on top of the infrastructure [4].

In this paper we discuss how we provide mobile people and devices with web-based services for "things" – the physical entities – in their environment. Things become web-present by embedding web servers in them or by hosting their web-presence within a web server. To enable services for web things we need a system that connects things and services. We will outline a system that leverages the power of the Web as a basis for nomadic computing, then highlight one important subsystem that uses DNS technology to name web things in a scalable and context dependent fashion.

Section 2 describes our use of the Web as 'just enough' middleware for nomadic computing, and describes web presence. Section 3 describes how we tackle the naming issues that arise in providing web presence for the things that mobile users encounter. Section 4 relates our work to that of others and concludes the paper.

2 The Web as middleware for nomadicity

The Web is the most promising basis for a nomadic computing infrastructure, for the following reasons:

Ubiquitous access. People use the Web at work and at home today; increasingly they will find it available when they shop or travel; eventually they will find adequate access "everywhere" they want electronic connections. The Web's accessibility supports mobility in two senses. First, resources on the web can be accessed from any device that supports the standard HTTP protocol, and this includes devices that the user carries: general-purpose devices such as personal digital assistants (PDAs) and laptops, and information "appliances" dedicated to a specific function, such as digital cameras. Equally, it is a simple matter to put HTTP *server* support in devices that nomadic users encounter, such as printers and projectors. Second, the web gives nomadic users access to resources in their home and workplace or anywhere else outside their current environment, through Internet connections. So the Web provides both a local communications standard and a standard global environment.

Just enough middleware. The diversity of the devices that mobile users carry and encounter argues against software support based on device type or even application type. It also argues against assuming that all devices will run a form of middleware that is heavy on the commitments it imposes at the application, language or resource level, such as Java or CORBA. Rather we believe that the web standards of HTTP and URLs have proven themselves adaptable to such diversity.

Local means local. We shall demonstrate how to deliver web services to mobile users without requiring a global wireless connection like a cell-phone or mobile IP. This has the advantage of minimizing how much of the infrastructure needs to be up and running in order for users to interact with local services. For example, a mobile user should be able to print a document at a nearby printer without necessarily having to contact any global services. A local web server should be enough, and often a simple peer-to-peer interaction will suffice.

2.1 Web Presence

A *point of web presence* for any physical entity (person, place, or thing) is a web resource that the system correlates with the entity. Like any web resource, the point of web presence has a URL and it is accessible using the standard HTTP protocol. The correlation is between the physical presence of the entity and the URL: the system automatically supplies the URL given the appropriate physical circumstances. For example, when a visitor Joe walks into a meeting room, he uses his PDA to pick up the URL of the room's point of web presence (called a web *portal*, in the case of a place such as a room) from an infra-red beacon on the wall. Joe browses the room's web portal on his PDA. A printer in the room has also automatically picked up the room's URL, and has registered its point of web presence as a service in the room. When Joe browses the room's web portal, he finds a link to that printer. He uses the 'WebSend' facility on his PDA to send a document to the printer: he *squirts* the URL of the document from his PDA to the printer, by infra-red communication. If Joe has the document with him, then the squirted URL refers to his device (he selects the document, and WebSend generates the URL for him). If Joe forgot to bring the document with him, he selects the document on a page he carries with him that represents his home environment. The printer gives the URL to the local rendering service registered with the room's portal. That service fetches the document, renders it and sends the resulting data to the printer. Note that Joe's resource-limited PDA is only involved in the transfer of content if that device has the only accessible copy of the document. Otherwise it acts only as a conduit and storage point for URLs.

At its simplest, a point of web presence provides a 'home' web page for an entity. For example, in one of our demonstrations a gallery provides visitors with the URL of the artifact that they are currently viewing, through a nearby infra-red beacon. A more sophisticated point of web presence provides an interactive service – a means of exchanging content with the entity, such as a printing service. Consider the printer in the meeting room. This device has two points of web presence. One is its *interior* electronic presence. This interior point of web presence allows Joe (and other devices and services) to query and manipulate the electronic state of the printer. For example, Joe can read the printer's toner level via a web page, set the desired print quality and print a document via a web page form. In some cases a web server in the printer provides this point of web presence, but if the printer does not have this capability then a web server in the place's infrastructure acts as an HTTP gateway and interacts with the printer using a printer-specific protocol.

The printer also has an *exterior* point of web presence. There is a tag on the printer's case – for example, a bar code, an RFID tag [20], an iButton [11] or another infra-red beacon. This tag provides an identifier for the printer. When Joe senses the identifier from the tag using his PDA (for example, using the bar-code scanner built into it), he obtains the URL for other pages about the printer. These contain links to the manufacturer's pages about the model of printer, and pages about this particular printer. If, for example, Joe finds that the printer is down, he can still look at the printer's external point of web presence, which includes the printer's service history. On it he finds that the fault has not been reported. So he adds a fault report. This results automatically in an email message being sent to the appropriate system administrator, containing the URL of the printer's point of web presence. The printer automatically registered its location (the meeting room) with that point of web presence when it was switched on there, so the system administrator knows where the faulty printer is.

In general, then, a single entity may have any number of points of web presence, and each of these may provide any services to the client. The 'client' may be a user with a browser, or a device or service accessing the point of web presence programatically.

2.2 Content-oriented programming

The printing scenario is one of several cases where our implementation experience shows that the HTTP and URL web standards can be straightforwardly applied to support nomadic computing. The Web allows HTTP servers and clients (on mobile devices and in their environments) to exchange any type of content, and it allows them to interpret that content in any way. In particular, content may be used to perform an operation on a device or describe its status. For example, a user can program her VCR through a web-page interface by entering the required settings in fields; a printer can report a malfunction to the local logging service by sending an XML document that describes the problem.

We are developing a web-based approach to what we call *content-oriented programming*. This approach supports nomadic computing by focusing on the content exchanged between components – which is what users are interested in – and minimising the complexity of interfaces – which otherwise potentially limit interoperability.

The distinguishing features of content-oriented programming are:

1. *minimal interfaces*: to use an interface, a client needs to know at most a name for the interface (not a procedure or method signature), and hints about the type of content to be sent or retrieved from it.
2. *by-value type checking*: programmers do not rely on language support to protect system integrity; they examine content received through their interfaces and process or reject it, on a best-effort basis
3. *content negotiation*: client and server (or peer and peer) support negotiation of the type of content that they will exchange.

UNIX pipes are a mechanism for a form of content-oriented programming that does not include negotiation: the producer writes content to a consumer, which processes (e.g. filters) it as best it can. Similarly, in some event-based systems such as Mushroom [13], events are self-describing objects. Objects that react to events have a single *notify*(*Event e*) method, which the event service calls when an event occurs. The notified object examines the event object, and reacts in whatever way it is programmed to react. Negotiation *per se* does not occur in an event system, but event producers advertise the types of event that they produce, and event consumers examine these and subscribe to whichever types of event suit their needs.

The Web already supports negotiation of content type to a limited extent, in that HTTP clients can provide a list of acceptable content types for the server to provide. Proposals such as Transparent Content Negotiation [8] exist to make negotiation more practical. We are developing the WebSend paradigm (c.f. JetSend [12]) to support negotiation for the URL-squirting paradigm in the printing example above. Simple content descriptions such as MIME types work well for many applications, for example when a user simply wants to upload an image from a camera to a web site. We expect that device manufacturers and others will define standards for content in specific domains, and some of these will be accepted and made widespread. This is the project of the Universal Plug and Play Consortium [22]. New devices will be built to conform to prevailing standards; gateways will be built to encompass legacy devices.

Web presence sets the user's expectations: they expect to be able to interact with "everything"; content-oriented programming allows loosely coupled systems to be designed to meet those expectations. The missing elements are mechanisms for connecting physical objects with their web presence, taking the context into account.

3 Web presence and naming issues

Naming is a significant issue for nomadicity, where the concept of *context* is an important one. In the mobile computing literature, context is a general abstraction of the environment, parameterised by such factors as the user's identity, location, current task, and the time of day. From the system's point of view, we interpret this as a naming context. Using a mobile device, users acquire URLs and object identifiers in their home environments and the places they visit, to obtain information and other services based around the things that they encounter. Some URLs are sensed directly from beacons and other sources in the environment. Others must be obtained by looking up a value in an appropriate naming context: by looking up services in yellow-pages directories, or by looking up identifiers found on tagged entities.

One type of naming context is provided by web portals – the web presence of places. These provide a service registration and discovery service, so that users can browse the available services, and clients can obtain information about services looked up from their attributes through a programmatic interface.

Here we concentrate on how identifiers attached to things are used to obtain their points of web presence.

3.1 Exterior web presence

An obvious way to identify an object's point of web presence is with a URL. This is very convenient for client devices – they sense the URL directly. We sometimes identify web-present places and things this way in HP's demonstration applications. But there are two limitations to identification by URLs. The first is that it is not current practice to label everyday goods with URLs[1], whereas many numeric identification schemes exist. Those numeric schemes are in widespread use, in forms that are amenable to electronic or electro-mechanical/optical sensing, such as Ethernet addresses and ISBN numbers.

The second limitation is that a URL designates one particular point of web presence, whereas an individual entity may require several exterior points of web presence, in addition to any interior one it may have. For example, an individual printer may have one point of web presence for information about its model; one for the purposes of

[1] Two-dimensional barcodes, e.g. PDF-417, can encode long character strings, and may be used for URLs.

maintaining that particular printer's service history; and another for inventory control. It should be possible for different groups of users to maintain their own point of web presence for a given object, independently of one another.

Therefore we provide a resolution process, which converts a pure identifier to a URL via a naming context. This level of indirection allows manufacturers, those who administer web-present places, and other groups of users to provide their own domain-specific services through their own naming contexts.

The resolution architecture

The building block of our resolution architecture is an *object-id resolver*, or *resolver* for short. This is a server that implements a naming context through an HTTP interface. A client takes the URL of the resolver, say *http://inventory.hp.com/webPresence,* and an identifier, say the UPC class 'A' barcode *01234567890*5. It then constructs the point of web presence for the identified entity from these data:

http://inventory.hp.com/webPresence?id=<ID><TYPE>UPCA</TYPE><N>012345678905</N></ID>.

– where the barcode value has been encoded in XML and appended to the resolver's URL. A client that read an identifier from, say, an iButton, would similarly package the identifier but with a different 'TYPE' value, so that it will be resolved unambiguously.

The resolver may implement the point of web presence directly, or it may send back a forwarding URL to the service that does (as far as the resolver knows) implement the service. The client application need not be involved in the forwarding process; the HTTP stub will take care of it transparently. This example is simplified, in that the URL constructed by the client may also include other information that encodes whether, for example, the client requires XML or HTML content. The URL may also include a token that makes it a capability for the point of web presence.

A related URL:

http://admin.hp.com/webPresence?id=<ID><TYPE>UPCA</TYPE><N>012345678905</N></ID>

uses a separate naming database at *admin.hp.com* for resolving the same identifier. Two clients using the two URLs for the same identifier will in general experience separate points of web presence.

Naming contexts may be composed, as in the Spring naming service [19]. Thus, for example, when Joe encounters a printer, his context could return a page with links to all the points of web presence supplied by other object-id resolvers known to it – the local context provided by the meeting room; the manufacturer's page for the printer; and his company's knowledge base for that particular type of printer. Joe's device may include a sophisticated id-resolution client that takes part itself in context composition, but we expect the normal case to be where the client selects a single resolver using a simple procedure, and that resolver relieves the client of any work.

Selecting a resolver

We are considering three models for selecting a resolver:

1. The user types in the URL of their preferred resolver, e.g. *http://myResolvers.com/Joe90*, in the client device. When Joe scans the printer's generic barcode (the one that identifies the model), the client on his PDA appends the identifier to form a full URL and fetches the web page for him. Joe sees whatever type of information about this type of printer his resolver is configured to provide – e.g. his own knowledge base about such printers.
2. The client uses a location-specific resolver, which it discovers automatically from the local web portal. For example, when Joe scans the printer-specific barcode he sees the host organisation's information about this particular printer.
3. The client derives the address of the resolver from the identifier itself, in order to obtain the web presence from a standard provider. For example, when Joe scans the printer's generic barcode, he sees the pages provided by whoever manufactured the printer.

With the first two options, the address of the resolver is known *a priori* or with minimal discovery effort. The system is as scalable as the web: there are as many object-id resolvers as there are groups interested to maintain and publish services based around identified entities. We are exploring both of these two options.

The third option is interesting from a systems point of view, because it requires a way to determine the address of the object-id server with little overhead. Fortunately, the hierarchic structure of barcodes and some other popular types of identifier allows us to use DNS for this purpose, without modification [7].

First, we introduce the concept of an *identification domain*. This is a DNS domain that is the root of a set of names of id-resolvers. At HP, for example, we propose an identification domain *entities.net*, using which manufacturers provide information and services based around their products.

An example resolver hostname is *0f98.a11b.ean.entities.net.* This consists of two bit-strings written in hexadecimal notation, *0f98* and *a11b*, prefixed to the identifier space *ean* (the EAN barcode standard) and the identification domain. A client knows *a priori* the identification domain *entities.net* and the identification space *ean*. It derives the strings *0f98* and *a11b* by applying a mask to the identifier in a well-known way, and expressing the result as hexadecimal strings.

Any hierarchic identification scheme such as the EAN barcode standard or Ethernet addresses incorporates a set of identification authorities (IAs), such as manufacturers. An identifier consists, in general, of three parts: the IA identifier, the IA-specific identifier of the entity, and some overhead bits such as checksums. For example, the EAN barcode standard encompasses eight decimal digits for IAs, each with five decimal digits to identify their entities.

The idea is to provide at least one object-id resolver for each IA, implementing the directory of entities that fall within that IA's control. For example, an EAN IA identifier in the US can be encoded as seven hex digits. To avoid overloading DNS servers, we add an extra layer to the DNS hierarchy at *ean.entities.net*, splitting the IA identifier into two quadruples of hex digits rather than using a single hex string, to get say *0f98.a11b*. Now the client can formulate the URL of the entity's point of web presence under this identification domain:

http://0f98.a11b.ean.entities.net/?id=……….

In this scheme, DNS servers for *ean.entities.net* will be expected to serve at most 2^{16} entries, and each server such as *a11b.ean.entities.net* will in turn serve at most 2^{16} entries. This is given as an example and is not necessarily the best choice, since the name space may not be used evenly. What is actually required is, for each identification scheme *I*, Internet-wide agreement on a mapping IA → (*minor*, *major*), which clients turn into an object-id resolver's DNS name of the form *hex*(*minor*).*hex*(*major*).*I.D*, where *D* is the identification domain.

Note that this scheme has to scale to the number of resolvers, not the number of identifiers. We expect that at worst each current domain would be an IA, and so would have an entry for an object-id resolver, just as it has an entry for a web server currently. We do not believe that this will put an unreasonable load on the DNS system. We expect that some of the manufacturers and governmental organisations who already provide the Internet infrastructure will also provide DNS servers such as *a11b.ean.entities.net*, which are needed to provide a two-level hierarchy for each identification space.

4 Summary and discussion

We have described web presence as a basis for bridging the physical world with the Web, and we have illustrated the utility of both interior (electronic) and exterior web presence for things. The notion of web presence and our demonstrations are built upon earlier work in our laboratory. This includes work on web-enabled devices [9], on embedded web server technology, and Java implementations of web servers augmented with object models, event models and service discovery [3].

We have been strongly influenced by work at other centres that are investigating how to augment physical entities with electronic services. These include Xerox PARC [23], Georgia Tech [21], the GUIDE project at Lancaster University 5], Stanford University [10], and the Satchel project at Xerox's European Research Centre [6].

The distinctness of our approach is in our use of the Web. We have argued that the web (rather than, say, JINI [1] or CORBA [18]) is the best middleware for nomadicity. It is widely deployed because of the robustness of its standards; it supports evolving standards to embrace new content; it can be implemented on devices with relatively little memory and other resources; and its user interface, the web page, enables users to perform a wide variety of tasks.

Other work on programming over the web appears in SOAP [2], WIDL [17] and WebL [15]. But we believe that those approaches miss the aspect of Web interaction that makes it so well suited for nomadicity. This is the content-oriented programming paradigm, in which devices and services provide content to one another in standard forms, and what is required is negotiation over the form in which that content will be exchanged. A host place that provides content-transformation services can accept whatever format the users have with them. By contrast, in an object-oriented programming paradigm, the values – that is, the content – passed in methods are expected to conform to the requirements of a specific target object. Such a system discourages the widespread acceptance and utilization of new types of content. The object-oriented paradigm has advantages such as automatic type-safety. But we believe that what users require can be most easily expressed in terms of the content they require, and so the framework should centre on content in preference to interface syntax.

We described how we use id-resolvers to support multiple points of web presence for the same entity. Although we discussed the resolution procedure for things, this works just as well for the web presence of people and places. In some cases, the users in a group need to maintain their own resolution contexts for entities – for example, the police and networks of garages and auto dealers can keep their own information about cars. In other cases, what is required is information from recognised sources such as manufacturers that can be obtained with little overhead by knowing the identifier alone. We explained a scheme for using the DNS to achieve this, which we are putting into practice.

Web presence for people, places and things raises many issues not covered in this paper. More information is available at our "CoolTown" web site [4].

References

[1] Arnold, K., Wollrath, A., O'Sullivan, B., Scheifler, R., and Waldo, J. *The Jini Specification (The Jini Technology Series).* Addison-Wesley Pub Co; ISBN: 0201616343.

[2] Box, D., Kakivaya, G., Layman, A., Thatte, S., and Winer, D. (1999). SOAP: Simple Object Access Protocol, *IETF draft* http://www.ietf.org/internet-drafts/draft-box-http-soap-01.txt.

[3] ChaiServer home page. http://www.chai.hp.com/chai_server.html.

[4] CoolTown web site http://www.cooltown.hp.com/.

[5] Davies, N., Cheverst, K., Mitchell, K., and Friday, A. (1999). Caches in the Air: Disseminating Information in the Guide System. *Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications* (WMCSA '99), New Orleans, Louisiana, U.S., 25-26 February 1999.

[6] Flynn, M., Pendlebury, D., Jones, C., Eldridge, M., and Lamming, M. *The Satchel System Architecture: Mobile Access to Documents and Services*. To appear in *Mobile Networks and Applications*, ACM.

[7] Foley, J. Joe Foley, MIT. Personal communication.

[8] Holtman K., and Mutz (1998). A. Transparent Content Negotiation in HTTP. RFC2295. http://www.ietf.org/rfc/rfc2295.txt?number=2295.

[9] HP device could make Web a device controller. In *Netscape World*, Dec. 1996. http://www.net-dev.com/netscapeworld/nw-12-1996/nw-12-newsbriefs2.html#HP.

[10] Huang, A.C., Ling, B., Ponnekanti, S., and Fox, A. (1999). Pervasive Computing: What Is It Good For? To appear, *Proceedings Workshop on Mobile Data Management*, in conjunction with ACM MobiCom 99.

[11] iButton home page. http://www.ibutton.com/.

[12] Jetsend home page http://jetsend.hp.com/JSHome.html.

[13] Kindberg, T. (1996). A stake in Cyberspace. *Proc. 7th. ACM SIGOPS European Workshop, Connemara*, Sept., pp. 83-88.

[14] Kindberg, T., Barton, J., Morgan, J., Becker, G., Caswell, D., Debaty, P., Gopal, G., Frid, M., Krishnan, V., Morris, H., Schettino, J., and Serra, B. (2000). People, Places, Things: Web Presence for the Real World. Tech report HPL-2000-16, Hewlett-Packard Laboratories.

[15] Kistler, T. and Marais, H. (1998). WebL, a programming language for the Web. *Computer Networks and ISDN Systems (proc. WWW7)*, April. pp. 259-270. http://gatekeeper.dec.com/pub/DEC/SRC/technical-notes/SRC-1997-029-html/.

[16] Kleinrock, L. (1997). Nomadicity: anytime, anywhere in a disconnected world, *Mobile Networks and Applications*, Vol. 1, No. 4., pp. 351-357.

[17] Merrick, P., and Allen, C. (1997). Web Interface Definition Language (WIDL). W3C Note, World Wide Web Consortium; http://www.w3.org/TR/NOTE-widl.

[18] The Object Management Group home page. http://www.omg.com/.

[19] Radia, S., Nelson, M., and Powell, M. (1993). The Spring Naming Service. Technical Report 93-16, Sun Microsystems Laboratories, Inc..

[20] RFID home page. http://www.aimglobal.org/technologies/rfid/.

[21] Salber, D., Dey, A.K., and Abowd, G.D. (1999). The Context Toolkit: Aiding the Development of Context-Enabled Applications. In *Proceedings of the 1999 Conference on Human Factors in Computing Systems (*CHI '99*),* Pittsburgh, PA, May 15-20. pp. 434-441.

[22] Universal Plug and Play Forum. http://www.upnp.org/.

[23] Want, R., Fishkin, K.P., Gujar, A., and Harrison, B.L. (1999). Bridging Physical and Virtual Worlds with Electronic Tags. In *Proceedings of the 1999 Conference on Human Factors in Computing Systems (*CHI '99*),* Pittsburgh, PA, May 15-20. pp. 370-377.

[24] Weiser, M. (1993). Some computer science issues in ubiquitous computing. *Communications of the ACM*, vol. 36, no. 7, pp. 74-84.