A Web-Based Nomadic Computing System

Tim Kindberg and John Barton

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

Abstract

CoolTown offers a web model for supporting nomadic users, based on the convergence of web technology, wireless networks and portable devices. This paper describes how CoolTown ties web resources to physical objects and places, and how users interact with resources using the information appliances they carry, from laptops to smart watches. Enabling the automatic discovery of URLs from our physical surroundings, and using localized web servers for directories, we create location-aware but ubiquitous systems. On top of this infrastructure we leverage device connectivity to support communication services.

Keywords

Web presence; nomadic computing; location-aware computing; ubiquitous computing; resource discovery.

1. Introduction

This paper describes work belonging to a research project at Hewlett-Packard Laboratories called *CoolTown.* CoolTown combines web technology, portable devices, and wireless communications to explore new kinds of systems, systems similar to nomadic [21] or ubiquitous [31] computing systems. Here we will show how a system created from elements of the CoolTown project form a basis for nomadic computing.

CoolTown operates from the premise that the Web and the technologies behind the Web provide a sound basis for pervasive nomadic computing. This premise is based on the web's potential to realize ubiquitous access, its lightweight software requirements, and its ability to operate locally as well as on a global scale. We start with a discussion of these properties.

Ubiquitous access. The Web supports pervasive computing because it is accessible in a large and rapidly growing number of places. This growth is a consequence of its design. The Web relies on a simple standard, the HTTP protocol [13], which can be implemented on a vast variety of devices. 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 and printers [11]. Furthermore, the Web supports mobile users through its flexible global addressing scheme, allowing them transparent access to resources outside their current environment. Finally, the web addressing and communications model can be bridged in to devices that do not support TCP/IP.

Just enough middleware. The diversity of the devices that mobile users may carry and encounter argues against specific software based on device type or even application type. It also argues against assuming that all devices will run a form of middleware, such as Java or CORBA, that is heavy on the commitments it imposes at the application, language, or resource level. Rather we believe that the Web standards of HTTP and URLs [4] have proven themselves to support adaptation to diversity. The form of interaction with particular devices and other entities should be encoded using XML [12] documents and MIME types, not binary formats and language-specific interface signatures. Our demonstrations include a variety of digital devices that we can integrate with minimal additional software.

Locality. We will demonstrate how to deliver web resources to nomadic users as they encounter new environments, without requiring a global wireless connection like a cell-phone, and without necessarily requiring any networking access beyond the immediate surroundings. 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 sometimes even a simple peer-to-peer interaction will suffice. Moreover, the user can discover and access these local resources without having to reconfigure their devices as they move from place to place.

While we believe these properties to be necessary for a pervasive computing solution, they are not sufficient to support nomadic users. Among the problems are:

- The Web grew out of the need for electronic document exchange, whereas nomadic users require information and interaction based around the physical entities in their environment.
- The Web relies on verbal or text-exchange of addresses for "resource discovery". For example, one Web user will tell another "look at *cooltown.hp.com*". Nomadic users will need new resource addresses continually and may not have easy access to keyboards: the Web model for resource discovery is not adequate.
- URLs are too bulky and volatile for routine use as identifiers.
- The Web is largely a read-only system supporting "browsers"; most content is placed on the Web through non-web avenues that therefore do not have its pervasive properties.

The CoolTown project seeks to expand the web model to support pervasive non-desktop computing and nomadic users in particular. We call the representation of physical entities on the web "web presence"; [20] describes our initial efforts to build and maintain web presence. We discuss the other three issues in this paper.

2. Nomadic computing model

CoolTown builds on HP Labs' experience in the development of embedded web servers [11] and systems of communicating appliances [19]. Our approach to nomadic users extends work on mobile web browsers ("wireless web") to support the augmentation of physical entities in the environment with web resources. In this way we see our work as combining the physical information systems work of Xerox PARC [27, 29] with the Web.

Our work describes an approach to system development based on assumptions about the hardware and communications resources available. To start then we need to layout those assumptions.

Nomadicity. Our first assumption is that users will be mobile: they will use computing resources in their homes, at work, as they shop and as they play.

Handy devices. Nomadic users carry some electronic devices with them: systems must support them.

Wireless communications. Nomadic users of portable devices expect them to employ simple, fast cableless I/O.

Heterogeneity. Nomadic users carry a diversity of devices, diverse by function – digital cameras, phones, and PDAs – and by communications – infrared, short range radio, cell phone.

Fixed resources. Nomadic users with communicating portable devices expect to use electronic resources – printers, kiosks, teller machines, Internet access points, and so on – wherever they encounter them.

Wired backbone. Nomadic users expect fixed resources to be strongly networked and they expect their devices to "use the web" in all locations, pervasively.

The CoolTown work is not on the devices or wireless communication layers, but rather on the layer of infrastructure that is needed to support nomadic users with portable devices. We work at the "web server" level and we see our work as extending web models to new arenas.

As a consequence of the above assumptions, most of our demonstration scenarios use a sensing mechanism, like a barcode reader or a short-range networking subsystem such as infrared, combined with long-range or wired web access. The sensor is used to obtain addresses (URLs) to access web resources. So we imagine our nomadic users carrying devices like:

- PDAs with 802.11 network access and barcode or other sensors, or
- Cellular telephones with additional short-range networking, or
- Cameras with short-range networks that interact with Internet-connected kiosks.

These are heterogeneous handheld devices with wireless connectivity to a web infrastructure. One such example is shown in Figure 1.

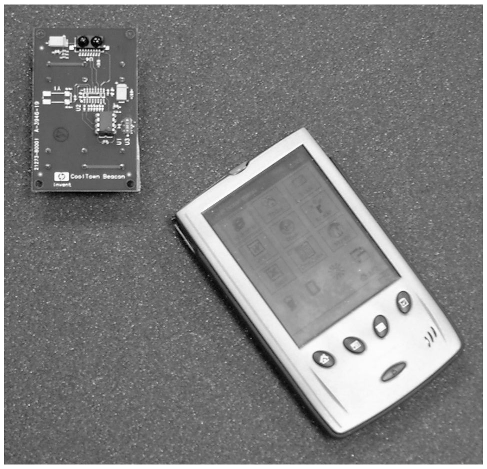

Figure 1. A palm-sized personal digital assistant (PDA) and an infrared beacon [15]. URL text stored in the beacon can be accepted through the PDA's infrared port then used to bring up a location-specific web page in the PDA's browser.

Present-day users of portable wireless devices view them as either adjuncts to larger desktop or workstation computers or as portable communications devices (phones or for e-mail). We view them as portable viewers of web pages and as sources of content, or as clipboards for URLs pointing to discovered content. Then we add support for obtaining location-dependent web pages and for moving content between mobile and fixed but online devices. Figure 2 shows how this support is organized.

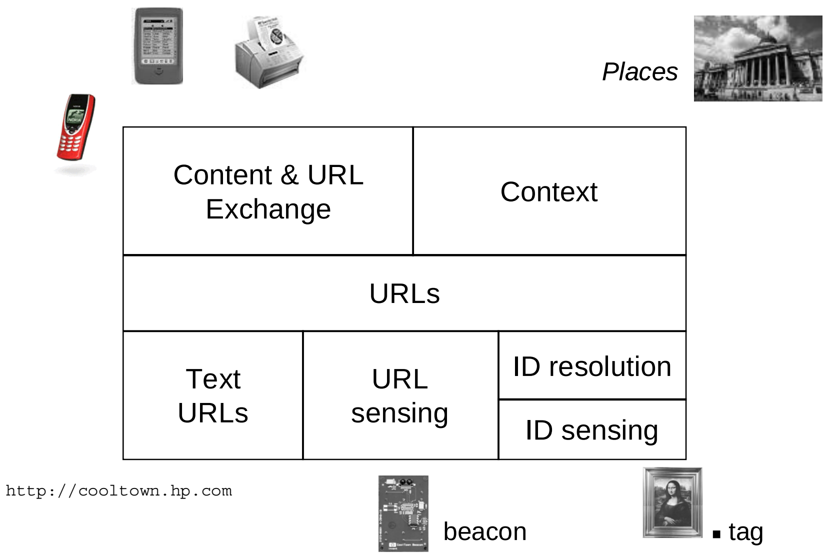

Figure 2. Component relationships. Sources of URLs, including keyboards, beacons, and resolved tags point to context or content.

The critical element of the system is the movement of URLs acting as Internet pointers. Components in the lower parts of the diagram provide addresses used in the upper layers. Keyboards and beacons provide URLs directly; tags on objects in the physical world are "resolved" to produce URLs. This layer is discussed in Section 3. The resolution of identifiers into URLs is context dependent and some of the URLs sensed in the environment lead to web servers that act as context for the resources in an environment; this is discussed in Section 4. URLs provide a means for accessing and directing content and they themselves can be part of content. Content exchange, discussed in Section 5, is critical to nomadic users as they require information on their environment and its resources or as they seek to modify that environment or its resources. Section 6 discusses our design in relation to related work. Section 7 concludes.

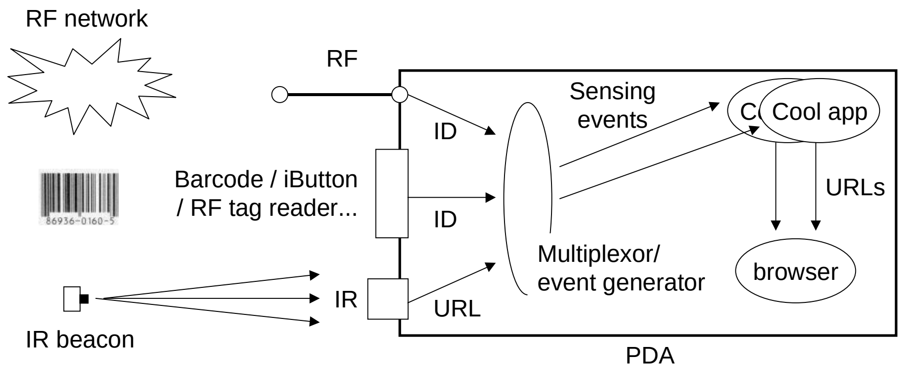

Figure 3. A sketch of a PDA equipped for sensing URLs and identifiers.

3. Sensing

In this section, we describe the bottom layer of our infrastructure, which enables the user to acquire URLs from their surroundings or from the physical entities in their surroundings. An obvious mechanism, with equally obvious drawbacks, is to write the URL on a label and attach it to the object. The user reads the label and types the URL into their browser.

Rather than being forced to use that cumbersome method, users should be able to identify conveniently what it is that interests them in their surroundings and obtain the corresponding URL *automatically*. To achieve this, their PDA is equipped with at least one short-range device that allows them to read URLs and other identifiers associated with places and individual objects in their surroundings, by pointing or positioning the device. We call this URL *sensing*.

Some options for sensing include:

- *IR & RF*. The infrared transceiver available on most PDAs and laptops can sense by communicating with a small, low-power ‘beacon’ device mounted at the place or on the object. Our beacon emits the identifier in infrared packets in the form of XML specifying a URL [15]. IR beacons can be configured to have a range of several meters. Similarly, short-range RF could be used.
- *Barcodes*. Barcode scanners are increasingly found integrated into PDAs, and are available as wands and other hand-held formats. Many objects are manufactured with barcodes – both generic ones and ones encoding their serial number; where they do not, barcodes can easily be printed and attached to objects.
- *Electronic tags*. A variety of small electronic tags exist containing a unique identifier with ~100 bits. All are available at relatively low cost. RFID tags can be read from a distance of about 10 cm with a hand-held reader. There are also tags such as iButtons [18] that are read on contact, which can be advantageous if tags lie close together.
- *Optical recognition*. The user's device may have a camera attached, in which case the identifier can be encoded in a form such as a 'glyph' [23] or digital watermark that is amenable to recognition software.

In *direct sensing*, the beacon or tag directly presents the URL of a web resource. In *indirect sensing*, it presents an identifier such as an ISBN or a UPC barcode. Then to obtain a URL we need a *resolver*, a service that returns a URL when given an identifier.

Direct sensing is the most concrete: it requires no resolver. An advantage of indirect sensing is that the same identifier can be looked up in different resolvers with different, context-dependent results. For example, sensing a book's ISBN barcode could lead to the local library’s entry for the book, if the resolver for the ISBN is the local library. It could lead to a catalog entry if resolved with the user's favorite bookseller's resolver. Or it could lead to the publisher's description of the book if resolved at the publisher's resolver.

Despite the variety of sensing technologies and resolution options, the implementation of URL sensing need not require significant software on handheld devices.

Figure 3 sketches a PDA with several sensing mechanisms (IR, barcode and RF) similar to prototypes we have developed in the CoolTown project. The device runs a web browser, and one or more CoolTown applications. The latter are simple applications that obtain URLs and present them as links, using the standard browser to display the corresponding web resources. As the figure shows, URLs and identifiers that are read by one or other of the sensing devices are multiplexed and announced within the device as standard 'sensing events', bearing the URL or other identifier and its source. The applications consume the events and handle them in various ways. All have a capacity for storing URLs and other identifiers as a record of what the user has physically encountered while engaged in their current activity, so that users can 'retrace their steps'.

One of those CoolTown applications provides indirect sensing. It can be configured to use any resolver of the user's choosing, but in particular it can automatically discover which is the local resolver when the user has designated which is the local context (see next section). When the user scans a tag such as a barcode, the application handles the sensing event by constructing a URN [3] from the identifier (if it is not in that form already). It uses the standard THTTP protocol [8] to request the local resolver to turn the URN into the URL of a resource. This URL is then made available to the browser.

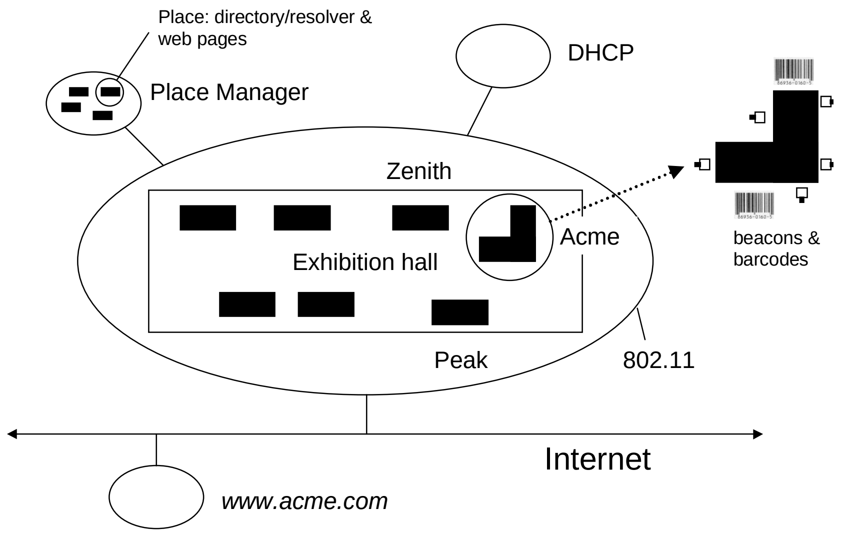

Figure 4. An exhibition hall 'place' containing the Acme stand and other places.

4. Context and Physical discovery

By attaching beacons or tags to easy-to-find points on physical entities or in physical places and by storing appropriate URL values in them (or in the resolver for them), we can create a connection between the entity or place and a web page. When a user approaches a beacon or tag they can use their device to find the corresponding web page. The user has "discovered" an electronic representation. This *physical discovery* involves the user in physically designating which places and things are of interest, a technique explored at PARC [29] and MIT [32].

Representation of places will be of particular interest to nomadic users. Web pages for places can help users orient themselves, providing maps or directions to nearby physical locations. But representation of places also provides *context*: virtual collections of related resources. Contexts provide organization: the desktop on the PC is an example of this idea. Our contexts take the form of web pages and a namespace for looking up web resources programmatically. We shall describe how place contexts are defined, discovered and used.

Places are distinguished from other types of context only by the fact that they are based on an underlying physical domain. The idea is for the place context to be made accessible when the user enters or nears the corresponding physical domain. Places may be fine-grained – for example, individual cubicles, exhibition stands and stores within a mall.

To get an idea of some of the issues in defining place contexts, imagine that Acme Software Inc. sets up a stand at the Wireless Web World exhibition (Figure 4). Acme has to define its own context within the exhibition, which it uses to provide information, printing, purchasing and other services to users who walk up to the stand. Visitors must be able to discover the Acme context when they walk up, distinguish it from that of neighboring exhibits, and, through their devices, these visitors must be able to browse and access the resources that Acme provides within that context. We are mainly interested here in resources that are based around the physical entities found at the exhibition stand. When the visitor reads a barcode attached, say, to a printer or a guest book on display, they are automatically shown the Acme web page offering them a service connected with it.

Some of the issues and our approach to them for place contexts include:

Discovery: Discovery is the acquisition of contexts and resources within contexts. Places are discovered in much the same way as individual physical objects, although they are typically extended and so may require several beacons or tags. Prominent “You Are Here” beacons or barcodes (or any other type of tag) attached to the Acme exhibition stand enable the user to pick up that place’s context. The user points the device at the beacon or scans the barcode. The result is the URL of the place: a set of web pages containing information and services that the user can browse on their device, and which devices can query.

Hierarchy and context overlap: Just as there may be many objects in their surroundings, there may be several place contexts to choose from: the user is by an exhibition stand, but there are others nearby; all are within the entire exhibition and, in turn, in a city. The web is a natural medium for presenting information to nomadic users because of its hypertext structure. The web pages for a place can include links to nearby places. The exhibition stand's context may contain links to resources that are not local to the stand itself – for example, to pages on the main Acme web site, and to the context of the entire exhibition.

Simple bootstrapping: To support nomadic computing, the system must take the user from switching on the device to obtaining a link to (i.e. the URL of) an Acme resource, with no user intervention except selection of which of the available places and resources interests them. Therefore, at least the root context for the exhibition appears directly as a URL that can be accessed once the device has fetched its IP configuration from a DHCP server. All other contexts (e.g. the Acme stand), and hence resources, can be discovered automatically from that root.

Scope: We re-emphasize that users are not forced to use the local place context when discovering the web resources associated with an object found in that place. Recall our example of looking up the ISBN for a book: the Acme Software resolver for ISBN might lead to special offers on Acme books, but a user should be able to use the ISBN with another resolver to see how special that offer really is. The user can choose any context in which the book's identifier will be looked up. The point of a place is to provide locally relevant resources, not necessarily all relevant resources.

Place Managers

Our implementation of place context is as a web server with appropriate content describing the place [6]. The points of web presence of the designated objects will appear as links in the place’s web pages. They will also be discoverable programatically, using lookup by identifier, and lookup by service type and other attributes. The 'local resolver' that we described in the previous section is an example of a resource that can be looked up automatically from the place's context.

A *place manager* provides access to and configuration of the data associated with one or more places. In the example of the exhibition hall, we shall assume that there is one place manager, which holds data about all the places within the exhibition hall. Different places are managed as separate objects. If this is not satisfactory from a security point of view, then the operators of a stand can supply their own manager for their place.

The place manager has the following functions:

- It maintains directories of the resources in each place, and provides an interface by which resources may be added, queried and removed.
- It acts as a resolver for each place, for looking up resources from their identifier.

- It acts as a web server, providing information about each place's resources in the form of HTML (for user browsing) and XML (for programmatic access).

Returning to the Acme Software convention stand example, the objects to be added to the Acme-stand place include samples of Acme-manufactured devices, a printer and a guest book. The Acme devices offer points of web presence that provide information and simple ways of interacting with them. The printer is for the convenience of visitors wanting to take material away with them. Visitors can write in the guest book in the conventional way but it also has a web presence, for those who prefer to fill in a web form.

Physical registration: defining a place

Physical registration is the dual of physical discovery: it is the mechanism for those who define the contents of places (“place masters”) rather than those who visit them. This involves physically designating what is to be bound in the place's context and where, physically, that context is deemed to be relevant.

Returning to the Acme Software exhibit booth example, consider the issues that face the Acme place master as she defines the place:

1) Instantiation: She must create a new place context at the Place Manager. She may wish to define a custom look and feel for the web pages that users will see when they browse the place.
2) Physical linkage for the place: On learning the URL for addressing this place context locally, the place master configures beacons or prints barcodes with the URL, and places them around the exhibition stand for the visitors to read.
3) Collection: She uses her PDA with an integrated barcode scanner to sense the identifiers of the Acme products, printer and guest book, whose web presences she wants to bind in the place's context.
4) Resolver configuration: As the place context does not yet exist, the identifiers sensed at the booth must be resolved elsewhere to obtain their web presences. In a mature environment, Acme will have a resolver to look up identifiers in an Acme “inventory” context, which contains an entry for every object in the Acme corporation's inventory. There are two resolvers in this process: the inventory resolver, and the place resolver. The former is accessible only by the place master (and other Acme employees). The latter is accessible by visitors to the place.
5) Registration: The collection of URLs is inserted into the newly instantiated place context. These are the URLs that will appear on the place’s web page.
6) Rendezvous (presence-tying). The place master next links the two points of web presence for each of the devices such as the printer. The printer runs a web server, which provides a web page enabling users to read its status, set its configuration and print. That web page is additional to the one that is hosted on an Acme server, and which was looked up in the inventory context in step 2. The problem to be overcome here is that the URL of the web page supplied by the printer depends on whatever IP number was assigned to it in the current environment: the Acme server has no way to predict its value and hence it cannot get a URL for the printer’s server.

Our goal for physical registration is simple operation. For most of the steps we have good solutions. The place master can initiate the place with a web-based form. The form processing can return a barcode to be printed and attached to her booth. She walks up to a series of objects with her PDA, reads their barcodes, and binds them in the new place. She can even include nearby objects that are not strictly within the physical extent of the exhibition stand – for example, a public printer in the exhibition hall. She does not have to be concerned with network topologies: she physically determines the resources in the place.

One of the most difficult problems is rendezvous (presence-tying). We have devised and are investigating three approaches to this problem. In one, the place master 'squirts' the URL of the place into an IR port on the printer. The printer then registers its attributes, including the URL, in the place's directory, where the place-definition application finds it. In the other, the printer has a barcode with its unique MAC identifier. The place-definition application multicasts a query for that identifier (in the manner of service discovery protocols), which the printer, and only that printer, responds to. In the third approach, the printer was configured with the URL of a global web presence where it registers its current IP address. The printer’s tag enables the place-definition application to retrieve the printer’s IP address, and thus obtain the URL of the point of web presence hosted on the printer itself.

5. Content exchange

Up to this point we have described how a nomadic user can enter a new place, obtain the address of the place context, and then browse the resource there. The level of function provided by (read-only) browsing is adequate for some nomadic tasks, but other tasks require more interactions with resources. In this section we discuss content exchange, a simple approach to interaction.

Content exchange supports the opposite of browsing: the nomadic user can push content into the pervasive infrastructure rather than just pulling content from it. The effect of pushing content can be varied. We shall discuss some of these various effects, then an application-level protocol approach that ties these together, ending with a discussion of some practical issues.

Direct content post

Pushing (or 'posting') content works between a source of content and a sink for content. Consider a nomadic user entering a conference room with a camera, photographing the content on a white board, then printing the image on the room's printer. The camera is the source for the image content; the printer is its sink. We could view this interaction as "printing from the camera". However, imagine the same user with the same camera and image, but interacting with the room's electronic projector. Or, perhaps the room is equipped with an electronic picture frame and the user wishes it to display the image. Or, the image is on the user's PDA instead of the camera, but is still sent to the same sink devices. In a pervasive computing environment, we need all combinations that users imagine should work to work. All of these scenarios can be handled similarly: the image just needs to be transferred from source device, the camera or PDA, to a sink device, the printer, projector, or picture frame.

To accomplish direct content transfer we need only have a simple push interaction: the content source opens a connection to the sink and writes the content (See Figure 5a). The only agreement we need between sources and sinks is the format of the data. Many imaging devices support JPEG format: it serves as the common format for images today. Since devices may use other common formats and since formats do evolve, effective interaction requires some introductory content preceding the data that identifies its format. The format of this introductory or "meta-" data has to be widely agreed upon and stable over time; our approach is to use an ASCII encoded XML format carried in a MIME [14] entity. Should the source attempt to transfer a type of content that the sink cannot handle, such as sending audio to a printer, the response indicates this error. We should also allow "content-negotiation", that is the source and sink should be able to exchange information on the formats they support to select a common and appropriate one [19].

Indirect content post

In addition to direct content like the image from a camera, we can also post content containing hypertext links to other content. One example is pushing multimedia: replace "image" in the example above with "hypertext multimedia document". For example, the content transferred from the PDA to the projector may be HTML that links images. As the projector (or the projection service that manages the projector) renders the HTML it resolves and fetches linked content (see Figure 5c). If the links point back to the PDA, it must act as a server for images; if the links point on to a network or the open Internet, the projector must have network or Internet access and fetch the images. This works similarly for sending the multimedia document to the printer or picture frame.

Another example is our IR beacon transmitting a simple kind of "document" containing a single URL. The beacon acts as a broadcast source of this URL; a PDA that picks up the beacon acts as a sink for the content, but it can also dereference the link – as the user views the indicated place, for example.

Transferring a URL instead of direct content has the advantage that the sink may be able to render the content with higher quality than the device that possesses the URL. For example, if a PDA transfers a link to a printer, then the printing service can download and print the document at a high resolution. This can be a distinct advantage over downloading the content to the PDA, rendering it on a tiny screen, and then printing the result.

Figure 5b shows indirect content exchange where the transmitted link refers to a third-party origin server.

Devices as 'clipboards'

One of the aspects of nomadic computing that we want to support is the use of appliances as 'clipboards' of content, or 'intermediaries'. Note that when a PDA is used to transfer an image to a projector, it is probably acting as an intermediary. That is, the original source of the image might have been a camera or a scanner or an online Web server rather than the PDA. The PDA is usually viewed as a temporary storage device for human input and output, but it can be used as a temporary storage device that ferries content from sources to sinks.

Link content is particularly interesting: because link content is small, small devices like wristwatches and key fobs can carry it. Thus we can consider a pervasive computing world in which nomadic users carry URLs in their watch pointing to their presentations. When they arrive at a meeting they "squirt" the URL into a projector which pulls the presentation down. Similarly, a user can use their watch to pick up the URL of the music currently being played on an 'Internet radio' with a beacon attached [22]. They can then take it to a friend's house and play the same piece on their Internet radio by pushing the link to it using their watch.

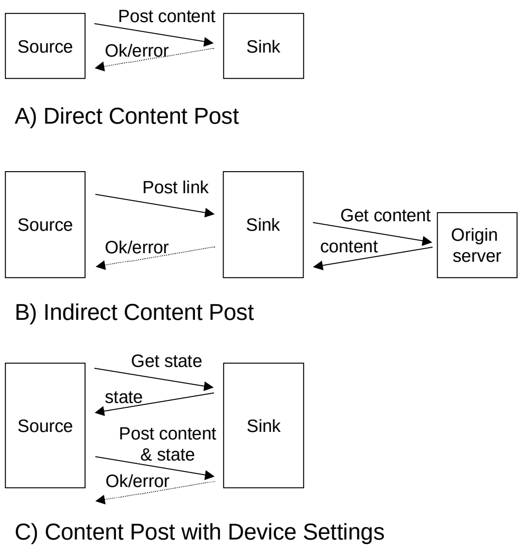

Figure 5. Some content-post component interactions. The arrows indicate application-level messages; dotted arrows are replies whose delivery does not affect the post processing at the sink. (A) Direct content post. (B) Indirect content post where the posted content links web content on a third-party origin server. (C) Direct content post with concurrent post of device settings; the settings would typically be modifications of previous settings obtained by a GET from the device.

Setting options on the sink

Transferring image content from a camera to a printer has a seemingly obvious consequence: the image gets printed. This obvious consequence must occur in a pervasive computing world that works intuitively. Sometimes we may want to alter the details of printing. Often users can accomplish this by front-panel settings on the printer. Alternatively we can use a “soft”, “remote” approach to changing the sink device operation. To maintain the intuitive nature of the operation, we should look for ways of changing the printing settings that does not disturb the simplicity of transferring an image from a camera to a printer. Thus we want something like "settings-enhanced" content-post: we would like to send the image from the camera to the printer together with some printer settings. Using a single operation like HTTP-POST ensures that correct settings are bound to a given content and it closely resembles the approach we use for cameras sending content to printers without settings.

A simple approach to settings-enhancement is to encode the setting values as XML in the transmitted web page. This allows a simple integration with web forms as we discuss next.

In a pervasive computing environment we cannot rely on preconfiguration of a camera with the available printer settings. We need some way for the camera to set the configuration of the printer in a device-independent manner. We can apply the technique used so successfully with web browsers: we have the printer provide a device-independent description of its device-dependent features as a web form and then render this description for the user on the user interface of the camera. For example, we can have the camera issue a GET to the printer's web server and have the printer return an XML or HTML form listing the possible options. Once the camera user has selected the settings, the new settings and the image can be transferred to the printer (see Figure 5c).

Exactly the same approach can apply to the interaction between the camera and the projector. The projector will send device-type dependent values to the camera for user selection that are different from those sent by the printer. However, the process of getting those options into the camera, rendering them for view, and sending them back to the projector are not dependent upon projection: the camera is acting like a web client with content to upload and the projector is acting like a web server ready to read content. This approach allows the camera-user to see devices like projectors, picture frames, and printers as pervasively available.

A detailed proposal for a content-post protocol with the properties described is described elsewhere [2].

6. Discussion and related work

Place contexts

In general, place contexts can be defined not only in terms of the physical domain but also in terms of other factors, such as the time of day, the identity of the user, the user's device, and the user's current activity. Some of these factors have been explored in tourist guides [9] and other applications such as overviews of working groups and meetings [26].

Section 4, for the sake of simplicity, shows only how places are defined irrespective of factors such as time of day. In fact, the data supplied by the Place Manager can be made a function of those factors [6]. Both the HTML and XML forms of web presence as supplied to the client are dynamically generated using annotations that the place manager provides. The annotations select which content to hand to the user according to attributes such as the current time and the user's identity.

Another aspect of places is that they may be active, and not the passive providers of web resources that we have described. For example, if a user from Nottingham enters a place, then those who are interested in meeting people from this, their home town, could be notified by the place. Stanford's iRoom [13] has such an event-based architecture. The CoolTown project is also investigating the possibility of active place contexts, which can alert users via a variety of means, such as on their mobile phone or pager. Users may have a point of web presence associated with their activities in a particular place, and the devices that they carry. This can be used to determine how to communicate with the user. It can also take account of the history of the user's involvement with the place when presenting it to them. For example, a repeated visitor to a café can be presented with suggestions based on their previous orders.

Other types of context

The resolver technology that we have been developing also supports contexts that are not tied to any physical domain. An individual entity such as a painting may possess a point of web presence not only in the context of the gallery where it currently hangs, but also in a variety of virtual art catalogues. It may even have a point of web presence in a private context shared by a group of art students and teachers, which they use to provide comments for one another. That context could contain an entry for each painting in any gallery in the world that students in the class have found interesting. When a student looks up a painting's point of web presence in the context, they will find comments left on that page by other students in the class.

Contexts have been established for published artifacts based on their 'distributed object identifier' [10], for products based on their UPC code, for articles in newspapers and magazines, and for artifacts in the office [29] and museums [24]. The research issue is how to choose the context (the resolver) for a given identifier so as to present appropriate resources, but to do so automatically wherever possible. Users require a choice, but a balance is needed between choice and ambiguity. The system must scale to trillions of points of web presence. At CoolTown, we are investigating resolver technologies that will support contexts shared between groups of individuals (the art class) or spanning organizations (the inventory context), while linking those contexts to more global ones where appropriate. Thus a printer's identifier could yield not only its pages within the host organization, but also pages from the manufacturer.

Discovery and registration mechanisms

There are three main discovery and registration mechanisms in addition to the physical mechanisms described in section 2: those based on zones or cells, triangulation or network discovery.

Zone or cell-based (for example, '911' cell-phone tracking and postal codes) are too coarse for fine-grained places such as stores and exhibition stands. On the other hand, they are useful for large-grained places such as sections of towns and cities.

Triangulation-based tracking systems include satellite-based (GPS), and terrestrial radio and acoustic schemes. These are inadequate for designating places that have fuzzy or complex topologies. The combination of indoor working and fine-grained localities argues against use of GPS; even radio frequency triangulation techniques may be of insufficient resolution; acoustic tracking requires line of sight.

Network discovery systems such as Jini [1] and SLP [16] use the ability to reach a resource over the network in a small number of hops as the criterion for what is local. This is often not an adequate criterion, as the exhibition example illustrates. Neighboring stands, for example Acme, Zenith and Peak in Figure 4, employ the same 802.11 network, which permeates the entire exhibition hall. Nothing in the network corresponds to those places. This is not an artificial example: the same is true of meeting rooms in buildings, for example.

We have developed our alternative, physical discovery and registration, because it allows users to decide exactly what it is that they are interested in based upon their understanding of their surroundings – not the happenstance of physical and networking topologies.

Privacy

Nomadic computing environments present extra potential threats to users' privacy that are currently not typically found, or only found to a lesser degree, in the virtual world of the Internet or in stores and other public places. The extra threats stem from the facts that nomadic environments potentially process detailed location information, and they provide points of exposure with which users are not currently familiar, such as utilization of a wireless networking infrastructure. We shall deal briefly with the question of location information.

In active badge systems [27, 30], locations sense users via the badges they wear, which emit a unique identifier. The system thus knows where each user is, without the user having to take any action. This is the inverse of the CoolTown approach, in which users decide whether to sense their location, and can do so without exposing any identifier except their IP number. The trade-off is between privacy and transparency.

Spontaneous networking

Entities in CoolTown are web present rather than, say, Jini-present [1] or CORBA-present [25]. This is because we believe that content-oriented computing as opposed to object-oriented computing has helped make the web successful, and that the same rationale applies in nomadic computing.

In its favor, Jini is an example of a system that provides answers to two key questions in nomadic computing: how to discover resources in a new environment, and how to utilize them. Jini uses network discovery to answer the first question. The resource is registered with specified attributes, and any device that knows the same descriptive format and vocabulary can discover that resource if it is in networking range. CoolTown's place manager satisfies that same function, except that we have added physical discovery and registration, for the reasons discussed above.

Jini's answer to the method of interaction between devices and the resources in their environment involves the download of Java object code to the device. This is better than systems based upon conventional, pre-installed device "drivers". Drivers contain code that runs on the content source to "drive" the operations of the sink. Conventional drivers are specific to the combination of particular sources and sinks. Consequently, in a world of many different devices, driver-based device control becomes a significant configuration problem. Jini leverages the processor-independence of a Java Virtual Machine to reduce the scale of the driver problem: one driver is needed for each sink device, but all source devices run the same driver and the driver is transparently downloaded. Jini requires all source devices to run Java Virtual Machines, implying considerable processing and memory resources.

The content-post approach we use supports simple device interaction without the need to install device drivers for every type of device encountered, dynamically or otherwise. It allows simple operations with minimal run-time support (a small layer on top of HTTP) without precluding the use of drivers for other operations. HP's JetSend [19] also embodies the principle of 'not modeling the device'.

7. Conclusion

We have outlined many of the important issues any system supporting nomadic users of pervasive systems must face, and provided web-based solutions to many of them. Critical to our approach is technology to connect physical objects and places with their corresponding web presence. We have a good deal of progress in that area. The web presence for places and its use as a context for nomadic users has been explored. Mechanisms for nomadic users to exchange content with each other and with devices that they encounter that have low software and configuration requirements have been developed. While much work remains before pervasive support for nomadic users will be a reality, our web-based approach has been easy to add to incrementally, suggesting that it will provide a sound basis for deployment.

Acknowledgements

The work described here contains contributions from many people in the Cooltown project. The concept and demonstration of the URL beacons is due to Jeff Morgan, Marcos Frid, Bill Serra, and John Schettino. The content transfer ideas came from the HP JetSend team with special thanks to Peter Willams and from Jeff Morgan. We also thank Gene Becker and Gita Gopal for helpful discussions.

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] Barton, John J., Simple Object Update Protocol, in preparation, available by request from John_Barton@hpl.hp.com.

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

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

[5] Box, D., Ehnebuske, D., Kakivaya, G., Layman, A., Mendelsohn, N., Frystyk Nielsen, H., Thatte, S. Winer, D. SOAP: Simple Object Access Protocol. http://www.w3.org/TR/SOAP/, 2000.

[6] Caswell, D. and Debaty , P. Creating web representations for places. *Proc. Second International Handheld and Ubiquitous Computing Symposium, HUC'2000*, Bristol, England, 2000.

[7] CoolTown home page. http://www.cooltown.hp.com/.

[8] Daniel, R. A trivial convention for using HTTP in URN resolution. Internet RFC 2169, 1997.

[9] Davies, N., Cheverst, K., Mitchell, K., and Friday, A. Caches in the air: disseminating information in the Guide system. *Proc. Second IEEE Workshop on Mobile Computing Systems and Applications* (WMCSA '99), New Orleans, Louisiana, U.S., 25-26 February 1999.

[10] Digital Object Identifier home page. http://www.doi.org.

[11] Embedding Web Access Mechanism in an Appliance for User Interface Functions Including a Web Server and Web Browser. *United States Patent no. 5956487*, Sep 21, 1999. Inventors: Venkatraman, C., and Morgan, J. Assignee: Hewlett-Packard Company, Palo Alto, Calif.

[12] Extensible Markup Language (XML) 1.0 http://www.w3.org/TR/REC-xml.html

[13] Fox, A., Johanson, B., Hanrahan, P., and Winograd, T. Integrating Information Appliances into an Interactive Workspace, *IEEE Computer Graphics & Applications*, 20(3), May/June 2000

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

[15] Frid, M. Personal communication.

[16] Guttman, E. Service Location Protocol: automatic discovery of IP network services. *IEEE Internet Computing*, Jul.-Aug. 1999. pp. 71-80.

[17] HTTP - Hypertext Transfer Protocol. http://www.w3.org/Protocols/.

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

[19] JetSend home page. http://www.jetsend.com/.

[20] 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. People, places, things: web presence for the real world. Tech report HPL-2000-16, Hewlett-Packard Laboratories, 2000.

[21] Kleinrock, L. Nomadicity: anytime, anywhere in a disconnected world. *Mobile networks and applications* 1(4), 1997, pp. 351-357.

[22] Krishnan, V., and Chang, G. Customized Internet radio. *Proc. Ninth International World Wide Web Conference*, Amsterdam, 2000.

[23] Moran, T., Saund, E., van Melle, W., Gujar, A., Fishkin, K., and Harrison, B. Design and technology for collaborage: collaborative collages of information on physical walls. *Proc. Twelth Annual ACM Symposium on User Interface Software and Technology*, 1999, pp. 197-206.

[24] Oberlander, J., Mellish, C., O'Donnell, M., and Knott, A. Exploring a gallery with intelligent labels. *Proc. Fourth International Conference on Hypermedia and Interactivity in Museums*, 1997, pp. 153-161.

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

[26] Salber, D., Dey, A.K., and Abowd, G.D. The Context Toolkit: aiding the development of context-enabled applications. In *Proc. 1999 Conference on Human Factors in Computing Systems (CHI '99),* Pittsburgh, PA, May 15-20, 1999, pp. 434-441.

[27] Spreitzer, M., and Theimer, M. Providing location information in a ubiquitous computing environment. *Proc. Fourteenth ACM Symposium on Operation System Principles*, 1993, pp. 270-283.

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

[29] Want, R., Fishkin, K.P., Gujar, A., and Harrison, B.L. Bridging physical and virtual worlds with electronic tags. In *Proc. 1999 Conference on Human Factors in Computing Systems (CHI '99),* Pittsburgh, PA, May 15-20, 1999.

[30] Want, R., and Hopper, A. Active badges and personal interactive computing objects. *IEEE Transactions on Information Systems*, 38(1), 1992, pp. 10-20.

[31] Weiser, M. Some computer science issues in ubiquitous computing. *Communications of the ACM*, 36(7), 1993, pp. 74-84.

[32] Yarin, P., and Ishii, H. TouchCounters: designing interactive electronic labels for physical containers. *Proc. 1999 Conference on Human Factors in Computing Systems (CHI '99),* Pittsburgh, PA, May 15-20, 1999. pp. 362-369.