

Informative Things: How to attach information to the real world

Rob Barrett *Paul P. Maglio*
IBM Almaden Research Center
650 Harry Road, NWED-B2
San Jose, CA 95120
408-927-{1911, 2857}
{barrett, pmaglio}@almaden.ibm.com

ABSTRACT

We describe a new method and implementation for managing information through the use of physical objects. In today's networked world, the trend is toward working in a global virtual environment. To transfer information, users are responsible for finding an appropriate storage location, naming the information, selecting the transport mechanism, and setting the access permissions. Much of the time, these burdens are needless and, in fact, stand in the way of productive use of the networked environment. In many circumstances, a physical floppy disk is the ideal medium for transferring information, as it eliminates these complications. Our *Informative Things* approach provides a "floppy-like" user interface that gives the impression of storing information on physical objects. In reality, our system stores information in the network, associating pointers to information with objects in the physical world. By hiding these details, we simplify information management. By linking the physical and virtual worlds, we leverage users' highly-developed ability to work in the real world.

Keywords: physical user interface, cooperative work, networked information

INTRODUCTION

To simply find a computer file, one often searches through several directories, through several hard disks, or even through data on several networked computers. Because of the sheer size of one's personal information space, keeping track of where things are located has become a chore. Modern human-computer interfaces provide many mechanisms that attempt to help, such as navigating a hierarchical file structure or following paths through the world wide web. Because people naturally think of such information spaces in physical terms [9,10,15], user interface designers often try to create virtual worlds that are

similar to the physical world (e.g., [3,16]). Nevertheless, the sheer volume of information available today inevitably leads to problems in managing it [11].

We propose a new approach to information management in which users interact with ordinary physical objects. A simple floppy disk is our canonical example of a physical object that holds information and that can be used to move information from place to place. We believe that in many cases, the floppy disk is exactly the right mechanism for moving information around. It simplifies the steps the user must go through to copy a file from one physical location to another; for instance, when one of us has updates for the paper we are writing together, we simply put the file on a floppy and walk it over to the other's office. Alternatives to this "sneaker net" method include emailing the file, putting the file on a shared disk, and posting the file on an ftp site or on the web. Each of these mechanisms, however, involves a series of decisions (e.g., where to put the file on the network so that the other can access it, what to name it, and so on) and a series of time-consuming steps (e.g., ftping the file or detaching it from an email message) that are largely irrelevant when using the floppy disk. In short, when using a physical object to transfer information, there is no question where the information is, what it is called, or how it can be obtained: it is contained by the physical object and can be obtained through the physical object.

One problem with floppy disks, of course, is that storage capacity is severely limited. But in a networked world, floppy disks need not actually contain the information that is presumably on them. They need only contain pointers to the information. More precisely, we propose a method for labeling physical objects and for associating those labels with network information sources. This approach, called *Informative Things*, relieves the user of many burdensome tasks involved in naming, finding, saving, and managing networked information by providing a simple, physical user interface to the information. One example is the floppy disk that can hold an unlimited amount of data, such as the installation for Netscape Communicator. Another is the Palm Pilot that can hold all files or email to be taken home at night. Each physical object, such as the floppy or the Palm Pilot, has a unique identifier, and when information is

copied to the object, it is in fact *attached to* the object's identifier in a central database. When the floppy is placed in the drive or when the Palm Pilot is hot synced, the data attached to object's identifier is found on the network and transferred to the local system. The key point is that the mechanism is hidden from the user, thus simplifying the user's job.

For the sake of usability, our method limits the flexibility of managing networked information. For example, in conventional networked computing, a file can be modified from anywhere. But in our system, the user who holds the physical object is the only one who can modify the information attached to that object. Other users cannot change what is attached to the object. Of course, this restriction makes sense when talking about real floppy disks, which users must physically possess to modify. Although our virtual floppy disks need not restrict who has modification rights for the attached data, we choose to limit this control to emulate the physical world. We believe the power of our approach lies in judiciously choosing constraints to maximize both usability and flexibility. In creating user interfaces, the temptation is always to widen the scope of applicability, giving the user ever more abilities. In this case, by taking operations away from the user (such as the ability to change the file an object points to), we have eliminated issues of giving permissions and deciding where to put files.

In what follows, we first situate our work in the context of related systems. We then present details of our Informative Things implementation. Next, we elaborate a series of user scenarios that are simplified or facilitated by Informative Things. We then discuss the basic user-level issues our approach addresses. Finally, we discuss open questions and future work.

RELATED WORK

Two basic mechanisms underlie our approach: physical objects can be associated with a unique identifier (ID), and arbitrary information can also be attached to an ID. The identifier serves as the key in a lookup table associating a physical thing with information. Of course, the general idea is not new. For example, electronic hotel keys and corporate identification badges are physical objects with IDs, but they are specialized for their purpose and do not allow arbitrary information to be attached to them. Smart cards and personal area networks [17] are used for transmitting unique IDs, but these IDs represent a person's identity, allowing a system to be customized to individuals. They are not used as general information containers.

The most closely related work is that of Ishii and colleagues [5,7,13,14]. The goal of their work is to supplant the conventional graphical user interface (GUI) with a *tangible* user interface (TUI). This is a natural extension of the move from command line interfaces, which are completely

abstract, to GUI's, which connect the abstract computer operations to metaphorical physical operations. The TUI seeks to hide the computer completely and at the same time manifest computational power. For example, mediaBlocks can associate video (or other media streams) with wooden blocks, allowing users to move data by moving the blocks from video cameras to video monitors [13]. Thus, the tangible interface of the mediaBlocks conceals the computer from the user. The TUI will clearly be important when computers are ubiquitous, blending into the background of everyday life. In addition, in specialized domains, such as graphical design or video manipulation, a tangible interface can be specially designed to produce a highly efficient work environment [5,13].

In contrast, our Informative Things approach is not a replacement for the GUI, but attempts to simplify user-computer interaction by connecting the virtual world to the physical when appropriate. The operations that users perform with Informative Things can be done through the GUI, yet connecting information management operations to physical objects simplifies the tasks of (a) finding a storage location, (b) globally naming the information, (c) selecting the transport mechanism, and (d) setting the access permissions. All these advantages are gained without sacrificing the power of operating within the GUI's virtual world. Our key point is that, in a fully networked world, the conventional GUI can be enhanced by reifying information storage and transfer.

THE INFORMATIVE THINGS IMPLEMENTATION

We have implemented a simple web-based system embodying the Informative Things concept. The system consists of a client, a server, and a physical object or *Thing*. The client provides the user interface, whereas the server implements the backend. The client determines the ID of the Thing, which it then uses to find the information attached to the Thing. The client relies on a server to find the information associated with an ID.

In this section, we first present our method for determining the ID of a Thing, and we then discuss the methods used for accessing information attached to a Thing and for attaching information to a Thing.

How to Read IDs

There are many methods for determining the ID of a Thing and only a few will be discussed here. As mentioned, the simplest case is a floppy disk. In the current system, the ID is read from a hidden file that is automatically generated when the floppy is first accessed by the system. A combination of timestamp, machine ID, and random number is used to virtually guarantee uniqueness and prevent guessing. Alternatively, the volume label and serial number could be used. The advantage of the floppy disk is that in most cases no new hardware or configuration is needed to read the ID.

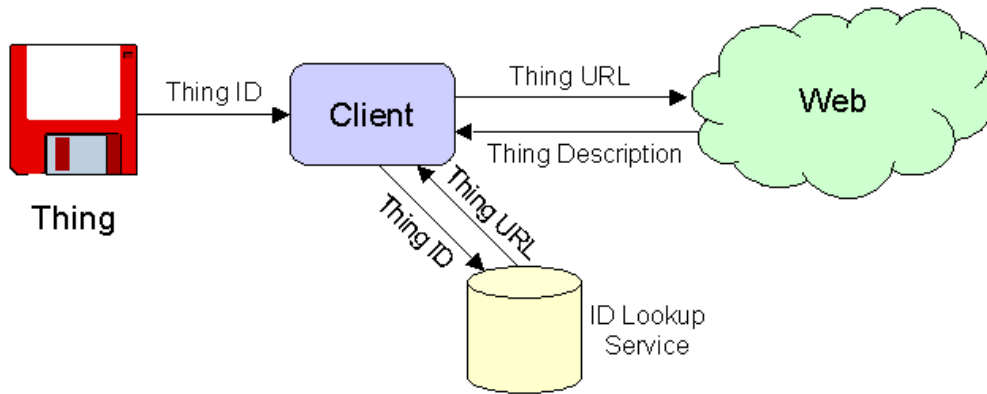


Figure 1. Client determines the Thing's ID, which it uses to lookup the Thing's URL. Given the Thing's URL, the client can find the Thing's description.

Other Informative Things can be accessed with special hardware. The communication mechanisms can be roughly separated into two classes: Things that actively broadcast their IDs and Things that can be passively read by an interested client. Active broadcasters include personal area network transmitters [17] and infrared transmitters. These devices contain their own internal power source and broadcast their IDs periodically, to be picked up by any receivers within range. By operating with a low duty cycle, such devices can have battery lives of several years and can avoid interference with other similar devices in the same area. These devices can be read at-a-distance and therefore do not have to be "plugged in" and do not suffer from unreliable electrical contacts.

Passive IDs are markings on a Thing that can be read by the client. For example, barcodes can be printed on many objects and can then be scanned by a barcode reader [7]. A more elaborate scheme for reading markings might involve a computer vision system [14]. Alternatively, a hardware connection to the client, such as through serial or parallel ports, can be used to determine an ID.

How to Access Information

Once the client has determined a Thing's ID, it can find the information attached to that Thing. Our current system uses two levels of indirection between the ID and the actual information (see Figure 1). A Thing ID is mapped to a Thing URL, which can then be used to find a description of the Thing. A Thing's description contains a list of attachment URLs, which in turn point to the data that is actually attached to the Thing.

The mapping from Thing IDs to Thing URLs allows the management of Thing information to be distributed. After reading a Thing ID, the client contacts an ID Lookup Service to determine the Thing URL. This lookup service could be a centralized well-known server, or it could be distributed (e.g., as the Domain Name Service system). In

either case, the client receives a Thing URL that can be used to find a Thing description. This ID-to-URL mapping allows the Thing descriptions to be stored in a distributed manner rather than all on one server.

In our current system, a Thing description is an XML document [2]. Figure 2 shows a sample Thing description, which contains information about the Thing as well as URLs through which the client can find the attached information. This method of encoding the contents of a Thing allows the client to choose its own rendering of the information, either through a web browser (using XSL [1] to convert the XML to HTML), or through a graphical user interface widget (see Figure 3). In another implementation, the client might export the contents of the Thing as a CIFS

```

<THING>
  <RANGE TYPE=INTERNET/>
  <NAME>My Thing</NAME>
  <FOLDER>
    <NAME>Today's News</NAME>
    <DOCUMENT>
      <NAME>Headlines</NAME>
      <URL>http://www.news.com/headlines</URL>
    </DOCUMENT>
    <DOCUMENT>
      <NAME>Sports</NAME>
      <URL>http://www.sports.com/</URL>
    </DOCUMENT>
  </FOLDER>
  <FOLDER>
    <NAME>Briefcase</NAME>
    <DOCUMENT>
      <NAME>ThingPaper.doc</NAME>
      <URL>http://thing.ibm.com/barrett/ThingPaper.doc</URL>
    </DOCUMENT>
  </FOLDER>
</THING>
  
```

Figure 2. Sample XML code for the information attached to an Informative Thing.

disk [8], making it available to the user's normal file

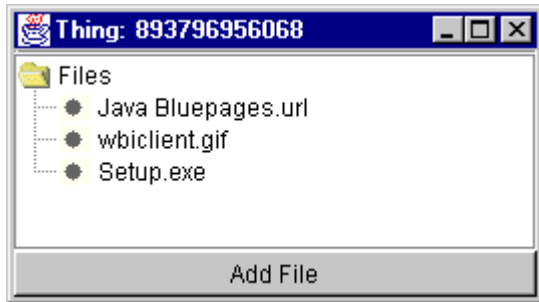


Figure 3. An example user interface that allows access to the information attached to a Thing.

system.

Once the client has obtained the attachment URLs from the Thing description, it can retrieve the data attached to the Thing by accessing those URLs. The data can be presented to user through a web interface or through a file system interface.

How to Store Information

The mechanism for attaching information to a Thing is similar. First, the information to be attached to the Thing must be made available on a web server. In our current system, a Thing can be made available in a particular range, either Local, Intranet, or Internet. If the range is Local, information to be attached to the Thing is put on the local machine, using "file:///" URLs. If the range is Intranet, the information is stored on a server within the corporate firewall. If the range is Internet, the information is stored on a server outside the firewall. This simplified access model provides the most basic permissions structure.

Once the attached data is stored, the client uses the ID Lookup Service to determine the Thing URL. If the Thing is new to the system, its ID and associated URL must be stored on the ID Lookup Service. In any event, given the Thing URL, the client can modify the Thing description (XML document) by reading the original contents, making any changes, and then re-storing the modified contents using an HTTP "PUT" request [4]. Alternatively, a CGI interface could be used for simple COPY, MOVE, RENAME, DELETE commands to manage the contents of the Thing description.

Security Issues

Maintaining strong security in a system such as this is difficult. A balance must be struck between security and usability. A number of attacks can be identified:

1. access to a Thing's attached data without ever having been in possession of the Thing
2. access to a Thing's attached data given previous possession of the Thing

3. access a Thing's attached data by users who have inappropriately gained possession of the Thing

In the first case, a user might either guess a Thing ID or eavesdrop on a network conversation that communicates the Thing ID. Our current system uses 64-bit Thing IDs, making the likelihood of guessing a valid ID quite small. SSL communication [6] with the ID Lookup Service can effectively eliminate eavesdropping on the Thing IDs. But even if someone has not guessed an ID or found one by snooping, because the data attached to a Thing is stored on the network using normal URLs, what is to keep other users from casually accessing the information with a normal web browser? First, some of the information stored on a Thing is generally available (for example, a pointer to the IBM Home Page). There is no need to protect these URLs. The issue comes for information that has been specifically exported to a web server for attachment to a Thing.

We would like to ensure that private information attached to a Thing can only be accessed by the user who is in possession of that Thing. Our approach has been to make the attachment URLs difficult to discover. This can be accomplished by the same means that make Thing IDs difficult to discover: (a) automatically produce complex URLs that are difficult to guess, and (b) use trustworthy clients that do not reveal the URLs that are being accessed. The first step is quite straightforward, but the second might imply that traditional web browser should not be used for accessing the information, as they casually reveal the URLs that are being accessed. A further protective mechanism would be to restrict access to exported data by requiring the Thing ID as a password.

The second security issue is somewhat more complicated. If a Thing were previously in the user's possession, that user could remember the Thing ID and use it later, even though the user is no longer in possession of the Thing. This would allow the user to manipulate the information attached to the Thing. To eliminate this kind of attack, the Thing can actively broadcast an ID that varies with time, thus reducing the ID's vulnerability to a small time window. Alternatively, the client software may again be trusted to not reveal the IDs of Things it handles. Of course, this protection can be defeated by programs that actively spy on the operation of trusted clients, for instance, by observing network traffic.

The third security issue involves preventing access by users who have come into illicit possession of a Thing. Our basic system assumes that possession implies permission to access attached information. But in some instances, additional security may be needed. Passwords or certificates can be added to the protocol for accessing the ID Lookup Service and for accessing the information attached to a Thing. This requirement creates the additional burden of maintaining passwords.

We have provided only a cursory look at some security aspects of Informative Things. Clearly, security in a dynamic distributed data environment is a complex issue, and will ultimately require more detailed consideration.

Scalability

To be useful on a large scale, Informative Things must be able to handle a large volume of transactions without any bottlenecks. The only concentration point in our system is the ID Lookup Service. As mentioned, this service could be operated in a hierarchical manner, similar to DNS, which can obviously scale as required. The load will be higher, however, if secure time-dependent Thing IDs are used or if the communications are encrypted, as with SSL.

The rest of our system is naturally distributed: Thing URLs and the attached data are stored on several servers. There is no problem with high numbers of hits on particular Thing URLs because only the user who is in possession of the physical object is allowed to access the attached information.

SCENARIOS

The Informative Things concept is broadly applicable, as it provides a general method for managing and transferring information. In this section, we describe several scenarios that illustrate some of this breadth.

The simplest example is the one we have been describing all along: two people sharing data in close physical proximity. The sender of the data takes a Thing and “connects” it to his or her computer. “Connect” may mean inserting a floppy disk, placing a PalmPilot in its port, scanning a barcode, or some other operation. In the background, the Informative Thing system reads the Thing ID, looks up the Thing URL and Thing Description, and displays an icon representing the Thing on the screen. The user then copies the files to the icon, disconnects the Thing, and hands it to a colleague. When the user copies the file, the system actually transfers the file to an appropriate web server and updates the Thing Description to reflect that this file is now attached to the Thing. When the receiving party connects the Thing to his or her computer, the attached data is now available for use. The transfer is simple, fast (assuming a fast network), and not limited by floppy disk capacity.

Another scenario involves a user who works with two different computers. Keeping files synchronized is always a chore. But using Informative Things and a personal area network (PAN) makes it simple. The user keeps a PAN card in his or her wallet. When the user sits at the keyboard of one computer, the PAN receiver picks up the PAN card's ID and displays an icon representing the card and its attached data. The user can simply use the files that are “on” the PAN card. When the user leaves the computer, the icon disappears. When the user sits down at the second computer, the same files are available there. The data is

actually stored and managed on a web server by the Informative Things system (with appropriate caching at the machine where the data is currently being used). But for all appearances, the user carries the data in his or her wallet.

Consider a more futuristic scenario in which information appliances such as viewboards are commonplace [12]. Users keep these simple interactive screens on their desks, carrying them around as needed to work with information. When enabled with Informative Things, a viewboard can be “connected” to a Thing and then the data attached to the Thing can be used. For example, a binder that once held paper copies of protocol specifications would simply have a barcode printed on it. When the barcode is scanned by the viewboard, its Thing ID is used to access the relevant specifications from the web. This physical-virtual notebook has some of the advantages of physical object (e.g., can be carried from place to place) and some advantages of a virtual object (e.g., seemingly limitless capacity).

By relaxing the constraint that only the holder of a Thing can modify its contents, we can envision a scenario involving a salesperson who needs an up-to-date sales presentation. By giving the salesperson an appropriate Informative Thing, he or she can always access the most recent version of the presentation while someone at the home office manages the versions. In fact, previous versions can also be maintained on the Thing so that new versions are never a surprise. This scenario raises issues of low bandwidth and sporadic connectivity, which can obviously limit the utility of Informative Things.

As a final example, we can imagine aiding navigation through physical spaces. Rather than telling a hospital patient to follow a red line to a green line to the radiology lab, the patient can be handed an Informative Thing that “knows” the pathway and destination. As the patient passes hallway monitors, they display appropriate directions. Alternatively, the situation could be reversed so that the hallway contains Informative Things and the patient carries an information appliance that know the destination. The hallway Things tell the appliance where the patient is and the appliance tells the patient the appropriate directions.

These examples are simply starting points for exploiting the idea of Informative Things. The concept provides the capability of attaching data to physical objects, which can be easily transported, remembered, shared, and organized. Though information is not actually stored “on” the object, the system provides the appearance that it is.

WHAT PROBLEMS ARE SOLVED?

The Informative Things system provides both a new conceptualization of managing networked information and also solutions to problems faced by real users. In this section, we discuss some of these problems and their solutions.

Namespace Resolution through Physical Objects

The networked computing environment promises “everything accessible by everyone, everywhere.” As we near this goal, the utility of universal connectivity is obvious. But a significant problem is that this environment requires a global namespace. One reason for the success of the world-wide web is that it helps to simplify the collection of hundreds of millions of URLs. Most pages of interest lie within several clicks of an easily remembered page, such as <http://www.yahoo.com>. The namespace is collapsed because pages can be reached by a relatively simple name and several mouse clicks. With Informative Things, we have collapsed the global namespace by connecting subspaces with physical objects. Though this method has intuitive appeal, computationally speaking, it does not provide more power than already exists. That is, in the networked world, any information can be accessed, but Informative Things restricts access: The user can only access information attached to a physical object. As we have demonstrated, by narrowing the namespace, this restriction facilitates some aspects of information management.

Information Migration to Shared Network Servers

The following sort of conversation occurs quite often among members of our group:

- A: Could you send me that file?
B: Should I mail it or should I put it on my ftp server?
A: Could you put it on the Unix server?
B: My password just expired. I'll put it on the web.
A: Great. What is the URL?
B: I'll make it “<http://w3/~barrett/temp/myfile.txt>”.
A: Do you spell “barrett” with one “t” or two?

This unfortunately realistic example points out that the networked world is quite complicated when it comes to spontaneously sharing information. A sharing mechanism must be decided upon, which often requires some level of negotiation. The information exporter must decide how to name the new resource and must then figure out how to transfer it. Finally, the global name must be successfully communicated to the information receiver.

Informative Things simplifies this by hiding the mechanism, naming, data transfer, and communication of the name. The exporter simply grabs a Thing, attaches information to it, and then hands it to the receiver. Furthermore, the range concept (which specifies Local, Intranet or Internet) provides a useful granularity for deciding what type of service to use for information storage.

Permissions

Managing permissions in distributed systems is complex. The most common models involve access control lists, group membership, and user ID/password management. Because of the difficulty of successfully using these affordances, most users make data either accessible to only

the author or accessible to the entire world. More complete permissions models result in significant headaches for the system administrators.

In the Informative Things approach, permission to access information is governed by possession of a physical object. This method corresponds to real-world security. Admittedly, this approach is inappropriate for geographically disparate users or for large numbers of users. However, as we have seen, there are many common shared data scenarios in which governing permission by physical possession is both adequate and useful.

OPEN QUESTIONS AND FUTURE WORK

Informative Things simplifies some kinds of data transfer. But we have not yet confronted all questions raised by this approach. In this section, we briefly discuss some additional issues.

Universal Connectivity

Throughout the discussion we have assumed universal, high-speed network connectivity. Although network technology is advancing rapidly, universal access has not yet arrived. Nevertheless, because many locations, such as universities and corporate offices, have sufficient connectivity, Informative Things is quite practical today in these settings. Further work on such technologies as replication, prefetching, and caching would make Informative Things applicable to a broad range of “sometimes connected” or low-speed connected users. For example, a user who occasionally connects via modem cannot assume that information attached to Things will always be accessible. Rather a method for indicating which information is to be replicated to the local computer must be devised.

Dynamic Information

In our current model, one set of information is attached to each Thing. However, it might be useful to have different sets of information attached to a single Thing depending upon who is viewing it. For example, consider attaching annotations to books. If all users see all annotations on a book, it would be like getting a book from the library that has been highlighted and marked up. Rather, we suggest that much of the time, different people would prefer to see only their own annotations attached to a book.

In other circumstances, one might want control of the information attached to an object while another is in possession of it. Again, this breaks our original model. One application of this would a floppy disk that always contains the latest operating system revision and drivers for a Thinkpad 760ED laptop computer, enabling the computer owner to always have pointers to the most recent software and to always be able to find the software simply by finding the physical floppy disk. However, because the “contents” of the disk can change while it is in the user’s physical possession, this application does not fit our metaphor.

Another example along these lines is the always up-to-date sales presentation discussed previously.

Ubiquity

A recurring problem in cooperative computing systems is that the enabling software must be present on all of the computers used by the cooperating parties. If one user hands another a file on a Thing, both users must have the base software installed on their machine, or else they cannot share the file in this way. This problem can be solved by using a floppy disk as the Thing. Because the floppy disk can contain arbitrary data as well as a Thing ID, we can put a copy of the enabling software on the floppy. So when one user gives another a floppy disk, it contains everything needed to access the file. The most attractive solution would be for the floppy to contain an HTML page and a Java applet that could be loaded by a browser and then automatically install itself on the host computer.

Loosening Constraints

Current computer networks allow users to (a) transfer data through a multitude of mechanisms and protocols, (b) store the data on shared devices, and (c) protect the data from improper access. The Informative Things concept takes this extremely flexible environment and constrains it to increase usability. By requiring possession of a physical object to access data, the user's power is decreased. As we work with the system, we find that it is always tempting to loosen these self-imposed constraints. For example, to allow the attached data to be modified by someone who does not possess the object (as described previously), or to allow users to hold on to Thing IDs for objects that they no longer possess. Similarly, we are tempted to allow more elaborate permissions schemes to handle particular security situations. But care must be exercised, because these loosened constraints quickly devolve back into the ordinary networked information model. Imposing the physical object limitation provides enhanced usability and should not be circumvented for temporary convenience.

CONCLUSIONS

In this paper, we have illustrated Informative Things, a new user interface model that simplifies many aspects of data transfer among physically close individuals. In standard networked computer environments, countless low-level decisions often stand in the way of productive computer use. For instance, to simply give a file away, a user must find an appropriate storage location, name the file, select a transport mechanism, and set appropriate access permissions. From the user's perspective, such details are often irrelevant. By combining both the promise of universal connectivity and the fact that people normally interact with physical objects, Informative Things moves information from the virtual world to the physical world, transforming ordinary physical actions (e.g., walking from place to place) into information actions (e.g., copying data from one computer to another). In the process, our

approach eliminates mundane problems of data transfer and opens up new possibilities for physical information interfaces.

ACKNOWLEDGMENTS

We thank Ted Selker and Dick Allen for conversations that helped us clarify our approach and implementation. We also thank the UIST reviewers and program committee for many detailed comments that helped us improve the paper.

REFERENCES

1. Adler, S. et al. *A Proposal for XSL*. World Microsoft Corporation, 1997. Available as: <http://www.w3.org/TR/NOTE-XSL.html>.
2. Bray, T., Paoli, J. & Sperberg-McQueen, C. M. *Extensible Markup Language (XML) 1.0. W3C Recommendation*. World Wide Web Consortium, 1998. Available as <http://www.w3.org/TR/1998/REC-xml-19980210.html>.
3. Dieberger, A. A city metaphor to support navigation in complex information spaces. In S. C. Hirtle & A. U. Frank (Eds.) *Spatial information theory: A theoretical basis for GIS (COSIT '97)*. Springer-Verlag, Berlin, 1997.
4. Fielding, R., Gettys, J., Mogul, J. C., Frystyk, H., Masinter, L., Leach, P., & Berners-Lee T. *Hypertext Transfer Protocol – HTTP/1.1*. World Wide Web Consortium, 1998. Available as <http://www.w3.org/Protocols/HTTP/1.1/draft-ietf-http-v11-spec-rev-03.txt>.
5. Fitzmaurice, G. W. Ishii, H., & Buxton, W. Bricks: Laying the foundations for graspable user interfaces. In *Proceedings of the ACM Conference on Human Factors in Computing Systems, CHI '95*. ACM Press, New York, 1995, 442-449.
6. Freier, A. O., Karlton, P., & Kocher, P. C. *The SSL Protocol*. Netscape Corporation, 1996. Available as <ftp://ietf.org/internet-drafts/draft-fraiser-ssl-version3-01.txt>.
7. Ishii, H. & Ullmer, B. Tangible bits: Towards seamless interfaces between people, bits, and atoms. In *Proceedings of the ACM Conference on Human Factors in Computing Systems, CHI '97*. ACM Press, New York, 1997, 234-241.
8. Leach, P. J., & Naik, D. C. *A Common Internet File System (CIFS/1.0) Protocol*. Microsoft Corporation, 1997. Available as <ftp://ietf.org/internet-drafts/draft-leach-cifs-v1-spec-01.txt>.
9. Maglio, P. P. & Barrett, R. On the trail of information searchers. In *Proceedings of the Nineteenth Annual Conference of the Cognitive Science Society* (Stanford University, Palo Alto, CA). LEA, Mahwah, NJ, 1997, 466-471.

10. Maglio, P. P. & Matlock T. Metaphors we surf the web by. In *Workshop on Personalized and Social Navigation in Information Space* (Stockholm, Sweden) 1998.
11. Mander, R., Salomon, G., & Wong, Y. Y. A "pile" metaphor for supporting casual organization of information. In *Proceedings of the ACM Conference on Human Factors in Computing Systems, CHI '92*. ACM Press, New York, 1992.
12. Selker, T. & Allison, J. Room with a view. Presented at *Computer Supported Cooperative Work '96* (Boston, MA), 1996.
13. Ullmer, B., Ishii, H., & Glas D. MediaBlocks: Physical containers, transports, and controls for online media. In *Proceedings of the 25th International Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '98*. ACM Press, NY, 1998.
14. Underkoffler, J. & Ishii, H. Illuminating light: An optical design tool with a luminous-tangible interface. *Proceedings of the ACM Conference on Human Factors in Computing Systems, CHI '98*. ACM Press, New York, 1998, 542-549.
15. Vicente, K. J. & Williges, R. C. Accommodating individual differences in searching a hierarchical file system. *International Journal of Man-Machine Studies*, 29 (1988), 647-668.
16. Waterworth, J. A. Personal spaces: 3D spatial worlds for information exploration, organization and communication. In R. Earnshaw and J. Vince (Eds.): *The Internet in 3D: Information, images, and interaction*. Academic Press, San Diego, 1997.
17. Zimmerman, T. Personal Area Networks: Near-field intrabody communication. *IBM Systems Journal*, 35 (1996).