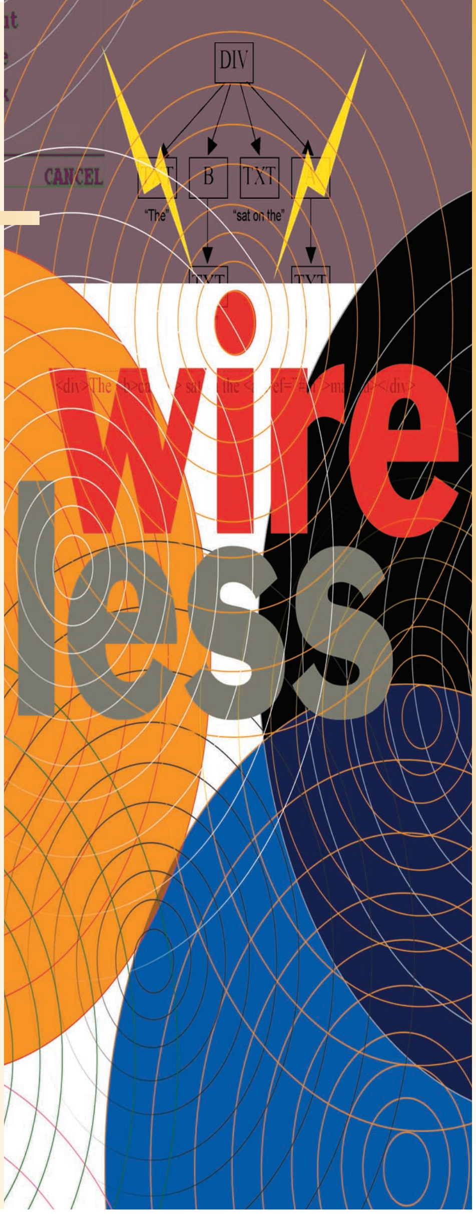
Web Interaction Using Very Small Internet Devices

Squeezing desktop Web content into smart phones and text pagers is more practical with separate interfaces for navigation and content manipulation. m-Links, a middleware proxy system, supports this dual-mode browsing, offering mobile users a range of actions on any Web link.



Bill N. Schilit
Intel Research

Jonathan Trevor
David M. Hilbert
FX Palo Alto Laboratory

Tzu Khiau Koh
Xerox Singapore

Most computer users interact with the Internet from a desktop or laptop computer running a Web browser such as Internet Explorer or Netscape. Interaction typically involves downloading and viewing documents that include content as well as links to Adobe PDFs, audio, video, Microsoft Office, and other HTML documents. Users tend to view content and links together, rapidly alternating between reading content and following links. When a user clicks on a link to a non-HTML document, the browser invokes a client-side plug-in application to display the linked content and in some cases lets the user manipulate it.

Because this traditional Web browser interaction model evolved on desktop computers, its user interface, hardware, and networking assumptions are uniquely suited to a desktop or laptop machine. In contrast, Internet-enabled cell phones, or "phonetops," typically accommodate only three to 12 text lines, and their design emphasizes portability and features such as battery life, audio clarity, and ease of selecting names from a phone book. Web interaction has, so far, been a secondary concern.

In some countries where cell-phone-based Internet users come close to outnumbering their desktop counterparts, many content providers create new content for very small displays. Yet overall, most Web content providers are neither equipped nor have the desire to do this tailoring, so handcrafted pages for phonetops represent only a fraction of the pages available to desktop users.

For the most part, automated techniques to address the feature gap between desktop and phonetop rely on the notion of transducing—translating HTML and images into formats compatible with small devices, which typically cannot handle HTML content. The "Fitting Desktop Content in a Small Display" sidebar describes fitting techniques in transducing and three other categories: scaling, manual authoring, and transforming.

Of the four techniques, transforming has the most potential for widespread use because, in the ideal case, it closely resembles professional content tailoring to a particular device yet without the manual overhead. A transforming system modifies both the content and the structure, or experience, of interacting with the content, as well as transducing

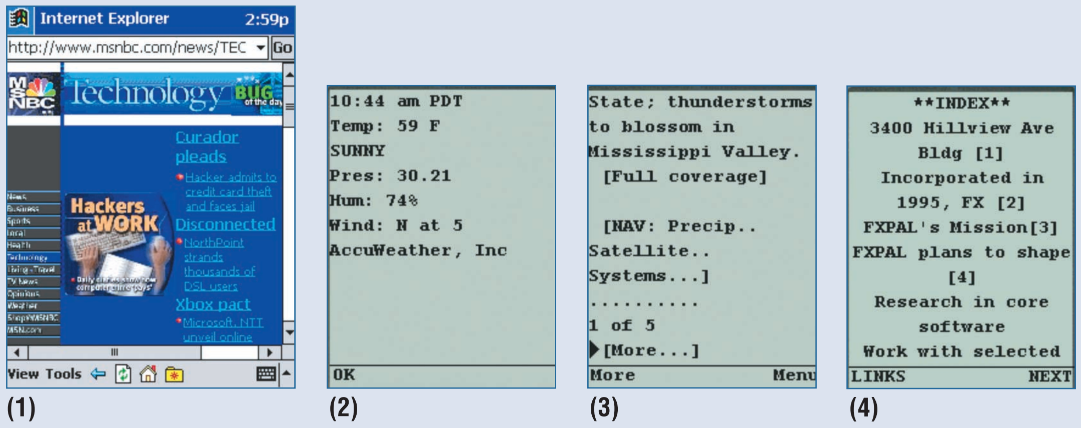
Fitting Desktop Content in a Small Display

Mobile Web devices span a range of capabilities. PDA-class devices like the Palm or Pocket PC can display many lines of text and graphics in a single screen. Midrange displays, such as the NeoPoint 1000, can show around 12 short lines of text with limited graphics capability. At the lower end are pagers and phones like the Samsung SCH-3500 with four lines of text and a maximum of 48 characters. We characterize very small devices as being sub-PDA in size, at the middle or low end of this range.

Methods for displaying Web information on these very small devices typically fall into four categories. Figure A shows how each method produces a different display.

Scaling

Web devices with high-resolution color displays, such as the Pocket PC with Pocket Explorer, provide a user experience that is the closest to desktop browsing. Figure A2 shows a display from the Pocket PC, which can render many types of Web content at full size with scrollbars to reposition the view. It can also scale down Web content in the viewer, using a fit-to-screen feature. However, although scaling can reduce scrolling, it also reduces readability and ease of use. As the device's screen size decreases, regardless of resolution, these problems become more severe. The other three techniques are better suited to the limited



```
10:44 am PDT
Temp: 59 F
SUNNY
Pres: 30.21
Hum: 74%
Wind: N at 5
AccuWeather, Inc
OK
```

```
State; thunderstorms
to blossom in
Mississippi Valley.
 [Full coverage]

 [NAV: Precip..
Satellite..
Systems...]
.........
1 of 5
▶[More...]
More                Menu
```

```
    **INDEX**
 3400 Hillview Ave
    Bldg [1]
  Incorporated in
   1995, FX [2]
FXPAL's Mission[3]
FXPAL plans to shape
       [4]
  Research in core
     software
 Work with selected
LINKS           NEXT
```

Figure A. Techniques for displaying Web content on small devices: (1)* Scaling *desktop content is possible on larger, graphics-capable, PDA-sized devices such as a Pocket PC. (2)* Manual authoring *by a professional designer produces a more custom and often less cluttered display for smaller devices. (3)* Transducing *literally converts desktop content into a linear sequence of subpages. (4)* Transforming *desktop content goes further by introducing new organizational structure, such as a table of contents.

to the appropriate markup language.

An example of a transformer for PDAs and laptops is the Web Reader's Digestor,[1] a proxy that intelligently splits a Web document into multiple subpages that better suit the smaller display and inserts new links for navigating among subpages. The system's goal is to relieve the Web designer of the need to reauthor Web pages for PDAs.

We experimented with using Digestor and found that, although it credibly transduced content for a range of small devices, such as PDAs, it broke down on very small devices, such as Internet phones. The resulting interface structure was just too difficult for users to understand and navigate.

As we explored alternatives to duplicating the desktop user's Web experience, we realized that browsing often involves both following links and reading, or more generally, navigating to information and then using it. Moreover, we saw that users often mailed, printed, saved, and even translated Web content—activities that a desktop interface could easily support, but not the limited interface of very small devices. Thus, rather than simply transforming pages for smaller screens, the direction should be to transform the Web browsing interaction into two separate modes: navigating to and acting on content.

On the basis of our Digestor experiments, we developed a Web browsing model that supports navigation and action in separate interfaces. To demonstrate the model, we created m-Links, a middleware proxy system that retrieves Web documents using HTTP, lets users navigate and apply services to Web content (URLs), and delivers a suitable user interface to a variety of small Web-capable wireless devices.

m-Links is both simple to use and powerful. Its simplicity comes from the navigation interface: Separating links from page content makes navigation a matter of selecting a link from a list. Its power comes from the action interface: Because users can apply various Web-based services to any link, they can do more with content than simply read it on their phones.

In creating m-Links, we met four important challenges of a modal Web browsing system:

screens of very small devices.

Manual authoring

Users typically have the best experience when the content provider—a professional Web or graphic designer—has tailored the Web content to fit their particular device. The content is laid out and summarized appropriately, and the interaction design takes into account known device limitations and idiosyncrasies. A similar approach employs manually authored page templates for each device type and populates these templates with content from a database. Unfortunately, because of the labor required, only a small fraction of Web content in Europe and the US is manually authored for any particular device. In Japan, where desktop computer penetration is lighter, the i-mode service provides many Web phone users with access to specifically authored compact HTML pages, but not to general Web content or other types of documents.

Transducing

Automated techniques for reauthoring Web content have become popular because they are both cost-effective and allow access to content that providers have not manually authored for very small devices. Many small devices use a markup language other than HTML.

Transducing is a basic automation technique that translates HTML and images into one of these other formats, which means that a client device can indirectly request the content through a proxy system, such as Mobile Google (http://mobile.google.com) or Wingman.[1] Such proxies retrieve the required content, transduce it into native formats, and compress and convert images to match device characteristics. Once transduced, a Web page appears to the user as a set of screens. Figure A3 shows an example. Most pages are too large to be transferred or rendered on the device as a single screen because of display size and network protocol limitations. Transducing systems, such as AvantGo (http://www.avantgo.com), which offers Web browsing and caching for wireless devices, can also work offline, giving users access to information when they are not connected.

Transforming

In addition to making Web content compatible with device formats, transforming systems modify content to transform the structure, or experience, of interacting with the content. The Digestor system,[2] for example, attempts to mimic an expert Web designer faced with the task of reauthoring Web pages for PDAs. It modifies the Web page layout, splitting it into multiple subpages (each better suited for the smaller display) and adding navigation links so that the user can navigate the subpages. Power Browser[3] adopts a similar approach for reading content on larger PDA-sized devices.

References

1. A. Fox et al., "Experience with Top Gun Wingman, A Proxy-Based Graphical Web Browser for the 3Com PalmPilot," *Proc. IFIP Int'l Conf. Distributed Systems Platforms and Open Distributed Processing* (Middleware 98), N. Davies, K. Raymond, and J. Seitz, eds., Springer-Verlag, London, 1998, pp. 407-424.
2. T.W. Bickmore and B.N. Schilit, "Digestor: Device-Independent Access to the World Wide Web," *Computer Networks and ISDN Systems,* vol. 29, nos. 8-13, 1997, pp. 1075-1082.
3. O. Buyukkokten et al., "Power Browser: Efficient Web Browsing for PDAs," *Proc. Conf. Human Factors in Computing Systems* (CHI 00), ACM Press, New York, 2000, pp. 430-437.

- *Link naming*. Separating links from page content also removes contextual cues that help users understand where links will lead. This is especially obvious when links have uninformative labels such as "click here." m-Links uses link-naming algorithms to improve link-label quality.
- *Unlinked data*. Much content of interest to mobile users, such as phone numbers and street addresses, is not linked using HTML tags, and therefore will not appear in a navigation view that shows only links. m-Links incorporates data detectors to extract these bits of useful information. Users can select these much like other links and invoke link-specific actions, such as placing a call.
- *Link overload*. A Web page often has more links than the device display has lines, so m-Links groups some links into categories that the device can display in a single line.
- *Unlimited content types*. The Web includes countless types of content not well suited to Internet phones. To maximize the ability to perform useful actions on any content, m-Links lets users apply multiple network-based services (similar to browser plug-ins) on any link. m-Links decides which services are available for each link on the basis of that link's attributes, such as MIME type.

In this article, we describe how we met these challenges. The technical details of the user interface and system are available elsewhere.[2,3]

M-LINKS INTERACTION

When phonetop users open a Web page through the m-Links proxy, they see a list of links from that page and can dig through the list in the same way they might dig through folders in a file dialog to select a filename.

When users find a link, they may invoke a service, analogous to right clicking on a document and using the context menu on a desktop interface. Although users can't do very much directly on the cell phone with content types such as PDF documents, MP3s, or MPEGs, m-Links users can always

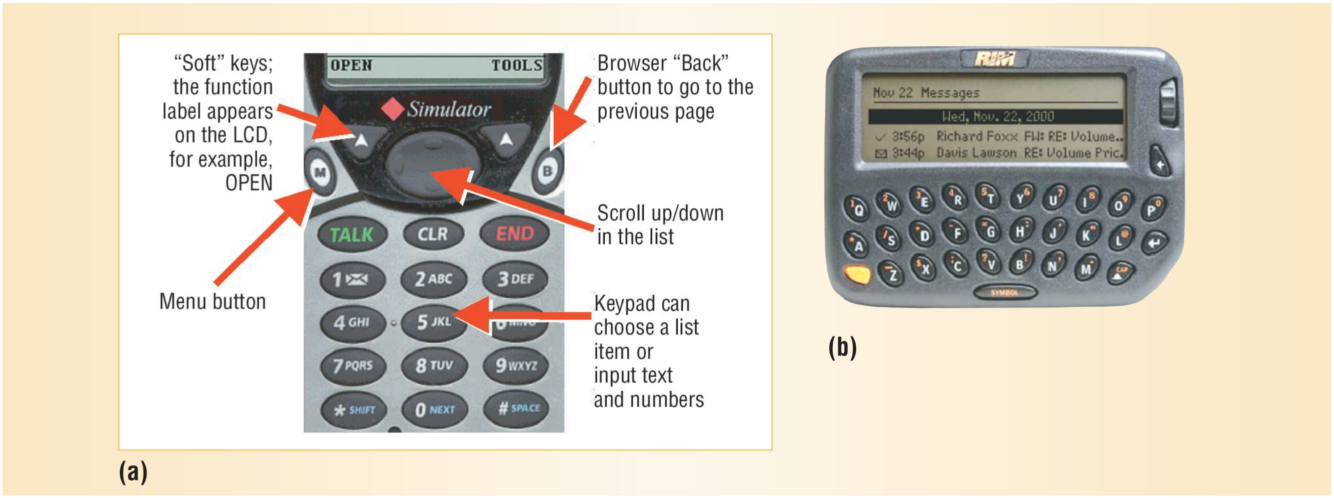
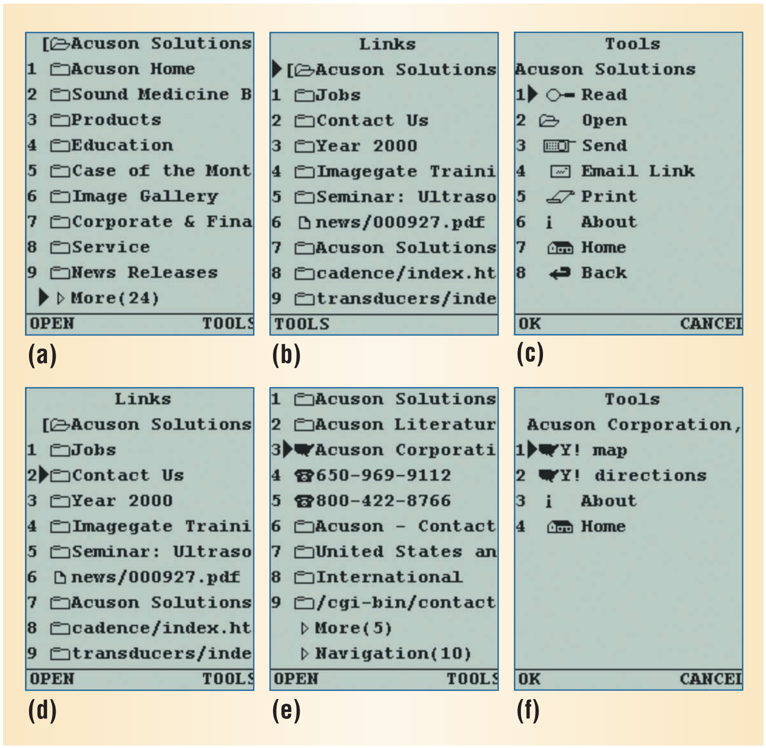

"Soft" keys; the function label appears on the LCD, for example, OPEN
Menu button
Browser "Back" button to go to the previous page
Scroll up/down in the list
Keypad can choose a list item or input text and numbers
OPEN
TOOLS
Simulator
Nov 22 Messages
Wed, Nov. 22, 2000
3:56p Richard Foxx FW: RE: Volume..
3:44p Davis Lawson RE: Volume Pric..
(a)
(b)

Figure 1. Typical very small Internet devices. (a) The Neopoint 1000 Internet-enabled cell phone and (b) the RIM BlackBerry 850 e-mail and Internet-enabled pager.



(a)
```
 [Acuson Solutions
1 Acuson Home
2 Sound Medicine B
3 Products
4 Education
5 Case of the Mont
6 Image Gallery
7 Corporate & Fina
8 Service
9 News Releases
 ▶ More(24)
OPEN          TOOLS
```

(b)
```
       Links
▶[Acuson Solutions
1 Jobs
2 Contact Us
3 Year 2000
4 Imagegate Traini
5 Seminar: Ultraso
6 news/000927.pdf
7 Acuson Solutions
8 cadence/index.ht
9 transducers/inde
TOOLS
```

(c)
```
        Tools
Acuson Solutions
1▶ Read
2  Open
3  Send
4  Email Link
5  Print
6  About
7  Home
8  Back
OK            CANCEL
```

(d)
```
       Links
 [Acuson Solutions
1 Jobs
2▶Contact Us
3 Year 2000
4 Imagegate Traini
5 Seminar: Ultraso
6 news/000927.pdf
7 Acuson Solutions
8 cadence/index.ht
9 transducers/inde
OPEN          TOOLS
```

(e)
```
1 Acuson Solutions
2 Acuson Literatur
3▶Acuson Corporati
4 650-969-9112
5 800-422-8766
6 Acuson - Contact
7 United States an
8 International
9 /cgi-bin/contact
  More(5)
  Navigation(10)
OPEN          TOOLS
```

(f)
```
        Tools
 Acuson Corporation,
1▶ Y! map
2  Y! directions
3  About
4  Home
OK            CANCEL
```

Figure 2. The m-Links interface on the Neopoint 1000. (a) The user starts with a list of links to navigate the Acuson Solutions homepage, which he reached by entering Acuson on the m-Links home page. The More item (bottom of screen) collects remaining links on the current page that would not fit on the current screen. (b) When the user selects More, m-Links displays the remaining links. (c) When the user presses the Tools soft key, the interface switches to a list of actions the user can perform on the current link. By pressing Read, for example, he could view all the text on the page. (d) Pressing Cancel returns the user to navigation mode. (e) By selecting Contact Us, he goes to Acuson's contact information page. The navigation item (bottom of screen) collects links that repeat across many Web pages. (f) The user selects Acuson's street address and presses the Tools soft key to show a list of actions. If he selects directions, for example, m-Links passes the street address to Yahoo Maps and returns the directions to the user in text form. The About and Home options let him get more information about the current link and return to the m-Links homepage.

do *something* with any content they find. For example, a user can navigate to a link to a PDF document and e-mail the PDF (or URL) to himself for later use at the desktop by selecting the link and applying the e-mail service.

Consider the Neopoint 1000 phone in Figure 1a. The phone has two soft keys, which can be programmed to execute HTTP requests when pressed. The programmed functions in this case are Open and Tools, which the phone displays on the screen immediately above their respective buttons. The user inputs text through the phone's keypad and uses the thumb pad to select an entry in a list or move the input cursor on the display.

Figure 2 shows a navigation sequence for a Neopoint 1000 user running m-Links. The sequence starts at the homepage of Acuson Solutions, a medical equipment manufacturer. Users typically start at the m-Links server homepage and enter a desired site. Because entering long URLs on the keypad takes time, users can enter just the core domain element, such as the company name. m-Links expands this by adding the "www" prefix and appending various top-level domain extensions—.com, .gov, co.uk, and so on—until it determines a valid site. It then returns the expanded list of valid URLs to the user for confirmation or correction.

Once the user selects the desired URL, m-Links removes the text on that page and displays only the links—the parts needed to support navigation. The user selects each link, represented by a closed folder, by using the thumb pad or pressing the corresponding keypad number. Selecting list items in this way is one of the few interactions that very small devices are designed for. When the Web page has more links than the phone has lines, m-Links splits the list into separate screens.

A document icon indicates links to non-HTML Web content, such as PDF or multimedia. At the top of each displayed list is the title of the Web page,

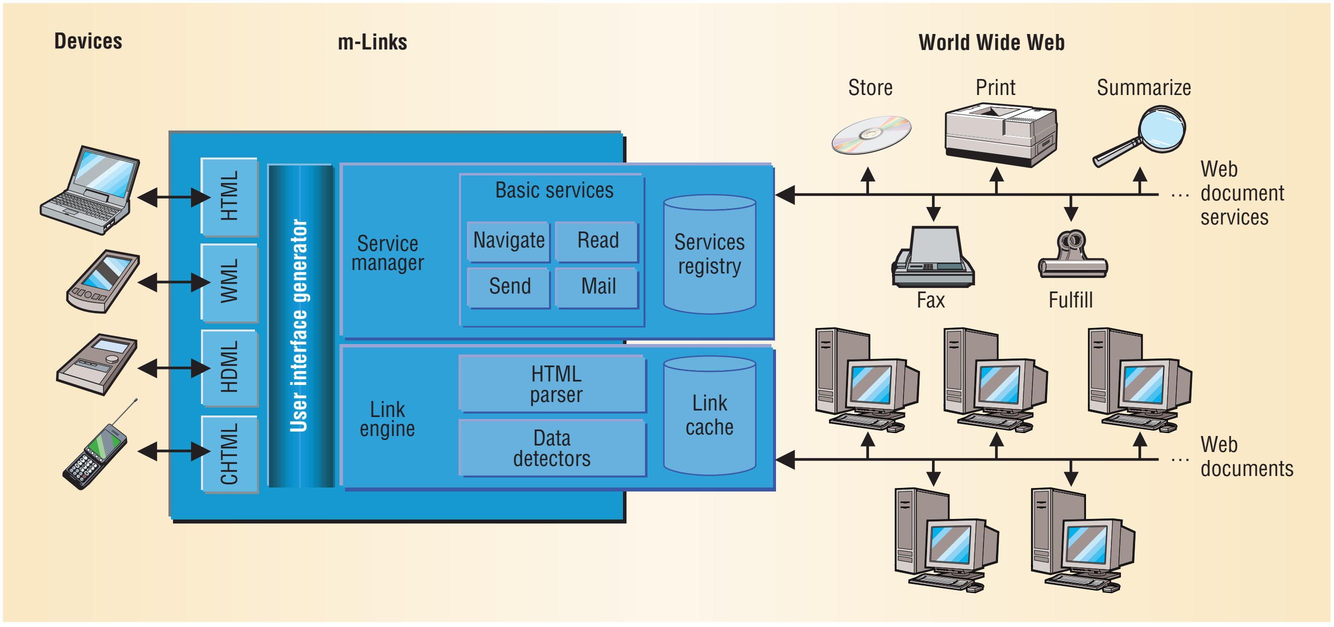


Figure 3. The m-Links architecture. The three main processing components are the **link engine***, which creates the navigation interface, the* **service manager***, which creates the action interface, and the* **multidevice user interface generator***, which converts the interfaces into forms suitable for the requesting device and browser. Formats include HTML, Wireless Markup Language (WML), Handheld Device Markup Language (HDML), and Compact HTML (CHTML).*

and next to it is an open folder. Once users select a link, pressing the Tools soft key causes the interface to switch to a list of actions they can perform on that link. Pressing the cancel soft key returns them to navigation mode.

As users continue deeper into the Web hierarchy, m-Links introduces categories for collecting similar links. One of these categories, navigation, is a repository for links that repeat across many pages on a Web site, such as Home, Products, and About Us. By collapsing repeated links, the navigation category promotes new links as users dig through a Web site.

A link is the basic unit of manipulation, but sometimes information the user wants may not be explicitly linked in HTML. m-Links solves this by scanning the text, using server-side data detectors to create new links, and then including these links in the list of links that can receive some user action.

The navigation interface exploits the user's familiarity with desktop file-selection dialogs. Satchel[4] also uses this metaphor, but it primarily accesses desktop documents; it does not operate on the Web's hypertext structure.

M-LINKS ARCHITECTURE

As Figure 3 shows, m-Links has three main parts: the link engine, the service manager, and the user interface generator.

Link engine

The link engine processes Web pages into a collection-of-links data structure that is stored in the link cache. A request to the navigation interface for a Web page involves four steps.

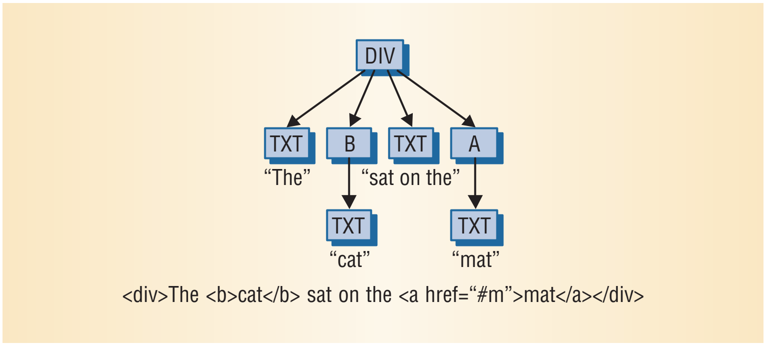


Figure 4. An HTML fragment represented in a parse tree.

Page parsing. In the first step, a parser—a variant of the Digestor's fast HTML parser—creates a parse tree that consists of the HTML tags and text on the Web page, as Figure 4 shows.

The parser also handles HTML frames, which many Web pages use extensively to partition pages into separate areas. `Frameset` and `Frame` tags describe the frame structure in HTML. Each `Frame` element points to a URL that the desktop browser fetches to populate that frame. Whenever the m-Links parser encounters a `Frameset` element, it recursively calls itself to iterate through each `Frame` element, inserting it into the tree and effectively flattening the structure. The user sees the resulting links as if they all came from the same URL, similar to the way desktop browsers present a page.

We deliberately excluded support for HTML scripting elements such as JavaScript or VBScript in m-Links for two reasons. First, scripts are often concerned with the rendering of the page. Second,

m-Links supports basic HTTP-accessible system, user, and content provider services that any site can host.

because most search engines also ignore scripting elements, site designers often provide an alternative static link.

Data detection. In the second step, m-Links invokes server-side data detectors to identify Web page elements that are not HTML anchors or tags but that could receive some user action. The detectors are functionally similar to Apple's data detectors,[5] but they run on the m-Links server, not on client devices. The current m-Links version has two detectors, one for phone numbers and one for street addresses. We will add more detectors as we identify other data types that users would like to perform actions on.

Once the link engine parses the page, it passes the parse tree to each data detector in turn. When a detector finds a match, it modifies the hierarchy to insert a virtual link or data-detection element that spans the text (and possibly other elements) containing the data. The detector assigns each data-detection element an `Href` property that contains target information about the virtual link. For an address, this is the street address as extracted. For a phone number this is a `WTAI`, a URL identifier that, when followed, lets a Web phone client dial the number.

Link naming. Once the data detectors have modified the parse tree, the link engine extracts links and gives them appropriate display names by traversing the tree looking for elements with an `Href` property (such as `<A>` or `<Area>` elements) and data-detected links. The link engine uses a link-naming algorithm to determine concise but meaningful link labels. The algorithm identifies a set of possible labels for each link and assigns each a quality value that represents how meaningful the label is likely to be. The lowest quality label is the link's URL. Because page authors generally make titles meaningful for book marking, the link engine assumes that the highest quality label is the document's title at the link's destination (for HTML pages). Other sources of label names are a link's anchor text, the alt-text associated with an image link, and the link's relative URL path (excluding the host name and so on).

When links to different documents have the same name label, the algorithm discards the label, moving to the next highest quality label in the link's label set. The link engine checks the new labels again to ensure that the new labels are unique, thus guaranteeing a distinct label for each link that appears in the final user interface.

Link categorization. Finally, in step four, after it labels links, the link engine categorizes them. Categorization essentially hides certain links from the user, resulting in a less cluttered and more compact interface. Categories can also help users quickly focus or direct their navigation actions. For example, a user who wants information about a company location or address can probably access it through a link in the navigation category. Otherwise, she would need to hunt for the link among all the other links on each page.

The current m-Links version supports two categories. The *offsite* category includes links that refer to documents at a different Web site from the document being processed. The *navigation* category holds links that a site uses throughout to navigate from anywhere to common index pages. The link engine uses page-layout heuristics to identify navigation links.[2,3]

Service manager

The service manager creates the action interface, which presents a menu of tools users can invoke once they have located a link with the navigation interface. The service manager supports compound interactions familiar to desktop users. For example, selecting a data-detected address link and retrieving directions in m-Links is analogous to a desktop user copying a street address from one Web page and pasting it into the Yahoo Maps page to get directions.

When the user selects a link in the navigation interface and chooses Tools, m-Links receives the request and passes the link to the service manager. The service manager constructs the action list by evaluating a set of rules that each service specifies against the link's attributes (such as its MIME type) and the characteristics of the user's device (such as what markup it supports). If the service manager determines that the service satisfies all the service rules, it adds the service to the action interface. Consequently, a service developed to play audio files will appear only when an audio file type is the selected link.

m-Links supports basic system, user, and content provider services—all of which are HTTP-accessible services that any site can host. The *basic system* services, which the m-Links server hosts, include reading HTML documents and sending URLs via e-mail and phone to phone. *User* services are third-party services that users would like to see for different kinds of links. Finally, content providers can include a services.xml file in the root directory and use this to include a set of customized *content provider* services. For example, http://www.patents.com could provide a service for overnight delivery of high-quality patent documents.

```
<service-group ID=”email-group”>
 <service ID=”email”>
   <rule>
     <accept match=”any”>
       <client-accepts>.*text/html.*
       </client-accepts>
       <client-accepts>.*text/x-hdml.*
       </client-accepts>
     </accept>
     <reject>
       <source-mimetype>URL/.*
       </source-mimetype>
     </reject>
   </rule>
   <execution>
       <execute>/mailto</execute>
   </execution >
   <presentation>
     <language ID=”en”>
       <service-name>email document
       </service-name>
       <short-name>Email Link</short-name>
       <description>
         Email a URL or the URL
         and its contents to yourself
         or a friend.
       </description>
     </language>
     <icon output-format=”text/x-hdml”>
         envelope1
     </icon>
   </presentation>
 ...
 </service>
...
</service-group>
```

Figure 5. A sample service specification for an e-mail service. Web site owners and users can use the service specification to introduce new services and encode possible ways to use them.

User interface generator

m-Links supports a variety of small devices with different user interface markup languages, including the Handheld Device Markup Language (HDML), Compact HTML (CHTML), and Wireless Markup Language (WML) for Web phones and HTML for most palm-sized devices. The generator uses a combination of screen template substitution and program inheritance to produce the appropriate markup interface for each device.

The user interface generator begins by identifying the type of device making the request. It then determines the appropriate type of response markup and dispatches to a markup handler. The handler, in turn, uses a screen template to help generate the content appropriate for the device. The generator uses the same process for both the navigation and action interfaces as well as a few associated screens.

The generator also supports multiple languages by using knowledge of the font encoding from the retrieved Web page to encode link labels in the font the author intended. Users can also specify the language for m-Links menus, prompts, and messages (currently English or Japanese); the generator simply chooses the interface screen template that matches those preferences.

DEFINING AND EXTENDING SERVICES

To introduce user and content provider services into m-Links, developers write a service specification document in XML. Our approach is similar to the Service Discovery Service[6] in that third-party developers can introduce new services and encode how to use them. However, our service specifications are simpler because we have tailored them to our application's specific requirements.

Figure 5 shows an abbreviated service specification for an e-mail service.

A service specification has three main sections:

- The *rule section* describes when m-Links should present the service as an option to the user, depending on the link's attributes, the user's device type, and the user's identity (e-mail address).
- The *execution section* describes what URL m-Links should execute when the user selects the service. An HTTP request with parameters that include the link to operate on and the user's e-mail address activates the service.
- The *presentation section* is in subsections for different languages. Each language element provides short and long labels to present to the user, as well as a longer description. A set of icon tags provides links to graphical elements that m-Links can also use when displaying services.

Developers can use one or more files for the service specification as well as URL references that m-Links resolves, validates, and substitutes. Thus, adding services to m-Links is as simple as submitting a URL with the service-specification file's location. m-Links then fetches and checks the description for validity and adds it to the service registry. The next time it generates a service menu for a link, m-Links checks the service rules and authorizes the service to appear in the menu.

We have experimented with several services, which we broadly categorize into reading, sending, printing, and mapping. *Reading* includes display-

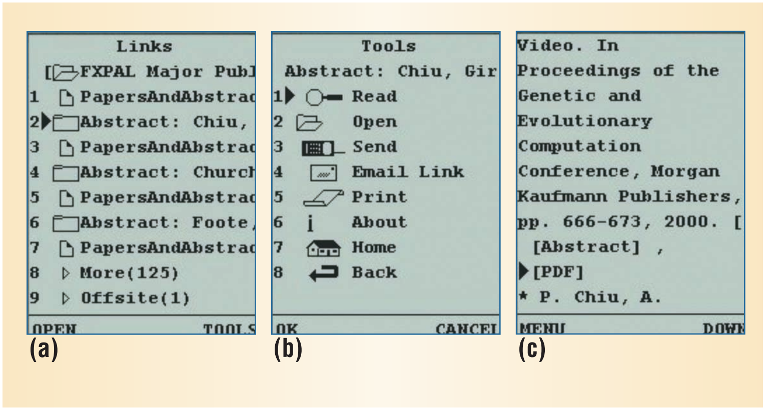


Figure 6. Tightly integrating a reading view with the navigation view. (a) The navigation interface shows links on the FXPAL publications page. (b) Selecting the Read action in the action interface (c) toggles the view to display the page text surrounding the selected link. Selecting Menu and Navigate toggles back to the navigation view in (a).

ing content. *Sending* includes moving the link or content to a user via e-mail or WAP (wireless application protocol) alerts—a messaging capability built into some Web phones that lets users send URLs phone to phone. *Printing* includes getting faxed or printed copies of content either at work or while mobile. *Mapping* includes getting directions and printing out maps to an address.

While reading services let users display content, such as PDF or HTML, we remain convinced that reading is not the primary action people want to perform on such small devices. However, we also have observed that people want to check content before proceeding with another action, such as faxing, and this checking is one of the most common reasons users employ reading services.

OBSERVATIONS

Our implementation of m-Links runs on a Java servlet engine under Microsoft's IIS Web server. We ran the system in a six-month trial on a Pentium III processor with 256 Mbytes of memory and a T1 network connection. We saw clearly in early prototyping the advantages of separating links from content, but we also saw the disadvantage of losing context and information that normally surrounds a link. This loss was especially pronounced on the data-detected phone number links—Was this a fax or a voice phone number?

To address this problem, we more tightly integrated the HTML reading service to provide the read-around feature, which Figure 6 illustrates. The Tools menu includes an option to toggle between the view of the links to a view of the text surrounding the selected link.

Portals

Portals like Yahoo or Excite can have hundreds of links, which means that finding an appropriate link can take a long time. Accommodating pages with this many links is an open problem, but we believe structural transformation systems like Digestor may perform better for these pages because these systems naturally separate page areas into distinct units for presentation. Consequently, one possible solution is a hybrid approach, in which the system collects links from separate page areas into categories much like m-Links already does for offsite and navigation links.

Browsing styles

m-Links best supports directed browsing, in which the user follows links in an attempt to find specific content. Casual browsing—following links to see if there is anything useful on a site—does not work as well. Indeed, we don't see casual browsing as compatible with very small devices anyway because of their display restrictions. However, we believe that directed browsing *plus* searching better supports the goals of casual browsing. For example, users can employ m-Links to perform actions on links that a search engine returns or to navigate into those sites.

Although users of very small Internet devices will benefit when Web content is designed specifically for their devices, it seems unlikely that content providers will be able to customize all new content, let alone existing content, in this way. Indeed, Web access for these users will undoubtedly become a more pressing concern as ever-smaller Internet devices become more pervasive.

The current version of m-Links has only the offsite and navigation link categories, but other categories are possible, such as those based on the link destination's MIME type (PDF, MPEG, or MP3 files) or layout characteristics (links separated into frames, table rows, columns, and cells). Research is required to determine when to introduce such categories and when doing so might be detrimental.

Research is also needed to streamline user input. Most users of very small devices find inputting even small amounts of text using a keypad unintuitive and frustrating. We have explored integrating the maturing technologies of speech recognition and Voice Extensible Markup Language, which would let m-Links seamlessly support mixed media input to dialogs and forms, as well as alternative output forms.

Despite these open issues, the modal interaction model underlying m-Links provides a foundation for using any Web content on very small devices—a foundation that is flexible enough to evolve as content types and user needs change. ■

References

1. T.W. Bickmore and B.N. Schilit, "Digestor: Device-Independent Access to the World Wide Web," *Computer Networks and ISDN Systems,* vol. 29, nos. 8-13, 1997, pp. 1075-1082.
2. B.N. Schilit et al., "m-Links: An Infrastructure for Very Small Internet Devices," *Proc. 7th Ann. Int'l Conf. Mobile Computing and Networking* (MobiCom 01), ACM Press, New York, 2001, pp. 122-131.
3. J. Trevor et al., "From Desktop to Phonetop: A UI for Web Interaction on Very Small Devices," *Proc. 14th Ann. ACM Symp. User Interfaces Software and Technology* (UIST 01), ACM Press, New York, 2001, pp. 121-130.
4. M. Lamming et al., "Satchel: Providing Access to Any Document, Any Time, Anywhere," *ACM Trans. Computer-Human Interaction*, vol. 7, no. 3, 2000, pp. 322-352.
5. B. Nardi, J. Miller, and D. Wright, "Collaborative, Programmable Intelligent Agents," *Comm. ACM*, Mar. 1998, pp. 96-104.
6. S. Czerwinski et al., "An Architecture for a Secure Service Discovery Service," *Proc. 5th Ann. Int'l Conf. Mobile Computing and Networking* (MobiCom 99), ACM Press, New York, 1999, pp. 24-35.

Bill N. Schilit *is a principal researcher and codirector of Intel Research Seattle, where his research focuses on ubiquitous and proactive applications, with an emphasis on information appliances and context-aware computing. Schilit received a PhD in computer science from Columbia University. He is a member of the IEEE Computer Society and the ACM. Contact him at bill.schilit@intel.com.*

Jonathan Trevor *is a senior research scientist at FX Palo Alto Laboratory, where he focuses on ubiquitous systems and computer-supported cooperative work. His current research interests are in the development of readily accessible groupware and human-computer interaction applications across a range of technologies and platforms. He received a PhD in computer science from the University of Lancaster. Contact him at trevor@fxpal.com.*

David M. Hilbert *is a research scientist at FX Palo Alto Laboratory. His research interests are in the design and evaluation of practical human-computer interaction, computer-supported cooperative work, and ubiquitous computing applications. He received a PhD in information and computer science from the University of California, Irvine. He is a member of the IEEE and the ACM. Contact him at hilbert@fxpal.com.*

Tzu Khiau Koh *is a senior member of the technical staff at Xerox Singapore's Software Centre, a division of Fuji Xerox Asia Pacific. Her research interests are in personal and mobile computing, social computing, and human-computer interaction. She received an MSc in computer science from the National University of Singapore. Contact her at kohtk@xssc.sgp.xerox.com.*