INFORMS Journal on Computing
Vol. 10, No. 4, Fall 1998

0899-1499/98/1004-0359 $05.00
© 1998 INFORMS

FEATURE ARTICLE

The World Wide Web: Opportunities for Operations Research and Management Science

HEMANT K. BHARGAVA / *Code SM-BH, Naval Postgraduate School, 555 Dyer Road, Room 214, Monterey CA 93943, Email: bhargava@nps.navy.mil*

RAMAYYA KRISHNAN / *The Heinz School, Carnegie Mellon University, Pittsburgh PA 15213, Email: rk2x+@andrew.cmu.edu*

(Received: March 1997; revised: December 1997, March 1998, June 1998; accepted July 1998)

The World Wide Web has already affected OR/MS work in a significant way, and holds great potential for changing the nature of OR/MS products and the OR/MS software economy. Web technologies are relevant to OR/MS work in two ways. First, the Web is a multimedia communication system. Originally based on an information pull model, it is—critically for OR/MS—being extended for information push as well. Second, it is a large distributed computing environment in which OR/MS products—interactive computational applications—can be made available, and interacted with, over a global network. Enabling technologies for Web-based execution of OR/MS applications are classified into those involving client-side execution and server-side execution. Methods for combining multiple client-side and server-side technologies are critical to OR/MS's use of these technologies. These methods, and various emerging technologies for developing computational applications, give the OR/MS worker a rich armament for building Web-based versions of conventional applications. They also enable a new class of distributed applications working on real-time data. Web technologies are expected to encourage the development of OR/MS products as specialized component applications that can be bundled to solve real-world problems. Effective exploitation, for OR/MS purposes, of these technological innovations will also require initiatives, changes, and greater involvement by OR/MS organizations.

The collection of concepts, languages, tools, and technologies that compose the *World Wide Web*[1] (WWW or Web[7]) is recognized to be as important and wide-reaching a development as that of the personal computer. The Web is a multimedia information system implemented on the Internet. At less than a decade old, it is much younger than the Internet and even younger than the personal computer, itself a key component in the phenomenal growth of the Web. Yet, it has already had a significant impact, more than the Internet itself did, on nearly every branch of work, including OR/MS (see [42] for a general discussion). A good example is the Web-based OR Data Library, which presents a more convenient, and easily discoverable, alternative to the earlier system of distribution via electronic mail.[4, 22]

What precisely is it, then, about the Web that has caused such an explosion of use that the basic Internet never experienced? How does it relate to OR/MS, and will, or should, OR/MS never be the same again? Specifically, what OR/MS activities can benefit from Web technologies and how can they benefit? This article addresses these questions, and has three main objectives.

1. Give the OR/MS readership a concise technical and capabilities overview of the WWW.
2. Illustrate, with inspirational examples, and explain, in terms of the underlying concepts and technologies, the potential positive impact that the Web technologies can make on the OR/MS profession—in research, education, practice, and general communication.
3. Describe a suite of Web-based technologies available to OR/MS workers for Web-enabling their applications and for setting up OR/MS products for execution in a distributed computing environment.

We begin with a short historical overview. The WWW grew from an effort at the European Laboratory for Particle Physics (CERN) aimed at sharing scientific information created by CERN scientists. The problem then was that the large number of file formats, computing platforms, access tools, and access protocols made it hard to access information on remote machines, let alone to jump from one information object to another. Berners-Lee, in a proposal[5] to CERN management in 1989 (significantly, only a year after CERN was formally connected to the Internet), suggested

[1] At the end of this article, we have included a glossary of technical terms (Appendix A) and a list of Web sites (Appendix B). To aid the reader, we use the following stylistic convention in the body of this article: Terms in the glossary, when first used, are italicized, and items for which Web-based links are available are underlined when they are first used.

Subject classifications: Computers, Information systems
Other key words: World Wide Web, OR/MS

development of a client-server distributed hypertext system to address the information management challenges at CERN. In a revised proposal[6] the following year, Berners-Lee and Cailliau proposed developing a world-wide web of information with easy readability and easy authorship. They suggested that links may point across machine boundaries and that this required solutions for problems such as different access protocols and different node content formats.

After initial development of the basic technologies at CERN, Marc Andreesen, a student at the University of Illinois at Urbana Champaign, created a graphical *browser* (Mosaic) that suddenly made the Web attractive and easy to use. Much has happened in the world of the Web since the early days. Today, the development and standardization of Web technology is driven by the *W3C* (World Wide Web Consortium), an international group of academic researchers and over 100 companies. However, from a practical marketplace perspective, Web technology is dominated by a few companies (primarily Netscape, Microsoft, and Sun Microsystems), their products (Netscape Navigator, Internet Explorer, and *Java*) and their extensions (*NSAPI* and *MSAPI*) of the general Web standards. Often, software products become available and widely used before there is much debate, analysis, or agreement about the underlying concepts. In our discussion, we attempt to remain at the conceptual level but are forced, at times, to discuss technology and concepts in terms of particular products.

The OR/MS worker is now presented a suite of tools with which to build new applications and to disseminate applications in new ways. The remainder of this article discusses these developments. Section 1 presents four vignettes that illustrate ways in which Web technologies can benefit OR/MS work. Section 2 is a gentle introduction to the architecture of the Web and the fundamental technologies underlying it. The Web is both a medium for communicating information and a distributed computer; the second perspective is crucial for OR/MS's effective exploitation of the Web. We present the tools underlying the first view in Section 3, and those relating to the second view in Section 4 and Section 5. Then we discuss emerging technologies that, we believe, will be instrumental in the use of Web-based tools for OR/MS applications (Section 6). In the conclusions (Section 7), we discuss some nontechnological issues concerned with the use of the Web for OR/MS work, and suggest that the OR/MS community should define and adopt policies and organizational changes that facilitate the use of the Web for OR/MS.

1. WWW and OR/MS: Vignettes of Opportunity

In this section, we present four illustrations of the role of the Web in OR/MS community use, education, practice, and research. All examples, even when they resemble and are inspired by real-world objects, are hypothetical. After presenting each vignette, we discuss, in an informal way, the enabling Web technologies and methods. Technical terms used in the examples are described in the sections that follow, and listed in the glossary.

Vignette 1: OR/MS Community Use of the Web

John, a Ph.D. candidate, is looking for a university job. Using his Web browser, he visits the placement service of INFORMS Online. There, he registers as a candidate and enters biographical information (including his *URL*) into a form; this information is added to the placement database and becomes visible to potential recruiters. John then searches the employment database. Four listings match his criteria. He follows up one at UCLA, and reads Web pages about UCLA's faculty, research programs, and professional programs. He then uses his Web browser to send electronic mail to the contact person at UCLA. The message includes URLs of his papers and his detailed résumé. At UCLA, Professor Jones—chair of the recruiting committee—receives this Email, views it using her Web browser, and has immediate access to John's résumé, publications, and other information.

Analysis. Although this scenario is not based on a true story, the technologies that enable it certainly exist. As with all Web applications, it requires participants to possess certain *client* and *server* technologies. These clients and servers allow the transfer—using the protocol best suited to the task—of information in many forms: text (formatted in *HTML*), graphics (encoded in the *GIF* format), sound (encoded in the *WAV* format), and video (encoded in the *MPEG* format). John, when he visits INFORMS Online, and Professor Jones when she views John's pages, must have a standard *HTTP* client (a Web browser). Because the browser can also serve as an *SMTP* and *IMAP* client, it can also be used to send and receive email. INFORMS Online uses HTTP server technology to serve both static pages and dynamically generated HTML pages. Because the placement service allows ad hoc queries, as well as the capture and storage of user data, it must have Web-database connectivity, achieved here via a *CGI* call to programs that implement *SQL* queries.

Vignette 2: Education

Professor Ram is discussing the role of sensitivity analysis in decision support systems, and Susie, a student, asks about analysis tools that use visualization and animation. Professor Ram, with her classroom computer, uses a popular Web *search engine*—*Lycos*—to locate relevant Web sites. Eventually, she clicks on the URL of an *applet* for traveling salesman problems. The *Java applet* gets downloaded to her machine, begins execution, and projects the display on her screen (Figure 1). Starting with initial city locations, it plots regions in which each city may be moved without affecting the optimal solution, and jumps to a new solution when the changes go outside these bounds. Thus, Susie is able to understand the impact, on the final solution, of changing elements of the problem data.

Analysis. The proliferation of information on the Web quickly led to the creation of general-purpose document search engines such as Lycos. Lycos, for example, catalogs millions of documents served by hundreds of thousands of Web sites from around the world. Lycos users, such as Professor Ram, need only have a standard Web browser, and the Lycos HTTP server provides access to the search

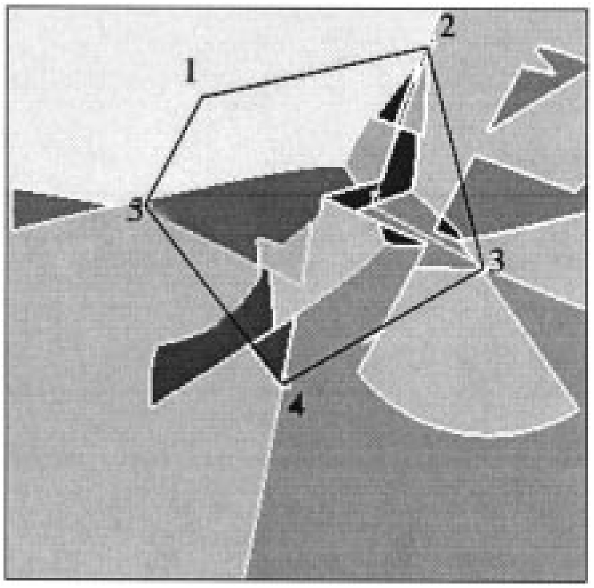


Figure 1. Java applet for animation of sensitivity analysis for the traveling salesman problem. Users can move each city within its region, without causing a change in the optimal tour; if a city is moved outside its region the applet computes and displays the new optimal tour. From Jones' *LP Animation* applet.

tools and database via a CGI call. The traveling salesman problem animation is implemented as a Java applet, a special type of mobile software application that can execute on a *Java Virtual Machine* (*JVM*) built into the Web browser.

Vignette 3: Research

Huang, a Ph.D. student, had to solve a complex engineering design problem that was formulated as a nonlinear program. His university did not have access to either the solution algorithms he wanted or the computational platform (a high-end workstation) on which to run them. Professor Ram, his advisor, suggested that he use NEOS (Network-Enabled Optimization System). Huang chose to solve his problem with Lancelot, a nonlinear programming solver at NEOS. Downloading the submission form (an HTML form; see Figure 2) on the NEOS site, he entered problem-specific information (i.e., number of variables, number of constraints, etc.) and the URLs of his FORTRAN routines (these defined the initial starting point, upper and lower bounds on variables, general constraints, and so on). He filled out the form, and submitted it to the NEOS server, which executed the solver and returned him the results.

Analysis. NEOS,[21] from the Optimization Technology Center (a joint program between Northwestern University and Argonne Laboratories), is an excellent example of the impact that the Web is likely to have on OR/MS. NEOS gives remote users the ability to execute specialized computational methods on high-performance servers. Huang needs to have a Web client as well as an HTTP server (to make Huang's FORTRAN routines accessible to NEOS). The client is used to process the HTML forms, submit parameters, interact with the NEOS server, and obtain the results of computation. The computational algorithms need not run on the NEOS server itself because NEOS can direct the request to registered remote servers maintained by the algorithms' developers anywhere on the Internet.

Vignette 4: Practice

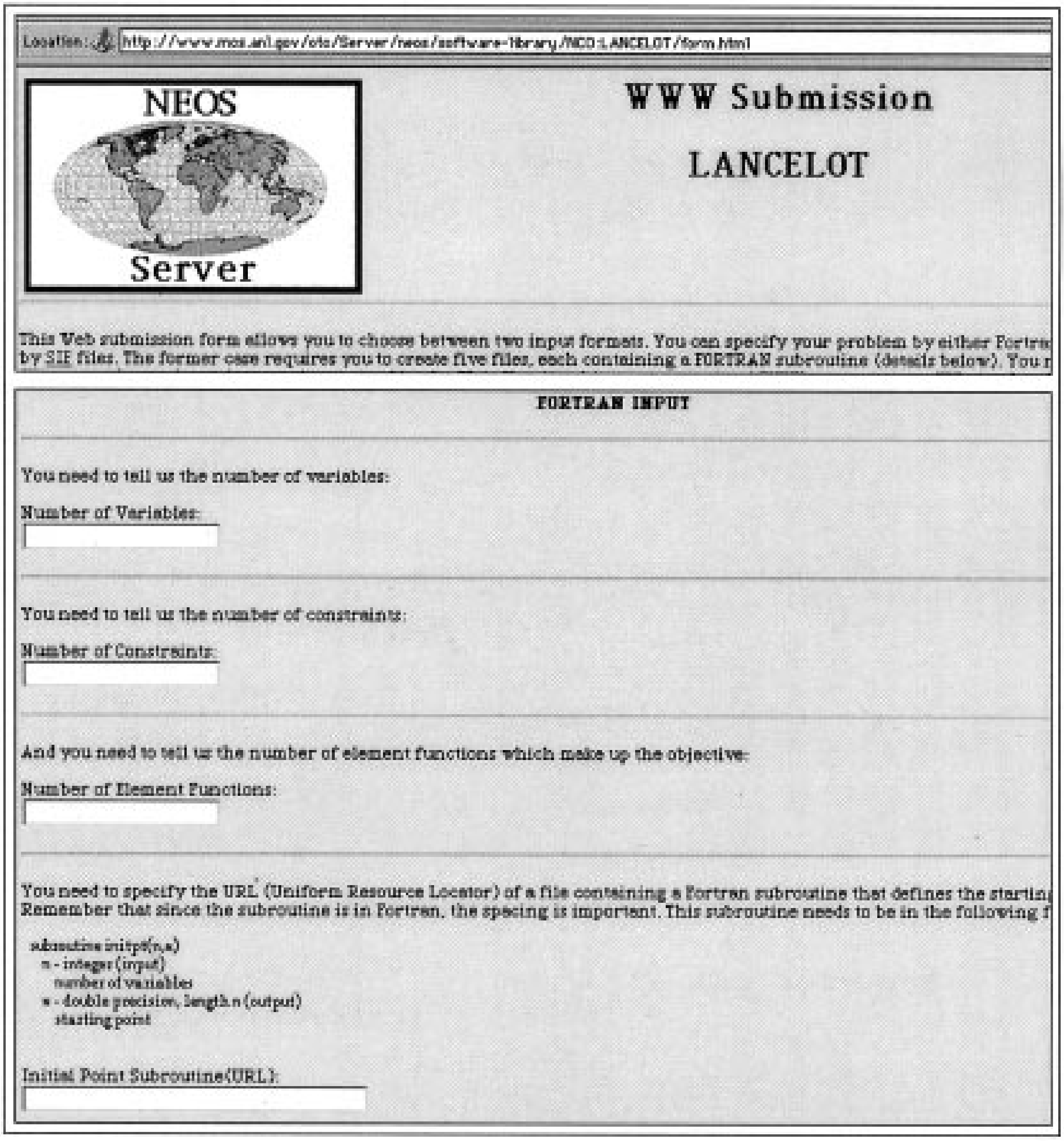
Financial consultant José needed to conduct fund portfolio analysis for Betty, an important client. After examining Betty's investment goals, José pointed his browser to FinNet, a hypothetical electronic brokerage for financial data and analytical tools. On FinNet's *yellow pages*, José found useful tools such as data on mutual fund families, an efficient frontier analysis tool for comparing alternative portfolios, and a visualization tool for displaying the results of the analysis. He created a FinNet *script* to (a) retrieve historical performance data from the specified mutual fund data servers, (b) format the data to feed the efficient frontier analysis model, and (c) create, using a visualization server on FinNet, a series of overlaid graphs depicting the efficient frontiers. FinNet executed the script and returned the report to José, who paid for use of the services and briefed Betty using the results of the analysis.

Analysis. Although this may appear futuristic, many components required to create such electronic markets and brokers for OR/MS products are available. José's Java-capable browser gets him access to FinNet via a Java applet that implements the user interface. The applet (referred to as the FinNet agent) obtains, using an object request broker (*ORB*), the services of several remote objects such as data servers, model servers, and visualization tool servers. The FinNet services required by the applet are implemented as *CORBA*-compliant objects,[39, 41] i.e., they are implemented and registered with an ORB.

Discussion: What does the Web Mean for OR/MS?

From an OR/MS point of view, the WWW (and the Internet) can be thought of as a medium and as a computer. The first perspective—illustrated in Vignette 1—is quite common and found in a majority of Web applications, and in existing OR/MS services such as INFORMS Online (we discuss these applications in more detail in Section 3). Here, the Web is used mainly as a communication medium to reach a large dispersed community, via one–many mass communication of static information (e.g., a professor's home page). The basic technologies underlying the Web (which we discuss in Section 2) are sufficient for most applications in this category. In addition, customization of information and many–many communication can also be achieved with minor extensions and little programming (e.g., chat rooms or discussion groups).

However, the full potential of the WWW is realized only when we think of the Web as a computer. That is, users accessing OR/MS Web sites or OR/MS-enabled Web sites not only obtain information about OR/MS or the OR/MS-enabled application but are able to interact with computational OR/MS products available on these sites. This, of course, requires online interactive computation. This can be achieved in several different ways ranging from simple CGI-



Location: http://www.mcs.anl.gov/otc/Server/neos/software-library/NCO:LANCELOT/form.html

NEOS Server

WWW Submission

LANCELOT

This Web submission form allows you to choose between two input formats. You can specify your problem by either Fortra
by SIF files. The former case requires you to create five files, each containing a FORTRAN subroutine (details below). You r

FORTRAN INPUT

You need to tell us the number of variables:

Number of Variables:

You need to tell us the number of constraints:

Number of Constraints:

And you need to tell us the number of element functions which make up the objective:

Number of Element Functions:

You need to specify the URL (Uniform Resource Locator) of a file containing a Fortran subroutine that defines the startin
Remember that since the subroutine is in Fortran, the spacing is important. This subroutine needs to be in the following f

```
subroutine initpt(n,x)
  n - integer(input)
    number of variables
  x - double precision, length n (output)
    starting point
```

Initial Point Subroutine(URL):

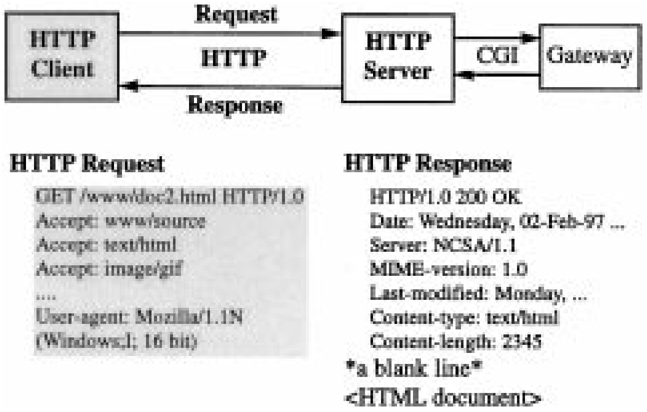
Figure 2. The submission form for the Lancelot solver on NEOS. The figure displays fragments of the form corresponding to the FORTRAN method of submission.

based architectures to more robust architectures that use distributed object technologies such as CORBA and competitors such as *DCOM*[18] and the *Java distributed object model*.[2] These were illustrated in Vignettes 2 through 4. We discuss this Web as a computer perspective, the enabling technologies, as well as their applications in OR/MS, in more detail in Sections 4, 5, and 6.

2. The WWW: An Architectural Overview

The Web, like the proverbial elephant in the old Indian fable "Five Blind Men and an Elephant," is different things to different people. For most users, it is a distributed, hypermedia repository of information. For creators of information, it is a publishing medium that offers high speed and wide reach at low cost. At least two other perspectives are valid: the Web is a digital library and, as a medium for electronic commerce, it is an electronic marketplace.

From an architectural point of view, though, the Web is a massive global network of client and server nodes that exchange information using the *hypertext transfer protocol*, which itself uses the Internet's underlying *TCP/IP* protocols (see [44] for a detailed discussion of TCP/IP). The widespread use and popularity of the Web is due, in part, to the existence of browsers. Browsers are client programs that interpret information encoded in a standard language (HTML), exploit its hypertext capabilities, and provide a



HTTP Request

```
GET /www/doc2.html HTTP/1.0
Accept: www/source
Accept: text/html
Accept: image/gif
....
User-agent: Mozilla/1.1N
(Windows;I; 16 bit)
```

HTTP Response

```
HTTP/1.0 200 OK
Date: Wednesday, 02-Feb-97 ...
Server: NCSA/1.1
MIME-version: 1.0
Last-modified: Monday, ...
Content-type: text/html
Content-length: 2345
*a blank line*
<HTML document>
```

Figure 3. World Wide Web: Architecture and Components. The client's request for a document (labeled HTTP Request) specifies the document being requested and types of documents that the client can accept. The server may return a stored document or one that is generated by a call to an external program. The server's response (labeled HTTP Response) identifies the document type and contains the document content.

point-and-click interface and transparent access to all Internet protocols. Figure 3 describes this basic architecture.

In the remainder of this section, we discuss components that define the basic Web architecture and that have been a part of the Web from its early days. This basic architecture, however, is quite limited from an OR/MS perspective. Fortunately, the technologies that make up the Web today are the result of continued innovations by various groups including W3C, academic researchers, and several commercial enterprises. These innovations make possible a wide variety of OR/MS-related applications on the Web, and are discussed in the next three sections.

2.1 Hypertext at the Network Level

Hypertext is a concept derived from Bush's proposal for a memex system.[17] It involves the organization of vast amounts of knowledge, and navigation of it, via a nonlinear collection of information and links. The earliest implementations of hypertext included Nelson's XANADU project,[38] OWL's GUIDE software,[30] and Apple's HyperCard software.[1] Until the emergence of the Web, hypertext features had to be implemented at the application level. A recognized research problem and goal was to implement hypertext at the system (or machine) level so that information objects in one application could be linked to objects in another application in an application-independent manner.

The Web's innovation regarding hypertext is that it advances the concept of hypertext to the network level. The Web allows object linking across any machines on the Internet network, independent of the operating system or hardware platforms involved. Link locations are identified using a unique address called the Uniform Resource Locator (URL). Hypertext has obvious value to designers and users of OR/MS applications. Benefits of hypertext for decision support were demonstrated at the application level by Bhargava, Bieber, and Kimbrough,[8] and included capabilities such as the automatic generation of links across various model components in a modeling system. Via the Web, such benefits are now available in a global distributed computation setting.

2.2 Hypertext Markup Language and Multimedia

What makes the Web click? Its popularity stems from the simplicity of its hypertext delivery vehicle—HTML.[29] HTML permits authors to write, format, and link text. In addition, through a special set of tags (e.g., IMG, applet, object), it permits developers to specify meta-information (e.g., URL) about nontextual digital media (e.g., sound, images, movies, animations, applets), encoded using standard *MIME* (multipurpose Internet mail extension) formats.

Consider the following code fragment.

```
⟨p ALIGN=CENTER⟩
⟨IMG SRC = “welcometoORMS.gif”⟩
⟨/p⟩
```

This code fragment illustrates the use of *HTML as container* for nontextual media. Note the reference to an image file in GIF format. This HTML feature enables delivery of multiple media types on the Web. In Section 3, we further discuss MIME formats and the concept of an OR/MS media type that would allow OR/MS to exploit HTML's property as a container for diverse media types.

HTML—in its role as the user interface definition language for Web-based applications—has several limitations regarding its use in OR/MS and in other computational applications. For example, it offers limited constructs (single-level menus, input fields in forms, and submit buttons) for user interaction. It also restricts an author to a few tools for controlling the position of objects on screen. These limitations are significant in the design of many OR/MS applications that involve a lot of user interaction (e.g., animations, as in Vignette 2), and that require complex visual representations both for user input and for display of outputs.

2.3 Server and Client (Browser) Software

Web server software allows computers to become providers of information on the Web. The information provided by Web servers can either be retrieved from prestored files or generated on request via the execution of some external program. This latter capability is important from an OR/MS perspective (see Section 2.5 and Section 4.1.1).

Web browsers, or clients, are programs that allow computers to request information from Web servers. On receiving a request from a client, the server delivers the requested information and supplies its MIME format. The browser interprets this information on the basis of its MIME format. If necessary, it selects a *viewer* (e.g., a streaming-video player or a spreadsheet program) to display this information. Again, this feature is useful in providing computational OR/MS content on the Web (see Section 3 for specific illustrations).

The use of MIME formats and the standardization of markup (via HTML), when coupled with Web client and server software, gives the WWW its two other important characteristics: universal readability (anyone equipped with a Web browser can read the documents) and universal authorship (anyone with access to a Web server can be a publisher of information).

2.4 Hypertext Transfer Protocol

The hypertext transfer protocol (HTTP) defines the set of rules that Web clients and servers must use to communicate. The communication itself takes place over a TCP/IP connection, where the TCP/IP protocols are responsible for reliable data delivery.

HTTP is optimized to handle certain kinds of transactions. In these typical Web transactions, a client makes an HTTP request (using the URL of the document) and the server returns the file. This process repeats with the client making another HTTP request to another server and so on. For each request, a TCP/IP connection is established and held open until that request is completed. This latter property of holding the connection open until the request is satisfied is referred to as the connection-oriented nature of HTTP. Because each request to a server is, in principle, a new one, HTTP servers (unlike, say, an FTP server) need not maintain any memory (or state information) about the request; this makes the protocol stateless. Finally, because only clients may initiate requests, HTTP is considered a directional protocol.

Thus, a critical assumption underlying HTTP is that sequential requests between a client and various servers are independent. Under this assumption, HTTP's connection-oriented and stateless properties are optimal. However, this optimization results in the protocol being inefficient when handling other kinds of transactions, particularly those involving computational applications (in which, typically, successive requests are not independent). The implications of these properties for OR/MS applications, and solutions to the problems that are created, are discussed further in Section 4.1.3.

We note that our discussion pertains to Version 1.0 of HTTP, which is the predominant version in use today. A newer version (HTTP 1.1) addresses some of the limitations (such as persistence of connections) discussed above.

2.5 Execution of External Programs

In response to some client requests, usually those made by filling out and submitting HTML forms, an HTTP server may invoke an external program called a script. The common gateway interface (CGI) defines an application programming interface (*API*), and specifies how data sent from the client to the server are passed to the script and how the results of executing the script are passed back to the server. The CGI approach is extremely powerful and flexible because, in principle, it allows a Web server to interface with any program written in any language (see Figure 3). We discuss CGI, and its applications in OR/MS, in more detail in Section 4.1.1.

2.6 Related Tools

Apart from the core technologies discussed in this section, the Web technology suite includes related tools and ideas. Search, cataloging, and indexing agents (see e.g., the Harvest system[15]) are programs that address the proliferation of information on the Web. Web spiders and worms are computer programs that query thousands of Web servers, index the documents available at these servers, and make the index available to users through search engines. Development and authoring tools (e.g., Microsoft's FrontPage editor) assist users in creating Web pages and Web sites. Accessory software for Web servers extends their basic functionality, e.g., by monitoring and summarizing usage and traffic.

3. Exploiting the Web for OR/MS: The Web as Media

A majority of Web-based applications is based on the concept that the Web provides a new medium for communication. Some applications even make excellent use of the unique features that the Web offers as an electronic medium. For example, INFORMS Online is an application that is primarily based on the Web as media idea. In addition to electronic versions of publications (such as OR/MS Today and the INFORMS conference bulletins) that were—and still are—available in physical form, the site contains a wealth of additional information (e.g., pointers to teaching materials) of considerable value to the OR/MS community. Further, the Web versions of even the standard publications offer unique and useful features. A good example is INFORMS Online's conference bulletins, which can be searched efficiently in many ways (by author, subject, session), and give users multiple views of relevant information. Finally, functionality such as online registration to an INFORMS conference, can be bundled along with the communication.

This section examines the Web as a communication medium and its uses in OR/MS. We organize the discussion along two dimensions—the alternative ways in which the medium could be organized and the formats (or data types) of information that can be communicated via the Web.

3.1 Push vs. Pull

The Web is, inherently, a user pull medium. Web-based information is available at all times and needs to be downloaded on command (i.e., pulled) by the reader. This is unlike, say, a radio or television broadcast that is sent to all potential receivers, and is transient. The act of requesting information from a Web site is similar to tuning in to a broadcast. In addition, with the use of HTML forms, Web readers can send information to the sender, making the medium interactive.

In this user pull model, users must discover information sources either by searching the document space on the Web (using a search engine such as Yahoo or Lycos) or by going to focused gateways (e.g., INFORMS Online or a journal's current contents pages). Further, users must remember to perform these searches with suitable regularity. Thus, obtaining relevant information can consume a considerable amount of user resources.

An alternative model that has begun to emerge is Webcasting (or information push). In typical applications of Webcasting, users locate relevant sites and register their interest profiles. Following this, the information server itself sends relevant information, at suitable times, to users without waiting for additional requests. This approach has been used to deliver news, stock quotes, and titles of journal articles.

Two leading examples of Webcasting technology are Pointcast and Netscape's Inbox direct service. Pointcast requires users to download and install (free) client software that communicates over a proprietary TCP/IP-based protocol with Pointcast servers. Publishers make their information available using Pointcast servers. Users register their interest profiles with the Pointcast client on their desktop. This client then initiates requests on the users' behalf and displays the results. Netscape's Inbox direct service does not need any new client. Interest profiles are registered at the server (versus the client in Pointcast). Based on these profiles, the server filters information from a set of Web sites and sends relevant information to users via electronic mail.

How might Webcasting be of use in OR/MS work? Consider, for example, registering your areas of interest with the INFORMS conference bulletin server and having the server send you the URLs of matching sessions. Similarly, in Vignette 1, the Ph.D. student could use Webcasting technology to have information about job postings (or candidates) that match his needs automatically sent to him.

We point out that the push technology is not limited to disseminating static information and news. It can be used to draw the attention of a user to the results of intensive data mining or computation that might trigger action. For example, one could imagine subscribing to an agent that processes large stores of financial data in the background and pushes results of its analysis tailored to interest profiles (e.g., risk profile) of users.

In closing, we note that the Web has also been used to support other community-oriented services traditionally available on the Internet. For example, Internet bulletin boards are approximated by chat groups on the Web, allowing for many–many communication. *Threaded discussions*, integrated with hyperlinks to obtain additional information, are also available on the Web.

3.2 Multimedia

A distinctive feature of the Web is the seamless way in which multimedia content such as audio, video, images, and other digital content can be integrated with text. The display and transport of this multimedia content are enabled by MIME formats. MIME[14] defines a standard way of formatting and encoding data that are exchanged between Web servers and clients. This formatting is accomplished using a collection of header fields (e.g., Content-Type, Content-Transfer-Encoding). These header fields use a collection of labels (e.g., application, audio, and multipart with Content-type) that convey information about the content and the structure of the message. A fragment of the response from a HTTP server containing Content-Type declaration of text/html is shown below.

```
HTTP/1.0 200 OK
  Date: Wednesday, 03-Apr-97 23:04:12 GMT
  Server: NCSA/1.1
  MIME-version: 1.0
  Content-type: text/html
```

These headers and labels are processed by MIME-aware clients. In the case of text/html, the Web browser knows it can display this MIME type. These capabilities can be used to deliver multimedia enabled OR/MS applications.

Although any kind of information can be digitized, a wide variety of encoding formats are available. For example, sound can be encoded in WAV format or in a streaming audio format such as *real audio*. The server needs to inform the browser about the format of the data being sent in response to a request in order that the browser may correctly interpret it. For example, if the sound file being downloaded is in WAV format, the browser could invoke a WAV viewer to playback the file. WAV is a subtype of the audio MIME format and corresponds to a particular way of digitally encoding audio. This method of delivering multimedia content relies on the following.

- *Creator/Generator of Data in Appropriate MIME Format*. These programs are used by publishers of information. For example, a user wanting to disseminate the results of an LP run as a GIF bar chart should have a program that can format this graphic in the desired format. Further, the server should know the MIME type of the information in order to supply it (as the value of the Content-Type header) to a client who wants to download and display the document.
- *Viewer of Data in Appropriate MIME Format*. The Web browser must either have the capability to display the MIME type (e.g., an image/GIF file) itself or with the help of a viewer, or be able to download a viewer of the appropriate type. Technologies such as *plug-ins*, *ActiveX* controls, and Java applets make this possible, each differently and with its own set of advantages and disadvantages (see Section 4 for details).
- *Transport of Data in MIME Format*. HTTP provides the transport from the server to browser.

This ability to deliver data of different MIME types is already being used by the OR/MS community and presents new opportunities for delivering OR/MS content. Here are some possibilities.

- *Make Viewers of OR/MS Content Available as Plug-ins, ActiveX Controls, or Java Applets*. Just as Adobe and Microsoft have made available free PDF (portable document format) and Powerpoint viewers, respectively, one could create executors (or viewers) of OR/MS content. For example, Excel spreadsheets may be easily disseminated over the Web and shared with users who have either the entire Excel application or the Excel Viewer made available by Microsoft. Another example is MathSoft's MathCad, a system for publishing and sharing mathematical models via the Internet. Similarly, OR/MS technology developers could provide—free of charge—viewers for their OR/MS products (implemented as e.g., applets, ActiveX controls, plug-ins, or *helper applications*), while selling the full-featured software for developing content.

- *Exploit Multimedia to Deliver Results of OR/MS Analysis.* Animations (e.g., simulations of manufacturing flow shops) can be a powerful way of disseminating content. Given the availability of viewers, particularly those that can be dynamically loaded, one could deliver results of complex analysis over the Web. If needed, MIME types customized to the needs of OR/MS applications may be developed and registered with the Internet Assigned Numbers Authority.

4. The Web as Computer: Publishing OR/MS Computational Products

Although the view of the Web as a medium encompasses several ways in which the Web can facilitate OR/MS work, one must expand this view—to that of the Web as a computer—to make fuller use of Web technologies for OR/MS work. In the following discussion, we use the phrase "publishing OR/MS applications on the Web" to mean giving access (to OR/MS products such as solvers, model schemas, and modeling languages) to users who are remote and on heterogeneous computing platforms. The question we seek to answer in this section, from the perspectives of OR/MS practitioners, educators, and researchers, is: How can I publish, in ways that overcome the limitations of the basic Web technologies, my OR/MS application on the Web?

From a technical perspective, the challenge is delivering a computational application in a distributed and highly heterogeneous computing environment, while preserving the Web's universal readability property. The Web delivers platform independence when static information is being communicated. However, the heterogeneity of client platforms on the Web makes it undesirable to transfer (i.e., download to client machine) OR/MS computational products (e.g., solver implementations) to users.

At the extreme, there are two ways to deliver execution of computational products via the Web: server-side computation (e.g., CGI scripts) and client-side computation (e.g., Java applets). In each category, there are methods (e.g., CGI, Java) that are, in theory, consistent with the Web's platform-independence principle. Other methods—including those involving plug-in components, viewers, ActiveX controls, or various scripting languages[40]—are also important though not necessarily platform independent. The technologies, and their advantages and disadvantages, are summarized in Table I.

4.1 Methods for Server-Side Computation

In methods for server-side computation, the HTTP server invokes an external program (usually resident on the server) upon receiving a request from a Web client (see Figure 4). The motivation behind server-side computation is that program execution, because it occurs on the server, is independent of the user's (client's) computing platform. This not only frees users from software installation and recompilation (or worse, code modifications and debugging), but facilitates sharing of specialized hardware and software technologies for solving OR/MS problems. The programs could be preexisting code written in any language that executes on the server, as long as the programs have input–output communication with the server software. Computing is done entirely on the server, and only data are exchanged between the client and server. A good example of this approach is the NEOS site that provides access to the AMPL modeling system and a variety of solvers.

Two methods are discussed in this category in Sections 4.1.1 and 4.1.2. Because both essentially rely on HTTP for data transport, they suffer the same limitations discussed in Section 2.4. We discuss methods for overcoming these limitations in Section 4.1.3.

4.1.1 Common Gateway Interface

In his initial proposal for the WWW, Berners-Lee[5] recognized the need for diverse computation. He proposed a common gateway interface that would allow a Web server to invoke other computational programs and to transfer to them data submitted by a Web client. The client's input interface is, typically, an HTML form that admits a variety of static formats for textual input. For example, Figure 5 displays the input (and output) interface used in a Web-based optimization system for solving waste disposal and recycling problems (Bhargava and Tettelbach[12]). Each form has, at most, one action tag that specifies the URL of its CGI script. The Web browser, through the HTTP server, communicates user inputs (in a format that maps input values to script variables) to the CGI script.

The script itself contains a list of variable definitions and some code to manipulate the input values. In simple Web applications, a CGI script could handle the entire computation. However, from an OR/MS perspective, it is more appropriate to think of a CGI script as an auxiliary program (also known as a wrapper) whose purpose is to: (a) gather, parse, and format the input data as required by an existing OR/MS solver, (b) initiate solver execution, and (c) obtain solver results and format them for the browser (typically as HTML text supplemented with images). In the waste disposal and recycling system, the script is actually a collection of CGI programs that execute various tasks as displayed in Figure 6. The model manipulation and solution, as well as data management, are done by external programs.

Thus, CGI provides a simple mechanism to extend the capability of the Web server and to migrate an existing command line-oriented desktop OR/MS technology (e.g., the GAMS[13] modeling system) to the Web. However, this simplicity comes with certain limitations. We discuss three limitations, the last two being particularly relevant to OR/MS work.

1. CGI implementation is customized to the features of the operating system on the Web server. For example, on Unix systems, data from the browser are passed through Unix environmental variables and results are returned through standard output. In contrast, on Windows NT systems, the script is launched as an application process

Table I. Technologies that Permit Use of the Web as Computer

Technology	Advantages	Disadvantages
CGI	Defines standard API CGI scripts can be written in a variety of programming languages	CGI scripts are invoked as processes Inefficient execution, high overhead
Server-side Scripting	Efficient execution (as threads) of scripts Support for development of gateways to external programs and databases	API is proprietary Lack of portability across Web servers
All Server-side	Shared access to resources (e.g., high-end server machines, quality software) Enables legacy programs to be accessed over the Web Client-side platform independence	Limited user interface (HTML forms) Server is called for every operation Dependence on network
Client-side Scripting	Executed within (ubiquitous) browsers on most client platforms	Not suited for complex programs Source code available to user
Plug-ins	Software components downloaded on demand Components remain on client machine Components are compiled binaries and execute efficiently	OR/MS plug-ins must be developed Components are platform specific Can only be invoked within browser Proprietary Netscape architecture
Java Applets	Platform independence (via JVM) Compile once, run anywhere model Powerful client-side capabilities (User interface and computation) Interoperable with remote components	Not suited for complex programs Programs must be converted to Java Applets need to be downloaded each time Inefficient execution
ActiveX Controls	Software components downloaded on demand Controls remain on client machine Linkable to other MS office controls on desktop Can integrate with Java applets and VBScript Controls are compiled binaries; hence execute efficiently	Require development of OR/MS controls Controls are platform specific High overhead during initial download Mainly a Windows technology
LiveConnect	Allows integration of client side scripting, Java Applets, and Javascript	Works only with Netscape browsers
All Client-side	Shifts processing from server to client Can build graphical user interfaces Makes client a more intelligent platform	Consume client computing resources Applications must be rewritten

The first two technologies (CGI and server-side scripting) involve computation on the server machine. The next five technologies involve client-side computing.

and data are passed through temporary files. This means that a CGI script developed for use on a Unix-based Web server will need modification before it can be used on an NT-based Web server. To keep this limitation in perspective, we note that it implies only server-side (and not client-side) platform dependence, and is relevant only when the application needs to be moved to a different server platform.

2. Each use of a CGI script requires the creation of an operating system process.[45] Multiple calls to invoke CGI scripts can consume considerable resources on the server. This limits the number of simultaneous requests that can be processed, particularly for computation-intensive OR/MS applications. Further, if the application is interactive and requires user input in a multistep sequence, then several input forms and scripts must be designed. Each step requires transport of data from the client to the server and invocation of a script.
3. The script-form interface of CGI is limited by the features of HTTP. It gives the user no control options other than to wait for the result or to abort the request. This presents a problem for OR/MS applications that need to provide

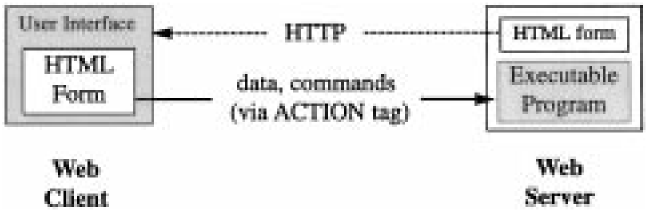


Figure 4. Server-side computing on the Web.

interactive capabilities to an end user or algorithm monitoring and control functions to an analyst.

4.1.2 Server-Side Scripting

In the typical client–server transaction using HTTP, the server retrieves and serves the requested HTML document without any regard to its content. The server-side scripting approach extends this basic model. Now the HTTP server (more specifically, the scripting engine on the server), on retrieving an HTML document, examines it for certain tags (the server tag in *JavaScript*) that identify special script statements. On finding these, it executes them and embeds the execution results into the document before passing it on to the client. The server-side scripting approach is exemplified in Netscape's server-side JavaScript technology[25] and in Microsoft's Active Server technology.[31]

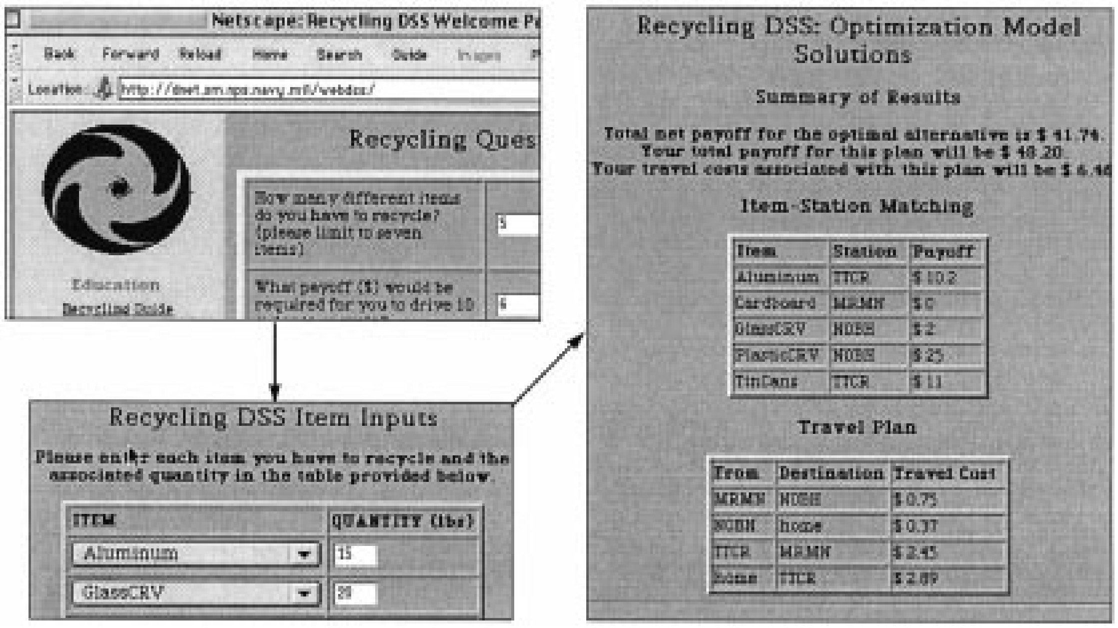
Script statements can, for example, manipulate some data, invoke an executable program, and provide connectivity to a database. Existing OR/MS applications can be published over the Web in this way: script statements communicate with the OR/MS code and the user interface is defined by HTML forms. Here, the interface to external programs is an application programmer interface (Netscape's NSAPI in the case of JavaScript) that is typically more efficient. NSAPI-initiated requests execute as *threads*[45] in contrast to CGI scripts, which are executed as processes.

Thus, compared to CGI-based scripts, server-side scripting is computationally efficient. Its disadvantage, though, is the proprietary nature of the technology. Although JavaScript is portable across server platforms, it can only be used with Netscape servers. It also suffers from the third limitation that is inherent in all HTTP-based implementations.

4.1.3 Overcoming Limitations of HTTP

Recall, from Section 2.4, that HTTP is connection-oriented, stateless, and directional. The needs of OR/MS applications (e.g., stateful, interactive, and computation-intensive) are not well suited to these HTTP features, but can be addressed via extensions and newer technologies.

- *Dealing with the Connection-Oriented Nature of HTTP*. HTTP's connection-oriented property means that the server keeps open the TCP/IP connection to the client as long as it takes to service the HTTP request. If the execution (on the server) of an OR/MS algorithm takes a long time, this results in inefficient use of network resources. Because HTTP is directional, the server cannot simply close the connection on receiving a request and open it again when the result is available. Solutions to this problem involve treating the Web as a push medium (see Section 3.1). One alternative complements HTTP with Email technology. When a request for extensive computation is made to a Web server, the server invokes the



Recycling Ques

How many different items do you have to recycle? (please limit to seven items)	5
What payoff ($) would be required for you to drive 10	6

Recycling DSS Item Inputs

Please enter each item you have to recycle and the associated quantity in the table provided below.

ITEM	QUANTITY (lbs)
Aluminum	15
GlassCRV	20

Recycling DSS: Optimization Model Solutions

Summary of Results

Total net payoff for the optimal alternative is $ 41.74.
Your total payoff for this plan will be $ 48.20.
Your travel costs associated with this plan will be $ 6.46

Item-Station Matching

Item	Station	Payoff
Aluminum	TTCR	$ 10.2
Cardboard	MRMN	$ 0
GlassCRV	NOSH	$ 2
PlasticCRV	NOSH	$ 25
TinCans	TTCR	$ 11

Travel Plan

From	Destination	Travel Cost
MRMN	NOSH	$ 0.75
NOSH	home	$ 0.37
TTCR	MRMN	$ 2.45
home	TTCR	$ 2.89

Figure 5. Waste disposal and recycling system showing remote interaction via CGI scripts. The user's input interface (left) and output interface (right) is in HTML.

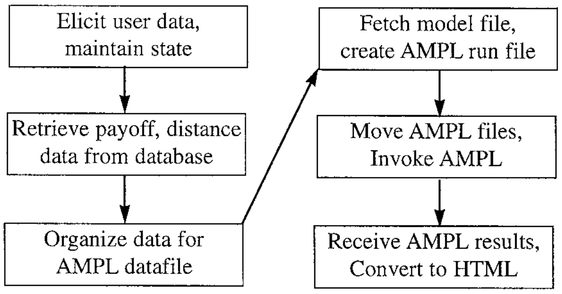

Elicit user data, maintain state
Retrieve payoff, distance data from database
Organize data for AMPL datafile
Fetch model file, create AMPL run file
Move AMPL files, Invoke AMPL
Receive AMPL results, Convert to HTML

Figure 6. Waste disposal and recycling system showing remote interaction via CGI scripts. The scripts perform a series of tasks and provide an interface between the optimization system, a database system, and the user.

model or algorithm and breaks the connection with the Web browser. Upon completion of the computation, the user is notified by electronic mail. With IMAP compliant mail servers, for example, the user can read this Email using a Web browser and retrieve the results of the computation by following the URL in the body of the email message.

- *Maintaining State.* A common feature of OR/MS applications is that user interaction involves a series of steps. Because HTTP is a stateless protocol, each request from a client to the server is considered to be independent of previous requests. This is a problem, for example, if instantiating data for an optimization model are to be collected in a sequence of interactions that are interdependent. Suppose that a user wishes to change only certain parameters (e.g., during what-if analysis) in a large dataset. If all data had to be sent together, every small change would require sending the entire dataset. The earliest, and still the most general, solution to this problem involves the use of hidden variables in HTML forms. For example, in the waste disposal and recycling system, user data (i.e., state information) are cumulatively stored on the server and mapped to an identifier. The hidden variables carry this identifier between the client and server, allowing the program to relate multiple interactions, and also to support multiuser access. Another, but proprietary, solution involves storage of session identifiers on the client machine: Web servers and clients process an additional HTTP header component called a cookie that can be used to uniquely identify session or transaction data that span multiple HTTP requests.
- *Support for Two-Way, Real-Time Interaction.* In HTTP, all the data from the client to the server are sent when the client initiates the transaction. Although the server can communicate with the client until it closes the connection, no further data can be sent from the client to the server as part of the ongoing transaction. This prevents the implementation of truly interactive applications over HTTP. Addressing this problem involves other methods, discussed in Section 5, that involve client-side computation and distributed computing technologies.

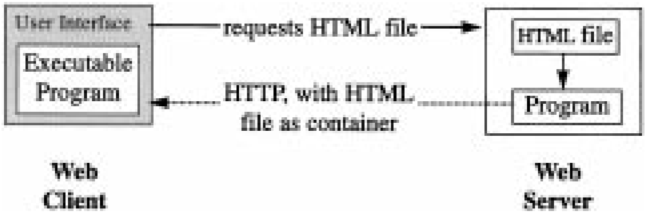


Figure 7. Client-side computing on the Web.

4.2 Methods for Client-Side Computation

Recall (from Section 2.3) that, in the basic Web transaction, a Web browser on the client machine does little more than interpret and display the HTML document sent by the server. In methods for client-side computation, the Web browser does much more. Richer types of documents can be delivered over the Web. These documents may be executable programs that run on the client machine (see Figure 7). We discuss five alternative client-side computation methods and their relative advantages and disadvantages.

4.2.1 Scripting Languages

Scripting languages were the first direct extension to plain HTML processing and display on the Web browser. The basic idea is to embed, within an HTML document, programs written in an interpreted language. Client-side JavaScript (from Netscape Corp.) and VBscript (from Microsoft Corp.) are two popular scripting languages. We use JavaScript to illustrate the concepts. Consider the following fragment of JavaScript code that is used to evaluate an expression specified in an HTML form.

```
⟨HEAD⟩
⟨SCRIPT LANGUAGE="JavaScript"⟩
function compute(form) {
  if (confirm("Are you sure?"))
    form.result.value = eval(form.
    expr.value)
  else
    alert("Please come back again.")
}
⟨/SCRIPT⟩
⟨/HEAD⟩
```

Note that the JavaScript program is specified in the HEAD of the HTML file. An HTML form in the body of the same file can reference and use this program as shown below.

```
⟨BODY⟩
⟨FORM⟩
Enter an expression:
⟨INPUT TYPE="text" NAME="expr" SIZE=15⟩
⟨INPUT TYPE="button" VALUE="Calculate"
  ONCLICK="compute(this.form)"⟩
⟨BR⟩
Result:
⟨INPUT TYPE="text" NAME="result" SIZE=15⟩
⟨BR⟩
⟨/FORM⟩
⟨/BODY⟩
```

The HTML form enables the user to enter an algebraic expression and, following a mouse click, use the JavaScript program to compute the expression. JavaScript is often used to implement context-sensitive interfaces through its ability to capture and respond to user events such as mouse clicks, form input, and page navigation. It can also be used as a full-fledged programming language to implement OR/MS algorithms. We now briefly discuss both uses of JavaScript.

- *Data Error Checking in HTML Forms*. In the standard Web architecture, HTML forms allow users to submit data to a Web server. Input data validation must be done at the server. This round trip to the server—even for minor data checks—degrades the quality of the interaction with an OR/MS application. Error checks on data are easily achieved with a client-side scripting language. Elements of HTML forms are associated with events that the user can trigger with keyboard or mouse actions (e.g., mouse clicks). These events invoke the script programs that can perform the desired check and provide immediate feedback to the user. Only valid data are sent to the server. This strategy also permits a clean separation between the user interface logic—implemented using JavaScript—and the algorithmic logic implemented on the server using the methods described in the previous section.
- *Implementing OR/MS Methods using JavaScript*. As noted earlier, JavaScript could also be used to implement OR/MS algorithms. This strategy would require the development of the interface and processing components of the application in JavaScript. WORMS is a leading example of this approach in the OR/MS arena. WORMS makes available JavaScript implementations of methods such as dynamic programming. Although this approach bundles the entire application for processing on the client platform, it does have several disadvantages. First, JavaScript, being an interpreted scripting language, is not well suited to developing and maintaining efficient implementations of complex algorithms. Second, because JavaScript implementations are embedded in HTML documents that are transported to the client, the source code for the implementation is available to the user. Third, if the implementation is complex and long, it can increase the time required to download the application.

4.2.2 The Plug-in Model

Although scripting languages enable computation on the client platform, they rely on the Web browser to serve as the language interpreter. This permits programs written in JavaScript to be processed by any Web browser on any platform. In contrast, plug-ins, part of a Netscape-defined architecture, are platform-specific compiled components. They provide a way of packaging and distributing existing C and C++ implementations of OR/MS algorithmic products for installation and use within a Web browser on the client platform. Plug-ins conform to an application programmer interface and are invoked by the browser when it attempts to load a file of a given MIME type.

Although plug-ins are typically used to process specialized content such as sound, video, and animation files, the same approach can be used to define and process OR/MS-specific MIME types. Thus, for example, one might define a nonlinear optimization model MIME type, causing the browser to invoke a nonlinear optimizer (packaged as a plug-in) to process the model. Because the plug-in software resides on the user's machine, version management becomes an issue. The plug-in architecture addresses this problem through methods for updating plug-ins without having to reinstall the entire software. We examine these advanced features later in this section.

4.2.3 Java Applets

Originally envisioned as a language for use in embedded applications such as home appliances (e.g., *set-top boxes* in applications such as WebTV), the Java[2] programming language (from Sun Microsystems) has fast become the vehicle for delivering interactive and computational applications on the WWW. The Java toolkit includes much more besides the language (e.g., database connectivity and remote method invocation; see [43] for a brief introduction to the various parts).

Java is an object-oriented language that shares certain syntactic similarities with C and C++. Due to its support for network protocols, it is especially suited to network-centric computing. Interested readers are referred to two gentle introductions ([28] and [47]) for the features and benefits of Java as a programming language. With respect to its use for OR/MS applications on the Web, Java's key feature is its platform independence (write once run anywhere) that results from compilation of Java programs into an intermediate platform neutral representation called *bytecode*. Bytecode is executed by interpreters (Java virtual machines[23]) that are available on major platforms. Because leading Web browsers implement the JVM, mobile Java programs (applets) can execute within the browser. This capability has captured the imagination of Web developers and several OR researchers (see e.g., work by Gagliardi and Spera[27] on classroom scheduling, Hochbaum on the RIOT project with several optimization problems, Jones on the TSP and LP animation, and Bradley on graph coloring heuristics).

Java applets use HTML as a container and are downloaded using HTTP from a Web server. For example, consider the following excerpt from an HTML file in Jones' LP animation. The applet (which interacts with the user) is embedded within applet tags in an HTML page. The width and height specify the space available on the Web browser screen for the applet. Parameters for the linear programming algorithm are specified using the PARAM tag.

```
⟨APPLET codebase=“Beta/Classes”
  code=“NULP.class” width=600 height=500⟩
⟨PARAM name=objsense value=“max”⟩
⟨PARAM name=objective value=“6 5”⟩
⟨PARAM name=constraints value=”1 2 < 18 2 1
  < 18 1 1 > 3 1 −1 < 6 −1 1 < 6”⟩
⟨PARAM name=graphwidth value=200⟩
⟨PARAM name=graphheight value=200⟩
⟨PARAM name=showrect value=”−1 −1 15 15”⟩
⟨PARAM name=varlabels value=“Ch Ta”⟩
```

```
⟨PARAM name=roundrhsto value=0.1⟩
⟨/APPLET⟩
```

In this example and the illustration in Vignette 2, the applet is a standalone implementation of an OR/MS method. Although this is feasible, there are certain problems. First, this often requires reimplementation of a method in Java. Second, as the size of the applet grows, there are delays both in downloading the applet over the network and in loading it on the Web page. Third, the client platform may not have the computational power to execute the model.

In such cases, Java applets can be used to implement platform-independent user interfaces that can be used with a Web browser on any platform. The user interface applet can communicate with a computationally intensive application (written in Java or another language) running on a server. There is considerable attention paid to such communication in Java between the applet and the server. In addition to HTTP, Java supports the remote method invocation (*RMI*, discussed in Section 5.1) method and the CORBA-based Internet InterOrb (*IIOP*) Protocol (CORBA and IIOP are discussed in Section 5.2). RMI and IIOP provide persistent two-way interaction between client and server. Using RMI or IIOP, Java applets can be used to implement interactive OR/MS applications on the Web.

4.2.4 LiveConnect: Integrating Plug-ins, JavaScript and Java Applets

LiveConnect is a Netscape object technology that allows each of the client-side technologies that we have discussed to be integrated within an object-oriented framework. This is done by making any Java applet (that gets downloaded into a Netscape browser) an object that can be referenced from a JavaScript program. After an applet is downloaded, a JavaScript program is able to access any Java public *classes* defined in the applet. Similarly, functions that are available in plug-ins can also be addressed as objects by a JavaScript program. When OR/MS technology is packaged as plug-ins and applets, JavaScript can be used to script these components (and potentially others) to deliver OR/MS-based applications to a remote desktop.

Although we are not aware of existing OR/MS applications that make use of LiveConnect, consider the following illustrative application modeled after an example on the Netscape Web site. A Java applet gathers live stock market information through the Internet. It calls a JavaScript function that implements a model (say, a neural network) for determining whether a user should buy or sell a certain stock. When a buy or sell condition is identified, this triggers an event that results simultaneously in an audio alert (raised by a plug-in) and a pop-up dialog box. The advantage of this technology is the ability to package OR/MS applications in rich multimedia contexts. The disadvantage is that it uses proprietary technology (LiveConnect and plug-ins) and can only be used with Netscape browsers. However, these technologies represent new and exciting opportunities for the distribution and packaging of OR/MS content, particularly instructional material.

4.2.5 ActiveX Controls

An ActiveX control is a reusable software component and is part of Microsoft's Component Object Model (COM) technology.[18] In contrast to Java components, ActiveX controls are platform-specific *binaries*. Although computationally intensive ActiveX controls are usually implemented in languages such as C++, simple controls can be implemented in popular scripting languages such as Visual Basic 5. Currently, ActiveX can only be used on Windows platforms. However, Microsoft has indicated plans to port its technology to Unix and Macintosh platforms as well.

Controls are embedded in client software called containers. Originally, ActiveX control containers were written in Visual Basic. Following the growth of the Web, Microsoft has designed the Internet Explorer browser to be an ActiveX control container as well. The browser, when encountering data that require a specialized viewer, loads a viewer (e.g., an Excel viewer or an OR/MS algorithm) implemented as an ActiveX control. The control may already be present on the client machine or it might have to be downloaded as shown in the following example.

```
⟨OBJECT
CLASSID="classid:B16553A0-06DB-101B-85B4-
  00000C0009BE05"
CODEBASE="http://www.traveling_salesman.
  com/mapshow.ocx"
ID=Mapdisplay
Data="http://www.traveling_salesman.com/
  mpadata.geo"
width=600 height=500⟩
⟨/OBJECT⟩
```

Because they are platform-specific binaries, ActiveX controls are resident after downloading on the client platform. This can pose security problems and the user has to ensure that the control is being downloaded from a trusted server. In contrast, Java applets have to be downloaded every time they need to be used and execute within the browser. The user of a Java applet is assured of using the latest version. With ActiveX programs, however, when a new version becomes available, the onus of fetching and updating the program falls on the user. The overhead incurred to download an ActiveX control (a control is usually much larger than a comparable applet) is offset by the faster speed of execution of the control (because it is an executable binary) when compared to an interpreted Java applet (even accounting for the availability of *Java just-in-time compilers*).

At present, ActiveX controls work best with Internet Explorer browsers. Netscape browsers require a plug-in to process ActiveX controls. Therefore, as an option for implementing and distributing OR/MS algorithms on the Web, ActiveX controls are a natural migration path optimized for users with Internet Explorer browsers on Windows machines.

Scripting ActiveX Controls

Just as JavaScript can be used to access public methods of Java applets, scripting languages can obtain access to methods implemented as ActiveX controls. This is usually done using VBScript and the process is conceptually similar to the process with JavaScript. This allows trapping of—and subsequent action on—events (such as mouse clicks) generated by a user interacting with an interface (such as a map display of a traveling salesman tour) generated by an ActiveX control.

The following example illustrates a VBScript program that accesses methods (in the example, SpinUp) of the ActiveX control using its ID tag, Mapdisplay. The script is capable of providing feedback as the user interacts with a traveling salesman tour being displayed by the ActiveX control.

```
⟨SCRIPT LANGUAGE="VBSCRIPT"⟩
Sub Mapdisplay_SpinUp( )
MsgBox "(Route Moved)"
End Sub
⟨/SCRIPT⟩
```

Integration with Java

ActiveX controls can be integrated with Java using certain Windows-specific extensions. ActiveX controls can be imported as Java classes into Java applications, and Java objects can be used wherever ActiveX controls are used. This integration is presently only supported by Microsoft's implementation of Java.[18] Because ActiveX controls are binaries, applications developed using this model will not be platform independent. However, it does allow the use—as ActiveX controls on the client platform—of existing software. Further, it offers OR/MS developers writing to Windows platforms the opportunity to draw on a large market in reusable ActiveX software components.

Summary

A key difference between ActiveX controls and Java applets is that ActiveX controls are binaries that are permanently installed on the client platform after they have been downloaded. They share this feature with plug-ins. The advantage of this approach is access to resources (superior execution speeds achievable with compiled code and access to other ActiveX controls) on the client platform. The disadvantages are concerns about security, the need to write controls for multiple target platforms, and dedicated use of client resources. A summary of the technologies discussed in this section is given in Table I.

5. The Web and Distributed Computing Technologies

The previous section described several recent technologies that allow the deployment of OR/MS and other computational products over the Internet. Because of the limitations of each alternative, no single alternative may be sufficient for implementing a Web-based version of an OR/MS product. For example, consider the vehicle routing and traveling salesman problems. Both problems require complex OR/MS solvers and sophisticated graphical user interfaces. CGI and related technologies, although allowing the use of a suitable solver, are inappropriate for the user interface because they are limited by the capabilities of HTML and HTTP. On the other hand, Java applets may deliver a nice user interface, but, because they are interpreted, are unsuitable for the solution algorithm.

More recent developments, enabled by the integration of distributed computing technologies, provide solutions to these problems. In this section, we seek to answer questions that relate to the view of the Web as a distributed computing environment for OR/MS application development: (a) How can OR/MS products be packaged as independent components with well-understood interfaces, and (b) How can these components communicate with other remote components and with remote users?

Techniques discussed in the previous section provide an initial, but unsatisfactory solution. For example, although CGI applications allow remote interaction, the external program that is invoked in response to a CGI call does not have a published interface. The interfaces are implemented as part of a CGI script. When there is an increase in either the number of external programs that have to be called from a CGI script or in the complexity of the interface, the CGI script becomes complex. This makes it difficult to debug and maintain.

Communication in CGI systems uses HTTP between the client and the server. Any other communication, either between an applet serving as an interface to an application on the server or between the CGI application and other remote servers, requires use of communication mechanisms such as *TCP/IP sockets* and cumbersome application-specific protocols. Once again, these solutions are difficult to maintain.

Recent developments such as the RMI system (defined as part of the Java object model) and the IIOP,[3, 37, 41] a messaging standard defined as part of CORBA[39, 41], address these shortcomings. Both approaches enable the development of objects with clearly specified interfaces—using an *IDL* (interface definition language) in the case of CORBA, and a special type of class called interface classes in the case of Java. IIOP is used in CORBA to enable communication between remote objects, whereas RMI does the same for Java objects executing on remote JVMs. The principal difference between the two is that RMI assumes that communicating processes run on a JVM (i.e., are written in the Java language), whereas the CORBA/IIOP solution admits a heterogeneous world with object implementations in various languages (presently, CORBA supports C, C++, Java, Smalltalk, Cobol, and Ada). We discuss these two technologies—and how they help answer both questions—in Sections 5.1 and 5.2.

5.1 Java RMI: Java-Based Distributed Computing

Section 4.2.3 described how an OR/MS method could be implemented as a stand-alone Java applet, contained in an HTML page, and transported using HTTP from a Web server to a client. However, once an applet is executing on a client JVM, it can communicate with objects executing on the Web server from which it was served. Similar communication is possible between two objects executing as Java applications on remote JVMs. In both cases, the enabling technology is the RMI system, introduced to integrate a distributed object model into the Java language. RMI defines a struc-

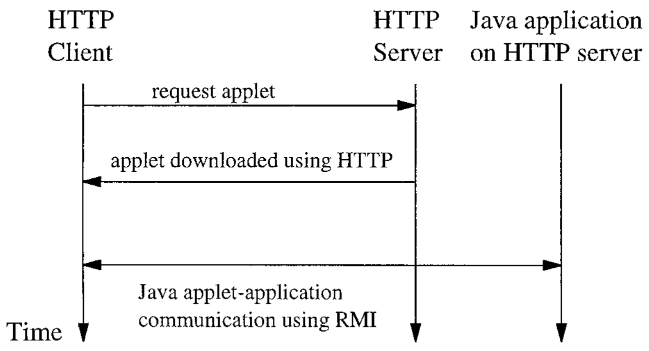

HTTP Client
HTTP Server
Java application on HTTP server
request applet
applet downloaded using HTTP
Java applet-application communication using RMI
Time

Figure 8. Distributed computing via Java RMI. An OR/MS algorithm residing on an HTTP server can present, as a Java applet, a graphical user interface to a remote user.

tured and transparent way in which a Java object, executing on one JVM, can invoke methods of another Java object executing on a remote JVM.

RMI is based on a simple concept. A Java object whose methods are to be invoked remotely defines its interface (i.e., a declaration of its methods). We will refer to such an object as the server. We will refer to the object that makes remote calls to the methods of the server object as the client. The server object is registered with the RMI system, a process that generates *stubs* and *skeletons*. A stub serves as a proxy or surrogate for the server object implementation by supporting the same set of interfaces as the server. The skeleton is a server-side entity that dispatches calls to the implementation of the server object. The implementation of the client (either an applet or an application) uses the stubs to make the calls to the methods of the remote server object. When the methods are invoked, the RMI system uses the skeleton to invoke the methods on the remote object and returns the result.

We describe two alternative ways in which this technology could be employed for implementing OR/MS applications.

5.1.1 Applet as User Interface with an OR/MS Algorithm on the Server

In this case, a Java applet manages the user interface. The method resides and executes on the server. The time-sequenced communication between the different components is shown in Figure 8. The applet is loaded into the browser using HTTP. Further communication between the applet and the application uses RMI.

As Figure 9 indicates, there are now two possibilities. The application (the OR/MS algorithm) itself may be written in Java (Option 1). An existing (say, C++) implementation of the algorithm can be wrapped in a simple Java application that executes it upon receiving an RMI request from an applet (Option 2). In both options, the wrapper or the algorithm is implemented as an RMI object using a specialized set of class and interface libraries. This is a simple and powerful way to make existing implementations (also referred to as legacy systems) available on the Web. The additional work involves redesigning the user interface as an applet and writing a wrapper application in Java. The first approach is taken in the design of CSLab,[19] a Web-accessible simulation environment for use in teaching and research. CSLab users construct, and interact with, simulation experiments—realized as Java classes on a remote server—through a worksheet implemented as a Java applet in their Web browser.

5.1.2 Applet Interface: Distributed Implementation

Remote method invocation is not limited to communication between an applet and a Java application on a server. As noted earlier, it can be used for communication between Java applications on different machines as well (see Figure 10). This approach permits distributed applications to be built out of well-tested and maintainable components with relative ease and robustness. For example, consider an application that requires the integration of a database server, a forecasting model, and a logistics planning model, all available as independent components. Now, each component is implemented as an RMI object on its own server. The client for each is a Java application executing on a Web server that also serves the user interface applet(s). This Java application mediates between the user interface applet and the components implemented on the remote servers. On receiving user requests (e.g., for a forecast, access to data, or the logistics planning model) communicated by the applet using RMI, it invokes methods—again using RMI—on the corresponding component objects.

Although this is a clear improvement over older TCP/IP socket-based approaches, the RMI system has some shortcomings. First, RMI is specific to Java. It is designed to enable objects to invoke methods on objects executing on remote JVMs. Second, and perhaps more importantly, it does not support features required to discover and invoke these methods at run time. Why might this be important to OR/MS application development? In the logistics planning scenario described above, RMI was used to construct a distributed OR/MS application. However, this assumes that stubs for RMI objects are available at the time clients for these RMI objects are written. Stated another way, RMI works under the assumption that clients for RMI-based services are written (using the server stubs) and compiled when the service is developed. If an application needs a new service, a new client for that service would have to be written, integrated into the (intermediary) application (which would have to be recompiled). Clearly, this is not feasible in an environment in which OR/MS products—packaged as components—are added and removed in a flexible manner and assembled together on demand (see e.g., [33]). This is essentially the underlying premise of the DecisionNet project.[9, 10, 11] DecisionNet seeks to create a flexible environment for creating OR/MS applications from a set of available components. The technology that makes this possible is CORBA,[3, 41] discussed in the next section.

5.2 CORBA and IIOP-based Distributed Computing on the Web

CORBA is an open cross-platform communications architecture. CORBA is based on two important concepts—an IDL

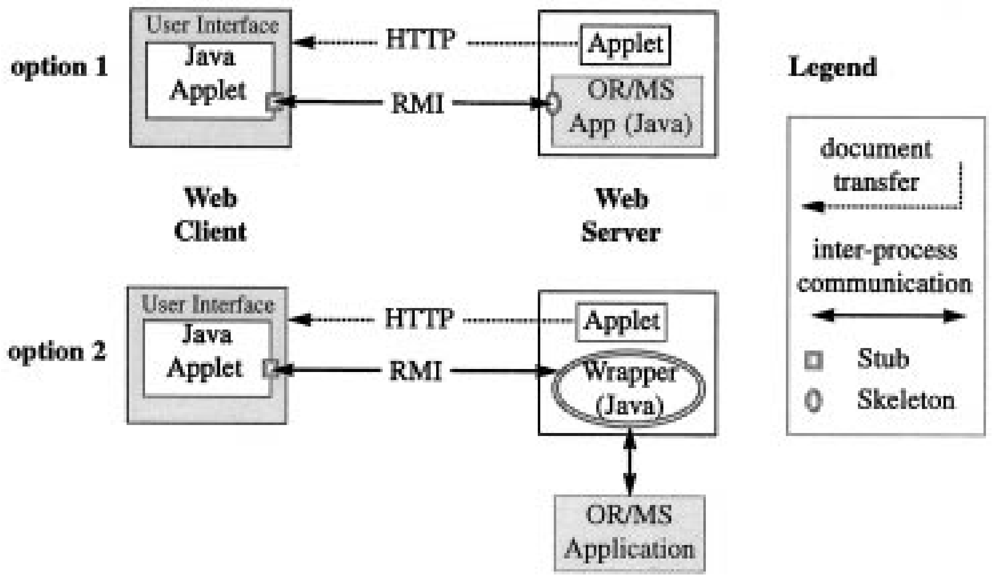


Figure 9. Applet-based interface, using Java RMI, to a server-based OR/MS Application.

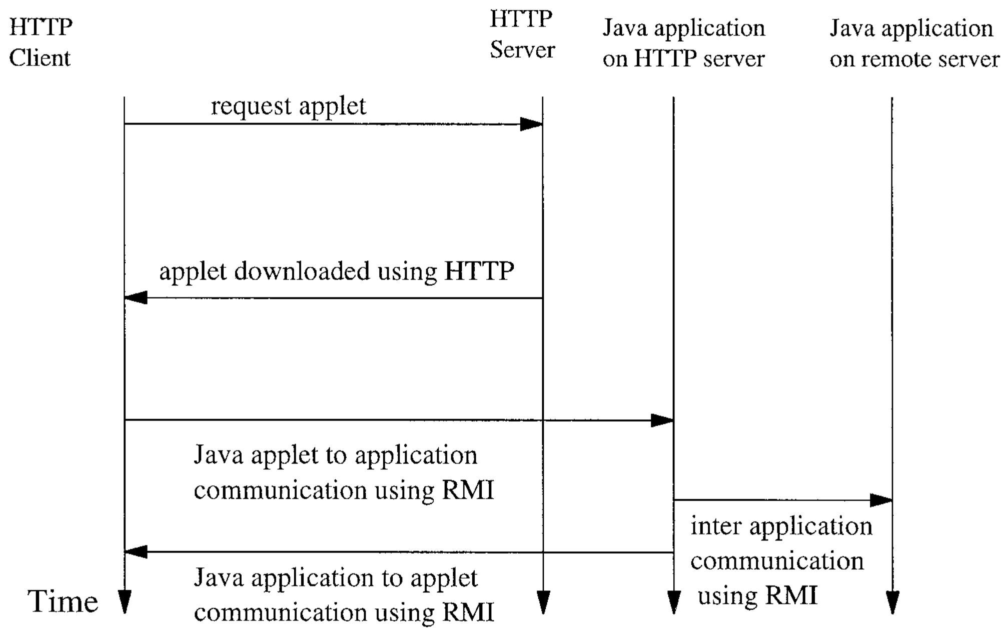

HTTP Client
HTTP Server
Java application on HTTP server
Java application on remote server
request applet
applet downloaded using HTTP
Java applet to application communication using RMI
inter application communication using RMI
Java application to applet communication using RMI
Time

Figure 10. Distributed computing via Java RMI. RMI makes possible communication between Java applications (not just applets) on remote machines.

and ORB. In contrast to RMI, CORBA allows the use of many programming languages (e.g., C, C++, ADA, Java). CORBA also provides a set of services—user interface, information management, systems management, and task management—required in any distributed implementation as infrastructural (horizontal) services. CORBA's relevance as an architecture for Web-enabled distributed computing has increased considerably following Netscape's decision to implement CORBA technology in its Web browser, giving Web users access to CORBA-compliant objects. Users with Java-enabled browsers that do not implement ORBs (e.g., Internet Explorer) may still obtain access to CORBA facilities by downloading an ORB implemented in Java.

The development of a CORBA-compliant object begins with a declaration of its interface in IDL. In essence, this defines an API to the object. IDL-specified methods can be written in and invoked from any language that is supported within CORBA. The ORB is the middleware that enables objects to make requests transparently to (e.g., to execute a simplex algorithm and return its results) and receive responses from other objects located locally or remotely. ORBs are developed by multiple vendors and interoperate on the Internet using IIOP.

5.2.1 Distributed Application Development using CORBA: Static Invocation Interface

The simple way to set up an OR/MS application for execution in a distributed environment is to perform the following steps. The interface to the OR/MS algorithm is specified using IDL. A suitable compiler (e.g., an IDL-C compiler if the algorithm is implemented in C) is used to produce skeletons in C for the algorithm server classes. The methods in the skeletons are implemented using the existing implementation of the OR/MS algorithm. Compilation of the IDL-specified methods results in the client stubs and the server skeletons. Binding the class definitions in the interface repository, and registering the run time objects in the implementation repository, completes the creation of a CORBA-compliant OR/MS algorithm server object.

A user interface to such an OR/MS algorithm server object may be constructed as an applet that is downloaded on demand by a user from a Web server. The client stubs generated during the creation of the OR/MS algorithm server are used in the construction of the user interface applet. The applet uses the services of an ORB on the Web browser to invoke, in a transparent manner, methods of the object (e.g., that implements the OR/MS algorithm) registered with the ORB on the Web server. The interorb communication is via the IIOP protocol.

This approach is based on the Static Invocation Interface (SII) of CORBA. The reader might note that, although the discussion used CORBA terminology, the remote invocation of server object methods from a client is similar to the Java RMI system. In both cases, the client needs a precompiled stub to invoke operations on the server object. Here, the primary advantage of SII over RMI is the ability to work with object implementations in a variety of implementation languages.

5.2.2 Distributed Application Development using CORBA: Dynamic Invocation Interface

The SII model is insufficient (as is Java RMI) when OR/MS algorithmic services are made available as components (as in a marketplace for computational services) that users may need to combine, on demand, to develop an OR/MS application. CORBA's dynamic invocation interface (DII) addresses this requirement. OR/MS algorithm servers offer new services and interfaces whenever they become available. Clients (such as the user interface applet) discover these objects at run time (e.g., by browsing the interface repository using the ORB) and construct the requests (i.e., calls to methods) using the interface information about the discovered object. Although DII is more complex than SII, it offers considerable flexibility and the opportunity for OR/MS developers to contribute to the electronic marketplace for specialized analytic services.

5.3 Comparison and Implications

It should be obvious that an OR/MS practitioner or researcher, as any other Web application developer, is faced with an array of choices for developing distributed Web-based applications. We discussed two alternatives that use Java/RMI and CORBA/IIOP (the user interface, in both cases, is assumed to be delivered as a Java applet). Within each category there are further variants. Besides these categories, one could also use a pure CGI approach (but with all the limitations of CGI) or a pure CORBA approach (however, users would no longer be able to use merely a Java-enabled Web browser to interact with the application). Table II summarizes the features of the Java/RMI and CORBA/IIOP alternatives. Interested readers may refer to two other articles for a comparison of technologies for building distributed applications on the Web:

1. Evans and Rogers[24] discuss how to build sophisticated, but easy to use, client software (interface) as Java applets that use CORBA to interact with (sharable) remote server software. They compare the pure CGI approach to a Java/CORBA approach along several dimensions: flexibility of design and development (Java/CORBA is better suited for complex applications); maintainability of components (Java/CORBA is better, given the more rigorous interface definitions, among other things); client deployment (with CGI, a user needs only a Web browser, but Java/CORBA wins over pure CORBA because users can still interact through a Java-enabled browser); responsiveness (Java/CORBA is better; many responses can be coded into a Java applet); and user interface intuitiveness (Java/CORBA offers many advantages in this area).
2. Baker, Cahill, and Nixon[3] discuss the role of CORBA as a bridging technology, and compare it to several alternatives, including Java (RMI and *Java Beans*), DCOM, and the CGI approach.

The technologies discussed in this section hold considerable promise for OR/MS developers thinking about implementing their techniques on the WWW. In addition to technological issues, several industry trends must be kept in mind when selecting a distributed computing platform to use for OR/MS application development. First, a set of conflicts involving major computing companies threatens the universal readability property of the Web. For example, Microsoft's Internet Explorer browser does not implement the complete Java specification (e.g., RMI is not implemented) in its Java Virtual Machine. Similarly, CORBA technologies are not supported in Microsoft browser and server products. Only Netscape servers (e.g., the Netscape ONE environment[16]) and browsers and Sun's HotJava browser implement these technologies. Second, in the area of distributed computing (where CORBA is the result of an industry-wide effort), Microsoft has offered a competing approach called DCOM. DCOM shares CORBA's objectives and supports both the static and dynamic invocation interfaces. However, it is currently available only on Windows NT and Windows '95 platforms. Furthermore, CORBA, which is in its third generation (having been in development since 1991), is presently considered to be more robust.

Table II. Technologies for Building Web-Based Distributed Applications

Technology	Advantages	Disadvantages
Algorithm (in Java) on server and user interface as Java applet	Simple to set up Separates user interface and algorithm	Must rewrite algorithm in Java limited to client server architecture
Wrapper (in Java) with algorithm on server and user interface as Java applet	Works with legacy implementations Algorithm can be coded in any programming language	Efficient wrappers only for languages (e.g., C/C++) for which Java has native language interface Limited to client server architecture
Algorithm components implemented in Java; distributed over network	Allows distribution of components and support for a peer-to-peer architecture No need for interface definition language; interfaces can be written in native Java classes RMI does garbage collection	RMI does not offer core services (e.g., discovery, security) Components must be predefined No equivalent protocol to IIOP for Inter-ORB communication, making RMI less scalable
CORBA static invocation interface	Can make use of core CORBA services Objects can be registered with multiple ORBs Inter-ORB communication through IIOP Enables robust and efficient distributed implementations	Need to write interfaces in IDL Few browsers support CORBA/IIOP (Java ORBs are available and supported) Client stubs and server skeletons predefined and registered; therefore not very flexible
CORBA dynamic invocation interface	Dynamic discovery and dynamic invocation of objects Robust and flexible architecture Clients discover, and request, new objects at run time	Need to write interfaces in IDL More complex than SII Few browsers support CORBA/IIOP (Java ORBs are available and supported) Flexibility of DII results in loss of efficiency

The first three alternatives progressively use Java RMI for communication. Advantages are inherited going from alternative top to bottom and disadvantages are inherited in the reverse order. The last two technologies are based on CORBA. In each case, we assume that the user interface is rendered as a Java applet.

6. Emerging Technologies and Trends

The technologies discussed in the previous two sections—though fairly new and underexploited for OR/MS applications—are relatively well established in the Web industry in comparison to some even newer developments that will soon be broadly available in commercial Web browsers and servers. These are XML (the extensible markup language[20, 32]) and RDF (the resource description framework). We believe that XML and RDF will be instrumental in enabling the use, for OR/MS applications, of the technologies previously discussed. In this section, we briefly discuss these developments and their implications for OR/MS.

6.1 XML: Extensible Markup Language

XML is a language being developed by the W3C to enable the use of SGML (standard generalized markup language[46]) on the WWW. To understand the connections between XML and HTML, it is important to recognize that SGML is the international standard metalanguage for defining markup languages. Markup languages provide tags that allow information providers to describe their content. HTML is an example of such a markup language and is defined (more precisely, as a document type definition or DTD) using SGML. XML also has its roots in SGML. However, like SGML and unlike HTML, XML is a metalanguage for creating markup languages (for a broader discussion of this subject, and the role of XML for document definition, see [34]).

Thus, XML permits information providers to design their own markup languages (i.e., introduce tags specialized to their needs and define the grammar of a document that uses these tags). For example, one could develop document type definitions for documents (model statements) created in algebraic modeling languages. These features are already being used to develop specialized markup languages.

Another relevant example of a specialized markup language is MathML (mathematical markup language), an XML application developed by the W3C. Consider the following fragment from the MathML site used to encode the expression $x^2 + 4x + 4 = 0$.

```
⟨EXPR⟩
  ⟨EXPR⟩
    ⟨EXPR⟩
      ⟨MI⟩x⟨/MI⟩
      ⟨POWER/⟩
      ⟨MN⟩2⟨/MN⟩
    ⟨/EXPR⟩
    ⟨PLUS/⟩
    ⟨EXPR⟩
      ⟨MN⟩4⟨/MN⟩
      ⟨TIMES/⟩
      ⟨MI⟩x⟨/MI⟩
    ⟨/EXPR⟩
    ⟨PLUS/⟩
    ⟨MN⟩4⟨/MN⟩
  ⟨/EXPR⟩
  ⟨E/⟩
  ⟨MN⟩0⟨/MN⟩
⟨/EXPR⟩
```

The various MathML tags used to encode the algebraic expression contain information that could be used both to display the expression as well as to compute it. The grammar of these tags is specified in an XML document-type definition that could be used by XML *parsers* to structurally validate the MathML document. However, the semantics that guide the processing of the tags (either for display or for computation) are not specified within XML. The semantics can be encoded in a scripting language (JavaScript programs) or in dynamically loaded applets written in languages such as Java. XML parsers will become widely available with the next generation of Web browsers, and APIs for Java are under development. This will facilitate distribution of content requiring specialized representations (e.g., OR / MS models and methods) and permit interoperability between information services that can now exchange structured data formatted in XML-specified markup languages. We discuss both XML-enabled capabilities from an OR / MS perspective in the next section.

6.1.1 Specialized OR/MS Markup Languages

Currently, OR / MS modeling systems and applications use specialized representations to specify models or input data. Examples include modeling languages such as AMPL[26] and the standard input data format (SIF) used to specify problems for nonlinear optimizers.

Specialized markup languages—say, an AMPL / ML (AMPL markup language) or an SIF / ML—with tags customized to express the concepts underlying these representations, could be defined in XML. Documents containing problems specified using these specialized markup languages can be parsed by XML-compliant Web browsers. Processing required to either display the documents or compute expressions is made possible using an API (e.g., the document object model API). This API permits applets implementing the semantics of the specialized tags to be downloaded and used as needed.

For instance, with AMPL / ML, one could imagine publishing models marked up in the AMPL / ML language. Users who would like to use the model would download the model—an AMPL / ML document. The parsing and validation of the AMPL model specification—an activity currently performed within the standalone AMPL system—would be performed by parsers available within an XML-compliant browser. The parser would use a document type definition of AMPL / ML downloaded from a server. The structures extracted by the parser would be handed over using a standard API to a Java applet. The applet would interpret the semantics associated with the AMPL / ML tags and interact with an AMPL server—using RMI or CORBA—to obtain access to the AMPL engine and the solvers.

6.1.2 Structured Document Interchange

As discussed in Sections 4 and 5, the Web will enable distributed OR / MS applications. In some of these applications, a Web client may have to mediate between two or more solvers. For example, a user might interact with a forecasting tool to develop a forecast and take the resulting forecast and use that to initialize parameters of a linear programming model. How might this application work currently? The user might interact with the forecasting application using HTML forms. However, the results of the forecasting application cannot be fed automatically into the linear programming application because a well-defined data interchange structure cannot be specified using the limited set of tags in HTML. Therefore, the forecasts may have to be re-entered by hand. XML, with its ability to define a specialized markup language, can be used to define a document interchange language between the forecasting and linear programming applications. This would permit a user to employ a drag and drop metaphor by dragging the XML document returned by the forecasting application and dropping it in as input to the linear programming application.

6.2 Resource Description Framework

The Resource Description Framework (RDF) provides the infrastructure for specifying *metadata*. Specifically, it provides a simple, yet expressive, data model that can be used to make assertions about Web resources. These RDF assertions can be used in a variety of application areas. Examples include resource discovery (in which current search engines provide only keyword-based search) and cataloging (providing a site map of content relationships between components at the site). RDF assertions can be expressed using XML. They leverage the capability of XML to define specialized markup languages.

There are two key principles underlying RDF. The first is the core data model built around the concept of nodes, properties, and values. Consider, for example, a fragment taken from the W3C's RDF pages. Consider expressing the statement "Ora Lassila" is the "author" of the Web page "http: / / www.w3.org / People / Lassila." This is represented as the RDF statement, author, [http: / / www.w3.org / People / Lassila], "Ora Lassila". The first element of the tuple is the author property of the object (the Web document) and the third element is the value of the property. This syntax provides a general language that can be used to make assertions about Web sites. The second important principle in RDF is support for reification. Reification permits an assertion (such as the example above) to be treated as an object and to have assertions made about it. Thus, one could assert the statement that "Ralph believes that Ora Lassila was the author of http: / / www.w3.org / People / Lassila." Space limitations prevent a more complete discussion. The interested reader is referred to the W3C site devoted to RDF.

What are the implications of RDF for OR / MS on the Web? As the Web grows as a platform for commerce, we expect to see commerce in OR/MS software components packaged as CORBA, Java, or ActiveX controls. Supporting the discovery of these resources in a distributed platform, such as the Web, will require more than the keyword-based search that is possible today with search engines such as Yahoo and Lycos. In particular, metadata about the computational features of a component (such as its inputs and outputs and their associated types, execution times on benchmark problems, and implementation language) can be asserted using RDF. Such assertions can be used by RDF-aware search engines to discover a component or a collection of components that could solve a problem faced by a user.

RDF can also be used to specify rankings and ratings of Web-based content. This will permit ratings along dimensions such as speed of solution, numerical accuracy, and stability of an OR/MS product to be reported and compiled. Once again, this sort of metainformation can be put to use to selectively discover resources of interest. Finally, it should be noted that RDF assertions can be supplied by either the content provider or by any third party in a secure manner (e.g., through the use of digital signatures). This latter option for specifying metadata could enable a whole class of third party OR/MS rating services.

7. Discussion and Conclusions

The Web, Internet, and associated technologies have been evolving at an astonishing rate. In this article, we have attempted to describe a wide range of Web technologies, with particular emphasis on technologies and methods that affect OR/MS work. We began with four examples that described the impact of Web technologies on OR/MS practice, research, education, and professional interaction. Following that, and a brief overview of the Web, we discussed the important developments in Web-related technologies, and ways in which OR/MS workers could exploit these new technologies. Our analysis can be summarized as follows.

1. The Web is a new—and high-speed—medium for communicating OR/MS materials and for communication between OR/MS professionals. Because of its multimedia capability and the MIME standards, the Web can be used to exchange a powerful and extensible set of data formats. However, for OR/MS to fully exploit this feature requires the creation of new OR/MS media types and corresponding viewer applications. Also, although the Web is basically a user pull medium, many OR/MS applications would require looking at associated technologies that allow the use of the Web as a push medium.
2. The Web and Internet technologies create a new computing environment for OR/MS applications. They offer OR/MS a new development environment and distribution channel in which software development is platform-independent and distribution occurs on demand. Alternatives for computation and user-interaction fall into two categories, involving either client-side computing (via client-side scripting, Java applets, ActiveX controls, plug-ins or viewers, and LiveConnect) or server-side computing (via CGI programs or server-side scripting). For most complex OR/MS applications it may be necessary to use a combination of technologies.
3. Three major alternatives are now available for distribution of computing tasks in a complex application. Each of these alternatives enable this distribution through the formal declaration of the interfaces. Java RMI allows communication between applets and applications written in Java and running on any platform. CORBA, an industry-wide approach, allows components programmed in any language, and resident on any computing platform, to be combined at run time. It also supports registration of computing resources and dynamic resource discovery. DCOM is a solution from Microsoft that has similar objectives as CORBA, but is presently limited to the Windows platform.
4. The widespread availability of XML in the near future will enable several features useful for OR/MS computing on the Web. These include the use of XML as a metalanguage to create specialized markup languages customized to the needs of OR/MS, definition of interchange languages to facilitate semantic interoperability between distributed applications, and the use of RDF with XML to define broad categories of metadata that will facilitate resource discovery in distributed repositories of OR/MS content.

7.1 The Web and the Future of OR/MS

Will the Web change the future of OR/MS? If so, in what ways? As the old joke about economists goes: it is hard to predict, especially the future. There is little doubt, given the variety of existing uses of the Web in OR/MS work, that the Web has already had an impact on OR/MS activities. However, at the risk of being wrong, we explore some more significant possibilities, that we arrange into two categories: impact of the Web on the nature of OR/MS products and on the OR/MS software economy.

7.1.1 The Nature of Future OR/MS Products

The Web continues a trend in which information technologies have had an important influence on the shape and functionality of OR/MS products. Any software product is the result of design tradeoffs, made implicitly or explicitly, between various criteria such as cost, functionality, ease of use, and performance. Web-related technologies—new programming paradigms and languages, new methods for interprocess communication, and powerful communication capabilities between remote nodes—will introduce new design tradeoffs and opportunities.

One possibility involves the interface between OR/MS models and data. Although most real-world problems have elements of uncertainty, corresponding models do not capture this directly because the resulting stochastic models would be very hard to solve. Often, uncertainty is caused by the dynamic nature of data, and lack of access to current or real-time data. This occurs, for example, in a military logistics planning problem (known as support requirements planning[36]).

Support requirements planning helps determine the set of combat support forces required to sustain a combat force that is to be deployed. Traditionally, methods for solving this problem relied on forecasts of demand imposed by the combat forces for various services. The quality of the planning was dependent on the quality of the forecasts. Adjustments to incorrect forecasts and data could not be easily obtained and integrated into the planning process. Krishnan and Padman[36] present an alternative approach—enabled by a Web-based architecture—that relies on the ability to integrate real-time data feeds from the field into the planning process and to deploy forces using such data on short notice. This type of just-in-time planning is enabled by the distributed computing architectures. As Web technologies become widely deployed in organizations, we believe that integration of real-time data feeds will result in the development of new types of planning and control systems.

7.1.2 The Future of the OR/MS Software Economy

The availability of several levels of Web-based computing solutions—basic technologies such as CGI and Java, distributed computing technologies such as CORBA, and emerging technologies such as XML and RDF—may cause fundamental changes in the way OR/MS products are developed and distributed. As discussed in this feature article, (distributed) component frameworks result in the creation of well-specified components.[35] That is, rather than one monolithic do-everything application, the market contains smaller components that users can combine to create tools either for their own needs or for use as services. For example, there is a market for user interface software components programmed in Microsoft's Visual Basic language. Similar markets will form—and have already begun forming—for OR/MS algorithm implementations. One could imagine markets—likely, electronic and hosted on the Web—in which one could purchase components or make use of their services. A good illustration of this trend is the recent release by DRA Systems of OR/MS algorithm implementations as Java components (OpsResearch). Another example is the DecisionNet system[10] in which OR/MS software components, capable of interoperation, are offered as services for specialized tasks or problems.

These capabilities for Web-based computing and interoperation open up a new market for OR/MS content providers in which services may be paid for in various ways (e.g., on a subscription basis or under a pay-per-use model). Access to these services could leverage the expected growth in simple network computing devices such as the Java station (from Sun Microsystems), the network computer (various hardware and software companies), and other hand-held or mobile communication and computing devices. This will permit electronic commerce in software services and we expect that OR/MS software designed for vertical industry segments (e.g., real estate, finance, or logistics) can deliver their services over the network.

7.2 Issues for the OR/MS Community and Leadership

To date, several initiatives that benefit the INFORMS community as a whole are, or were created out of, individual volunteer efforts. Examples—in the areas of research, practice, teaching, and communication—include the electronic journal *ITORMS* (also, see Bradley's article on interactive and dynamic *Research Publications using Java*), *Practice Online*, Greenberg's *Mathematical Programming Glossary*, and *Michael Trick's OR page*. As this feature article demonstrates, the Web presents many more opportunities for positively affecting OR/MS work.

But, besides technology, what else is required to realize the potential of the Web? Must individuals fend for themselves, or can the INFORMS organization do something to facilitate the process? Imagine, for example, in the absence of the INFORMS Online pages about our national conferences, that individual authors tried to disseminate information about their role in the conference. Can certain structures or generic systems serve as catalysts? Can certain software be developed at the community level rather than be developed multiple times at the individual level? For example, if referees are to be given the capability to test and execute algorithms described in a submitted paper, can generic software be developed that makes it easy for authors to place these algorithms for access and execution by referees? In this section, we examine some of these questions with the aim of provoking further discussion among various INFORMS constituents.

7.2.1 Electronic Community

Given the widespread impact that the Web has had on society and, in particular, on academic communities, we believe INFORMS can play an important role as a catalyst to educate members in the technology and facilitate the use of the technology in the following ways.

- *Technology Demonstration Sites*. We believe that INFORMS can play an active role in educating its membership by hosting technology demonstration sites. These sites should be designed to facilitate inspection (e.g., one should be able to look at well-documented code and be able to understand how it works) and should help answer questions that arise in implementing OR/MS applications. Emphasis should be placed on both basic technologies and on the software component technologies that we expect will revolutionize development, deployment (e.g., embedded systems), and distribution of computational products. The online tutorials hosted by information systems magazines such as *Network Computing* are models worthy of study.
- *Monitoring Web Technology Developments*. INFORMS should monitor and inform its membership on key technology developments such as electronic commerce, electronic payment systems, and encryption. OR/MS developers will be content providers in evolving electronic markets and need to be aware of developments on these fronts. ISWORLD is an example of a voluntary effort in the information systems community that could serve as a good model.

7.2.2 Setting Standards

- *Involvement in Standard Setting Bodies.* OR/MS products are computational applications. As such, they benefit from all the innovations in distributed computing such as the Web and distributed object technologies. However, they do have specialized requirements. They are computation and data intensive and usually designed to support interactive problem solving. Although custom solutions demanded by these requirements can always be developed for specific applications, it would be useful to make such features available as part of the infrastructure. A case in point is the need to maintain state in HTTP/CGI-based OR/MS implementations. There are several alternative ways in which applications implement the features required to maintain state. If a state maintenance feature was made part of the HTTP/Web infrastructure, this would make development of OR/MS products much easier. INFORMS could play a role to ensure that the needs of OR/MS products are taken into account in the standardization process either formally, by becoming involved in W3C, or by actively commenting on all public drafts of proposed standards.
- *Developing INFORMS Standards.* The availability in the near future of XML will enable the creation of custom markup languages. Interoperability between applications depends on a standard set of tags with well understood rendering and processing semantics. Although the custom languages will likely be developed by individuals and vendors, INFORMS should provide an online forum—like W3C does for Web technologies—for public discussion of proposals that will likely have an impact on the INFORMS community.

In conclusion, it is evident that Web technologies present a major opportunity for the OR/MS community. This implies that individual OR/MS practitioners, educators, and researchers need to be made aware of, and need to be able to take advantage of, developments in this area. We hope that this feature article provides a broad and readable introduction to Web technologies and their relevance to OR/MS, and encourages exciting new Web-based developments in OR/MS.

Acknowledgment

We are extremely grateful to the Feature Article Editor, Ed Wasil, for his thorough reviews and suggestions for revision. These have significantly contributed to improvements in the style and content of the feature article. This work was partially funded by ARO grant DAAH04-96-10385 and ARPA grant DASW 0197R007.

Appendix A: Glossary of Terms

ActiveX A software component technology from Microsoft, and part of the Component Object Model. ActiveX controls are platform-specific binaries that can be downloaded and used over the Web.

API Application Programmer Interface. A set of classes and associated methods that enables an external program to obtain access to a system.

Applet A (Java) program that can be downloaded over the network and executed on a Java Virtual Machine.

Binary Compiled version of a program designed to be executed on a specific operating system and CPU.

Browser A program that is used to browse the World Wide Web. It provides transparent access to a variety of other Internet protocols such as FTP, Gopher, SMTP, IMAP4, and POP3 that are used to transfer files and send and receive electronic mail.

Bytecode Intermediate target representation for Java compilers. Bytecodes, in turn, are interpreted by Java virtual machines that then execute them on the target machine.

CGI Common Gateway Interface. Defines the standard application programmer interface by which a Web server transfers data (obtained from a Web client) to an external program.

Class In object-oriented programming, a class defines the structure and behavior of a family of objects.

Client Defined as part of a client–server architecture. Generates requests for service in a protocol understood by the server. All Internet protocols work using a client–server architecture resulting in the need for HTTP clients, FTP clients, IMAP clients, and so on.

Cookie A mechanism by which a Web server can remember several users' status in their interaction with the server. It involves placing state information on each user's machine.

CORBA Common Object Request Broker Architecture. Middleware, standardized by the Object Management Group, that enables the implementation of robust distributed systems using object-oriented concepts.

DCOM Distributed Component Object Model. A Microsoft technology that enables software components to communicate directly with each other across networks, including the Internet and intranets. Based on COM.

GIF Graphic Interchange Format. A standard for storing graphic files; most Web browsers can directly display GIF files.

Helper Application An application launched by the Web browser to display or process a MIME format that it is incapable of processing. Examples are audio players and video players.

HTML Hypertext Markup Language. The language for encoding, and creating links between, information in textual and certain other forms.

HTML as Container Use of HTML tags to contain URL references to nontextual media.

HTTP Hypertext Transfer Protocol. The rules that define communication between Web servers and clients.

Hypertext Transfer Protocol See HTTP.

IDL Interface definition language. A declarative language, independent of any operating system or programming language, used to define the interfaces that object implementations provide and client objects call. It is used in distributed-component architectures such as CORBA and COM.

IIOP Internet InterOrb Protocol. Standard protocol for communication over TCP/IP networks.

IMAP Internet Mail Access Protocol. A standard protocol used to receive mail on the Internet.

Java A network-centric programming language. Compilation of Java programs yields platform-independent bytecodes that can be executed on any platform that has a Java Virtual Machine.

Java Applet See applet.

Java Beans A software component technology defined for the effective reuse of Java software components.

Java Distributed Object Model A distributed object model supported within the Java programming language. Makes use of Java RMI.

Java Just-in-time Compiler A Java just-in-time compiler converts a Java bytecode representation of a program into machine-executable instructions for a specific platform when the program is to be executed.

Java RMI A system that permits a Java object (part of an applet/application) executing on a Java Virtual Machine to invoke methods of another Java object executing on another Java Virtual Machine.

JavaScript Standard scripting language for use on Web browsers. JavaScript can also be used with Netscape servers to facilitate server side scripting.

Java Virtual Machine Interpreter of Java bytecodes; maps them into executable instructions for the given platform. It is the key to realizing the “compile once, run anywhere” philosophy of Java, and is implemented within leading Web browsers.

JVM See Java Virtual Machine.

LiveConnect A Netscape-defined technology that enables developers to combine Java applets, JavaScript, and plug-ins.

Lycos A popular search engine and catalog of Internet sites.

Metadata Metadata are data about data. The type of metadata recorded (and its format) is a function of its anticipated use. On the Web, metadata are used to support the identification, description, and location of networked electronic resources.

MIME Multimedia Internet Mail Extension. Defines a standard way of formatting and encoding data that are exchanged between Web servers and clients.

MPEG Standard format, defined by the Movie Producers and Experts Group, for video on the Web.

MSAPI Microsoft Application Programmer Interface. It is the API used on Microsoft Web servers to provide CGI functionality.

NSAPI Netscape Application Programmer Interface. It is the API used on Netscape Web servers to provide CGI functionality.

ORB Object Request Broker. A key component of the CORBA architecture. ORBs provide a range of services required to build robust distributed systems based on object-oriented concepts.

Parser A program that processes and validates expressions to ensure that they conform to some specified grammar.

Plug-in A software component installed on the client platform that is part of a Netscape-defined architecture. Plug-ins can be invoked by the browser (as a helper application) and scripted using JavaScript.

RDF Resource Description Framework. A metadata description framework under development by the World Wide Web Consortium.

Real Audio A standard for streaming audio files. It is widely used to disseminate sound files on the Internet.

RMI See Java RMI.

Script Programs written in a high-level scripting language such as JavaScript or VBScript. They are useful in tying software components together.

Search Engine Technology that is used to develop an indexed collection of Web sites and to search it. Alta Vista from Digital is a popular search engine.

Server Defined as part of the client–server architecture, the server is software that responds to requests from a client.

Set-top Box A hardware device containing a CPU that is used in conjunction with television sets. It enables computational processing, e.g., Internet access, using a TV as a monitor.

Skeleton The skeleton for a remote object is a server-side entity that dispatches calls to the actual remote object implementation.

SMTP Simple Mail Transfer Protocol. A standard protocol used to send mail on the Internet.

SQL Structured Query Language. The standard language used to query relational databases.

Stub The object that serves as a surrogate for an actual implementation of a remote object. The stub has the same set of remote interfaces defined by the implementation of the remote object.

TCP/IP A suite of protocols for networking computers and transmission of data between them. IP (Internet Protocols) breaks data into packets and routes them in best-effort delivery. TCP (transmission control protocols) provides reliability of the delivery.

TCP/IP Socket In TCP/IP-based networks, a socket defines a logical address for a program. Sockets allow communication between client and server programs.

Thread A single sequential flow of control within a program. Traditional programs consist of a single thread. Modern programming languages permit programs consisting of multiple threads. These threads can execute concurrently. Using multiple threads allows a server to handle clients’ requests in parallel, instead of artificially serializing them or creating one server process per client.

Threaded Discussion An Internet-based discussion forum that allows users to join or follow an individual discussion, i.e., a message and a sequence of responses to it, in a newsgroup or bulletin board.

URL Uniform Resource Locator. An address that specifies the access protocol used for access, the Internet node (by domain name, or by IP address), and a complete path to the resource being requested.

VBScript Microsoft-defined scripting language that has been adapted for use with Microsoft Web servers and browsers.

Viewer Specialized software that is designed to display or process data encoded according to a specified format (e.g., MIME type). Often used as helper applications on the Web.

W3C World Wide Web Consortium. A group of representatives from several companies and universities, it is the official body for setting standards for Web technologies.

WAV One of several standards for encoding audio on the Web.

World Wide Web A distributed hypermedia information system implemented on the Internet.

XML Extensible Markup Language. A metalanguage for markup languages under development by the World Wide Web Consortium.

Yellow Pages A listing (usually indexed) of service, goods, and products available in an electronic market. Modeled after the familiar Yellow Pages phone directory.

Appendix B: List of URLs

ActiveX	http://www.microsoft.com/activex/default.htm
Browser	http://w3c.org/WWW/
CGI	http://hoohoo.ncsa.uiuc.edu/cgi/
Classroom Scheduling DSS	http://turing.dmq.unisi.it/allocate.html
Client	http://w3c.org/WWW/
Cookie	http://developer.netscape.com/find/index.html
CORBA	http://www.omg.org
DecisionNet	http://dnet.sm.nps.navy.mil/
Document interchange using XML	http://sunsite.unc.edu/pub/sun-info/standards/xml/why/xmlapps.htm
Graph coloring heuristic	http://dubhe.cc.nps.navy.mil/~gbradley/JavaPaper/JavaPaperFeb96/7-nodeColoring.html
GUIDE	http://wwwis.win.tue.nl/2L670/static/guide.html
HTTP	http://w3c.org/Protocols/
HyperCard	http://hypercard.apple.com/
IIOP	http://www.omg.org/corba/corbaiiop.htm
IMAP	http://www.imap.org/
INFORMS Online	http://www.informs.org/
Internet Assigned Numbers Authority	http://www.isi.edu/div7/iana/
ISWORLD	http://www.isworld.org/index.html
ITORMS	http://catt.bus.okstate.edu/itorms/index.html
Java	http://java.sun.com/
JavaScript	http://developer.netscape.com/one/javascript/index.html
LiveConnect	http://developer.netscape.com/library/documentation/communicator/jsguide4/livecon.htm
LP Animation	http://weber.u.washington.edu/~cvj/animalp.html
Lycos	http://www.lycos.com/
Mathematical Programming Glossary	http://www-math.cudenver.edu/~hgreenbe/glossary/glossary.html
MathML	http://www.w3.org/TR/WD-math-970515/section2.html
Mathsoft	http://mwww.mathsoft.com
Media type	ftp://ftp.isi.edu/in-notes/iana/assignments/media-types/media-types
Michael Trick's OR page	http://mat.gsia.cmu.edu/
Microsoft	http://www.microsoft.com/
MIME	http://www.fokus.gmd.de/mtl/mime/entry.html
NEOS	http://www.mcs.anl.gov/otc/Server/
Netscape	http://home.netscape.com
Network Computing journal	http://techweb.cmp.com/nc/docs/default.html
OpsResearch.com	http://opsresearch.com
OR Data Library	http://\-mscmga.ms.ic.ac.uk/\-info.html
ORB	http://www.omg.org/news/begin.htm
Plug-in	http://developer.netscape.com/one/plugins/index.html
Practice Online	http://silmaril.smeal.psu.edu/pol.html
RDF	http://w3c.org/Metadata/RDF/Overview.html
Research Publications using Java	http://dubhe.cc.nps.navy.mil/~gbradley/JavaPaper/versionsJava.html
RIOT	http://riot.ieor.berkeley.edu/riot/
Server	http://w3c.org/WWW/
SIF	http://www.rl.ac.uk/departments/ccd/numerical/lancelot/sif/sifhtml.html
SMTP	whatis.com/smtp.htm
Sun Microsystems	http://www.sun.com/
TSP (Jones)	http://weber.u.washington.edu/~cvj/tsp/tspnew.html
URL	http://www.w3c.org/pub/WWW/Addressing/
VBScript	http://www.microsoft.com/vbscript/
W3C	http://www.w3c.org/
WORMS	http://www.maths.mu.oz.au/~worms
XANADU	http://www.xanadu.net/the.project
XML	http://w3c.org/XML

References

1. APPLE COMPUTER, INC., 1988. *HyperCard Script Language Guide*, Addison-Wesley, Reading, MA.
2. K. ARNOLD and J. GOSLING, 1996. *The Java Programming Language*, Addison Wesley Publishing Company, Reading, MA.
3. S. BAKER, V. CAHILL, and P. NIXON, 1997. Bridging Boundaries: CORBA in Perspective, *IEEE Internet Computing 1:6*, 52–57.
4. J. BEASLEY, 1990. OR-library: Distributing Test Problems by Electronic Mail, *Journal of the Operational Research Society 41*, 1069–1072.
5. T. BERNERS-LEE, 1989. Information Management: A Proposal, Technical Report, European Laboratory for Particle Physics (CERN).
6. T. BERNERS-LEE and R. CAILLIAU, 1990. World Wide Web: Proposal for a Hypertext Project, Technical Report, European Laboratory for Particle Physics (CERN).
7. T. BERNERS-LEE, R. CAILLIAU, A. LUOTONEN, H. NIELSEN, and A. SECRET, 1994. The World-Wide Web, *Communications of the ACM 37*, pp. 76–82.
8. H.K. BHARGAVA, M. BIEBER, and S. KIMBROUGH, 1988. Oona, Max, and the WYWWYWI Principle: Hypertext and Model Management in a Symbolic Programming Environment, in *Proceedings of the Nineth International Conference on Information Systems*, Minneapolis, MN, J.I. DeGross and M.H. Olson (eds.), 179–192.
9. H.K. BHARGAVA, R. KRISHNAN, M. CASEY, D. KAPLAN, S. ROEHRIG, and R. MÜLLER, 1997. Model Management in Electronic Markets for Decision Technologies: A Software Agent Approach, in *Proceedings of the Thirtieth Hawaii International Conference on System Sciences*, Maui, HI, R. Sprague (ed.), pp. 1–11.
10. H.K. BHARGAVA, R. KRISHNAN, and R. MÜLLER, 1997. Decision Support on Demand: Emerging Electronic Markets for Decision Technologies, *Decision Support Systems 19*, 193–214.

11. H.K. BHARGAVA, R. KRISHNAN and R. MÜLLER, 1997. Electronic Markets for Decision Technologies: A Business Cycle Analysis, *International Journal of Electronic Commerce 1*, 109–127.
12. H.K. BHARGAVA and C.G. TETTELBACH, 1997. A Web-Based Decision Support System for Waste Disposal and Recycling, *Computers, Environment, and Urban Systems 21*, 47–65.
13. J. BISSCHOP and A. MEERAUS, 1982. On the Development of a General Algebraic Modeling Language, *Mathematical Programming Study 10*, pp. 1–29.
14. N. BORENSTEIN, 1993. MIME: A Portable and Robust Multimedia Format for Internet Mail, *Multimedia Systems 1*.
15. C.M. BOWMAN, P.B. DANZIG, D.R. HARDY, U. MANBER, and M.F. SCHWARTZ, 1994. The Harvest Information Discovery and Access System, in *Proceedings of the Second International World Wide Web Conference*, Chicago, Illinois, 763–771.
16. D. BREWER, 1997. *Netscape One Sourcebook*, John Wiley & Sons, New York, NY.
17. V. BUSH, 1945. As We May Think, *Atlantic Monthly 176*, 106–107.
18. D. CHAPPELL, 1996. *Understanding ActiveX and OLE*, Microsoft Press, Redmond, WA.
19. S. CHATTERJEE, M. PARAMASIVAM, and W.J. YAKOWENDO, 1997. Architecture for a Web-Accessible Simulation Environment, *IEEE Computer 30*, 88–90.
20. D. CONNOLLY, 1997. XML: Principles, Tools, and Techniques, *World Wide Web Journal 2*.
21. J. CZYZYK, J. OWEN, and S.J. WRIGHT, 1997. Optimization on the Internet, *OR/MS Today 24*, 48–51.
22. J.J. DONGARRA and E. GROSSE, 1987. Distribution of Mathematical Software via Electronic Mail, *Communications of the ACM 30*, 403–407.
23. T. DOWNEY and J. MEYER, 1996. *The Java Virtual Machine*, McGraw Hill, New York, NY.
24. E. EVANS and D. ROGERS, 1997. Using Java Applets and CORBA for Multi-User Distributed Applications, *IEEE Internet Computing 1*, 43–58.
25. D. FLANAGAN, 1997. *Javascript: The Definitive Guide*, O'Reilly & Associates, Sebastopol, CA.
26. R. FOURER, D.M. GAY, and B.W. KERNIGHAN, 1990. A Modeling Language for Mathematical Programming, *Management Science 36*, pp. 519–554.
27. M. GAGLIARDI and C. SPERA, 1997. A Java DSS for Solving University Scheduling Problems, Technical Report, University of Siena, Siena, Italy.
28. J. GOSLING, 1997. The Feel of Java, *IEEE Computer 30*, 53–57.
29. I.S. GRAHAM, 1996. *The HTML Sourcebook: A Complete Guide to HTML 3.0*, John Wiley & Sons, New York, NY.
30. W. HERSHEY, 1987. GUIDE (Hypertext Package), *BYTE 12*, 244–246.
31. S. HILLIER and D. MEZICK, 1997. *Active Server Page Programming*, Microsoft Press, Seattle, WA.
32. S. HOLZNER, 1997. *XML Complete*, McGraw-Hill, Boston, MA.
33. M. JEUSFELD and T. BUI, 1995. Interoperable Decision Support System Components on the Internet, in *Proceedings of the Fifth Workshop on Information Technologies and Systems*, Amsterdam, Holland, S. Ram and M. Jarke (eds.), RWTH Aachen, Fachgruppe Informatik, 56–67.
34. R. KHARE and A. RIFKIN, 1997. XML: A Door to Automated Web Applications, *IEEE Internet Computing 1*, 78–87.
35. D. KRIEGER and R.M. ADLER, 1998. The Emergence of Distributed Computing Platforms, *IEEE Computer 31*, 43–51.
36. R. KRISHNAN and R. PADMAN, 1997. On Using Web Technologies to Architect DSS: The Case of Support Requirements Planning, in *Proceedings of the Fourth International Society for Decision Support Systems Conference*, Y. Pigneur (ed.), Lausanne, International Society for DSS, 257–276.
37. G. MINTON, 1997. IIOP Specification: A Closer Look, *Unix Review 14*, 41–50.
38. T. NELSON, 1990. On the Xanadu Project, *BYTE Magazine 15:9*, 298–299.
39. R. ORFALI, D. HARKEY, and J. EDWARDS, 1996. *The Essential Distributed Objects Survival Guide*, John Wiley & Sons, Inc., New York, NY.
40. J.K. OSTERHOUT, 1998. Scripting Languages: Higher-Level Programming for the 21st Century, *IEEE Computer 31*, 23–30.
41. J. SIEGEL, 1996. *CORBA Fundamentals and Programming*, John Wiley & Sons, New York, NY.
42. M.S. SODHI, 1995. An OR / MS Guide to the Internet, *Interfaces 25*, 14–29.
43. K. SRINIVAS, V. JAGANNATHAN, Y.V.R. REDDY, and R. KARINTHI, 1997. Java and Beyond: Executable Content, *IEEE Computer 30:6*, 49–52.
44. W.R. STEVENS, 1996. *TCP/IP Illustrated, Volume 3*, Addison Wesley Longman, Reading, MA.
45. A. TANNENBAUM, 1995. *Distributed Operating Systems*, Prentice Hall, Inc., Saddle River, NJ.
46. E. VAN HERWIJNEN, 1994. *Practical SGML*, Kluwer Publishers, Dordrecht, The Netherlands.
47. A. VAN HOFF, 1997. The Case for Java as a Programming Language, *IEEE Internet Computing 1*, 51–56.