World-Wide Web: Whence, Whither, What Next?

Henning Schulzrinne
GMD Fokus
schulzrinne@fokus.gmd.de

1 Introduction

In a time span of about five years, the World-Wide Web (WWW) [1] has become, next to electronic mail, the most popular Internet application. It has been a major contributor in turning the Internet from an obscure data network for scientists and computer programmers to a household word. The World-Wide Web allows to retrieve text and multimedia objects from servers located throughout the world, with objects connected by hypermedia links. This article aims to provide a snapshot of the World-Wide Web after about half a decade, speculating at the same time where this young medium might be improved and which directions it might take from a technical perspective.

Like most (successful) Internet technologies, the underlying central functionality of the World Wide Web (WWW) is rather simple: a naming mechanism for files (the universal resource locator or URL), a typed, stateless retrieval protocol (HTTP) and a minimal formatting language with hyperlinks (HTML). Building a minimal "web retriever" in a string-oriented language like Tcl or Perl requires a few dozen lines of code, with even less effort to build a server. Indeed, simply telnet'ing to port 80 and issuing a command like `GET index.html` is sufficient to retrieve documents. All of the basic technologies were around prior to the "invention" of the world-wide web, generally credited to Tim Berners-Lee and Robert Cailliau at CERN around 1989[1]. However, the major accomplishment was not an individual protocol, but rather the integration of disparate pieces into a new, more powerful way of using networks. However, only after the original ASCII-only browser was replaced by one based on X (namely Mosaic from the National Center for Supercomputer Applications at the University of Illinois), did the world-wide web really take off. While it was originally conceived to integrate existing retrieval and access mechanisms, in particular, the file transfer protocol (ftp), gopher as a menu-oriented retrieval system, and telnet for remote login and interaction with databases, the core WWW protocol (http) has far surpassed usage of all three of these. There are other reasons for the rapid proliferation of WWW, making its rise, in hindsight, a bit less surprising than R. Lucky describes [2]: The technology works reasonably well for access speeds from 2400 baud modems on up since it retrievals can be restricted to text only, and newer Internet users, accustomed to graphical user interfaces, are far less tolerant of the command-based interfaces of traditional retrieval mechanisms like ftp or telnet. Also, in the early 90's, the lowest common denominator computing platform shifted from terminals and DOS-based PCs to PCs running windowing systems, X-terminals and workstations, allowing rapid uptake of the WWW-based multimedia content, while the basic functionality remained accessible to those still restricted to ASCII. Also, the Internet itself did not have to offer any new capabilities or "service models", beyond a reliable domain name system. Since WWW retrievals are TCP-based, they share the available bandwidth reasonably fairly, and require no new resource allocation mechanisms in the network. Finally, the cost of entry for "consumers" and

[1] See http://www.w3.org/hypertext/WWW/History.html for a timeline of WWW developments.

“providers” alike was extremely low, because the software was (and is) largely free and Web usage, for corporate and university users, incurs no additional network charges. Only now is the sheer quantity of WWW transfers one of the major forces for the increasing congestion in many areas of the Internet, particularly in Europe and the trans-atlantic connections.

This article will try to present a survey of some of the open areas within the WWW framework, both those which are the subject of current on-going standardization efforts, and those which may impose longer-term fundamental limitations on the WWW. We will investigate the three principal components, namely the primary transfer protocol (HTTP) in Section2, HTML as the germane web data type in section 3 and URLs as its naming and addressing mechanism in Section 4. Some ideas on how browsers might develop are presented in Section 5. Section 6 gives some background on the impact of WWW on the Internet and how it can be made to scale. Section 7 points out some more long-term limitations of the WWW model and how other applications could be integrated with the Web. The final section summarizes some of the new applications and alternatives for information delivery that might be viable in the near term. This article does not discuss the important topic of how to organize the resources that can be accessed via the world-wide web; Lynch [3] offers a survey of these more generic issues.

2 HTTP

The “native” WWW protocol for data retrieval is the hypertext-transfer protocol, HTTP [4]. HTTP is an application level-protocol that currently is used probably exclusively with the Transmission Control Protocal (TCP), although there is no reason it could not be used with other reliable transport protocols. HTTP is a client-server protocol: The client, typically a WWW browser, asks the WWW server for some information via a GET request or, less frequently, transfers information to the server. Currently, version 1.0 (and rarely, version 0.9) of the protocol is in use, with version 1.1 being worked on within the IETF.

typical HTTP request is shown in Fig. 1. It consists of the request line and a number of parameter-value header lines describing the request, e.g., what kind of data types are acceptable as answers. Each request is handled by its own TCP connection and is completely independent of any previous request, i.e., for each document and graphics on a page, the web browser opens a new connection to the web server. The server closes the connection to signal that the data has been transferred completely. Given this description, you can build your own web browser by using a telnet client to connect to port 80 (the standard WWW port) on any web server and typing the first GET line in Fig.1 followed by a blank line. The information exchanged by HTTP can be any data type and is not limited to HTML.

This simple protocol has the advantage that clients and servers can be stateless, that is, they do not have to remember anything beyond the transfer of a single document. It is also pseudo-anonymous, in that the server sees only the host IP address of the client. Two requests from the same IP address are likely to come from the same individual, but, with firewalls and multi-user systems, certainly do not have to. For retrieving documents, these two properties are sufficient and desirable. But they make it difficult to, say, maintain a “virtual shopping cart” for a particular visitor, have an advertisting agency track how visitors move through their site or to customize pages for specific visitors. There are at least three approaches to add state:

- One could generate custom links from a home page, so that instead of simply pointing to “chapter1.html”, a link would point to “17348/chapter.1.html”, where the latter has the same content, but is specific to visitor 17348. While this scheme works without any support from the browser, it also defeats any caching mechanisms. It also does not work beyond a single visit to a home page. Generating custom pages also requires somewhat more processing by the server.

- Forms can contain “hidden fields”, which are not visible to the user, but can carry values identifying a particular visitor.
- Netscape has proposed “HTTP cookies”, where the server response for a page contains a parameter-value pairs, an expiration date and a URL range. The client should then store these and return appropriate parameter-value pairs when accessing the given range of URLs.
- HTTP could be extended to maintain a single TCP connection across several requests [5]. While this is desirable for performance reasons (see below), it does not work across several visits separated by a larger time span.

```
GET index.html HTTP/1.0
If-modified-since: Sat, 29 Oct 1994 19:43:41 GMT
Accept: audio/*;q=0.2, audio/basic
```

Figure 1: Simple HTTP Version 1.0 GET request

An observant reader will have noticed in Fig. 1 that the GET request did not contain the whole URL, but rather only the file name part, `index.html` in this case. While this saves a few bytes and slightly simplifies parsing by the server, it causes problems for the popular approach of “virtual hosting”, where a single server “fronts” for a number of URLs, e.g., a host host.ipp.com may have aliases www.company1.com, www.company2.com, etc. Unless it acquires several IP addresses, the server on host.ipp.com cannot tell which of its hosted URLs a request is for. HTTP version 1.1 fixes this by making it mandatory to include the host part of the URL in the request header.

One of the greatest current problems with HTTP is its relative inefficiency. For a typical HTML page, the client first retrieves the HTML page itself, then discovers the potentially dozens of images contained within the page and issues a separate HTTP request for each. (Cunha et al. [6] have measured the average HTML page retrieved at 6.4 kBytes, with roughly 7 images per page.) Each HTTP retrieval requires at least one round trip time, plus the TCP connection setup of 3 one-way delays and a connection tear-down of another two round trips. Since there is some overlap possible, the smallest retrieval time is 4 round trip delays. Thus, for typical long-distance round-trip times of around 300 ms, the minimum latency even with an infinitely fast link would be 1.2 seconds. In addition, the TCP congestion control mechanism [7] reduces the achievable throughput until the window has been fully opened, again without regard to the link speed. Experiments [8] show that using a TCP connection to transfer only 2 Kbytes for a link with 70 ms of round-trip delay results in a throughput less than 10% of the best-case value, increasing to only 50% with 20 kByte transfers.

Typical browsers open several TCP connections at once. While this avoids some of the latency induced by the numerous figures on a page, it also can easily overload slow links, since TCP congestion control, being based on acknowledgements, cannot control the emission of the first data packet[2]. Thus, most browsers seem to restrict parallel retrievals to about four, also to avoid unfairness (since a user can increase her overall throughput by opening several connections in paralell).

If an item on a page has access protection, an additional exchange is necessary: the server refuses the first request, telling the client how to authenticate itself, and then the client tries again, with the proper credentials.

[2] Particularly the heavily loaded transatlantic links seem to suffer under this high number of short connections.

To avoid data corruption due to sequence number reuse, the operating system of TCP endpoints needs to maintain TCP state information for a few minutes after the connection has been closed. For a busy server, this can add up to thousands of connection records. Some of the TCP-related latencies and the connection record problems can be reduced by a modified version of TCP, called transactional TCP (T/TCP) [9].

HTTP 1.1 suggests extensions that allow a single TCP connection to stay open for several HTTP transfers. This also allows servers to proactively transfer a HTML page, say, with all its icons, without the client explicitly asking for each.

Browsers also reduce perceived latency by showing text before all the images have been retrieved. This, however, is only possible if the image contains hints about the size to be used. This feature is an HTML 3 extension.

HTTP is a textual protocol, that is, all headers are transferred as (mostly ASCII) text. This simplifies the writing of simple browsers, but also increases parsing costs for high-speed servers since the server has to look at every single character to pick apart the header and might have to do some string processing like line continuations, escaping of special characters and date parsing. The textual representation for HTTP is also fairly verbose, so that the headers can easily be larger than the actual content transferred. As with all Internet textual protocols, the protocol gets rather more complicated once more than one character set is to be supported. For HTTP, this affects mostly URLs (see Section 4) and a few other less-important fields.

The most frequent HTTP operations are GET, PUT and HEAD (to get header information only). HTTP 1.1 also defines operations to have HTTP take over the remaining functions of the File Transfer Protocol (ftp): the ability to delete, link and rename files. Since HTTP offers automatic data compression, format, language and character set negotiation and avoids the need to maintain two TCP connections, it seems likely that ftp will gradually be replaced by HTTP. (Currently, there is no explicit HTTP directory command; rather, the client has to rely on the server to generate a HTML representation of a directory listing when the URL points to a directory rather than a file. It would be preferable to have a structured directory representation so that the client could decide on the amount of detail, date format or content representation to be rendered.)

Probably the most notable feature of HTTP separating it from, say, ftp, is its ability to negotiate media types, and (in version 1.1) even character sets and language. The negotiation can be preemptive, where the client indicates preferences for different media types, or reactive, where the server returns a list of possible types. Note that media type negotiation implies that a URL does not necessarily refer to one document, but a single URL could point to the English, German and French versions, for example. It also avoids having to tailor a server's response to a particular browser make, model and operating system, so that, say, a browser would automatically get the proprietary compression format, while others can obtain a lower quality "standard" rendition. While these capabilities are rather convenient in many circumstances, it also complicates caching. In an interesting omission, price is not one of the limits that can be imposed, only size.

The content negotiation can be rather complex, taking into account five different quality scales, and seems unlikely to be amenable to a readily comprehensible user interface. A basic problem is that the client does not know which types the server considers exchangeable for the same URL. If, for example, the user assigns a higher preference to MPEG audio over telephone-quality audio and these desirability values happen to be higher than the one for text, a clever server may interpret that as saying that the text should be read (which could actually be quite sensible for providing access to a blind audience). There is also an efficiency problem in that the client has to send its complete preference description for every retrieval, since it cannot know which are completely irrelevant. Needless to say, this has not been implemented by any browser or server that the author is aware of. (For servers, it breaks the convenient notion of mapping URLs more-or-less one-to-one to file names.) For some media types, notably audio and video, additional parameters such as supported sampling rates or pixel depths would be desirable [10], although it seems likely that most systems capable of displaying multimedia objects will soon have the maximum useful capability of 16-bit audio and 24-bit pixels.

There are some efforts to replace HTTP with a binary, ASN.1-based version that supports pipelining of several objects and asynchronous retrievals[10]. Since T/TCP and the extensions of HTTP discussed will probably reach a large fraction of the throughput of a revised protocol and the textual parsing overhead is only relevant for the relatively small number of servers, displacement of HTTP by a different protocol does not seem imminent.

3 HTML

HTML is a mixture of a presentational and descriptive markup system [11]. Presentational markup systems indicate *how* some text is to be rendered, e.g., in bold face of a certain font, with a given paragraph width. Descriptive markup tags the structure of a document, i.e., whether a certain piece of text is a heading, the abstract, a quotation, and the like. Descriptive markup allows the browser to render content according to the capabilities of the end system such as its screen resolution or the preferences of the reader in terms of fonts, line width, line spacing, and the like. Thus, descriptive markup is inherently suitable for a heterogeneous environment like the Web, where braille readers, ASCII terminals and small-screen PDAs coexist with 19" workstation screens. However, descriptive markup seems to work only for a limited set of documents and has been most successful in relatively technical fields, e.g., for the coding of scientific articles or technical standards. Most of the web pages use the HTML descriptive elements to achieve layout and font effects. For example, the lower-ranked headings are used to produce small print.

LaTeX [12] and some nroff macro sets also have both presentational and descriptive markup properties, but they differ from HTML in that they are programmable, i.e., contain the ability to write (small) programs, declare and invoke macros or functions, or conditionally execute certain parts of the description. This adds flexibility and significantly eases the global manipulation of text. For example, it is possible to define a function that takes a name, email address and other information as arguments, and then define that function once to render it in different ways (say, as rows in a table or as a list). It is also easy, for example, to render the same text as a single column or as several columns, without changing the text itself. HTML does not offer this capability. For this and other reasons, HTML is often generated automatically from more capable systems. While this is reasonably straightforward for standard sectioned text, it is rather difficult as soon as tables, equations or columnar output are desired.

HTML intentionally does not contain these "programming" capabilities since they greatly complicate parsing at the receiver. They would make it very difficult to simply ignore tags that a client does not (yet) know. Once they reach the client, HTML documents are also self-contained, that is, they do not reference any external definitions and thus avoid the problems of missing or incompatible external references. (It should be noted that some servers can dynamically piece together a HTML document with so-called server-side includes.)

Unfortunately, the descriptive capabilities of HTML are limited mostly to low-level constructs such as emphasis or indented lists. There are no standardized mechanism to indicate common document parts such authors, abstract, keywords, table of content, references. (HTML 3.0 provides figures with captions.) Abstract, author and keyword elements would also significantly improve automatic indexing and searching. In addition to the "TITLE" element, it would be very useful to have an element indicating the home (front) page for a particular document, so that indices could point to that rather than some random location in the text. In fairness it should be noted that a generic, content-oriented encoding of even English text [13] is rather complicated. A particularly relevant omision, however, is document navigation. Almost all HTML pages contain

version	“typical” browser	features
1.0	historic	headers, emphasis, lists
2.0	Mosaic	inline images (IMG), forms
”2.1”	Netscape, Microsoft	alignment, tables, appearance
3.0	Arena	math, floating figures, style sheets

Table 1: History of HTML

navigation features, that is, links to a “previous” or “next” document, the home page, an index, and the like. However, despite the similar functionality, each document uses its own sets of buttons and links, requiring re-learning of the “user interface” for each document. HTML elements for navigation would allow browsers to present a uniform, user-configurable interface.

HTML is oriented towards display rather than printing or storage. First, HTML follows a “scroll model”; it has no notion of pages. This is appropriate for display, but makes it difficult to print pleasing output. Also, the appropriate delivery units for display are small, to reduce transfer latency and scaling, but this means that a single article has to be pieced together from numerous HTML files to be printed or saved locally, each piece likely containing rather distracting navigation icons. The print quality is further diminished by the lack of a vector-based graphics standard within the world-wide web; the common graphics formats GIF, JPEG and X bitmaps all render bitmaps and usually look rather poor on the printed output where the resolution is three to ten times higher. Chaining of page sequences and standardized HTML navigation tags would avoid the necessity of maintaining a separate viewing and printing representation of the same text.

Because of its non-procedural nature, HTML parsers can be rather forgiving of any HTML coding violations. A large fraction of web pages, even those professionally authored, contain coding “mistakes”, often used to achieve particular layout effects on popular browsers.

Only since November 1995 has there been an IETF standard [14] for HTML. However, a large fraction of HTML pages use various supersets of this standard, e.g., to display tables, colored and textured page backgrounds or fonts of different sizes. Many commercial servers have started to interpret the HTTP header field identifying the browser software release to deliver custom-tailored renditions of their material[3]. Clearly, this does not scale as the number of browsers increases.

Each new browser and browser release seems to introduce a new set of HTML tags or new parameters to older tags. While browsers usually just skip unrecognized tags, a conscientious content developer still has to test the material with a bevy of different browsers to make sure it looks acceptable on all. Most of the new tags seem to be geared to satisfy advertisers. This has yielded the blink tag, marquees that stay on the screen even when the text is scrolled and audio that plays automatically (and continuously) whenever a particular page is viewed. There seems to be little urgency in getting tags for mathematical typesetting into browsers, but it has been defined as part of HTML 3.0. From both the descriptive and the presentational camps, there are efforts to integrate other text formats into the WWW environment. This usually just requires that there is a way to embed links in that text format and somehow integrate the display engine with a browser. There is also work on building full-fledged SGML browsers that could be customized with a document type definition to render any SGML document, including those in HTML. However, the syntax of SGML was clearly not designed by compiler writers. It is not representable as a regular expression or other simple languages. SGML allows a number of abbreviations and tag omissions whose full support is rather cumbersome. A document type definition requires arcane expertise and is unreadable (and probably semantically incomplete) without

[3] This has even led a large PC software vendor to identify its product to the server as that of its competition.

copious comments. It also has some strange remnants concerning spaces, character sets and line endings dating back to punched cards and fixed-length records. Despite its shortcomings, it is unlikely that it will be replaced any time soon.

A somewhat more modest (and complementary) effort adds so-called *style sheets* to HTML. Style sheets describe how a particular tag is to be rendered, e.g., that a heading is to have a certain color or font size. This relieves some of the pressure on HTML to provide ever new tags for new presentation forms, yet allows page designers and readers control over the rendering of material. Currently, many high-end graphical pages use bit maps for full layout control. Bit-mapped pages and additional text types have one severe disadvantage: they cannot be easily automatically indexed by various "web spiders" and search engines. Also, it is not easy to have a single client-side search if there are several different text formats.

HTML has basically three kinds of hyper links: The first wraps a block of text or an image in an anchor HTML tag. Typically shown underlined, it leads to the replacement of the current web page by the one the link points to. Since there is only one display, it imposes a linear navigation sequence, where it is difficult to retrace one's steps beyond a few mouse clicks. (History lists are also generally only linear, so that side excursions quickly let one loose one's place in the web of links.) The second, not yet standardized, displays the document pointed to in a new browser window. Finally, the IMG tag, not generally perceived as a link, displays some image in-line, as part of the document. Only recently have there been efforts to generalize the in-line display so that most any data type, including active content like Java applets or animations, can be part of a document.

Also are more "light-weight" anchors invoked simply by leaving the pointer on a page position and displaying a small, transient window with a small amount of information. These would be appropriate for displaying, say, a brief help message for a button or the definition of a word, similar to the "balloon help" feature in some operating systems. Similarly, it would be helpful to be able to define a default link so that a region of text is passed to a search engine defined by the reader or the page creator, for example a dictionary or a translation engine.

Currently, links can only be added by the author of a document. A different model of hypertext separates document and links, so that a single document can feature different sets of links and links can be updated without touching the document itself. This also allows annotations to be added by readers. Annotations were an original experimental feature of WWW which seem to have disappeared almost completely. Certainly, anything but personal annotations raise all kinds of liability issues. Since the content of the same URL often changes or a single URL can refer to several documents, as discussed earlier in the context of content negotiation, it is also difficult to come up with a reliable, persistent annotation mechanism.

Within the Internet, HTML is already replacing a number of similar text systems, such as MIME rich text [15] or nroff manual pages.

While there have been extensions of SGML to the presentation of continuous media (HyTime), they appear overly complex, yet still do not offer the full programming flexibility of a client-side programming and scripting language such as Safe-Tcl or Java. Proprietary solutions, particularly those that have widely-deployed authoring tools, seem to be dominating this application.

4 URLs and URNs

Universal resource locators (URLs) are just one of the names used to designate objects within the world wide web. The whole family is technically known as universal resource identifiers, of which URLs [16, 17] name the physical location of an object, URNs the identity without regard to location, and Uniform Resource Citations (URCs) describe properties of the object. Only URLs are in any widespread use. They consist of an identifier for the protocol (http, ftp, ...), the network location (host and port) and a path name within that named server. While that path is typically mapped directly onto a file name by the server, this is a pure server convenience. A server could just as well use this path as a key into a database or as a function name and arguments to dynamically generate a document.

Note that a URL says nothing about the type of object it points to, even though most URLs give some informal hints to the initiated. An example of a URL is `http://www.w3.org/`. Most home-page URLs pass the "business card test" whose failure doomed X.400-style email addresses, but they are still punctuation-heavy for reading over the telephone or the radio. This has lead to the http being dropped as the default and browsers assuming that any URL starting with `www` is to be accessed via HTTP.

Despite its apparent hierarchical nature, the domain name space is practically flat, as the large majority of names are drawn from the `.com` domain. Even countries as large as Germany have a flat second-level domain name. Since registration costs only about $50 a year, companies have taken to register every single product of theirs as a domain name. This is relatively harmless, but causes inevitable clashes since the Internet domain name space lacks the distinction of type of trade, country and region that allows reuse of popular names in the area of traditional trademarks. Thus, the regional naming structure found in the `.us` domain will have to become the common case rather than the exception.

Letting a URL name a single host has a number of disadvantages: If that host crashes or is overloaded, the retrieval fails. Large sites have found a number of work-arounds, like DNS aliases, or distributing different name information to different subsidiary name servers.

In a better system, the domain name service used to transform Internet host names into IP network addresses would offer a per-domain service listing, extending that used for email today. Most organizations have a so-called mail exchange record which names one or most hosts responsible for all electronic mail for that domain. A mail transport agent goes down the sorted list until it finds a working remote host. A similar system could be introduced for WWW service. There are also current proposals to include a host's geographical location in its DNS record. Given a list of alternate, equivalent servers and their geographical location, browsers could look locally first before sending a request across the ocean[4].

URLs were often considered to be temporary artifacts until a more powerful naming mechanism could be deployed. However, URLs seem to experience the same longevity as email addresses, also long condemned to be replaced by lookups by name in a global X.500 directory. The basic idea has permanent names, URNs, lead through a directory to URCs describing their properties and URLs listing their current location. However, somebody has to maintain and pay for this directory. The same URLs that change frequently are also likely to be the ones that their maintainer forgets to update in the directory since she has lost access or to interest in the content. Finding documents would only be made easier if the number of registries is relatively small and if authors of documents, including all the mixed-media documents, VRML worlds, and the like, can agree on a common set of meta data (URCs). For electronic publications of well-known publishers and scientific societies, the address of individual documents within their library is not very likely to change. Thus, it appears unlikely that users will be willing to pay either in monetary units, the additional look-up delay or the decreased reliability to avoid "stale" links. For some forms of documents, in particular university technical reports and scientific periodicals, an electronic directory seems appropriate and likely to take shape, since the organizational infrastructure and incentive is largely in place. Also, it would indeed be very useful to have a "repository of last resort" for digital artifacts, the equivalent of the U.S. Library of Congress.

[4] This assumes that geographical distance is equal to network distance. However, the shortest network distance between two European cities is often through Washington, D.C.

5 Browsers

One of the factors driving the success of the WWW is its ability to attract both content providers and serve as a base for new applications. While in the past, a corporate library may have written its own user interface to its library catalogue, it now appears much easier to base this on a web server and browsers. This avoids having to write a new user interface for each new client platform or operating system, and automatically lets the system participate in advances like security.

There seem to be two contradictory directions for WWW applications: the browser that can do everything and having every application have WWW capabilities. The latter makes it difficult to integrate several data types, but it is certainly desirable to be able to have applications recognize URLs and control a browser to view the URL content. Browsers are already incorporating mail tools, news readers and very primitive file system managers. Soon, they will also feature text editors, at least for HTML. While this integration has the advantage of hiding the difference between local and remote operations to a large extent, it also leads to huge applications and less choice between vendors. Other mechanisms to integrate different applications are currently being deployed, for example, so-called plug-ins, where applications communicate directly with the browser and share some of its window area.

6 Network Impact and Requirements

Since the Internet has gone commercial, it is difficult to ascertain exactly what fraction of wide-area traffic is generated by WWW browsers and HTTP. Informal indications indicate, however, that the fraction is well above half the total traffic. WWW stresses the Internet in that browsing has low latency requirements, as a human is waiting impatiently. The data transfers can be anything from a short burst for a small image to several tens of megabytes for a video or audio clip. While text and images on a single page usually come from the same host, hyperlinks traversed have no real spatial correlation. The low spatial correlation will strain any IP-over-ATM mechanism that attempts to set up individual switched virtual circuits for each web retrieval.

For any of the more popular or high-bandwidth services, the world-wide web can only scale if information content is mirrored and cached. A mirror provides a complete copy of some server, with the master server explicitly updating its mirrored copies. Mirrors are trusted, at least to some extent, by the data source. Mirrors can be configured by manual selection or, possibly, through a domain name service mechanism. If URN directories ever come into widespread use, these could obviously also return several candidate locations for a document. A busy server could also send a redirect answer to a browser, but it better be sure that the server it is redirecting users to, is not busy or down.

Caches are placed between client and WWW server, for example on the premises of a company or a university, and have no direct trust relationship with the server. Typically, the browser is manually configured with the location of a single cache server. Caches may connect to other caches. The client connects to the cache and requests the document, which the cache server either has stored already or in turn retrieves from the actual server, keeping a copy to satisfy future queries. Caches can be quite effective [6], but can also easily become bottlenecks and are then avoided by users. Also, many documents that look static are actually generated anew for each request and thus are not cacheable. Since many servers want to keep a running hit count for their offerings, they actively defeat caching by setting expiration dates as immediate or otherwise marking information as non-cacheable. Thus, some mechanism has to be found to enable caches while providing accurate access counts to the main server. Also, documents with access authorization cannot currently be cached; in the future, this may be a large fraction of popular documents. If a hiearchy of caches is to be built, there has to be a routing mechanism that determines which cache(s) should be queried for a document. For that, a client may need know the actual location of the document so that it can avoid contacting a cache where the actual location of the resource is closer. It remains to be proven how much can be gained with multiple cache levels, as there is probably a stronger affinity of interest on the level of a company or department than a whole country.

7 Limitations of the WWW Model

Despite all the press and publicity, the WWW model is currently rather limited: retrieve an object (text, audio or video) and render it. Even with forms, the capabilities of a Web page are roughly that of a page-oriented mainframe terminal, with some graphics spice added. Some inherent capabilities of the web model have not been developed, in particular the ability to store content through the server. This could be quite useful for collaboration and for maintaining corporate information within intranets[5] and across fire walls, particularly once client authentication is better developed. It is likely that future browsers will cease to be display-only and allow users editing and storing back documents, at least those written in plain ASCII or HTML. This would make them more competitive with other asynchronous computer-supported cooperative work environments.

Client-side interaction is currently limited to filling out simple forms and clicking on buttons and bitmaps (so-called image maps). There are some efforts to provide more direct feedback to the user rather than having to fill out the whole form and then being told that some field is wrong or clicking on parts of a bit map without any feedback as to what, if anything, might happen. Client-side imagemaps store the coordinates of sensitive areas so that the browser can provide local feedback. Client-side scripts or applets would allow to provision of interactive help or correctness checks. These could also provide richer user interfaces, so that the content of a page can adapt to user action rather than simply by reloading a page from the server.

The integration of multimedia is currently very primitive: A video or audio file is transferred via HTTP and then played with some sufficient buffering or from local temporary storage. Playing audio and video as it arrives from the network avoids waiting minutes for it to download completely (only to discover that it probably was not worth the wait). However, unless the minimum network bandwidth is the access bandwidth, assumed known, the user has no way to choose an appropriate encoding or know ahead of time how long the media content needs to be buffered to assure playout without interruption. A number of solutions can be envisioned. First, non-TCP protocols such as Realtime Transport Protocol (RTP) [18] that provide congestion feedback can be used to adaptively tune their buffering and encoding to the available bit rate. In addition, for both TCP and other protocols, resource reservation could guarantee a minimum bandwidth [19]. Adaptive applications require no changes to the Internet, but are still subject to glitches when adapting quality downward. RSVP, the IETF Internet resource reservation protocol [19, 20], will likely not be deployed widely for a number of years, as it requires router modifications, some form of authentication and usage-based billing to prevent abuse.

Any type of interactive games (often called "twitch games) will likely bypass the web protocols completely, even though they might use it to locate services or partners to play with. The same is also true for multimedia-on-demand applications. The existing media-on-demand applications use the WWW browser only to present some VCR-style controls and a play list. There is no possibility of embedding "active areas" (i.e., hyperlinks) within the audio or video content or synchronize textual and video content.

For content delivery, the web is "pull-only", that is, the browser has to explicitly check whether there is new content. There are some HTML extensions where the browser can be instructed to do this at preset inter-

[5] Intranets are networks within organizations that may or may not be connected to the Internet, but use some of the same technology and protocols.

vals, and browsers can be told to check a subset of the book marks for any changes, but neither is particularly efficient. There are also services that regularly poll web pages for changes and then send electronic mail to the subscribers of this page, saving unnecessary queries for popular servers. As an alternative, it would be close to optimal in terms of bandwidth usage and speed of notification to have servers send announcements to by IP multicast to all interested parties. This, however, works only for systems that are continuously connected to the Internet.

Web browsing is a solitary occupation. A visitor to a web page cannot see if there are other like-minded people reading the same page or interacting with the same 3D-world. There have been a number of proposals to create mutual awareness. The *Virtual Places* [21] protocol uses a separate server to track visitors, represented by small cartoons or pictures displayed on a custom browser. A visitor sees the movement of others on the page and can move her image to touch that of another visitor. As images touch, a packet voice connection is established. Tour buses can be used to visit web sites together, with "passengers" on the bus reachable by audio and video through IP multicast. People that are on the same page can also "chat" with each other by typing messages. However, all of these interactions only use the browser as a platform and bypass WWW protocols. Such web-based interactions might also be feasible by using scripting languages and multicast notifications of presence and location. This offers an enhanced version of the room metaphor employed in some video conferencing systems [22].

Chat rooms are quite popular on on-line services and there have been attempts to use WWW technology. Without some external protocols, this is rather clumsy, as a user would have to reload a page to see what, if anything, others have typed.

8 New Applications and the competition

The simplicity of the Web technology is in sharp contrast with some other efforts offering networked multimedia, for example those build on the MHEG description, an ISO standard [23, 24]. While MHEG tries to provide a generic multimedia model that supports data types from text to synchronized audio and video as well as dynamic user interaction elements. However, producing even simple information requires specialized authoring tools. To offer more than text, much higher access bandwidth is needed, thus limiting this to video-on-demand applications running over cable TV and set-top boxes. For many of these, it would seem to be sufficient to simply send back timing and pointer coordinate information to the server rather than having the set-top box interpret MHEG "programs". In the early 1990's, there seemed to be a rush by every cable and local telephone company to trial various forms of video-on-demand services. Some of the additional services such as home shopping and banking would arguably be similar to what is now slowly emerging on the world-wide web, but the pool of providers would have been strictly controlled by the cable or telephone operator. Experiments showed that most households only watched three or four paid movies per month. Given that the premium over walking to the neighborhood video rental store has to be fairly small (unless there is a blizzard outside), the additional revenue turned out to be rather small and not enough to compensate for the large investment in video servers, set-top boxes and network infrastructure. It remains to be seen whether cable modems, allowing packet traffic from cable headend to subscriber at around 10 Mb/s rate, shared between all subscribers on that cable segment, and a much lower upstream data rate, are a viable alternative. They would seem to offer the bandwidth for small-scale video-on-demand and near video-on-demand services via WWW protocols as well as immediate access to the Internet at large, while avoiding the customer having to purchase a special-purpose computer disguised as a set-top box. For these cable modems, very cheap "Internet appliances" or slightly enhanced video game machines may also be attractive for the large majority of households that do not own a personal computer.

However, should video-on-demand be deployed, the digital infrastructure would also be usable for Internet services. The use of WWW with a set-top box and a TV as a monitor would be possible, but not likely to find widespread use. Despite increased resolution of recent models, TVs still cannot reproduce a 640x480 pixel image, the minimum acceptable for displaying WWW pages with significant amounts of text. Clicking on hypertext links could be difficult as they are often too close together to aim at with an air mouse; an interface, similar to ASCII WWW browsers, using arrow keys to move from one hyperlink to the next, however, might be workable. Textual input would be extremely tedious unless the remote control featured a small keypad – a good role for a personal organizer with an infrared interface.

The roles of on-line services relative to the Web has seen much discussion, with some dismissing the on-line service providers as parts of a bygone area. Certainly, a number of new services seem to be heading towards being "gated Internet communities", rather than using proprietary technology, shifting their focus from providing content to providing Internet access. However, the large majority of residential users still use on-line services, with occasional forays into the Internet. On-line services not only provide access, but also

- technical customer support services;
- a basic menu of standard content;
- aggregated, averaged billing for a range of content;
- parental access control features;

Particularly the value of providing billing services should not be underestimated, for both content providers and consumers. A single, predictable bill for all content makes a service more attractive, even if an individual actually uses only a fairly constant tiny fraction of it. Besides advertising support, there does not seem to be a viable means of creating revenue for individual web sites. Most people currently do not spend enough time at a single site to make subscriptions attractive. (An average customer of an on-line service spends about six to eight hours a month on-line.) Pay-per-view services have generally not been very successful in other areas, unless content could not be obtained elsewhere. Efforts to develop micropayment schemes might provide a way to charge some small amount for each page retrieved, but nobody knows how users will react to "hearing the meter ticking". The September Project [25] is investigating some of these issues in the context of scientific publications.

On corporate intranets, Lotus Notes™ has been used for many of the same functions as the Web. While the basic access mechanisms are similar in spirit, if very different in detail, to a combination of electronic mail, Usenet newsgroups and HTTP document retrieval, the back-end differs. While most web servers base their document structure on the local file system, Notes uses a distributed database system containing compound documents. The individual databases are synchronized periodically or on demand on a per-record basis. Similar to the just developing server and client side scripting capabilities of the Web such as cgi-bin scripts, Java and Javascript, Notes has various languages and application programming interfaces to construct business applications and has tools to develop those. The web server script industry is still very much in its infancy. Its servers have the capability to integrate with corporate mainframe databases, something web servers are only just developing. The web still needs to develop the workflow capabilities offered by Notes. (Workflow capabilities automate the typical flow of forms and correspondence for anything from purchase orders to personnel record changes and travel accounting.) Unfortunately, it is likely that this capability will be just as proprietary in the Internet as in the Notes environment. Thus, for publishing relatively static documents and simple scripts, WWW and Notes offer similar capabilities, while Notes seems to have a headstart on the “backend”, that is, server, side. Recently, Notes databases have become accessible from web browsers [26].

9 Conclusion

The success of the World Wide Web has, at least in the eyes of the non-technical public, made it and the Internet nearly synonymous. However, there are many interesting Internet-based services which are largely independent of the Web, and have quite different requirements. As an example, real-time, interactive multimedia services would provide an alternative to the telephone network, immediately offering many of the services promised by the telephone carriers through Advanced Intelligent Networks (AIN), but they require extensive additions to network infrastructure, in particular some form of resource reservation and significantly enhanced bandwidth. A Web-centric Internet would be giving up one of the fundamental strengths of the Internet: the ability to quickly deploy new services.

Overall, the continued growth success of the world-wide web as a global delivery mechanism for multimedia content will ride on technical issues as much as on economic, social and political ones. Some questions that we can only raise here include: Can an advertising-only medium prosper or will there be ways (and willingness) to pay for quality content? Will jurisdictions force Internet service providers to restrict access to parts of the web which are considered in violation of local criminal statutes? Will encryption and user certification be widespread and sufficiently easy to use that electronic commerce can prosper?

References

[1] T. Berners-Lee, R. Cailliau, A. Luotonen, H. F. Nielsen, and A. Secret, “The world-wide web,” *Communications ACM*, vol. 37, pp. 76–82, Aug. 1994.

[2] R. W. Lucky, “Where die the web come from?,” *IEEE Spectrum*, vol. 32, p. 15, July 1995.

[3] C. A. Lynch, “Networked information resource discovery: An overview of current issues,” *IEEE Journal on Selected Areas in Communications*, vol. 13, pp. 1505–1522, Oct. 1995.

[4] T. Berners-Lee, R. Fielding, and H. Frystyk, “Hypertext transfer protocol – http/1.0,” Internet Draft, Internet Engineering Task Force, Aug. 1995. Work in progress.

[5] R. Fielding, H. Frystyk, and T. Berners-Lee, “Hypertext transfer protocol – http/1.1,” Internet Draft, Internet Engineering Task Force, Nov. 1995. Work in progress.

[6] C. Cunha, A. Bestavros, and M. Crovella, “Characteristics of WWW client-based traces,” Tech. Rep. BU-CS-95-010, Boston University, Computer Science Department, Boston, MA 02215, July 1995.

[7] W. R. Stevens, *TCP/IP illustrated: the implementation*, vol. 2. Reading, Massachusetts: Addison-Wesley, 1994.

[8] V. N. Padmanabhan and J. C. Mogul, “Improving HTTP latency,” in *Proc. of the Second World Wide Web Conference ’94: Mosaic and the Web*, (Chicago, Illinois), Oct. 1994.

[9] R. Braden, “T/TCP – TCP extensions for transactions functional specification,” RFC 1644, Internet Engineering Task Force, July 1994.

[10] S. Spero, "HTTP-NG architectural overview." Electronic document, Feb. 1995.

[11] J. H. Coombs, A. H. Renear, and S. J. DeRose, "Markup systems and the future of scholarly text processing," *Communications ACM*, vol. 30, pp. 933–947, Nov. 1987.

[12] L. Lamport, *LaTeX: A Document Preparation System*. Reading, Massachusetts: Addison-Wesley, 1986.

[13] C. M. Sperberg-McQueen and L. Burnard, eds., *Guidelines for Electronic Text Encoding and Interchange*. Chicago and Oxford: Association for Computers and the Humanities (ACH), May 1994.

[14] T. Berners-Lee and D. Connolly, "Hypertext markup language - 2.0," RFC 1866, Internet Engineering Task Force, Nov. 1995.

[15] N. Borenstein, "The text/enriched MIME content-type," RFC 1563, Internet Engineering Task Force, Jan. 1994.

[16] T. Berners-Lee, L. Masinter, and M. McCahill, "Uniform resource locators (URL)," RFC 1738, Internet Engineering Task Force, Dec. 1994.

[17] R. Fielding, "Relative uniform resource locators," RFC 1808, Internet Engineering Task Force, June 1995.

[18] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, "RTP: a transport protocol for real-time applications," RFC 1889, Internet Engineering Task Force, Jan. 1996.

[19] L. Zhang, S. Deering, D. Estrin, S. Shenker, and D. Zappala, "RSVP: a new resource ReSerVation protocol," *IEEE Network*, vol. 7, pp. 8–18, Sept. 1993.

[20] R. Braden, L. Zhang, and S. Berson, "Resource reservation protocol (RSVP) – version 1 functional specification," Internet Draft, Internet Engineering Task Force, Nov. 1995. Work in progress.

[21] E. Shapiro, "Virtual places – a foundation for human interaction," in *Proc. of the Second World Wide Web Conference'94*, (Chicago, Illinois), Oct. 1994.

[22] S. R. Ahuja and J. R. Ensor, "Call and connection management: making desktop conferencing systems a real service," *ACM Computer Communication Review*, vol. 22, pp. 10–11, Mar. 1992.

[23] T. Meyer-Boudnik, "MHEG: Ein Austauschformat für interaktive Multimedia–Präsentationen," Tech. Rep. TR-94-002, Praktische Informatik IV, Universität Mannheim, 1994.

[24] R. Price, "MHEG: An introduction to the future international standard for hypermedia object interchange," in *Proc. of ACM Multimedia 93*, (Anaheim, California), pp. 121–128, June 1993.

[25] H. Schulzrinne, J. Brassil, and A. Choudhury, "Internet access to IEEE journals: The September trial," in *Proc. of Documents in Public Networks – Applications and Experiences (Dokumente in öffentlichen Netzen – Anwendungen und Erfahrungen)*, (Darmstadt, Germany), Nov. 1995.

[26] M. Zisman, "Lotus Notes web strategy." Transcript of press briefing, Dec. 1995.