The Web is the Next Platform

5/27/95, bens (version 5)

Note: I've included a lot of material in this memo. If you don't have time to read it all, please be sure to read at least the first 4 chapters.

Table of Contents

1. Summary

The Web (as I will loosely refer to the Internet and it's evolving data formats and protocols) exists today as a collection of technologies that deliver some interesting solutions today, and will grow rapidly in the coming years into a full-fledged *platform* that will rival -- and even surpass -- Microsoft Windows.

Coincidentally, as I've been constructing my vision of the future of the Web, The Forrester Report, "Sizing The Internet", Volume Nine, Number 5, April 1995, published a vision of the "SuperWeb" that is very much along the lines I've been pursuing. The rest of this memo describes my vision of how we can create the *Microsoft SuperWeb*.

I have had a great deal of help from many folks at Microsoft in forming the opinions herein, so I give those people credit for any great ideas and deep insights that I may have written down. Any errors in facts or logic, on the other hand, are mine alone. Thanks to brianf, chrisjo, drosen, and johnlu for detailed comments on earlier drafts of this memo.

I will try to make and support the following key points in the rest of this memo:
1. **The Web** is an **application platform** (complete with APIs, data formats, and protocols) that threatens Windows -- many corporate developers and ISVs could develop and deliver their solutions more quickly, to a wider audience, with the Web than they can with Windows or MSN as it exists today.
2. If **Microsoft** is to influence the Web, we must have **broad, standards-based Web support** in our products -- we have to be *the product supplier of choice* for all key existing Web technologies -- clients, servers, and publishing tools, at a minimum.
3. Once we have **market and mind share** on the Web with our products, we can take a leadership role in **expanding and shaping the Web**.

1.1. Why is the Web a Threat to Windows?

The Web today is a rapidly maturing ***application delivery platform*** -- you can shop for and buy wine, play hangman, Rubik's cube, and chess, read and augment converstation threads, check up-to-date weather forecasts and stock prices, read the latest news headlines, get dealer costs for cars, look up 1-800 phone numbers, download movie trailers, music clips from major recording labels, look up zip codes based on addresses, get real-time photographs from San Fransisco and U. of Washington, and order food from restaurants in 8 different cities. ***And these are all done with a simple HTML 2.0 web viewer*** (like Netscape, Mosaic, or our Internet Explorer)!

You can also conduct a phone call with anyone anywhere else in the world ("internet phone", just CB radio quality now, but that will improve), read USENET news groups (NNTP), join a chat session (Internet Relay Chat), join a 3D chat session (Worlds Away), and hear streaming, low-quality audio (RealAudio -- ex-MS folks robg, philba, martind).

My nightmare scenario is that the Web grows into a rich application platform in an operating system-neutral way, and then a company like Siemens or Matsushita comes out with a $500 "WebMachine" that attaches to a TV. This WebMachine will let the customer do all the cool Internet stuff, plus manage home finances (all the storage is at the server side), and play games. When faced with the choice between a $500 box (RISC CPU, 4-8Mb RAM, no hard disk, ...) and a $2KPentium/P6 Windows machine, the 2/3rds of homes that don't have a PC may find the $500 machine pretty attractive!

The following attributes of the Web are paramount:
1. Server-side information and *interactive applications* are key (the viewer is just enabling technology)
2. Universal data formats and viewers enable the web to grow in richness and power -- the Web is a platform that no one controls and everyone can enhance.

Examples of some cool, interactive uses of the web today that may not have been imagined or intended by the creators of HTTP & HTML:
1. Virtual Vineyards (http://www.virtualvin.com) -- examine their list of wines, see graphic charts of wine flavors and tastes, add wines to a shopping basket, and order the selected wines. They put a "session id" in the "redirect" URL that you get when you first visit, and use this to track state (your shopping basket) for you on the server.
2. Play Rubik's cube (http://vadim.www.media.mit.edu/cube.htm) -- presents a 3x3x3 Rubik's cube graphic, complete with "mirrors" to see the hidden sides. You click on arrows on the corners to rotate sections, and it sends you the new rotation. Uses a sensitive map.
3. Send a web postcard (http://persona.www.media.mit.edu/Postcards) -- let's you pick a picture and send a message to someone's e-mail box. Recipient gets an e-mail message telling them to go to a special URL that has a page with the postcard -- the picture plus the message you composed.

4. Movie database and rating system (http://www.msstate.edu/Movies) -- this is an Internet user-maintained movie database, more movies than Cinemania, reviews by any random web surfer, credits, actors, actresses, producers, directors, etc. 45,926 titles, 4,569 plot summaries, 6,516 have been rated by surfers, 79,619 actors, 43,942 actresses, 10,289 directors. Shows use of HTML as a user interface to a database.

1.2. Why do we need to start from the Web today?

If we don't quickly become the supplier of choice for Internet technology, the Internet will grow and change under someone else's influence, and we risk losing the standard setting role (with the attendant profit margins) we have come to enjoy with MS-DOS and Windows (and Office).

There was a time when we thought that we could just "build it and they will come" with MSN, hence all the non-Internet technologies we developed (Marvel RPC, incompatible Mail & News protocols, MOSView, etc.) for MSN. These technology choices are unfortunate, for (in hindsight) I think it is clear that MSN would have been much further along now if we had started from the existing Web and enhanced it.

The battle for control of the online world is much bigger than anything Microsoft has ever faced, and we are coming to the battle a bit on the late side. With BASIC, we were almost alone. With MS-DOS, we had a few, small competitors. Even with Windows, there were perhaps a half-dozen competitors. In the online world, we have all of our favorite software companies (Novell and Lotus) and hardware companies (Sun, IBM), new companies (Netscape), phone companies (MCI, AT&T Interchange), internet service providers (PSI, Netcomm), banks, and who knows who else. Most of these companies recognize that their best bet for trying to attack Microsoft is through the Internet (since they have failed in the OS wars). The promise of gigantic revenue streams and healthy profit margins (not to mention the cachet) are drawing established companies big and small and energetic start-up companies into the battle.

Now, the UNIX model is one possible way to see the Internet evolving -- several companies start from a good idea, but go in their own proprietary directions and thus are unable to achieve the true leverage of an open, common standard that enables a large market of products. It is possible that if Microsoft forges ahead with it's current MSN plan (BlackBird, OLE everwhere, COM/DCOM, etc.), and only pays the Internet lip service, we may "pull a Windows" and end up dominating the online world. All of these other players will spend all of their time bickering about IETF standards and shipping incompatible extensions, and the Internet will end up a mish-mash of incompatible solutions.

On the other hand, it is also possible that some company will "pull a Windows" by taking a leadership position of enhancing the Web -- this is certainly the strategy that Netscape is pursuing! We have to assume that at least some of our competitors have figured out how Windows won, and are trying to recreate that strategy on the Web.

When I reflect on some of our previous "big bang" efforts -- OS/2 and LanMan -- the key mistake we made was not to focus on compatibility enough. With OS/2 (where I spent my first 5.5 years at Microsoft, working primarily on MS-DOS compatibility), we didn't support **all** MS-DOS applications, and we didn't support **any** MS-DOS device drivers, and we didn't even multi-task MS-DOS applications until IBM shipped OS/2 2.0. Regardless of all the cool, sexy features in OS/2 (multi-tasking, better graphics API, memory protection), it was not a no brainer upgrade from MS-DOS -- customers had to **give something up** in order to switch to OS/2: their existing software! Only with Windows 95 (where we have focused on compatibility to an amazing extent) are we finally going to enable to move customers away from MS-DOS.

With LanManager, the compatibility point was Novell Netware. We told customers they had to toss their existing Novell networks in order to run LanMan **and** they would have to accept slower performance from LanMan file servers vs. their existing Netware servers. So, not only did LanMan have the OS/2 albatross hanging around it's neck, it also was not a no-brainer upgrade from Novell. With Windows NT and Windows 95 embracing Netware, we're finally starting to gain some ground here.

1.3. How should we extend the Web?

At a high-level, this is very clear. We should support all of the key internet standards and become key suppliers of Internet technology to all comers. In parallel, we should be extending the web with as many Microsoft technologies as possible, even if we have to modify those technologies in ways not original intended by their designers. If we look at the reasons for our success with Windows, certainly one important aspect was the quality of our development tools and the support we give our ISVs.

I think a key focus for us as a company is to become the place to go for Internet tools -- servers, clients, publishing management systems, editing tools (Word, graphics, etc.), system management tools, billing, search tools, etc.

2. Goals & Vision

Goals

1. Microsoft needs to ensure that we ride the success of the Web, instead of getting drowned by it.
2. Microsoft needs a consistent plan to enable ourselves and our customers to *author*, *manage*, and *distribute* large volumes of ***information*** quickly and easily on CD-ROM, LAN, and the Internet.
3. Microsoft needs a consistent plan to enable ourselves and our customers to *author*, *manage*, and *distribute* highly interactive, multimedia ***applications*** on CD-ROM, LAN, and the Internet.
4. Microsoft needs a consistent plan to enable ***billing*** for the online world, so that our STT becomes pervasive.
5. Microsoft needs to ensure that our **technology platforms** are appropriate to capture the **two-thirds of US homes** that don't have PCs today.

Observations

1. If Microsoft doesn't enhance the Web, there is a nightmare scenario where an OS-neutral Web platform arises, and then a company like Matsushita or Siemens could come out with a $500 "Web Box" that runs web applications (with no need for Windows, or MS-DOS compatibility, or Intel compatibility), and consumers make the obvious choice between a $2000 Windows PC and the $500 Web Box. Say good-bye to Windows.
2. Only one-third of US homes have PCs today -- content creation tools like Office are not going to help us capture the remaining two-thirds; Entertainment and Information content and services are.
3. Popular web sites 3 years from now look like CD-ROM titles -- multimedia, highly interactive, engaging -- they will not filled with lots of static (Word, WP, etc.) documents. This will happen with or without Microsoft involvement.
4. These sites will be hugely popular, rivaling traditional use of the Windows platform. We either embrace this usage or get left behind.
5. We currently have at least 4 different online publishing efforts going in almost as many different directions. We should focus our efforts on the existing web platform and build up from there.
6. The impact of Web-enablement will be pervasive throughout Microsoft. This is at least as fundamental a change as the current wave of Win32/OLE2 enablement has been.

Evolution of Existing Businesses

1. Make NT the best Web server platform
2. Make Windows the best Web client platform
3. Make Office & Developer Tools the best Web authoring tool

3. Key Issues

I've tried to call out here only the most controversial/complicated issues:

1) **What do we do with the current BlackBird client/server, Forms cubed work?**
   ==> There may be a time in the future where this approach is more compelling than a web-based approach, but for now I think BlackBird should focus on publishing for standard web data formats and protocols, and lay the other stuff aside. If we have the resource to continue pursuing the original BlackBird viewer/server design without impacting our web publishing efforts, that's fine.

2) **What is our object model? docObject/OCX, or some subset or alternate?**
==> paulma has asked chrisz and me to go sort this out (by writing comparative code). My goal is to have this resolved no later than 12/95. I'm hoping that some subset of docObject/OCX with some subset of the current OLE run-time penalty will be the outcome, and everyone will be happy.
3) **What is our overall electronic commerce strategy?**
==> STT is a great solution for electronic *credit cards*, but we have a fair bit of infrastructure to implement. We also have the MSN billing system. And what about eCash (FirstVirtual, CyberCash, etc.)? Do we need a "low-end" solution in addition to STT?
==> One idea is to see if we can extend the MSN billing system out onto the Internet in a secure manner, so that high-volume, low price services can be used by MSN customers. Do we need to get into the eCash business
4) **What is our future in Content -- Creation, Publishing, or Both?**
==> Consumer today is primarily a publishing effort. Will this continue, or will we branch out into content creation? Will we be R. R. Donnelly, or Disney, or both?
5) **How can we coordinate and leverage our various authoring/publishing efforts?**
==> We should view BlackBird, Word Assistant/Office, Media View and MOS View, Multimedia Tools, VRML tools, and 4D authoring as all solving parts of a bigger problem, and we should coordinate/coalesce efforts as appropriate.
6) **Where do we draw the line on the editing abilities between OS and Office?**
==> O'Hare will enhance HTML as much as possible to make it a great *application* platform, and the HTML Control will at least have to be able to edit all the HTML it supports. However, O'Hare will not focus on the tremendous ease-of-use (spell checking, autocorrect, Intellisense, wizards, etc.) that make Word such a great authoring tool.
7) **What do we need to do about SMTP and NNTP clients and servers?**
==> Do we need to have a small SMTP and NNTP server (targeted at small and medium sized businesses?) Do we just allow others to do this? Do we switch MSN from private mail and news protocols to SMTP/POP and NNTP?
8) **How open are our Client and Server offerings, and what distribution channels do we use?**
==> Should we provide the base functionality with a good tool set and let others write the specific end user apps and server apps? Or should we try to provide complete solutions? Examples: would we want others (e.g. Netscape) to take our components, add a lot of value and then distribute to customers under their brand?
9) **How do we balance our MSN service business with our (NT) WWW server software business?**
==> Do we want to offer a Microsoft-branded service that allows millions of consumers access to the WWW? If so, what do we need to have a market leading offer? How do we do end-to-end testing of this offer? Who determines pricing of the components? (e.g. do we give away the WWW explorer? Should we price the NT WWW stuff to gain maximum market share, even if it mean allow others fostering a consumer offering to have an equivalent/better price structure than our own?)

3.1. A Step too Far: BlackBird?

BlackBird has many aspects that are similar to the SuperWeb I sketch out here, but there are several fundamental problems:

1. **Building Client, Server, Protocols, and (some) Data Formats from the ground up takes a long time.**
We followed this approach with OS/2 and LanManager in many ways, providing an inadequate bridge from the past (MS-DOS, Novell) to our future products. MSN is also guilty of this, using non-standard mail and news protocols (instead of SMTP, NNTP) and "2D" protocols MPC and MOSView (instead of HTTP and HTML and TCP/IP). In contrast, Windows embraced MS-DOS and slowly helped customers transition to Win apps while still running most of their MS-DOS drivers, TSRs, and apps. Similarly, O'Hare tries to take an evolutionary, compatible approach to the web.
2. **Forcing information providers to choose a proprietary data format may cause them to balk**
I met with the KB folks in PSS & ITG today (5/9/95), and they currently publish the PSS KnowledgeBase on: The Web, CIS, AOL, Genie, MSDN, TechNet, and internally to PSS technicians. Most of these formats and viewing tools are different, so there is a lot of inertia here that prevents them from really improving the quality of their presentation and query abilities.

The web is growing at a very fast rate, and if we force information providers to make a choice between creating great web content or great MSN content, they may choose the former. The only strength MSN has now is billing, but that will evaporate in the next year or two with STT and other technologies for eCommerce.

3. **Using OLE as a Fundamental Framework adds Complexity/Size, reduces Speed**
   "If OLE were not complicated, fat, and slow, it would be wonderful!" -- bens, 5/9/95. I love the idea of supporting OCXs in HTML and docObjects in my viewer frame window, but **only** as a way to leverage the existing and forthcoming body of controls that we and our customers and ISVs are developing. To burden the common case (no custom controls in an HTML page) with the overhead of OLE just reduces the quality of the viewer (bigger, slower). Standard Windows controls let me do everything I need to do more simply, with less RAM, and faster. My current perspective on OLE (5/27/95) is that it can be fast if you just use COM and build just the supporting libraries you for yourself (as Forms cubed folks have done). For the Universal Viewer, the default model would not support cross-process scenarios nor docfiles, for example (loading OLE to accomplish these things only when a docObject like Word was loaded).
4. **Using a Custom, RPC-based Protocol**
   Many people say that RPC is fast, but when I look at groups that have used RPC (MSN, Mail), I find that the overall use of RPC has made the resulting solution slow. I wouldn't doubt it if this were just to poor coding (MS-Mail used to make 13 RPC calls to send a message, the number is down to 1 now, I believe), but the mere fact that RPC hides from you the network call makes it hard for a programmer to be aware of the problem (I liken this to the similar problem of object oriented programming in C++ -- it's hard to tell what is expensive and what isn't by looking at the source code). So, I'm not opposed to RPC in principal, but it's "network call hiding" properties have to be carefully monitored during development.
5. **Assuming that Fixed 2D Layout is the Primary Design Paradigm adds Complexity/Size**
   If you've tried viewing a FAX on a 640x480 screen, or an Adobe Acrobat "PDF" file, I think you'll agree that having to scroll horizontally and vertically to see the content is obnoxious. Fixed 2D layout (i.e., Forms cubed/VB/MS-Draw/etc.) is harder to author than the "1D" layout that HTML provides, and it is harder for the customer to view. A minimum Forms cubed form is 300 bytes (last time I talked to johnshew) -- A minimum HTML document is 1 bytes (the letter A, for example).

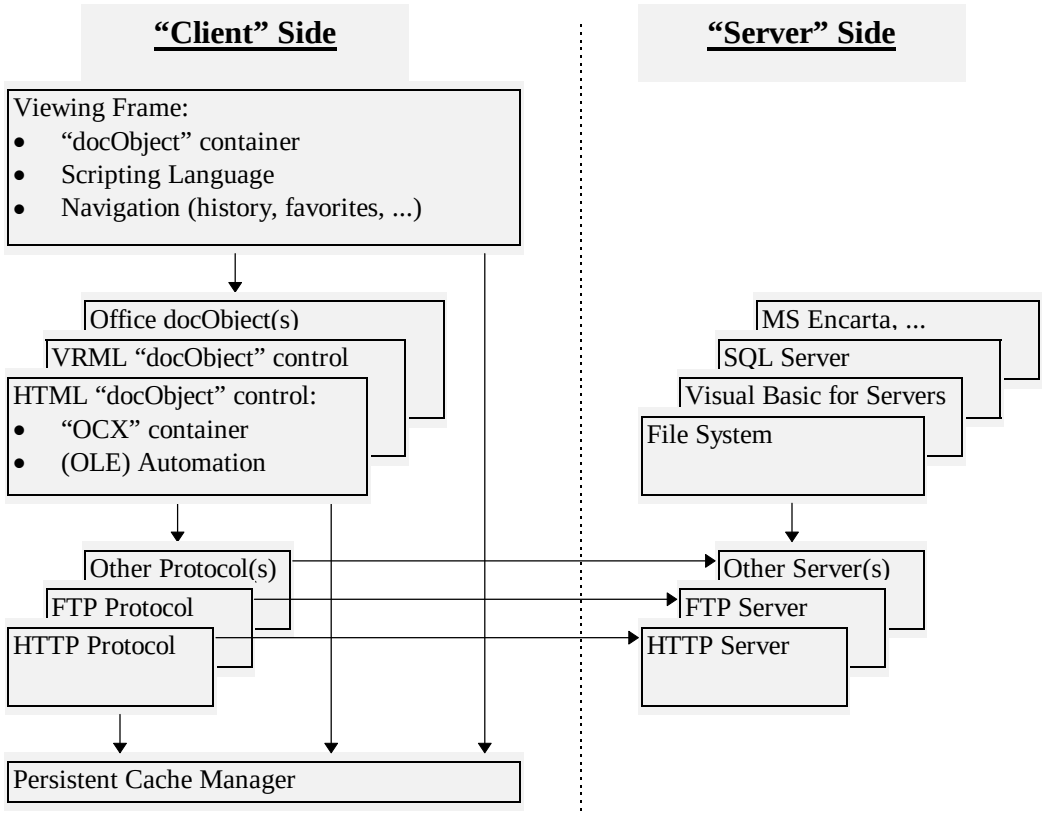
A more gradual approach of starting from existing technology and enhancing it (as I'm suggesting in this memo) has proven in the past (with Windows) to be more successful than a Big Bang approach (as with OS/2, Lan Manager, Exchange?), at least in the "platform" business. Certainly we did efforts like Excel and WinWord (the latter which took 5 years, I believe) that we're Big Bang (especially Excel), but there the compatibility requirements were lighter, and these are not really platforms to the extent that Windows and the Web are.

4. Proposal: Microsoft "SuperWeb" Architecture

I think of the web as "OLE without the E" -- MIME (Multimedia Internet Mail Exchange) types identify the objects (HTML, GIF, JPEG, AU, etc.), and URLs (Uniform Resource Locators) are the links. Embedding is achieved not by storing one object inside another, but rather visually -- an HTML document contains an <IMG SRC=url> tag, and the HTML viewer fetches the image (typically a GIF file) and "embeds" the image in the rendered HTML. Instead of the complex, general purpose, flexible "C++" interfaces of OLE, the web has very simple binding (for MIME type x, run viewer program y) of objects to code, and has some fixed support for visual embedding (the HTML viewer).

This "OL" approach is highly suited to low-bandwidth client/server connections -- no embedding (i.e., docfiles) means that a view is constructed in a composite fashion, linked together by URLs, so the client can cache the heck out of these individual data objects. And, while it may seem that ASCII text HTML files are inefficient, the streaming nature of access, coupled with progressive rendering (draw immediately as data is received in most cases, for both text and graphics), and the fact that typically HTML files are *smaller* than equivalent Word doc files, means that the Web is quite zippy.

The following picture illustrates the architecture of the Microsoft web platform as I see it in 1996:



With this model, we're saying that the *default* application model is "dumb" client and "smart" server, returning to the mainframe model of the past -- except that the client is *much more capable* than a 3270 terminal. You could also think of this as X-Windows or Display Postscript, except that the language is higher-level than either of these, and the client has a lot of user-interaction code smarts.

Recognizing that this is not sufficiently powerful and extensible for all applications, we augment this with the ability to use custom controls (native code) and scripting. This provides a very powerful platform: you still probably cannot write Excel 7.0 with performance comparable to native code, but you can construct very rich client/server applications.

Benefits of this Architecture

1. Applications can be developed for both local and LAN and WAN deployment, using the same data formats and engine code -- the universal viewer is the same in all cases, and the server code can either be a Win32 process on a local machine (for consumer applications, MSDN, etc.) or on a separate server machine over the LAN or WAN.
2. Client and server versions are decoupled -- client and server can be improved independently, speeding progress of both.
3. Management of client machines is simpler -- don't have to be constantly installing new applications, whether in a corporate setting or on the web.
4. Reduces resource requirements on client machines -- a universal viewer saves disk space, and since it is universal we can tune the heck out of it to be small and fast
5. Develop applications is faster than VB -- since most common user interaction is handled by the universal client, application developer is writing simple UI descriptions (assuming no or minimal need for scripting), and focusing primarily on core engine code for the application. UI consistency is easier to achieve because there is less code to write for UI. UI also looks better, because the layout of text, graphics, and controls is automatic (no more nudging controls in a VB form!), and richer (background bitmaps, sounds, etc.)

6. Client can choose to render/degrade information, but still supply core application features -- Rashid wants to use HTML + Video support for his ITV low-end set top box. Could use HTML for "winpad" companion devices.

5. Details on Microsoft "SuperWeb" Architecture

5.1. Universal Viewer Frame

This "frame window" has the title bar, menus, and toolbars for "docObject" HTML, VRML, WinWord, etc. viewing controls. It maintains the "session" history of visited URLs, provides access to Favorite Places, and provides common controls for navigating (back, forward, open) and viewing (zoom, maybe different viewing layouts). The Windows UI elements (title bar, menus, toolbar, status bar, etc.) may be hidden (by customer and/or by viewer control?) to provide a "kiosk" mode.

5.2. Scripting for HTML and VRML

The difficult part about scripting is not choosing a language (though there are obvious religious and practical issues here that may be at odds with one another), but it defining a safe run-time environment that is powerful but does not permit virus/Trojan horse behavior (formatting your C: drive, or sending your Visa card number or your Quicken file or your password list file to an anonymous remailer).

In addition to permitting a flexible platform for running web applications (HTML UI, scripting, and/or native code controls), scripting can also be used to for simple consumer automation (enhance the Plus! System Agent, for example).

ThomasRe/BenS met with Jonathan Tisdale (Microsoft Institute in Australia, he's 2 years away from a Ph.D. on secure scripting) on 5/8/95 and basically sketched this model:

1. Scripting "virtual machine" -- using the common scripting engine, we provide a set of inputs and outputs that the script language may operate on in the "HTML World" and "VRML World".
2. Give each script a private space (protected from other scripts) as well as a global script space. These spaces would consist of persistent state (in the file system and registry, but access is restricted to a special subtree in each case, so the script cannot read or write any system/user data, so scripts cannot search for credit card numbers or userid/passwords, or modify any system settings, programs, user data files, etc.)
3. Dynamic state would also exist both per-script and global, and would include UI state, a variable store, and "pipes" to permit inter-script communication.
4. To support "unsafe" scripting, we would use digital signatures. A digitally signed script would be permitted to load a digitally signed DLL, and only Microsoft would be permitted to sign such scripts and DLLs (at least for now).

The obvious choices of languages include: VBA, Java (Sun's HTML client scripting language), TCL, PERL, and REXX. VBA is clearly preferred, but we need to evaluate it's size and performance (IEXPLORE.EXE is <600K right now), and if another language (like Java) has immense momentum, we would have to consider going with that language. Victor Stone (in ACT) is building (has built?) a lightweight VBA subset, so that is obviously **very** attractive. Netscape has announced (5/23) they are incorporating Java into the next release of their browser.

Of course, if we move toward the web as the shell for Memphis, then we automatically get shell scripting! Now, you might think that adding scripting to the HTML control gives us the equivalent of VB, and then ask "aren't we giving VB away in Windows"? Well, the run-time environment will not be quite as flexible (since we're preventing virus/trojan horse behavior), so it will be somewhat limited versus VB (cannot manipulate any random local file, for example).

5.3. HTML Control

The key focus here is to extend HTML to make it the premier "1.5D" multimedia publishing format (we will have limited 2D layout control, but not as much as Forms cubed, for example). Here is a list of features (by no means exhaustive) we're thinking about for future versions of HTML:

1. Embed custom controls (think OCX and also Windows controls) -- this permits pretty arbitrary extensibility of the browser. Control can then talk to system or server in any appropriate manner.
2. Scripting "virtual machine" -- using the common scripting engine, we provide a set of inputs and outputs that the script language may operate on in the "HTML World". We would permit the script a private space (protected from other scripts) as well as a global script space. These spaces would consist of persistent state (in the file system and registry, but access is restricted to a special subtree in each case, so the script cannot read or write any system/user data, so scripts cannot search for credit card numbers or userid/passwords, or modify any system settings, programs, user data files, etc.) Dynamic state would also exist both per-script and global, and would include UI state, a variable store, and "pipes" to permit inter-script communication.
3. HTML 3.0 tables (Netscape 1.1 has these now) -- variable number of rows and columns, controllable border sizes and styles, each cell can contain any HTML tags (including another table -- you can nest tables), cell height and width can be unspecified, fixed size, or percentage of viewing window height and width. These permit control similar to the Forms cubed fixed layout, but also resize nicely for different screen/window sizes.
4. Enhanced tables -- support simple math functions (ala Word), support sorting ascending/descending by clicking on a column header, resizable column width, drag & drop to reorder columns, outlining -- collapse/expand.
5. Fixed top/bottom/side regions -- most MediaView/MOSView titles (Encarta, Bookshelf), etc. have a non-scrollable region of the screen with information or controls. We add <headbar>, <sidebar>, <footbar> tags to support this. Down-level viewers still get this content, and <headbar>/<footbar> sections are just displayed as they are encountered.
6. Win95 USER controls -- just add simple <input> tags to call up all the new controls -- spin buttons, progress bars, etc.
7. Virtual List Box -- many MV titles have an edit control and a list box that are tied together: you type letters into the edit control and the list box scrolls to that range of entries that have the typed in prefix. This control would speak some protocol (HTTP or something new) to a server that has the contents, and an HTTP GET (for example) would be issued on every change to the edit control to update the contents of the list box.
8. Sound effects -- a <sound src=... loop> tag would specify "mood music" to be played when a particular "page" is loaded; we could also add sound effects to control actions (pressing buttons, anchors, etc.) on a per-control basis.
9. Transition effects -- borrowing an idea from PowerPoint slide transitions, we would add a TRANS= attribute to the anchor (link) tag, so that when we go to the new URL, we perform a dissolve, fade-in, wipe, etc. visual effect.
10. Read-ahead -- have PROBABILITY=0.0...1.0 attribute in anchor tag, viewer will read-ahead on highest probability anchor. These probabilities can be computed by dynamically by the server based on actual usage patterns (and even can be customized to the individual client user?) -- Patent application will be filed here. Help's make the TRANS tag more useful on a slow link!
11. Pop-up windows -- similar to WinHelp's pop-up text boxes for short descriptions, we would add a POPUP attribute to the anchor tag, and render the fetched contents (typically HTML, but could be sound, graphic, etc.) in a pop-up "window" that goes away when you click outside of that window. Could also have a <popup> tag that has the HTML inline and doesn't show it until the text/image is clicked (but wouldn't work as nicely for down-level viewers).
12. Overlay text/graphics -- make nice looking images quickly by downloading image once and drawing text on top of it (using our new <font face=...> tag we have for v1.0).
13. <include src=> tag -- lets you composite a "single" HTML page from multiple HTML sources; gives better client performance due to caching, and simplifies server data management (perhaps); a variation on this theme also lets one HTML page contain another HTML page visually embedded in it (in a rectangle, presumably).
14. Custom tags -- Using the scripting language, permit the definition of custom tags and their behaviors; <define tag=foo src=script URL>; <foo> ... </foo> causes all text between these two custom tags to be run through the specified script for foo. This is another way to optimize for low-bandwidth connections.

15. Add TAZZ text-to-speech tag -- low-bandwidth way to get voice; also, as viewer option, permits visually impaired folks to view web content and use any web application; if shell is a web application, then we're automatically enabled to help visually impaired folks.
16. For ITV (Rashid) and IHPC (GabeN), add "video" support -- need to be able to specify a video source as either the desktop background or the client area background of the HTML viewer, and also need to be able to have another video source in a rectangle contained in the HTML in the client area (for preview/channel selection). Normal HTML is used to display video programming (schedule) information, adjust volume, color, channel, recording, etc. Special video card does MPEG decoding, compositing of video signal and HTML "VGA out".
17. Define a compressed HTML format (HTC?) -- use MSZIP/GZIP/PKZIP compression to increase effective bandwidth of low-speed links: this will yield better results than any modem compression scheme.
18. Support wavelet-compressed images.

5.4. Custom Controls for HTML

To demonstrate the types of controls people might right, consider a "virtual list box with edit control" like those found in Bookshelf, Encarta, and Exchange. There is an edit control that you can type text into, and the list box automatically "scrolls" it's content (sorted) to show the position in the virtual list whose prefix matches the typed in text. This is implemented by having the edit/list box control ask the server (via HTTP?) for new list box content every time a change is made to the edit control text.

5.5. Other Protocols

HTTP is very general purpose, but has a simple GET/PUT model (i.e., no LSEEK). We might consider adding another protocol, not as complicated as RPC, that supports higher-frequency client/server interaction. Do this only if we cannot easily extend HTTP to get similar performance.
MediaView will also add a protocol to support their "generic MV viewer" application.

5.6. Low-End HTML Editing

HTML editing in the HTML Control is to Word as VB form design is to Corel Draw (or some other high-end grahics package) -- it enables construction of simple highly interactive, multi-media web applications, and (as a side-effect) passable HTML static documents, but it doesn't make it easy to construct awesome, Word-quality documents.

We're not going to compete with Word or PowerPoint or Publisher, but by adding simple editing to the viewer (perhaps with an add-on DLL?), we ensure that as we add HTML rendering features customers can immediately author these features. This would be a modal thing (like Edit.Edit Mode), since typically customers won't be editing HTML. Since HTML is just an object stream, all we have to support is the ability to insert, delete, move, and edit these objects. We don't have the complicated 2D fixed layout commands that VB, Forms cubed, etc. have (align objects, grids, etc.), and the properties for our objects are pretty simple. You can imagine editing mode looking similar to VB, having an object tool palette and a property sheet for the selected object.

5.7. VRML Control

Virtual Reality Markup Language is an evolving standard that you can think of as the 3D analog of HTML. A VRML viewer fetches a VRML file which contains a "scene description" 3D objects, texture maps, light and sound sources. The scene would typically be composited by the viewer from multiple URLs -- the VRML would just list URLs for the objects in the scene -- as this permits caching and reuse of objects. Over time, scripts (similar to the HTML scripting environment) could be attached to the objects so that they could have autonomous behavior (like a sprite, or a water fountain, or a machine-gun wielding zombie), and their could also be remote control of these 3D objects to support multi-user games, interaction (as in mannyv's Virtual MUD/MOO project -- this control should be the basis of his work, and, indeed, perhaps he should write it).

We're (thomasre/johnshew) close to licensing a basic VRML browser (InterVista, Community Company) that runs on top of Reality Lab.

5.8. Persistent Cache Manager

Internet Explorer today caches everything that is downloaded (HTML, GIF, JPEG, AU, etc.). Over time, we'll probably combine this with the Shadowing work that is going on for Nashville (wassefh). As this is hooked in at the Protocol level, we'll document the interfaces for authors of new protocols. The Cache manager takes care of flushing heuristics and user-settable UI for how much disk space to devote to the cache.

5.9. Servers

As described in the architectural picture, there are many types of servers running on a web server:

1. File system (FAT, NTFS, OFS)
2. SQL server
3. Visual Basic for Servers
4. MediaView server -- (I've discussesd this with KaviS) leverage MediaView's MVB/HLP title storage to contain HTML, sound, graphics, etc. data, along with an index. In the near term, MediaView titles will be native code linked with the media view library, using the HTML Control to view the HTML content on the client. Over time, we may migrate to the HTML + "embedded" control model, assuming that it will be easier to author such titles and the titles and their UI will be better and more flexible.

5.10. Publishing (document management, workflow, editing)

This is a really big problem (at least a lot of work, if not a hard problem). BlackBird, MSDN, RSegal, and others are all trying to solve this class of problem. We really need what is akin to a newspaper or magazine back office publication system.

Key issues:

1. Manage database of data (stock images, photos, sound, 3D objects) to reduce redundancy (also improves client performance because it can reuse cached data).
2. Handle publishing/versioning -- trickiest part is supporting testing content with links locally, and then publishing content with real links, and doing the publish in a coordinated fashion to the live web server without having to take it down or causing momentary bad links.
3. Feedback system to permit customers to submit "bug" reports about problems with pages, and have these reports flow back to editors to review and act upon.
4. Workflow system -- support compose, copy edit, senior edit, layout review workflow, multiple reviewers, deadlines (date/time, or "issue" focused), reporting on status, etc.
5. Automatic "back issue" archiving, searching -- lots of sites have the current issue and then an old issue archive -- you can search or browse this material.

There is probably a lot of overlap here with the Repository work in davidv's group, whomever is doing workflow stuff at MS, the Office group, etc.

6. Recommendations

I've arranged these according to the groups that are (or should be) working on products either enhance the Web or use the Web to deliver content. This is how we create the **Microsoft SuperWeb**.

DISCLAIMERS:

1. My knowledge of the current activities and future plans of these various groups is in some cases very spotty. Nevertheless, in broad strokes I believe that the following priorities are generally correct. If I have mischaracterized your group in any way, I apologize profusely in advance!
2. I am not a fan of OLE. This is not a judgment based on the people who designed and implemented OLE, or the people inside and outside Microsoft who are using OLE -- I salute them for their ability to wield this amazing technology! Rather, it is based merely upon my observation that the complexity, size, and speed costs of OLE seem to outweigh the benefits in most cases (and certainly in the world of the Web). For convenience, I have used OLE terms in some cases where I actually think that a more lightweight solution may be sufficient (Windows control instead of OCX, for example). However, I remain open-minded about OLE, and hope that we can achieve a balanced approach where we use an OLE subset to achieve my goals. Full OLE would be

loaded only when a full-blown OLE object was present (as Internet Explorer does today when it needs to drag and drop).

3. Many statements and opinions below are presented in black and white, when really there are many shades of gray. My purpose here is to try to bring the issues and solutions into clear focus, so I if my statements appear to strident, please try to read through them to see the underlying intent.

6.1. [New] Internet Marketing Team

As drosen suggests, we need a single team to own and drive the Internet strategy:

1. Set priorities
2. Communicate strategy to internal Microsoft folks and external customers and partners and ISVs
3. Coordinate packaging/products/pricing
4. Manage business relationships related to the Internet

6.2. NT

A major push for NT is to get caught up with Netscape and others in many areas, and also add value places where others are behind:

1. Scalability and Reliability enhancements to ensure NT owns the server farm business -- clustering, replication, fault tolerance, monitoring/maintenance, mirroring, etc. MOS has a challenge in being able to roll out a network to support the millions of customers they will have on MSN, and Microsoft doesn't have basic technology (like DNS server) that they could modify to meet their needs. The MS folks who manage www.microsoft.com cannot use the standard NT replication service because it doesn't deal with open files.
2. Visual Basic for Servers -- to enable rapid development of rich web applications, we need to provide a Visual Basic-like development environment for NT, except that instead of creating VB forms, the developer will author web page (templates) and/or code to construct HTML on the fly. Integration with the publishing environment is critical, so perhaps this instead belongs in the BlackBird team, which would use the BGI hooks that NT is supplying in their Internet Server.
3. Query Services -- just as Netscape & Verity announced recently (5/9/95, see "Verity and Netscape Team up to Bring Topic Agent Technology to the Internet; Netscape to Provide Popular Topic Search Engine for Netscape Servers" on page 15), provide core full-text/keyword search services: right now, OFS is it's own little world -- it doesn't scale well (how can you index the Internet?). OFS should be split up so that the storage, indexing, and querying of data can be performed each on separate machines -- just like the Lycos model today (web servers store the original data, a battery of web crawling machines fetch documents and produce abstracts, and then a battery of query servers respond to queries); Lycos has (as of 5/9/95) **3.85 million** URLs cataloged, by virtue of fetching **566K** web pages. The uncompressed abstracts take up about 1.4 gigabytes (you still need the inverted indices to make searching rapid).
   I met with the MS Research NLP/Decision Theory folks on 5/12, and they have some great technology for us to harness (in concert with OFS). JohnMs is doing an application of "AutoClass" on the Lycos abstracts, so we'll be able to add a "Find Similar Pages" button to Internet Explorer -- it will return an HTML page with a list of links to pages that are "similar" to the page that was being viewed.
4. Merchant Server -- do all the STT stuff for electronic Visa (eCard) transactions.
5. Add performance enhancements to work faster with IExplore (http keep-alive, read-ahead hints)
6. Logging and Profiling hooks (let's web site managers know what areas are popular, enables setting advertising rates, server tuning, etc.)

6.3. O'Hare

My team owns the universal client (at least for HTML, mannyv may own VRML). Our Internet Explorer is in beta now with Plus!, and we RTM 7/14. We have lots to do after 7/14 -- I hope to get this all done by 7/1/96 (with interim releases in Sep 95 and Feb 96):

1. Break Internet Explorer into: Frame window, HTML "docObject" control, Protocols, Caching
2. Support (light-weight) OCX "embedding" in HTML
3. Support scripting for HTML (and VRML)
4. Provide HTML OCX to VB, Access, etc.
5. Appropriate HTML 3.0 tags (especially tables)

6. Add multimedia enhancements to HTML (background sounds, sprites, panes, etc.)
7. Low-End HTML Editing
8. Extend secure code model (introduced in IExplore v1)
9. Secure Sockets Layer (SSL, HTTPS)

6.4. Windows Shell

We have the idea that the Windows Shell should move toward a "page and link" model of user interaction, so the Windows Shell team (joeb, kurte, et. al.) need to aggressively pursue figuring out both:

1. Does a page and link model really make the shell easier, cooler, more powerful?
2. How do we transition customers from the Win95 shell to this web shell?
3. Can we do something useful with VRML for Win97?
4. Tight integration of Internet Explorer and the Shell (shell would provide a "shell viewer control" that fits into the universal viewer frame).

6.5. BlackBird and MSN

Trying to do client, protocol, server, and publishing work is a big, big effort. The BB/MSN team should:

1. Refocus on HTML/HTTP as a foundation
2. Leave the client to the O'Hare team
3. Solve the high-end publishing problem (workflow, document management, versioning, ...)
4. Focus on HTTP server extensions in support of solving the publishing problem.
5. Leave the query services to the NT team
6. *Fun Director* -- MS Research has "AutoClass" technology that groups together customer preferences, documents, etc. into "equivalence classes", permitting MS to recommend movies, books, audios, web pages, and products that the customer has a high probability of liking (based on the systems' knowledge of the customer's preferences and the preferences of other people that match up with the customer). This has a tremendous potential for revenue to MS by replacing direct marketing efforts by consumer companies the world over!

6.6. Office

Our Office applications (especially Word and PowerPoint) remain our premier editing applications. Office needs to:

1. Make all Office apps great docObjects (for use in Viewer Frame)
2. Continue to improve Internet Assistant's "save as HTML" feature
3. Provide great integration with the HTML Publishing tool
4. Add "save as HTML" to PowerPoint
5. Create DLLs that convert on-the-fly between native office data formats (DOC, XLS, PPT, etc.) and HTML (so non-Windows/Mac platforms can view native Office documents. Web server would cache translated versions to improve performance.
6. Stop trying to be a web browser -- there is no way Word can compete in size or speed or multimedia functions with a focused web browser like Internet Explorer or Netscape. Instead, with Word as a docObject, and Internet Explorer supporting these in its client area, there is no need for Word to do HTML viewing.

6.7. WinHelp

WinHelp is an ideal candidate for turning into a web application. The WinHelp team should:

1. Scale back to a minimum follow through on current commitments
2. Strip out the Windows UI code, producing an HTML web app server that has the code to read and search *.HLP WinHelp files and produce HTML as output -- HTML is the UI. If it makes sense, we may also update the *.HLP format to store HTML natively for the content.
3. This "WinHelp HTML Server" must be capable of running over a LAN/WAN (what protocol should it use to talk to the HTML viewer -- HTTP?), as well as being a local Win32 process.

6.8. MediaView/MOSView

Similar to WinHelp, this team should recast their current model (write Windows client code that links to Viewer libraries, talks to MV "server") as follows:

1. Use standard protocol (HTTP?) to communicate between HTML viewer and MV HTML server
2. Ship HTML back and forth for UI and results
3. Run MV HTML server both on local machine as well as across LAN/WAN

NOTE: MV folks (and Rick Segal) have commitments to both MS-internal groups (Consumer) and customers to continue delivering and enhancing MV. Depending upon headcount and focus issues, this work may need to be done in the O'Hare and/or BlackBird groups.

6.9. Multi-Media Tools (Rick Segal)

Rick has commitments for MV that he has to meet, and he also wants to do HTML publishing/authoring. His team should:

1. Meet existing commitments
2. Try to minimize additional investments in MV
3. Move over to help BlackBird with publishing systems

6.10. 4D Authoring (movie making)

These folks are doing a publishing system (possibly fancier than some of the 2D systems discussed elsewhere). We should try to leverage their work, or they should try to leverage ours.

6.11. Repository

We should try to get this team to focus on the needs of the publishing system folks (BlackBird), so that we don't reinvent the wheel.

6.12. MSDN

MSDN currently has it's own MV client code and a private publishing system that they are actively working on to improve. MSDN should:

1. Donate their "publishing system" folks to BlackBird
2. Switch to the MV HTML Server platform
3. Move their content into SGML/HTML

6.13. KnowledgeBase

Similar to the MSDN group, they should get their content into SGML/HTML, and leverage these common tools being built in other groups.

7. Business Opportunities on the Web

I've tried to identify all the money-making areas on the Web (that Microsoft might be interested in), and indicate their relative priority to us and which groups should (probably) own each area:

1. Higher priority areas are ones we want to attack immediately and gain large market share.
2. Lower priority areas we might make tactical investments in to "prime the pump", but we would probably not seek to gain large market share.

Naturally, the more software-intensive areas are highest priority.

The "Total Market $s" column is my random estimate of the relative revenue opportunities of these market, not how much money Microsoft should make -- we'll focus on the higher margin markets.

Area	Priority	Total Market $s	Group	Comments
Client Software	1	0 (or small)	PSD	The client just helps us sell Windows; we

				need to make it ubiquitous so that we can control the evolution of the Web.
Server Software	1	$$	BSD	Scalability and Reliability are key features here (see Server Farms), but also a Query Engine, Back Office, natural language processing, and integration with publishing tools.
Publishing Tools	1	$$$	MOS & Office	Like a newspaper/magazine publishing system: manage workflow, media (text, graphics, sound, etc.) management, versioning, indexing, extensible to support individual media editors.
eCommerce	1	$$$$$...	AT & BSD	Getting STT established ASAP is our only chance at this lucrative revenue stream.
Content	2	$$$$$	MOS & Consumer	As a company, we have to decide how much publishing vs. creation we do.
Server Farms	2	$$$	MOS	A phone/cable company business -- just leasing disk space, CPU cycles, and bandwidth to customers to put up their content. Margins are probably not high.
Plumbing	3	$$$$$	MOS	A phone/cable company business -- plumbing is the routers and backbones and cables. Margins are probably not high.

8. Our Uncoordinated Approach to the Internet So Far

Microsoft has approached the Web much as the blind men approached the elephant, with different groups "feeling" different parts of the beast and coming to different conclusions about the Web as a whole. Hence, we have not had a consistent view of what the Web is, nor determined how to exploit the Web to achieve our goal of a PC on every desk and in every home running Microsoft software:

1. The Word team views the web as primarily an information retrieval system, and insist that the weak document formats (HTML) be replaced by the infinitely richer DOC, XLS, etc. formats that Microsoft (and our customers) have invested in so heavily.
2. The MOS team views the web as primarily on on-line service, and insist that the existing "weak" protocols **and** document formats be replaced by proprietary protocols and document format (MOSView, MPC, BlackBird; no support SMTP, NNTP, HTTP, HTML, etc.).
3. The NT team views the web as primarily another market place to sell NT and BackOffice, and so are focusing on performance, manageability, and HTTP-to-SQL server links. Netscape, by contrast, is focused on merchant servers, protocol enhancements (SSL), and server enhancements (see the 5/9/95 press release putting the Verity search engine into the Netscape server to provide higher quality searching).
4. The O'Hare team are viewing the Web as a new application platform, and are strenuously avoiding OLE, OCX, COM, and other non-Web technologies in the quest for smaller size and faster speed.
5. There is no coordinated marketing effort toward understanding the Internet and communicating a coherent Microsoft Internet strategy to our customers and the world.

9. Netscape & Verity 5/9/95 Press Release

```
Verity and Netscape Team up to Bring Topic Agent Technology to the Internet; Netscape
to Provide Popular Topic Search Engine for Netscape Servers

  MOUNTAIN VIEW, Calif.--(BUSINESS WIRE)--May 9, 1995--Two leading providers of
products for the Internet, Verity, Inc., and Netscape Communications
Corporation, today announced that the two companies are teaming up to bring
Topic agent technology to the Internet. Netscape will be able to embed Verity's
Topic Search Engine in its Netscape servers and will also resell Verity's Topic
Agent Server technology.
```

Verity's Topic search engine is the embedded engine of choice for hundreds of application developers and is used in many well-known software products including Adobe Acrobat and Lotus Notes.

Verity's new technology brings agents and topic query objects to the Internet. Topic Agents allow users and on-line providers to filter incoming information against interest profiles and send automatic alerts via personal HTML pages, electronic mail or fax. Topic objects also allow information to be automatically categorized and browsed by subject area. With this new metaphor, information providers now also have the tools to capture editorial expertise in topic objects and make them available to their subscribers.

"Making it easy for users to find and access the information they want is key to the Internet's success as a mainstream communications channel," said Jim Barksdale, president and CEO of Netscape. "Verity's Topic search engine and state-of-the-art topic agent technology combined with our Netscape servers will make locating, filtering, and accessing information fast and simple. Verity's tight integration with Adobe Acrobat also complements our support of Adobe's Portable Document Format in our Netscape Navigator and Netscape server products."

"Netscape and Verity share similar visions of ubiquity, deployability, cost effectiveness, ease-of-use and a belief in the web metaphor and Adobe's PDF format," said Philippe Courtot, chairman and CEO of Verity, Inc. "Netscape is becoming synonymous with the Web and we are extremely delighted to be working with them."

Knight-Ridder's NewsHound service is one example where technologies from both Verity and Netscape are already in use together. Knight-Ridder was an early adopter of the topic agent technology which is at the core of its NewsHound service. Netscape servers were selected to provide access to the Internet subscriber base.

The Netscape Internet Applications(TM) family is a line of turnkey software applications that enables companies to conduct full-scale electronic commerce on the Internet. Netscape Internet Applications incorporate Netscape Commerce Server(TM), which includes the Secure Sockets Layer (SSL) open protocol to enable secure commerce to be conducted over global networks.

Netscape Communications Corporation is a premier provider of open software to enable people and companies to exchange information and conduct commerce over the Internet and other global networks. The company was founded in April 1994 by Dr. James H. Clark, founder of Silicon Graphics, Inc., a Fortune 500 computer systems company; and Marc Andreessen, creator of the NCSA Mosaic(TM) research prototype for the Internet. Privately held, Netscape Communications Corporation is based in Mountain View, California.

Verity, Inc., headquartered in Mountain View, California develops and markets the Topic family of information retrieval tools and applications for publishing and disseminating information across the enterprise, the Internet and CD-ROM. The company's products and services are used by more than 650 corporations and organizations worldwide as well as by hundreds of development partners. Verity's Topic search engine is the engine of choice for Adobe Systems, Lotus Development Corp., Frame Technology Corp., Saros Corp., PC Docs, Odesta Systems Corp., Documentum, Inc. and many other software developers.

Note to Editors: For more information, contact Verity at info@verity.com or by calling 415/960-7600 or at the World Wide Web site http://www/verity.com/. Verity and TOPIC are registered trademarks of Verity, Inc in the United States and other countries. Verity, Inc. is not related to the International Stock Exchange of the United Kingdom and the Republic of Ireland Limited, which provide computerized information services under the mark Topic. Netscape Communications, Netscape, Netscape Internet Applications, Netscape Commerce Server and Netscape Navigator are trademarks of Netscape Communications Corporation. NCSA Mosaic is a trademark of the University of Illinois. All other product names are trademarks of their respective companies.

CONTACT: Verity, Inc.
Sue Barsamian, 415/960-7600
or
Netscape Communications Corporation
Roseanne Siino, 415/528-2619
or
Fineline Communications
Mary Carlisle, 408/867-9070

The End