***Abstract:*** *The Internet, which had been around for a quarter of a century, has recently bloomed with astounding speed into a household technology; so much so that T-shirts have been printed saying "The Internet is full: go away!". The catalyst for this overnight transformation was the World Wide Web. But it all went so fast! What happened? What's happening? This article attempts to replay the interesting parts in slow motion.*

# Success story: net risks melt-down as web hits critical mass

***Frank Stajano***
*fms@orl.co.uk, http://www.orl.co.uk/~fms*

*Olivetti / Oracle Research Laboratory,*
*Cambridge, UK*

Invented in 1989 by Tim Berners-Lee, around Christmas 1993 the World Wide Web was still only known to a small circle of insiders; Netscape hadn't been founded yet and those early enthusiasts were all excited about the new graphical browser called Mosaic. One year later, the web was the most fashionable thing in popular computing magazines. Two years later, some large companies were ready to bet that they could make lots of money by building cheap computers almost solely dedicated to surfing the web. Today, for a company not to have a web site whose address can be listed at the bottom of the advertisement is almost as unfashionable as not having a fax number on the business card. This really is explosive growth. Why did the web catch on like this?

## *Critical mass*

In many fields, not necessarily limited to computing, success depends on whether a critical mass of supporters can be reached. Below critical mass, the idea (or product or standard) can be arbitrarily good but, even if supported by fervent believers, it will never become mainstream. Above critical mass, instead, the mere presence of the existing supporters is sufficient to attract bystanders and turn them into supporters, thus fuelling a positive feedback cycle. The more supporters there are, the more the idea becomes mainstream. The more it becomes mainstream, the more the undecided start to gravitate towards it. The more undecided get attracted into the mass of supporters, the stronger the attraction force becomes, and the easier it becomes for the idea to gain even more supporters. This pattern — critical mass, then self-sustained explosive growth through positive feedback — is a good model for many physical, social and economic systems. Its applicability is even more evident in cases related to communications, like the spreading of the fax machine, where the presence or absence of a sufficient number of other parties to connect to is an essential criterion for a newcomer deciding whether to join in or not. An idea whose aura of influence is below critical mass, like Betamax video-tape, is one that the general public won't willingly adopt.

It is obvious that today's web, that by most estimates currently doubles in size every few months, is well above critical mass: how did it reach that crucial threshold?

## *Bootstrapping to critical mass*

1. For one thing, the web is really easy. Click on the blue underlined words and something happens. Everyone gets that. Easy action, instant reward. A straightforward, consistent user interface that requires only ten seconds of training. A triumph of software ergonomics. This aspect may at first seem trivial, but it isn't. Some might object that users are not *that* stupid and that it would be offensive to insinuate that what stopped them before was their inability to master anything more complex than "click on the underline and you go there". But those critics miss the point. Usability does make a difference. Even the so-called power user, accustomed to the complexity of today's software tools, welcomes like a breath of fresh air those precious few applications whose workings are sufficiently intuitive that they don't require looking up in the post-it note on the side of the monitor. One doesn't have to be stupid to appreciate simplicity.

2. Next, the software is free, and always has been. From the first line-mode browser to the first web server and to the first popular graphical browser, Mosaic, the creators of the web have always given the software away for free so that people could join the experiment. Commercial products followed, but they couldn't compete with the existing free reference implementations unless they offered their own software at the same price, at least on a trial basis. The leading browser Netscape Navigator, whose primary architect, Marc Andreessen, had been the author of Mosaic, conquered the market around the time when critical mass was being reached; but it was always given away for free to individuals. When it became clear that the web was over critical mass, software giant Microsoft decided that it was worth fighting for it; and, to undermine Netscape's near-monopoly, even Microsoft had to go against its nature and had to start giving away its Explorer for free. Quite apart from any judgement of value on the strategies behind this corporate generosity, we cannot avoid to note that the user is spoilt for choice, and the choice is between several excellent products, all of them free. Almost too good a deal to be true: a key factor in making the web spread like wildfire.

3. Another key factor is that, from a systems standpoint, the web respects and reuses existing goodies. The web is largely made of existing material: physical infrastructure such as cables, routers and hosts; layers of communication protocols such as TCP/IP and FTP; standardized client-server applications such as FTP, mail and telnet; file formats such as GIF, JPEG and MIDI. All these existed and worked before the web came to be; and the web, in the wisdom of its creators, didn't arrogantly reinvent the wheel: it took advantage of what worked and strove to be compatible with anything relevant. This not only allowed it to leverage on the corpus of available program code; it also made the web friendly to developers wanting to write for it. A particularly telling example of the web's openness is the format which defines its documents, the HyperText Markup Language or HTML: unlike other proprietary hypertext document formats such as Apple HyperCard, Windows Help or Adobe Acrobat, HTML is purely ASCII text with "in-band signalling" of links and formatting features. Simple labels between angle brackets, such as <b> for boldface, govern all the typographical and hypertextual features. HTML can be read and written with any existing text editor. This too was a crucial factor in bootstrapping up to critical mass: it allowed budding web authors to build the first corpus of pages before any specialised editor had been written.

4. Finally, web pages are visually appealing. A well-known 80-20 software design rule states that, although in the long run the perceived usefulness of a program will depend for 80% on how well it gets the job done and for 20% on its looks, the first impact on the user will depend for 80% on looks and only for 20% on functionality. When a net-virgin user asks for the assistance of a net-savvy friend for a bibliographic search, a character-mode telnet connection to a large library will yield meaningful but dull results, while a web search will return pages in a format that invites them to be read.

## *Hyperlinks*

There are two core ideas behind the World Wide Web: one is hypertext, the other is simple and uniform access to heterogeneous resources. Hypertext, a word coined by Ted Nelson in the 60's, but an idea going back to Vannevar Bush's description of a knowledge machine called the memex (1945), is essentially text containing navigable cross references. Bill Atkinson's HyperCard authoring system (1987) was the first popular software product to implement the concept, which has since been widely used in CD-ROMs and

on-line help systems. But self-contained hypertexts were inherently limited: all the promises of hypertext were only really fulfilled by the web, when links were first allowed to cross network boundaries. Uniform access means that the many and varied tools of the Internet, from FTP to Gopher, from telnet to mail, can now all be accessed in the same easy and consistent way, bypassing their arcane UNIX-inspired command languages. The Mosaic name was a particularly inspired metaphor: it evoked the harmonized combination of many tiny tesserae, demonstrating a system where the whole is far greater than the sum of its parts.

These two ideas together combine all the on-line resources of the net (including images, sounds etc.) into one immense hypermedia document — the web itself — to which authors can add cross-references by writing some HTML glue. New content can be written from scratch, but most of what's already on-line can become part of the web without any reformatting. The existing resources become much more easily accessible. The three basic services of the net (news, FTP and mail) are significantly enhanced by the change. News, where postings generally refer to previous messages in the discussion, can now contain machine-generated hyperlinks to the messages they quote. Files available on FTP servers can be designated unambiguously by their URL (Uniform Resource Locator), and this URL can be embedded in a mail message as a hyperlink, so that the recipient of the mail message can retrieve that file simply by clicking on the link. But while the hyperlink-enhanced traditional services are interesting, the new service — browsing, with its natural complement, publishing — has, since 1995, being generating more traffic on the Internet backbone than any of them.

## *One-click stegosaur*

Net citizens have evolved their own cyber-ecology based on the principle of conservation of scarce resources such as bandwidth. When the web was introduced, the instinctive reaction of the cyber-ecologists was that the web was an antisocial waster. Normally, to see a picture of a stegosaur you would go to the appropriate FTP site with an FTP client, fetch the file, save it to your local disc, uncompress it, and finally display it with a viewer program. If you wanted to look at it again the next day, and the next month, you would simply redisplay your local copy, without bothering the FTP site again. With the web, instead, you'd go to the stegosaur page every time you wanted to view the picture, thus reloading the same data over and over again in a throwaway fashion. What a horrible sin! But the web's approach proved to be the right one: it made the process of seeing that stegosaur a whole lot easier, both for non-computer people who wouldn't otherwise have done it at all, and for computer experts who, though capable, might in most cases have decided not to bother with the rigmarole. Besides, web clients soon evolved their own caching mechanisms (not so much for ecologist reasons as in order to improve their speed) which meant that, mostly, those extra downloads didn't actually happen after all. No, the real reason why the web is melting down the Internet is not because it's inefficient: it's simply because it's good enough that it gets used a lot.

The critical mass is attracting more and more users to the net. The population of the net doubles every few months. Of course this trend can't go on forever (we'd run out of people on Earth in a few years) but it's worth noting that, while it lasts, an immediate consequence of this exponential growth is that less than half the population of the net has been on-line for more than those few months — and in turn only a small proportion of those has been on the net for, say, a couple of years or more. Everybody, save a tiny few, is a complete newcomer. One would intuitively assume that, as time goes by, the number of knowledgeable users would grow: after all, nobody has time to die within the hectic computer-distorted time scale, and an individual's experience can only increase with time; instead, the exponential growth rate makes the proportion of knowledgeable users minuscule. Another reason why it's important to be able to get at that stegosaur with just one click.

The user interface of the web is among the most elegant that exist. To browse, you only really need one command: click on a link to follow it. All the other gadgets that adorn the browser, except perhaps for the button that goes back, are not fundamental and most users don't even know what they do anyway. Such a drastic simplification of the command set is a huge step forward in usability. It is also a hint that more complex interfaces could be made simpler by unfolding them onto this flat model. Surprisingly many small applications, especially in corporate intranets, can be reexpressed as web pages, perhaps with some behind-the-scenes CGI programming on the server or with the help of some Java executable content. Many application commands (out-of-band controls that a user has to learn) may then be unfolded into in-band

hyperlinks that are part of the data and, as such, no longer need explaining. When this is done properly, the new pseudo-applications are much more readily accepted than their real counterparts, because users run them within the familiar environment of the browser which they can already drive.

## *Indexing, robots and Java*

Naturally there are, and always will be, many more readers than authors. But authors grow exponentially too, and so does the number of available pages on the web. We have already reached a stage where almost any imaginable subject has at least some pages devoted to it, although not always with the depth and accuracy that you might desire. The hard part now is finding the right stuff. You can rely on lists of known good places compiled by your friends, but these are rarely up to date. Interestingly, the problem of finding the right resources existed in the pre-web days too: lists of good places were published in specialized newsgroups and an automated indexing tool, archie, was written to dynamically compile a list of all the software available for FTP anywhere on the net. If you wanted a given file you could ask archie which, if it could find it in its most recent listing, would tell you its location. Both approaches have been revisited for the web: the lists of hand-culled links are the basis of classified directory services like Yahoo, whereas the automatically generated indices of anything that's on-line are the fuel of search engines like Digital's Altavista. The web would drown in its own noise without these aids, which work with surprising accuracy and effectiveness. One could comment that is now only the software robots sent around by the search sites that have the time and patience to visit web pages. I recently wrote a Java applet and uploaded it to a public repository for free distribution; the repository allowed visitors to give any applet a score from 1 to 10 by clicking on a hyperlink. I came back after a few days and found my average score to be 5.5, which I thought wasn't brilliant. For calibration purposes I looked at other scores only to find that *all* the applets had an average score of 5.5. Suspicious… After some investigation with the site administrator it became clear that this was due to the automated web crawlers regularly hitting all the links from 1 to 10, with an average of 5.5, outweighing any scores from genuine human visitors! Another lesson I learned from this is that web crawlers have rendered access counters rather meaningless as a measure of the popularity of a given page.

Java, as we all know, is a programming technology that enables pages to include executable contents: when the page is loaded by the browser the Java portion of the page, known as an applet, is interpreted as a program and executed locally on the client side. This technology, publicly released by Sun in 1995 and now incorporated in all the major browsers, has provoked much excitement in the web community. But some are sceptical about its true merits and deny having ever seen a useful applet. What are applets good for? Let's attempt a rough classification. The most commonly seen applets (probably the ones that give Java a bad name) are eye-catchers with no other purpose than to spice up a web page: scrolling LED signs, moving billboards and the infamous animations. Then there are the solo games: nothing special, you visit the page to play. Then there are what I call the viewers: this is an interesting category. Your page contains some data that doesn't fit into any of the classical formats understood by the browser, so you ship a viewer applet along with it that can render it at the client end. The classical example would be a 3D model of an object, which you could embed in the page with an applet allowing people to view it from every angle; but I'd like to also mention a viewer I recently wrote for a friend who is a chess expert: this applet lets you embed a chess game in your page, in such a way that people can single-step at their leisure through the moves and see the position of the pieces on the board. I feel that viewers alone justify the introduction of executable contents. A fourth category, perhaps the most interesting, covers client-server applets that cooperate with a back-end program over the network. The site you're visiting provides a service (and a server to implement it) and gives you the matching client when you visit the web page. A classical example would be a stock ticker. But the most fascinating example of this kind is probably Javatel (teleporting in Java), developed at ORL in 1996: the server maintains an X Windows session for the user, for which the browser acts as the display. The user can move to anywhere in the world and, by simply visiting the password-authenticated Javatel page through any available web browser, he can instantly teleport the display of his personal X session to that browser.

## *Web publishing*

Is the web just a well-designed graphical user interface around the rest of the net? It might have been at the beginning, but now the answer is a definite no. The web has invented its own new service, publish/browse, which is popular in its own right. The beauty of this new service is that it only gives out information on demand. Unlike news or mailing lists, where you have to unconditionally subscribe and receive lots of useless data before finding a nugget of information, with the web you browse when you need it, looking for what you need. This paradigm opens the door to new uses for the net, not the least of which is ecological advertising. While sending unsolicited mail ads would be a no-no, setting up a web site with all the possible information about your company's products is a genuine service. Whoever is interested in that information will come to you to get it, voluntarily, and grateful that you've put it on-line.

A common observation is that the web, by letting anyone publish documents at a modest cost, enables a practical freedom of the press, more democratic than the conventional one which is actually only available to those few with enough money to buy and run a printing press. What is seldom noted, however, is that pages published on the web are much more volatile than those printed on paper. In the pre-web Internet, when a document was made public by leaving it on an FTP server, people would at least download it and save it to their local disc to read it. With the web, the custom is to simply go to the page, without saving anything. The web is seen as an extension of your disc. If you think you'll want to refer to that page again, you simply bookmark it. What if the author then changes the page subtly? In the FTP case, you'd still have the old copy; but with this new procedure, where you rely on the net to store it for you, you won't: you'll get the new one without being able to refer to the old, unless you went through cumbersome and unusual steps to save the previous version when you first viewed it. Without going into discussions of cryptographic protocols, let's simply remark that web publishing as it is currently practised has the potential for Orwellian denials — changing the past when it is no longer liked.

## *How can we speed it up?*

The net seems about to melt down under the success of the web. The web can be very slow, especially at the wrong time of day, and what's more frustrating is that I can't buy a faster modem to solve the problem: the bottleneck is not between me and my provider, it's between my provider and the rest of the net, or somewhere further down. Apparently, many providers are overselling bandwidth by too large a margin. Of course the wide area network infrastructure will become faster: thanks to the magic of fibres, we can still expect improvements of several orders of magnitude in the carrying capacity of the physical layer. We can also be confident that we'll soon get a much better deal on the bits we receive over the phone line. The current tariffs for modem-based data transfer are based on the traditional fees for the voice phone service, but are grossly overpriced, in terms of dollars per bits, compared to the cost of the technologies used to provide the links. This can be readily appreciated by comparing data rates and prices of telephone with those of cable television. Even the phone companies themselves acknowledge that this tariff is an anomaly; they'd be ready to supply twenty times the bandwidth for the same price, and the primary reason why they don't do it yet is that they want to avoid someone else resplitting the new big link and subletting old-style voice links for a twentieth of the price. It is an inescapable fact that bits are inherently much cheaper to transport than what we are currently charged; so, thanks to competition, we're bound to get some interesting offers in the near term.

Notwithstanding that, though, more users will join in; and, on top of that, we are pretty good at filling up whatever new capacities technology can offer us (just think of the smallish gigabyte hard disc in your current laptop, which would have felt infinitely spacious in 1990). We'll have no remorse at throwing continuous media at the net: even the background of your ordinary web page will become a true-colour moving video clip with hi-fi stereo soundtrack. Finally, the increasing importance of mobile communications, and the consequent use of wireless channels, will take us back to capacities well below those delivered by fibres. So there will always be circumstances in which bandwidth will be scarce for our intended purposes. What can be done? Well, there still is great potential in making efficient use of whatever is available. Not all the significant improvements are necessarily measured in bits per second, as our own engineering optimizations sometime (mis)lead us to believe. One of the primary reasons why Netscape quickly took over from Mosaic was that it was much faster. Did it get any more bits per second out of the

network? No, it simply displayed the page while loading it and allowed you to scroll and follow links before having received it completely. A brilliant example of maximization of the user's *perceived efficiency*. This is an area in which there are still many opportunities for creative contributions.

On the subject of perceived efficiency, we shouldn't think of bandwidth as the only indicator of speed: latency often counts even more, especially in interactive user interfaces. The low resolution of an otherwise smooth video clip (bandwidth problem) is not as annoying as the ten second delay after each mouse click while browsing (latency problem). One way to reduce latency is through caching and prefetching. Expect advances in this area as browsers become smarter and take advantage of extra hints on what you're likely to do next: while you're watching a page, your browser could spend its otherwise idle time prefetching all the pages pointed to by this one, just in case you might hit one of these links. And, having done that, it could move to the second level of links. Of course this strategy runs out of cache space rather quickly, but it could be improved by fetching links according to their score — a score sent out by the server and based on the popularity of the various pages on that site, gathered from past statistics. More hints could come from the user's own bookmarks, which could be periodically prefetched in anticipation for a potential visit. The user might even mark some pages as particularly interesting, making them hard to throw out of the cache, or (in the case of volatile pages like news) forcing the browser to regularly reload them in the background. When continuous media becomes commonplace on the web, even more sophisticated caching and prefetching techniques will have to be perfected to ensure that the clip starts up reasonably quickly when the user clicks on the link.

## *Is it just fashion?*

Someone might notice all the media hype and wonder whether the web is just a passing fad. It isn't. Every ten to fifteen years there is a great event that fundamentally changes the face of computing, and I personally consider the birth of the web to be such an event, the most significant innovation since the introduction of the personal computer. The personal, a computer on everyone's desk, didn't solve everything: it often crashes, it's often a configuration nightmare and it changes beneath our feet faster than we can learn its new functions. Still, we'd never want to give it up: the versatility with which it allows us to manipulate information outweighs its disadvantages. For the web, it's similar. The unification of the information sources of the world into one gigantic hypermedia document to which anyone can contribute won't solve every problem, and won't be perfect from the start; but it's a fundamental technology with a profound social impact whose trace in the computing universe is already visible from very far away.

## *Further reading*

For an exhaustive panorama of what's available on the Internet and how to make it work:

KROL, Ed, *The Whole Internet User's Guide & Catalog*, 2nd ed, O'Reilly, 1994.

For visionary opinions on what the future of the net could be:

NEGROPONTE, Nicholas *Being Digital*, Hodder and Stoughton, 1995.

GATES, Bill, *The Road Ahead*, Viking, 1995.

To dispel the myth that the net and computers in general are necessarily a good thing:

STOLL, Clifford, *Silicon Snake Oil*, Macmillan, 1995.

To take the pulse of the web revolution (a monthly fashion magazine for computer buffs and an incredibly up to date source of excellent and thought-provoking information, if you can stand the psychedelic colours and layouts):

*Wired*.