**W3C meeting on “Web Efficiency and Robustness”.**

# A Trip Report and some reflections.


## Harald Skardal, FTP Software Inc.


## April 22, 1996.

Note: The following is my personal recollection and impressions of the meeting.

Jim Gettys of DEC/W3C invited to a meeting to discuss the topic of “Web Efficiency and Robustness”. Some key themes were: Very young Web technology is used for corporate mission critical applications; Popular servers are almost unreachable due to congestion; “Flash crowds” generated from a URL in a commercial during Superbowl brings the Web to it’s knees; etc. How can we improve the Web to handle these requirements better?

The attendees covered a wide spectrum: content providers, ISP’s, telecom companies, and Web software developers. In addition Jim Clark and Van Jacobson brought in a lot of experience from having built the Internet over the last 25 years.

Jim Gettys used his opening statement to describe some of the issues, and issued the personal opinion that he saw the proxy server as the handle that most easily could be used to solve the problem.

A lot of the meeting was centered around “caching”: being able to “serve” content from a place “closer to” the user than the original source. Unfortunately I think the debate suffered from the lack of appropriate language: caching covers everything from a server side mirroring strategy to a client side statistical based caching; I believe strongly we have to device a language that recognizes all the different levels of “caching”. They are indeed very different. We need to understand the different constraints and requirements so that we can provide the appropriate solutions and strategies for each level.

### What I remember from the presentations:

Steve Beckhardt - from Iris/Lotus talked about Notes. Fyi, replication wasn’t done until Lotus bought them, and they wanted to synchronize projects between Westford and Cambridge. Replication is typically “carrier grade”, not “fault tolerant”. This means that documents are only replicated “pretty often”. There are mechanisms for providing a “get me the most up-to-date version”. These are seldom used. Also, Notes also has no truly “master object”, an object is a named thing which could be anywhere in the network. Location is not a part of the name. The system makes sure that there is “at least one instance” out there. I believe this is important.

Margo Seltzer from Harvard University has been carrying out analysis of Web server log files. Her findings suggests that the highly requested portion of a servers content is much less than 10% of the content, but also that which fraction that are popular may change frequently. This suggests that there is a very strong benefit to some caching/replication scheme as long as it happens “close to” the server. An important point: Web traffic is still growing exponentially, it is doubling every three months.

Mike Little from Bellcore described an attempt to correlate improvements to the server and perceived benefits to the user. The motive is to develop metrics such that we can narrow down where are the biggest bang’s for the least investments.

John Ellis from Open Market talked about their OM-Express, a client side proxy server based Web utility used to download content “packages” that the user has subscribed to. The user can read the content at full speed. Included is a

allows the user to customize on top of this. John's main point for this discussion was that while we are figuring out how to improve the "network and content infrastructure" in order to avoid melt-downs, programmers are coming out with "hacks" that solve smaller specific problems. We should study them carefully, and also be aware that many problems will be solved that way.

## The discussions:

The discussions were quite unstructured to begin with, reflecting a vast problem space. After some failing attempts we decided to spend 30 minutes each on a few topics: naming, caching, replication. I had to leave before the final word, so I did not catch the last hour or so.

A couple of key statements were made that helped me organize the problem: Van stated that caching is always more efficient closer to the server (the root) than closer to a user community, (the branches). This is a known fact from multi-level memory systems, and it was underpinned by Margo's analysis. Secondly, Van stated that the biggest problem with "caching" in the Web is that the Web is like a tree where the root is moving around all the time depending upon the popularity of each site. Van stated that all evidence suggests that "manually configured" caches do not work. My interpretation: statistics and a simple policy is the only base for a cache.

There are phenomenons in the Web working against caching. One is the "hit count problem" where content providers modify the URL's to be CGI/non-cacheable URL's in order to improve "hit count", which is needed for advertisement counting.

John Ellis also stated that the Web is too slow for business, $60M Web business compared to $6,000,000M total business in 1995.

I suggested that to start with we define some language to let us describe the different problems without mixing them. Initial suggestion: replication is controlled by the server and or the "operating system", it is based on some pre-assigned strategy/policy. Caching is a statistical based mechanism, often closer to the end user. In retrospect, the concept of "mirroring" should be included here too. The important point is to get some precision into the language so that we can discuss more constructively.

Van suggested that we use the URL, which is a unique name, as a basis for a universal name. The server name is just a part of the name, it also "happens" to be the source of origin for the document.

## Some further thinking:

The following summarizes my thinking on this topic. The first step in solving the problem is to partition the problem space into all the different aspects of "content copying" that are relevant for this problem: copying: mirroring, replication, caching; location: server, network, proxy, client side; strategy: statistical, user/system driven; etc. According to what we know we should be able to estimate, and prove by experiments, the value of each selected aspect and option.

Secondly I suggest looking at the problem as at least having three layers: the server with content, the network, and the clients. Additionally there is the organization's network (Intranet). These levels have different requirements and properties. The borderlines between them are important transition points for the traveling content.

The "caching/replication" problem can be handled at several levels, under control of different parties. The content provider, who hopes to become a very popular site, is interested in spreading content across many servers/connections in order to spread the load for flash-crowds. Content providers can cooperate in creating a mirroring network to be prepared for flash-demand. They will not all be popular at the same time.

A next problem is to create a mechanism for mapping a server name to many physical addresses. This can be done by the content server with Web re-direction, or by a new "DNS distributing mechanism" in the network layer. The latter

The network itself can contain caches. The larger ISP's, such as AOL, CompuServe, PSI; would most likely get high hit rates out of their caches since they represent if not the trunk, so at least thick branches.

The intranet is typically connected to the Internet via a firewall/proxy server. This connection point is perhaps the most obvious attack point for caching, but may not yield as much performance improvement as hoped for.

The end user client may have pre-fetch capability, for instance based on content "subscriptions", or at runtime based on "load all pages referenced from the current one". This may improve the condition for the user, but could overall increase network load considerably.

I do believe that the current "hit count" solution is only temporary, eventually content providers will have to rely on other mechanisms. When everybody subscribes to Pathfinder, how does Time-Warner know that I actually read all that I downloaded? The problem could be better solved if the content itself, the page, could send a signal to a "hit count server" when a set of content was actually read. This way intermediate caches, or mirror servers would not hide the actual reading audience from the content owners. Java/ActiveX applets could provide this mechanism.

## In the end:

I have not thought about this in any detail, nor developed any statistics to indicate that one particular issue nor solution is more or less important than any other. The above is simply an attempt to help us see the big problem as a set of smaller problems. These smaller problem areas and the relationships between them can be analyzed and solved much more easily. By limiting the scope, we can also much more easily determine the importance of each area for the different symptoms that we are observing or expecting to observe over time, and develop and provide specific solutions more quickly.