Thinternet: Life at the End of a Tether

H. Shrikumar <shri@cs.umass.edu>
Rehmi Post <rehmi@twain.ucs.umass.edu>

Abstract

As the Internet continues its exponential growth (with increased commercialisation of its latent value), the user profile is also changing. Many of the newer hosts are personal machines and a good number connect via dialup or other slow links. We examine some factors that motivate or mandate "thin" (low-bandwidth) connections to the Internet. We notice different motivations in the west and in developing countries, yet there are underlying technical similarities. This helps us draw a profile of today's typical user. We use these observations to make a case for some techniques and methods that could make Internet practical and adequately efficient even at the end of a tether. We are exploring the application of these methods to routine Internet use, from a site in India (a software development laboratory, multi-user LAN) and from mobile computers (e.g., HP100LX and Gateway Handbook) in the US. In each case the user sits at the end of a relatively thin link.

We consider **transparent caching** *and* **replay** *of application protocols to be a useful and practical way to implement transaction spoofing, in a manner transparent to the application, user, and network. Cache effectiveness can also be improved by* **prefetching. Progressive encoding** *exploits the ability of the user to judge the value of an information object based on a preview of certain fragments, and diminishes most of the inconveniences associated with casual Internet browsing from the end of a Thin Wire. We suggest that such encoding methods apply not only to graphic image files, but also to large archive files and real time communication.* **Filtering** *and* **relevance feedback** *have been recognised as effective tools in overcoming information overload. We find that Usenet surfing lends itself to a some filtering methods particularly well because of its nature.*

I. Introduction

The Connected Internet is still growing exponentially, in both scope and traffic volume; recent developments in commercialisation of the net have only fueled this expansion. New connectivity and usage patterns are being established. One noticeable difference is the change in demographics of the Internet - the rapid increase in the fraction of smaller machines and personal computers, the number of dial-up interconnections in the west, and the number of sites and nets in developing countries. This is driven both by the increased availability of this class of machines, and a larger class of new users who are used to the PC rather than UNIX workstations on an Ethernet. Such "Thinternet" connections, whether via dial-up, wireless, cellular or VSAT, are all characterised by limited bandwidth, usage charges largely or entirely insensitive to net bytes of traffic, but charged by the minute of usage, and demand-based end-host-initiated connection management.

Some services (e.g., audio and video multicasts) implicitly assume that the user's network connection will be "thick" enough to match the requirements. In fact, the bandwidth required to be a first-class Internet participant can become quite high.

It used to be a rule that participants in the Internet were all connected first by 56kbps links and progressively upgraded to T1 and T3 capacity pipes. But today we are seeing an addition of one new hookup to the Internet every 10 minutes[1], a sizable number of them being dial-up. And that segment is growing fast. Some users are now being served by auxiliary communication services, such as CATV based systems[2], and these also hold a promise of higher throughput. New local-loop technology such as ADSL (Asymmetric Digital Subscriber Loop) also hold a promise of bringing greater bandwidth in one direction. But the deployment of these services cannot be taken for granted in all areas and in all countries of the world.

It has been pointed out[3] that the much-vaunted Information Superhighway would demand a whole new approach to the problems of network management, and consequently would not be born overnight. The deployment rate of network "infrastructure" is likely to be constrained by these and other, financing related issues. This is another reason we expect that low-bandwidth channels, including high-speed dial-up, will be around for quite some time, supporting a large community of Thinternet subscribers.

Given the wide acceptance and brisk marketing of the V.32bis/V.42bis 14.4kbps modem standard, a very large user base of these are likely to be around for a while. Also in developing countries, such dial-up could well prove to be the only option due to various regulatory and technical circumstances[4]. This is a viable connection strategy, because today's 14.4 kb/sec modems can provide transfer rates (over high-quality telephone lines) of about 1.2 kB/sec or 4.2 MB/hour for raw binary data, after the average protocol overhead is subtracted. Although this bandwidth is not adequate for current audio or video multicasts, it is evident that such links could be put to more effective use than is currently done.

In a leased Internet link, the line aggregates traffic from multiple users, and packet queues tend to even out traffic to utilise the links better. Yet in the case of a personal computer at the end of a dial-up PPP link, the traffic tends to come in burst, with long idle periods. It is these idle periods which we wish to utilise more effectively.

In this work we attempt to focus on the needs and circumstances of users who might not have the fortune of high-speed leased line connectivity, and show how they could still participate effectively in the breadth of the currently popular Internet activities.

Our methods focus on making observations about how Internet applications have changed since its inception, and on taking advantage of particular properties of the current most popular or valuable applications. We also suggest methods to extend the availability of "Internet access" to larger world communities.

II. Users of the Thinternet

In trying to identify the needs of the "typical" Thinternet subscriber, it is as easy as it would be incorrect to stereotype the user too narrowly. For instance, it is easy to assume that dial-up subscribers simply cannot afford wider connections, and would if only they could justify the expense. Such a user might have chosen dial-up connection to the Internet for reasons other than affordability. For example, many organizations with fairly well equipped LANs and computer equipment choose to obtain only a dial-up connection to the IP Internet, as it adequately meets their perceived needs.

While there is a lot of dial-up usage in developing countries, we note that they are not the only dial-up participants of the Internet. Interest in wireless[5], cellular, and mobile Internet connectivity[6] has been increasing. Laptops, palmtops and other portable computers are becoming significant as points of contact with the Internet. Some of these devices, such as the Apple's Newton, IBM's Simon, and InFolio's Personal Digital Assistant, and the Bell Northern Research Orbitor are evolving into powerful pocket-ready appliances. It is easy to foresee that such devices will be a common interface to the World Wide Web[7] and similar applications in the near future.

Our techniques also benefit parties already connected to the Internet by leased lines, by reducing duplication of traffic[8]. Many of these connections are already straining under the demands of many of the newer applications as well as the volume of more traditional traffic. Software distributions available via FTP are increasing in size and NNTP traffic is always on the rise. In almost every aspect, volume is steadily increasing.

II.A. The typical Thinternet user

For the purpose of this discussion, our "typical" user has some sort of personal computer with a "comfortable" amount of secondary storage, as typified by the new generation of high density hard disks. Our user also has access to a **network service provider** with a thick connection to the Internet backbone, perhaps through a network server running Unix, or a terminal server. This access would be through a **dial-up** link, a **voice-grade** leased line, or a **wireless** connection. Such links are not assumed to be available all of the time because of usage charges, unreliability, geographic coverage, or a combination of these factors. All the same, most users have no wish to be relegated to a second-class citizenship on the Internet and will not be content to just thumb a ride along the Information Superhighway.

III. Thinternet Design Principles

In this section we briefly touch on some design principles which guide our approach. These are not new ideas; in fact, many of them are getting significant attention, particularly those which deal with Information Retrieval[9] and Information Overload[10].

These are the particular methods, from among a number of possible solutions, that we find elegantly suited to the kinds of computers, networks, and connectivity styles that we expect in the near future.

III.A. Filtering

Much interest has come to bear on methods of filtering and presenting streams of information so as to increase their value. Commercial services such as *DowQuest* and Prodigy's *Journalist* provide means to watch various news sources for stories which have high potential value to the user. Evaluation of this potential is based on a dynamically maintained profile of the user's interests. Other issues concerning the design of filters in messaging systems are also discussed elsewhere in this conference[9].

We also notice that the Usenet Group Conferencing system (and by extension the forthcoming **X.gc** standard) has some unique characteristics that greatly simplify the filtering problem, permitting some very simple methods that work surprisingly well. We have achieved good results by running an *indistinguishably complete* Usenet feed into a Hewlett-Packard 100LX palmtop computer despite its limited bandwidth and storage space, and are experimenting with the same at a multi-user site in India served by a slow dial-up link.

III.B. Transparent Caches and Prefetching

Transparently maintained caches and mirrors of frequently accessed repositories of information have also proven their utility in reducing the dependence of central machines. They also alleviate the load on communication lines.

Many of the currently popular Internet applications are client-server oriented, and lend themselves to simple caching methods. We have found that several of these protocols can be spoofed using a simple server technique. We are in the process of experimenting with this to provide "virtual Internet" connectivity to a site in India with a LAN hosting several PC/Windows and Sun class workstations, and to personal networks in the United States. However, while this solution is easy to deploy and retrofit, its elegance is marred by quirks and deficiencies in some of the protocols it spoofs. We urge greater attention to these issues (discussed later) in protocol revision and evolution.

It also appear that cache hit ratios can be improved considerably by pre-fetching objects based on the branching probabilities of threads of browsing. These can either be statically determined or dynamically discovered. Quantitative information concerning patterns of reference and locality in the Web is hard to come by, but this appears to be an avenue worth exploring.

III.C. Found+Lost file systems.

Transparent caches become much easier to implement and maintain with an addition to the filesystem in the underlying OS. We submit that it is possible, considering the amount of disk space now available even at personal workstations, to maintain local caches of such data. Many Web browsers (mosaic, lynx, and others) already cache recently fetched documents, but delete the cache after each session to avoid clutter. If should be feasible to efficiently allow these caches to be retained and garbage collected when necessary. This would be particularly useful if the maintenance of this data and cache occurred transparently to the user, and did not require active partitioning of disk-space into separate areas for routine applications and caching.

With such management of disk-space, a user would not have to be burdened with actively managing the disk space allocated to the cache, and would not be forced to make a taxing decisions of what to keep and what to age. Also he or she would not be required to maintain mappings of data objects obtained across the networks, and their URLs to local files. These are the two issues that are a big bother for many Internet surfers, and partly the reason why some Web Home Pages are accessed so very often.

The system software would allocate disk blocks from currently free and idle data areas to such caching. On demand for more conventional file space the system would automatically delete the least valuable piece of data from the cache of network objects, and release that space for regular use. We will show how this can be done for DOS and Unix based machines.

III.D. Anytime encoding

Files compression has become the norm, but the methods used today are not very sensitive to bandwidth restricted circumstances. We must remember that a lot of "surfing" on the Internet today is oriented towards browsing, with serious attention paid only of some of the objects. it is apparent that "Progressive" or "Anytime" methods of encoding would be useful in reducing the quantum of information that must be fetched, without requiring any structural changes to the current applications. Image files are conventionally compressed in the JPEG, MPEG, or GIF formats. The JPEG standard also has a mode known as *progressive encoding.*[11] This mode is perhaps more suited to bandwidth limited applications. The fetching of the image and the display can be pipelined easily in the application, and the display of a progressively encoded image can be begun even when the transfer is still in progress. The transfer can be aborted anytime, when the information so far received has satiated the users initial curiosity, typical in the browsing mode of usage.

Progressive JPEG however is computationally intensive, so we are considering alternatives such as Discrete Cosine Transforms, Haar transforms, and recursive delta codings. However, we would like to emphasise that such a method can be used provided that the application protocols are extended to support multi-threaded communication, and allow aborts without catastrophic consequences, which is a feature in what we refer to below as Personal Protocols. Anytime encoding is not restricted only to Progressive Encoding of images but has a wider application. TAR and ZIP files and other similar large archives can also benefit from such methods. Reordering of files within the archive could make this of high utility, e.g., README files could be sorted to the beginning of the archive. This sort of reordering is easily done with TAR files, but ZIP files keep their directories at the end of the archive file. This is one reason that file archive formats should become transparent to the Thinternet user.

III.E. Personal Protocols

Communication between a Personal computer and a network service provider might need to keep several threads open within one single application context in order to achieve the aims of Anytime Encoding. For instance, image objects embedded in a HTML document can be transmitted in parallel with the rest of the document, thus the transfer of the text of the document is not held up by a big GIF object just preceding it. This would allow the user to browse or glance at the text matter even before the image inset has fully arrived. Notably this does not require the HTML to be re-engineered, it only needs a superficial modification to the HTTP access protocol.

The image can be encoded further using Anytime methods so the user sees images gradually get more and more clear with passage of time, and he can abort and proceed ahead by clicking any hypertext button anytime.

We propose a lightweight protocol, the Personal Protocol, which can run above a traditional TCP transport layer. This can maintain such multiple contexts and their containments efficiently.

Another problem that bothers IP users at the end of low-bandwidth links is large and variable queueing delays in the routers enroute. This can make on-line communication like **irc(1)** and **talk(1)** very frustrating. Interestingly it turns out a very useful chat application can be built over such a Personal Protocol, which can help maintain Causal Ordering of Broadcast and can probably achieve usable multi-party on-line chat over limited facilities. We intend to do some experiments in this area between a site in the US and multiple scattered sites in India connected to the backbone via a mesh of thin links.

III.F. Personal Protocols - Close Coupling

Another application of Personal Protocols is in blurring the distinction between a network host and a dial-up client. Some existing applications already do this, such as Tim O'Reilly's **term** program insofar as they allow a dial-up client to share the IP space of the host they are dialing into. This has high utility in very resource constrained machines like Hewlett-Packard's HP100LX, in situations where the host becomes a backing store that is not always available. This also obviates the need for user managed file transfer protocols like **ZModem** and **kermit,** and can multi-task a link with other simultaneous use.

This is one way to boost a dial-up link utilisation. With this background, the Personal Protocols is being designed with some RealTime properties and to queue different classes of traffic based on priorities. It can also be argued that this provides an adequate short term solution to the IP name space problem, till IPng[12] is widely deployed. We plan to exploit this in providing unrestricted IP client services to the LAN in India out of a very small IP allocation available from our dial-up service provider.

III.G. Dynamic URL replacement

The Uniform Resource Locator convention, formally introduced by the World-Wide Web to support the hypertext metaphor, looks something like this - **http://www.cern.ch/WWW94/Welcome.html** This gives an opportunity for a nearby server to edit the URL on the fly to refer to an image of the object within its domain. This server can then act as a mirror, and reuse documents recently fetched in a different communication to conserve the number of accesses to the real server. There are several HTTP mirrors that work on a principle similar to this.

It also suggests to us that a hierarchical system with chaining and reference passing, as in the X.500 directory service[13] is well-suited to the purpose. It would seem a good idea to import some of these ideas into the current Web architecture.

IV. Implementing the Thinternet

IV.A. NSFNET statistics

Data gathered by the Merit corporation[14] shows that FTP accounted for 40% of the traffic volume (in bytes) during February of 1994. Including the volume due to NNTP, SMTP, TELNET, Gopher, and HTTP - all in rank order - we recognize 70% of the total NSFNET volume. In fact, these are the 6 top-volume TCP protocols in use.

To put these proportions in context, note that the total NSFNET traffic has almost exactly doubled every year since 1991, with a volume of about 10 terabytes (10^{13} bytes) in March of 1994 alone, in contrast to 1.27 terabytes in March of 1991. The total number of hosts on the Internet, as determined by Mark Lottor's ZONE program has also been doubling yearly.

Finally, the newest of these protocols - Gopher and HTTP - have been rapidly gaining volume since their introduction, and are likely to be very important in the future. Many commercial operators, such as CommerceNet and America Online are adopting the Web as a saleable metaphor. Popular Web servers are already under tremendous loads, and the future promises even greater demands.

We draw several conclusions from this.

First, in the emerging Internet, Web/Gopher, FTP, USENET, besides different flavours of store-and-forward mail would account for most of the traffic. (Based on current Internet usage patters, these account for about 74% (by packet) and 90% (by volume) of Internet traffic).

Second, the backbone is getting overloaded because of replicated traffic which could in many cases be handled by a hierarchy of caches.

Lastly, the Thinternet user's connection is more and more likely to be swamped unless efficient filtering is applied for unsolicited subscribed traffic such as Usenet.

IV.B. Postel spoofing

Of the 6 top-volume protocols, all but TELNET belong to a family of what we refer to (perhaps naively) as "Bhushan-Postel protocols", or "Postel protocols" for short[15, 16]. Such a protocol is distinguished by its simple, line-oriented syntax. They implicitly define an abstract association control and have a strict lock-step progression. SMTP and FTP are classic examples of this style, with their four-letter command names and three-digit response codes.

We first observe that **FTP, Gopher,** and **HTTP** are essentially idempotent, lockstep protocols. By idempotent we mean that a transaction or series of transactions in one connection can be repeated (in order) with the same result. The protocols operate in lockstep because each request/reply transaction must complete before another can begin.

Service	% Pkts	% Bytes	Thinternet?	
ftpdata	20.632	39.266	spoofing	☜
(other)	19.026	12.582	-	
telnet	14.686	5.670	-	☜
nntp	8.456	9.751	quern	☜
smtp	8.046	6.711	MX/POP*	☜
domain	5.872	2.869	spoofing	☜
icmp	3.945	1.788	-	
ip	3.467	5.563	-	
irc	2.567	1.448	-	☜
gopher	2.455	3.470	spoofing	☜
ftp	2.000	0.854	spoofing	☜
www	1.843	3.044	spoofing	☜
X0	0.816	0.930	-	
talk	0.708	0.385	-	☜
vmnet	0.677	1.263	-	
finger	0.524	0.232	spoofing	☜

Table 1. NSFNET top 16 ports for 2/94.

Now consider a client connecting to a remote HTTP server through a router, and imagine that the router just happens to log all HTTP transactions. When the client requests a document, it sends a GET command and a URL. The server will respond with a stream encoding of the requested document. The client's GET command and the server's response comprise a logical transaction, which the router can supply to future HTTP clients, provided that it spoofs the entire communication.

With little more than the addition of logging and spoofing abilities, the router can be equipped as an HTTP cache. It may now intercept all HTTP traffic to receive document requests. If a requested document does not reside in the cache, then the router must act as an intermediary and contact the server as before.

More interestingly, the same technique, and in fact the same spoofer can work for all the application protocols we list above. Noting that this covers a lions share of usage in the emerging Internet, we conclude that a good hit-rate in such a spoofer can multiply the usability of a "ThinWire" link very much.

The spoofer essentially decodes all TCP traffic much as port-based access control in current routers works. It then caches traffic as a non-deterministically branching sequence of lock-step Postel transactions. On demand this is replayed, and when any deviation is detected, the real remote server is connected up and brought-up to sync, again by replay, after which the client and real server are connected in direct communication.

While this would work reasonably well, it is not very elegant. It should still prove adequate to provide our site in India to have significant Web browsing facilities. Because spoofing breaks the end-to-end semantics of communications, some Web based services such as the providers of periodically updated information (e.g., BasketBall scores)[17] and in the future, commercial information updates would break. Still, most of the form-filling and other use Web use (even in the new CommerceNet Plan) would work.

A far better solution would be to reorganise HTTP to have a hierarchical structure. We dwell on this a little later.

Our plan is to manage the cache in the Found+Lost filesystem, with objects being aged purely by being driven off the disk in LRU order. This should be adequate for most of the casual Web browsing we anticipate in the short run from our LAN in India; perhaps a better Web architecture will emerge[18] in the meantime.

IV.C. Anytime image compression

Consider now some of the time constraints that drive user interface design. One such constraint comes from the way the human eye constructs a scene from pieces to build a complete mental image. An artist takes this effect into account by composing a scene so that the eyes flows smoothly from one point of interest to the next. Such compositional techniques also apply to the layout of text and images on a page. User interfaces can also be crafted according to this principle.

If you have used the Web with a graphical browser such as Mosaic are probably aware of the way that a page is displayed from the top down. Images embedded in the page cause pauses to occur while they are fetched from distant hosts or heavily-loaded servers. While these delays are annoying, they can be alleviated by using a local HTTP gateway which transparently caches remote documents. The odds that a desired document will be in the cache roughly improve as more users browse the Web through the cache.

This caching (which could be done at the level of Postel spoofing as discussed above) makes Web navigation more pleasant for users on a local network, but does very little for the dialup user. Mosaic already allows the user to request that image transfers be postponed until the text of a page has been displayed. It even allows the user to stop a transfer in progress, so that browsing can continue.

We are exploring a browser which uses anytime encoding of images for dialup Web users, with a display strategy very similar to Mosaic's except for three differences. First, the user does not explicitly cancel transfers, but instead simply chooses any desired link which is on the screen. When a link reference is followed, all fetches pertaining to the current page are canceled by the browser, and attention turns to following the link.

The second important difference is that images are encoded so that their level of detail increases as their transmission progresses. The image "comes into focus" as the user watches, and this is made possible by the choice of image format. The encoding algorithm is discussed below in more detail.

The third difference is the order in which page elements are transmitted. First to be sent is HTML description of the current page. When the on-screen portion of text has been rendered, the browser requests that all on-screen images be sent concurrently over separate high-priority channels of the personal protocol, while switching transmission of the HTML document to a lower priority. This is followed by concurrent transmission of any on-screen images over several channels. These are sent at high priority, while the remainder of the HTML document and its associated images are queued for lower-priority transmission.

IV.D. Quern: a netnews filter

USENET has its origins the UUCP world, and hence has some properities that naturally make it suitable for dial-up applications. However, the original designs did not anticipate the tremendous volume that USENET traffic has grown to be, feeding on the considerable bandwidth of the Internet backbone.

We have given some thought to this new context and have made some experiments in making this incarnation of the USENET news more accessible to a user with dial-up access. First we note that the net demand for news style information to a users sensory system is of the same magnitude as that provided by dial-up links, and argue that with proper filtering closer to the backbone prove quite adequate.

DowQuest uses inverted indices and scores to evaluate the relevance of a news item to a subscriber. We however use simpler methods that are more tuned to the Usenet context and exploit some patterns of usage that make it easy to filter. In particular, one of us (HS) has shifted his entire interaction with the USENET system to a palmtop which is always conveniently carried along, shifting news browsing to the otherwise idle moments in a day.

Our approach differs from those such as [20] which applies WAIS-like queries to the entire amorphous spool of Usenet submissions to locate articles of potential interest. To develop the concept, first we recognise that acceptable Usenet browsing can be done dialed into a host with a full feed. Here the dial-up line is still a bottleneck and the users rate of reading limits the information transmitted to not more than a page a second. We ask if such filtering cannot be done at the previous level.

An application called **quern(1),** currently implemented in perl scripts on the host with a full Usenet feed and some enhancement to the DOS more(1) utility, filters news in a two stage process with relevance feedback.

In a first stage, interesting threads from pre-subscribed newsgroups are selected in a quick session every day. This is not unlike the thread selection step in the popular **trn(1)** newsreader. The full text of the articles are merely stashed away for transfer via dial-up, and perusal at a different time. Articles of particular interest are notified back to the node connected to the Internet, where they are retrieved and stored for additional future reference. And also used to build a profile.

With such filtering, the bandwidth of interaction with the USENET system reduces from the current traffic of over 40 MB a day to barely 128 KB. This however proves to be too efficient, and one begins to perhaps even miss the low SNR of the USENET environment **:-).** Also, we observed an increasing isolation and disconnection with new threads developing on the network.

Our solution to this problem is to "fuzz" the feed by deliberately lowering the SNR. We are currently experimenting with one technique to achieve this effect in a useful manner. We notice that the areas of interest of individuals active on the net tend to correlate very well. With this observation, our current experiment picks out articles on any topic written by, or referencing the FQDN email address of those Usenet contributors that the user tends to read the most. This exploits the particular statistical properties of group interaction that is unique to the Usenet context.

V. Discussion

V.A. Spoofing vs. URL editing

Spoofing of Postel Protocols has the advantage of not needing any modification to the clients, which especially in the case of the PCs, turn out to be commercial software for which source code is unavailable. These newer clients, mostly GUI based, tend to exhibit very consistent patterns of APDU usage, which are simple start-end transactions and not complicated trees of possibilities, which improves the hit-rate to usable amounts.

Also it is very transparent to the user, who may not be a very computer literate person. A URL communicated via a personal mail would still work and benefit from the cache. Of course, the cache may be not only demand-filled, but also filled by routine updates of objects known to be popular in the wide internet and CD-ROM distributions of mirrors.

On the flip side, it does have its disadvantages. It is conceivable that a distinct policy module might be needed for different services to age and manage the cache differently - a simple LRU among all the files cached which we are now trying out, may or may not be adequate. Also these policy modules might need to be keyed by site names and even object identifiers, so that dynamic information (such as basketball scores)[17] are aged rapidly when the events are in progress.

We conclude with an observation that this approach should serve the purposes of one subject of our experiment very well - a team of professional software developers and casual Web surfers in India, with a dial-up connection to the Internet tunneled via public packet networks. They have a current active interest in anonymous ftp repositories for freely available software and for general browsing on the Web and Internet Gopher, besides intensive usage of Usenet and E-Mail.

V.B. A Hierarchy of Web Servers

However a far more elegant solution in the long term would be to adopt a hierarchy of servers, not unlike the DNS and X.500 DSAs, for all these applications. Client programs should always contact their mirror or local cache server first. The local server should transparently accept connections to URLs that point elsewhere, and look in its cache for those objects. In the event of failure, a chain can be performed (so the object can be cached this time) or a reference can be passed back to the client.

We hardly need to draw more attention to this as a good design idea than to point out that popular WWW Web Home Pages are being overwhelmed by a fast growing load of connections to them at any given time. Also, some flags about cacheability or and explicit aging policy of individual documents need to be added to the HTTP GET result, so that cache management gets as close to the semantics desired by the author of an HTML object.

VI. Conclusions: Who benefits from this?

Connectivity to the Internet via dial-up or other low-bandwidth demand-based links has value not only to users in developing countries but also to large certain significant niches in the developed west as well. In the methods we have discussed, we have attempted to address the needs of both these user communities.

	By packets	By bytes
Total Internet traffic	100 %	100 %
Typ. Thinternet use	68 %	74 %
Methods apply to	50 %	66 %
Thinternet coverage	73 %	89 %

Table 2. Coverage by service.

E-Mail is already well-addressed by various store forward mechanisms, including proxy receipts with the help of DNS/MX records. We have augmented this coverage with the application of the methods we discuss in this paper.

As is evident in Table 2, which is based on current Internet usage patterns, our methods address about 74% (by packets) and 90% (by volume) of the traffic which we can expect a Thinternet user to generate. Sustained growth on Web traffic at the current rate would only boost these numbers favorably, since Web like transactions are particularly amenable to our methods.

We also make note of the fact that networks such as SITA, SWIFT and SkyTel[21, 22] are strategically viewing dial-up as one significant way to increase their reach in several market areas that are now beginning to open up. News and information dissemination and retrieval services are also considering such methods as options to on-line access. Some of the methods we discuss and some lessons learned from experiments such as these might also be of use to these applications.

VII. References

References

1. Win Treese, “Internet Index,” *EDUPAGE,* EDUCOM (February 7, 1994).
2. PSI, inc., “Continental Cable Press Release” (August 24, 1993).
3. Editorial, *Data Communication,* p. 31 (March 1994).
4. T. H. Choudary, “Dialup in India: Needs, Demand, and Regulation.,” *INET’94* (June 1994).
5. Bennett Kobb, “Personal Wireless,” *IEEE Spectrum,* p. 20 (June 1993).
6. Routing Support for IP Mobile Hosts, IP Mobility Working Group, *Internet Engineering Task Force Drafts* (October 1993).
7. T.J. Berners-Lee, R. Cailliau, N. Pellow, A. Secret, “The World Wide Web Initiative,” *INET ’93 Proceedings,* CERN, San Francisco (1993.).
8. Peter B. Danzig, Richard S. Hall, Michael F. Schwartz, “A Case for Caching File Objects Inside Internetworks,” University of Colorado at Boulder (March 1993).
9. Jacob Palme, Jussi Karlgren, Daniel Pargman, “Issues When Designing Filters in Messaging Systems,” *INET’94* (June 1994).
10. Michael F. Schwartz, Alan Emtage, Brewster Kahle, B. Clifford Neuman, “A Comparison of Internet Resource Discovery Approaches,” *Computing Systems* (August 1992).
11. Tom Lane, Personal communication (April 3, 1994).
12. Stev Knowles, “IPng - A vendor’s perspective,” *INET ’94* (June 1994). A look at the next generation of IP.
13. ITU-TSS, “X.500 Directory Services” (1988).
14. Merit Inc., NSFNET monthly usage statistics, *ftp://ftp.merit.edu:statistics/nsfnet.* This directory contains various traffic statistics.
15. A. Bhushan, “RFC-114, File Transfer Protocol”

(April 10 1971).

16. J. Postel, S. Sluizer, “RFC-772, Mail Transfer Protocol” (September 1980).

17. Eric Richard, Sport information server, *http://www.mit.edu:8001/services/sis/sports.html.* This URL carries current basketball scores, etc.

18. Tim Berners-Lee, “Universal Resource Identifiers in WWW,” *Internet Engineering Task Force Drafts* (March 12, 1994).

19. Finke, Ronald A., “Principles of mental imagery,” MIT Press, 1989., Cambridge, Mass..

20. Stanford Netnews Server, Usenet posting, *http://woodstock.stanford.edu:2000* (February 1994). To access, send mail to netnews@DB.Stanford.edu.

21. David Knight, Personal communication (March 1994).

22. Jean Mourain, Personal communication (April 1994).

Acknowledgements

One of us, (H.Shrikumar), would like to thank the support provided by Sunil Kejariwal, Amit Shah, Ashwin Maru, Vishaka Rao and Makarand Kulkarni, all at Temporal Systems, Bombay, India. Their enthusiasm in participating in these experiments, and being early guinea pigs for many of these ideas has been one inspiring factor to get the Internet closer to them, despite all odds.

Author Information

H.Shrikumar is the President, Temporal Systems, India, and has been involved in planning and architecting the network and business plan of one of the first private value added network service providers in India. This work has also led to the the development of a X.400 based Message Handling System positioned to survive in the difficult telecom scenario like the one in India, with unique implementation features that were crafted to work around both technical and regulatory pitfalls. Earlier, he was involved in the setting of India’s academic Internet, ERNET. He received a BE(Hons) EEE, from BITS, Pilani, India in 1987.

Currently Shrikumar is pursuing research in the area of monitoring and fault tolerance for Predictable hard Real-Time systems, at the University of Massachusetts, Amherst MA, USA.

Rehmi Post is a student of Physics and Computer Science. He has written systems which use PostScript as a dynamic protocol to coordinate distributed computation, including the c2ps C-to-PostScript compiler and an extensible interpreter which makes libraries network-accessible. Earlier he worked with FTP Software, a major vendor of PC networking software, and worked extensively on RFC-1001 NetBIOS.

His current research interests are in collaborative programming environments and computational physics. His current work in physics involves computer-generated diffractive optical elements, particularly as they relate to computer networks.