Defining High-Speed Protocols: Five Challenges and an Example that Survives the Challenges

Joseph D. Touch, *Member, IEEE*

Abstract—**The First IEEE Gigabit Networking (GBN) Workshop defined a set of characteristics of "interesting" high-speed applications. The GBN criteria ensure that the application addresses a significant problem, and that it actually requires a gigabit network. This paper presents five challenges that augment the GBN criteria. These challenges ask whether gigabit applications require new research into different protocols, or can be supported by existing protocols that merely run faster.**

I. Introduction

At the First IEEE Gigabit Networking Workshop (GBN), held in Toronto, Ontario, Canada, prior to IEEE Infocom '94, a number of gigabit applications were presented [10]. The GBN submission criteria for "gigabit applications" were defined in the call-for-papers by a list of characteristics that ensures that significant user bases exist, and that a gigabit network is required.

During the past several years the networking research community has been considering the problem of gigabit protocols, especially how they differ from their slower counterparts [9]. There are two primary issues—increased speed or performance of existing protocols, and domains where existing protocols may not suffice. This paper characterizes the latter by a list of challenges developed to complement the GBN criteria.

This paper first summarizes the GBN criteria and justifies the need for additional challenges. The challenges are then presented. Finally, as a challenge unanswered is not entirely useful, an application is presented that survives these challenges. This is done to open the door to broader consideration of some unconventional trade-offs that use bandwidth as a resource[1] rather than as a constraint.

A. The GBN Criteria—A Review

The GBN criteria were described in its call-for-papers [10]. They are designed to ensure a significant user base, and that a gigabit network is required. They are:

1) *Realistic consumer or business application (current or future),*
2) *Minimum bandwidth per user of many megabits per second,*
3) *Minimum potential base of thousands of simultaneous users,*
4) *Number of users* $\times$ *application bandwidth in excess of 1 Tb/s,* and
5) *Consumer video applications must be more sophisticated than broadcast or simple video-on-demand multicast.*

Criteria 1 and 3 ensure that the application addresses a significant user population. Although there are many interesting applications in telemedicine, distributed simulation, and parallel computation that require gigabits, they address too small a user community. An exception would be tele-*stuff*, of which telemedicine is one example, where the general class of applications may have a substantial user base as an aggregate.

Criteria 2, 3, and 4 ensure that gigabit bandwidth is required, and is not a result of aggregation of users or groups of independent applications.

The first four criteria ensure that the application addresses substantial user communities and requires gigabit bandwidth. There are many applications that satisfy these criteria but are considered "uninteresting," because they can already be implemented with existing protocols. Criterion 5 is an attempt by GBN to filter some of these out.

The GBN criteria are conditions where gigibits are useful, but not where they are necessary. Criterion 5 does not sufficiently exclude classes of gigabit applications for which solutions already exist. This is why this paper augments these criteria with further challenges.

B. The Killer Application and Killer Protocol

The goal of this discussion is to refine the GBN criteria to exclude cases where existing protocols suffice. The result is to define a set of criteria that require new protocols to use gigabit networking for real applications.

The application that exhibits this goal is the World Wide Web (WWW) client server system [2]. It exhibits gigabit requirements when real-time interactive constraints are added [13].

The WWW is emerging as a dominant (and thus realistic) consumer and business application [2]. Although originally developed as an interface to Internet navigation of file transfer client/server systems, its current use is evolving toward a distributed interactive application. As such, the requirements

Manuscript received June 12, 1994; revised January 17, 1995. This work was supported in part by the Advanced Research Projects Agency through Fort Huachuca Contract DABT63-91-C-0001. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Department of the Army, the Advanced Research Projects Agency, or the U.S. Government. This paper was presented in part at the First IEEE Gigabit Networking Workshop, Toronto, Ontario, Canada, May 1994.

The author is with the USC/Information Science Institute, Marina del Rey, CA 90292 USA.

IEEE Log Number 9410352.

[1] A resource that, like any other resource, is subject only to the arguably external constraint of cost.

on response time are narrowing. As users begin to expect real-time interactive response, they demand response times in the 100–200 ms range.[2]

Whereas its minimum user bandwidth is not many Mb/s for conventional client/server operation, it can be for interactive use. Consider a 100 ms transmission and switching latency. That leaves around 50 ms for request creation, server parsing, server retrieval, and server response. Assuming that the other components consume a minimal 10 ms, only about 40 ms remain for file transmission itself.

The typical WWW HTML (hypertext-only) document is only around 6 kbytes, but small images can be around 60 kbytes. Transmitting a 60-kbyte file in 40 ms requires 12 Mb/s. As the file sizes increase to 200 kbytes (for larger still images), the bandwidth increases to 40 Mb/s. Whereas the minimum bandwidth to the user is not many megabits per second yet, as the demand for still images and short video clips increases, so too do file sizes. As other latencies remain constant, the bandwidth requirements increase as a result. And the expectation of interactive response time is growing. This is an issue that "rides the technology curve."

The minimum potential base of WWW is easily millions of simultaneous users, and the aggregate bandwidth could be in excess of 1 Tb/s as well, satisfying the other GBN criteria.

C. Other Considerations

Consider the case where a 300 ms satellite hop is in the path. In this case, the response time of a direct request is longer than the acceptable limit.

There are an average of 10–20 hypertext links/page and 6 kbyte/file for Web pages (measured). If the user spends 20 s reading each page, the entire contents of every possible "next page" can be sent in 20 links * 6 kbyte/link/20 s = 50 kb/s. If the user scrolls through at 1 page/s, that increases to 1 Mb/s.

These are modest bandwidth requirements, but they increase where the Web pages have images, averaging 60 kbytes/page, or where the pages have embedded "IMG" icons, PostScript (100 kbytes average), or executable binaries (200 kbytes average, at NCSA). Even higher bandwidth is required for video clips (1 Mbyte), around 160 Mb/s. When a user begins poking around video archives, the interaction speed increases to 0.2 s, resulting in a single-user bandwidth requirement of 800 Mb/s.

Further, users currently pay for maximum bandwidth, but not per-packet. This should continue (the PTT's use this as a design criteria). Unused bandwidth is wasted—there is not necessarily a need to charge for extra bandwidth. Use of this bandwidth to reduce the user-perceived latency is a win.

These WWW modifications are also useful in low bandwidth environments, where the channel is idle in-between requests. The common characteristic is that of surplus bandwidth-delay product. In one case, the product is "vertical"—the increase in BW results in a "tall" pipe that can not be filled. In the other case, the product is "horizontal"—idle periods form longitudinal gaps in the pipe that are not filled. The solutions proposed here enable interactive WWW applications that require surplus bandwidth-delay product, regardless of which type.

[2]The 100–200 ms number is well-established in the human-factors and user-interface community, although the specific value is debated.

II. Primary Issues

There are two primary issues—that the protocol does not run fast enough, and that the bandwidth-delay product pipe is not full. Each has several ways of being dealt with, ways that determine the additional challenges.

A. Protocol Does Not Run Fast Enough

If the protocol does not run fast, it must be speeded up. Basically, the data path is too slow, the control path is too slow, or the latency is too large for feedback control.

1) Data Path Is Not Fast: If the data path is not fast, make it faster. Increase the clock rate of the data, or parallelize the data path.

2) Control Path Is Not Fast: If the control path is not fast enough, reduce the amount of control required. With a higher transmission rate, this results in the same amount of control for a larger amount of data [14]. Slowing the control down is accomplished by making the data packets larger (e.g., NetBLT or UltraNet), or transmitting multiple packets per control packet. Slower control also requires more assumptions about the stability of state in-between control messages, to ensure stability in the absence of as many control messages [14].

3) End-to-End Latency Is Too Large: In this case, the data can be sent quickly and the control computed quickly, but the end-to-end latency is causing the interaction loop to be too large. One solution is to relocate everything, i.e., to get rid of the need to communicate in the first place. Caching is one way to relocate, and another is to circulate the data [1], [4].

B. The Pipe Is Not Kept Full

Assuming speed exists, a method is needed that keeps the bandwidth-delay product pipe full to achieve high throughput. If the bandwidth-delay product is not larger than in a current WAN, current protocols already suffice. That means a gigabit LAN is fine, but a MAN or LAN will not work, because speed is not the issue—keeping the pipe full is. There are two main problems—either the data pipelining mechanism fails, or it runs out of pipeline data.

1) The Pipe Is Empty Because the Window Is Small: In this case, there is a flaw in the implementation. There is nothing about sliding-windows protocols that necessitates a particular window size or window granularity, only the implementation has these properties. Increase the number of bits for the window sequence or "count" over larger window components [5]–[7].

2) Even with Large Windows, There Is Not Enough Stuff: Even if the windowing mechanism allows large amounts of pipelining, there may not be enough data with which to fill the pipe. A gigabit WAN has 30–100 Mb in the pipe–4–12 Mbytes. That is more RAM than many systems have, and certainly larger than most messages an application has to transmit.

One solution is to use multiplexing to share the channel among user processes [8]. This is a parallel of process-

swapping in OS—when one process runs out of data to send, another is activated. This works, provided that the process activation is deterministic, i.e., that the other side of the channel knows what the process activation order is [11]. This is also tantamount to not having a gigabit application protocol—each application does not use a gigabit channel, so bandwidth needs to be aggregated over a set of multiplexed applications.

The case where the application order is not predictable is more interesting, as will be discussed later.

III. Five Challenges

There are five challenges for gigabit protocols. The main question is probably:

- *Why are these challenges challenging?*

They distinguish between incremental performance increases and places where current kinds of protocols are not useful, regardless of how well-engineered.

The questions this paper addresses are:

- *Do gigabit networks NEED new protocols?*
- *Are any new protocols really NEW?*

One reason gigabit systems have low performance is that applications "run out of stuff to fill the pipe with." Some believe this problem will disappear, but other investigations indicate it will not [11], [12]. The transient environment has already been observed—when files were much larger than the bandwidth-delay product. The current ratios are not transient, but will continue. It is now recognized that bandwidth needs to keep pace with processing and storage evolution.

A. Challenge #1—Increase the Clock Rate

One easy way to increase the speed of a protocol is to increase the clock rate of the processor, interface, and transmission lines. If the protocol is slow because you are using CMOS processors, it might work fine in ECL technology.

There are a few issues here. First, fast protocols require fast operating systems, fast interfaces, fast transmission lines, fast memory, fast disks, and fast everything else. If the problem is completely solved by speeding up the clock rate of the system, then this is not a protocol issue. It is a clock rate issue.

Also consider the fact that fast TCP code relies on precompiled branch prediction (so-called "fast-path"), a very old and well-known optimization technique. Some of the other protocol stack optimizations are just recognition that general interfaces are slow, and specific interfaces can be optimized, also a well-known trade-off. None of these change the protocol—they are implementation enhancements.

There is a need to distinguish between protocol issues and issues of overall speed. If the protocol itself is not keeping up with the speeds of the rest of the system, then protocol issues are indeed involved. Parallelization addresses this case, but does not always help [14].

In general, data path parallelization works well, but control path parallelization is, by definition, poor. Control paths can not be factored efficiently without synchronization between components, which adds overhead that defeats the parallelization.

Data path parallelization is a way to speed the "clock rate" of the data. Control parallelization works where packets are unrelated—e.g., UDP, but not for TCP-like protocols [14]. In the latter case, regardless of partitioning (per packet, per function, etc.), the parallelism is limited to about five processors per connection.

The real issue is that of "protocol relativity" [12]. A protocol does not know the clock rate—only the number of bits in transit between components. Protocol speedup is a control and feedback issue, sensitive to the bandwidth-delay product only.

B. Challenge #2—Multiplex (Deterministic)

Multiplexing has been proposed as a solution to the "do not have enough stuff to send" issue, as mentioned before [8]. This is equivalent to not having a gigabit application—the gigabits are shared among a set of applications on a workstation. Using multiple hosts, users, or processes are all ways of providing aggregate gigabits only.

In the deterministic multiplexer case, this avoids the domain examined here [11]. In the nondeterministic case, the problem has just moved down a level in the protocol stack. In the original case, there was not enough data to send, primarily because data was sent to be sufficient for the current state of the other end of the channel, and not any possible subsequent states. If there were enough for subsequent states, that data would have been sent too, and so on, increasing the amount of data available to send *ad infinitum*. This is how sliding windows works—by predicting subsequent states, in a linear manner.

If the subsequent state is not predictable, neither is the subsequent data [12]. It does not matter whether it is the application state, or the multiplexer state. Nondeterministic multiplexing moves the state prediction problem to the multiplexer-synchronization level, i.e., lower in the protocol stack.

C. Challenge #3—Use Large Payloads

Using large payloads is another way to shut off the protocol, and increase the effective speed of the control protocol [13]. Versions of TCP run at 1 Gb/s by using 64-kbyte packets, i.e., by this technique (e.g., UltraNet).

Using large payloads slows down the control protocol. Header frequency determines the rate of the protocol. The ratio of header to payload determines the stability of the protocol [14].

Large payloads are also an attempt to amortize the cost of context switches. Increased payload transfers between the host and network reduce the effective overhead of the transfer setup costs. This is an attempt to overcome an existing problem in the host—if context switches are this costly, making network I/O faster is the least of the worries.

D. Challenge #4—Increase Window Size

As mentioned before, increasing the window size increases the amount of information an implementation of a sliding-

windows protocol can pipeline. This helps fill the pipe only if the application has sufficient data to supply. It addresses an implementation deficiency only. The main difficulties are backward compatibility and acquiring the consensus of standards bodies [5]–[7].

The window size parameter is also an example of a compile- or run-time parameter that is unfortunately treated as a specification constant. There are several such parameters—maximum protocol data unit, timeout values, window granularity and range, etc. The constancy of these parameters is a limitation of implementations only.

E. Challenge #5—Relocate Everything

If the data can be copied or cached, i.e., if it is stable enough that there is no need for separating it from the application, then a copy can be put near the application via a low-speed channel to avoid the need for high-speed communication altogether. A protocol is not needed if there is no communication, or more precisely, if there is no feedback of state between two separated entities.

Another way to relocate data is to circulate it among the nodes of a protocol [4]. Variants of this protocol rely on predictive behavior of data reuse to govern caching [1]. These are useful techniques, but are protocol extrapolations of previously known methods.

IV. WWW Interactive Applications

Although it is useful to eliminate domains where gigabit protocols are not needed, it begs the question of where they are.

One application that requires a gigabit protocol is an interactive client/server system with real-time response. Several years ago when the problem of latency and high speed protocols was analyzed, the conditions were specified under which bandwidth and bandwidth-delay product could be used to compensate for latency. The goal is to reduce the perceived latency, to give the illusion of low latency. This work began as "Mirage" (a model) and continues as "Parallel Communication" (a protocol based on Mirage) [12].

Latency compensation is possible using source-based anticipation (presending). The composition of two presending channels (back-to-back) is the more common receiver-based anticipation, i.e., prefetching. Presending differs because it is source-based.

There are several advantages of presending over prefetching. Presending distributes the computational effort between source and receiver. It also avoids unnecessary prefetch messages from the receiver, allowing better use of asymmetric communications channels (e.g., satellite, cable-TV, or high-speed digital telephone with dial-up feedback).

This solution requires knowledge of the state space evolution of the other end of the channel, where the state evolution has moderately-constrained branching properties. The domain was described where source-anticipation would help, specifically distributed hypermedia navigation [12], [13].

This describes the WWW, used as a real-time interactive distributed system [2]. The WWW browsers are currently used as client/server interfaces, where response time tolerance is high. Casual users have come to begin to expect a level of real-time interaction that does not match the client/server design of the system.

In addition, HTML (the document notation language of the WWW) has come to be an effective high-level application language. Systems use WWW to drive bulletin-board services, interactive query systems, on-line forms systems, etc. This further drives the expectation of interactive response time from these WWW interactive applications (WWWia's).

V. Defining Characteristics

This section lists a set of characteristics that helps define applications that are capable of using gigabit networks and keeping the "pipe" full of data sufficient to reduce perceived latency.

A. Char. #1—Requires Feedback

Non-interactive applications, i.e., those that pipeline data to fill the bandwidth-delay product, can be accommodated with existing transport protocols. These include streaming data applications, such as digital audio or video, as used in teleconferencing.

WWWia's require feedback between the client and server. Even though the servers are stateless, they keep soft-state that helps govern source-based anticipation.

Although there are caching proxies for WWW servers, they will not help for the first-use of documents. If the response time is very large, even for some small percentage of the time, the interactive nature of the WWWias will be defeated. Also, the WWW drives the interaction toward first-use, because the clients themselves have caches.

B. Char. #2—"Nonlinear" Communication

The feedback needs to be nondeterministic. Otherwise simple pipelining again works fine, as in the case of sending a very large file or database in total [1], [4].

WWWia's have a branching control structure with recursion, as indicated by the URL links and the "history" of the browser (user interface).

Large windows or packets help only during the transmission of a branch item. The branching structure cannot be accommodated by current sliding-windows protocols, and inhibits use of large linear windows or large packets [12].

The combination of feedback and nonlinear communication defines a rich control structure. It is this structure that the source uses to guide its presending. Making the data chunks larger reduces the richness of the control. WWWia's are reasonably rich, because the branching of the control is reasonably large (7–10 links/page), even though the data chunk is small (6 kbytes for HTML text).

C. Char. #3—"Well-Defined" App.-App. BW

This characteristic helps determine that the data can not be moved, and that the distributed application has not been broken in a particularly bad place (for high bandwidth) with

no other justification. The system should not require dissection or detailed constraints to evidence the application-application bandwidth. WWWia's are well-defined—the server is one side, the browser is the other.

VI. An Example that Survives the Challenges

This section explains how WWWia survives the challenges (and exhibits the characteristics). Specifically, it describes WWWia's modified with server-based preloading of the browser cache [13].

A. Survival

The WWWia's survive the five challenges as follows:

1) *Increase the clock rate:* Current WWW would just get each hypertext page faster. It would still take a 100–400 ms latency hit each time you clicked on an item, defeating its interactive nature.
2) *Use deterministic multiplexing:* This effectively reduces the per-user bandwidth, increasing the user-perceived latency. As noted before, it reduces the per-application bandwidth below gigabits.
3) *Use large payloads:* Even if each WWW page is one packet, performance is not helped in the case where propagation latency is larger than 200 ms. In the case where propagation latency is smaller, the bandwidth required for a direct response is determined by the amount of time left (200 ms prop. latency), and the size of the response (4 kbytes, 60 kbytes, 200 kbytes, etc.).
4) *Speed TCP code/increase TCP windows:* Same as #3—WWW files are too small to matter with even existing TCP window size.
5) *Requires feedback:* WWW is an "interactive" system—more so as it evolves.

B. Exhibits Characteristics

The WWWia's also exhibit the three main characteristics:

1) *Requires feedback:* WWWia's require feedback—files from the server, and "next request" from the browser.
2) *"Nonlinear" communication:* WWWia's have a branching set of possible next requests from the browser. The stream of requests is nonlinear.

   The control is reasonably rich with respect to the packet stream. The data chunks are large (30–60 kbytes), but the control is much richer than "buffer empty/full" as in current protocols—it specifies a unique file on the server.
3) *"Well-defined" app.-app. bandwidth:* Server to browser, browser to server.

VII. WWWia's Architecture

The design for a WWWia architecture augments the existing WWW client/server with a presending *pump* and browser *filter* (Figs. 1 and 2) [13]. The pump and filter are supported by either the existing transport protocols or their more recent extensions [3], or by an augmented transport protocol [12]. The

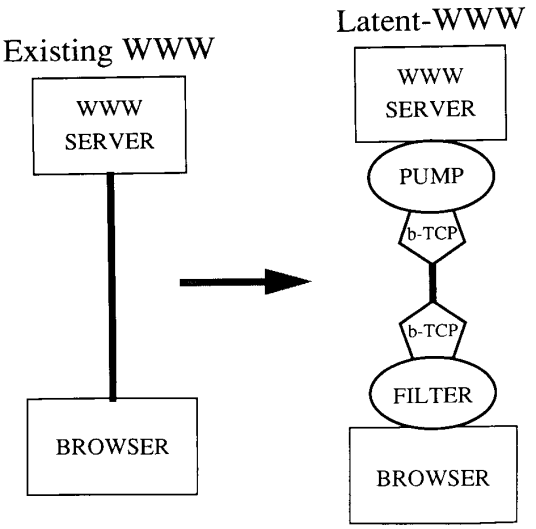

Fig. 1. Implementation of the WWW intermediaries called the *pump* and *filter*.

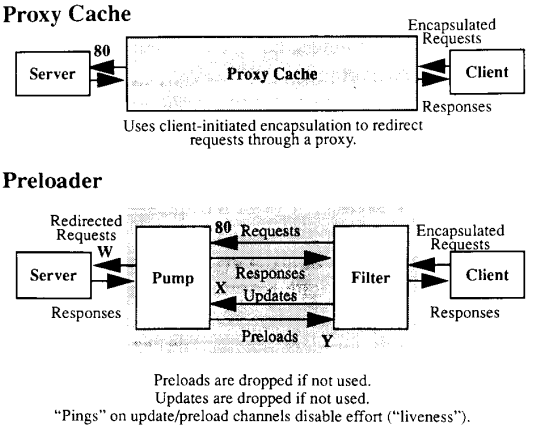

Fig. 2. Design of the pump and filter appears to the server and client as if it were a proxy cache.

pump and filter implement the Web-equivalent of the Parallel Communication protocol [12].

The pump acts as a proxy for the browser at the server. It keeps soft state indicating the last request received from the browser, and peeks into the data stream to find URL's embedded in replies from the server. The pump then makes requests for URL's on the same server to be forwarded to the filter. The pump and filter together appear as a proxy cache to the client and server (Fig. 2). The protocol is outlined in Figs. 3 and 4.

The pump permits two kinds of HTML replies to be sent to the browser—direct replies, and present replies. The present replies are tagged to be saved on the disk by the filter. In a high bandwidth-delay product network, these tags may not be necessary, because the present documents arrive just as they are needed at the browser. The most disk space required is the larger of the bandwidth-delay product and the bandwidth-"idle-

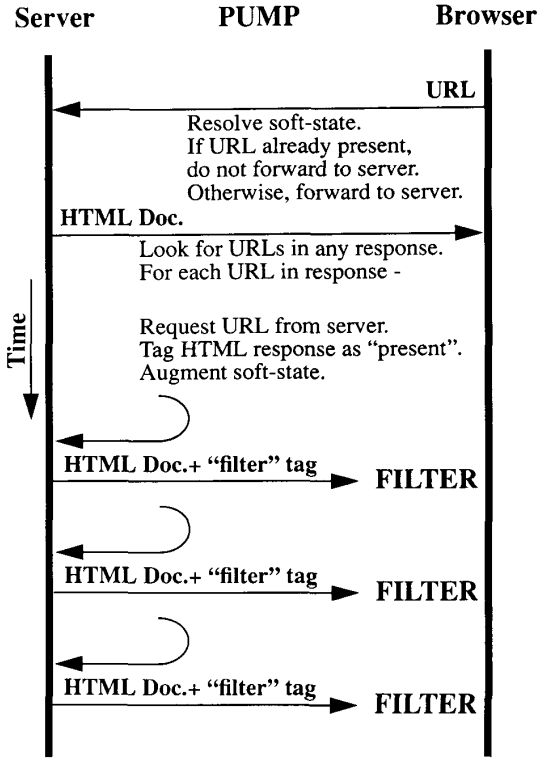

Fig. 3. Pump operation.

time" product. If there is some upper-bound on reasonable disk usage for the filter to cache present data, that can be indicated to the pump, to avoid wasted effort.

The filter stores forwarded server replies to the disk. It also intercepts URL requests from the browser. If the URL is on the disk, the filter responds with the request and forwards the URL to the pump (not to be forwarded to the server). If the URL is not on the disk, the request is sent to the pump to be forwarded to the server (Fig. 3).

Note that in either case, the URL is sent to the pump. This provides feedback to the pump. In the case where the file has not yet been sent, it indicates a corrective action to the pump. In the case where the pump has already sent the file to the disk, it indicates which file was used, and permits the pump to focus further preloading.

The branching-TCP extensions support the tags indicated in the figures and provide application-layer signalling of excess bandwidth that can be used for latency reduction.

The pump manages the sending of all possible next requests, and manages the possible states of the client. The pump uses the server-side TCP signal of excess bandwidth to initiate presending, and the branching window allows the pump to send alternate streams of messages to the client. As the pump emits these messages, the branching in the server-side TCP increases.

The filter allows the browser to receive only those messages that correspond to a particular state. This client-side TCP also indicates branch selections to the server-side TCP, to perform state resolution.

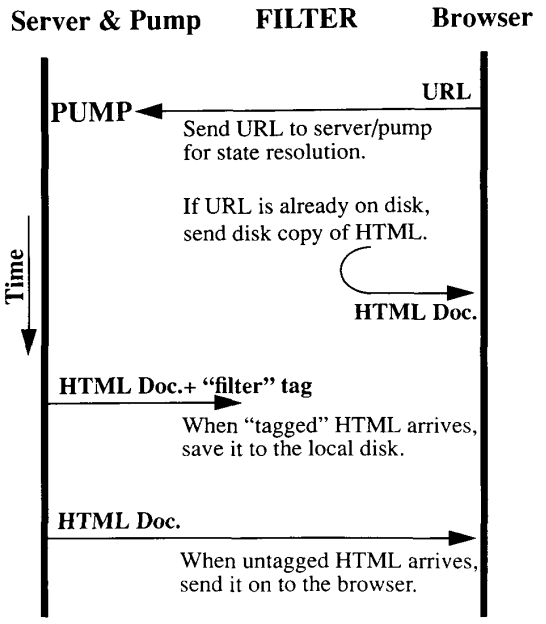

Fig. 4. Filter operation.

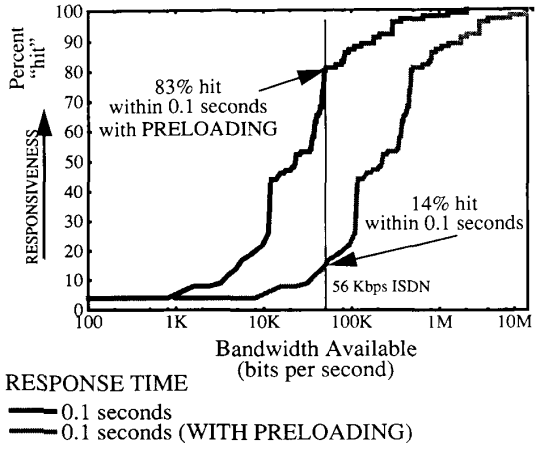

Fig. 5. Response time (probability of a 0.1 s response) as bandwidth increases.

VIII. Observations

Some measurements have been taken to indicate the effectiveness of this mechanism. These measurements were performed on existing Web servers, so reflect current Web design, which revolves around formatted text (average page size of 6 kbytes). As available bandwidth increases, servers are expected to more fully utilize embedded icons, images, and video clips, increasing the required bandwidth by a factor of 100.

One observation is that current Web cannot be supported interactively by ISDN lines (14% hit rate within 0.1 s). By augmenting the protocol to support server preloading of receiver caches, the same bandwidth can support 0.1 s response 83% of the time (Fig. 5).

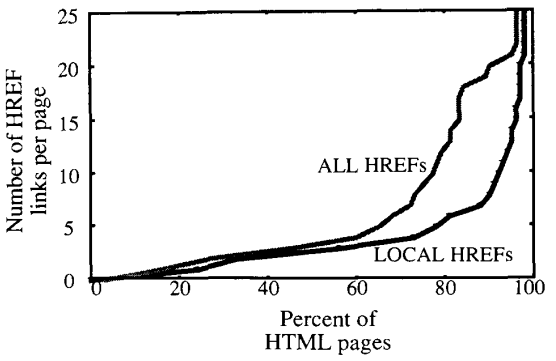

Fig. 6. Number of hypertext links (HREF's) per page in the web (local only, and all).

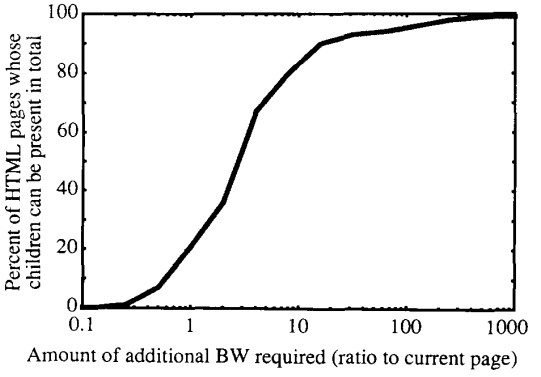

Fig. 7. Amount of additional bandwidth required (relative to the current page).

The bandwidth required for source preloading of receiver caches in the Web has also been measured. Specifically, number of links per page (Fig. 6) and the amount of bandwidth for general preloading (Fig. 7) were measured. The links per page is measured both in general terms, as well as local to the server (where preloading is possible). The bandwidth is a comparison of the bytes per page versus the total bytes required for the files pointed to by the links on that page.

A. Performance

The performance of this mechanism can be evaluated relative to several metrics—channel utilization, effective bandwidth, effective latency, and overall cost. The goal of this mechanism is reduced latency, and it assumes an acceptable increase in bandwidth used.

The channel utilization can be measured, where the goal is a load of 100%. Conventional request/response systems achieve loads near 50%, because the response channel is idle in-between and during requests. The goal is to keep the server-to-client channel busy 100% of the time.

The bandwidth of the messages that are actually received (effective bandwidth) can also be measured. This will always be at least as large as the effective bandwidth of a request/response system, because guessed messages are not counted, and because a direct request always overrides this protocol.

Similarly, the effective latency is always reduced relative to a conventional protocol. Responses that are anticipated reduce the measured latency, and responses not anticipated cost the same as in the conventional case.

The overall cost is difficult to measure without externally imposed network cost functions. The cost can be expressed in terms of the bandwidth used, but it is of little meaning due to the number of variables. The real result is that a set of conditions must exist.

- bandwidth must be available in excess of that required by the conventional protocol
- the expense of the excess bandwidth must be acceptable, i.e.:
  - —external cost is perceived acceptable
  - —latency reduction is not feasible by any other method, so any cost is acceptable.

B. Bandwidth Requirements

Available bandwidth implies a high peak-rate allocation in guaranteed-bandwidth systems, or that the server pump subjects itself to feedback from a rate- or burst-limiting mechanism, and avoids preloading that violates the rate or burstiness guarantees. Server preload messages should be tagged as "droppable available-bit-rate" traffic. In this way, bandwidth in excess allocation can be used when available, and shared among preloading sources. "Droppable" ABR traffic assumes a mechanism that provides bandwidth and latency performance to untagged traffic equivalent to the case where no droppable ABR traffic exists, i.e., a preemptive packet scheduler.

C. Other Requirements

The feasibility of this mechanism also implies the availability of sufficient cache storage at the receiver and server capability. The amount of cache storage required is one bandwidth-delay product, where delay is the time between user requests, due to either round trip latency or idle user activity. The server must also be able to supply anticipatory information at the channel bandwidth; if the server is already loaded, or if its internal bandwidth is the bottleneck, performance will be compromised. Note that in the case where cache space is limited, or where the server is loaded or has insufficient bandwidth, the mechanism degenerates to its existing performance with conventional protocols.

This discussion also assumes the availability of sufficient information (i.e., hypermedia links) to support server-based preloading. There need not be a correlation between users (ensemble) or a repeated history of a single user's actions (temporal); the requested item need only be from among the links on a page, rather than overridden by typing in an arbitrary URL. The URL's within links on a page are information the server can use to optimize the response latency; arbitrary URL's are (by nature) unpredictable, and will require a conventional client/server interaction (and its associated latency).

IX. Conclusion

This paper has discussed five challenges for gigabit applications that indicate where existing protocols may not work, and where new protocols are required. It has shown a class of applications—interactive distributed multimedia, namely interactive real-time WWW access—that survive the challenges. It has also shown how source presending is a way to use excess bandwidth-delay product to reduce the browser response time, and how this is one example of a truly gigabit protocol.

Acknowledgment

This paper is the result of discussions with and feedback of J. Sterbenz of GTE Labs, Waltham, MA, J. Postel and S. Hotz of USC/ISI, Marina del Rey, CA, G. Woodruff at the University of Toronto, Toronto, Ontario, Canada, and S. Banerjee at University of Pittsburgh, Pittsburgh, PA, as well as the other participants of the IEEE Gigabit Networking Workshop held in Toronto, Ontario, Canada, May 1994.

References

[1] S. Banerjee, V. O. Li, and C. Wang, "Distributed database systems in high-speed wide-area networks," *IEEE J. Select. Areas Commun.*, vol. 11, no. 4, May 1993, pp. 617–630.

[2] T. J. Berners-Lee, R. Cailliau, J.-F. Groff, and B. Pollermann, "World-wide web: The information universe," *Electronic Networking: Research, Applications and Policy*. Westport, CT: Meckler, pp. 52–58.

[3] R. Braden, "Extending TCP for transactions—Concepts," RFC-1379, USC/Inform. Sci. Inst., Nov. 1992.

[4] G. Herman, G. Gopal, K. Lee, and A. Weinrib, "The data-cycle architecture for very high throughput database systems," in *Proc. ACM SIGMOD Conf.*, 1987, pp. 97–103.

[5] V. Jacobson and R. Braden, "TCP extensions for long-delay paths," RFC-1072, LBL and USC/Inform. Sci. Inst., Oct. 1988.

[6] V. Jacobson, R. Braden and L. Zhang, "TCP extensions for high-speed paths," RFC-1185, LBL and USC/Inform. Sci. Inst., Oct. 1990.

[7] V. Jacobson, R. Braden, and D. Borman, "TCP extensions for high performance," RFC-1323, LBL, USC/Inform. Sci. Inst., and Cray Res., May 1992.

[8] L. Kleinrock, "The latency/bandwidth tradeoff in gigabit networks," *IEEE Commun. Mag.*, vol. 30, no. 4, pp. 36–40, Apr. 1992.

[9] "Research priorities in networking and communications," NSF Rep. 92-109, Oct. 1992.

[10] J. Sterbenz *et al.*, *Gigabit Networking Workshop '94*, <http:// info.gte.com/ieee-tcgn/conference/gbn94>.

[11] J. D. Touch and D. Farber, "Reducing latency in communication," *IEEE Commun. Mag.*, vol. 31, no. 2, pp. 8–9, Feb. 1993.

[12] J. D. Touch, "Parallel communication," in *Proc. IEEE Infocom*, Mar. 1993, pp. 505–512.

[13] J. D. Touch and D. Farber, "An experiment in latency reduction," in *Proc. IEEE Infocom*, May 1994, pp. 175–181.

[14] J. D. Touch, "Protocol parallelization," *Protocols for High-Speed Networks III*. Amsterdam, The Netherlands: Elsevier, 1994.

Joseph D. Touch (S'83–M'92) received the B.S. (Hons.) degree in biophysics and computer science from the University of Scranton, Scranton, PA, in 1985, the M.S. degree from Cornell University, Ithaca, NY, in 1988, and the Ph.D. degree from the University of Pennsylvania, Philadelphia, in 1992, both in computer science.

He joined USC/Information Sciences Institute, Marina del Rey, in 1992, and is currently a Project Leader in the High Performance Computing and Communications Division there, directing the ATOMIC-2 and PC-ATOMIC tasks. He is also a Research Assistant Professor in the Department of Computer Science, University of Southern California, Los Angeles, where he teaches Advanced Operating Systems. Since 1988, he has been addressing issues of latency and source-anticipative protocols. In 1994, he received a U.S. patent for a device for latency-reducing processor-memory interface. He is also interested in issues of telecommuting and on-line city services and in response-time reducing extensions to the World-Wide Web.

Dr. Touch is a member of the program committees of IEEE Infocom '94 and '95, Protocols for High Speed Networks '94, and Physcomp '94. He is a member of Sigma Xi.