Reprinted from IEEE Infocom '94, Toronto Can., pp.175-181, June

An Experiment in Latency Reduction

Joseph D. Touch and David J. Farber

USC / Information Sciences Institute (touch@isi.edu)
Univ. of Pennsylvania (farber@cis.upenn.edu)

Abstract [1,2]

Communication latency can be reduced by increasing bandwidth via a sender-based anticipation technique called Parallel Communication. Here we apply this method to anonymous FTP. Our analysis of log files indicates that latency can be reduced to 2 round-trip times, as small as 0.6 round-trip time/file, for a 7x increase in bandwidth. This technique applies to up to 95% of the FTP traffic. This method is expected to be especially useful in reducing latency in automated FTP access, such as in the World-WideWeb.

1: Introduction

Current gigabit networking research addresses limitations of communication and computation "imposed by factors other than the finite speed of light" [10]. Communication limits computation via latency, and computation limits communication, especially regarding coding and compression [13]. There is a need for an "information theoretic theory of networks" [10]. Just as computation is analyzed at various levels of abstraction, from the Boolean/gate logic levels, through the abstract machine to algorithms, communication must go beyond the bit-level coding and communication theory, to an algorithmic abstraction.

To this end, we are addressing the 'finite' limits of communication latency. Whereas the speed of light remains a fundamental limit, propagation latency alone is not a limit to interaction latency in many cases. There are often ways to reduce the user-perceived communication latency to below that of speed-of-light propagation [13].

Network rate increases do not address the problem of propagation latency given effectively infinite bandwidth[3]. What will this available bandwidth be used for? What will be changed as a result? We propose using excess bandwidth to reduce propagation latency. This assumes that bandwidth is a plentiful resource and latency is not.

We are interested in considering bandwidth as a means, rather than an end. We want to investigate the real fundamental limit of communication - that of propagation latency, which is not currently being addressed [10].

1.1: Is this a problem?

From its inception in 1969 through 1988, available communication bandwidth remained constant in the Internet backbone as other computer resources grew. Bandwidth is resuming its own technology curve [13]. We notice that its rate of increase is challenging that of other computational resources, such as CPU rate, workstation memory, disk capacity, etc.

On a per-year basis, workstation sizes increase at a rate of 1.26x [12]. DRAMs increase at a well-established rate of 1.59x, equivalent to 4x in 3 years. Disk capacity increases at 1.26x, equivalent to 2x in 3 years [5]. Microprocessors power increases at 1.38x (2x in 2 years, sustained since 1979), and recently at up to 2x per year. The Internet communications rate has increased at 2.34x in '86-'90 (T-1 to T-3, 30x in 4 years), and at 1.28 in '90-'95 (T-3 to OC-3, 3.44x in 5 years), after a period of dormancy ('79-'86). The Internet is expected to increase at 1.52x in the '95-2000 period (OC-3 to OC-24, 8x in 5 years).

Figure 1 depicts these rates as slopes. Workstation system (system) and disk rates (disk) are identical. Microprocessors are shown both as their sustained rate of increase

1. This research was partially sponsored by the Advanced Research Projects Agency through Ft. Huachuca Contract No. 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.

2. This work is also based on research supported by the Information Science and Technology Office of the Advanced Research Projects Agency, under contract NAG-2-639, and by an AT&T Graduate Research Fellowship, grant #111349.

3. Specifically, we consider the limit of finite values of BW, as those values approach infinity.

(μP) and their peak rate (μP*); the latter includes architectural phases that cannot be expected to continue [5]. The Internet backbone rates are shown as a shaded area, where the recent sustained increases are shown darker. The figure illustrates that communication speed increases are expected to keep pace with microprocessors, but not memory, although the growth rates are too close to definitely call[1].

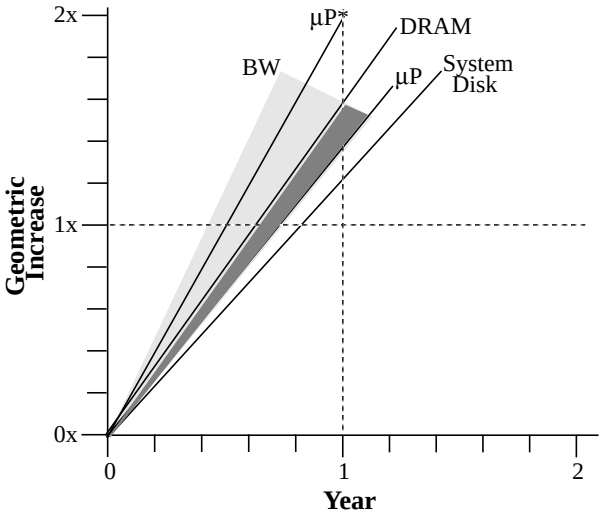

FIGURE 1. Comparative rates

1.2: Existing Approaches

Existing methods to accommodate bandwidth increases include high performance protocols (XTP [4], VMTP [3], Delta-t [14]), and methods to increase the window size of sliding-window mechanisms [7][8][9].

High performance protocols primarily address implementation issues. If protocols cannot be implemented effectively at high speeds, neither can user applications. However, if context switching is expensive for protocols, it is for user applications, too. In fact, many so-called 'protocol performance issues' reduce to workstation performance issues. This is to say that they are interesting and important, but not specifically communication issues. We have shown a sort of protocol 'relativity', that a protocol does not know the speed at which it operates, only the number of bits in transit between communicating entities [12]. As a result, protocols are sensitive to the bandwidth-delay product, not the raw bandwidth rate. A gigabit LAN is equivalent to a kilobit WAN, in that respect. There are significant implementation issues that accompany raw rate increases, but existing protocols suffice[2].

Methods to increase window sizes require that sizes grow as communication rates increase. Existing methods that address high bandwidth-delay product environments will not be adequate in the future. These methods are aimed at increasing the window size of sliding window transport protocols, such as TCP. Measurements of anonymous *ftp* access here at ISI have shown that the average file size transferred is 65KB, and 210KB at NCSA (a large *ftp* site with available log files). At rates below 1.5 Tbps in a LAN (100m), and 25Mbps in a WAN, the window size does not affect the transmission rate at ISI. The minimum rates are 500 Gbps LAN, and 8Mbps WAN for window size to affect NCSA transfers. As a result, we address gigabit WAN networks, because nothing new need be done in a LAN at the expected rates.

1.3: Anticipation

Anticipation has been used at the CPU, system, and application level in systems to reduce latency. Methods include caching and prefetching have been used in all these arenas to address access latency, usually directed at an I/O latency bottleneck [5].

Source anticipation has been used in file systems, so that when a file is accessed, its entire contents are transmitted. It has also been used in database systems, and, as with file systems and lower layer instances, solutions are directed at I/O (bandwidth) bottlenecks.

Other research in databases has recently considered using I/O bottleneck solutions to address propagation latency as well. The I/O bottleneck was addressed by the Datacycle architecture at Bellcore [6]. In this system, the database was repeatedly pumped at receivers, where high-speed filters extracted the appropriate reply. High-bandwidth fiber was used from the pump to the receivers, and a low-bandwidth return path was used for update requests.

The "Send-on-Demand" (SOD) architecture uses the Datacycle integrated with many-to-many communication to provide distributed database performance with Datacycle anticipation [1]. SOD periodically pumps information out to the participant pumps, in the manner of periodic state update via timers of the Delta-t protocol [14]. Their goal, as in Mirage [11], is rollback-free anticipation. Mirage performs breadth-first search(BFS) Time-Warp-like anticipation, where the BFS alternates cover the space of possible states, thus avoiding rollback.

1. Tthese are rough estimates at best., so further statistical analysis is not warranted.

2. Technology required includes parallel implementations, high-performance hardware or software implementations, and better integration of protocol processing with other workstation processing. These require changes in protocol implementations, rather than changes of the protocols themselves.

1.4: Interactive Systems

One opportunity for latency-reduction methods is interactive systems, especially those involving user-adaptive interfaces. Latency reduction techniques may be required for their effective use, to provide responsive interaction. Interaction with graphical or pictorial information, especially where highly structured, is a prime candidate.; this means hypertext and hypermedia. Hypermedia encodes structural cues to access in its links, which provide application-independent access to application-specific information, of the specific type required by our methods.

We can also use statistical optimizations to enhance interactive access to visualization data. Current research programs emphasize the "intelligent automatic sequencing and spatial organization of visual or auditory information, to match the...goals of the user." [10] One particular goal of these optimizations is the user perception of low latency, regardless of propagation latency.

2: An illustrative experiment - file transfer latency reduction

The following is an illustrative experiment in latency reduction, applied to an interactive *ftp* session. The experiment determines the extent of gain and cost in using Parallel Communication methods to reduce propagation latency. Parallel Communication is a method of sender-based anticipation using coarse remote state, and is based on Mirage, an abstract model of latency in communication protocols [11] [13]. The gain is the reduction in per-transaction propagation latency; the cost is the increased bandwidth required. In the case of *ftp*, coarse state is the current working directory (*cwd*) of the user. When a user types "cd", she indicates a change in that state. The server anticipates the requests that follow that state change by sending "every possible next request" [11]. In *ftp*, this reduces to sending the entire contents of the *cwd* after each "cd".

We chose *ftp* because it exhibits a limited communication parallelism, and because we could experiment using available log information. It isn't optimal because the parallelism isn't multilevel, as in hypermedia.

The following results were obtained from file transfer logs of anonymous access at ISI (*ftp.isi.edu*) and NCSA (*ftp.ncsa.uiuc.edu*). Ideally, we'd perfer to measure the utilization of a distributed hypermedia system, e.g., the World-Wide Web [15], but here we'll start with some locally-available information, and see where it leads.

Currently, in *ftp*, users usually "cd" to a directory, and retrieve files from that directory. A user session is defined as the contiguous set of file transfers initiated by a user, and is composed of a number of directory sessions. A directory session is defined as a contiguous set of transfers of a single user in a single directory.

What is the effect of a 'degenerate', 1-step case of Parallel Communication, in which the entire first-level contents of a directory[1] are sent at the beginning of a directory session (i.e., no subdirectories)? What would the user-perceived latency reduction be? What would the server load be? What channel bandwidth would be required? What would the effective channel utilization (bytes kept/sent) be? I.e., what are the benefits (latency reduction), and what are the costs (load, bandwidth, utilization)?

2.1: Measurements

The following data are the result log files kept at ISI, and publicly available via anonymous *ftp* from NCSA (Table 1). The ISI logs represent 2.1 gigabytes of transfers, consisting of 32,000 files transferred in 9,000 sessions. The NCSA logs represent 12 gigabytes of transfers, consisting of 54,000 files transferred in 38,000 sessions. The numbers are represented as average±standard deviation.

TABLE 1. File and transfer sizes

	ISI	NCSA
File size	65K ± 3K	210K ±3K
bytes/user	270K ± 13	350K ± 4
bytes/directory	6.9M ± 0.15	2.1M ± 0.016

File sizes are 65KB±2.5KB for ISI[2], and 210K±2.3K for NCSA (Table 1). User sessions consist of 1.1±0.005 directory sessions on both systems, and directory sessions consist of 3.6±0.1 file transfers at ISI, and 1.4±0.009 at NCSA (Table 2). A user retrieves a total of 4.1±0.2 files from ISI and 1.6±0.02 from NCSA (Table 2), for a total of 270K±13K bytes at ISI, and 350K±4K at NCSA (Table 1).

TABLE 2. Session sizes

	ISI	NCSA
Sessions/user	1.1 ± 0	1.1 ± 0
Files/session	3.6 ± 0.2	1.4 ± 0
Files/user	4.1 ± 0.2	1.6 ± 0
Files/directory	114 ± 3	11 ± 0.1

1. We model *ftp* access as "cd"/"get"/"get"... We could have modelled it as "cd"/"ls"/"get"/"get"... This adds negligibly to the model we present, as will be discussed later.

2. This may seem to be a very narrow range, but note that it represents weighting by frequency of byte and file. Certain files dominate the traffic - e.g., the text "info" files.

First, some justification of this analysis. The conventional wisdom of *ftp* access is that the dominant portion of transfers are of single files, because most users of anonymous *ftp* know what they came for. Conventional wisdom also is that *ftp* files are fairly large, i.e., 100KB and up.

At ISI, only about 38% of directory sessions involve multiple file transfer, but about 83% of files transferred are solo (because group-transferred files are often transferred as part of a large set of files) (Figure 2). In terms of the raw bytes transferred, 61% of the bytes are part of multi-file sessions. The skew towards single file sessions is probably the result of the RFC files at ISI (qualitatively verified by the logs[1]).

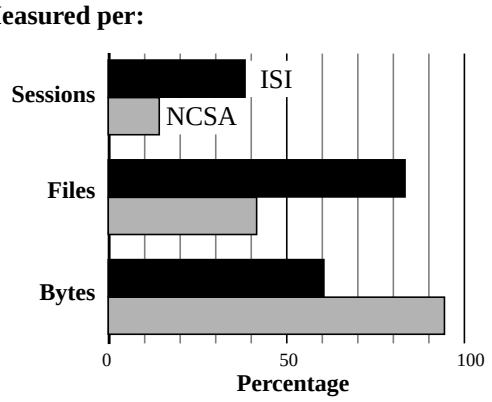

FIGURE 2. Percent of transfers in multiple files

At NCSA, 14% of directory sessions involve multiple file transfer, but about 42% of files transferred are solo (Figure 2). In terms of the raw bytes transferred, over 95% of the bytes are part of multi-file sessions.

If we were to send the entire directory every time a user "cd's" to it, we'd send 114.9±2.8 files at ISI, but only 11±0.1 at NCSA (Table 1), and achieved about 3.4% per-byte efficiency (3.4% of the bytes sent would be those requested) at ISI, and 15% efficiency at NCSA (Table 4).

TABLE 3. Cost and efficiency

	ISI	NCSA
per-byte efficiency	3.4%	15%
BW cost	30x	6.7x
BW/latency eff.	14-27%	29-56%

2.2: Analysis - existing *ftp* method

Consider this session, in the optimistic case that a "cd" costs 2 trip times of latency (i.e., 1 rtt {round-trip time}), and files are transferred using TCP (Figure 3). TCP requires 3 trip times to open a connection, where the first trip time can overlap with the most recent "cd" request. A file can be sent as part of that exchange, and the close can occur via time-out. The total for TCP, per file, assuming transmission without interruption, costs 1.5 rtts + file transfer time. As BW goes to infinity[2] the file transfer time approaches zero[3], and the open/close protocol dominates (Figure 4). The file transfer itself is limited by the server capacity, rather than the rtt.

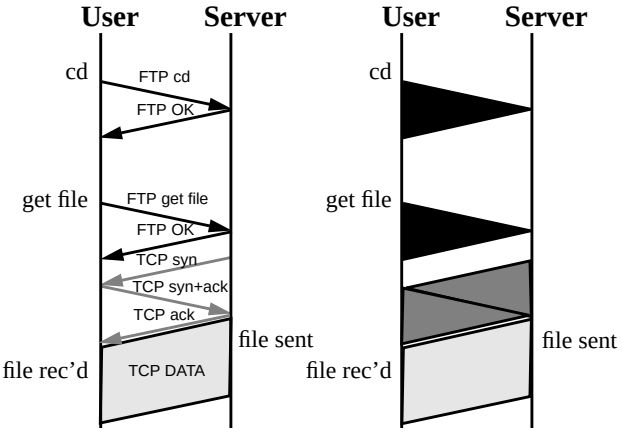

FIGURE 3. *ftp* action sequence, symbolic notation

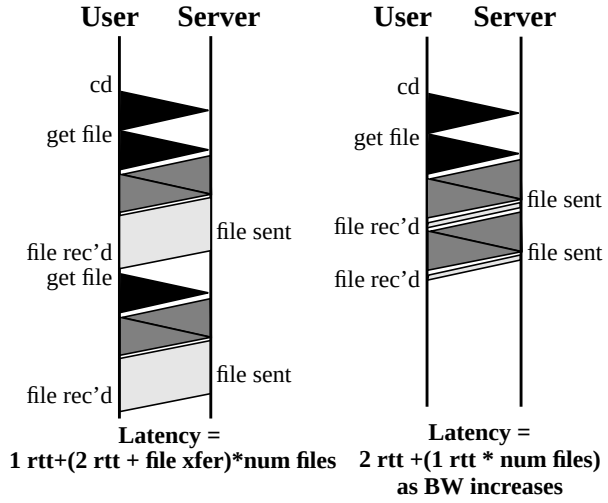

FIGURE 4. Transaction time as BW increases

1. We note that the RFCs uniform average is 56K±3K. Considering only text RFCs (96% by file count), that's 46K±2K. But even these numbers aren't relevant, because they need to be byte-weighted by transfer. We are performing this analysis now.

2. BW is never infinite - that would violate physical laws, by requiring infinite negative entropy in a finite space-time. We can however consider an increasing series of finite BW values.

3. Never actually zero, as a result of the finiteness of the BW value.

For the average 1 “cd” plus 3.6 files transferred, a total of 1 + 3.6 * 2, or 8.2 rtts are expended at ISI (Table 4). At NCSA, the average is 1 + 1.4 * 2 = 3.8 rtts. The latency is reduced to 2 RTTs in the conventional case, and 1 in the transaction-TCP case. The per file latency is reduced from 2.5 RTTs down to around 0.6 for ISI, and 1.4 for NCSA.

TABLE 4. Trip time reduction

	ISI	NCSA
RTT original	8.2 RTTs	3.8 RTTs
RTT new	1-2 RTTs	1-2 RTTs
orig. RTT per file	2.3 RTTs	2.7 RTTs
new RTT per file	0.3-0.6 RTT	0.7-1.4 RTTs

The total bandwidth consumed by sending the entire directory vs. sending only those files requested is the reciprocal of the bandwidth “efficiency” (BW efficiency is defined as [bytes used]/[bytes sent], i.e., [sum of file sizes requested]/[sum of file sizes of the directory]). The total bandwidth consumed is approximately 30x larger (1/3.4%) at ISI, and only 6.7x (1/15%) at NCSA.

2.3: Analysis- Parallel Communication method

Parallel Communication indicates sending the entire directory in reply to the “cd” (Figure 4). The file reply latency can vary, depending on the method used. We propose using Transaction-TCP [2], which should be open for the entire *ftp* session anyway. The “cd” is a transaction, and the files follow on the heels of the “cd”. As a result, the “cd” and entire directory send occur in 1 rtt. If instead we perform the “cd” separately, and send the entire directory as 1 TCP transfer (1 rtt when overlapped with the “cd”, as shown), we end up with 2 rtts cost. The result drops to between 1 and 2 rtts, depending on whether you count the send as a remote TCP transfer.

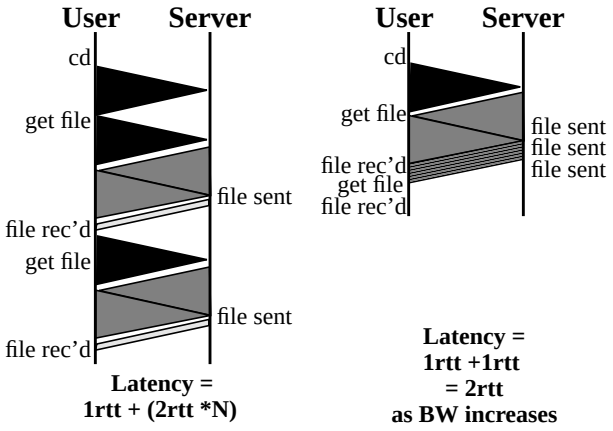

FIGURE 5. Pre-sending directory after a *cd* request

At ISI, this result reduces 8.2 rtts to between 1 and 2 rtts (Table 4). The latency reduction, assuming bandwidth availability, is between 8.2x and 4.1x speedup. At NCSA, it reduces 3.8 rtts to between 1 and 2, for a total of 3.8x to 1.9x speedup.

At ISI, a 30x bandwidth increase supports a speedup between 8.2x and 4.1x, or a ratio cost of between 3.7:1 and 7.3:1 (Table 3). The bandwidth increase is between 27% and 14% effective at ISI. At NCSA, a 6.7x bandwidth increase supports a speedup between 3.8x and 1.9x, or a ratio cost of 1.8:1 and 3.5:1. The bandwidth increase is between 56% and 29% effective at NCSA.

In order to perform sender-based anticipation, the entire directory must be in transit after a request[1]. The entire directory transmission time should be some fraction (less than 1/5) of the BW-delay product. The average entire directory size (weighted by use[2]) is 6.9 M bytes (±150KB) at ISI, and 2.1 M ±16KB at NCSA (Table 1).

There are two ways to look at latency and bandwidth as they correlate to directory size. For a given directory size and bandwidth, there is a minimum latency at which sending the entire directory is better than sending a direct file as a response. Alternately, for that same directory size and a given latency, there is a minimum bandwidth required.

At ISI, at 1 Gbps, this requires 55 ms to transmit the entire directory as a response to the “cd” request (Table 5). The overall propagation latency would need to be about 300 ms latency, for propagation costs to dominate. This time is achieved in satellite links (vs. 20 ms speed-of-light and 100 ms total ground propagation delay within the USA). At NCSA a 1 Gbps line needs 16 ms latency in-transit, or 80 ms propagation, achieved in the Internet today within the USA (coast-coast).

TABLE 5. Break-even points

	ISI	NCSA
1 Gbps delay req.	55-300ms	16-90ms
USA rate min.	3 Gbps	800 Mbps

Alternately, at 20 ms latency ISI requires a link speed of 2.75Gbps, at 100 ms it requires 0.6 Gbps in-transit. Assuming a factor of 5x where the data transmission becomes negligible compared to propagation latency, this means 14 Gbps speed-of-light, or only 3 Gbps propagation

1. We could also send a directory listing with a “cd”, modelled after a “cd”/“ls”/“get”/“get”/... interaction. The “ls” output represents about 10-100 chars per file. Upper-bound, that's about 10,000 chars at ISI, and 1,000 chars at NCSA. That's noise on these numbers, though.

2. We weight averages by number of times a file is transferred, and by the number of bytes in the file. Thus the weight represents the effect a directory has on the overall communication byte stream.

(Table 5). At 20 ms NCSA requires a link speed of 800 Mbps. Assuming a factor of 5x as before, this requires 4 Gbps speed-of light, or 800 Mbps propagation.

Multiple file sessions are not required here, because the directory contents are transferred as soon as the "cd" request is indicated. A more aggressive implementation would send from the 'last known node' in the directory structure, going beyond the single directory level[1]. The result is that the server is driven at the network bandwidth. This replaces a physical limitation with a performance limitation. It is also a good argument for using the network itself as storage, to avoid server processing costs. The work of the server is increased to provide latency reduction without increasing the work of the receiver (other latency reduction methods, such as prefetching and caching, increase receiver workload to reduce latency).

3: Why is this interesting?

This experiment is based on the Mirage model of latency in communication [11]. In Mirage, latency induces imprecision of state, which is resolved via feedback. Latency is accommodated via source anticipation. Mirage is the name of the model; Parallel Communication is the name of the protocol derived from the model [13]. The model is based on analogies between protocols and latent interaction and quantum physics [12].

In source anticipation, the state of the receiver is partitioned into subsets, and messages are pre-sent labelled for each subspace. The result is a breadth-first search of the remote space, in contrast to the depth-first search of Time-Warp-like protocols. As a result, Mirage never backs-up; instead, it resolves the ambiguous remote state into one of the sub-states.

Latency reduction in FTP is a one-level, non-recursive Mirage protocol. Mirage indicates that the extent of the latency reduction is a function of the bandwidth available, the branching of state, and the linearity within a single state. The branching here corresponds to the number of files per directory, and the linearity corresponds to the sizes of the files. The feedback corresponds to the "cd" requests. The efficiency and effectiveness of Mirage depend on the partitioning of the space, which here corresponds to the power-set of files actually selected, vs. the files in the directory.

In effect, this experiment embeds Parallel Communication on the *i-node* structure of the FTP directories, together with the request-response FTP protocol itself. The result is a system which uses bandwidth to compensate for latency, using known structure of the interaction to beat the "speed-of-light" limit.

4: What next?

Actually, *ftp* isn't quite what we had in mind. *Ftp* users are interactive, and don't care about propagation latency. *Ftp* also has a very flat structure, especially here at ISI and at NCSA- *ftp* directories aren't very deep, and files are often rooted in a single directory only. We'd prefer to measure the speedup of a system that shows a truly interactive nature, with large chunks of information, and a rich structure.

One candidate of particular interest is the World-Wide Web [15]. WWW uses some of the same mechanisms as *ftp*, but uses embedded textual "cookies" to indicate hypermedia links to other files, rather than the *ftp* directory structure. We are in the process of looking into log and raw file access of WWW components, for further investigation.

4.1: Conclusions

We must see beyond the existing implications of the changes in communication bandwidth rates now occurring. Communication protocol research requires new models, rather than additional mechanisms and implementations. New paradigms of communication and interaction, made possible by the changing nature of high bandwidth channels, need to be exploited.

In this experiment we used Parallel Communication to reduce the propagation latency of *ftp*. At 1 Gbps, 80-300ms latency is required for bandwidth to be used to reduce propagation latency. Alternately, at 100ms, a bandwidth between 800Mbps-3 Gbps is required. Propagation latency can be reduced to 2 round-trip times, to as little as 0.6 round-trip time/file. This can be achieved at a cost of 7x additional bandwidth. These results are impressive when we consider *ftp* was not the optimal application to analyze.

We believe that propagation latency can and should be addressed. We're nearing a point where bandwidth can be seen as a means to that end, rather than as merely an end to itself.

4.2: Acknowledgments

This paper is the result of discussions with and feedback of Rafael Saavedra, Bob Felderman, Mike Carlton, and Jon Postel at USC/ISI, Nick Short at NASA/GSFC, and Sanjita Banerjee at University of Pittsburgh, as well as feedback from the reviewers of Infocom.

1. This is an argument for self-restructuring *i-node*s (i.e., physical restructuring without logical effect). Thus accessing the data would have an effect on the structure of that data. Essentially, this is Parallel Communication on the *i-node* structure of the server directory.

5: References

[1] Banerjee, S., Li, V.O., Wang, C., "Distributed Database Systems in High-Speed Wide-Area Networks," *IEEE Journal on Selected Areas in Communications*, V. 11, N. 4, May 1993, pp. 617-630.

[2] Braden, R., "Extending TCP for Transactions -- Concepts," RFC-1379, USC/ISI, Nov. 1992.

[3] Cheriton, D.R., "VMTP: A Transport Protocol for the Next Generation of Communication Systems," *Computer Communication Review*, Aug. 1986, p. 406-415.

[4] Chesson, G., et. al., *XTP Protocol Definition*, Protocol Engines, Inc., Dec. 1988.

[5] Hennesy, H.L., and Patterson, D.A., *Computer Architecture: A Quantitative Approach*, Morgan Kaufmann, 1990.

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

[7] Jacobson, V., and Braden, R., "TCP Extensions for Long-Delay Paths," RFC-1072, LBL and USC/Information Sciences Institute, Oct. 1988.

[8] Jacobson, V., Braden, R., and Zhang, L., "TCP Extensions for High-Speed Paths," RFC-1185, LBL and USC/Information Sciences Institute, Oct. 1990.

[9] Jacobson, V., Braden, R., and Borman, D., "TCP Extensions for High Performance," RFC-1323, LBL, USC/Information Sciences Institute, and Cray Research, May 1992.

[10] NSF Report 92-109, "Research Priorities in Networking and Communications," Oct. 1992.

[11] Touch, Joseph D., *Mirage: A Model for Latency in Communication*, Ph.D. dissertation, Dept. of Computer and Information Science, Univ. of Pennsylvania, 1992. Also available as Dept. of CIS Tech. Report MS-CIS-92-42 / DSL-11.

[12] Touch, J.D., "Physics Analogs in Communication Models," *Proc. PhysComp '92*, Oct. 1992, p.248-252.

[13] Touch, J.D., "Parallel Communication," *Proc. IEEE Infocom*, Mar. 1993, p. 505-512.

[14] Watson, R.W., "The Delta-t Transport Protocol: Features and Experience," *Protocols for High Speed Networks*, Elsevier, 1989, p. 3-17.

[15] Berners-Lee, T.J., Cailliau, R., Groff, J-F, Pollermann, B., "World-Wide Web: The Information Universe," *Electronic Networking: Research, Applications and Policy*, Meckler Publishing, Connecticut, Spring 1992, p.52-58.