# How Slow Is One Gigabit Per Second?

*Craig Partridge*

**Abstract**

At first blush, one would expect that increasing data network transfer rates by two orders of magnitude (from the ubiquitous 10 Mbit speed of today's LANs to the greater than 1 gigabit-per-second speeds we expect of networks in the early 1990s) would severely impact our choice of network protocols and architectures. This report presents the strawman argument that, in fact, moving to one-gigabit data rates presents surprisingly *few* problems.

## 1. Introduction

Recently there has been considerable interest in gigabit-per-second networks for three reasons:

(1) The demand for bandwidth on data communications networks is growing rapidly (the current growth curve is steeply exponential) and it appears that by the mid-1990s there will be an aggregate demand for network bandwidth that approaches one gigabit per second.

(2) Innovative applications under development in research labs have large bandwidth requirements, often approaching a gigabit per second for a *single* application.

(3) It has been an article of faith for many researchers in the networking community that the current networking architectures (and in particular, the datagram-oriented architectures) would have difficulty scaling to very high speeds. The exact speed at which the architectures would break down has never been stated, but the popular impression is that one gigabit per second is sufficiently fast to exceed the limitations of the current architectures.

For several months I have participated in a research effort intended to help shape the architecture of gigabit-per-second networks. This effort involved spending considerable time trying to find the constraints that gigabit-per-second data rates placed on network architectures. In other words, Accepting the idea in point three above that gigabit-per-second networks would severely strain or break current architectures, I hoped that by discovering where those strains or breaks occurred, I would be better able to recommend changes or innovations necessary for a new, gigabit-oriented, network architecture.

The research yielded little reason to believe that a gigabit-per-second data rate alone would force us to adopt a new network architecture. This is not to say that gigabit-per-second networks present no difficult problems to the researcher, but rather that the problems presented are less extraordinary than one might initially suspect and do not appear to require us to develop a new network architecture to solve them.[1]

---

This paper was written under DARPA Contract MDA903-89-C-0020. The opinions in this paper are the author's and do not necessarily reflect those of the Defense Advanced Research Projects Agency. The author's address is BBN Systems and Technologies Corporation, 10 Moulton St, Cambridge MA 02138, or via electronic mail, craig@bbn.com.

[1] We may still employ a novel network architecture when building gigabit-per-second networks. Some of the research work on gigabit applications suggests that we may need new mechanisms for communicating between network peers, and some researchers have argued that as networking matures, we must incorporate additional functionality such as policy routing into our networks. Such new mechanisms or functionality might cause us to develop a new network architecture to support them. This report simply argues that speed alone is not a good justification for change.

By examining both the performance of individual components and more systemic issues, such as problems of flow control and buffering, on gigabit-per-second networks, this paper attempts to support the contention that moving to one gigabit data rates presents relatively few problems. The discussion is meant to remain neutral about the particular architecture to use; the goal of the paper is simply to show that neither the virtual-circuit nor the datagram architectures are precluded by gigabit-per-second speed networks.[2]

## 2. Basic Assumptions

This argument rests on two basic assumptions:

- The average datagram or packet sizes on gigabit networks will be larger than those on current networks, such as the Internet.
- It is possible to make a good estimate of the processing capacity of future routers and hosts.

The remainder of this section explains these assumptions.

### 2.1. Average Datagram or Packet Sizes

The average datagram or packet[3] size on a network largely determines the rate at which routers need to be able to switch data and influences how hosts use the network. As a result, estimating the average packet size is essential to any argument about a router or host's ability to send or receive at gigabit speeds.

For the purposes of this paper, assume that the average packet sizes for hosts and routers on the Internet (about 100 octets for gateways and somewhere between 100 and 500 octets for hosts) is a conservative estimate of the average packet size on a gigabit network.[4]

The current packet size is believed to be a conservative estimate for two reasons.

First, the proportion of bulk data transfer to interactive data transfer on gigabit networks will probably increase. Currently, one of the primary reasons for building gigabit networks is that researchers want to be able to exchange large quantities of data very fast. Thus one would expect to see more large packets (carrying bulk traffic) on the network in the future than we do currently. Additional large packets will increase the average packet size.

Second, the maximum transmission units (MTU) on gigabit networks will probably be higher than the MTU on current networks. Certainly the trend is toward large MTUs.[5] The FDDI MTU is substantially larger than the Ethernet MTU. Larger MTUs imply that bulk data transfers can use fewer and larger packets; again, all other factors being equal, the average packet size will increase.

Averages don't tell the whole story, however. The average packet size can vary dramatically depending on what network users are doing at a given moment. Smaller packets make it harder to achieve high speeds, and larger packets make high speeds easier. But predicting how

[2] The fact that many of the numbers in this paper are derived from TCP/IP networks reflects the fact that this information was most easily available.

[3] From here on, the word *datagram* will be reserved for use only when networks using a datagram architecture are being discussed. In cases where the architecture could be either datagram or virtual circuit, the term *packet* will be used.

[4] The average packet size estimates come from a variety of sources. RFC 1009 estimates the packet size at routers to be about 100 octets. The packet size for hosts comes from observations on networks at BBN, ISI, MIT, and Sun Microsystems.

[5] Except for ATM networks, but ATM is essentially a multiplexing scheme. Hosts and routers will almost certainly use a larger transmission unit, which is then broken down into ATM cells for transmission.

gigabit networks will be used, in enough specificity to do detailed estimates of traffic mixes, is all but impossible. Therefore, estimating requirements based on conservative average packet sizes seems a good approach.

### 2.2. Building Future Hosts and Routers

To discuss the computing limitations of hosts and routers in a gigabit network, one must identify candidate hardware that could be used to build them. This paper discusses components that that are currently in test production, as a reasonably conservative way to predict what hardware would be available in the early 1990s. Furthermore, the focus is on inexpensive components, the sorts of processors and memory one would expect to have in a workstation class machine. Cray routers would be interesting, but probably prohibitively expensive in real life.

The key characteristics of the candidate architectures are:

- A 60 to 70 million instructions-per-second RISC CPU. The logical near-term successor to the CPUs in workstations today are the RISC CPUs currently in test production or advanced development. Intel currently has CPUs of this speed in test production.
- 64-bit data paths and word sizes. Currently most machines have 32-bit data paths, but there is plenty of room on most VLSI chips for more pins, and the industry is apparently concluding that they need the larger data paths. Note that using a 64-bit data path, a one gigabit-per-second transfer speed requires only a 16-Megahertz clock. Inexpensive RAM memory already exists that can clock out bits at this speed. Thus, modulo the problems of multiple address ports, discussed in the section on interfaces below, one can assume that a system can move data internally at gigabit rates.

## 3. Network Components

A likely place to start the strawman argument is with the components of the network themselves. Clearly, if our choices for gigabit speed network components are very limited, our choice of network architecture may be constrained.

This section examines three types of network components: the hosts or hosts that use the network, the switches or routers that direct the data flowing between hosts, and the physical links over which the traffic actually passes. This troika, in some form, appears in all network architectures. The discussion progresses bottom-up, starting with the links and working its way up to the hosts.

The general goal of this section is to suggest that we have enough processing or computing power in all parts of the network to handle gigabit-speed data traffic.

### 3.1. Link Speed And Interfaces

The first question is whether we can transmit data at one gigabit per second between systems. The answer is yes, and indeed the problems involved seem to be relatively well understood, and there is a broad choice of methods for building gigabit networks. See Acampora and Karol’s article for a good discussion [2].

### 3.2. Routers

Routers are responsible for switching data through the network between an originating host and a destination host. In varying forms, with different mixes of hardware and software, they are a component of all major network architectures. Examples include routers between IP networks and X.75 gateways between X.25 networks.

Historically, the dominant cost in a router has been the cost of switching the packet. As a result, the key question in this section is: Do we have enough CPU bandwidth to do per-packet processing required?

The model used here is the basic architecture used by most of today's packet switches: one or more general-purpose CPUs, memory, and network interfaces, plus the possible addition of some special-purpose hardware (often used for address or route lookup). In these configurations, the bottleneck is usually the CPU.

Why is the CPU the key bottleneck? Because routers need to examine only a small part of each packet (the header). Therefore the other obvious bottleneck, the data bus, is underloaded, unless an entire packet needs to be moved across the bus. Newer systems work very hard not to transfer entire packets; they put multiple interfaces on the same I/O board, sharing the same memory. In these configurations the CPU need simply copy the header from and to the I/O memory and set a bit on the device to switch a packet from one interface to another. Thus, in practice, the major resource that is usually in short supply is the CPU.

To estimate how much CPU time we have per packet, we estimate the number of packets per second that a router will see at gigabit speeds. The method used is to extrapolate from typical performance expected today.

Current wisdom in the TCP/IP community is that a router between two Ethernets must be able to switch at between 6,000 and 10,000 packets per second.[6] Doing a simple linear extrapolation, if we need to switch 6,000 to 10,000 packets at Ethernet speeds (10 Mbit/sec) we will need to switch 600,000 to 1,000,000 packets at gigabit rates. Or, to look at the problem another way, we need to be able to switch a packet in 1 to 1.6 microseconds. Note that if we have a pipelined architecture, we can take longer to switch the packet, but no stage in the pipeline can take longer than 1 to 1.6 microseconds.

Consider a router built with one processor (currently most routers use at least two processors). Given the expected processor speed of 60 MIPS, we have 60 to 100 instructions in our single CPU with which to handle the packet.

Sixty instructions per packet is almost certainly acceptable for virtual circuit architectures which amortize circuit set-up costs over several packets. Once a circuit is set up, switching from one interface to another can be done with a handful of instructions.[7] The real question is whether 60 to 100 instructions is enough to switch a packet.

The answer is, probably. Certainly it is fast enough to switch simple packets. The DEC LANBridge 100 switches Ethernet frames using about 50 instructions per packet (with some hardware assist).[8] More complex packets such IP packets are somewhat harder to switch.

Using 32-bit processors, current routers typically switch an IP packet using between 100 and 150 instructions plus some code to run the device drivers. That instruction count should be reduced somewhat for a 64-bit processor because many operations (such as additions in the checksum and checking header fields against bit masks) would use 64-bit integer or word sizes instead of 32 bits. So, if our device drivers are sufficiently simple to drive, or if we can make some switching functions faster through use of hardware, we should be able to switch even moderately complex packets like IP at speeds of one gigabit per second.

---

[6] This value is consistent with an average packet size of a little over 100 octets for gateways. If we assume all packets are switched off the Ethernet, these rates convert to an average packet size of between 100 and 200 octets.

[7] For example, using the Butterfly gateway-to-gateway header protocol, which is used to specify a path through a series of Butterfly gateways, switching a packet requires fewer than 30 instructions.

[8] Personal communication from Ross Callon.

### 3.3. Hosts

Determining what is required of a host on a gigabit network is more difficult than determining what is required of a router, because hosts and routers are very different.

At least three key differences affect this evaluation:

- Historically, a single host has not been expected to consume the entire bandwidth of the network it is attached to. The introduction to this paper notes that some applications currently being developed may require that their host systems be able to support gigabit transfer rates, but these applications also tend to have special traffic characteristics (i.e., they use maximum-size packets) that make them different from run-of-the-mill host traffic.
- Even ignoring the question of new applications, the average packet size for hosts appears to vary widely, from about 100 to over 500 bytes, depending on traffic mix. In particular, hosts that participate in distributed filesystems and do paging over the network appear to have larger average packet sizes.
- Finally, hosts usually want to look at the data, so any calculation of resource requirements must allow for some computation on the data received.[9]

Note that hosts also always need to checksum their data, while routers often do not. This difference does not play a role the discussion below, because it is reasonably easy to put a checksum into hardware, which is operated in parallel with other protocol operations.

### 3.4. Host Computational Requirements

To determine a host's ability to process data at one gigabit per second, we first need some estimate of the computation that an end system is expected to do on a packet. This estimate can be divided into three parts: protocol processing, application processing, and operating system overhead.

For an estimate of protocol processing costs I used the measurements by Clark, Jacobson, Romkey, and Salwen for TCP/IP processing overhead [2]. Note that their measurements were done on code that does not include many of the current innovations, such as header prediction, that can dramatically reduce processing time. Their measurements showed that sending or receiving a TCP segment took about 300 CISC instructions, inflated to 400 instructions for RISC processors. These numbers include IP processing, TCP processing (exclusive of checksum), and buffer processing, but not device driver support. The checksum does not need to be done in software, so we don't have to add this in. Device driver code can be very large or very small, depending upon the device. For this discussion, assume a device driver of moderate complexity requiring about 400 instructions. This gives a protocol processing cost of about 800 instructions.

To this estimate some instructions for context switching and other operating system overhead must be added. I've estimated this value at about 200 instructions.

Finally, some time must be allowed for processing the data. The data can be processed only as fast as every word can be touched. In other words, a segment can't be processed faster than the host can read each word. Thus the total estimated per-packet processing cost is around $1000+l/w$ instructions, where $l$ is the packet length and $w$ is the host's word size.

---

[9] Note that some bulk-data applications do not examine their data. For example, receivers of file or video transfers are often moving data straight from the network buffers into disk or video buffers, without further application processing. Including application processing time may overstate the host computation requirements in some cases.

Although these numbers are taken from datagram networks, I believe they are the same or lower for virtual circuit networks.[10]

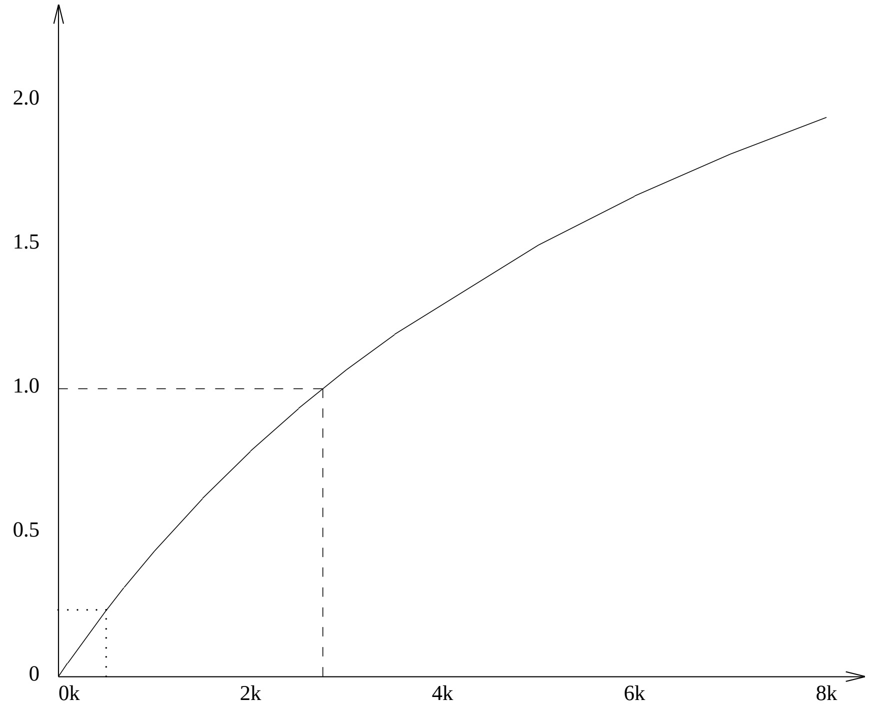


**Figure 1: Packet Size (in bytes) vs. Bandwidth (in $10^9$ bits)**

### 3.5. Hosts And Variable Packet Sizes

The average packet size for hosts appears to vary dramatically depending upon the applications the host is currently running. Casual observations at various sites show average packet sizes ranging from 100 to 700 bytes and a moderate variation over time, as application mixes change during the working day. As a result, I've chosen, to make the packet size the free variable in the equation relating bandwidth, packet sizes, and available instructions.

Figure 1 shows the bandwidth a host can consume versus the packet size used to send the data. To generate the graph, the number of instructions required to process one packet of a given size was computed. This number was then used to compute the maximum number of packets that could be processed by a 60-MIPS CPU, and then the packet size was multiplied by the number of packets that could be processed to get the consumable bandwidth.[11] The packet sizes range up to 8 Kbytes, which is double the FDDI packet size. (If FDDI, which is 10 times faster than Ethernet, uses an MTU somewhat more than twice as big as Ethernet's, it does not seem unreasonable

[10] The transport processing (exclusive of checksum), application, and operating system costs are the same. The differences, if any, appear in the network layer and device driver code. In practice, the device drivers for circuit networks tend to be a bit more complex than those for datagram networks, because a single device supports multiple circuits, and the multiplexing and demultiplexing of data and commands for multiple circuits into the command registers of a single device is complex and often CPU-intensive. However, the network layer code in virtual circuit networks is typically somewhat simpler than the comparable code for datagram networks.

to assume that gigabit network, 10 times faster than FDDI, might use an MTU twice again as big).

At least two pieces of information are clear from the graph. First, achieving gigabit speeds will be possible for hosts. As the dashed line shows, to achieve one gigabit throughput, a host (or application) need only have an average packet size of about 3 Kbytes, which will probably be well within the MTU size of gigabit networks (FDDI, for example, has a 4 Kbyte MTU).

The second item of note is that, even with small average packet sizes, a 60-MIPS host could consume a large fraction of a gigabit. The dotted line shows that for a 512-byte average packet size, the system could use nearly a quarter of a gigabit. Even at a 100-byte average size, a system could use nearly 50,000,000 bits per second, about one twentieth of a gigabit capacity.

Both of these observations are encouraging. They suggest that in normal use (with average packet sizes below 1,000 bytes) hosts will still be able to use an appreciable fraction of a gigabit bandwidth, and that, should those systems have applications that need to fill a gigabit pipe, gigabit speeds can be achieved using a moderately large packet size.

Is it fair to ask applications to use large packet sizes to achieve gigabit speeds? In almost all cases, yes. Larger packets are more efficient for the network to process, as well as for routers and hosts to process. Even interactive traffic such as remote terminal applications can usually benefit from aggregation of single keystrokes into larger packets: witness Nagle's algorithm [5] and the DEC LAT protocol. The one exception to this rule is applications with strict time constraints, such as digitized speech. But these applications typically have low bandwidth requirements, and do not need to achieve gigabit throughput between peers.

### 3.6. Yet Another Way to Look at Things

Here is one more way to look at the question of processing and resources. So far, this discussion has typically measured packet costs in instructions or bandwidth consumed. But another useful way to look at the costs is in terms of time. Given a gigabit data stream, packed as tightly as possible with packets of size *n*, how long does a component have (in seconds), to process that packet?

Figure 2 shows this relationship between packet size and time, with time measured in microseconds.

This graph gives a generally comforting picture. The time available to process even small packets is not impossible to achieve, and for larger packets, such as those of 6 Kbytes or larger, the processing time available is feasible today (the DEC LANbridge switches a packet in 50 μs).

## 4. Flow Control and Buffering

This section of the paper concerns systemic problems. Having argued that individual components of a one gigabit network can work comfortably at gigabit speeds using either a datagram or virtual circuit model, I will now consider whether the gigabit network as a whole is stable.

This issue is not examined simply for completeness; it is a pressing question. Gigabit networks will be moving immense amounts of data at very large speeds. Errors or faults in the

---

[11] As a sanity check on this model, I also did a small test with some real life numbers. Jacobson has reported that he can saturate an Ethernet with a file transfer using a 2-MIPS workstation and 1536-byte packet sizes. He further noted that the workstation still had free cycles. Assuming the same 1000 instructions + 384 instructions for handling data and an additional 384 instructions for the software checksum (which we assume is handled in hardware at gigabit speeds), we get an expected speed of nearly 14 megabits, consistent with Jacobson's observations. (Actually, he estimates his maximum bandwidth to be somewhat higher thatn 14 megabits/second).

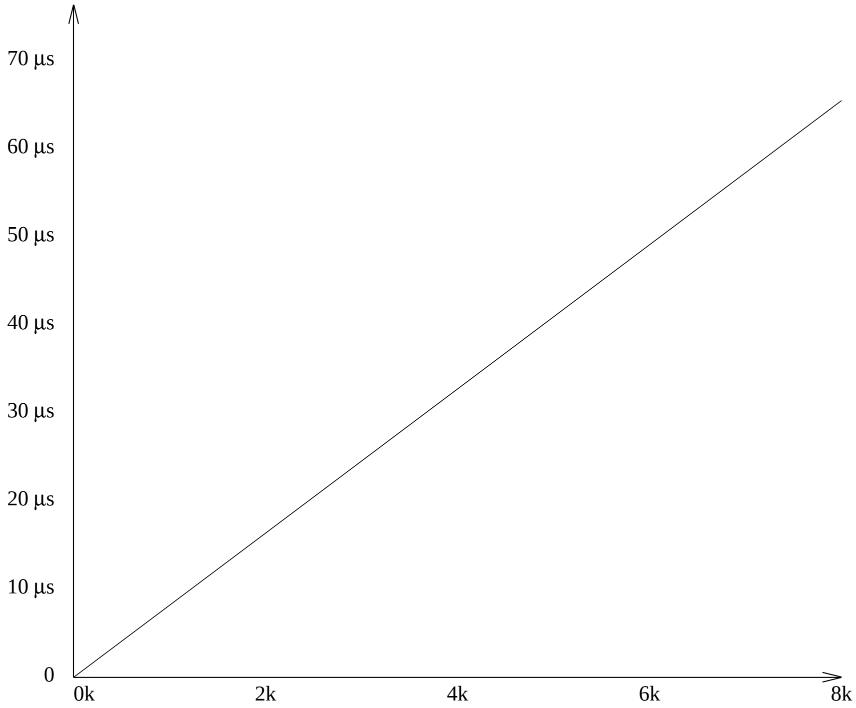


**Figure 2: Packet Size (in bytes) vs. Time to Process (in µ-sec)**

system may in a short time lead to megabytes or even gigabytes of misrouted, misdelivered, rerouted or, possibly, discarded data. The ease with which large amounts of data can meet such fates is often viewed as a serious problem for gigabit networks. Furthermore, this problem appears likely to get worse as speeds increase. Gigabit speed networks pass data at the speed of light. Additional bandwidth will be achieved by packing more bits per centimeter of light, not by reducing the network delay, so the delay-×-bandwidth product, the amount of data that can be in-flight at any one time, will continue to increase. Thus, if there is a problem with managing large flows, it can only get worse as networks go faster.

However, the problems of moving large data are not unmanageable. First, measured in the context of a gigabit speed network, megabytes of data, however mishandled, are much like misdirecting a few large packets in a current-day network. For example, discarding a megabyte of data in transit on a gigabit network is comparable discarding a single 576-byte IP datagram on today's 56-kilobit networks. In other words, we have to recalibrate our impressions of ''big'' to conform to a gigabit world. Second, once you are willing to redefine ''big,'' our current control algorithms, perhaps with minor tweaking, actually appear to scale reasonably well to the gigabit realm.

### 4.1. Host Buffering

One concern about gigabit networks is how much buffer space is required by hosts. To transfer data reliably from one host to another, the sending host must buffer data which has been sent but not acknowledged. At the same time, for efficient use of bandwidth, it is important that a sender of data be able to put multiple packets of data in flight before waiting for an acknowledgement. It is also important for attaining good throughput that the receiving host buffer (as opposed to discarding) any packets received out of order [4].

Clearly, if the buffering requirements are unattainable, hosts will be unable to achieve gigabit transfer speeds, because they are unable to buffer enough data to keep a gigabit link filled.

### 4.1.1. Basic Buffering Requirements

How much buffering is required? The maximum amount of buffering required is the delay-×-bandwidth product for one-round trip time over the longest possible path.[12] The key problem is estimating the round-trip time.

A naive method is simply to compute the round-trip time through pure fiber for a plausible longest path and determine how much buffering is required in the worst-case situation, where all data is being sent via the longest possible path. For this paper the longest path will be assumed to be a round-trip across North America, about 6,000 miles. At the speed of light through fiber, this distance takes about 47 milliseconds to traverse. The delay-×-bandwidth product is 5.9 megabytes.

A more reasonable estimate of the round-trip time is that of the long lines carriers, which reportedly promise a delay of 60 milliseconds between points-of-presence, which gives a round-trip time of no less than 120 milliseconds. The delay-×-bandwidth product is 15 megabytes.

Six megabytes of buffering is not unreasonable. Current workstations vary widely in the amount of memory they support, but a memory size of between 4 and 12 megabytes is common. A nice rule of thumb is that memory sizes roughly double every two years for constant price. So following the practice of trying to keep price constant, we might expect to see memory sizes of 8 to 24 megabytes, and in four years, 16 to 48 megabytes. Certainly, at the four-year mark, reserving 6 megabytes of memory for network buffers seems plausible.

Buffering 15 megabytes appears more difficult. It is certainly possible to add enough memory to buffer 15 megabytes (today's large DRAM chips allow us to cram enough memory on a board), but the extra memory will increase the cost of our systems.

### 4.1.2. The Effects of Switching Delays on Buffering

Another point to consider about buffering is how much delay is introduced by switching between networks; i.e., how much additional host buffering do we need to accomodate the delays in our gigabit equivalents of IP routers or X.25 switches?

The answer appears to be that switching delay should have a negligible effect. Recall that Section 3.2 contained the calculation that no stage in a packet switch or router could take longer than 1 to 1.6 microseconds if it was to keep up with its expected traffic load at gigabit rates. To increase the necessary host buffering by one megabyte on a gigabit network, the delay must increase by 8 milliseconds. Thus, we would have to insert 5,000 to 8,000 switching elements into the round-trip path between hosts to cause the host buffering requirements to increase by one megabyte. Or, thinking of this another way, if we put 100 switching elements into the round-trip path, the additional buffering required is only 12.5 to 20 Kbytes![13]

---

[12] Why one round-trip time? Two reasons: (1) if one uses less than one round-trip time's worth of buffering, and if the network is otherwise idle, the host will have to stop sending before the network is full; (2) if on the other hand, one uses more than one round-trip time's worth of buffering, since fiber is very reliable, the excess buffering is almost never used and is wasteful to allocate.

[13] Note that this calculation is done without considering the effects of congestion, because congestion increases the round-trip time delay, but doesn't increase the amount of data that should be outstanding. Given a network capable of moving $n$ bits per second, sending $n$+1 bits in a second doesn't increase the delivery rate — it just causes the extra bit to be delayed in a queue somewhere. You are better off just sending $n$ bits. In other words, when a network is congested, you should not try to put more data into it.

### 4.2. Flow Control

Gigabit networks have at least two important flow control problems. Both are caused by delay. Neither circuit nor datagram architectures completely solve either problem.

Because the network transmission delay cannot be faster than the speed of light in fiber, gigabit networks will pack a relatively large number of bits into a length of cable. This high concentration of bits has at least two important consequences. First, hosts that want to fill a gigabit link will need mechanisms capable of managing a connection where megabytes of data will be in transit at any given time. A second consequence, which is in some sense an inverse of the first, is that establishing a connection between two hosts becomes more expensive, in terms of data not sent. In the time that a host spends waiting for a call request or connection open to go through, several megabytes of data could have been sent. In other words, an opportunity cost is associated with connection set-up.

The first problem, managing large amounts of data in flight, appears to be the easier of the two to solve.

In the virtual circuit world, the problem is trivial. A virtual circuit with sufficient capacity is created and the data is sent through it. Enough buffering is provided in the circuit to support the maximum flow.

Datagram architectures have somewhat greater difficulty supporting large transfers. All the basic datagram control mechanisms are believed to work a gigabit speeds. The slow start algorithm can still be used to probe the bandwidth of the path, and discarding data when the network is severely congested will not have catastrophic effects. But datagram architectures have historically not provided guaranteed bandwidth, and on gigabit networks there may be applications (like real-time video) in which guaranteed bandwidth is necessary.

There have been several suggestions for how guaranteed bandwidth could be incorporated into a datagram architecture. Some of the more interesting ideas involve having routers infer the existence of guaranteed bandwidth flows from their traffic patterns or specially tagging datagrams affiliated with a guaranteed bandwidth flows. Both schemes would appear preserve the basic characteristics of datagram networks (especially recoverable state).

The problem of the opportunity cost of a connection set-up looks less tractable. This problem can be broken up into three closely related problems:

(1) How long before the first bit comes back from the remote system? If we send a request to a remote system, how long do we wait to get the start of the reply? This question is critical for distributed systems, where the delay time for network transactions can have a large impact on system performance. Unfortunately, the reply cannot come back faster than the speed of light delay. This has critical consequences for distributed systems, which will soon find that modulo small changes in host processing times, getting the first bit back in a network transaction will never get any faster.

(2) How much can we send on the first round-trip? One possible way to cope with the long delay waiting for the first bit is to send more data back in response to the original request. Again, unfortunately, this is difficult. Until at least one round-trip is made, the capacity of the network cannot be learned, and thus sending more than a trivial amount of data (a few kilobytes) risks causing mild congestion.[14]

---

[14] Note that one connection sending lots of data won't affect congestion or round-trip times very much, but several starting up at once could have catastrophic effects – particularly if there are delay-sensitive applications, like voice, using the network. So, to be conservative, each connection has to start slowly.

(3) How long before we can send at full network capacity? If we can't get more than a small amount of data back in one round-trip time, how long before we can send at whatever bandwidth the network is willing to give us. For virtual circuits, we have to wait until the virtual circuit is established, usually one-round trip time.[15] For datagram architectures, the approved method is to probe the network using exponentially larger amounts of data on each round-trip time, until the network capacity is reached [3]. This takes several round-trip times but starting with one 8-kbyte packet, the elapsed time before one gigabit is reached is well under 2 seconds. That is less time than it takes to make a phone call, which makes it likely that users would not find the delay upsetting. However, a 2-second delay is too long for some applications, and it may be that the datagram network needs to be able to provide more information about its current capacity so that applications can send at high speed after only one or two round-trips.[16]

## 5. Conclusions

This paper has focused narrowly on the problem of achieving transfer speeds of one gigabit per second and has suggested that speeds of one gigabit per second are within reach of both datagram and virtual circuit architectures. It is perhaps easier to achieve this speed using virtual circuits (the reader may note that much of the flow control section was concerned with scaling datagram architectures to gigabit speeds), but either architecture is capable working at one gigabit, and thus the choice of datagram or virtual circuit architecture for a gigabit network is one of preference, not technological necessity.

The next question is whether these arguments still apply at higher speeds, say 3 gigabits or 10 gigabits per second. Good arguments can be made both pro and con. Simply scaling some of the numbers to 10 gigabits gives a scary set of requirements: a 600-MIPS CPU, switching times of a few hundred nanoseconds, and 150 megabytes of buffer space. Those requirements are sufficiently stiff to make one question the network architectures that force them on us. But doomsayers have a historically poor record in computer science. Five years ago, the shift from silicon to gallium arsenide was widely predicted. It hasn't happened yet. Time was that networking experts generally felt that TCP was a processing-intensive transport protocol. Van Jacobson has shown how to process a TCP segment in under 20 instructions. More generally, processors continue to get faster (last year SUN was reportedly predicting a 300-MIPS CPU in the early 1990s); memory continues to get bigger and cheaper, and parallel processing still waits to be fully exploited.

All this leads me to be optimistic and suggest that the architectures will scale up to higher speeds. Having concluded that one gigabit is not that fast, I find it hard to accept that a mere factor of ten change will make a big difference.

## Acknowledgements

Several people have helped me shape my thinking in this paper. Special thanks to Debbie Deutsch, Ira Richer, John Robinson, and members of the Internet End-To-End Services Task Force, which is chaired by Bob Braden, and the Internet Architecture Task Force, which is chaired by Dave Mills. Of course, I take full responsibility for any errors or misguided opinions.

[15] The exception comes when the network is being heavily used, in which case the level of service requested may not be available and the equivalent of a busy signal will result.

[16] Various mechanisms like those proposed for flow indicators could be used. Again, the basic datagram model doesn't break.

## References


[1] Anthony S. Acampora and Mark J. Karol, ‘‘An Overview of Lightwave Packet Networks,’’ *IEEE Network*, Vol. 3, No. 1, January 1989, pp. 29-41.

[2] David D. Clark, Van Jacobson, John Romkey, and Howard Salwen, ‘‘An Analysis of TCP Processing Overhead,’’ *IEEE Communications*, Vol. 27, No. 6, July 1989, pp. 23-29. Most of the computation in this report was done with the earlier version of the paper, published in the Proc. 13th LAN Conference (Minneapolis, 1988).

[3] Van Jacobson, ‘‘Congestion Avoidance and Control,’’ *Proc. ACM SIGCOMM ’88*, Stanford, Calif., August 1988, pp. 314-329.

[4] R. Jain, ‘‘On Caching Out-of-Order Packets in Window Flow Controlled Networks,’’ Tech-Report 342, Digital Equipment Corporation, January 1985.

[5] J. Nagle, ‘‘Congestion Control in IP/TCP Networks,’’ *ACM Computer Communication Review*, Vol. 14, No. 4, October 1984, pp. 11-17.