Data Networking and the Universal Speed Limit

Craig Partridge
BBN Systems and Technologies Corporation

1. Introduction

This note was motivated by the need to organize my thoughts for an INARC panel session on limits of the Internet architecture. In particular, among other topics I'd been induced to talk about very widely distributed file systems (''galatic filesystems'' in Dave Mills' terminology) and since file systems were not my area of expertise, I felt I'd better write down my thoughts and think them over before I said anything in public. This note is largely the accumulation of those notes. I want to emphasize that much of this note is simply thinking out loud on paper, and that I can (and probably will) change my mind on some points.

2. A Small Observation

The basic observation that motivates this note is that, in gigabit speed networks, the network delay will be largely a function of the distance between end-points, not delays in the routers or switches or the amount of data sent. The logic behind this observation is that to be able to keep up with potentially very small packets at very high speeds, routers and switches will have to be able to process packets quickly, in a few microseconds or less. But even at the speed of light, the delay across moderate distances is measured in milliseconds. Thus for delays in routers or switches to have an effect there must be hundreds or thousands of routers in a network path (or, potentially, queues of hundreds or thousands of packets on some routers). Such a large number of routers seems somewhat implausible.

Furthermore, as data networks get faster, their speed increase will come by packing more bits into a unit length of fiber, not by lowering the delay of the network (which in fiber networks is already the speed of light, the universal speed limit). Thus a given packet will take of less space in the fiber (thus forcing routers to process packets faster), but the first bit of the packet will take the same time to get from point A to point B that it did before.

Eventually (and I think eventually is sooner rather than later) we will find ourselves in the situation shown in Figure #1. Figure #1 shows the time to transfer *n* megabytes of data one way over a 10 gigabit link between two endpoints roughly three thousand miles apart. The transmission delay resulting from the distance between end-points dwarfs the delay due to packet size. It is also interesting to compare this graph with Figure #2, which shows the same transfer, but for a network where the two endpoints are a few hundred miles apart. Comparing the two figures leads me to believe that the key networking question for distributed systems, including file systems, will soon be not ''How much data needs to be moved?'' but ''How far does the data need to be moved?'' In other words, systems will become much less sensitive to the amount of data being transferred, and much more sensitive to how far away the data is.

3. Musing About Galatic File Systems

So how do these observations about transfer delay affect widely distributed file systems? I'd like to suggest two possible consequences:

- Distributed file systems will have place a premium on getting files close to the applications accessing those files.
- Distributed system designers will have to rethink their models of RPC.

The desire to get files close to the applications that use them should be clear. Reading on a per-block basis from a file that is three thousand miles away is going to be vastly more expensive than transferring the entire file to a cache on the local network and doing all operations locally. At the same time, I think that the advantages of diskless workstations (in particular low maintenance requirements) makes it clear

---

This note was written under DARPA Contract MDA903-89-C-0020. The opinions in this paper are those of the author 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.

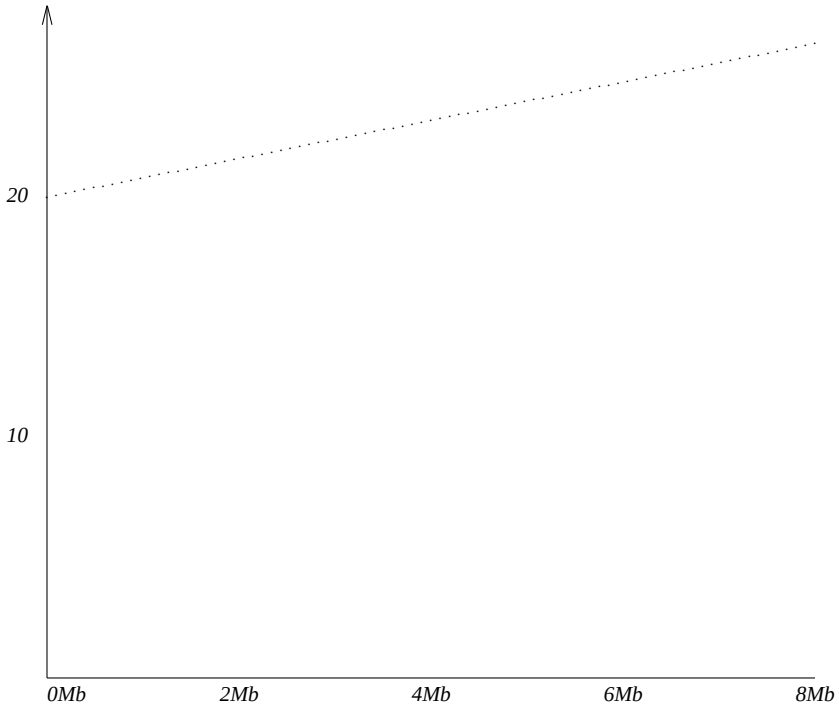

Figure #1: Long Distance Transfer Delay (in ms) vs. Data Size

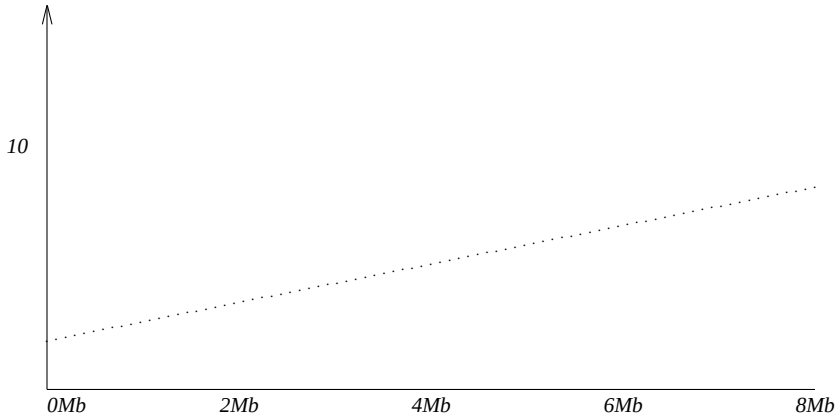

Figure #2: Transfer Delay (in ms) vs. Data Size (in megabytes)

that we don't expect every computer that participates in a galatic file system to have a disk. As a result, I suspect we'll see architectures where file system clients communicate with local servers that both maintain the local portions of the file system and also serve as gigantic caches for remote files that are currently in local use.

Local caching raises interesting consistency problems. What if there are two applications, say in California and Massachusetts, that want to share a file between them? How do they make sure they both see each other's writes? A consistency check across country will take at least 50 milliseconds (roughly 25 milliseconds each way). Or to think of that another way, the reading application is typically 50 milliseconds behind the writing application. I don't have any good ideas about how one might deal with this problem.

The consistency problems leads naturally to RPC models. They are flip sides of the same coin. Unless we repeal the universal speed limit, a cross-country remote procedure call will always take at least 50 milliseconds, no matter how fast the computers at each end run. As a result, remote procedure calls will get more expensive (measured in instructions spent waiting) as computers get faster. Personally, I always felt one of the benefits of RPC was that it gave a clean application interface to the network – the network is made to look like a procedure call. Now, one can see the possibility that a remote procedure call, especially a long distance RPC, will take dramatically longer than the average procedure call.

An oft-stated goal of some members of the distributed systems community is that the network should be transparent to the user. It is clear that, at least in the long run, it will not be easy to make the network transparent. Its topology will leak through, in the form of slower response times for long-distance remote procedure calls (or equivalent operations). If we are to make the network transparent I think we will have to find ways to anticipate applications' data requirements and use smart schedulers that either move the applications closer to their data (e.g. swap the application out to a machine which is physically close to the data it is currently using) or move the data closer to the applications (e.g. anticipating the data the application is likely to access and pre-loading it into a local cache).