Implementing TCP/IP on a Cray computer.

David A. Borman†

Cray Research, Inc.

ABSTRACT

Cray Research has been supplying TCP/IP networking code with its UNICOS‡ operating system since February 1986. In the last three years, this code has been improved and optimized, yielding up to two orders of magnitude improvement in performance. This paper addresses some of the issues that were addressed to achieve these changes, and some of the other changes that have been made to fit the TCP/IP code into a Cray environment.

A Brief History

In 1985, Cray Research entered into a contract with The Wollongong Group, for Wollongong to provide TCP/IP support for Cray Research's 1.0 release of the UNICOS operating system. For the original port to the Cray computer, Wollongong used their port of the 4.2BSD TCP/IP code to System V. After Wollongong had delivered a working product, Cray Research took over the support and further development of the TCP/IP package. Originally there were two source trees for the TCP/IP code, one for the CRAY-2 implementation of the UNICOS operating system, and one for the CRAY X-MP implementation of the UNICOS operating system. With the 3.0 release of the UNICOS operating system, these two source trees were merged into a single source tree. With the 5.0 release of the UNICOS operating system, the TCP/IP code has been upgraded to be compatible with the 4.3 BSD Tahoe release.

Cray Research's view of TCP/IP

A Cray computer is not cheap. Prices for Cray computers range between 2.5 million and 20 million dollars. When someone invests this kind of money, they want their Cray computer to be accessible to a large numbers of users. By having the TCP/IP protocol suite available on the Cray computer, it is possible to easily add the Cray system to an established TCP/IP network, providing instant access to the Cray system from any of the machines in the network. The corporate mission statement of Cray Research states that "Cray Research designs, manufactures, markets, and supports the most powerful computer systems available." With this in mind, it makes sense that Cray Research should also strive to have the most powerful implementation of TCP/IP available.

The initial implementation of TCP/IP on the Cray computer had some serious performance problems. Running a memory to memory copy over a TCP/IP connection between two processes on the

---

† David A. Borman, Cray Research, Inc, Networking and Communications, 1440 Northland Drive, Mendota Heights, MN 55120. Mr. Borman has worked at Cray Research since December of 1985, and is the project leader for the TCP/IP networking code that Cray Research releases with its UNICOS operating system. Prior to working at Cray Research, he worked for two years at Digital Equipment Corporation in Merrimack, NH, as a kernel developer for DEC's ULTRIX-11 product. He received his BA in Mathematics, with a concentration in Computer Science, from St. Olaf College, in May of 1983.

same Cray computer ran around six to seven megabits per second. When running between two machines, the performance was even more dismal, on the order of one to two megabits per second. Today, with the 5.0 release of the UNICOS operating system, when doing a memory to memory copy between two processes on the same Cray computer, it is possible to obtain TCP/IP transfer speeds in excess of a half a gigabit per second. The speed of a memory to memory copy between two Cray computers has been demonstrated close to two hundred megabits per second. This change of two orders of magnitude is due to many things. Improvements to the TCP/IP code, improvements to the base UNICOS operating system, improvements to the C compiler, and new hardware have all contributed to this increase in performance.

TCP/IP code changes

The TCP/IP code in the UNICOS operating system has undergone many changes in its 3 year history. There have been numerous bug fixes and minor modifications, and it is not really worth going into great detail about them. However, there are a few areas that are worth talking about.

The Checksum Routine

Perhaps the one area that has provided the most visible improvement in performance from revision to revision is the checksum routine, used by both IP and TCP. The original checksum routine, as delivered by Wollongong, was written in C, and was character oriented, meaning that the routine read the data to be checksummed from memory one byte at a time. The Cray computer is a sixty-four bit word addressable machine. Each word will hold eight bytes. A character pointer is stored as a word address, and a byte offset into the word for the character. The C compiler automatically takes care of this, but it means that walking through an array of characters a byte at a time will cause each word to be read from memory, shifted and masked eight times to extract the eight individual bytes. Needless to say, this is not optimal.

The first revision to the checksum routine was to rewrite it, still all in C, so that it would calculate the checksum a word at a time. This change alone caused the memory to memory copy test to go from 6 megabits per second to 20 to 30 megabits per second.

A second revision was to hand code in CAL (Cray Assembly Language) the section of it that added up the checksum a word at a time. This was done using scalar instructions. Hand coding it in CAL allowed memory wait states to be eliminated.

The last revision that was done was to take the CAL code, and rewrite it to a vectorized loop. This vectorized version is not used all the time, because of startup costs involved with using vectors in the kernel. For short segments of data, the scalar version is used, and for longer data segments, the vectorized version is used. On a CRAY Y-MP, with our latest version of TCP/IP, using the scalar version of the checksum routine provided a transfer rate of over 250 megabits per second. When the vectorized version of the checksum routine was enabled, the throughput shot up to over 550 megabits per second.

Obviously, the conclusion that can be drawn is that the checksum routine should be highly optimized for any implementation, because it can have a very profound effect on overall TCP/IP throughput.

Mbufs

Mbufs are the basic data structures used by the networking code for holding data and control structures. The original Berkeley implementation had mbufs that were 128 bytes long, with the possibility of pointing the mbuf to an external piece of memory 1024 bytes long. These were designed for use on a virtual memory machine, to facilitate page mapping. On a Cray computer, there is no virtual memory hardware. There are two registers that describe a process; one points to where the process starts, and the other contains how long the process is in words. The mbuf code has had several major modifications made to it to make it efficient on a Cray computer. Currently, there is an array of mbuf headers, each eight words long, and an array of data segments, each 1K bytes long. There is a one to one mapping between the headers and the data segments. In addition, several contiguous mbufs may be allocated together, to provide larger than 1Kbyte data areas. When mbufs are not in use, they are

returned to the free pool, which is an ordered linked list of all the partial mbufs. If an mbuf is freed adjacent to an already free mbuf, it is merged with it. Allocating a 32Kbyte mbuf can get expensive, because the free pool has to be searched for 32 mbufs that are contiguous, and then all 32 mbuf headers have to be linked together. To eliminate this overhead, a cache has been added to keep free lists of various commonly used sizes of mbufs. If a 32Kbyte mbuf is in this cache, it is no more expensive to allocate it than it is to allocate a 1Kbyte mbuf.

Variable MTU

The term MTU (Maximum Transmission Unit) is used to describe how much data can be sent in one piece over the networking hardware. The networking hardware used to connect to a Cray computer does not impose any limit on the size of an IP datagram, of which the maximum length allowed by the IP protocol is 65535 bytes. Practical lengths are imposed by the software. In a heterogeneous environment, different vendors may have different limits on sizes of data that they can send and receive. Cray Research has introduced the idea of a variable MTU, where the MTU of a given interface specifies how large of a packet the Cray computer is willing to receive on that interface, and the IP to hardware address mapping table for the interface has an MTU associated with each entry, to limit how large of a packet the Cray computer can send to each machine. With this scheme, the Cray computer is able to use large packets when connected to a machine that accepts large packets, and small packets when talking to smaller machines that can only handle smaller packets.

The changes in the mbuf code facilitate using large (more than 16Kbytes) MTUs to achieve performance improvement. Prior to adding the mbuf cache, using MTUs greater than 16Kbytes actually degraded performance, due to the overhead of allocating large mbufs. With the mbuf cache, using a 32K byte MTU improves throughput, as it should.

Windowing

One of the first things done to the Cray networking code was to increase the TCP window. The original 4.2 BSD code had only 2K of buffering; this was bumped up to 16K or 32K, depending on how much memory the Cray computer had. One area where the TCP protocol is lacking is that it is impossible to determine how much buffering the remote side has for sending data, i.e. how much data the remote side can send before it blocks waiting for some of the data to be acknowledged. A common strategy used in TCP is to delay ACKs to acknowledge several packets at once. The Berkeley code assumed that the remote side had as much buffering space for sending as the local side had. The Cray computer would get into situation where the remote side had sent 2K of data, and then would sit there waiting for an ACK, while the Cray computer sat there thinking "Well, 2K out of a 32K window, I'll not ACK the data just yet, because he can send a lot more data into my window." The net effect was that sending data to the Cray computer with a small send space was very slow. To resolve this problem, the networking code keeps an estimate of how large the remote sides sending space is, by keeping track of the largest amount of data ever sent between ACKs. Now, when the remote side has sent its 2K of data, the Cray computer will realize that that is probably all the remote side can send, and immediately send an ACK, rather than delaying it.

Porting Base

Originally, the code that was ported to the Cray computer was based on the 4.2 BSD release. In the summer of 1988, a re-port from the 4.3 BSD release was done, incorporating the recent work done by Van Jacobson and Mike Karels at U.C. Berkeley, most notably the "slow-start" code, congestion control code, and better round-trip time calculations. With the amount of excellent work being done in protocol development, it is silly not to make use of and build upon what is available.

The UNICOS operating system

Cray Research is about to release the 5.0 version of the UNICOS operating system. When the TCP/IP code was first implemented, it was with the 1.0 release of the UNICOS operating system. Many bugs have been found and fixed in the last 4 releases, many new features have been added, and many sections of code have been re-written to improve performance. Individually, these changes may

not make a large impact on TCP/IP performance, but when added up they definitely make a noticeable difference. However, since these changes have been spread out over a three year period, there are no hard figures on how much these changes account for the improvements in the TCP/IP performance.

Probably the first interesting bug in the operating system on the CRAY-2 computer that affected the TCP/IP code was that there was a small window in which interrupts would not be noticed, and would not be processed until another interrupt occurred. The networking hardware seemed to consistently hit this window. When timings were done to measure the effect of changing the data transfer size, an erratic, step like graph emerged. After isolating and fixing this delayed interrupt processing, when the same measurements were done a nice smooth curve emerged.

The C Compiler

During the last several years, the C compiler group at Cray Research has not been sitting still. The original C compilers did not have very well developed optimized code generation, and had no vector processing capability. The current C compiler has greatly improved code generation, and is also able to automatically recognize and vectorize many constructs. While there are no hard numbers available for how the compiler changes have changed the performance numbers of TCP/IP, one particular test application that did many small reads and writes to a TCP socket, had its overall execution time cut by over 40 percent when switching to a new version of the C compiler on a CRAY X-MP computer (both the application and the kernel were re-built with the new compiler).

Hardware Changes

Since the introduction of TCP/IP on the Cray computer, there have been improvements to the Cray computers themselves, and to the networking hardware available to attach to the Cray computer. The original port of the TCP/IP code, and the original measurements, were done on a single processor CRAY-2 computer with 16 megawords of memory, one of the original prototype CRAY-2 computers, and on a two processor CRAY X-MP, with a 9.5 nanosecond clock and two megawords of memory. Current CRAY-2 computers being built have had logic and memory improvements, so that the hardware itself will give an overall improvement of more than 30%. The CRAY X-MP EA computers now have an 8.5 nanosecond clock, up to 64 megawords of memory, and the newest machine, the CRAY Y-MP, has a 6.0 nanosecond clock, with 8 processors, and up to 128 megawords of memory.

Networking hardware has also improved. When the TCP/IP code was first ported, the only networking hardware was NSC HYPERchannel†, connected to a Cray computer's low speed channel. The low speed channel is rated at 50 or 100 megabits per second. The best performance that TCP/IP has gotten over HYPERchannel is around 12 megabits per second, using a 16Kbyte transfer size. NSC has recently introduced its HYPERchannel DX, and over this new hardware, Cray computers have been able to obtain transfer speeds of 23 megabits per second. Cray Research has also developed a high speed channel, HSX, which is rated at 850 megabits per second. TCP/IP transfer speeds of around 190 megabits per second have been demonstrated between two Cray computers connected with HSX hardware.

Other Features

Other features have been added to the networking code to make it more configurable and usable. The features listed here are some of the major features added that do not exist in the standard Berkeley releases.

Policy Routing

Currently Cray Research does not supply any routing daemons to run on the Cray computer. Routing is done through installing static routes at boot time, and automatically adding new routes due to receiving ICMP redirects. Because of the cost of Cray computer systems, many times the machine will be used by several different departments or companys.. In order to keep the Cray computer from becoming a path for people from one department or company to access another department or company,

† NSC and HYPERchannel are registered trademarks of Network Systems Corporation.

the ability to restrict who can use IP routes has been implemented. When a route is installed, it may be tagged to either exclude, or only include, certain groups. If a user is a member of an exclusive group list, or not a member of an inclusive group list, then when the kernel looks up a route to use for a connection by the user it will not allow the restricted route to be used. Additionally, individual routes can be marked so that the IP forwarding code will skip them.

Line Mode TELNET

The TELNET protocol has been typically implemented such that all the terminal character processing is done by the server machine, meaning that the user's data is sent across the network, and echoed back across the network, a byte at a time. Cray computers are number crunching machines, and were not designed to deal with lots of single character interrupts. To alleviate this problem, Cray Research has developed what is known as Line Mode TELNET. When running with Line Mode, all the character processing and character echoing is done on the client side of the TELNET connection, and only the completed line of text is sent to the remote machine. This is also very useful when using TELNET over a long delay network, because the user will get what he types echoed immediately, rather than having a many second delay between what is typed and what is echoed. The implementation allows for the server to dynamically switch between line mode and single character mode, so that things like visual editors will work properly.

Futures

Cray Research will continue trying to improve performance of the TCP/IP networking code. Currently the code is using less than one third of the potential bandwidth of the HSX channel, it is felt that there is room for improvement. Many third party vendors are coming out with hardware products that can connect directly to the Cray computer, giving Cray customers a wider choice of how to connect their Cray computer into their local networks. Cray Research is actively testing and verifying these products to make sure that they can stand up to the demands of Cray customer environments. Cray Research is also active in standards development, such as the Internet Engineering Task Force, helping to broaden the experience on which the standards are based. Cray Research is also actively exploring and addressing issues such as network management and network security. Many of the lessons learned from implementing TCP/IP are generic to implementing networking protocols, and can be applied to new protocols as they become available.

Conclusions

The following table summarizes some peak of the transfer rates that have been obtained during dedicated machine time. In all these tests, there was 512K bytes of buffering space on both sides of the connection, the TCP window was opened to a full 64K bytes, and the MTU was set to 32K bytes. The application that generated these numbers timed a memory to memory copy over the connection, no disk time is involved.

Machine(s)	Transfer Rate
CRAY-2 to CRAY X-MP over HSX	190 Mbits
CRAY-2 software loopback	310 Mbits
CRAY X-MP EA software loopback	400 Mbits
CRAY Y-MP software loopback	550 Mbits

Over the past three years, many things have changed to improve the performance of TCP/IP on the UNICOS operating system. The overall system performance, of the operating system, compilers, and utilities, has an effect on how well the TCP/IP code performs, and should not be ignored when evaluating the performance of any networking implementation. There are also areas of networking code that are specific to the machine that it is being implemented on, and these are some of the more obvious areas that should addressed when trying to improve the performance of any networking protocol. Much time and effort can be saved by cooperating and sharing code and ideas with other developers working on improving the same networking code.