M.I.T. Laboratory for Computer Science

Request for Comments No. 225
May 26, 1982

IBM Networking Progress Report

by Wayne C. Gramlich

1. Introduction

Within the past 6 months, the Computer Systems Communications group has obtained three IBM personal computers. A project was undertaken to integrate these computers into the M.I.T. local network environment. A previous document [Gramlich 81] discussed the initial ideas on how to accomplish this task. This document discusses progress that has been made to date, as well as future plans.

2. Progress to Date

The scheme that has been adopted to connect the IBM PC's to the M.I.T. network environment is to have a front-end processor. The front-end processor connists of some serial lines and at least one network interface. Each IBM PC is connected to one RS-232 serial line. The front-end processor that has been chosen is the DEC LSI-11. The primary reason for this choice is that the LSI-11 is the lowest cost processor for which network interfaces are readily availble. The software that runs on the LSI-11 is the "C-Gateway". The C-Gateway is a multi-protocol gateway that is being developed for interconnecting various networks. The C-Gateway is written in C and runs under the MOS operating system.

There are basically two ways of thinking about this front-end processor. First, the front-processor can be thought of as a dedicated processor whose only task is the connection of IBM PC's to the network. Second, the processor can be thought of a network gateway that has the additional

capability of connecting IBM PC's to the network. The second case is in some sense more cost effective because a single processor is being used to both interconnect several networks, as well as, connecting several IBM PC's to the network environment.

There are two basic packages that are necessary to connect the IBM PC's to the network environment. The first package provides the user with the ability to transfer files between the IBM PC's and remote hosts. The TFTP/UDP/IP [Sollins 81] protocol is to be used for this purpose, although eventually the BLAST/UDP/IP [Theriault 82] protocol will probably be used instead. The second package permits a user to log into a remote host via an IBM PC. The TELNET/TCP/IP protocol is to be used for this purpose, where the IBM PC will appear like a Heath H19 terminal to the remote computer.

The current status is that only the TFTP/UDP/IP protocol has been implemented to transfer files between two IBM PC's via a 9600 baud serial line. This implementation was part of a Bachelor's thesis by Karl Wright which has subsequently been released as an LCS Technical Memo [Wright 82]. In summary, it was possible to transfer a 20,000 byte file at 3500 bits per second over a 9600 baud line. Places where bandwidth was lost include:

- ~ Each 8-bit character is transmitted using 10-bits of bandwidth. This can consume up to 20% of the bandwidth, but if fact consumes only about 10% due to the fact that the line is not 100% utilized.
- ~ The protocol headers consume roughly 10% of the bandwidth.
- ~ The software delays and operating system overhead consume roughly 40% of the bandwidth.

Higher speed disks using winchester technology and higher speed serial lines should eventually permit the transfer rate to increase. This hypothesis has been tested by modifying the file transfer program to move a file completely in primary memory without the disk. Data rates of 10K bits per second were acheive with a 19.2K baud line.

3. Software Tools Development

A fair number of tools to support software development have been implimented:

- ~ A C Compiler and assembler that produces 8086 code has been ported over to the CSR PDP-11/45.

~ A linker has been implemented to produce files that are directly executable on an IBM PC.

~ A down-load program has been implemented to transfer executable files from the PDP-11/45 to an IBM PC.

~ A standard Unix-like I/O library has been implemented.

~ An interrupt driven I/O package has been developed.

~ A program that makes an IBM PC emulate an H19 terminal has been implemented.

A fairly large amount of effort has gone into the development of these tools.

Most of the initial project effort was expended on the development of software tools. It was decided to implement as much software in C as possible. Towards this end, a version of the portable C Compiler that was modified by Chris Terman to produce code for an 8086 microprocessor was ported from the RTS VAX/780 to the CSR PDP-11/45. The major problem encountered with the C Compiler was that the 8086 code generator was very new and contained some bugs. Thus, occasionally it was necessary to rewrite some portions of code that were semantically correct but for which the compiler produced incorrect code. This was a fairly minor inconvenience. The alternative to using the C compiler was to write code in IBM Pascal. An assembler was necessary since not all code could be directly written in Pascal. Since it was not clear when an assembler was going to be released, an implementation in Pascal was not seriously considered. In retrospect, the decision to go with the C cross compiler was a good one.

It was necessary to implement a new linker from scratch that ran on the PDP-11/45, since the VAX linker could not be easily ported from the VAX. The major advantage of the new linker is that it supports some features that would have been very difficult to support using the VAX linker. The primary example of such a feature is that the new linker directly supports a program profiling package to help decide where a program is spending most of its time. The major disadvantage of writing a new linker was that linkers are complicated programs and the new linker was no exception. For example, while the new linker did not initially support routine libraries, it became necessary to add code to the linker to support this feature. Thus, it has been necessary to dedicate man-power to maintain the linker that could have been directed to more useful activity.

It was necessary to implement a simple down-load program to transfer files from the PDP-11/45 to the IBM PC's. The down-load program consists two parts, a program that runs on the PDP-11/45 and a program that runs on the IBM PC. The part that runs on the PDP-11/45 was written in C. Initially, the part that ran on the IBM PC was written in BASIC. Everything worked very well, until for no apparent reason the BASIC program stopped working. It became necessary to write a second down-load program for both ends that was far less complicated in order to get the BASIC side working again. This cost almost an entire week of lost productivity. Shortly thereafter, the BASIC side was replaced with a program written in C.

It was necessary to provide a set of routines to permit a C programmer to access the file system of the IBM PC. It was decided to make this package look as similar as possible to the standard I/O package that is implemented by Unix. This package was a modified version of an I/O package developed by the author for his home computer. All of the commonly used routines in the standard I/O library have been implemented. The more complicated and less frequently used routines will be implemented as they are needed.

In order to connect the IBM PC's to the network environment it was necessary to have a package for interfacing to the serial I/O port. It was desirable to operate the serial I/O port in an interrupt driven mode in order to permit more concurrency. The documentation on the serial I/O port and interrupt system was completely inadequate. A large amount of experimentation was necessary before it was possible to get the interrupt driven serial I/O package working. Even now, it is not entirely clear why the current package works and other very slight variations on the package do not work.

The lack of terminals made it desirable to quickly implement a VT-52 terminal emulator program. The initial program was written in BASIC and had effective baud rate of 300 baud over a 9600 baud line. This program has been rewritten in C and now emulates an H19 terminal. It can run at 9600 baud without getting behind. This terminal emulator will be used as a piece of the TELNET/TCP/IP protocol.

About the only tool that is missing on IBM PC is a decent symbolic debugger. There is a version of DDT that is written in C that can probably be modified to run on the IBM PC's without too much difficulty. This would probably also require some modifications to the new linker to produce symbol tables that the debugger could access.

A large amount of effort has been expended developing software development tools for the IBM PC's. However, in the long run it is believed that the total amount of time spent developing software will have been less than if the time had not been expended developing the software tools.

4. Future Plans

The most critical task that remains to be accomplished is the addition of the interface code to the C-Gateway for the IBM PC's. The way this code is structured is to treat all of the IBM PC's as if they were attached to a single logical network with its own sub-network number. This interface code should permit an IBM PC to send a packet to 1) itself, 2) another IBM PC, and 3) another host on the network. The low-level protocol to be used over the serial line has already been specified. The interface specifications to the network are specified by the C-Gateway. The fact that both specifications exist should immensely simplify the design of this code.

To aid in the debugging of the C-Gateway inteface code, both a packet echo (ICMP) program and a name user program are being produced. A ping user is a program that sends a packet out to a specific internet address and a reply packet is directly sent back. The C-Gateway supports ping protocol. Ping protocol is about the simplest protocol that there is. The ping program will have a variety of debugging features in it to permit the dumping of both sent packets and received packets. The name user is a program to access a data base that maps names to internet addresses. This program will also have debugging facilities built into it.

The TIU (Terminal Interface Unit) is an LSI-11 with both serial lines and a network interface used for connecting terminals to a network via TELNET/TCP/IP. This PDP-11 assembly code has been written and debugged at SRI. As an interim measure, the TIU will be used to connect the IBM machines to the network for remote login.

As was mentioned earlier, it is necessary to implement a version of TELNET/TCP/IP to permit users to perform remote login. This summer the TELNET/TCP/IP program will be worked on by Lou Konopelski. The terminal emualation package has already been developed for this program. The C-Gateway will be critical to debugging of this program. In addition, it is quite possible that the serial line will introduce performance bottlenecks. This may necessitate the migration of some of the functionality of the TCP/TELNET program over to the C-Gateway. Performance measurements will help determine whether this will be necessary.

In the internetwork environment, it is possible for one host to send a packet and have that packet get fragmented into several pieces on its journey to its destination. When these fragments arrive at the destination it is necessary to reassemble them into a single packet. This code has yet to be implemented for the IBM PC. In the implemention of reassembly care is going to be taken to try to minimize the amount of copying.

As soon as TFTP works with the C-Gateway, it will be possible to implement a couple of programs for printing files from the IBM PC. The easiest program will be a program that spools a file on the line printer connected to the CSR PDP-11/45. This simply requires TFTP to spool the file to be printed and a request file over to the PDP-11/45. It would also be desirable to be able to print files on the Dover. The easiest way to accomplish this would be to implement a server on the CSR PDP-11/45 that converts an ascii text file to a press file suitable for printing on the dover. The software for converting an ascii file to a press file already exits on the PDP-11/45.

As was mentioned earlier, it is suspected that TFTP will eventually be superseded by the BLAST protocol. The BLAST protocol is a protocol that does not have any acknowledgements sent back to the sender until the entire file has been transmitted. It is suspected that this protocol will be at least twice as fast as TFTP (but not necessarily with the IBM PC's).

It would be nice to be able to send and receive mail with an IBM PC. An initial version of this program has been implemented by Michael Patton as part of a Bachelor's thesis [Patton 82]. It is hoped that this program will be expanded upon sometime during the summer. In particular, the user interface needs to be improved.

5. Conclusions

A significant amount of progress has been in the IBM networking project. A large number of tools have come into place. Not as much actual networking code has actually been generated, but more is expected shortly. A version of TFTP has been implemented that achieves a through-put of 3500 bits per second.

References

[Gramlich 81]
Wayne C. Gramlich.
A Project to Explore the Neworking of Low-Cost Personal Computers.
RFC 215, MIT LCS, December, 1981.

[Patton 82]
Michael A. Patton.
Integrating Disconnected Personal Computers into anElectronic Mail System.
thesis, MIT, May, 1982.

[Sollins 81]
Karen R. Sollins.
The TFTP Protocol (Revision 2).
IEN 133, MIT LCS, June, 1981.

[Theriault 82]
Dan Theriault.
BLAST, an Experimental File Transfer Protocol.
RFC 217, MIT LCS, March, 1982.

[Wright 82]
Karl D. Wright.
A File Transfer Program for a Personal Computer.
TM-217, MIT/LCS, April, 1982.