Strategies for Operating Systems in Computer Networks

Robert M. Metcalf, MIT Project MAC

Abstract

Experience with the impact of the ARPA Computer Network on attached operating systems is briefly discussed. Early indications of good strategy in network operating systems design are reported.

KEY WORDS AND PHRASES: computer networks, operating systems design, interprocess communication.

CR CATEGORIES: 3.81, 4.39.

Introduction

The communications subnet of the ARPA Computer Network has been increasingly operational since late 1969. A significant amount of operating system programming has been invested in joining computers to the subnet and in providing the basic tools for utilizing remote computer resources. It seems appropriate at this time to examine the interaction between the ARPANET and the various operating systems to which it is connected -- to look for preliminary indications of what constitutes good strategy in network operating systems. Our experience has shown that computer networks will have a dramatic effect on how we organize and use computing resources.

ARPA Computer Network

The ARPA Computer Network (ARPANET) is a transcontinental, message-switching, store-and-forward, digital communications network joining more than twenty computer sites <1, 2, 4, 5, 7, 8>. At each ARPANET site is an Interface Message Processor (IMP), i.e., a switching node of the communications subnet <2, 8>. Each IMP supports from one to four HOST computers. A HOST is typically an interactive time-sharing system (e.g., Multics, Tenex, ITS), a relatively powerful number-cruncher (e.g., IBM 360/91, CDC 6600), or a mini-computer (e.g., DDP-316, PDP-11) providing terminal access to distributed ARPANET resources <8>. Constructed under the direction of Dr. Lawrence G. Roberts of the Advanced Research Projects Agency (ARPA), the ARPANET has made significant progress in promoting resource sharing in the nationally distributed ARPA research community.

In connecting a computer to the ARPANET it is necessary (1) to construct a bi-directional, asynchronous, bit-serial interface between the computer and its local IMP <2, 6>, (2) to write a Network Control Program (NCP) providing logical communication paths among user processes on the computer and others on remote computers <5, 10>, and (3) to code specific function-oriented programs to provide system services on the ARPANET via NCP communication paths <7, 11>.

Experience has shown that (1) system programmers are surprised to find that an IMP interface, with attached ARPANET, is an autonomous device requiring HOST cooperation, (2) an NCP is a system module which as yet has no real home in an operating system, and (3) interprocess communication, even within a single operating system, is still difficult business. It is not that IMPs, NCPs, and interprocess communication are troublesome or a waste of time, but rather that these essential components of computer communications systems are deserving of careful consideration in the design of computer operating systems.

IMP Interfaces

An IMP interface, with attached ARPANET, is a device unusual in its insistence that the relationship between it and its HOST is symmetrical. A HOST computer likes to treat its devices with a certain paternalism -- in a master-slave relationship. A HOST is naturally concerned with the continued well-being of its IMP, but is somewhat put off by the IMP's reciprocating concern. Treating the ARPANET as if it were a reader/punch-like device is much like wagging a rabid dog by its tail.

Network Control Programs

Different approaches have been taken in the implementation of various ARPANET NCPs. On the MAC Dynamic Modeling/Computer Graphics (DMCG) PDP-10, for example, the NCP is an

integral part of the Incompatible Time-sharing System (ITS) <13> at the same level as disk and teletype handlers <9>. On Multics, the current NCP implementation is one in which NCP functions are distributed -- from hard core where device interrupts are dispatched, to user rings where subroutines handle user-dependent NCP functions, to a "network daemon" (i.e., a background job) which manages user-independent activities in general support of ARPANET communications. On the Lincoln Laboratory IBM 360/67, the NCP runs on its own virtual machine under CP/67. At UCSB, the NCP occupies a partition under OS/MVT on a 360/75.

In the ARPANET's early stages when NCP specifications were threateningly tentative, implementers placed a high premium on modularity and maintainability in NCP construction. The differences between implementations can be described in terms of the differing techniques in each of the systems for maintaining modular, modifiable system code. The similarities among NCP implementations (below) are important in that they suggest essential features of the impact of networking on computer operating systems.

The ARPANET can best be understood in terms of its levels of communication protocol. There are protocols among IMPs, among HOSTs, and among remote user processes (i.e., jobs). One advantage of the explicit structuring of protocol into levels is that errors can be detected where the potential for timely recovery is high. Errors at the IMP level are usually transmission errors routinely corrected by retransmission. At the NCP level, errors are likely to be protocol violations made by malfunctioning NCPs. User processes must deal with user program bugs, user protocol inadequacy, and (possibly) unrecoverable errors from lower protocols.

As of this writing, it can be said that NCP implementations have been very casual about error handling. Few HOSTs are prepared to retransmit a message when notified by their IMP of the rare need to do so. Even the inadequate error control features of ARPANET HOST-HOST protocol are largely ignored in current implementations. The approach taken so far has been to abort all of a HOST's ARPANET communications at the drop of a hat, rather than attempt to make a possibly graceful recovery. Fortunately, the occurrences of errors have been sufficiently infrequent to make such unforgiving implementations tolerable. But, as traffic increases, so too will errors. As substantive use increases, so too will error intolerability.

The reason for this observed approach to error handling in NCP implementations can be traced to the fact that centralized operating systems and their implementers do not expect to treat errors as a matter of routine. There is so little potential for error recovery in centralized computing environments that operating systems do not lose significantly by responding to an otherwise minor malfunction with an immediate crash. In computer networks, because of their dependence on the cooperation of processes created by different people in different places at different times, errors are not the exception, but the rule. Distributedness and redundancy in computer networks, however, significantly increase the potential for error recovery. Network operating systems will need to become more failure tolerant, more robust -- more resilient.

Another observation we can make about the ARPANET and its operating systems is that out of their interactions come daemons -- i.e., apparently userless, often consoleless, lurking background processes with responsibility for recurring events in system services. In almost every NCP implementation we find at least one network daemon. The network daemon is usually given responsibility for managing HOST-HOST protocol and the flow of data through the IMP interface. The ubiquity of these daemons is the result of the aforementioned emphasis on modularity and is an indication of an important facet of network operating systems design.

At the heart of operating systems there are what we call "embedded daemons" which handle many of the basic system functions. There is usually, for example, an embedded daemon handling the arrival of disk buffers destined to fill user requests. In the MAC DMCG and Harvard PDP-10 systems, the NCP is just such an embedded daemon. The characteristics of embedded daemons are (1) that their programs are wired down and intimately intertwined with other supervisor code, and (2) that their scheduling is usually done in priority interrupt hardware at small expense. The proliferation of non-embedded daemons in NCP implementations has bought improved modularity and maintainability, at the cost of scheduling (in software) "user" processes to perform frequent "system" functions.

Networks will continue to increase the demand for distributed multi-process activities and will continue to call for efficient daemon handling. Indications of increased daemon requirements in the ARPANET are already to be seen in the planning for implementation of a distributed file system for the ARPANET <12>. Daemons seem to be a fundamental structure of distributed computing. A good strategy, then, will be to design operating systems in which nonembedded daemons (processes) can function efficiently.

Interprocess Communication

Networking is interprocess communication. In interfacing computer systems to the ARPANET, good interprocess communication in the existing operating environment made implementation easier.

Interprocess communication is a controversial subject in computer operating systems. Managing asynchronous processes over a shared data base is a singularly important difficulty in building large operating systems. Let us assert that much of the unmanageability of cooperating asynchronous processes is removed when the processes involved are constrained to cooperate as if remote, i.e., as if joined only by a thin wire <14>. Under this constraint, cooperating processes must deal with each other via explicit data exchanges over controllable communication paths using well-defined protocols. There is evidence in the ARPANET experience to suggest that operating systems should be constructed using clean "thin-wire" communications interfaces among cooperating processes <14, 15, 16>. We propose that good strategies for distributed interprocess communication be used more generally in computer operating systems (1) because they have a clarifying effect on the management of multi-process activity and (2) because they generalize well as operating systems themselves become more distributed <14>.

For some time NCP implementers believed that ARPANET interprocess communication could be treated in the same way as device or file manipulation. We have discovered that such treatment, though convenient in specific cases, is inadequate in general. Interprocess communication in a distributed environment is more susceptible to malfunction and random timing than other resources in an operating system. Processes need more controls on such communications to deal effectively with their vagaries.

Experience has shown that it is good strategy to make it possible for a process to avoid blocking, i,e,, to avoid unintentional dependence on unpredictably long external delays. If a process has important work to do while waiting for long-distance communications, he must be able to tell, before committing himself, whether issuing a particular communication request will block him. This is accomplished in some systems by permitting processes to test for blocking conditions. In other systems, an argument to sensitive system calls specifies whether the caller should be blocked waiting for some external intervention and, if so, for how long before timing out. By giving such facilities to processes, we can substantially improve the returns to network parallelism and resilience.

Summary

We have had some experience with the impact of computer networks on operating systems. Our experience indicates that this impact will be important. In taking a preliminary look at the characteristics of this impact, we conclude that it will be good strategy in network operating systems (1) to make them more resilient, better able to detect and recover from component malfunction, (2) to be more supportive of nonembedded modular daemons for distributed multi-process activities, and (3) to make "thin-wire" interprocess communication an underlying structure of operating system organization.

Acknowledgements

Work reported herein was supported in part by Project MAC, an M.I.T. research project sponsored by the Advanced Research Projects Agency, Department of Defense.

References

<1>. L.G. Roberts, B.D. Wessler, "Computer Network Development to Achieve Resource Sharing", AFIPS Conference Proceedings, May 1970.

<2>. F.E. Heart, et al, "The Interface Message Processor for the ARPA Computer Network", AFIPS Conference Proceedings, May 1970.

<3>. L. Kleinrock, "Analytic and Simulation Methods in Computer Network Design", AFIPS Conference Proceedings, May 1970.

<4>. H. Frank, I.T. Frisch, W. Chou, "Topological Considerations in the Design of the ARPA Computer Network", AFIPS Conference Proceedings, May 1970.

<5>. C.S. Carr, S.D. Crocker, V.G. Cerf, "HOST-HOST Communication Protocol in the ARPA Netowrk", AFIPS Conference Proceedings, May 1970.

<6>. "Specifications for the Interconnection of a HOST and an IMP", Bolt Beranek and Newman Inc., Report No. 1822, February 1971.

<7>. S. Crocker, J. Heafner, R. Metcalfe, J. Postel, "Function-Oriented Protocols for the ARPA Computer Network", AFIPS Conference Proceedings, May 1972.

<8>. S.M. Ornstein, et al, "The Terminal IMP for the ARPA Computer Network", AFIPS Conference Proceedings, May 1972.

<9>. R.D. Bressler, "Interprocess Communication on the ARPA Computer Network", MS Thesis, Civil Engineering, M.I.T., June 1971.

<10>. J. Newkirk, et al, "A Prototypical Implementation of the NCP", ARPA Network Working Group (NWG) Request for Comments (RFC) No. 55 (RFC #55), June 1970.

<11>. T. O'Sullivan, et al, "TELNET Protocol", RFC #158, May 1971.

<12>. A.K. Bhushan, et al, "The File Transfer Protocol", RFC #265, November 1971.

<13>. D. Eastlake, et al, "ITS 1.5 Reference Manual", M.I.T. Project MAC Artificial Intelligence Memo Number 161A, revised June 1968.

<14>. R.M. Metcalfe, "Strategies for Interprocess Communication in a Distributed Computing System", Proceedings of the Conference on Computer-Communications and Teletraffic, Polytechnic Institute of Brooklyn, 1972.

<15>. R.D. Bressler, D. Murphy, D.C. Walden, "A Proposed Experiment with a Message Switching Protocol", NIC #9926, RFC #333.

<16>. D.C. Walden, "A System for Interprocess Communication in a Resource Sharing Computer Network", Communications of the ACM, Volume 15, Number 4, April 1972.