1990

The effects of high speed networking on the operation of a distributed system software design paradigm

Schneidenwind, Norman F.

The Effects of High Speed Networking on the Operation of a Distributed System Software Design Paradigm

Norman F. Schneidewind

Code 54Ss
Naval Postgraduate School
Monterey, CA 93943

Abstract

The effect of high speed networking on the operation of a previously developed paradigm [1] for the system and software design of distributed systems is presented. The effect of high speed on the several design principles is discussed. The following aspects of design are discussed: use of objects, use of generalized modules, control of module interaction, separation of control and data, and management of state information.

Introduction

We explain why some of our design principles are even more important to use in a high speed environment. In addition, we point out the modifications that must be made to the paradigm to account for high speed. Finally, we note that many design principles are independent of speed.

We examine a previously developed distributed system and software design paradigm [1] in the light of emerging high speed networks and identify those design principles of the paradigm that must be modified in order to accommodate high speed. In addition we identify those design principles that are independent of speed. Our objective is to produce a revised paradigm that will be applicable to high speed networks.

High speed networks and protocols are subjects of considerable current interest. One approach to the development of high speed networks involves focusing on level of performance, efficiency, scalability, and diversity of applications [2]. Another approach looks at an internet model consisting of connection-oriented protocol, a more flexible addressing scheme, and subnetwork descriptive information for making intelligent routing decisions [3]. These approaches seem to have more to do with increasing network functionality and efficiency than with speed per se.

Another view is that achieving gigabit-per-second networks is well within the reach of current network architectures [4]. This is in line with the thesis that TCP in the Internet is not the primary source of overhead (the host OS is) and that TCP could support high speed if implemented properly [5].

Objects

Representation

A complex system, like a distributed computer network, requires a model for identifying, relating, structuring and controlling system objects. We use objects as a means of representing system entities so that the representation of resources and

0-8186-2088-9/90/0000/0161/$01.00 © 1990 IEEE

processes can be completely general and become specific only at the point of naming, addressing or invoking. This method of representation provides for the latest possible binding of objects to physical resources [6] and for mobility of objects across physical nodes, since the identification of entities is not tied to physical nodes nor to host addresses. **The use of this method of system representation is independent of network speed.**

Named

We use named objects to represent system and software entities so that users can have location-independent access to objects. Thus users are not constrained in their access to resources by host identity or geographic location. Instead, users are free to access resources by type of service. **This attribute, which is desirable from the user's standpoint because he can access network resources by name or subject rather than by host numeric address, is feasible in high speed networks; it is not feasible in low speed networks. A low speed network does not have the speed to handle the overhead associated with additional message transfer generated by the need to look up host numeric address when presented with name or subject.**

Function

We associate a major function with each object and specify objects and functions by the services they perform. An object and its associated major function can be conveniently specified by the major service which the object performs for a process and by the relationships which exist between this object and other objects which assist in providing the service. **The act of relating objects to functions and services is independent of network speed.**

Response

We design objects so that they always respond the same way to the same stimuli. That is, an object should not respond one way at one time to a given input (e.g., a message to be processed FIFO) and another way at another time to the same input (e.g., same message to be priority processed), even though network operating conditions may have changed (e.g., congestion in the network). Any differences in instantiations of the objects, given the same inputs, requires complex program development and maintenance and does not allow for code sharing. In the example, if the message is handled in a special way due to performance considerations (e.g., priority message handling), this capability should not be incorporated in the many server modules of the network. Rather, the exception handling conditions should be placed in a single interface module. If this is not done, exception handling conditions would have to be incorporated in every module rather than in one module. If exception handling changes in the future, many modules would have to be changed rather than one. **Although it is true that, from a performance standpoint, this consideration is not as important in a high speed network, it is still important for software development and maintenance, as noted above.**

Generalized Modules

We specify generalized and independent functional modules

(i.e.,server modules, such as terminal management) for performing all non-unique application processing in order to: (1) reduce redundancy of effort and documentation in module development, (2) reduce the redundancy of storage space and execution time during network operation, and (3) confine the effects of changes to the software to a small number of standardized modules. This approach is important in distributed system design because of the large geographic dispersion of distributed systems. **This approach has a dramatic effect on reducing redundancy of resources and confining effects of change but is independent of network speed.**

Module Interaction

We minimize module interaction (i.e., elaborate handshaking message exchanges) by using message communication instead of the remote procedure call (RPC) so that reduced software development and maintenance effort and decreased computer execution time and communication channel transmission time will result from a reduction in the number and types of messages flowing in the network. This result follows from the fact that a reduction in number of control messages will reduce overhead time. In addition, a reduction in types of control messages will reduce software development and subsequent maintenance. **This is another design principle that is independent of network speed. True, as before, high speed will help mask the inefficiencies of network communication, but software development and maintenance will still be aided by a reduction in the number of message types.**

Control

Separation of control data from user data assists maintenance: changes in control data procedures and message formats will not affect user data protocols and vice versa [1]. This is accomplished by using separate logical data links and ports for each. Since control functions must be exercised prior to data transfer (e.g., handshake before data transfer), higher priority must be assigned to control (i.e., higher priority data links and ports). **In a high speed network the need for separation and prioritization is acute so that the processing and transmission of control data does not become a bottleneck to the transmission of user data.**

State Information

We use system state information as soon as it becomes available for the purpose of planning resource utilization as far in advance as possible in the execution of system operations so that unnecessary message exchange and module interaction can be avoided. We confine knowledge of transaction state information (i.e., status and progress of a transaction) to a single module. By this method we hide major design decisions and, in addition, hide important control information during system execution, as well. **This is another example of a design principle appearing to be less critical in a high speed network because greater speed would lessen the need for advanced planning in the allocation of resources. However, the need to confine information that is likely to change to a limited number of modules (e.g., one) is independent of speed.**

Conclusions

Although it might appear that radical changes would be necessary to adopt a distributed software design paradigm to high speed networks, in fact there is little that would be changed. This is due to the fact that the overriding consideration is the use of design principles that enhance the development and maintenance processes. These are largely independent of speed. The results of this analysis are summarized in Table 1.

Table 1

Principle	High Speed Effect
Objects	
Representation	No
Named	**Yes**
Function	No
Response	No
Generalized Modules	No
Module Interaction	No
Control	**Yes**
State Information	No

References

[1] N. F. Schneidewind, 'Distributed System Design Paradigm with Application to Computer Networks', IEEE Transactions on Software Engineering, Vol. 15, No. 4, April 1989, pp. 402-412.

[2] Gurudatta M. Parulkar, 'The Next Generation of Internetworking', ACM Computer Communication Review, Vol. 20, No.1, January 1990, pp. 18-43.

[3] Gurudatta M. Parulkar and Jonathan S. Turner, 'Towards a Framework for High-Speed Communication in a Heterogeneous Networking Environment', IEEE Network, Vol. 4, No. 2, March 1990, pp. 19-27.

[4] Craig Partridge, 'How Slow is One Gigabit Per Second', ACM Computer Communication Review, Vol. 20, No.1, January 1990, pp. 44-53.

[5] David D. Clark, Van Jacobson, John Romkey and Howard Salwen, 'An Analysis of TCP processing Overhead' IEEE Communications, Vol. 27, No. 6, June 1989, pp. 23-29.

[6] Richard W. Watson, 'Identifiers (naming) in Distributed Systems', Distributed Systems-Architecture and Implementation, B.W. Lampson, et. al., (eds.), Springer-Verlag, 1981, pp. 191-210.