Transport Issues in the Network File System

Bill Nowicki

Sun Microsystems
March 8, 1989

The Sun Network File System (NFS) is a popular protocol to access files across a network [Sandberg 86]. NFS is implemented on machines as different as Personal Computers, and Cray-2 supercomputers. It uses Sun Remote Procedure Call [Sun 88] and External Data Representation [Sun 87] specifications, which can be used on a variety of transport protocols. The original design goal for NFS was to support small clusters of machines on local area networks. Thus a datagram protocol (UDP) was chosen to simplify transport.

Using a connection-less transport protocol meant that nothing explicitly needed to be done to handle server or network failures. However, the use of NFS has expanded to cover much larger and more complicated networks than was ever anticipated. This has caused some NFS users to experience the problems of congestion and flow control that transport protocols have been designed to address. This paper discucsses some alternatives for NFS transport protocols. Other issues such as inter-domain authentication, are not discussed here.

1. Why Not Use TCP?

One approach would be to use an existing connection-oriented transport protocol such as TCP [Postel 81] instead of UDP. Some people [Chesson 87] claim that the overhead of TCP would reduce the throughput significantly, but experiments have shown that TCP overhead is not significantly different than that of UDP [Clark 1988].

On the other hand, connection management code would need to be added to handle server and network failures which would add complexity to both the client

---

Author's address: 2550 Garcia Avenue, Mountain View CA, 94043. Internet: nowicki@Sun.COM. This paper presents preliminary research results, which may be published more formally at a later time. It is not a product announcement or commitment.

guidance for chosing these values; the result was that the defaults intended for local networks were used in almost all cases, even over long-haul networks.

A first step to the solution was to add retransmission timers using the algorithms that have evolved for effective TCP congestion control. [Jacobson 88]. These estimate the round-trip times and their deviations, with only a few shifts and adds per transaction. Only two words of storage per timer are needed to store these values on the client. No modifications at all were required on the server, so it could be done totally transparently.

Since we are now timing not only the network round-trip time, but also the service time at the file server, the times are less well behaved than for a pure transport protocol like TCP. For example, Read and Writes are the only kinds of requests with large amounts of data in them, and can have very different response times depending on the speed of the disk, locality of reference, amount of memory for caching, etc. The solution is to implement the round-trip estimation algorithm multiple times, with an overall estimate as well as estimates of the three different kinds of operations (Reads, Writes, and all others).

4. Transfer Size Adjustment

In addition to adaptive timers, congestion control requires adjustment of the size of each request. This is complicated in NFS since most requests and responses are small, except for read responses and write requests. Luckily, the sizes of both of these are determined by the client, so again inter-operability can be preserved by making simple modifications to the client to allow dynamic transfer size adjustment.

Unfortunately deciding on the transfer size policy was very difficult. The problem is distinguishing a slow server from a congested network. If the server is slow or busy, sending many small requests instead of fewer large ones will be counter-productive, causing even greater load on the server. On the other hand, a congested network requires smaller transactions so that each one is separately acknowledged (unless selective retransmission is used).

5. Some Experiments

With dynamic retransmission timers and transfer sizes, an experimental NFS implementation was able to adapt to networks that vary in speed from 100 Mbits/second to 9.6 Kbits/second. On the slow networks the transfer size is properly adapted downwards to the minimum limit. Further experiments were performed over the Arpanet, using NFS to read the "Request For Comments"

and server implementations. NFS allows any number of requests to be outstanding simultaneously, and responses can be returned in any order. This property provides high performance when, for example, one request can be satisfied from a cache, even before another request that arrived previously but requires actual disk access can be satisfied. The TCP connection management sublayer could handle the multiplexing of a single connection among several simultaneous requests, but this would add to complexity and cause inefficiency when packets are lost. The strict ordering of TCP is not necessary.

One primary concern with NFS was for inter-operability. Obviously clients that used TCP could not communicate with servers that used only UDP, and vice versa. One approach would be to have converters that would translate NFS requests from UDP to TCP and back again, but this would require some complicated policy to set up connections and recover from failures. Note that idempotency of NFS operations would still be required if TCP were used. The fact that a request was received and acknowledged at the transport level does not imply that the request was actually performed at the NFS level -- the server could crash either before or after the operation was performed, and the two situations cannot be distinguished by the client.

2. How about VMTP?

Other transport protocols have been proposed for the support of ''transaction-oriented protocols'' such as NFS. One of these is the Versatile Message Transaction Protocol [Cheriton 88]. However, using a protocol like VMTP would have the same drawbacks given for TCP: strict ordering is not required, and inter-operability would suffer. There were also no stable implementations of VMTP at the time of this work. VMTP has since been ported to SunOS, so this option should be re-examined, along with consideration of other transaction-oriented transport protocols.

3. UDP with Timers

The original NFS implementation used fixed retransmission timers with exponential back-off, but the base values had to be set per file-system by a system administrator. Measurements showed that a significant fraction of certain kinds of requests (such as Write operations, which normally require an actual disk access) were retransmitted. A later implementation included a table of scale factors, so that the retransmission time would be scaled by a time dependent on the kind of operation. This worked much better on local networks, but had the same congestion problems over larger networks. System administrators had little

and other information that are available on the SRI Network Information Center machine. Most Arpanet links are 56 Kbits/second terrestrial circuits.

Finally, a file system from the University College, London, was mounted using NFS across both Arpanet and the Satnet, a trans-atlantic sattelite network [Seo 1988]. The NFS implementation did work, although performance was very poor due to the long delays. Because the read-ahead performed by the NFS client is done on a logical block basis, but the transfer size adapted down to a smaller amount, the protocol degenerated into stop-and-wait, which of course does not perform well on slow networks. On the other hand, the smaller packets make it less likely that the slow network will experience congestion, allowing other traffic (such as TCP-based mail transfer, probably the most common use of such long-haul networks) to use the network more effeciently.

6. Conclusions

Transport protocol issues cannot be ignored, even for protocols like NFS that are not connection-oriented. Protocols like TCP and VMTP should be considered in the future, but substantial improvements can be obtained through the use of better timer algorithms and transfer size adjustment to avoid congestion, while remaining inter-operable with current implementations.

References

[Cheriton 88]

D. Cheriton, ''VMTP: Versatile Message Transaction Protocol'', RFC 1045, SRI Network Information Center, Menlo Park, CA, February 1988.

[Chesson 87]

G. Chesson, ''Protocol Engine Design'' in *Proceedings Summer 1987 USENIX Conference*, Phoenix, AZ, June 1987.

[Clark 88]

D. Clark, J. Romkey, and H. Salwen, ''An Analysis of TCP Processing Overhead'', in *Proceedings of the 13th Conference on Local Area Networks*, Minneapolis, MN, 1988.

[Jacobson 88]

V. Jacobson, ''Congestion Avoidance and Control'', in *Proceedings of the*

SIGCOMM 88 Symposium on Communication Architectures and Protocols, Stanford CA, August 1988.

[Postel 81]

J. Postel, "Transmission Control Protocol - DARPA Internet Program Protocol Specification", RFC 793, SRI Network Information Center, Menlo Park, CA, September 1981.

[Sandberg 86]

R. Sandberg, D. Goldberg, S. Kleiman, D. Walsh, and B. Lyon, "Design and Implementation of the Sun Network Filesystem", in *Proceedings Summer 1985 USENIX Conference*, Portland OR, June 1985.

[Seo 88]

K. Seo, J. Crowcroft, P. Spolling, J. Laws, and J. Leddy, "Distributed Testing and Measurement across the Atlantic Packet Satellite Network", in *Proceedings of the SIGCOMM 88 Symposium on Communication Architectures and Protocols*, Stanford CA, August 1988.

[Sun 87]

"XDR: External Data Representation Standard", RFC 1014, SRI Network Information Center, Menlo Park, CA, June 1987.

[Sun 88]

"RPC: Remote Procedure Call Protocol Specification", RFC 1057, SRI Network Information Center, Menlo Park, CA, June 1988.

[Sun 89]

"NFS: Network File System Protocol Specification", RFC 1094, SRI Network Information Center, Menlo Park, CA, March 1989.