Capacity Testing A HYPERchannel-Based Local Area Network*

W. Bruce Watson

Lawrence Livermore National Laboratory Livermore, California 94550

ABSTRACT

Given the amount of resources they have invested in their local area network, network managers at Lawrence Livermore National Laboratory have an abiding if not critical interest in determining its remaining life span. Using a hardware monitor, researchers at Livermore have tested the capacity of their mature (over 10 years old), heavily loaded (5 Crays, 3 Ethernet gateways, 13 terminal concentrators, 5000 terminal, 40 node) two-trunk, HYPERchannel based, high data rate local area network. Results indicate that the current load could be intensified by 40% before the network reaches saturation. Further, as the load approaches saturation, overly persistent, low-level protocols begin thrashing, effectively destabilizing the network and destroying much of its potential capacity. Finally, it is clear what steps need to be taken to extend its life span and improve its performance under heavy loads.

INTRODUCTION

System designers at the Lawrence Livermore National Laboratory (LLNL) are designing, implementing, and debugging a new, distributed, client/server, message passing operating system, the Network Livermore Time-Sharing System (NLTSS) [1-6]. The underlying communications software and hardware comprise one of the structures that play a critical role in overall system performance. To aid in achieving optimum network performance, systems designers at LLNL have acquired a hardware monitor for their Network Systems Corporation (NSC), HYPERchannel-based, high data rate local area network. This monitor, the HYPERtools Inc., Enhanced HYPERchannel Monitor Device (EHMD), copies the headers of all frames traversing every HYPERchannel trunk into internal buffers. Simultaneously, the EHMD's 80386-based processor analyzes the data in these buffers to reveal the network's performance, e.g., message size, delay and interarrival time distributions, source-destination path activity, utilization, throughput, etc. The HYPERchannel is NSCs 50 Mbps (megabytes per second), baseband LAN product utilizing carrier sense multiple access (CSMA) with collision avoidance (CA) and capable of interconnecting a plethora of diverse computers and peripherals.

In 1984, I concluded a study funded by the National Bureau of Standards [7] to characterize the traffic in LLNL's

HYPERchannel-based network using these and other metrics; an early prototype of the EHMD proved indispensable.

In comparing the current intensity of network activity to that reported in the 1984 study, I observed that a dramatic increase had taken place. In August 1987, peak afternoon traffic as high as 350 messages per second was observed, whereas in 1984, the overall afternoon network traffic had been about 150 messages per second. During this same interval, the network had swelled to 40 nodes (adapters) from 25 nodes. I began to wonder how much network capacity remained and how best to use the EHMD to measure it. Given that we rely heavily on this network and have made a considerable investment of money and effort in it, we intend to prolong its useful lifespan and to extract every nickel's worth of value and every line of coding's worth of sweat from it.

CAPACITY TESTING

Consider an arbitrary and small moment of network time containing some network traffic, characterized by message sizes and interarrival times, as well as source/destination pairings. Suppose we had at our disposal the means to deny, in varying amounts, some fraction of that moment to the traffic normally present. We could then gradually increase the fraction of network time denied to normal traffic to the point where not enough time remained to accommodate that traffic. Suppose that by such a process we discover that the network collapses if we deny more than 40% of the available time. We could conclude that the current load consumes no less than 60% of the current capacity.

One way of denying time to the real load is to consume it with artificial load, but the artificial load, whatever it is and however it is generated, must not perturb the real load in any way whatsoever. It cannot alter receiver/sender availability or the true load's interarrival rates or message lengths. Note that for the purpose of capacity testing, we do not need to know the true load's characteristics; we need only to force it to fit in less available time.

I have avoided any reference to the term "bandwidth," often used to denote capacity. Network load and network capacity are inextricably related; load creates capacity as well as consumes it. If the messages are very small, one can achieve

* This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract number W-7405-Eng-48.

message rates in excess of 3,000 per second (Figure 1) but, because of protocol overhead, not realize very significant utilization, about 4.8% (0.024 Mbps). On the other hand, if the messages are very large (about 900 Mbits), one can achieve very high utilizations, 90% (about 45 Mbps), but not very high message rates, about 0.05/second. Given that the real load exhibits some time-dependent distribution of sizes and source-destination pairings, the network capacity will vary correspondingly. Thus, while it is these distributions that delimit network capacity, it is the load's intensity, as characterized by the distributions of its interarrival times, that consumes network capacity. For example, a load comprised of single byte messages arriving at a rate of 3,000 per second will consume 100% of network capacity (Figure 1).

For the purposes of our capacity test, we relied on two Amdahl 5840s that had been attached to the network but could not be in service until their network interfaces had been debugged, i.e., no other node in the network knew of their presence. The capacity test was conducted during the two weeks between successful debugging of the network drivers and their full integration into the network. The capacity test proceeded as follows: each afternoon from 1 to 5 p.m. the network was subjected to a known load test load. That is, one Amdahl would transmit messages of known length, mode, and intensity over the network to its counterpart. Table 1 summarizes the test loads. We used the EHMD to verify that the load was in fact what we wanted it to be. For example, the EHMD message trace in Figure 2 shows stream (burst) mode messages consisting of 16,399 bytes each arriving about every 7.6 milliseconds with a transmission delay of about 6.6 milliseconds (1 millisecond interarrival time). This amounts to a load that denies 87% of the available time to the real load:

$$= \frac{0.006626}{0.216690 - 0.209067}$$

$$= 87\% \text{ (of 1 trunk).}$$

CAPACITY		
	(Mbps)	(msg/sec)
bits 8	~45	0.05
9x10 ⁸		
msgs 8	0.024	~3000

Figure 1. Asymptotic bounds on HYPERchannel capacity.

However, because our network consists of two trunks, this load effectively amounts to 43% (test load 7 in Table 1).

METRICS

Once we had a means of generating a known load, we needed some means of measuring the network's response to that load. We decided on the following metrics:

- Idle time: any interval of time of longer duration than an End Delay (103 microseconds in our network)—the period of time that must elapse before an adapter knows no other adapter wants to use its slot time (called Priority Delay by NSC). Line 6 in Figure 3 shows the end of an idle period lasting 889 microseconds (i.e., 992 - 103 microseconds).

Table 1. Test loads.

Test (No.)	Load (%)	Message length (Kbytes)	Mode	Interarrival time (msec)
1	10	16	Multiplex	13
2	14	32	Multiplex	13
3	17	60	Multiplex	13
4	25	32	Burst	13
5	31	60	Burst	13
6	38	8	Burst	1
7	43	16	Burst	1
8	49	60	Burst	1

LLNL-HYPERTools Inc. EHMD Message Trace

on 10/09/87, at 15:31:26

rel(secs)	(fr>to)	mode	length	delay
0.201395	(08>09)	bm	16399	0.006626
0.209067	(08>09)	bm	16399	0.006626
0.216690	(08>09)	bm	16399	0.006627
0.224315	(08>09)	bm	16399	0.006626
0.232023	(08>09)	bm	16399	0.006626
0.239632	(08>09)	bm	16399	0.006626
0.247244	(08>09)	bm	16399	0.006625
0.254868	(08>09)	bm	16399	0.006626
0.262493	(08>09)	bm	16399	0.006625
0.270105	(08>09)	bm	16399	0.006626
0.277728	(08>09)	bm	16399	0.006626
0.285351	(08>09)	bm	16399	0.006626
0.292972	(08>09)	bm	16399	0.006626
0.300592	(08>09)	bm	16399	0.006617
0.308208	(08>09)	bm	16399	0.006624

Figure 2. An EHMD message trace. The fields are: the message's arrival time in seconds relative to the beginning of data collection [rel(secs)], its source/destination adapter pairs [fr>to], its mode [mode] whether stream (burst) or multiplex, its length [length] in bytes and its transmission delay [delay] in seconds.

- Transit delay: the elapsed time from the first attempt by a sender to establish contact with a receiver (called a reserve request) to the actual beginning of data transmission, e.g.,

$$\text{transit delay} = 36.777225 - 36.77666 = 0.56 \text{ msec}$$

(Figure 3).

- Dither: any sustained sequence of rejected reserve requests (Figure 4).

The phenomenon of dither is sufficiently complex and pervasive to warrant special elaboration. Each adapter attached to a HYPERchannel trunk has a unique slot during which it can gain access for transmission (CSMA/CA). There are two NSC trunk level protocols: multiplex mode and burst mode. In burst mode, once an adapter has acquired access to a trunk, it will retain control until all data has been successfully transmitted. In multiplex mode, all active adapters take turns using the trunk in a round robin fashion. When an adapter wishes to transmit data to some other adapter, it must first reserve itself and then reserve the destination adapter before it can send the data. Once an adapter has been reserved, by itself or by some other adapter, it will reject all subsequent reserve requests until the current reservation has been released (i.e., when the data has been

emptied out of the sending adapter, delivered to the receiving adapter, and emptied out of the receiving adapter into its attached host). Should an adapter be rejected in its reserve attempt, it will retry during its next slot and repeatedly, using its next 1000 or so slots. This process can consume tens of milliseconds. Unfortunately, the retrying adapter is self-reserved throughout the retrying.

This incessant and long-winded retrying impedes the progress of other adapters doing legitimate work because the retrying adapter will use access slots that could have been used by the others. In addition, retrying is a contagious condition; should some other adapter attempt to reserve either one of the working pair or the one retrying, it too will begin retrying, further impeding the working pair as well as the retrying adapter. Thus collectively they require a longer time either to finish transmitting the data or to finish their retry sequences. As more and more adapters bog down in retrying, the probability that some other adapter will "catch" it increases, because the opportunity to do so is increasing (the presence of an increasing number of infected, potential destination adapters) and the period of contagion increasingly prolongs itself.

As a metric, dither is simply the ratio of the number of rejected reserve requests to the total number of frames.

LLNL-HYPERTools Inc. EHMD Frame Trace

on 10/09/87, at 14:31:26

rel (secs)	delta-t	func	(fr>to)	tr	length
36.775615	662	ackno	(09>08)	3	0000
36.775641	26	clrf9	(08>09)	3	0000
36.775647	6	ackno	(09>08)	3	0000
36.775667	20	clrf0	(08>09)	3	0000
36.775674	7	ackno	(09>08)	3	0000
36.776666	992	RESF8	(08>09)	3	0000
36.776672	6	rject	(09>08)	3	0000
36.776716	44	RESF8	(08>09)	3	0000
36.776722	6	rject	(09>08)	3	0000
36.776825	103	RESF8	(08>09)	3	0000
36.776832	7	rject	(09>08)	3	0000
36.776935	103	RESF8	(08>09)	3	0000
36.776941	6	rject	(09>08)	3	0000
36.777044	103	RESF8	(08>09)	3	0000
36.777050	6	rject	(09>08)	3	0000
36.777175	125	RESF8	(08>09)	3	0000
36.777182	7	rject	(09>08)	3	0000
36.777225	43	copyr	(08>09)	3	0000
36.777232	7	ackno	(09>08)	3	0010

Figure 3. An EHMD frame trace. The fields are: the frame's arrival time in seconds relative to the beginning of data collection [rel(secs)], the inter frame elapsed time in microseconds [delta-t], its NSC trunk protocol frame function mnemonic [func], its source/destination adapter pairs [fr>to], its trunk of origin [tr], and its frame length [leng].

LLNL-HYPERTools Inc. EHMD Frame Trace

on 10/09/87, at 14:31:26

rel (secs)	delta-t	func	(fr>to)	tr	length
0.018447	57	RESF8	(83>01)	2	0000
0.018454	7	rject	(01>83)	2	0000
0.018558	104	RESF8	(83>01)	2	0000
0.018565	7	rject	(01>83)	2	0000
0.018623	58	RESF8	(83>01)	2	0000
0.018630	7	rject	(01>83)	2	0000
0.018734	104	RESF8	(83>01)	2	0000
0.018741	7	rject	(01>83)	2	0000
0.018845	104	RESF8	(83>01)	2	0000
0.018852	7	rject	(01>83)	2	0000
0.019010	158	RESF8	(83>01)	2	0000
0.019017	7	rject	(01>83)	2	0000
0.019075	58	RESF8	(83>01)	2	0000
0.019082	7	rject	(01>83)	2	0000
0.019187	105	RESF8	(83>01)	2	0000
0.019193	6	rject	(01>83)	2	0000
0.019251	58	RESF8	(83>01)	2	0000
0.019258	7	rject	(01>83)	2	0000
0.019362	104	RESF8	(83>01)	2	0000
0.019369	7	rject	(01>83)	2	0000

Figure 4. An EHMD frame trace showing dither between adapters 83 and 01.

RESULTS

The bar graphs in Figure 5 contrast the percent of idle time during a normal Friday afternoon with that for an afternoon during which a 25% test load was applied to the network. The graphs not only indicate that the test load was having an effect, but also provide a means of assessing its impact during the course of the afternoon.

Figures 6 and 7, the average percentage of idle time and transit delay, respectively, depict the response of the network to test loads 1 (10%) through 7 (43%). Both measurements exhibit an exponential dependence on the test load as it approaches high intensities. The network does not respond to a test load presented as messages in multiplex mode in the same way it does to messages presented in stream mode, the former taxing the network more. The reason for this is not clear and needs further study.

It appears that the current characteristics of network load at LLNL are such that:

- Our two-trunk HYPERchannel can never be less than about 20% idle and still be productive.

Figure 5. A comparison of normal and loaded (25%, 4, table 1) network behavior as measured by the idle percent.

- During peak times, we operate at about 57% of network capacity with an average transit delay of about 4 milliseconds.
- Our second trunk is critical to successful network operation during peak loads.
- Our second trunk no longer provides the redundancy we require.

We must distinguish between "percent of capacity" and "utilization." As long as there is no significant change in other characteristics, the applied load can be increased 75% before the network will saturate ($0.57 \times 1.75 = 1.00$). At this point the applied load will consume about 100% of available network capacity and the network will be 80% utilized.

Our network can function under a 43% test load, albeit not without some difficulty (e.g., average transit delays are about 3 times normal). The implication is that as our real

Figure 6. Idle percent versus test load percent for the 7 values of table 1 showing its abrupt drop near critical values of test load ($>40\%$). An M indicates multiplex mode test messages, a B, burst mode.

Figure 7. Transit delay versus test load percent for the 7 values of table 1 showing its abrupt increase for values $>30\%$. An M indicates multiplex mode test messages, a B, burst mode.

load increases, the transit delays will increase up to a maximum average value of about 10 milliseconds. Our results indicate that as the test load is increased beyond 43% and approaches the 49% mark, the network will collapse abruptly because of dither. And indeed, when subjected to the 49% test load (8, Table 1), the network collapsed so quickly due to dither that we were unable to collect any idle time or transit delay data. However, Figures 8 (normal state) and 9 (collapsed state) depict this collapse and the dither that caused it in a very graphic way.

SHORTCOMINGS

It was not until after the test period and we had studied the results that we discovered the flaws in our method. First, at relatively high test loads, it was difficult for the test load to defeat the CSMA/CA mechanism. That is, a 43% test load

was sometimes prevented from consuming 43% of the available time by the presence of real load. Because the test load did not always get onto the network when it needed to, our results are somewhat optimistic. Second, our message transit delay measurements included those associated with the messages in the test load itself. This, too, led to optimistic results.

These two defects in our approach account for the tendency of the transit delays to "flatten out" at high loads (Figure 7).

RECOMMENDATIONS

Because the cause of network collapse (dither) is clear, so too is its remedy—the reduction, if not total elimination, of needless and futile retrying. It should be clear at this point

Figure 8. A From/To graph (source adapters along the y-axis, destination adapters along the x-axis) for a normal afternoon, showing those adapters with < 30% of their frames rejected ■; those with > 30% but < 70% rejected ▨; and those with > 70% rejected ▩.

that adding more trunks will not prevent such collapse, because once an adapter is reserved, it is reserved across all trunks. However, a variety of tactics have been explored and have either proven or promise to be of benefit:

1. Implement some kind of binary exponential backoff/retry algorithm in the adapter driver interface to limit the number of times a host system will cause its attached adapter to embark on these long-winded, reserve/reject sequences.
2. Modify the adapter microcode to reduce the of number of retries from something often approaching the thousands to a more reasonable number, e.g., tens.
3. Abandon the idea of "adapter reservation" and move to an adapter architecture that supports full multiplex operation (the ability of a node to support more than one,

simultaneous session with other nodes), such as the NSC N-series adapters, which are compatible with the older, A-series adapter trunk level protocols.

CONCLUSIONS

The HYPERchannel-based local area network, which constitutes the communications backbone for the Livermore Computer Center at LLNL, operates at about 60% of its capacity during periods of peak load. Experiments predict that as the network load approaches its maximum limit, its interaction with overly persistent protocols at the trunk, link, and possibly higher layers will cause the network to collapse abruptly. The maximum sustainable load is not easily characterized as some fraction of the HYPERchannel's raw bandwidth or maximum message rate.

Figure 9. A From/To graph for the network during the collapse caused by the 49% test load (test 8 in Table 1).

REFERENCES

- [1] J. E. Donnelley, *The Internal Structure of NLTSS*, LINC internal document, Lawrence Livermore National Laboratory, Network Systems Division, Livermore, Calif., June 1983.
- [2] J. E. Donnelley, *The NLTSS Message System: Definition and Implementation for the Cray-1 and Cray X-MP Computers*, LINC internal document, Lawrence Livermore National Laboratory, Network Systems Division, Livermore, Calif., May 1984.
- [3] J. E. Donnelley, *NLTSS Network I/O*, LINC internal document, Lawrence Livermore National Laboratory, Network Systems Division, Livermore, Calif., April 1982.
- [4] J. G. Fletcher, R. W. Watson, *An Overview of the LINC Architecture*, Lawrence Livermore National Laboratory, Livermore, Calif., UCID-19294.
- [5] R. W. Watson, J. G. Fletcher, *An Architecture for Support of Network Operating System Services*, Computer Networks, vol. 4, no. 1, pp. 33-49, 1984.
- [6] R. W. Watson, *LINC Transport Service Specification*, Lawrence Livermore National Laboratory, Livermore, Calif., UCID-19296, 1984.
- [7] W. B. Watson, *Performance Monitoring a Large, High-Data-Rate Local Network*, Lawrence Livermore National Laboratory, Livermore, Calif., UCRL-53661, 1985.