https://doi.org/10.48009/2_iis_2005_317-323

MANAGEMENT OF LAN DESIGN FOR BUSINESS APPLICATIONS USING HIERARCHICAL SWITCHING: SIMPLICITY VERSUS ADDED DELAY

Dr. Paul Safonov, St. Cloud State University, MN, USA, safonov@stcloudstate.edu
Dr. Dennis Guster, St. Cloud State University, MN, USA, dcguster@stcloudstate.edu
Amit Parnerkar, St. Cloud State University, MN, USA, amit@bcrl.stcloudstate.edu
Chuck Hall, St. Cloud State University, MN, USA, chall@bcrl.stcloudstate.edu

ABSTRACT

The paper was designed to determine the effect that nesting Ethernet switches had on round trip delay of packets sent from a client to a server and back. A test bed of switches nested seven layers deep was used to collect data. Four experiments were run using both single client server pairs and four simultaneous client server sessions following the same physical path. These experiments were run on lightly and moderately loaded switches. The results revealed minimal delay on the physical switch level, but potentially dangerous delay when multiple client server sessions took the same physical path. Recommendations are provided regarding how to deal with the results from a hardware, software and personnel management level.

Keywords: LAN (local area network), Network Design, Network Performance, Switching.

INTRODUCTION

The LAN design process has made great strides since the days of Ethernet shared media supported by hubs [4]. The contention problems common in shared networks have been eliminated by using switches instead of hubs [3]. While it is true that linking devices together via switches eliminates packet collisions and the delay caused by waiting for that line to reset, there are other factors that may cause delay in switched based networks, such as message length, line capacity, and chip delay within the connecting devices [9]. End-users are very sensitive to delay. If the response happens within 100 ms (milliseconds) or $1/10^{th}$ of a second, most do not notice any delay [4], but beyond this threshold users get frustrated waiting for the network to display a Web page, echo a typed character, download an e-mail, etc. Furthermore, this delay should not vary from day to day, a good rule of thumb is that the variation should be less than 1-2%. E.g., for a goal of an average delay of 40 ms, the variation should be under 400-800 microseconds [7].

In any network design it is often difficult to meet such maximum average delay requirements. This may be attributed in part to the complexity and size of many LANs and the software applications they support. When LANs first were implemented they often only encompassed a single functional work group. Now it is common that every functional work group in an organization have connectivity not only to its members, but the whole company and potentially everyone in the world. This situation makes it clear that networks need to evolve according to some organized plan. Such plan typically involves a hierarchical switching relationship just like most file systems use a hierarchical tree structure. In the single workgroup model one switch could provide connectivity for up to approximately 50 workstations. At some sites the number of workstations is often in the thousands, a value much greater than the capacity of a typical

Ethernet LAN switch. Therefore, to effectively design LANs with more than one work group it is necessary to configure and deploy multiple switches. Hierarchical switching makes it easy to organize and manage workgroups in multi-level tree structures (e.g. Figure 1).

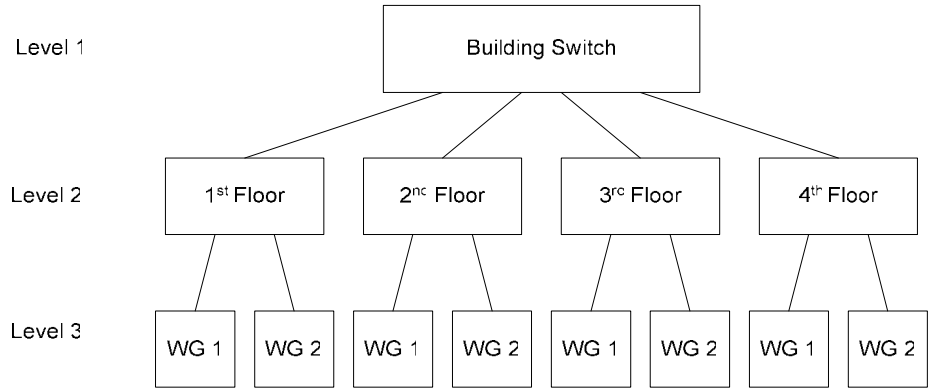

Figure 1. Hierarchical Switching Example

Hierarchical Switching: Simplicity of Design versus Performance

While the ability to organize workgroups into hierarchies is very appealing from a design and simplicity point of view, what happens to delay as additional switches are added to the network? For most switching devices the delay for the basic chip is calculated and available from the manufacturer. For example, the Tycho Electronics MAMUSM0008 [6] has published a maximum delay of 50 pico seconds. However, once integrated into a multi-port switching device with possible busy states it would be expected that this theoretical value would not be achievable. In fact, while testing the NTP (network time protocol) in a switched network Cisco [2] suggests a default delay of 3 ms. This value would easily fit into the confidence interval for the example listed earlier in which the target threshold was 40 ms, especially in a single work group single switch solution. However, the network time protocol is less intensive than most business applications, but the expectation of transferring between client server applications in a 3 ms time frame is encouraging.

However, would the performance still be adequate as additional switches are added in a nested topology? If the delay (in the NTP example above) was linear one could nest up to 13 devices and still maintain the 40 ms delay (it least on the switching level). That number may not be realistic because other factors such as propagational delay through the wire connecting the switches and the workstations, workload profiles that cause busy conditions in any given switch, and interactions among the state of all switches in the hierarchy [5].

Specifically, it is the delay caused by the interaction of switches that will be addressed by this paper. Simply stated, as a packet travels through the switch hierarchy it must wait until it acquires a path through a given switch and once through that switch it must wait again at the next switch in the path and so forth until it passes through all switches [8]. Therefore, the delay across the entire switch hierarchy would be somewhat analogous to traveling across town on a main street on which cross traffic is controlled by traffic lights. If your timing is just right and the lights are synchronized you will complete your trip relatively quickly (similar to not busy

switches). However, if your timing is poor and the lights are not synchronized it may take you significantly longer to travel the same distance (one or more switches are in a busy state). Therefore, to ascertain the delay profile caused by hierarchical switching a number of experiments will be undertaken in which the depth of the switching nests will be varied and delay will be recorded and analyzed.

METHODOLOGY

To collect the data a time stamping client/server program was written in Java. The client program generates UDP (user datagram) packets, timestamps them, and sends them across a switch grid to the server process. The server process records their arrival and sends a response packet back to the client which timestamps its arrival. The program was written without receive queues and could be viewed as using "blocking logic". This approach was used to provide an understanding of the busy conditions that occur in links and ultimately affects delay. The program is capable of varying the packet arrival distribution and the size of the packet payload.

A series of four experiments were devised to ascertain the round trip delay caused by nesting switches under a variety of conditions. Experiment 1 was designed to determine the delay that could be expected with a single client/server packet flow going through a series of lightly loaded switches. The second experiment was designed to determine the delay expected if four client/server pairs took the same path simultaneously through the series of lightly loaded switches. Whereas, the third experiment was designed to determine the delay expected for a single client/server pair going through a series of moderately loaded switches. The last experiment was to determine the interaction between multiple processes accessing the same port on a switch moderately loaded with traffic on its other ports. To guide the investigation four respective null hypotheses were devised:

- **H1**: The increase of the number of *lightly loaded* switches, which *a single* client/sever communication flow passes through, does not cause a significant difference in delay.
- **H2**: The increase of the number of *lightly loaded* switches, which *four* client/sever communication flows pass through, does not cause a significant difference in delay.
- **H3**: The increase of the number of *moderately loaded* switches, which *a single* client/sever communication flow passes through, does not cause a significant difference in delay.
- **H4**: The increase of the number of *moderately loaded* switches, which *four* client/sever communication flows pass through, does not cause a significant difference in delay.

Lightly loaded is defined here as packet traffic on the other ports in the switches not directly involved in the experiments being loaded by less than 20 packets per second. Whereas, *moderately loaded* corresponds to more than 200 packets per second. The test bed environment was variable from one to seven switches configured in a hierarchical format. Furthermore, to separate switch delay from propagational delay, where appropriate, the experiment was also run with just a crossover cable (a hard wire connection between two devices). The packet size was set close to the maximum for Ethernet (~1500 bytes) to maximize throughput.

RESULTS

Experiment 1 results are presented in Table 1. The middle column displays the number of lost packets out of the 3,000 transmitted for each session of the experiment. When the client and server machines were directly connected via the crossover cable there were 99 lost packets. At first it looked strange as normally the hardware should easily support this level of transmission. However, we determined that the server process was the weak link responsible for the packet loss, since the data collection program was designed to be blocking in nature. In a non-blocking program the packets would have gone into a queue and then processed later when the process was not busy. To help clarify the concept of "busyness" (on the packet level) in this experiment we used ratio of the number of lost packets to the number of packets sent. For the first entry in Table 1: (99/3000*100 = 3.3%). If packet loss occurred it was clear that as the number of packets lost increased so did the delay. In fact, if packets lost exceeded 400, the delay increases non-linearly. The difference in delay between the crossover and the single switch was .64 ms, which gives an idea of the proportion of switch to propagational delay. After adjusting for the increased delay caused by packet loss one can generally state that there is a uniform linear increase of about .12 ms as switches are added. The variation can be attributed to the fact that the distribution of the packet inter-arrival rates within the experimental flow is randomly generated. Although there was a .63 ms increase in delay from one to seven switches, it is insignificant comparing to maximum allowable delay target of 40 ms. Thus, hypothesis one can be accepted: there is no significant difference in delay as the packets travel through up to seven switches.

Table 1. Single Client/Server Pair on Lightly Loaded Switch(es)

Number of Switches	Number of lost packets	Roundtrip delay (ms)
Crossover	99	1.60
1	64	2.24
2	0	1.98
3	68	2.35
4	50	2.56
5	0	2.55
6	7	2.65
7	8	2.87

In **Experiment 2** the four client/server pairs would be analogous to having four (network active) windows open simultaneously. Therefore, these four processes could generate a theoretical load four times greater on the single physical path than in the previous experiment. The results are the average of the four sessions that were recorded. At first, the results (see Table 2) are quite confusing. However, when evaluating the delay in conjunction with number of lost packets it is clear that packet loss is skewing the delay. On the crossover, one and two switch levels' packets are spit out so fast that server process hangs up and needs time to recover, hence the massive delays. Starting at the three switch level the added delay of going through the additional switches smoothens out the flow and provides excellent transfer results and a nice linear pattern from the three to the seven switch level. So therefore, it appears sometimes a little delay is a good thing (in a blocked link). It is clear that there is significant difference in delay among the switch levels and therefore hypothesis 2 must be rejected. However, the high delay unexpectedly occurred at the 1 and 2 switch levels at the opposite end of the expected spectrum.

Table 2. Average of Four Client/Server Pairs on Lightly Loaded Switch(es)

Number of Switches	Number of Lost Packets	Roundtrip Delay (Ms)
Crossover	747	95.00
1	1149	130.25
2	763	83.50
3	49	6.25
4	24	6.50
5	52	6.75
6	52	7.00
7	372	7.25

Experiment 3. Since its purpose was to determine the effect of moderate traffic forwarded through the other ports in the switches, the crossover configuration was not applicable. It is interesting that the packet loss is not as severe in the one and two switch categories. This may be in part due to the added delay caused by the competing traffic being forwarded through the other switch ports. As in Table 1 the problem of overloading the server process prevents the delay from following a linearly increasing roundtrip delay progression. Once again, as packet loss occurred so did the delay. In fact the maximum packet loss table value, 139, exhibited the longest delay. After adjusting for the increased delay caused by packet loss one can generally state that there is a uniform linear increase of about .06 ms as switches are added. Comparing the lightly loaded switch values (Table 1) to the moderately loaded switch values (Table 3), the additional load only added about .20-.40 ms of delay within each category. These values are minimal and would allow us to accept hypothesis 3. So, there is no significant difference between lightly and moderately loaded switches supporting a single client/server pair session.

Table 3. Single Client/Server Pair on moderately loaded Switch(es)

Number of Switches	Number of Lost Packets	Roundtrip Delay (Ms)
Crossover	N/A	N/A
1	35	2.62
2	136	6.31
3	59	4.87
4	68	2.77
5	195	2.78
6	20	2.73
7	201	3.03

Experiment 4 was designed to determine the interaction between multiple processes accessing the same switch port and a switch moderately loaded with traffic on its other ports (Table 4). Because the goal was to determine the worst case scenario the maximum values of the four sessions are reported. Generally speaking the delay was much greater than in Table 2. However, the spiking at switch 1 did not occur. It occurred at switch 2 though, but after that an increasing progression took place (if adjustments were made for packet loss). The values at the 6 and 7 switch levels exceed 100 ms. All row entries except switch 1 exceed the 40 ms threshold. The interaction between the moderate competing loads from devices connected to the switch(es) and the four logical sessions being pumped through the single physical path results in deteriorating performance. The fact that delay increases from ~15 to ~138 ms causes us to reject Hypothesis 4.

Table 4. Maximum of Four Client/Server Pairs on Moderately Loaded Switch(es)

Number of Switches	Number of Lost Packets	Roundtrip Delay (Ms)
Crossover	N/A	N/A
1	371	15.49
2	936	96.50
3	455	50.78
4	370	50.57
5	199	61.35
6	1333	120.04
7	372	138.01

CONCLUSIONS AND RECOMMENDATIONS

The original premise of the paper that nesting switches too deep would result in an increasing exponential delay just did not occur at least up to the seven switch level. Certainly, it would be possible to nest switches deeper, but it is not likely to happen in most LAN applications. However, there was a number of other findings that have major implications on the network design process. Users routinely have multiple client windows open at the same time, which in effect is multiplexing their available physical connection. How many of these sessions they have open and how network active each session can have a major effect on the delay they can expect. Based on the data herein, with four active sessions their delay might increase to ~130 ms in a moderately loaded switch structure.

If one accepts the premise that any value that exceeds the 40 ms threshold is a potential problem, then what network design principles can be invoked to minimize that occurrence? It seems logical to break the design process into three components: hardware software, and personnel. On the hardware level the results indicate minimal delay and excellent scaling as switches and competing workload are added. It appears that the hierarchical strategy is effective and there is little to gain from its revision because the values are so small already. On software level how well the client and the server stubs interact with the network is very important. In these experiments UDP was used as the transport protocol and the payload size is definable through the stub logic. In this study, this payload size was set at 1450 bytes close to the maximum allowed by Ethernet. Varying this value can have a major impact on performance and throughput across the network and in many cases this value is set to an arbitrary value such as 1024 (used in many sample programs). This value may work well in many networks but, poorly in others. Ideally, the software needs to be tested in its home network under a variety of conditions so that the optimum value can be determined. Further, the resiliency of the server process used herein was limited especially as the workload offered increased. At 100mbs it is possible to pump out a lot of packets in a hurry. However, the protocol stacks can seldom operate at the same transfer rate [1]. This study used UDP, which can often transfer at the 60 Mbs range, but TCP (transport control protocol) is often limited to the 40mbs range. Moreover, the client/server processes themselves have limitations. They can only process information at certain rates. Their speed is typically a function of processing power, available memory, development software used and coding efficiency. Based on the client/server software used herein it appears that their performance curve is exponential in nature. The client/server pair provides adequate performance up to a point and when the performance goes bad it spikes in a hurry. Proactive designers need to test the client/server software on its home network to determine if that point of massive

deterioration will be reached and if so devise a strategy to correct it. So, based on this study we can derive the following **rule of thumb**:

> *Nesting switches seven layers deep does not add significant delay on lightly or even moderately loaded switches unless multiple simultaneous users are present.*

Although this paper provided useful baseline data about the effect of nesting switches, it raises several additional research questions. First and foremost, is a little delay sometimes a good thing? From the data depicted in Table 2 it appears to be the case. In a computer network there is a multitude of devices that operate a variety of different speeds, and the delay from one device to another can help smoothen out some of those speed differences (although the basic network design premise should still be to minimize delay). Second, UDP was used as the transport protocol in this study, however many client server applications use TCP (transmission control protocol). UDP features best effort delivery, but does not keep track of lost packets and therefore is unable to recover them. Whereas, TCP keeps track of packets and asks for a retransmission when a packet with a given sequence number is missing. One would expect that the added overhead of this recovery service in TCP would impact delay, but to what extent is not clear perhaps it is case specific. It would be interesting to adapt a client/server application so that it could run either on UDP or TCP and to subject each version's performance to experiments similar to those carried out herein. Third, the fact that the switch delay was so minimal as additional switches were added to the hierarchy needs to be replicated within multiple LAN settings. If further replications support these findings then the burden of improving network design and the resulting performance is shifted even more to the services world, particularly how well developed client/server applications interact with their supporting network.

REFERENCES

1. bf3Net performance figures and memory requirements (2003). Ethernet data links. http://www.windmill-innovations.com/products/bf3Net/bf3Net_testing.htm.
2. CISCO Catalyst 5000 Series Switches. (2003). Administering the Switch. http://www.cisco.com/en/US/products/hw/switches/ps679/products_configuration_guide_chapter09186a008007f341.html.
3. Ethernet Switching Information (1998), KJM White Paper. http://www.kmj.com/kmjinfo/smenu.html.
4. Guster, D. & Holden, M. (1996). Integrating High-Speed Switches into Existing Networks as a Means of Bridging to the 100Mbs World. *Proceedings of the Small College Computing Symposium*, St. Cloud, MN, April 18-20.
5. Lloyd-Evans, R. (1996). *Wide Area Network Performance and Optimization*. Harlow, UK: Addison Wesley Longman Ltd.
6. MAMUSM0008 Switch Specifications. (2003). http://catalog.tycoelectronics.com/TE/bin/TE.Connect?C=17142&P=107037&M=PROP&N=1
7. Oppenheimer, P. (1999). *Top-Down Network Design: A Systems Analysis Approach to Enterprise Network Design***,** Macmillan Technical Publishing (Cisco Press), Chapter 2: Analyzing Technical Goals and Constraints, http://www.groupstudy.com/bookstore/samples/Oppenheimer/.
8. Perlman, R. (1992). *Interconnections: Bridges and Routers*. Reading, MA: Addison Wesley Co.
9. Sarwar M. (1999). Comparative Study of Circuit Switching and Packet Switching http://cseweb.uta.edu/~sarwar/#3.1%20Delay%20Comparison.