Case Study #132

Findings WEB Server Application Issue

Briefing

School districts throughout the entire state use a centralized datacenter where applications are consolidated to reduce cost. Applications from this location perform functions such as tracking enrollment, grades, teachers and class schedules. The 100,000s of end users, accessed by teachers, students and parents or guardians which can be access within the schools or remotely for the convenience of all end users via a web portal.

Back ground: the application was changed out in the off season (June and July) in preparation for the upcoming school year. It was installed under strict supervision from both the IT department and the application developer vendor. The vendor had promised that it was ready for prime-time. As the school season began, end users began to have problems such as timeouts and lockups. The most common error was a WEB-speed error. Every inch of the network device, servers and applications had been reviewed and tweaks had been made, leading to less PC locking up, but the problem continued. The problems had been continuing for 5 months.

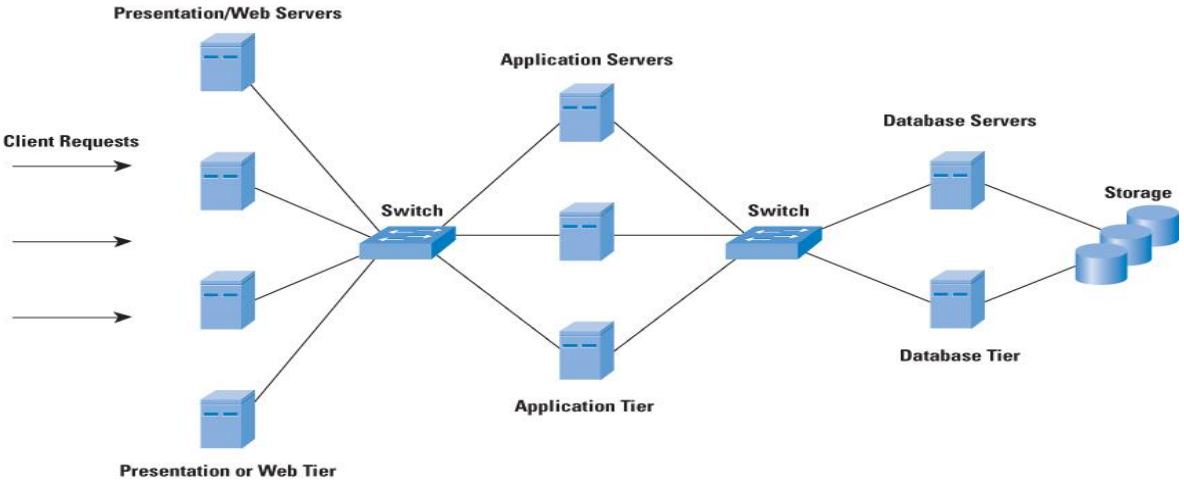
Network Detail: The network is a typical 3 tier architecture, remote client accessing a front end WEB server and the backend was a SQL database running on Microsoft Windows 2000 Advanced Server and Datacenter Server operating systems, supporting Load Balancing.

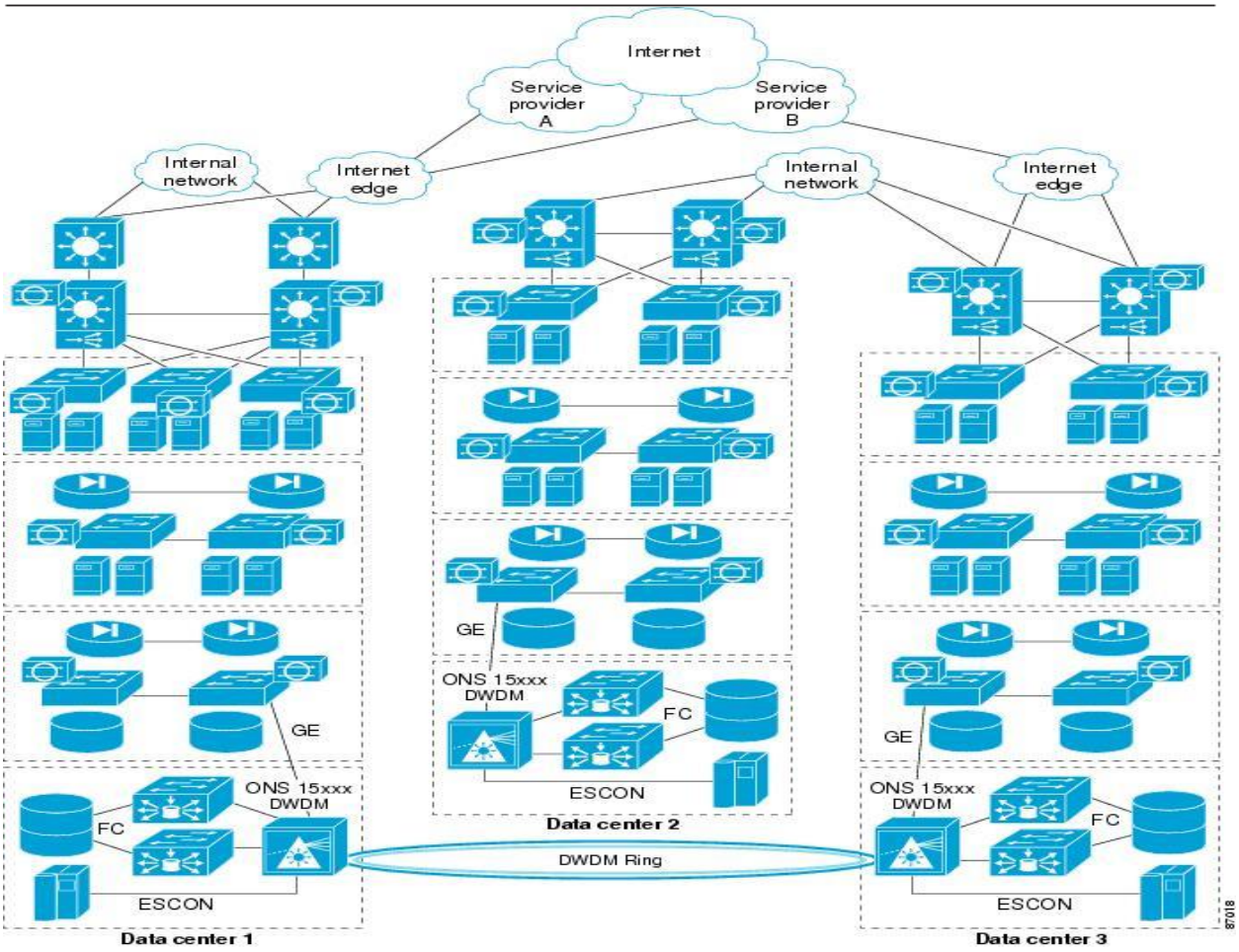
Network Load Balancing, a clustering technology included in the Microsoft Windows 2000 Advanced Server and Datacenter Server operating systems, enhances the scalability and availability of mission-critical, TCP/IP-based services, such as Web, Terminal Services, virtual private networking, and streaming media servers. This component runs within cluster hosts as part of the Windows 2000 operating system and requires no dedicated hardware support. To scale performance, Network Load Balancing distributes IP traffic across multiple cluster hosts. It also ensures high availability by detecting host failures and automatically redistributing traffic to the surviving hosts.

In addition the Web server and the SQL database are also running VM environments. Network access included public and private enterprise connections to over 100 school districts. Internal networks within the datacenters include Load Balancers, Firewalls and an assortment of routers and switches to create the connectivity. In addition remote Traffic within the remote school districts was being used to prioritize the network packets.

Gearbit's protocol analyzer installed within the core of the datacenter and connected to a SPAN port in between the core routers and the Load Balancer. Majority of the capture traces were taken at this

point, but some were taken on the backend, capturing the SQL traffic from the WEB servers and SQL database servers.





Network Analysis

The problem was identified with TCP half opens at the WEB Server. In the diagram shown below, it shows a normal TCP Close. The TCP session the client would issue a TCP finish at the end of the HTTPS transaction. The response from the Web Server would be an acknowledgement, and a TCP finish to the client PC. Then the final set, an acknowledgment from the PC to the Web Server.

Client PC ----------(TCP-FIN)----------> Web Server

Client PC <---------(TCP-FIN & ACK)--------- Web Server

Client PC ----------(ACK)----------> Web Server

Example 1 TCP Close with FIN followed with FIN-ACK then ACK

The trace files revealed that the TCP sessions were being closed very differently from what would be normal. Sessions were being closed from the Web Server itself, also Resets were also used. The Reset is

a quick way to end a session. In some applications this has become a tradition way to close a session, but primarily one way or the other is used, not both.

Client PC <----------(Reset)---------- Web Server

Client PC <---------(TCP-FIN & ACK)--------- Web Server

Client PC ----------(ACK)----------> Web Server

Example 2 Web Server initiated the TCP close with a Reset (not normal)

Throughout the traces abnormal TCP close types were seen shown in both examples 2 and 3. The abnormal condition the Web Server should not be closing the session. In a rare occasion if for example an application had a problem with the database, then you might see this type of close. So both examples 2 and 3 the Web Server indicated the abnormal TCP Close.

Client PC <----------(TCP-FIN & ACK)-------- Web Server

Client PC ----------(Reset)---------->Web Server

Example 3 Web Server initiated the TCP close with a TCP FIN (not normal)

The results with TCP half opens occurring at the Web Server, resources within TCP get used up. Within the Microsoft operating system there are limitations such as how many concurrent TCP sessions can be open, how long they can remain open, etc. The end result, if they reach the maximum limit, strange occurrences begin to occur, such as we're seeing with the Student Record application.

For more information concerning the TCP problems seen within the trace files see Appendix A

Application Analysis

There are no complete concussions in the packet traces other than the half-open conversations and some unexplained resets. These types of half-open conversations and resets can be indicative of problems with the application running on the web servers. Half-open conversations and resets may result from:

1) An application that fails suddenly do to an exception condition that has not been handled by the application – such as the failure of a back-end database request or a programming error (i.e. an infinite loop or exhaustion of available memory to memory leaks).
2) An application that has not been designed to scale well in a high load environment. The application may run out of available resources or become locked at which point the operating

system/web server stack may terminate the application or the application may simply yield control and no longer respond.

3) The web-server/OS stack may not be properly tuned to manage a higher application load. Such things as number of available ports, the length of time ports may be released after use for other clients.
4) The application is being deployed at the client site using a CGI approach as opposed to an ISAPI approach. CGI applications are typically run as separate processes under the web server. Memory, thread and process constraints could be affecting the ability of the web server to fork and manage those CGI processes. Also, as connections are passed to these CGI processes the number of connections could be constrained either for the web server or for the child processes.
5) Communication between a web server and its child CGI processes is inherently more distant and problematic because they no longer share the same process space. This can sometimes lead to delays in socket shutdowns and/or exception handling management

Short Term Solution

To test this to insure that we added four web servers to the web environment for School District A, bringing the total to number of web servers to eight. This was only to increase the number of TCP ports available. The results were positive with no sessions failing that caused the application to error.

Other Things Observed

The overall network health was in great standing. Organized, design in good standing, no packet drops, routing working properly and minimal latency were observed. The normal transaction time seen at the outbound side of the Load Balancer was sub millisecond, (.000900). Very impressive response time for the Web Server to handle the HTTPS request and get the response from the database server and app server, then send the HTTPS data response in sub millisecond. This shows that all servers, network connections, load balancers, firewall and routers are working well.

Also throughout the trace files that we analyzed from the School District A the network things look very healthy. The transaction response time was good, sub second, and all data was being delivered as expected. Again showing that routing, packet shaping, Ethernet switches, connections, firewalls, and other network devices working as they should.