Performance of Microsoft's File Sharing over Wide Area Networks Version 1.0[123]

The following describes some recent work in the area of network performance over Wide Area Networks (WAN). In particular, this looks at the performance of Microsoft's file sharing via its Server Message Block (SMB) protocol. When operating over a WAN environment SMB is layered over the TCP transport layer protocol. Therefore when looking at why network performance is less than expected, you need to look at both protocols individually, and you also need to look at their interaction for a complete answer.

Limitation Associated with the SMB Protocol

The SMB protocol has two main characteristics lending to performance issues when operating over wide area networks:

- **Chattiness**: The SMB protocol is extremely chatty. Most of the managing of the actual data transfer is performed via a series of request/response transactions
- **Window Based Flow Control**: The SMB protocol uses a window mechanism for flow control. This value of this window along with the round trip time characterizes the minimum time required to transfer the bytes of the file through the network.

The Round Trip Time (RTT) through the network plays an important factor in each of these issues and how they affect performance. The RTT time through a LAN should be on the order of few milliseconds, However the RTT though a WAN may be on the order of tens or hundreds of milliseconds, depending on the characteristics of the links through the network; thus, orders of magnitude in difference. The larger the RTT, the longer it takes for a response of some stimuli to return to the source.

Chattiness

For example, let's assume you log into a system and display a directory. This may take over 50 transactions, most of them are serialized. If the RTT is only 10 msec, typical of a LAN environment, it would only take 0.50 seconds to complete the transactions and display this information. On the other hand, if the RTT is 500 msec- typical if a satellite in present, it would take at least 25 seconds to display the same information.

Window Based Flow Control

Now, let's assume we need to upload a 1 Meg file from the client to the server. Since we are going to upload the file, *core* mode within the SMB protocol is used and its default buffer size is 4K. This requires about (1 Mbyte / 4192 KByte) 240 blocks. SMB requires that after each block is sent, an application layer acknowledgement is required before the next block is transmitted. Therefore in the LAN environment each of the 240 blocks takes a minimum of 10 msec to transmit the data and receive the application acknowledgement; therefore it will take at a minimum of 2.40 seconds to upload the file. In our WAN environment, each of the 240 blocks

---

[1] 1Approved for Public Release; Distribution Unlimited

[2] ©2005 The MITRE Corporation. All Rights Reserved

[3] The research described in this paper was carried out for the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration

takes 500 msec to transmit and receive that application layer acknowledgement; therefore it may be uploaded in 120 seconds.

If we assume we are using *raw* mode within the SMB protocol and the file is being transferred from server to client, the default buffer size of 60K is used. At 60K blocks it only takes 16 blocks to transfer this file. Again at LAN RTTs the file will be transferred in 16 * 10 msec or 160 msec. In the WAN environment, the file will be transferred in 16 * 0.250 seconds or 8 seconds.

The default buffer sizes may be changed. However, since the buffer size is stored in an 8 bit value, the maximum buffer size is only 64K. Therefore the above analysis for a *raw* mode transfer is close to a theoretical best.

Interaction of TCP and the SMB protocol

Now let's briefly describe the interactions of TCP and the SMB protocol. They generally fall into three major categories:

- TCP loss indication and recovery
- TCP flow control
- TCP congestion control

TCP Loss Indication and Recovery

The TCP protocol degrades network performance when packet loss occurs. Before discussing the interaction with congestion control, let's look at how packet loss is indicated via TCP. TCP has two techniques to detecting which packet and subsequent retransmission:

- **3 Duplicate Acknowledgements:** For this method, TCP requires at least 3 data packets for that data stream to be received after a packet is lost. Given the transaction style techniques of the SMB protocol, this may not be possible. For the request/response negotiation, not even a single packet would be available for transaction, much less three.
- **Retransmission Timer:** This is TCP's stop gap method. If a TCP retransmission timeout occurs, severe performance degradation will occur.

TCP Flow Control

The TCP protocol also contains a window based flow control mechanism. The TCP window is used for flow control to ensure the sender does not overwhelm the receiver's buffers. In actuality, both the TCP sender and receiver have buffers. The sender uses the buffers as a retransmission buffer (i.e., to store data for possible retransmission) while the receiver uses these buffers as an out-of-sequence queue (i.e., to save packets received by the sender that are not in sequence to avoid the sender from possibly retransmitting them.) The key point in the following: if the buffer sizes are different, only the minimum buffer size of the two is used for flow control. In terms of performance the window size really refers to the maximum amount of data that can be in-flight. The default window size for Windows XP I believe is about 17K bytes.

Therefore if TCP's window size is less than SMB's window size, then TCP's window will be the limiting factor.

TCP Flow Control Part II

Let's take a closer look at the data transfer phase of SMB. Based on the above information, it would appear larger buffer sizes are better; rherefore we will assume a SMB buffer size of 60K and will assume TCP window sizes are sufficiently large enough to not constrain performance for the following discussion. Let's start looking at a random place during the data transfer – an entire

60K block of data has been transferred and we are waiting for the application layer acknowledgement before emitting new data. Once that acknowledgement is received, 60K of new data is immediately emitted in the network. Assuming standard ethernet sized frames and a MSS of 1460 - 41 packets are immediately burst. Given a single file transfer a burst of 41 packets has the ability to cause packet loss due to router queue overflow. For networks containing many nodes simultaneously transferring files, this results in an extraordinary about of network traffic. Increasing the probability of network loss, causing data to be retransmitted; and only increase the file transfer time.

SMB and TCP Performance Enhancing Proxies

There are various techniques out there to improve the performance of TCP. TCP PEPs are a common technique to improve the performance of TCP based application in challenged environments. PEPs, operating at the transport layer, increase the performance of TCP based applications where native performance suffers due to characteristics of a link or subnetwork in the path. These PEPs do not modify the application protocol, thus applications operate end-to-end. These PEP techniques are totally transparent to the applications. Technically, these gateways perform a technique called spoofing in which they intercept the TCP connection in the middle and terminate that connection as if the gateway were the intended destination. PEP, typically bracketing the satellite network, split a single TCP connection into three separate connections. These gateways communicate via standard TCP when talking to each of the end-systems and a third connection using an optimized rate based protocol is used when transferring the data between them. This technique allows one to isolate the properties of satellite networks that degrade performance from manifesting themselves to TCP. For example, corruption loss on the satellite network does not cause the transmission rate to be cut in half, and congestion loss on the terrestrial network does not cause data to be retransmitted consuming satellite bandwidth that may be a scarce resource.

These techniques can work very and sometimes extremely well when the application is akin to a bulk transfer - large amounts of data going from A to B. These techniques do not work as well when the application traffic is either chatty in nature or does not provide a constant stream of data to be transmitted over the challenged environment.