Key Components of WAN Optimization Controller Functionality

Ashton, Metzler & Associates

Leverage Technology & Talent for Success

Introduction and Goals

One of the key challenges facing IT organizations relative to application and service delivery is ensuring that the applications and services exhibit acceptable performance. Because of the importance of this challenge, an extensive set of techniques have been implemented by WAN Optimization Controllers (WOCs). Some of the primary roles that these techniques play are to:

- Reduce the amount of data sent over the WAN;
- Ensure that a WAN link is never idle if there is data to send;
- Reduce the number of round trips (a.k.a., transport layer or application turns) necessary for a given transaction;
- Overcome the packet delivery issues that are common in shared networks that are typically over-subscribed;
- Mitigate the inefficiencies of protocols and applications;
- Use multiple paths from origin to destination where appropriate.

This white paper has two goals. One of those goals is to describe some of the factors that lead to poor performance. The other goal is to describe some of the optimization functionality provided by WAN Optimization Controllers (WOCs) that can improve application and service delivery.

Impediments to Acceptable Performance

Factors that Impact Throughput

One factor that can have a negative impact on application and service delivery is packet loss. The effect of packet loss on TCP has been widely analyzed[1]. Mathis, et al. provide a simple formula that offers insight into the maximum TCP throughput on a single session when there is packet loss. That formula is:

Figure 1: Factors that Impact Throughput

$$Throughput <= (MSS/RTT)*(1 / sqrt\{p\})$$

where:	MSS =	maximum segment size
	RTT =	round trip time
	p =	packet loss rate.

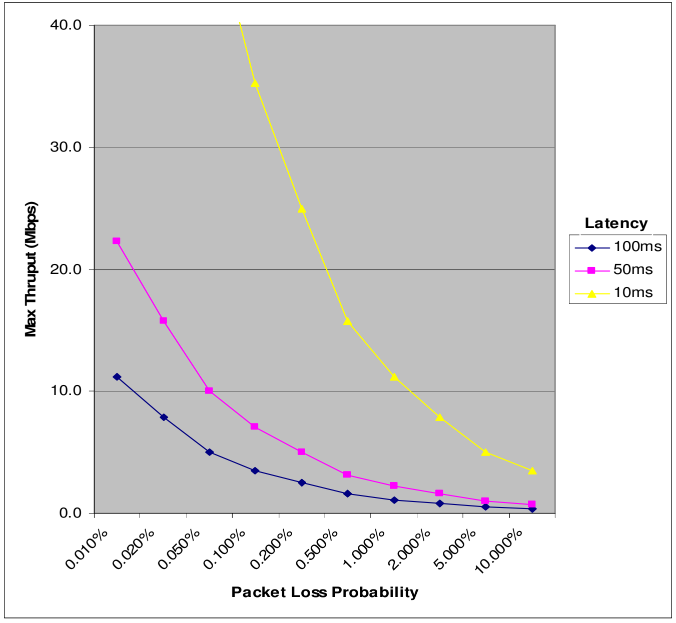
The preceding equation shows that throughput decreases as either the RTT or the packet loss rate increases. To illustrate the impact of packet loss, assume that MSS is 1,420 bytes, RTT is 100 ms. and p is 0.01%. Based on the formula, the maximum throughput is 1,420 Kbytes/second. If, however, the loss were to increase to 0.1%, the maximum throughput drops to 449 Kbytes/second. **Figure 2** depicts the impact that packet loss has on the throughput of a single TCP stream with a maximum segment size of 1,420 bytes and varying values of RTT.

[1] The macroscopic behavior of the TCP congestion avoidance algorithm by Mathis, Semke, Mahdavi & Ott in Computer Communication Review, 27(3), July 1997

One conclusion we can draw from **Figure 2** is:

Small amounts of packet loss can significantly reduce the maximum throughput of a single TCP session.

Figure 2: Impact of Packet Loss on Throughput



For example, on a WAN link with a 1% packet loss and a round trip time of 50 ms or greater, the maximum throughput is roughly 3 megabits per second no matter how large the WAN link is.

Quantifying Application Response Time

A model is helpful to illustrate the potential performance bottlenecks in the performance of an application. The following model (**Figure 3)** is a variation of the application response time model created by Sevcik and Wetzel[2]. Like all models, the following is only an approximation and it is not intended to provide results that are accurate to the millisecond level. It is, however, intended to provide insight into the key factors impacting application response time. As shown below, the application response time (R) is impacted by a number of factors including the amount of data being transmitted (Payload), the goodput which is the actual throughput on a WAN link, the network round trip time (RTT), the number of application turns (AppTurns), the number of simultaneous TCP sessions (concurrent requests), the server side delay (Cs) and the client side delay (Cc).

Figure 3: Application Response Time Model

$$R \approx \frac{Payload}{Goodput} + \frac{(\#\,of\ AppsTurns * RTT)}{Concurrent\ Requests} + Cs + Cc$$

[2] Why SAP Performance Needs Help

WOC Functionality

Table 1 lists some of WAN characteristics that impact application delivery and identifies WAN optimization techniques that a WOC can implement to mitigate the impact of those characteristics.

Table 1: Techniques to Improve Application Performance

WAN Characteristics	WAN Optimization Techniques
Insufficient Bandwidth	Data Reduction: • Data Compression • Differencing (a.k.a., de-duplication) • Caching
High Latency	Protocol Acceleration: • TCP • HTTP • CIFS • NFS • MAPI Mitigate Round-trip Time • Request Prediction • Response Spoofing
Packet Loss	Congestion Control Forward Error Correction (FEC) Packet Reordering
Network Contention	Quality of Service (QoS)

Below is a description of some of the key techniques used by WOCs:

- ***Caching***
  A copy of information is kept locally, with the goal of either avoiding or minimizing the number of times that information must be accessed from a remote site. Caching can take multiple forms:

  - *Byte Caching*
    With byte caching the sender and the receiver maintain large disk-based caches of byte strings previously sent and received over the WAN link. As data is queued for the WAN, it is scanned for byte strings already in the cache. Any strings resulting in *cache hits* are replaced with a short token that refers to its cache location, allowing the receiver to reconstruct the file from its copy of the cache. With byte caching, the data dictionary can span numerous TCP applications and information flows rather than being constrained to a single file or single application type.

  - *Object Caching*
    Object caching stores copies of remote application objects in a local cache server, which is generally on the same LAN as the requesting system. With object caching, the cache server acts as a proxy for a remote application server. For example, in Web object

caching, the client browsers are configured to connect to the proxy server rather than directly to the remote server. When the request for a remote object is made, the local cache is queried first. If the cache contains a current version of the object, the request can be satisfied locally at LAN speed and with minimal latency. Most of the latency involved in a cache hit results from the cache querying the remote source server to ensure that the cached object is up to date.

If the local proxy does not contain a current version of the remote object, it must be fetched, cached, and then forwarded to the requester. Either data compression or byte caching can potentially facilitate loading the remote object into the cache.

- ***Compression***
  The role of compression is to reduce the size of a file prior to transmitting it over a WAN. Compression also takes various forms.

  - *Static Data Compression*
    Static data compression algorithms find redundancy in a data stream and use encoding techniques to remove the redundancy and to create a smaller file. A number of familiar lossless compression tools for binary data are based on Lempel-Ziv (LZ) compression. This includes zip, PKZIP and gzip algorithms.

    LZ develops a codebook or dictionary as it processes the data stream and builds short codes corresponding to sequences of data. Repeated occurrences of the sequences of data are then replaced with the codes. The LZ codebook is optimized for each specific data stream and the decoding program extracts the codebook directly from the compressed data stream. LZ compression can often reduce text files by as much as 60-70%. However, for data with many possible data values LZ generally proves to be quite ineffective because repeated sequences are fairly uncommon.

  - *Differential Compression; a.k.a., Differencing or De-duplication*
    Differencing algorithms are used to update files by sending only the changes that need to be made to convert an older version of the file to the current version. Differencing algorithms partition a file into two classes of variable length byte strings: those strings that appear in both the new and old versions and those that are unique to the new version being encoded. The latter strings comprise a delta file, which is the minimum set of changes that the receiver needs in order to build the updated version of the file.

    While differential compression is restricted to those cases where the receiver has stored an earlier version of the file, the degree of compression is very high. As a result, differential compression can greatly reduce bandwidth requirements for functions such as software distribution, replication of distributed file systems, and file system backup and restore.

  - *Real Time Dictionary Compression and De-Duplication*
    The same basic LZ data compression algorithms discussed above and proprietary de-duplication algorithms can also be applied to individual blocks of data rather than entire files. This approach results in smaller dynamic dictionaries that can reside in memory rather than on disk. As a result, the processing required for compression and de-compression introduces only a relatively small amount of delay, allowing the technique to be applied to real-time, streaming data. Real time de-duplication applied to small

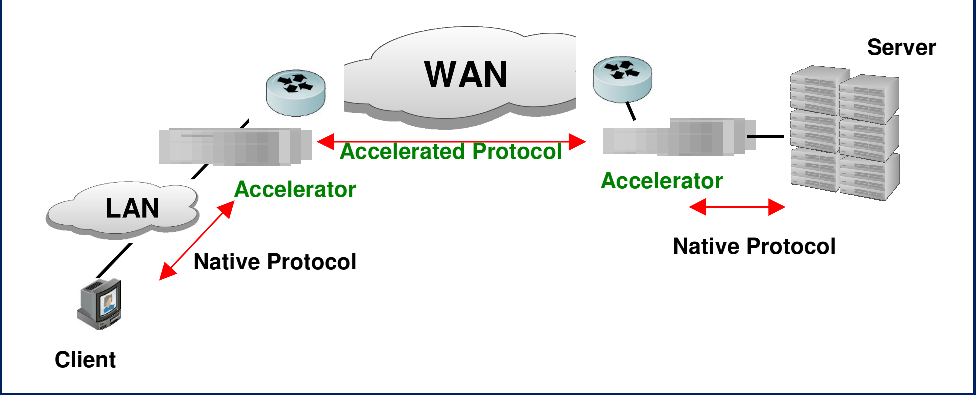
chunks of data at high bandwidths requires a significant amount of memory and processing power.

- ***Congestion Control***
  The goal of congestion control is to ensure that the sending device does not transmit more data than the network can accommodate. To achieve this goal, the TCP congestion control mechanisms are based on a parameter referred to as the *congestion window*. TCP has multiple mechanisms to determine the congestion window.

- ***Forward Error Correction (FEC)***
  FEC is typically used at the physical layer (Layer 1) of the OSI stack. FEC can also be applied at the network layer (Layer 3) whereby an extra packet is transmitted for every *n* packets sent. This extra packet is used to recover from an error and hence avoid having to retransmit packets. A subsequent subsection will discuss some of the technical challenges associated with data replication and will describe how FEC mitigates some of those challenges.

- ***Protocol Acceleration***
  Protocol acceleration refers to a class of techniques that improves application performance by circumventing the shortcomings of various communication protocols. Protocol acceleration is typically based on per-session packet processing by appliances at each end of the WAN link, as shown in **Figure 4**. The appliances at each end of the link act as a local proxy for the remote system by providing local termination of the session. Therefore, the end systems communicate with the appliances using the native protocol, and the sessions are relayed between the appliances across the WAN using the accelerated version of the protocol or using a special protocol designed to address the WAN performance issues of the native protocol. As described below, there are many forms of protocol acceleration.

Figure 4: Protocol Acceleration



  - *TCP Acceleration*
    TCP can be accelerated between appliances with a variety of techniques that increase a session's ability to more fully utilize link bandwidth. Some of these techniques include dynamic scaling of the window size, packet aggregation, selective acknowledgement, and TCP Fast Start. Increasing the window size for large transfers allows more packets to be sent simultaneously, thereby boosting bandwidth utilization. With packet aggregation, a number of smaller packets are aggregated into a single larger packet, reducing the overhead associated with numerous small packets. TCP selective

acknowledgment (SACK) improves performance in the event that multiple packets are lost from one TCP window of data. With SACK, the receiver tells the sender which packets in the window were received, allowing the sender to retransmit only the missing data segments instead of all segments sent since the first lost packet. TCP slow start and congestion avoidance lower the data throughput drastically when loss is detected. TCP Fast Start remedies this by accelerating the growth of the TCP window size to quickly take advantage of link bandwidth.

- *CIFS and NFS Acceleration*
  CIFS and NFS use numerous Remote Procedure Calls (RPCs) for each file sharing operation. NFS and CIFS suffer from poor performance over the WAN because each small data block must be acknowledged before the next one is sent. This results in an inefficient ping-pong effect that amplifies the effect of WAN latency. CIFS and NFS file access can be greatly accelerated by using a WAFS transport protocol between the acceleration appliances. With the WAFS protocol, when a remote file is accessed, the entire file can be moved or pre-fetched from the remote server to the local appliance's cache. This technique eliminates numerous round trips over the WAN. As a result, it can appear to the user that the file server is local rather than remote. If a file is being updated, CIFS and NFS acceleration can use differential compression and block level compression to further increase WAN efficiency.

- *HTTP Acceleration*
  Web pages are often composed of many separate objects, each of which must be requested and retrieved sequentially. Typically a browser will wait for a requested object to be returned before requesting the next one. This results in the familiar ping-pong behavior that amplifies the effects of latency. HTTP can be accelerated by appliances that use pipelining to overlap fetches of Web objects rather than fetching them sequentially. In addition, the appliance can use object caching to maintain local storage of frequently accessed web objects. Web accesses can be further accelerated if the appliance continually updates objects in the cache instead of waiting for the object to be requested by a local browser before checking for updates.

- *Microsoft Exchange Acceleration*
  Most of the storage and bandwidth requirements of email programs, such as Microsoft Exchange, are due to the attachment of large files to mail messages. Downloading email attachments from remote Microsoft Exchange Servers is slow and wasteful of WAN bandwidth because the same attachment may be downloaded by a large number of email clients on the same remote site LAN. Microsoft Exchange acceleration can be accomplished with a local appliance that caches email attachments as they are downloaded. This means that all subsequent downloads of the same attachment can be satisfied from the local application server. If an attachment is edited locally and then returned to via the remote mail server, the appliances can use differential file compression to conserve WAN bandwidth.

- ***Request Prediction***
  By understanding the semantics of specific protocols or applications, it is often possible to anticipate a request a user will make in the near future. Making this request in advance of it being needed eliminates virtually all of the delay when the user actually makes the request.

Many applications or application protocols have a wide range of request types that reflect different user actions or use cases. It is important to understand what a vendor means when it says it has a certain application level optimization. For example, in the CIFS (Windows file sharing) protocol, the simplest interactions that can be optimized involve *drag and drop*. But many other interactions are more complex. Not all vendors support the entire range of CIFS optimizations.

- ***Request Spoofing***
  This refers to situations in which a client makes a request of a distant server, but the request is responded to locally.

Conclusion and Next Steps

WOCs provide a wide range of functionality that can overcome factors such as packet loss and high latency that cause the performance of applications to degrade. Because a WOC will provide varying degrees of benefit to an enterprise based on the unique characteristics of its environment, third party tests of these solutions are helpful, but not conclusive. In order to understand the performance gains that will result from deploying WOCs, network organizations must test those WOCs in an environment that closely reflects their production environment.