Design Best Practices for Latency Optimization

This whitepaper addresses options for mitigating key sources of latency within financial data network implementations. Topics surveyed include propagation delay, processing and serialization delay, packet size, queuing delay, transport-layer implementation, middleware, applications, server operating systems, and security/compliance considerations.

Propagation Delay

Networking, as with anything, is subject to the laws of physics. Light travels through a vacuum at 186,000 mps, but—due to the refractive index of glass or the electron propagation delay in copper—data slows down so that typical real world numbers are closer to 122,000 mps. That works out to 8.2 microseconds per mile or 0.82 ms per 100 miles. Table 1 summarizes the overall effect of increasing distances.

You can minimize this effect by reducing the distance data must travel. Brokerages should work with their service provider (SP) to understand where facilities are located and how these facilities connect to the brokerage data center. Brokerages and SPs must coordinate efforts to minimize the distance between sites.

Co-locating the algorithmic trading servers at the service provider location—or even at the exchange—can give a brokerage a huge advantage. It also removes the need to continually upgrade links to the FSP or the exchange.

However, many brokerages cannot co-locate their servers due to security and compliance considerations.

Table 1 Propagation Delay and Distance

Distance	Propagation Delay (milliseconds)
1 mile	8.2 microseconds
5 miles	41 microseconds
20 miles	0.164 ms
100 miles	0.82 ms
200 miles	1.64 ms

Processing and Serialization Delay

Each router or switch in the data path adds a finite amount of delay as the packet is received, processed, and then forwarded. Each value-added feature, such as NAT or access lists, can add additional latency. Using features that are supported with hardware assistance will greatly reduce latency.

The once disparate worlds of LAN switching and WAN routing have been converging. The data communications industry has gone through a transition from TDMA-based serial lines to high-speed Metro Ethernet.

The added benefit from using Metro-Ethernet boxes is that they support hardware-assisted forwarding, which can greatly reduce latency.

Latency with a hardware-assisted switch will be in the 4-to-20 microsecond range. The most reasonable processing delay that you can expect in practice should be 25 microseconds per hop. The processing delay on a software-based router can be considerably higher.

Cut-through switching is often considered as an option to reduce serialization-related delay in a switch. Cut-through switching has diminishing returns with contemporary data rates. The advantage of cut through switching is that the switch can start transmitting the packet out the destination port before it has received the full packet on the incoming port.

With cut-through switching, you save the time it takes to transmit/receive the entire packet. Back in the days of 10 Mbps LAN links this meant a lot. The time it takes to transmit a packet at 10 Mbps is between 51.2 and 1200 microseconds for a 64 or 1500 byte packet. Today, at 1 Gbps, this drops to between 0.512 and 12 microseconds. At 10 Gbps, it further reduces to between 0.0512 and 1.2 microseconds. Table 2 summarizes serialization delay effects associated with various link types for 64- and 1500-byte packet sizes.

Table 2 Serialization-related Delay Summary

Packet Size	Link Size	Serialization Delay
64 bytes	256 Kbps	2 ms
	1.5 Mbps	0.35 ms
	100 Mbps	5.1 microseconds
	1 Gbps	0.51 microseconds
	10 Gbps	0.051 microseconds
1500 bytes	256 Kbps	46.98 ms
	1.5 Mbps	8 ms
	100 Mbps	120 microseconds
	1 Gbps	12 microseconds
	10 Gbps	1.2 microseconds

Another potential drawback to cut-through switching is the inability to perform any of the value-added features, such as quality of service (QoS), network address translation (NAT), and others. The packet will be switched before any of those features can perform their functions. Cut-through switching is often considered in the high-speed Data Center, where those features may not be required.

Smaller Packets: Less Bandwidth and Compression

Network utilization and latency are usually inversely proportional. Smaller packets will be transmitted over the network faster and therefore will have lower latency. However, many smaller packets require greater network overhead (IP headers and Ethernet headers) than fewer larger packets.

Compression has always been a trade-off between using less bandwidth with smaller packets and the CPU cycles to compress and decompress the packets. Today there are efficient compression algorithms that take less time and CPU power than the bandwidth savings associated with sending smaller packets. In a sense, CPU power has now exceeded the speed of light.

Queuing Delay

When packets are coming into a router faster than they can leave, you will have queuing. The best way to avoid packet-queuing latency is to avoid congestion. This translates into over-engineering your network to handle traffic bursts. When some level of congestion is inevitable then QoS methods such as Low-Latency Queuing (LLQ) should be used. In converged networks—with many different traffic types—market data must be integrated into the overall QoS strategy.

Weighted Random Early Detection (WRED) may not work so well with market data. It randomly drops packets as we approach congestion by looking at the Differentiated Services Code Point (DSCP) bits. The idea is that Transmission Control Protocol (TCP) will detect the drops and adjust its window size. The routers do not differentiate between TCP and User Datagram Protocol (UDP)—and UDP streams will not back off. Note that messaging software *will* detect the packet loss and start throttling back the application. This will increase the overall latency of the system which might be better than experiencing packet loss and retransmissions. However, the preferred strategy is to avoid congestion in the first place.

Transport Layer and the TCP/IP Stack

There are many options in the protocol stack that can effect the efficiency of the data delivery. You must understand the characteristics of the version of the stacks that you are running and that they are compatible with the versions and options on the other stacks.

For example, Nagle's algorithm—which is very useful for minimizing network overhead by concatenating packets together—interacts very badly with TCP delayed acknowledgements (another optimization option).

Some customers might want to disable Nagle's algorithm under certain circumstances and most stacks have a way to disable Nagle.

Another factor to consider is socket buffer tuning. The best example this type of tuning is to increase receive UDP buffers. UDP has typically been used for low level query/response work such as Domain Name System (DNS) and Network Time Protocol (NTP). Some of the kernel limits for these buffers were designed in the 1980s when networks ran much slower. These are now outdated and should be increased for today's networks.

Middleware Characteristics

The characteristics of the messaging protocol will affect latency. The overall efficiency in which the messaging bus performs during setup, tear down and retransmissions must be optimized.

How chatty the protocol is will affect the number of packets on the wire and cause all the other latency issues discussed previously.

The middleware also defines how many multicast groups will be used by the application.

Too few groups will have all the data going to all the users and there will be a very large user domain. A bad receiver will be able to affect a large community. Alternatively, too many groups and the network and the application might not be manageable.

The application developers must strike the right balance to divide up the data into logical groups that will give you some efficiency in the network and on the receiver side.

For example, NASDAQ divides up the data into streams for A-to-E, F-to-N, and so on. The tech stocks are divided up among all the letters. There will be people that want to listen to just the tech stocks and will get all the instruments. We must understand the usage pattern for the data stream and break it up into efficient components.

This requires a holistic approach to understand the application, the data usage pattern, and the multicast addresses—and to then make them all work together.

Application Architecture

The challenge in designing the application architecture is determining how to scale the trading application without adding to the overall latency. Some of the typical trade-offs that must be considered are the number of tiers the application will have, centralized vs. distributed processing, and service-oriented architecture vs. event-driven architecture. There is also the trade-off that application developers must make between rapid development cycles and efficient execution.

A common mistake is that application developers test code in lab conditions, sometimes on a single subnet, and do not speak with the networking department until the application is about to go into production. The lack of coordination between the application developers and the network team can lead to inefficiencies in the application performance or the network design.

Another important factor is the type and location of the data store. Some questions to consider: Is the data coming from different sources? Where are these sources and is there sufficient bandwidth to all of these locations? What type of database management system is being used?

Common solutions in application architecture:

- Grid computing is one answer to the issue of reducing application processing time. This works by processing different parts of the application in parallel on multiple physical servers. This solution is used for applications such as risk modeling or other simulations which are part of the middle- or back-office.
- Event Stream Processing (ESP) deals with the task of processing multiple streams of event data with the goal of identifying the meaningful events within those streams with almost no latency. ESP employs multiple techniques, such as: detection of complex patterns of many events; event correlation and abstraction; detection of event hierarchies; understanding the relationships between events such as causality, membership and timing; and, event

driven processes. The job of ESP is to consume multiple streams of event-oriented data, analyze those events to discover patterns, and then act on the inferred events it finds—in milliseconds. ESP is typically used in the front-office applications, such as algorithmic trading engines.

- In-memory database products can reduce data access time by keeping the data in high-speed memory caches. There is a trend to replace traditional databases with real-time, in memory database products.

  The in-memory caching method is also used by clusters, where the data is shared by multiple servers. The data is stored in structures called Distributed Shared Objects (DSO). The use of DSOs ensures that when one server fails, the application state information is instantly available to the redundant server.

Server/OS Architecture

Server hardware and software components, such as the CPU, hard disk, memory, and the operating system (OS), also contribute to overall latency.

Depending on the type of RAM used, typical access times vary from 9 to 70 nanoseconds. In a conventional network stack implementation, data must be copied by the CPU between network buffers and application buffers. This overhead is compounded by the fact that memory speeds have not kept up with increases in CPU speeds. For example, processors like the Intel Xeon are approaching four GHz, while RAM chips hover around 400 MHz. This is a 10:1 ratio in clock speeds. That difference means the processor must wait 10 clock cycles for every cycle it takes the RAM chips to fetch and send the data, when the processor needs to retrieve a data item that is not located in its memory cache. Source Intel: http://www.intel.com/technology/ioacceleration/306517.pdf

The obvious impulse is to throw hardware at the problem: 64 bit processors, multiple CPUs, dual- and quad-core CPUs, faster memory, faster disks. On the other hand, the OS and the applications must support parallel processing, in order to take advantage of the multiple CPUs. Trading systems usually run real-time versions of Linux such as Montavista or RTLinux with finely tuned buffers and process priorities to take advantage of these improvements.

A less obvious solution is to use technologies like Infiniband which combine hardware and software acceleration techniques. High-end servers can be connected via an InfiniBand switch which provides two benefits for low-latency clustering: kernel bypass and Remote Direct Memory Access (RDMA). InfiniBand streamlines the processing of communications traffic so that most of the work is carried out on the network interface card—not in the kernel. This frees server processing cycles to focus on the application instead of communication overhead. In addition, RDMA protocols allow an application running on one server to access memory on another server through the network with minimal communications overhead. This lowers network latency to as little as five microseconds as opposed to tens or hundreds of microseconds for traditional non-RDMA TCP/IP communication.

The concept behind RDMA is analogous to Direct Memory Access (DMA) in traditional Unix architectures. In the latter case, each CPU has its own memory, but other CPUs or devices in the same machine can share access to this memory by taking control of the bus and making the transfer themselves. Otherwise, the main CPU would be tied up copying the data itself. In the case of Infiniband, each server in the cluster has its own memory, but can also access the memory of other servers in the same cluster through the server fabric switch.

Security and Compliance

The fastest network or application can be bogged down by what may seem like *red tape*—security policies implemented in firewalls, intrusion detection and protection devices, encryption, load-balancers, traffic monitors; anything that is a *bump on the wire*.

It is of utmost importance that all the stakeholders collaborate in finding the optimal balance between policy and business agility. A rule-of-thumb in security policy design is that simple is safer. Eliminating undue complexity from both the policy and the network design is the starting point of this collaboration.

From a technical point-of-view, the solution should have as many security features as possible processed in hardware to reduce the processing time.

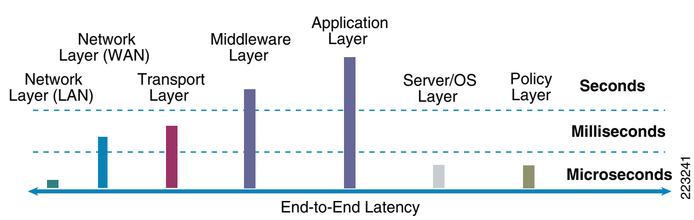
From an architectural point-of-view, the trading functions can be split into DMZ functions and behind-the-firewall functions with separate policies for each category.

From an operational point-of-view, most of the monitoring and event processing can be done out-of-band.

Comparing Latency Effects

The preceding descriptions surveyed the software and hardware elements of a networking implementation that can be sources of latency. The next question is: How do they all compare? Figure 1 illustrates that application and middleware layers have the most impact on overall latency. Their relative effects can be measured in terms of seconds, as opposed to milliseconds or microseconds. The most room for latency improvement—with the greatest impact on performance—is in the application and middleware components.

Figure 1. Comparison of Latency Effects



Low Latency Monitoring

Traditional network monitoring tools operate with minutes or seconds of granularity. Next-generation trading platforms, especially those supporting algorithmic trading, require latencies of less than 5 milliseconds and extremely low levels of packet loss. On a gigabit LAN, a 100 ms microburst can cause 10,000 transactions to be lost or excessively delayed. In these environments, latency must be measured at a much more granular level. Additionally, measurement of latency must break down the end-to-end transaction so that each step can be properly measured. Understanding where the bottleneck occurs is necessary to effectively tune any system.

There are several tools that can be used to measure latency in a trading environment. These include Bandwidth Quality Manager (BQM), IP SLA, and Application Oriented Networking solution (AONS) Latency Monitoring. Brief descriptions of each follow.

Bandwidth Quality Manager (BQM)

Bandwidth Quality Manager (BQM) 4.0 is a next-generation network application performance management product that enables customers to monitor and provision networks for controlled levels of latency and loss performance. While BQM is not exclusively targeted at trading networks, its microsecond visibility—combined with intelligent bandwidth provisioning features—make it ideal for these demanding environments.

Cisco BQM 4.0 implements a broad set of patented and patent-pending traffic-measurement and network-analysis technologies that give the user unprecedented visibility and understanding of how to optimize the network for maximum application performance.

Cisco BQM is now supported on the Cisco Application Deployment Engine (ADE) product family. The Cisco ADE product family is the platform of choice for Cisco network management applications.

More information on BQM can be found at the follow URL: http://www.cisco.com/go/bqm

IP SLA

IP SLA is an embedded network management tool in Cisco IOS that allows routers and switches to generate synthetic traffic streams which can be measured for latency, jitter, packet loss, and other criteria. One device acts as a probe and other devices act as responders. These routers and switches can be production routers in the network or—if there are performance concerns—tests can be run on dedicated, non-productions equipment.

The tests can be configured through the CLI or from a management station using SNMP. Cisco has several different management products that use IP SLA, as well as partners that have developed applications that run tests with IP SLA and report the results.

Today, IP SLA does not support IP Multicast test streams. Multicast support is in the process of being developed.

More information on IP SLA can be found at the following URL: http://www.cisco.com/go/ipsla

Application Oriented Networking Solution Latency Monitoring

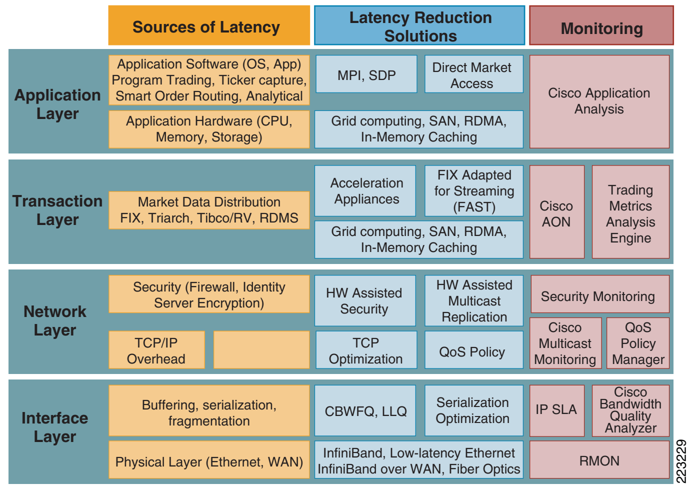
Cisco's Application Oriented Networking solution (AONS) can be used to monitor latency in Financial Services applications. AONS can track the FIX orders as they are sent to the exchange and then measure the latency of the trade acknowledgements that are received from the exchange.

More information on how AON can monitor market data—and then produce reports historically and in real time—can be found can be found at the following URL: http://www.cisco.com/en/US/netsol/ns340/ns394/ns224/netbr0900aecd804b0abe.html

Summary of Latency Design Best Practices

The approach to minimize latency must be a *holistic* effort that takes into consideration the entire market data system from end-to-end and focuses on reducing latency throughout the design. Figure 2 summarizes strategies for minimizing latency based on the respective protocol layer and the sources of latency. It lists the latency reduction solutions available and options for monitoring the effects of latency on performance.

Figure 2. Summary of Latency Management Hierarchy



Corporate Headquarters
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-1706
USA
www.cisco.com
Tel: 408 526-4000
800 553-NETS (6387)
Fax: 408 526-4100

European Headquarters
Cisco Systems Europe
11, Rue Camille Desmoulins
92782 Issy-les-Moulineaux
Cedex 9
France
www-europe.cisco.com
Tel: 33 1 58 04 60 00
Fax: 33 1 58 04 61 00

Americas Headquarters
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-1706
USA
www.cisco.com
Tel: 408 526-7660
Fax: 408 527-0883

Asia Pacific Headquarters
Cisco Systems, Inc.
Capital Tower
168 Robinson Road
#22-01 to #29-01
Singapore 068912
www.cisco.com
Tel: 65 317 7777
Fax: 65 317 7799