Pragmatic Network Latency Engineering Fundamental Facts and Analysis

Rony Kay, Ph.D.
President/CTO cPacket Networks

Low latency networks for distributed applications have become a competitive advantage for financial institutions with algorithmic trading platforms, where delay in trades impacts profits. The latency requirements for applications like high frequency trading, derivative pricing, and latency arbitrage are much stricter than for traditional web applications, such as VoIP and network gaming. While traditional applications can tolerate more than 100 milliseconds of one-way packet latency, algorithmic trading is sensitive to milliseconds, microseconds, or less. Market demand for *ultra low latency* networking is growing rapidly and the time resolution scale has shifted by several orders of magnitude from milliseconds to microseconds and less. Meeting these more stringent performance constraints requires finer resolution and more accurate measurements of latency and jitter than ever before.

Latency is a business and operational issue and must be treated as such. Improving network latency relies on measuring and understanding the underlying trade-offs. This paper provides facts and observations to help with understanding the current situation, identifying critical bottlenecks, and implementing systemic improvements. It aims for a pragmatic overview of *network latency engineering*, while remaining vendor agnostic. The goal is to reconcile vendors' marketing claims with the reality of the fundamental laws of physics and practical technology limitations.

Unlike bandwidth, latency and jitter depend on the specific context of network topology and traffic conditions. Latency variability needs to be monitored with the same diligence that traders monitor changing market prices, correlations, and trends. Otherwise, the variability in network performance could drown out the sophistication of the trading algorithms. Spending money on expensive equipment, bandwidth, and buzzwords with no understanding of the underlying trade-offs is not an effective latency engineering strategy and will lead to poor results.

If you cannot measure network latency, you cannot control it and cannot improve it.

Abbreviated Table of Contents

TABLE OF CONTENT

1 Objective and Scope

The basic concept of "you cannot improve what you cannot measure and understand" must be applied for the optimization of the latency of automatic trading platforms. The goal of the paper is to enhance understanding of latency sources and underlying trade-offs, to facilitate effective measurement and analysis, and to support development of a consistent latency monitoring and improvement strategy.

The L-word buzz has been increasing with the proliferation of network applications with end-to-end latency requirements. Traditionally, applications that have latency requirements include: VoIP and interactive video conferencing, network gaming, high-performance computing, cloud computing, and automatic algorithmic trading. For example, one-way latency for VoIP telephony should generally not exceed 150 milliseconds (0.15 seconds) to enable good conversation quality. Interactive games typically require latencies between 100 and 1000 milliseconds depending on the game genre. However, the requirements for automated algorithmic trading are much more strict. A few extra milliseconds (10^{-3} second), or even a few extra microseconds (10^{-6} second) or less, can enable trades to execute ahead of the competition, thereby increasing profits[1].

Rapid evolution of automated algorithmic trading has boosted the commercial interest in ultra low network latency and in related monitoring and measurement. Algorithmic trading relies on the network connectivity between stock exchanges to data centers, where the automatic trading applications run. The technology platform to support this environment must be reliable, scalable, and provide minimal latency while handling market data speeds of tens of thousands of ticks per second - each millisecond, microsecond, or even less could be a competitive differentiator.

According to the Tabb Group, automatic algorithmic trading and direct market access are the biggest disruptors in modern-day markets. In a report from April 2008 they observe: *"… handling the speed of the market is of critical importance because latency impedes a broker's ability to provide best execution. In 2008, 16% of all US institutional equity commissions are exposed to latency risk, totaling \$2B in revenue. As in the Indy 500, the value of time for a trading desk is decidedly non-linear. TABB Group estimates that if a broker's electronic trading platform is 5 milliseconds behind the competition, it could lose at least 1% of its flow; that's \$4 million in revenues per millisecond. Up to 10 milliseconds of latency could result in a 10% drop in revenues*". According to the Tower Group "*low latency finally comes into its own"*. In an interview with Wall Street & Technology in July 2009, a senior analyst observed that the level of awareness among consumers of ultra-low latency, as it relates to the competitive advantage in electronic trading, has substantially increased in 2009. Furthermore, new regulations put more responsibility on brokers and exchanges to ensure that trades are executed at the best available price.

Distributed trading applications with their ultra low latency requirements, where milliseconds and microseconds matter, coexist with other networking market trends[2] of agile data centers, higher speed LAN, and faster WAN transport. The combination of increasing speed and increasing applications complexity creates new challenges for latency measurement and monitoring. Thus technologies, which might have been sufficient in the past, do not scale to the new requirements of enhanced visibility, higher accuracy, and more real-time reporting.

[1] *As an anecdote, the 100 meter sprint race in the Olympics is measured to the thousandth (1/1,000) of a second and rounded up to the next hundredth of a second: 10 milliseconds and occasionally 1 millisecond can distinguish between win and lose (photo finish).*

[2] *Including emergence of cost-effective 10 gigabits-per-second Ethernet that is lowering the barrier for economical and pervasive deployments of cluster technologies, distributed storage, and clouds.*

Financial institutions are driving development of the low latency market. They are lucrative customers with strong buying power and their interest has sparked marketing hype from vendors of networking equipment and monitoring solutions. Unfortunately, some of the vendors' claims are inaccurate, confusing, and outright misleading.

1.1 Organization

This paper aims to help develop intuition about trade-offs and reconcile marketing buzz with practical technology limitations and the fundamental laws of physics. It is a pragmatic technical overview, while remaining vendor agnostic.

The subsequent sections are organized as follows: Section 2 describes the fundamental theoretical limits and contrasts them with simple illustrative examples from real networks; Section 3 defines latency and jitter, makes a distinction between host latency and network latency, and reviews the various latency sources in actual networks; Section 4 delves into engineering principles and analysis of bandwidth, latency, jitter, and microbusts in the specific context of modern packet switched networks; Section 5 reviews practical considerations and a methodology for one-way and round-trip latency and jitter measurement in high-speed production environments; Section 6 briefly highlights pitfalls of latency measurement solutions; and Section 7 summarizes key observations.

2 Basic Facts and Observations

This section has two parts: first, it describes the fundamental theoretical barrier on how fast a bit of data can move from one location to another. Second, it provides simple illustrative experiments to demonstrate the impact of additional factors, such as packet size and traffic conditions, on the network latency profile.

2.1 Fundamental Speed of Light Barrier

The speed of light is a fundamental constraint of the universe according to the current understanding of the laws of physics. The speed of light in a vacuum is exactly 299,792,458 meters per second, or about 300,000 kilometers per second (186,411 miles per second). The speed of light is as fast as one can move a bit of data from one location to another. Neither the buying power of financial institutions nor equipment vendors' claims can alter the fundamental laws of physics.

Signals in fiber or copper cables can travel at roughly ~70% of the speed of light. The signals slow down in the cable because of the physical construction of the media. For example, when a light signal travels through a transparent material it slows down due to the interaction with the electrons bound to the atoms in the material; the refractive index implies the slow-down factor relative to vacuum - for glass it is about 1.3 to 1.5. Similarly, when an electrical signal travels in a copper cable, it slows down due to the effects of inductance, capacitance, and resistance of the cable. Practically, the signals can travel in the fiber or copper cables at about 70% of the speed of light, which is 210,000 kilometers per second. Presently, the long distance signal transmission technology is more efficient and economical for fiber cables (optical) than for copper cables (electrical).

It takes ~3.3 microseconds in a vacuum for a signal to propagate 1 kilometer at the speed of light (1 kilometer divided by 300,000 kilometers per second). In fiber optic cables the signal slows down to 70%, so the propagation delay for 1 kilometer is about 4.76 microseconds (1 kilometer divided by 210,000 kilometers per second).

Consider the 4,125 kilometers distance between San Francisco and New York. If a fiber cable were stretched in a straight line, it would take the signal about 19.6 milliseconds (0.0196 seconds) to travel that distance. The round trip for one bit of data going from San Francisco to New York and immediately coming back without any processing delay is about 39.2 milliseconds (0.0392 seconds).

The 39.2 milliseconds round trip from San Francisco to New York can put a financial institution with an algorithmic trading platform in San Francisco out of business. Tabb Group asserts that "… *the value of time for a trading desk is decidedly non-linear … if a broker's electronic trading platform is 5 milliseconds behind the competition, it could lose at least 1% of its flow; that's $4 million in revenues per millisecond. Up to 10 milliseconds of latency could result in a 10% drop in revenues*".

The trivial example of a one bit of information travelling from San Francisco to New York and back with zero processing delay at the end-points is hypothetical, but it provides an approximate lower bound. Brokers in San Francisco can draw some practical conclusions:

- If your electronic trading system needs to meet a 5 millisecond round trip delay to the New York exchanges, the automatic algorithmic trading platform must be relocated from San Francisco to within no more than 525 kilometers (~328 miles) of New York City.

- Even if money could buy some "magic" communication equipment, where signals travel at 100% of the speed of light, the round trip of one bit from San Francisco to New York is still 27.5 milliseconds. No capital spending on new equipment can close this gap from 27.5 milliseconds to 5 milliseconds.

The emerging reality of automatic algorithmic trading coupled with the laws of physics imply that an automatic trading platform in San Francisco might not be competitive for trades in New York Exchanges. However, the speed of light is not the only parameter that impacts latency and there are additional factors to consider for optimizing a trading platform.

2.2 Illustrative Experiments

An example of simple measurements of actual network latencies demonstrates that other significant factors beyond the speed of light barrier affect latency.

The following round trip delays are reported using the standard Ping[3] and Traceroute[4] utilities from a host in the Silicon Valley in California to five destinations:

Ping Destination	Round Trip Delay (milliseconds)	Round Trip Min-Max (milliseconds)	Number of hops (from trace route)
www.cmu.edu	100	99-101	14
www.nyu.com	88	87-90	13
www.stanford.com	21	20-24	12
www.google.com	36	35-38	14
www.yahoo.com	19	17-21	9

The ping utility also allows specifying the payload size of the ICMP packet. Following are the results for 100, 500, and 1,000 bytes payload size:

Ping Destination	Round Trip Delay Buffer = 100 Bytes (milliseconds)	Round Trip Delay Buffer = 500 Bytes (milliseconds)	Round Trip Delay Buffer = 1,000 Bytes (milliseconds)
www.cmu.edu	101	108	116
www.nyu.com	90	96	102
www.stanford.com	20	26	34
www.google.com	35	40	46
www.yahoo.com	19	25	33

The data provided in the tables shows that:

- The reported round trip delays are much longer than the signal travel time in the cable. For example the round trip from Silicon Valley to Pittsburgh is more than twice the signal travel time at 70% of the speed of light. Compare the 100 milliseconds round trip to CMU in Pittsburgh and 90 milliseconds to NYU in New York with the

[3] *Ping and Traceroute are standard utilities in most operating systems. Ping sends a special packet to the destination, which immediately sends a response back. It provides only coarse measurement, which may not be sufficient for trading applications.*

[4] *The number of hops counted by Traceroute is only an approximate number of router hops between the source and destination and is limited to the path in one direction and only for a sample of the available paths.*

calculated 39.2 milliseconds round trip to New York. Clearly, the speed of light is only a small portion of the measured latencies. The speed of light is much less than 50% for long distances and much less than 20% for short distances.

- The reported round trip delays depend on the packet size. The packets' size seem to have a larger relative effect, e.g. 50%, on the nearer destinations (Stanford, Google, Yahoo), than on the far destinations (CMU in Pittsburgh, NYU in New York).

- The number of reported hops is not directly correlated to the physical distance and to the total delay.

- The round trip delay time is not constant; it varies within a wide range. There seems to be a non-deterministic factor that impacts latency variability (jitter).

Figure 1 shows the round trip delays of ping packets over a period of three hours. The pings were sent (using *fping*) every three seconds and the data was collected and analyzed for 30 seconds windows (i.e. 10 data points in each time window), including: average, maximum, and minimum delay per window. The chart shows the average and the maximum delays over three hours period.

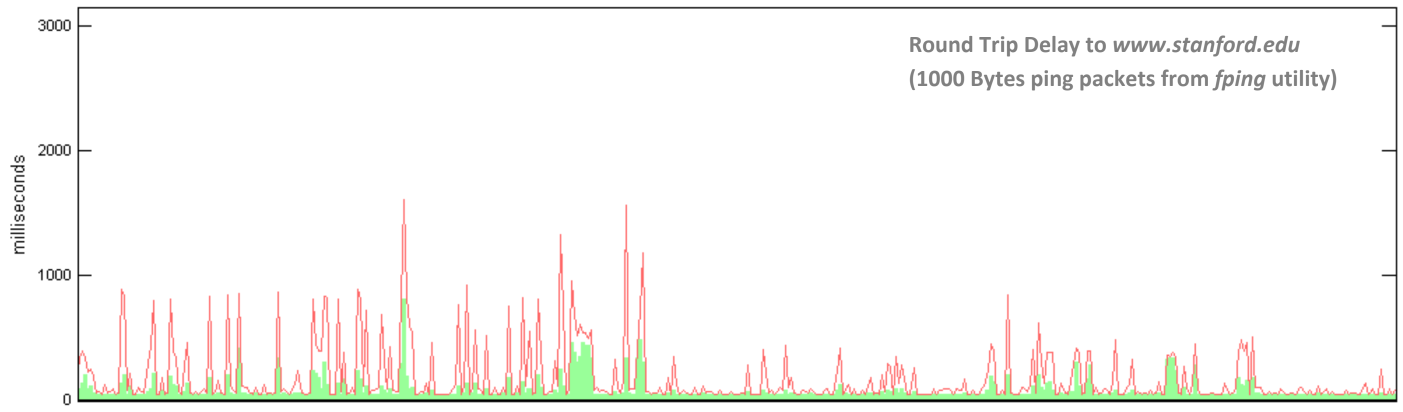

Figure 1: Roundtrip delays over a period of about three hours at 3 seconds intervals. The light green is an average of 10 data points over 30 seconds and the red line is the maximum over the same window of 30 seconds.

More than 97% of the 30 seconds time windows had a minimum delay of less than 40 milliseconds; more than 33% of the windows had a maximum delay of more than 100 milliseconds; and about 14% of the windows had a maximum delay of more than 400 milliseconds. Detailed examination revealed about 12% of the windows had a spread of more than an order of magnitude. Namely, more than 12% of the 30 seconds windows included *both* a delay of less than 40 milliseconds and a delay of more than 400 milliseconds. Those large variations (10x) and frequent occurrences (above 10% of the time) create a serious concern for electronic trading applications.

The next section defines terminology and discusses deterministic and non-deterministic latency sources.

3 Definitions and Latency Sources

3.1 Overview

Latency is the elapsed time (*delay*) between event A and event B. In this paper, the term *latency* refers to the delay between sending and receiving a message over a packet-switched network. Generally, the overall application latency is comprised of the time to process messages at both the sending and receiving hosts (*host latency*) and the delays which occur inside the network (*network latency*).

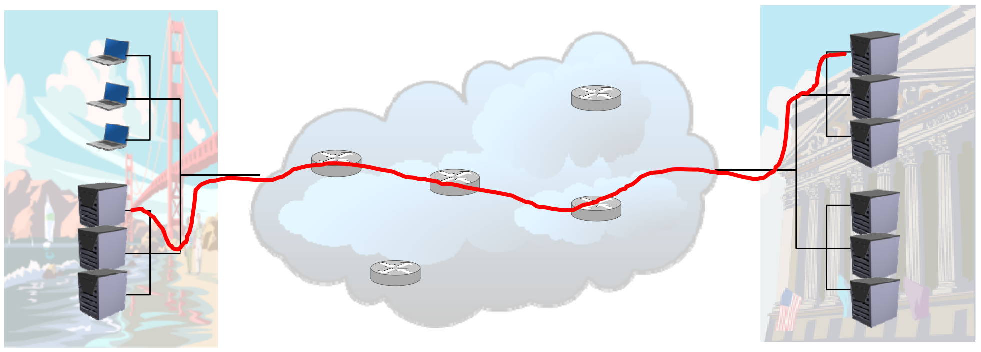

Figure 2: One-way network latency is the elapsed time (delay) between sending a message from one computer to receiving it at another computer; the red line illustrates a packet path between the computers.

The data (messages) sent between the hosts is encapsulated in the payload of network packets (for the purpose of this paper the terms *packet* and *frame* refer to abstract data units and are conceptually interchangeable)[5]. Different applications and messaging protocol implementations differ in how they encapsulate data inside packets, e.g. each packet can include a message, a part of a message, or a few messages. The specific packet structure is application and protocol dependent. However, in order to deliver any message, at least one packet (data unit) needs to be sent over the network communication channel. The observations in this paper are not limited to any specific application or protocol suite.

3.2 Formal Definition of Network Latency

The overall performance of distributed applications depends on the hosts and the network connectivity between them. *Network latency* is the delay that is introduced by the network; it excludes the hosts' software processing at the source and destination end-points. Network latency can be measured either *one-way* (the time from the source sending a packet to the destination receiving it), or *round-trip* (which is the sum of the one-way latency from source to destination plus the one-way latency from the destination back to the source).

Definition: *network latency* of packets is the delay from the time of the *start* of packet transmission at the sender host to the time of the *end* of packet reception at the receiver host.

Some literature and tools define network latency as the delay from the time of the start of packet transmission at the sender to the start of packet reception at the receiver. The different definitions can lead to substantial discrepancies in

[5] *For TCP/IP over Ethernet the terms packets and frames usually mean Layer 3 and Layer 2 data-units respectively. The Layer 2 Ethernet frames encapsulates the Layer 3 IP packets.*

certain situations. To avoid confusion, be sure to interpret latency metrics correctly according to specific and relevant definitions. It is also important to distinguish between network latency and propagation delay: propagation delay is the travel time of one bit from one side to the other inside of the cable, while network latency is the time to deliver an entire data unit (packet) from one host to the other.

In general, it is easier to measure round-trip latency than one-way latency, because both the send and receive events occur at the same physical location. Yet, round trip delays provide no information about asymmetric latency in both directions. Furthermore, in trading environments it is often more useful to benchmark *one-way delay*, because market data is received via one communications path from the exchange and orders are sent to the exchange from the broker's system via an independent path. A critical challenge in measuring one-way delay is how to accurately correlate event timing from distinct physical locations.

3.3 Network Performance Metrics

Bandwidth is perceived by many users as the primary measure of network *performance*. It is an absolute and convenient figure of merit, which depends only on the equipment and can be measured in isolation. In contrast, network latency is context dependent and hard to characterize. Usually, network equipment vendors prominently publish their interface bandwidth (of switches, routers, and gateways), while remaining "shy" about latency and jitter characteristics. This section explains nuances that should be taken into consideration when analyzing vendors' claims.

Bandwidth in computer networks represents the overall *capacity* of a connection as the amount of data that *can* pass via that connection over a time period - it is measured in bits-per-second (bps). *Throughput* or *maximum throughput* is sometimes used as a synonym for bandwidth, but the terms are not equivalent. While bandwidth is the theoretical *capability* of the connection, the throughput is the *actual* data rate of a specific application.

The *actual throughput* is the amount of data that passes via the connection over a *time period*. In practice, it is important to set the time period according to the relevant context. If the period is too long, it averages out spikes and bursts. Figure 3 shows that examining average throughput over windows of 10 seconds indicates "healthy" 50-60% bandwidth utilization (in yellow color). But the 10 second averages conceal the fact that about 10% of the time, the link capacity is saturated to the maximum available throughput, as shown by the red line that represents the maximum rate per second during each 10 seconds window. Unfortunately, common monitoring tools are often limited to reporting averages over long intervals (e.g. hour, 5 minutes, or 30 seconds) that hide link saturation and spikes. Observe that the longer the averaging period is, the less information it provides about temporal variability, spikes and bursts which may impact network latency and jitter.

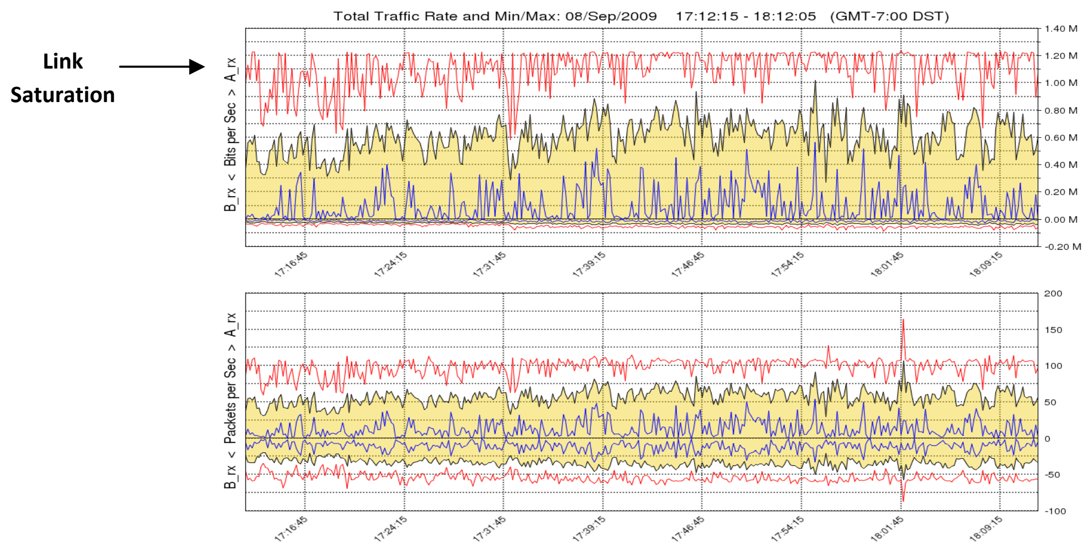

Figure 3: One hour chart of the effective throughput (top) and packet rate (bottom) of Internet access of a small office. The yellow color represents average throughput for 10 seconds windows. The red line represents the maximum throughput and the blue line the minimum throughput over 1 second period (i.e. max and min per second for each of the 10 seconds window).

In computer networks, *bandwidth* is an upper bound on *maximum available throughput.* The available throughput is implementation dependent. For example, consider a connection between two hosts based on IP packets with encapsulated UDP[6] protocol over an Ethernet link. The protocol layers consume bandwidth and reduce the maximum available throughput for effective data rate. The ratio between the bandwidth and the maximum data throughput is shown in the following table for three packet sizes:

Protocol Overhead	Description	64 Byte Frame/Packet	512 Byte Frame/Packet	1500 Byte Frame/Packet
Layer 1	Inter-frame gap and preamble: 20 Bytes	23.81%	3.76%	1.32%
Layer 2	MAC header and CRC (no VLAN): 18 Bytes	21.43%	3.38%	1.18%
Layer 3	IP header (no option fields): 20 Bytes	23.81%	3.76%	1.32%
Layer 4	UDP Header: 8 Bytes	9.52%	1.50%	0.53%
Payload	Application Data payload	**21.43%**	**87.59%**	**95.66%**
Total		100%	100%	100%

* Percentage are rounded to 2nd decimal digit.

The pie charts in Figure 4 show the maximum effective data throughput as a percentage of link bandwidth. The maximum data throughput varies widely from 21.43% for 64 bytes packets to 95.66% for 1500 bytes packets.

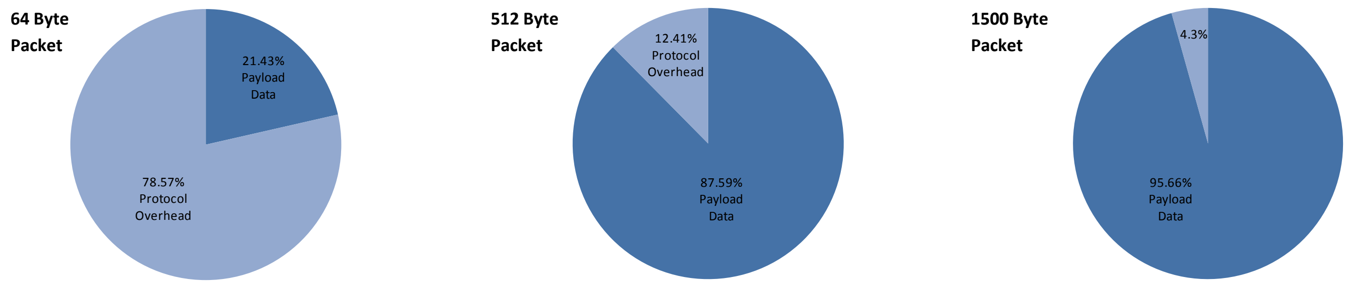

Figure 4: Ratio of effective data throughput (in dark blue) and protocol overhead (in light blue) for different packet sizes. The maximum effective data throughput for 64 bytes packets is only a small fraction, 21.42%, of the connection bandwidth; for 1500 byte packets the maximum effective data throughput is more than 95%.

The bandwidth is an upper bound that approximates the *maximum* data throughput, but it can lead to substantial inaccuracy. For example, estimating the performance of data transfer in a distributed application that transmits many small data units at high rate over a short distance can lead to discrepancies if the bandwidth is used instead of the data throughput.

Keep in mind that dependencies of bandwidth and latency are context specific; improving the bandwidth may or may not be sufficient to improve latency. In practice, improving latency is harder than improving bandwidth. Linear scaling of bandwidth can be simply achieved by *bundling* several identical links in parallel, but it will not improve the network latency *per* packet. Furthermore, even upgrading the native physical link bandwidth may or may not have a significant impact on network latency as shown in Figure 7 and Figure 8. Conversely, reducing excessive latency bottlenecks along the packets' path can improve the actual throughput by eliminating time-outs and retransmissions.

Network Jitter is another important figure of merit related to measuring network performance: it measures latency variations caused by congestion, route changes, queuing, etc. The jitter metric applies to packet sequences (streams) between two observation points. The jitter measures network performance *consistency*; when a network does not

[6] Note that *TCP incurs a higher protocol overhead.*

introduce any variability, the jitter is zero. Real networks introduce substantial jitter and the next section describes the fundamental sources of non-deterministic network behavior.

3.4 Life of a Packet: Application Latency and Network Latency

Packets are forwarded from source to destination and traverse multiple network links including LAN switches, gateways, and routing *hops* as shown in Figure 5 below. The combination of various latency sources such as buffering, propagating, switching, queuing, and processing delays produces a complex and variable network latency profile.

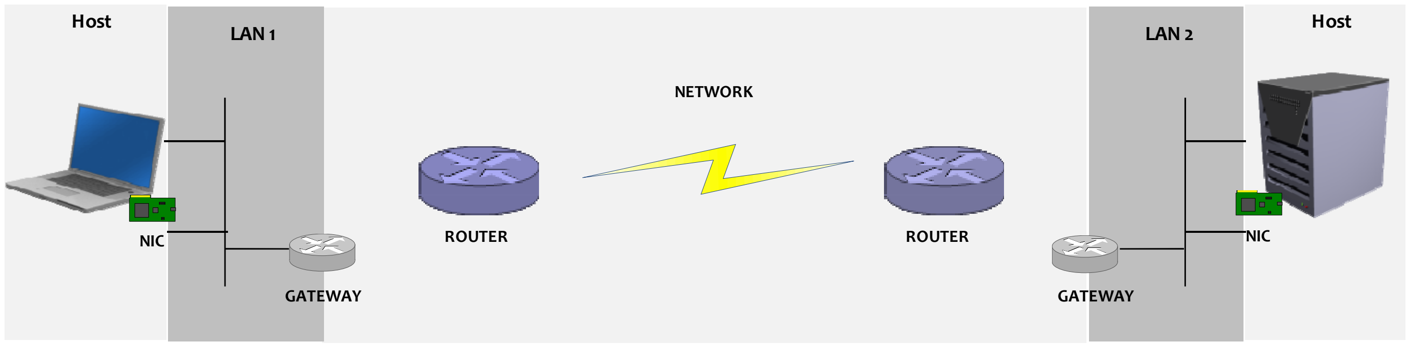

Figure 5: Packets traverse a path through the LAN and gateway at both the sender and receiver sides and routing hops in between; the hosts' network interfaces are the demarcation between the host domain and the network domain.

A simplistic description of the life of a packet is as follows:

- At the sender host, a software application sends data to the receiving host via the socket API of the host operating system.
- The operating system packetizes the data and puts it in a hardware memory buffer.
- The network interface hardware encodes the bits in the corresponding memory buffer and transmits modulated signals over a cable to the next hop.
- At the network, elements like switches, gateways, and routers forward the packet from hop to hop toward the receiving host.
- At the receiving host, the signals reach the network interface, where they are decoded and the data bits are placed in a hardware memory buffer.
- The operating system de-packetizes the received data.
- The software application gets access to the received data in the corresponding hardware memory buffer via the socket API of the host operating system.

This simplistic description abstracts many details of operating system's network stack (e.g. TCP/IP) and low level networking[7]. Conceptually, packetized data is sent and received via hardware memory buffers that serve as the *demarcation* points between the host software (application and operating system) and the network. At those demarcation points, the hosts hand the control over the packet to the network. The host software application does not begin to process a packet until the end-of-packet is received and the packet is *fully* stored in a memory buffer.

[7] *Typically, the sending and receiving hosts are using a standard or customized network stack, such that the host operating system abstracts the lower level networking details from the programmer via the socket API.*

The latency of a distributed application depends on the performance of the hosts and the network transit time (*overall latency = host processing latency + network latency*). The next section describes specific network latency sources in more detail.

3.5 Network Latency Sources

Understanding the sources of network delays is important for effective latency engineering and it is specifically important for optimizing algorithmic trading platforms. Importantly, different environments may require different latency optimization approaches.

A distributed application's performance depends on the host processing time and the delays introduced by network communication. The host processing time is a combination of the application logic and the operating system services. Exchanging data over the network adds latency which impacts overall system performance. Recall that the network interface memory buffers are the demarcation points between the host domain and the network domain.

The sources of network latency include:

1. *Network interface delay* for serialization, including signal modulation and data framing (packetizing). It is the time to convert data packets into or from signals in a physical link to bits in a dedicated memory buffer. The interface delay depends on the size of transmitted packets and varies with link bandwidth: it is equal to the size of the data-unit divided by bandwidth. For example, over 1 gigabit per second Ethernet (GigE) it takes 10 microseconds to receive a 1,250 Bytes packet (10,000 bits) from the physical connection. In comparison, over a T1 link of 1.544 megabits per second the same task takes 6.477 milliseconds (i.e. 6,477 microseconds).

2. *Network processing delays* are incurred while gateways, firewalls, or other network elements like switches and routers determine what to do with a newly arrived packet (e.g. filtering, encapsulation, MTU fragmentation, etc.). The delay depends on the network equipment technology and the specific processing function.

3. *Signal propagation delay* is the time it takes for signals to travel in the physical cable. As discussed earlier, the propagation time depends on the distance and is governed by the fundamental laws of physics. The rule of thumb is that the propagation delay in cables is about 4.76 microseconds per kilometer (~70% speed of light). Therefore, 100 kilometer one-way propagation delay in the cable is 0.476 milliseconds and the round trip propagation time is almost one millisecond.

4. *Inherent router and switch delay* is the time to shift packets from an ingress port to an egress port. The switch (router) converts signals at the ingress port (optical or electrical) to bits and packets, buffers data, performs a look-up to determine an egress port according to the data-unit's destination address (Layer 3 for router or Layer 2 for switch), and forwards the packets from the egress port by converting the bits back to modulated signals. The delay depends on the switch architecture (store and forward or cut-through), look-up time, and other hardware implementation details. It also depends on the switch/router interface speeds which impact the associated interface serialization delay (similar to item 1 above for the hosts' network interfaces)[8].

5. *Queuing delay* occurs in switches and routers when packets from different ingress ports are heading to the same egress port concurrently. Since only one packet can be transmitted at a time from the egress port, the other

[8] *The delay over routers and switches includes look-up, forwarding, and data serialization at the interfaces.*

packets must be queued for sequential transmission. This resource contention is called head-of-line blocking and the queuing can lead to substantial latency.

The total end to end *network latency* is the sum of all the delays of the links between the sending and receiving hosts and the switches and routers in between.
Figure 6 is the summation of latency sources along the packet path for a total end-to-end network latency:

$$\mathit{Network\ Latency} := \mathit{sender} + \mathit{receiver} + \sum_{\substack{\mathit{switches\ ,routers,} \\ \mathit{gateway,,appliances} \\ \mathit{in\ path}}} (\mathit{processing} + \mathit{forwarding} + \mathit{queuing}) + \sum_{\substack{\mathit{links} \\ \mathit{in\ path}}} \mathit{propagation}$$

Figure 6: Total network latency is the sum of the sender and receiver interface delays, the network elements along the path (including: processing, switching, and queuing), and the propagation delay in the links along the path.

The combination of multiple network latency sources produces a complex latency profile. The relative contribution of different latency sources to the overall performance of distributed applications is implementation specific; it depends on the software implementation, hosts, network hardware, protocols, and configuration. Different optimizations apply to different situations, according to the specific context.

4 Engineering Considerations

This section covers latency engineering concepts in more depth, including: modeling of basic tradeoffs, sources of latency variability and jitter, impact of head-of-line blocking phenomena, microbursts at output ports, and latency context for switch features like cut-through and non-blocking.

4.1 Simple Latency Trade-Off Model

The simplified model in this subsection assumes an "ideal" sender, which transmits 1,250 bytes (10,000 bits) packets to the receiver over a one kilometer one gigabit per second link (1G). The network latency, from the start of transmission of a packet to the end of receiving it, is the sum of 4.76 microseconds propagation delay (1 kilometer at 210,000 kilometers per second) and the network interface delay of 10 microseconds (10,000 bits divided by 1,000,000,000 bits per second), for a total network latency of 14.76 microseconds.

For the same simplified model, assuming a 10 gigabit per second link (10G), the latency reduces to 5.76 microsecond which is about 60% of the latency of 1 gigabit link. The total network latency is the sum of 4.76 microsecond propagation-delay and 1 microsecond network interface delay (10,000 bits divide by 10,000,000,000 bits per second).

The chart in Figure 7 summarizes the network latencies for the one kilometer fiber cable for both the 1G and 10G line rate. Upgrading from 1G to 10G improved the latency by a factor of three:

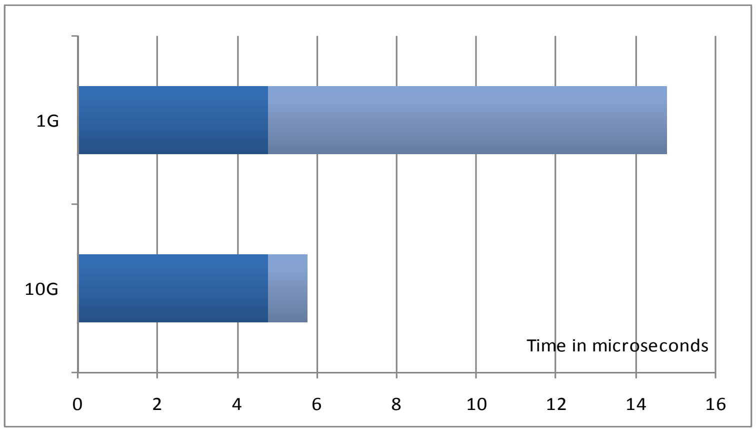

Figure 7: For one kilometer direct fiber link, propagation delay (dark) and network interface delay (light), 1,250 bytes packet. At 1 Gbps (Gigabit per second) and 10 Gbps. The propagation delay remains constant. The overall latency reduction by upgrading from 1 Gbps to 10 Gbps is ~60%.

Similarly, consider a 100 kilometer distance of direct fiber cable connection and ideal transmitter of 1,250 bytes packet size. The network latency over the 1 gigabit per second connections is 486 micro seconds (476+10), and the latency over the 10 gigabit per second connection is 477 microseconds (476+1). It implies that in this context, upgrading from 1 gigabit per second to 10 gigabit per second yields less than 2% latency improvement.

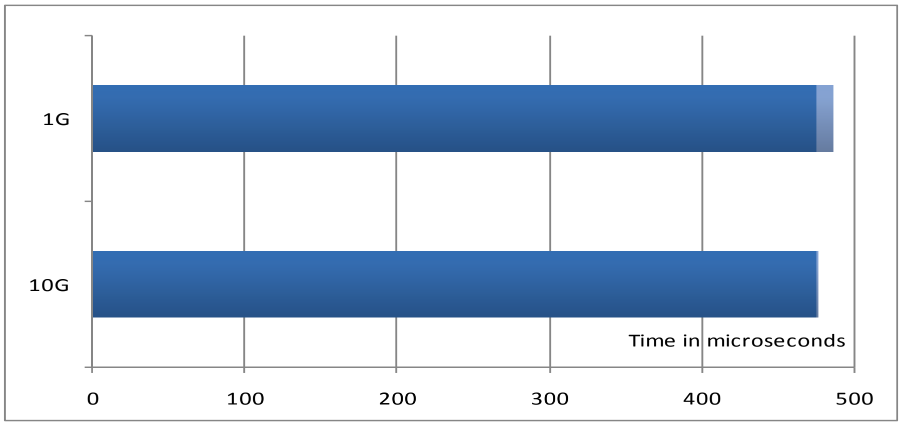

Figure 8: For 100 kilometer direct fiber link, propagation delay (dark) and network interface delay (light) at 1 Gbps (Gigabit per second) and 10 Gbps. The overall improvement from upgrading from 1Gbps to 10 Gbps is less than 2%.

The simple examples (Figure 7 and Figure 8) highlight the context dependent trade-offs:

- In the first example, for 1 kilometer distance, upgrading the connection from 1 gigabit per second to 10 gigabit per second reduced the network latency by about 60%, from 14.76 microseconds to 5.76 microseconds.

- In the second example, for 100 kilometer distance, upgrading the connection from 1 gigabit per second to 10 gigabit per second reduced the network latency by less than 2%, from 486 microseconds to 477 microseconds.

- In both examples, the network latency improvement from upgrading the connection speed by a factor of 10 was much smaller (less than 60% and 2% respectively).

These simplified examples utilize a direct cable connection, but in real networks packets will traverse multiple links and network elements. The switches, routers, and links between the sending and receiving hosts have a cumulative latency effect. Switching latency includes constant forwarding delays and variable queuing delay. Under certain network conditions, even at a low or moderate load, switch latency can vary drastically from less than a microsecond to more than 100 microseconds. Network elements such as firewalls, intrusion prevention systems, WAN optimization, etc. can introduce additional latencies of tens of milliseconds (or more).

Effective latency engineering relies on understanding the latency profile in the context of real (relevant) traffic.

4.2 Variability of Latency and Network Jitter

The actual network behavior is nondeterministic, as evident from browsing a web site at different times and experiencing different response times (also see Figure 1). The general observations in this section about latency variability apply to any packet switched network, regardless of specific equipment or protocol[9].

The nondeterministic network latency depends on:

1. *Practical implementation factors* related to networks being a *complex dynamic system* with many "moving parts" including: applications, equipment, configuration, people, etc.
2. *Fundamental factors* related to *mathematics of stochastic models* of interdependent queues with random arrival times, subject to head of line blocking.

The practical implementation factors include interdependent applications, physical links, protocols, packets, routers, switches, hosts, software, bugs, people, and processes, which are coupled in a complex circular cause and effect. The random variability includes inter packet arrival time, packet size mix, human intervention, changing routing paths, ARP table aging, TCP back-off, network congestion, etc.

The fundamental theoretical[10] variability is due to the underlying *many-to-one* queuing structure. A switch (router) connects multiple devices to each other: it has multiple input ports and output ports, such that incoming data-units (packets, frames) are channeled from input ports to output ports taking the data unit toward its intended destination. The output port for each data unit is determined via table look-up of a destination address that is embedded in the data unit. Generally, a packet can be forwarded from any input port to any output port based on the look-up result. Since only one packet at a time can be transmitted from each port, other packets that arrive during the transmission must be queued (buffered) and wait for their turn.

Switching is susceptible to *oversubscription*. For example, when two input ports forward all the packets to the same output port, it may be oversubscribed with more traffic than it can sustain. Oversubscription can overwhelm the hardware packet buffers and lead to packet loss: if the queue exceeds the size of the physical hardware buffer, packets are dropped. But beyond oversubscription, switches and routers introduce substantial latency variability and jitter even under moderate or low traffic conditions, as shown in the next subsection.

4.3 Fundamental Head-of-Line Blocking Phenomenon

At any time, only one packet can be transmitted from each physical output port of a switch. Resource contention might happen when two packets arrive from separate input ports to the same output port (e.g. uplink) at about the same time. When the output port is busy transmitting a packet, other packets need to be queued (buffered) waiting their turn. The phenomenon is called *head of line blocking* and it leads to inherent latency variations and jitter, even at low traffic load. This inherent jitter can be substantial and may cause an avalanche effect at subsequent network hops.

[9] *The fundamental observations apply to packet switched technology (including Infiniband and storage oriented protocols). Indeed, credit based scheduling aims to manage the congestion (e.g. by managing virtual circuits), but from a holistic system perspective the inherent variability is simply pushed elsewhere into the overall system.*

[10] *The mathematical model applies to both switches and routers, for convenience we use the term switch more often. Also the terms data unit, packet, and frame can be used interchangeably for the purpose here and for convenience we usually use the term packet.*

To analyze the head of line blocking we consider a theoretical queuing model of switching functionality as illustrated in Figure 9 below.

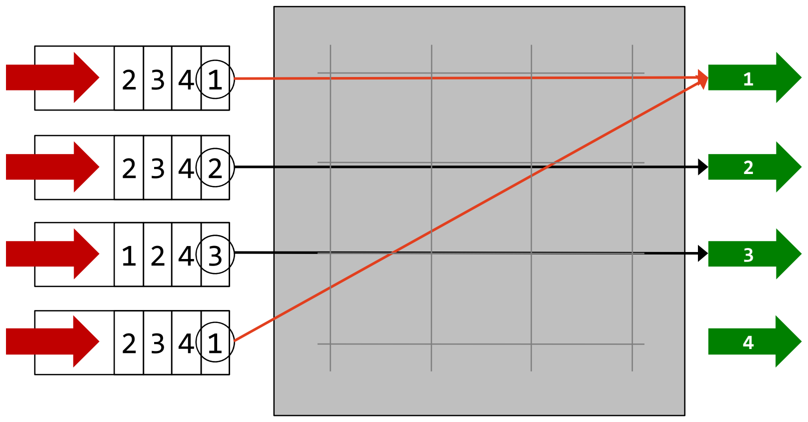

Figure 9: Model of a switch with four inputs (ingress ports) and four outputs (egress ports). The packets arrive at input ports; the switch forwards each packet to a specific output according to a look-up of a destination address embedded inside each packet. The thin red lines show two packets that are forwarded to the same output at the same time and cause head of line blocking.

The jitter at the output ports is a property of a queuing system with random inter-arrival times. A Matlab simulation model enables us to perform quantitative analysis of the queuing behavior[11]. The model can generate pseudo random input streams according to configurable parameters. It assumes an ideal switch implementation with no extra overhead for look-ups or other practical constraints. The simulation is an approximation that computes a switch latency profile under various conditions.

For example, consider a switch with four input ports and one uplink, where each input port is at stable 16.6% bandwidth utilization and the uplink is at aggregated 66.7% bandwidth utilization. The computed latency spread between minimum and maximum is large: at uniform distribution of packet sizes between 64 to 1518 bytes, the latency spread is over 50X. At a discrete even distribution of 64, 512, and 1518 bytes packets, the simulated latency spread is over 60X[12].

Similarly, for three input packet streams at less than 20% bandwidth utilization per input port and 60% output bandwidth utilization the computed latency spread is in the following table:

MATLAB Model	Simulation	Min (microsecond)	Average (microsecond)	Max (microsecond)
3 inputs -> 1 output (each input at 20% uplink at 60%)	Uniform 64 – 1518	0.7	8.4	33.08
	Discrete 64, 512, 1518	0.67	9.3	36.35

Note that the number of input ports impacts the latency spread at the output. For six input ports at 10% bandwidth utilization, the simulated latency spread at the uplink is likely larger than for three ports at 20% bandwidth utilization (while the uplink bandwidth utilization remains 60% in both). The intuitive reasoning is that the probability of the head of line blocking at the output increases with the number of inputs.

[11] *Note that the simulation model applies to both store-and-forward and cut-through switch chip implementations.*

[12] *In this subsection, the minimum and maximum are measured in simulation of 10,000–100,000 packets.*

In addition to simulations, measurements were conducted for actual switches using the experimental setup shown in Figure 10.

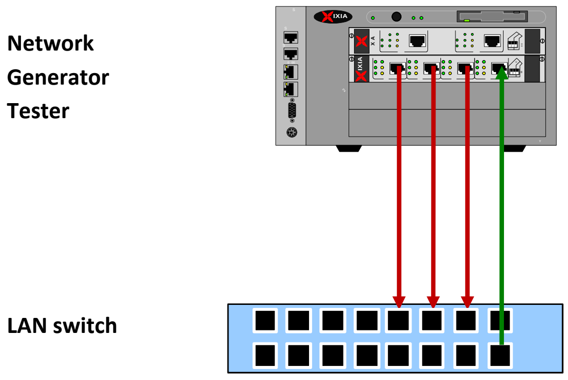

Figure 10: Experimental setup for measuring switch latency in a controlled lab environment. The traffic generator injects synthetic traffic stream with the desired characteristic to three input ports (brown) and it receives the traffic back from one output port (green).

The experimental results are given for three GigE LAN switches from different vendors with 8, 24, and 48 ports. Results are summarized in the following table. The experiments include uniform packet size distribution between 64-1518 bytes and discrete IMIX[13] packet size distribution.

3 inputs -> 1 output (input at 20% uplink at 60%)	Lab Test	Min (microsecond)	Average (microsecond)	Max (microsecond)
Switch vendor 1 (8 ports)	Uniform 64 – 1518	2.22	8.72	39.50
	Discrete IMIX	2.24	7.71	66.68
Switch vendor 2 (24 ports)	Uniform 64 – 1518	3.88	9.22	40.82
	Discrete IMIX	3.88	8.57	64.44
Switch vendor 3 (48 ports)	Uniform 64 – 1518	3.46	10.52	40.20
	Discrete IMIX	3.44	9.12	66.48

Both, the simulation results and the hands-on lab measurements show large latency spread between the minimum and the maximum values. The actual measurements differ from the simulation results because the actual hardware introduces implementation overhead of look-ups, pipelining, physical interfaces, etc. The results of the simulation are consistent with the actual measurements.

Figure 11 summarizes the measured latency spread for a 24-port GigE LAN switch (Switch vendor 2) as a function of the traffic load[14]. It shows that the variability is small when one input port is transmitting all the traffic to one output port at *full* line rate (1-to-1), if the other ports are quiet. But the variability increases drastically when three input ports forward traffic simultaneously to one output port (3-to-1), even for light traffic load.

[13] *IMIX flavors have been defined for network testing tools like Agilent, Ixia, and Spirent. The traffic mix contains discrete frame sizes in a ratio to each other, which is supposed to approximate the composition of frame sizes in the real Internet.*

[14] *Note that latency of modular chassis switches with multiple line cards is typically worse than latency for simple pizza box switches like the ones tested here. The pizza boxes typically have simpler hardware (e.g. single switch chip) and stripped down functionality.*

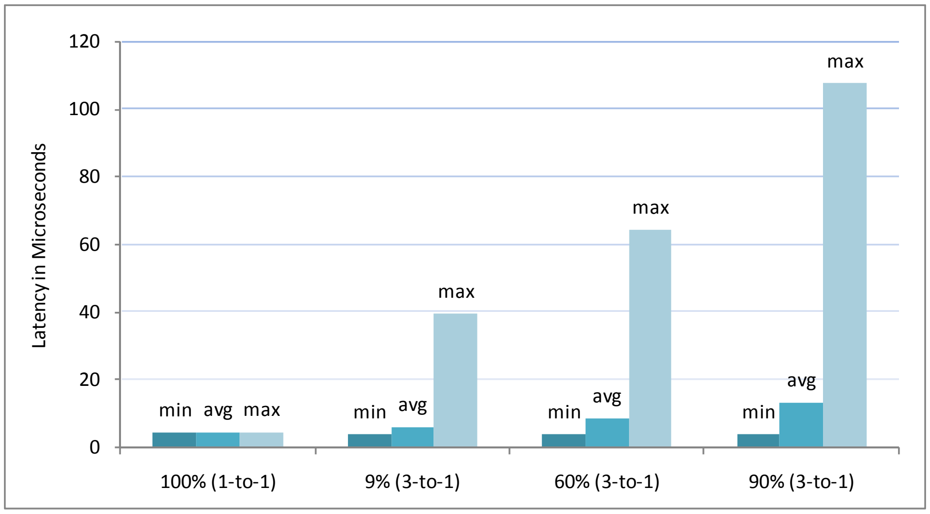

Figure 11: Impact of traffic load on the latency of a 24-port GigE LAN switch. It shows average/minimum/maximum for different traffic loads. The first (left) scenario is one input port forwarding fixed size packets at 100% line rate to one output port, when all other ports are quite. The other scenarios are for three input ports forwarding traffic streams with no spikes to one output port (3-to-1), such that the aggregated bandwidth utilization at the output is 9%, 60%, and 90% respectively. A high variability can significantly impact algorithmic trading applications.

In particular, the traffic dependent jitter needs to be taken in account when designing adaptive trading clusters, which react to load by adding computing resources. Due to the increased fan-in and corresponding random jitter, the cure may be worse than the disease.

Rigorous simulations and lab measurements show that switch latency variability and jitter are inherent and significant (order of magnitude or more between min and max latency). Low bandwidth utilization does not eliminate the jitter caused by head of line blocking and queuing. Jitter can be significant even for low and moderate traffic load (e.g. 9% to 60% range).

4.4 Spikes and Microbursts

Even at low traffic, the head of line blocking phenomenon causes queuing. The output port transmits the queued packets in bursts of back-to-back packets. It implies short periods where the instantaneous bandwidth can reach maximum utilization (e.g. 100%), which can cause operational hazards by pushing equipment to its operational limits.

Microbursts were accurately measured for an experimental setup, as shown in Figure 12. A network generator transmitted three input streams at stable 16.6% bandwidth utilization with no spikes to three input ports of the device under test, which forwarded all the packets to one output port; so the average output bandwidth utilization is 50%.

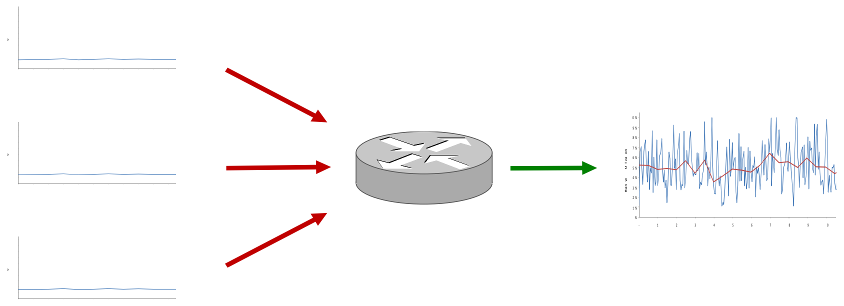

Figure 12: Three inputs streams of random size packets at 16.6% bandwidth utilization to one output port at aggregated 50% average utilization. The 50% average hides the bursts and jitter caused by the head of line blocking and queuing.

Figure 13 shows granular measurements[15] for a 24-port GigE switch. Despite incoming traffic with no spikes and moderate output utilization of only 50%, a detailed analysis shows multiple spikes at 100% utilization.

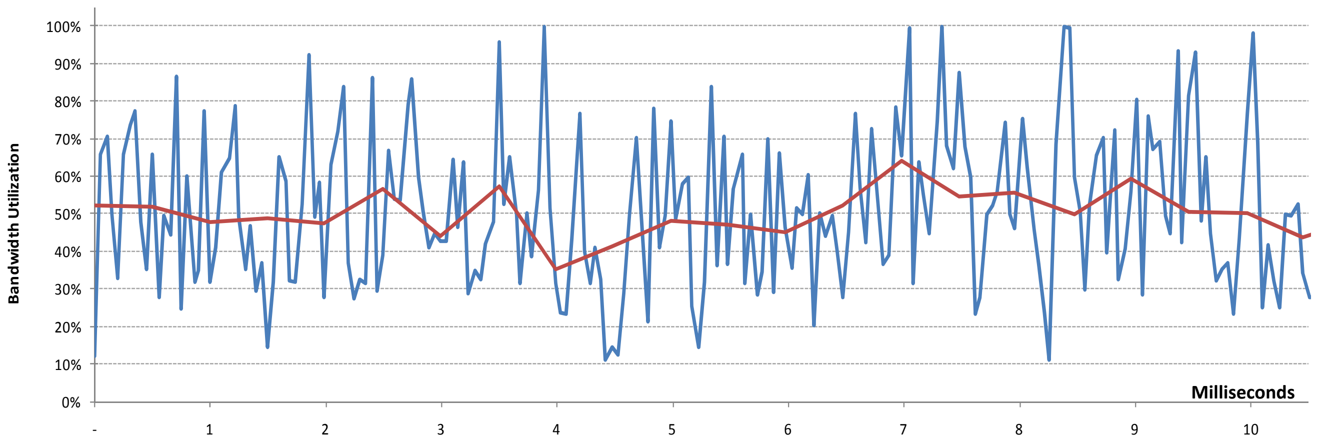

Figure 13: Accurate measurements at an output of a 24-port LAN switch. While the aggregated average utilization is 50%, the instantaneous traffic spikes to 100% utilization .The bandwidth utilization over sequences of 100 packets is in red and over shorter sequences of 10 packets is in blue. The averages conceal the micro behaviors.

Earlier, it was shown that averaging traffic over 10 second periods can hide link saturation and spikes that are apparent at *one* second granularity (see Figure 3). But Figure 13 illustrates a higher resolution view, where the whole X axis represents a 10 millisecond period.

[15] *Measurements were taken at the hardware transceiver (PHY chip) with no intermediaries or buffers that hide the bursty behavior; cPacket hardware probes provided the required measurement accuracy.*

The microbursts can cause an avalanche effect, amplification, and packet loss in subsequent hops, so that hosts at the end-points may experience inconsistent performance and packet loss with no visibility to the root cause. Therefore, continuous on-the-fly monitoring in real time is needed in order to identify trends, generate alerts, support corrective actions, and minimize adverse effects on the business operation[16].

4.5 Switch Features in Latency Perspective

The previous sections prove that network latency is context dependent. This subsection reviews terminology like "cut-through" and "non-blocking" switches and how it pertains to the latency profile.

Cut-through switches are designed to begin forwarding of packets from an input to an output before the end of the packet is received and soon after examining the destination address at the packet header. On the other hand, a store-and-forward switch receives the entire packet into a buffer before it begins to transmit it from the output. If the actual traffic does not cause head of line blocking, the cut-through mode has lower latency than store-and-forward mode by ~one packet buffering delay (at most). Recall that the packet buffering delay can be estimated by dividing the packet size by the link bandwidth; for example, at 1 Gbps the buffering delay for 1,250 bytes packet is 10 microseconds and at 10 Gbps the buffering delay is 1 microsecond. Modern switch chips often implement some hybrid architecture of cut-through and store-and-forward, where cut-through applies when there is no head of line blocking. Keep in mind that cut-through switches are susceptible to the inherent latency variability and jitter like any other switches.

Nonblocking switching was historically used for telephony[17] circuit switching and meant that enough paths are available to allow any call (input) to reach a line (output) without encountering a busy condition. In the packet switching world (including Ethernet, SCSI, Infiniband), nonblocking performance applies to traffic patterns, where input and output ports are (temporarily) paired to allow the input to forward traffic at full line-rate to the corresponding outputs. The non blocking performance excludes head-of-line blocking conditions. Namely, for a certain time period the traffic is subject to a rigid one-to-one topology. Such one-to-one pairing, however, does not cover situations where two (or more) hosts need to concurrently send packets over the same uplink, which is very common in IP networks. Usually, IP networks aggregate links to uplinks and rely heavily on fan-in/fan-out topologies; as such, head-of-line blocking is a common occurrence. Therefore, latency and jitter can vary drastically also for nonblocking switches.

Generally speaking, modular chassis-based switches with multiple line cards cause more latency than single-chip pizza box switches that have simpler hardware and stripped down functionality. The latency of the chassis equipment comprises the delays within the line cards and the delay between the line cards, thus the latency and jitter are higher.

In practice, the most significant part of latency variability and jitter is caused by head of line blocking phenomenon. The switch latency spread - as measured by the ratio between min, average, and max – can be an order of magnitude or more (e.g. see Figure 11). Features like cut-through and non-blocking architectures cannot eliminate that inherent variability. Note that while the spread ratio is inherent, upgrading a switch speed (e.g. from 1 Gpbs to 10 Gbps links) can reduce the magnitude of latency variability range proportionally.

[16] *Unfortunately, switches and routers do not provide detailed information about bursts, until the hardware buffers are overwhelmed and cause packet loss.*

[17] *Manual switch boards required a human operator to physically connect an input call to an output line with a cable.*

5 Measuring Network Latency

Low latency is a critical advantage for algorithmic trading platforms. Excessive delays in executing trades are a disadvantage against competitors. Random jitter (e.g. Figure 11) lowers the algorithmic predictability and reduces the profit potential.

If network latency and jitter of distributed automatic trading platforms are not monitored adequately, not only does it conceal inconsistent performance and random variability, it also exposes the trading systems to latency arbitrage, hostile manipulations, and malicious attacks. Latency variability and jitter strongly depend on actual network topology and traffic conditions; therefore, it is important to conduct granular network latency measurements continuously in the real environment.

The following subsections overview practical considerations, measurement methodology, and time synchronization aspects. We keep the discussion vendor agnostic.

5.1 Practical Considerations for Measurement Strategy

Specifically, the development of measurement strategy should address the following questions:

1. What to measure – definitions of metrics, temporal granularity, and accuracy target.
2. Where to measure – map of the logical and physical locations of the observation points.
3. How to measure – methodology, dedicated equipment deployment, and reporting mechanism.

New deployment projects can be structured according to standard business procedures: setting measurable objectives; articulating use cases; defining metrics and indicators; designing the setup and deployment model; structuring the data collection and reporting infrastructure; evaluating the results; and driving a continuous improvement process.

Note that lab measurements are easier than monitoring the production environment, but they do not provide full coverage of all the traffic conditions and corner cases in the real production networks.

Practical implementation aspects include the following considerations:

- Dedicated hardware probes versus integrated software feature on the application host: integrating measurement into the application software enables deployment on the same host and does not require additional hardware. However, letting the hosts measure themselves can compromise the accuracy and reliability of measurements due to the *observer effect*. The observer effect in experimental research refers to changes that the act of observing will have on the phenomenon being observed. In information technology, it refers to the challenge of observing a process while it is running without modifying its behavior[18]. External network probes can be used to independently measure network latency and analyze it separately from the software application processing on the host. Independent passive hardware probes eliminate the risk of inadvertently altering application behavior and provide more reliable and accurate information.

[18] *In physics, the observer effect relates to the Heisenberg uncertainty principle, which states that certain interdependent physical properties cannot be measured and known to an arbitrary precision; the more precisely one property is known, the less precisely the other can be known.*

- One-way or round trip measurement: round-trip latency is easier to measure than one-way latency because both the send and receive events occur at the same location. In contrast, one-way measurement involves the more complicated task of *correlating* events time from disparate locations. Trading environments typically need to measure one-way delay because market data is received via one communications path from the exchange and orders are sent via another path from the brokers' trading system.

- Accuracy: the relevant accuracy target is important and should be determined in the context of relevant applications. Higher accuracy provides more information, but accuracy should not be confused with simply adding to a report more digits after the decimal point. For example, if the inherent jitter is in the order of 100 microseconds, it may not be relevant to aim for 0.001 microsecond (nanosecond) accuracy. As another example, if a switch SPAN port jitter introduces random 10-100 microsecond measurement error, aiming for nanosecond accuracy might be irrelevant. It is important to focus on the needed accuracy and results without misinterpreting irrelevant digits that give a false sense of accuracy[19].

- Deployment model and observation points: one way latency and jitter measurement requires at least two observation points. Complex distributed environments, mesh topologies, hop-to-hop analysis, and SLA verification may require several observation points along the packet's path. Distributed observation points provide more granular drill down into performance bottlenecks and excessive variability sources.

- Real time or retroactive analysis: frequent on-the-fly monitoring of a production environment enables timely detection and remediation of critical latency issues. Real-time analysis requires different equipment that does more than capture packets on a disk for retrospective analysis. Exhaustively capturing packets at multiple observation points creates an overhead of multiple copies of the same data. In high speed networks, capturing all the packets to disk does not scale effectively. However, computing resources can be scaled down by applying selective filtering and data reduction.

- Latency average versus variability profile and jitter analysis: average latency is an important figure of merit. But the latency profile over time provides a more information of variations, spread, and trends. The jitter metric provides complementary information about consistency that is critical for optimizing algorithmic trading engines.

- Spikes, bursts, and bandwidth utilization: monitoring latency and jitter coupled with visibility to network spikes, bursts, and bandwidth utilization, supports better root cause analysis and remediation of performance bottlenecks.

- Actual application data or synthetic test data: synthetic traffic generation provides more control over traffic parameters, but it does not provide real time information about actual application behavior.

[19] *The Hawthorne Effect is a form of reactivity whereby subjects improve an aspect of their behavior being experimentally measured simply in response to the fact that they are being studied and not in response to a particular experimental manipulation. It is named after Hawthorne Works that commissioned a study to see if its workers would become more productive in higher or lower levels of light. The workers' productivity seemed to improve when changes were made and slumped afterwards But it was later proved that the study ignored critical variables and led to the wrong conclusions about the impact of light.*

- Exhaustive or filtered: measurements can be applied to all packets and streams, or to a relevant filtered subset, or to a random sample, as appropriate. Applying exhaustive analysis in high speed networks with high traffic volume necessitates substantial computing resources, which can be reduced by analyzing only the relevant subset of the traffic.

The above list is by no way complete, but it highlights important topics to consider when developing a measurement and monitoring strategy. The next subsection is an overview of a solution methodology.

5.2 Methodology Overview

Consumers of the latency information and analytics should have easy access to measurement data with summary tables and graphs of both real-time and historical data.

In general, methodologies for latency measurement include common steps:

1. Extracting time stamped events at the observation points
2. Correlating the events between the observation points
3. Calculating delay metrics like latency and jitter
4. Reporting the measurement data and potential archiving

Relevant events are the times at which specific packets traverse the observation points, where they are time-stamped. Extracting time stamped events from the network can be achieved by deploying dedicated hardware network probes at critical observation points. Note that using software agents on the application hosts and letting them measure themselves can lead to inaccurate results due to the observer effect.

For example, consider measuring one-way network latency and jitter for packets that are flowing through the network between point A and point B, as shown in Figure 14.

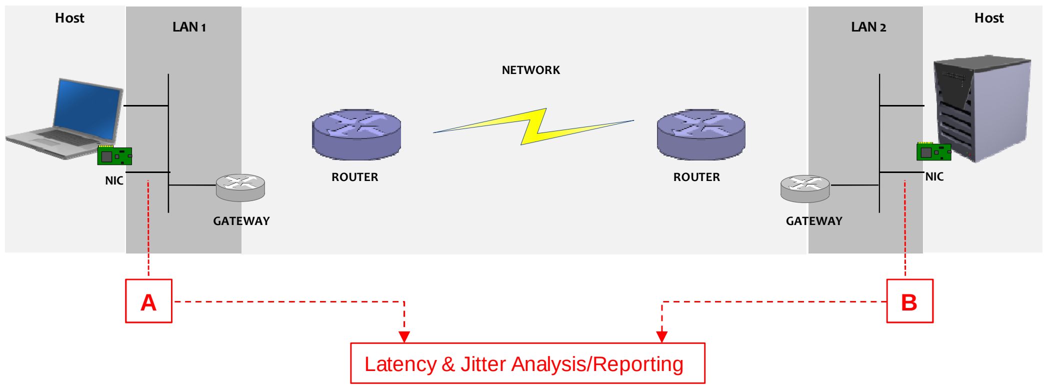

Figure 14: Example of measuring network latency and jitter for traffic between observation points A and B. The events are time-stamped at the observation points and subsequently collected and correlated by a central analysis/reporting engine.

Common approaches for measuring and correlating events include:

- Synthetic test streams with embedded IDs, sequence numbers, tags, and time stamps to facilitate the analysis - allows high degree of control of the test traffic parameters (e.g. packet size mix and rate), but does not provide information about the actual application traffic.

- Application protocols with integrated identifiers and time tags - integration into the application suite is convenient, but does not provide separation between host application delays and network delays. Measurements are subject to the observer effect and may inadvertently impact the application behavior.

- Passive listening to actual application traffic with independent hardware time stamps - allows measurement of application behavior in the real environment and actual context. It relies on header and payload packet inspection to correlate events between the observation points. Distributed hardware probes can support hop-to-hop measurement by deploying multiple observation points along the packets path.

The events that need to be time-stamped and correlated are the times when a specific packet traversed observation points A and B; we denote the corresponding event times as TA_i and TB_i respectively (for packet ID = *i*). The packet latency, L_i, is the difference between the events times ($L_i = TB_i - TA_i$). The jitter for a sequence of packets is the latency variation over the sequence ($L_2 - L_1$, $L_3 - L_2$, …, $L_{i+1} - L_i$, …). For example, a basic data structure for analyzing latency and jitter between two observation points A and B is shown in the following table.

Packet ID	**Time at Point A**	**Time at Point B**	**Latency**	**Jitter**		**Statistics**
1	TA_1	TB_1	$L_1 = TB_1 - TA_1$	---		*Average, Std, Min, Max, etc. over L_1 to L_m*
2	TA_2	TB_2	$L_2 = TB_2 - TA_2$	$L_2 - L_1$		
.	.	.	.	.		
m	TA_m	TB_m	$L_m = TB_m - TA_m$	$L_m - L_{m-1}$		
.	.	.	.	.		⋮
I	TA_i	TB_i	$L_i = TB_i - TA_i$	$L_i - L_{i-1}$		
.	.	.	.	.		
n-m+1	TA_{n-m+1}	TB_{n-m+1}	$L_{n-m+1} = TB_{n-m+1} - TA_{n-m+1}$	$L_{n-m+1} - L_{n-m}$		*Average, Std, Min, Max, etc. over L_{n-m+1} to L_n*
.	.	.	.	.		
n-1	TA_{n-1}	TB_{n-1}	$L_{n-1} = TB_{n-1} - TA_{n-1}$	$L_{n-1} - L_{n-2}$		
N	TA_n	TB_n	$L_n = TB_n - TA_n$	$L_n - L_{n-1}$		

The table shows *n* rows that represent packets. Each row includes correlated time stamps at points A and B, derived latency metrics, and statistics. The 1st column is the packet ID; the 2nd and 3rd are the measured times at which the packet traversed point A and point B respectively; the 4th column is the packet latency that is the difference between the previous two columns; the 5th column is the jitter (latency variation) that is the difference between the latency of subsequent packets; the last column is a place holder for various statistics of latency and jitter that can be calculated for groups of rows or for time intervals. The statistics in the last column might include basic metrics like average, standard deviation, minimum, maximum; and more advanced metrics like moving average, momentum, trend lines, etc.

Hop-to-hop latency analysis can identify specific bottlenecks and sources of variability along the communication path. It requires multiple observation points as shown in Figure 15. Measurement at the intermediate points is performed by hardware probes that passively listen to the traffic and extract time stamped events. The events from all the observation points are collected centrally, correlated, and processed to produce the relevant reports.

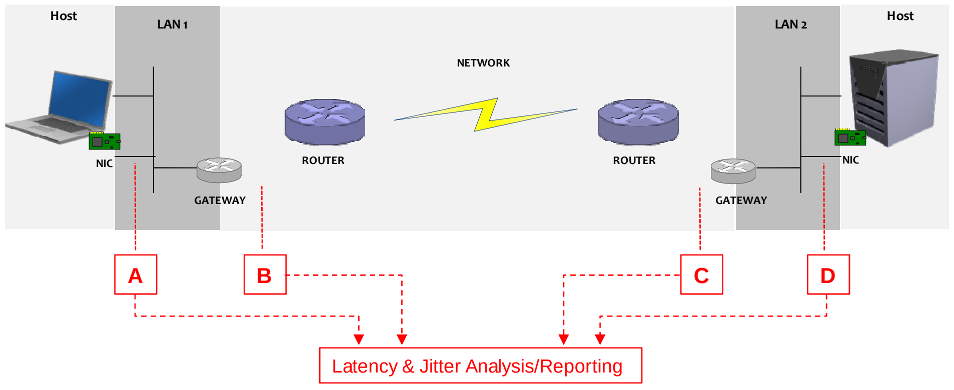

Figure 15: Hop to hop latency analysis relies on deploying multiple observation points along communication path. It enables analysis of the relative contribution of different network segments to the overall latency and jitter. Transparent hardware probe should be deployed without affecting the traffic.

Measuring one way delay and hop-to-hop analysis relies on extracting time stamped events from multiple locations and collecting the data centrally.

For accurate measurement of latency sensitive applications in high speed network environments, it is most appropriate to use transparent hardware probes that time-stamp events on-the-fly and do not create the risk of altering software application behavior.

The next subsection describes common approaches for distributing a common time reference to multiple observation points.

5.3 Time Synchronization Principles

One-way latency analysis is performed by subtracting event times (time stamps) from two or more observation points. For meaningful analysis the time-stamps should have a common time-reference[20] for all of the observation points. This subsection reviews synchronization of clocks in computer networks.

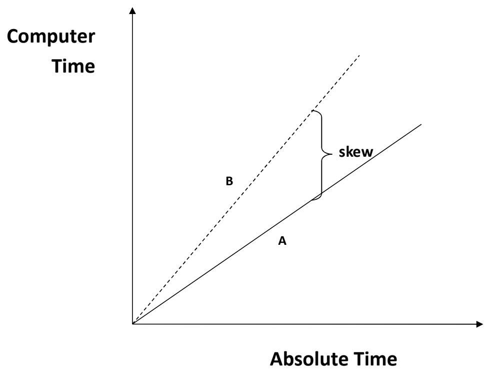

Figure 16: Built-in oscillators in independent computers "tick" at different rates and drift over time, which leads to a clock skew.

[20] *The general concept of simultaneous events, clock synchronization, and time frame-of-reference relates to Einstein's Relativity Theory. Those aspects are not discussed here.*

Clocks built into PCs, servers, and network equipment are low-cost crystal oscillators that drift from each other over time, as shown in Figure 16. Even if two oscillators begin with “perfect” alignment they drift apart, as measured in ppm (parts per million). PC clocks may drift by an order of ~100 ppm, which amounts to one millisecond every 10 seconds or 8.64 seconds a day (more precise clocks may drift by 1 to 10 ppm, which amounts to one millisecond every 100 to 1,000 seconds). Stand-alone atomic clocks are orders of magnitude more precise, but significantly more costly. An atomic clock may drift by less than ~two parts per 10^{14}, which amounts to less than microsecond in a year and a half.

Clock synchronization[21] is a known problem in distributed systems and is by no way unique to latency measurement of financial trading applications. The underlying principle is to use periodic synchronization to limit the *skew* caused by the drift between local clocks. Clock synchronization schemes are susceptible to variability introduced by networks, hosts, and operating systems [22].

The important elements of a synchronization scheme of distributed systems include:

1. Clock source: a common *reference* for synchronizing disparate devices. Clock sources might be GPS or CDMA feeds or high precision clock servers (Atomic Clocks). GPS is commonly used, but it is not always practical to attach a GPS receiver to each synchronized device. Furthermore, GPS signal reception may be limited inside data centers due to signal interference and noise. Often, a GPS receiver is installed next to a window or on the roof and it feeds the clock signal to multiple synchronized devices over a local communication channel.

2. Communication channel for distributing a clock source: allows a clock source to synchronize connected devices. The communication media can be a direct cable connection, standard LAN connectivity, or the Internet. For example, a point to point cable connection provides a more consistent and predictable communication channel than the Internet. Generally, more consistent and predictable communication channel supports higher synchronization accuracy.

3. Time distribution protocol: defines the synchronization messaging sequence. A trivial protocol used for direct cable connection is “pulse-per-second”. More elaborate time synchronization protocols over a network include NTP (Network Time Protocol, RFC1305) and PTP (Precision Time Protocol, IEEE1588). NTP originally targeted autonomous systems that are widely dispersed over the Internet, like servers, work stations, routers, etc. The more recent PTP originally targeted a group of relatively stable components cooperating locally over a local network or few subsegments.

Accuracy of clock synchronization schemes depends on context. NTP over a standard *Internet* connection may provide accuracy of a few milliseconds accuracy, while careful selection of the clock source with fine-tuning of the network connection to that source may improve it to sub millisecond accuracy and better. For distributing a reference clock (e.g. from a GPS receiver on the building roof) over a *local network*, IEEE 1588 is designed to provide accuracy of a few microseconds or better assuming a stable, consistent, and symmetrical network connection (but will not reach this accuracy if the connection is not consistent and symmetrical).

A logical architecture for clock synchronization of multiple devices in a few physical locations can be layered as shown in Figure 17. The common global reference clock can be a GPS signal that is globally available with dedicated receivers.

[21] *The inherent time offset for moving synchronization messages from one location to the other is limited by the speed of light; in vacuum it takes ~3.3 microseconds to move one bit over 1 kilometer and ~3.3 nanoseconds over 1 meter.*

[22] *Such variability can be significant depending on workload conditions.*

Local clock sources in separate physical locations are dedicated clock servers that periodically synchronize to the global clock reference. Synchronizing multiple devices and probes to a local clock server can leverage standard LAN cabling and protocols like PTP or NTP.

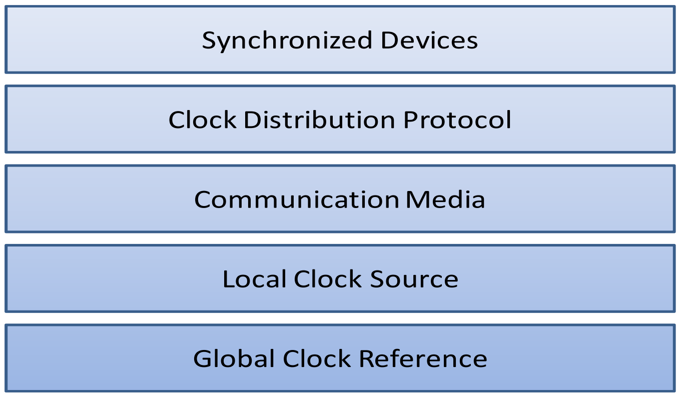

Figure 17: Layered architecture for distributing a common global clock-reference to distributed devices in disparate physical locations.

Different synchronization methods and protocols provide different accuracy levels in different environments. Choosing a particular protocol is not sufficient for determining an accuracy level. Critical parameters for tuning accuracy include clock source accuracy and consistency (symmetry) of synchronization messages transit times.

Clock synchronization is not required for round-trip latency measurements. Furthermore, *one-way jitter* analysis also does not require clock synchronization to a common reference.

6 Common Misconceptions and Errors

The marketing buzz surrounding ultra low latency trading has led to misconceptions and errors. This section highlights a few perceptions and pitfalls of latency measurement solutions.

Practical implementations depend on customers' view of their needs and on the capabilities of available technology in a relevant price range. At the time of writing this paper, the market perceptions about required accuracy seem to vary widely for different applications. For example: low frequency trading and Alpha trading require order of ~100 millisecond accuracy; market making and prime brokerage service require ~10 millisecond accuracy; derivative pricing, direct market access (DMA) services, and high frequency trading (HFT) require ~1 millisecond accuracy or better; and latency arbitrage is sensitive to fractions of a milliseconds, a few microseconds, and less. Measurement solutions, which were sufficient in the past, do not scale to stricter needs of optimizing cutting edge algorithmic trading platforms.

The arms race between vendors marketing departments about accuracy can lead to misunderstanding and unproductive deployments. While accuracy is extremely important, it should not be confused with number of decimal digits in some time-stamp or report. For example, emphasizing a nanosecond accurate (10^{-9} second) time-stamp for one-way analysis can be misleading, if the clock synchronization error between the end points is 10 microseconds (10^{-5} second) which imply four orders of magnitude larger measurement error. Time stamping is only one element of a solution and the actual measurement accuracy depends on all the solution elements.

Claims about measurement accuracy should be evaluated in holistic perspective of all the solution elements. To calibrate the expected measurement accuracy, keep in mind the following facts:

	Description	**Order of Magnitude in nanoseconds**
i	Propagation delay of one bit in 100 meter fiber optics cable	476
ii	Time to serialize packet of 1,250 bytes to network interface memory buffer at 10 Gbps	1,000
iii	Clock period for client interface of transceiver (PHY chip) for 10 gigabit per second Ethernet (XGMII)	6.4
iv	SPAN port random jitter due to queuing (head of line blocking)	10,000 to 100,000
v	A single random-access to DRAM of high-end Server CPU	few 10s
vi	Time-circuit interrupt frequency of operating systems like Linux[23]	order of 1,000,000
vii	Asymmetric network transit times and random jitter impact on clock synchronization	100s to 1,000s
viii	Discrepancy between two commercial GPS receivers[24]	30 to 500 or more

[23] *Linux Kernel 2.6 and above allow time-circuit interrupt frequency upto one kilo-Hertz, which maps to one millisecond intervals. Those interrupts are the beat time (ticks) for all activities in the system. Shorter ticks result in higher resolution timers, which help smoother multimedia playback and I/O multiplexing (e.g. poll and select). However, shorter ticks require the CPU to spend more time in Kernel Mode and less time in User Mode.*

[24] *Some people argue that: "A man with one clock knows what time it is. A man with two clocks is never sure."*

Observe that the total measurement error is cumulative over all the elements of the solution's architecture including: clock synchronization, random jitter of SPAN ports and aggregation devices, variability of operating system, etc. Implementing measurement at a few microseconds accuracy or below in production environment of high speed networks requires attention to all the solution details.

For accurate measurements at millisecond, microseconds, or below, lumping together the host latency and network latency prevents a detailed analysis of dominant latency factors, which might lead to unproductive optimization decisions. Moreover, relying on hosts to measure themselves inherently causes errors due to the observer effect, meaning that the act of observing will make changes on the phenomenon being observed. Effective and accurate measurement methodology should provide detailed analysis of the contributions of host latency and network latency to the overall application performance. For high accuracy, passive transparent hardware probes should be deployed inline with no intermediate switches, span ports, or aggregating devices.

7 Conclusion

Unlike bandwidth, latency depends on context. For example, upgrading link bandwidth by 10 times may or may not improve network latency (e.g. Figure 7 and Figure 8). In real networks, there are multiple latency sources that impact the performance of distributed applications like algorithmic trading. While the speed of light is a fundamental limit for moving bits from one location to another, additional practical factors include interface delays, processing delays, and queuing delays. The impact of various latency sources on distributed applications performance can vary widely across different environments. A key observation is that misinterpretation of the relative contribution of host latency and network latency to application performance can lead to expensive and unproductive optimization efforts.

Inherent latency variability in packet switched networks causes jitter and microbursts (e.g. Figure 11 and Figure 13 respectively). A mathematical model, backed by measurements of real switches, proves that variability is caused by queuing due to head-of-line blocking. Generally speaking, higher speed networks tend to reduce random jitter *magnitude*, if deployed correctly. However the inherent latency variability, as measured by the ratio between the min, and max over a time period, is *not* avoided by the use of cut-through or non-blocking switches. In a complex production environment, assessing the actual latency variability requires a consistent distributed measurement methodology.

Developing an effective methodology for accurate real-time measurements poses several implementation challenges. A critical challenge is circumventing the Observer Effect, where the act of observing changes the behavior of the phenomenon being observed. Making accurate measurements at high speed requires independent hardware probes that do not impact the application performance. These passive transparent probes should be dropped in-line at critical observation points with no intermediaries like span ports, switches, or aggregation devices that introduce random variability and measurement error.

To avoid business disruptions and inferior trading results, continuous real-time measurement should identify trends, detect outliers, and enable quick remediation of excessive latency and jitter. Distributed passive hardware probes with a centralized dashboard deliver reliable information of one-way and hop-to-hop latency and jitter without disrupting the host applications or network performance. Lack of real-time monitoring may expose algorithmic trading platforms to latency arbitrage, hostile traffic manipulations, and malicious attacks.

cPacket Networks offers a range of solutions for accurate latency and jitter measurement, analysis, and reporting based on unique hardware probe technology and easy to use software. Contact us at latency@cpacket.com.