Networking for Quants and Traders

Ben Van Vliet

October 26, 2009

1. Introduction

For automated trading, hardware and networking components are incredibly important. So, at a minimum, a foundation of knowledge in these topics is crucial.

In a network environment that enables trading, the fewer the number of hops between the front-end application and the exchange, the better the performance (at least measured in milliseconds, and maybe in dollars). Achieving better performance requires both analysis and synthesis, that is to say, a holistic approach. Trading is not a hardware or software problem; it is a hardware/software problem. Latency may be added in either the hardware, or the software, or in the interaction between the two. Every component of a trading system must be examined individually on its own merits, but also in light of the whole. As my friend Alex Deitz says, "Everything has an impact, and everything can influence everything else."

Obviously, there are thousands of resources, including books and websites, which describe all manner of network technologies. In this paper I simply describe the hardware and network components that make automated trading work, from microprocessors, to computers, to routers, to networks. I will start with the chip and move outwards.

2. Microprocessors

At the heart of every computer is a microprocessor, an integrated circuit made of silicon with millions of transistors etched on it. A central processing unit (CPU) chip is a microprocessor. CPU performance is measured in millions of instructions per second (MIPS), a number which is a function of clock speed. Modern processors can execute at a rate of about two instructions per clock cycle. A microprocessor executes instructions to perform three basic tasks:

- Perform mathematical operations using its ALU (Arithmetic Logic Unit)
- Move data from one location to another.
- Make logical decisions and jump to new instructions.

This microprocessor has:

- An address bus for sending and receiving addresses to and from memory.
- A data bus for sending and receiving data to and from memory.

These buses and lines connect either to RAM or ROM, generally both.

- An read and write line for telling memory whether it wants to set or get the addressed location
- A clock line for pulsing the clock sequence
- A reset line that resets the program counter to zero (or whatever) and restarts execution

Let's assume that both the address and data buses are 8 bits wide in this example. A latch, or flip-flop, is an electronic circuit which has two stable states and thereby can store one bit of information. Here are the components of this simple microprocessor:

- Registers are simply latches.
- The address latch is just like registers.
- The program counter is a latch with the extra ability to increment by 1 when told to do so, and also to reset to zero when told to do so.
- The ALU could be as simple as an 8-bit adder, or it might be able to add, subtract, multiply and divide 8-bit values.
- The test register is a special latch that can hold values from comparisons performed in the ALU. An ALU can normally compare two numbers and determine if they are equal, if one is greater than the other, etc. The test register can also normally hold a carry bit from the last stage of the adder. It stores these values in flip-flops and then the instruction decoder can use the values to make decisions.
- The tri-state buffers can pass a 1, a 0 or it can essentially disconnect its output (imagine a switch that totally disconnects the output line from the wire that the output is heading toward). A tri-state buffer allows multiple outputs to connect to a wire, but only one of them to actually drive a 1 or a 0 onto the line.
- The instruction register and instruction decoder are responsible for controlling all of the other components.

Coming into the instruction decoder are the bits from the test register and the clock line, as well as the bits from the instruction register.

If we have an address bus 8 bits wide and a data bus 8 bits wide, the microprocessor can address 2^8 or 256 bytes of memory, and it can read or write 8 bits of the memory at a time. Let's assume that this simple microprocessor has 128 bytes of ROM starting at address 0 and 128 bytes of RAM starting at address 128.

A ROM chip is programmed with a permanent collection of pre-set bytes. The address bus tells the ROM chip which byte to get and place on the data bus. When the read line changes state, the ROM chip presents the selected byte onto the data bus.

RAM contains bytes of information, and the microprocessor can read or write to those bytes depending on whether the read or write line is signaled.

On a PC, the ROM is called the BIOS. When the microprocessor boots, it begins executing instructions it finds in the BIOS. The BIOS instructions do things like test the hardware in the machine, and then it goes to the hard disk to fetch the boot sector. This boot sector is another small program, and the BIOS stores it in RAM after reading it off the disk. The microprocessor then begins executing the boot sector's instructions from RAM. The boot sector program will tell the microprocessor to fetch something else from the hard disk into RAM, which the microprocessor then executes, and so on. This is how the microprocessor loads and executes the entire operating system.

The BIOS looks for boot (short for bootstrap) devices. The BIOS initiates the boot sequence from device to device using the bootstrap loader. The bootstrap loader loads the operating system into memory launches it by dividing up memory between the operating system, data and applications. The loader then builds the data structures for communication within and between the various sub-systems and applications. Finally, the loader cedes control of the computer to the operating system.

The collection of instructions a microprocessor can perform is implemented as bit patterns, each one of which has a different meaning when loaded into the instruction register. This collection is called the assembly language of the processor. An assembler can translate the language into their bit patterns and then placed it in memory for the microprocessor to execute.

Every instruction can be broken down as a set of sequenced operations that manipulate the components of the microprocessor in the proper order. Some instructions might take two or three clock cycles. Others might take five or six clock cycles. The number of transistors available has a huge effect on the performance of a processor. A typical instruction in a processor may take 15 clock cycles to execute, and because of the design it may take some 80 cycles just to do one 16-bit multiplication. With more transistors, much more powerful, single-cycle speeds are possible.

More transistors allow for pipelining, where instruction execution overlaps. So, even though it might take five clock cycles to execute each instruction, there can be five instructions in various stages of execution simultaneously.

Microprocessor design has primarily been toward full 32-bit ALUs with fast floating point processors built in and pipelined execution with multiple instruction streams. But, 64-bit processing is almost here. Also, new is the addition of hardware virtual memory support and L1 caching on the processor chip. These new technologies increase transistor count, and therefore MIPS, exceeding one billion instructions per second.

64-bit processors enable enlarged address spaces. A 64-bit RAM address space is essentially infinite for the foreseeable future, 2^{64} bytes is a lot of RAM. With a 64-bit address bus and wide, high-speed data buses on the motherboard, 64-bit machines also offer faster input and output speeds.

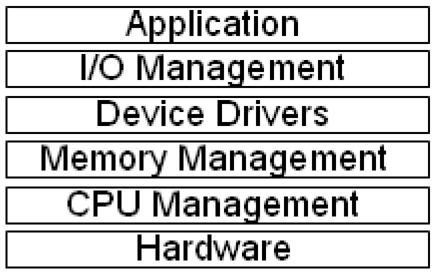
3. Computer

A computer is a machine built around a microprocessor. The main components of a computer are:

- CPU chip.
- Memory, including, as we have seen:
  - RAM for temporarily storing data.
  - ROM permanent storage.
  - BIOS ROM used to establish basic communication when the computer is turned on.
  - Cache, which is the storing of frequently used data in extremely fast RAM that connects directly to the CPU
  - Virtual memory space on a hard disk used to temporarily store data and swap it in and out of RAM as needed
- Motherboard, which is the main circuit board that all other components connect to.
- Hard drive, which is permanent storage used to hold data.
- Operating system, which is the software that allows the user to interface with the computer.
- Integrated Drive Electronics (IDE) Controller, the primary interface for the hard drive, CD and DVD ROMs.
- Peripheral Component Interconnect (PCI) Bus, which is the most common way to connect additional components to the computer, PCI uses a series of slots on the motherboard that PCI cards plug into.
- Small Computer System Interface (SCSI), which is for adding external devices.
- Accelerated Graphics Port (AGP), a very high-speed connection used by the graphics card.
- Sound card for recording and playing audio.
- Graphics card for translating image data for display.

4. Operating System

For our intents and purposes, all computers have operating systems—either Windows, Macintosh, or UNIX/LINUX.



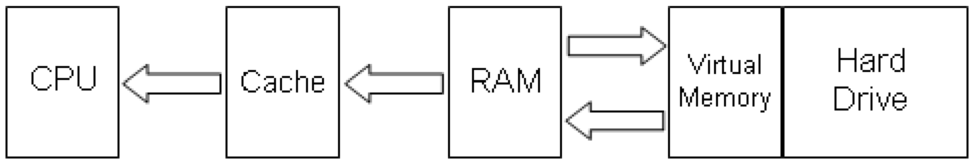
At the simplest level, an operating system does two things:

- Manages the hardware and software resources of the system.
- Provides a stable, consistent way for applications to interface with the hardware.

Managing the hardware and software resources is very important since programs and input methods all compete for time on the central processing unit (CPU), demand memory, and I/O bandwidth. The operating system makes sure that each application gets what it needs, while at the same time playing nice with everyone else.

The OS also provides a consistent application interface, allowing developers to write applications on one computer that will run on other computer with the same OS, even if the physical aspects of the machines differ or hardware upgrades occur. This is because the operating system, not the executable application, manages the hardware and the distribution of time and data storage resources. Keeping an OS flexible enough to run hardware from the thousands of hardware vendors is obviously a challenge.

When an operating system manages the computer's memory, each application must be allocated enough memory to run properly. An application’s memory cannot collide or conflict with the memory space of other processes. Memory must be optimized so that each process can run most effectively. The operating system sets up boundaries in memory for individual applications.



During the boot process, the OS goes to the top of available memory and then backs up far enough to meet its own needs. The OS then goes to the bottom of available RAM and climbs up to allocate memory to the applications that drive the hardware. Once everything is completely loaded, the remaining memory is available for application processes.

When applications are loaded into memory, the OS loads in blocks, and every process will be given a block or several blocks of memory. These blocks ensure that applications won't collide. So, what happens then when all memory is filled?

A processor can only access one memory location at a time, so at any given moment, most stored data is not being used. Also, since disk space is cheap compared to RAM, moving unused data in RAM to the hard disk can greatly expand memory space. This is called virtual memory management. Of course, disk storage is the slowest memory of all. In order of speed, the types of memory are:

- **High-speed cache.** This is a small amount of memory available through the fastest connection. A cache controller predicts which data the CPU will need and copies it from main memory to the high-speed cache. This enhances performance.
- **Main memory.** This refers to the megabytes of RAM in a computer.
- **Secondary memory.** This is virtual memory controlled by the OS.

The OS balances the needs of the various processes with the availability of the different types of memory, moving data in blocks between them. Once loaded, the OS performs six tasks:

- **Processor management**. Breaking down tasks into chunks and prioritizing them before execution.
- **Memory management**. Orchestrating the movement of data between the cache, RAM and virtual memory.
- **Device management**. Providing an interface to devices.
- **Storage management**. Directing data storage on the hard drive.
- **Application management**. Providing a standard interface between software programs.
- **User interface management**. Providing a way for users to interact with the computer.

5. Motherboard

The main task of the motherboard is to hold the CPU chip and allow other devices to connect to it. Every device that runs the computer or boosts its performance is either a part of the motherboard or plugs into a slot or port on it. The layout of components on a motherboard is called the form factor. Some motherboard standards are the:

- CPU socket.
- Chipset, which consists of two bridges that connect the CPU to other devices.
- BIOS chip.
- Real time clock chip, which is a battery-operated chip that maintains the system time.

The slots and ports on a motherboard include:

- Peripheral Component Interconnect (PCI) for video, sound and network cards.
- Accelerated Graphics Port (AGP) for video cards.
- Integrated Drive Electronics (IDE) for hard drive interfaces.
- Universal Serial Bus or FireWire for external peripherals.

- Memory slots.
- Some motherboards also use PCI Express, a newer protocol that acts more like a network than a bus.

The faster the CPU, the faster the computer runs. It used to be that any CPU chip would fit into any motherboard via a standard Pin Grid Array, called Socket 7.   Nowadays, different chip makers use different PGAs, with more pins to handle new features and provide more power.

The chipset, which consists of two bridges, connects the CPU to the rest of the motherboard. One bridge connects to the CPU via the front side bus (FSB) and enables fast access to the memory. The other, slower bridge actually connects to the first bridge and enables the USB ports and hard disk connections. Manufacturers optimize chipsets to work with certain CPUs.

A bus is a circuit that connects one part of the motherboard to another. The speed of the bus, which refers to the front side bus, which connects the first bridge to the CPU, is measured in megahertz (MHz) and shows to how much data can move across the bus at any given time. The faster the bus, the faster the computer, though a fast bus cannot cover for a slow CPU chip or chipset.

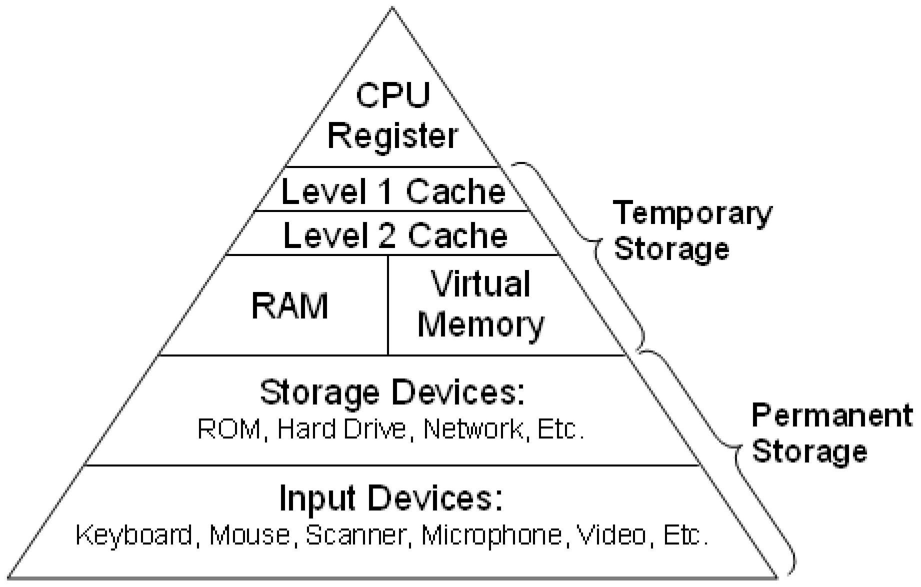
Since the CPU reaches the memory controller though the bridge, FSB speed significantly affects performance. There are other busses, too:

- Back side bus connects the CPU with the level 2 (L2) cache.
- Memory bus connects the first bridge to memory.
- IDE or ATA bus connects the second bridge to the hard disk drives.
- AGP bus connects the video card to the memory and the CPU.
- The PCI bus connects PCI slots to the second bridge. PCI Express is likely to replace both PCI and AGP busses.

Te speed of the CPU determines how fast a computer thinks. The speed of the chipset and busses determines how quickly the computer can communicate between its parts. The speed of the memory bus determines how fast the computer can access instructions and data. A fast CPU with slow memory is going to be slow. The more RAM the better.

6. Memory

The CPU accesses memory hierarchically, though wherever data comes from, it must go into RAM first. From there, the CPU stores some data in a cache, and maintains certain special instructions in the register.



On start up, the computer loads data from ROM and, as part of the power-on self-test (POST), the memory controller checks all of the memory addresses with a read/write operation to ensure that the memory chips are free of errors. When a software application is opened, the CPU loads it into RAM, though to minimize memory usage, many software applications load only the essential parts of the program initially and then load other pieces as needed. After loading, any files that are opened for use in that application are loaded into RAM.

During execution, the CPU requests data from RAM, processes it and writes new data back to RAM in a continuous cycle, millions of times every second. Fast CPUs need fast access to data in order to maximize performance. A fast CPU may end up waiting for data. The problem is that memory that can keep up with a 1-gigahertz CPU is expensive. Most computer designs solve the cost problem by tiering memory, where a small amount of expensive memory is backed up with larger quantities of cheaper memory, say the hard drive.

The bit size describes the number of bytes the CPU can access from RAM at any one time. For example, a 16-bit CPU processes 2 bytes at a time, a 64-bit 8 bytes. A CPU’s Megahertz (MHz) describes its processing speed, or clock cycle, in millions per second. A 32-bit 800-MHz Pentium III can process 4 bytes at one time at a rate of 800 million times each second. Now, a computer's RAM is not fast enough to keep up with the CPU, which is why the computer uses a cache. Nevertheless, faster RAM is better. Most chips have a cycle rate of 50 to 70 nanoseconds.

RAM speed is controlled by bus width and bus speed. Bus width describes the number of bits that can be sent to the CPU at one time. Bus speed is the number of times a set of bits can be sent each second. A bus cycle occurs when a set of bits travels from RAM to the CPU.

A 66 MHz 32-bit bus is theoretically capable of sending 4 bytes of data to the CPU 66 million times per second. Changing the bus width from 16 bits to 32 bits and the speed from say 66 MHz to 100 MHz will pass three times as much data to the CPU every second.

In reality, latency prevents RAM from operating at optimum speed. Latency is the number of clock cycles needed to read a bit. RAM with a 100 MHz speed can send a bit in 0.00000001 seconds. However, it may take 0.00000005 seconds to start the read process for the first bit. To compensate for this added latency, CPUs use a burst mode.

Burst mode makes the assumption that data requested by the CPU will be stored sequentially in memory. The memory controller anticipates that the data the CPU is working with will continue to come from this same sequence of memory addresses, so it reads several consecutive bits at a time. So, only the first bit is will encounter the full effect of latency.

The burst mode rate is normally expressed with four numbers. The first number is the number of clock cycles needed to begin a read operation. The second, third and fourth numbers are the number of cycles needed to read each consecutive bit. A burst mode rate of 5-1-1-1 shows that it takes five cycles to read the first bit and one cycle for each ensuing bit.

A technique called pipelining is another means of minimizing the latency. Pipelining organizes data retrieval into an assembly line process. The memory controller simultaneously reads one or more pieces of data from memory, sends the current data to the CPU and writes one or more data to memory. Used together, burst mode and pipelining can dramatically reduce latency.

The speed and width of the memory's bus should match the system's bus. There is no advantage to using 100 MHz memory designed in a 66-MHz system, nor a 32-bit memory on a 16-bit bus. Even with a fast and fat bus, it still takes longer for data to get from RAM to the CPU than it takes for the CPU to process the data. When the microprocessor accesses the main memory (RAM), it does it in about 60 nanoseconds (60 billionths of a second), which is much slower than the typical microprocessor. Microprocessors can have cycle times as short as 2 nanoseconds, so to a microprocessor 60 nanoseconds seems like an eternity. This is where caches come in.

Caches alleviate the memory bottleneck by making the data used most often by the CPU instantly available. The level 1 cache is a small amount of memory, usually 2 to 64 kilobytes, built right into the CPU. Level 1 cache enables memory access at full microprocessor speed (10 nanoseconds and 4 kilobytes to 16 kilobytes in size). The level 2 cache, usually 256 kilobytes to 2 megabytes, resides on a memory card located on the motherboard near the CPU and has a direct connection to the CPU. (Many high performance CPUs now have the level 2 cache actually built into the CPU chip itself.) Level 2 cache enables memory access at speeds around 20 to 30 nanoseconds and 128 kilobytes to 512 kilobytes in size. Data needed by the CPU is accessed via cache hits approximately 95 percent of the time, greatly reducing latency added by going to main memory. The size and location of the level 2 cache is a major determining factor in the performance of a CPU.

Caches use extremely fast static RAM (SRAM). SRAM consists of an external gate array, known as a bistable multivibrator, where elements flip-flop between two states to maintain data. So, SRAM does not need to be continually refreshed like DRAM. SRAM can be asynchronous or synchronous. Synchronous SRAM matches the speed of the CPU, while asynchronous does not, and matching the CPU's clock speed is beneficial. Always look for synchronized SRAM although it is expensive.   The idea behind caching is to use a small amount of expensive memory to speed up large amounts of slower, less-expensive RAM.

CPU registers are memory locations built right into the CPU that contain the arithmetic and logic unit (ALU). As an integral part of the CPU itself, registers are controlled directly by the compiler that sends data to the CPU.

7. Bus

A bus transfers data between components inside a computer or between computers. Unlike a point-to-point connection, a bus can connect several components over the same set of wires, where each bus defines its own connectors between devices, cards or cables. An internal, or local, bus connects the CPU and memory. An external bus connects external components to the motherboard.

Buses can use parallel and bit-serial connections or can be connected by switched hubs, like in USB. Parallel buses carry data in parallel across multiple wires, or serial buses, which carry data in bit-serial form. As rates of financial data increase, timing skew, power consumption, electromagnetic interference and crosstalk problems across parallel buses become harder to work around.

Network connections, such as Ethernet, are not considered buses, although the difference is really just a matter of semantics. InfiniBand and HyperTransport technologies are blurring the boundaries between networks and buses.

8. Message Queue

A message queue is a software component used for communication between processes or threads. With asynchronous message queuing the sender (or publisher or producer) and receiver (or subscriber or consumer) of the message do not need to interact directly. The sender places messages onto the queue, which stores them until the receiver grabs the message off the queue. Many message queues function within an operating system. In synchronous queuing, the receiver makes a connection to the sender, makes a request for data and waits for a reply.

The nature of the queuing system depends upon the nature of the problem. For example, the sender may only need to notify a consumer that new data has arrived, but the sender does not need to wait for a confirmation response from the consumer. Other applications may publish market data to any number of consumers.

In many financial systems, having an asynchronous queuing subsystem will boost performance.

9. Network

A computer network, such as a LAN or a WAN, is group of computers that are interconnected and can exchange data. A network communicates over twisted-pair copper wire cable, coaxial cable, optical fiber, and various wireless technologies.

In a network, a master scheduler controls the data traffic. When data is transferred, the requesting computer sends a message to the scheduler, which queues the request.   The scheduler prioritizes requests, notifies the appropriate receiver as soon as the bus is available, and transfers the data.   The benefit of a bus is that any computer can be accessed directly and messages can be sent quickly, although the bus needs a scheduler to prioritize the traffic.

10. Nodes and Hubs

A node is a point in a network at which lines intersect or branch, a device attached to a network, or a terminal or any other point in a network where messages can be created, received, or transmitted. On a TCP/IP network, a node is any device with an IP address. A computer node is called a host.

In the most basic type of network found today, nodes are simply connected together using hubs. A hub is a device that connects multiple twisted pair or fiber optic Ethernet devices, making them act as a single segment. As we will see, hubs work at the physical layer of the OSI model. As a network grows, potential problems arise with this configuration:

- **Scalability**. In a hub network, limited shared bandwidth makes it difficult to grow without sacrificing performance.
- **Latency**. Since each node in a hub-based network has to wait for an opportunity to transmit in order to avoid collisions, the amount of time that it takes a packet to get to its destination.can increase rapidly as more nodes are added.

- **Network failure**. In a typical network, one device can cause problems for other devices due to incorrect speed settings.
- **Collisions.** If two nodes send out packets at the same time, a collision occurs and the packets may be lost.

11. Server

A server is a multi-user computer that executes specific, long-running applications. These applications describe the server itself, with names such as web servers, email servers, database servers, and gateway servers. These server applications may be divided among several servers over a large geographical area if need be. The server performs its service application for connected clients by sending responses.

While any computer can act as a server, server computers often include faster processos and memory, more RAM, larger hard drives, higher reliability, redundant power supplies, redundant hard drives, compact size and shape, modular design, rack or cabinet mount-ability, serial console redirection, etc.

12. Packet

Networks that transfer data around in packets are called packet switched networks. On the Internet, the network breaks messages into parts called packets, usually containing 1000 to 1500 bytes of information. Each packet carries the information that will help it get to its proper destination, including the sender's IP address, the intended receiver's IP address, and packets the message has been broken into and the number of the particular packet. Packets carry data in the protocols that the Internet uses—Transmission Control Protocol/Internet Protocol (TCP/IP). Each packet is sent to its destination via the best available route, which may be different for each packet in the message.

The network balances the load across various pieces of equipment on a millisecond-by-millisecond basis. If there is a problem at one node, packets can be routed around them, which ensures the delivery of the entire message. Most packets consist of three parts:

- **Header**. The header may include:
    - Length of packet.
    - Synchronization bits that help the packet match up to the network.
    - Packet number in a sequence of packets.
    - Protocol.
    - Destination IP address.
    - Origination IP address.

- **Payload.** This is the actual data content. The payload may be padded with blank bytes to make it a fixed size.
- **Trailer.** The trailer, or footer, contains a couple of bits that signal the end of the packet. It may also have some type of error checking. The trailer may contain a Cyclic Redundancy Check (CRC). CRC takes the sum of all the 1s in the payload and stores it in the trailer. The receiving device adds up the 1s in the payload and compares the result value in the trailer. If the values match, the packet is good. If not, the receiving device knows to send a request to resend the packet.

Routers look at the destination address in the header and compare it to their lookup table to find out where to send the packet. Once the packet arrives at its destination, the receiving computer strips the header and trailer off each packet and reassembles entire message based on the numbered sequence of the packets.

13. TCP/IP

The Internet Protocol (IP) is a protocol for communicating data across a packet-switched network. The Transmission Control Protocol (TCP) is a core of the Internet protocol suite. TCP provides reliable, in-order delivery of packets. TCP controls the size and rate at which messages are exchanged between the server and client.

TCP is connection-oriented, meaning that it requires a handshake to set up a connection. TCP manages message acknowledgment, retransmission and timeout, where many attempts to reliably deliver the message may be made. If a packet gets lost, the receiving device will request the lost packet to be resent, so with TCP, there will be no missing data. In the case of multiple failures or timeouts, the connection is dropped. With TCP also, if two messages are sent along a connection, the first message will reach the receiving application first. Out of order packets are held by the TCP layer and rearranged later for delivery to the application to the application. As a result, before any real data is sent, TCP is heavy, needing three packets just to set up a socket connection.

14. UDP

User Datagram Protocol (UDP) is also a core of the Internet protocol suite. Using UDP, applications on networked devices can send and receive short messages, called datagrams, as a stream. UDP does not guarantee delivery or ordering like TCP, and so is lightweight, but unreliable. Datagrams may arrive out of order, be duplicated, or disappear without notice. UDP avoids the overhead of checking delivery of every packet, which makes it faster and more efficient, at the expense of guaranteed delivery. Many real-time applications use UDP when dropped packets are preferable to delayed packets. UDP is also stateless, so it is also useful for servers that deliver small amounts of data to a huge numbers of clients. Unlike TCP, UDP is compatible with packet broadcast and multicasting to a group of subscribers.

UDP is a simple connectionless protocol, where there is no effort made to setup a dedicated end-to-end connection or handshake. Communication is effected by sending data in one direction, from source to destination, with no check to confirm that the destination is still there.

15. Frame Relay

Frame relay is an efficient data transmission technology for sending digital information quickly and cheaply. It is a message forwarding relay-race-style system in which packets, or frames, are passed from one or many points to one or many destinations via a series of intermediate nodes.

Network providers commonly implement frame relay for data as an encapsulation technique, used between LANs over a WAN. Each end-user gets a leased line to a frame-relay node. The frame relay network handles the transmission over a frequently-changing path transparent to all end-users.

16. OSI Model

The Open Systems Interconnection Basic Reference Model (OSI Model) is an abstract description of communications and network design. The core of this standard is the OSI Model, a set of seven layers that define the different stages that data must go through to travel from one device to another over a network. Each layer of the model is a collection of related functions that provides services to the layer above it and receives service from the layer below it. The OSI model does not include user interfaces. The seven layers are divided into the application set (layers 5-7) and the transport set (layers 1-4):

Layer 7: Application. This is the layer interacts with the operating system or software application whenever the user attempts to transfer files, read messages or perform other network related activities.

Layer 6: Presentation. This layer takes the data provided by layer 7 and converts it into a standard format that other layers can understand.

Layer 5: Session. This layer establishes, maintains and ends communication with the receiving device.

Layer 4: Transport. This layer maintains flow control of data and provides for error checking and recovery of data. Flow control checks integrates each application's data into a single stream for the network layer.

Layer 3: Network. This layer handles the logical protocols, routing and addressing before sending the data to the receiving device.

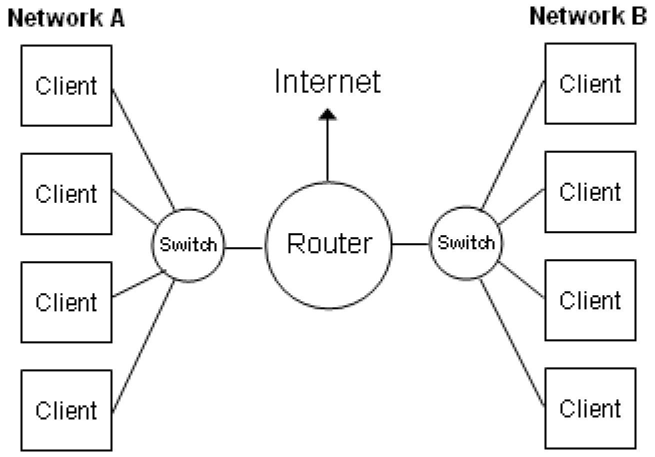
Layer 2: Data. This layer assigns the appropriate physical protocol to the data and defines the type of network and the packet sequencing.

Layer 1: Physical. This is the hardware layer, and it defines the physical characteristics of the network such as connections, voltage levels and timing.

In practice, the OSI Model is just a guideline. Actual protocol stacks usually combine one or more of the OSI layers into a single layer. A protocol stack is a group of protocols that all work together to allow the technology to perform some function. The TCP/IP protocol stack uses four layers: Network Interface (OSI 1 and 2), Internet (OSI 3), Transport (OSI 4), Application (OSI 5-7).

17. Router

Routers are specialized devices (at OSI Layer 3) that send messages to their destinations along thousands of pathways. Much of the work to get a message from one computer to another is done by routers, because they're the crucial devices that let messages flow between networks, rather than within networks.



A router links networks and connects networks to the Internet. The router is the only device that sees every packet of every message sent by every computer on both networks. A router uses a configuration table to decide where packets should go. A configuration table includes:

- Information on which connections lead to which groups of IP addresses.
- Priorities for connections.
- Rules for handling data traffic.

A router ensures that packets do not go where they are not intended to go. A router ensures that data makes it to the intended destination. Since the Internet is one gigantic network, which itself is comprised of thousands of smaller networks, routers are an absolute necessity.

Internet data travels over a packet switching network. In this system, each packet is sent via the best available route. The routers can reconfigure the paths that packets take, because routers look at the data surrounding the payload, and are able to tell each other about delays in pieces of the network.

Of all the devices that make up a network, a router is the only one of these devices that examines each packet as it arrives and makes a decision as to exactly where it should go. To make these decisions, routers must first know addresses and network structure.

Routers that are part of the Internet move millions of packets every second. But, simply moving packets along isn't enough. Routers find the best possible route. Depending on the time of the trading day and day of the week, some parts of the huge public packet-switched network may be busier than others. The routers that make up this system will communicate with one another so that traffic can be sent through less congested routes.

18. Switch

A switch is a gate device (at OSI Layer 2) used to connect and disconnect a circuit. Switches are another fundamental part of networks because they allow different nodes of a network to communicate directly with one another in a smooth and efficient manner.

A hub is like an intersection, where all packets have to stop. A switch, on the other hand, is like a cloverleaf on an expressway, where each packet can take an exit ramp at full speed to get to its destination. A difference between a hub and a switch is that all the nodes connected to a hub share the bandwidth between themselves, while a device connected to a switch has the full bandwidth all to itself.

19. Ethernet

Ethernet is the most popular and most widely deployed network technology in the world. Ethernet, standardized as IEEE 802.3, is a family of frame-based networking technologies for LANs. It defines a number of wiring and signaling standards for the physical layer (OSI Layer 1), through a common addressing format. Twisted pair versions of Ethernet for connecting to the network combined with fiber optic versions is the most widespread Ethernet LAN technology.

At most, Ethernet devices can have a few hundred meters of cable between them, making it impractical for geographically dispersed networks. Modern advancements have allowed Ethernet networks to cover tens of miles.

Ethernet nodes communicate in short messages called frames, which are variably sized chunks of data. The Ethernet protocol specifies a set of rules for constructing frames, including explicit minimum and maximum frame lengths and required bits of data such as the destination address and a source address.

InfiniBand is a switched fabric communications link for high-performance computing. The InfiniBand specification defines a connection between processors and high performance I/O nodes. It replaces Ethernet and features quality of service and failover guarantees. Furthermore, it is scalable.

20. Leased/Dedicated Line

A leased line is a dedicated telecommunications line connecting two locations. In the UK, a leased line is known as a private circuit or data line. High speed leased lines are usually presented using T1 circuit with 1 to 24 56k or 64k timeslots. For many purposes, leased lines are gradually being replaced by DSL and Ethernet. Leased lines may not be a single cable, but will provide the guarantee of consistant bandwidth and nearly constant latency, which are unattainable over the Internet.

21. Conclusion

For automated trading, hardware and networking components are incredibly important. The correct approach, when looking at all things technology, is cost/benefit analysis.   That is, how much of a latency decrease can be gotten from integrating a new technology; or how much more business value can we cram into the same latency budget utilizing a new technology? Typically, from a performance perspective, fewer hops between the Automated Trading Application and the exchange are better. So co-location, choice of networking fabrics (from the exchange and among client machines), available protocols and API's make a huge difference in how you achieve high performance.

Colocation means placing multiple servers in a single location, and installing or running applications in a data center in common with the exchange. Virtualization is an example of colocation, where a host server provides a virtual platform for running one or more instances of software. When it comes to networking fabrics you have three choices—1Gb Ethernet, 10Gb Ethernet, and Infiniband. Infiniband is fast, but Ethernet will likely catch up. LANs are faster than WANs.

Achieving performance requires both analysis and synthesis in a holistic approach. Trading is not a hardware or software problem; it is a hardware/software problem. Latency could be in the code; it could be the network stack; it could be the protocol; it could be the switch; it could be your connectivity; it could be the exchange.   Every component has to be examined individually on its own merits, but must be examined in light of the whole; there are interaction effects between the hardware and the software. Everything has an impact, and everything can influence everything else.

So, performance metrics—resolution of metrics, the ability to measure, measuring in the right places, measuring the right things—determine if a trading system is achieving anything like what you set out to achieve.