Bus Systems and Sensors Paper

	Professor's comments:	
Date: 10.05.2020		Author: Ching Wei Hsieh
Professor: Pawel Buczek		

Table of Contents

Introduction

In this report, we are going to discuss a few topics, including some detailed explanation regarding different models and how they interact and work with each other. We will have a deep dive especially on the topics of: Ethernets, WiFi, IP, TCP, UDP, USB, CAN, SATA and Firewire.This paper consists of many citations, the referenced website links and resources would be recorded in the last few pages.

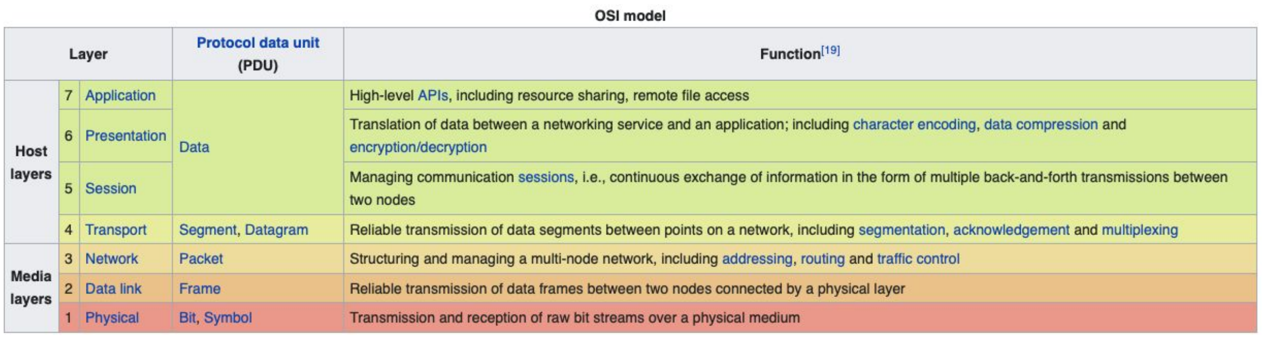
OSI Model/layers vs TCP/IP

First, let's have an introduction about different protocols including OSI and TCP/IP that would help us have an understanding of the concepts that will be mentioned later.

OSI:

The Open Systems Interconnection model is a conceptual model that gives us a systematic overview of the communication functions of a telecommunication system and was also standardized by the International Organization for Standardization so it is a recognized and common Model to reference from.

From the chart below, we could have a brief overview of how each layer is distributed with different functions and how they connect with each other.



OSI model

Layer			Protocol data unit (PDU)	Function[19]
Host layers	7	Application	Data	High-level APIs, including resource sharing, remote file access
	6	Presentation		Translation of data between a networking service and an application; including character encoding, data compression and encryption/decryption
	5	Session		Managing communication sessions, i.e., continuous exchange of information in the form of multiple back-and-forth transmissions between two nodes
	4	Transport	Segment, Datagram	Reliable transmission of data segments between points on a network, including segmentation, acknowledgement and multiplexing
Media layers	3	Network	Packet	Structuring and managing a multi-node network, including addressing, routing and traffic control
	2	Data link	Frame	Reliable transmission of data frames between two nodes connected by a physical layer
	1	Physical	Bit, Symbol	Transmission and reception of raw bit streams over a physical medium

(Online image from Wikipedia)

TCP/IP:

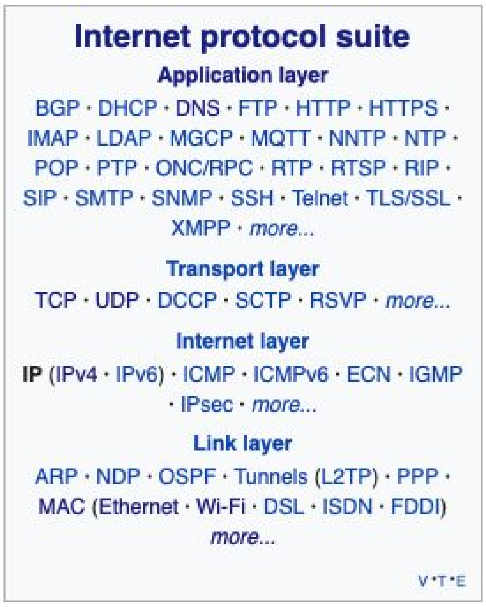
The Internet protocol suite is the conceptual model and set of communications protocols used in the Internet and similar computer networks. It is commonly known as TCP/IP because it consists of the most fundamental and important protocols which are the Transmission Control Protocol (TCP) and the Internet Protocol(IP).

So after having a brief idea on which layers there are and their definitions, we would be able to categorize that Ethernet belongs in the Physical and Data link layer, while WiFi only belongs to the Data link layer, we will have a comparison and discussion about that later. Lastly, TCP/UDP belongs in the Transport layer . It is an interesting topic to learn about how each layer interacts with each other and how they pass data from layer to layer. For more detailed description regarding the different layers and how to identify them, you could visit the Wikipedia website that was referenced above.

These two protocols are basically introducing a similar concept and division of layers but slightly differs on the number of layers. For more detailed information on the details of these two models, we could reference to *Tanenbau, Computer Networks* page 49.[1]

Internet Protocol

First, we will start understanding the protocol from the bottom layers to the top. In the most bottom layer we have the Link layer, which consists of the MAC layer including the Etherent and WIFI. Later on, we have the Internet Layer, Network Layer, Application Layer. We will have a brief introduction about all the layers in this paper.



Internet protocol suite

Application layer

BGP · DHCP · DNS · FTP · HTTP · HTTPS · IMAP · LDAP · MGCP · MQTT · NNTP · NTP · POP · PTP · ONC/RPC · RTP · RTSP · RIP · SIP · SMTP · SNMP · SSH · Telnet · TLS/SSL · XMPP · *more...*

Transport layer

TCP · UDP · DCCP · SCTP · RSVP · *more...*

Internet layer

IP (IPv4 · IPv6) · ICMP · ICMPv6 · ECN · IGMP · IPsec · *more...*

Link layer

ARP · NDP · OSPF · Tunnels (L2TP) · PPP · MAC (Ethernet · Wi-Fi · DSL · ISDN · FDDI) *more...*

V·T·E

(Online image from Wikipedia)

Ethernet

Ethernet[1] is a way of connecting computers together in a local area network or LAN. It has been the most widely used method of linking computers together in LANs since the 1990s. The basic idea of its design is that multiple computers have access to it and can send data at any time. [2]

There may be a confusion whether Ethernet = Internet?
It is important to note that Ethernet is a LAN[2] (Local Area Network), therefore having an Ethernet connection does not necessarily equal grant you access to the public internet we use nowadays to surf on websites. However, it is possible to use a ethernet to connect the LAN networks to a WAN[3] (Wide Area Network) in order to have access to the public internet.

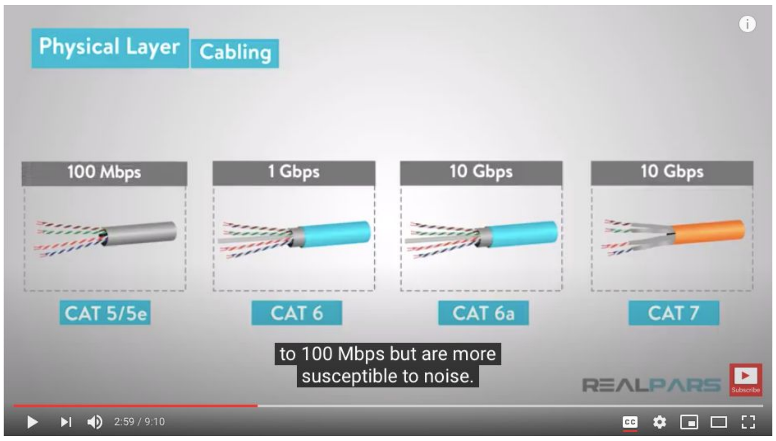
Ethernet consists of two layers - Physical Layer and Data Link layer. First we will have a discussion about the Physical Layer.

[1] https://en.wikipedia.org/wiki/Ethernet
[2] https://simple.wikipedia.org/wiki/Local_area_network
[3] https://en.wikipedia.org/wiki/Wide_area_network

Physical Layer

We could divide the physical layer of ethernet into two components - cabling and devices.

Cabling:
Ethernet is a wired system that started with using coaxial cable and has later evolved in twisted pair cables and fiber optic cable. The most commonly used cable nowadays is the twisted pair cable that varies from different speed or reliability such as CAT 5/5e, CAT 6, CAT 6a, CAT 7.[3]



(Online image from Youtube)

The twisted pair cables are connected with pin connectors in order to transmit or receive data, either with the half-duplex mode or full-duplex mode[4]. The half-duplex mode only transmits data in one direction at a time, but is able to transmit/receive data in both ways just unable to transmit/receive at the same time. While full-duplex (consists of two wires) mode allows data to transmit in both directions at the same time.

Fiber optic cable uses glass or plastic optic fiber as a conduit of light pulses to transmit data, it has allowed ethernet to travel farther distances at higher speeds.[4] In order to optimize higher speeds using fiber optic cables, we need to have an ethernet to fiber converter[5] to connect the different cables.

Devices:
Ethernet devices consist of computers, printers or any device that either has an internal Network Interface Card (NIC), or an external one that is USB or PCI based.

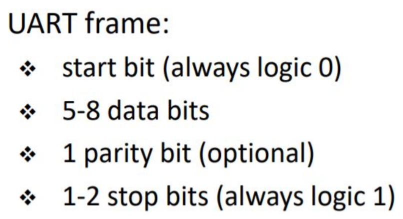
Switches and Routers act as the director of the network and connect multiple computers or even networks together to enable communication between all the different devices. Gateway and Bridges are used to connect multiple Ethernet networks together and allow communication across them. Gateways connect two dissimilar networks together while a bridge connects two similar networks together so that you only see one network.

Data Link Layer

[4] https://en.wikipedia.org/wiki/Duplex_(telecommunications)
[5] https://en.wikipedia.org/wiki/Fiber_media_converter

This layer is often considered as the most complicated layer as we could divide this layer into two sublayers - Media Access Control (MAC)[6] and Logical Link Control (LLC)[7]. When this layer receives data from the physical layer it checks for transmission errors and then packages the bits into data frames[8]. From there, this layer manages the physical addressing methods for the MAC or LLC layers. [5]

In general, a data frame (packet) contains some of the components in the following: Addresses, ERC, length, payload, priority. According to each different system we are using there are different data frame formats to them, for example we have learnt the UART frame format in our Bus Systems Lecture 2. We will have a more detailed discussion with variations of protocols in the next few sections.


UART frame:
- start bit (always logic 0)
- 5-8 data bits
- 1 parity bit (optional)
- 1-2 stop bits (always logic 1)

(Image from Bus Systems Lecture PDF)

The LLC establishes paths for data on the Ethernet to transmit between devices. The MAC uses hardware addresses that are assigned to NIC to identify a specific computer or device to show the source and destination of data transmissions.

Ethernet transmits data packets in this data link layer by using an algorithm called Carrier Sense Multiple Access with Collision Detection (CSMA/CD)[9]. CSMA/CD is used to reduce data collisions and increase successful data transmissions.[6] The algorithm first sends a bit of data to check if there is traffic on the network, if it does not find any it will send out the first bit of information to see if a collision will occur. If this first bit is successful, then it will send out other bits while still testing for collisions. If a collision occurs, the algorithm calculates a waiting time and then starts the process all over again until the full transmission is complete.[7]

The task for checking collisions is mainly done in the MAC layer, while the LLC acts as an interface between the MAC and the network layer. As we could see these two sublayers work together to be able to bring us to another level of the Protocol- the network layer.

WiFi - Data Link Layer

WiFi also belongs in the MAC layer of the Data Link Layer of the Protocol.

WiFi uses radio waves, just like cell phones, televisions and radios to transmit data. Wi-Fi most commonly uses the 2.4 gigahertz (120 mm) UHF and 5 gigahertz (60 mm) SHF ISM radio bands;

[6] https://en.wikipedia.org/wiki/Medium_access_control
[7] https://en.wikipedia.org/wiki/Logical_link_control
[8] https://en.wikipedia.org/wiki/Ethernet_frame
[9] https://en.wikipedia.org/wiki/Carrier-sense_multiple_access_with_collision_detection

these bands are subdivided into multiple channels.[8] Channels can be shared between networks but only one transmitter can locally transmit on a channel at any moment in time.

The steps for transmitting data by WiFi would be

1. A computer's wireless adapter translates data into a radio signal and transmits it using an antenna.
2. A wireless router receives the signal and decdoes it. The router sends the information to the Internet using a physical, wired Ethernet connection.

The process also works in reverse, with the router receiving information from the Internet, translating it into a radio signal and sending it to the computer's wireless adapter.[9]

We will have a brief comparison on the differences between WiFi and Ethernet in the following chart.

WiFi	Ethernet
transmits data through wireless signals	transmits data through cables
Slower(up to 9.6Gbps)	faster (up to 10GGbps)
Less reliability and security	Greater reliability and security
More convenient	Less convenient
Higher latency	Lower latency

Graph created by: ChingWei Hsieh

Ethernet Frame

Ethernet frames have been developed in many different versions throughout history as the later versions get more and more complicated, we are just going to discuss the version that was first introduced, Ethernet II as it is more simple. This section would be about how to transmit our bits correctly in order to match the correct format.

A data packet on the wire and the frame as its payload consist of binary data. Ethernet transmits data with the most-significant octet (byte) first; within each octet, however, the least-significant bit is transmitted first. [10]

*This part might be a bit confusing, but here we want to clarify the differences between the Ethernet protocol physical layer and the Ethernet protocol data link layer. Note that in the previous section we have discussed the Ethernet physical layer and Ethernet data link layer but it was more of a general introduction regarding Ethernet. In contrast, here we are in a depth view from the protocol (or frame format) perspective of the Ethernet.

Not only do the twisted pair cables and fiber optics that we mentioned above belong to the physical layer in the OSI model but the preamble and SFD that we will learn about later also belongs there too. Similarly, not only does the MAC and LLC we mentioned a while ago belong to the Data Link Layer in the OSI model but also more detailed components such as Header, Payload, FCS also belong there. So in the later section, we are going to have a deep dive in the compositions of the Ethernet Protocol.

Ethernet II

Preamble	SFD	Destination Address	Source Address	Type	Data Payload	FCS
7 Bytes	1 Byte	6 Bytes	6 Bytes	2 Bytes	46 - 1500 Bytes	4 Bytes

Graph created by: ChingWei Hsieh

Physical Layer[10]

Unlike higher-level protocols, the lower layers must know the details of the underlying network including its packet structure, addressing, etc. to correctly format the data being transmitted to comply with the network constraints. We will be able to see different Bit/Byte allocations for different frames and protocol headers in the following sections.

In the Ethernet Packet, only Preamble and SFD belong to the physical layer, while the rest belong to the data link layer.

An Ethernet packet starts with a seven-octet preamble and one-octet start frame delimiter (SFD). The preamble consists of a 56-bit (seven-byte) pattern of alternating 1 and 0 bits, allowing devices on the network to easily synchronize their receiver clocks, providing bit-level synchronization.

The SFD is the eight-bit (one-byte) value that marks the end of the preamble, which is the first field of an Ethernet packet, and indicates the beginning of the Ethernet frame. The SFD signals a break in the bit pattern of the preamble and signals the start of the actual frame. [10]

Interpacket gap is idle time between packets. After a packet has been sent, transmitters are required to transmit a minimum of 96 bits (12 octets) of idle line state before transmitting the next packet.

Data Link Layer

The data link layer of the Ethernet protocol consists of - Headers, Type, Payload and Frame Check Sequence (FCS).

The header features destination and source MAC addresses (each six octets in length), the EtherType field.

The EtherType field is two octets long and it can be used for two different purposes. Values of 1500 and below mean that it is used to indicate the size of the payload in octets, while values of 1536 and above indicate that it is used as an EtherType, to indicate which protocol is encapsulated in the payload of the frame. When used as EtherType, the length of the frame is determined by the location of the interpacket gap and valid frame check sequence (FCS) [11].

[10] https://en.wikipedia.org/wiki/Ethernet_frame

The Type/Length field is used to identify what higher-level protocol is being carried into the frame. (Example: TCP/IP)

Payload is the most important part in the frame as it is the data that we would be transmitting.

The frame check sequence (FCS) is a four-octet cyclic redundancy check (CRC) that allows detection of corrupted data within the entire frame as received on the receiver side.

The FCS field contains a number that is calculated by the source node based on the data in the frame. This number is added to the end of a frame that is sent. When the destination node receives the frame the FCS number is recalculated and compared with the FCS number included in the frame. If the two numbers are different, an error is assumed and the frame is discarded.

The FCS provides error detection only. Error recovery must be performed through separate means. Ethernet, for example, specifies that a damaged frame should be discarded and does not specify any action to cause the frame to be retransmitted. Other protocols, notably the Transmission Control Protocol (TCP), can notice the data loss and initiate retransmission and error recovery.

For Ethernet and other IEEE 802 protocols, the standard states that data is sent least significant bit first, while the FCS is sent most significant bit (bit 31) first. [12]

For detailed description on the format and fields of Ethernet, we could reference to https://ethernethistory.typepad.com/papers/EthernetSpec.pdf

IP protocol - Network Layer

From this part on, we are starting to look at the Internet layer of the Internet Protocol.

Similarly to the Ethernet protocol, we also have IP protocols which will be able to define and enable internetworking at the internet layer of the Internet Protocol Suite. In essence it forms the Internet we are familiar with nowadays. It uses a logical addressing system in which we will introduce below and perform routing.

With IP addresses[11], there are multiple rules to help define it. IPv4 addresses may be represented in any notation expressing a 32-bit integer value. They are most often written in dot-decimal notation, which consists of four octets of the address expressed individually in decimal numbers and separated by periods which is also known as the CIDR notation. CIDR notation combines the address with its routing prefix in a compact format, in which the address is followed by a slash character (/) and the count of leading consecutive 1 bits in the routing prefix (subnet mask) to help masking addresses in order to define a range of addresses in shorthand notation. The Internet Engineering Task Force (IETF) and IANA have restricted from general use various reserved IP addresses for special purposes. So in some ranges, IP addresses are reserved only for public addresses while others may be reserved for private addresses or other special purpose uses. [13]

[11] https://en.wikipedia.org/wiki/IPv4

Two different versions of IP are used in practice today: IPv4 and IPv6. To decide which version to use, we could reference this website on a comparison of the different pros and cons for IPv4 and IPv6:

https://www.geeksforgeeks.org/differences-between-ipv4-and-ipv6/

Headers

So in order to be able to send data correctly, we have a IP protocol format we must follow.

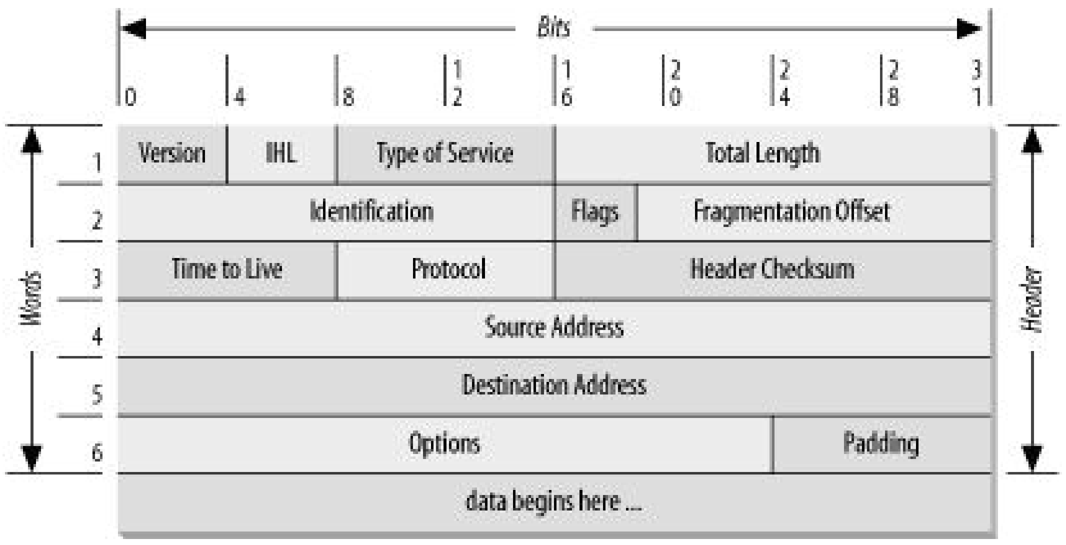
An IP packet consists of a header section and a data section. The data section is just simply the data we would like to send therefore we will have a deep dive on the IP protocol header.

An IP header is header information at the beginning of an Internet Protocol (IP) packet. An IP packet is the smallest message entity exchanged via the Internet Protocol across an IP network. IP packets consist of a header for addressing and routing, and a payload for user data. The header contains information about IP version, source IP address, destination IP address, time-to-live, etc. [14]

The data section includes the payload, the payload of an IP packet is typically a datagram or segment of the higher-level transport layer protocol, but may be data for an internet layer (e.g., ICMP or ICMPv6) or link layer (e.g., OSPF) instead. [15]

Since the header is such a large part and extensive, the IP datagram is typically used to help understand on which sequence the bits should be in and to have a better knowledge on the allocations of bits. There is too much to go into details and in order to build an accurate IP protocol header, we should reference this website on wikipedia or for other references we could read this book:

Semeria, 1996, Understanding IP Addressing: Everything You Ever Wanted To Know



TCP - Transport Layer

Transmission control protocol (TCP) is one of the main protocols in TCP/IP (transmission control protocol/Internet protocol), the suite of communications protocols that is used to connect hosts on the Internet and on most other computer networks as well. [16]

TCP provides reliable, ordered, and error-checked delivery of a stream of octets (bytes) between applications running on hosts communicating via an IP network. Major internet applications such as the World Wide Web, email, remote administration, and file transfer rely on TCP, which is part of the Transport Layerof the TCP/IP suite. SSL/TLS which belongs to the application layer often runs on top of TCP. [17]

TCP uses error correction[12] and data stream control techniques to ensure that packets arrive at their intended destinations uncorrupted and in the correct sequence, thereby making the point-to-point connection virtually error-free. Packets are the most fundamental unit of data transmission on TCP/IP networks.

At the lower levels of the protocol stack, due to network congestion, traffic load balancing, or unpredictable network behaviour, there may be loss of data through transmission of IP packets. TCP detects these problems through various methods that requests retransmission of lost data, rearranges out-of-order data and even helps minimize network congestion to reduce the occurrence of the other problems. The reliability of TCP could be achieved by methods such as Three-way handshake, retransmission and error-detection through a network socket interface. With the help of these guarantee services, TCP will guarantee that all bytes received will be identical and in the same order as those sent. If the data still remains undelivered, the source is notified of this failure. Once the TCP receiver has reassembled the sequence of octets originally transmitted, it passes them to the receiving application. [18]

TCP is connection-oriented, and a connection between client and server is established before data can be sent. Applications that do not require reliable data stream service may use the User Datagram Protocol (UDP), which provides a connectionless datagram service that prioritizes time over reliability. [19]

For example with retransmission which we mentioned above, this will require the receiver to respond with a confirmed message as it receives the data. The sender keeps a record of each packet it sends and maintains a timer from when the packet was sent. The sender re-transmits a packet if the timer expires before receiving the acknowledgement. The timer is needed in case a packet gets lost or corrupted.

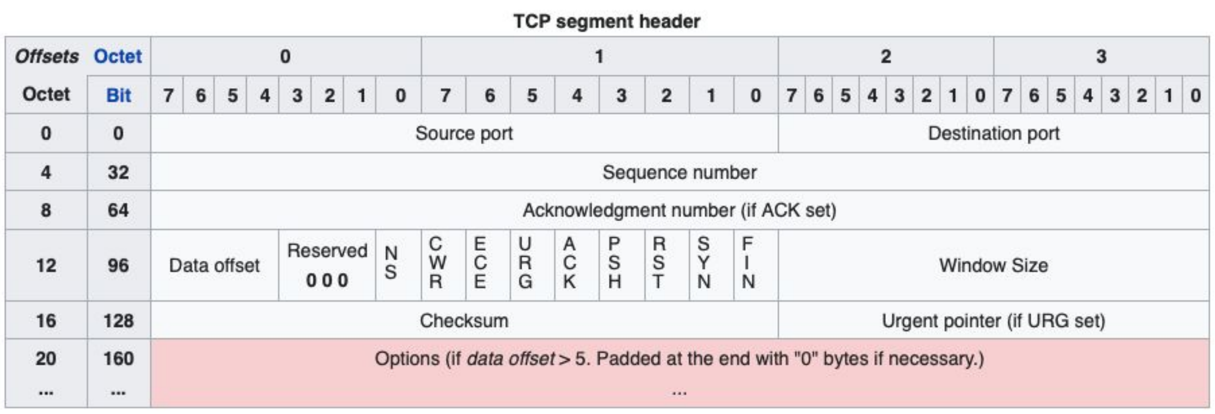
Transmission Control Protocol accepts data from a data stream, divides it into chunks, and adds a TCP header creating a TCP segment. The TCP segment is then encapsulated into an Internet

[12] http://www.linfo.org/tcp.html

Protocol (IP) datagram.A TCP segment consists of a segment header and a data section. The segment header contains 10 mandatory fields, and an optional extension field (Options, pink background in table). The data section follows the header and is the payload data carried for the application. [20]

Since the TCP header is such a large part and extensive, and for accurate locations of Octets we could reference to this website on wikipedia or for other references we could read this book:

Tanenbau, 2003, Computer Networks, section 6



TCP segment header

Offsets	Octet	0								1								2								3							
Octet	Bit	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
0	0	Source port																Destination port															
4	32	Sequence number																															
8	64	Acknowledgment number (if ACK set)																															
12	96	Data offset				Reserved 0 0 0			NS	CWR	ECE	URG	ACK	PSH	RST	SYN	FIN	Window Size															
16	128	Checksum																Urgent pointer (if URG set)															
20 ...	160 ...	Options (if *data offset* > 5. Padded at the end with "0" bytes if necessary.) ...																															

[(Online image from Wikipedia)](#)

Congestion Control

Transmission Control Protocol (TCP) uses a network congestion-avoidance algorithm that includes various aspects of an additive increase/multiplicative decrease (AIMD) scheme, along with other schemes including slow start and congestion window, to achieve congestion avoidance.[21]

To avoid congestive collapse, TCP uses a multi-faceted congestion-control strategy. For each connection, TCP maintains a congestion window, limiting the total number of unacknowledged packets that may be in transit end-to-end. This is somewhat analogous to TCP's sliding window used for flow control.

TCP uses a mechanism called slow start to increase the congestion window after a connection is initialized or after a timeout. It starts with a window, a small multiple of the maximum segment size (MSS) in size. Although the initial rate is low, the rate of increase is very rapid; for every packet acknowledged, the congestion window increases by 1 MSS so that the congestion window effectively doubles for every round-trip time (RTT). When the congestion window exceeds the slow-start threshold, ssthresh, the algorithm enters a new state, called congestion avoidance. In congestion avoidance state, as long as non-duplicate ACKs are received the congestion window is additively increased by one MSS every round-trip time. [22]

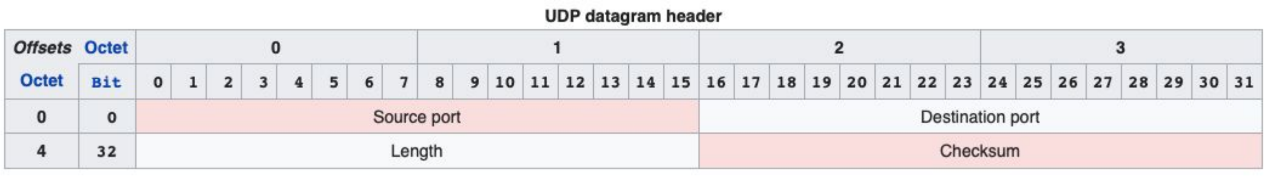
In TCP, the congestion window is one of the factors that determines the number of bytes that can be sent out at any time. The congestion window is maintained by the sender and is a means of stopping a link between the sender and the receiver from becoming overloaded with too much traffic.

UDP - Transport Layer

We have briefly mentioned UDP when we were introducing TCP, as we know that it lacks reliability but in contrast it does not require prior connections and priorities time.

UDP uses a simple connectionless communication model with a minimum of protocol mechanisms. UDP provides checksums for data integrity, and port numbers for addressing different functions at the source and destination of the datagram, there is no guarantee of delivery, ordering, or duplicate protection.[23]

Lacking reliability, UDP applications must be willing to accept some packet loss, reordering, errors or duplication. If using UDP, the end user applications must provide any necessary handshaking such as real time confirmation or other error detection methods to ensure that the message has been received.


UDP datagram header

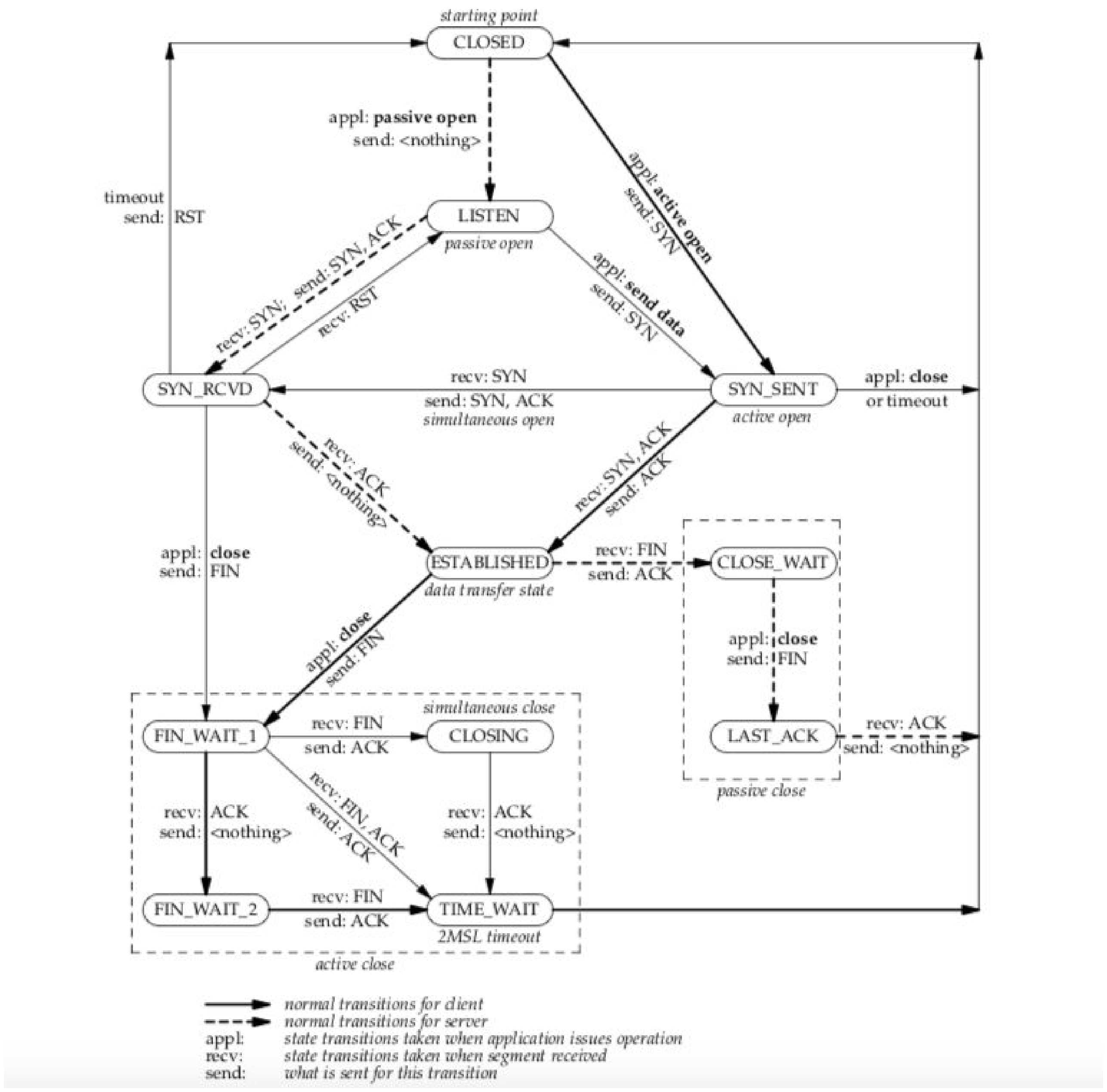
Offsets	Octet	0								1								2								3							
Octet	Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	Source port																Destination port															
4	32	Length																Checksum															

(Online image from Wikipedia)

Differences between UDP and TCP

Here we will have a look at the TCP state diagram and have knowledge on how prior connection between server and clients should be first established before data transmission could take place.

TCP State Diagram



Online image from RFC793

Comparison for UDP and TCP

TCP	connection-oriented	data is bi-directional	reliable	ordered	heavyweight	steaming
UDP	connectionless	data is one direction	unrelia -ble	not ordered	lightweight	no congestion control

Graph created by: ChingWei Hsieh

Two Examples of Application Layer Protocols

Application layers are usually where most users are familiar with, when we are using an application service we most of the time don't care about what is beneath it and how it interconnects with other lower layers. Therefore application layer protocols are also often known as client-server applications. But it is important to know that it is due to these lower layers that enable us to have stable network connections and no loss of data while transmission.

DNS

The process of DNS resolution involves converting a hostname (such as www.example.com) into a computer-friendly IP address (such as 192.168.1.1). It works like a translator so DNS servers eliminate the need for humans to memorize IP addresses such as 192.168.1.1 (in IPv4), or more complex newer alphanumeric IP addresses such as 2400:cb00:2048:1::c629:d7a2 (in IPv6). [24]

The DNS can be quickly and transparently updated, allowing a service's location on the network to change without affecting the end users, who continue to use the same hostname. Users take advantage of this when they use meaningful Uniform Resource Locators (URLs) and e-mail addresses without having to know how the computer actually locates the services. [25]

The Internet maintains two principal namespaces, the domain name hierarchy and the Internet Protocol (IP) address spaces. The Domain Name System maintains the domain name hierarchy and provides translation services between it and the address spaces. Internet name servers and a communication protocol implement the Domain Name System. A DNS name server is a server that stores the DNS records for a domain; a DNS name server responds with answers to queries against its database. [26]

There are four DNS servers involved in loading a webpage:[27]

- DNS recursor - The DNS recursor is a server designed to receive queries from client machines through applications such as web browsers. Typically the recursor is then responsible for making additional requests in order to satisfy the client’s DNS query.
- Root nameserver - The root server is the first step in translating (resolving) human readable host names into IP addresses.
- TLD nameserver - This nameserver is the next step in the search for a specific IP address, and it hosts the last portion of a hostname (In google.com, the TLD server is “com”).
- Authoritative nameserver - The authoritative nameserver is the last stop in the nameserver query. If the authoritative name server has access to the requested record, it will return the IP address for the requested hostname back to the DNS Recursor that made the initial request.

There are various things related to a DNS server, such as recursive DNS servers, authoritative nameservers, DNS lookups, DNS queries and DNS caching. We will not have an introduction to these topics but there are websites and books (such as the *Tanenbaum, Computer Networks*) that have an explanatory description to help understand better.

HTTP

HTTP is an application layer protocol designed within the framework of the Internet protocol suite. Its definition presumes an underlying and reliable transport layer protocol, and Transmission Control Protocol (TCP) is commonly used. HTTP resources are identified and located on the network by Uniform Resource Locators (URLs), using the Uniform Resource Identifiers (URI's) schemes http and https. [28]

HTTP functions as a request–response protocol in the client–server computing model. A web browser, for example, may be the client and an application running on a computer hosting a website may be the server. The client submits an HTTP request message to the server. The server, which provides resources such as HTML files and other content, or performs other functions on behalf of the client, returns a response message to the client. [29]

There are various request methods such as *connect*, it enables HTTP to perform various actions, such as interacting with proxy[13] servers which could provide us different services such as privacy, security or control the complexity of the request.

The request message from the client side consists of the following:[30]

- a request line (e.g., GET /images/logo.png HTTP/1.1, which requests a resource called /images/logo.png from the server)
- request header fields (e.g., Accept-Language: en)
- an empty line
- an optional message body

The response message from the server side consists of the following:[31]

- a status line which includes the status code and reason message (e.g., HTTP/1.1 200 OK, which indicates that the client's request succeeded)
- response header fields (e.g., Content-Type: text/html)
- an empty line
- an optional message body
-

Additional Resources

We could learn more about the specifics of HTTP calls, statuses on this website:

https://www.w3.org/History/19921103-hypertext/hypertext/WWW/Protocols/HTTP.html

And to learn about how HTTP interacts with the transport layer with protocols like TCP, we could reference to this website:

https://blog.cloudflare.com/http3-the-past-present-and-future/

For more detailed information about HTTP, we could read this tool:

https://tools.ietf.org/html/draft-ietf-quic-http-22

[13] https://en.wikipedia.org/wiki/Proxy_server

USB v.2,3 and C

A USB, also known as Universal Serial Bus is specified to be an industry standard extension to the PC architecture with a focus on Computer Telephony Integration (CTI), consumer, and productivity applications. [32] Universal Serial Bus (USB) is an industry standard that establishes specifications for cables and connectors and protocols for connection, communication and power supply (interfacing) between computers, peripherals and other computers.[33]

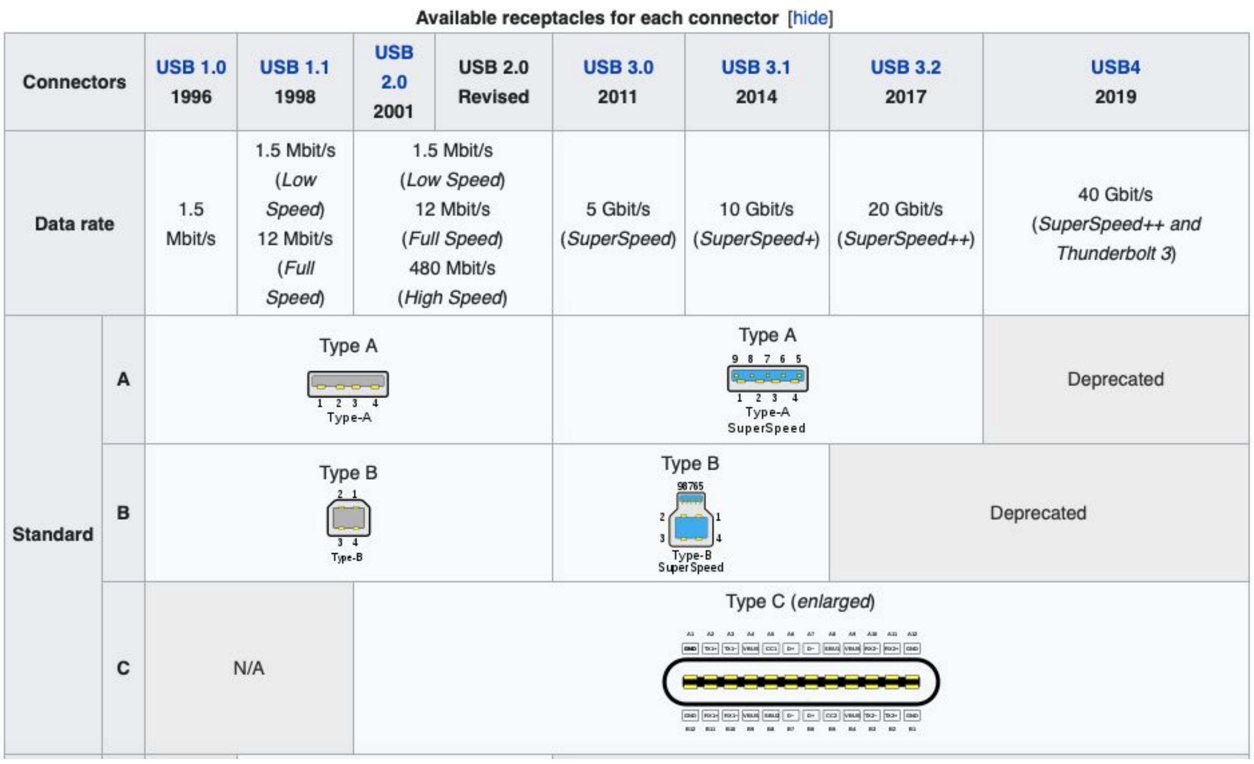
There have been four generations of USB specifications: USB 1.x, USB 2.0, USB 3.x and USB4.[34] USB-C (formally known as USB Type-C) is a 24-pin USB connector system with a rotationally symmetrical connector.[35] Which also means that you would not encounter the problem where you plug in on the wrong side and damage the chip.

Physical Layer

The Physical Layer represents the PHY and the actual physical connection between two ports.

Physical connection

The physical portion of the link represents the two differential data pairs: one transmit path and one receive path. [36] According to different versions of the USB and different types, we could find a chart which shows a comparison on how different versions also have different types which also results in different data rates.



Available receptacles for each connector [hide]

Connectors		USB 1.0 1996	USB 1.1 1998	USB 2.0 2001	USB 2.0 Revised	USB 3.0 2011	USB 3.1 2014	USB 3.2 2017	USB4 2019
Data rate		1.5 Mbit/s	1.5 Mbit/s (*Low Speed*) 12 Mbit/s (*Full Speed*)	1.5 Mbit/s (*Low Speed*) 12 Mbit/s (*Full Speed*) 480 Mbit/s (*High Speed*)		5 Gbit/s (*SuperSpeed*)	10 Gbit/s (*SuperSpeed+*)	20 Gbit/s (*SuperSpeed++*)	40 Gbit/s (*SuperSpeed++ and Thunderbolt 3*)
Standard	A	Type A (Type-A)				Type A (Type-A SuperSpeed)			Deprecated
	B	Type B (Type-B)				Type B (Type-B SuperSpeed)		Deprecated	
	C	N/A		Type C (*enlarged*)					

Online picture from Wikipedia

On its 2.0 Standard release, USB permits three speeds of communication: Low Speed 1.5M bits/s, Full Speed 12 Mbits/s and High Speed 480 Mbits/s. In order to get a higher data rate, the physical cables or electrical conventions are adjusted accordingly also. For example, for superspeed data rate we need the most twisted pair cables and with some versions we have half duplex while the others use full duplex.

At the high speed frequency it is difficult to synthesize HDL code without modification. That is why the USB Transceiver Macrocell Interface (UTMI) Standard is defined. [37]

PHY

A PHY chip is integrated into most USB controllers in hosts or embedded systems and provides the bridge between the digital and modulated parts of the interface.

The Physical Layer receives the 8bit data from the link layer, scrambles it to reduce EMI, encodes the scrambled data into 10bit symbols, and serializes the data to be sent over the cable. At the receiving end, the reverse operation is performed. The transmitter functions of the Physical Layer include data scrambling, 8b10b encoding, and serialization. The receiver functions of the Physical Layer include de-serialization, 8b10b decoding, data descrambling, and receiver clock and data recovery. [38]

The USB PHY under verification has multiple implementation options based on if it is host, device or OTG controller[14], and if it is single or multi-ported. PHY implementations can support several data signaling rates such as high speed (HS), full speed (FS) and low speed (LS).[39] In general, a host or OTG PHY supports all data signalling rates. The interface between PHY and Serial Interface Engine, (SIE) for host, device and OTGimplementations is the UTMI+ interface with small internal changes to increase observability and controllability. For more detailed information we could refer to the UTMI specification. [40] For the PHY architecture we could refer to this datasheet, Embedded USB2 (eUSB2) Physical Layer Supplement to the USB Revision 2.0 Specification

For detailed information we could refer to this datasheet: Datasheet

Here we will have a see a block diagram on how the physical connections are made on the USB and how the hardware components are connected to each other.

[14] https://en.wikipedia.org/wiki/USB_On-The-Go

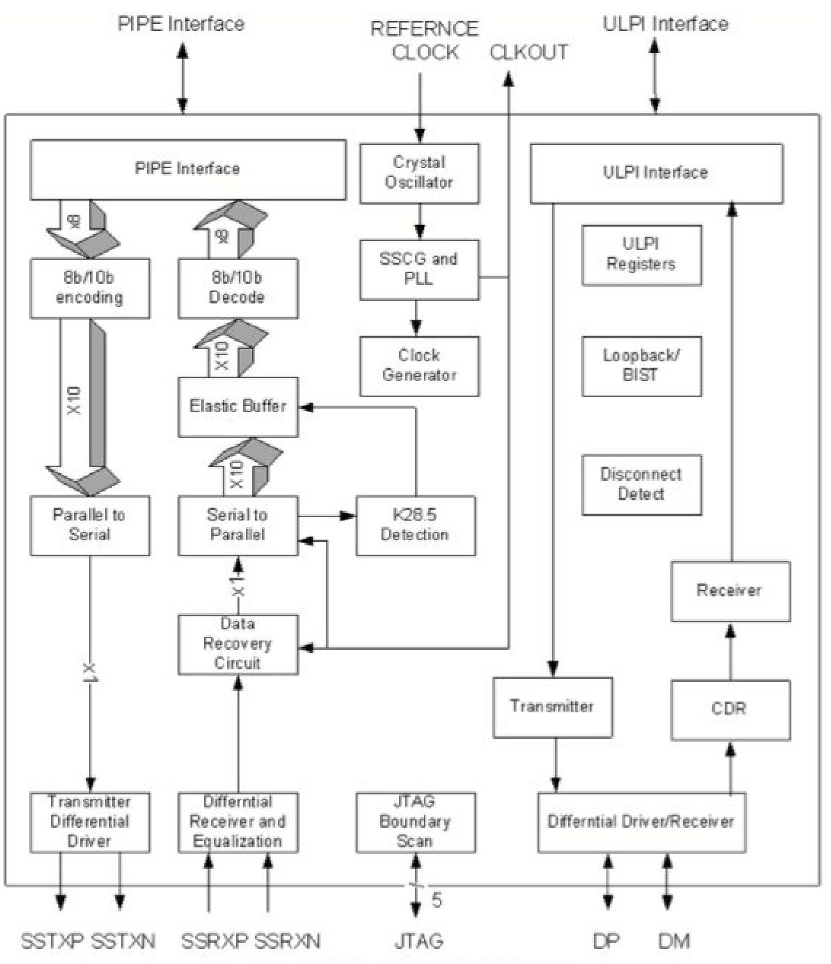


Figure 1-2. Functional Block Diagram

Online Image from Datasheet

Data Link Layer

In the general SuperSpeed architecture, the Link Layer manages the port-to-port flow of data between the host and the device. A Link is the logical and physical connection between two ports (an upstream-facing port and a downstream-facing port).

Header Packet Framing

The Link Layer also handles Header Packet Framing, by taking the 12 bytes of each Header Packet created by the Protocol Layer and framing them with eight more bytes of information. These include 4 bytes to indicate the start of the Header Packet and the end of the Packet Framing, 2 bytes of cyclic redundancy check (CRC), and a 2 byte Link Control Word.

Link Commands

Link Commands are used between two linked ports to communicate information between the upstream and the downstream port, ensuring link-level data integrity, flow control, and link power management. These commands are only sent from the transmitter link layer to receiver link layer, and are not routed onwards to other links.

A Link Command Word handles packet acknowledgement and error recovery, packet flow control, and link power management. The Command Word includes fields with which a receiving port can indicate whether the header packet is good, bad or should be retried. Other fields provide information about requests to change power status, and a ‘device present’ indicator. [41]

In practical terms, the Link Layer contains the media access layer including state machines for link training, flow control and status. This is connected, via the PHY/MAC interface, to Physical Layer functions such as the physical coding sublayer, which handles the 8b10b encoding/decoding, buffering and receiver detection, and the physical media attachment layer, which includes a 10bit interface, SerDes and analog buffers. For more details about the how the decoding works, we could refer to this website:

https://www.mindshare.com/files/resources/MindShare_Intro_to_USB_3.0.pdf

PIPE

The interface between the PHY and the MAC is known as the PIPE interface. The data crossing the interface can be 8, 16 or 32bit wide. A PCLK is recovered from an incoming connection and passed from the PHY to the MAC. There are also signals on the PIPE interface to indicate whether what is on the data bus is real data or special symbols; 12 or 16bit command signals to control the PHY layer; or other status signals. [42] For more detailed information regarding the PIPE, we could refer to the datasheet. Datasheet

Network Layer

In USB 3.0, the protocol layer converts requests from the functional layer into transactions consisting of packets, and manages the end to end data flow between the host and the device.

The protocol layer’s functions include:

- Ensuring end-to-end reliability for packets
- Effective power management
- Effective use of bandwidth

Packets in USB 3.0 begin at the transmitter protocol layer and end at the receiver protocol layer.

Like USB 2.0, the SuperSpeed bus carries data, address, status, and control information. Four packet types are defined. Breaking down the header packets, two of them, the Transaction Packet (TP) and Data Packet (DP), remain the same as in USB 2.0. Two additional header packet types, Isochronous Timestamp Packet (ITP) and Link Management Packet (LMP), are newly introduced by USB 3.0. [43] For more detailed information on header packets and the flow control, we could reference to this website.A brief summary of packet types, flow control and handshakes

The header and data packets include an address (holding the device address, endpoint number, and direction); and a route string describing the path between the host and device.

The application data is embedded in the data packet payload. The host starts all the data transfers, with packets being routed through all the intermediate hubs to the target device. The USB data is transferred on a differential serial line (USB DP and USB DM), using NRZI[15] coding. [44]

Devices respond to, or defer, the packet. All packets from the device are routed to the host. Deferred requests are restarted asynchronously by the device. The bus enters low-power mode whenever transactions are not happening.

There are 4 main USB packet types :Token, Data, Handshake and Start of Frame . (Each packet is constructed from different field types, namely SYNC, PID, Address, Data, Endpoint, CRC and EOP.) [45]

USB packet types

USB token packet: used to access the correct address and endpoint.

FIELD	SYNC	PID	ADDRESS	ENDPOINT	CRC5	EOP
#BITS	8/32	8	7	4	5	3

Graph created by: ChingWei Hsieh

USB data packet: may be of variable length, dependent upon the data.

FIELD	SYNC	PID	DATA	CRC16	EOP
#BITS	8/32	8	0-8192	16	3

Graph created by: ChingWei Hsieh

USB handshake packets: used to signal the status of a transaction ie pass/fail.

FIELD	SYNC	PID	EOP
#BITS	8/32	8	3

Graph created by: ChingWei Hsieh

USB Start of Frame packet: To complete a full message the packets are grouped into frames. To identify the frame you need a Start of Frame packet (SOP).

FIELD	SYNC	PID	Frame Number	CRC5	EOP

[15] https://en.wikipedia.org/wiki/Non-return-to-zero#NRZI

#BITS	8/32	8	11	5	3

Graph created by: ChingWei Hsieh

Header Packet example - Transaction packets/types

The Transaction Packet traverses all the links in the path directly connecting the host and a device, and is used to control the flow of data packets, configure devices and hubs, and so forth.

There are four transaction types:

- Bulk Transactions (Large sporadic transfers using all remaining available bandwidth, but with no guarantees on bandwidth or latency e.g., file transfers)
- Control Transactions (mainly performs control read and control write)
- Interrupt Transactions (Devices that need guaranteed quick responses (bounded latency) such as pointing devices, mice, and keyboards
- Isochronous Transactions (At some guaranteed data rate (for fixed-bandwidth streaming data) but with possible data loss e.g., realtime audio or video) [46]

Each type of transaction has both IN and OUT directions. [47] These transactions are able to perform tasks such as error detection and retry, control read and control write.

For more detailed explanation and information about USBs, we could refer to these websites :

- 2000, Universal Serial Bus Communication (http://sdphca.ucsd.edu/lab_equip_manuals/usb_20.pdf)
- FTDI, 2009, USB Data Packet Structure, https://www.ftdichip.com/Support/Documents/TechnicalNotes/TN_116_USB%20Data%20Structure.pdf
- 2000, Universal Serial Bus Communication (http://sdphca.ucsd.edu/lab_equip_manuals/usb_20.pdf)
- https://www.beyondlogic.org/usbnutshell/usb6.shtml
- https://www.techdesignforums.com/practice/technique/usb-3-0-protocol-layer-2/

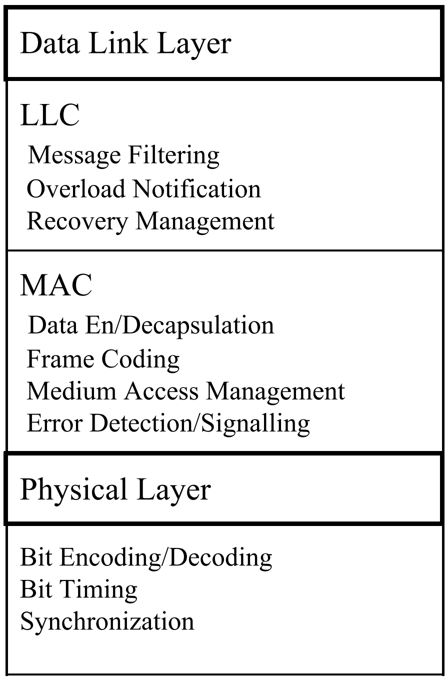
CAN

The Controller Area Network (CAN) protocol is an asynchronous serial bus with Non-Return to Zero (NRZ) bit coding designed for fast, robust communications in harsh environments, such as automotive and industrial applications. The CAN protocol allows the user to program the bit rate, the sample point of the bit, and the number of times the bit is sampled. With these features, the network can be optimized for a given application. [48] CAN is widely used in large automobile companies for various applications.

Being a widely used bus protocol in automotive distributed embedded systems, the limitation of having a communication bandwidth (up to 1 Mbps) and payload size (up to 8 Bytes) limits the applicability. CAN with flexible data rate (CAN-FD) is an improved CAN-based communication protocol, with higher communication bandwidth (up to 8 Mbps for the payload) and increased payload size (up to 64 Bytes). [49] For higher speed performance and flexible data rate, it is common to use CAN-FD nowadays and there are several papers that feature it. To have a better knowledge on CAN-FD, we could reference to the datasheets websites that will be attached in index 1.

In contrast to traditional networks, such as USB and Ethernet, CAN does not send large data blocks point-to- point under the supervision of a central bus master. In a CAN network, many short messages are broadcasted to the entire network, which provides data consistency in every node of the system. Therefore it is also called a broadcasting bus.

The ISO 11898 standard of CAN describes the physical and the data link layer. The lower protocol levels of CAN (Physical Layer and Data Link Layer) are standardized in the ISO/OSI layer model. Protocols based on layer 7 (application layer) are summarized in different, partly manufacturer-specific standards. Examples for these CAN-based protocols are CANopen and J1939. [50]



Graph created by: ChingWei Hsieh

Physical Layer

The Physical Layer defines how signals are actually transmitted and therefore deals with the description of Bit Timing, Bit Encoding, and Synchronization.[51]

And of course, we also have the physical/electrical property standards for building a CAN.

Electrical Properties

Cables and most components are defined in the ISO 11898, as it specifies hardware requirements such as the impedance of the cables and the usage of twisted pair cables. In this ISO, the two wired balancing signal scheme was used in this CAN standard therefore it is also known as "high-speed CAN".

Many devices can be connected to the CAN bus, ranging from complex electronic control units to simple I/O devices. Each device is called a node. Each CAN node transmits differentially over two wires: CAN High and CAN Low.

CAN transmits differentially over two lines, CAN High and CAN Low. There are two logic states:

- Dominant – Logic 0
- Recessive – Logic 1 [52]

Bit Timing

Each bit on the CAN bus is, for timing purposes, divided into at least 4 quanta. The quanta are logically divided into four segments:

- the Synchronization Segment
- the Propagation Segment
- the Phase Segment 1
- the Phase Segment 2

Most CAN controllers allows the programmer to set the bit timing using the following parameters:

- A clock prescaler value
- The number of quanta before the sampling point
- The number of quanta after the sampling point
- The number of quanta in the Synchronization Jump Width, SJW [53]

Usually two registers are provided for this purpose: btr0 and btr1. [54] According to different hardwares, it is important to reference the datasheet before configuring.

Synchronization

In order to adjust the on-chip bus clock, the CAN controller may shorten or prolong the length by an integral number of quanta. The maximum value of these bit time adjustments are termed the Synchronization Jump Width, SJW (which we briefly mentioned above in the Bit Timing section) in order to match the bit timing. There are two types of synchronizations the CAN could perform.

- Hard synchronization: is performed whenever there is a ’recessive’ to ’dominant’ edge during BUS IDLE.
- Resynchronization: All other ’recessive’ to ’dominant’ edges fulfilling the rules 1 and 2 will be used for RESYNCHRONIZATION with the exception that a node transmitting a dominant bit will not perform a RESYNCHRONIZATION as a result of a ’recessive’ to ’dominant’ edge with a positive PHASE ERROR, if only ’recessive’ to ’dominant’ edges are used for resynchronization. [55][56]

Data Link Layer

Remember that the Data Link layer consists of MAC and LLC, The MAC sublayer represents the kernel of the CAN protocol. It presents messages received from the LLC sublayer and accepts messages to be transmitted to the LLC sublayer.

The CAN bus is a broadcast type of bus. This means that all nodes can ‘hear’ all transmissions. There is no way to send a message to just a specific node; all nodes will invariably pick up all traffic including the transmitting device. The CAN hardware, however, provides local filtering so that each node may react only on the interesting messages. [57] For each device the data in a frame is transmitted sequentially but in such a way that if more than one device transmits at the same time the highest priority device is able to continue while the others back off. [58]

Framing

CAN has four frame types:

- Data frame: a frame containing node data for transmission.
- Remote frame: a frame requesting the transmission of a specific identifier.
- Error frame: a frame transmitted by any node detecting an error.
- Overload frame: a frame to inject a delay between data and/or remote frame.

For each frame type, according to the configuration the CAN network is possible to be presented in two different frame formats: the base frame format (CAN 2.0A & CAN 2.0B) which supports 11-bit identifiers, and the extended frame format (CAN2.0B only) which supports 29-bit identifiers by allowing the addition of an 18-bit identifier extension. An identifier extension bit (IDE) determines if the 18-bit ID extension is being used. According to different needs, you should choose wisely between base frame format or the extended version as it brings different pros and cons.

Each frame type has their own specifications on the allocations of the bits but we will not go into the deep details in this paper, for information we have to refer to the datasheet:

http://esd.cs.ucr.edu/webres/can20.pdf

Brief introduction to some important fields for a base frame format:

- Identifier(Arbitration): A identifier which also represents the message priority
- The Data Length Code: indicates the length of data in bytes.

- The RTR bit (Remote transmission request): determines between data frames (0) and remote frames (1).
- CRC: stands for cyclic redundancy check, which is used for error detection.
- Ack slot bit: All nodes that receive a frame without finding any errors transmit a dominant 0, which overrides a recessive 1 sent by the transmitter.
- End of frame: is confirmed by the transmission of 7 recessive 1s.

The frame segments START OF FRAME, ARBITRATION FIELD, CONTROL FIELD, DATA FIELD and CRC SEQUENCE are coded by the method of bit stuffing. Whenever a transmitter detects five consecutive bits of identical value in the bit stream to be transmitted it automatically inserts a complementary bit in the actual transmitted bit stream. The remaining bit fields (DATA LENGTH CODE, RTR, ACK, EOF) are in fixed format and not bit stuffed. The bit stream in a message is coded according to the Non-Return-to-Zero (NRZ) method. This means that during the total bit time the generated bit level is either 'dominant' or 'recessive'. [59]

EOF is functioning as message validation for our bus, as it has different meanings depending on the different roles of the transmitter or receiver.

Transmitter: The message is valid for the transmitter, if there is no error until the end of END OF FRAME. If a message is corrupted, retransmission will follow automatically and according to prioritization. In order to be able to compete for bus access with other messages, retransmission has to start as soon as the bus is idle.

Receivers:The message is valid for the receivers, if there is no error until the last but one bit of END OF FRAME. The value of the last bit of END OF FRAME is treated as 'don't care', a dominant value does not lead to a FORM ERROR.

Arbitration (Priorities)

Any CAN controller may start a transmission when it has detected an idle bus. This may result in two or more controllers starting a message at the same time. The conflict is resolved in the following way. The transmitting nodes monitor the bus while they are sending. If a node detects a dominant level when it is sending a recessive level itself, it will immediately quit the arbitration process and become a receiver instead. The arbitration is performed over the whole Arbitration Field and when that field has been sent, exactly one transmitter is left on the bus. This node continues the transmission as if nothing had happened. The other potential transmitters will try to retransmit their messages when the bus becomes available next time. No time is lost in the arbitration process.[60]

Error checking/signalling

The CAN protocol defines five different ways of detecting errors. Each of them are able to raise an ERROR FLAG to notify the system that an error has occurred.

Bit Monitoring (Bit error): Each transmitter on the CAN bus monitors (i.e. reads back) the transmitted signal level. If the bit level actually read differs from the one transmitted, a Bit Error is signaled.

Bit Stuffing (Stuff error): When five consecutive bits of the same level have been transmitted by a node, it will add a sixth bit of the opposite level to the outgoing bit stream. The receivers will remove this extra bit. This is done to avoid excessive DC components on the bus, but it also gives the receivers an extra opportunity to detect errors: if more than five consecutive bits of the same level occurs on the bus, a Stuff Error is signaled.

Frame check (Form error): Some parts of the CAN message have a fixed format, if a CAN controller detects an invalid value in one of these fixed fields, a Form Error is signaled.

Acknowledgement Check (Acknowledgement error): If the transmitter can't detect a dominant level in the ACK slot, an Acknowledgement Error is signaled.

Cyclic Redundancy Check (CRC error): Each message features a 15-bit Cyclic Redundancy Checksum (CRC), and any node that detects a different CRC in the message than what it has calculated itself will signal an CRC Error. [61] [62]

Fault confinement

The main purpose of the fault confinement function is to perform a self-checking mechanism for distinguishing short disturbances from permanent failures. Therefore, we could act accordingly to know which approach we should make to recover.

Every CAN controller along a bus will try to detect the errors outlined above within each message. If an error is found, the discovering node will transmit an Error Flag, thus destroying the bus traffic. The other nodes will detect the error caused by the Error Flag (if they haven't already detected the original error) and take appropriate action, i.e. discard the current message.

Each node maintains two error counters: the Transmit Error Counter and the Receive Error Counter. There are several rules governing how these counters are incremented and/or decremented. In essence, a transmitter detecting a fault increments its Transmit Error Counter faster than the listening nodes will increment their Receive Error Counter.

A node starts out in Error Active mode. When any one of the two Error Counters raises above 127, the node will enter a state known as Error Passive and when the Transmit Error Counter raises above 255, the node will enter the Bus Off state.

- An Error Active node will transmit Active Error Flags when it detects errors.
- An Error Passive node will transmit Passive Error Flags when it detects errors.
- A node which is Bus Off will not transmit anything on the bus at all.

The rules for increasing and decreasing the error counters are somewhat complex, but the principle is simple: transmit errors give 8 error points, and receive errors give 1 error point. Correctly transmitted and/or received messages causes the counter(s) to decrease. [63]

Message filtering

The purpose of filtering is to decide which messages received by the LLC sublayer are actually to be accepted.

Message filtering is based upon the whole Identifier. Optional mask registers that allow any Identifier bit to be set ’don’t care’ for message filtering, may be used to select groups of Identifiers to be mapped into the attached receive buffers. If mask registers are implemented every bit of the mask registers must be programmable, i.e. they can be enabled or disabled for message filtering.[64]

Overload notifications

If a CAN node receives messages faster than it can process the messages, then the CAN module will signal an overload condition and then send an overload interrupt to the CPU.

Causes of an Overload Frame

- A receiver node needs more time to process current data before receiving the next Message Frame .
- The detection of a dominant bit during the INTermission field [65]

Serial ATA

Serial ATA is a high-speed serial link replacement for the parallel ATA attachment of mass storage devices (hard drive (HDD), a solid-state drive (SSD), or an optical drive) and a ribbon cable is typically used to connect the two interfaces. The following section will have a deeper discussion about the evolution and discussion about the parallel and serial interfaces. Throughout the years, there have been many revisions on the versions and has been adapted with speed, data rates, backward compatibilities and more.

The serial link employed is a high-speed differential layer that utilizes Gigabit technology and 8b/10b encoding.

State machines are heavily encoded in this bus system and the control flow is mainly determined within. The Transport control state machine and the Link state machine are the two core sub-modules that control overall operation.

The Link state machine controls the operation(s) related to the serial line and the Transport control state machine controls the operation(s) relating to the host platform. The two state machines coordinate their actions and utilize resources to transfer data between a host computer and attached mass storage device.

The host Link state machine communicates via the serial line to a corresponding Link state machine located in the device. The host Transport machine also likewise communicates with a corresponding device Transport state machine. The two Link state machines ensure that control sequences between the two Transport control state machines are properly exchanged. [66]

Parallel vs Serial

Serial ATA succeeded the older Parallel ATA (PATA) standard, offering several advantages over the older interface: reduced cable size and cost (seven conductors instead of 40 or 80), native hot swapping, faster data transfer through higher signaling rates, and more efficient transfer through an (optional) I/O queuing protocol. Which results in the most significant difference from Serial ATA and Parallel ATA. At the hardware interface level, SATA and PATA devices are completely incompatible: they cannot be interconnected without an adapter.

SATA host adapters and devices communicate via a high-speed serial cable over two pairs of conductors. In contrast, parallel ATA uses a 16-bit wide data bus with many additional support and control signals, all operating at much lower frequency. To ensure backward compatibility with legacy ATA software and applications, SATA uses the same basic ATA and ATAPI command sets as legacy ATA devices.

Therefore, PATA has mostly been replaced by SATA for any use.

Physical Layer

Electrical Properties

SATA encompasses two ports: The data connector and the power connector. The former is the short, L-shaped, seven-pin connector, while the latter is the more extended 15-pin connector — the taller "L" of the two.

Both connectors are typically reversed on the drives they allow connections for, with the bases of their respective "L" shapes facing one another. Beyond length, they can be told apart by the cables that connect to them. Where the SATA data cable is usually made up of solid plastic, which extends into a flat, single-band cable, the SATA power connector will continue from its head to multiple, thin, rounded wires of different colors.

Both cables are required for SATA devices to work, and both do different jobs. The data cable provides the high-speed connection to the rest of the computer, transferring information back and forth as requested, while the power cable is what gives the drive the electricity to run in the first place. [67]

Similarly to the previous sections, the hardware specifications should be carefully and detailedly checked with the datasheet. It includes the impedance, which type of cables, which connectors, voltages,pins, etc. we should use. The datasheet also includes the specifications for the transmitter and receiver characteristics also the BER and jitter effects we could reference to.

PHY

The Phy layer is divided into a transmitter, interconnect, and a receiver.

The PHY layer is responsible for detecting the other SATA/device on a cable, and link initialization. During the link-initialization process, the PHY is responsible for locally generating special out-of-band signals by switching the transmitter between electrical-idle and specific 10b-characters in a defined pattern. Once link-initialization has completed, the link-layer takes over data-transmission, with the PHY providing only the 8b/10b conversion before bit transmission. [68]

Jitter

In SATA systems, random jitter is a significant portion of the total jitter causing occasional errors to occur. When a bit error occurs, the error is detected when an entire frame of bits is received. The bit error is corrected by retransmitting the frame. If two bit errors occur within a single frame, the corrective action is the same. The data throughput on the channel is diminished when frames are retransmitted. Jitter tests are compliance tests done on an individual SATA component, a device, host, or interconnect to ensure system performance.

Another performance measurement is the Frame Error Rate. A frame error rate test is a system performance test done on a combination of SATA compliant components.

States

This system is heavily relied on state diagrams and most of the functional logic was implemented according to the complicated state diagrams. Phy relies on detection of received ALIGN primitives for state transitions to be enabled to proceed further for different tasks.

Different states include: HR_RESET, HR_AwaitCOMINIT, HR_AwaitNoCOMINIT, HR_Calibrate, HR_COMWAKE, HR_AwaitCOMWAKE, HR_AwaitNoCOMWAKE, HR_AwaitAlign, DR_Reset,, DR_COMINIT, DR_AwaitCOMWAKE, etc.

Here are some states we chose to have a brief knowledge on how the states interact with the system for example.

OOB Signaling Tests: OOB signaling is used to signal specific actions during conditions where the receiving interface is in an active mode, a low interface power state, or a test mode.

Idle Bus Status: During the idle bus condition, the differential signal diminishes to zero while the common mode level remains.

COMRESET always originates from the host controller, and forces a hardware reset in the device. It is indicated by transmitting bursts of data separated by an idle bus condition.

COMINIT always originates from the drive and requests a communication initialization.

COMWAKE may originate from either the host controller or the device. It is signaled by transmitting six bursts of data separated by an idle bus condition.

For more detailed configurations, jitter synchronizations, calibrations and information on how the state flows and what the state's meaning is, we should refer to the datasheet, Serial ATA:High Speed Serialized AT Attachment section 7.

Error Detections on PHY layer when:

- No device present
- OOB signaling sequence failure
- Phy internal error (loss of synchronization of communications link)

Data Link Layer

After the PHY-layer has established a link, the link layer is responsible for transmission and reception of Frame Information Structures (FISs) over the SATA link. FISs are packets containing control information or payload data. Each packet contains a header (identifying its type), and payload whose contents are dependent on the type. The link layer also manages flow control over the link. [69]

The Link layer transmits and receives frames, transmits primitives based on control signals from the Transport layer, and receives primitives from the Phy layer which are converted to control signals to the Transport layer. [70]

Frame Transmission: When requested by the Transport layer to transmit a frame, the Link layer provides the following services:

- Resolves arbitration conflicts if both host and device request transmission
- Inserts frame envelope around Transport layer data

- Receives data in the form of Dwords from the Transport layer
- Calculates CRC on Transport layer data
- Transmits frame
- Provides frame flow control (in response to requests from the FIFO or the peer Link layer)
- Receives frame receipt acknowledge from peer Link layer
- Reports good transmission or Link/Phy layer errors to Transport layer
- 8b/10b encoding
- Scrambles data Dwords in such a way to distribute the potential EMI emissions over a broader range

Frame Reception: When data is received from the Phy layer, the Link layer provides the following services:

- Acknowledges to the peer Link layer readiness to receive a frame
- Receives data in the form of encoded characters from the Phy layer
- Decodes the encoded 8b/10b character stream into aligned Dwords of data
- Removes the envelope around frames
- Calculates CRC on the received Dwords
- Provides frame flow control in response to requests from the FIFO or the peer Link layer
- Compares the calculated CRC to the received CRC
- Reports good reception or Link/Phy layer errors to Transport layer and the peer Link layer
- Descrambles data Dwords received from a peer Link layer [71]

Coding structure:

This part is very extensive and complicated, since the Link Layer performs different types of encoding/decoding structure as needed for each action.

Information to be transmitted over Serial ATA shall be encoded by a byte at a time along with a data or control character indicator into a 10-bit encoded character and then sent serially bit by bit. Information received over Serial ATA shall be collected ten bits at a time, assembled into an encoded character, and decoded into the correct data characters and control characters. The 8b/10b code allows for the encoding of all 256 combinations of eight-bit data. A subset of the control character set is utilized by Serial ATA. [72]

Transport Layer

The Transport layer simply constructs Frame Information Structures (FIS's) for transmission and decomposes received Frame Information Structures. The Transport layer maintains no context in terms of ATA commands or previous FIS content. It is mostly describing the applications we could apply to our SATA.

This layer has the responsibility of acting on the frames and transmitting/receiving the frames in an appropriate sequence. The transport layer handles the assembly and disassembly of Frame Information Structures. In an abstract fashion, the transport layer is responsible for creating and encoding FIS structures requested by the command layer, and removing those structures when the frames are received. [73]

Versions of SATA

In this section we will briefly go through the different versions and applications of SATA throughout the history.

Slimline connector: SATA 2.6 is the first revision that defined the slimline connector, intended for smaller form-factors such as notebook optical drives.

eSATA: eSATA enters an external storage market served also by the USB and FireWire interfaces. Most external hard-disk-drive cases with FireWire or USB interfaces use either PATA or SATA drives and "bridges" to translate between the drives' interfaces and the enclosures' external ports; this bridging incurs some inefficiency. An eSATA later has also developed a version known as eSATAp, an eSATAp port combining the four pins of the USB 2.0 (or earlier) port, the seven pins of the eSATA port, and optionally two 12 V power pins. [74]

Mini-SATA: The physical dimensions of the mSATA connector are identical to those of the PCI Express Mini Card[16] interface. Common applications of Mini-SATA include notebooks, laptops and other devices with a smaller footprint. [75][76]

SATA Express: SATA Express, initially standardized in the SATA 3.2 specification, is an interface that supports either SATA or PCI Express storage devices. The host connector is backward compatible with the standard 3.5-inch SATA data connector, allowing up to two legacy SATA devices to connect and allowing bandwidths of up to 2 GB/s. Instead of the otherwise usual approach of doubling the native speed of the SATA interface, PCI Express was selected for achieving data transfer speeds greater than 6 Gbit/s. [77]

M.2: Formerly known as the Next Generation Form Factor (NGFF), is a specification for computer expansion cards and associated connectors. It replaces the mSATA standard, which uses the PCI Express Mini Card physical layout. Having a smaller and more flexible physical specification, together with more advanced features, the M.2 is more suitable for solid-state storage applications in general, especially when used in small devices such as ultrabooks and tablets. [78]

[16] https://en.wikipedia.org/wiki/PCI_Express#PCI_Express_Mini_Card

Firewire

The IEEE 1394 interface is a serial bus interface standard for high-speed communications and real-time data transfer. The interface also is known as Firewire® by Apple® later on, Sony created its own trademark for 1394 called iLINK.

IEEE 1394 can connect up to 63 peripherals in a tree topology, as opposed to Parallel SCSI's Electrical bus. It allows peer-to-peer device communication, such as communication between a scanner and a printer, to take place without using system memory or the CPU. IEEE 1394 also supports multiple hosts per bus. It is designed to support plug-and-play and hot swapping. [79]

Comparison to USB

IEEE 1394 can transmit data at three rates: 98.304, 196.608 and 393.216Mb/s (megabits per second), which are usually rounded to 100Mb/s, 200Mb/s and 400Mb/s and officially labelled S100, S200 and S400 respectively. As you can see even the lowest rate is nearly 10 times that of USB.

Like USB, IEEE 1394 is designed to allow hot swapping. Devices can be connected and/or disconnected without turning off the power or resetting, etc. The device which is performing the bus manager role constantly monitors bus status, and reconfigures it dynamically whenever nodes are added or disconnected. [80]

Comparison to Thunderbolt

The main difference between Firewire and Thunderbolt is speed. Thunderbolt was initially designed to use fiber optic cables rather than traditional copper wires which Firewire uses. Therefore, Firewire has a maximum throughput of around 3.2Gbps while Thunderbolt is capable of achieving 10Gbps.

The number of devices you could connect is also another topic which draws the attention of the differences. Firewire is superior when it comes to numbers as it can daisy chain up to 63 devices at once in comparison to Thunderbolt which could only connect 6 devices at a time. [81]

Firewire applications

FireWire devices are generally multimedia based; examples are Digital Cameras and Camcorders, Digital VCRs, and Pro-audio equipment like DAT stations and other multi-track recorders. However, they can also be general PC peripherals like Hard Disk Drives, CD-ROM/DVD Drives, and Laser Printers or advanced computer graphics systems.

Due to the flexibility of the peer-to-peer communication, a digital camcorder can stream video and audio data to both a digital VCR and a DVD recorder at the same time via an IEEE 1394 network with no need for assistance from another network controller device.

Physical Layer

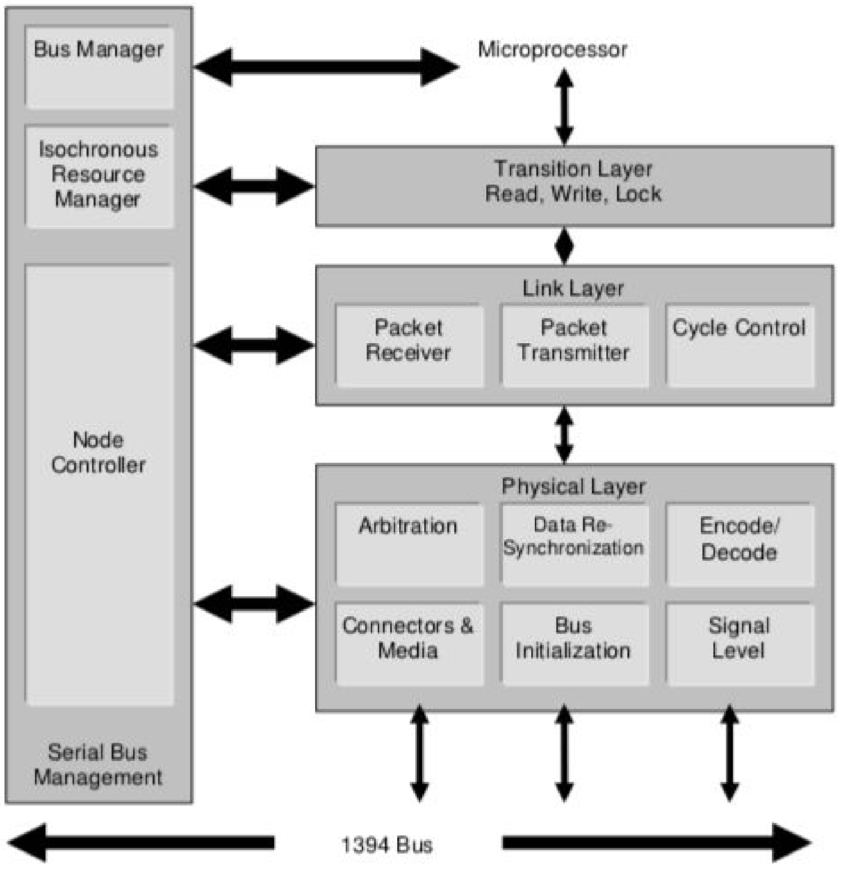


Figure from IEEE 1394 Architecture

This layer implements the electrical and mechanical interface required for transmission and reception of data bits (packets) across the serial bus.

Twisted pairs

The IEEE 1394 Data and Strobe signals are sent on cables with two separately shielded twisted-wire pairs, called TPA (twisted pair A) and TPB (twisted pair B). Transfers in one direction have the Strobe signal on TPA and the Data signal on TPB, while those in the opposite direction have the Data signal on TPA and the Strobe signal on TPB. [82]

For specifications on the connector, required voltages please reference to the datasheet.

PHY

The physical layer (the PHY) has three primary functions: transmission and reception of data bits, arbitration, and provision for the electrical and mechanical interface. The cable and backplane environments have different physical layers. The physical layer (PHY) translates the logical symbols used by the link layer into electrical signals using differential Data-Strobe (D-S) bit level encoding.

Link Layer

The link layer handles addressing, data checking and data framing for all packet transmission and reception, plus the provision of isochronous (same time) data transfer service directly to the application, including the generation of a cycle signal used for timing and synchronization.

Data Transmission

IEEE 1394 data packets have a 64-bit address header, which is divided into a 10-bit network address, a 6-bit node address and the remaining 48 bits for data (or mode control) memory addresses at the receiving node. This gives IEEE 1394 the ability to address 1023 networks of 63 nodes, each with up to 281TB (terabytes) of data addresses. [83]

All data is sent along the IEEE 1394 bus in serial four byte (32-bit) words, called quadlets. And the quadlets are encoded together with their clock signal onto two NRZ (non return to zero) bus signals, using a technique known as Data-Strobe (DS) coding. [84]

Data is transferred on the IEEE 1394 bus in addressed packets, and is transaction-based. The transfers can be asynchronous or isochronous.

Asynchronous Mode

Asynchronous transfers are used mainly for bus configuration, setting up transfers and handshaking, but are also used for bulk data transfer to and from hard disk drives. Provides guaranteed delivery, reliability is more important than timing and provides packet delivery service between the physical and transaction layers.

To transmit data in asynchronous mode, a 1394 device first requests control of the physical layer. Data are transmitted along with the sender's and the receiver's address. Once the receiver accepts the packet, a packet acknowledgment is sent to the sender by the receiver. To improve throughput, the sender may transmit up to 64 packets without waiting for the acknowledgment. If a negative acknowledgment is returned, error recovery follows.[85]

Asynchronous Packets

According to different modes, the packets transmitted are also different, here we will take a look at the Asynchronous packets.

- destination address – consisting of target bus, node, and device within that node
- source address – consisting of the source bus and node address
- transaction type – type of packet being sent
- transaction label – allows matching of requests and acknowledgements
- response code – part of a packet sent in response to a request for “completion status”.
- CRC – standard error-checking checksum for a request or response packet.

- acknowledge code – returned by the recipient of a packet to verify receipt
- acknowledge parity – a simple parity bit for the acknowledge code packet
- payload - the data we would like to transmit [86]

Isochronous Mode

Isochronous transfers are used for transporting time- sensitive data like digital video and audio. Provides guaranteed timing, late data is useless and there are no retries for bad or lost packets and the link layer performs packet delivery service directly between the physical layer and the application program interface (API).

In isochronous mode, the sender requests an isochronous channel with a specific bandwidth. Once the isochronous channel ID is granted, the sender sends the ID followed by packet data. The receiver monitors the data's channel ID and accepts data with the specified ID. Up to 64 (063) isochronous channels may be opened based upon user application requirements. The bus cycle master has an 8-kHz clock which generates cycles for data transfers . The duration for each cycle (also known as packet frame time) is 125µs. [87]

Isochronous Packets

Since isochronous packets do not require a response from or acknowledgements from the receiver, the packet format is simpler than the asynchronous packets that we have seen from above.

- channel number – isochronous senders and receivers use a shared channel number
- transaction type – identification code
- data
- CRC [88]

Bus Arbitration

When a node wishes to transmit a packet on the bus it needs to request permission to do so. The procedure for determining which node gets control of the bus is known as *arbitration*. There are two different kinds of arbitration depending on what kinds of nodes are on the bus: arbitration with only asynchronous devices and arbitration with isochronous devices.

Asynchronous-only arbitration is based on fair scheduling with priorities. After a 20µs gap (bus idle time) a new “arbitration round” starts. If a node wishes to transmit, it sends an arbitration request, which propagates up to the root node. If multiple nodes request arbitration at the same time, the node closest to the root node wins arbitration. After a 10µs gap the remaining nodes arbitrate for their turn to use the bus; this continues until all nodes that wish to use the bus during that round have used it. Following another 20µs gap a new round starts over. [89]

The arbitration mechanism gets more complicated when isochronous devices are connected on the bus, because those devices have already been guaranteed bandwidth. The root node broadcasts (sends a message to node 63, the broadcast address) a *cycle start packet*, which begins the arbitration round. All interested nodes send an arbitration request as before, and the winning node is still the node closest to the root. After .04µs of bus idle time, the remaining nodes arbitrate for control of the bus. [90]

Transaction layer

This layer supports the asynchronous protocol read, write, and lock commands. Using read, the originator (of the command) reads data from another node. Using write, the originator of the command sends data to the receiver (of the command). Lock combines the functions of the write and read commands by producing a data routing round trip between the sender and the receiver, which includes processing by the receiver.

The IEEE 1394 subsystem provides a callback mechanism for interaction with the bus. Two important data structures exist that allow users to register callback functions with the system to be called when certain events occur: hpsb_highlevel_ops and hpsb_address_ops. [91]

The first of these structures contains functions for host-level operations. These functions are called by the system when, for example, a host is added or removed, or when an isochronous packet is received. The second structure allows users to provide operations related to a specific address range. The functions provided include read, write, and lock. When a device on the bus attempts to read or write an address in the range specified for a particular hpsb_address_ops, the function in that structure is called by the system. [92]

Applications using Linux:

The IEEE 1394 subsystem in Linux provides a convenient interface to the bus in the form of the raw1394 module. The raw1394 module is both a *hpsb_highlevel_ops* and a character device. Being an *hpsb_highlevel_ops* allows the module to know about all hosts that exist on the system so that users can access all buses available on the system. As a character device, raw1394 provides the standard *fops* functions that allow reading, writing, etc. Reading and writing are accomplished in a slightly different manner than is traditionally used for character devices. There is a data structure, *raw1394_request*, which has data members for all important information regarding an IEEE 1394 transaction including type, address, and buffers for the actual data. An instance of this structure is passed to the read and write methods instead of actual buffers of data as is traditionally done. [93]

References

[1] Tanenbaum, 2012, Computer Networks 5 edition
[2] https://en.wikipedia.org/wiki/Ethernet
[3] DINTEK Electronic Limited, 2020,
https://medium.com/@dintekdci/difference-between-fiber-optic-cables-and-ethernet-cables-d0fc48e99ab9
[4] DINTEK Electronic Limited, 2020,
https://medium.com/@dintekdci/difference-between-fiber-optic-cables-and-ethernet-cables-d0fc48e99ab9
[5] LANTRONIX,
https://www.lantronix.com/resources/networking-tutorials/ethernet-tutorial-networking-basics/
[6] IEEE, 2012, https://standards.ieee.org/standard/802_3-2012.html
[7] Burg, Urs; Kenney, Martin, 2003, "Sponsors, Communities, and Standards: Ethernet vs. Token Ring in the Local Area Networking Business"
[8] Vangie Beal, https://www.webopedia.com/TERM/W/Wi_Fi.html
[9] Marshall Brain, https://computer.howstuffworks.com/wireless-network.htm
[10] IEEE Standard for Ethernet, 2015
[11] Bittel, R., "On Frame Check Sequence (FCS) Generation and Checking" ANSI working paper X3-S34-77-43, 1977
[12] The Ethernet, 1980, 6.2.4 https://ethernethistory.typepad.com/papers/EthernetSpec.pdf
[13] M. Cotton; L. Vegoda; R. Bonica, Special-Purpose IP Address Registries, 2013
[14] Understanding IP Addressing: Everything You Ever Wanted To Know, Chuck Semeria, 1996
[15] Wikipedia, 2020, https://en.wikipedia.org/wiki/IP_header
[16] Transmission Control Protocol (TCP),
https://www.sdxcentral.com/resources/glossary/transmission-control-protocol-tcp/
[17] Inter networking with TCP/IP: Principles, Protocols, and Architecture, Comer Douglas E., 2006
[18] Wikipedia, 2020, https://en.wikipedia.org/wiki/Transmission_Control_Protocol
[19] [20] [21] Tanenbaum, 2012, Computer Networks 5 edition
[22] W. Stevens, 1997, TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms
[23] Clark, M.P, 2003, Data Networks IP and the Internet, 1st
[24] RFC781, https://tools.ietf.org/html/rfc781
[25] Erik Nygren, Ramesh K. Sitaraman, Jennifer Sun,
The Akamai Network: A Platform for High-Performance Internet Applications
[26] RFC1035, https://tools.ietf.org/html/rfc1035
[27] https://www.cloudflare.com/learning/dns/what-is-dns/
[28][29] RFC2616, https://tools.ietf.org/html/rfc2616
[30][31] M. Bishop Ed, 2019, Hypertext Transfer Protocol Version 3 (HTTP/3)
https://tools.ietf.org/html/draft-ietf-quic-http-22
[32] SP5301, Universal Serial Bus, http://www.datasheet.es/PDF/495648/SP5301CN-pdf.html
[33] Garfinkel, 1999, USB deserves more support
https://simson.net/clips/1999/99.Globe.05-20.USB_deserves_more_support+.shtml
[34] Hachman, 2019, The new USB4 spec promises a lot: Thunderbolt 3 support, 40Gbps bandwidth, and less confusion
https://www.pcworld.com/article/3347403/the-new-usb4-spec-promises-a-lot-thunderbolt-3-support-40gbps-bandwidth-and-less-confusion.html
[35] Hruska, 2019, USB-C vs. USB 3.1: What's the difference?
http://www.extremetech.com/computing/197145-reversible-usb-type-c-finally-on-its-way-alongside-usb-3-1s-10gbit-performance
[36] Collins, 2013, The USB 3.0 Physical Layer
https://www.techdesignforums.com/practice/technique/usb-3-0-physical-layer/
[37] https://www.design-reuse.com/articles/15011/usb-2-0-phy-verification.html
[38] Wikipedia, 2020, https://en.wikipedia.org/wiki/PHY
[39] Anderson, Introduction to USB 3.0,
https://www.mindshare.com/files/resources/MindShare_Intro_to_USB_3.0.pdf

[40] Cavalcanti, USB 2.0 PHY Verification
https://www.design-reuse.com/articles/15011/usb-2-0-phy-verification.html
[41] https://www.techdesignforums.com/practice/technique/usb-3-0-link-layer/
[42] Collins, 2013, The USB 3.0 Physical Layer
https://www.techdesignforums.com/practice/technique/usb-3-0-physical-layer/
[43] Govindaraman, 2010,
https://www.electronicdesign.com/technologies/digital-ics/article/21790555/usb-30the-nextgeneration-interconnect
[44][45] FTDI chip, 2009,
https://www.ftdichip.com/Support/Documents/TechnicalNotes/TN_116_USB%20Data%20Structure.pdf
[46] Collins, 2013, https://www.techdesignforums.com/practice/technique/usb-3-0-protocol-layer-1
[47] Wikipedia, https://en.wikipedia.org/wiki/USB#cite_note-spec_3.0-19
[48] Richards, 2001, http://ww1.microchip.com/downloads/en/appnotes/00754.pdf
[49] Andrade; Hodel; Justo; Lagana; Santos; Gu; 2018, https://ieeexplore.ieee.org/document/8338047
[50] Vector,
https://web.archive.org/web/20160425131652/http://vector.com/vi_controller_area_network_en.html
[51]Bosch, 1991, CAN specification, http://esd.cs.ucr.edu/webres/can20.pdf
[52] Pico technology, https://www.picotech.com/library/oscilloscopes/can-bus-serial-protocol-decoding
[53][54][55]Kvaser, https://www.kvaser.com/can-protocol-tutorial/##bitTiming
[56] Bosch, 1991, CAN specification,http://esd.cs.ucr.edu/webres/can20.pdf
[57] Kvaser, https://www.kvaser.com/can-protocol-tutorial/
[58] Wikipedia, https://en.wikipedia.org/wiki/CAN_bus#cite_ref-3
[59] Bosch, 2003, CAN specification, http://esd.cs.ucr.edu/webres/can20.pdf
[60] Kvaser, https://www.kvaser.com/can-protocol-tutorial/
[61] Kvaser, https://www.kvaser.com/can-protocol-tutorial/##bitTiming
[62] Bosch, 1991, CAN specification, http://esd.cs.ucr.edu/webres/can20.pdf
[63] Kvaser, https://www.kvaser.com/can-protocol-tutorial/##bitTiming
[64] Bosch, 1991, CAN specification,http://esd.cs.ucr.edu/webres/can20.pdf
[65] Motorolla,
https://www.mi.fu-berlin.de/inf/groups/ag-tech/projects/ScatterWeb/moduleComponents/CanBus_canover.pdf
[66] Serial ATA, 2003, High Speed Serialized AT Attachment
[67] Martin, 2020, Digital Trends, https://www.digitaltrends.com/computing/what-is-sata/
[68] Wikipedia, 2020, https://en.wikipedia.org/wiki/Serial_ATA#cite_note-SATA1a-3
[69] Wikipedia, 2020, https://en.wikipedia.org/wiki/Serial_ATA#cite_note-SATA1a-3
[70] Serial ATA, 2003, High Speed Serialized AT Attachment
[71] Serial ATA, 2003, High Speed Serialized AT Attachment
[72] Serial ATA, 2003, High Speed Serialized AT Attachment
[73] Wikipedia, 2020, https://en.wikipedia.org/wiki/Serial_ATA#cite_note-SATA1a-3
[74] Wikipedia, 2020, https://en.wikipedia.org/wiki/ESATAp
[75] Amazon, 2020,
https://web.archive.org/web/20130802043631/http://www.amazon.com/gp/richpub/syltguides/fullview/RBX0KM9DMNFEJ
[76] Jansen, 2007, SATA-IO Advances Technology With The SATA Revision2.6,
https://sata-io.org/system/files/member-downloads/SATA-IOAdvancesTechnologyWithTheSATARevision2.6Spec.pdf
[77] Wikipedia, 2020,https://en.wikipedia.org/wiki/SATA_Express
[78] Wikipedia, 2020,https://en.wikipedia.org/wiki/M.2

[79] IEEE Standard for a High Performance Serial Bus

[80] https://www.imore.com/whats-difference-between-usb-c-and-thunderbolt
[81] http://www.differencebetween.net/technology/difference-between-firewire-and-thunderbolt/
[82] http://www.cse.unsw.edu.au/~cs2121/LectureNotes/08s2/firewire.pdf

[83] IEEE Standard for a High Performance Serial Bus

[84] IEEE Standard for a High Performance Serial Bus

[85] IEEE Standard for a High Performance Serial Bus

[86] IEEE Standard for a High Performance Serial Bus

[87] IEEE Standard for a High Performance Serial Bus

[88] IEEE Standard for a High Performance Serial Bus

[89] IEEE Standard for a High Performance Serial Bus

[90]Anderson, Don. FireWire System Architecture. Addison Wesley Longman Inc., 1998
https://web.archive.org/web/20120426040422/http://www.tindel.net/Firewire/firewire.html

[91] IEEE Standard for a High Performance Serial Bus

[92] IEEE Standard for a High Performance Serial Bus

[93] IEEE Standard for a High Performance Serial Bus

[92] Rubini, Alessandro. Linux Device Drivers. O'Reilly & Associates, Inc., 1998.