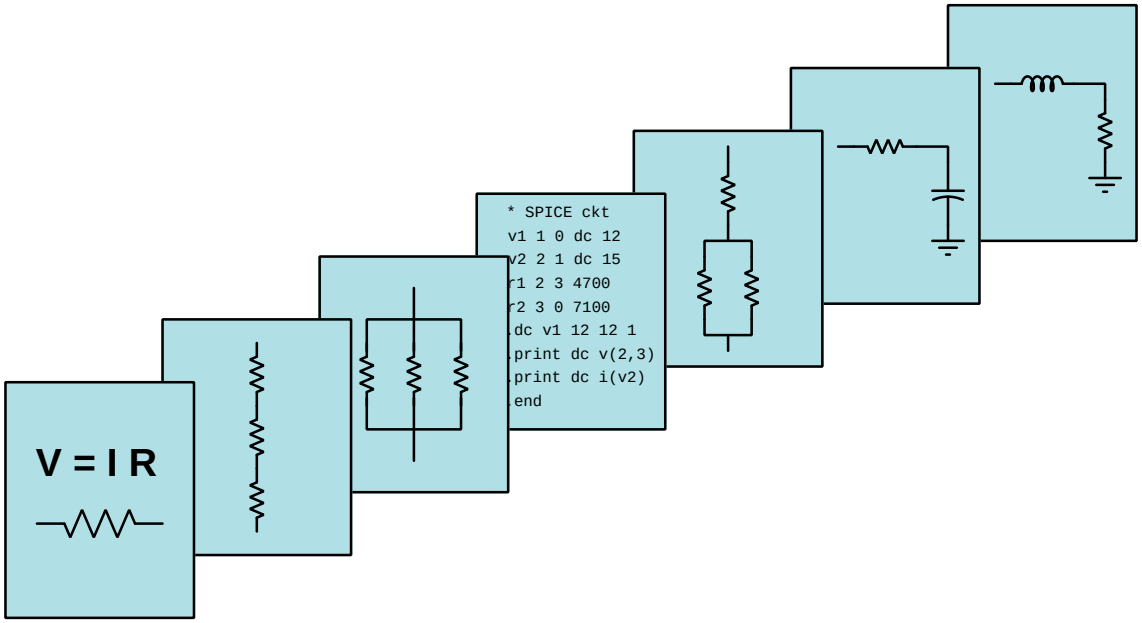
Modular Electronics Learning (ModEL) project



Ethernet Networks

© 2020-2026 by Tony R. Kuphaldt – under the terms and conditions of the Creative Commons Attribution 4.0 International Public License

Last update = 19 February 2026

This is a copyrighted work, but licensed under the Creative Commons Attribution 4.0 International Public License. A copy of this license is found in the last Appendix of this document. Alternatively, you may visit `http://creativecommons.org/licenses/by/4.0/` or send a letter to Creative Commons: 171 Second Street, Suite 300, San Francisco, California, 94105, USA. The terms and conditions of this license allow for free copying, distribution, and/or modification of all licensed works by the general public.

Contents

1 Introduction **3**
- 1.1 Recommendations for students . . . 3
- 1.2 Challenging concepts related to Ethernet networks . . . 5
- 1.3 Recommendations for instructors . . . 6

2 Tutorial **7**
- 2.1 Serial data communication and shift registers . . . 8
- 2.2 Ethernet overview . . . 12
- 2.3 Repeaters (hubs) . . . 13
- 2.4 Ethernet cabling . . . 16
- 2.5 Ethernet data frame format . . . 20
- 2.6 Switching hubs . . . 21
- 2.7 Error detection . . . 23

3 Derivations and Technical References **25**
- 3.1 The OSI Reference Model . . . 26
- 3.2 Hash algorithms . . . 28

4 Questions **31**
- 4.1 Conceptual reasoning . . . 35
  - 4.1.1 Reading outline and reflections . . . 36
  - 4.1.2 Foundational concepts . . . 37
  - 4.1.3 Bob Metcalfe's idea . . . 39
  - 4.1.4 Arbitration methods . . . 40
  - 4.1.5 Full-duplex Ethernet . . . 41
  - 4.1.6 Ethernet data frames . . . 42
  - 4.1.7 Power over Ethernet . . . 43
  - 4.1.8 RJ-45 connector circuitry . . . 44
- 4.2 Quantitative reasoning . . . 45
  - 4.2.1 Miscellaneous physical constants . . . 46
  - 4.2.2 Introduction to spreadsheets . . . 47
  - 4.2.3 Preamble time and distance . . . 50
  - 4.2.4 MAC address space . . . 50
  - 4.2.5 Ethernet collision probability . . . 51

4.3 Diagnostic reasoning . . . 53
4.3.1 Checking data using a hash function . . . 54

A Problem-Solving Strategies **55**

B Instructional philosophy **57**
B.1 First principles of learning . . . 58
B.2 Proven strategies for instructors . . . 59
B.3 Proven strategies for students . . . 61
B.4 Design of these learning modules . . . 62

C Tools used **65**

D Creative Commons License **69**

E References **77**

F Version history **79**

Index **80**

Chapter 1

Introduction

1.1 Recommendations for students

Ethernet was invented in the 1970's, but remains the dominant communications standard almost fifty years later for local-area-networks (LANs). It offers a high bandwidth, is easy for end-users to install and manage, and is embedded in most computing platforms both large and small. This module explores the basic principles of Ethernet technology and some of the common devices used within.

Important concepts related to ethernet include **bus arbitration**, **DTE** versus **DCE** devices, the **OSI Reference Model**, **transmission lines**, **terminating resistors**, data **collisions**, **differential** versus **single-ended** signals, data **frames**, **MAC addresses**, collision **domains**, **parity**, and **frame check sequences**.

Here are some good questions to ask of yourself while studying this subject:

- How might an experiment be designed and conducted to explore the relationship between network load and collision frequency? What hypothesis (i.e. prediction) might you pose for that experiment, and what result(s) would either support or disprove that hypothesis?
- What was innovative about Ethernet at the time of its invention?
- How do multiple Ethernet devices manage communication along a shared network cable without interfering with one another?
- What is the purpose of a hub?
- How do switches differ in operation from hubs?
- What cabling standards exist for Ethernet networks?
- How are signal reflections mitigated in Ethernet network cabling?
- What physical form(s) do Ethernet signals take?

- What is a "collision domain" and how do we manage these in Ethernet networks?
- How are errors detected in Ethernet transmissions, and how does this method compare with that used by EIA/TIA-232?
- How are different Ethernet devices distinguished from one another in the same network?

1.2 Challenging concepts related to Ethernet networks

The following list cites concepts related to this module's topic that are easily misunderstood, along with suggestions for properly understanding them:

- **Network arbitration** – the same is true for channel arbitration techniques such as CSMA/CD, master-slave, and others. It is helpful to imagine a set of devices all requiring access to a common channel of communication, and stepping through each of the protocols to see how they manage this access without having multiple devices "talk over" one another. This, like so many other things, simply takes time to digest and cannot be rushed.
- **Shift register modes** – the concepts of serial versus parallel data exchange for shift registers can be confusing, but experience has shown that likening these to the movement of packages on and off a conveyor belt is helpful in clarifying the concepts. In this analogy, the packages are data bits while the conveyor belt is the shift register.
- **Synchronous versus Asynchronous communication** – synchronous communication is when two or more digital devices communicate in lock-step with each other due to the simultaneous transmission of a clock signal. Asynchronous communication is when two or more digital devices use internal clocks to keep pace with each other, but rely on a "start" signal of some kind to synchronize themselves together at the start of each data frame. SPI is a synchronous communication standard.

1.3 Recommendations for instructors

This section lists realistic student learning outcomes supported by the content of the module as well as suggested means of assessing (measuring) student learning. The outcomes state what learners should be able to do, and the assessments are specific challenges to prove students have learned.

- **Outcome** – Demonstrate effective technical reading and writing

  Assessment – Students present their outlines of this module's instructional chapters (e.g. Case Tutorial, Tutorial, Historical References, etc.) ideally as an entry to a larger Journal document chronicling their learning. These outlines should exhibit good-faith effort at summarizing major concepts explained in the text.

  Assessment – Students present their "Conceptual Toolbox" Theory Journal entries listing important concepts applied in the Tutorial text(s). A "Conceptual Toolbox" serves as inspiration when solving problems, each concept suggesting a potential problem-solving action. Challenge students to collaboratively generate a Conceptual Toolbox for the day's topic during the class session and then use it when solving problems together, thus fostering the habit of reasoning from principles rather than recalling procedures!

- **Outcome** – Apply the concept of network channel arbitration

  Assessment – Explain how various types of network arbitration methods function; e.g. pose problems in the form of the "Arbitration methods" Conceptual Reasoning question.

- **Outcome** – Apply the concept of hash algorithms to data sets

  Assessment – Use hash algorithms to test two sets of plain-text data for identity; e.g. pose problems in the form of the "Checking data using a hash function" Diagnostic Reasoning question.

- **Outcome** – Independent research

  Assessment – Locate a user manual for common Ethernet hardware such as hubs and switches, and properly interpret some of the information contained in that document such as number of ports, management capabilities, etc.

Chapter 2

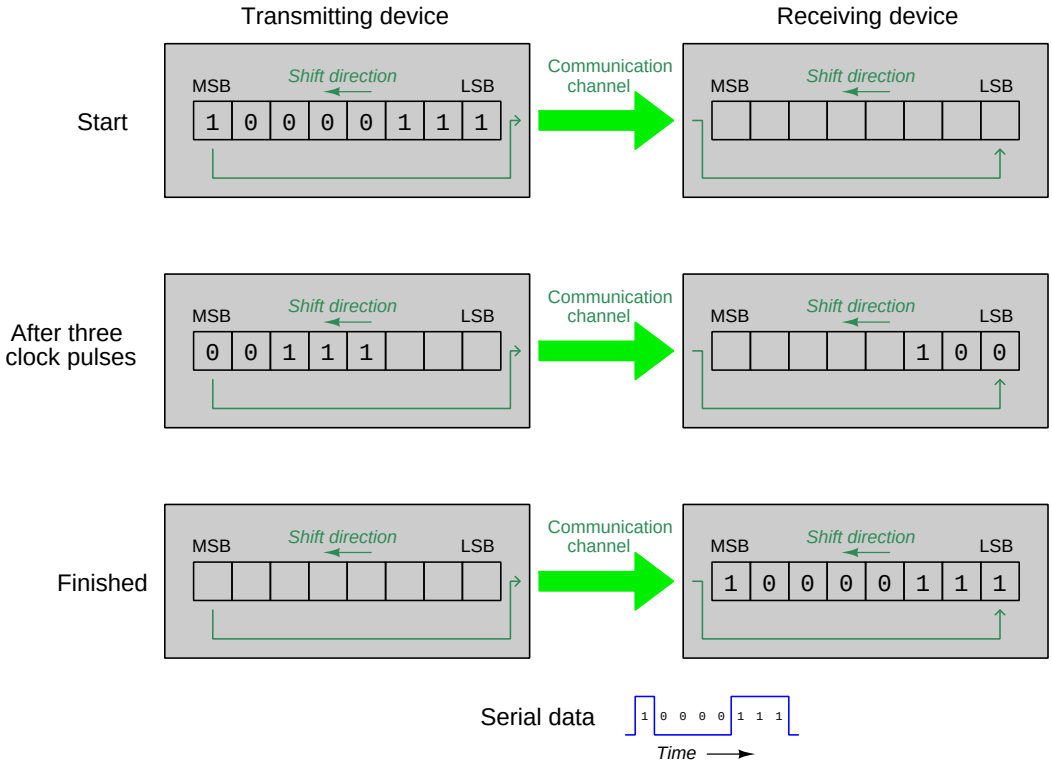
Tutorial

2.1 Serial data communication and shift registers

Digital data is comprised of *bits* of information, each bit being either a 1 or a 0, a high or a low, a true or a false. Collections of bits are generally known as *words*, although some specific numbers of them have unique names, such as a *byte* for an 8-bit word, or a *nybble* for a 4-bit word.

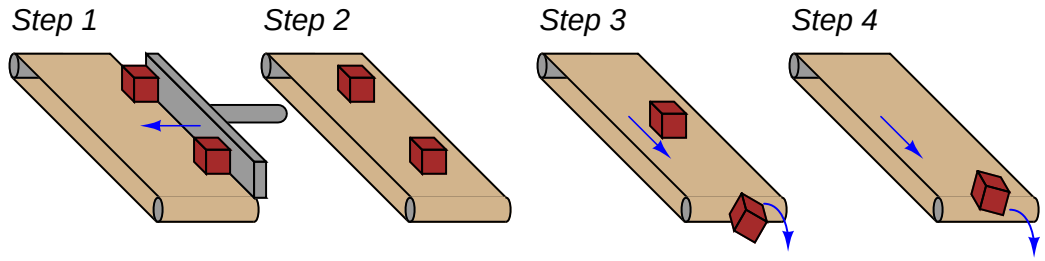
One way to communicate multi-bit digital words from one device to another is to do so one bit at a time over the same conductor(s), and this is known as *serial* communication because the bits are sent in a series over time rather than all at once using one conductor per bit (which is called *parallel* communication). Serial data communication is of course slower than parallel, but enjoys the decided advantage of using fewer conductors which is important in multiple contexts: for long-distance communication, fewer conductors means less expensive and less bulky cable; for communication between ICs on a circuit board, fewer conductors means fewer terminals (pins) necessary on the ICs to exchange that data which in turn means physically smaller devices are possible.

Multiple methods have been standardized for serial data communication, but fundamentally they all involve the use of digital devices known as *shift registers*. The following illustration shows a block diagram of a serial communication system where eight bits (one byte) are shifted out of the transmitting device's shift register, one bit at a time in sequence, and received at the receiving device's shift register to form a complete byte of data. This shifting of bits out of and in to the shift register circuits happens at the command of a signal called a *clock pulse*:

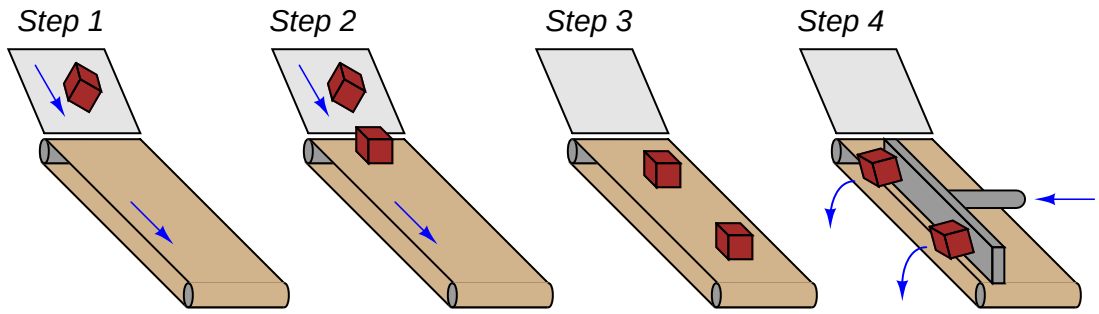


Shift registers are comprised of *flip-flops*, and are designed to move bits of data along from one flip-flop to another in sequence at the command of the clock pulse signal. The action of a shift register may be likened to a conveyor belt where packages move on to and off of the belt in various ways as shown below, the packages representing bits and the conveyor's motion representing the clock pulse:

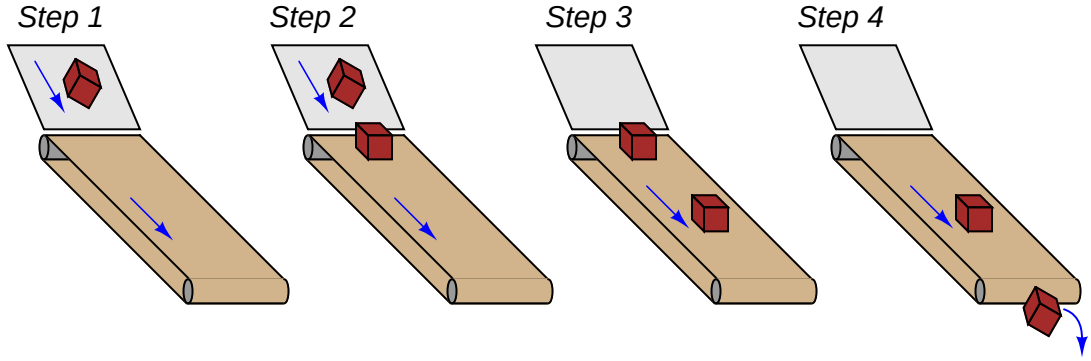
Parallel-in, serial-out (used by serial transmitting devices)



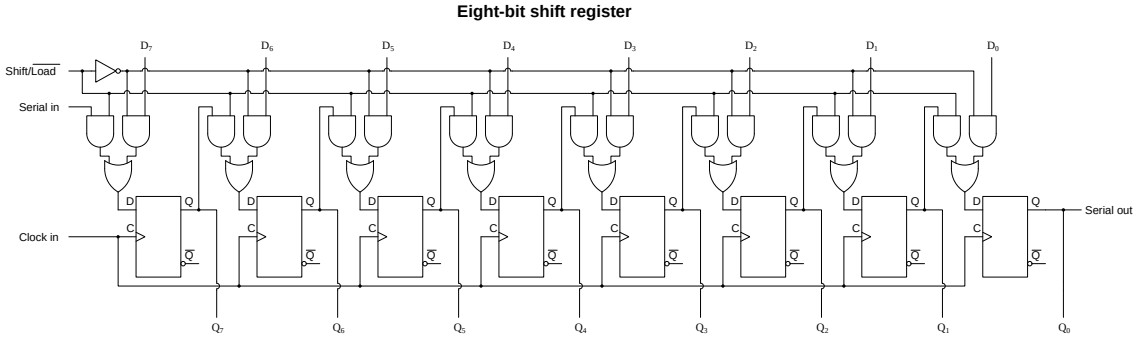
Serial-in, parallel-out (used by serial receiving devices)



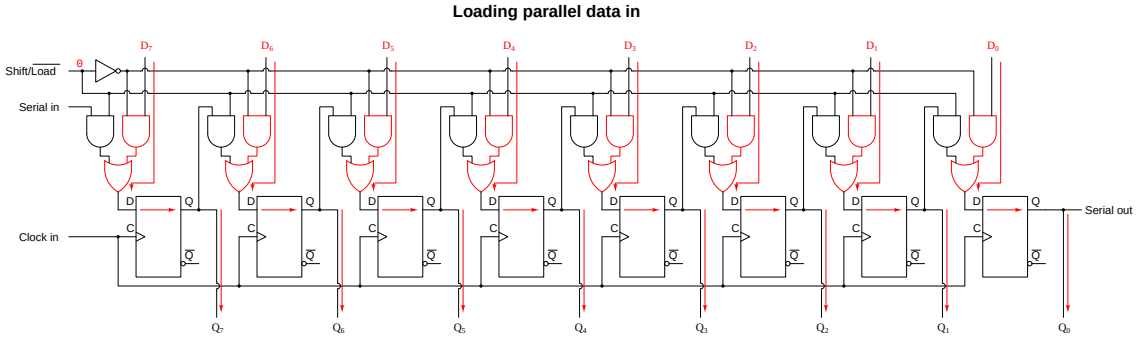
Serial-in, serial-out



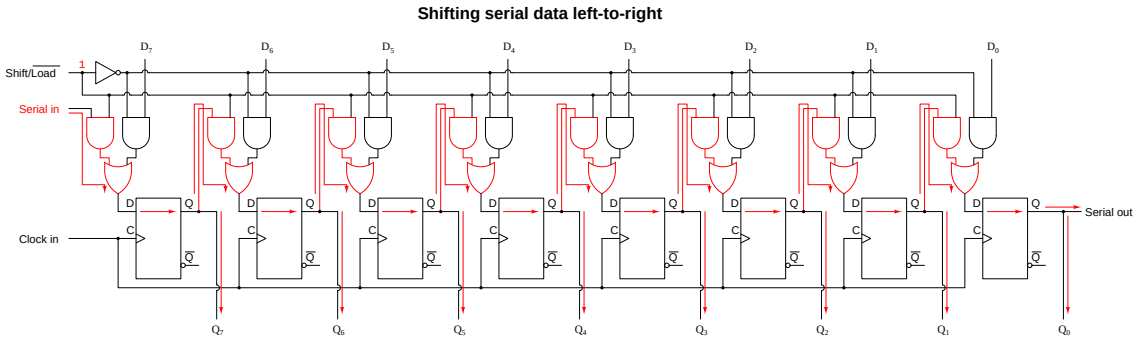
Shown below is a schematic diagram for an eight-bit (one-byte) shift register capable of all three modes of operation illustrated on the previous page. Lines D_0 through D_7 are for parallel data input, while lines Q_0 through Q_7 are for parallel data output. Serial data enters the shift register one bit at a time through the single "Serial in" line on the left, and exits the shift register one bit at a time through the single "Serial out" line on the right. A single "Clock" input receives a pulse signal causing parallel data to be fed into all the D-type flip-flops when the control line is in the Load state (i.e. "low"), and causing serial data to shift from left to right through all the D-type flip-flops when the control line is in the Shift state (i.e. "high"):



The AND/OR gate networks *steer* digital data to those flip-flops from different sources. In the "Load" mode of operation these steering gates route data from the parallel input terminals to the eight flip-flops as shown in red below:



In the "Shift" mode of operation these steering gates route data from one flip-flop stage to the next as shown in red below:



While all serial data communication systems employ shift registers at some level to convert between parallel and serial data formats, a range of different options distinguishes one standard from another. Some of this options include:

- The rate(s) at which the serial data may be communicated
- The manner in which clock pulses are synchronized between transmitter and receiver
- The exact voltage levels or other electrical characteristics of the serial data bits
- Properties of the communication channel connecting transmitter and receiver(s) together, for example the number of wires used and whether the signals are *single-ended* or *differential*

For example, a serial communication standard may specify certain allowable bit rates (i.e. bits per second), or let this be up to the end-user to decide. A serial communication standard may utilize a common clock pulse signal to drive both transmitter and receiver (called a *synchronous* network), or may use separate clock pulse generators at the transmitting and receiving ends which must be synchronized periodically to stay in-step with each other (called an *asynchronous* network). A serial communication standard may utilize common digital logic voltage levels to represent the 1 and 0 logical states of each bit, or it may use a much different set of voltage levels, or may even use some form of signal other than "high" and "low" voltage states to represent bits (e.g. using different audio-frequency tones). Many combinations exist on which to formulate a serial communications standard, which is why we have so many to choose from: EIA/TIA-232, EIA/TIA-485, SPI, I2C, 1-Wire, and Ethernet to name a few.

2.2 Ethernet overview

An engineer named Bob Metcalfe conceived the idea of Ethernet in 1973, while working for the Xerox research center in Palo Alto, California. His fundamental invention was the CSMA/CD method of channel arbitration, allowing multiple devices to share a common channel of communication while recovering gracefully from inevitable "collisions." In Metcalfe's vision, all of the "network intelligence" would be built directly into "controller" devices situated between the DTE devices (computers, terminals, printers, etc.) and a completely passive coaxial cable network. Unlike some other networks in operation at the time, Metcalfe's did not rely on additional devices to help coordinate communications between DTE devices. The coaxial cable linking DTE devices together would be completely passive and "dumb," performing no task but the conduction of broadcast signals between all devices. In that sense, it served the same purpose as the "luminiferous ether" once believed to fill empty space: conducting electromagnetic waves between separated points.

The CSMA/CD ("Carrier Sense Multiple Access with Collision Detection") method of bus arbitration works by giving each Ethernet device the ability to sense an idle channel as well as sense if it happens to "collide" with another transmitting device. In the event of a collision, the colliding devices both cease transmission, and set random time-delays to wait before re-transmission. The individual time delays are randomized to decrease the probability that a re-collision between the same devices will occur after the wait. This strategy is analogous to several peers in one group holding a conversation, where all people involved are equally free to begin speaking, and equally deferential to their peers if ever two or more accidently begin speaking at the same time. Occasional collisions are perfectly normal in an Ethernet network, and should not be taken as an indication of trouble unless their frequency becomes severe.

Metcalfe's original network design operated at a data rate of 2.94 Mbps, impressive for its time. By 1980, the three American computer companies DEC (Digital Equipment Corporation), Intel, and Xerox had collaborated to revise the Ethernet design to a speed of 10 Mbps, and released a standard called the *DIX Ethernet* standard (the acronym "DIX" representing the first letter of each company's name). Later, the IEEE Local and Metropolitan Networks Standards Committee codified the DIX Ethernet standard under the numeric label 802.3. At the present time there exist many "supplemental" standards underneath the basic 802.3 definition, a few of them listed here:

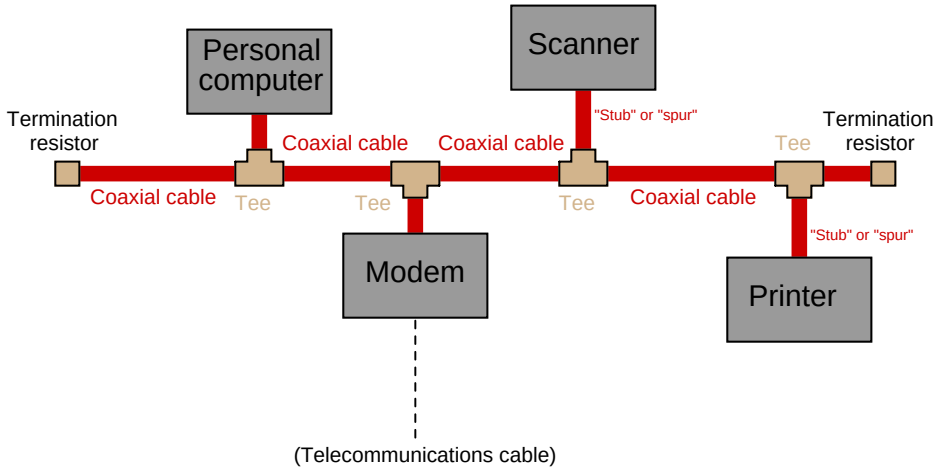
- 802.3a-1985 *10BASE2 "thin" Ethernet*
- 802.3d-1987 *FOIRL fiber-optic link*
- 802.3i-1990 *10BASE-T twisted-pair cable Ethernet*
- 802.3u-1995 *100BASE-T "Fast" Ethernet and Auto-Negotiation*
- 802.3x-1997 *Full-Duplex standard*
- 802.3ab-1999 *1000BASE-T "Gigabit" Ethernet over twisted-pair cable*

The IEEE 802.3 standard is limited to layers 1 and 2 of the OSI Reference Model: the "Physical" and "Data link" layers. In the physical layer (1), the various supplements describe all the different ways in which bits are electrically or optically represented, as well as permissible cable and connector types. In the data link layer (2), the IEEE standard describes how devices are addressed (each one with a unique identifier known as a *MAC address*, consisting of a 48-bit binary number usually

divided into six bytes, each byte written as a two-character hexadecimal number), the CSMA/CD channel arbitration protocol, and also how data frames are organized for Ethernet transmissions.

2.3 Repeaters (hubs)

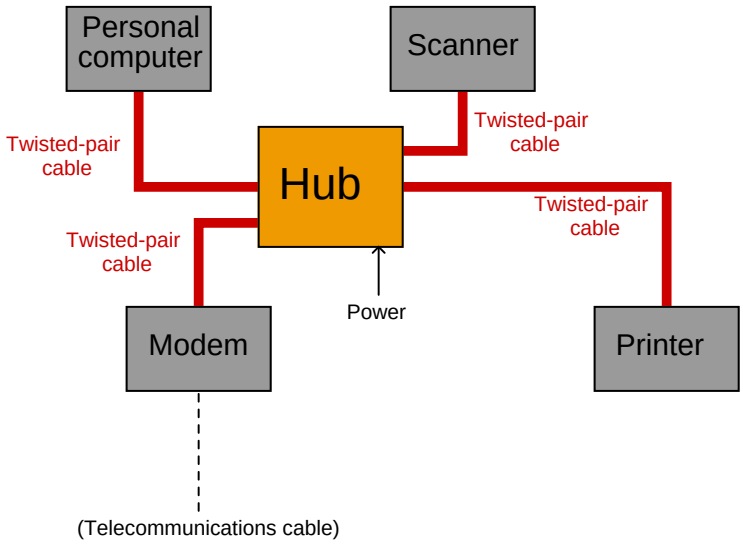
Bob Metcalfe's original design for Ethernet consisted of DTE devices connected to a common coaxial cable through the use of "tee" connectors, like this:



This cabling arrangement suffered several problems. First, it was inconvenient to run through an office building, since each DTE device needed to be coupled rather closely to the main "trunk." Short cable segments (called *stubs*, *spurs*, or *drops*) joining the main trunk line to each DTE device could not be too long, or else they would could cause multiple signal reflections to occur in the main line. Secondly, the signal strength decreased with each "tee" connector: every time the signal branched, it would lose power. Thirdly, the need for termination resistors – necessary at each end of any long cable conducting high-frequency signals to prevent debilitating signal "reflections" – posed the problem of those terminating resistors failing, falling off, or simply forgotten during installation or maintenance[1].

[1] These very same problems may arise in an industrial data network standard called FOUNDATION Fieldbus, and for the exact same reason: the cabling is passive (for increased reliability). This makes FOUNDATION Fieldbus instrument systems challenging to properly install for most applications (except in really simple cases where the cable route is straightforward), which in my mind is its single greatest weakness at the time of this writing (2009). I strongly suspect Ethernet's history will repeat itself in FOUNDATION Fieldbus at some later date: a system of reliable "hub" devices will be introduced so that these problems may be averted, and installations made much simpler.

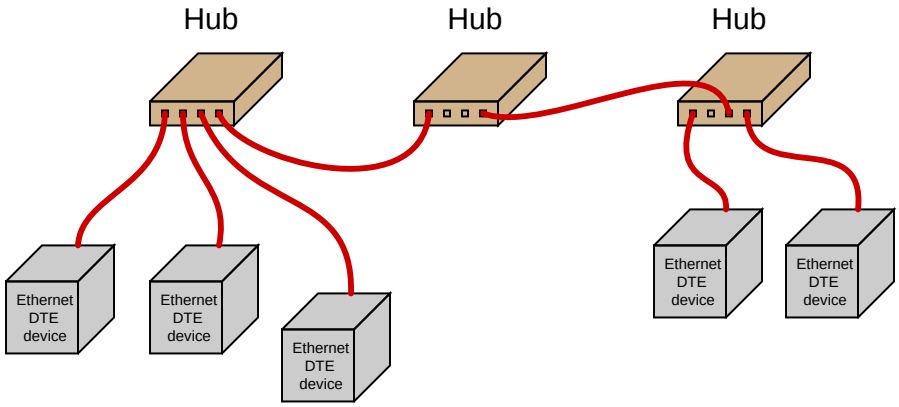
As Ethernet evolved as a practical networking standard, one of the many improvements added to its design was the concept of a *repeating hub*. A "repeater" is an active device designed to re-broadcast a signal, usually to overcome inevitable power losses incurred as that signal propagates along a cable. Repeaters are common in the telecommunications industry, where telephone, television, and computer signals must travel hundreds or thousands of miles between points of transmission and reception. A "repeating hub" is a repeater with multiple ports for many cables to plug into, where any signal entering on any cable is repeated to *all* ports on the device. Thus, a repeating hub (or simply "hub") allows multiple Ethernet devices to interconnect with no degradation in signal quality:



Not only do hubs improve system performance by boosting signals' voltage levels, but they also eliminate the need for termination resistors in the network. With a hub-based system, each and every cable terminates at either a DTE or DCE device, which is (now) designed with the proper termination resistance built-in to their internal transceiver circuitry. This means each and every Ethernet cable is automatically terminated with the proper impedance simply by plugging it in to the Ethernet port of *any* device. "Stub" or "spur" cables with their length restrictions are also a thing of the past, since no cable ever splits or branches in a hub-based network system.

Hubs are considered OSI "layer 1" devices, because they operate purely on the physical layer of Ethernet: all they do is receive Ethernet signals and re-broadcast those signals in boosted form to all other devices plugged into the hub. As a piece of interconnecting hardware, a hub is considered a DCE (Data Communications Equipment), as opposed to the end-of-cable devices such as computers and printers which are DTEs (Data Terminal Equipment).

Repeating hubs may be connected together to form larger networks[2]:

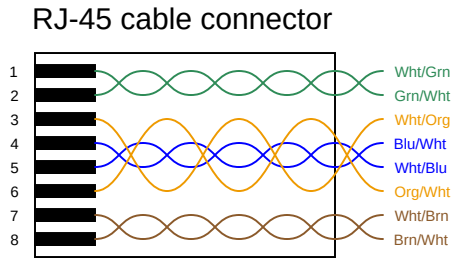


Since hubs are merely "layer 1" devices, mindlessly boosting and re-broadcasting signals received to their ports, their presence does not mitigate collisions between transmitting devices. As far as collisions between those devices is concerned, they might as well be directly connected together on a single piece of coaxial cable. One way to express this concept is to say that all portions of the network are part of the same *collision domain*. In other words, any devices on this network are able to collide with each other, because all transmissions are sensed by all the devices. This is analogous to a small room with several people in it: the room is small enough that everyone can hear everyone else talking, which means only one person in that room is able to speak at a time.

[2]There are practical limits as to how many hubs may be "daisy-chained" together in this manner, just as there are practical limits to how long a twisted-pair cable may be (up to 100 meters). If too many hubs are cascaded, the inevitable time delays caused by the process of repeating those electrical impulses will cause problems in the network. Also, I have neglected to specify the use of *crossover* cables to connect hubs to other hubs – this is a topic to be covered later in this book!

2.4 Ethernet cabling

Along with hubs came another form of Ethernet cable and connector: *unshielded, twisted pair* (UTP) wiring and *RJ-45* "flat" connectors. These cables use multiple twisted pairs of wires instead of the coaxial cable specified in Metcalfe's original Ethernet. The purpose of using twisted-wire pairs is to reduce magnetic signal coupling. In an effort to reduce electric signal coupling, modern Ethernet systems use *differential signaling* rather than ground-referenced (single-ended) signaling, which explains why a pair of conductors is used for each signal path (transmit and receive):



For 10 Mbps Ethernet over UTP cable (called 10BASE-T) and for 100 Mbps Ethernet (called 100BASE-TX), only two[3] out of four available wire pairs are used:

Pin number	Assignment	Abbreviation
1	Transmit Data (+)	TD+
2	Transmit Data (−)	TD−
3	Receive Data (+)	RD+
4	(not used)	
5	(not used)	
6	Receive Data (−)	RD−
7	(not used)	
8	(not used)	

[3]With only half the available wire pairs used in a standard 10 Mbps or 100 Mbps Ethernet cable, this opens the possibility of routing *two* Ethernet channels over a single four-pair UTP cable and RJ-45 connector. Although this is non-standard wiring, it may be a useful way to "squeeze" more use out of existing cables in certain applications. In fact, "splitter" devices are sold to allow two RJ-45-tipped cables to be plugged into a single RJ-45 socket such that one four-pair cable will then support two Ethernet pathways.

It should be noted that 1000 Mbps ("Gigabit") Ethernet over twisted-wire pairs does in fact use all four pairs in an eight-wire cable, a departure from traditional UTP Ethernet cable wiring:

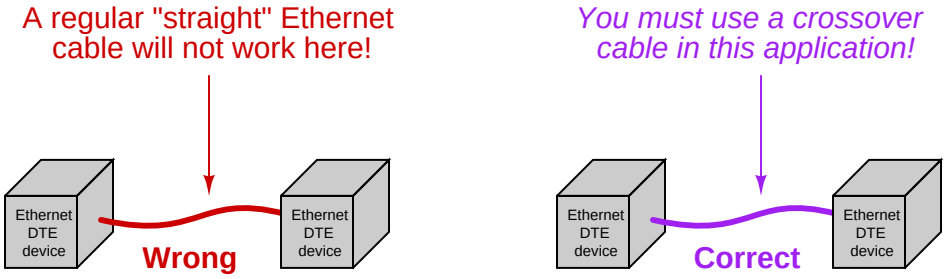
Pin number	Assignment	Abbreviation
1	Pair "A" (+)	BI_DA+
2	Pair "A" (−)	BI_DA−
3	Pair "B" (+)	BI_DB+
4	Pair "C" (+)	BI_DC+
5	Pair "C" (−)	BI_DC−
6	Pair "B" (−)	BI_DB−
7	Pair "D" (+)	BI_DD+
8	Pair "D" (−)	BI_DD−

Gigabit Ethernet also requires cabling built to handle higher-frequency signals in order to maintain signal integrity over long distances. Standard "Category 5" cable suitable for 10 Mbps and 100 Mbps Ethernet is not sufficient for 1 Gbps or higher Ethernet data rates, so instead we have "Category 5e" (the "e" standing for *enhanced*) which is the same cable design as Cat 5 cable but built to tighter tolerances. Beyond that, we have Category 6 cable built to have a thicker sheath and tighter twist rates of its wire-pairs, and Category 6a (*augmented*) adding shielding to protect against electric field interference.

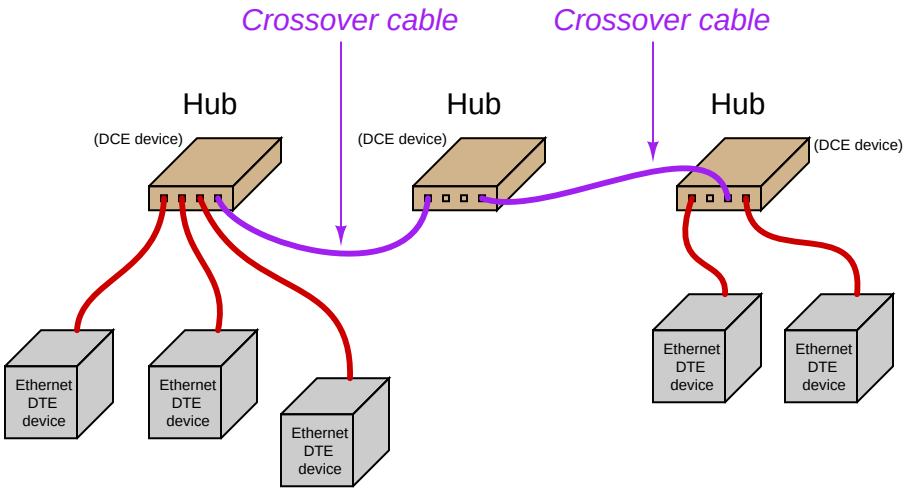
Along with twisted-pair cables and RJ-45 connectors came a significant alteration to the basic electrical scheme of Ethernet. Metcalfe's original design used a simple coaxial cable as the "ether" connecting devices together. Such cables had only two conductors, meaning each device needed to transmit *and* receive data over the same two conductors. With UTP cable's four pairs of conductors, transmission and reception of signals occurs over different wire pairs[4]. This means connections made between Ethernet devices must employ a "swap" between TD and RD wire pairs in order for communication to take place, so that the "receiver" circuitry of one device connects to the "transmitter" circuitry of the other, and vice-versa. This is precisely the same characteristic inherent to EIA/TIA-232 and four-wire EIA/TIA-485 networks, where separate wire pairs are dedicated to "transmit" and "receive" functions.

[4]This means modern Ethernet is capable of full-duplex communication between two devices, whereas the original coaxial-based Ethernet was only capable of half-duplex communication.

In a typical Ethernet system, the interconnecting hubs perform this transmit/receive swap. Hubs are considered DCE devices, while computers and other end-of-the-line devices are considered DTE devices. This means the pin assignments of DTE and DCE devices must be different in order to ensure the transmit/receive pin swap necessary for straight-through cables to work. This also means if someone ever wishes to directly connect two Ethernet DTE devices together without the benefit of a hub in between, a special *crossover* cable must be used for the connection, identical in function to the *null modem* cable used to connect two EIA/TIA-232 DTE devices together:



Furthermore, the same problem exists when multiple hubs are connected to form larger networks. Since each hub is a DCE device, a straight-through cable connecting two hubs together will pass transmitted signals from one hub directly to the "transmit" pins of the other hub, not the "receive" pins as it needs to. Consequently, a "crossover" cable should be used to connect two Ethernet hubs together in order to avoid this problem:



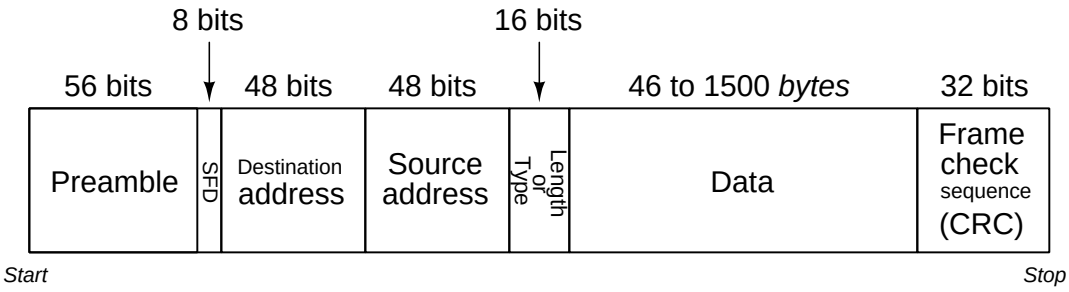
Some early Ethernet hubs provided a different solution to the "crossover" problem, and that was a crossover *switch* built into the hub, allowing a person to manually switch the transmit and receive wire pairs with the push of a button. In this next photograph of a four-port Ethernet hub, you can see the "Normal/Uplink" pushbutton on the right-hand side of the front panel, controlling the furthest-right port of the hub. This switch is supposed to be placed in the "Normal" position if the device plugged into that port is a DTE device, and placed in the "Uplink" position if the device is a DCE device (e.g. another hub):

Note the LED indicator lights by each port on the hub. One LED indicates whether or not the cable is active (when a powered Ethernet DTE device is plugged into that port of the hub), while the other LED indicates traffic on the cable (by blinking). These LEDs are very helpful for identifying a crossover problem. This hub even has an LED indicating the occurrence of collisions (the "Col" LED just below the main power LED), giving simple visual indication of collision frequency.

Newer Ethernet DTE and DCE devices use auto-sensing technology to perform any necessary transmit/receive pin swaps, rendering crossover cables and crossover pushbuttons unnecessary for either DTE-to-DTE or hub-to-hub connections. Auto-sensing is a standard feature of 1000BASE-T ("Gigabit" Ethernet).

2.5 Ethernet data frame format

Every Ethernet[5] frame consists of a pattern of bits following this format:

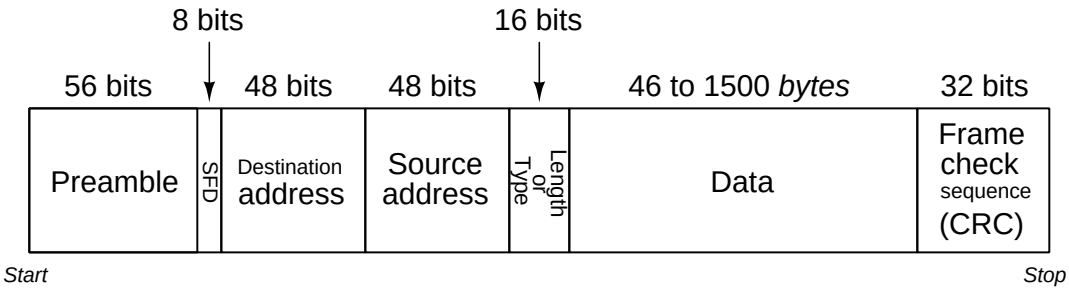


- **Preamble** – this is a string of alternating 1 and 0 bit states (i.e. `1010101010` . . .) intended to announce the transmission of a new Ethernet frame to all connected devices for the purpose of bus arbitration. It is during this period of time that simultaneously-transmitting devices will sense a "collision" and stop their transmissions. This alternating bit pattern also gives the *phase-locked loop (PLL)* circuitry in every Ethernet network interface time to "lock in" to the transmitted data rate, which is also how multi-rate devices "know" whether the bit rate will be 10 MBps, 100 Mbps, 1 GBps, or otherwise.
- **SFD** – this is the "Start Frame Delimiter" consisting of the single binary byte `10101011`. That last pair of repeated "1" bits announces the beginning of the rest of the data frame.
- **Destination address** – this is the 48-bit MAC address assigned to the Ethernet device that the message is destined for. Every Ethernet device in the world has a unique MAC address!
- **Source address** – this is the 48-bit MAC address assigned to the Ethernet device that the message is being sent from.
- **Type or Length** – this two-byte section declares the nature or size of the data payload shortly following. Values equal to 1500 or less specify the literal payload length in bytes. Values greater than 1500 signify a standardized *EtherType* code, declaring a payload following some well-known format attached to a particular communication protocol. The *IEEE Registration Authority* is responsible for assigning these EtherType codes, one for every standardized payload format. For example, EtherType `0x0800` (decimal value 2048) designates the payload as *Internet Protocol version 4* or *IPv4*.
- **Data** – this is the sequence of bits representing the actual message to be communicated. All else is "overhead" needed by Ethernet to do its job.
- **Frame Check Sequence** – this is four bytes (32 bits) representing a form of error-checking code used. It is similar in purpose to *parity bits* though much more sophisticated and more appropriate for data frames as large as Ethernet.

[5] Technically, this is the frame format for what is called *Ethernet II* or *DIX Ethernet* which is by far the most popular at the time of this writing (2026).

2.6 Switching hubs

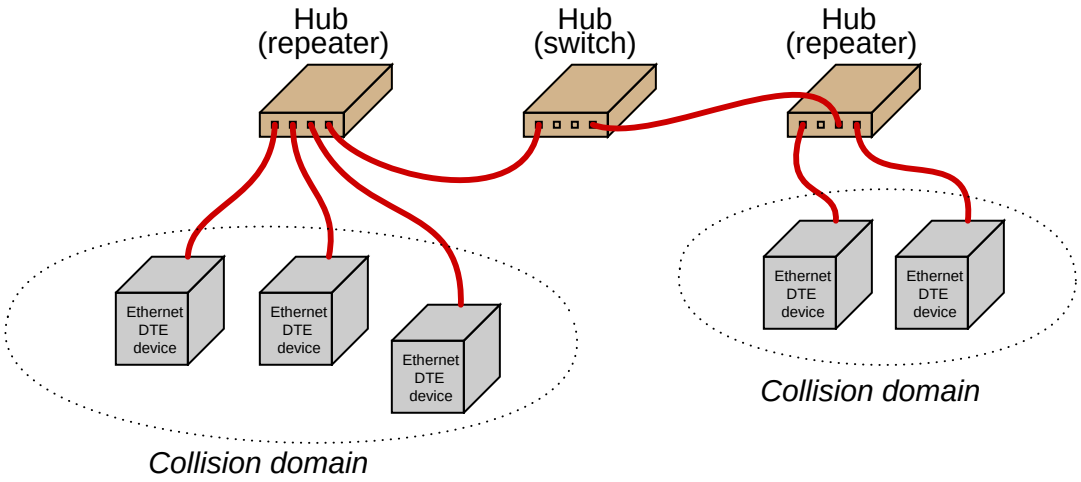
The next evolutionary step in Ethernet network connections was the introduction of a *switching hub*, or simply *switch*. A "switch" looks exactly like a repeating hub, but it contains intelligence to route transmitted signals only to specific ports, rather than broadcasting every received data frame to all ports. What enables this to happen is the information contained in each Ethernet frame transmitted by DTE devices:



Note that part of the frame includes both a source address and a destination address. These refer to the 48-bit "MAC" addresses uniquely identifying each and every Ethernet device. A switching hub "learns" the identities of all devices plugged into each of its ports by remembering the "source" addresses received through those ports. When a switch receives an Ethernet frame with a destination address it recognizes as residing on one of its ports, it *only* repeats that frame to that specific port, and not to the other ports. In other words, an Ethernet switch does not mindlessly broadcast all messages to all of its ports the way an Ethernet hub does. The switch's targeted direction of messages reduces the amount of "traffic" seen at the other ports, and also avoids unnecessary collisions because messages only get sent to their intended destinations.

If a switch receives a data frame with an unrecognized destination address, it defaults to basic "hub" behavior by broadcasting that frame to all ports. If a device plugged into one of that switch's ports replies to that data frame, the MAC address of that device is noted for future traffic direction to that port.

The presence of a switching hub in a larger network has the effect of dividing that network into separate collision domains, so that a collision occurring in one domain does not "spill over" into another domain where it would delay communication between those devices:



Of course, collisions between these two domains may still occur, for instance if a device in the first domain tries to transmit to a device in the second domain at the exact same time that a device in the second domain attempts to transmit to a device in the first.

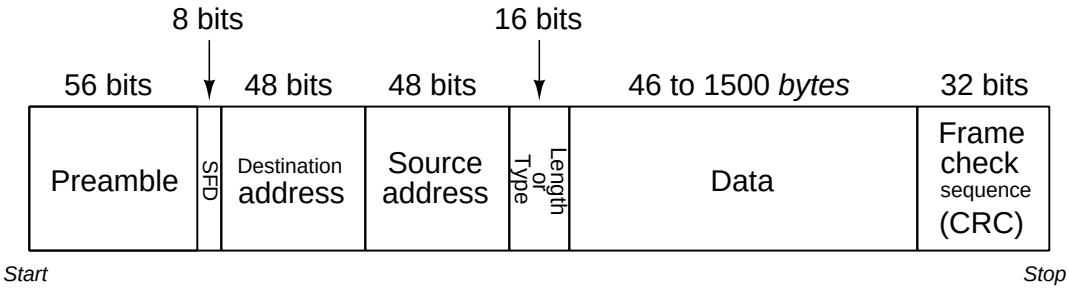
With this added intelligence, switching hubs are considered "layer 2" devices, since they operate not just at the physical layer of electrical impulses, but also at the next OSI model layer of device addressing. Since switching hubs add benefit beyond repeating hubs without any drawbacks[6], most people elect to use switches whenever possible.

[6] Even the cost difference is negligible. It should be noted, though, that switches may exhibit unintended behavior if a cable is unplugged from one of the ports and re-plugged into a different port. Since switches internally map ports to device addresses, swapping a device from one port to another will "confuse" the switch until it re-initializes the port identities. Re-initialization may be forced by cycling power to the switch, if the switch does not do so on its own.

2.7 Error detection

In simple serial networks a method for detecting errors is the *parity bit*: an extra bit added to the message frame based on the evenness or oddness of the "1" bits contained in the message. The principle is that a corruption in the data will cause the received parity to not match the transmitted parity – whether the corruption occurs in the data block or in the parity bit itself is irrelevant – and thereby warn the receiving device that a corruption has occurred. Parity-checking is not perfect, as it will fail to detect corruption if it occurs in any *even* number of bits, but it is better than nothing and fairly reliable when the message's data is only several bits in length.

Ethernet data frames typically carry data payloads consisting of *tens of thousands of bits*, which makes parity checking an unreliable[7] error-detection mechanism. Instead, the IEEE 802.3 standard uses a more sophisticated method called a *frame check sequence*. Like a parity bit, the content of the frame check sequence is generated by the transmitting device based on the data contained in the message, and the same algorithm is run by the receiving device to check to see that the result agrees with the received frame check sequence bits:



An Ethernet message's frame check sequence is a collection of bits mathematically calculated by the transmitting device based on the content of the data. The mathematical calculation is called a *hash algorithm*, and it always generates an output with a fixed number of bits (called a *hash code* or a *digest* or simply a *hash*). In the base of the IEEE 802.3 Ethernet standard, the hash algorithm employed is the 32-bit Cyclic Redundancy Check, or *CRC32*. Like a parity algorithm, the transmitting device processes the data bits to create the hash and then appends that hash to the end of the data field. The receiving device then takes the received data field and performs the exact same hash algorithm to see that its hash matches the one transmitted. As it is highly unlikely that any random corruption of bits during transmission would result in identical hash codes, this method is more robust for error-checking than parity.

Like parity, frame check sequences do not indicate *where* the errors lie, and also like parity they are imperfect. A chance always exists that just the right combination of errors may occur in transmission causing the frame check sequence values at both ends to match even though the data

[7] Consider the probability that any even number of bits (2, 4, 6, etc.) may become corrupted when the data contained in a serial message is only seven or eight bits. Now consider the probability that any even number of bits may become corrupted when the data is *thousands* of bits. Clearly, the odds are higher that two bits out of a thousand might become corrupted compared to two bits out of eight (i.e. a 0.2% corruption rate versus a 25% corruption rate, respectively)!

is not identical[8], but this is highly unlikely (calculated to be one chance in 10^{14} for Ethernet's 32-bit CRC). It is certainly better than having no error detection ability at all.

If the communications software in the receiving device is configured to take action on a detection of error, it may return a "request for re-transmission" to the transmitting device, so the corrupted message may be re-sent. This is analogous to a human being hearing a garbled transmission in a telephone conversation, and subsequently requesting the other person repeat what they just said.

[8] This is called a *hash collision*.

Chapter 3

Derivations and Technical References

3.1 The OSI Reference Model

Digital communication is a fairly complex topic, and there are a great many ways in which one digital device might communicate information to another. In order to help describe how different methods (protocols) function in forming a complete communication system, the *OSI Reference Model* was developed. This is not a standard itself, but rather a taxonomical system for *describing* and *classifying* various aspects of any communications system.

For example, one communication system might use discrete voltage pulses in a non-return-to-zero (NRZ) format to represent 0 and 1 bit states, while another might use analog audio tone frequency voltage signals to do the same. According to the OSI model, the choice over electrical signal types is a *Layer 1* matter. The patterns in which bit states make up larger serial data frames is another matter which must be standardized in any communication system, and according to the OSI model that is the domain of *Layer 2*. At the highest level of the OSI model we have *Layer 7* which regards the practical meaning of the bits communicated, whether those be alphanumeric codes representing an internet web page's contents, Modbus control functions, or anything else.

Layer 7 **Application**	This is where digital data takes on practical meaning in the context of some human or overall system function. *Examples: HTTP, FTP, HART, Modbus*
Layer 6 **Presentation**	This is where data gets converted between different formats. *Examples: ASCII, EBCDIC, MPEG, JPG, MP3*
Layer 5 **Session**	This is where "conversations" between digital devices are opened, closed, and otherwise managed for reliable data flow. *Examples: Sockets, NetBIOS*
Layer 4 **Transport**	This is where complete data transfer is handled, ensuring all data gets put together and error-checked before use. *Examples: TCP, UDP*
Layer 3 **Network**	This is where the system determines network-wide addresses, ensuring a means for data to get from one node to another. *Examples: IP, ARP*
Layer 2 **Data link**	This is where basic data transfer methods and sequences (frames) are defined within the smallest segment(s) of a network. *Examples: CSMA/CD, Token passing, Master/Slave*
Layer 1 **Physical**	This is where data bits are equated to electrical, optical, or other signals. Other physical details such as cable and connector types are also specified here. *Examples: EIA/TIA-232, 422, 485, Bell 202*

Students of philosophy might see a parallel between the OSI model's description of digital communication versus the ancient Greek philosopher Aristotle's classification of *causes*. Aristotle proposed four different ways to describe what causes a thing to exist or to occur. I have arranged these four causes in an order analogous to the OSI model's layers:

Aristotle's Four Causes

Layer 4 **Final**	This is the end-purpose for which something exists or has happened.
Layer 3 **Efficient**	This is the means by which something came into existence or occurred.
Layer 2 **Formal**	This is the structure or form taken by something.
Layer 1 **Material**	This is the fundamental substance something is made of.

For example, we might classify the four causes of a textbook as follows:

- **Final cause** – to educate and enlighten the reader
- **Efficient cause** – a human author wrote the book, and it came into existence through the operation of a printing press
- **Formal cause** – a series of characters and images arranged into words, paragraphs, diagrams, graphs, equations, etc. following one or more standards of language (e.g. English, international math notation, Cartesian-coordinate graphs)
- **Material cause** – paper and ink

All of these causes describe different aspects of what was required for a book to come into existence. Similarly, with the OSI model, each level of the model is like one of Aristotle's causes in that it describes something necessary for the communication to take place.

3.2 Hash algorithms

A *hash algorithm* is a mathematical/logical procedure by which a large set of data generates a result of fixed-bit-width that is fairly unique. We see hash algorithms applied as error-checking mechanisms in such communication standards as IEEE 802.3 Ethernet.

Interestingly, hash algorithms are applicable to more than just transmitted data in a digital network. They may also be used to validate the integrity of *any* collection of digital data, including *files* existing within a computer's memory or stored on a digital drive. Comparisons of hash codes for such data would then reveal tampering or corruption of that data[1].

To demonstrate, we will use the same frame check sequence algorithm used by Ethernet (called *CRC32*) to process a dataset consisting of ASCII text characters from the following passage taken from Henry David Thoreau's book *Walden*:

> If I wished a boy to know something about the arts and sciences, for instance, I would not pursue the common course, which is merely to send him into the neighborhood of some professor, where anything is professed and practised but the art of life; – to survey the world through a telescope or a microscope, and never with his natural eye; to study chemistry, and not learn how his bread is made, or mechanics, and not learn how it is earned; to discover new satellites to Neptune, and not detect the motes in his eyes, or to what vagabond he is a satellite himself; or to be devoured by the monsters that swarm all around him, while contemplating the monsters in a drop of vinegar. Which would have advanced the most at the end of a month – the boy who had made his own jackknife from the ore which he had dug and smelted, reading as much as would be necessary for this – or the boy who had attended the lectures on metallurgy at the Institute in the meanwhile, and had received a Rodgers' penknife from his father? Which would be most likely to cut his fingers?... To my astonishment I was informed on leaving college that I had studied navigation! – why, if I had taken one turn down the harbor I should have known more about it. Even the poor student studies and is taught only political economy, while that economy of living which is synonymous with philosophy is not even sincerely professed in our colleges. The consequence is, that while he is reading Adam Smith, Ricardo, and Say, he runs his father in debt irretrievably.

[1] I once used an industrial control system that relied on UV-erasable EPROM memory ICs to store the code defining the system's functions. These EPROM chips were susceptible to data loss by accidental exposure to light, and so as a precautionary measure we used the programming device to generate a hash (which we called a "checksum") and hand-wrote that hash code on the piece of tape we used to cover up the glass UV-exposure window on the IC. If ever there was doubt that a particular EPROM chip had become corrupted, we could place that chip into the programmer machine's DIP socket and re-run the hash algorithm to see if the hash code it generated still matched what was written on the tape. A mis-match indicated data corruption.

Applying the CRC32 hash algorithm to this text[2] generates the following digest:

```
a1c544e8
```

Now, to demonstrate the sensitivity of the hash algorithm to even the smallest change, consider the result when we alter a single character from Thoreau's text (changing the question-mark symbol following the word *fingers* into a blank space character, this substitution representing a single-bit difference between the ASCII characters: `0100000` for the space character versus `0100001` for the exclamation point) and re-run the CRC32 hash algorithm on that edited text:

```
9719f081
```

Of course, these disparate hash codes tell us nothing about the location or extent of the data corruption, only that *some* corruption occurred and a message re-transmission is necessary.

[2]Note that CRC32 is not the only hash algorithm in existence – it just happens to be simple and fast to compute which is why it was chosen as the frame check sequence algorithm for Ethernet. The relatively high speed of Ethernet serial communication requires a hash algorithm that is simple enough to be executed for every single Ethernet frame by end-devices having limited computing power. An example of a more robust hash algorithm is *SHA1*, which is simple to test on any computer with a command-line interface and the necessary SHA1 application installed. Simply save the data to a file (e.g. `walden.txt`) and then type the command: `sha1sum walden.txt` to see the hash (digest) printed to the console. For the *Walden* passage example, the original SHA1 digest is `d48055e2065e9324fefb0ea54747d0990e855041` and the SHA1 digest for the corrupted passage is `14f0df7536e301978148d6d46ecf8cff7c6edb35` which as you can see is quite different.

Chapter 4

Questions

This learning module, along with all others in the ModEL collection, is designed to be used in an inverted instructional environment where students independently read[1] the tutorials and attempt to answer questions on their own *prior* to the instructor's interaction with them. In place of lecture[2], the instructor engages with students in Socratic-style dialogue, probing and challenging their understanding of the subject matter through inquiry.

Answers are not provided for questions within this chapter, and this is by design. Solved problems may be found in the Tutorial and Derivation chapters, instead. The goal here is *independence*, and this requires students to be challenged in ways where others cannot think for them. Remember that you always have the tools of *experimentation* and *computer simulation* (e.g. SPICE) to explore concepts!

The following lists contain ideas for Socratic-style questions and challenges. Upon inspection, one will notice a strong theme of *metacognition* within these statements: they are designed to foster a regular habit of examining one's own thoughts as a means toward clearer thinking. As such these sample questions are useful both for instructor-led discussions as well as for self-study.

[1] Technical reading is an essential academic skill for any technical practitioner to possess for the simple reason that the most comprehensive, accurate, and useful information to be found for developing technical competence is in textual form. Technical careers in general are characterized by the need for continuous learning to remain current with standards and technology, and therefore any technical practitioner who cannot read well is handicapped in their professional development. An excellent resource for educators on improving students' reading prowess through intentional effort and strategy is the book textitReading For Understanding – How Reading Apprenticeship Improves Disciplinary Learning in Secondary and College Classrooms by Ruth Schoenbach, Cynthia Greenleaf, and Lynn Murphy.

[2] Lecture is popular as a teaching method because it is easy to implement: any reasonably articulate subject matter expert can talk to students, even with little preparation. However, it is also quite problematic. A good lecture always makes complicated concepts seem easier than they are, which is bad for students because it instills a false sense of confidence in their own understanding; reading and re-articulation requires more cognitive effort and serves to verify comprehension. A culture of teaching-by-lecture fosters a debilitating dependence upon direct personal instruction, whereas the challenges of modern life demand independent and critical thought made possible only by gathering information and perspectives from afar. Information presented in a lecture is ephemeral, easily lost to failures of memory and dictation; text is forever, and may be referenced at any time.

General challenges following tutorial reading

- Summarize as much of the text as you can in one paragraph of your own words. A helpful strategy is to explain ideas as you would for an intelligent child: as simple as you can without compromising too much accuracy.
- Simplify a particular section of the text, for example a paragraph or even a single sentence, so as to capture the same fundamental idea in fewer words.
- Where did the text make the most sense to you? What was it about the text's presentation that made it clear?
- Identify where it might be easy for someone to misunderstand the text, and explain why you think it could be confusing.
- Identify any new concept(s) presented in the text, and explain in your own words.
- Identify any familiar concept(s) such as physical laws or principles applied or referenced in the text.
- Devise a proof of concept experiment demonstrating an important principle, physical law, or technical innovation represented in the text.
- Devise an experiment to disprove a plausible misconception.
- Did the text reveal any misconceptions you might have harbored? If so, describe the misconception(s) and the reason(s) why you now know them to be incorrect.
- Describe any useful problem-solving strategies applied in the text.
- Devise a question of your own to challenge a reader's comprehension of the text.

General follow-up challenges for assigned problems

- Identify where any fundamental laws or principles apply to the solution of this problem, especially before applying any mathematical techniques.
- Devise a thought experiment to explore the characteristics of the problem scenario, applying known laws and principles to mentally model its behavior.
- Describe in detail your own strategy for solving this problem. How did you identify and organized the given information? Did you sketch any diagrams to help frame the problem?
- Is there more than one way to solve this problem? Which method seems best to you?
- Show the work you did in solving this problem, even if the solution is incomplete or incorrect.
- What would you say was the most challenging part of this problem, and why was it so?
- Was any important information missing from the problem which you had to research or recall?
- Was there any extraneous information presented within this problem? If so, what was it and why did it not matter?
- Examine someone else's solution to identify where they applied fundamental laws or principles.
- Simplify the problem from its given form and show how to solve this simpler version of it. Examples include eliminating certain variables or conditions, altering values to simpler (usually whole) numbers, applying a limiting case (i.e. altering a variable to some extreme or ultimate value).
- For quantitative problems, identify the real-world meaning of all intermediate calculations: their units of measurement, where they fit into the scenario at hand. Annotate any diagrams or illustrations with these calculated values.
- For quantitative problems, try approaching it qualitatively instead, thinking in terms of "increase" and "decrease" rather than definite values.
- For qualitative problems, try approaching it quantitatively instead, proposing simple numerical values for the variables.
- Were there any assumptions you made while solving this problem? Would your solution change if one of those assumptions were altered?
- Identify where it would be easy for someone to go astray in attempting to solve this problem.
- Formulate your own problem based on what you learned solving this one.

General follow-up challenges for experiments or projects

- In what way(s) was this experiment or project easy to complete?
- Identify some of the challenges you faced in completing this experiment or project.

- Show how thorough documentation assisted in the completion of this experiment or project.
- Which fundamental laws or principles are key to this system's function?
- Identify any way(s) in which one might obtain false or otherwise misleading measurements from test equipment in this system.
- What will happen if (component X) fails (open/shorted/etc.)?
- What would have to occur to make this system unsafe?

4.1 Conceptual reasoning

These questions are designed to stimulate your analytic and synthetic thinking[3]. In a Socratic discussion with your instructor, the goal is for these questions to prompt an extended dialogue where assumptions are revealed, conclusions are tested, and understanding is sharpened. Your instructor may also pose additional questions based on those assigned, in order to further probe and refine your conceptual understanding.

Questions that follow are presented to challenge and probe your understanding of various concepts presented in the tutorial. These questions are intended to serve as a guide for the Socratic dialogue between yourself and the instructor. Your instructor's task is to ensure you have a sound grasp of these concepts, and the questions contained in this document are merely a means to this end. Your instructor may, at his or her discretion, alter or substitute questions for the benefit of tailoring the discussion to each student's needs. The only absolute requirement is that each student is challenged and assessed at a level equal to or greater than that represented by the documented questions.

It is far more important that you convey your *reasoning* than it is to simply convey a correct answer. For this reason, you should refrain from researching other information sources to answer questions. What matters here is that *you* are doing the thinking. If the answer is incorrect, your instructor will work with you to correct it through proper reasoning. A correct answer without an adequate explanation of how you derived that answer is unacceptable, as it does not aid the learning or assessment process.

You will note a conspicuous lack of answers given for these conceptual questions. Unlike standard textbooks where answers to every other question are given somewhere toward the back of the book, here in these learning modules students must rely on other means to check their work. The best way by far is to debate the answers with fellow students and also with the instructor during the Socratic dialogue sessions intended to be used with these learning modules. Reasoning through challenging questions with other people is an excellent tool for developing strong reasoning skills.

Another means of checking your conceptual answers, where applicable, is to use circuit simulation software to explore the effects of changes made to circuits. For example, if one of these conceptual questions challenges you to predict the effects of altering some component parameter in a circuit, you may check the validity of your work by simulating that same parameter change within software and seeing if the results agree.

[3] *Analytical* thinking involves the "disassembly" of an idea into its constituent parts, analogous to dissection. *Synthetic* thinking involves the "assembly" of a new idea comprised of multiple concepts, analogous to construction. Both activities are high-level cognitive skills, extremely important for effective problem-solving, necessitating frequent challenge and regular practice to fully develop.

4.1.1 Reading outline and reflections

"Reading maketh a full man; conference a ready man; and writing an exact man" – Francis Bacon

Francis Bacon's advice is a blueprint for effective education: reading provides the learner with knowledge, writing focuses the learner's thoughts, and critical dialogue equips the learner to confidently communicate and apply their learning. Independent acquisition and application of knowledge is a powerful skill, well worth the effort to cultivate. To this end, students should read these educational resources closely, journal their own reflections on the reading, and discuss in detail their findings with classmates and instructor(s). You should be able to do all of the following after reading any instructional text:

☑ Briefly SUMMARIZE THE TEXT in the form of a journal entry documenting your learning as you progress through the course of study. Share this summary in dialogue with your classmates and instructor. Journaling is an excellent self-test of thorough reading because you cannot clearly express what you have not read or did not comprehend.

☑ Demonstrate ACTIVE READING STRATEGIES, including verbalizing your impressions as you read, simplifying long passages to convey the same ideas using fewer words, annotating text and illustrations with your own interpretations, working through mathematical examples shown in the text, cross-referencing passages with relevant illustrations and/or other passages, identifying problem-solving strategies applied by the author, etc. Technical reading is a special case of problem-solving, and so these strategies work precisely because they help solve any problem: paying attention to your own thoughts (metacognition), eliminating unnecessary complexities, identifying what makes sense, paying close attention to details, drawing connections between separated facts, and noting the successful strategies of others.

☑ Identify IMPORTANT THEMES, especially GENERAL LAWS and PRINCIPLES, expounded in the text and express them in the simplest of terms as though you were teaching an intelligent child. This emphasizes connections between related topics and develops your ability to communicate complex ideas to anyone.

☑ Form YOUR OWN QUESTIONS based on the reading, and then pose them to your instructor and classmates for their consideration. Anticipate both correct and incorrect answers, the incorrect answer(s) assuming one or more plausible misconceptions. This helps you view the subject from different perspectives to grasp it more fully.

☑ Devise EXPERIMENTS to test claims presented in the reading, or to disprove misconceptions. Predict possible outcomes of these experiments, and evaluate their meanings: what result(s) would confirm, and what would constitute disproof? Running mental simulations and evaluating results is essential to scientific and diagnostic reasoning.

☑ Specifically identify any points you found CONFUSING. The reason for doing this is to help diagnose misconceptions and overcome barriers to learning.

4.1.2 Foundational concepts

Correct analysis and diagnosis of electric circuits begins with a proper understanding of some basic concepts. The following is a list of some important concepts referenced in this module's full tutorial. Define each of them in your own words, and be prepared to illustrate each of these concepts with a description of a practical example and/or a live demonstration.

DTE

DCE

CSMA arbitration

MAC address

Transmission line

Signal reflection

Repeater

OSI model

Collision

Collision domain

Coupling, magnetic

Single-ended voltage signal

Differential voltage signal

Null modem

Auto-sensing

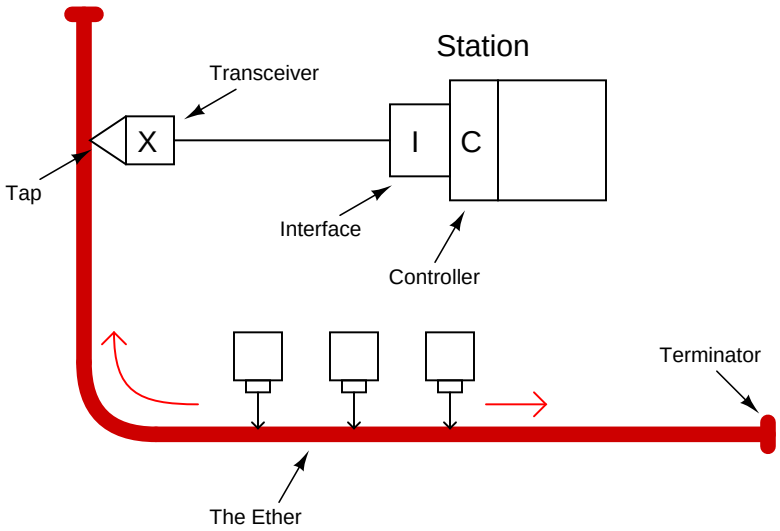
Switching hub

Frame check sequence

Hash

4.1.3 Bob Metcalfe's idea

In 1976, an engineer named Bob Metcalfe designed a new type of digital communication standard he dubbed *Ethernet*. A sketch he drew of his new system looked like this:



Explain the basic concept behind Metcalfe's Ethernet system, and why he chose the word "ether" to name it. Also, identify the bit rate (speed) at which his original Ethernet communicated at.

Challenges

- Identify some disadvantages of the original coaxial-based Ethernet versus modern twisted-pair (with hubs) Ethernet networks.

4.1.4 Arbitration methods

When multiple digital devices "talk" together on a common network, they must have some way of arbitrating who gets to talk, and in what order. Otherwise there will be data loss as multiple devices inevitably attempt to transmit at the same time.

A variety of protocols exist to handle this problem. A few are listed here:

- Master/slave
- Token passing
- TDMA (*Time Division Multiple Access*)
- CSMA/CD (*Carrier Sense Multiple Access / Collision Detect*)
- CSMA/BA (*Carrier Sense Multiple Access / Bitwise Arbitration*)
- CSMA/CA (*Carrier Sense Multiple Access / Collision Avoidance*)

Explain how each of these protocols works, and identify which one is used in the *Ethernet* communication standard (IEEE 802.3).

Challenges

- *Wireless* communications such as WLAN (IEEE 802.11) absolutely *cannot* use CSMA/CD protocol. Explain why this is.
- Explain what "jabbering" is, in your own words, and what effect it will have on networks using each of these five different protocols.
- Identify ways to identify a "jabbering" problem occurring in a network.

4.1.5 Full-duplex Ethernet

What do the phrases *full-duplex* and *half-duplex* mean in relation to copper-based (i.e. connected by wire cables) serial data networks?

What does the term *determinism* mean with regard to serial data networks?

Why is determinism more of a factor in networked industrial control systems than it is for general information-technology systems such as residential internet service?

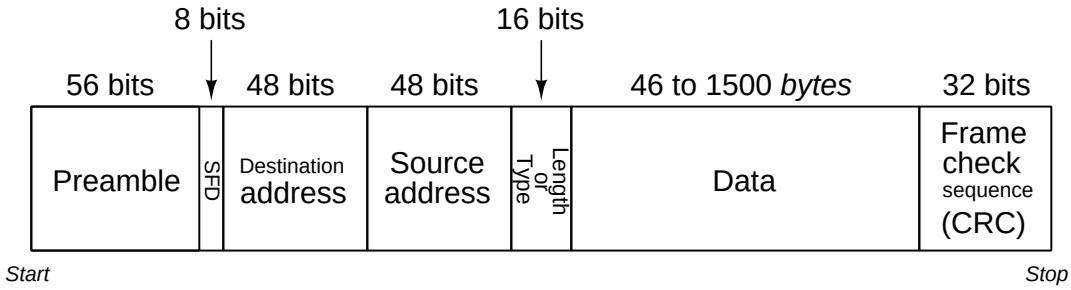
How is the determinism of an Ethernet network between two DTE devices affected if that network is full-duplex rather than half-duplex?

Challenges

- Explain why it is impossible to form a full-duplex network between *more than two* Ethernet DTE devices.

4.1.6 Ethernet data frames

Data in an Ethernet network is transmitted in a series of bits known as a *frame*. A basic organizational illustration for an Ethernet frame is as follows (according to the IEEE 802.3 standard):



Explain the purpose of each frame section:

- Preamble:
- SFD:
- Destination address:
- Source address:
- Type (or Length):
- Data:
- Frame check sequence:

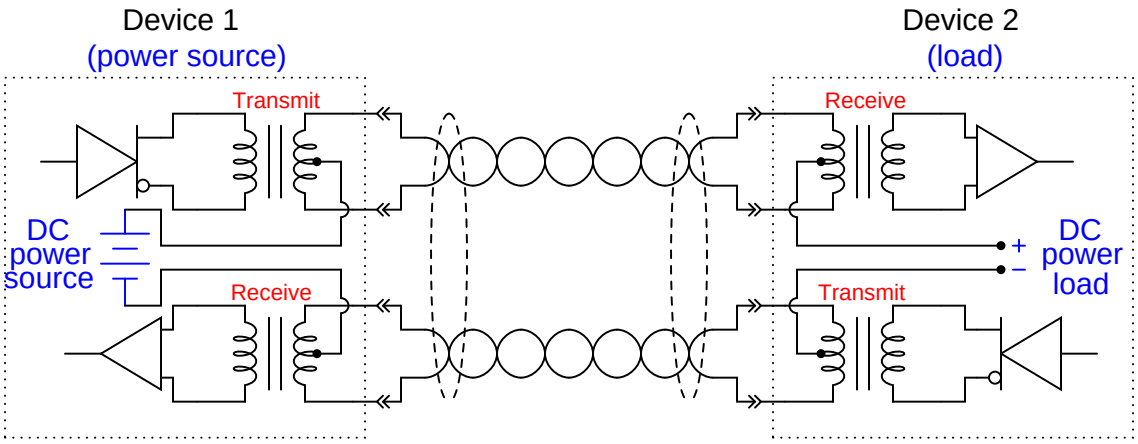
Challenges

- Ethernet data frames do not use a "parity" bit as is the case with data frames of RS-232 and other (simpler) serial network standards. Why is a parity bit unnecessary with Ethernet? Explain why a parity bit would be far less useful (if it was used) in Ethernet than it is in an RS-232 data frame.

4.1.7 Power over Ethernet

Ethernet is a popular communications standard for many digital devices, personal computers included. Originally, Ethernet was intended to be a network standard for conveying digital data only, without power. In later years, however, upgrades to the standard allowed DC power to be conveyed over the same wire pairs. The IEEE standard `802.3af` is one example of a power-over-Ethernet standard.

Shown here is a schematic showing how two Ethernet devices connect together over a Category 5 ("Cat 5") twisted-pair cable, with DC power conveyed over the same wire pair:



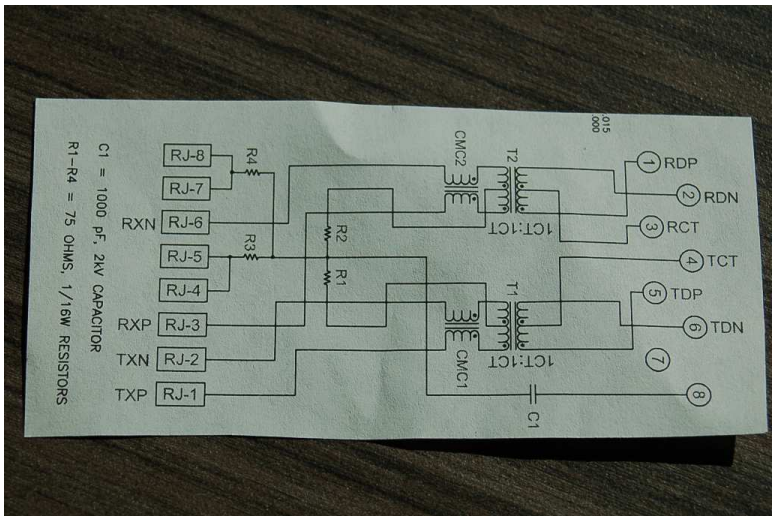
Explain what function(s) the transformers provide in this system, and how they allow DC power to travel through the wire pairs from source to load without interfering with the Ethernet data signals, which are AC.

Challenges

- Identify a practical application for power-over-Ethernet.
- If the power source were AC rather than DC, would interference with the Ethernet signal still be avoided?

4.1.8 RJ-45 connector circuitry

The following schematic diagram is for a PCB-mount RJ-45 connector, used to transfer Ethernet signals from a twisted-pair cable (plugged into the RJ-45 jack) onto traces on a printed circuit board:



Identify the following based on this schematic:

- Which terminals on the schematic diagram represent the pins of the RJ-45 connector?
- Which terminals on the schematic diagram represent the pins soldered to traces on the PCB?
- What is the purpose of components T1 and T2?
- What is the purpose of components CMC1 and CMC2?
- Between which terminals would we expect to see voltage signals representing *transmitted* data?
- Between which terminals would we expect to see voltage signals representing *received* data?

Challenges

- Does this connector have the ability to convey PoE (Power Over Ethernet) signals?

4.2 Quantitative reasoning

These questions are designed to stimulate your computational thinking. In a Socratic discussion with your instructor, the goal is for these questions to reveal your mathematical approach(es) to problem-solving so that good technique and sound reasoning may be reinforced. Your instructor may also pose additional questions based on those assigned, in order to observe your problem-solving firsthand.

Mental arithmetic and estimations are strongly encouraged for all calculations, because without these abilities you will be unable to readily detect errors caused by calculator misuse (e.g. keystroke errors).

You will note a conspicuous lack of answers given for these quantitative questions. Unlike standard textbooks where answers to every other question are given somewhere toward the back of the book, here in these learning modules students must rely on other means to check their work. My advice is to use circuit simulation software such as SPICE to check the correctness of quantitative answers. Refer to those learning modules within this collection focusing on SPICE to see worked examples which you may use directly as practice problems for your own study, and/or as templates you may modify to run your own analyses and generate your own practice problems.

Completely worked example problems found in the Tutorial may also serve as "test cases[4]" for gaining proficiency in the use of circuit simulation software, and then once that proficiency is gained you will never need to rely[5] on an answer key!

[4] In other words, set up the circuit simulation software to analyze the same circuit examples found in the Tutorial. If the simulated results match the answers shown in the Tutorial, it confirms the simulation has properly run. If the simulated results disagree with the Tutorial's answers, something has been set up incorrectly in the simulation software. Using every Tutorial as practice in this way will quickly develop proficiency in the use of circuit simulation software.

[5] This approach is perfectly in keeping with the instructional philosophy of these learning modules: *teaching students to be self-sufficient thinkers*. Answer keys can be useful, but it is even more useful to your long-term success to have a set of tools on hand for checking your own work, because once you have left school and are on your own, there will no longer be "answer keys" available for the problems you will have to solve.

4.2.1 Miscellaneous physical constants

Note: constants shown in **bold** type are *exact*, not approximations. Values inside of parentheses show one standard deviation (σ) of uncertainty in the final digits: for example, the magnetic permeability of free space value given as $1.25663706212(19) \times 10^{-6}$ H/m represents a center value (i.e. the location parameter) of $1.25663706212 \times 10^{-6}$ Henrys per meter with one standard deviation of uncertainty equal to $0.0000000000019 \times 10^{-6}$ Henrys per meter.

Avogadro's number (N_A) = **6.02214076** $\times 10^{23}$ **per mole** (mol^{-1})

Boltzmann's constant (k) = **1.380649** $\times 10^{-23}$ **Joules per Kelvin** (J/K)

Electronic charge (e) = **1.602176634** $\times 10^{-19}$ **Coulomb** (C)

Faraday constant (F) = **96,485.33212...** $\times 10^{4}$ **Coulombs per mole** (C/mol)

Magnetic permeability of free space (μ_0) = $1.25663706212(19) \times 10^{-6}$ Henrys per meter (H/m)

Electric permittivity of free space (ϵ_0) = $8.8541878128(13) \times 10^{-12}$ Farads per meter (F/m)

Characteristic impedance of free space (Z_0) = 376.730313668(57) Ohms (Ω)

Gravitational constant (G) = $6.67430(15) \times 10^{-11}$ cubic meters per kilogram-seconds squared ($m^3/kg\text{-}s^2$)

Molar gas constant (R) = **8.314462618... Joules per mole-Kelvin** (J/mol-K) = 0.08205746(14) liters-atmospheres per mole-Kelvin

Planck constant (h) = **6.62607015** $\times 10^{-34}$ **joule-seconds** (J-s)

Stefan-Boltzmann constant (σ) = **5.670374419...** $\times 10^{-8}$ **Watts per square meter-Kelvin**^{4} ($W/m^2 \cdot K^4$)

Speed of light in a vacuum (c) = **299,792,458 meters per second** (m/s) = 186282.4 miles per second (mi/s)

Note: All constants taken from NIST data "Fundamental Physical Constants – Complete Listing", from `http://physics.nist.gov/constants`, National Institute of Standards and Technology (NIST), 2018 CODATA Adjustment.

4.2.2 Introduction to spreadsheets

A powerful computational tool you are encouraged to use in your work is a *spreadsheet*. Available on most personal computers (e.g. Microsoft Excel), *spreadsheet* software performs numerical calculations based on number values and formulae entered into cells of a grid. This grid is typically arranged as lettered columns and numbered rows, with each cell of the grid identified by its column/row coordinates (e.g. cell `B3`, cell `A8`). Each cell may contain a string of text, a number value, or a mathematical formula. The spreadsheet automatically updates the results of all mathematical formulae whenever the entered number values are changed. This means it is possible to set up a spreadsheet to perform a series of calculations on entered data, and those calculations will be re-done by the computer any time the data points are edited in any way.

For example, the following spreadsheet calculates average speed based on entered values of distance traveled and time elapsed:

	A	B	C	D
1	Distance traveled	46.9	Kilometers	
2	Time elapsed	1.18	Hours	
3	Average speed	= B1 / B2	km/h	
4				
5				

Text labels contained in cells `A1` through `A3` and cells `C1` through `C3` exist solely for readability and are not involved in any calculations. Cell `B1` contains a sample distance value while cell `B2` contains a sample time value. The formula for computing speed is contained in cell `B3`. Note how this formula begins with an "equals" symbol (`=`), references the values for distance and speed by lettered column and numbered row coordinates (`B1` and `B2`), and uses a forward slash symbol for division (`/`). The coordinates `B1` and `B2` function as *variables*[6] would in an algebraic formula.

When this spreadsheet is executed, the numerical value `39.74576` will appear in cell `B3` rather than the formula `= B1 / B2`, because 39.74576 is the computed speed value given 46.9 kilometers traveled over a period of 1.18 hours. If a different numerical value for distance is entered into cell `B1` or a different value for time is entered into cell `B2`, cell `B3`'s value will automatically update. All you need to do is set up the given values and any formulae into the spreadsheet, and the computer will do all the calculations for you.

Cell `B3` may be referenced by other formulae in the spreadsheet if desired, since it is a variable just like the given values contained in `B1` and `B2`. This means it is possible to set up an entire chain of calculations, one dependent on the result of another, in order to arrive at a final value. The arrangement of the given data and formulae need not follow any pattern on the grid, which means you may place them anywhere.

[6] Spreadsheets may also provide means to attach text labels to cells for use as variable names (Microsoft Excel simply calls these labels "names"), but for simple spreadsheets such as those shown here it's usually easier just to use the standard coordinate naming for each cell.

Common[7] arithmetic operations available for your use in a spreadsheet include the following:

- Addition (`+`)
- Subtraction (`-`)
- Multiplication (`*`)
- Division (`/`)
- Powers (`^`)
- Square roots (`sqrt()`)
- Logarithms (`ln()` , `log10()`)

Parentheses may be used to ensure[8] proper order of operations within a complex formula. Consider this example of a spreadsheet implementing the *quadratic formula*, used to solve for roots of a polynomial expression in the form of $ax^2 + bx + c$:

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

	A	B
1	x_1	= (-B4 + sqrt((B4^2) - (4*B3*B5))) / (2*B3)
2	x_2	= (-B4 - sqrt((B4^2) - (4*B3*B5))) / (2*B3)
3	a =	9
4	b =	5
5	c =	-2

This example is configured to compute roots[9] of the polynomial $9x^2 + 5x - 2$ because the values of 9, 5, and -2 have been inserted into cells `B3`, `B4`, and `B5`, respectively. Once this spreadsheet has been built, though, it may be used to calculate the roots of *any* second-degree polynomial expression simply by entering the new a, b, and c coefficients into cells `B3` through `B5`. The numerical values appearing in cells `B1` and `B2` will be automatically updated by the computer immediately following any changes made to the coefficients.

[7] Modern spreadsheet software offers a bewildering array of mathematical functions you may use in your computations. I recommend you consult the documentation for your particular spreadsheet for information on operations other than those listed here.

[8] Spreadsheet programs, like text-based programming languages, are designed to follow standard order of operations by default. However, my personal preference is to use parentheses even where strictly unnecessary just to make it clear to any other person viewing the formula what the intended order of operations is.

[9] Reviewing some algebra here, a *root* is a value for x that yields an overall value of zero for the polynomial. For this polynomial ($9x^2 + 5x - 2$) the two roots happen to be $x = 0.269381$ and $x = -0.82494$, with these values displayed in cells `B1` and `B2`, respectively upon execution of the spreadsheet.

Alternatively, one could break up the long quadratic formula into smaller pieces like this:

$$y = \sqrt{b^2 - 4ac} \qquad z = 2a$$

$$x = \frac{-b \pm y}{z}$$

	A	B	C
1	x_1	= (-B4 + C1) / C2	= sqrt((B4^2) - (4*B3*B5))
2	x_2	= (-B4 - C1) / C2	= 2*B3
3	a =	9	
4	b =	5	
5	c =	-2	

Note how the square-root term (y) is calculated in cell C1, and the denominator term (z) in cell C2. This makes the two final formulae (in cells B1 and B2) simpler to interpret. The positioning of all these cells on the grid is completely arbitrary[10] – all that matters is that they properly reference each other in the formulae.

Spreadsheets are particularly useful for situations where the same set of calculations representing a circuit or other system must be repeated for different initial conditions. The power of a spreadsheet is that it automates what would otherwise be a tedious set of calculations. One specific application of this is to simulate the effects of various components within a circuit failing with abnormal values (e.g. a shorted resistor simulated by making its value nearly zero; an open resistor simulated by making its value extremely large). Another application is analyzing the behavior of a circuit design given new components that are out of specification, and/or aging components experiencing drift over time.

[10] My personal preference is to locate all the "given" data in the upper-left cells of the spreadsheet grid (each data point flanked by a sensible name in the cell to the left and units of measurement in the cell to the right as illustrated in the first distance/time spreadsheet example), sometimes coloring them in order to clearly distinguish which cells contain entered data versus which cells contain computed results from formulae. I like to place all formulae in cells below the given data, and try to arrange them in logical order so that anyone examining my spreadsheet will be able to figure out *how* I constructed a solution. This is a general principle I believe all computer programmers should follow: *document and arrange your code to make it easy for other people to learn from it.*

4.2.3 Preamble time and distance

Calculate the amount of time required for an Ethernet device operating at 10 million bits per second to transmit just the *preamble* of the data frame.

Also, calculate the distance the first bit of an Ethernet frame preamble will travel along a copper Ethernet cable during the preamble period, assuming a transmission speed of 10 million bits per second, a cable velocity factor of 0.7 and a cable temperature of 65 degrees Fahrenheit.

Challenges

- The *preamble* section of an Ethernet frame is extremely important for practical reasons, although at first blush it appears to be useless (an alternating sequence of 1's and 0's?). Explain why the preamble is a necessary component of the Ethernet frame.

4.2.4 MAC address space

Each Ethernet device manufactured in the world today possesses a unique identifying number, known as a *MAC address* or *hardware address*. This address is 6 bytes, or octets, long (48 bits). An example of a valid Ethernet MAC address is shown here:

```
D2-48-1C-30-EA-B5
```

Given the number of bits in a MAC address field, how many unique identifier addresses can exist in the world?

Challenges

- Ethernet MAC addresses are very interesting, in that each one is unique to a single manufactured device. Explain why this is.
- Describe what might happen in an Ethernet network if two or more devices shared the same MAC address.
- In cyber-crime investigations, MAC addresses can be important. Explain why, and also describe how the MAC address of a cyber-criminal might be identified.
- On your personal computer, open up a command-line interface ("cmd" on Microsoft Windows operating systems) and issue the command `arp -a` to see a listing of known IP addresses and their corresponding MAC addresses on your computer's network.

4.2.5 Ethernet collision probability

A fact of life in Ethernet networks is an event called a *collision*. While collisions are normal for an Ethernet network, too high of a collision rate will definitely slow down data transfer.

If we imagine a worst-case scenario, where every device (node) on an Ethernet network is always trying to send data, the probability of a node being delayed due to another node transmitting in the same time slot is $1 - \frac{1}{N}$, where N is the number of nodes, and the probability value lies between 0 and 1 inclusive. As you can see, the probability of collision for a 1-node Ethernet network is zero (there are no other nodes to interfere with), while the probability of delay is 1 (100% chance = absolutely guaranteed all the time) in an Ethernet network having an infinite number of nodes.

The probability that any one node is able to transmit without being delayed by any other node is equal to the probability that all the other nodes on the network are getting delayed by its success. This probability P is equal to:

$$P = \left(1 - \frac{1}{N}\right)^{N-1}$$

The average number of time slots (M) that an Ethernet node must wait before it may transmit depends on this probability:

$$M = \frac{1 - P}{P}$$

Build a computer spreadsheet to calculate both the probability of no-delay transmission (P) and the average number of time slots waiting to transmit (M), then see how these numbers are affected by the number of nodes (N) on the Ethernet network. The following example layout uses yellow shading for the one cell where you enter the number of nodes, and blue shading for those cells containing calculated values (the color-shading being entirely optional):

	1	2	3	4	5
1	# of nodes =				
2					
3	P (no delay) =				
4	Avg slots =				
5					

Do the results surprise you? If so, how?

Challenges

- Examining the formula $1 - \frac{1}{N}$ and imagining the cases of 1 node ($N = 1$) versus an infinite number of nodes ($N = \infty$) is an exercise mathematicians refer to as *limits*. Formally written,

the limit as N approached infinity is: $\lim_{N \to \infty} \left(1 - \frac{1}{N}\right) = 1$. Even though a quantity like "infinity" cannot be handled by a calculator or a spreadsheet program, the concept of imagining what a mathematical function will do as a variable *approaches* infinity is still very useful. Identify the problem-solving technique listed in question 0 that most closely resembles this concept.

4.3 Diagnostic reasoning

These questions are designed to stimulate your deductive and inductive thinking, where you must apply general principles to specific scenarios (deductive) and also derive conclusions about the failed circuit from specific details (inductive). In a Socratic discussion with your instructor, the goal is for these questions to reinforce your recall and use of general circuit principles and also challenge your ability to integrate multiple symptoms into a sensible explanation of what's wrong in a circuit. Your instructor may also pose additional questions based on those assigned, in order to further challenge and sharpen your diagnostic abilities.

As always, your goal is to fully *explain* your analysis of each problem. Simply obtaining a correct answer is not good enough – you must also demonstrate sound reasoning in order to successfully complete the assignment. Your instructor's responsibility is to probe and challenge your understanding of the relevant principles and analytical processes in order to ensure you have a strong foundation upon which to build further understanding.

You will note a conspicuous lack of answers given for these diagnostic questions. Unlike standard textbooks where answers to every other question are given somewhere toward the back of the book, here in these learning modules students must rely on other means to check their work. The best way by far is to debate the answers with fellow students and also with the instructor during the Socratic dialogue sessions intended to be used with these learning modules. Reasoning through challenging questions with other people is an excellent tool for developing strong reasoning skills.

Another means of checking your diagnostic answers, where applicable, is to use circuit simulation software to explore the effects of faults placed in circuits. For example, if one of these diagnostic questions requires that you predict the effect of an open or a short in a circuit, you may check the validity of your work by simulating that same fault (substituting a very high resistance in place of that component for an open, and substituting a very low resistance for a short) within software and seeing if the results agree.

4.3.1 Checking data using a hash function

Copy and paste each of the following text passages into its own plain-text file, then run any hash algorithm you prefer (CRC32, SHA1, SHA256, etc.) to determine whether or not the two passages are character-for-character identical to each other:

First passage:

> Romeo. He jests at scars that never felt a wound.
>
> (Juliet appears above at a window.)
>
> But soft! what light through yonder window breaks? It is the east, and Juliet is the sun! Arise, fair sun, and kill the envious moon, Who is already sick and pale with grief, That thou her maid art far more fair than she: Be not her maid, since she is envious; Her vestal livery is but sick and green, And none but fools do wear it; cast it off. It is my lady; O, it is my love! O, that she knew she were! She speaks, yet she says nothing: what of that? Her eye discourses, I will answer it. I am too bold, 'tis not to me she speaks: Two of the fairest stars in all the heaven, Having some business, do entreat her eyes To twinkle in their spheres till they return. What if her eyes were there, they in her head? The brightness of her cheek would shame those stars, As daylight doth a lamp; her eyes in heaven Would through the airy region stream so bright That birds would sing and think it were not night. See how she leans her cheek upon her hand! O that I were a glove upon that hand, That I might touch that cheek!

Second passage:

> Romeo. He jests at scars that never felt a wound.
>
> (Juliet appears above at a window.)
>
> But soft! what light through yonder window breaks? It is the east, and Juliet is the sun! Arise, fair sun, and kill the envious moon, Who is already sick and pale with grief, That thou her maid art far more fair than she: Be not her maid, since she is envious; Her vestal livery is but sick and green, And none but fools do wear it: cast it off. It is my lady; O, it is my love! O, that she knew she were! She speaks, yet she says nothing: what of that? Her eye discourses, I will answer it. I am too bold, 'tis not to me she speaks: Two of the fairest stars in all the heaven, Having some business, do entreat her eyes To twinkle in their spheres till they return. What if her eyes were there, they in her head? The brightness of her cheek would shame those stars, As daylight doth a lamp; her eyes in heaven Would through the airy region stream so bright That birds would sing and think it were not night. See how she leans her cheek upon her hand! O that I were a glove upon that hand, That I might touch that cheek!

Challenges

- A hash "collision" is when non-identical data sets end up generating the exact same hash value. Is this more likely with CRC-32 or with SHA-256?

Appendix A

Problem-Solving Strategies

The ability to solve complex problems is arguably one of the most valuable skills one can possess, and this skill is particularly important in any science-based discipline.

- Study principles, not procedures. Don't be satisfied with merely knowing how to compute solutions – learn *why* those solutions work.
- Identify what it is you need to solve, identify all relevant data, identify all units of measurement, identify any general principles or formulae linking the given information to the solution, and then identify any "missing pieces" to a solution. Annotate all diagrams with this data.
- Sketch a diagram to help visualize the problem. When building a real system, always devise a plan for that system and analyze its function *before* constructing it.
- Follow the units of measurement and meaning of every calculation. If you are ever performing mathematical calculations as part of a problem-solving procedure, and you find yourself unable to apply each and every intermediate result to some aspect of the problem, it means you don't understand what you are doing. Properly done, every mathematical result should have practical meaning for the problem, and not just be an abstract number. You should be able to identify the proper units of measurement for each and every calculated result, and show where that result fits into the problem.
- Perform "thought experiments" to explore the effects of different conditions for theoretical problems. When troubleshooting real systems, perform *diagnostic tests* rather than visually inspecting for faults, the best diagnostic test being the one giving you the most information about the nature and/or location of the fault with the fewest steps.
- Simplify the problem until the solution becomes obvious, and then use that obvious case as a model to follow in solving the more complex version of the problem.
- Check for exceptions to see if your solution is incorrect or incomplete. A good solution will work for *all* known conditions and criteria. A good example of this is the process of testing scientific hypotheses: the task of a scientist is not to find support for a new idea, but rather to *challenge* that new idea to see if it holds up under a battery of tests. The philosophical

principle of *reductio ad absurdum* (i.e. disproving a general idea by finding a specific case where it fails) is useful here.

- Work "backward" from a hypothetical solution to a new set of given conditions.
- Add quantities to problems that are qualitative in nature, because sometimes a little math helps illuminate the scenario.
- Sketch graphs illustrating how variables relate to each other. These may be quantitative (i.e. with realistic number values) or qualitative (i.e. simply showing increases and decreases).
- Treat quantitative problems as qualitative in order to discern the relative magnitudes and/or directions of change of the relevant variables. For example, try determining what happens if a certain variable were to increase or decrease before attempting to precisely calculate quantities: how will each of the dependent variables respond, by increasing, decreasing, or remaining the same as before?
- Consider limiting cases. This works especially well for qualitative problems where you need to determine which direction a variable will change. Take the given condition and magnify that condition to an extreme degree as a way of simplifying the direction of the system's response.
- Check your work. This means regularly testing your conclusions to see if they make sense. This does *not* mean repeating the same steps originally used to obtain the conclusion(s), but rather to use some other means to check validity. Simply repeating procedures often leads to *repeating the same errors* if any were made, which is why alternative paths are better.

Appendix B

Instructional philosophy

B.1 First principles of learning

- **Anyone can learn anything** given appropriate time, effort, resources, challenges, encouragement, and expectations. Dedicating time and investing effort are the student's responsibility; providing resources, challenges, and encouragement are the teacher's responsibility; high expectations are a responsibility shared by both student and teacher.

- **Transfer is not automatic.** The human mind has a natural tendency to compartmentalize information, which means the process of taking knowledge learned in one context and applying it to another usually does not come easy and therefore should never be taken for granted.

- **Learning is iterative.** The human mind rarely learns anything perfectly on the first attempt. Anticipate mistakes and plan for multiple tries to achieve full understanding, using the lessons of those mistakes as feedback to guide future attempts.

- **Information is absorbed, but understanding is created.** Facts and procedures may be memorized easily enough by repeated exposure, but the ability to reliably apply principles to novel scenarios only comes through intense personal effort. This effort is fundamentally creative in nature: casting new concepts in one's own words, running mental experiments to perceive patterns and trends, etc. These acts of making knowledge "one's own" need not be perfect in order to be effective, as the value lies in the activity and not necessarily the finished product. Depth of understanding and permanence of memory are both directly related to mental effort during the learning process – i.e. *easy learning is shallow and fleeting.*

- **Education trumps training.** There is no such thing as an entirely isolated subject, as all fields of knowledge are connected. Training is narrowly-focused and task-oriented. Education is broad-based and principle-oriented. When preparing for a life-long technical career, education beats training every time.

- **Character matters.** Poor habits are more destructive than deficits of knowledge or skill. This is especially true in collective endeavors, where a team's ability to function depends on trust between its members. Simply put, no one wants an untrustworthy person on their team. An essential component of education then, is character development.

- **People learn to be responsible by bearing responsibility.** An irresponsible person is someone who has never *had* to be responsible for anything that mattered enough to them. Just as anyone can learn anything, anyone can become responsible if the personal cost of irresponsibility becomes high enough.

- **What gets measured, gets done.** Accurate and relevant assessment of learning is key to ensuring all students learn. Therefore, it is imperative to measure what matters.

- **Failure is nothing to fear.** Every human being fails, and fails in multiple ways at multiple times. Eventual success only happens when we don't stop trying.

B.2 Proven strategies for instructors

- Assume every student is capable of learning anything they desire given the proper conditions. Treat them as capable adults by granting real responsibility and avoiding artificial incentives such as merit or demerit points.
- Create a consistent culture of high expectations across the entire program of study. Demonstrate and encourage patience, persistence, and a healthy sense of self-skepticism. Anticipate and de-stigmatize error. Teach respect for the capabilities of others as well as respect for one's own fallibility.
- Replace lecture with "inverted" instruction, where students first encounter new concepts through reading and then spend class time in Socratic dialogue with the instructor exploring those concepts and solving problems individually. There is a world of difference between observing someone solve a problem versus actually solving a problem yourself, and so the point of this form of instruction is to place students in a position where they *cannot* passively observe.
- Require students to read extensively, write about what they learn, and dialogue with you and their peers to sharpen their understanding. Apply Francis Bacon's advice that "reading maketh a full man; conference a ready man; and writing an exact man". These are complementary activities helping students expand their confidence and abilities.
- Use artificial intelligence (AI) to challenge student understanding rather than merely provide information. Find productive ways for AI to critique students' clarity of thought and of expression, for example by employing AI as a Socratic-style interlocutor or as a reviewer of students' journals. Properly applied, AI has the ability to expand student access to critical review well outside the bounds of their instructor's reach.
- Build frequent and rapid feedback into the learning process so that students know at all times how well they are learning, to identify problems early and fix them before they grow. Model the intellectual habit of self-assessing and self-correcting your own understanding (i.e. a cognitive *feedback loop*), encouraging students to do the same.
- Use "mastery" as the standard for every assessment, which means the exam or experiment or project must be done with 100% competence in order to pass. Provide students with multiple opportunity for re-tries (different versions of the assessment every time).
- Require students to devise their own hypotheses and procedures on all experiments, so that the process is truly a scientific one. Have students assess their proposed experimental procedures for risk and devise mitigations for those risks. Let nothing be pre-designed about students' experiments other than a stated task (i.e. what principle the experiment shall test) at the start and a set of demonstrable knowledge and skill objectives at the end.
- Have students build as much of their lab equipment as possible: building power sources, building test assemblies[1], and building complete working systems (no kits!). In order to provide

[1] In the program I teach, every student builds their own "Development Board" consisting of a metal chassis with DIN rail, terminal blocks, and an AC-DC power supply of their own making which functions as a portable lab environment they can use at school as well as take home.

this same "ground-up" experience for every new student, this means either previous students take their creations with them, or the systems get disassembled in preparation for the new students, or the systems grow and evolve with each new student group.

- Incorporate external accountability for you and for your students, continuously improving the curriculum and your instructional methods based on proven results. Have students regularly network with active professionals through participation in advisory committee meetings, service projects, tours, jobshadows, internships, etc. Practical suggestions include requiring students to design and build projects for external clients (e.g. community groups, businesses, different departments within the institution), and also requiring students attend all technical advisory committee meetings and to dialogue with the industry representatives at those meetings.
- Repeatedly explore difficult-to-learn concepts across multiple courses, so that students have multiple opportunities to build their understanding.
- Relate all new concepts, whenever possible, to previous concepts and to relevant physical laws. Challenge each and every student, every day, to *reason* from concept to concept and to explain the logical connections between. Challenge students to verify their conclusions by multiple approaches (e.g. double-checking their work using different methods). Ask *"Why?"* often.
- Maintain detailed records on each student's performance and share these records privately with them. These records should include academic performance as well as professionally relevant behavioral tendencies.
- Hold mandatory "check-in" meetings between all program faculty and each new student during their first term. Offer these to all other students as an option, except for any students continuing to manifest unprofessional behaviors, poor academic performance, or who have some other need for a face-to-face meeting with faculty.
- Address problems while they are small, before they grow larger. This is equally true for tutoring technical concepts as it is for helping students build professional habits.
- Build rigorous quality control into the curriculum to ensure every student masters every important concept, and that the mastery is retained over time. This includes (1) review questions added to every exam to re-assess knowledge taught in previous terms, (2) cumulative exams at the end of every term to re-assess all important concepts back to the very beginning of the program, and (3) review assessments in practical (hands-on) coursework to ensure critically-important skills were indeed taught and are still retained. What you will find by doing this is that it actually boosts retention of students by ensuring that important knowledge gets taught and is retained over long spans of time. In the absence of such quality control, student learning and retention tends to be spotty and this contributes to drop-out and failure rates later in their education.
- Finally, *never rush learning.* Education is not a race. Give your students ample time to digest complex ideas, as you continually remind yourself of just how long it took you to achieve mastery! Long-term retention and the consistently correct application of concepts are always the result of *focused effort over long periods of time* which means there are no shortcuts to learning.

B.3 Proven strategies for students

The single most important piece of advice I have for any student of any subject is to take responsibility for your own development in all areas of life including mental development. Expecting others in your life to entirely guide your own development is a recipe for disappointment. This is just as true for students enrolled in formal learning institutions as it is for auto-didacts pursuing learning entirely on their own. Learning to think in new ways is key to being able to gainfully use information, to make informed decisions about your life, and to best serve those you care about. With this in mind, I offer the following advice to students:

- **Approach all learning as valuable**. No matter what course you take, no matter who you learn from, no matter the subject, there is something useful in every learning experience. If you don't see the value of every new experience, you are not looking closely enough!
- **Continually challenge yourself**. Let other people take shortcuts and find easy answers to easy problems. The purpose of education is to stretch your mind, in order to shape it into a more powerful tool. This doesn't come by taking the path of least resistance. An excellent analogy for an empowering education is productive physical exercise: becoming stronger, more flexible, and more persistent only comes through intense personal effort.
- **Master the use of language**. This includes reading extensively, writing every day, listening closely, and speaking articulately. To a great extent language channels and empowers thought, so the better you are at wielding language the better you will be at grasping abstract concepts and articulating them not only for your benefit but for others as well.
- **Do not limit yourself to the resources given to you**. Read books that are not on the reading list. Run experiments that aren't assigned to you. Form study groups outside of class. Take an entrepreneurial approach to your own education, as though it were a business you were building for your future benefit.
- **Express and share what you learn**. Take every opportunity to teach what you have learned to others, as this will not only help them but will also strengthen your own understanding[2].
- Realize that **no one can give you understanding**, just as no one can give you physical fitness. These both must be *built*.
- **Above all, recognize that learning is hard work, and that a certain level of frustration is unavoidable**. There are times when you will struggle to grasp some of these concepts, and that struggle is a natural thing. Take heart that it will yield with persistent and varied[3] effort, and never give up! That concepts don't immediately come to you is not a sign of something wrong, but rather of something right: that you have found a worthy challenge!

[2] On a personal note, I was surprised to learn just how much my own understanding of electronics and related subjects was strengthened by becoming a teacher. When you are tasked every day with helping other people grasp complex topics, it catalyzes your own learning by giving you powerful incentives to study, to articulate your thoughts, and to reflect deeply on the process of learning.

[3] As the old saying goes, "Insanity is trying the same thing over and over again, expecting different results." If you find yourself stumped by something in the text, you should attempt a different approach. Alter the thought experiment, change the mathematical parameters, do whatever you can to see the problem in a slightly different light, and then the solution will often present itself more readily.

B.4 Design of these learning modules

"The unexamined circuit is not worth energizing" – Socrates (if he had taught electricity)

These learning modules, although useful for self-study, were designed to be used in a formal learning environment where a subject-matter expert challenges students to digest the content and exercise their critical thinking abilities in the answering of questions and in the construction and testing of working circuits. Every effort has been made to embed the following instructional and assessment philosophies within:

- The first goal of education is to enhance clear and independent thought, in order that every student reach their fullest potential in a highly complex and inter-dependent world. Robust reasoning is *always* more important than particulars of any subject matter, because its application is universal.
- Literacy is fundamental to independent learning and thought because text continues to be the most efficient way to communicate complex ideas over space and time. Those who cannot read with ease are limited in their ability to acquire knowledge and perspective.
- Articulate communication is fundamental to work that is complex and interdisciplinary.
- Faulty assumptions and poor reasoning are best corrected through challenge, not presentation. The rhetorical technique of *reductio ad absurdum* (disproving an assertion by exposing an absurdity) works well to discipline student's minds, not only to correct the problem at hand but also to learn how to detect and correct future errors.
- Important principles should be repeatedly explored and widely applied throughout a course of study, not only to reinforce their importance and help ensure their mastery, but also to showcase the interconnectedness and utility of knowledge.

These learning modules were expressly designed to be used in an "inverted" teaching environment[4] where students first read the introductory and tutorial chapters on their own, then individually attempt to answer the questions and construct working circuits according to the experiment and project guidelines. The instructor never lectures, but instead meets regularly with each individual student to review their progress, answer questions, identify misconceptions, and challenge the student to new depths of understanding through further questioning. Regular meetings between instructor and student should resemble a Socratic[5] dialogue, where questions serve as scalpels to dissect topics and expose assumptions. The student passes each module only after consistently demonstrating their ability to logically analyze and correctly apply all major concepts in each question or project/experiment. The instructor must be vigilant in probing each student's understanding to ensure they are truly *reasoning* and not just *memorizing*. This is why "Challenge" points appear throughout, as prompts for students to think deeper about topics and as starting points for instructor queries. Sometimes these challenge points require additional knowledge that hasn't been covered in the series to answer in full. This is okay, as the major purpose of the Challenges is to stimulate analysis and synthesis on the part of each student.

The instructor must possess enough mastery of the subject matter and awareness of students' reasoning to generate their own follow-up questions to practically any student response. Even completely correct answers given by the student should be challenged by the instructor for the purpose of having students practice articulating their thoughts and defending their reasoning. Conceptual errors committed by the student should be exposed and corrected not by direct instruction, but rather by reducing the errors to an absurdity[6] through well-chosen questions and thought experiments posed by the instructor. Becoming proficient at this style of instruction requires time and dedication, but the positive effects on critical thinking for both student and instructor are spectacular.

An inspection of these learning modules reveals certain unique characteristics. One of these is a bias toward thorough explanations in the tutorial chapters. Without a live instructor to explain concepts and applications to students, the text itself must fulfill this role. This philosophy results in lengthier explanations than what you might typically find in a textbook, each step of the reasoning process fully explained, including footnotes addressing common questions and concerns students raise while learning these concepts. Each tutorial seeks to not only explain each major concept in sufficient detail, but also to explain the logic of each concept and how each may be developed

[4] In a traditional teaching environment, students first encounter new information via *lecture* from an expert, and then independently apply that information via *homework*. In an "inverted" course of study, students first encounter new information via *homework*, and then independently apply that information under the scrutiny of an expert. The expert's role in lecture is to simply *explain*, but the expert's role in an inverted session is to *challenge*, *critique*, and if necessary *explain* where gaps in understanding still exist.

[5] Socrates is a figure in ancient Greek philosophy famous for his unflinching style of questioning. Although he authored no texts, he appears as a character in Plato's many writings. The essence of Socratic philosophy is to leave no question unexamined and no point of view unchallenged. While purists may argue a topic such as electric circuits is too narrow for a true Socratic-style dialogue, I would argue that the essential thought processes involved with scientific reasoning on *any* topic are not far removed from the Socratic ideal, and that students of electricity and electronics would do very well to challenge assumptions, pose thought experiments, identify fallacies, and otherwise employ the arsenal of critical thinking skills modeled by Socrates.

[6] This rhetorical technique is known by the Latin phrase *reductio ad absurdum*. The concept is to expose errors by counter-example, since only one solid counter-example is necessary to disprove a universal claim. As an example of this, consider the common misconception among beginning students of electricity that voltage cannot exist without current. One way to apply *reductio ad absurdum* to this statement is to ask how much current passes through a fully-charged battery connected to nothing (i.e. a clear example of voltage existing without current).

from "first principles". Again, this reflects the goal of developing clear and independent thought in students' minds, by showing how clear and logical thought was used to forge each concept. Students benefit from witnessing a model of clear thinking in action, and these tutorials strive to be just that.

Another feature of these learning modules is that they do not treat topics in isolation. Rather, important concepts are introduced early in the series, and appear repeatedly as stepping-stones toward other concepts in subsequent modules. This helps to avoid the "compartmentalization" of knowledge, demonstrating the inter-connectedness of concepts and simultaneously reinforcing them. Each module is fairly complete in itself, reserving the beginning of its tutorial to a review of foundational concepts.

To high standards of education,

Tony R. Kuphaldt

Appendix C

Tools used

I am indebted to the developers of many open-source software applications in the creation of these learning modules. The following is a list of these applications with some commentary on each.

You will notice a theme common to many of these applications: a bias toward *code*. Although I am by no means an expert programmer in any computer language, I understand and appreciate the flexibility offered by code-based applications where the user (you) enters commands into a plain ASCII text file, which the software then reads and processes to create the final output. Code-based computer applications are by their very nature *extensible*, while WYSIWYG (What You See Is What You Get) applications are generally limited to whatever user interface the developer makes for you.

The `GNU/Linux` computer operating system

> There is so much to be said about Linus Torvalds' `Linux` and Richard Stallman's `GNU` project. First, to credit just these two individuals is to fail to do justice to the *mob* of passionate volunteers who contributed to make this amazing software a reality. I first learned of `Linux` back in 1996, and have been using this operating system on my personal computers almost exclusively since then. It is *free*, it is completely *configurable*, and it permits the continued use of highly efficient `Unix` applications and scripting languages (e.g. shell scripts, Makefiles, `sed`, `awk`) developed over many decades. `Linux` not only provided me with a powerful computing platform, but its open design served to inspire my life's work of creating open-source educational resources.

Bram Moolenaar's `Vim` text editor

> Writing code for any code-based computer application requires a *text editor*, which may be thought of as a word processor strictly limited to outputting plain-ASCII text files. Many good text editors exist, and one's choice of text editor seems to be a deeply personal matter within the programming world. I prefer `Vim` because it operates very similarly to `vi` which is ubiquitous on `Unix`/`Linux` operating systems, and because it may be entirely operated via keyboard (i.e. no mouse required) which makes it fast to use.

Donald Knuth's TEX typesetting system

Developed in the late 1970's and early 1980's by computer scientist extraordinaire Donald Knuth to typeset his multi-volume magnum opus *The Art of Computer Programming*, this software allows the production of formatted text for screen-viewing or paper printing, all by writing plain-text code to describe how the formatted text is supposed to appear. TEX is not just a markup language for documents, but it is also a Turing-complete programming language in and of itself, allowing useful algorithms to be created to control the production of documents. Simply put, TEX *is a programmer's approach to word processing*. Since TEX is controlled by code written in a plain-text file, this means anyone may read that plain-text file to see exactly how the document was created. This openness afforded by the code-based nature of TEX makes it relatively easy to learn how other people have created their own TEX documents. By contrast, examining a beautiful document created in a conventional WYSIWYG word processor such as Microsoft `Word` suggests nothing to the reader about *how* that document was created, or what the user might do to create something similar. As Mr. Knuth himself once quipped, conventional word processing applications should be called WYSIAYG (What You See Is *All* You Get).

Leslie Lamport's LATEX extensions to TEX

Like all true programming languages, TEX is inherently extensible. So, years after the release of TEX to the public, Leslie Lamport decided to create a massive extension allowing easier compilation of book-length documents. The result was LATEX, which is the markup language used to create all ModEL module documents. You could say that TEX is to LATEX as `C` is to `C++`. This means it is permissible to use any and all TEX commands within LATEX source code, and it all still works. Some of the features offered by LATEX that would be challenging to implement in TEX include automatic index and table-of-content creation.

Tim Edwards' `Xcircuit` drafting program

This wonderful program is what I use to create all the schematic diagrams and illustrations (but not photographic images or mathematical plots) throughout the ModEL project. It natively outputs PostScript format which is a true vector graphic format (this is why the images do not pixellate when you zoom in for a closer view), and it is so simple to use that I have never had to read the manual! Object libraries are easy to create for `Xcircuit`, being plain-text files using PostScript programming conventions. Over the years I have collected a large set of object libraries useful for drawing electrical and electronic schematics, pictorial diagrams, and other technical illustrations.

`Gimp` graphic image manipulation program

Essentially an open-source clone of Adobe's `PhotoShop`, I use `Gimp` to resize, crop, and convert file formats for all of the photographic images appearing in the ModEL modules. Although `Gimp` does offer its own scripting language (called `Script-Fu`), I have never had occasion to use it. Thus, my utilization of `Gimp` to merely crop, resize, and convert graphic images is akin to using a sword to slice bread.

`SPICE` circuit simulation program

`SPICE` is to circuit analysis as TEX is to document creation: it is a form of markup language designed to describe a certain object to be processed in plain-ASCII text. When the plain-text "source file" is compiled by the software, it outputs the final result. More modern circuit analysis tools certainly exist, but I prefer `SPICE` for the following reasons: it is *free*, it is *fast*, it is *reliable*, and it is a fantastic tool for *teaching* students of electricity and electronics how to write simple code. I happen to use rather old versions of `SPICE`, version 2g6 being my "go to" application when I only require text-based output. `NGSPICE` (version 26), which is based on Berkeley `SPICE` version 3f5, is used when I require graphical output for such things as time-domain waveforms and Bode plots. In all `SPICE` example netlists I strive to use coding conventions compatible with all `SPICE` versions.

Andrew D. Hwang's `ePiX` mathematical visualization programming library

This amazing project is a `C++` library you may link to any `C/C++` code for the purpose of generating PostScript graphic images of mathematical functions. As a completely free and open-source project, it does all the plotting I would otherwise use a Computer Algebra System (CAS) such as `Mathematica` or `Maple` to do. It should be said that `ePiX` is *not* a Computer Algebra System like `Mathematica` or `Maple`, but merely a mathematical *visualization* tool. In other words, it won't determine integrals for you (you'll have to implement that in your own `C/C++` code!), but it can graph the results, and it does so beautifully. What I really admire about `ePiX` is that it is a `C++` programming library, which means it builds on the existing power and toolset available with that programming language. Mr. Hwang could have probably developed his own stand-alone application for mathematical plotting, but by creating a `C++` library to do the same thing he accomplished something much greater.

`gnuplot` mathematical visualization software

Another open-source tool for mathematical visualization is `gnuplot`. Interestingly, this tool is *not* part of Richard Stallman's GNU project, its name being a coincidence. For this reason the authors prefer "gnu" *not* be capitalized at all to avoid confusion. This is a much "lighter-weight" alternative to a spreadsheet for plotting tabular data, and the fact that it easily outputs directly to an X11 console or a file in a number of different graphical formats (including PostScript) is very helpful. I typically set my `gnuplot` output format to default (X11 on my Linux PC) for quick viewing while I'm developing a visualization, then switch to PostScript file export once the visual is ready to include in the document(s) I'm writing. As with my use of `Gimp` to do rudimentary image editing, my use of `gnuplot` only scratches the surface of its capabilities, but the important points are that it's *free* and that it *works well*.

`Python` programming language

Both Python and C++ find extensive use in these modules as instructional aids and exercises, but I'm listing Python here as a *tool* for myself because I use it almost daily as a *calculator*. If you open a Python interpreter console and type `from math import *` you can type mathematical expressions and have it return results just as you would on a hand calculator. Complex-number (i.e. *phasor*) arithmetic is similarly supported if you include the complex-math library (`from cmath import *`). Examples of this are shown in the Programming References chapter (if included) in each module. Of course, being a fully-featured programming language, Python also supports conditionals, loops, and other structures useful for calculation of quantities. Also, running in a console environment where all entries and returned values show as text in a chronologically-ordered list makes it easy to copy-and-paste those calculations to document exactly how they were performed.

Appendix D

Creative Commons License

Creative Commons Attribution 4.0 International Public License

By exercising the Licensed Rights (defined below), You accept and agree to be bound by the terms and conditions of this Creative Commons Attribution 4.0 International Public License ("Public License"). To the extent this Public License may be interpreted as a contract, You are granted the Licensed Rights in consideration of Your acceptance of these terms and conditions, and the Licensor grants You such rights in consideration of benefits the Licensor receives from making the Licensed Material available under these terms and conditions.

Section 1 – Definitions.

a. **Adapted Material** means material subject to Copyright and Similar Rights that is derived from or based upon the Licensed Material and in which the Licensed Material is translated, altered, arranged, transformed, or otherwise modified in a manner requiring permission under the Copyright and Similar Rights held by the Licensor. For purposes of this Public License, where the Licensed Material is a musical work, performance, or sound recording, Adapted Material is always produced where the Licensed Material is synched in timed relation with a moving image.

b. **Adapter's License** means the license You apply to Your Copyright and Similar Rights in Your contributions to Adapted Material in accordance with the terms and conditions of this Public License.

c. **Copyright and Similar Rights** means copyright and/or similar rights closely related to copyright including, without limitation, performance, broadcast, sound recording, and Sui Generis Database Rights, without regard to how the rights are labeled or categorized. For purposes of this Public License, the rights specified in Section 2(b)(1)-(2) are not Copyright and Similar Rights.

d. **Effective Technological Measures** means those measures that, in the absence of proper authority, may not be circumvented under laws fulfilling obligations under Article 11 of the WIPO Copyright Treaty adopted on December 20, 1996, and/or similar international agreements.

e. **Exceptions and Limitations** means fair use, fair dealing, and/or any other exception or

limitation to Copyright and Similar Rights that applies to Your use of the Licensed Material.

f. **Licensed Material** means the artistic or literary work, database, or other material to which the Licensor applied this Public License.

g. **Licensed Rights** means the rights granted to You subject to the terms and conditions of this Public License, which are limited to all Copyright and Similar Rights that apply to Your use of the Licensed Material and that the Licensor has authority to license.

h. **Licensor** means the individual(s) or entity(ies) granting rights under this Public License.

i. **Share** means to provide material to the public by any means or process that requires permission under the Licensed Rights, such as reproduction, public display, public performance, distribution, dissemination, communication, or importation, and to make material available to the public including in ways that members of the public may access the material from a place and at a time individually chosen by them.

j. **Sui Generis Database Rights** means rights other than copyright resulting from Directive 96/9/EC of the European Parliament and of the Council of 11 March 1996 on the legal protection of databases, as amended and/or succeeded, as well as other essentially equivalent rights anywhere in the world.

k. **You** means the individual or entity exercising the Licensed Rights under this Public License. **Your** has a corresponding meaning.

Section 2 – Scope.

a. License grant.

1. Subject to the terms and conditions of this Public License, the Licensor hereby grants You a worldwide, royalty-free, non-sublicensable, non-exclusive, irrevocable license to exercise the Licensed Rights in the Licensed Material to:

A. reproduce and Share the Licensed Material, in whole or in part; and

B. produce, reproduce, and Share Adapted Material.

2. Exceptions and Limitations. For the avoidance of doubt, where Exceptions and Limitations apply to Your use, this Public License does not apply, and You do not need to comply with its terms and conditions.

3. Term. The term of this Public License is specified in Section 6(a).

4. Media and formats; technical modifications allowed. The Licensor authorizes You to exercise the Licensed Rights in all media and formats whether now known or hereafter created, and to make technical modifications necessary to do so. The Licensor waives and/or agrees not to assert any right or authority to forbid You from making technical modifications necessary to exercise the Licensed Rights, including technical modifications necessary to circumvent Effective Technological Measures.

For purposes of this Public License, simply making modifications authorized by this Section 2(a)(4) never produces Adapted Material.

5. Downstream recipients.

A. Offer from the Licensor – Licensed Material. Every recipient of the Licensed Material automatically receives an offer from the Licensor to exercise the Licensed Rights under the terms and conditions of this Public License.

B. No downstream restrictions. You may not offer or impose any additional or different terms or conditions on, or apply any Effective Technological Measures to, the Licensed Material if doing so restricts exercise of the Licensed Rights by any recipient of the Licensed Material.

6. No endorsement. Nothing in this Public License constitutes or may be construed as permission to assert or imply that You are, or that Your use of the Licensed Material is, connected with, or sponsored, endorsed, or granted official status by, the Licensor or others designated to receive attribution as provided in Section 3(a)(1)(A)(i).

b. Other rights.

1. Moral rights, such as the right of integrity, are not licensed under this Public License, nor are publicity, privacy, and/or other similar personality rights; however, to the extent possible, the Licensor waives and/or agrees not to assert any such rights held by the Licensor to the limited extent necessary to allow You to exercise the Licensed Rights, but not otherwise.

2. Patent and trademark rights are not licensed under this Public License.

3. To the extent possible, the Licensor waives any right to collect royalties from You for the exercise of the Licensed Rights, whether directly or through a collecting society under any voluntary or waivable statutory or compulsory licensing scheme. In all other cases the Licensor expressly reserves any right to collect such royalties.

Section 3 – License Conditions.

Your exercise of the Licensed Rights is expressly made subject to the following conditions.

a. Attribution.

1. If You Share the Licensed Material (including in modified form), You must:

A. retain the following if it is supplied by the Licensor with the Licensed Material:

i. identification of the creator(s) of the Licensed Material and any others designated to receive attribution, in any reasonable manner requested by the Licensor (including by pseudonym if designated);

ii. a copyright notice;

iii. a notice that refers to this Public License;

iv. a notice that refers to the disclaimer of warranties;

v. a URI or hyperlink to the Licensed Material to the extent reasonably practicable;

B. indicate if You modified the Licensed Material and retain an indication of any previous modifications; and

C. indicate the Licensed Material is licensed under this Public License, and include the text of, or the URI or hyperlink to, this Public License.

2. You may satisfy the conditions in Section 3(a)(1) in any reasonable manner based on the medium, means, and context in which You Share the Licensed Material. For example, it may be reasonable to satisfy the conditions by providing a URI or hyperlink to a resource that includes the required information.

3. If requested by the Licensor, You must remove any of the information required by Section 3(a)(1)(A) to the extent reasonably practicable.

4. If You Share Adapted Material You produce, the Adapter's License You apply must not prevent recipients of the Adapted Material from complying with this Public License.

Section 4 – Sui Generis Database Rights.

Where the Licensed Rights include Sui Generis Database Rights that apply to Your use of the Licensed Material:

a. for the avoidance of doubt, Section 2(a)(1) grants You the right to extract, reuse, reproduce, and Share all or a substantial portion of the contents of the database;

b. if You include all or a substantial portion of the database contents in a database in which You have Sui Generis Database Rights, then the database in which You have Sui Generis Database Rights (but not its individual contents) is Adapted Material; and

c. You must comply with the conditions in Section 3(a) if You Share all or a substantial portion of the contents of the database.

For the avoidance of doubt, this Section 4 supplements and does not replace Your obligations under this Public License where the Licensed Rights include other Copyright and Similar Rights.

Section 5 – Disclaimer of Warranties and Limitation of Liability.

a. Unless otherwise separately undertaken by the Licensor, to the extent possible, the Licensor offers the Licensed Material as-is and as-available, and makes no representations or warranties of any kind concerning the Licensed Material, whether express, implied, statutory, or other. This includes, without limitation, warranties of title, merchantability, fitness for a particular purpose, non-infringement, absence of latent or other defects, accuracy, or the presence or absence of errors,

whether or not known or discoverable. Where disclaimers of warranties are not allowed in full or in part, this disclaimer may not apply to You.

b. To the extent possible, in no event will the Licensor be liable to You on any legal theory (including, without limitation, negligence) or otherwise for any direct, special, indirect, incidental, consequential, punitive, exemplary, or other losses, costs, expenses, or damages arising out of this Public License or use of the Licensed Material, even if the Licensor has been advised of the possibility of such losses, costs, expenses, or damages. Where a limitation of liability is not allowed in full or in part, this limitation may not apply to You.

c. The disclaimer of warranties and limitation of liability provided above shall be interpreted in a manner that, to the extent possible, most closely approximates an absolute disclaimer and waiver of all liability.

Section 6 – Term and Termination.

a. This Public License applies for the term of the Copyright and Similar Rights licensed here. However, if You fail to comply with this Public License, then Your rights under this Public License terminate automatically.

b. Where Your right to use the Licensed Material has terminated under Section 6(a), it reinstates:

1. automatically as of the date the violation is cured, provided it is cured within 30 days of Your discovery of the violation; or

2. upon express reinstatement by the Licensor.

For the avoidance of doubt, this Section 6(b) does not affect any right the Licensor may have to seek remedies for Your violations of this Public License.

c. For the avoidance of doubt, the Licensor may also offer the Licensed Material under separate terms or conditions or stop distributing the Licensed Material at any time; however, doing so will not terminate this Public License.

d. Sections 1, 5, 6, 7, and 8 survive termination of this Public License.

Section 7 – Other Terms and Conditions.

a. The Licensor shall not be bound by any additional or different terms or conditions communicated by You unless expressly agreed.

b. Any arrangements, understandings, or agreements regarding the Licensed Material not stated herein are separate from and independent of the terms and conditions of this Public License.

Section 8 – Interpretation.

a. For the avoidance of doubt, this Public License does not, and shall not be interpreted to, reduce, limit, restrict, or impose conditions on any use of the Licensed Material that could lawfully

be made without permission under this Public License.

b. To the extent possible, if any provision of this Public License is deemed unenforceable, it shall be automatically reformed to the minimum extent necessary to make it enforceable. If the provision cannot be reformed, it shall be severed from this Public License without affecting the enforceability of the remaining terms and conditions.

c. No term or condition of this Public License will be waived and no failure to comply consented to unless expressly agreed to by the Licensor.

d. Nothing in this Public License constitutes or may be interpreted as a limitation upon, or waiver of, any privileges and immunities that apply to the Licensor or You, including from the legal processes of any jurisdiction or authority.

Creative Commons is not a party to its public licenses. Notwithstanding, Creative Commons may elect to apply one of its public licenses to material it publishes and in those instances will be considered the "Licensor." Except for the limited purpose of indicating that material is shared under a Creative Commons public license or as otherwise permitted by the Creative Commons policies published at `creativecommons.org/policies`, Creative Commons does not authorize the use of the trademark "Creative Commons" or any other trademark or logo of Creative Commons without its prior written consent including, without limitation, in connection with any unauthorized modifications to any of its public licenses or any other arrangements, understandings, or agreements concerning use of licensed material. For the avoidance of doubt, this paragraph does not form part of the public licenses.

Creative Commons may be contacted at `creativecommons.org`.

Appendix E

References

"Demystifying Ethernet Category Types", version 2.0, Datatronix white paper.

Horak, Ray, *Telecommunications and Data Communications Handbook*, John Wiley & Sons, Inc., New York, NY, 2007.

Horak, Ray, *Webster's New World Telecom Dictionary*, Wiley Publishing, Inc., Indianapolis, IN, 2008.

Newton, Harry, *Newton's Telecom Dictionary*, CMP Books, San Francisco, CA, 2005.

Park, John; Mackay, Steve; Wright, Edwin; *Practical Data Communications for Instrumentation and Control*, IDC Technologies, published by Newnes (an imprint of Elsevier), Oxford, England, 2003.

Spurgeon, Charles E., *Ethernet: The Definitive Guide*, O'Reilly Media, Inc., Sebastopol, CA, 2000.

Appendix F

Version history

This is a list showing all significant additions, corrections, and other edits made to this learning module. Each entry is referenced by calendar date in reverse chronological order (newest version first), which appears on the front cover of every learning module for easy reference. Any contributors to this open-source document are listed here as well.

19 February 2026 – corrected an error where I listed the EtherType code for IPv4 as `0x8000` instead of the correct value `0x0800`.

4 January 2026 – added a new section to the Tutorial on Ethernet data frame format.

30 December 2025 – added a new Conceptual Reasoning question on full- versus half-duplex Ethernet and the effects on determinism.

17 September 2025 – changed the title of the "Overview" section to "Ethernet overview" in order to make section-title search results more meaningful. Also added a Challenge question.

4 October 2024 – added more content on modern Ethernet cable categories.

15-18 September 2024 – divided the Introduction chapter into sections, one with recommendations for students, one with a listing of challenging concepts, and one with recommendations for instructors. Also made some minor edits to the Tutorial.

2 June 2024 – added new Tutorial section on the use of shift registers within all serial communication networks.

14 February 2024 – added instructor notes.

1 January 2024 – edited image_3915 to label the two-byte "Length" field of Ethernet frames as being either "Length or Type" so as to not omit proper nomenclature for the Ethernet II frame standard.

28 December 2023 – corrected an omission in the Diagnostic Reasoning question "Checking data

using a hash function" where I never really stated the purpose of the question.

29 November 2022 – placed questions at the top of the itemized list in the Introduction chapter prompting students to devise experiments related to the tutorial content.

12 August 2022 – minor edits to the Tutorial.

23 July 2022 – added a Conceptual Reasoning question to analyze the schematic diagram of a transformer-isolated RJ-45 jack.

26 June 2022 – fixed an omission in the table showing UTP cable conductor assignments for 10 Mbps Ethernet (The Receive Data line did not show "RD−" as it should have).

9 July 2021 – replaced some TeX-style italicizing markup with LaTeX-style.

10 May 2021 – commented out or deleted empty chapters.

17 February 2021 – corrected some typographical errors.

22 September 2020 – added content to the Introduction chapter, some new index entries to the Tutorial, and some more instructor notes to questions.

5 September 2020 – significantly edited the Introduction chapter to make it more suitable as a pre-study guide and to provide cues useful to instructors leading "inverted" teaching sessions.

25 June 2020 – added a Technical Reference section on the OSI Reference Model.

21 June 2020 – document first created.

Index

Adding quantities to a qualitative problem, 56
Annotating diagrams, 55
Asynchronous data transfer, 11

Bit, 8
Byte, 8

Checking for exceptions, 56
Checking your work, 56
Clock pulse, 8
Code, computer, 65
Collision domain, Ethernet, 15
CRC, 23
CRC32 hash algorithm, 28
Crossover cable, 18
CSMA/CD channel arbitration, 12
Cyclic redundancy check, 23

Data Communications Equipment, 14
Data Terminal Equipment, 14
DCE, 14
Differential signal, 11
Differential signaling, 16
Digest, 23
Dimensional analysis, 55
DIX Ethernet, 12
Drop, 13
DTE, 14

Edwards, Tim, 66
EIA/TIA-232 serial communication, 17
EIA/TIA-485 serial communication, 17
Electric signal coupling, 16
Ethernet, 12
EtherType code, 20

Fieldbus, 13
Flip-flop, 9
FOUNDATION Fieldbus, 13
Frame check sequence, 23

Graph values to solve a problem, 56
Greenleaf, Cynthia, 31

Hash algorithm, 23
Hash collision, 24
How to teach with these modules, 63
Hwang, Andrew D., 67

Identify given data, 55
Identify relevant principles, 55
IEEE Registration Authority, 20
Intermediate results, 55
Internet Protocol, 20
Inverted instruction, 63
IPv4, 20

Knuth, Donald, 66

Lamport, Leslie, 66
Layer, OSI Reference Model, 13, 14, 22
Limiting cases, 56
Luminiferous ether, 12

MAC address, Ethernet, 13
Magnetic signal coupling, 16
Metacognition, 36
Metcalfe, Robert, 12
Moolenaar, Bram, 65
Murphy, Lynn, 31

Null modem, 18
Nybble, 8

Open-source, 65
OSI Reference Model, 13, 14, 22

Parallel communication, 8
Parity, 23
Phase-locked loop, 20
PLL, 20
Preamble, 20
Problem-solving: annotate diagrams, 55
Problem-solving: check for exceptions, 56
Problem-solving: checking work, 56
Problem-solving: dimensional analysis, 55
Problem-solving: graph values, 56
Problem-solving: identify given data, 55
Problem-solving: identify relevant principles, 55
Problem-solving: interpret intermediate results, 55
Problem-solving: limiting cases, 56
Problem-solving: qualitative to quantitative, 56
Problem-solving: quantitative to qualitative, 56
Problem-solving: reductio ad absurdum, 56
Problem-solving: simplify the system, 55
Problem-solving: thought experiment, 55
Problem-solving: track units of measurement, 55
Problem-solving: visually represent the system, 55
Problem-solving: work in reverse, 56

Qualitatively approaching a quantitative problem, 56

Reading Apprenticeship, 31
Reductio ad absurdum, 56, 62, 63
Resistor, termination, 13
RS-232 serial communication, 17
RS-485 serial communication, 17

Schoenbach, Ruth, 31
Scientific method, 36
Serial communication, 8
SHA1 hash algorithm, 29
Simplifying a system, 55
Single-ended signal, 11
Single-ended signaling, 16
Socrates, 62
Socratic dialogue, 63
SPICE, 31
Spur, 13
Stallman, Richard, 65
Stub, 13
Switching hub, Ethernet, 21
Synchronous data transfer, 11

Termination resistor, 13
Thought experiment, 55
Torvalds, Linus, 65

Units of measurement, 55
Unshielded, twisted pair (UTP) cable, 16
UTP cable, 16

Visualizing a system, 55

Word, 8
Work in reverse to solve a problem, 56
WYSIWYG, 65, 66