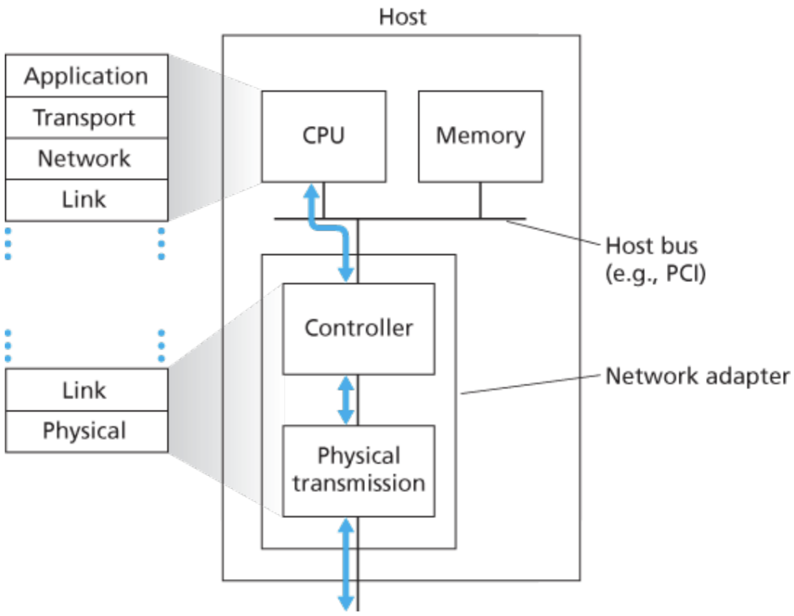
Justin Ciocoi Oct. 23, 2023

CSCI 379 Computer Networking

Textbook Notes 6.1-6.3

6.1: Introduction to the Link Layer

- In this chapter, any device that runs a link-layer protocol will be referred to as a *node*
  - Nodes include hosts, routers, switches, and Wi-Fi access points
- The communication paths between nodes connecting them will be referred to as *links*
- Over any given link, a transmitting node encapsulates the datagram in a *link-layer frame* and transmits the frame into the link
- **6.1.1: Services Provided by the Link Layer**
  - Provided services can vary from one link-layer protocol to the next
  - Possible services provided by link-layer protocols include
    - *Framing*, which refers to the encapsulation of a network layer datagram in a link-layer frame prior to transmission
    - *Link access*, such as through the medium access control (MAC) protocol, which deals with managing access and coordination of frame transmission over different nodes
    - *Reliable Delivery*, which similar to TCP refers to the guarantees made by the link-layer in terms of guaranteed and in-order delivery using acknowledgments and retransmissions
    - *Error Detection and Correction*, which functions like the checksum properties in the network layer, but generally uses more sophisticated error checking and correcting methodologies
- **6.1.2: Where is the Link Layer Implemented**
  - For the most part, in user-side end systems, the link-layer is implemented in the form of a *network adapter* and *network interface card (NIC)*



- At its heart is the link-layer controller, which is generally a single, special-purpose chip that implements many of the link-layer's services
- So, as we can see, the majority of a link-layer controller's functionality is implemented in hardware
- Until the late 1990s, network adapters were almost exclusively physically separate cards (PCMCIA, PCI, PCIe), but in the modern day, motherboards almost always have an on-board LAN configuration
- Nowadays, many motherboards. even have on-board Wi-Fi configurations
- While the link-layer *is* largely implemented in hardware, software side implementations also assist in controller interrupts, error handling and datagram passing to the network layer

---

6.2: Error-Detection and -Correction Techniques

- We have noted throughout the text that error-detection and -correction techniques are two services that are often provided by the link layer, as well as the transport layer
- In this section, we will examine some of the simplest techniques that can be used to detect, and in some cases, correct bit errors

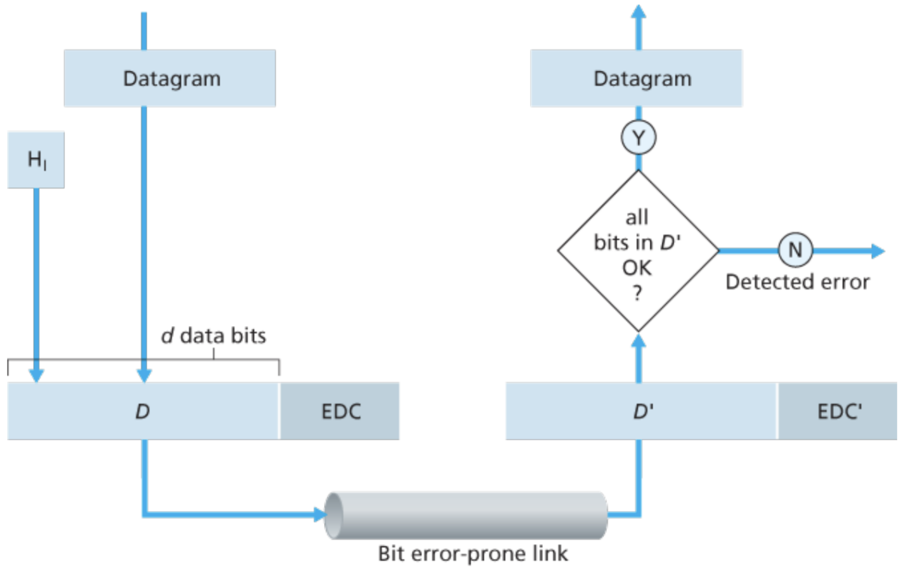


Figure 6.3 Error-detection and -correction scenario

- Here, we can see that at the sending node, D represents data to be protected against bit errors
    - This data will be augmented with error-detection and -correction bits (EDC)
- The data to be protected typically does not only include the datagram passed down from the network layer, but also link-level addressing information and other fields in the link frame header
- At the receiving node, D' and EDC' are received, which might differ from the original D and EDC
- Now, the receivers challenge is to determine whether the D' which it has received is the same as the original D
- Error-detection and -correction techniques allow the receiver to sometimes, *but not always*, detect when bit errors have occurred
- There can still be *undetected bit errors*, which means the receiver might deliver a corrupted datagram to the network layer, or be unaware the contents of a field in the frame's header has been corrupted
- Thus, we want to choose an error-detection scheme that keeps the probability of such occurrences small
- In general, more sophisticated error-detection and -correction algorithms incur more overhead due to the need for more computation and transmission of EDC bits

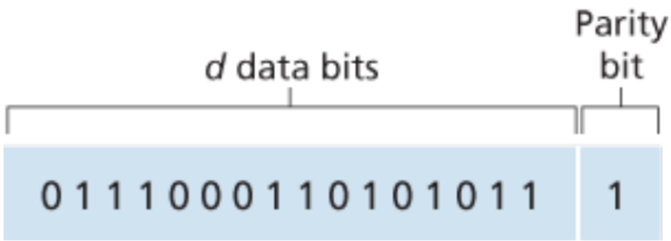
- **6.2.1: Parity Checks**
  - The simplest form of error detection is the use of a single *parity bit*
  - Suppose that the information we are sending, D, has d bits
    - In an *even parity scheme* the sender includes one additional bit such that the total number of 1s in the $d + 1$ bits is even
    - For *odd parity schemes*, the parity bit value is chosen such that there is an odd number of 1s
-


Figure 6.4 One-bit even parity

  - Receiver operation is also fairly simple when utilizing a parity bit
  - The receiver need only count the number of 1s in the received d+1 bits
    - If an odd number of 1-valued bits are found within an even parity scheme, the receiver knows that at least one bit error has occurred
    - More precisely, the receiver knows that some *odd* number of bit errors have occurred
  - But what happens when an even number of bit errors occurs?
    - This would result in an undetected error
  - If the probability of bit errors is small, and errors can be assumed to occur independently from one another, the probability of having multiple bit errors in a single packet is extremely small
  - However, measurements have shown that, rather than occurring independently, errors are often clustered together in *bursts*
    - Under burst conditions, the probability of undetected errors in a frame protected by single-bit parity can approach 50%

  - Clearly, a more robust error detection scheme is needed for practical use, but before examining such a scheme, we will consider a generalization of one bit parity that will provide us with some insight into error-correction techniques

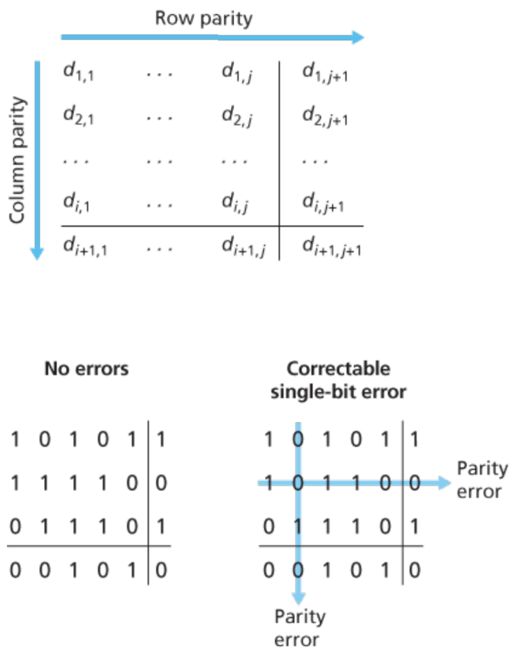
- 

Figure 6.5 Two-dimensional even parity

  - As shown in the above diagram, a two-dimensional generalization can help us better understand error correction techniques
  - Here, the d bits in D are divided into i rows and j columns, and a parity value will be computed in each row and each column
    - The resulting $i + j + 1$ parity bits comprise the link-layer frame's error detection bits
  - Now, suppose that a single bit error occurs in the original d bits of information
  - In this scheme, the parity of both the column and the row containing the flipped bit will be in error
    - Therefore, the receiver can use the indices of the row and column bits in error to not only detect that a bit is in error, but also identify the corrupted bit, and *correct* the error
  - Two dimensional parity schemes can also detect, but not correct, any combination of two errors in a single packet

    - The ability of the receiver to both detect and correct errors is known as *forward error correction (FEC)*
    - FEC techniques are valuable since they can decrease the number of sender transmissions required
    - It also avoids the round-trip propagation of ACK and NAK packets and retransmissions, which can prove integral to real-time web applications
- **6.2.2: Checksumming Methods**
    - In *checksumming techniques*, the d bits of data are treated as a sequence of k-bit integers
    - One simple checksumming method would be to sum these k-bit integers and use the resulting sum as the error-detection bits
        - The *Internet checksum* is based on this approach; bytes of data are treated as 16-bit integers and then summed
    - Checksumming methods require relatively little overhead
        - For example, the checksums in TCP and UDP use only 16-bits
    - However, they provide relatively weak protection against errors when compared to other error detection and correction methods that will be later discussed
- **6.2.3: Cyclic Redundancy Check (CRC)**
    - A widely used error detection technique in modern computer networks is based on *cyclic redundancy check (CRC) codes*
        - These are also called *polynomial codes*, since it is possible to view the bit string to be sent as a polynomial whose coefficients are the 0 and 1 values in the bit string, with operations on the bit string interpreted as polynomial arithmetic
    - **CRC codes operate as follows**
        - Consider the d-bit piece of data, D, that the sending node wants to send to the receiving node
        - The sender and receiver must first agree on an $r + 1$ bit pattern, known as a *generator* which will be denoted by G
        - The leftmost bit of G must be a 1

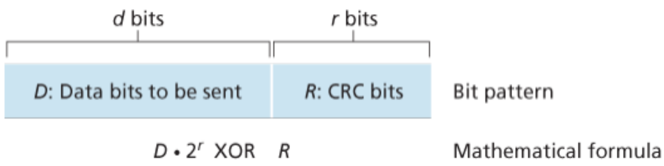
        - For D, the sender will choose r additional bits, R, and append them to D such that the resulting $d+r$ bit pattern is exactly divisible by G using modulo-2 arithmetic
        - 

Figure 6.6 CRC
        - The process of error checking is fairly simple with CRCs
            - The receiver computes:
                - $$\frac{d+r}{G} \, mod \, 2$$
            - If the result of this computation is nonzero, the receiver knows that an error has occurred, and otherwise the data is accepted as being correct
    - Given D and R, the formula $D \cdot 2^r \ XOR \ R$ yields the $d+r$ bit pattern shown in Figure 6.6
    - **How will the sender compute R?**
        - We want to find R such that there is an n which satisfies
        - $$D \cdot 2^r \oplus R = nG$$ $$D \cdot 2^r = nG \oplus R$$
        - Thus, if we divide $D \cdot 2^r$ by G, the value of the remainder is precisely R
        - So, we can calculate R as
        - $$R = remainder(\frac{D \cdot 2^r}{G})$$
    - Below is an example of the calculations described above
    -

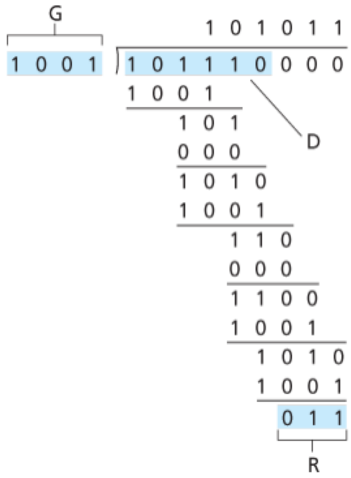


Figure 6.7 A sample CRC calculation

- International standards have been defined for 8, 12, 16, and 32-bit generators
- The CRC-32 32-bit standard, which has been adopted in a number of link layer IEEE protocols, uses a generator of $G_{CRC-32} = 100000100110000010001110110110111$
- Each of the CRC standards can detect burst errors of fewer than $r + 1$ bits
- Each of the CRC standards can also detect any odd number of bit errors

---

6.3: Multiple Access Links and Protocols

- There are two different types of network links
  - A *point-to-point* link consists of a single sender at one end of the link and a single receiver at the other end of the link
    - Many link-layer protocols have been designed for point-to-point links, such as the *point-to-point protocol (PPP)* and *high-level data link control (HDLC)*
  - The second type of link, a *broadcast link*, can have multiple sending and receiving nodes which are all connected to the same, single, shared broadcast channel
    - Here, the term *broadcast* is used here because when any one node transmits a frame, each of the other nodes will receive a copy
    - Ethernet and wireless LANs are examples of broadcast link-layer technologies
- Now, we must consider how to coordinate the access of multiple sending and receiving nodes to a shared broadcast channel; the *multiple access problem*

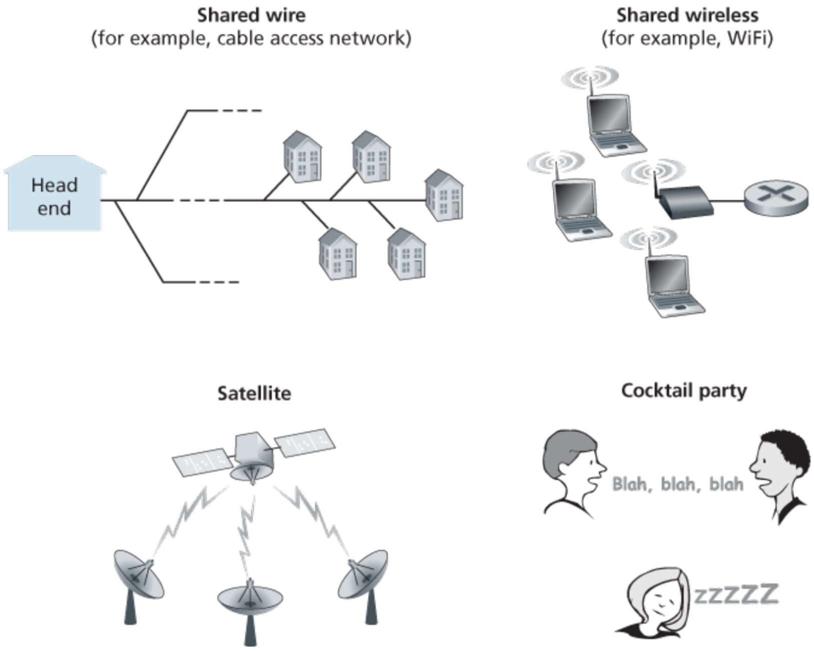
- Television has been using the idea of broadcasting since long before the advent of computers
  - However, traditional television is a one-way broadcast link, while nodes on a computer network broadcast channel can both send *and* receive
  - Networks have protocols known as *multiple access protocols*, by which nodes regulate their transmission into the shared broadcast channel
  - 

Figure 6.8 Various multiple access channels
- In practice, hundreds or even thousands of nodes can directly communicate over a broadcast channel
- Since all nodes are capable of transmitting frames, more than two nodes can transmit frames at the same time
- When this happens, all receiving nodes receive multiple frames at the same time, such that the transmitted frames *collide* at the receiver
- Typically, when there is a collision, none of the receiving nodes can make sense of any of the frames that were transmitted
- Clearly, if many nodes want to transmit frames frequently, many transmissions will result in collisions, and thus much of he channel's bandwidth will be wasted
- In order to ensure that the broadcast channel performs useful work when multiple nodes are active, we must somehow coordinate the transmissions of the active nodes

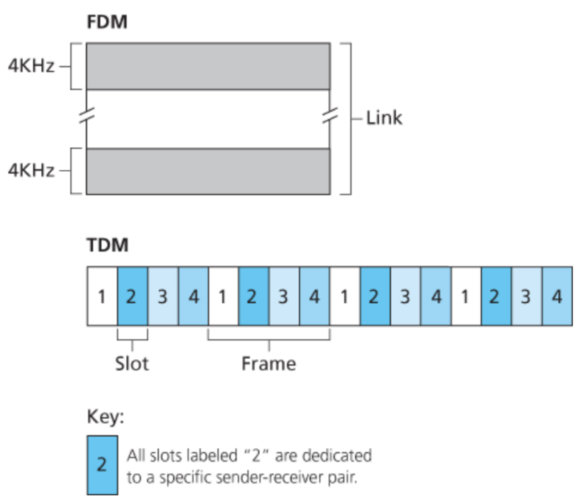
- This coordination is the job of the *multiple access protocol*
- Overall, dozens of multiple access protocols have been implemented in a variety of link-layer technologies, however all can generally be classified as belonging to one of three categories
  - *Channel partitioning protocols*
  - *Random access protocols*
  - *Taking-turns protocols*
- Ideally, a multiple access protocol for a broadcast channel of rate R bits per second should have the following desirable characteristics
  - When *only one* node has data to send, that node has a throughput of R bps
  - When M nodes have data to send, each node has a throughput of R/M bps on average (the instantaneous rates may not be constant)
  - The protocol is decentralized; that is, there is no master node that represents a single point of failure for the network
  - The protocol is simple such that it is inexpensive to implement
- **6.3.1: Channel Partitioning Protocols**
  - We must recall that time-division multiplexing and frequency-division multiplexing are two techniques which can be used to partition a broadcast channel's bandwidth among all nodes sharing that channel
  - 

Figure 6.9 A four-node TDM and FDM example

    - For example, assume a channel with rate R bps supports N nodes
        - Time-division multiplexing divides time into *time frames* and further divides each time frame into N *time slots*
        - Each slot will be assigned to a node, and when a node has a packet to send, it will do so during its allotted time slot in the TDM time frame
    - This scheme is appealing due to the fact that it eliminates the previously mentioned issue of collision and is perfectly fair between all nodes
    - Each node gets an average transmission rate of R/N bps over the course of each time frame
    - However, this structure also has two major drawbacks
        - A node is limited to a *maximum* transmission rate of R/N bps, even if it is the only node with packets to send over the R bps channel
        - A node also must wait for its allotted time slot, even when, again, it is the only node with any packets to be sent
    - While TDM shares the broadcast channel in terms of time, FDM divides the R bps channel into different frequencies, each with a bandwidth of R/N and assigns each frequency to one of the N nodes
    - FDM shares both the advantages and drawbacks with TDM
        - It avoids collisions and divides the bandwidth fairly among the N nodes
        - A node is limited to a *maximum* bandwidth of R/N, even when it is the only node with packets to be sent
    - A third channel partitioning protocol is *code division multiple access (CDMA)*
    - CDMA assigns a different *code* to each node, and each node then uses its unique code to encode the data bits it sends
    - CDMAs also have the unique property that different nodes can transmit simultaneously and yet have their respective receivers correctly receive a sender's encoded data bits
- **6.3.2: Random Access Protocols**
    - The second broad class of multiple access protocols are random access protocols

- In a random access protocol, a transmitting node always transmits at the full rate of the channel, namely, R bps
- If there is a collision, each node involved in the collision will repeatedly retransmit its frame until its frame gets thorough without a collision
- The node does not immediately retransmit the node, however, instead waiting a *random delay* before retransmitting the frame
- There are dozens, if not hundreds, random access protocols described in the literature
- We will describe some of the more commonly used ones in this section, starting with *Slotted ALOHA*
- **Slotted ALOHA**
  - The slotted ALOHA protocol is one of the simplest random access protocols
  - In slotted ALOHA, we will assume the following
    - All frames consist exactly of L bits
    - Time is divided into slots of size L/R seconds
      - This means a slot equals the time to transmit one frame
    - Nodes start to transmit frames only at the beginnings of slots
    - The nodes are synchronized so that each node knows when the slot begins
    - If two or more frames collide in a slot, then all the nodes detect the collision event before the slot ends
  - We then let p be a probability (a number between 0 and 1)
    - When the node has a fresh frame to send, it waits until the beginning of the next slot and transmits the entire frame in the slot
    - If there is no collision, the node has successfully transmitted its frame and thus need not consider retransmitting the frame
    - If there *is* a collision, the node detects the collision before the end of the slot, and the node retransmits its frame in each subsequent slot with probability p until the frame is transmitted without a collision
  - The probability p decides whether the node will retransmit the frame in that slot, or wait until the next slot to try a "biased coin-flip" deciding whether the node will be retransmitted in that frame
  - All nodes involved in a collision toss their coins independently
  - Unlike channel partitioning, slotted ALOHA allows a node to transmit continuously at the rate R if that node is the only active node
  -

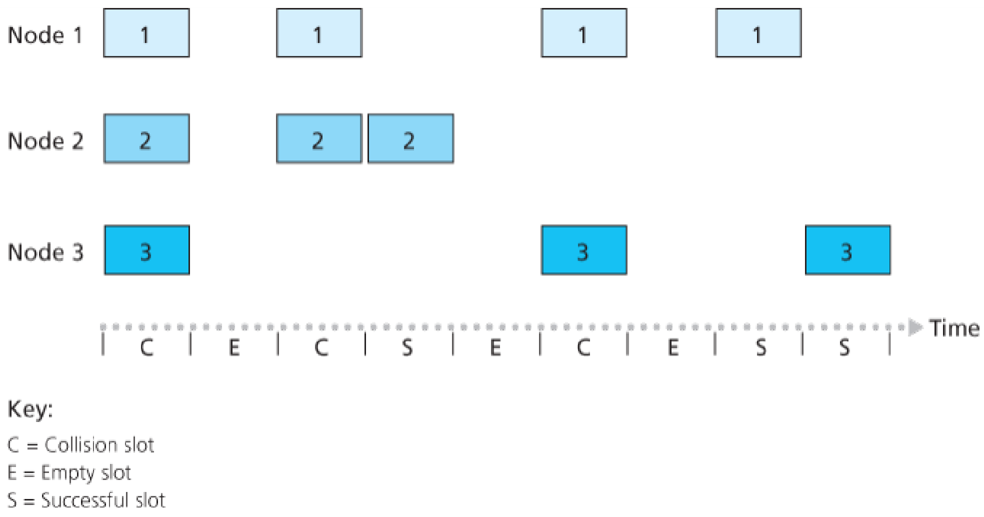


Figure 6.10 Nodes 1, 2, and 3 collide in the first slot. Node 2 finally succeeds in the fourth slot, node 1 in the eighth slot, and node 3 in the ninth slot

- Now, let us derive the maximum efficiency of slotted ALOHA
  - To simplify this derivation, we will modify the protocol such that every node always has a packet ready to transmit, and each frame, whether fresh or a retransmit, will be transmitted with probability p
  - Suppose there are N nodes
  - Then the probability that a given slot is successful is the probability that one of the nodes transmits, and the remaining $N-1$ nodes do not
    - The probability a node transmits is p
    - The probability that the remaining nodes do not is $(1-p)(N-1)$
  - Since there are N nodes, the probability that any one of the given nodes is successful is $Np(1-p)(N-1)$
  - In order to find the maximum efficiency of slotted ALOHA, you must find the value for p that maximizes the value of the above expression
    - This value ends up being $\frac{1}{e} \approx 0.37$, meaning only 37% of the nodes are doing useful work at maximum, making the effective transmission rate of this channel $0.37 * R$ bps

- ALOHA
  - As opposed to slotted ALOHA, the first ALOHA protocol was actually an unslotted, fully decentralized protocol
  - In pure ALOHA, when a frame first arrives, the node immediately transmits the frame in its entirety into the broadcast channel
  - If a transmitted frame experiences a collision with one or more other transmissions, the node will then immediately retransmit its frame with probability

p, or waits for a frame retransmission where it will once again transmit with probability p

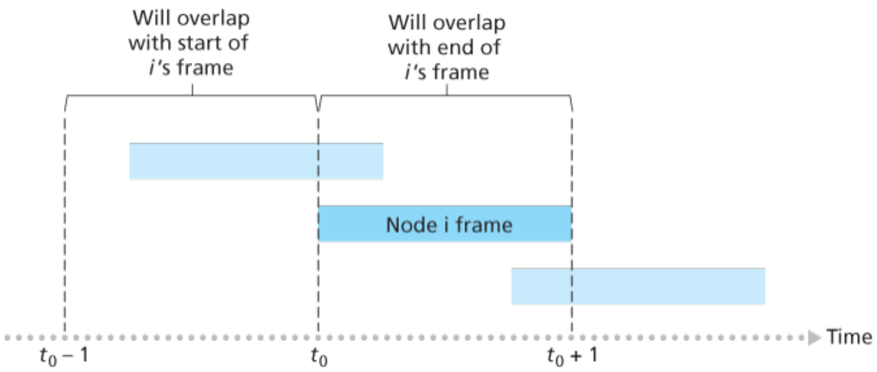
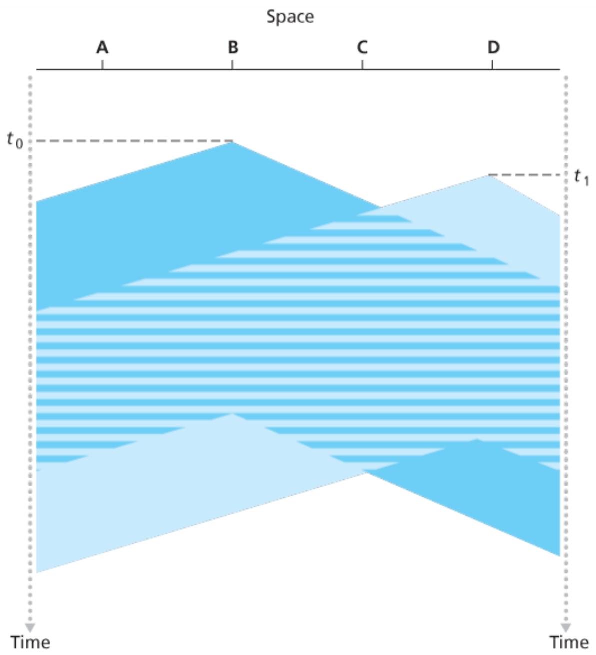
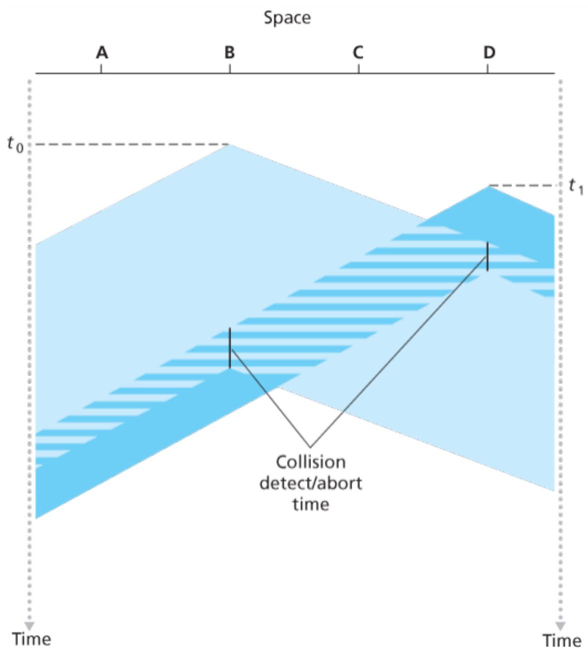
- To determine pure ALOHA's maximum efficiency, we will focus on an individual node
- At any given time, the probability that a node is transmitting a frame is p
- Suppose this frame begins transmission at time t_0
    - In order for this frame transmission to be successful, no other nodes can begin their transmission in the interval $[t_0 - 1, t_0]$, since such a transmission would overlap with the beginning of our node's frame
- By doing a derivation similar to that of slotted ALOHA, we can find that the maximum efficiency of the ALOHA protocol is $\frac{1}{2e} \approx 0.185$, which is exactly half that of the slotted ALOHA protocol, which is the major downside when compared to slotted ALOHA
- 

Figure 6.11 Interfering transmissions in pure ALOHA

- **Carrier Sense Multiple Access (CSMA)**
    - In both of the previous protocols, a node's decision to transmit is made independently of the activity of the other nodes attached to the broadcast channel
    - In the CSMA protocols, there are two important rules to limit collisions
        - *Carrier sensing*, such that a node will listen to the channel before transmitting and wait if a frame is being transmitted by another node
        - *Collision detection*, such that a transmitting node listens to the channel while it is transmitting, and if it detects another node sending an interfering frame, it will wait a random time interval and retransmit
    - With the property of carrier sensing in place, we then might ask how collisions can even occur in the first place
    - This can be best illustrated using a space-time diagram

- 

- At time t_0, node B senses the channel is idle and begins to transmit, with its bits propagating in both directions along the broadcast medium
- At time t_1, where $t_1 > t_0$, node D has a frame to send, and node B's message has yet to reach D and thus D believes the channel is open for its transmission
- In this example, no collision detection occurs
- Let us instead consider the example where collision detection is implemented in the protocol
- 

        - As we can see from the lower amount of interfering transmissions, this can assist in our channel's efficiency by reducing the amount of useless data sent over the channel
        - Let us now summarize the operation of the CSMA with collision detection protocol from the perspective of an adapter in a node attached to a broadcast channel
            - The adapter obtains a datagram from the network layer, prepares a link-layer frame, and puts the frame adapter buffer
            - If the adapter senses that the channel is idle, it starts to transmit the frame, otherwise the adapter will wait until it senses no signal energy and then begin transmission
            - While transmitting, the adapter monitors for the presence of signal energy coming from other adapters using the broadcast channel
            - If the adapter transmits the entire frame without detecting signal energy from other adapters, the adapter is finished with the frame
            - Otherwise, the adapter will abort the transmission, wait a random amount of time, and sense again whether or not the channel is idle
        - To calculate the efficiency of CSMA, we will let
            - d_{prop} be the maximum time it takes signal energy to propagate between any two adapters
            - d_{trans} be the time to transmit a maximum size frame
        - Then, we can approximate the efficiency, E, as
            - $$E = \frac{1}{1 + 5\alpha}$$
            - *where*,
                - $$\alpha = \frac{d_{prop}}{d_{trans}}$$
- **6.3.3: Taking-Turns Protocols**
    - There are many of these protocols, and each has many variations, but we will discuss two of the more important ones

- The first is the *polling protocol*, which requires one of the nodes in the broadcast channel to be designated as the *master node*
- The master node then *polls* each of the nodes in a round-robin fashion
  - This functions by telling node 1 it can transmit up to some maximum number of frames, and then telling node 2 the same once that amount is reached or it senses no signal energy on the channel
- One drawback of this protocol is that it introduces a *polling delay* since it requires time to notify a node that it can transmit
- The second taking-turns protocol is the *token-passing protocol*, which does not require the designation of a master node
- Instead, a small, special-purpose frame, known as the *token* is exchanged among the nodes in some fixed order, achieving a similar effect as the polling protocol

---