University of Science and Technology
Faculty of Computer Science and Information Technology
Computer Science Department
Computer Networking /Semester (7)
Lecture (4)

5.3.2 Random Access Protocols

The second broad class of multiple access protocols is **random access protocols**.
In a random access protocol, **a transmitting node always transmits at the full rate of the channel, namely, *R* bps.**

What happened when there is a collision?

When there is a collision, each node involved in the collision repeatedly retransmits its frame (that is, packet) until its frame gets through without a collision.

How nodes retransmit packets to avoid new collisions?

A node doesn't necessarily retransmit the frame right away. *Instead it waits a random delay before retransmitting the frame*. **Each node involved in a collision chooses independent random delays**. Because the random delays are independently chosen, it is *possible that one of the nodes will pick a delay that is sufficiently less than the delays of the other colliding nodes and will therefore be able to sneak its frame into the channel without a collision.*

In this section we'll describe a few of **the most commonly used random access** protocols—the **ALOHA** and the **Carrier Sense Multiple Access (CSMA)** protocols. **Ethernet deployed CSMA protocol.**

1. Slotted ALOHA

Let's begin our study of random access protocols with one of the simplest random access protocols, the slotted ALOHA protocol. In our description of slotted ALOHA, we assume the following:

1- All frames consist of exactly L bits.
2- Time is divided into slots of size L/R seconds (that is, a slot equals the time to transmit one frame).
3- Nodes start to transmit frames only at the beginnings of slots.
4- The nodes are **synchronized so that each node knows when the slots** begin.
5- If two or more frames collide in a slot, then all the nodes detect the collision event before the slot ends.

How a node transmit a packet using ALOHA

Let *p* be a probability, that is, a number between 0 and 1. The operation of slotted ALOHA in each node is simple:

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.

2- If there isn't a collision, the node has successfully transmitted its frame and thus need not consider retransmitting the frame. (The node can prepare a new frame for transmission, if it has one.)

3- If there is a collision, the node detects the collision before the end of the slot. The node retransmits its frame in each subsequent slot with probability p until the frame is transmitted without a collision.

By retransmitting with probability *p*, we mean that the node effectively tosses a biased coin; the event heads corresponds to "retransmit," which occurs with probability *p*. The event tails corresponds to "skip the slot and toss the coin again in the next slot"; **All nodes involved in the collision toss their coins independently**.

Advantages of Slotted ALOHA

1-Unlike channel partitioning, slotted ALOHA allows a node to transmit continuously at the full rate, *R*, when that node is the only active node.

2-Slotted ALOHA is **also highly decentralized**, because each node detects collisions and independently decides when to retransmit.

Disadvantage

1-Slotted ALOHA does, however, require the slots to **be synchronized** in the nodes.

When Slotted ALOHA does work well?

Slotted ALOHA works well when there is only one active node in the network.

How efficient is it when there are multiple active nodes?

When there are multiple active nodes, there are two possible efficiency concerns:

1- A certain fraction of the slots will have collisions and will therefore be "wasted."

2- Another fraction of the slots will be *empty* because all active nodes refrain from transmitting as a result of the probabilistic transmission policy.

See Figure 5.10 below.

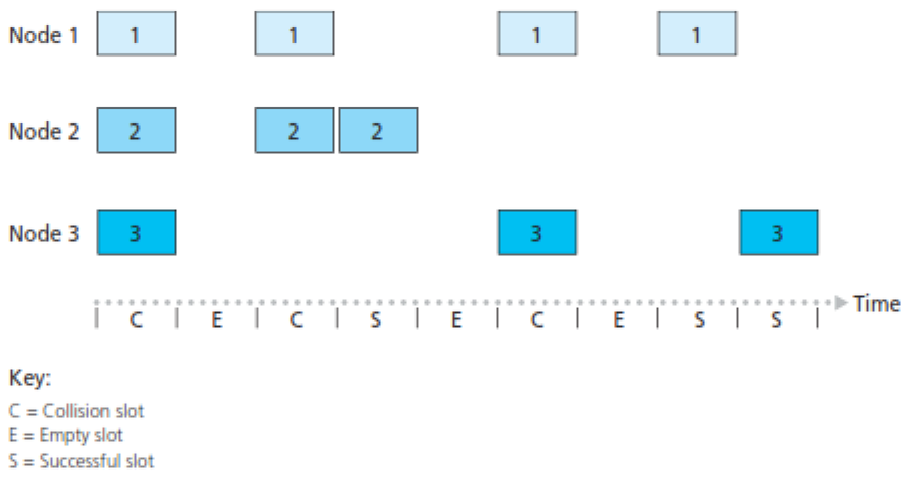

Figure 5.10 ♦ Nodes 1, 2, and 3 collide in the first slot. Node 2 finally succeeds in the fourth slot, node 1 in the eighth slot, and node 3 in the ninth slot

---

To avoid the synchronize feature in Slotted ALOHA a new Aloha protocol is proposed.

2. Pure ALOHA Protocol

Pure ALOHA protocol is an unslotted, fully decentralized protocol. It works as follows:

1- When a frame first arrives (that is, a network-layer datagram is passed down from the network layer at the sending node), the node **immediately transmits** the frame in its entirety into the broadcast channel.

2- If a transmitted frame experiences a collision with one or more other transmissions, the node will then immediately (after completely transmitting its collided frame) retransmit the frame with probability *p*.

3- Otherwise, the node waits for a frame transmission time. After this wait, it then transmits the next frame with probability *p*, or waits (remaining idle) for another frame time.

3. Carrier Sense Multiple Access (CSMA)

In both slotted and pure ALOHA, a node's decision to transmit is made independently of the activity of the other nodes attached to the broadcast channel.

In particular, a node neither:

- pays attention to whether another node happens to be transmitting when it begins to transmit, nor

-stops transmitting if another node begins to interfere with its transmission.

As humans, we have human protocols that allow us not only to behave with more civility, but also to decrease the amount of time spent "colliding" with each other in conversation and, consequently, to increase the amount of data we exchange in our conversations. **Specifically, there are two important rules for polite human conversation:**

- ***Listen before speaking***. If someone else is speaking, wait until they are finished. In the networking world, this is called **carrier sensing**—a node listens to the channel before transmitting. If a frame from another node is currently being transmitted into the channel, a node then waits until it detects no transmissions for a short amount of time and then begins transmission.

- ***If someone else begins talking at the same time, stop talking***. In the networking world, this is called **collision detection**—a transmitting node listens to the channel while it is transmitting. If it detects that another node is transmitting an interfering_frame, it *stops transmitting and waits a random amount of time before repeatin*g the sense-and-transmit-when-idle cycle.

These two rules are embodied in the family of **carrier sense multiple access (CSMA)** and **CSMA with collision detection (CSMA/CD)** protocols. Many variations on CSMA and CSMA/CD have been proposed. **Here, we'll consider a few of the most important, and fundamental, characteristics of CSMA and CSMA/CD**.

CSMA

In CSMA, a node will refrain (stop) from transmitting whenever it senses that another node is transmitting.
So, the first question that you might ask about CSMA is:

Why, if all nodes perform carrier sensing, do collisions occur in the first place?

The answer to the question can best be illustrated using space-time diagrams. Figure 5.12 shows a space-time diagram of four nodes (A, B, C, D) attached to a linear broadcast bus. The horizontal axis shows the position of each node in space; the vertical axis represents time.

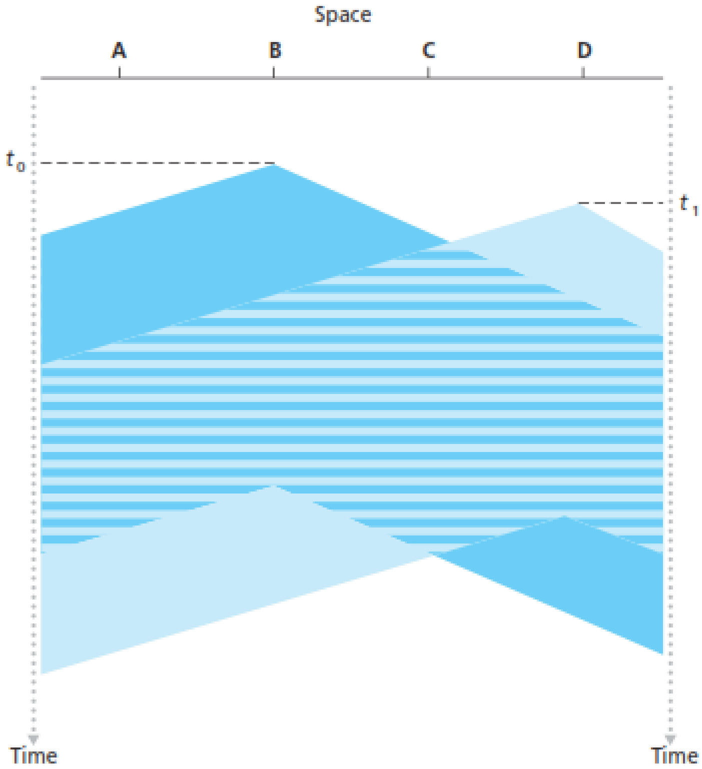

Figure 5.12 ♦ Space-time diagram of two CSMA nodes with colliding transmissions

---

- At time t_0, node B senses the channel is idle, as no other nodes are currently transmitting. Node B thus begins transmitting, with its bits propagating in both directions along the broadcast medium.
- At time t_1 ($t_1 > t_0$), node D has a frame to send. Although node B is currently transmitting at time t_0, the bits being transmitted by B have not yet reach D, and thus D senses the channel idle at t_1. In accordance with the CSMA protocol, D thus begins transmitting its frame.
- A short time later, B's transmission begins to interfere with D's transmission.

From Figure 5.12, it is evident that the end-to-end **channel propagation delay** of a broadcast channel—**the time it takes for a signal to propagate from one of the nodes to another**—will play a crucial role in determining its performance.
It is clear that, **the longer the propagation** delay, **the larger the chance** that a carrier-sensing node is not yet able to sense a transmission that has already begun at another node in the network.

In Figure 5.12, nodes do not perform collision detection; both B and D continue to transmit their frames in their entirety even though a collision has occurred.

When a node performs collision detection, it stop transmission as soon as it detects a collision. Figure 5.13 shows the same scenario as in Figure 5.12, except that the two nodes each abort their transmission a short time after detecting a collision.

.

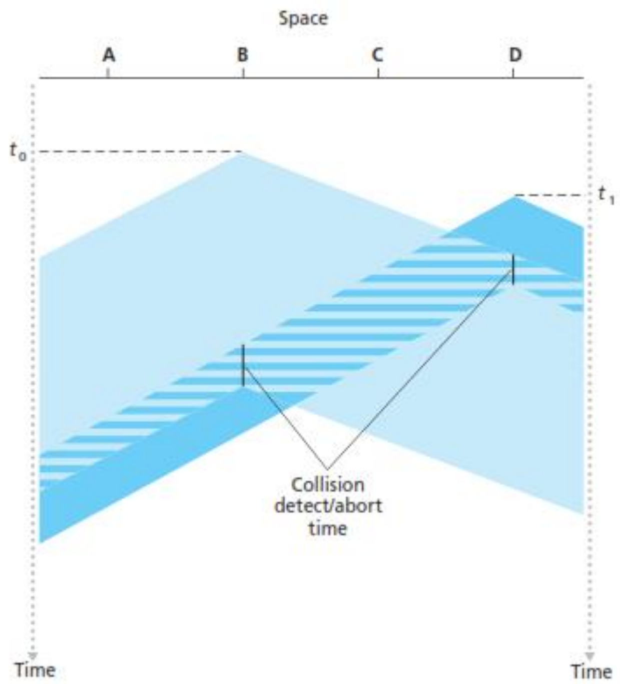

Figure 5.13 ♦ CSMA with collision detection

What is the benefit of Collision Detection?

Clearly, adding collision detection to a multiple access protocol will increase the protocol performance by not transmitting a useless, damaged (by interference with a frame from another node) frame in its entirety.

Carrier Sense Multiple Access with Collision Detection (CSMA/CD)

This protocol is CSMA with collision detection capabilities.

How an adapter works with CSMA/CD?

The adapter works as follows:

1. The adapter obtains a datagram from the network layer, prepares a link-layer frame, and puts the frame in the adapter buffer.

2. If the adapter senses that the channel is idle (that is, **there is no signal energy entering the adapter from the channel**), it starts to transmit the frame.

Else if the adapter senses that the channel is busy, it waits until it senses **no signal energy and then** starts to transmit the frame.

3. While transmitting, the adapter monitors for the presence of signal energy coming from other adapters using the broadcast channel.

4. If the adapter transmits the entire frame without detecting signal energy from other adapters, the adapter is finished with the frame.

If, on the other hand, the adapter detects signal energy from other adapters while transmitting, it aborts the transmission (that is, it stops transmitting its frame).

5. After aborting, the adapter waits a random amount of time and then returns to step 2.

The Question now is:

What is the suitable interval time the adapter has to wait?

The need to wait a random (rather than fixed) amount of time is hopefully clear, if two nodes transmitted frames at the same time and then both waited the same fixed amount of time, they'd continue colliding forever.

But what is a good interval of time from which to choose the random back off time?

If the **interval is large and the number of colliding nodes is small**, nodes are likely to wait a large amount of time (**with the channel remaining idle**) before repeating the sense-and-transmit.

On the other hand, if **the interval is small and the number of colliding nodes is large**, it's likely that the chosen random values will be nearly the same, and transmitting **nodes will again collide**.

What we'd like is an **interval that is short when the number of colliding nodes is small, and long when the number of colliding nodes is large.**

The **binary exponential backoff** algorithm, used in Ethernet solves this problem.

CSMA/CD Efficiency

When only one node has a frame to send, the node can transmit at the full channel rate (e.g., for Ethernet typical rates are 10 Mbps, 100 Mbps, or 1 Gbps). However, if many nodes have frames to transmit, the effective transmission rate of the channel can be much less.

We define the **efficiency of CSMA/CD** **to be the fraction of time during which frames are being transmitted on the channel without collisions when there is a large number of active nodes, with each node having a large number of frames to send.**

In order to present an approximation of the efficiency of Ethernet:

- let d_{prop} denote the maximum time signal energy takes to propagate between any two adapters.

- Let d_{trans} be the time to transmit a maximum-size frame (approximately 1.2 msecs for a 10 Mbps Ethernet).

Then the efficiency of CSMA/CD is given by the following approximation:

$$\text{Efficiency} = \frac{1}{1 + 5d_{prop}/d_{trans}}$$

From this formula, we see that as d_{prop} approaches 0, the efficiency approaches 1. This matches the fact that if the **propagation delay is zero, then all nodes (adapters) senses signal energy coming at the same time, and no collision occur or colliding nodes will abort immediately without wasting the channel**.

Also, as d_{trans} becomes very large, efficiency approaches 1. This is also intuitive because when a frame **grabs the channel, it will hold on to the channel for a very long time; thus, the channel will** be doing productive work most of the time.