Multiple Access Protocols for Link Layer

Multiple Access Protocols

- **Single shared broadcast channel**
- **Two or more simultaneous transmissions by nodes: interference**
  - collision if node receives two or more signals at the same time

<u>Multiple access protocol</u>

- **distributed algorithm that determines how nodes share channel, i.e., determine when node can transmit**
- **communication about channel sharing must use channel itself!**
  - Typically no out-of-band channel for coordination

Ideally what we want?

<u>**Broadcast channel of rate R bps**</u>

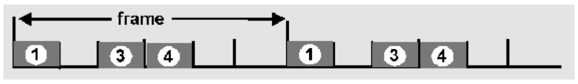
- **1. When one node wants to transmit, it can send at rate R.**
- **2. When M nodes want to transmit, each can send at "long-run average" rate R/M**
- **3. Fully decentralized:**
  - no special node to coordinate transmissions
  - no synchronization of clocks, slots

Types of Multiple Access

- **"Channelization"**
  - Divide channel into smaller "pieces" (time slots, frequency, code etc.)
  - Allocate a piece to node for exclusive use
- **Random Access**
  - channel not divided
  - prevent collisions, or allow and recover from collisions
- **"Taking turns"**
  - Nodes take turns, but nodes with more to send can take longer turns

Channelization

- Examples: Time-division multiple access (TDMA) or Frequency-division multiple acesss (FDMA).
- Take TDMA example:



- Unused slots go idle.
- Inefficient when load is unbalanced or unknown.

Random Access Protocols

- No *a priori* "pieces" (slots, frequency or code).
- Packets are transmitted with little or no coordination. Collisions possible.
- Issue: how to prevent collisions, or recover from collisions?
- Techniques
  - Aloha (slotted and unslotted)
  - CSMA, CSMA/CD, CSMA/CA

Aloha Protocols

- Initially developed in 1970 for radio communication between a computer and several terminals in Univ. of Hawai.
- Mother of all random access protocols, even though very inefficient.
- We present an idealized version for easy analysis.

Slotted Aloha

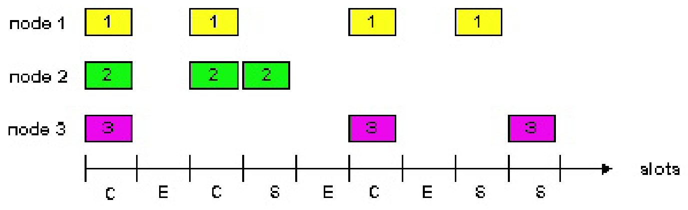
Assumptions

- All frames same size
- Time is divided into equal size slots, time to transmit 1 frame
- Nodes start to transmit frames only at beginning of slots
- Slots are synchronized
- Collision = more than one node transmit in a slot.

Operation

- When node has a new frame to transmit, it transmits in the next slot
- If collision, node retransmits frame in each subsequent slot with prob. p until success

Slotted ALOHA



Pros

- single active node can continuously transmit at full rate of channel
- highly decentralized: only slots in nodes need to be in sync
- simple

Cons

- collisions, wasting slots
- idle slots
- nodes may be able to detect collision in less than time to transmit packet
- clock synchronization

Slotted Aloha efficiency

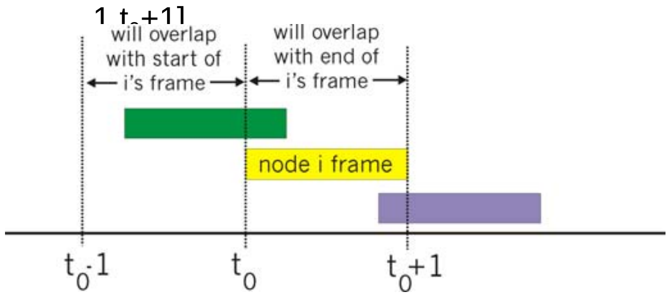
Efficiency is the long-run fraction of successful slots when there are many nodes, each with many frames to send

- Suppose N nodes with many frames to send, each transmits in slot with probability *p*
- prob that node 1 has success in a slot $= p(1-p)^{N-1}$
- prob that any node has a success $= Np(1-p)^{N-1}$
- For max efficiency with N nodes, find p* that maximizes $Np(1-p)^{N-1}$
- For many nodes, take limit of $Np^*(1-p^*)^{N-1}$ as N goes to infinity, gives 1/e = .37

At best: channel used for useful transmissions 37% of time!

Pure (unslotted) ALOHA

- **unslotted Aloha: simpler, no synchronization**
- **when frame first arrives**
  - transmit immediately
- **collision probability increases:**
  - frame sent at t_0 collides with other frames sent in $[t_0-1, t_0+1]$


will overlap with start of i's frame
will overlap with end of i's frame
node i frame
t_0-1
t_0
t_0+1

Pure Aloha efficiency

P(success by given node) = P(node transmits) ·

P(no other node transmits in $[p_0-1, p_0]$ ·

P(no other node transmits in $[p_0-1, p_0]$

$$= p \cdot (1-p)^{N-1} \cdot (1-p)^{N-1}$$

$$= p \cdot (1-p)^{2(N-1)}$$

… choosing optimum p and then letting n -> infty ...

Even worse ! **= 1/(2e) = .18**

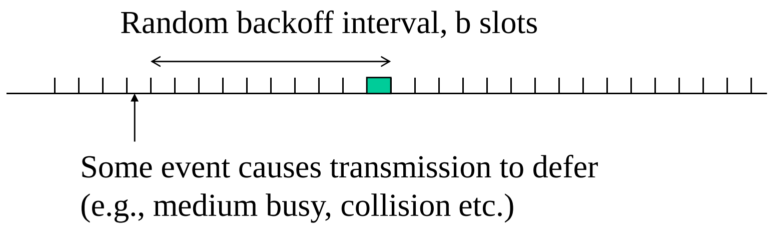
Carrier Sense Multiple Access (CSMA)

- **In Aloha, nodes transmit immediately when a new packet arrives.**
- **What if we are able to sense carrier and tell busy and idle periods apart?**
- **Busy carrier indicates ongoing transmission. Wait until carrier idle. Typically, needs a "carrier sense interval" for detecting the state of the carrier.**
- **Can be used in both slotted and unslotted systems.**

CSMA : Persistence

- **New arrival senses busy carrier**
  - Continue sensing until idle (persistent CSMA).
  - Retry after a random time interval (non-persistent CSMA). Retry again if still busy.
- **Two types of persistent CSMA**
  - Transmit new arrival as soon as carrier becomes idle (1-persistent CSMA).
  - Transmit new arrival after a random time interval after carrier becomes idle (p-persistent CSMA).

Note on Backoff


Random backoff interval, b slots
Some event causes transmission to defer
(e.g., medium busy, collision etc.)

- **Trying to transmit in subsequent slots with some probability p is equivalent to sending after a random interval.**
- **Length of random interval in slots = b.**
- **Prob [b=k] = $p(1-p)^{(k-1)}$**

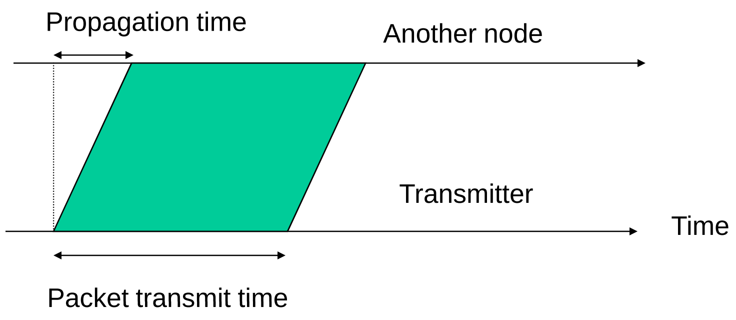
Collision Detection

- **Need to know whether transmission is successful.**
- **First idea: Detect collision. Retransmit after backoff (as in Aloha) if collision.**
- **CSMA/CD – 1-persistent CSMA with collision detection. Used in Ethernet. IEEE LAN standard 802.3.**
- **Listen while transmitting. If notice transmission being garbled, flag collision.**

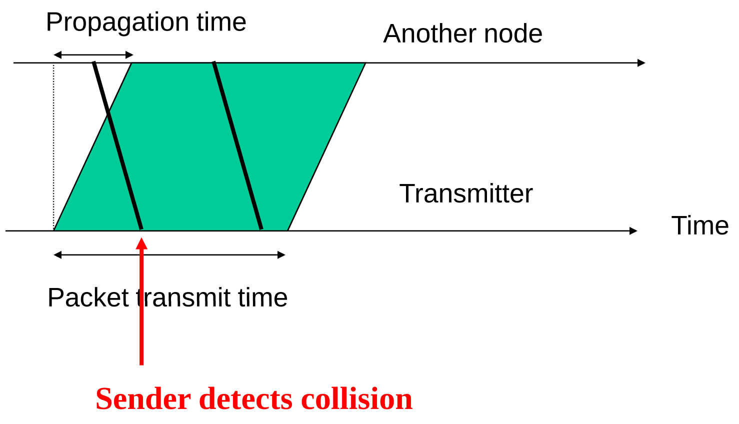
Ethernet CSMA/CD

- **Important parameter a = max. signal propagation delay in network / smallest packet transmission time.**
- **Packet transmission time = packet size / channel bandwidth.**
- **Parameter a must be less than 0.5 so that the packet lasts long enough at the transmitter for it be able to detect collision if it happens *anywhere* in the network.**

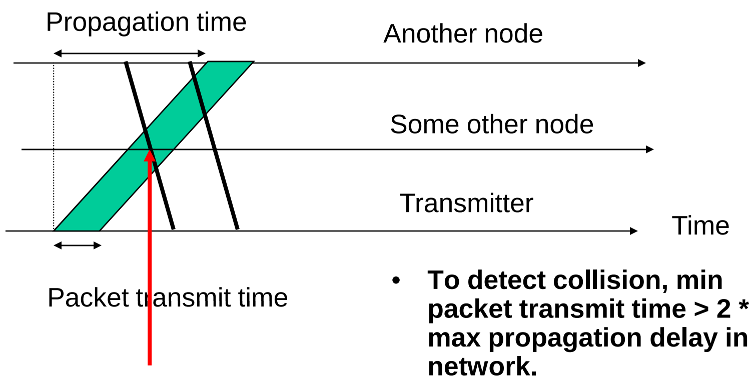
Collision Detection



Collision Detection



Collision Detection



Collision at some node. Sender unable to detect.

- **To detect collision, min packet transmit time > 2 * max propagation delay in network.**
- **Thus Ethernet has a max "cable" length and a minimum packet size.**

Exponential Backoff In Ethernet

- **Transmitter transmits a “jam” signal after collision is detected to make sure every transmitter aborts transmission.**
- **Then it enters an exponential backoff phase. Retransmits the colliding packet after 512*K bit time.**
  - K is random from $\{0,1,2,3,\ldots,2^{m-1}\}$, m = min(n,10).
  - n = no. of collisions for this packet.

Performance Metrics

- **Throughput: Bits/sec or packets/sec received correctly at the receiver.**
- **Offered load: Bits/sec or packets/sec actually transmitted on the network.**
- **Normalized throughput or load: The above normalized with respect to bandwidth.**
- **Delay: Delay between when the first bit of the packet starts transmitting for the first time and the when the last bit of the packet correctly received.**

Less Common Metrics

- **Stability: Whether throughput decreases with increase in load.**
  - Could happen in a random access scheme, if collision increases with increase in load.
  - Ideally, the protocol should try to admit less load if overload (frequent collisions) is detected.
- **Fairness: Whether all nodes with packets to transmit have equal shots at transmitting. i.e., all contending nodes get "equitable" share of the bandwidth; nobody "hogs" the channel.**

CSMA/CD Performance

- **Much harder to analyze analytically than Aloha.**
- **Max. throughput is very dependent on parameter a = ratio of prop and transmit time.**
  - With some conservative assumptions, max. normalized throughput = 1/(1+constant*a).
  - Very small a close to zero -> almost ideal throughput. Large "a" means more waste before collision detect. Bad for CSMA/CD.
  - constant ≈ 5 for Ethernet LAN. Find 'a' from Ethernet specs.

CSMA on Wireless

- **Cannot depend on collision detection ability.**
  - Full duplex radios are complex and expensive.
  - Even if sender is able to detect collision, it may so happen that sender senses no collision, but collision happens at the receiver. This is due to path loss of wireless signals. Doesn't happen in Ethernet.

Busy Tone Multiple Access (BTMA)

- **Divide the available frequency band into two channels: message (data) channel and control channel.**
- **When a receiver receives a packet, it sets off a tone on the control channel. Any "hidden" terminal upon hearing this "busy tone" will refrain from transmitting.**
- **Carrier sense on the busy tone only. Solves both hidden and exposed terminal problems.**
- **Problems:**
  - Use of two channels makes radios complex. Also, channels may need to well-separated.
  - Propagation characteristics on two channels may be different.

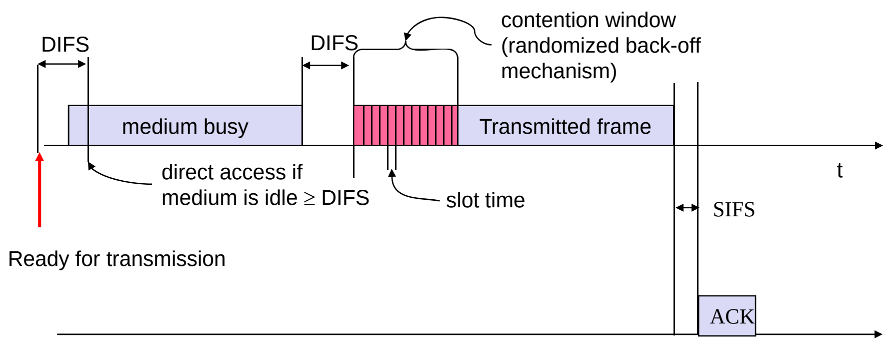
CSMA/CA, IEEE 802.11 WLAN

- CSMA with collision avoidance. A form of p-persistent CSMA.
- Used in IEEE standard 802.11 for Wireless LAN.
- Transmit after the channel is continuously sensed idle for an amount of time equal to or greater than the Distributed Inter Frame Space (DIFS).
- If channel is busy, defer until the channel is sensed idle for DIFS duration. Then go through an additional random backoff much like Ethernet. Transmit after the backoff timer expires.
- The timer counts down as long as the channel is idle, but stops counting during the time the channel is busy.

CSMA/CA Contd.

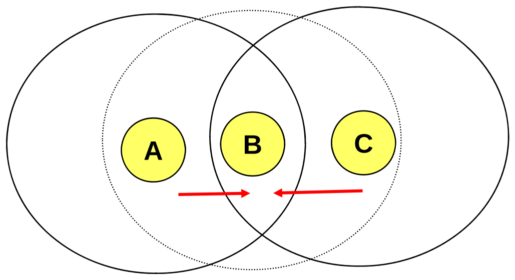
- Avoids collision by randomization, but can't eliminate it. So, still need feedback from receiver unlike CSMA/CD.
- Receiver sends a short ACK packet after a SIFS (Short Inter Frame Spacing) period after correctly receiving a packet.
- No ACK -> sender retransmits, but the max value of backoff timer is now doubled much like Ethernet (exponential backoff).

CSMA/CA in 802.11



- **Backoff clock stops if medium is sensed busy.**
- **DIFS > SIFS**

Hidden Terminal Problem



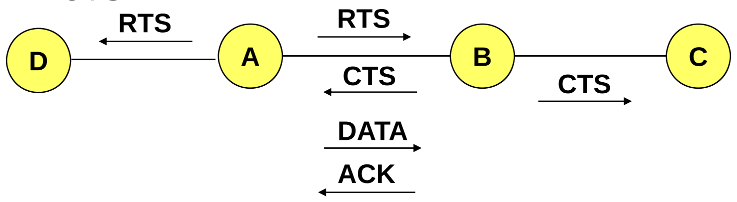
- **A and C cannot hear each other.**
- **When A transmits to B, C cannot detect the transmission using carrier sense.**
- **If C transmits as well, packets collide at B.**
- **A and C are “hidden” from each other.**

Hidden Terminal Problem

- **Cannot happen in wired Ethernet. Peculiar to wireless!**
- **Need to sense carrier at receiver, not sender!**
- **Solution: Do "virtual carrier sensing". Ask receiver whether it can hear anything. If it does, behave as if channel busy.**

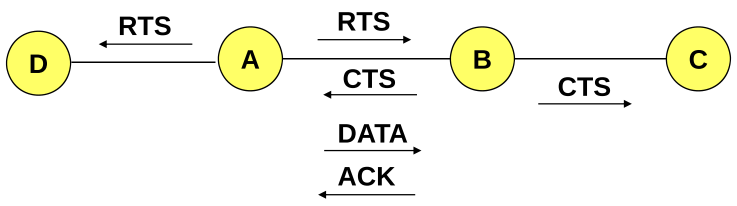
Solving Hidden Terminal Problem: RTS-CTS Handshake

- **When A wants to send a packet to B, node A first sends a short *Request-to-Send (RTS)* packet to B.**
- **On receiving RTS, node B responds by sending *Clear-to-Send (CTS).***
- **When a "hidden" node (e.g., C) overhears a CTS, it keeps quiet for the duration of the whole transfer.**
  - Transfer duration is included in both RTS and CTS.

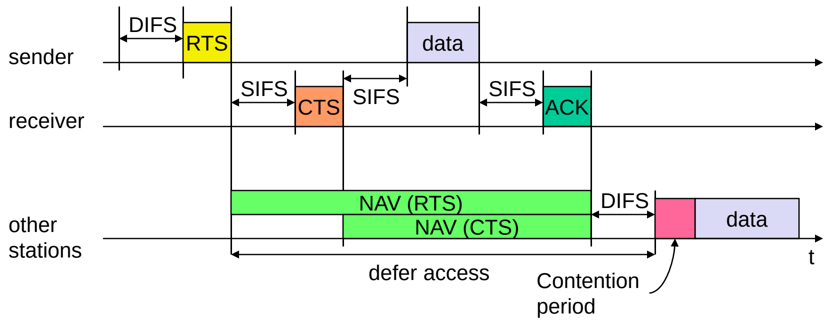


RTS-CTS Handshake Contd

- Note: Any node that hears RTS or CTS must keep quiet for the duration of the entire transfer (until end of ACK).
- Note the RTS packets may collide. Probability small as they are short. Retransmit RTS after backoff if no CTS.

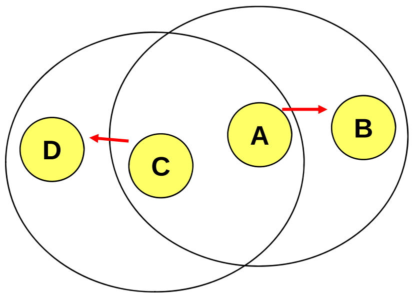


Timeline



- Note Data > RTS/CTS/ACK > DIFS > SIFS. Times not to scale for clarity.
- NAV = network allocation vector. During the period NAV is set the node must act as if medium busy. Thus a packet from them ready during this time must go through a contention period.

Exposed Terminal Problem



- In principle, A->B and C->D transmissions can go in parallel without collisions.
- But A and C can hear each other. C will wait for A->B to end before starting C->D.
- A and C are “exposed” terminals.
- Solutions??