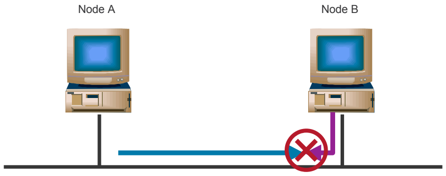
IEEE 802.11, Token Rings

Medium Access Control

- Wireless channel is a shared medium
- Need access control mechanism to avoid interference
- Why not CSMA/CD?

Ethernet MAC Algorithm



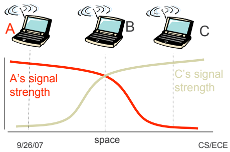
- Listen for carrier sense before transmitting
- Collision: What you hear is not what you sent!

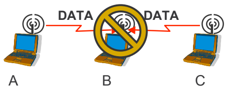
CSMA/CD in WLANs?

- Most (if not all) radios are half-duplex
  - Listening while transmitting is not possible
- Collision might not occur at sender
  - Collision at receiver might not be detected by sender!

Hidden Terminal Problem

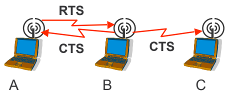
- Node B can communicate with both A and C
- A and C cannot hear each other
- When A transmits to B, C cannot detect the transmission using the carrier sense mechanism
- If C transmits, collision will occur at node B

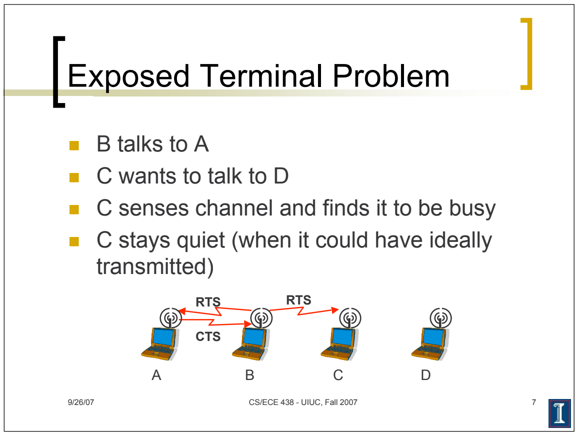




MACA Solution for Hidden Terminal Problem

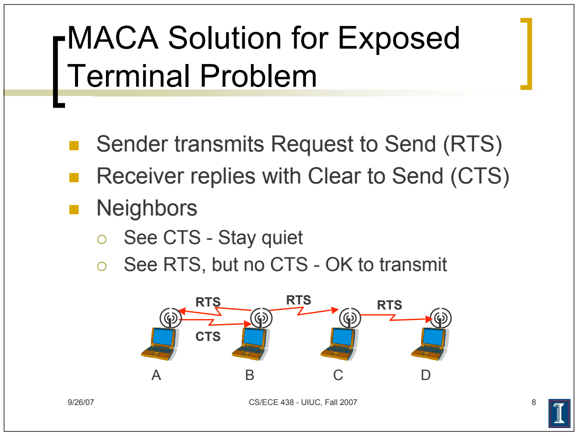
- When node A wants to send a packet to node B
  - Node A first sends a Request-to-Send (RTS) to A
- On receiving RTS
  - Node A responds by sending Clear-to-Send (CTS)
  - provided node A is able to receive the packet
- When a node C overhears a CTS, it keeps quiet for the duration of the transfer





Exposed Terminal Problem

- B talks to A
- C wants to talk to D
- C senses channel and finds it to be busy
- C stays quiet (when it could have ideally transmitted)



MACA Solution for Exposed Terminal Problem

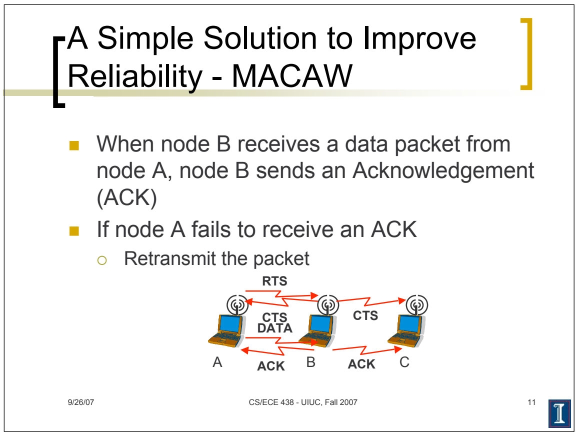
- Sender transmits Request to Send (RTS)
- Receiver replies with Clear to Send (CTS)
- Neighbors
  - See CTS - Stay quiet
  - See RTS, but no CTS - OK to transmit

Collisions

- Still possible
  - RTS packets can collide!
- Binary exponential backoff
  - Backoff counter doubles after every collision and reset to minimum value after successful transmission
  - Performed by stations that experience RTS collisions
- RTS collisions not as bad as data collisions in CSMA
  - Since RTS packets are typically much smaller than DATA packets

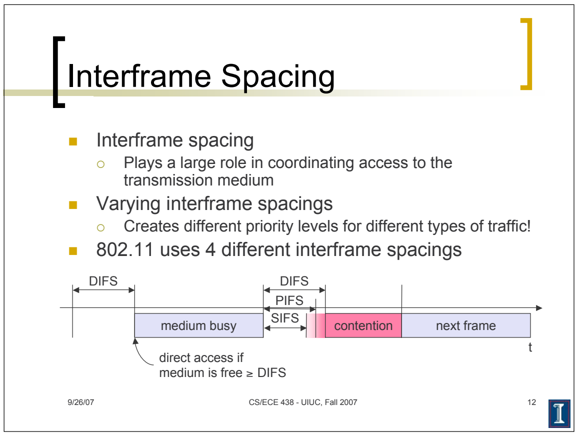
Reliability

- Wireless links are prone to errors
  - High packet loss rate detrimental to transport-layer performance
- Mechanisms needed to reduce packet loss rate experienced by upper layers



A Simple Solution to Improve Reliability - MACAW

- When node B receives a data packet from node A, node B sends an Acknowledgement (ACK)
- If node A fails to receive an ACK
  - Retransmit the packet

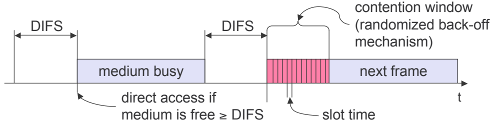


Interframe Spacing

- Interframe spacing
  - Plays a large role in coordinating access to the transmission medium
- Varying interframe spacings
  - Creates different priority levels for different types of traffic!
- 802.11 uses 4 different interframe spacings

IEEE 802.11 - CSMA/CA

- Sensing the medium
- If free for an Inter-Frame Space (IFS)
  - Station can start sending (IFS depends on service type)
- If busy
  - Station waits for a free IFS, then waits a random back-off time (collision avoidance, multiple of slot-time)
- If another station transmits during back-off time
  - The back-off timer stops (fairness)



Types of IFS

- SIFS
  - Short interframe space
  - Used for highest priority transmissions
  - RTS/CTS frames and ACKs
- DIFS
  - DCF interframe space
  - Minimum idle time for contention-based services (> SIFS)

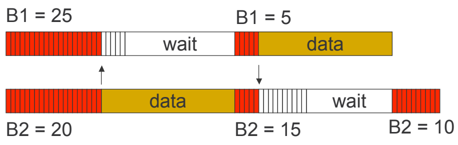
Types of IFS

- PIFS
  - PCF interframe space
  - Minimum idle time for contention-free service (>SIFS, <DIFS)
- EIFS
  - Extended interframe space
  - Used when there is an error in transmission

Backoff Interval

- When transmitting a packet, choose a backoff interval in the range [0,cw]
  - cw is contention window
- Count down the backoff interval when medium is idle
  - Count-down is suspended if medium becomes busy
- When backoff interval reaches 0, transmit RTS

DCF Example



cw = 31

B1 and B2 are backoff intervals at nodes 1 and 2

Backoff Interval

- The time spent counting down backoff intervals is a part of MAC overhead
- Large cw
  - Large backoff intervals
  - Can result in larger overhead
- Small cw
  - larger number of collisions (when two nodes count down to 0 simultaneously)

Backoff Interval

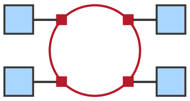
- The number of nodes attempting to transmit simultaneously may change with time
  - Some mechanism to manage contention is needed
- IEEE 802.11 DCF
  - Contention window cw is chosen dynamically depending on collision occurrence

Binary Exponential Backoff in DCF

- When a node fails to receive CTS in response to its RTS, it increases the contention window
  - cw is doubled (up to an upper bound)
- When a node successfully completes a data transfer, it restores cw to Cw_{min}
  - cw follows a sawtooth curve

Token Ring

- Example Token Ring Networks
  - IBM: 4Mbps token ring
  - IEEE 802.5: 16Mbps



Token Ring

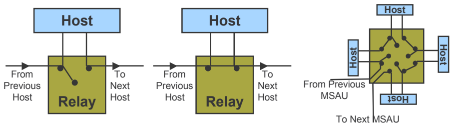
- Focus on Fiber Distributed Data Interface (FDDI)
  - 100 Mbps
  - Was (not is) a candidate to replace Ethernet
  - Used in some MAN backbones (LAN interconnects)
- Outline
  - Rationale
  - Topologies and components
  - MAC algorithm
  - Priority
  - Feedback
  - Token management

Token Ring

- Why emulate a shared medium with point-to-point links?
- Why a shared medium?
  - Convenient broadcast capabilities
  - Switches costly
- Why emulation?
  - Simpler MAC algorithm
  - Fairer access arbitration
  - Fully digital (802.3 collision detection requires analog)

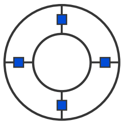
Token Ring: Topology and Components

- Relay
  - Single Relay
  - Multistation access units



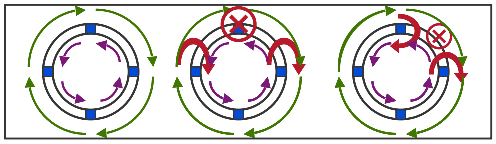
Token Ring: Dual Ring

- Example Token Ring Networks
  - FDDI:		1000Mbps
    - Fiber Distributed Data Interface



FDDI

- Dual ring configuration
  - Self-healing
    - Normal flow in green direction
    - Can detect and recover from one failure



Multistation Access Unit

- Each station imposes a delay
  - E.g. 50 ms
- Maximum of 500 Stations
- Upper limit of 100km
  - Need 200km of fiber
- Uses 4B/5B encoding
- Can be implemented over copper

Token Ring: Basic Concepts

- Frames flow in one direction
  - Upstream to downstream
- Token
  - Special bit pattern rotates around ring
- Stations
  - Must capture token before transmitting
  - Must remove frame after it has cycled
  - Must release token after transmitting
- Service
  - Stations get round-robin service

Token Ring: Basic Concepts

- Immediate release
  - Used in FDDI
  - Token follows last frame immediately
- Delayed release
  - Used in IEEE 802.5
  - Token sent after last frame returns to sender

Token Release

Token Ring: Media Access Control Parameters

- Token Holding Time (THT)
  - Upper limit on how long a station can hold the token
  - Each station is responsible for ensuring that the transmission time for its packet will not exceed THT
- Token Rotation Time (TRT)
  - How long it takes the token to traverse the ring.
  - TRT ≤ ActiveNodes x THT + RingLatency
- Target Token Rotation Time (TTRT)
  - Agreed-upon upper bound on TRT

802.5 Reliability

- Delivery status
  - Trailer
    - A bit
      - Set by recipient at start of reception
    - C bit
      - Set by recipient on completion on reception

802.5 Monitor

- Responsible for
  - Inserting delay
  - Token presence
    - Should see a token at least once per TRT
  - Check for corrupted frames
  - Check for orphaned frames
    - Header
      - Monitor bit
        - Monitor station sets bit first time it sees packet
        - If monitor sees packet again, it discards packet

Token Maintenance: 802.5

- Monitoring for a Valid Token
  - All stations should periodically see valid transmission (frame or token)
  - Maximum gap
    - = ring latency + max frame < = 2.5ms
  - Set timer at 2.5ms
    - send claim frame if timer expires

Timing Algorithm: 802.5

- Each node measures TRT between successive tokens
  - If measured-TRT > TTRT
    - Token is late
    - Don't send
  - If measured-TRT < TTRT
    - Token is early
    - OK to send
- Worse case:
  - 2xTTRT between seeing token
  - Back-to-back 2xTTRT rotations not possible

Traffic Classes: FDDI

- Two classes of traffic
  - Synchronous
    - Real time traffic
    - Can always send
  - Asynchronous
    - Bulk data
    - Can send only if token is early

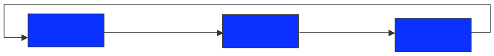
Timing Algorithm: FDDI

- Each station is allocated S_i time units for synchronous traffic per TRT
- TTRT is negotiated
  - $S_1 + S_2 + \dots + S_N$ + RingLatency ≤ TTRT
- Algorithm Goal
  - Keep actual rotation time less than TTRT
  - Allow station i to send S_i units of synchronous traffic per TRT
  - Fairly allocate remaining capacity to asynchronous traffic
  - Regenerate token if lost

Timing Algorithm: FDDI

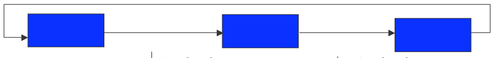
- When a node gets the token
  - Set TRT = time since last token
  - Set THT = TTRT – TRT
  - If TRT > TTRT
    - Token is late
    - Send synchronous data
    - Don't send asynchronous data
  - If TRT < TTRT
    - Token is early
    - OK to send any data
      - Send synchronous data, adjust THT
      - If THT > 0, send asynchronous data

FDDI example



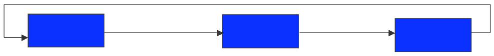
- Assume
  - RingLatency=12 μs
  - Three active stations
  - Each with Si=20 μs
  - TTRT=100 μs
  - Stations have unlimited supply of asynchronous traffic.

FDDI example



Arrival time	TRT	s/a	Arrival time	TRT	s/a	Arrival time	TRT	s/a
0	0	0/0	4	0	0/0	8	0	0/0
12	12	20/88	124	120	20/0	148	140	20/0
172	160	20/0	196	72	20/28	248	100	20/0
272	100	20/0	296	100	20/0	320	72	20/28
372	100	20/0	396	100	20/0	420	100	20/0
444	72	20/28	496	100	20/0	520	100	20/0

FDDI example



0	0	0	0	0	0	0	0	0
12	12	88	124	120	0	148	140	0
172	160	0	196	72	28	248	100	0
272	100	0	296	100	0	320	72	28
372	100	0	396	100	0	420	100	0
444	72	28	496	100	0	520	100	0
544	100	0	568	72	28	620	100	0
644	100	0	668	100	0	692	72	28
744	100	0	768	100	0	792	100	0
816	72	28	868	100	0	892	100	0
916	100	0	940	72	28	992	100	0
1016	100	0	1040	100	0	1064	72	28
1116	100	0	1140	100	0	1164	100	0
1188	72	28	1240	100	0	1264	100	0
1288	100	0	1312	72	28	1364	100	0
1388	100	0	1412	100	0	1436	72	28
1488	100	0	1512	100	0	1536	100	0
1560	72	28	1612	100	0	1636	100	0
1660	100	0	1684	72	28	1736	100	0
1760	100	0	1784	100	0	1808	72	28

FDDI Performance

- Synchronous traffic may consume one TTRT worth of time
  - TRT > TTRT
- Worst case
  - TRT < 2*TTRT
  - Any asynchronous traffic plus RingLatency ≤ TTRT
  - Synchronous traffic < TTRT

FDDI Performance

- Can’t have two consecutive TRT = 2*TTRT
  - After a cycle with TRT = 2*TTRT, no asynchronous traffic will be sent

Token Maintenance: FDDI

- Lost Token
  - No token when initializing ring
  - Bit error corrupts token pattern
  - Node holding token crashes
- Monitoring for a valid token
  - Should see valid transmission (frame or token) periodically – within 2*TTRT
  - Maximum gap = RingLatency + MaxFrame ≤ 2.5ms