MAC
Propagation Delay and Transmit time:

Question: How long does it take for a transmitter to fully transmit a packet to the receiver?

Propagation delay (PD) = \((t_j - t_0) = \frac{d_{TR}}{c}\)

Transmit time (TT) = \((t_k - t_0) = (t_n - t_j) = \frac{L}{R}\)

Total time = PD + TT
Medium Access Control (MAC)

How to share a resource in a distributed manner?

Resource is the "channel"

Idea 1

- I want to transmit
- Just transmit

Problem 1
- But all transmitters will collide.

Idea 2
- Listen before you talk to ensure no one else transmitting

Problems 2
- Simultaneous transmission cannot be avoided.
- Propagation delay problems, i.e., Bob cannot hear you until propagation delay. Can talk before that even after listening.
Idea 3

Listen before talking and also while talking. ⇒ If some foreign signal is heard, conclude there is a collision ⇒ Prepare to retransmit

- Collision still possible.

You ➔ Me ➔ Bob

Problem 3

Idea 4

Make pkt size longer, i.e.,

Transmit time > Max network prop. delay.

OR

Require Ack from the Receiver.

You ➔ Me ➔ Bob

3 Detect Collision
So current protocol:

1. I have a pkt to send
2. Listen to channel
3. Transmit Packet & keep listening to channel
4. Prepare to retransmit packet
5. Is channel idle
6. If not, repeat

Problem 4: Many nodes want to transmit. But all of them experience busy channel. So they all wait for channel to become idle. Then they all transmit $\Rightarrow$ collision for sure.
Wait for random times after channel becomes idle.

- Node may receive unfair share of channel just due to bad luck. Starvation possible (although low probability).

Don't reset random no. if unable to transmit. Instead, freeze the random countdown once channel becomes busy. Resume countdown once channel idle again.
So current protocol:

1. I have a pkt to send
2. Listen to channel
3. Pick Random No. R
4. Is Channel idle?
   - Yes: Count down R to zero
   - No: Keep listening while counting down
5. Is Channel Busy?
   - Yes: Freeze countdown
   - No: Transmit and keep listening
6. Is Tx done?
   - Yes: Prepare retransmission
   - No: Is Foreign Signal?
     - Yes: Abort Tx
     - No: Wait for ACK
Problem 6

• How do you cope with collision?

Do you pick new random # and try again? Can there be collision again?

Idea 7

Increase random number range after collision. Pick new rand #.

This is a form of admission control, i.e., reduce the rate of pkt injection into the network.

Problem 7

• But congestion may have gone and you still pick large random #s.

Idea 8

Once pkt is successful (i.e., no congestion) then reset your random # range back to minimum value.
• Collision still possible even if 2 nodes have picked different random #s.

Make the countdown as multiples of slots, where each slot time is greater than prop. delay.

Bob doesn't transmit. No collision.
So current protocol:

- I have a pkt to send
  - Count down R to zero
  - Listen to channel
  - Pick Random No. R
  - Is channel idle?
    - Yes, go to 'Is Channel Busy?'
    - No, go to 'Is countdown = zero?'
    - Yes, go to 'Freeze countdown'
    - No, go to 'Transmit and keep listening'
  - Is Channel Busy?
    - Yes, go to 'Is Foreign Signal?'
      - No, go to 'Abort Tx'
    - No, go to 'Prepare retransmission'
  - Is Foreign Signal?
    - Yes, go to 'Abort Tx'
    - No, go to 'Transmit and keep listening'
  - Freeze countdown
  - Is Tx done?
    - Yes, go to 'Wait for ACK'
    - No, go back to 'Pick Random No. R'
  - Increase Random # range
  - Pick new Random # R+

- Keep listening while counting down
- Reset Random # range to min. value
Some key points.

1. Collision happens always at the Receiver. Transmitter may detect collision by observing a foreign signal, but that doesn't mean collision is at Tx.

2. Channel is wasted because of random count down \(\Rightarrow\) called BACKOFF. This is the price to be paid for distributed coordination.

3. The above protocol assumes that a Tx can transmit and listen at the same time. Possible in wired networks like Ethernet. Harder in wireless networks.

4. Tx detects foreign signal and can tell for sure that collision is happening at Rx. This assumes channel is identical at Tx and Rx. True for wired networks, not for wireless.