Institute for the Wireless Internet of Things
at Northeastern University

EECE5155: Wireless Sensor Networks and the Internet of Things

Josep Miquel Jornet
Associate Professor, Department of Electrical and Computer Engineering
Director, Ultrabroadband Nanonetworking Laboratory
Member, Institute for the Wireless Internet of Things
Northeastern University
jmjornet@northeastern.edu
www.unlab.tech

Institute for the Wireless Internet of Things at Northeastern

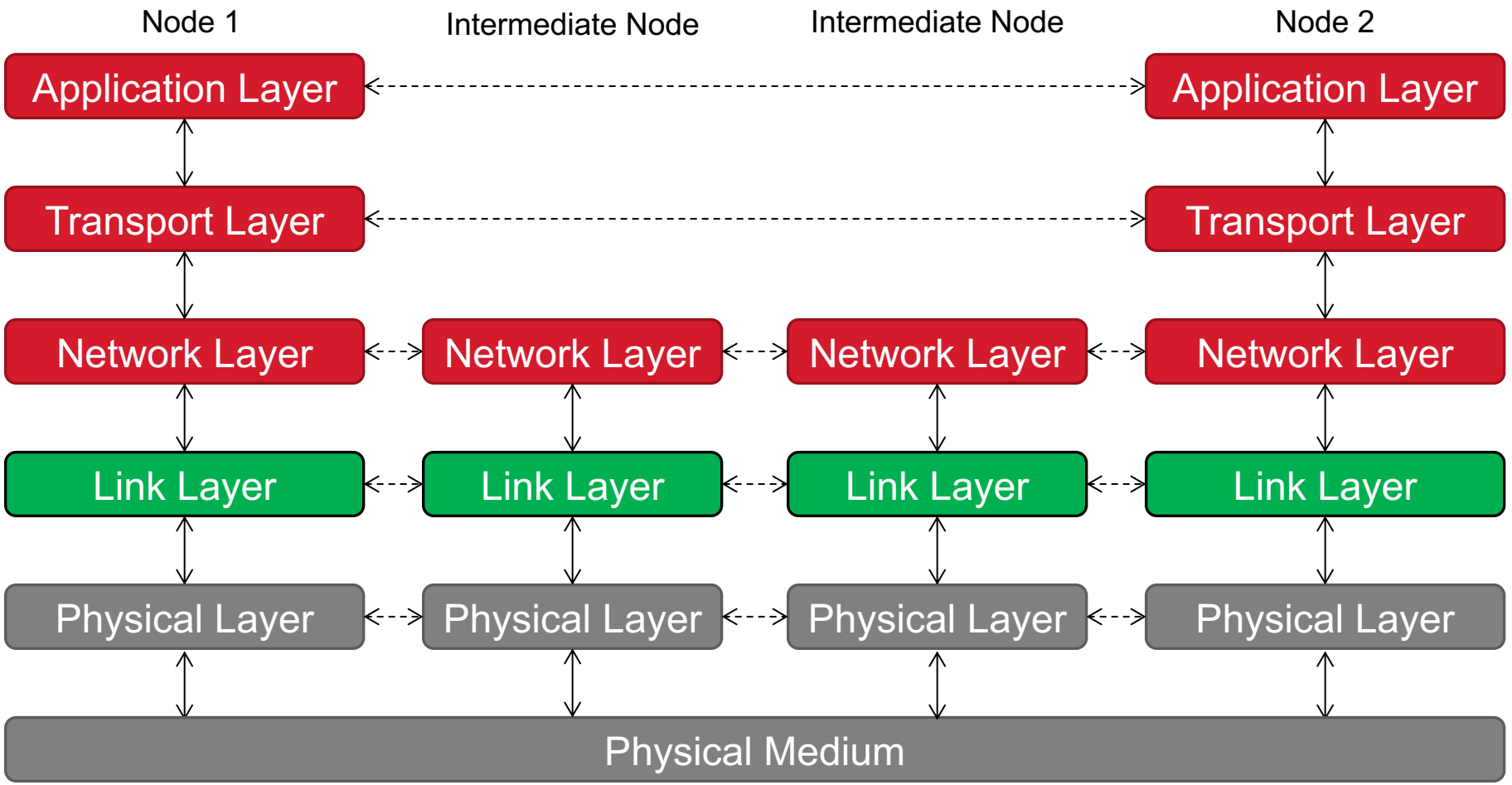
Module T4: Data Communication

Institute for the Wireless Internet of Things at Northeastern

Part 2: Protocols and Standards

The Protocol Stack

Institute for the Wireless Internet of Things at Northeastern



Institute for the Wireless Internet of Things at Northeastern

The Link Layer

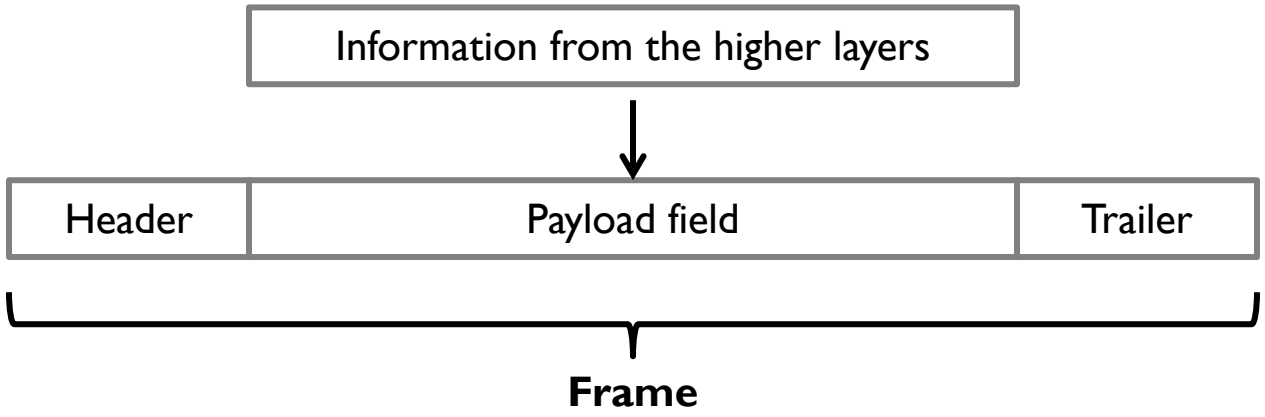
Functions of the Link Layer

- Framing
- Addressing
- Error Control
- Flow Control
- Medium Access Control (MAC)

Framing

Institute for the Wireless Internet of Things at Northeastern

- **Goal:** to group or encapsulate bits into packets/frames
  - Strictly speaking, at the link layer, the word "frame" should be used, as opposed to "packets" which is the term reserved for the network layer
    - The reality as that the two terms are usually used interchangeably
- Each frame is composed of
  - **Header:** contains control and link layer information
  - **Payload:** contains the packet from the network layer (i.e., the actual information)
  - **Trailer:** determines the end of a frame and contains additional control information



Header

- Common fields in the header include:
  - **A synchronization sequence:**
    - A unique (short) combination of 0s and 1s, know by both the transmitter and the receiver, that is used to indicate the beginning of the frame
      - The receiver "looks for it" usually by exploiting the correlation function
        - (More in a signal processing/communications course)
  - **A pilot or training sequence:**
    - A collection of 0s and 1s, known by both the transmitter and the receiver, that is used to "learn" the behavior of the channel
      - The receiver will compare what it receives with what is transmitted and compensates for the differences in a process known as channel estimation and equalization
        - (More in a signal processing/communications course)

Header

- Other fields:
  - **Frame length:**
    - A field indicating how many bits the frame has
      - The receiver will stop processing the incoming signal after reaching the end
    - Alternatively, the trailer can indicate the end of the frame (reciprocal to the header)
  - **Origin and destination addresses:**
    - Each node should have a unique ID in the network
    - Each frame should indicate who is sending the information to who
      - At the link layer, we usually refer to them as MAC or physical addresses
        - As opposed to the IP or logical addresses (more at the Network Layer)
  - **Error control-related fields**
    - More in the next slides!

Error Control

- Errors do happen…
  - … because of noise, interference, lack of (tight) synchronization, hardware malfunctioning, etc.

- **Objectives** of error control:
  - **Detect the presence of errors:**
    - The link layer should not send "incorrect" information to the upper layers of the protocol stack
  - **Recover from errors:**
    - By leveraging error correction codes, error detection codes, positive & negative acknowledgements and retransmissions

Error Control Mechanisms

- **Forward Error Correction (FEC)**
- **Automatic Repeat reQuest (ARQ)**
- **Hybrid ARQ**

Forward Error Correction

Institute for the Wireless Internet of Things at Northeastern

- Based on the use of error correction codes
  - Can restore (a fraction of) the erroneous bits by leveraging *carefully computed* redundant bits

- Most error correction common codes:
  - Hamming codes
  - Binary convolutional codes
  - Reed-Solomon codes
  - Low-Density Parity Check codes

Definitions

- **m =** data bits
- **r =** redundant bits
- **n = m+r =** total number of bits
- **(n,m) code =** a code with n total bits out of which only m are data
- **n-bit codeword =** a n-bit unit containing the data and the redundant bits
- **m/n =** code rate = the fraction of the codeword that carries information that is not redundant
- $\mathbf{2^m}$ **=** total number of possible data messages
- $\mathbf{2^n}$ **=** total number of possible codewords (not all of them might be used)

Some types of codes

- **Block code:** the *r* redundant bits are computed solely as a function of the *m* data bits with which they are associated (e.g., as if obtained from a table)

- **Systematic code:** the *m* data bits are sent directly, along with the *r* redundant bits, rather than being encoded themselves before they are sent

- **Linear code:** the *r* redundant bits are computed as a linear function of the *m* data bits (e.g., XOR or modulo 2)

Definitions

- **Hamming distance between two codewords:**
  - The number of bits in which two codewords differ:
  - Can be easily computed with the XOR function:

$$\begin{array}{r} 0\ 1\ 1\ 0\ 1\ 1\ 1\ 1 \\ 0\ 1\ 0\ 0\ 0\ 1\ 1\ 0 \\ \hline 0\ 0\ 1\ 0\ 1\ 0\ 0\ 1 \end{array} \quad d=3$$

  - If two codewords are *d* apart, *d* single-bit errors are needed to convert one codeword into the other

- **Hamming distance of a code:**
  - The smallest Hamming distance between any two codewords in the complete code
  - To detect *d* errors, you need a code with distance *d+1*
    - The erroneous codeword is still different than any other codeword
  - To correct *d* errors, you need a code with distance *2d+1*
    - The erroneous codeword is still closer to the original codeword

Example

- **Code with 4 codewords and distance equal to 5**:
  - 0000000000, 0000011111, 1111100000, and 1111111111

- If 0000000111 is received:
  - If we expect **only single or double errors**, we can tell that the originally transmitted codeword was 0000011111
  - If we expect **triple errors**, we can just detect that there are errors, but we don't know whether this codeword comes from 0000000000 or from 0000011111

- We are identifying the errors and finding the codewords **by inspection**, but this cannot be done when you have a large number of codewords
  - We need a systematic way to do this

Single-bit Error Correction Codes

- Consider a (n,m) code which allows us to correct single-bit errors:
  - Each of the 2^m data messages has n messages at a distance 1 from it
    - These can be obtained by systematically inverting each of the n bits in the n-bit codeword formed from it
      - E.g., 000000 → 000001, 000010, 000100, 001000, 010000, 100000
  - Each data message "occupies" n+1 possible codewords,
    - I.e., the correct codeword and n codewords we should not use
  - Therefore, the minimum number of redundant bits needed to correct single-bit errors should satisfy:

$$(n+1)2^m \leq 2^n$$

$$(m+r+1) \leq 2^r$$

Hamming Codes

- **Linear systematic block codes** that achieve the lower bound for the minimum number of redundant bits needed to correct any single-bit error
  - It can be shown that they have distance 3
- There are many hamming codes:
  - Hamming(3,1) → Triple repetition code (we send 3 times each bit!)
  - Hamming(7,4) → We send the original 4 bits followed by 3 "smartly computed" redundant bits
  - Hamming(11,7) → …
  - Hamming(15,11)
  - …
  - Hamming(2^r-1,2^r-r-1)
- Easy to construct
  - Out of the scope of this course… → Look for courses on communications, information and coding theory

Convolutional Codes

- **Linear** codes, in which an encoder processes a sequence of input bits and generates a sequence of output bits
- **Not a block code**: there is no natural message size or encoding boundary
- **Not systematic**: we don't transmit data+redundancy, but just a "new" bitstream
- The output depends on the current and previous input bits
  - The encoder has memory
- The Viterbi algorithm is used to decode the received bits
  - Beyond the scope of this course…

Reed-Solomon (RS) Codes

- Linear block codes that:
  - Operate with *m* bits symbols rather than individual bits
  - For *m*-bit symbols, the codewords are 2^m-1 symbols long
    - E.g., for *m*=8 bits, each codeword is 255 symbols long
    - Popular choice: RS (255,223) code = adds 32 redundant symbols (bytes) to 223 data symbols
- Basic idea:
  - An *n* degree polynomial is uniquely determined by *n+1* points
    - E.g., the line *ax+b* is determined by 2 points, more points are redundant
    - If we send two data points that represent a line and two additional check points on the same line…
    - … if one of the points is received erroneously, we can still recover the data by fitting the point to the line

Low-Density Parity Check (LDPC) Codes

- Linear block codes in which:
  - Each output bit is formed from only a fraction of the input bits
    - This leads to a matrix representation of the code that has a low density of 1s, hence the name for the code
- The received codewords are decoded with an approximation algorithm that iteratively improves on a best fit of the received data to a legal codeword
  - This corrects errors
- LDPC codes are practical for large block sizes and have excellent error-correction abilities that outperform many other codes in practice

Observation

- All the techniques seen so far attempt to aid the receiver to recover from channel errors, by adding redundant data to the frame before being transmitted (i.e., *forward* or in advance)

- Adding redundant bits ultimately means transmitting less *useful* data in the same amount of time
  - Sometimes, **simpler error detection codes** might suffice to discover unlikely errors…
  - Other times, even very strong error correction techniques cannot overcome all the errors…

Error Detecting Codes

Institute for the Wireless Internet of Things at Northeastern

- Parity Check Codes
- Checksum
- Cyclic Redundancy Checks (CRC)

Parity Check Codes

- **Simplest case:** append a bit to each codeword, so that the number of 1 bits in the codeword is even (or odd)
  - E.g., 0100011 → 01000111 → Even Parity
  - E.g., 0100011 → 01000110 → Odd Parity

- A code with a single parity bit has a hamming distance of 2, i.e., can only detect single-bit errors
  - Cannot correct anything!

Parity Check Codes

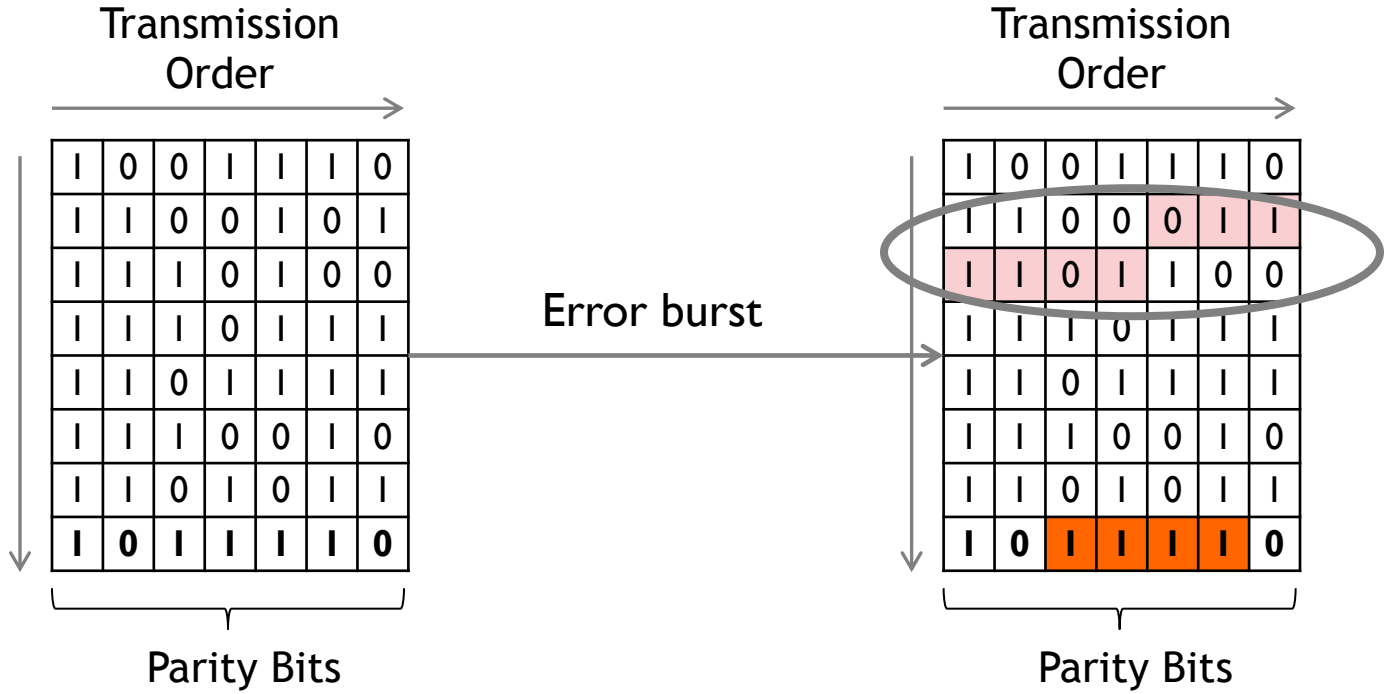
- Some numbers:
  - Channel bit error rate: 10^{-6} → 1 erroneous bit every 1 Mbit
  - Total data to transmit: 1 Mbit
  - Block size: 1000 bits

- **Error correction codes:**
  - From Slide 17, 10 redundant bits per block are needed to be able to correct single-bit errors
  - For 1000 blocks (1 Mbit), **10000 additional bits are transmitted**

- **Error detection codes:**
  - 1 parity check bit is added per block
  - For 1000 blocks (1 Mbit), 1000 additional bits are transmitted
  - 1 block needs to be retransmitted, i.e., 1001 bits
  - **Total additional bits: 2001 bits**

Interleaving

- To overcome error bursts, bits can be transmitted in a different order in relation to how the parity bits are computed
- Example:



Other Options

- **Checksum:** A group of check bits associated with a message
  - E.g., 16-bit Internet checksum used in every IP packet
    - Sum of the message bits divided into 16-bit words in one's complement arithmetic
    - Stronger than parity check, as it can detect errors that leave the parity unchanged (e.g., 2 zeros becoming 2 ones or *vice versa*)
    - Still weak as it does not detect the deletion or addition of zero data nor swapping parts of the message
- **Cyclic Redundancy Check (CRC):** Also known as polynomial codes:
  - Based upon treating bit strings as representations of polynomials with coefficients of 0 and 1 only
  - The idea is to append a CRC to the end of each frame in such a way that the polynomial represented by the checksummed frame is divisible by a common generator polynomial

- Out of the scope of this course…

Observation

- Error detection codes only allow the receiver to realize about the presence of errors
- Error correction codes cannot always recover a frame, specially when there are many errors or these come in bursts
- Some frames can be simply not be received at all due to noise an interference…

What can be done at this layer to create a reliable link?

Automatic Repeat reQuest (ARQ)

- A set of error control techniques based on the use of **acknowledgements** and **timeouts** to achieve reliable data transmission
  - **Acknowledgements:** messages sent by the receiver to indicate that a data frame has been correctly (ACK) or incorrectly (negative ACK or NACK) received
  - **Timeouts:** maximum waiting time during which an acknowledgement is expected to be received
- **Also used as part of *flow control* at the link layer**

Flow Control

- **Objective:** to avoid an overflow of the receiver due to a mismatch in the data handling capabilities of transmitter and receiver:

- Two types:
  - **Feedback-based Flow Control:** the receiver sends back information to the sender giving it permission to send more data

  - **Rate-based Flow Control:** the protocol in use between transmitter and receiver incorporates a built-in mechanism that limits the rate at which messages are transmitted
    - **We will look more into this when discussing the Transport Layer**

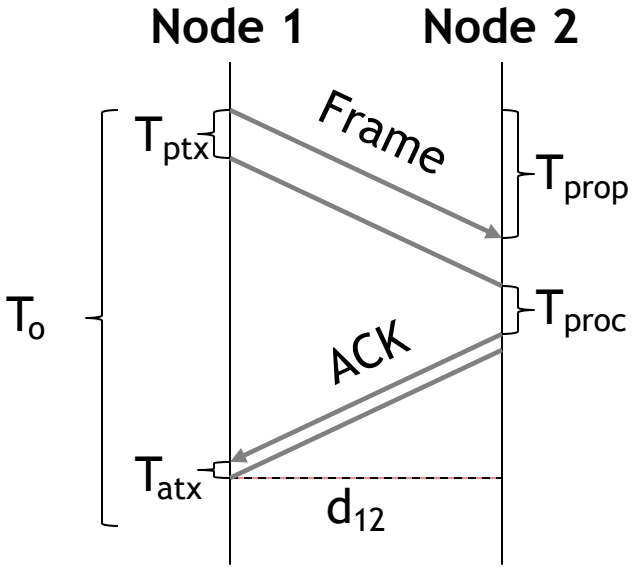
Types of ARQ Protocols

- Stop and Wait
- Sliding Window
- Selective Repeat

Stop and Wait

- After transmitting one frame, the sender waits for an acknowledgment before transmitting the next frame

- If the acknowledgment does not arrive after a certain period of time, the sender times out and retransmits the original frame



$$T_{ptx} = \frac{\text{Frame length}}{\text{Bit-rate}} = \text{Packet transmission time}$$

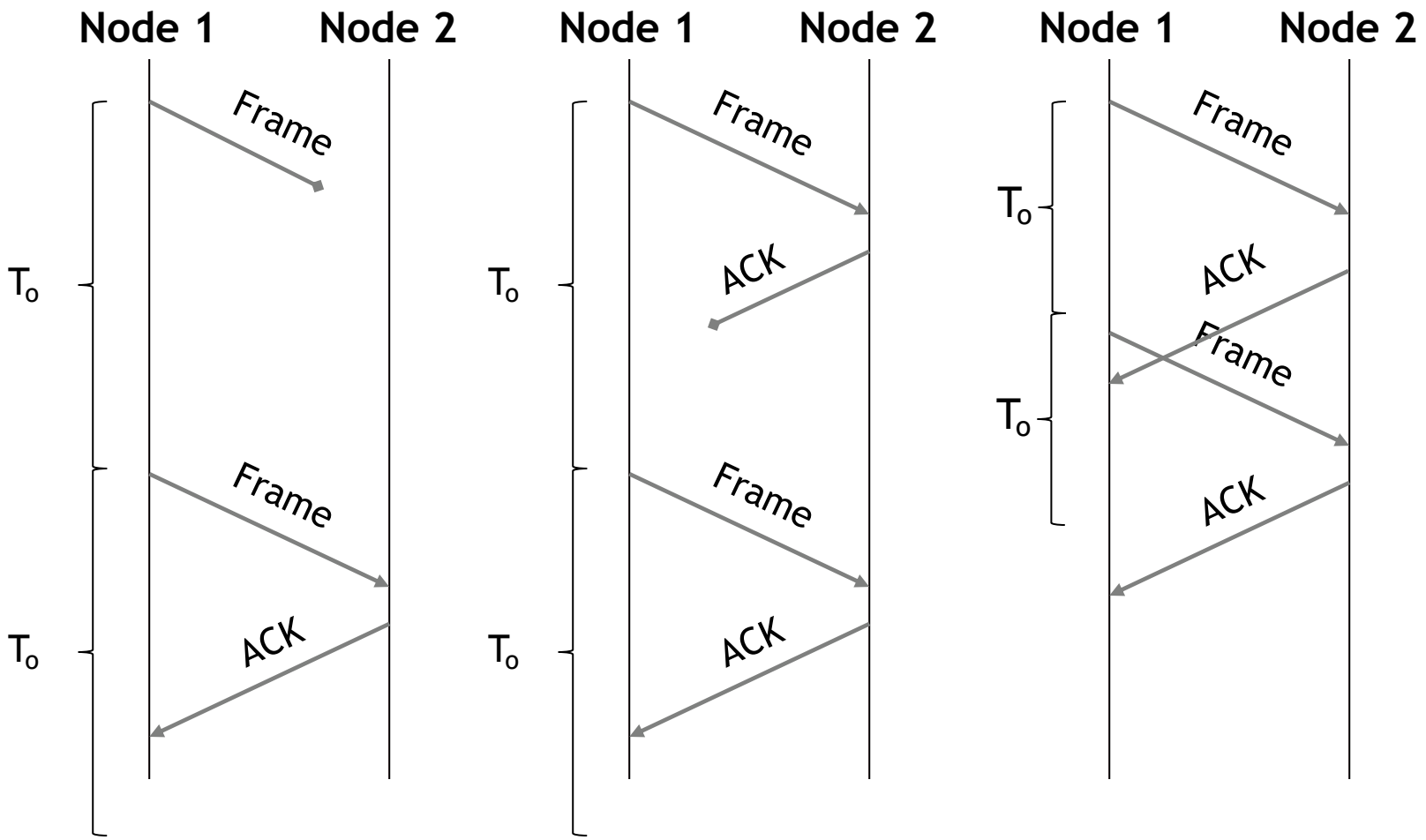
$$T_{prop} = \frac{d_{12}}{v_p} = \text{Propagation time}$$

T_{proc} = Processing time

$$T_{atx} = \frac{\text{ACK length}}{\text{Bit-rate}} = \text{Acknowledgement transmission time}$$

T_o = Time-out

Stop and Wait



Stop and Wait

- **Problem:**
  - Only one frame is sent at a time:
    - If the propagation time is very long, the effective rate at which the information is transmitted is much lower than the actual data rate at the physical layer

$$2T_{prop} + T_{ptx} + T_{proc} + T_{atx} \gg T_{ptx}$$

Sliding Window

- Can drastically increase the effective rate at which the information is transmitted, by allowing multiple frames to be transmitted while waiting for the acknowledgements

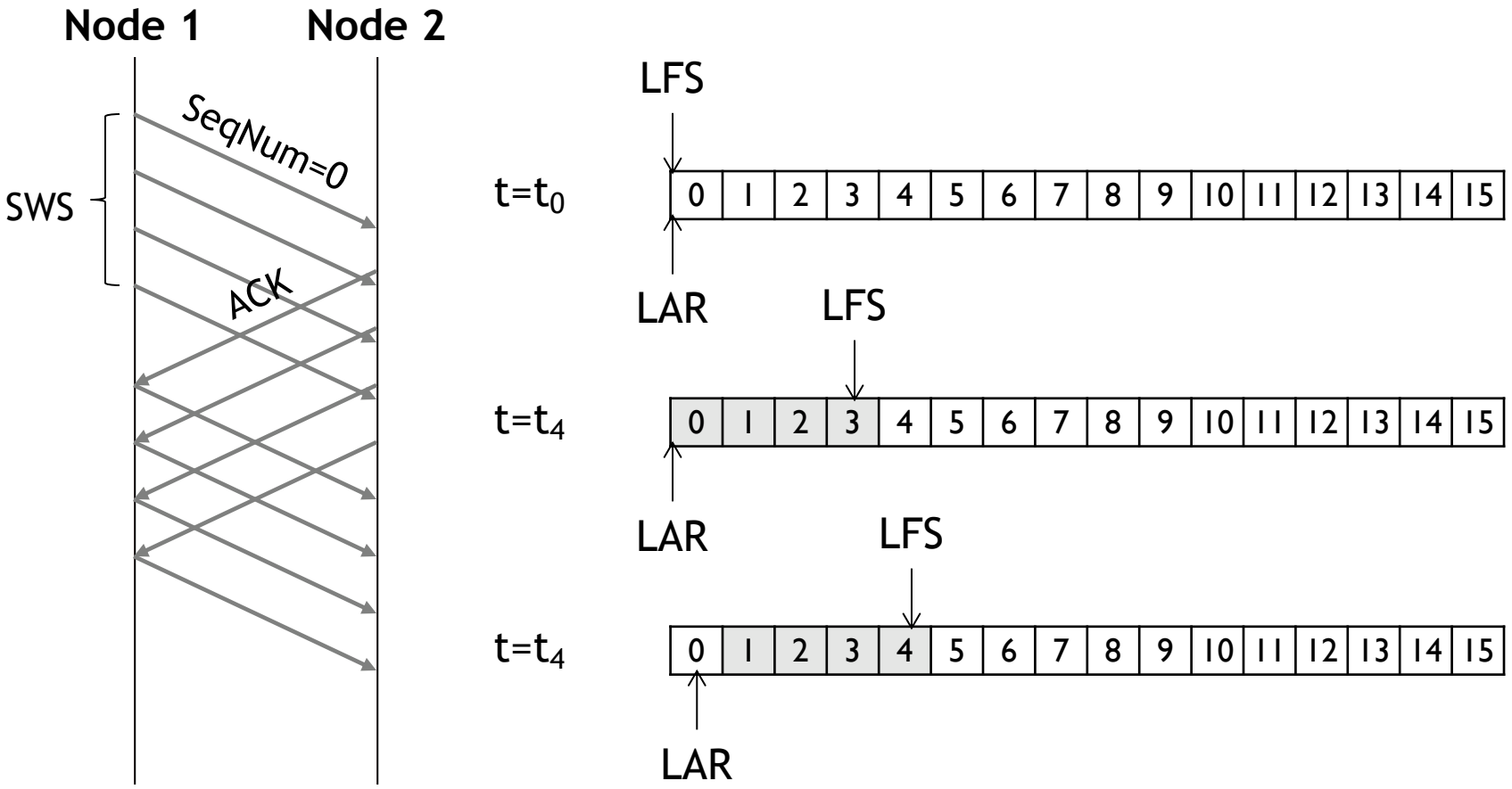
- Some definitions:
  - *Last Frame Sent (LFS)*
  - *Last Acknowledgement Received (LAR)*
  - *Sender Window Size (SWS)* = Maximum number of frames that can be transmitted without being acknowledged
  - The following condition needs to be satisfied

LFS-LAR≤SWS

Transmitter Behavior

- The sender assigns a *sequence number (SeqNum)* to each frame
- The sender sends up to SWS frames consecutively without waiting for an ACK
- When an ACK arrives, the sender moves LAR to the right thus allowing the sender to transmit another frame
- The sender associates a timer with each frame it transmits, and it retransmits the frame should the timer expire before an ACK is received
  - The sender has to be willing to buffer up to SWS frames since it must be prepared to retransmit them until they are acknowledged

Transmitter's Sliding Window



Receiver Behavior

- More definitions:
  - *Receiver Window Size (RWS)* = Maximum number of out-of-order frames that the receiver is willing to accept
  - *Largest Acceptable Frame (LAF)*
  - *Last Frame Received (LFR)*

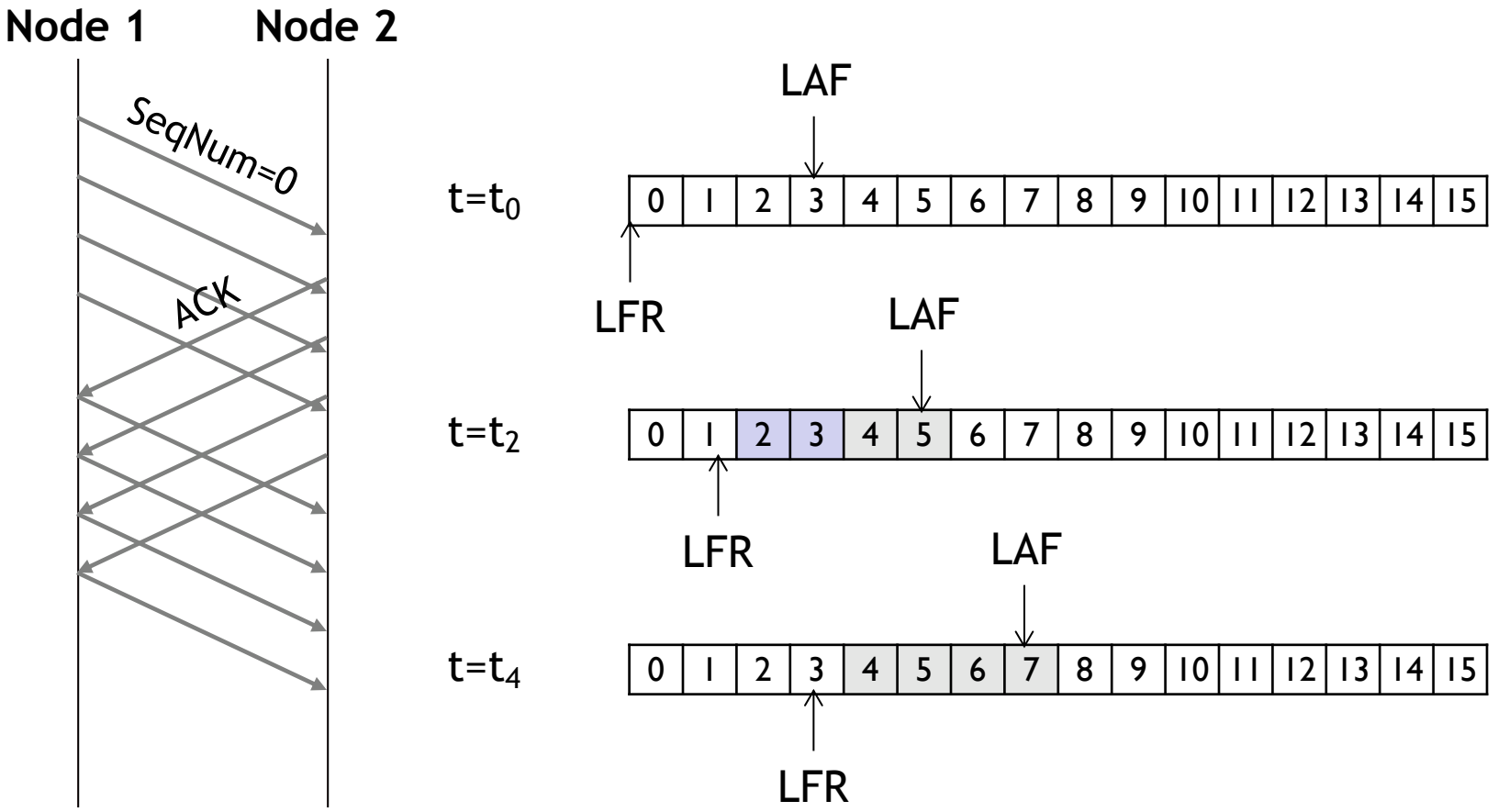
- Functioning:
  - When a frame with sequence number *SeqNum* arrives:
    - If $SeqNum \leq LFR$ or $SeqNum > LAF$, then the frame is outside the receiver's window and it is discarded
    - If $LFR < SeqNum \leq LAF$, then the frame is within the receiver's window and it is accepted
      - Should an acknowledgment be sent?

Receiver Behavior

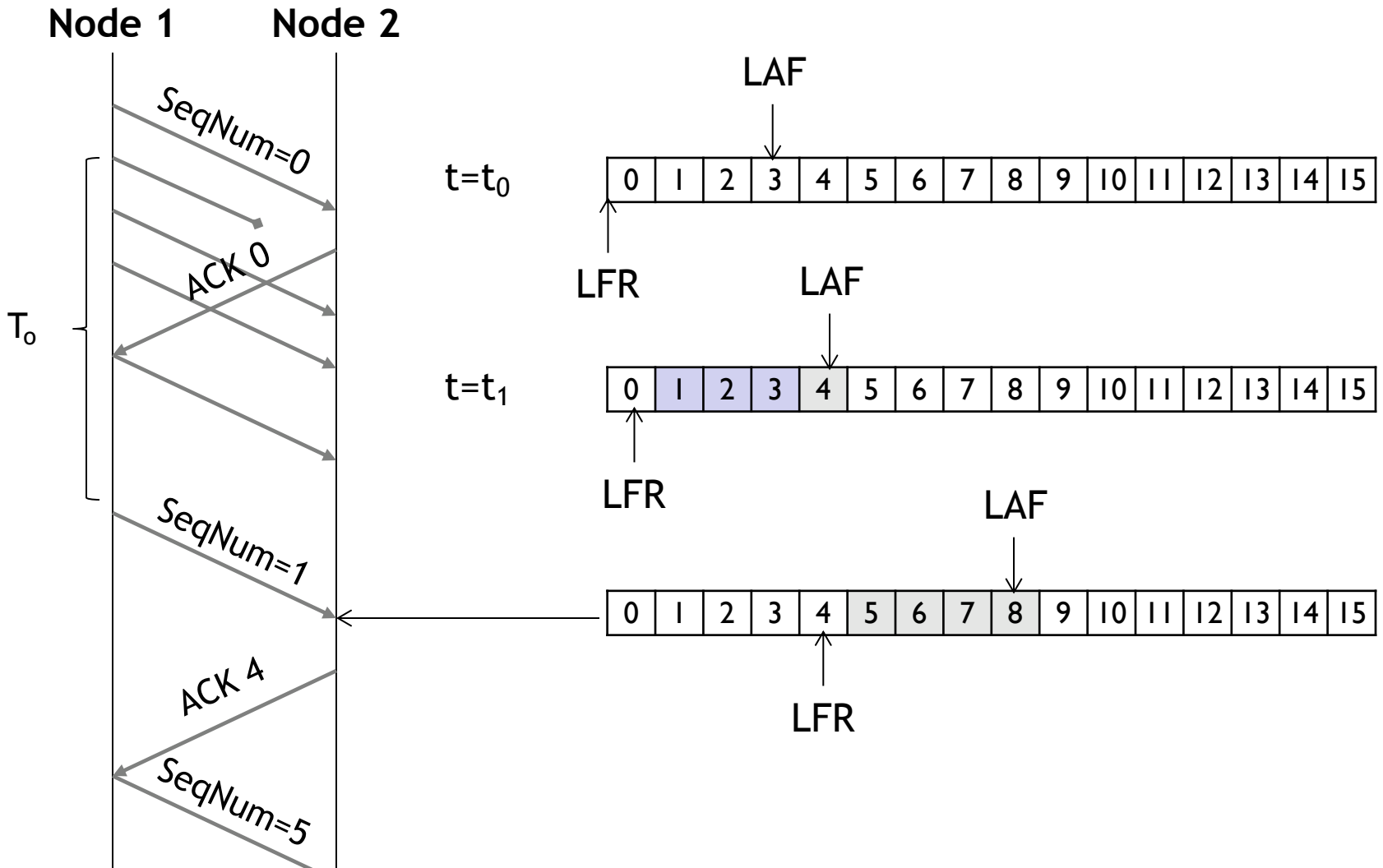
- *SeqNumToAck* = largest sequence number not yet acknowledged, such **that all frames with sequence numbers less than or equal to *SeqNumToAck* have been received**

- The receiver acknowledges the receipt of SeqNumToAck, even if higher numbered packets have been received
  - This is called *cumulative acknowledgement*
  - It then sets:
    - *LFR* = *SeqNumToAck*
    - *LAF* = *LFR* + *RWS*

Receiver's Sliding Window



Receiver's Sliding Window



Variations of the Sliding Window

- **Negative Acknowledgement:**
  - In our example, a NACK for frame SeqNum=1 could be sent when frame with SeqNum=2 is received
- **Selective Acknowledgement:**
  - In our example, frames SeqNum=2 and SeqNum=3 could be acknowledged directly, without using the cumulative acknowledgement
    - The receiver could add a field specifying that SeqNum=1 is missing → This is called selective repeat
- **Go back N:**
  - A particular case with SWS=N and RWS=1

Medium Access Control

- Controls how the shared medium (i.e., the channel) is used by different devices
- Determines
  1. when to send a packet
  2. when to listen for a packet

Perhaps the two most important operations in a wireless network!!!

Link Layer Jargon

- **Packet Collisions:** two or more packets reach the receiver at the same time with comparable power
  - Extreme case of multi-user interference
  - The receiver experiences it
  - The transmitter might not know about it:
    - It can only learn of it after it has occurred:
      - Through a negative acknowledgement packet
      - Because of the lack of a positive acknowledgement packet

- **Back-off Time:** after a collision, or a failed transmission attempt (e.g., because of noise), nodes do not immediately attempt to transmit again or retransmit, but wait for a random time
  - This is supposed to minimize the probability of colliding again

Some Common Language

Institute for the Wireless Internet of Things at Northeastern

- **Packet Delay:** The total time between the moment a packet is generated at the transmitter to the moment the receiver acknowledges its proper reception
  - It takes into account everything:
    - Processing time
    - Transmission time
    - Propagation time

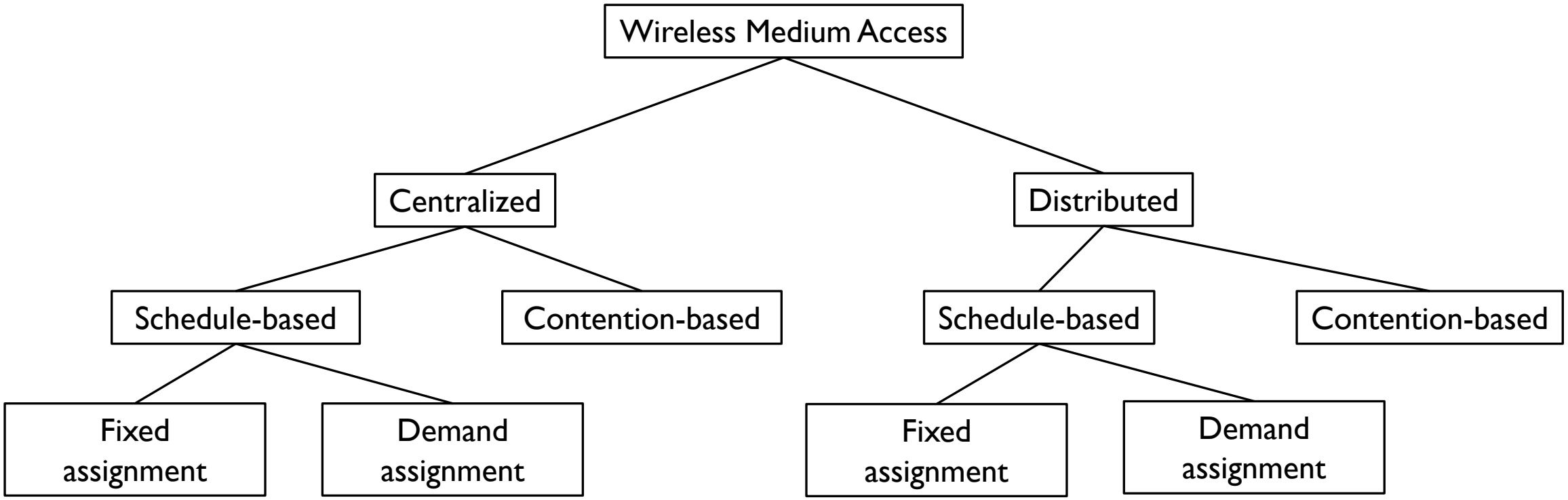
  from the very first message (e.g., handshake) to the very last message (e.g., acknowledge)

- **Throughput:** the effective rate (bits/second) at which actual information is transmitted between a transmitter and a receiver, defined as the actual data-bits divided by the packet delay
  - This can be much lower than the physical layer data-rate!

- **Packet Successful Delivery Probability:** the probability that the packet is ultimately properly received (after retransmissions, if needed)

Type of MAC Protocols

- Static (Fixed) VS Dynamic (On demand) Channel Access
- Contention-free (Scheduled) VS Contention-based Channel Access
- Centralized VS Distributed Channel Access



Objectives of a MAC Protocol

Institute for the Wireless Internet of Things at Northeastern

- Collision Avoidance
  - Reduce Retransmissions
- Energy Efficiency
  - Avoid Idle Listening
- Scalability
- Latency
- Fairness
- Throughput
- Bandwidth Utilization

Static/Fixed Channel Access

- Based on the use of multiplexing techniques at the physical layer
- Fixed distribution of the channel resources among N nodes:
  - **Frequency Division Multiple Access (FDMA):** each one of the N nodes utilize a fraction of the bandwidth all the time
    - Useful only when the bandwidth to share is reasonably large
  - **Time Division Multiple Access (TDMA):** N nodes take turns at occupying the entire bandwidth for a limited time
    - Requires tight synchronization among users
  - **Code Division Multiple Access (CDMA):** N nodes occupy the entire bandwidth all the time by using different spreading codes
    - Requires accurate power control to avoid the near-far problem
  - **Space Division Multiple Access (SDMA)*:** N nodes occupy the entire bandwidth all the time but only in a confined region of space
    - * Only for wireless networks with directional antennas

Particular Case

- **Frequency Division Duplex (FDD):** two nodes establish a full-duplex link between them by transmitting and receiving at different frequencies

- **Time Division Duplex (TDD):** two nodes establish a full-duplex link between them by taking turns in the utilization of the channel

Static Channel Access

- Different nodes can have different number of resources assigned (sub-bands, time slots, etc.) based on their expected communication needs
- However, if the nodes' needs change with time, a static channel access scheme cannot guarantee an effective use of the resources
  - E.g., if a node has nothing to transmit, its time slots or frequency sub-bands will be just wasted…

In most of the applications, there is a dynamic component in the Medium Access Control

Dynamic Channel Access

- Resources are used (or at least requested) *on-demand* by the different nodes in the network
- Some definitions:
  - **Independent Traffic:** the model consists of N independent nodes, each one of them generating frames by following a *Poisson distribution* with rate λ frames/second
  - **Single Channel:** a single channel is available for all communication and all nodes can transmit on it and receive from it
  - **Observable Collisions:** if two frames are transmitted at the same time and overlap in time, the resulting signal is garbled → This is called a collision
  - **Continuous or Slotted Time:** the transmission of a frame can begin at any instant (continuous time) or only at specific discrete intervals (slotted time)
  - **Carrier Sense or No Carrier Sense:** the nodes can either check if the channel is in use before trying to use it (carrier sense) or just send the frame directly (no carrier sense)

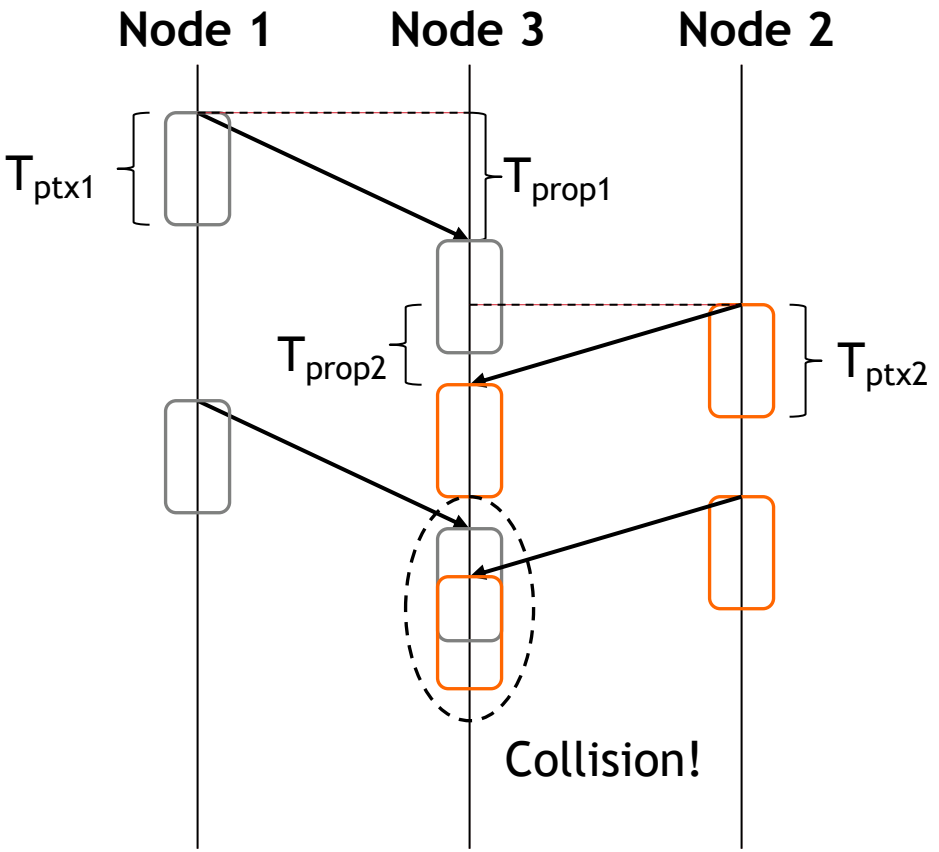
Multiple Access Protocols

- **ALOHA**
  - Pure ALOHA
  - Slotted ALOHA
- **Carrier Sense Multiple Access (CSMA)**
  - Persistent and Non-persistent CSMA
  - CSMA with Collision Detection (CSMA-CD)
  - CSMA with Collision Avoidance (CSMA-CA)
- **Others**
  - Token Ring/Bus
  - Polling

Pure ALOHA

Institute for the Wireless Internet of Things at Northeastern

- A node transmits a frame whenever it needs to, without checking whether the channel is occupied or not
- If two or more frames reach the receiver at the same time, there is a collision, and the frames need to be retransmitted after waiting for a random back-off time

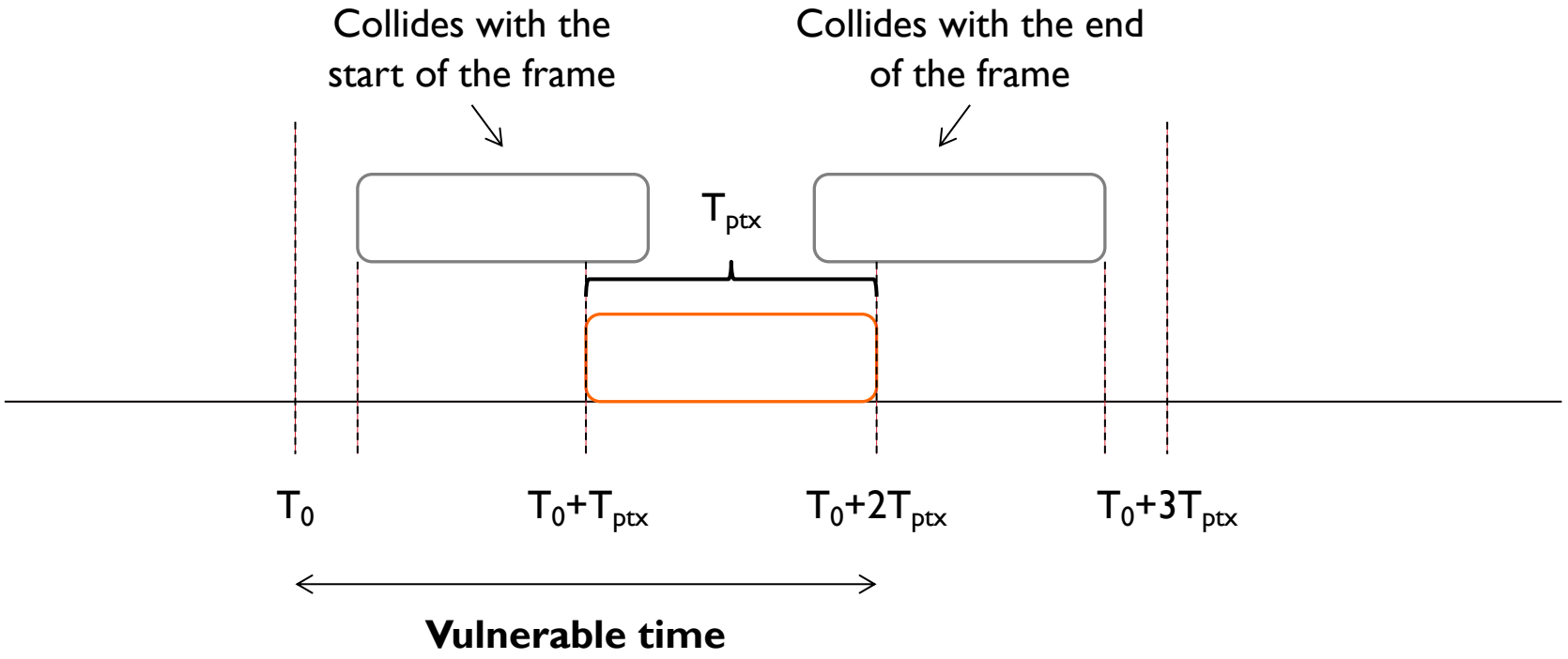


Pure ALOHA

- Some definitions:
  - **New Frame Generation Rate (N):** normalized average number of new frames generated per frame time
    - If N>1, nodes are generating frames at a higher rate than what the channel can handle (i.e., only 1) → Expect many collisions and, thus, many retransmissions
    - Preferably, 0<N<1
  - **Total Offered Load (G):** normalized total number of transmitted frames per frame time
    - Accounts for both new frames and retransmitted frames
    - If there are many collisions, G>>N
    - At low load (N<<1), G≈N
  - **Throughput (S):** average rate of successfully delivered frames per frame time
    - $S=GP_0$, where P_0 is the probability a frame does not suffer a collision

Throughput of Pure ALOHA

- Probability of no collision:
  - A collision at the receiver will not occur as long as no other frames are received within the vulnerable time



Throughput of Pure ALOHA

- The probability of having k frames transmitted during a frame time is given by the Poisson distribution

$$P[k] = \frac{G^k e^{-G}}{k!}$$

- The probability of having 0 frames transmitted during a frame time is thus given by

$$P[0] = e^{-G}$$

- Given that the vulnerable time is equal to two times the frame time, the probability of not having a collision is given by

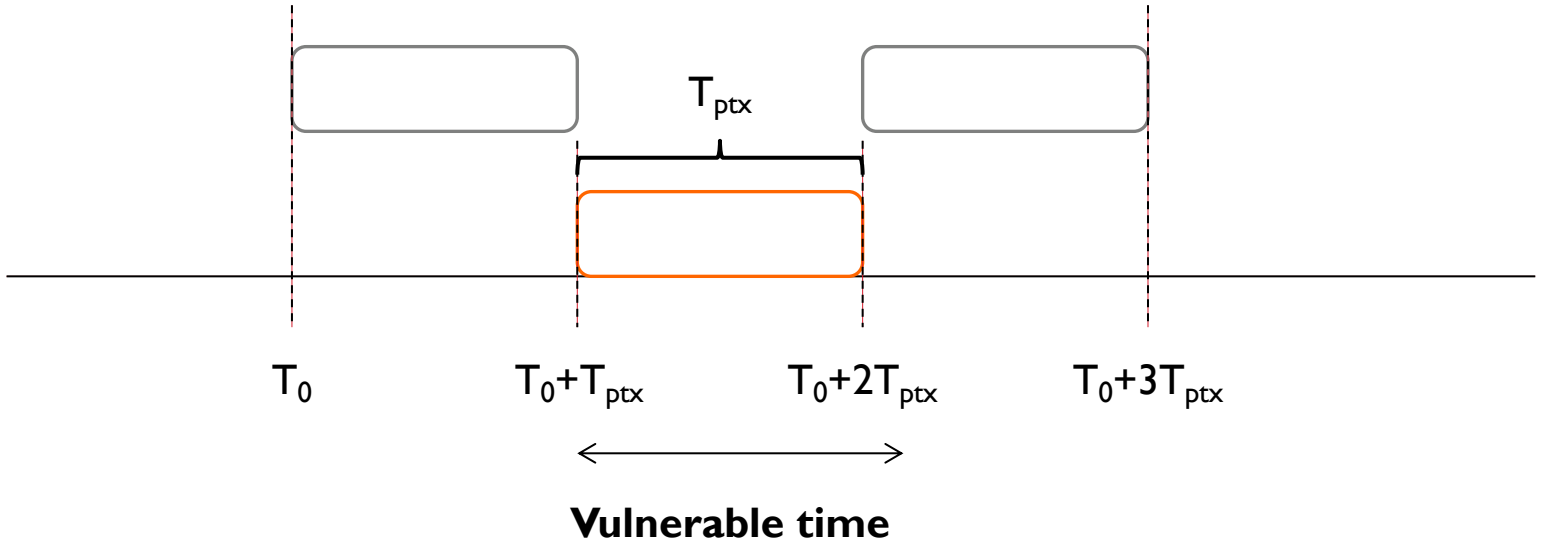
$$P_0 = e^{-2G}$$

- Finally, the throughput S of Pure ALOHA is

$$S = Ge^{-2G}$$

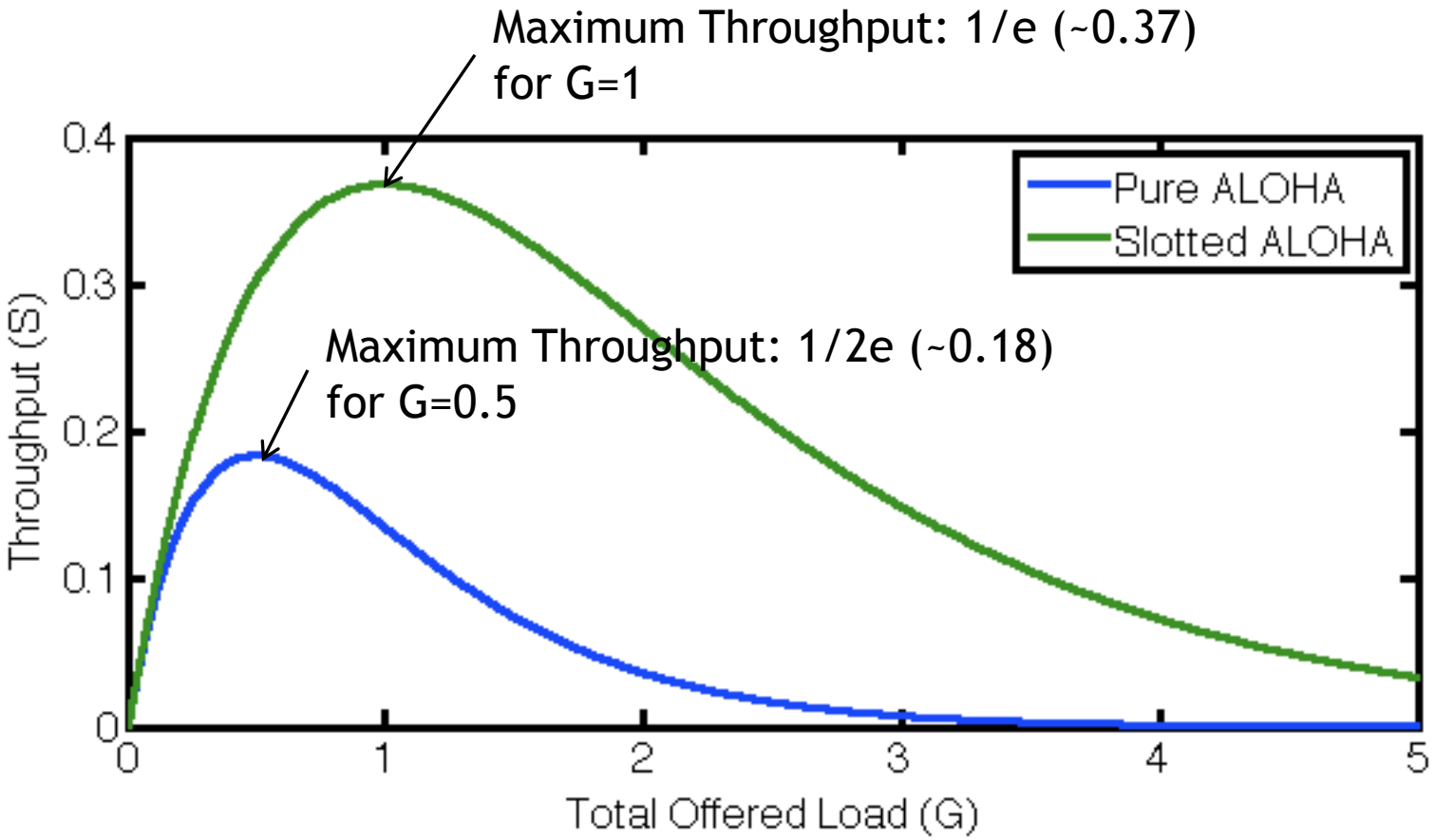
Slotted ALOHA

- Time is divided into slots, whose boundaries are agreed by all the nodes
- When a node wants to transmit a frame, it waits for the beginning of a new time slot
  - The vulnerable time is cut by 2! → Throughput increases
  - Requires synchronization…
  - Wasted slots…



Throughput: $S = Ge^{-G}$

Throughput of Pure and Slotted ALOHA



Expected Number of Retransmissions

- Probability of no-collision $p_0 = e^{-G}$
- Probability of collision $p_{col} = 1 - p_0 = 1 - e^{-G}$
- Probability of k retransmissions $p_k = e^{-G}\left(1 - e^{-G}\right)^{k-1}$
- Expected number of retransmissions $E = \sum_{k=1}^{\infty} k p_k = \sum_{k=1}^{\infty} k e^{-G}\left(1 - e^{-G}\right)^{k-1} = e^{G}$
- Increases exponentially with the offered load…

Carrier Sense Multiple Access

- **Basic Idea:** Before transmitting a frame, a node should check whether the channel is occupied by another node

- Many variations of CSMA exist based on how the nodes should act once the channel is found available or not:
  - **Persistent and Non-persistent CSMA**
  - **CSMA with Collision Detection (CSMA-CD)**
  - **CSMA with Collision Avoidance (CSMA-CA)**

Persistent and Non-persistent CSMA

- **1-persistent CSMA**
  - When a node needs to transmit a frame:
    1. Listens to the channel to see if anyone else is transmitting
       1. If the channel is free, it directly sends the frame
       2. If the channel is busy, the station just waits until it becomes idle and then transmits the frame
    2. If there is a collision, the node waits for a random time (back-off time) and starts all over again

This is called 1-persistent because the node will transmit as soon as it finds the channel idle

1-persistent CSMA

- If two nodes decide to transmit a frame in the middle of a third node's transmission:
  - Both will wait until the transmission ends and both will begin transmitting exactly simultaneously…
  - → Collision!

- If after a node starts transmitting a frame, at the same, another node becomes ready to send and senses the channel
  - Due to the propagation delay, it is possible for the second node to find the channel available and start transmitting because the first node signal has not yet reached it
    → Collision!

Persistent and Non-persistent CSMA

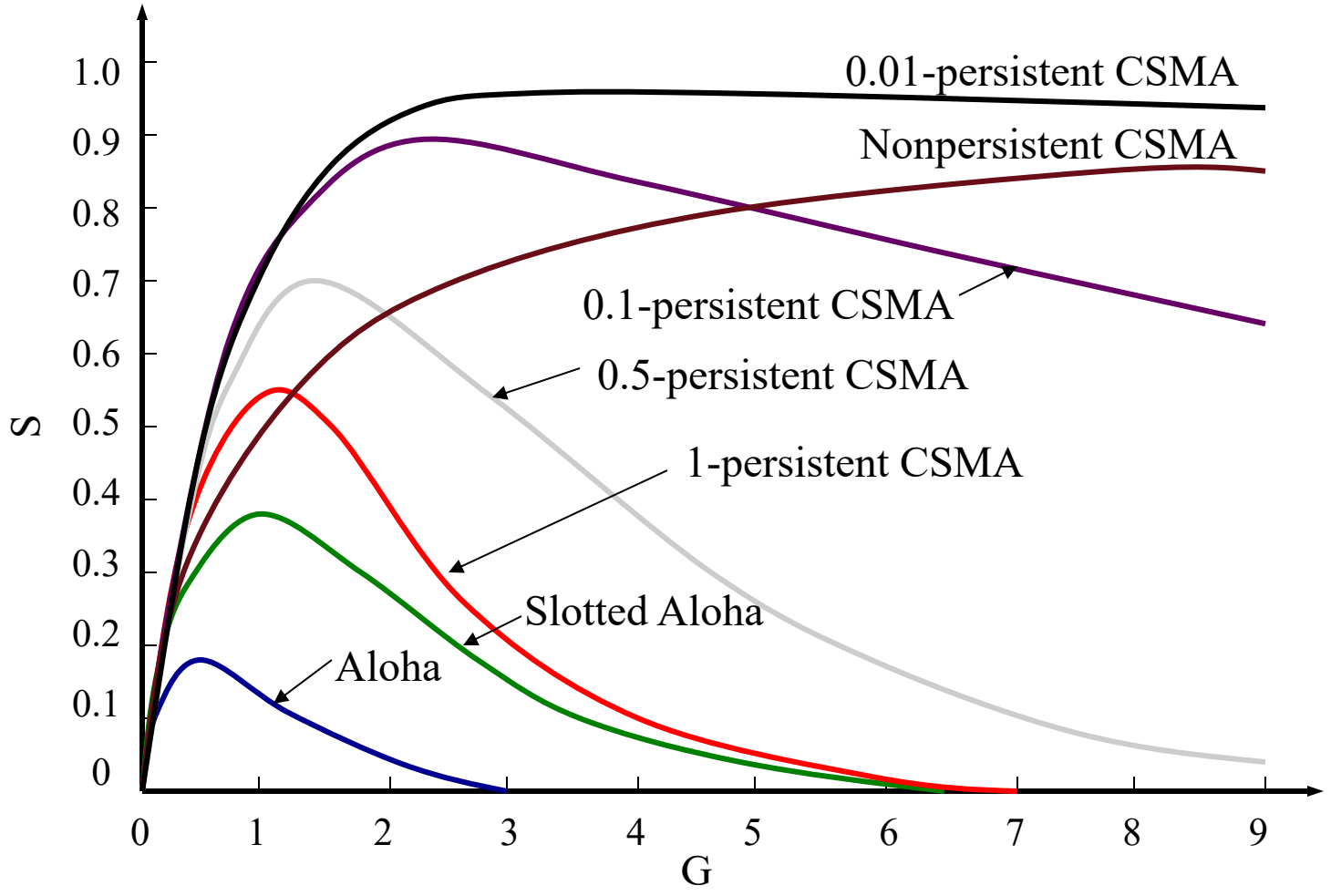
- **Non-persistent CSMA**
  - When a node needs to transmit a frame:
    1. Listens to the channel to see if anyone else is transmitting
       1. If the channel is free, it directly sends the frame
       2. If the channel is busy, it waits a random amount of time and repeats the algorithm
    2. If there is a collision, the node waits for a random time (back-off time) and starts all over again

The channel utilization is higher than 1-persistent CSMA, but expect longer delays

Persistent and Non-persistent CSMA

- **p-persistent CSMA**
  - Applies to slotted channels
  - When a node needs to transmit a frame:
    1. Listens to the channel to see if anyone else is transmitting
       1. If the channel is free,
          1. With probability p, it transmits its frame
          2. With probability q=1-p, it waits till the next slot and repeats the algorithm
       2. If the channel is busy, it waits for the next slot and repeats the algorithm
    2. If there is a collision or another node takes the slot, the node waits for a random time (back-off time) and starts all over again

Throughput Comparison



CSMA with Collision Detection

- Allows nodes to quickly detect that there has been a collision

- **Basic idea:** the sender of a frame listens to the channel while transmitting
  - If the received signal is different than the transmitted signal, there has been a collision

- **Problem:** This only works if the amplitude of the transmitted signal and that of the received signal are comparable in magnitude → **not the case in wireless communications!**

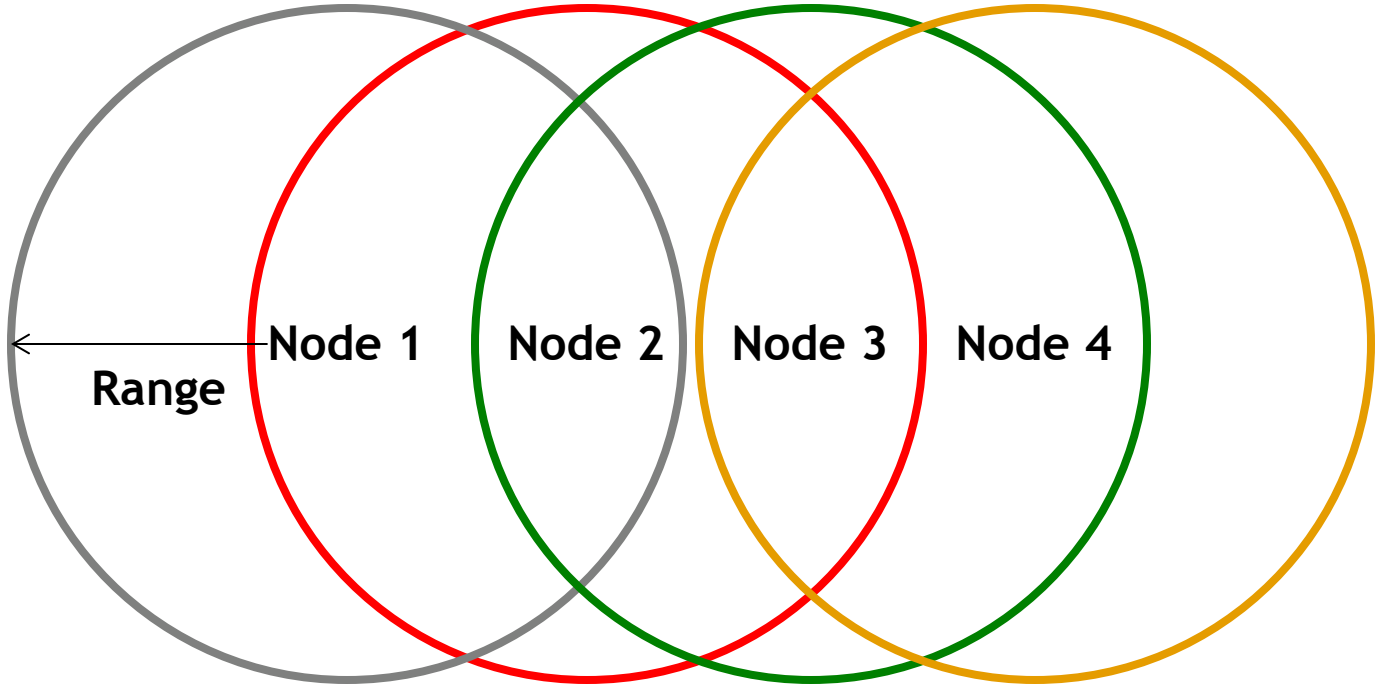
CSMA with Collision Avoidance

- Collisions in wireless networks cannot be usually detected because:
  - Usually a node cannot **transmit and receive simultaneously** when using the same radio interface
  - Even if it could, the received signal strength might be several orders of magnitude weaker than the transmitted signal

  - **In any case… collisions happen at the receiver!**
    - The transmitter might not know/hear the frames received at the receiver node at all…
    - Not all the nodes in the network might be able to hear all the ongoing transmissions…

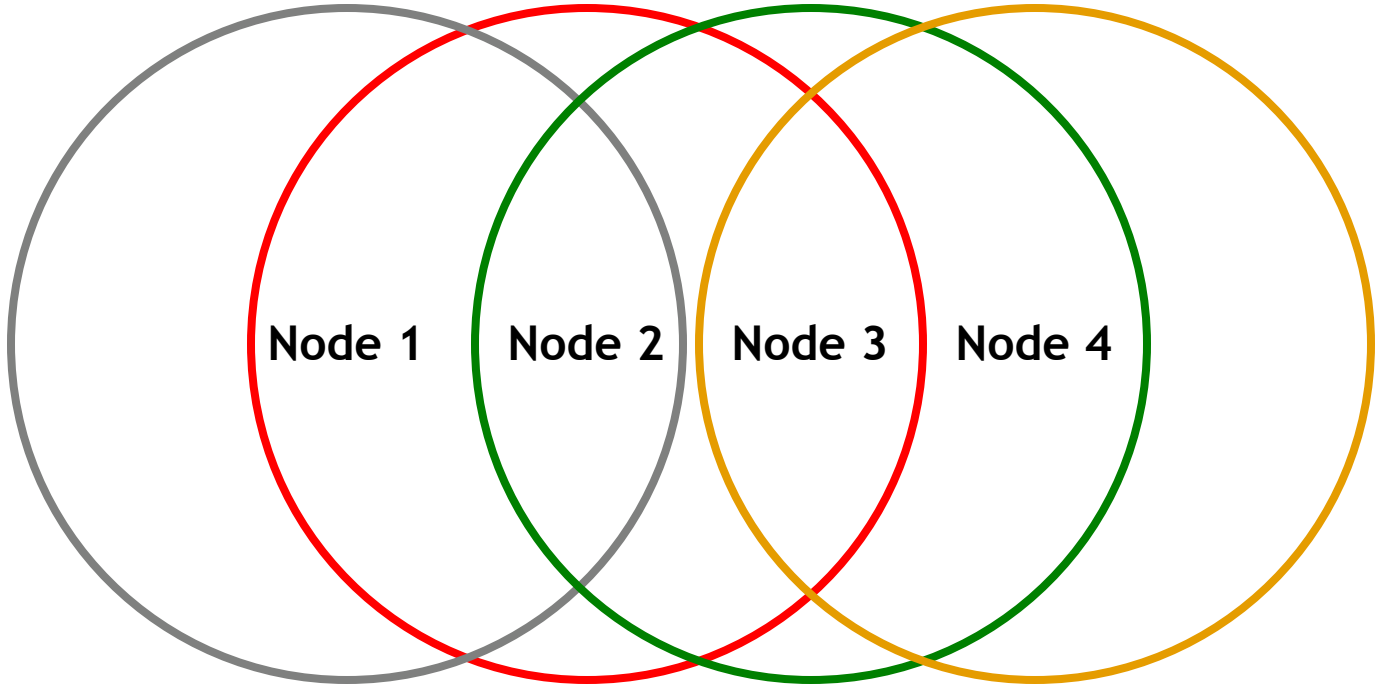
Hidden Terminal Problem

- The transmitter cannot detect a possible channel competitor because the two nodes are too far one from each other:
  - E.g., nodes 1 and 3 transmit at the same time to node 2



Exposed Terminal Problem

- Two nodes might prevent each other from transmitting, despite their transmissions would not collide at the receiver:
  - E.g., nodes 2 and 3 transmit at the same time to node 1 and 4, respectively



CSMA with Collision Avoidance

- When a node needs to transmit a frame:
  1. Listens to the channel to see if anyone else is transmitting
     1. If the channel is free, it waits for a short period of time and transmits its data
     2. If the channel is busy,
        1. The station sets a counter with a random exponential back-off time
        2. While the channel is utilized, the counter is stopped
        3. While the channel is free, the counter decreases its value
        4. When the counter reaches 0, it repeats the algorithm
  2. If there is a collision (no positive ACK received), the node waits for a random truncated binary exponential back-off time, and then repeats the algorithm

Request-to-Send / Clear-to-Send

- **Optionally,** nodes can make use of an initial handshake to avoid data frames collisions

- When a node needs to transmit a frame:
  - Listens to the channel to see if anyone else is transmitting
    - If the channel is free, it waits for a short period of time and transmits a Request-To-Send (RTS) frame to the receiver
    - If the receiver properly receives the RTS and no other nodes are active in its neighborhood, it replies with a Clear-To-Send (CTS) frame after waiting for a very short time
    - If the transmitter properly receives the CTS and no other nodes are active in its neighborhood, it transmits the data frame after waiting for a very short time
    - If the receiver properly receives the data frame and no other nodes are active in its neighborhood, it transmits an ACK frame after waiting for a very short time

  - If there are collisions in any of the steps, the nodes go into an exponential back-off time as in the previous case

Request-to-Send / Clear-to-Send

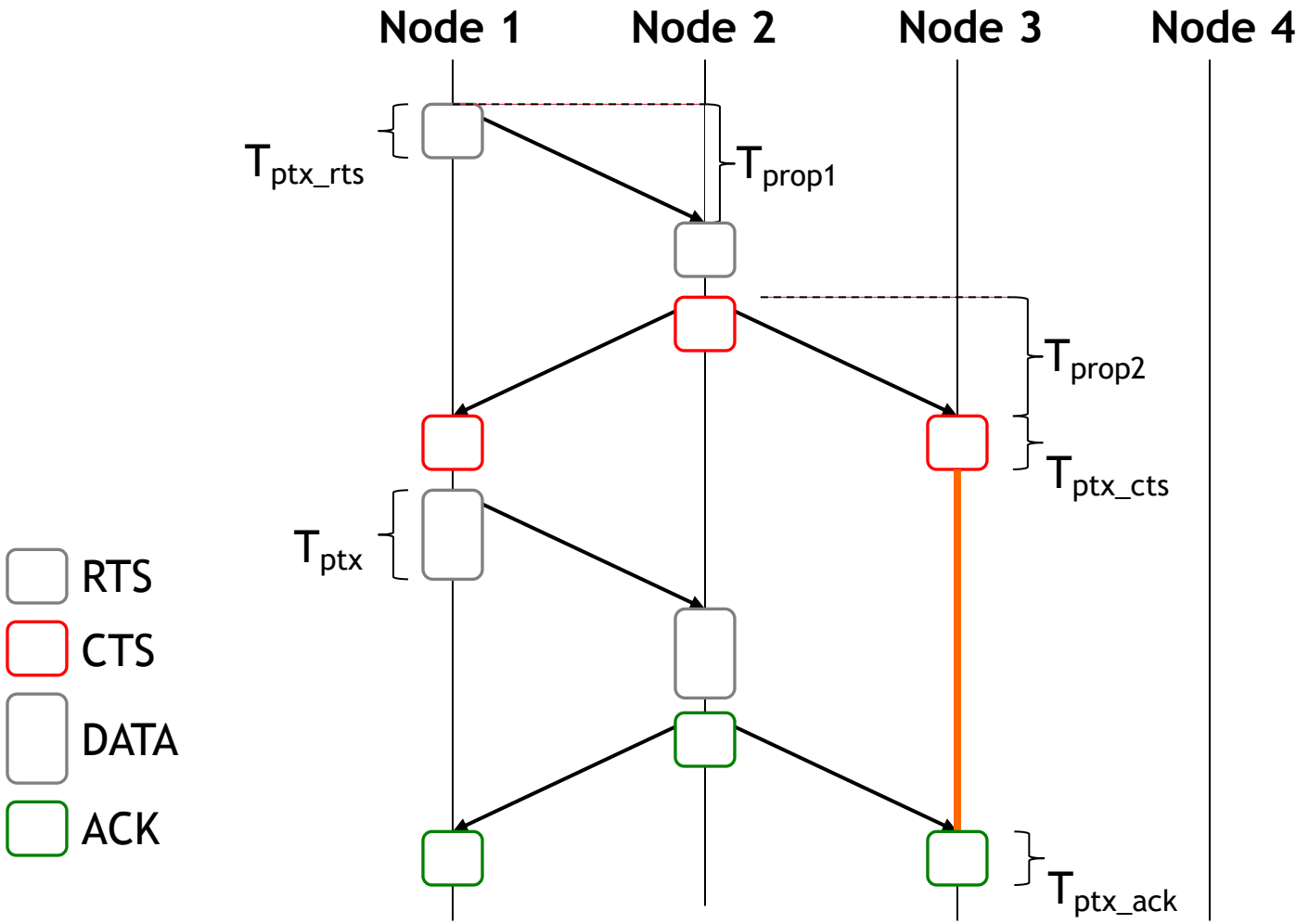
- The **RTS frame** specifies how much data the node needs to transmit (so the receiver can compute for how long will the channel be occupied)

- The **CTS frame** specifies for how long the channel will be occupied (so its neighbors do not attempt to transmit during that time)
  - In the particular case of IEEE 802.11 Wi-Fi Standard, this is called Network Allocation Vector or NAV

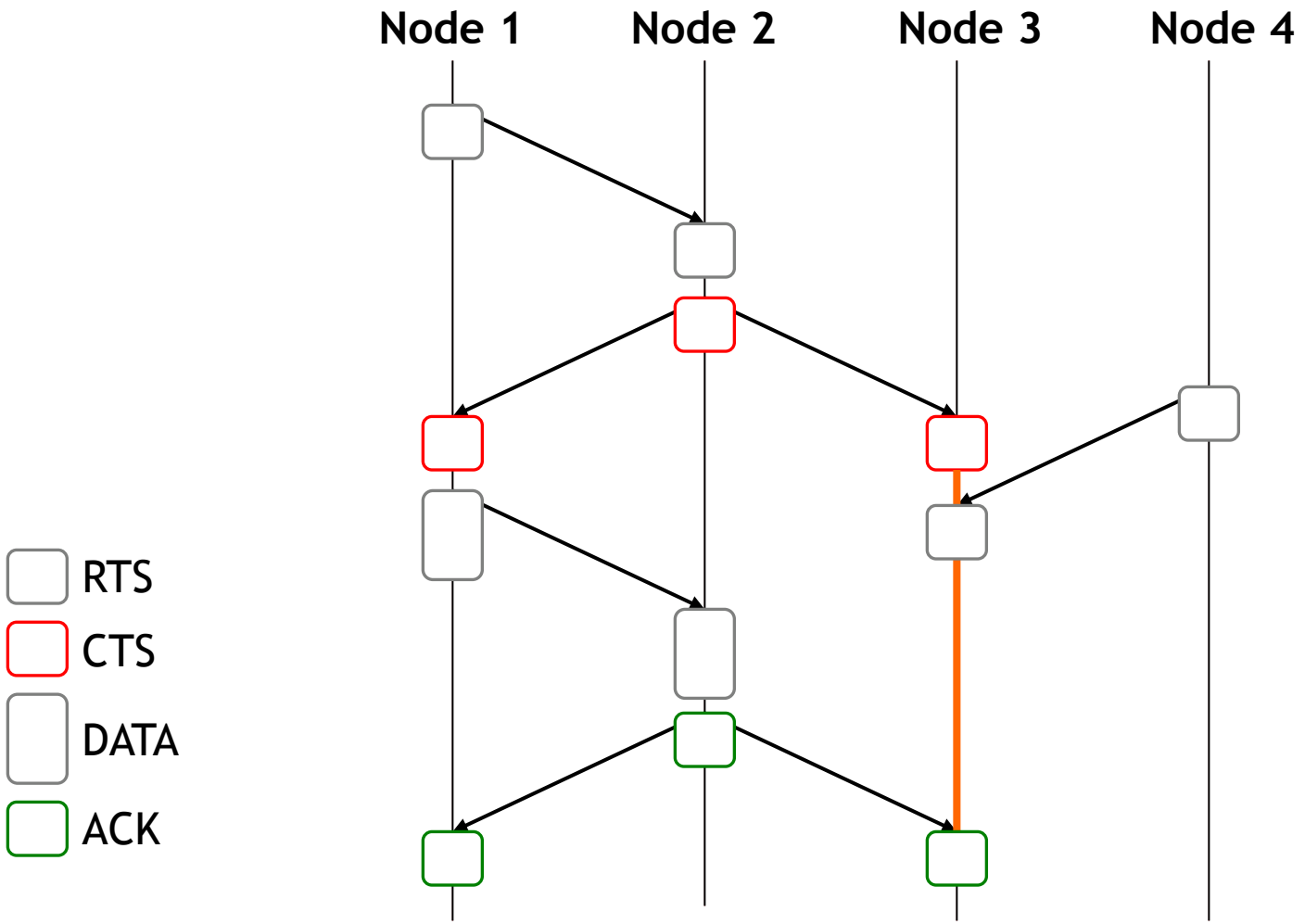
- The length of the RTS and CTS frames is much shorter than the length of the data frame:
  - If they collide, not many "resources" (e.g., time, energy) have been wasted

CSMA with Collision Avoidance with RTS/CTS

Institute for the Wireless Internet of Things at Northeastern

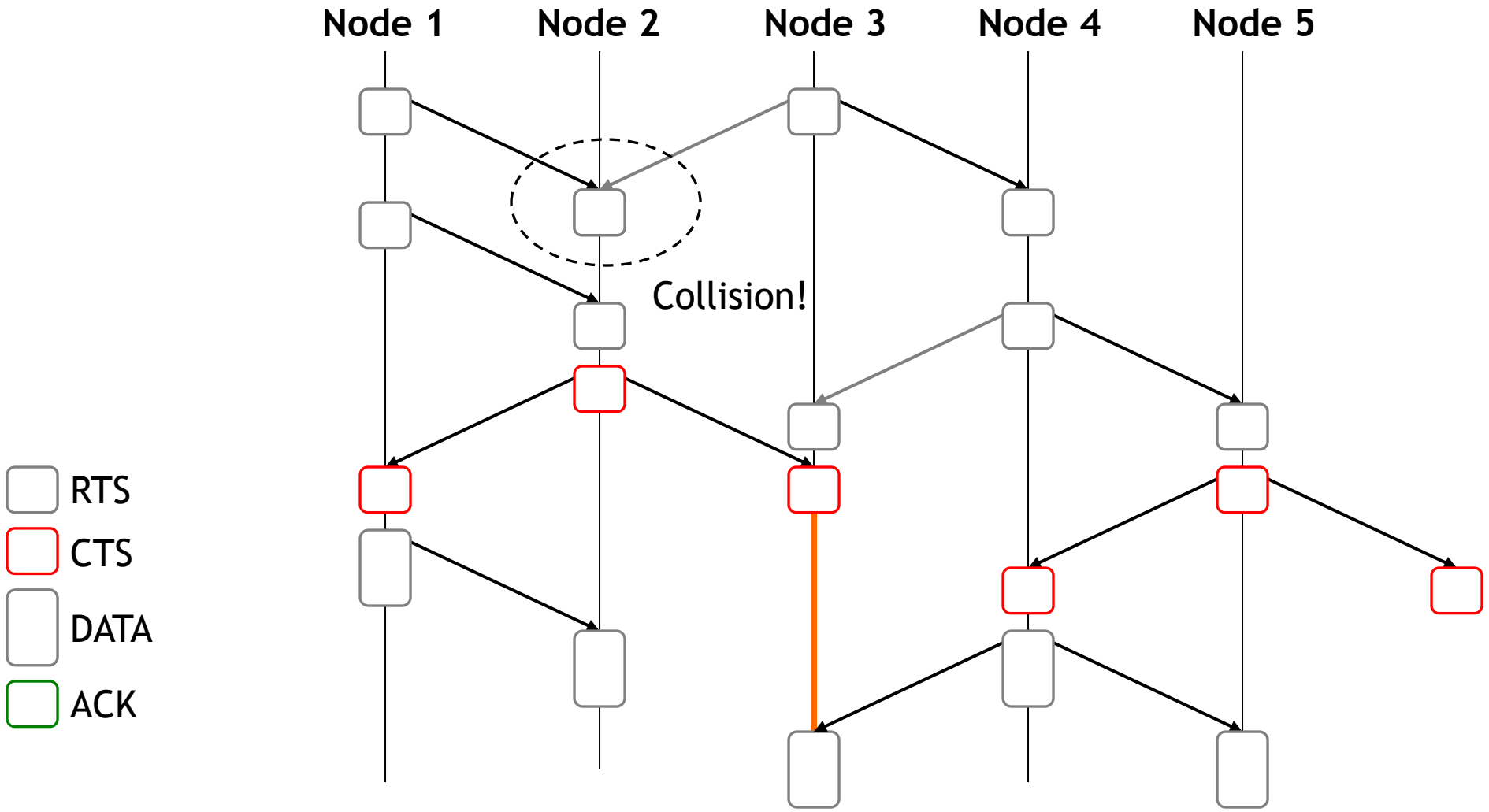


CSMA with Collision Avoidance with RTS/CTS



CSMA with Collision Avoidance with RTS/CTS

Institute for the Wireless Internet of Things at Northeastern



Is RTS/CTS always better?

- Why to "book" the channel?
  - If the network is almost always empty…
    - The time needed to complete the data transaction can actually double! :-(
  - If the network is congested…
    - Collisions among RTS or CTS packets are much shorter, consume less time and less energy

Problem

- A node always needs to be "ready to receive" a message
  - This leads to very high energy consumption…
    - This is a problem particularly for the IoT!

How can we save energy?

S-MAC: Sleep MAC

- **Problem:** "Idle Listening" consumes significant energy
- **Solution:** Periodic listen and sleep

- During sleeping, radio is turned off
- Duty Cycle:10% (Listen for 200ms and sleep for 2s)
- **Result:**
  - Energy? ☺
  - Latency? ☹

S-MAC

- Each node goes into periodic sleep mode during which it switches the radio off and sets a timer to awake later
- When the timer expires it wakes up and listens to see if any other node wants to talk to it
- The duration of the sleep and listen cycles are application dependent and they are set the same for all nodes
- Requires a periodic synchronization among nodes to take care of any type of clock drift
  - I.e., deviation in the actual clock frequency (again, think of cheap clocks for cheap IoT devices)

Periodic Sleep and Listen

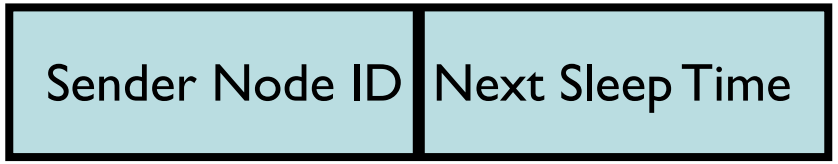
- All nodes are free to choose their own listen/sleep schedules
- To reduce control overhead, only neighboring nodes are synchronized together
- Preferably, they listen at the same time and go to sleep at the same time

Synchronization

Institute for the Wireless Internet of Things at Northeastern

- SYNC packets are exchanged periodically to maintain schedule synchronization

SYNC PACKET



- **Synchronization period:** Period for a node to send a SYNC packet
- Receivers will adjust their timer counters immediately after they receive the SYNC packet

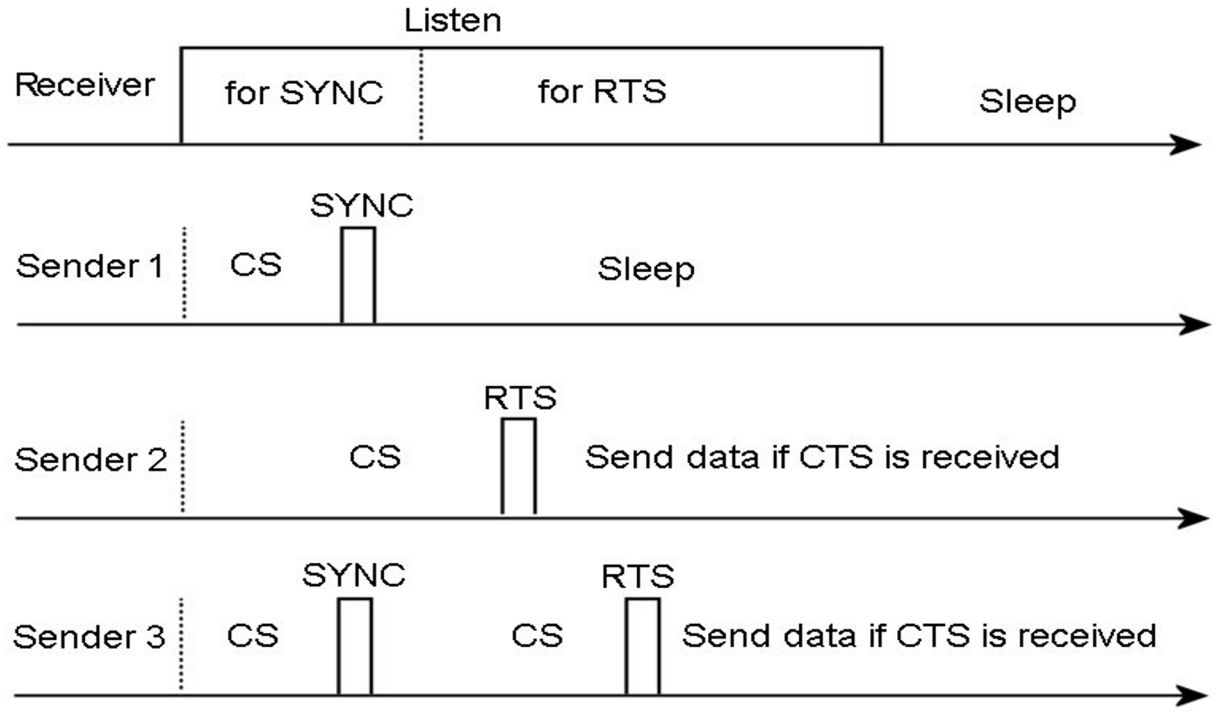
Choosing and Maintaining Schedules

Institute for the Wireless Internet of Things at Northeastern

- Each node maintains a schedule table that stores schedules of all its known neighbors
- For initial schedule, do:
  - A node first listens to the medium for a certain amount of time (at least the synchronization period)
  - If it does not hear a schedule from another node, it randomly chooses a schedule and broadcasts its schedule with a SYNC packet immediately
  - This node is called a Synchronizer
  - If a node receives a schedule from a neighbor before choosing its own schedule, it just follows this neighbor's schedule, i.e. becomes a Follower, waits for a random delay and broadcasts its schedule

Collision Avoidance

- S-MAC is based on contention, i.e., if multiple neighbors want to talk to a node at the same time, they will try to send when the node starts listening
  - Similar to CSMA/CA, i.e. use RTS/CTS mechanism to address the hidden terminal problem
- Perform carrier sense before initiating a transmission



Some Thoughts on MAC

Institute for the Wireless Internet of Things at Northeastern

- There are (tens of) thousands of MAC protocols for wireless sensor networks and, by extension, for the Iot
- All these protocols are based on the same fundamental ideas that we have seen in this module, just with some fine tuning (e.g., "salt and pepper").
- If you understand the foundations, you can understand any existing protocol.