
# Data Link Layer: Error Detection, Multiple Access

A/PROF. DUY NGO

# Link Layer and LANs

**our goals:**

understand principles behind link layer services:

- error detection, correction
- sharing a broadcast channel: multiple access
- link layer addressing
- local area networks: Ethernet, VLANs

instantiation, implementation of various link layer technologies

# Learning Objectives (1 of 9)

**6.1 introduction, services**

**6.2** error detection, correction

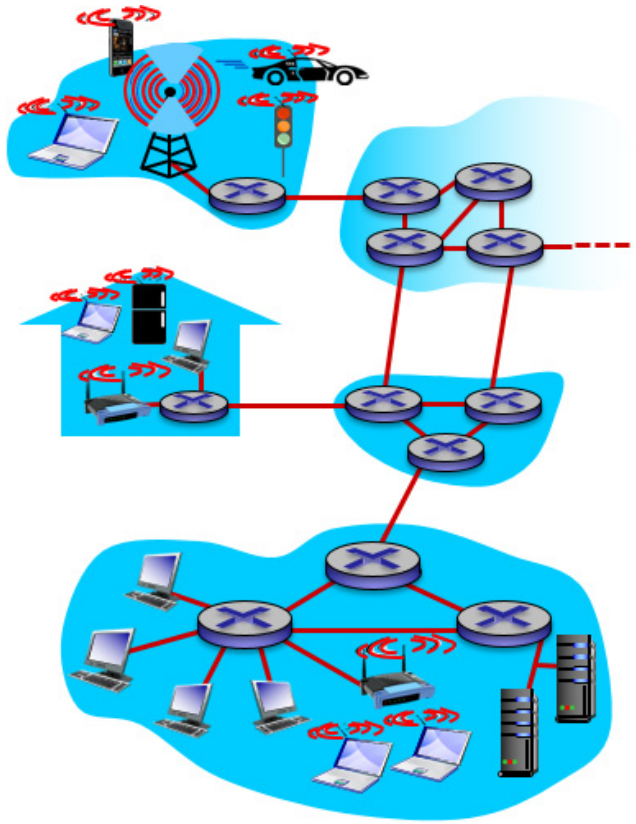
**6.3** multiple access protocols

# Link Layer: Introduction

**terminology:**

- hosts and routers: **nodes**
- communication channels that connect adjacent nodes along communication path: **links**
  - wired links
  - wireless links
  - LANs
- layer-2 packet: **frame,** encapsulates datagram

**data-link layer** has responsibility of transferring datagram from one node to **physically adjacent** node over a link



# Link Layer: Context

- datagram transferred by different link protocols over different links:
  - e.g., Ethernet on first link, frame relay on intermediate links, 802.11 on last link
- each link protocol provides different services
  - e.g., may or may not provide rdt over link

**transportation analogy:**

- trip from Princeton to Lausanne
  - limo: Princeton to JFK
  - plane: JFK to Geneva
  - train: Geneva to Lausanne
- tourist = **datagram**
- transport segment = **communication link**
- transportation mode = **link layer protocol**
- travel agent = **routing algorithm**

# Link Layer Services (1 of 2)

**framing, link access:**

- encapsulate datagram into frame, adding header, trailer
- channel access if shared medium
- "MAC" addresses used in frame headers to identify source, destination
  - different from IP address!

**reliable delivery between adjacent nodes**

- we learned how to do this already (chapter 3)!
- seldom used on low bit-error link (fiber, some twisted pair)
- wireless links: high error rates
  - **Q:** why both link-level and end-end reliability?

# Link Layer Services (2 of 2)

**flow control:**

- pacing between adjacent sending and receiving nodes

**error detection:**

- errors caused by signal attenuation, noise.
- receiver detects presence of errors:
  - signals sender for retransmission or drops frame

**error correction:**

- receiver identifies **and corrects** bit error(s) without resorting to retransmission

**half-duplex and full-duplex**

- with half duplex, nodes at both ends of link can transmit, but not at same time

# Where is the Link Layer Implemented?

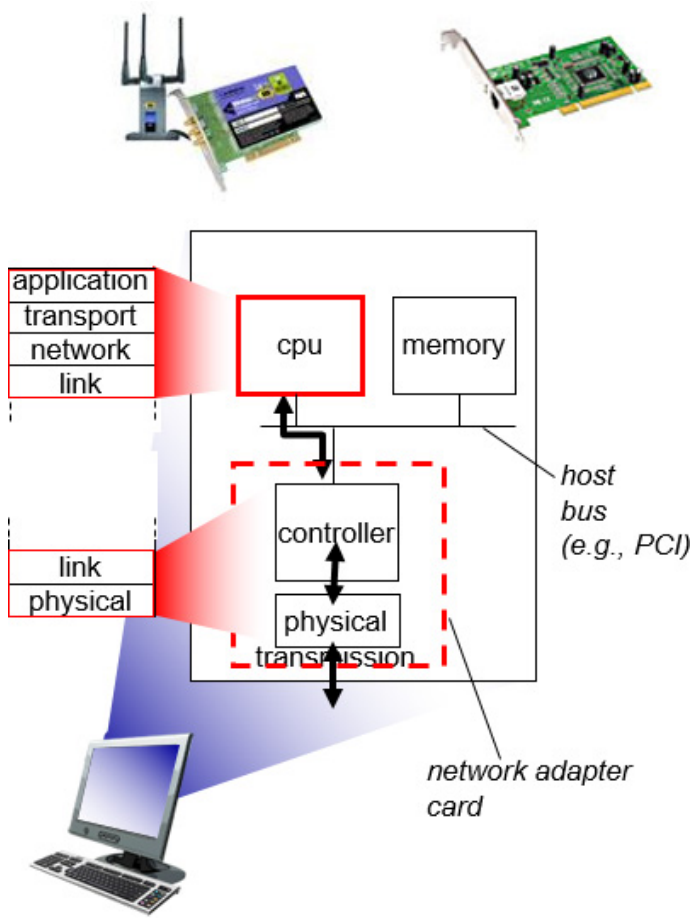
in each and every host

link layer implemented in “adaptor” (aka **network interface card** NIC) or on a chip

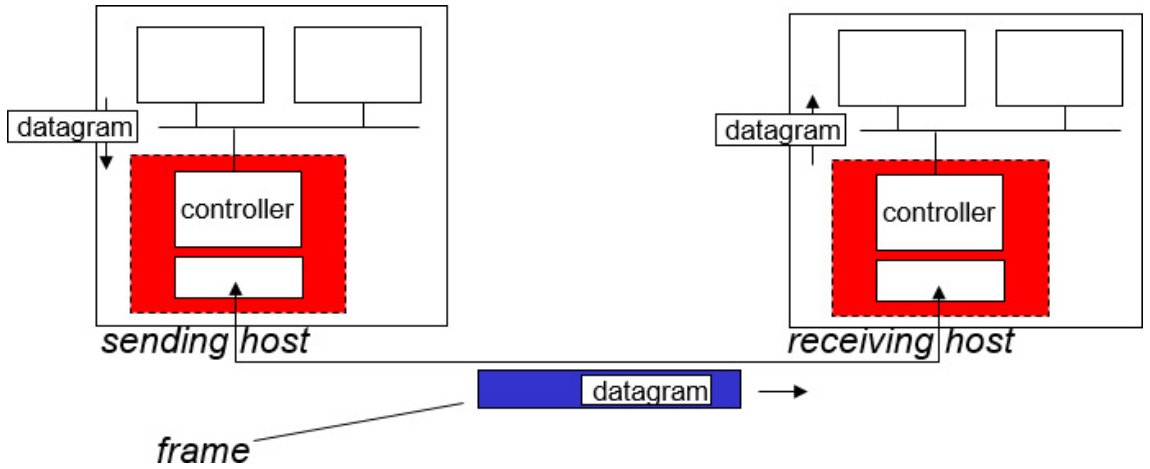
- Ethernet card, 802.11 card; Ethernet chipset
- implements link, physical layer

attaches into host’s system buses

combination of hardware, software, firmware



# Adaptors Communicating



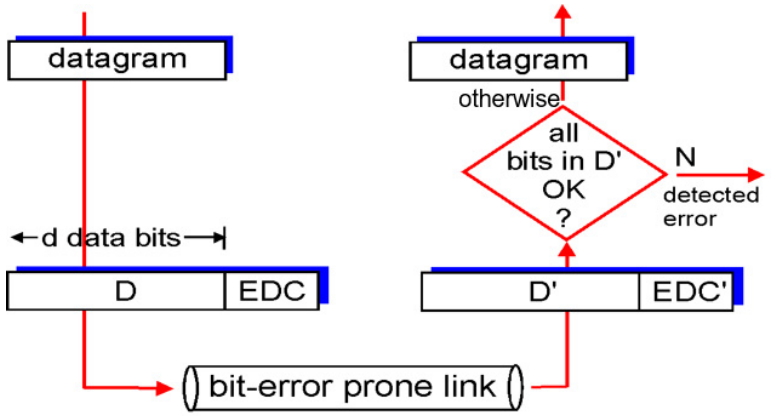

- sending side:
  - encapsulates datagram in frame
  - adds error checking bits, rdt, flow control, etc.
- receiving side
  - looks for errors, rdt, flow control, etc.
  - extracts datagram, passes to upper layer at receiving side

# Error Detection

EDC = Error Detection and Correction bits (redundancy)

D = Data protected by error checking, may include header fields

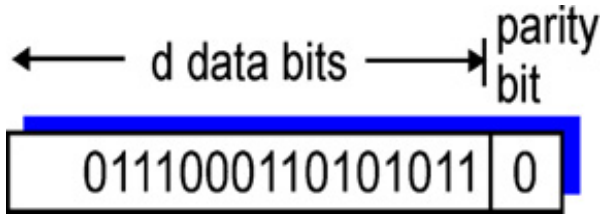
- Error detection not 100% reliable!
  - protocol may miss some errors, but rarely
  - larger EDC field yields better detection and correction



# Parity Checking

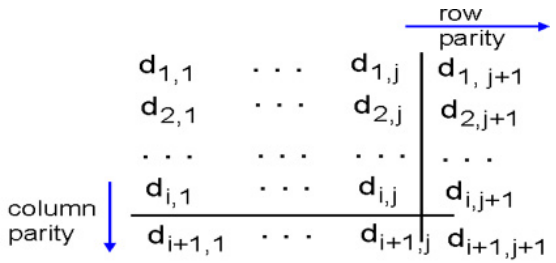
**single bit parity:**

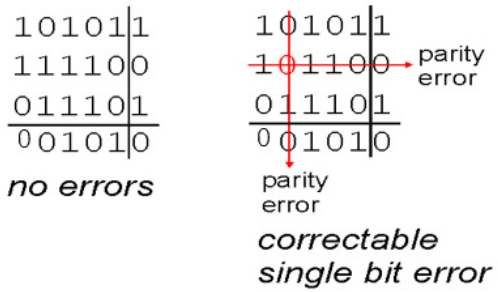
detect single bit errors




**two-dimensional bit parity:**

detect and correct single bit errors




* Check out the online interactive exercises for more examples: http://gaia.cs.umass.edu/kurose_ross/interactive/

# Internet Checksum (Review)

**goal:** detect "errors" (e.g., flipped bits) in transmitted packet (note: used at transport layer only)

**sender:**

- treat segment contents as sequence of 16-bit integers
- checksum: addition (1's complement sum) of segment contents
- sender puts checksum value into UDP checksum field

**receiver:**

- compute checksum of received segment
- check if computed checksum equals checksum field value:
  - NO - error detected
  - YES - no error detected. **But maybe errors nonetheless?**

# Cyclic Redundancy Check

more powerful error-detection coding

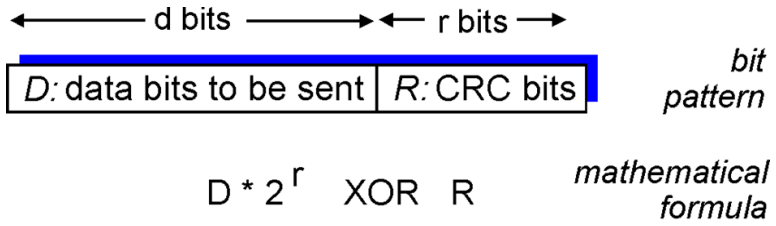
view data bits, **D**, as a binary number

choose r+1 bit pattern (generator), **G**

goal: choose r CRC bits, **R**, such that

- <D,R> exactly divisible by G (modulo 2)
- receiver knows G, divides <D,R> by G. If non-zero remainder: error detected!
- can detect all burst errors less than r+1 bits

widely used in practice (Ethernet, 802.11 WiFi, ATM)



# CRC Example

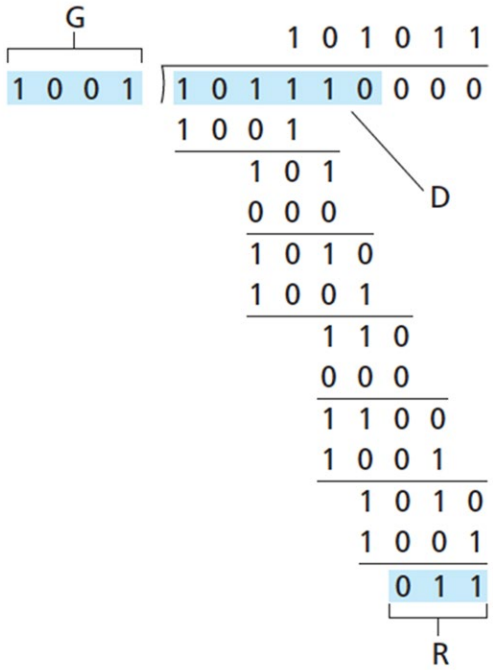
**want:** $D.2^r \text{ XOR } R = nG$

**equivalently:** $D.2^r = nG \text{ XOR } R$

**equivalently:** if we divide $D.2^r$

by G, want remainder R to satisfy:

$$R = \text{remainder}\left[\frac{D.2^r}{G}\right]$$




* Check out the online interactive exercises for more examples: http://gaia.cs.umass.edu/kurose_ross/interactive/

# Multiple Access Links, Protocols

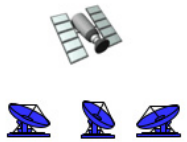
two types of “links”:

point-to-point
- PPP for dial-up access
- point-to-point link between Ethernet switch, host

**broadcast (shared wire or medium)**
- old-fashioned Ethernet
- upstream HFC
- 802.11 wireless LAN

shared wire (e.g., cabled Ethernet)

shared RF (e.g., 802.11 WiFi)



shared RF (satellite)

humans at a cocktail party (shared air, acoustical)

# 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

**multiple access protocol**

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

# An Ideal Multiple Access Protocol

**given:** broadcast channel of rate R bps

**MAC scenario:**

1. when one node wants to transmit, it can send at rate R.
2. when M nodes want to transmit, each can send at average rate R/M
3. fully decentralized:
   - no special node to coordinate transmissions
   - no synchronization of clocks, slots
4. simple

# MAC Protocols: Taxonomy

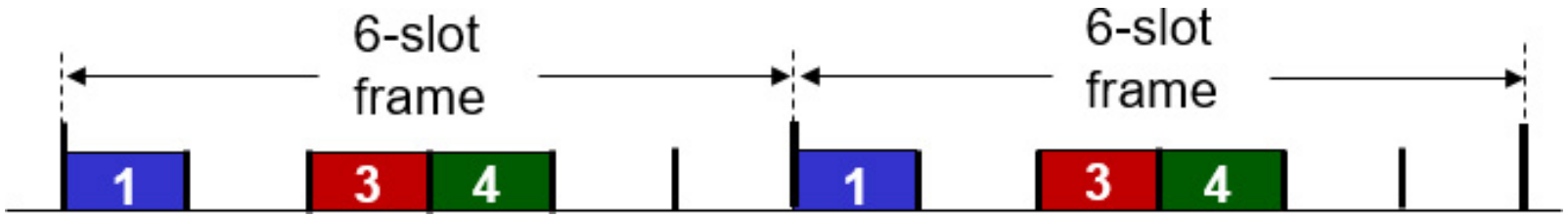
Three broad classes:

- **channel partitioning**
  - divide channel into smaller “pieces” (time slots, frequency, code)
  - allocate piece to node for exclusive use
- **random access**
  - channel not divided, allow collisions
  - “recover” from collisions
- **“taking turns”**
  - nodes take turns, but nodes with more to send can take longer turns

# Channel Partitioning MAC Protocols: TDMA

**TDMA: time division multiple access**

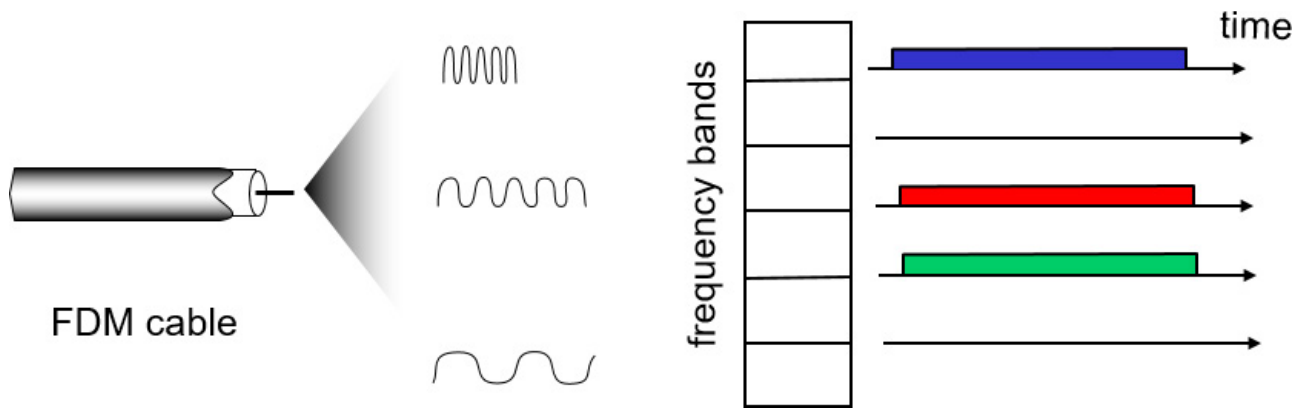
- access to channel in "rounds"
- each station gets fixed length slot (length = packet transmission time) in each round
- unused slots go idle
- example: 6-station LAN, 1,3,4 have packets to send, slots 2,5,6 idle



# Channel Partitioning MAC Protocols: FDMA

**FDMA: frequency division multiple access**

- channel spectrum divided into frequency bands
- each station assigned fixed frequency band
- unused transmission time in frequency bands go idle
- example: 6-station LAN, 1,3,4 have packet to send, frequency bands 2,5,6 idle



# Random Access Protocols

- when node has packet to send
  - transmit at full channel data rate R.
  - no *a* **priori** coordination among nodes
- two or more transmitting nodes → "collision"
- random access MAC protocol specifies:
  - how to detect collisions
  - how to recover from collisions (e.g., via delayed retransmissions)
- examples of random access MAC protocols:
  - ALOHA
  - Slotted ALOHA
  - CSMA, CSMA/CD, CSMA/CA

# Slotted ALOHA

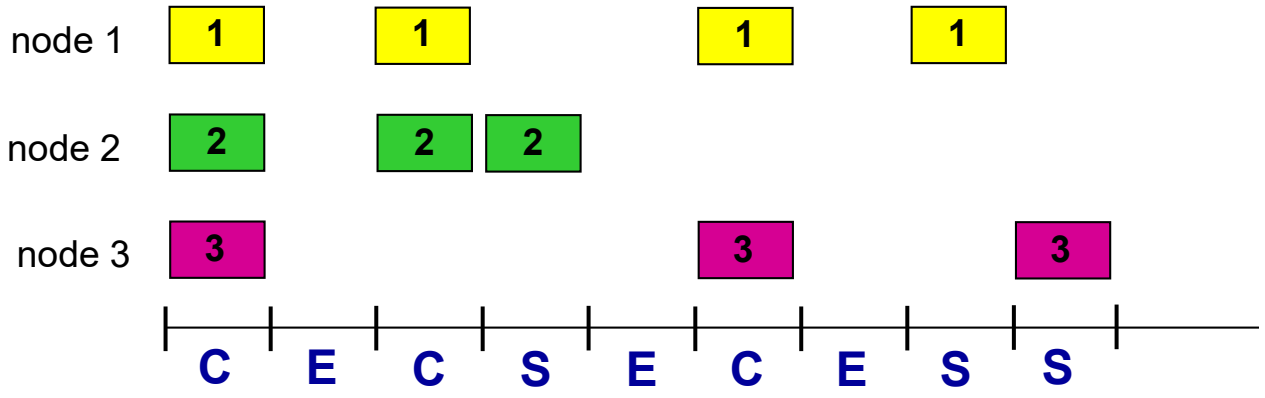
*assumptions:*

- all frames same size
- time divided into equal size slots (time to transmit one frame)
- nodes start to transmit only slot beginning
- nodes are synchronised
- if 2 or more nodes transmit in slot, all nodes detect collision

*operation:*

- when node obtains fresh frame, transmits in next slot
  - *if no collision:* node can send a new frame in the next slot
  - *if collision:* node retransmits the frame in each subsequent slot with probability *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*: long-run fraction of successful slots (many nodes, all with many frames to send)

- *suppose:* $N$ nodes with many frames to send, each transmits in slot with probability $p$
- prob that a given node has success in a slot = $p(1-p)^{N-1}$
- prob that *any* node has a success = $Np(1-p)^{N-1}$
- max efficiency: find $p^*$ that maximises $Np(1-p)^{N-1}$
- for many nodes, take limit of $Np^*(1-p^*)^{N-1}$ as $N$ goes to infinity, gives:

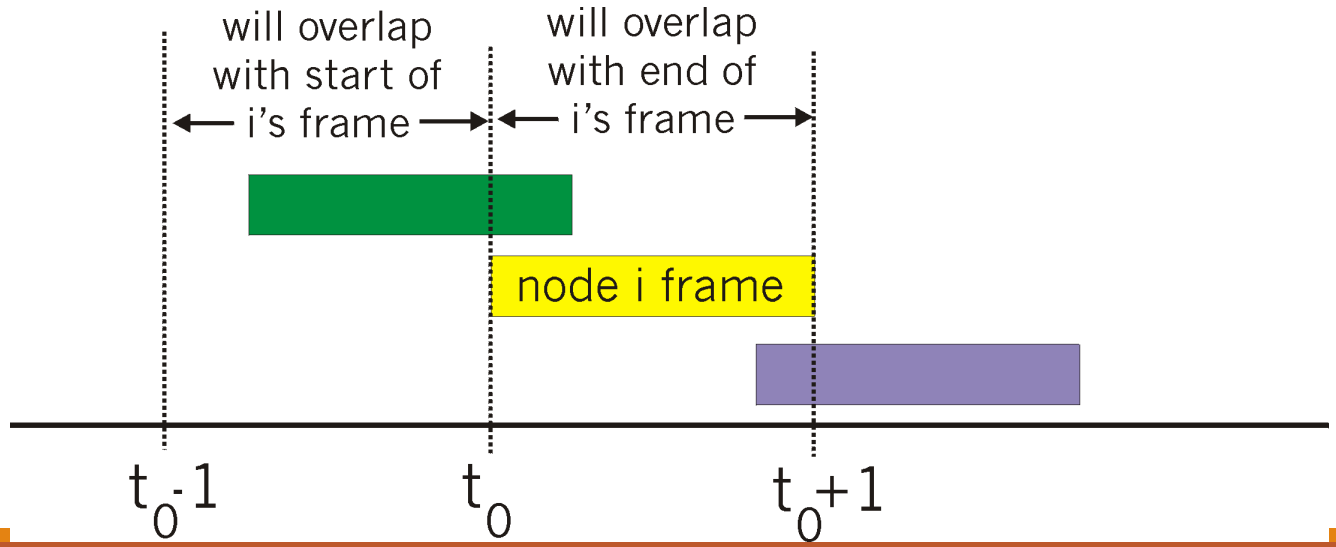
*max efficiency = 1/e = .37*

*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 $[t_0-1, t_0+1]$



# Pure ALOHA efficiency

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

P(no other node transmits in $[t_0\text{-}1, t_0]$) ×

P(no other node transmits in $[t_0\text{-}1, t_0]$)

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

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

By choosing optimum p and then letting $n \rightarrow \infty$

Maximum efficiency = 1/(2e) = .18

even *worse* than slotted Aloha!