Technische Universität Berlin

# Computer Networks

Data Link Layer

# Chapter

1. Introduction
2. Protocols
3. Application layer
4. Web services
5. Distributed hash tables
6. Time synchronization
7. Error control
8. Transport layer
9. Network layer
10. Internet protocol
11. **Data link layer**
    - **Addressing**
    - **Error control**
    - **Medium access**
12. WLAN

Top-Down-Approach

| |
|---|
| Application Layer |
| Presentation Layer |
| Session Layer |
| Transport Layer |
| Network Layer |
| Data link Layer |
| Physical Layer |

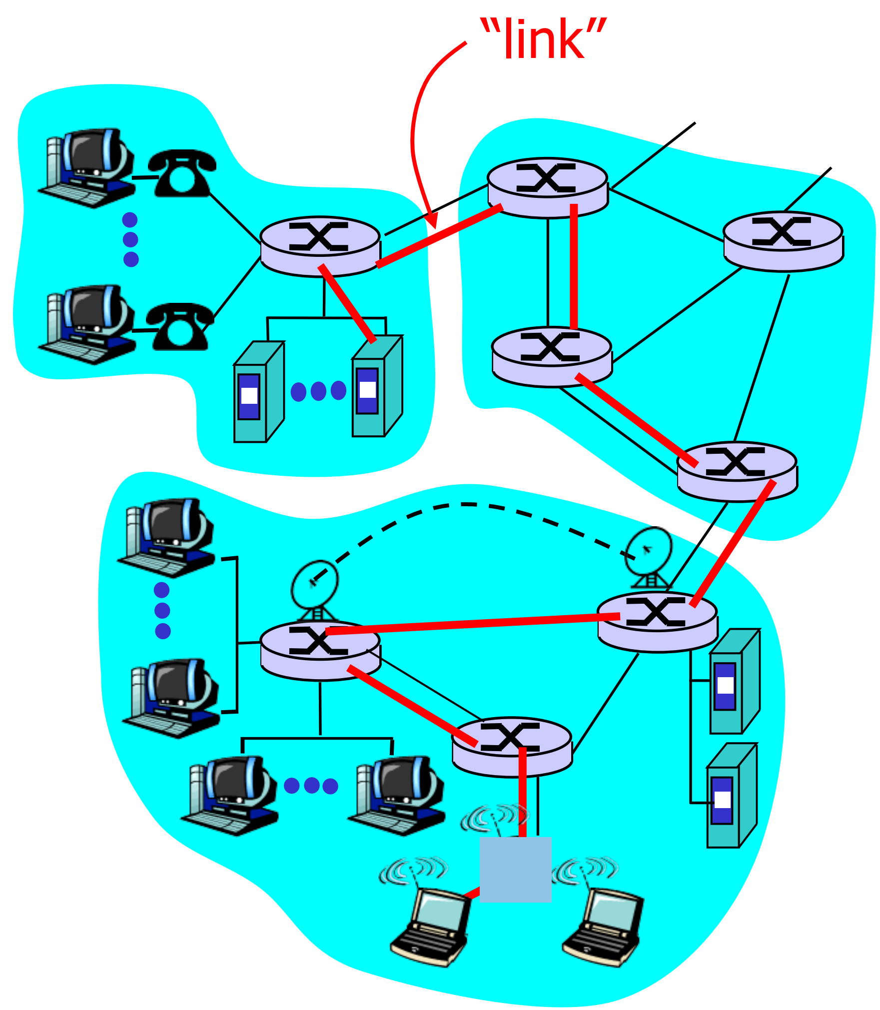
# Data Link Layer

# Data Link Layer

- Terminology
  - Hosts and routers are **nodes**
  - Communication over **links**
  - Protocol data units are called **frames**

- Examples
  - Local networks, Ethernet
  - Optical networks
  - WiFi

- Data link layer is responsible for transfer of frames over these links to one or more nodes



# Data Link Layer

- Tasks of data link layer
  - Transmit frames from network layer, receive frames for network layer
  - **Addressing**: every frame contains the **physical address** of the nodes
  - **Error control**
  - **Medium access (MAC)**
  - Sometimes also flow control

- Network interface
  - Most functions implemented in network interface adapter, but in part in the operating system (driver)
    - Often mix of HW/SW/FPGA
  - Programmed I/O (PIO): CPU transfers data form memory to adapter using registers and interrupts
  - Direct Memory Access (DMA): adapter reads and writes by itself

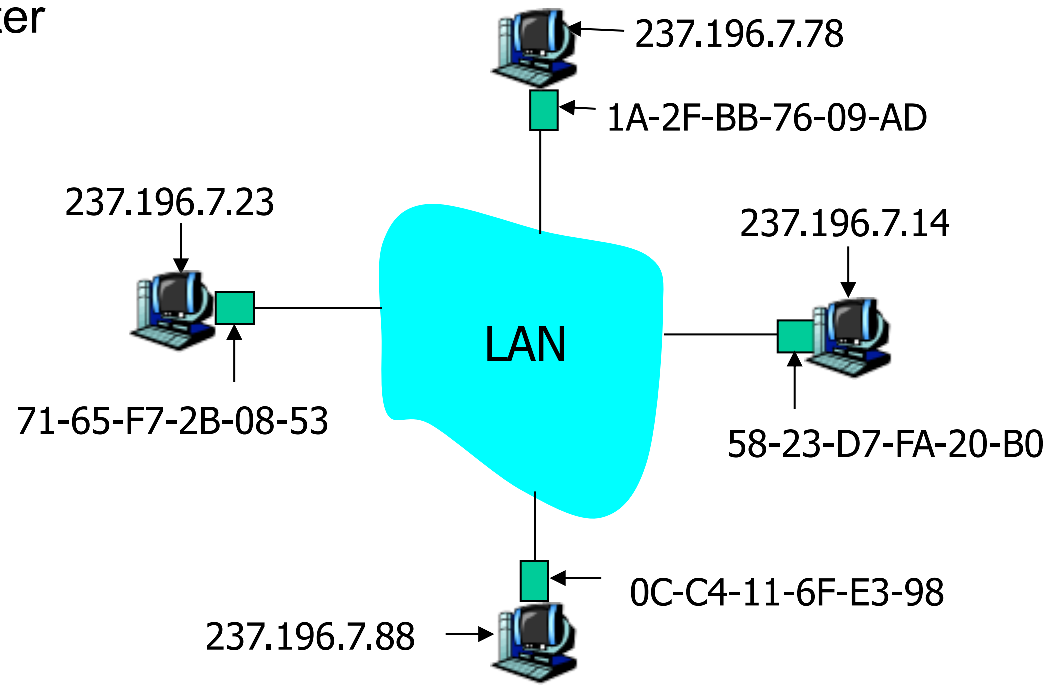
# Data Link Layer

- Realized in form of WANs, LANs, PANs, BANs, …
  - **Local Area Network (LAN)**
    - **Switched Ethernet** (star or tree topology, hubs, switches, twisted pair, fiber), IEEE 802.3a …
    - **Wireless LAN / WiFi, IEEE 802.11** …
  - **Personal Area Network (PAN)**
    - Bluetooth, IEEE 802.15.1: connecting ”personal” devices, data plus multimedia
    - ZigBee, IEEE 802.15.4: mostly used for internet of things (IoT), low data rates but also low energy footprint

# Addressing

# Addressing

- Physical address
  - Also known as MAC address or LAN address
  - 48 bit / 6 byte, written as 6 hexadecimal numbers
  - Fix in ROM of interface adapter
  - Assigned by IEEE, usually in blocks to different vendors
  - Globally unique
  - No logical structure
  - $\neq$ IP address!
  - Question: if IP address of destination is known, how do we get the MAC address?



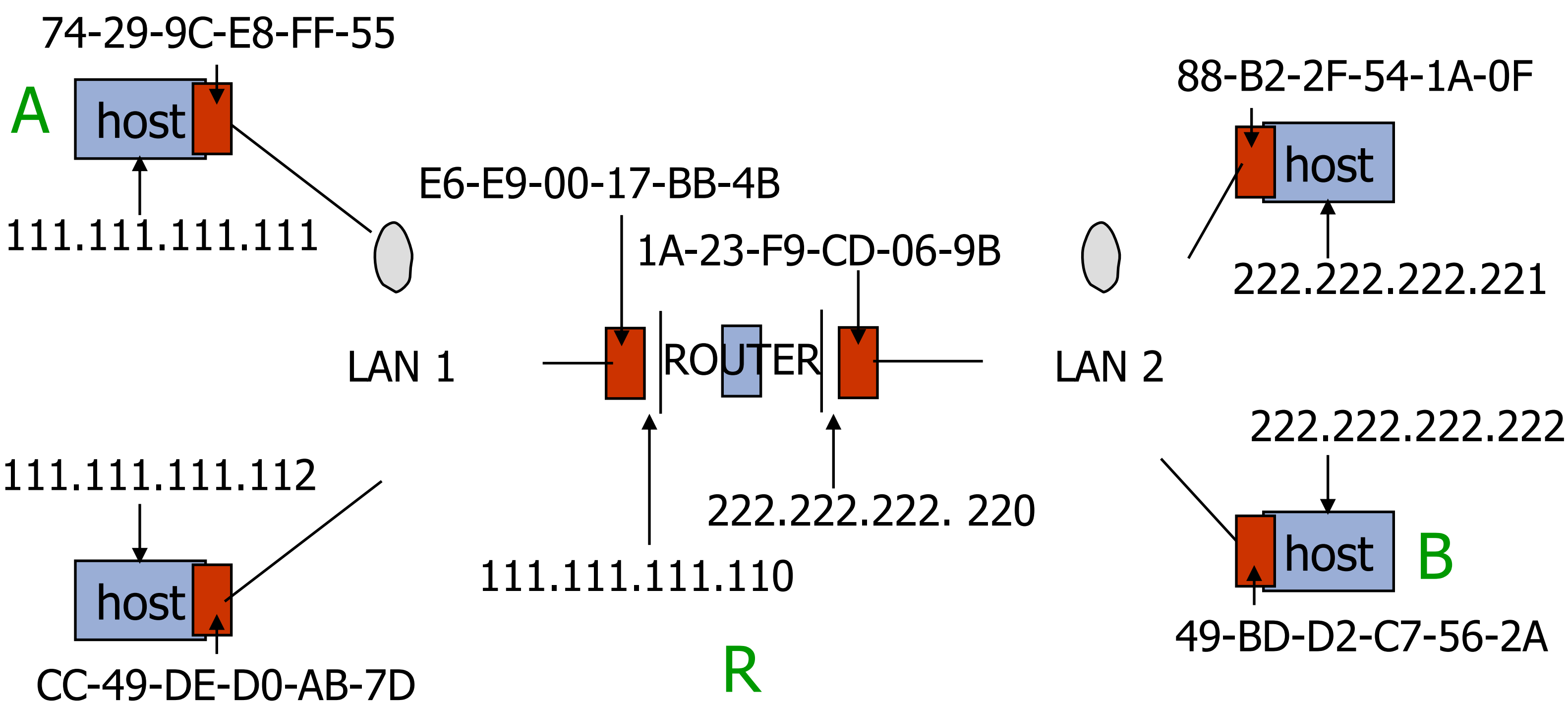
# Address Resolution Protocol (ARP)

- Every node has ARP table (IP address, physical address, TTL)
- Time To Live (TTL): validity of entry (e.g., 20 minutes)

- Node A wants to send frame to node B but only knows IP address of B
- A sends ARP request as a broadcast (address FF-FF-FF-FF-FF-FF) containing its own MAC address and the IP address of B
- B identifies itself based on the IP address, sends ARP reply containing its own MAC address as unicast to A
- A saves MAC to IP address relation in its ARP table
- Soft state approach!

# ARP: Example

- A in LAN1 sends IP packet from A to B in LAN2 via R
- A knows IP address of B
- R requires an ARP table for every interface (**why?**)






Telecommunication Networks Group

# ARP: Example (continued)

- A wants to send IP packet from A to B
- A identifies R using its routing table (network layer!)
- A uses ARP to find MAC address of R
- A sends frame with its own MAC address as source and MAC address of R as destination (original IP packet is encapsulated in this MAC frame)
- R receives the frame, extracts IP packet and, using its routing table, figures that B is in LAN2
- R uses ARP to find MAC address of B
- R sends from with its own MAC address as source and MAC address of B as destination (again, original IP address is encapsulated in this MAC frame)
- B receives the frame, extracts IP packet and continues processing on the network layer

# Error Control

# Error Control

- Source of (bit) errors
  - Thermal noise, electromagnetic interference, radioactive radiation
  - Typical bit error probabilities: $10^{-3}$ radio waves to $10^{-12}$ fiber optic
  - Bit errors mostly in **bursts**

- Error control
  - **Error detection**: checksums to identify bit errors (e.g., parity bits, cyclic redundancy check)
  - **Error correction**: forward error correction using coding, receiver can detect and (partially) correct errors
    - n bit user data transmitted in m bit frames, m > n
    - Degree of redundancy larger than just error detection, good for very noisy channels and for high latency requirements, but overhead!

# Cyclic Redundancy Check (CRC)

- Principles
  - Bit sequences are interpreted as coefficients of a binary polynom: $(b_{n-1}, b_{n-2}, \ldots, b_1, b_0)$ is interpreted as $b_{n-1}\, x^{n-1} + b_{n-2}\, x^{n-2} + \ldots + b_1\, x^1 + b_0\, x^0$ e.g., (10011001) can be written as $x^7+x^4+x^3+x^0$
  - Payload data D with d bit, checksum R with r bit, sending of (D,R)
  - Generator polynom G, r+1 bit
  - Sender selects R such that (D,R) can be divided by G without remainder:
    - R is remainder of $D \cdot 2^r \div G$
    - Now (D,R) is $D \cdot 2^r + R$ can can be divided by G without remainder
  - Receiver divides (D,R) by G, no error in case of remainder = 0

| ← d bit → | ← r bit → |
|---|---|
| D: data bit to be sent | R: CRC |

$D * 2^r$ XOR R

# Cyclic Redundancy Check (CRC)

- Implementation
  - Euclidian division can be done very fast in HW using shift registers and XOR gates
  - Examples for generator polynomes:
    - CRC-8 = $x^8 + x^2 + x + 1$
    - CRC-12 = $x^{16} + x^{15} + x^2 + 1$
    - CCITT-16 = $x^{16} + x^{12} + x^5 + 1$
    - CCITT-32 = $x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$
- Properties
  - Single bit errors can be detected if if coefficients of $x^r$ and $x^0$ are one
  - Double bit errors can be detected if G contains indivisible factor of at least 3 terms
  - Even numbered bit errors can be detected if G contains factor (x+1)
  - Error bursts shorter than r bit can be detected



Telecommunication Networks Group

# Medium Access

# Medium Access

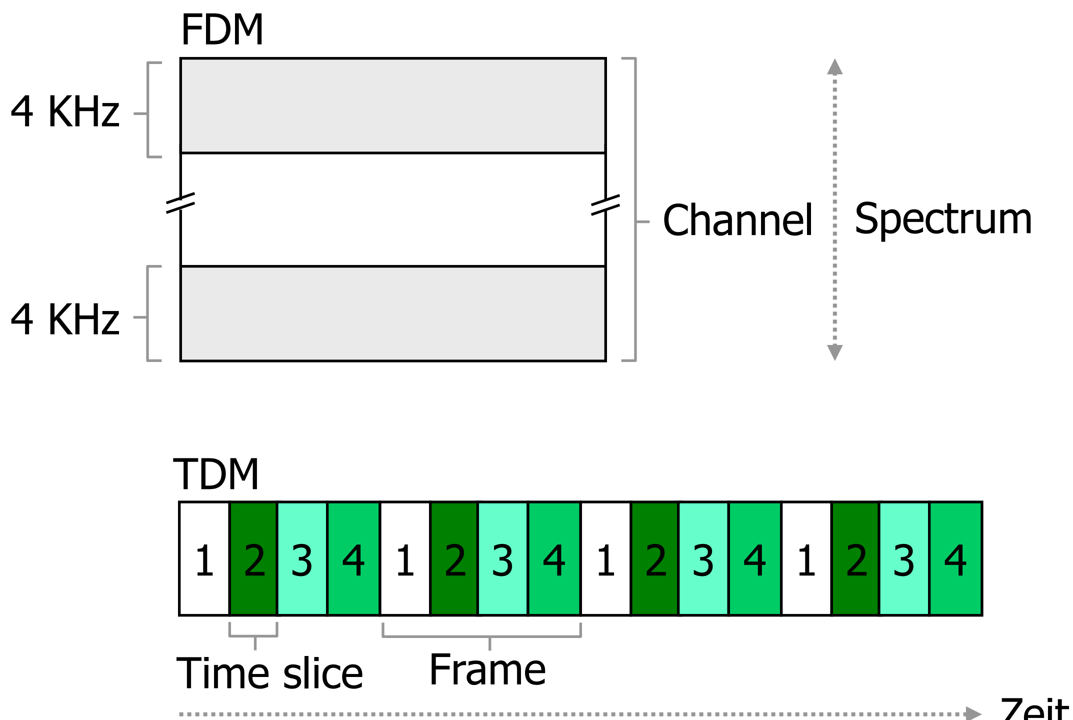
- Point-to-point connections
  - Only two end systems access the medium
  - Examples
    - Point-to-Point Protocol (PPP) between host and router via a telephone line
    - Connection between routers over a fiber optics cable
  - No or at least no complicated coordination required
- Multi-access media
  - Examples
    - Shared bus: old Ethernet, CAN, system bus like PCI
    - Shared radio channel: WLAN, Bluetooth, ZigBee
    - Internet access via cable TV
  - Requires **(distributed) coordination of medium access** (Medium Access Control, MAC)

# Medium Access

- Options for multi-access
  - **Multiplexing**
    - Results in fixed channels for every communication
    - We already studied frequency, time, and code multiplex
    - Disadvantage: inefficient for data communication
  - **Random access**
    - Stations access the medium randomly, thus, simultaneous transmissions (collisions) need to be treated
    - Examples: ALOHA, later CSMA variants
  - **Cyclic access**
    - Centralized: polling via centralized coordinator
    - Distributed: permission to send using a rotating token, e.g., Token Ring, USB, Profibus

# Multiplexing

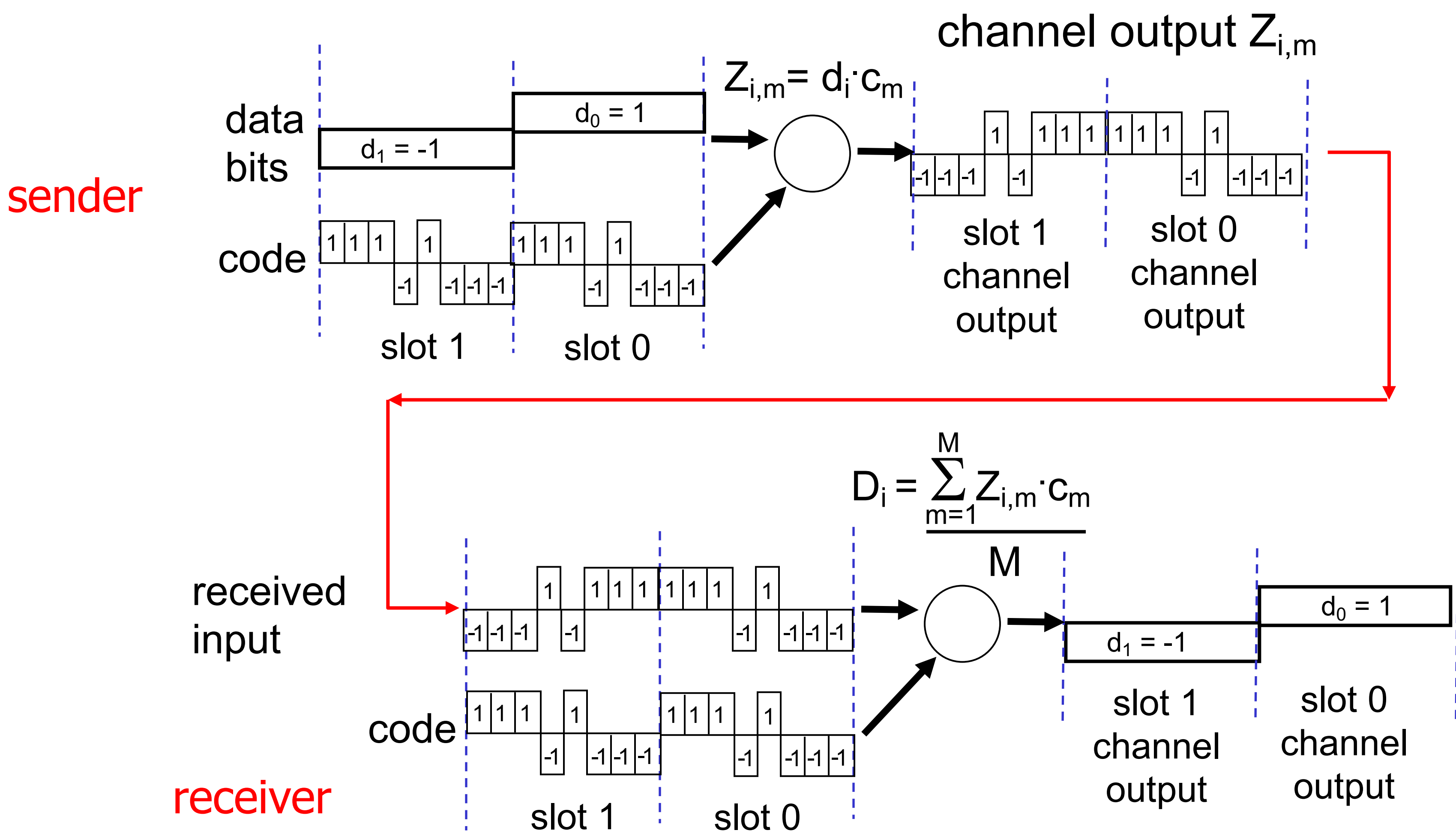
- Frequency Division Multiplex Access (FDMA): nodes use different parts of the frequency spectrum
- Time Division Multiplex Access (TDMA): nodes use full spectrum for dedicated time slices



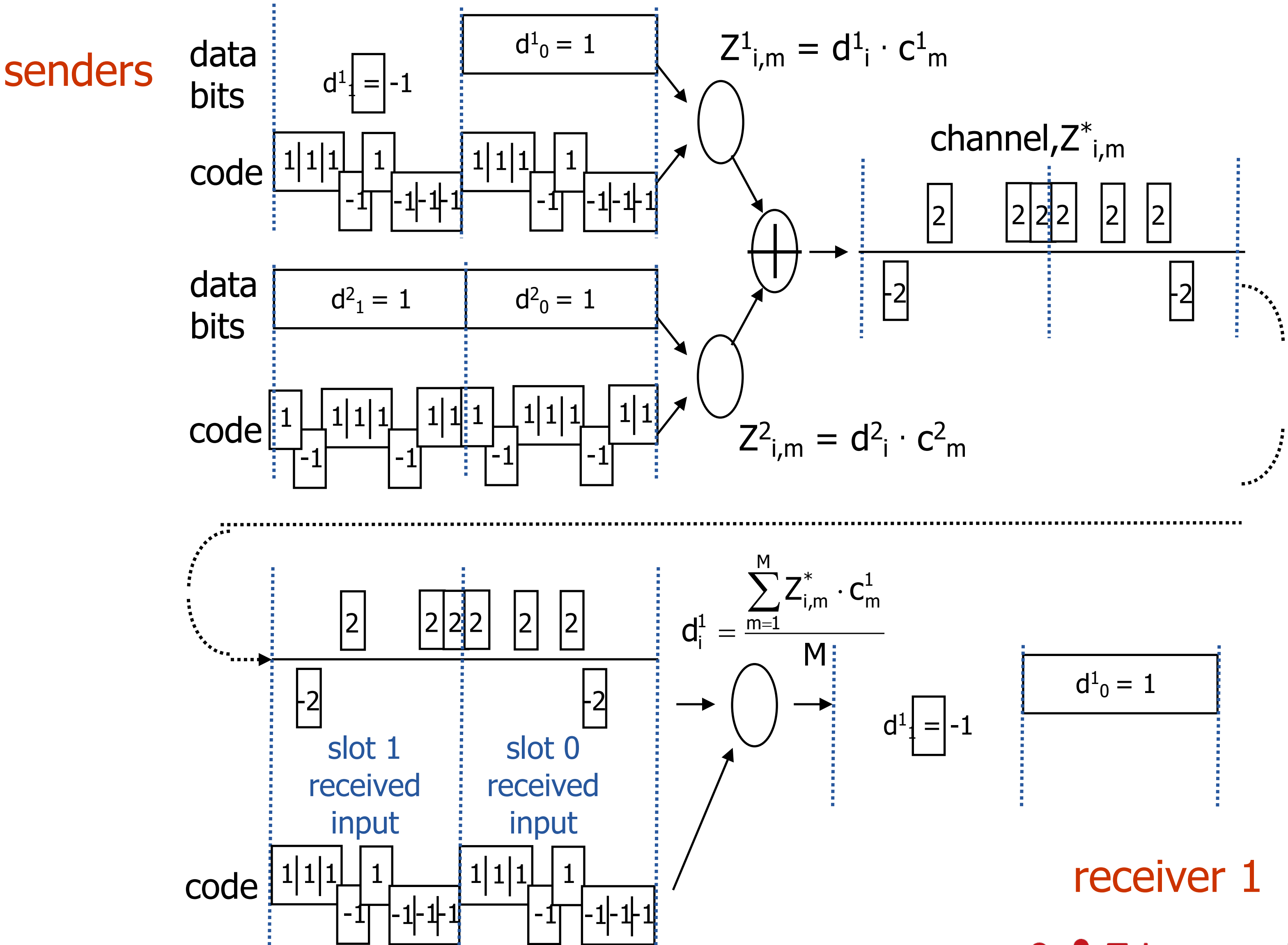
# Multiplexing

- Code Division Multiplex Access (CDMA)
  - **Spreading technique:** sender multiplies every bit with a **chipping code**
    - This results in a signal with of frequency; the signal is sent using the full bandwidth and time resources
    - All these signals overlay on the medium
    - The receiver can now reconstruct the original bit sequence wugin the same chipping code
- Alternative: **frequency hopping**
  - The sender now jumps through the channels and the receiver follows the same hopping sequence
  - Only nodes following the same hopping sequence can communicate
  - Originally used in Second World War, now basis for, e.g., Bluetooth

# CDMA Example: One sender, one receiver

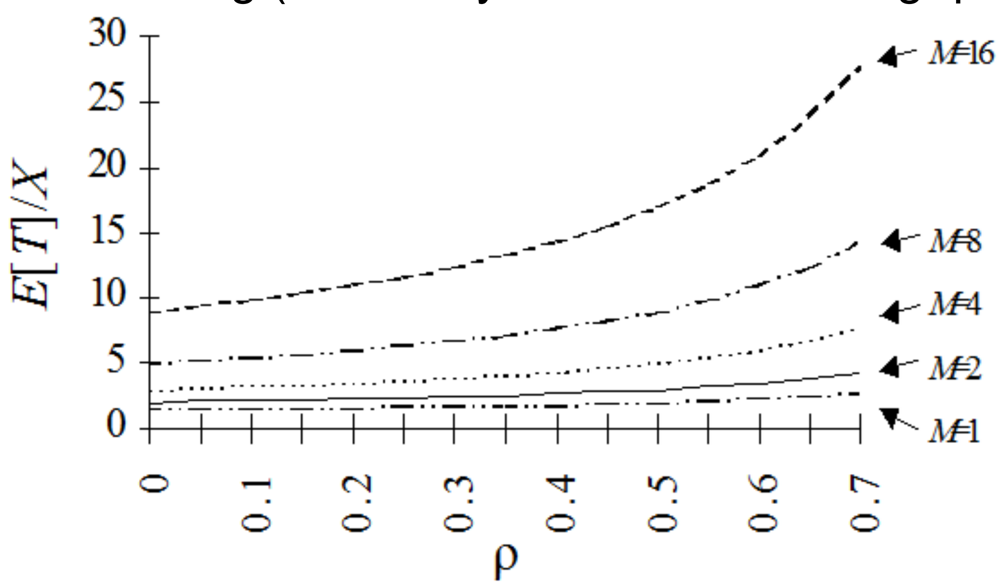


# CDMA Example: Two senders, one receiver



# Multiplexing

- Efficiency of fixed channel resource distribution
  - Data communication is **bursty**, thus, fixed channel resource distribution is inefficient
  - Example: increasing load ρ per node while equally sharing the channel resources into M = 1 ... 16 subchannels (we assume random, exponentially distributed packet send times and packet lengths); plotted is the mean wait time due to buffering (can easily be calculated using queuing theory) :

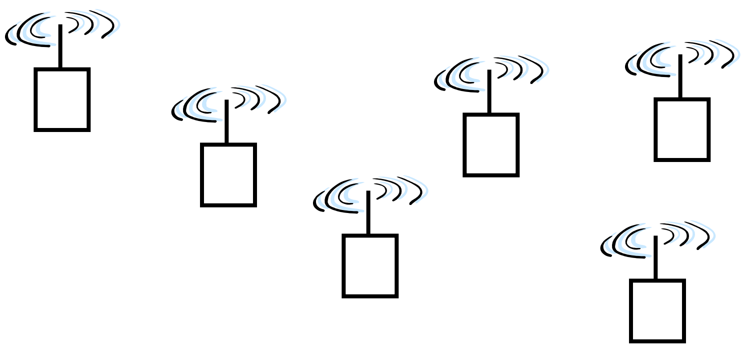


# Random Access

- When a node sends, it uses the full channel resources (i.e., sends at maximum bit rate)
- If two nodes send simultaneously, the signals interfere (often called a packet collision), solved by repeated transmissions
- Basic idea: for low load, collisions happen rarely

- Concepts for avoiding and detecting collisions
  - ALOHA, slotted ALOHA
  - Carrier Sense Multiple Access (CSMA)
  - CSMA with Collision Detection: CSMA/CD (e.g., in Ethernet)
  - CSMA with Collision Avoidance: CSMA/CA (e.g., in WLAN)

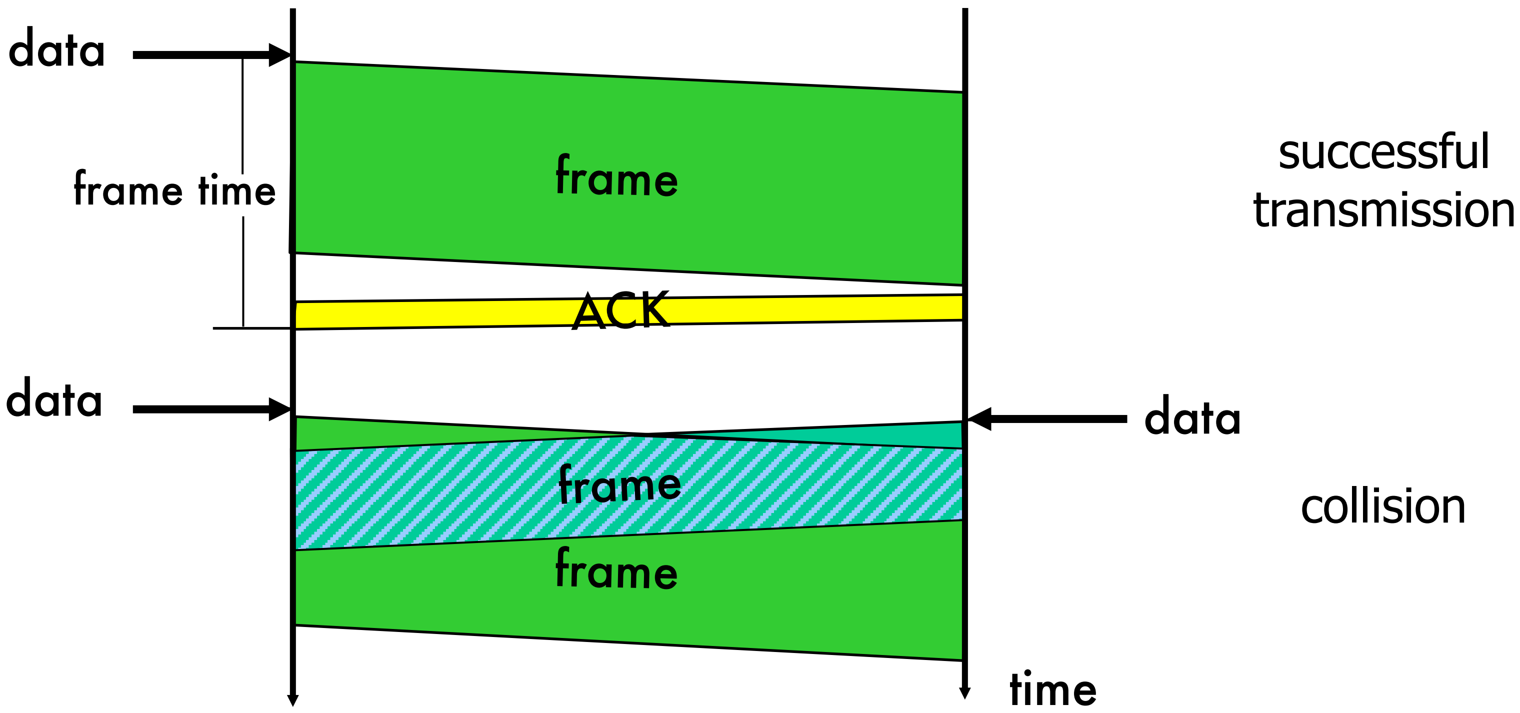
# ALOHA



- ALOHA
  - Developed in the 1970ies to interconnect computers of Hawaii University
  - Shared radio channel for all nodes
  - Procedure:
    - If a frame is ready to send (i.e., it is received by the MAC layer), the frame is sent **immediately**
    - If the receiver receives a frame without bit errors, it sends a positive **acknowledgement (ACK)** to the sender
    - If the sender does not receive the ACK within a **timeout**, the sender waits a **random backoff time** and repeats sending the frame
  - Collisions are handled like bit errors in the error control function
  - Very simple protocol, does not require coordination between sender and receiver

# ALOHA

- Examples for frame transmission using ALOHA:



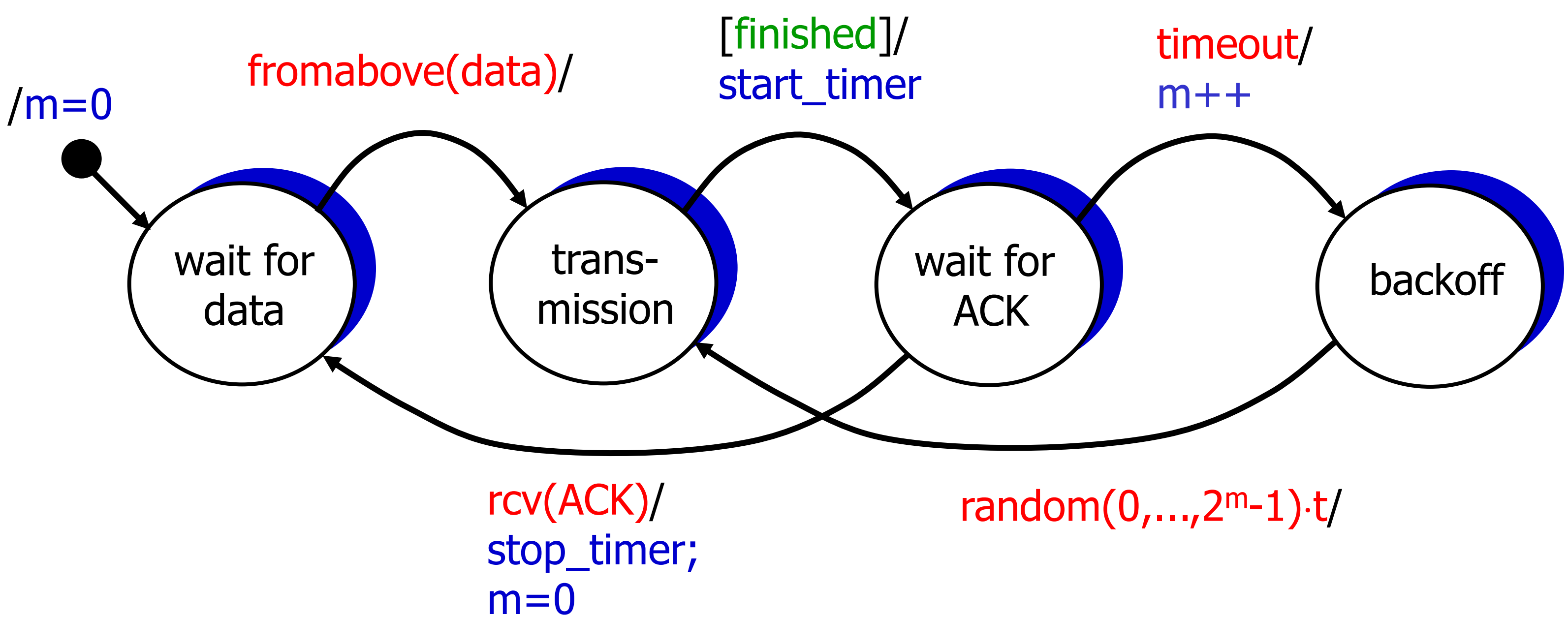

- Please note that the bandwidth-delay-product is very small, thus, it looks like send-and-wait

# ALOHA: Binary exponential backoff

- Algorithm
  - First collision: random, uniformly distributed choice of K ∈ {0,1}
  - Second collision: random, uniformly distributed choice of K ∈ {0,1,2,3}
  - …
  - m-th collision: random, uniformly distributed choice K ∈ $\{0,1,2,3,4,\ldots, 2^m-1\}$
  - Backoff time is K x $\tau$, with $\tau$ being a time slot adequately chosen for the used data rate and propagation time
  - After a maximum number M of retransmissions (e.g., M=10), the MAC layer stops and reports an error to the network layer
- Basic Idea: Adapt backoff time to current load in network
  - Low load: probably only few nodes are involved in collision, small K is sufficient to avoid collisions next time
  - High load: number of nodes involved in collision increases, thus, wider range of random backoff choices is needed

# ALOHA

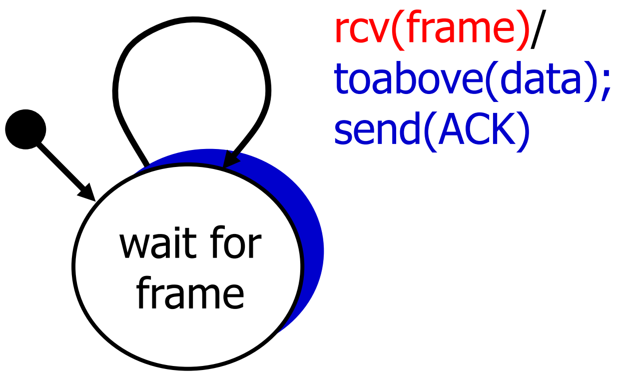
- Protocol statechart: Sender
  - Note: bit errors are treated as lost messages




m = #collisions
t = constant time

# ALOHA

- Protocol statechart: Receiver
  - Very simple, every received frame is delivered to the higher layer and an ACK is sent



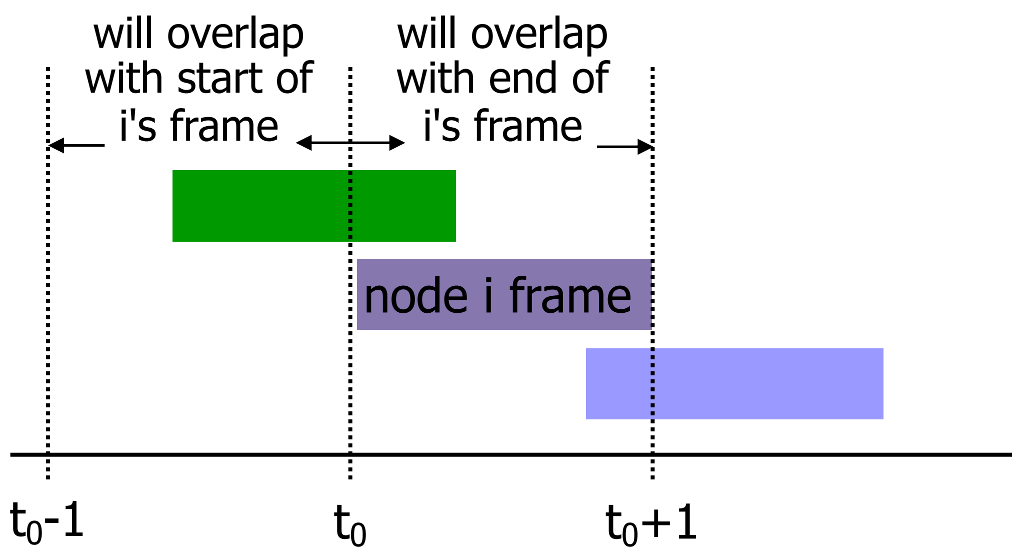
# ALOHA: Performance

- Performance seems to be very sensitive to collisions and resulting backoff times

- In the following, we use some simplifying assumptions
  - All frames have a constant length, the time to send a frame defines as “slot time”
  - Every node has the same probability p to send a frame (whether for the first time or as a retransmission); this is the most critical assumption
  - Specific impacts of radio communication are ignored (in reality, frame error probability is a function of signal quality)
  - We assume no bit errors at all

# ALOHA: Performance

- Preconditions for a collision
  - If a node starts sending at time $t_0$ and a second node starts sending in the interval $[t_0-1, t_0+1]$, we observe a collision



# ALOHA: Performance

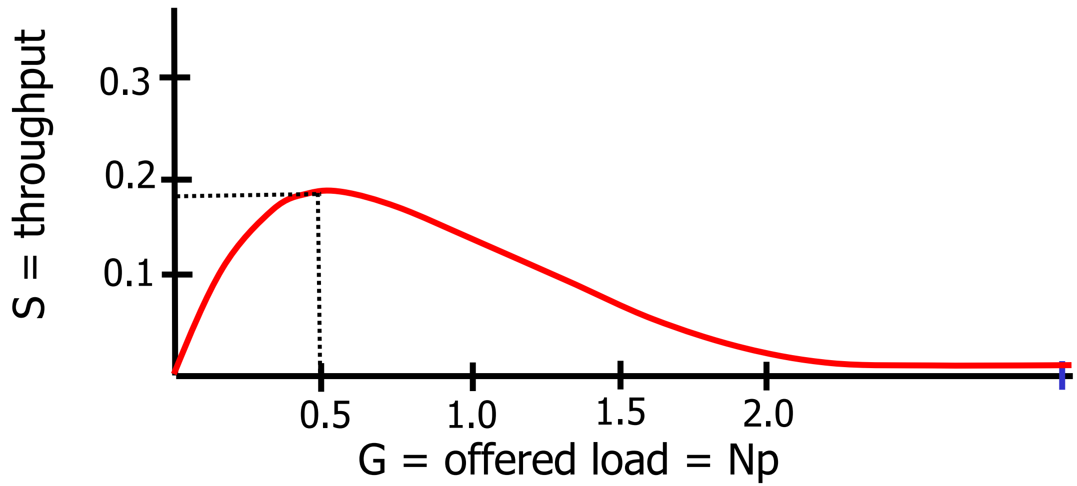
- Resulting throughput
  - N nodes
  - Probability that a specific node sends in any slot without collisions
    $= P(\text{node sends}) \cdot$
    $P(\text{no other node sends in } [t_0-1, t_0]) \cdot$
    $P(\text{no other node sends in } [t_0, t_0+1])$
    $= p \cdot (1-p)^{N-1} \cdot (1-p)^{N-1}$
    $= p \cdot (1-p)^{2(N-1)}$
  - Probability that any node sends in any slot without collisions
    $= N \cdot p(1-p)^{2(N-1)}$
    = average number of successful slots
    = **normalized throughput S**

# ALOHA: Performance

- Let G = Np (**offered load**, i.e., the rate of send attempts in a slot), then p = G/N
- Using this in the throughput equation $S = Np(1-p)^{2(N-1)} = G\left(1-\frac{G}{N}\right)^{2(N-1)}$
- For large N, we know $\lim_{n\to\infty}(1+x/n)^n = e^x$
- Thus $\lim\limits_{N\to\infty} S = \lim\limits_{N\to\infty} G\left(1-\frac{G}{N}\right)^{2(N-1)} = \lim\limits_{N\to\infty} G\left(1-\frac{G}{N}\right)^{2N}\left(1-\frac{G}{N}\right)^{-2} = Ge^{-2G}$
- From $\frac{d}{dG}Ge^{-2G} = (1-2G)\,e^{-2G} \overset{!}{=} 0$
  we get the maximum throughput $S_{max}$ = 1/2e ≈ 0.18 at G = 0.5
- Interpretation: even when trying to optimize the offered load, we can only achieve a resulting throughout of **18%** of the theoretic limit

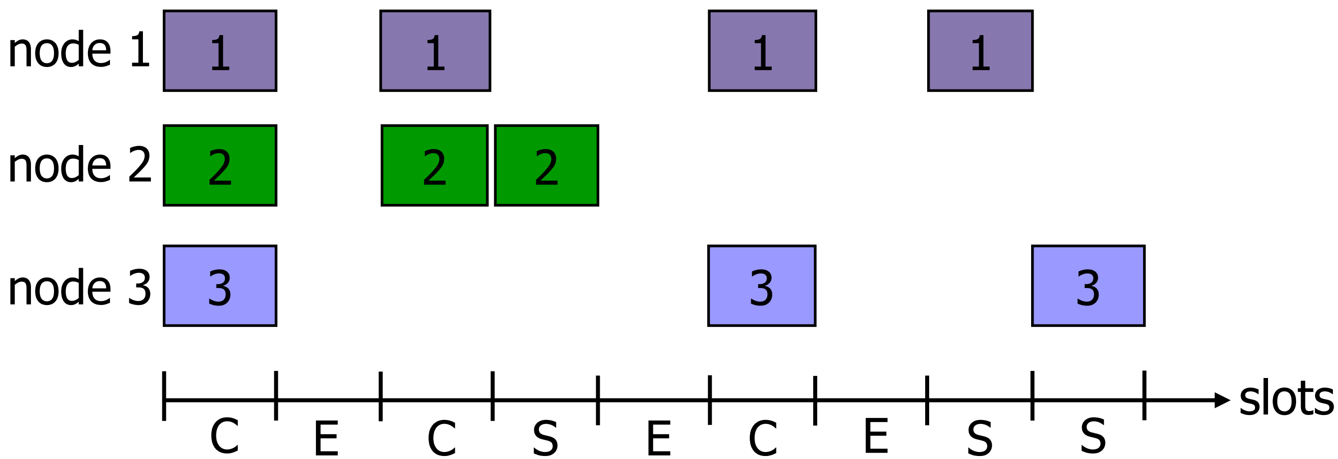
# ALOHA: Performance

- Throughout vs. offered load for ALOHA



# Slotted ALOHA

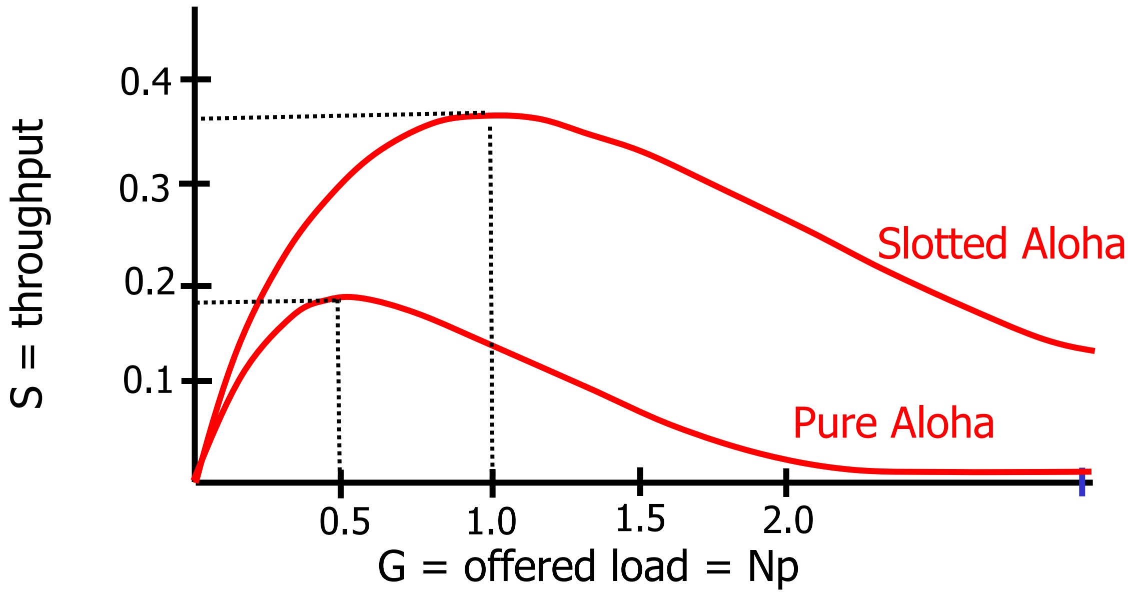
- Improvement: all nodes synchronize their slots (e.g., using a centralized time signal)
- Nodes may only send at the beginning of a slot → critical collision interval is reduced to one slot time




- Probability that any node sends in any slot without collisions: $Np(1-p)^{(N-1)}$
- Following the same considerations as with ALOHA, we get $S = G \cdot e^{-G}$ and $\mathbf{S_{max} = 1/e \approx 0.37}$ at G = 1 (this doubles the throughout)

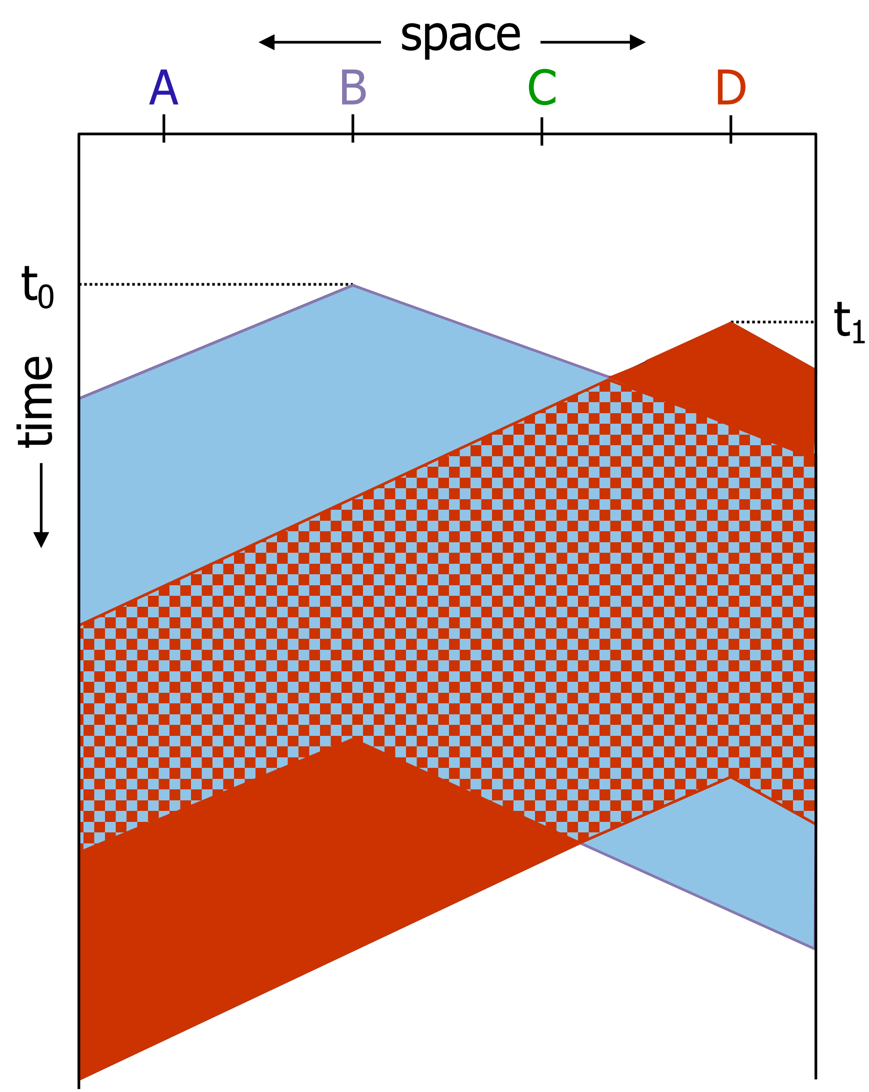
# Slotted ALOHA

- Comparison of ALOHA and Slotted ALOHA
  - Throughout vs. offered load



# Carrier Sense Multiple Access (CSMA)

- Nodes first sense whether medium is busy (**listen before talking**)
- Significantly reduces collisions
- Precondition: propagation delay < send time of frames (otherwise, useless)
- Collisions are still possible, e.g., when a node starts sending before the signal of another message was able to propagate to its position

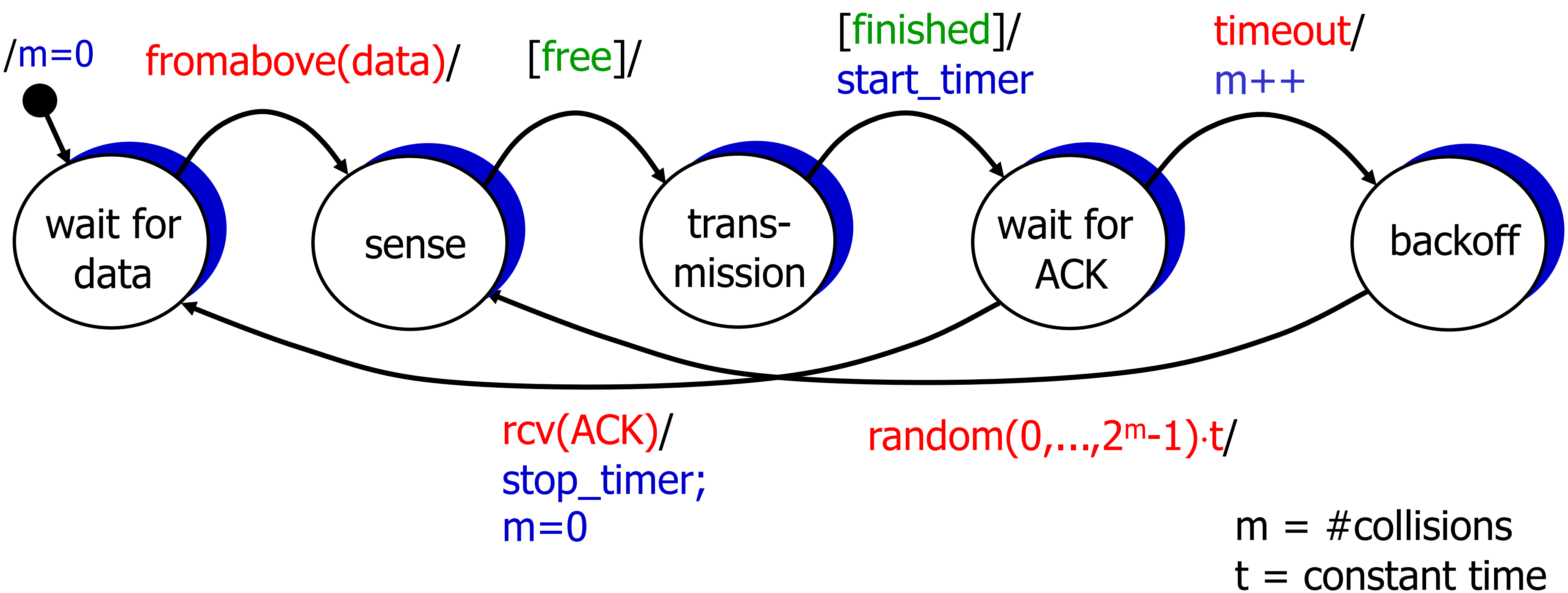


# Carrier Sense Multiple Access (CSMA)

- Procedure:
  - If a frame is ready to send (i.e., it is received by the MAC layer), the sender first checks if the medium available **(listen before talking)**; only if available, the frame is sent
  - If the receiver receives a frame without bit errors, it sends a positive **acknowledgement (ACK)** to the sender
  - If the sender does not receive the ACK within a **timeout**, the sender waits a **random backoff time** and repeats sending the frame

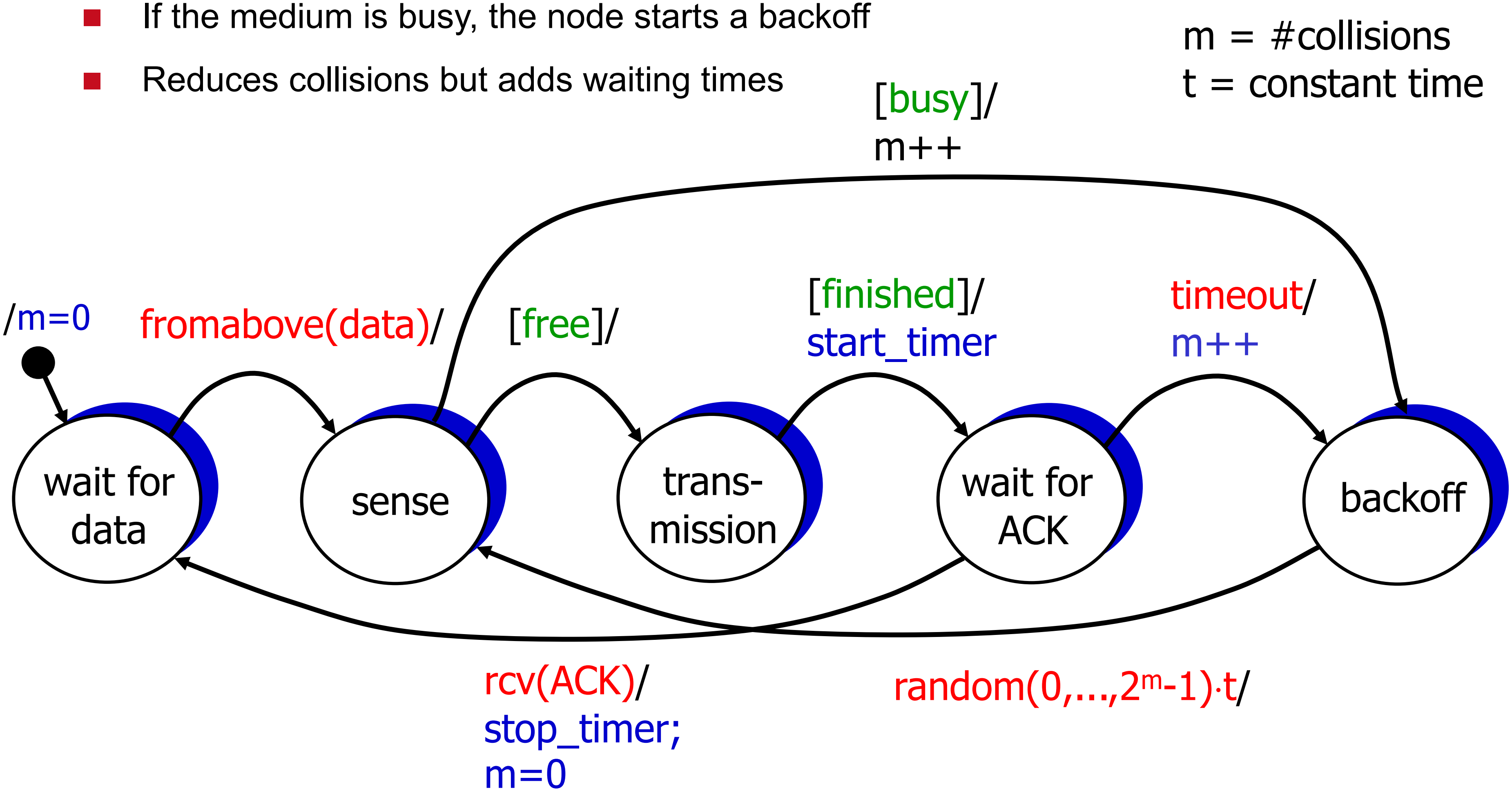
# CSMA Variants

- **1-persistent**
  - If the medium is busy, the node waits until it becomes available, then it immediately sends
  - Very short waiting time but possibility of synchronized collisions if multiple nodes wait for the medium



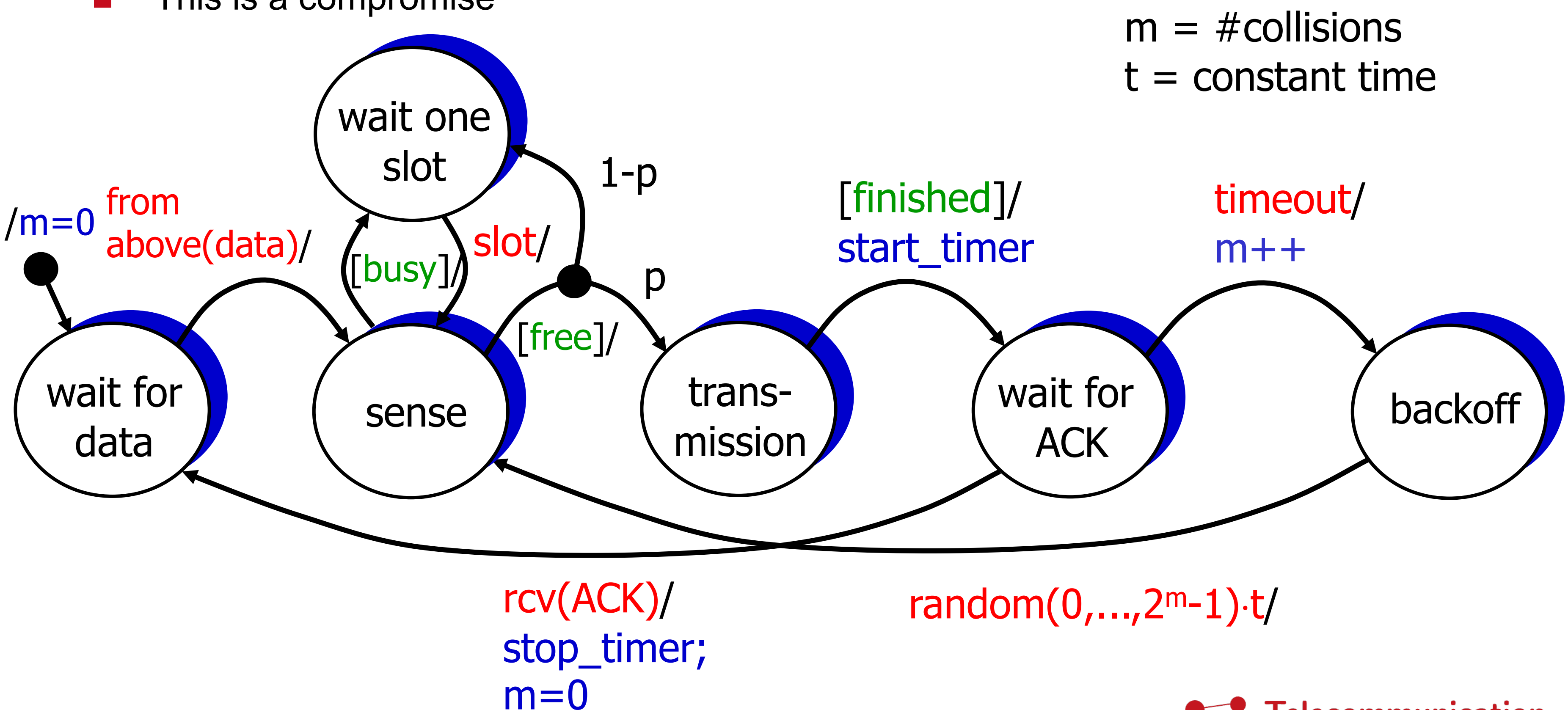
# CSMA Variants

- **non-persistent**
  - If the medium is busy, the node starts a backoff
  - Reduces collisions but adds waiting times

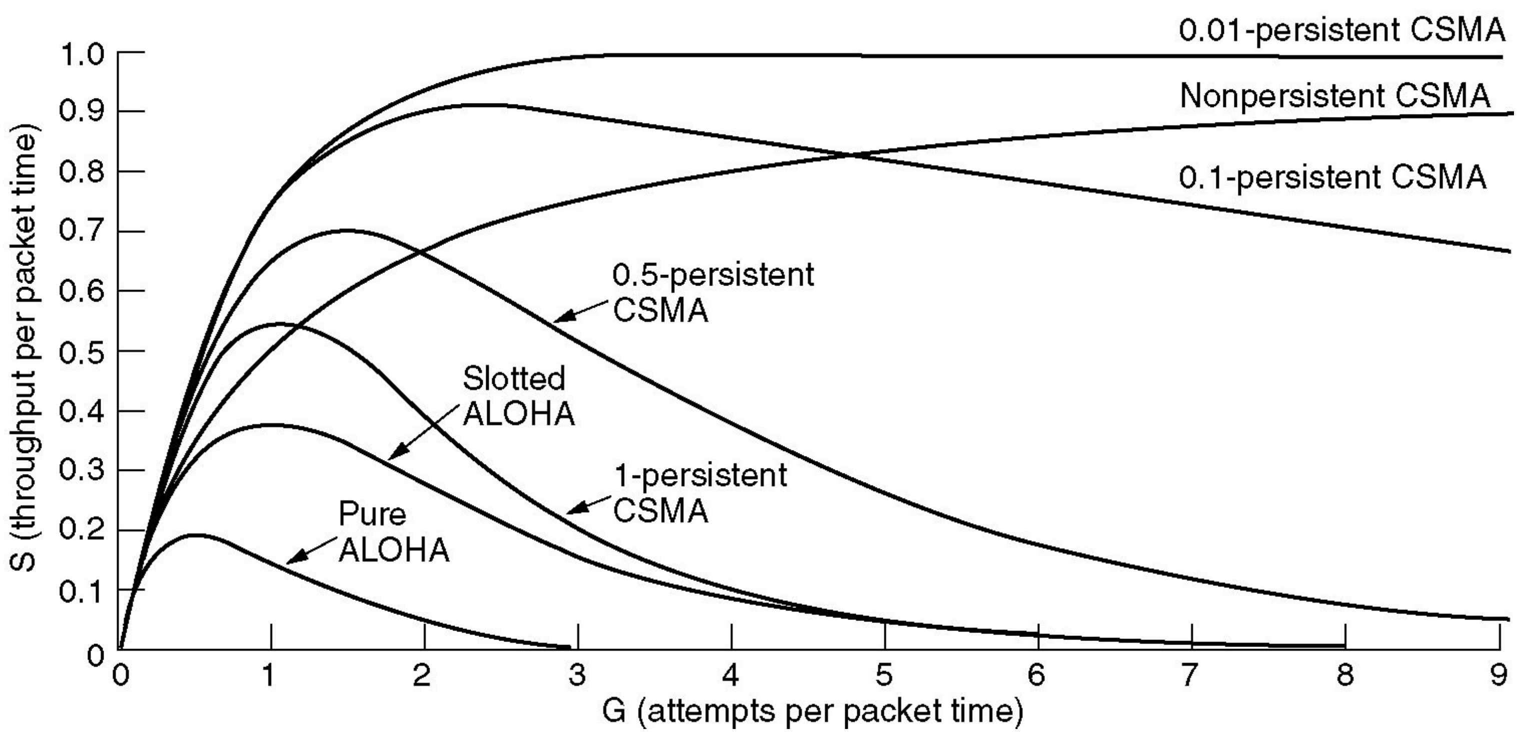


# CSMA Variants

- **p-persistent**
  - If the medium is busy, the node waits until it becomes available, then it immediately sends with probability p or waits another slot with probability 1-p
  - This is a compromise

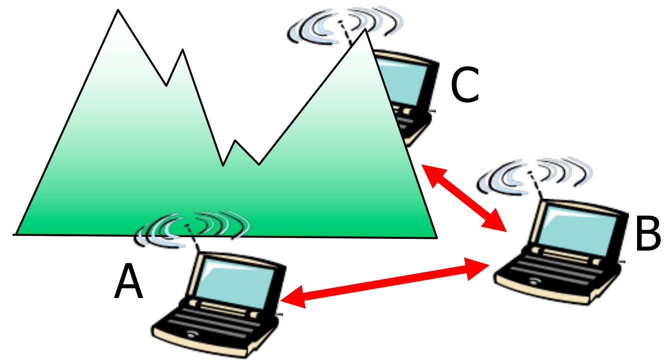


# Performance of CSMA Variants

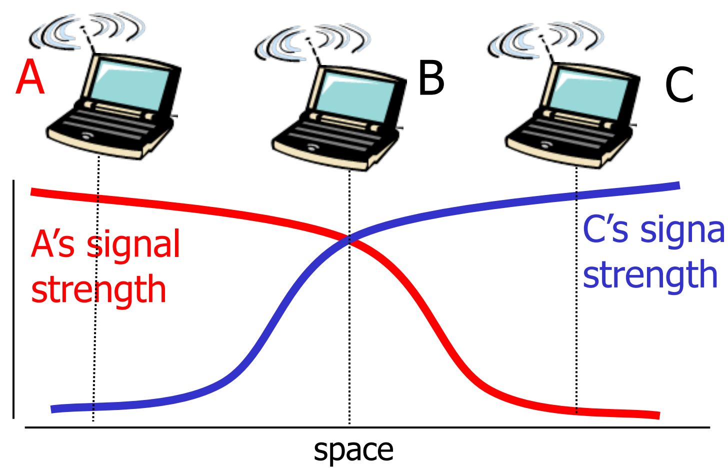


# CSMA Issues

- Hidden terminal problem
  - A, B are in communication range
  - C, B are in communication range
  - A, C cannot hear each other,
    A and C do not know about possible collisions at B

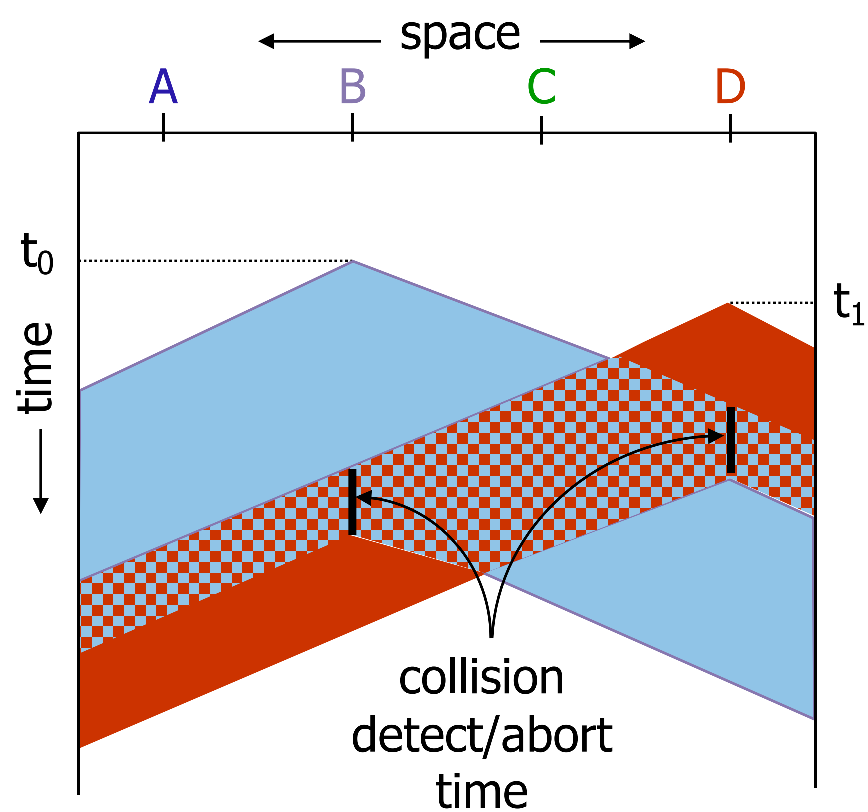



  - Also possible due to signal atttenuation



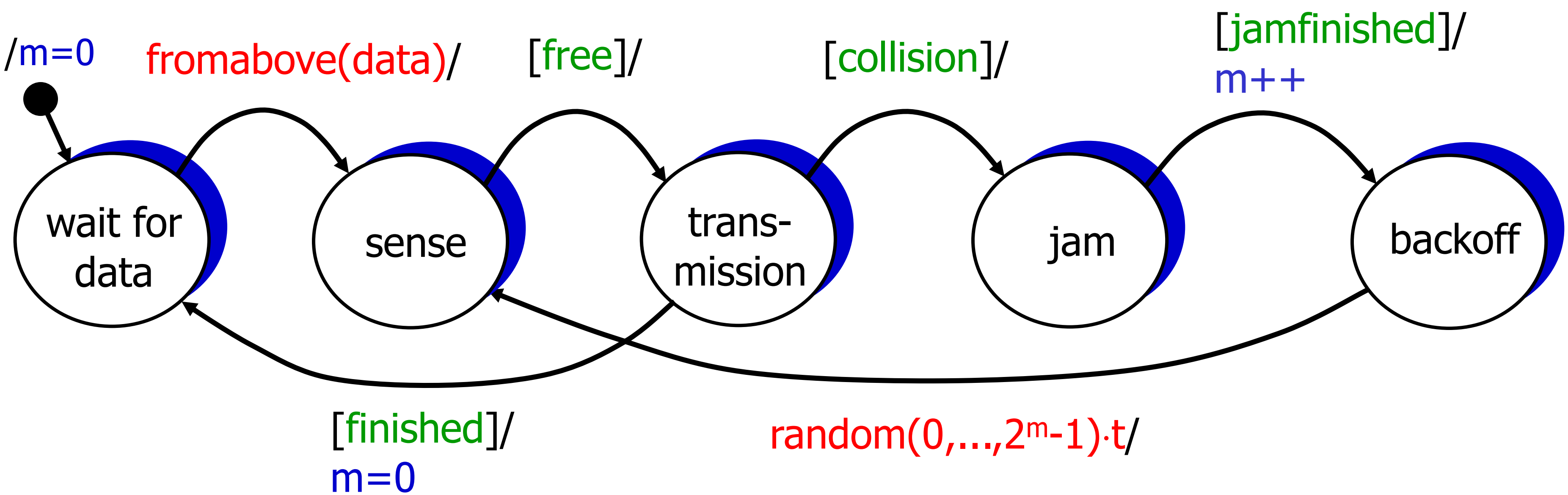
# CSMA/CD

- CD = Collision detection
  - Node use additional HW to detect collisions during sending (**listen while talking**)
  - After **collision detection**, sending is terminated (reduced abortion time) and a **jamming signal** is sent → ensuring collision detection at other stations
  - No ACKs required
  - Can be combined with all CSM variants



# CSMA/CD

- 1-persist CSMA/CD sender



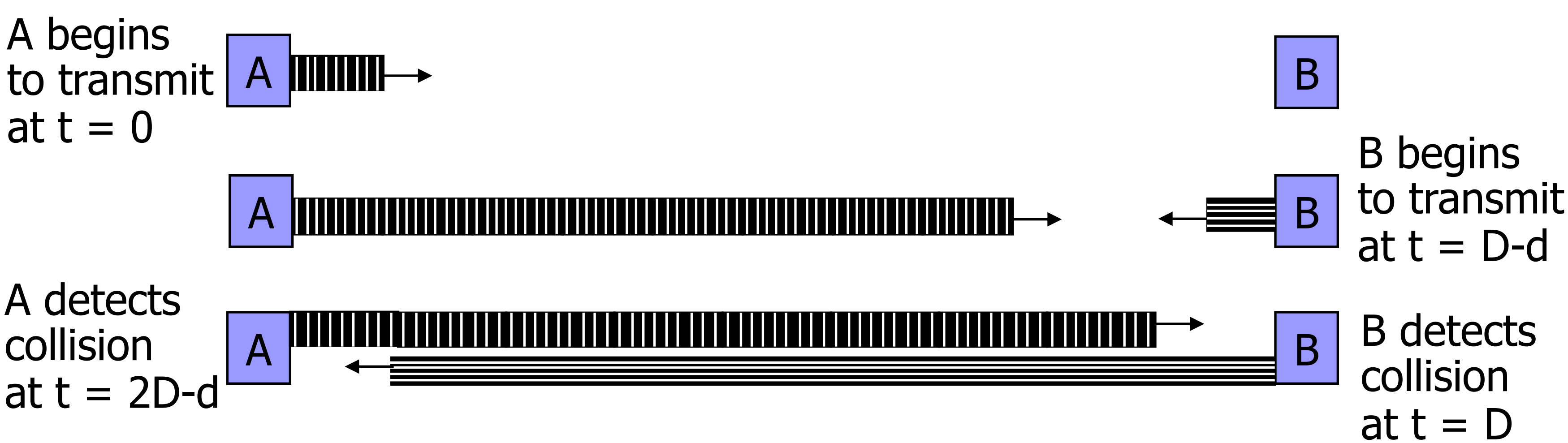

m = #collisions
t = constant time



Telecommunication Networks Group

# CSMA/CD

- Minimal frame size for CSMA/CD
  - Let D be the maximum propagation delay between any two nodes
  - It will take at most 2D until a collision is detected by all nodes
  - Assuming a bit rate R, then the minimum frame length L **must** be large enough so that L/R > 2D



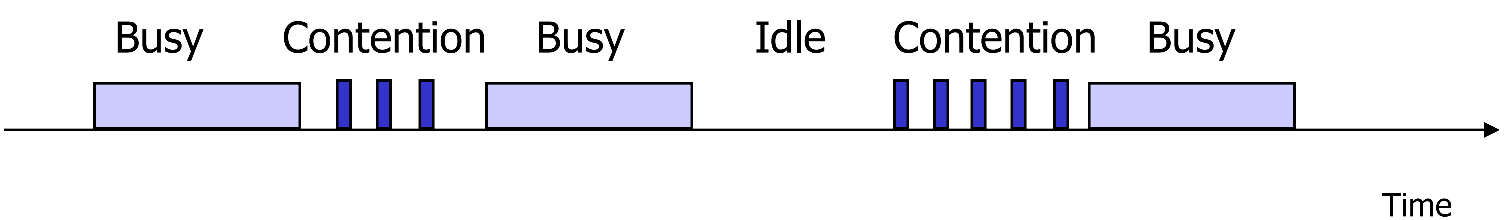
# CSMS/CD: Performance

Technische Universität Berlin

- Nodes are switching between sending, idle, and contention
- Sending a frame takes L/R time units
- Collisions are resolved of after 2D
- Contention subdivided into slots of length 2D
- N nodes, each tries to send in a particular with probability p
- Contention terminates, if exactly one node sends:

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

- As for Slotted ALOHA, we can derive that for p = 1/N the success probability is maximal: $P_{Success}^{max} = 1/e$






Telecommunication Networks Group

# CSMS/CD: Performance

Technische Universität Berlin

- Thus, on average, the contention phase takes e ≈ 2.718 slots
- For maximum throughout, we assume zero idle phases, thus, the system alternates between sending and contention
- After sending a frame, it takes one propagation time D until the frame is received by all nodes
- Thus:

$$S_{max} = \frac{\text{Sendephase}}{\text{Sendephase} + \text{Ausbreitung} + \text{Wettbewerbsphase}}$$
$$= \frac{L/R}{L/R + D + e2D} = \frac{1}{1+(1+2e)DR/L} = \frac{1}{1+(1+2e)a} \approx \frac{1}{1+6.4a}$$

- a = RD/L is the buffer capacity of the channel, which is critical for the performance of the CSMA/CD network



Telecommunication Networks Group

# CSMS/CD: Performance

- Maximum throughout of random access schemes as a function of a