INF3190 - Data Communication

Data Link Layer

Carsten Griwodz

Email: griff@ifi.uio.no

most slides from: Ralf Steinmetz, TU Darmstadt

and a few from Olav Lysne, J. K. Kurose og K. W. Ross

Function, Services and Connection Management

L1 Service

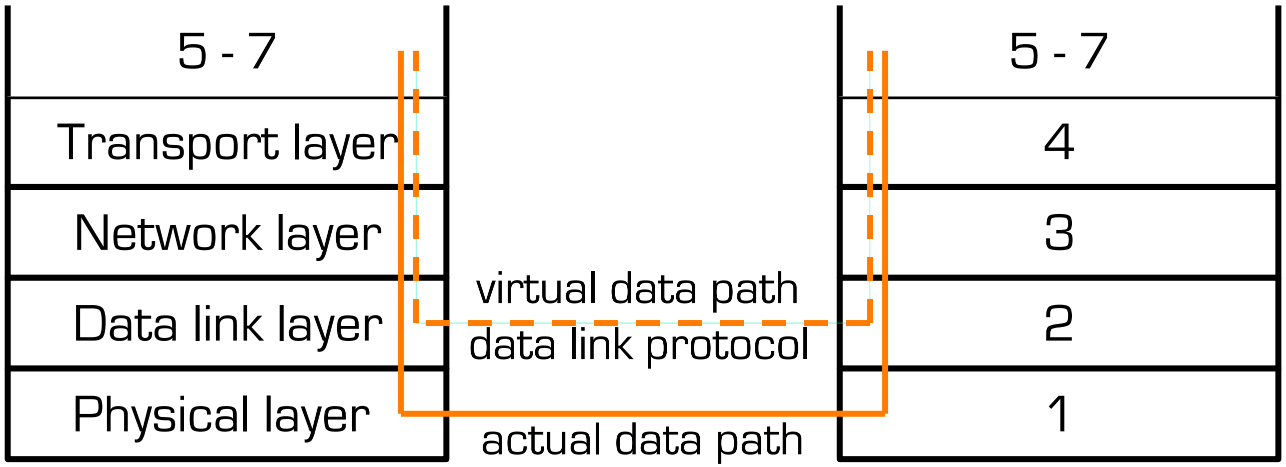
- transmission of a bit stream ("unreliable bit pipe")
  - without sequence errors
- problems of L1
  - finite propagation speed (limited data rate)
  - loss, insertion and changing of bits possible

L2 Service

- provide transfer of frames
- data transfer between adjacent stations
  - may be between more than 2 stations
  - adjacent: connected by one physical channel

L2 Functions

- data transmission as **frames**
- **error detection** and correction
- **flow control**
- configuration management

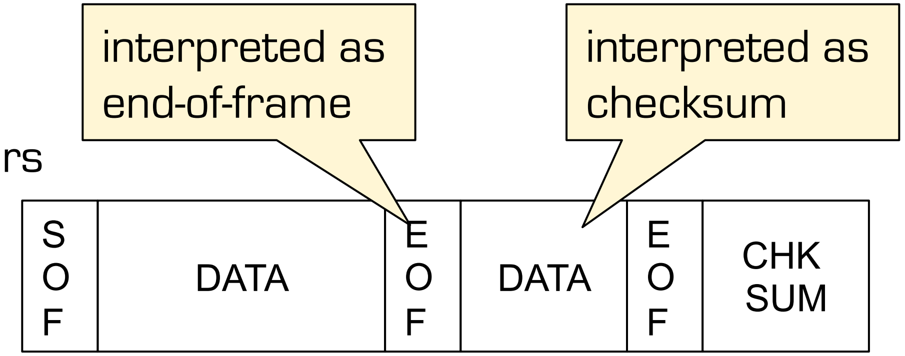


Framing

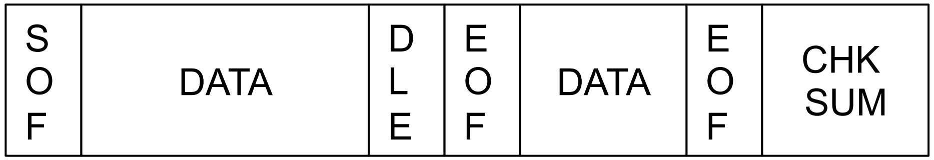
Framing: Character-oriented Protocols

Features
- Smallest unit is a character
- Alphabet size is predefined
  - Baudot: 5 bit, ASCII: 7 bit, Byte: 8 bit
- Control characters delimit frame start, frame end, and additional functions
- Frame has arbitrary length

Problem
- user data may contain control characters

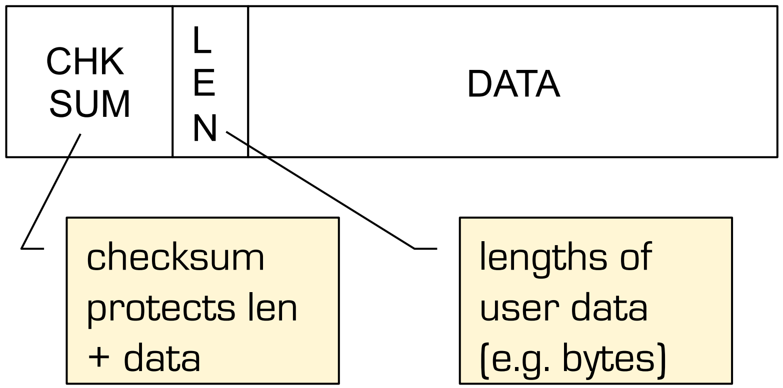


Solution
- **Character Stuffing**



- each control character in user data is preceded by a DLE (Data Link Escape)
- only control characters preceded by DLEs are interpreted as such

Framing: Count-oriented Protocols



Features

- frame contains a Length Count Field
- all symbols can be present in user data
- max. frame length determined by number of bits reserved for Length Count Field

Problem

- transmission error may destroy checksum and length count
- sender and receiver cannot recover understanding of frame start and frame end

Consequence

- no good solution for bit errors without Data Link Escape Symbol for SYN markers
- entire frame must be read before computing or verifying checksum

→ Rarely rarely used

Framing: Bit-oriented Protocols

Most used today

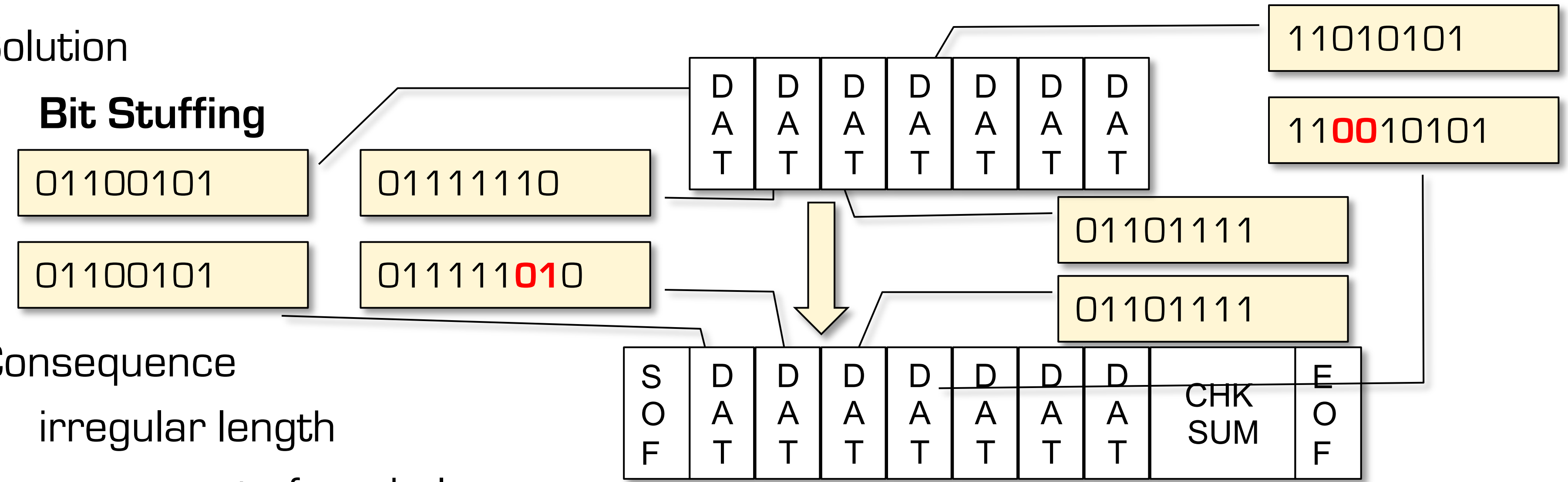
- independent from encoding block definition
- unique bit pattern for start-of-frame (or end-of-frame)
- frame can be corrupted, but re-synchronization is simple: wait for next start-of-frame

Problem

- start-of-frame marker can occur in user data or checksum

Solution

- **Bit Stuffing**



Consequence

- irregular length
- no concept of symbol

Error detection

Error Detection

Bit Error

- Modification of single bits

Burst Error

- Modification of a sequence of bits

Causes for error	Kind of disruption
thermic noise: electron movement generates background noise	infrequent bit errors
impulse disruptions (often last for 10 msec), e.g. due to glitches in electric lines, thunderstorms, switching arcs in relays, etc.	burst errors
crosstalk in adjacent wires	frequent bit errors
echo	infrequent bit errors
signal distortion (dampening is dependent on frequency)	burst errors

Burst Errors are more frequent than isolated Bit Errors

Code Word, Hamming Distance

Frame (= code word) contains

- data
- checking information

```
w1      10001001
w2      10110001
w3      10110011
```

Code = set of all valid code words

```
    w1 10001001
XOR w2 10110001
  =    00111000
 =>   Δ=3
```

Hamming distance of two words of the code

- number of bits that differ between two words

Hamming distance of a code

- minimal Hamming distance of all pairs of words

```
w1      10001001
w2      10110001
w3      10110010
Δ(w1,w2)=3
Δ(w1,w3)=5
Δ(w2,w3)=2
 =>   Δ=2
```

Error Detection (according to Hamming)

Detection of f 1-bit errors:

- if we make sure that the Hamming distance of a code is **d**

  $$d \geq f + 1$$

- **f** and fewer errors generate an invalid code word and are detected

			parity bit p
w1	0	0	0
w2	0	1	1
w3	1	0	1
w4	1	1	0

d = 2:

i.e. maximum value for f: f=1

detection of *one* 1-bit error

Cyclic Redundancy Check (CRC)

Basic idea:

bit strings are treated as polynomials

n-bit string: $k_{n-1} \cdot x^{n-1} + k_{n-2} \cdot x^{n-2} + \ldots + k_1 \cdot x + k_0$

where $k_i = [0,1]$

Example: **1 1 0 0 0 1** → $x^5 + x^4 + 1$

Polynomial arithmetic: modulo 2

Sender and receiver agree on a polynomial G(x)

Cyclic Redundancy Check (CRC)

Sender and receiver agree on a polynomial G(x)

Sender wants to send bitstring B(x)
Sender computes
  B(x)00…0 / G(x) ➔ result
  Q(x) and rest R(x)
Sender sends B(x) and R(x)

Receiver computes B(x)|R(x) / G(x)
result Q'(x) and rest R'(x)
if R'(x)=0, no bit error was found
else at least one bit error

Error Detection

Algorithm

with

B(x) ... Block polynomial

G(x) ... Generator polynomial of degree r

- r < degree of B(x)
- highest and lowest order bit = 1

1. Add r 0-bits at the lower order end of B

- Let result be B^E and corresponds to: $x^r * B(x)$

2. Divide $B^E(x)$ by G(x)

- modulo 2: subtraction and addition are identical to XOR
- result: Q(x) + R(x)

3. Subtract R(x) from B^E(modulo 2)

- And transmit the result

Error Detection

Example: frame: 1101011011

Generator G(x), degree 4: 10011

Frame with 4 attached 0-bits: 11010110110000

```
             1100001010
       10011 11010110110000
             10011
              10011
              10011
               00001
               00000
                00010
                00000
                 00101
                 00000
                  01011
                  00000
                   10110
                   10011
                    01010
                    00000
                     10100
                     10011
                      01110
                      00000
Remainder ---->        1110
```

Transfered frame: 11010110111110

Error Detection

Discovering polynomials with “nice” properties is an art form

Standardized polynomials

CRC-1: this is the parity bit

CRC-5-EPC $= x^5 + x^3 + 1$ (used for RFID)

CRC-16-IBM $= x^{16} + x^{15} + x^2 + 1$ (used for USB)

CRC-16-CCITT $= x^{16} + x^{12} + x^5 + 1$ (used in Bluetooth, SD memory)

for example, CRC-16-CCITT recognizes

- all single and duplicate errors
- all errors with odd bit numbers
- all burst errors up to a length of 16
- 99.99 % of all burst errors of a length of 17 and more
- if x+1 is a divider of the CRC, no odd bit error can escape

Flow control

Flow Control and Error Treatment

Problem

- sender can send faster than receiver can receive

Without flow control

- receiver loses frames despite error-free transmission

With flow control

- sender can adapt to receiver's abilities by feedback

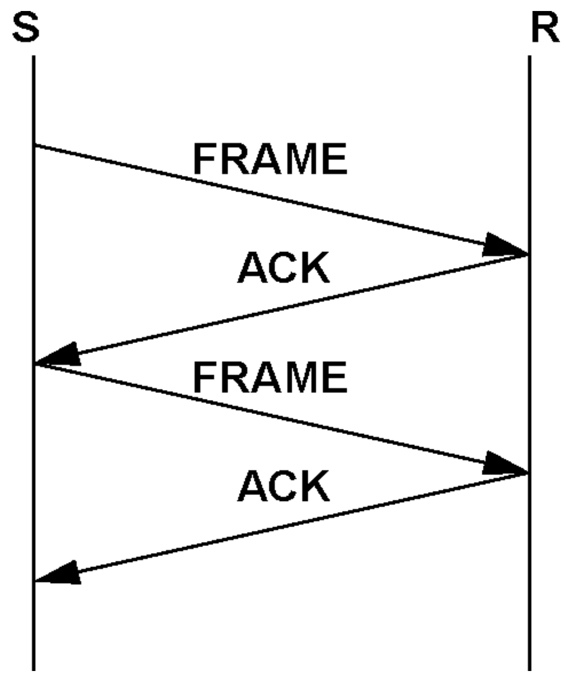
Comment

- error control and flow control are usually interlinked
- rate control
  - controls sending speed as well
  - but defines sequencing of send operations
  - whereas flow control defines conditions for next send operation

Protocol: Basic Stop-and-Wait

Assumptions

- error-free communication channel
- NOT [infinitely large receiving buffer]
- NOT [receiving process infinitely fast]



Further

- simplex mode for actual data transfer
- acknowledgement requires
  at least semi-duplex mode

Flow control necessary: **Stop-and-Wait**

- receiving buffer for a frame
- communication in both directions (frames, ACKs)

Basic Stop-and-Wait in insufficient

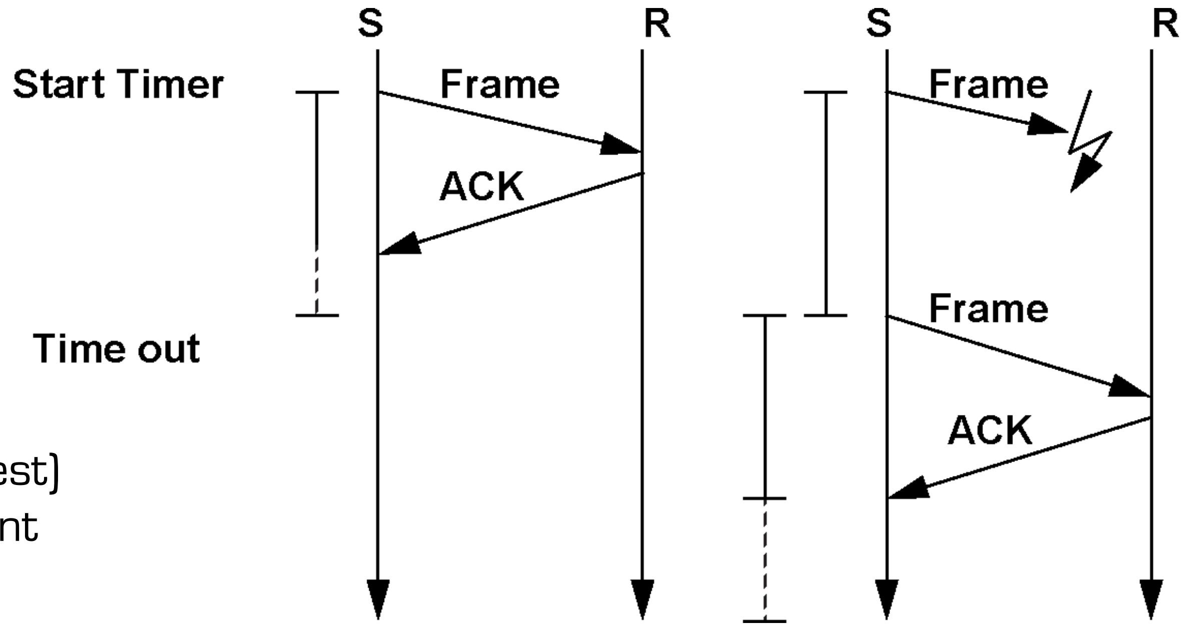
- fails with lost data frames and lost ACK frames

Protocol: Stop-and-Wait with ARQ

Assumptions
- NOT [error-free communication channel]
- NOT [infinitely large receiving buffer]
- NOT [receiving process infinitely fast]

Problem
- basic Stop-and-Wait blocks when a frame is lost

Solution: add a **timer**



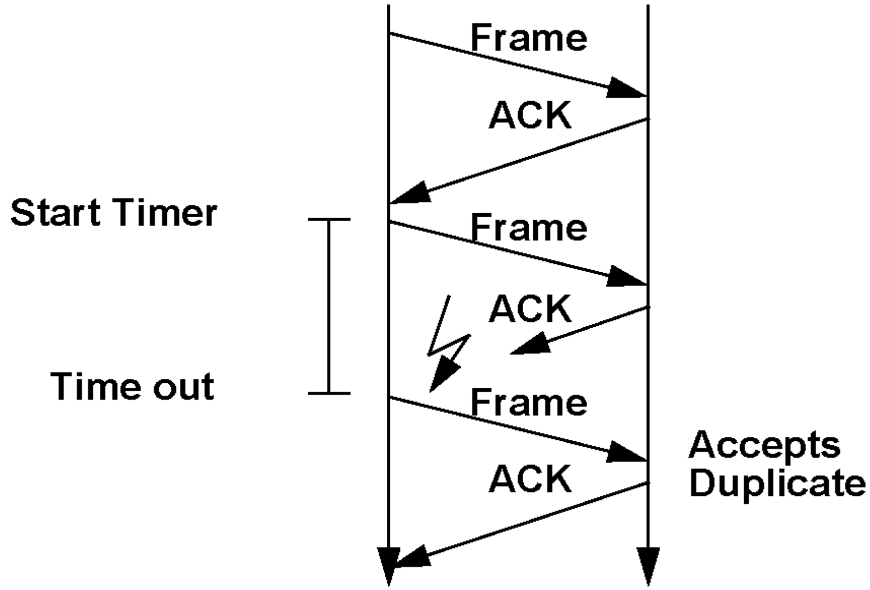
Two variants
- **ARQ** (Automatic Repeat reQuest)
- **PAR** (Positive-Acknowledgement with Retransmit)

Timeout interval:
- Too short: unnecessary sending of frames
- Too long: unnecessary long wait in case of error

Protocol: Stop-and-Wait with ARQ & SeqNo

Problem

- cannot distinguish loss of frames and loss of ACKs
- loss of ACKs may lead to duplicates



Solution: sequence numbers

- each block receives a sequence no.
- sequence no. is kept during retransmissions
- range
  - in general: [0, ..., k], k=2n-1

Stop-and-Wait: 0,1

Protocol: Stop-and-Wait with NAK, ACK & SeqNo

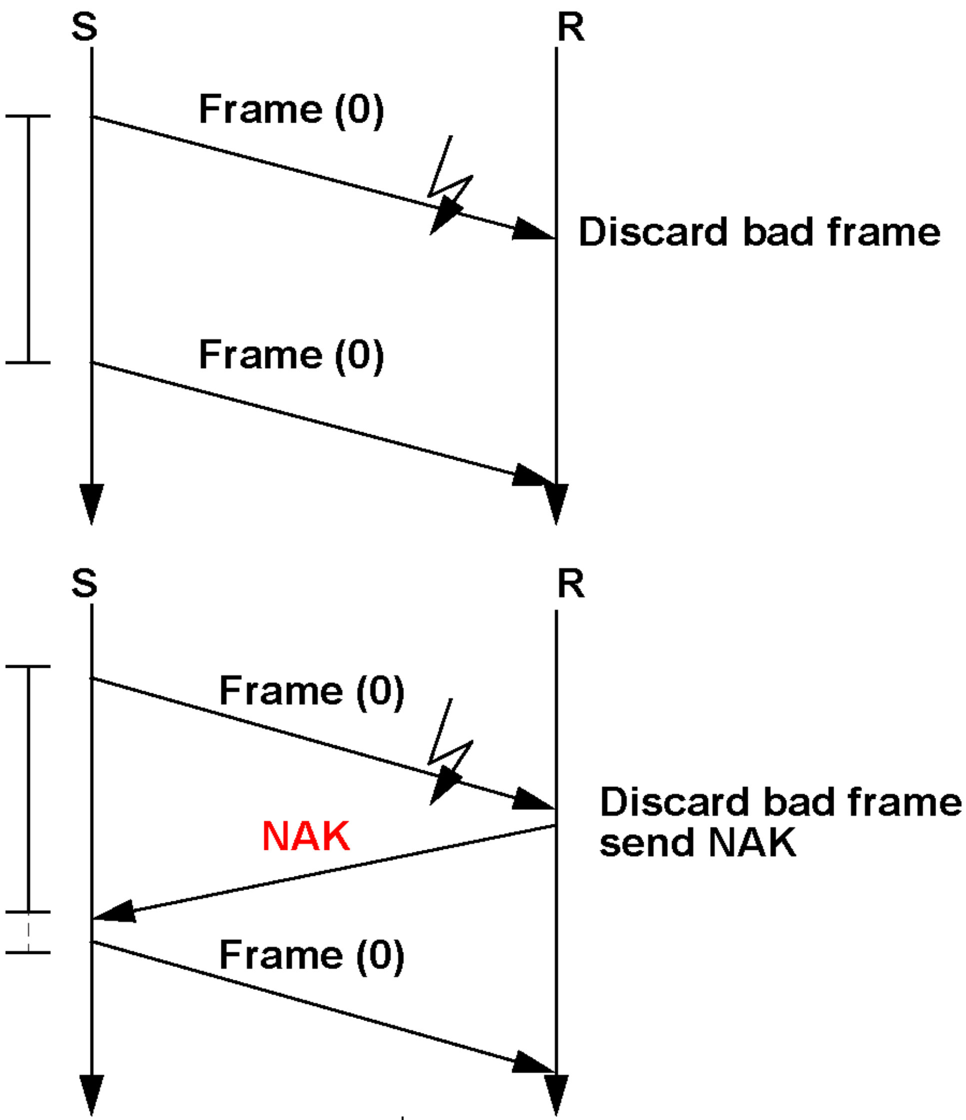
Until now passive error control

- no differentiation between
  - missing frames (cannot be recognized as frames)
  - faulty frames (recognized but checksum indicates bit errors)

- even if receiver knows the error, it has to wait for the timer
  - time consuming

Alternative: Active error control

- include negative ACK (NAK)
- in addition to ACK



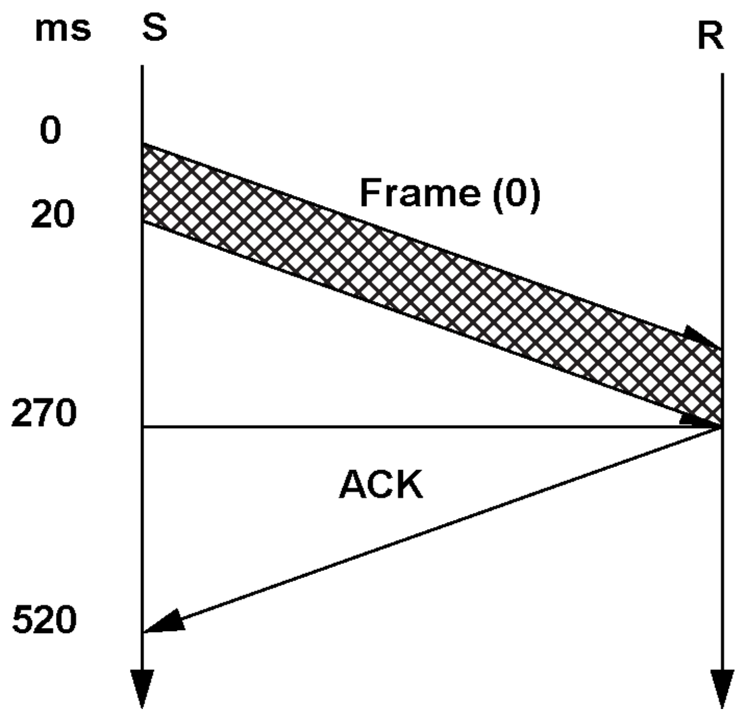
Channel Utilization and Propagation Delay

Stop-and-Wait

- sender can never send new frame before ACK, or NAK, or timeout
- channel is unused most of the time
- poor **utilization** of the channel

Satellite channel

- transmission rate: 50 kbps
- roundtrip delay 500 ms (2*250 ms)
- frame size: 1000 bit
- in comparison
  → ACK is short and negligible

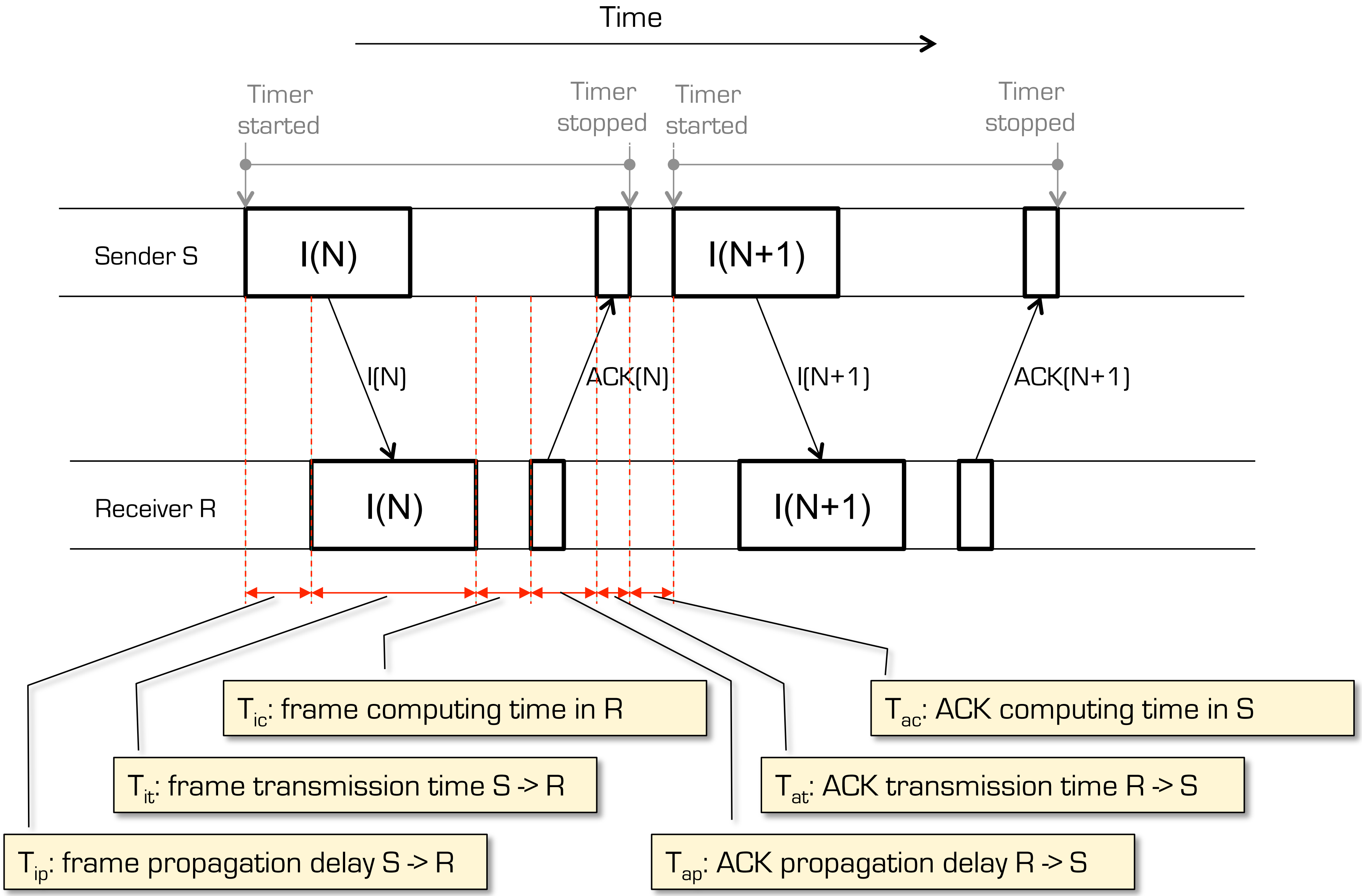


this means

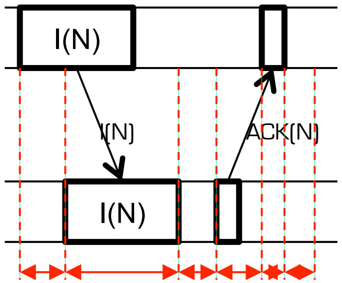
- sending takes 1000 bit / 50.000 bps = 20 ms
- sender is blocked for 500 ms of 520 ms

→ Channel utilization < 3.8%

Channel Utilization and Propagation Delay



Channel Utilization and Propagation Delay



T_{ip} : frame propagation delay
T_{it} : frame transmission time
T_{ic} : frame computing time
T_{ap} : ACK propagation delay
T_{at} : ACK transmission time
T_{ac} : ACK computing time

Best-case **utilization** of Stop-and-Wait

– best-case: only the error-free case is considered

$$U = \frac{T_{it}}{\sum T_{\text{information + acknowledgement}}} = \frac{T_{it}}{T_{ip} + T_{it} + T_{ic} + T_{ap} + T_{at} + T_{ac}}$$

usually we can approximate

$T_{ip} = T_{ap}$ – bits on the wire need same time both directions

$T_{ic} = T_{ac} \ll T_{ip}$ – the *protocol* computing time is negligable

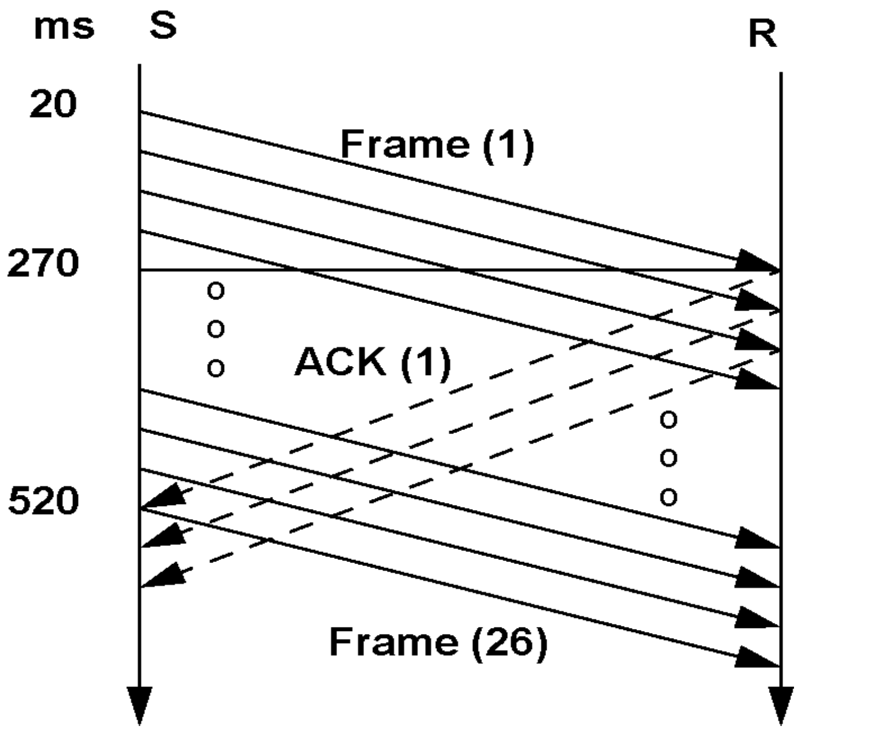
$T_{at} \ll T_{it}$ – data frame transm. time much larger than ACK frame transm. time

Approximate best-case utilization of Stop-and-Wait: $U = \frac{T_{it}}{T_{it} + 2T_{ip}} = \frac{1}{1 + 2\frac{T_{ip}}{T_{it}}}$

Improving Utilization: Sliding Window

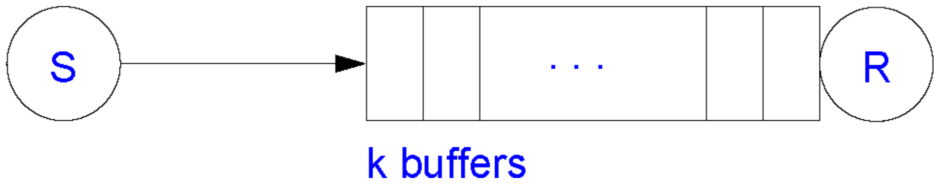
Improve utilization: pipelining

Flow control: **sliding window** mechanism



Sliding Window: Concept

1st goal of link layer flow control:
receiver buffer must not overflow



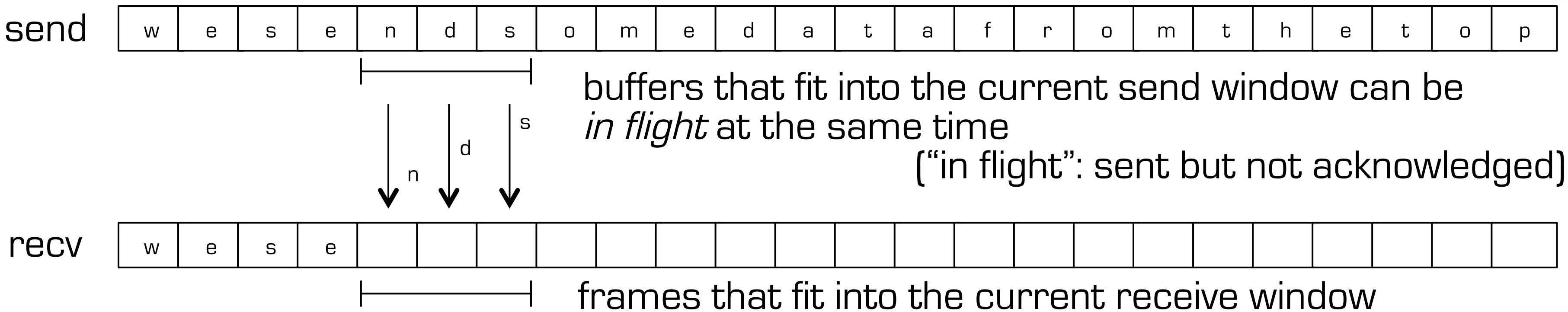
Assumptions:
each buffer can contain one frame

Two windows per communication relationship
Sender Window (or Send Window)
- frames that were sent but not yet acknowledged

Receiver Window (or Receive Window)
- frames that can be accepted

frames are identified by sequence numbers

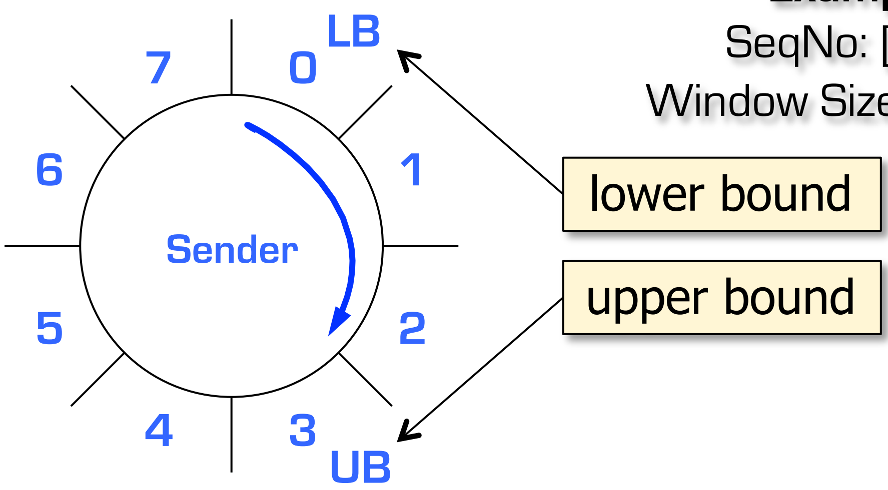
but the frames keep coming and these numbers will wrap eventually



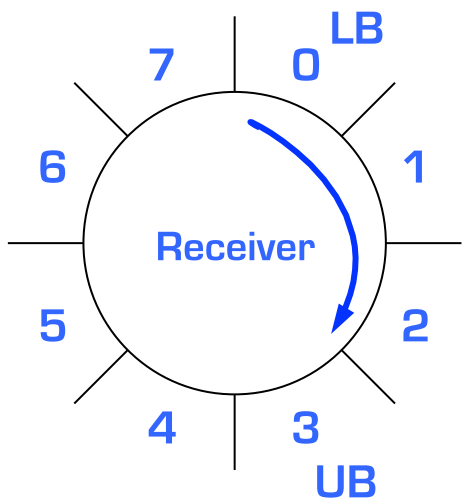
buffers that fit into the current send window can be *in flight* at the same time
("in flight": sent but not acknowledged)

frames that fit into the current receive window are considered valid and are acknowledged on arrival

Sliding Window: Concept



Example
SeqNo: [0..7]
Window Size (**WS**): 3



Sender:
- LB: oldest seqno that still unconfirmed
- UB: next seqno to be sent

Receiver:
- LB: lowest valid seqno that can be received
- UB: highest valid seqno that can be received+1

advancing sender seqno (modulo 8)
- LB: when ACK is received
- UB: when sending a frame

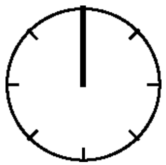
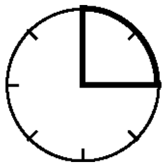
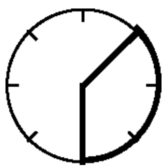
advancing receiver seqno (modulo 8)
- LB: when frame is received
- UB: when sending an ACK

UB=LB ← idle state → UB=(LB+WS) % 8

Sliding Window: Examples

Assuming

- 8 sequence numbers [0..7]
- max window size 3

Sender: Sliding Window	UB - LB	Situation
	0	sender may send up to 3 frames
	2	sender may send 1 frame
	3	sender is blocked

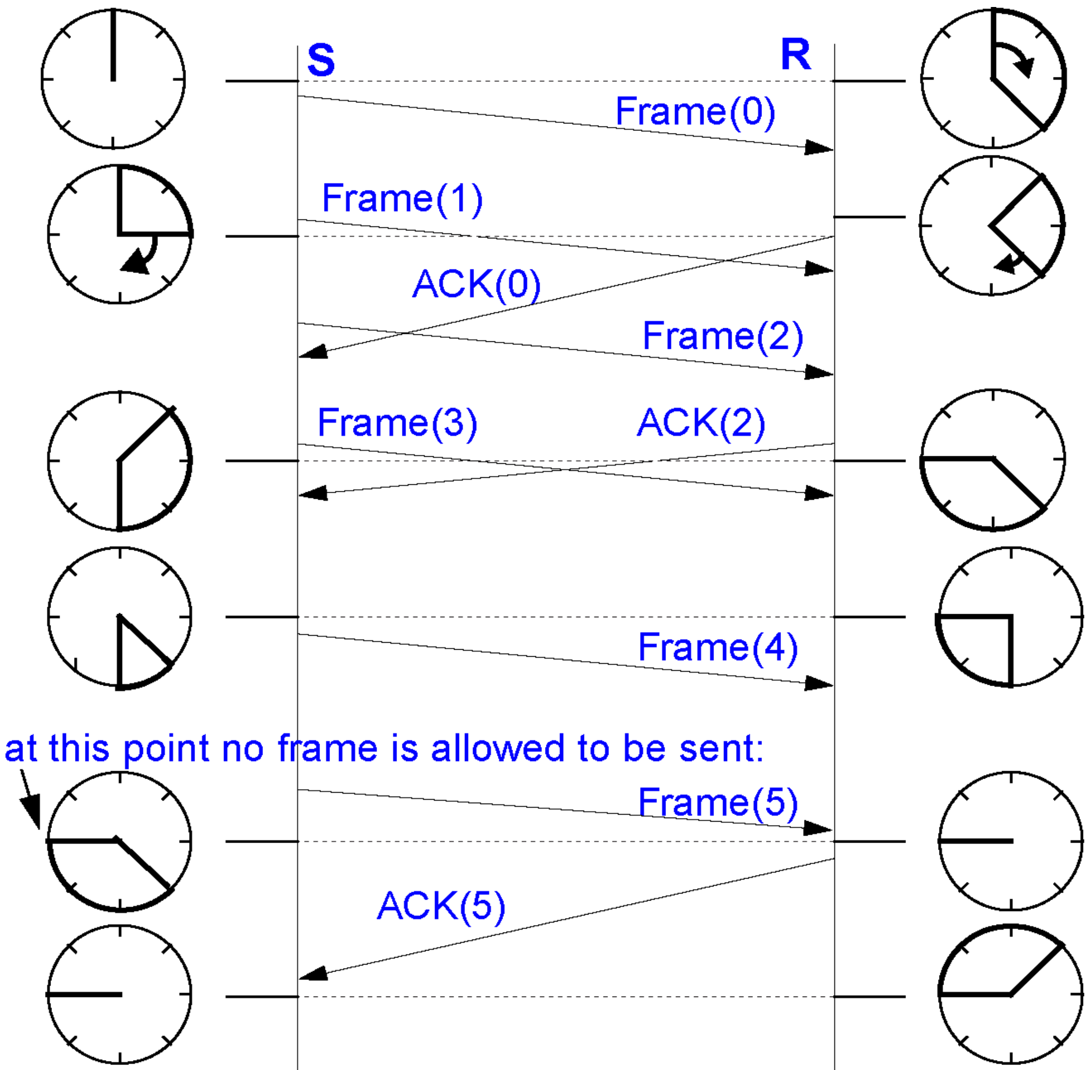
Sliding Window: Examples

Assuming

- 8 sequence numbers [0..7]
- max window size 3

ACK contains SeqNo

- like in Stop-and-Wait/ACK/SeqNo
- but ACK(SeqNo) ***may*** be interpreted as ACK for **all frames up to** SeqNo
- not every lost ACK frame leads to a timeout and retransmission



SEND action moves the upper bound, RECEIVE action moves the lower bound, w = 3

© Ralf Steinmetz, Technische Universität Darmstadt

Sliding Window

Stored frames at the sender

- maximum number defined by sender's window size (here 3)
- the frames not yet acknowledged by the receiver

Stored frames at the receiver

- **not necessary to store any frames**
  - the sender must store all unacknowledged frames and be able to retransmit
  - a receiver can NAK the lost frame and all higher seqnos (or not ACK)
- no use to store more than one receiver window size

ACK sent by receiver if frame

- **has been** identified as being correct
- **can be** transmitted correctly to the network layer
  - *correct* includes "in the right order"
  - *correct* includes usually: free from detected bit errors (but not always)

Sliding Window: Influence of the Window Size

Expected order

- if window size = 1
  - sequence always correct
- if window size n (n>1)
  - no requirement to comply with the sequence
  - but, size limited by the window size

Efficiency depends on (among other things)

- type and amount of errors on L1
- amount of data (in one frame) and rate of data
- end-to-end delay on L1
- window size

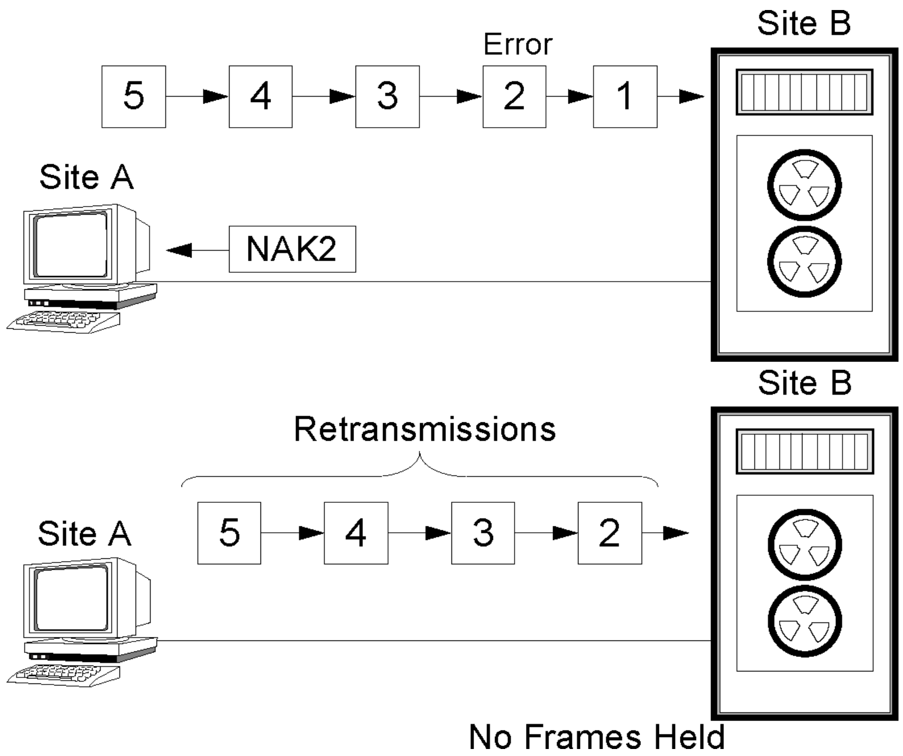
Performance consideration:

- if the window size is small
  - less memory needed per L2 relation
  - shorter average end-to-end delays at the L2 service interface
    also for higher error rates
    - this does not not necessarily mean shorter end-to-end delays for L7 !

Sliding Window: Go-Back-N (Error Treatment)

Procedure

- after a **faulty frame** has been received
  - receiver
    drops all frames with higher SeqNo
    until correct frame has been received

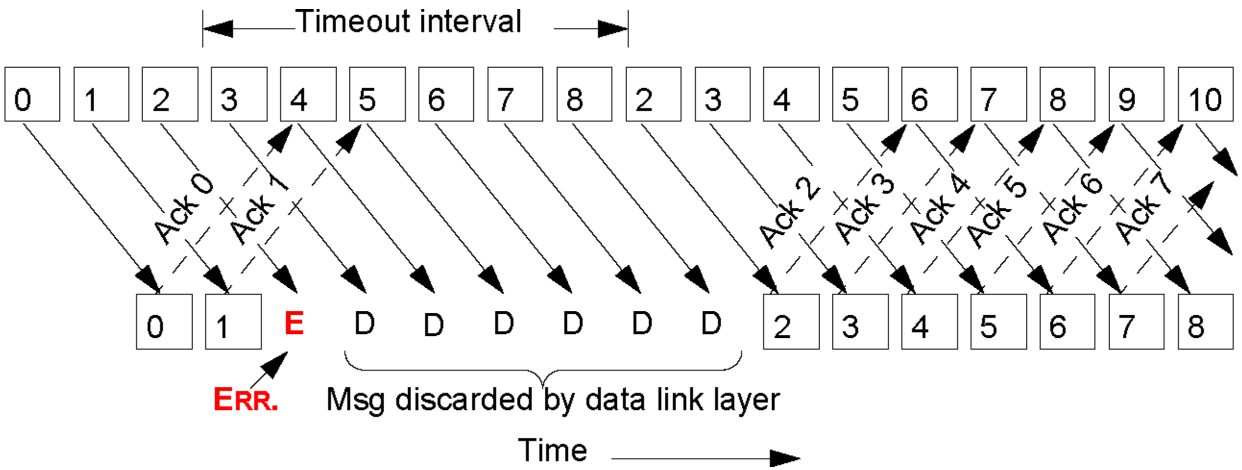


Evaluation

- simple
- receiver needs no buffers
- still quite poor utilization

Sliding Window: Go-Back-N

Example: sender: error detection by timeout



Sliding Window: Go-Back-N

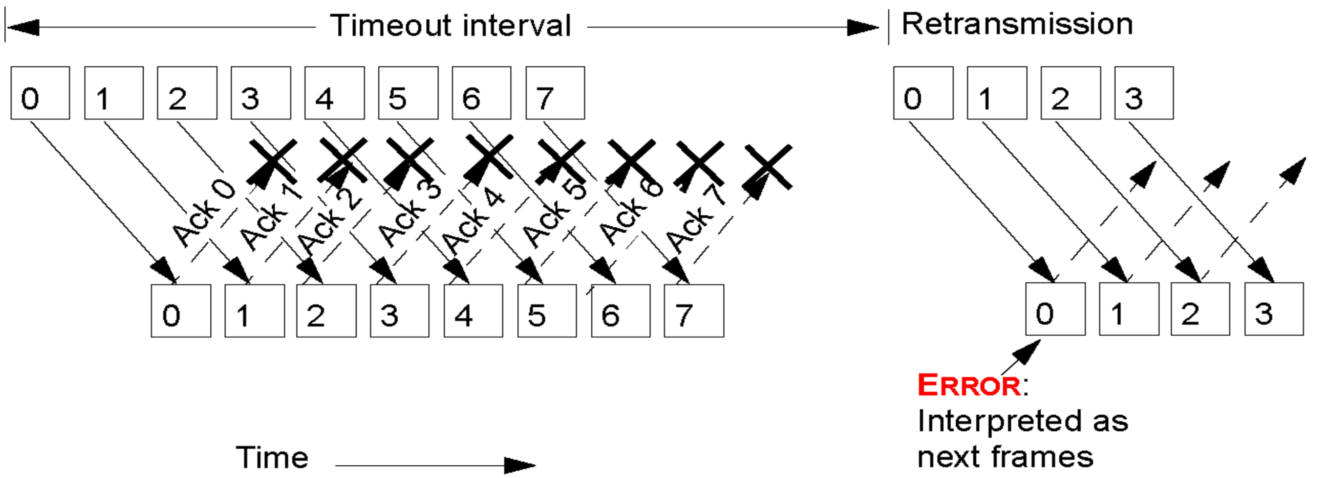
Correlation between

- window size and
- number of possible sequence numbers

→ at least *max. window size* ***strictly less than*** *range of sequence numbers*

Example for incorrect window size:

- amount of sequence numbers 8
- window size 8
- all ACKs lost



Sliding Window: Selective Repeat (Error Treatment)

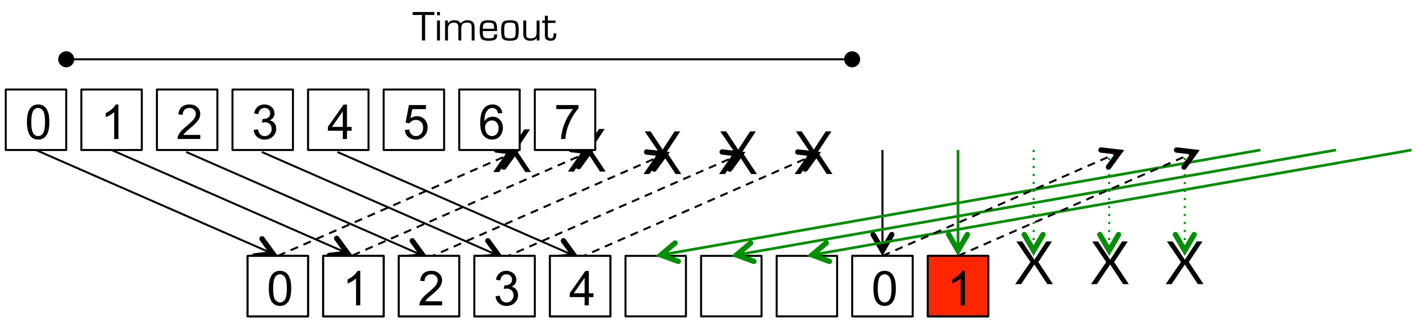
Procedure

- receiver stores all correct frames following a faulty one
- if sender is notified about an error
  it retransmits only the faulty frame
  - (i.e. not all the following ones, too)
- if received properly
  - receiver may have up to *max window size-1* frames in its buffer
- benefit
  - frames are delivered from L2 to L3 in correct sequence

Note: delivery from L2 to L3 can be *bursty*

- after a successful repeat
  receiver's L2 entity can deliver to receiver's L3 entity
  faster than
  sender's L2 can transmit to receiver's L2

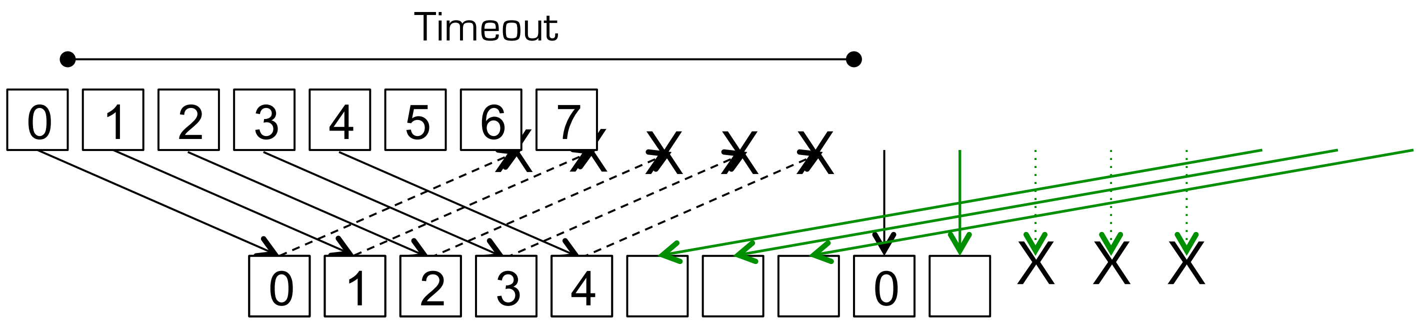
Sliding Window: Selective Repeat



'

- amount of sequence numbers 8
- window size 5
- **all ACKs are lost**, and the frame that has been lost last is the first one to arrive at the receiver again

Repeat of previous slide for non-animated use



Correlation between

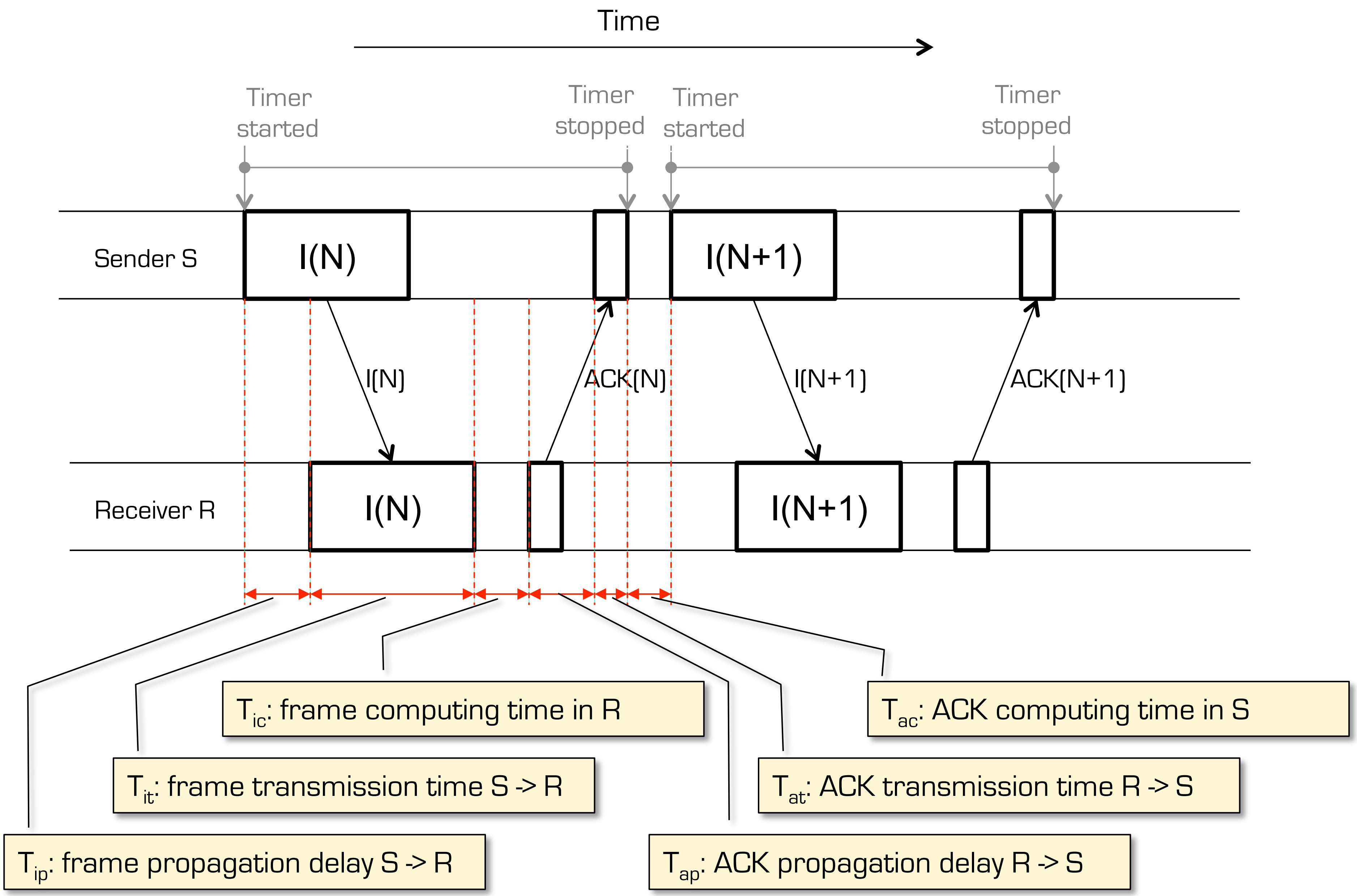
- window size and
- number of possible sequence numbers

→ max. window size <= 1/2 range of sequence numbers

Example for incorrect window size:

- amount of sequence numbers 8
- window size 5
- **all ACKs are lost**, and the frame that has been lost last is the first one to arrive at the receiver again

Recap: Utilization of Stop-and-Wait



Recap: Utilization of Stop-and-Wait

Best-case **utilization** of Stop-and-Wait

$$U = \frac{T_{it}}{T_{ip} + T_{it} + T_{ic} + T_{ap} + T_{at} + T_{ac}}$$

T_{ip} : frame propagation delay
T_{it} : frame transmission time
T_{ic} : frame computing time
T_{ap} : ACK propagation delay
T_{at} : ACK transmission time
T_{ac} : ACK computing time

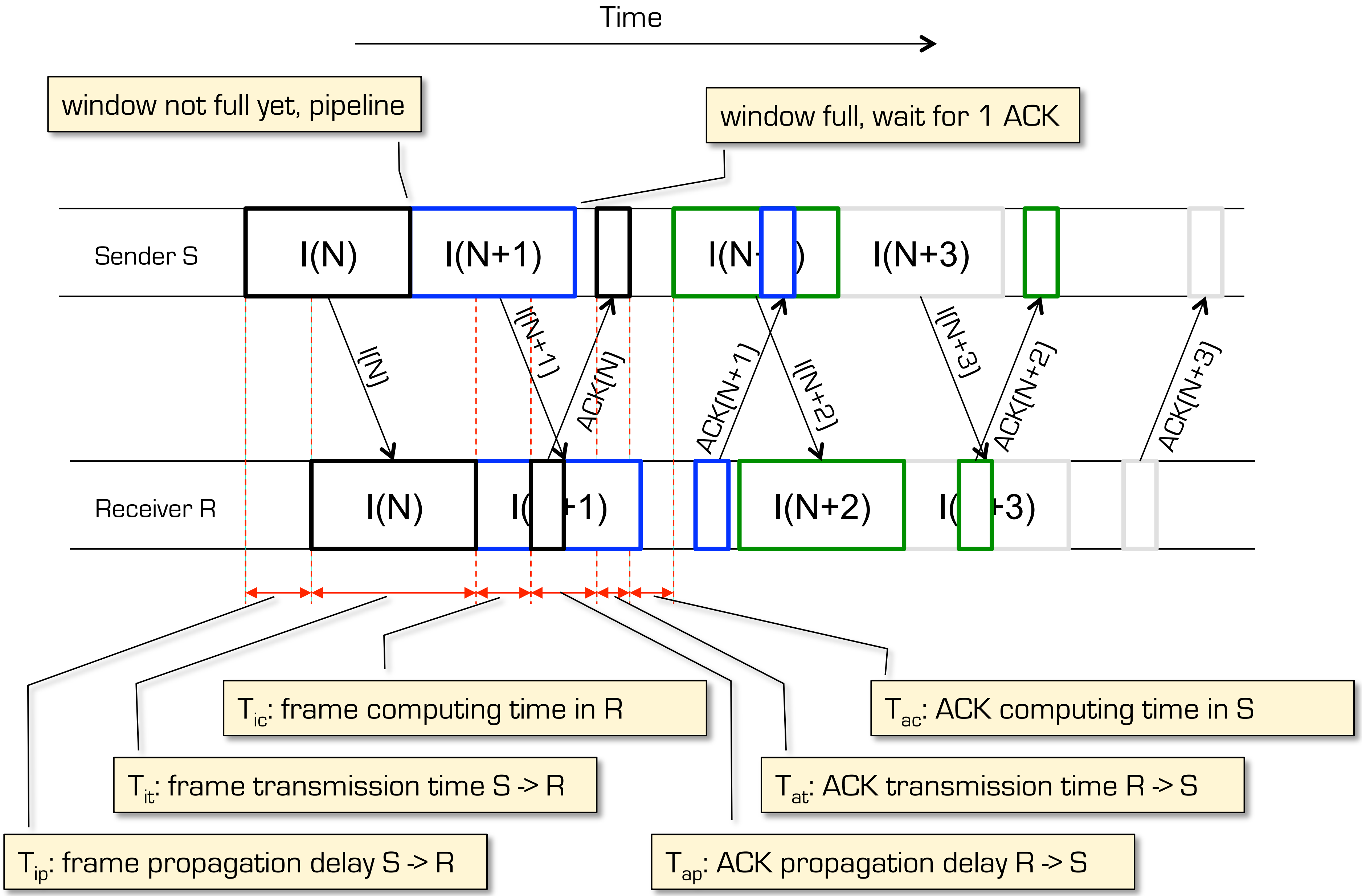
with the approximation

- $T_{ip} = T_{ap}$ – bits on the wire need same time both directions
- $T_{ic} = T_{ac} \ll T_{ip}$ – the *protocol* computing time is negligable
- $T_{at} \ll T_{it}$ – data frame transm. time much larger than ACK frame transm. time

Approximate best-case utilization of Stop-and-Wait:

$$U = \frac{T_{it}}{T_{it} + 2T_{ip}} = \frac{1}{1 + 2\frac{T_{ip}}{T_{it}}}$$

Utilization of Sliding Window



Utilization of Sliding Window

T_{ip} : frame propagation delay
T_{it} : frame transmission time
T_{ic} : frame computing time
T_{ap} : ACK propagation delay
T_{at} : ACK transmission time
T_{ac} : ACK computing time

Approximation

- $T_{ip} = T_{ap}$ bits on the wire need same time both directions
- $T_{ic} = T_{ac} << T_{ip}$ the *protocol* computing time is negligable
  note that T_{ac} is even less relevant because of pipelining
- $T_{at} << T_{it}$ data frame transm. time much larger than ACK frame transm. time

Two cases

- let the window size be k
- if $kT_{it}<2T_{ip}$: even in the best case, the sender must wait for an ACK
  the channel cannot be filled
- otherwise: the channel can be filled

$$U = \begin{cases} \dfrac{kT_{it}}{T_{it} + 2T_p} = \dfrac{k}{1 + 2\dfrac{T_{ip}}{T_{it}}} & \text{if } \left(k < 2\dfrac{T_{ip}}{T_{it}}\right) \\ 1 & \text{otherwise} \end{cases}$$

Note: The best case is identical for Go-Back-N and Selective-Repeat

INF3190 - Data Communication

Data Link Layer (cnt'd)

Carsten Griwodz

Email: griff@ifi.uio.no

most slides from: Ralf Steinmetz, TU Darmstadt

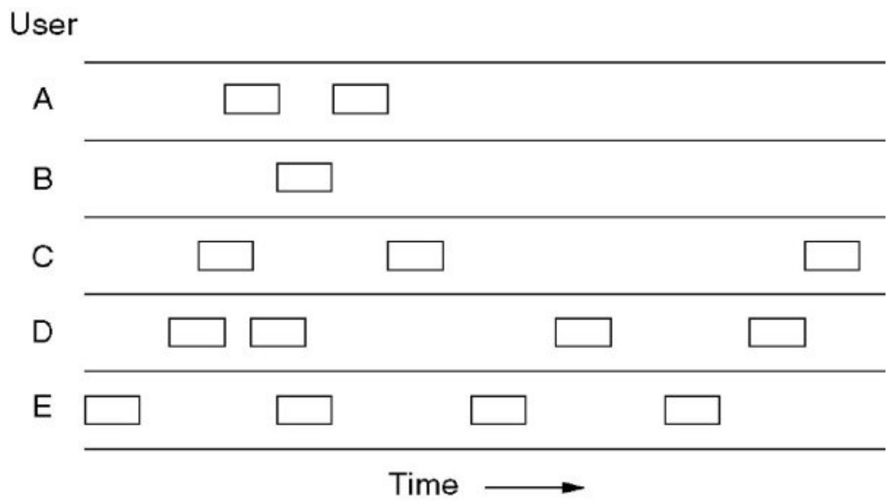
and a few from Olav Lysne, J. K. Kurose og K. W. Ross

MAC sublayer

Medium Access Control (MAC)

Need for a MAC sub-layer

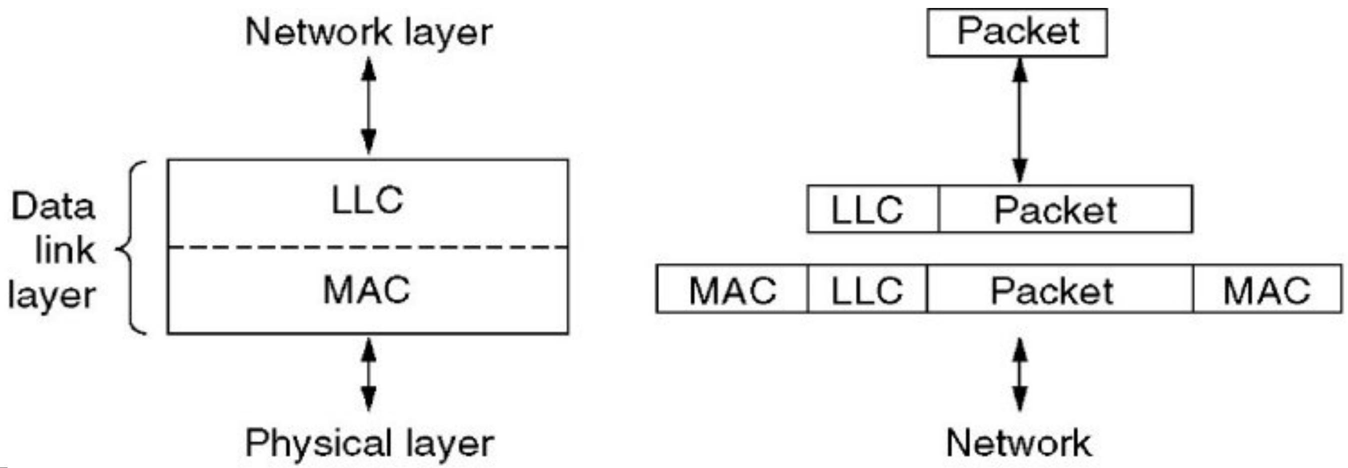
- IF several senders share a channel/medium
- THEN it is very likely that two or more will start communicating at the same time



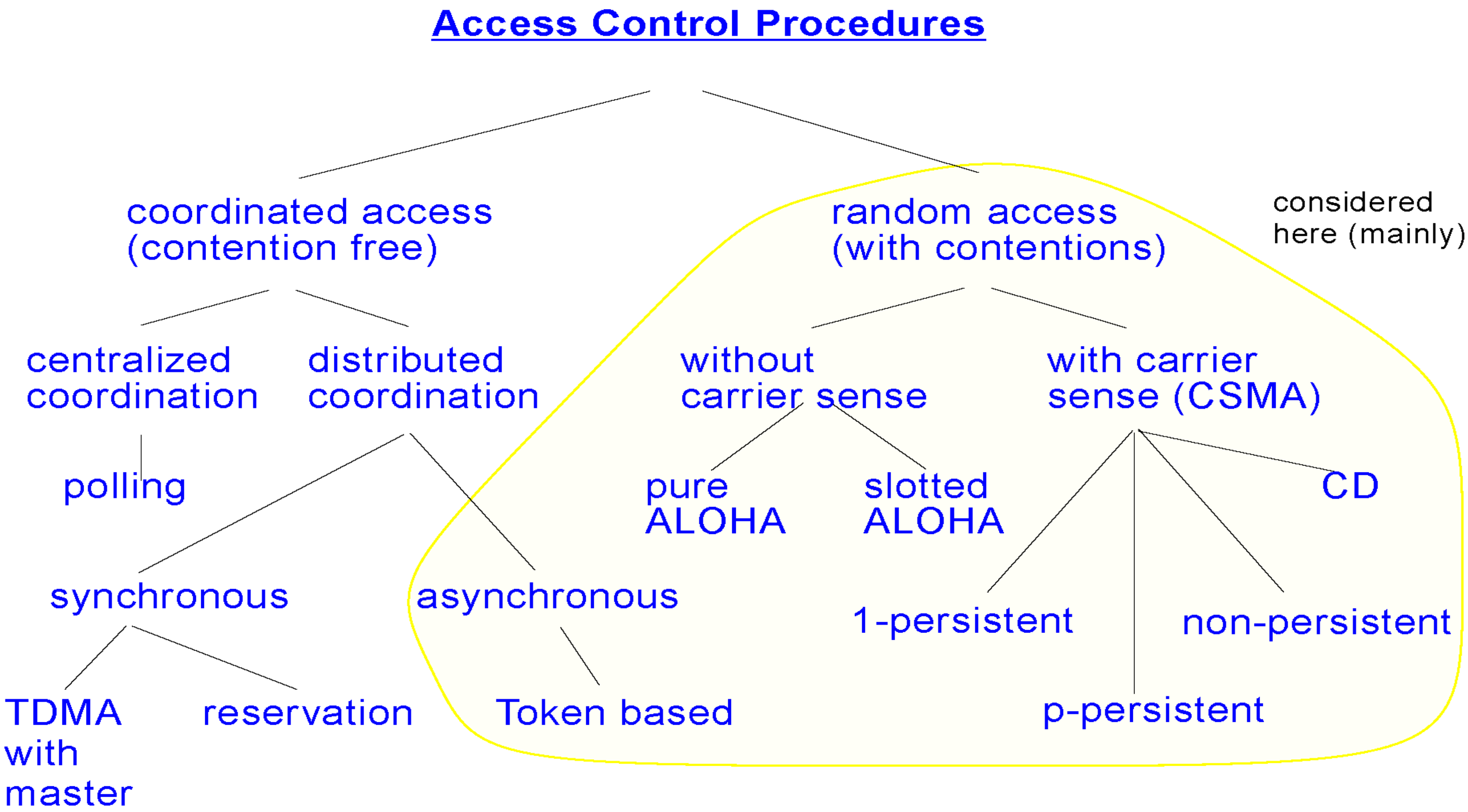
MAC "avoids chaos"

Important "sub layer" of L2

- lower part of L2

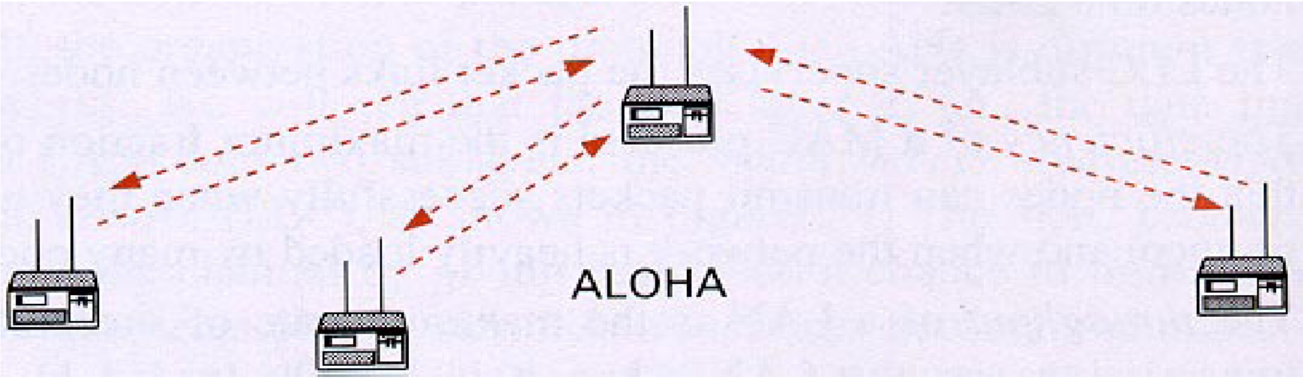


Dynamic Channel Allocation Schemes



MAC sublayer
Random access protocols

ALOHA



History

- University of Hawaii, 1970
- originally via radio station with 9.600 bps
  - 413 MHz: centralized sender (to everybody) on earth
  - 407 MHz: return channel used by all receivers

Principle

- sending without any coordination whatsoever
- sender listens to the (return-) channel (after sending)
- in case of collision
  - retransmits after a random time interval

CSMA (Carrier Sense Multiple Access)

ALOHA

- station sends and realizes only *afterwards* if it was actually able to send

CSMA Principle

- check the channel *before* sending
- channel status
  - busy:
    - no sending activity
    - wait until channel is re-checked
      OR
      keep checking continuously until channel is available
  - available:
    - send
    - still possibility for collision exists!
  - collision:
    - wait for a random time

CSMA/CD Applet

CSMA Variation Non-Persistent

Principle

- Request to send → check channel
- channel status
  - busy:
    - wait without checking the channel continuously,
    - channel re-check only after a random time interval
  - available:
    - send
  - collision:
    - wait for a random time, then re-check channel

Properties

- assumption that other stations want to send also
  therefore it is better to have the intervals for the re-checks randomly determined
- Improved overall throughput
- longer delays for single stations

CSMA Variation 1-Persistent

Principle

- Request to send → channel check
- channel status
  - busy:
    - continuous re-checking until channel becomes available
  - available:
    - send
    - i. e. 1-persistent: send with probability 1 immediately when both data is available and the channel is free
  - collision:
    - wait random time, then re-check channel

- Properties
  - if channel is available: send with probability 1 (thus 1-persistent)
  - minimize the delay of sending station
  - but a lot of collisions at higher load (low throughput)

CSMA Variation P-Persistent

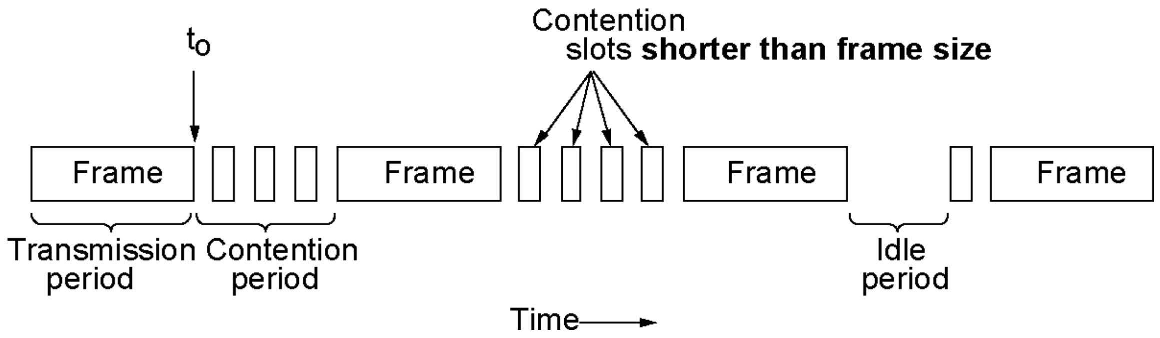
Principle

- Requires an understanding of "slot", e.g. a maximum frame duration
- Request to send → channel check
- channel status
  - busy:
    - wait for the next slot, re-check (continuously)
  - available:
    - Send with Probability p,
    - wait with probability 1-p for the next slot,
    - check next slot
      - busy: wait random time, re-check channel
      - available: send with probability p,
        wait for next slot with probability 1-p, ...etc.
      - collision: ..etc
  - collision:
    - wait random time, re-check channel

Properties

- compromise between delay and throughput
- defined by parameter p

CSMA Variation CD



Carrier Sense Multiple Access with Collision Detection

- CSMA 1-persistent with CD

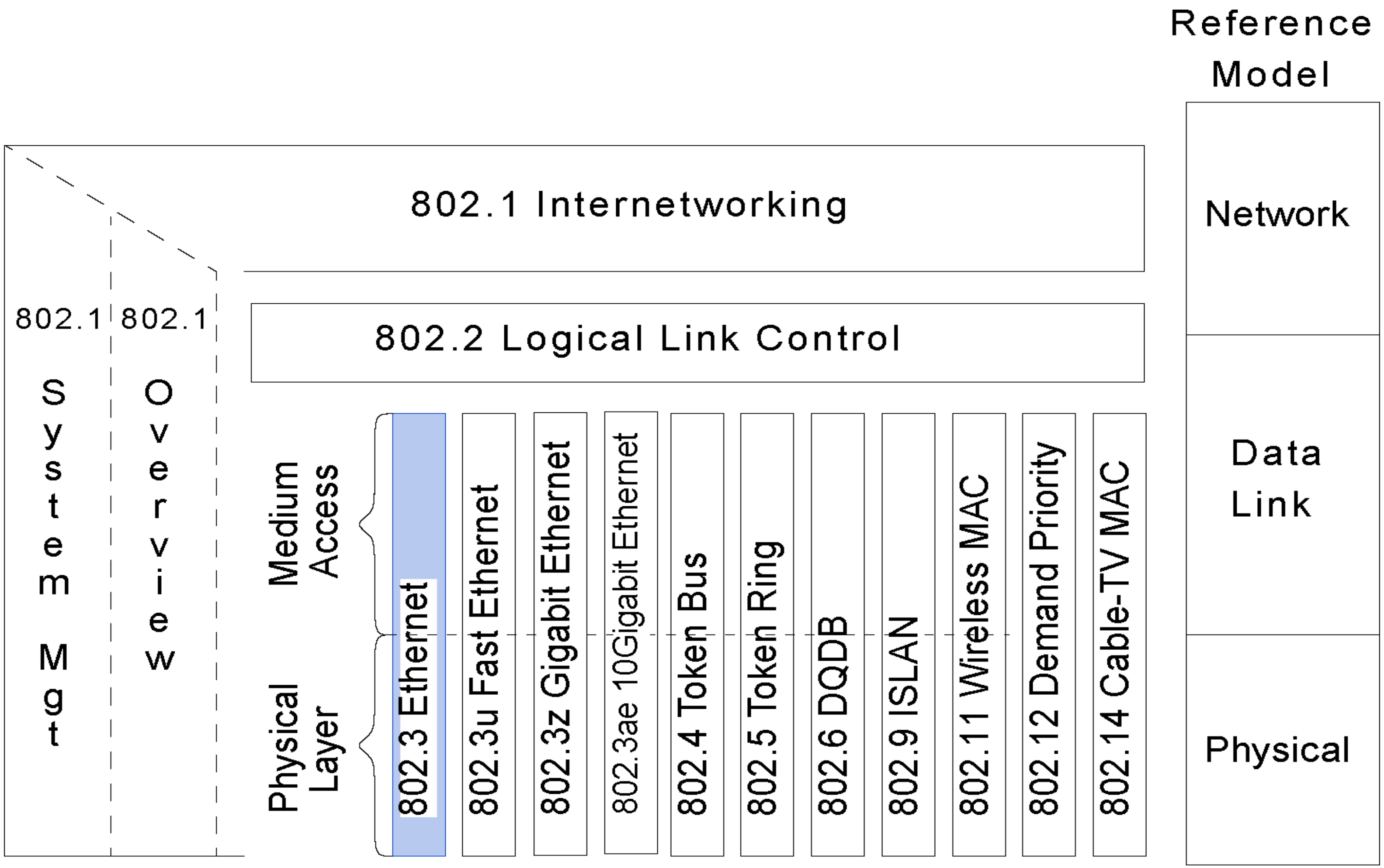
Principle:

- sending station interrupts transmission as soon as it detects a collision
  - saves time and bandwidth
  - frequently used (802.3, Ethernet)
  - station has to realize DURING the sending of a frame if a collision occurred

Comparing ALOHA, CSMA.., CSMA CD

<table>
<tr><th colspan="2" rowspan="2"></th><th colspan="3">channel is checked (regarding decision to send, not with regard to collision)</th><th colspan="3">behavior in case of desire to send and if one of the following states has been determined</th><th rowspan="2">Time slot</th></tr>
<tr><th>before</th><th>during</th><th>after</th><th>busy</th><th>available</th><th>collision</th></tr>
<tr><td>ALOHA</td><td>pure</td><td></td><td></td><td>X</td><td colspan="2">sender does not know these conditions</td><td>re-transmit after random time interval</td><td></td></tr>
<tr><td rowspan="3">CSMA</td><td>nonpersist</td><td>X</td><td></td><td>(X)</td><td>re-check channel only after random time interval</td><td rowspan="2">sends immediately</td><td rowspan="3">wait random time interval then re-check channel and send (if possible) (depending on algorithm "available/ busy")</td><td></td></tr>
<tr><td>1 persist.</td><td>X</td><td></td><td>(X)</td><td>Continuous wait until channel is Available</td><td></td></tr>
<tr><td>p persist.</td><td>X</td><td></td><td>(X)</td><td>initially: continuous wait until chnl/slot available</td><td>sends with probability p, waits with probability 1-p (for next slot, then re-checks status)</td><td>X</td></tr>
<tr><td colspan="2">CSMA/CD</td><td>X</td><td>X</td><td></td><td colspan="2">depending on procedure, (see above) 1-persistent is e.g. Ethernet</td><td>Terminates sending immediately, waits random time</td><td></td></tr>
</table>

802.3: History and Basics



IEEE 802.3: CSMA / CD

History

- 1976
  - Ethernet by Xerox, Robert Metcalf (2,94 Mbps)
- 1980
  - Ethernet industrial standard by Xerox, Digital Equipment (today part of HP) and Intel (10 Mbps)
- 1985
  - IEEE 802.3 based on Ethernet

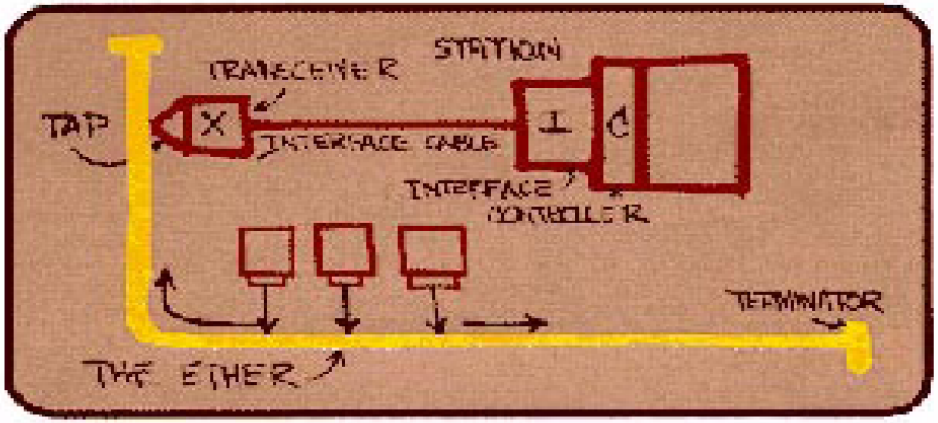


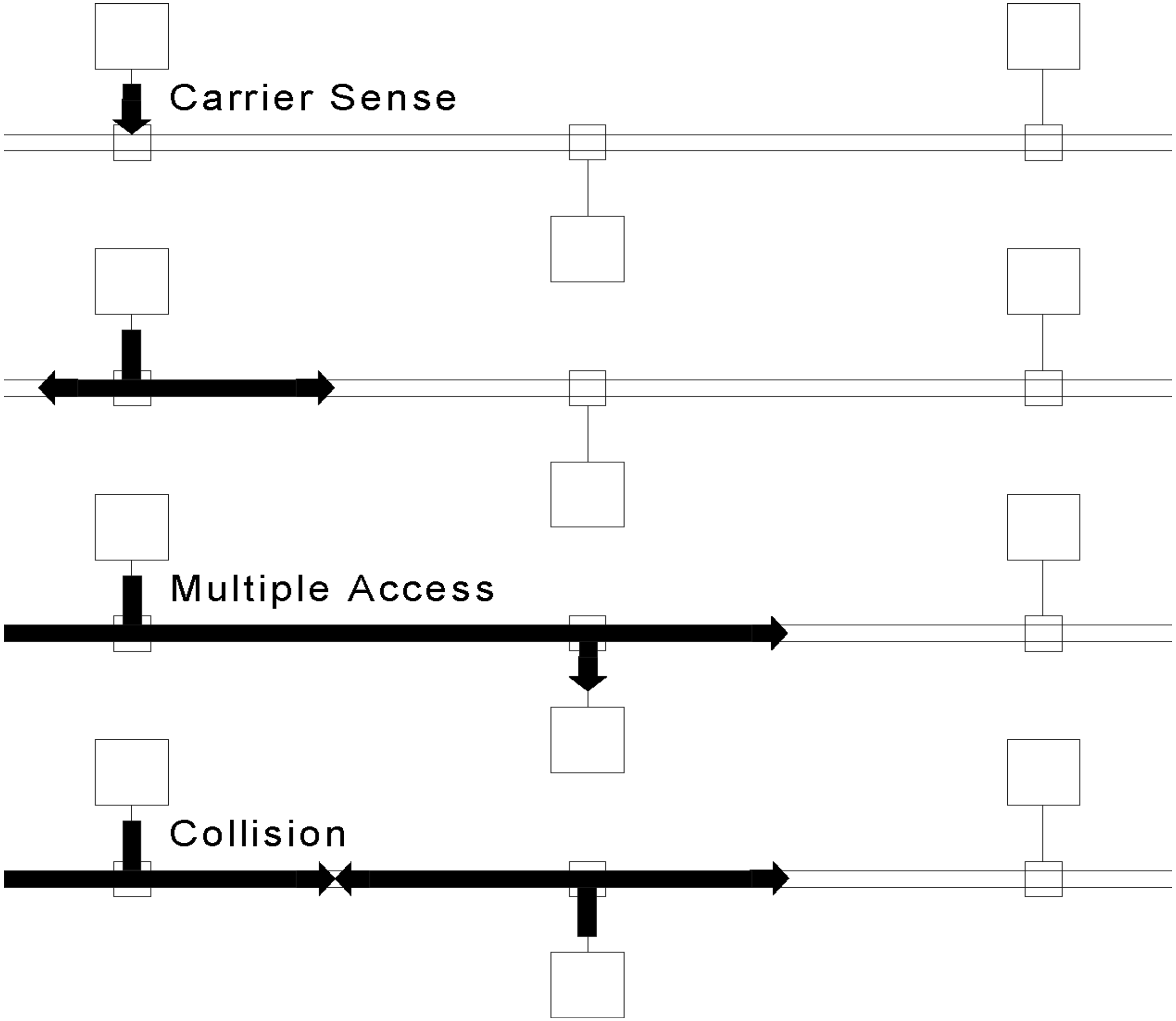
Figure 1. Robert Metcalfe's drawing of the first Ethernet design.

IEEE 802.3

- specifies a family based on the 1-persistent CSMA/CD systems
- (1 -) 10, 100 Mbps, 1, 10, 100/40, ... Gbps on different media
- standards specify also L1

1-persistent CSMA / CD

IEEE 802.3: CSMA / CD



802.3: Frame Format

Frame Length

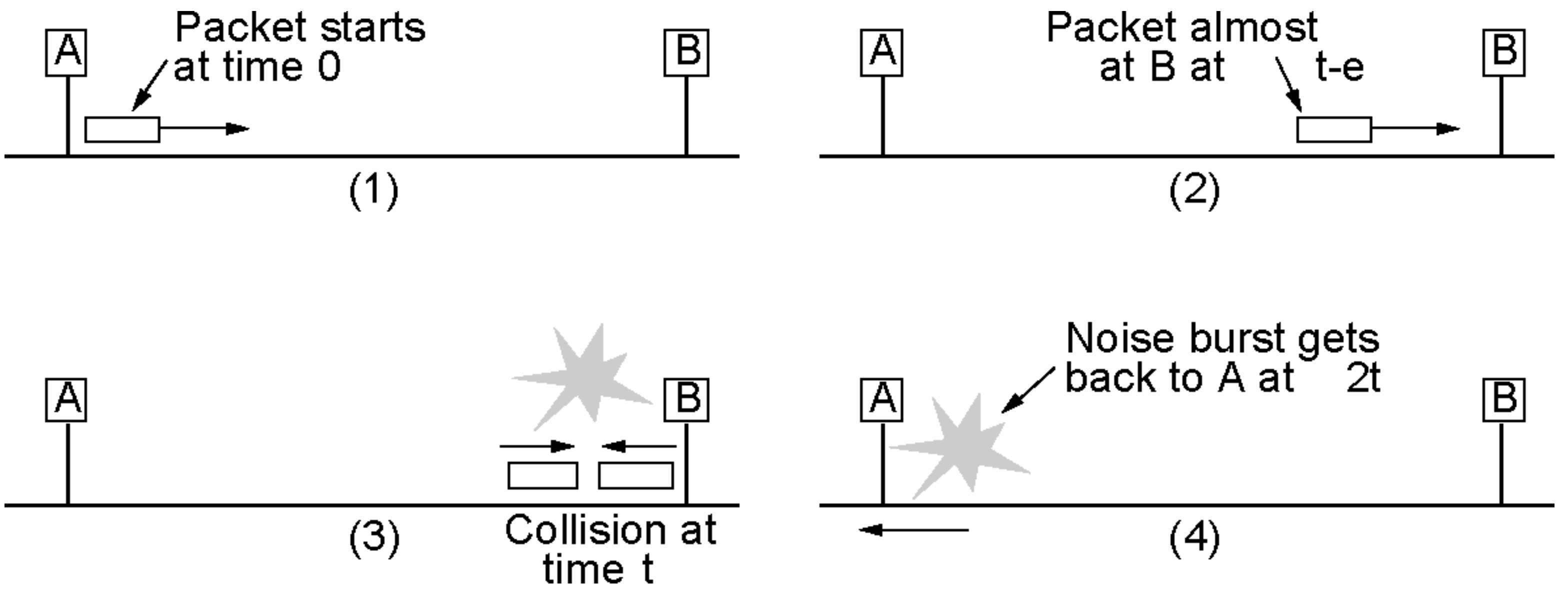
- IEEE 802.3 frames have ***minimum size*** restrictions based on network bandwidth (64 bytes, of these payload 46)

- The first bit of the frame must have reached every other station and the collision must be visible to the sender if the collision occurs between the most distant senders

- When necessary, the data field should be padded (with octets of zero) to meet the 802.3 minimum frame size requirements

- Padding is not part of the packet delivered to L3

802.3: Illustration for Minimum Length



802.3: Behavior at a Collision

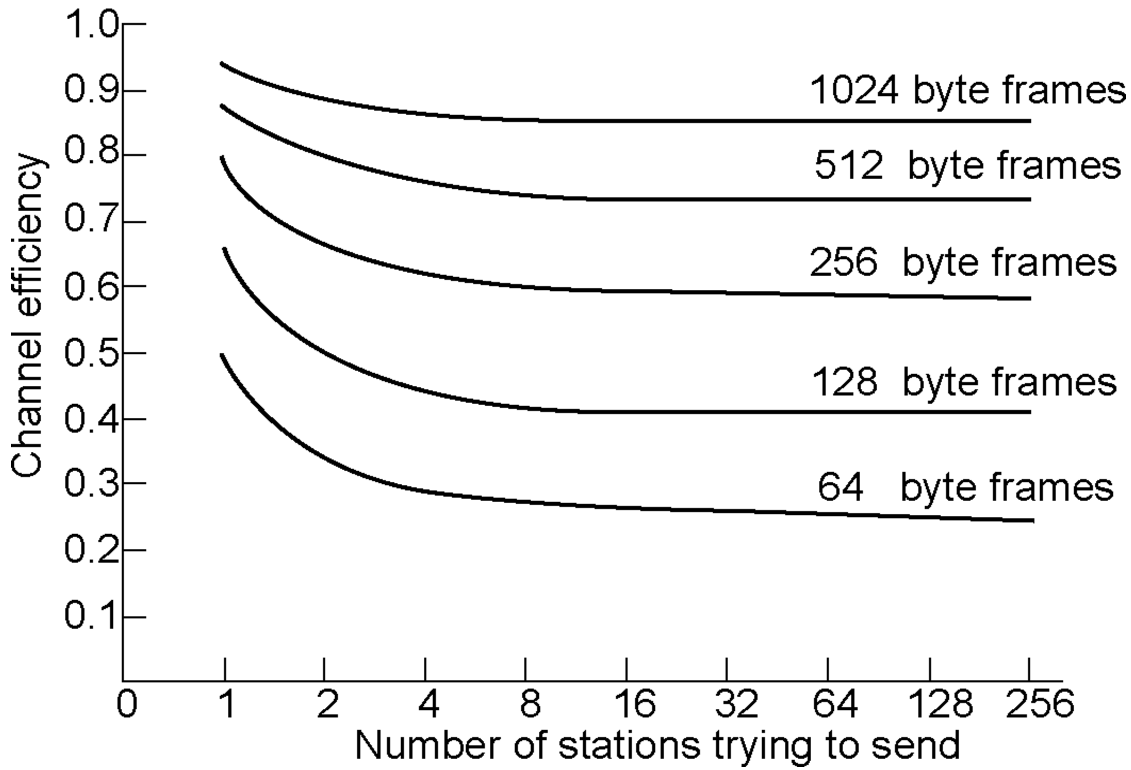
... collision after first request to send	next attempt after a waiting ... frames
1st	0 or 1
2nd	0, 1, 2 or 3
3rd	0, 1, 2, 3, 4, 5, 6 or 7
...	
nth	0,..., 2^n-1
16th	error message to L3

Binary Exponential Backoff Algorithm

802.3: Behavior at a Collision

Behavior

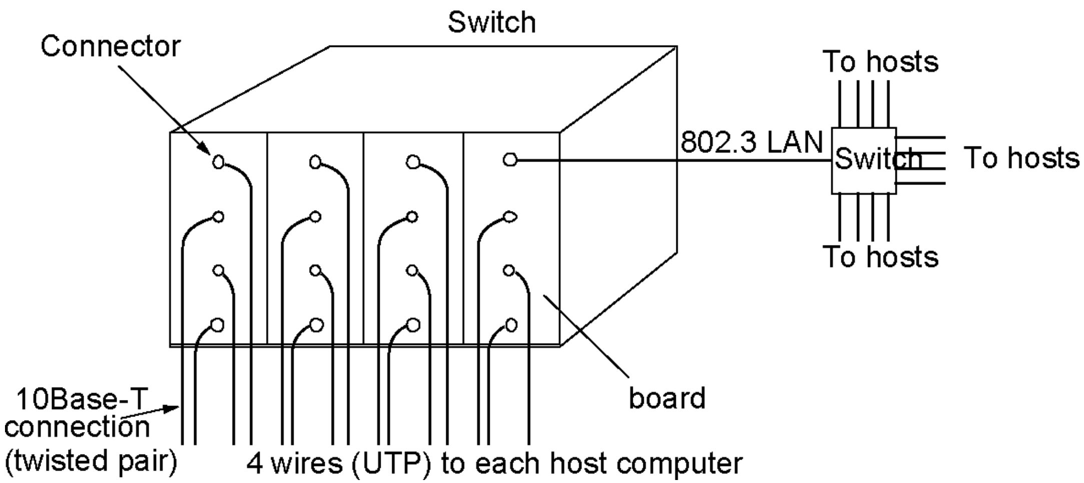
- while increasing load → longer waiting periods
- if more stations → lower utilization
- if longer frames → higher utilization



Switched 802.3 LANs

Increasing the throughput of 802.3 versions

Switch as relaying center



- station sends frame
- switch tries to locate receiver
  - remember (cache) port of stations that have been **senders** before
  - if unknown, send to all

Collision domain

- the stations that can affect each other through collisions
  - when receiver is known: senders addressing same receiver at same time
  - when receiver is unknown: all stations

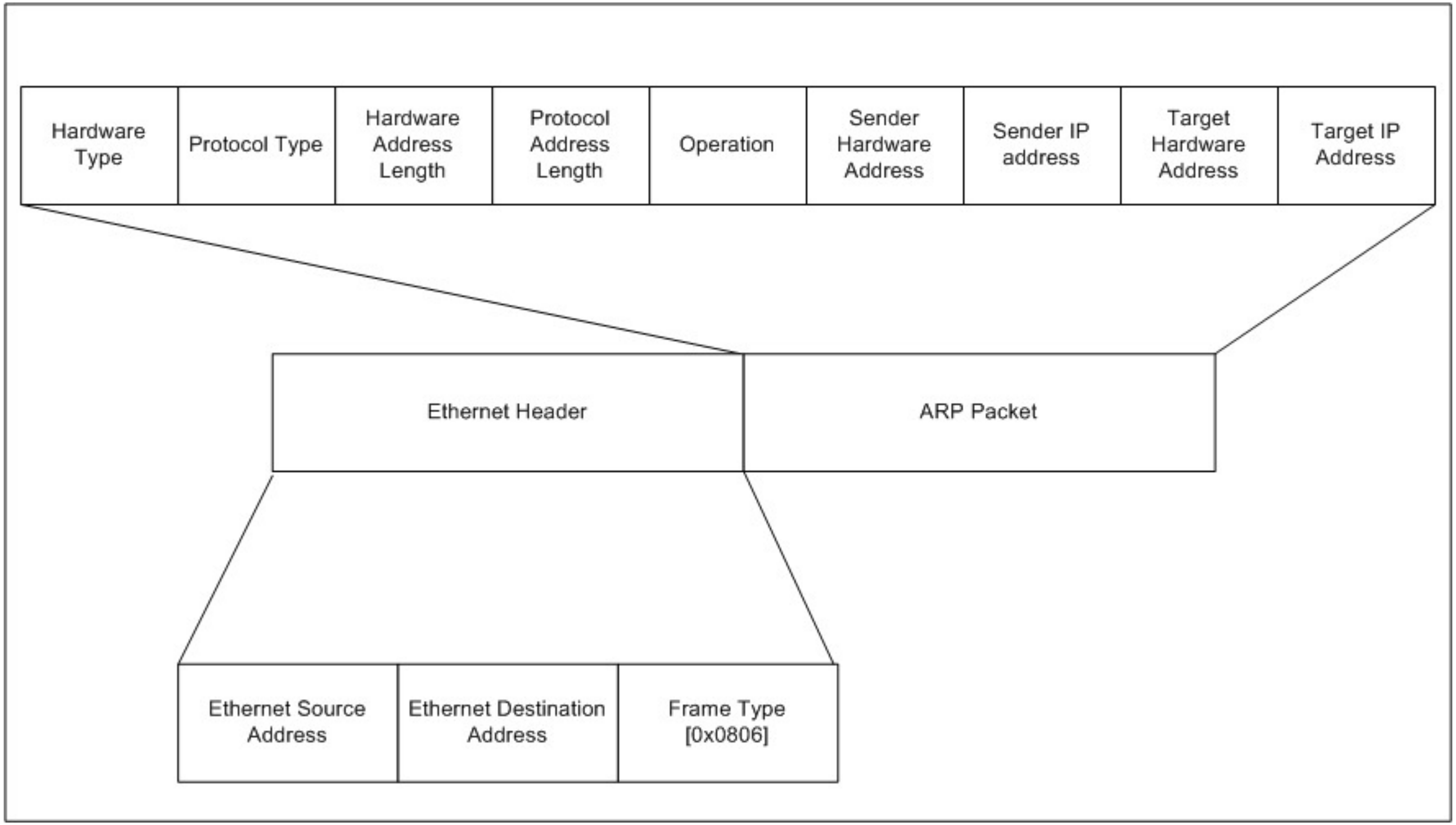
802.3: Conclusion CSMA / CD

Properties

- + most widely spread
- + stations connect without shutting down the network
- + practically no waiting period during low workload

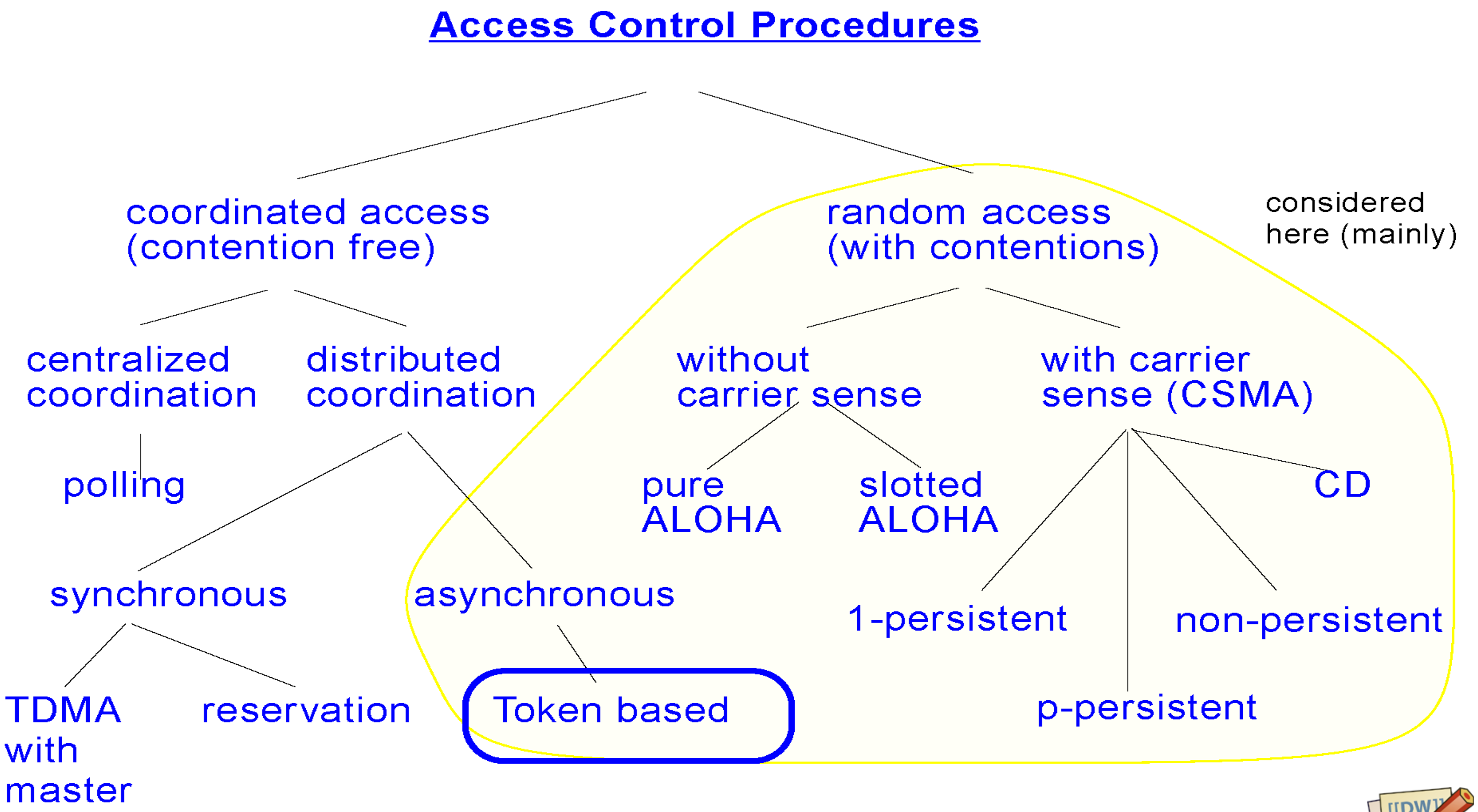
- - analog components for collision recognition
- - minimum frame size (64 bytes)
- - not deterministic (no maximum waiting period)
- - no prioritizing
- - when load increases, collisions also increase

What is ARP ?


Hardware Type
Protocol Type
Hardware Address Length
Protocol Address Length
Operation
Sender Hardware Address
Sender IP address
Target Hardware Address
Target IP Address
Ethernet Header
ARP Packet
Ethernet Source Address
Ethernet Destination Address
Frame Type [0x0806]

MAC sublayer
Token Ring

IEEE 802.5: Token Ring



Token Ring Trainer Applet

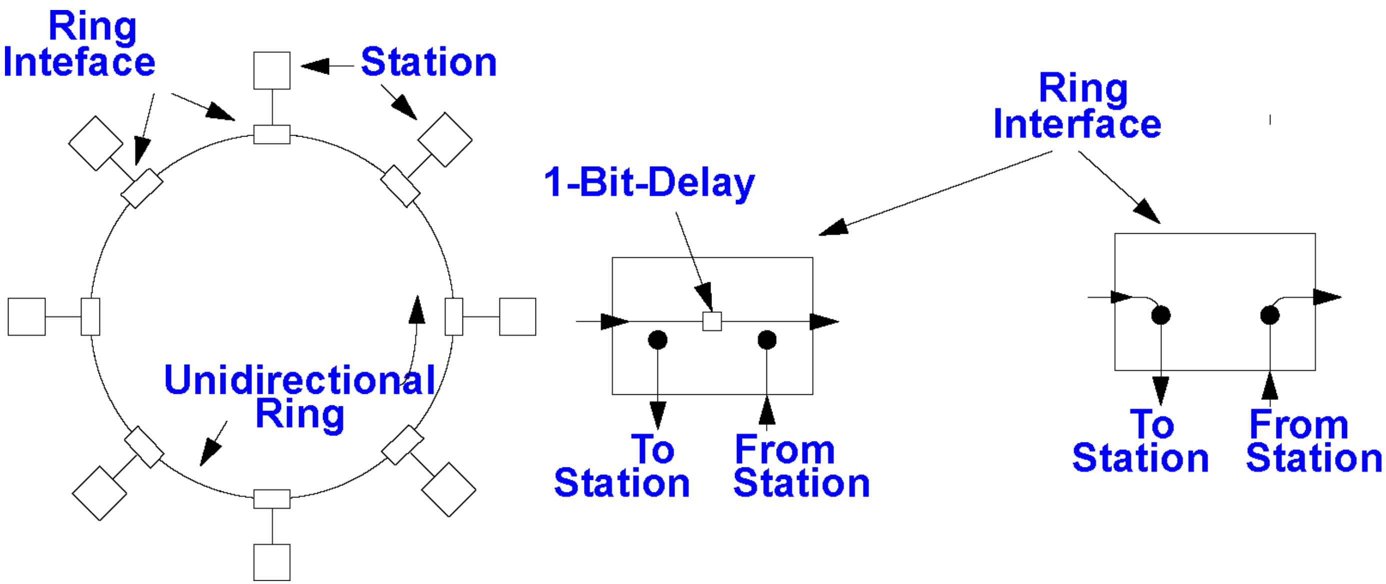
802.5: Ring Topology

Ring

- not really a broadcast medium, but
  - a multitude of point-to-point lines

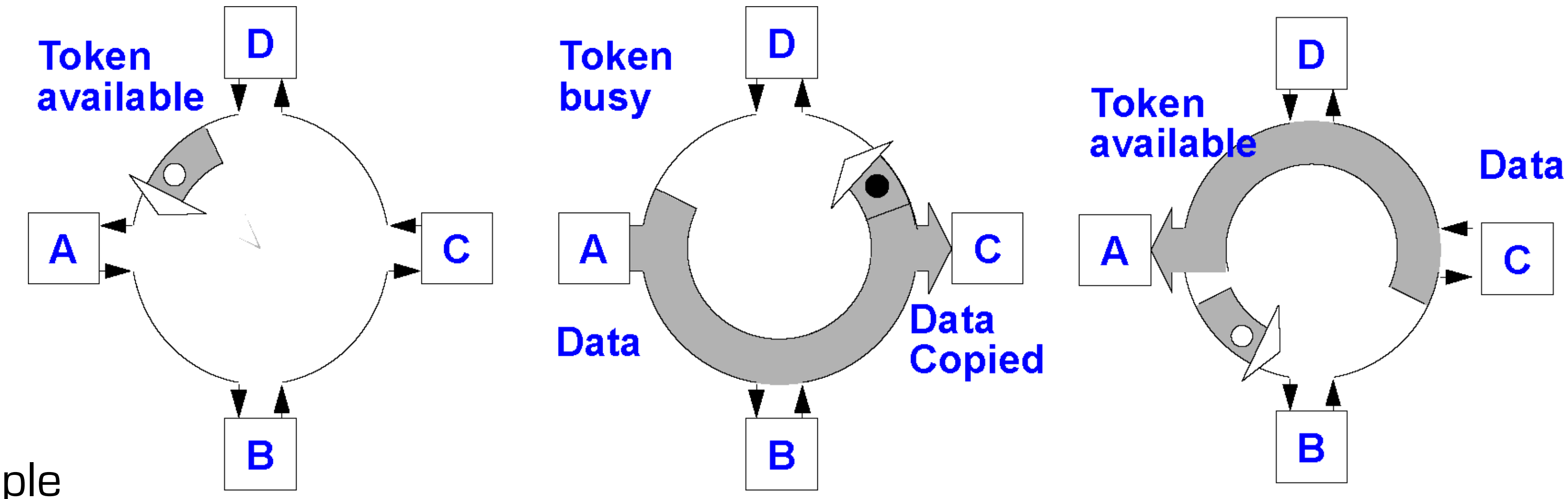
Station

- copies information bit by bit from one line to the next (active station)



802.5: MAC Protocol

Token Protocol



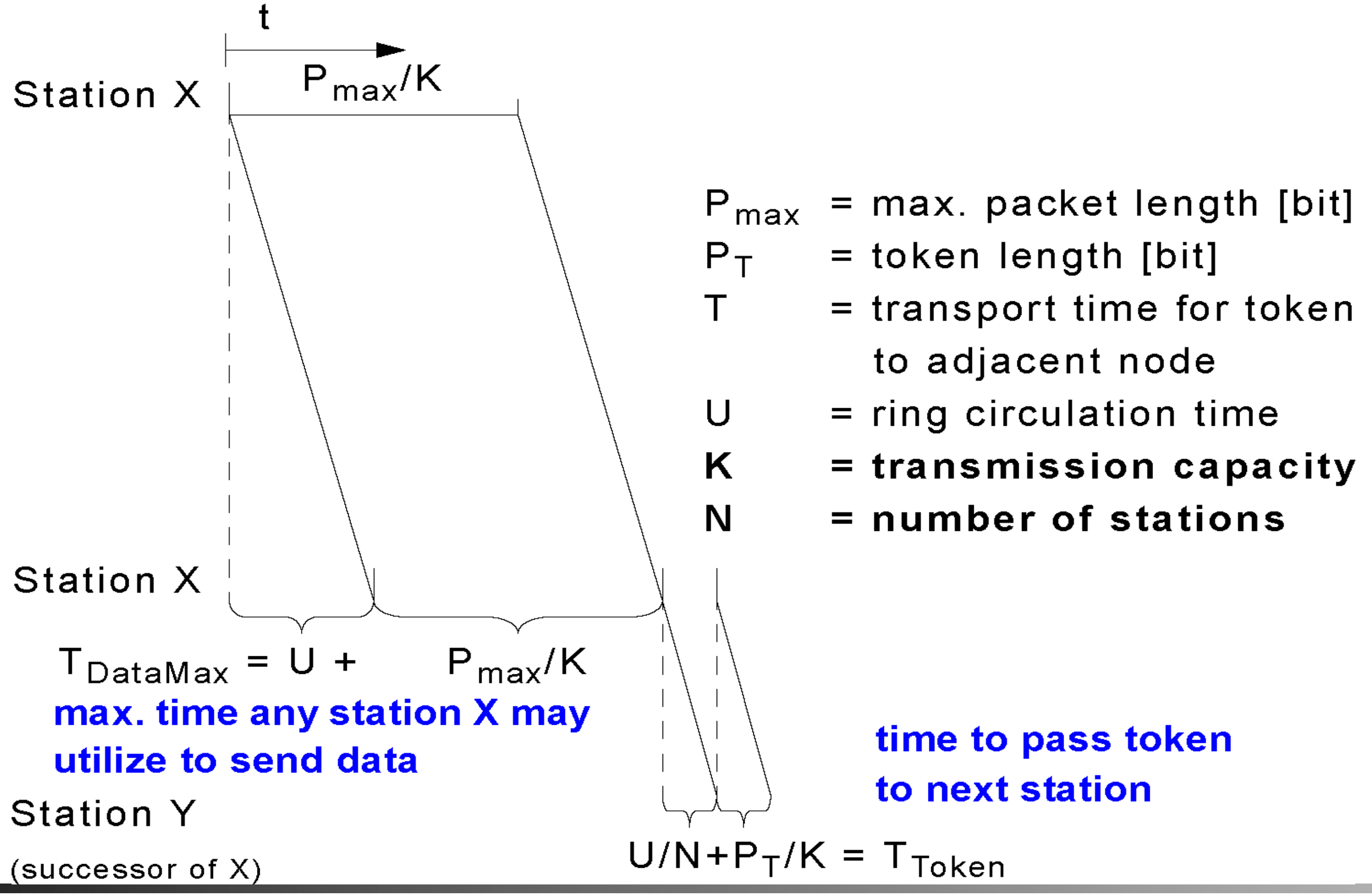
Principle

- Token
  - frame with special bit pattern
- one token circulates on the ring
  - 1: before station is permitted to send
    - it must own and remove the token from the ring
  - 2: station may keep the token for a pre-defined time and may send several frames
  - 3: after sending
    - the station generates a new token

802.5: Maximum Waiting Period

What is the maximum waiting period for a station before it receives permission to send again?

- i.e. all stations want to send with the max. amount of allowed time



802.5: Maximum Waiting Period

What is the maximum waiting period for a station before it receives permission to send again?

W = maximum waiting period:

W = all others are sending + token rotates x-times

$= (N-1)(P_{max}/K + U) + N(P_T/K + U/N)$

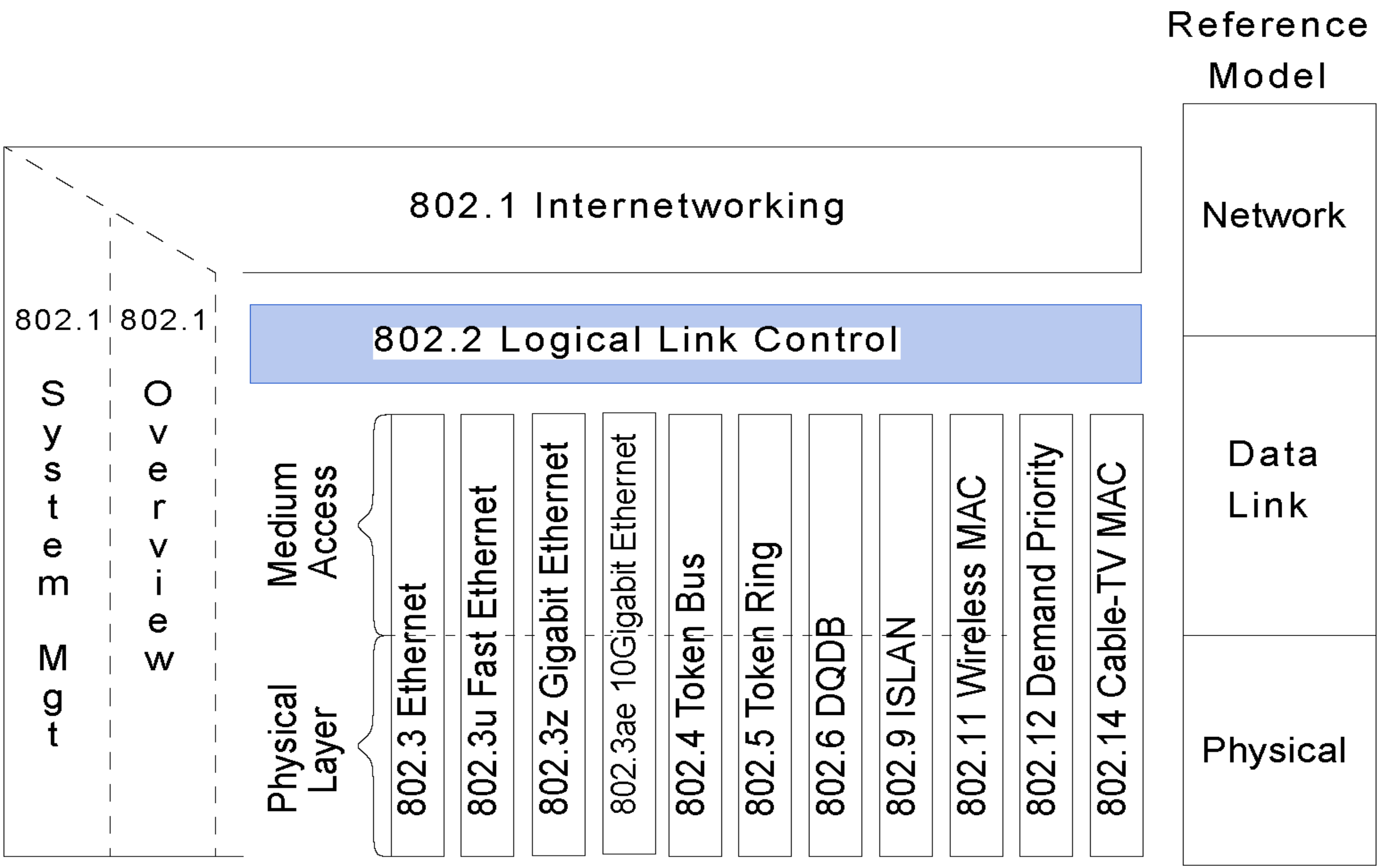
$= (N-1)(P_{max}/K + U) + NP_T/K + U$

$= (N-1)(P_{max}/K + U) + U$

Note: $NP_T/K = 0$ for $P_T \ll P_{max}$

LLC sublayer
IEEE 802.2

802.2: Logical Link Control



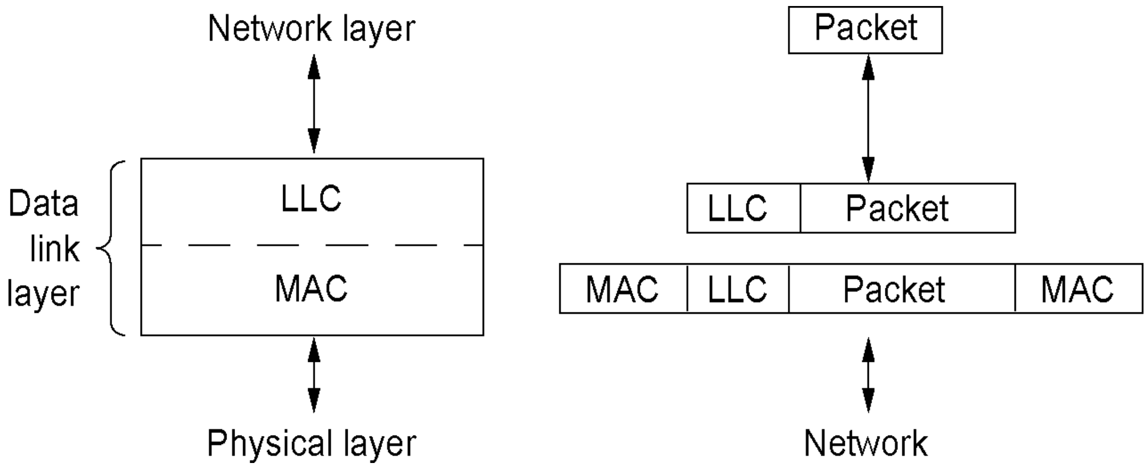
802.2: Logical Link Control

- Function
  - subset of HDLC
    - High Level Data Link Control HDLC
  - common interface
    - to L3 for all underlying LAN/MAN/WAN components

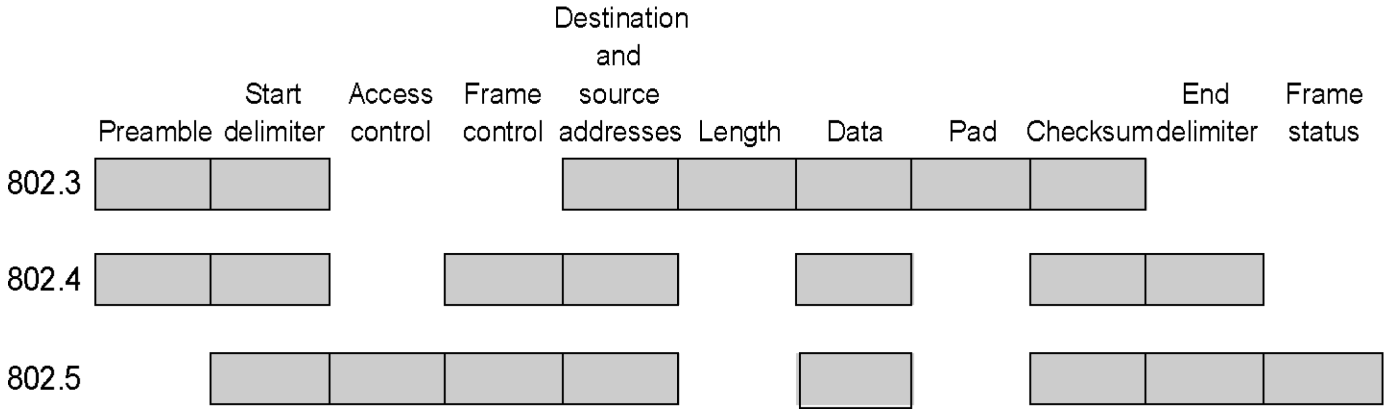
- Services
  - unacknowledged connectionless (unreliable datagram)
    - upper layers ensure
      - that sequence is maintained, error correction, flow control
  - acknowledged connectionless (acknowledged datagram)
    - each datagram is followed by exactly one acknowledgement
  - connection oriented
    - connect and disconnect
    - data transmission incl. acknowledgement, guaranteed delivery to receiver
    - maintaining the sequence
    - flow control

LLC Frame

- Format
  - includes LLC Service Access Points SAPs for source and destination



- Varying AC frames:
  - formats



Ethernet variants

Standardizing Ethernet

802.2 Logical Link Control

802.3 Contention Bus Standard 10base 5 (Thick Net)

- ~~802.3a Contention Bus Standard 10base 2 (Thin Net)~~
- ~~802.3i Twisted-Pair Standard 10base T~~
- ~~802.3j Contention Bus Standard for Fiber Optics 10base F~~
- 802.3u 100-Mb/s Contention Bus Standard 100base T
- 802.3x Full-Duplex Ethernet
- 802.3z Gigabit Ethernet
- 802.3ab Gigabit Ethernet over Category 5 UTP
- 802.3ae 10 Gigabit Ethernet over fiber
- 802.3av 10 Gigabit Ethernet over Passive Optical Network (EPON)
- 802.3bm 100G/40G Ethernet for optical fiber
- ...

IEEE 802.3u: Fast Ethernet

- History
  - High-Speed LAN compatible with existing Ethernet
  - 1992:
    - IEEE sets objective to improve existing systems
  - 1995:
    - 802.3u passed as an addendum to 802.3
    - (alternative solution containing new technology in 802.12)

- Principle
  - retain all procedures, format, protocols
  - bit duration
    - reduced from100 ns to 10 ns

- Properties: CSMA/CD at 100 Mbps
  - cost efficient extension of 802.3
  - very limited network extension
    - sender has to be able to recognize collision during simultaneous sending
      - network extension must not exceed the size of the min. frame
      - frame at least 64 byte, i.e. 5 ms at 100 Mbps per bit
    - i.e. extension only a few 100 meters "collision domain diameter" = 412 m
      - (instead of 3000m)
  - many collisions (lower utilization)

IEEE 802.3u: Fast Ethernet

- Basics
  - actually 10Base-T (Unshielded Twisted Pair)
  - ***Hub*** on L2

- Medium

Name	Cable	Max. segment	Advantages
100Base-T4	**Twisted pair**	**100m**	**Uses category 3UTP**
100Base-TX	**Twisted pair**	**100m**	**Full duplex at 100Mbps (5UTP)**
100Base-F	**Fiber optics**	**2000m**	**Full duplex at 100Mbps**

- 100Base-F (fiber optics):
  - maximum segment length of 2000 m too long for collision recognition
    → may be used only in context with buffered hub ports
    - collisions not possible

- usually improved procedure required
  - for 100 Mbps and more
  - to transmit data in real time

IEEE 802.3z: Gigabit Ethernet

Desirable principle

- if 100% compatible
  - retain all procedures, formats, protocols
  - bit duration reduced from 100 ns over 10 ns to 1 ns
- but, then
  - maximum extension would also be
    - 1/100 of the 10 Mbit/s Ethernet,
  - i. e. (depending on the type of cable) approx. 30 m

IEEE 802.3z: Gigabit Ethernet

Principle for

point-to-point links

- full duplex mode
- interconnected by switch function
- with 1 Gbps in both directions
- no change of packet size

→ i.e. no need for further details

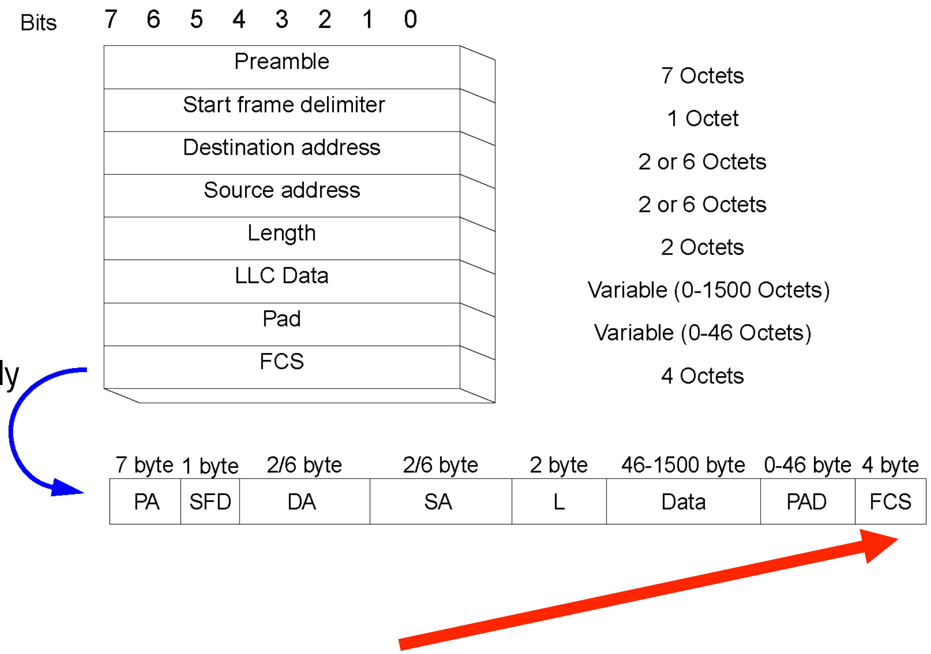
shared broadcast mode

- half duplex mode
- CSMA/CD
- interconnected by hub function
- tradeoff between distance and efficiency

→ i.e. see the following details

IEEE 802.3z: Gigabit Ethernet: Shared Broadcast Mode

- Principle:
  - maintain (as far as possible)
    - CSMA-CD with 64 byte minimum length
  - introducing two features
    - carrier extension
    - frame bursting



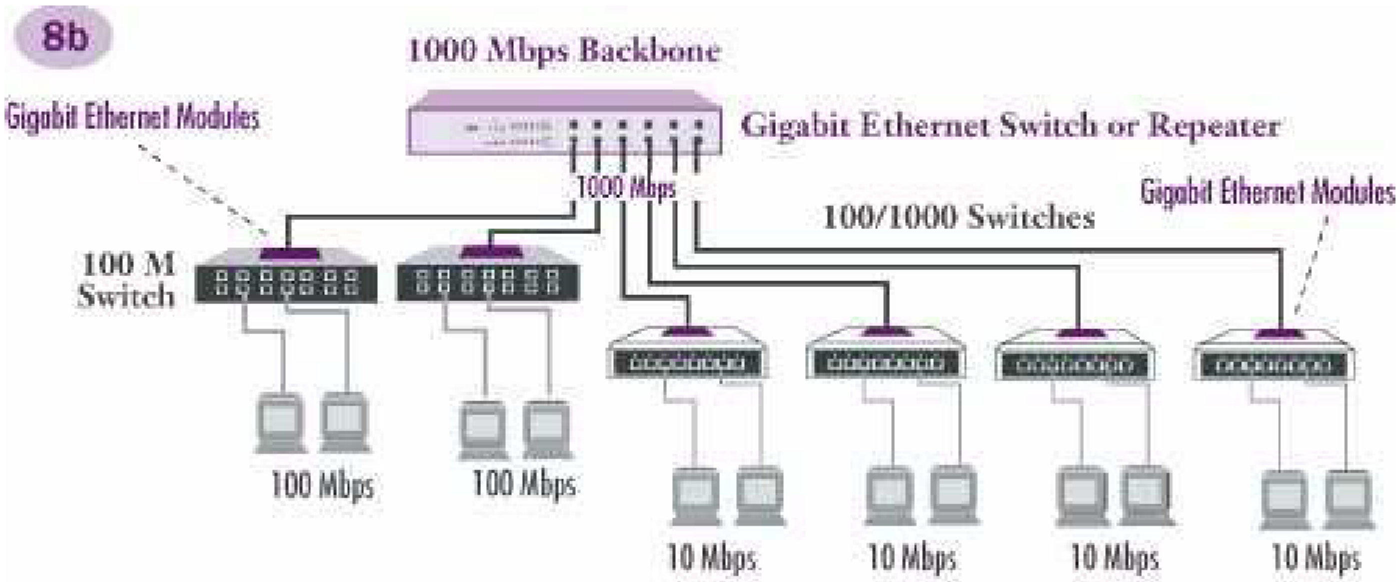
- Carrier extension
  - from 512 bit (64 byte) length, previously
  - to 512 byte length
  - i. e. by attaching a new extension field
    - following the FCS field (Frame Check Sum)
    - to achieve the length of 512 byte
  - Doing:
    - added by sending hardware and
    - removed by receiving hardware
    - software doesn't notice this
  - low efficiency
    - transmit 46 byte user data using 512 byte: 9%

- Frame bursting
  - allow sender to transmit CONCATENATED SEQUENCE OF MULTIPLE FRAMES in single transmission
    - needs frames waiting for transmission
    - better efficiency

IEEE 802.3z: Gigabit Ethernet: Shared Broadcast Mode

Maximum extension of a segment (i.e. of a Collision Domain)

- 5 UTP 100 m
- coax 25 m
- multimode fiber 550 m
- single mode fiber 5 km



IEEE 802.3ae: 10Gbit Ethernet

History

- 1999: IEEE 802.3ae task force founded
- 2002: approval as a standard

Objectives

- to preserve 802.3 frame format
  - incl. minimal and maximal frame sizes
- to support full duplex operation only
  → no CSMA/CD required

Type of media used

- works over optical fiber only, no UTP or coax

Supported distances:

- 850nm: 300 m
- 1310nm: 10 km
- 1550nm: 40 km

IEEE 802.3ba: 40Gb/s and 100Gb/s Ethernet

Requirements

- To support full-duplex operation only
- To preserve the 802.3 frame format utilizing the 802.3 MAC
- To preserve minimum and maximum FrameSize of current 802.3 standard
- To support a bit error ratio (BER) better than or equal to 10^{-12} at the MAC service interface