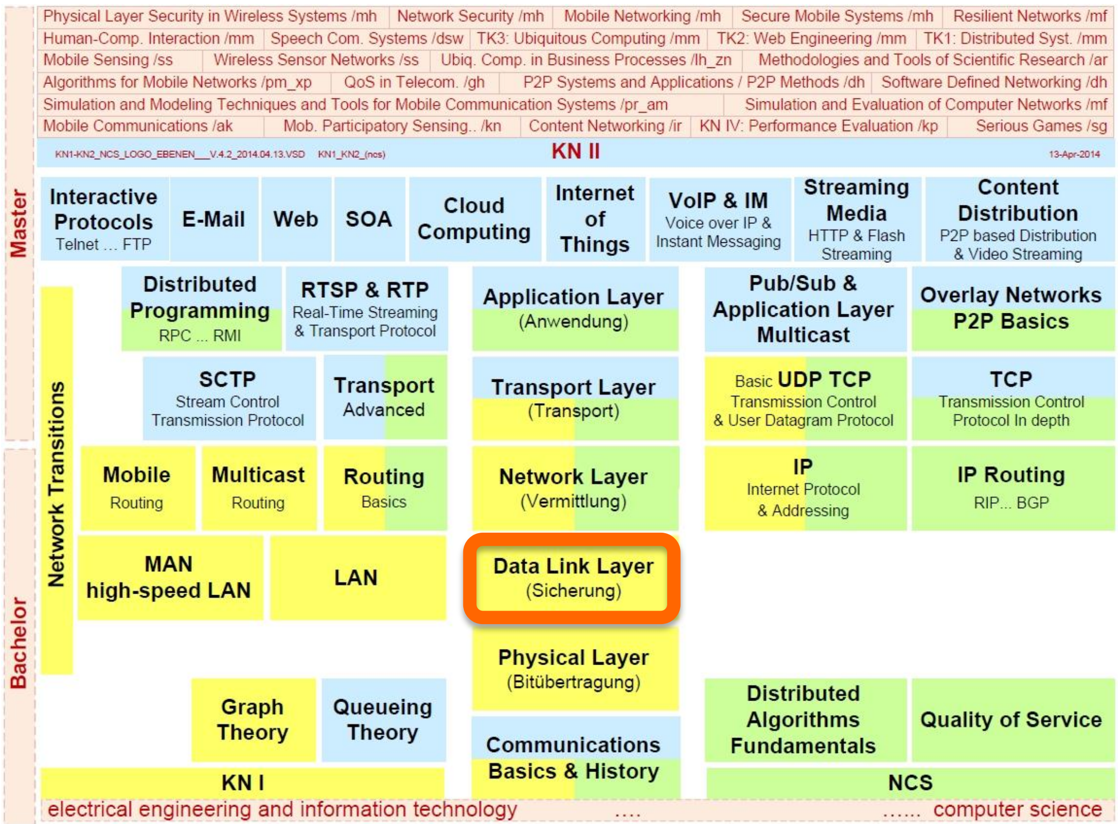
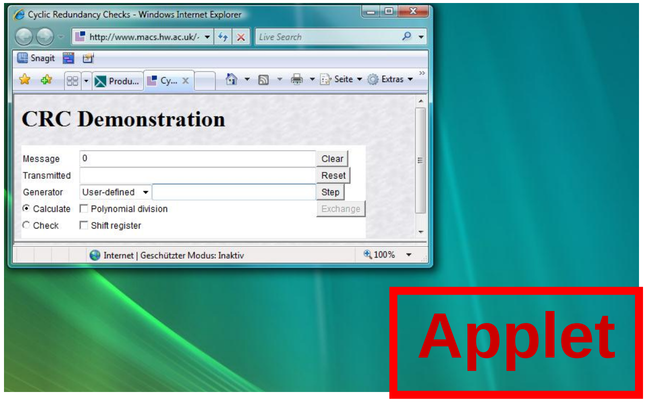
Communication Networks I

Data Link Layer



Prof. Dr.-Ing. **Ralf Steinmetz**
KOM - Multimedia Communications Lab

Overview

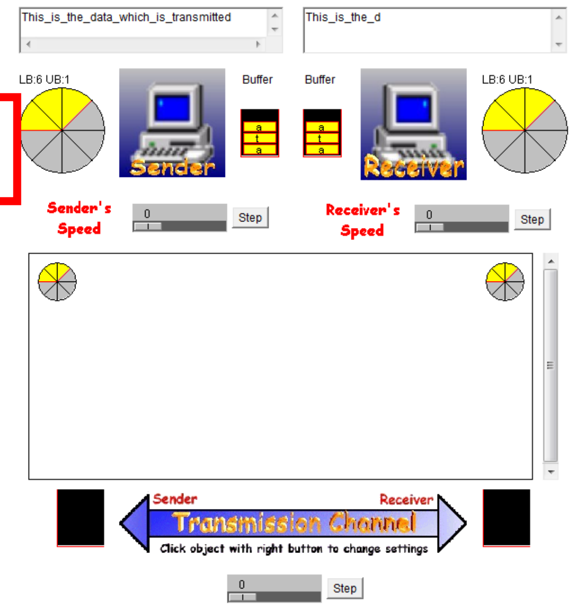


Cyclic Redundancy Checks - Windows Internet Explorer
CRC Demonstration
Message
Transmitted
Generator
User-defined
Clear
Reset
Step
Exchange
Calculate
Check
Polynomial division
Shift register
Applet

Overview

- 5 Sliding Window – Flow Control & Error Treatment
  - 5.1 Channel Utilization and Propagation Delay
  - 5.2 Sliding Window: Concept
  - 5.3 Sliding Window – Active Learning Object

Applet



- 6 Sliding Window: Remarks & Refinement
  - 6.1 Sliding Window: Influence of the Window Size
  - 6.2 Sliding Window: Piggybacking
  - 6.3 Sliding Window: Go-Back-N (Error Treatment)
  - 6.4 Sliding Window: Selective Repeat (Error Treatment)
  - 6.5 Channel Utilization
  - 6.6 Comparing Protocols

- 7 Protocols: HDLC Family
  - 7.1 HDLC: Principle
  - 7.2 Three Types of frames: (differ in control field)

- 8 Protocols: at Internet Layer 2
  - 8.1 Internet: Serial Line IP (SLIP)
  - 8.2 Internet: Point-To-Point Protocol (PPP)

- 9 Protocols: Perspective – L2 Communication Design Tasks

L1 Service

- Transmission of a bit stream (“unreliable bit pipe")
  - without sequence errors
- ’Malign’ features of the L1 service (& the communication channel)
  - Finite propagation speed
    - between sending and receiving operations at L2
  - Limited data rate

  → i. e. loss, insertion and changing of bits possible

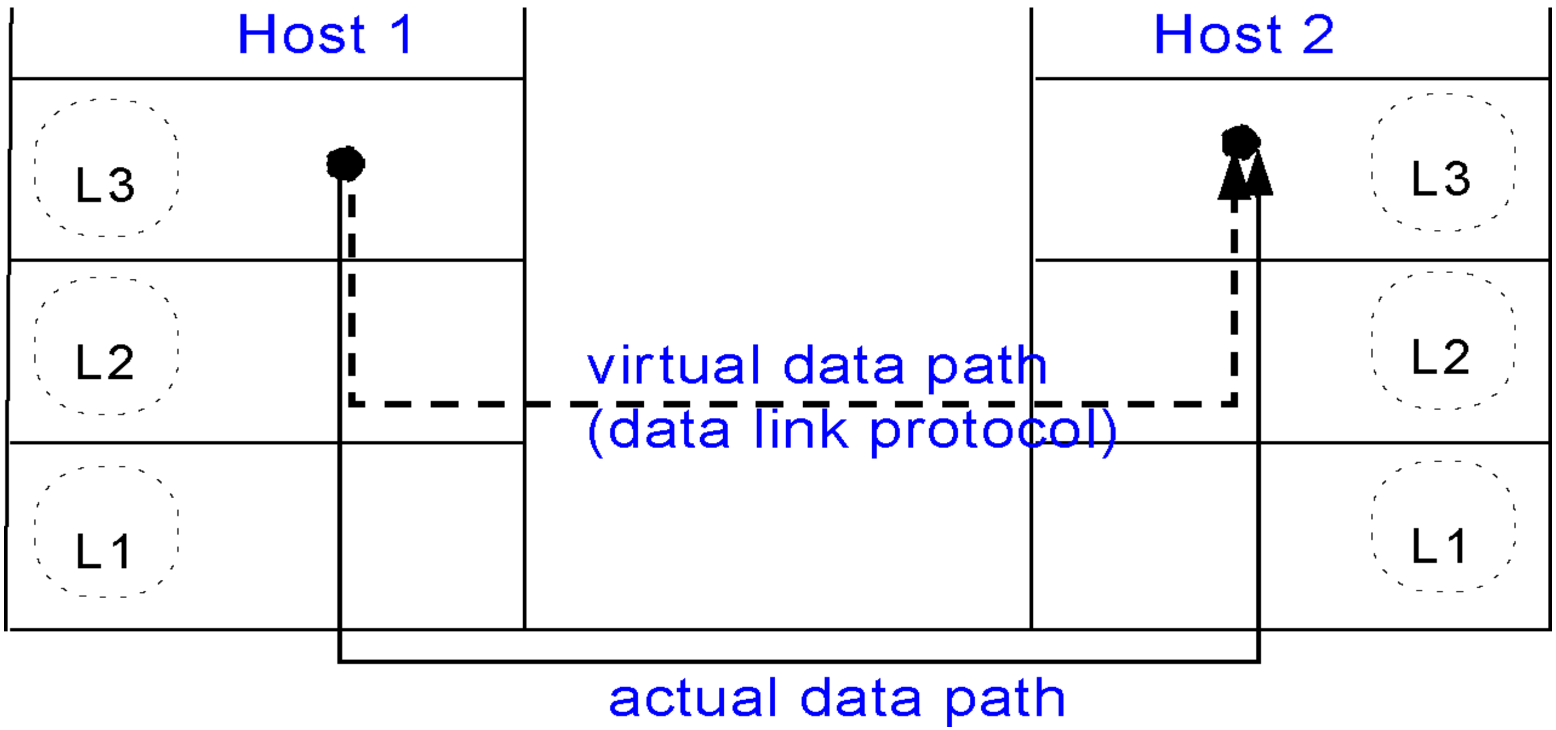
L2 Service

- (Reliable), efficient data transfer between ADJACENT stations
  - May be between more than 2 stations
  - Adjacent = connected by one physical channel
    - Wireless, coax, optical fiber, ...

L2 Functions

- Data transmission as FRAMES
- ERROR control and correction
- FLOW CONTROL of the frames
- Configuration management

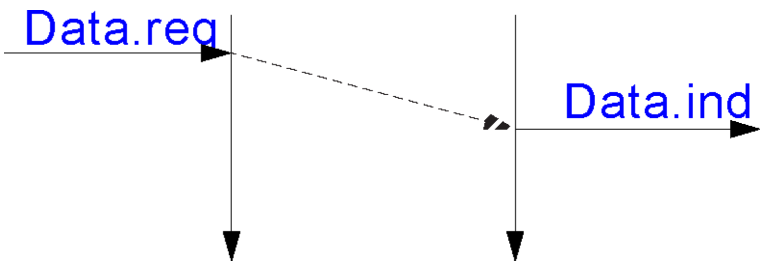
Actual data path and virtual data path



1.1 L2 Service Class “Unconfirmed Conn.less Service”

Transmission of isolated, independent units (Frames)

- Loss of data units possible
  - L2 does not try to correct this
  - L2 transmits only correct frames



Features

- No flow control
- No connect or disconnect

Applications

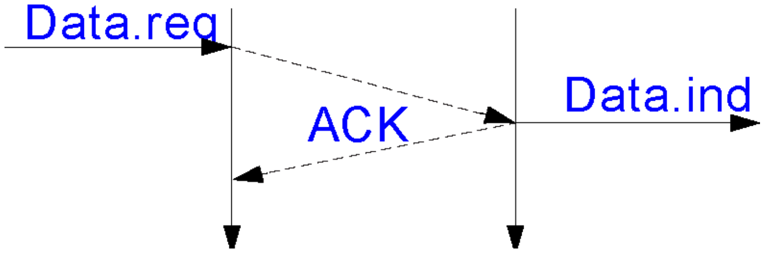
- On L1 communication channels with VERY LOW ERROR RATE
  - Corrections will possibly be done at a higher level then
- Possibly during real time data transfer like interactive voice communication
  - Timing errors probably more critical than errors in the voice data
- Often used in LANs

1.2 L2 Service Class "Confirmed Conn.less Service"

TECHNISCHE UNIVERSITÄT DARMSTADT

Receipt of data units (implicitly) acknowledged

- No loss (each single frame is acknowledged)
- Timeout and retransmit (if sender does not receive an acknowledgement within a certain time frame)



Features

- No flow control
- No connect or disconnect
- Duplicates and sequence errors may happen due to "retransmit"

Application

- L1 communication channel with high error rate e.g. mobile communication

Comments

- Acknowledging on L2 is only for optimization but is not indispensable, because this can also be done at a higher level (L4), however
  - L4 message usually consists out of n (e.g. 20) L2 frames
  - If there is an error in a frame

  ⇒the whole message will be retransmitted

  ⇒loss of time and efficiency

L2 Service Class “Connection-Oriented Service”

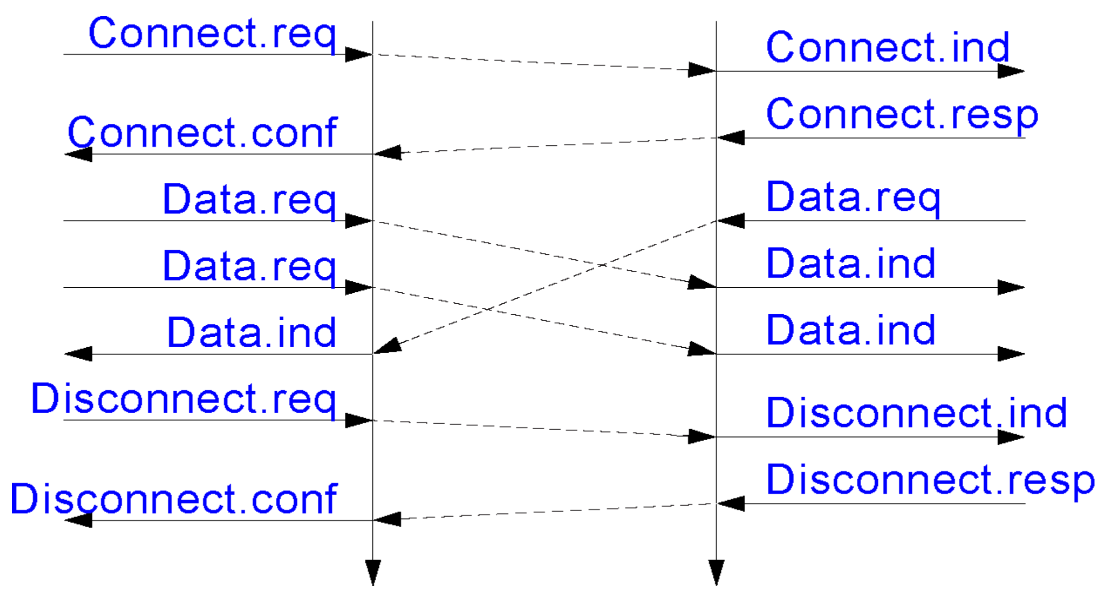
TECHNISCHE UNIVERSITÄT DARMSTADT

Connection over error free channel

- No loss, no duplication, no sequencing error
- Flow control

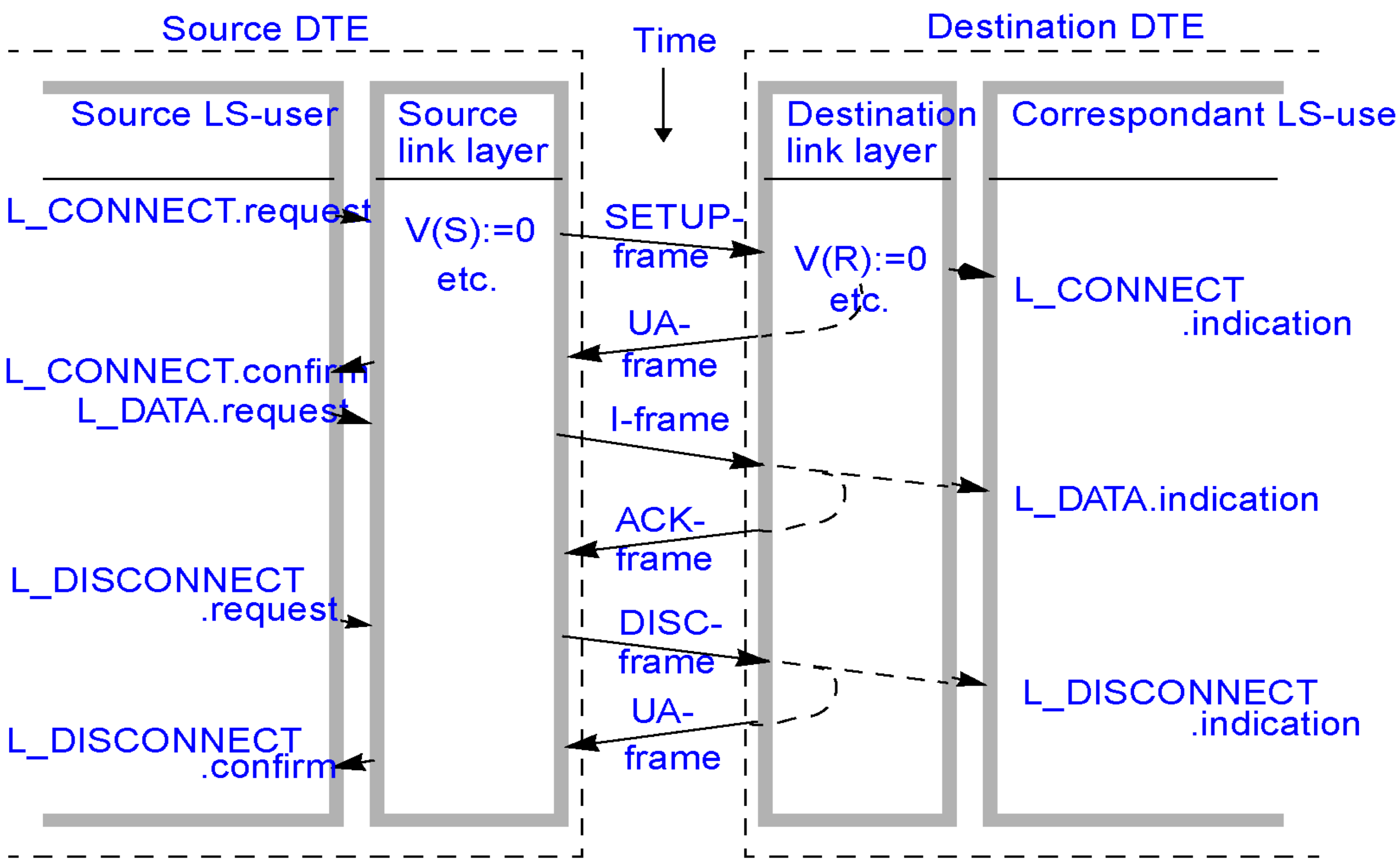
3-phased communication

1. Connection (initializing the counters/variables of the sender and receiver)
2. Data Transfer
3. Disconnection



1.3 Connection Management

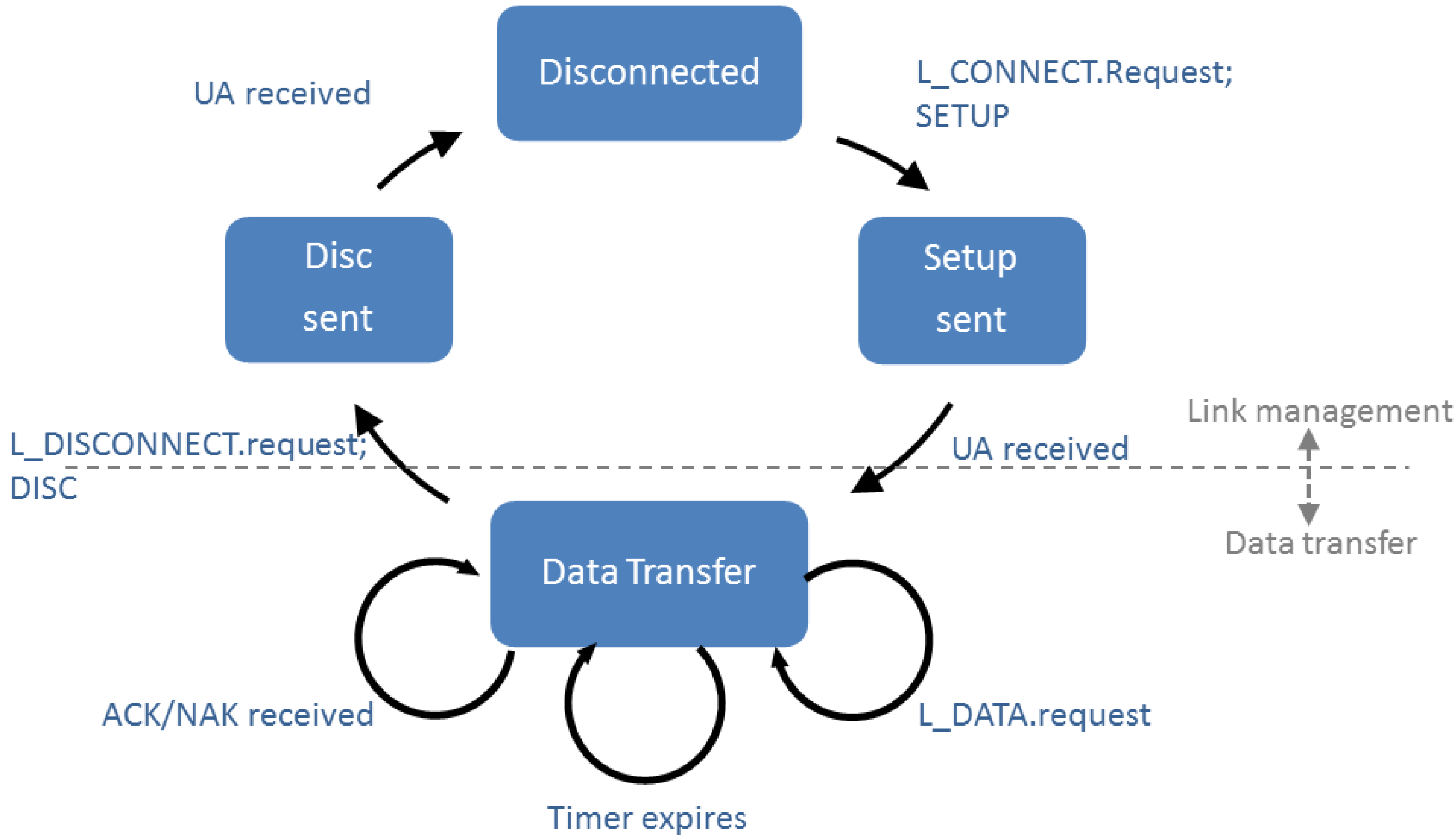
Presentation of the transmitted frame sequences, an example



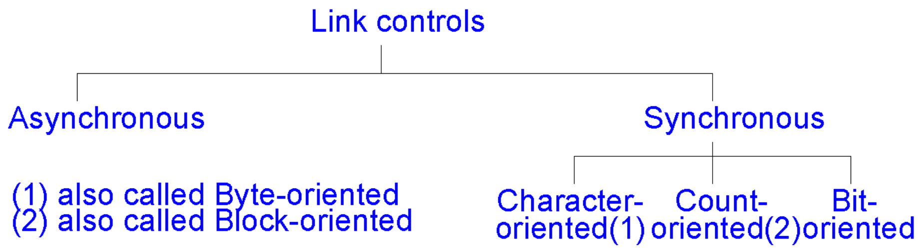
- UA: Unnumbered Acknowledgement

Connection Management

State presentation (as Finite State Machine)

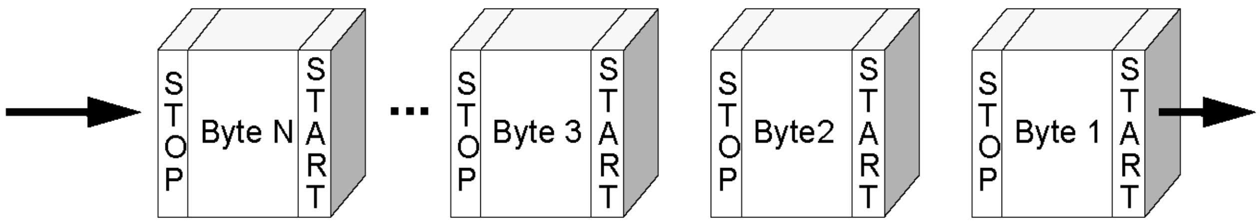


Operating Mode: Asynchronous and Synchronous



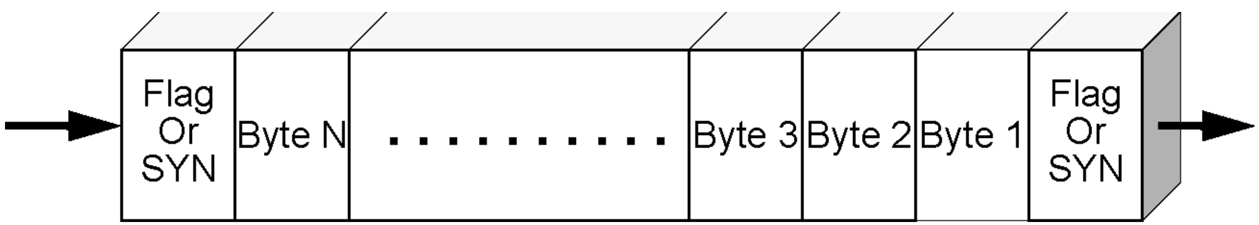
Asynchronous transmission

- Each character is bounded by a start bit and a stop bit
- Simple + inexpensive, but low transmission rates, often up to 200 bit/sec



Synchronous transmission (to be discussed in more detail in the following)

- Several characters pooled to frames
- Frames defined by SYN or flag
- More complex, but higher transmission rates



Synchronous Data Transmission: Framing

L2 forms a frame from the L1-bits

- Which (as a unit) undergoes error correction

Possibilities for bounding frames

- Bound frames by idle times
  - Critical challenges
    - Networks (L1) usually cannot relate to the notion of time
    - Possibly loss of efficiency
- Character-oriented
- Count-oriented
- Bit-oriented
- Using invalid characters of the physical layer

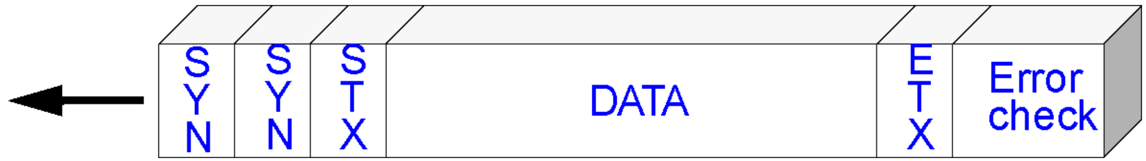
Comment

- Combinations may be used in L2, e.g.
  - Count-oriented and Bit-oriented
  - The transmission is flawless/OK only if both match

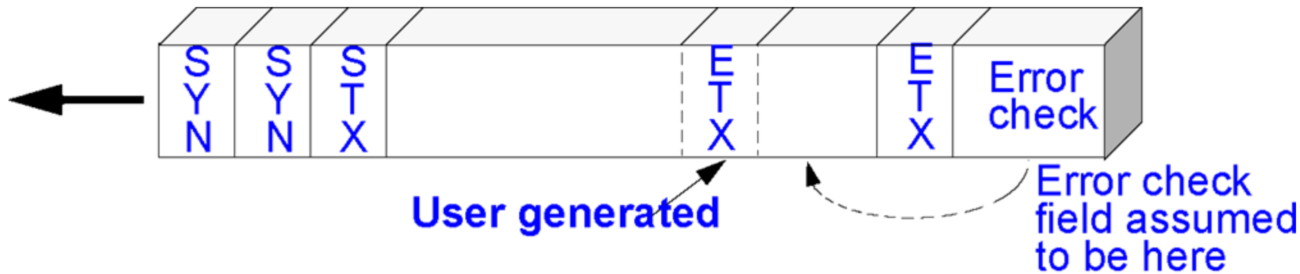
2.1 Character Oriented Protocols

Control Fields

- Flag frame areas
- depend on encoding (e.g. ASCII)

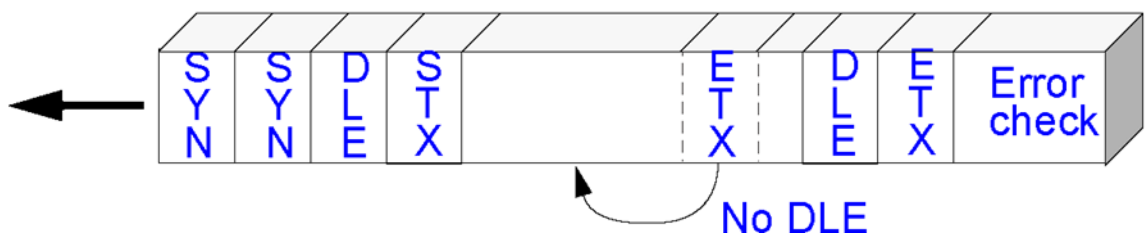


Problem: user data may contain “control characters”



Solution: CHARACTERSTUFFING

- SENDER:
  each control character is preceded by a DLE (Data Link Escape), (but not in user generated data)
- RECEIVER:
  only control characters preceded by DLEs are interpreted as such



Character Oriented Protocols

Problem

- User generated data contains DLE

Solution

- The sender inserts an additional DLE before the DLE in the user's data
- The receiver ignores the first of two back-to-back DLEs

(a)		STX	A		DLE	B		ETX
(b)	DLE	STX	A	DLE	DLE	B	DLE	ETX
(c)		STX	A		DLE	B		ETX

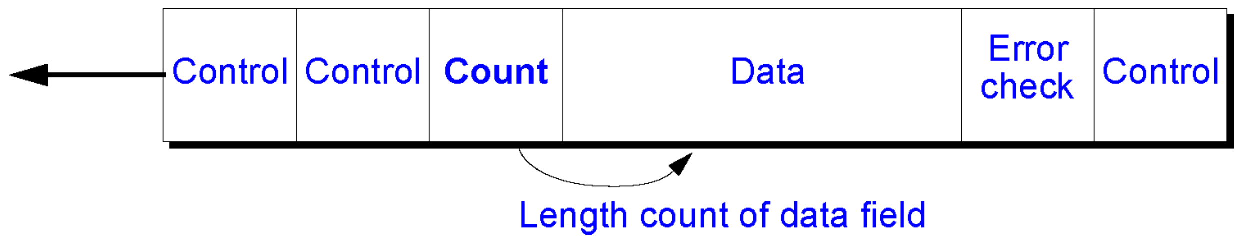
Stuffed DLE

Disadvantages

- DLE insertion requires additional effort/time
- usually a derivation of an ASCII 8 bit encoding used
  - i.e. conversion required (if codes are different)

2.2 Count Oriented Protocol

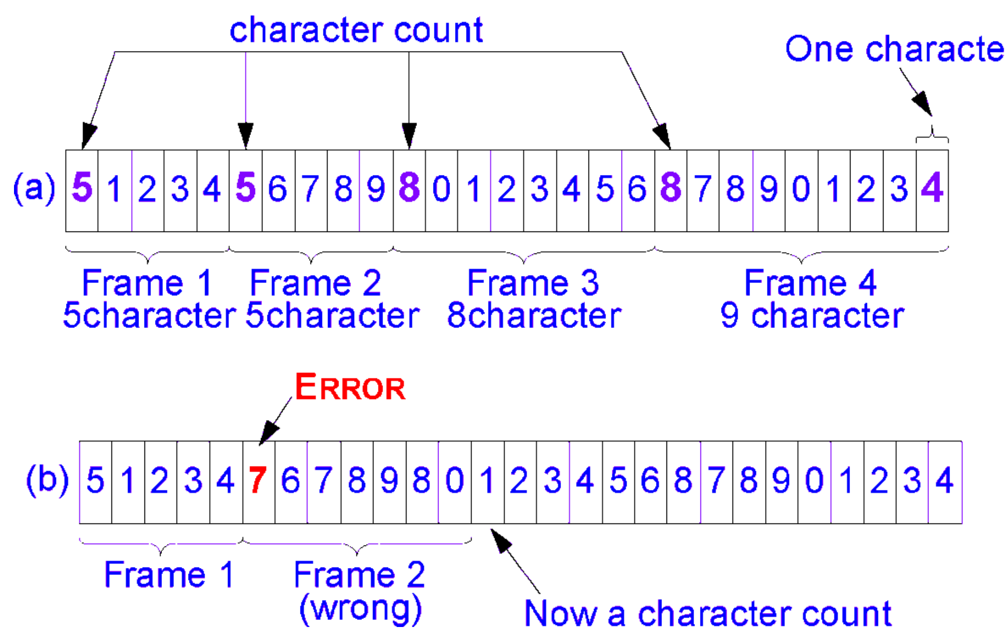
Frame contains LENGTH COUNT FIELD



Problem: Transmission error destroys length count

- Sender and receiver are not synchronized anymore
- That means
  - Where does the next frame start?
  - Where do retransmitted frames start?

→ **Therefore not widely spread!**

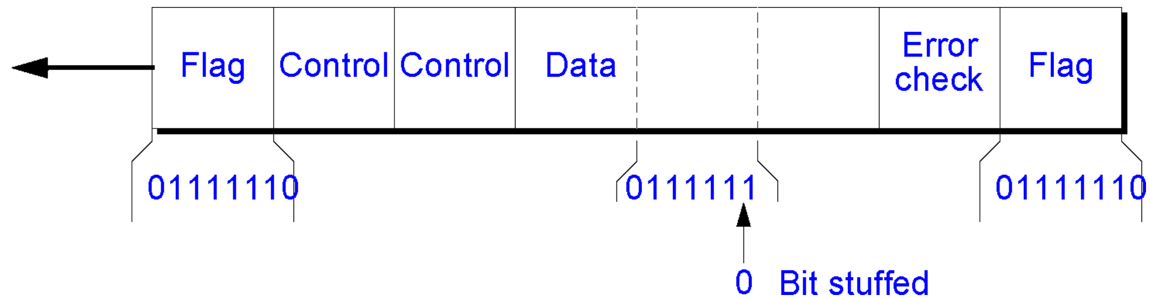


2.3 Bit Oriented Protocols

TECHNISCHE UNIVERSITÄT DARMSTADT

In use at most of today's protocols

- Independent from encoding
- Block definition



Problem: " Flag " in user Data

Solution: Bit Stuffing

- Sender: Inserts a "0" bit after 5 successive "1" (only in the user data stream)
- Receiver: Suppresses "0" after 5 successive "1"

(a) 0 1 1 0 1 1 1 1 1   1 1 1 1 1   1 1 1 1 1   1 0 0 1 0

(b) 0 1 1 0 1 1 1 1 1 **0** 1 1 1 1 1 **0** 1 1 1 1 1 **0** 1 0 0 1 0

Stuffed Bits

(c) 0 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 0

2.4 Protocol with Invalid Characters

Invalid

- With regard to the layer in consideration: in this case the physical layer

Method

- L1 defines digital encoding

Example

- Return-to-Zero (RZ)
  - 1: clock pulse (double frequency) during the interval
  - 0: low level
- There is always a combination of “high-low” or a sequence of “low”
- There is never a “high-high” combination (invalid symbol)
  - i. e. define an invalid symbol in L2 as the bit boundary

Comment

- Effective
- But, actually inconsistent with the layer model

BIT ERROR: Modification of single bits

BURST ERROR: Modification of a sequence of bits

Causes for errors

- Thermic noise: electron movement generates background noise
- Impulse disruptions (often last for 10 msec):
  - Cause: glitches in electric lines, thunderstorms, switching arcs in relays, etc.
  - Most common cause for errors
- Crosstalk in adjacent wires
- Echo
- Signal distortion (dampening is dependent on frequency)

→ errors usually occur in bundles: BURST ERROR

Error detection

- Inserting redundancies so that receiver is able to detect an error
- Retransmission in case of an error

Error correction

- Inserting redundancies so that receiver is able to detect and correct an error

Basics: Code Word, Hamming Distance

Frame (= code word) contains
- Data
- Checking information (Error correction / error detection)

Code = set of all valid code words

Hamming distance of two words w1 and w2
- Number of bit positions by which w1 and w2 differ
- Example:

```
        w1  10001001
XOR     w2  10110001
        =   00111000   → d = 3
```

Hamming distance of a code
- Minimum Hamming distance between two words of a code
- Example:

```
w1  10001001
w2  10110001
w3  10110011   → d = 1
```

Hamming distance between w1/w2 = 3; w2/w3 = 1; w1/w3 = 4;
→ the Hamming distance of the code is d = 1

How to calculate the Hamming distance

- Number of digits may be selected freely
- Hamming distance is calculated step by step

Applet

Quiz included

- To check whether or not topic is understand correctly

http://www.frontiernet.net/~prof_tcarr/HammingDistance/applet.html#APPLET

3.2 Error Detection (according to Hamming)

Hamming Distance determines

- A code's error detection and correction properties

Hamming-Distance Applet

Error Detection (according to Hamming)

DETECTION of *f 1*-bit errors

- If the Hamming distance of code *d*

$$d \geq f + 1$$

- *f* and less errors generate an invalid code word and are detected
- More than *f* errors possibly generate a valid but wrong code word
- Example:

```
        parity bit
        p
0   0   0
0   1   1
1   0   1
1   1   0
    d = 2:
    i.e. maximum value for f: f=1
    detection of a 1-bit error
```

Error Detection (according to Hamming)

1- dimensional

- Allows detection of 1-bit errors

2-dimensional

- Allows detection of 1-, 2- and 3-bit errors
- (example from page 92 "Peterson Davie", German book)

example 1-dimensional

		parity bit p
0	0	0
0	1	1

example 2-dimensional

parity bits last column and last row

0	0	1	1	0	0
1	1	1	1	1	1
1	0	0	0	0	1
1	1	0	0	0	0
0	0	0	0	1	1
1	0	0	0	0	1

Error Correction (according to Hamming)

CORRECTION of f 1-bit errors:

- If the Hamming distance of code *d*

  d ≥ ...*f + ... ??

- THEN *f* and less errors transcribe word *w* into an invalid word, which is "closer" to *w* than to any other word

- Example: *d* = 5 → *f* = 0,1 or 2

```
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 1 1 1 1 1
1 1 1 1 1 0 0 0 0 0
1 1 1 1 1 1 1 1 1 1
```

- Correction of two 1-bit errors (*f* = 2) in the following word:

```
0 0 0 0 0 0 0 1 1 1 →
```

- Result: the NEXT POSSIBLE WORD the following is ..

```
0 0 0 0 0 1 1 1 1 1 →
```

CORRECTION of *f* 1-bit errors:

- Example: d=4 → f = 2 ??

- Correction of two 1-bit errors (f = 2) in the following word:

```
0 0 0 0 0 0 0 0
0 0 0 0 1 1 1 1
1 1 1 1 0 0 0 0
1 1 1 1 1 1 1 1

0 0 0 0 0 0 1 1
```

- The next possible words the following are ..

```
0 0 0 0 0 0 0 0    or!!
0 0 0 0 1 1 1 1
```

→ No correction possible !

Error Correction (according to Hamming)

CORRECTION of *f* 1-bit errors:

- If Hamming distance of code is given by d

$$d \geq 2f + 1$$

Lower bound for the number of check-bits for correcting

1-bit errors: $(m + r + 1) \leq 2^r$

(m is # data bits; r is # check-bits)

e. g.

m = 8 → r = 4

m = 1000 → r = 10

Procedure:

- According to Hamming, 1950
- According to Trellis

Properties:

- Only possibility for simplex operation
- But high redundancy in each block

→ **usually less efficient than error detection**

3.4 Further Error Detection

CYCLIC REDUNDANCY CODE (CRC) is one of the error detection procedures see e.g. http://de.wikipedia.org/wiki/Cyclic_Redundancy_Check

Basic idea:

- Bit strings are treated as polynomials

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

  whereas $k_i = [0,1]$

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

Polynomial arithmetic: modulo 2 ("algebraic field theory")

```
Sender                                        Receiver
            /*sends block B*/
        B(x)/G(x) = Q(x)+ R(x);
                (B, R)
send --------------------------------------> receive;
                                              B(x)-R(x))/G(x) =Q(x)+R'(x)
                                              if R'(x) = 0
                                              then Accept B
                                              else Reject B
```

CRC + CRC-Applet

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 correlate 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
                  1110
```

Transfered frame: 11010110111110

TECHNISCHE UNIVERSITÄT DARMSTADT

Standardized polynomials:

$$\text{CRC} - 12 = x^{12} + x^{11} + x^{3} + x^{2} + x^{1} + 1$$

$$\text{CRC} - 16 = x^{16} + x^{15} + x^{2} + 1$$

$$\text{CRC} - \text{CCITT} = x^{16} + x^{12} + x^{5} + 1$$

CRC - 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

TECHNISCHE UNIVERSITÄT DARMSTADT

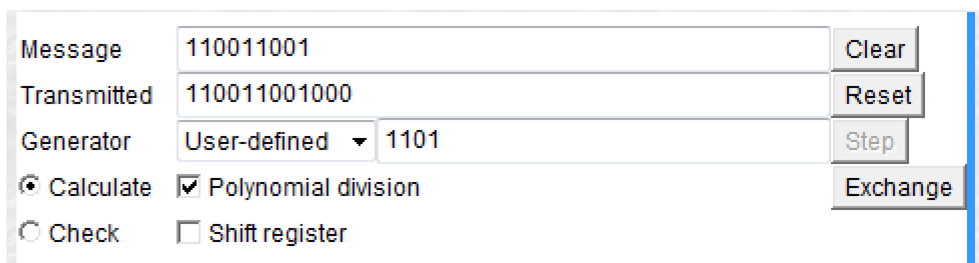
Applet

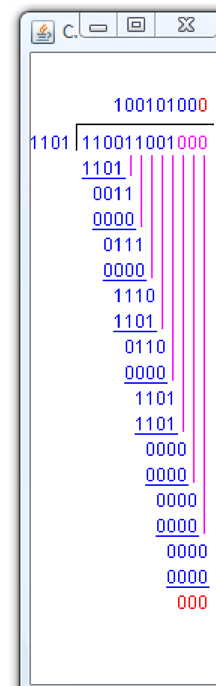
Calculates the CRC

- Message may be freely chosen
- Generator may be freely chosen
  - Or a predefined generator may be used

Stepwise polynomial division

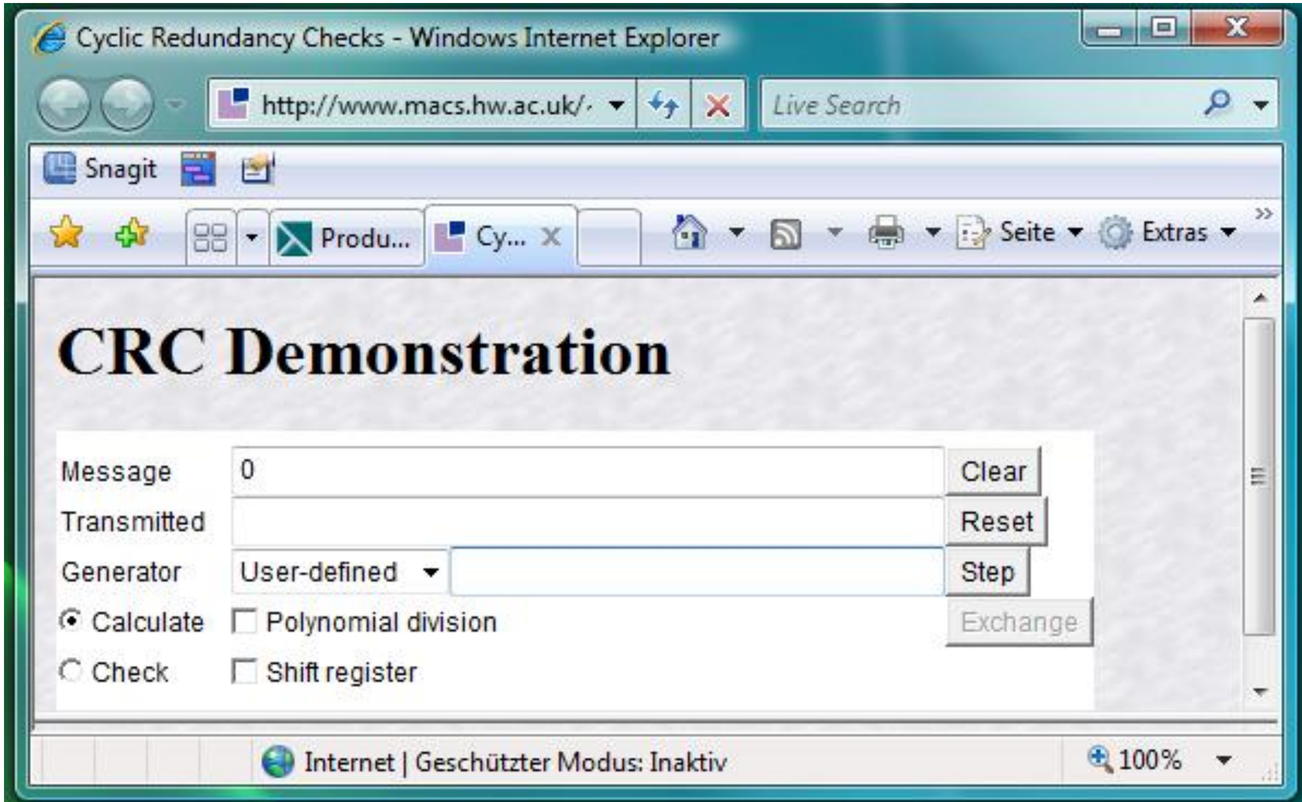
- Opens an extra window
  - Java based





http://www.macs.hw.ac.uk/~pjbk/nets/crc/

Cyclic Redundancy Check – Active Learning Object



Applet

Basics, problems statement:

- sender can send faster than receiver can receive

WITHOUT FLOW CONTROL:

- sender can send faster than receiver can receive
- that means that the receiver loses frames despite error-free transmission

WITH FLOW CONTROL:

- sender can adapt to receiver's abilities by feedback

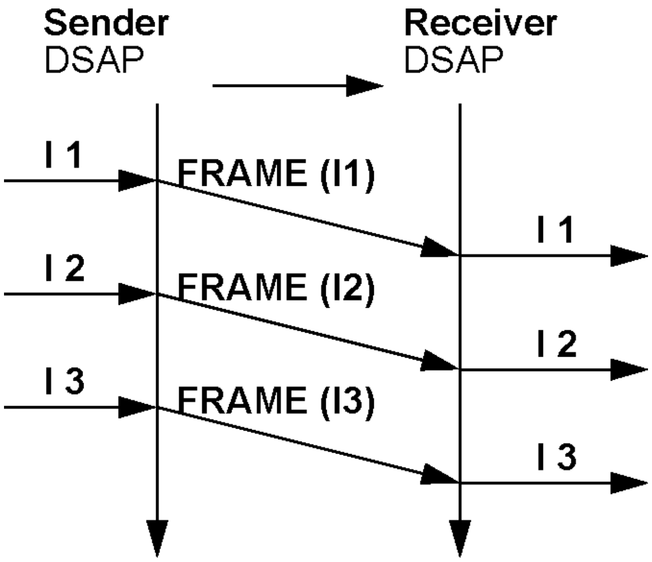
Comment:

- error control and flow control are usually interlinked
- rate control (as opposed to flow control)
  - reference to frame sequencing (not single frames)
  - used with continuous data (audio, video)

4.1 Protocol 1: Utopia

Assumptions:

- error-free communication channel
- receiving buffer infinitely large
- receiving process infinitely fast



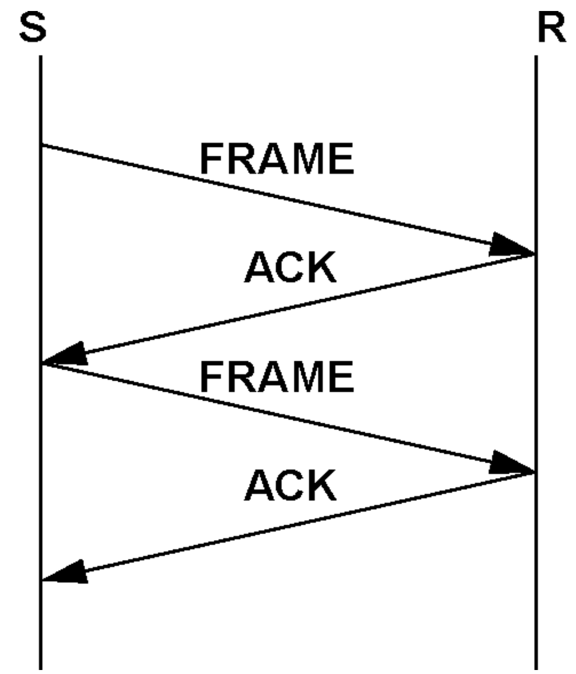
DSAP: Data link (layer) Service Access Point

- but: finite buffer, finite processor output

TECHNISCHE UNIVERSITÄT DARMSTADT

Assumptions:

- error-free communication channel
- NOT [infinitely large receiving buffer]
- NOT [receiving process infinitely fast]
  - but always fast enough for processing one (1) frame



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)

  but: additionally, faulty communication channel (loss of frames)...

4.3 Protocol 3a: Stop-and-Wait / PAR

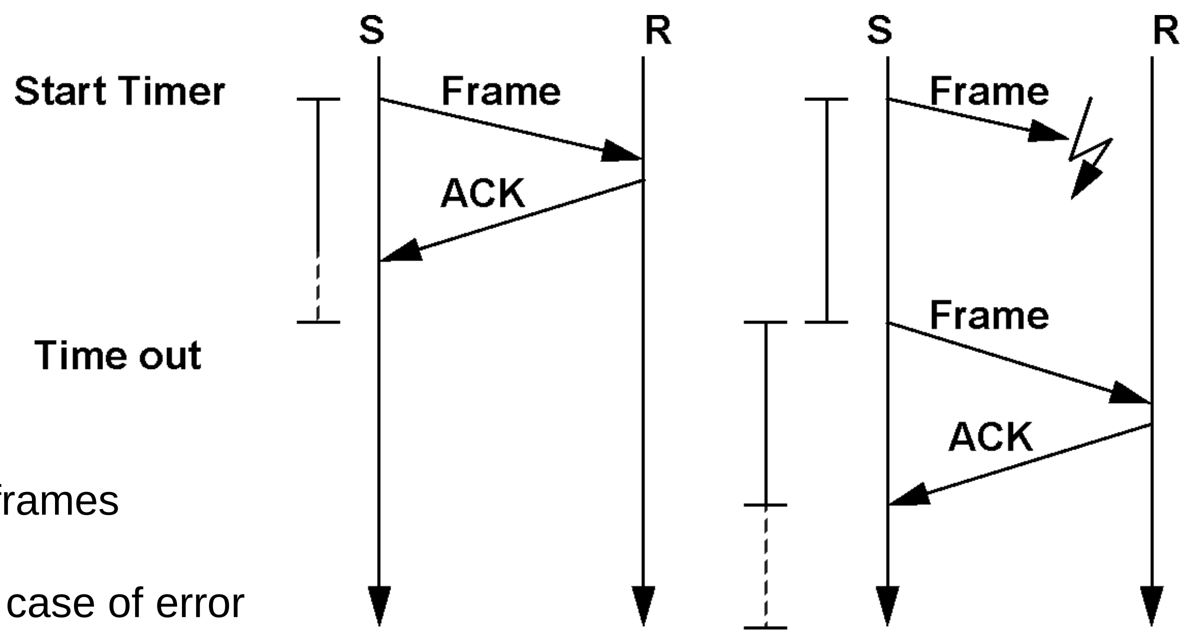
Assumptions:

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

Problem: protocol blocks at loss of both frames and ACKs

Solution:

- PAR (Positive-Acknowledgement with Retransmit)
- also called ARQ (Automatic Repeat reQuest)



Timeout interval:

- TOO SHORT:
  - unnecessary sending of frames
- TOO LONG:
  - unnecessary long wait in case of error

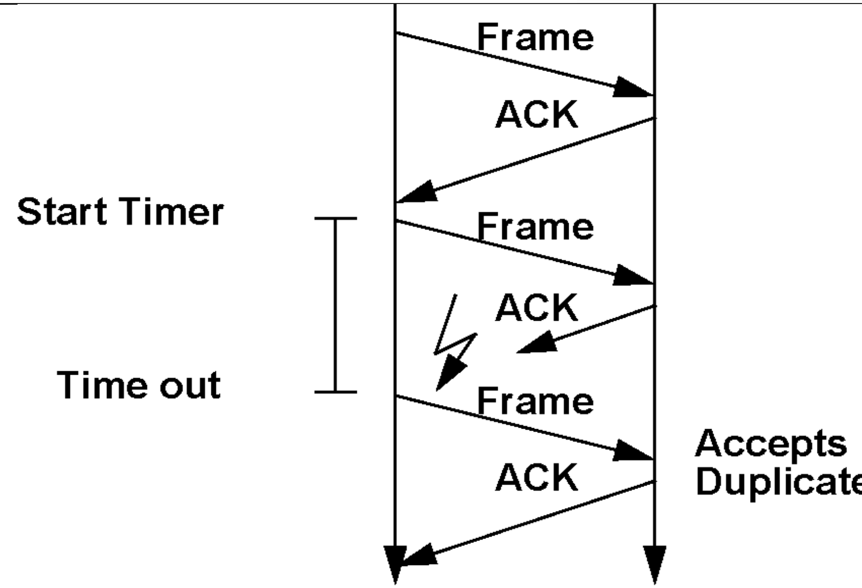
  but: in addition if ACK is lost

  ...

4.4 Protocol 3b: Stop-and-Wait / PAR / SeqNo

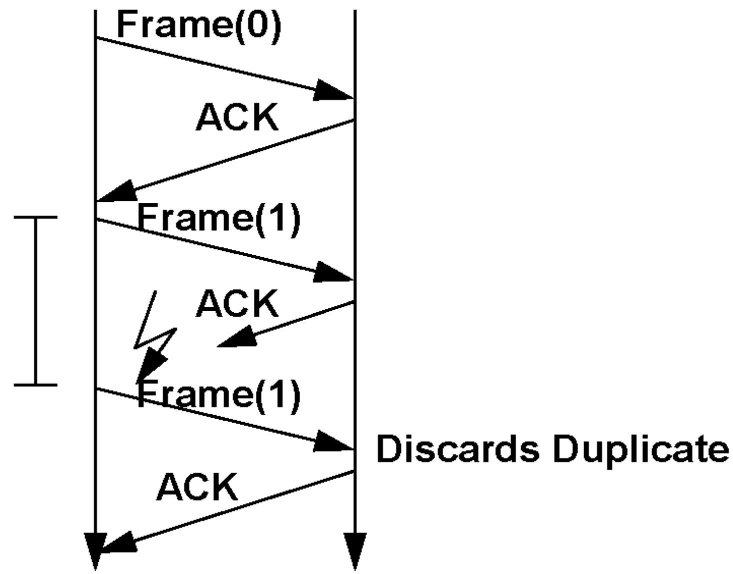
Problem:

- loss of ACKs may lead to duplicates



Solution: sequence numbers

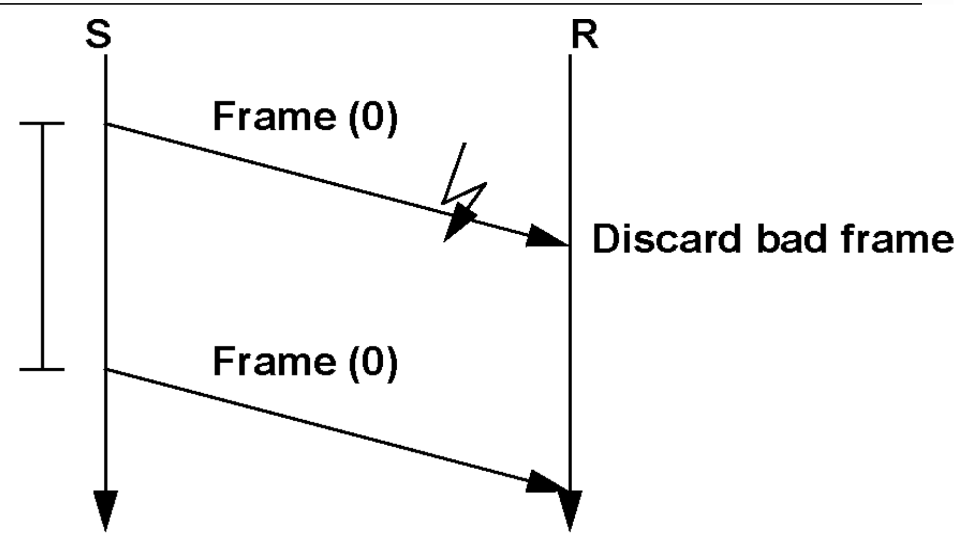
- each block receives a sequence no.
- sequence no. is kept during retransmissions
- range
  - in general: [0, …, k], k=2n-1
  - Stop-and-Wait: 0,1



4.5 Protocol 3c: Stop-and-Wait / NAC+ACK / SeqNo

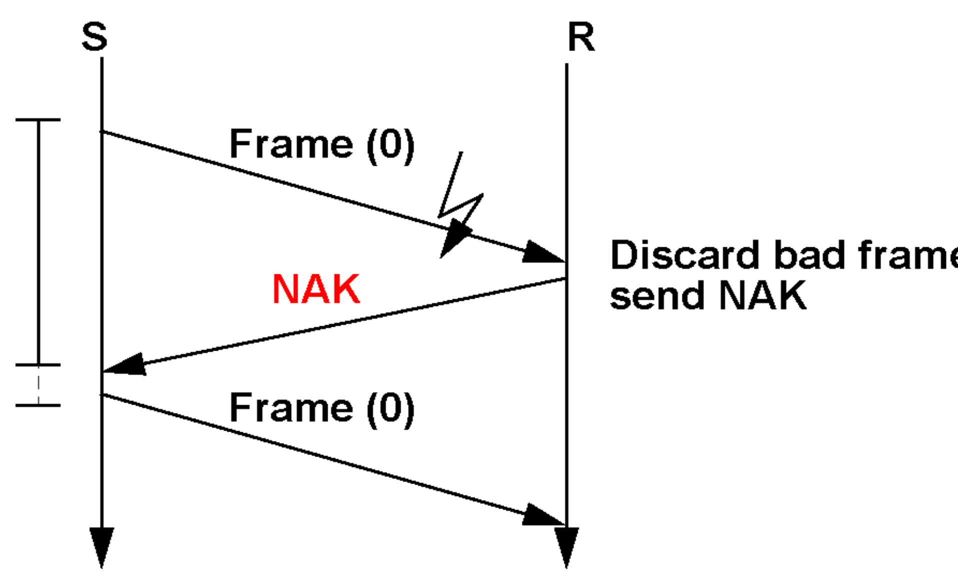
Until now passive error control

- no differentiation between
  - missing and
  - faulty frames
- 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

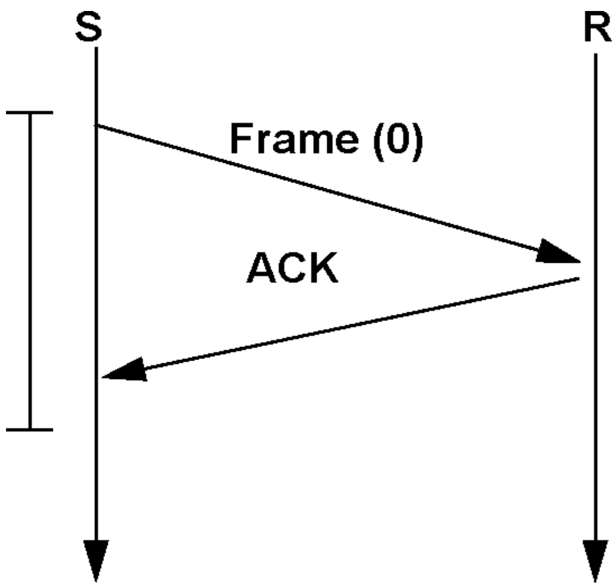


Protocol 3c: Stop-and-Wait / NAC+ACK / SeqNo

1. Situation: OK

- Frame correctly transmitted

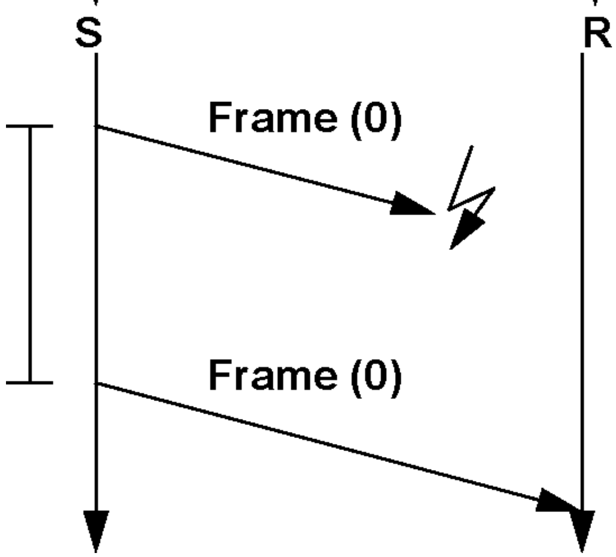
→ ACK sent



2. Situation:
Break at path "sender → receiver"

- frame did not arrive

→ timer issues retransmit



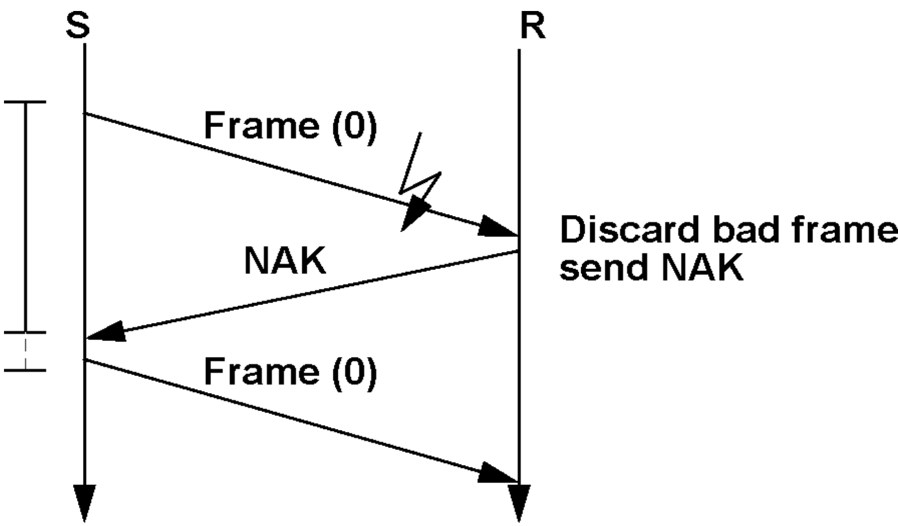
Protocol 3c: Stop-and-Wait / NAC+ACK / SeqNo

3. Situation:

Break at path "sender → receiver"

- faulty frame arrives

→ NAK issued



4. Situation:

Break at path "receiver → sender"

- NAK issued

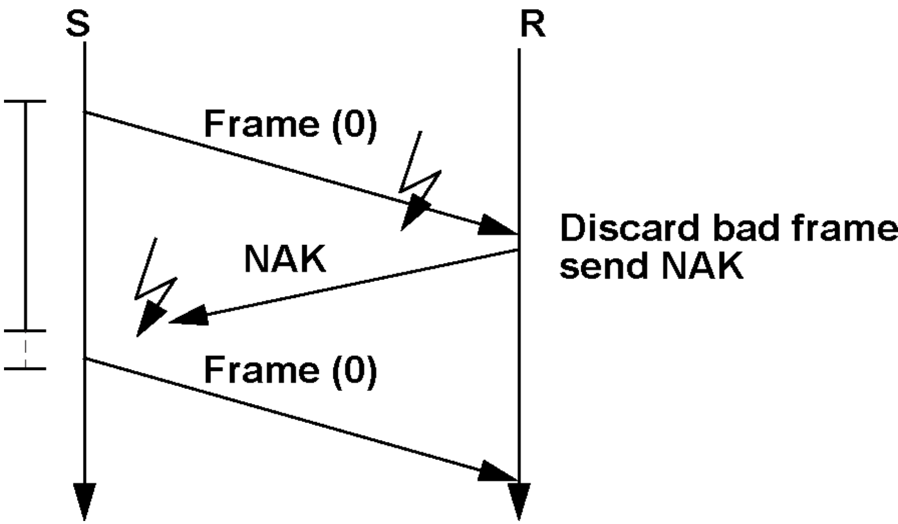
but,

- NAK does not arrive

or

- NAK arrives damaged

→ timer issues retransmit



4.6 Example of Matching Frame Formats, Seq.No.

Example: using a character oriented protocol

SOH	
N(S)	SEQ.NO. Field
STX	
…	Frame contents = printable characters
…	
…	
…	
ETX	
BSC	Error check field (block sum check)

I-frame format (information, data frame)

SOH/STX
ACK
N(R)
BSC

ACK-frame format

SOH/STX
NAK
N(R)
BSC

NAK-frame format

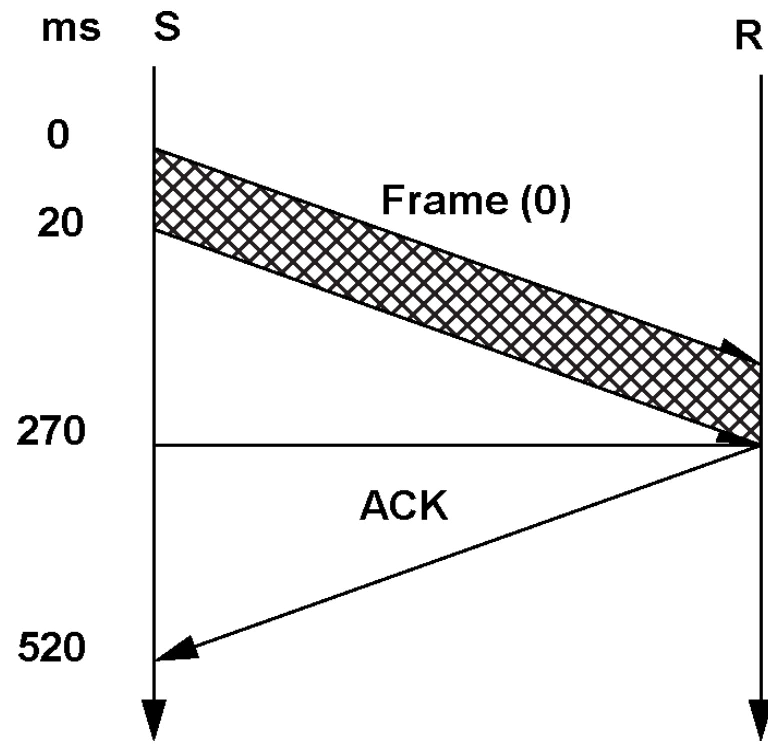
Channel Utilization and Propagation Delay

Sliding Window: Concept

5.1 Channel Utilization and Propagation Delay

Stop-and-Wait:

- non-defined state (parallelism) with simultaneously
  - lost frames, modified frames and
  - premature time-out
- poor utilization of the channel



Example: satellite channel

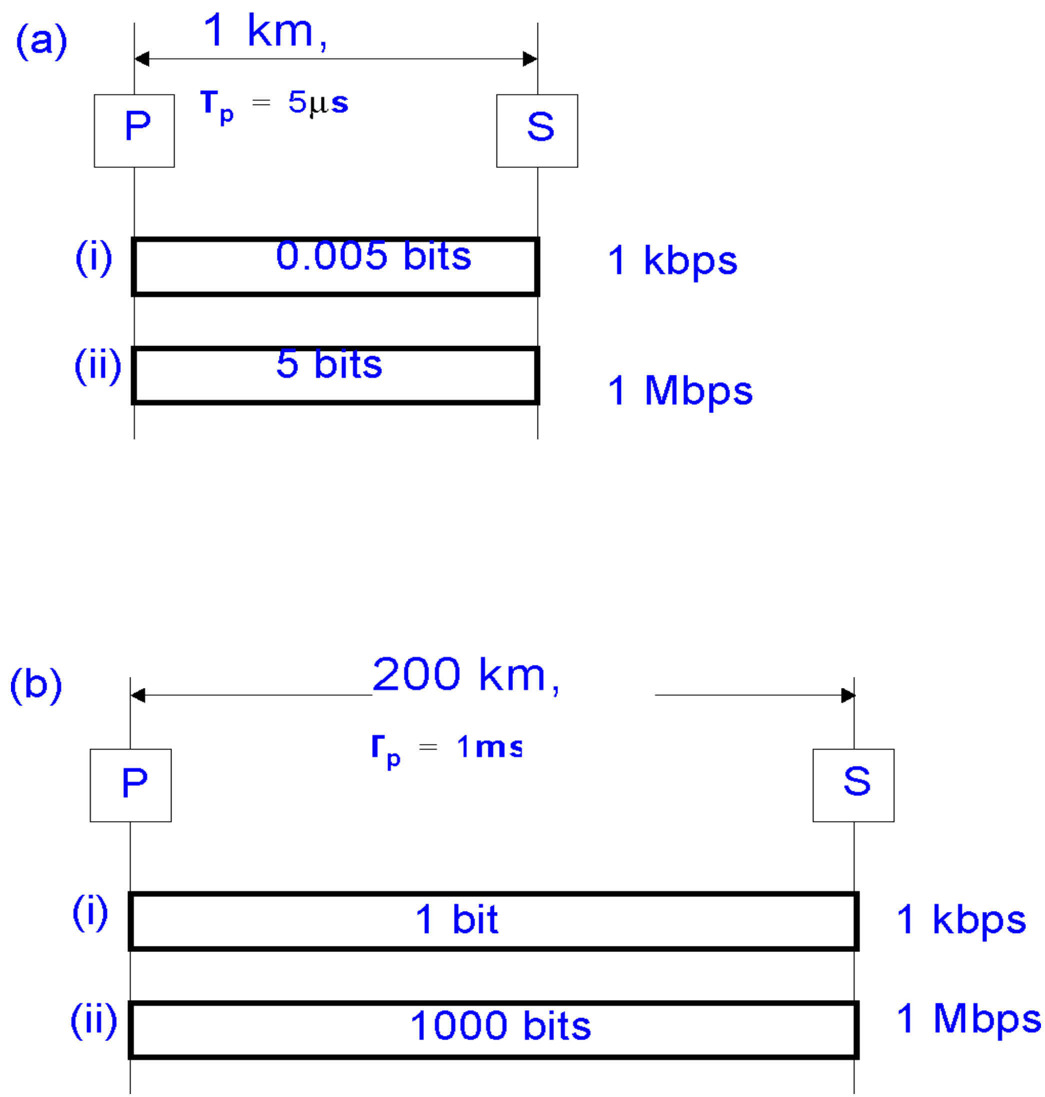
- transmission rate: 50 kbps
- roundtrip delay 500 ms (2*250 ms)
- frame size: 1000 bit
- in comparison
  → ACK is short and negligible

this means

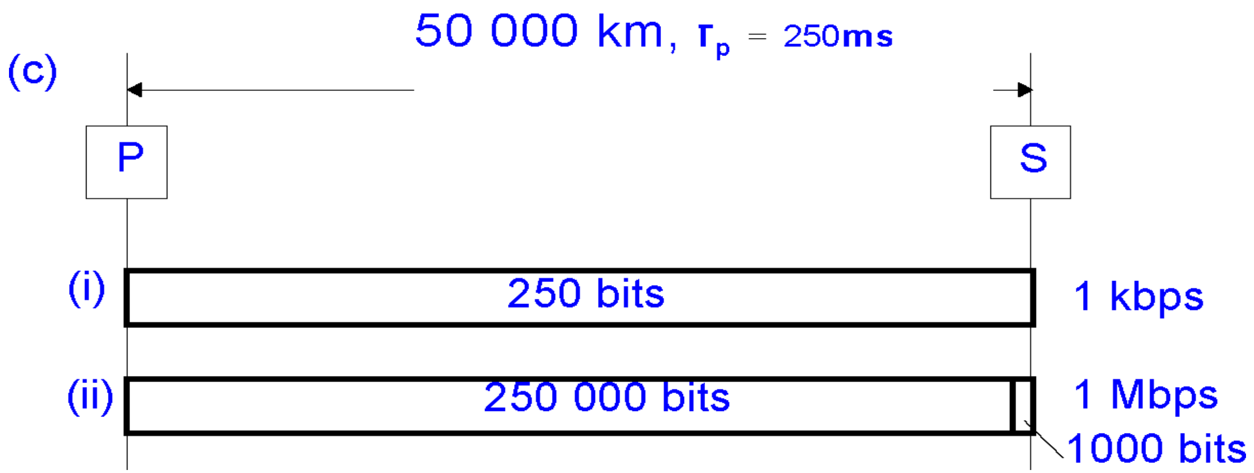
- sending takes 1000 bit / 50.000 bps = 20 ms
- sender is blocked for 500 ms of 520 ms

→ Channel utilization < 4%

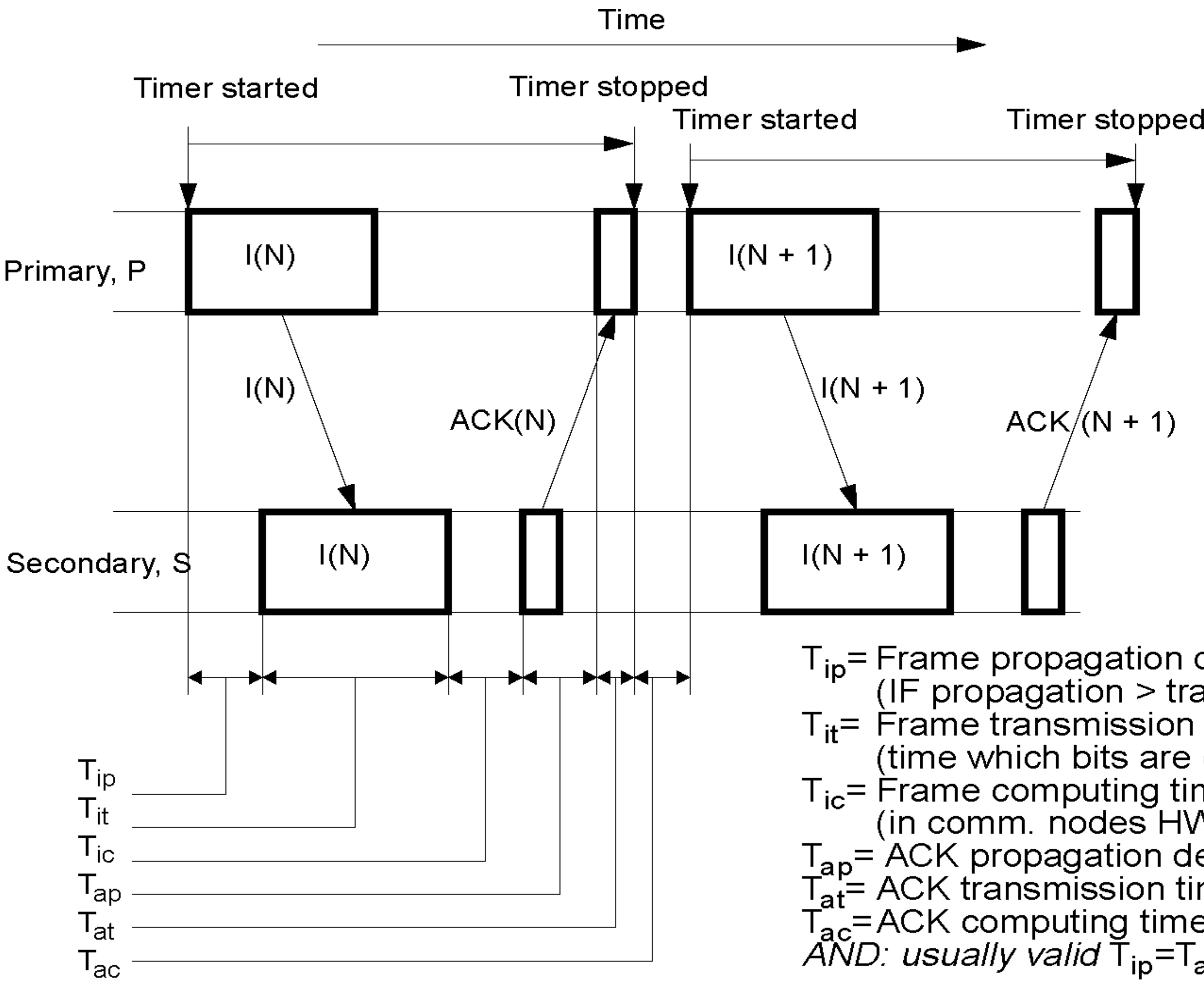
Channel Utilization and Propagation Delay



Channel Utilization and Propagation Delay



Channel Utilization and Propagation Delay



T_{ip}= Frame propagation delay (**P** → **S**)
(IF propagation > transmission)
T_{it}= Frame transmission time (**P** → **S**)
(time which bits are on channel)
T_{ic}= Frame computing time in *S*
(in comm. nodes HW & SW)
T_{ap}= ACK propagation delay(**S** → **P**)
T_{at}= ACK transmission time(**S** → **P**)
T_{ac}=ACK computing time in *P*
AND: usually valid $T_{ip}=T_{ap} = T_p$

Channel Utilization and Propagation Delay

exact formula (note: some values based on assumptions):

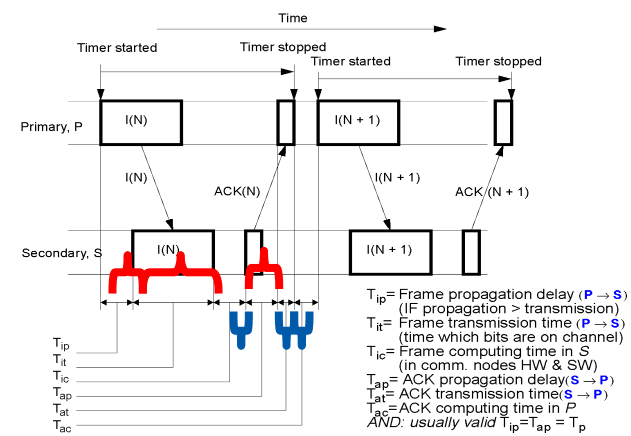
$$U = \frac{T_{it}}{\sum T_{information + acknowledgement}} = \frac{T_{it}}{T_{ip} + T_{it} + T_{ic} + T_{ap} + T_{at} + T_{ac}}$$

approximated formula:

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

- AND: usually valid
  $T_{ip} = T_{ap} = T_p$
  $T_{ic, ac}$ computing << $T_{ip, ap}$ propagation delay
  T_{it} information frame transm. >> T_{at} ack information frame transmission

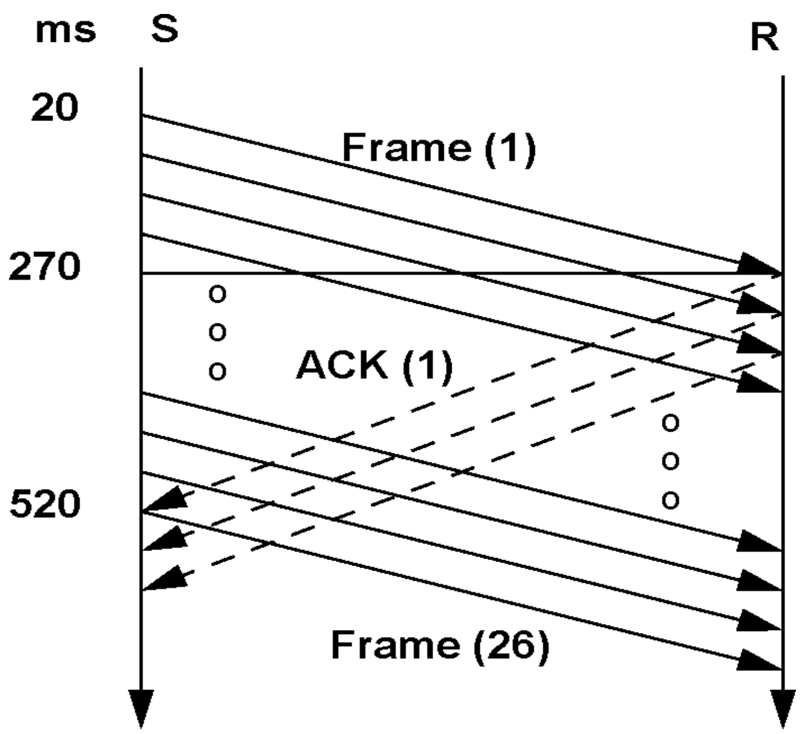
T_{ip} = Frame propagation delay (IF propagation > transmission)
T_{it} = Frame transmission time (time which bits are on channel)
T_{ic} = Frame computing time in S (in comm. nodes HW & SW)
T_{ap} = ACK propagation delay
T_{at} = ACK transmission time
T_{ac} = ACK computing time in P



better: Pipeline ... Sliding Window

Solution: pipelining

Flow control: sliding window mechanism



5.2 Sliding Window: Concept

Flow control: receiving buffer must not flood

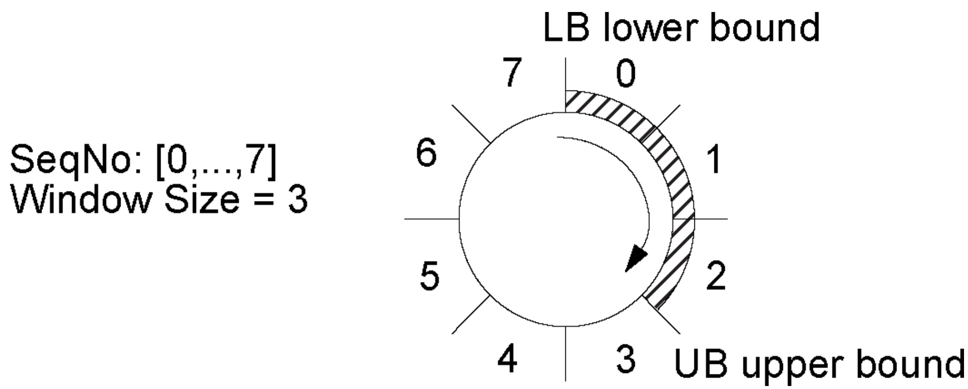
Sender and receiver window per connect/communication relationship

R-WINDOWS:

- sequence numbers, which can be accepted

S-WINDOW:

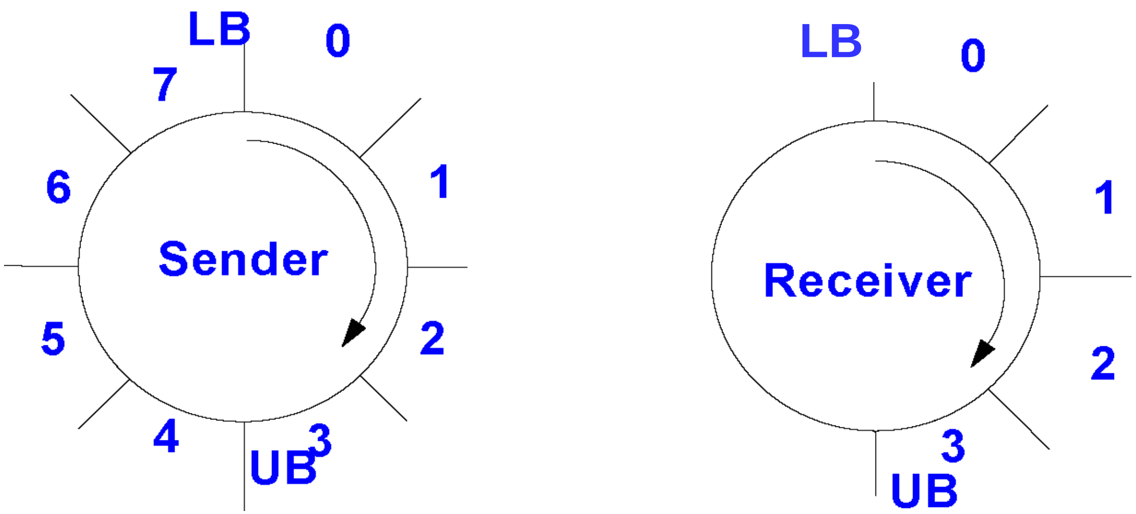
- sequence numbers, which were sent but not yet acknowledged



Initial window size:

- R-Window: number of buffers reserved
- S-Window: maximum number of blocks, which may still be open for acknowledgement

Sliding Window: Concept



SeqNo: [0,...,7]
Window Size = 3

Lower Bound & Upper Bound

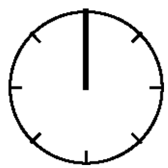
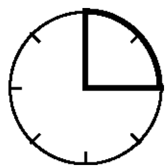
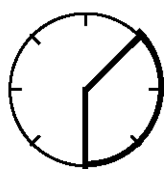
	Sender	Receiver
LB	oldest not yet confirmed seqno.	next, to be expected seqno.
UB	next seqno. to be send	highest seqno. to be accepted

Manipulation: increment(LB), increment(UB), if

	Sender	Receiver
LB	when receipt of an ACK	when receipt of a frame
UB	when sending of a frame	when sending of an ACK

Sliding Window:
Go-Back-N Applet

Sliding Window: Examples

Sender: Sliding Window	Stored Frames	Situation
	0	in this case sender may send up to 3 frames
	2	in this case sender may send 1 frame
	3	sender is not permitted to send anything, sender’s L3 must not transmit further data to L2

TECHNISCHE UNIVERSITÄT DARMSTADT

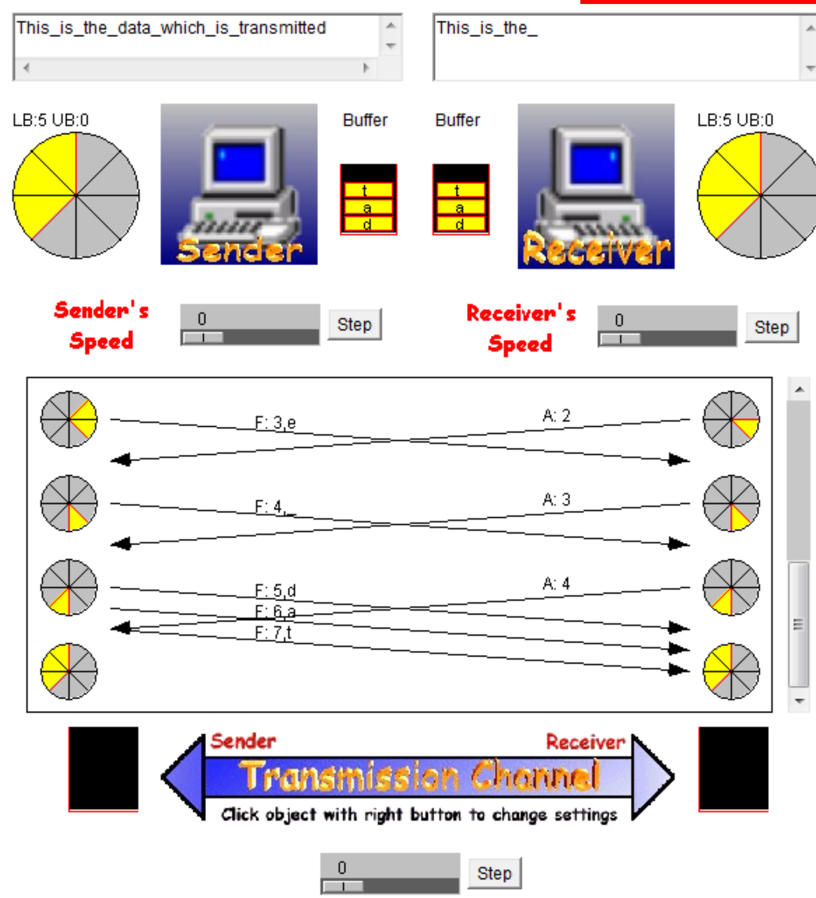
Simulates Sliding Windows

- Go-back-n
- Selective Repeat

Multiple parameters

- Sender’s / receiver’s transmission speed
- Simulation speed
- Timeouts
- Error rate

Applet

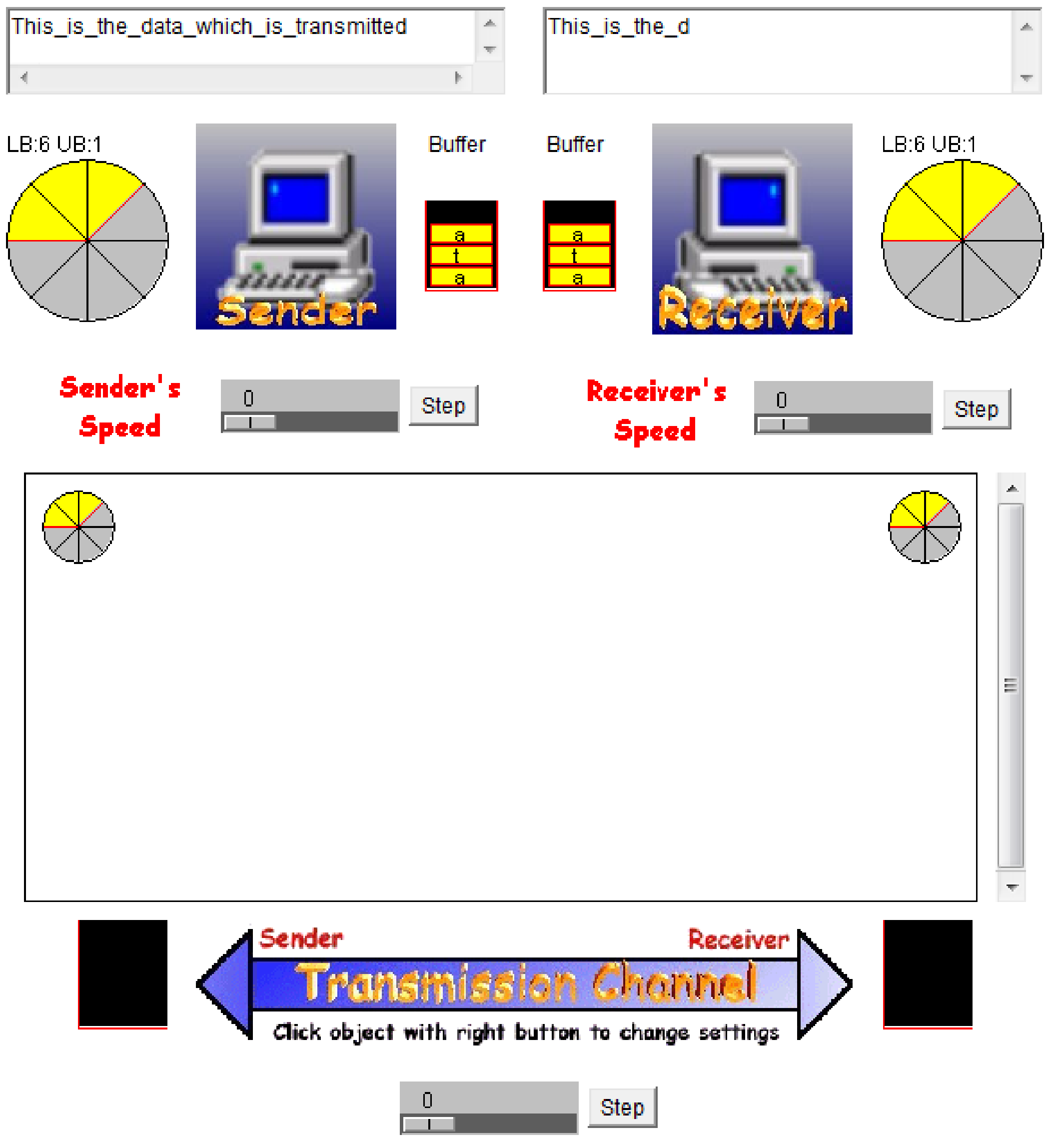


http://www.site.uottawa.ca/~elsaddik/abedweb/applets/Applets/Sliding_Window/sliding-window/index.html

Sliding Window – Active Learning Object

TECHNISCHE UNIVERSITÄT DARMSTADT

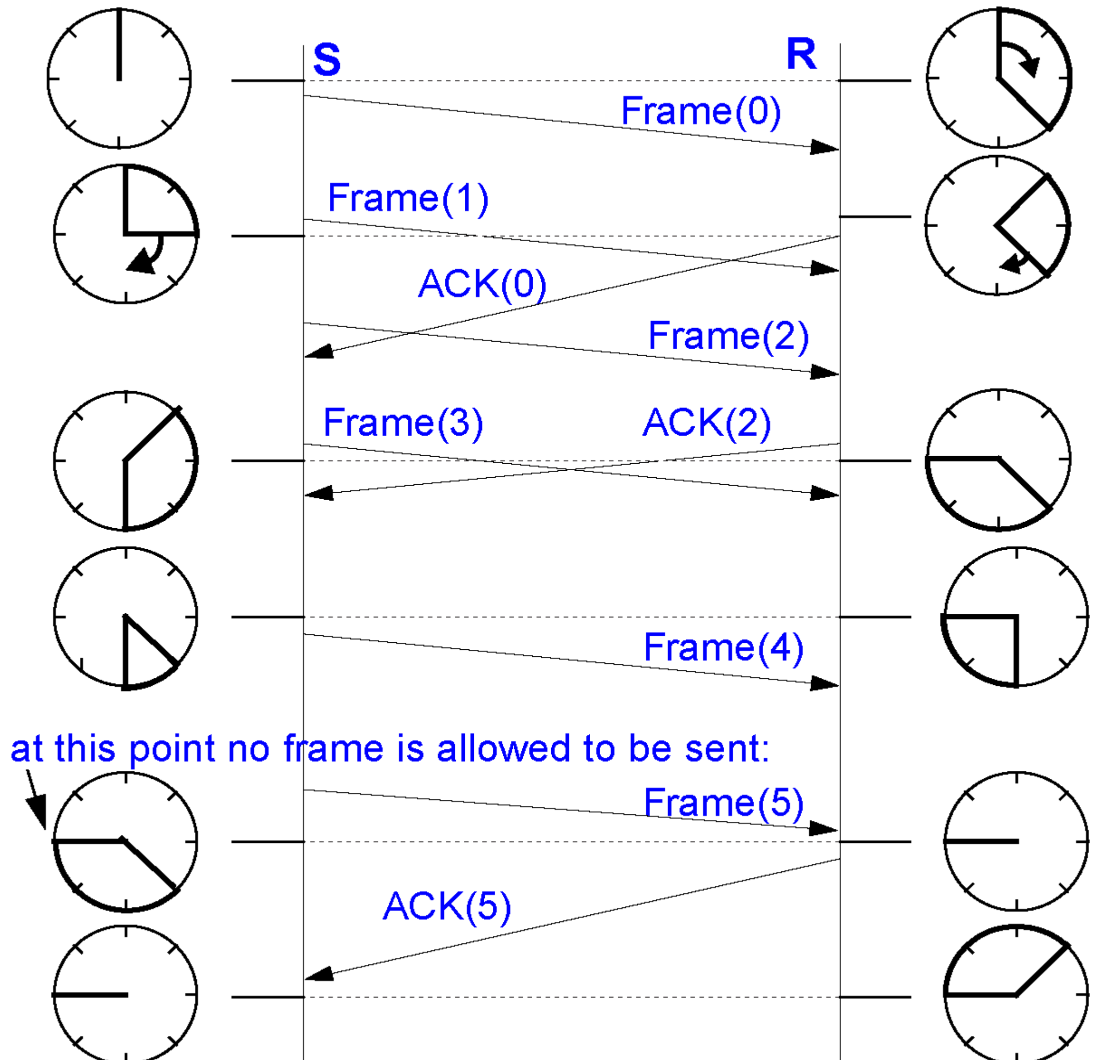
Applet



Example Including Acknowledgement

Including Acknowledgement

- ACKs contain SeqNo
- that means ACK(SeqNo) confirms all frames(SeqNo') with
  - SeqNo' = SeqNo and
  - SeqNo' before SeqNo



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

Example: Description

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

- maximum number determined by receiver’s window size (here 3)
- the frames not yet acknowledged to the sender

ACK sent by receiver if frame

- has been identified as being correct
- has been transmitted correctly to the network layer (or a corresponding buffer)

6 Sliding Window: Remarks & Refinement

- Sliding Window: Influence of the Window Size
- Sliding Window: Piggybacking
- Sliding Window: Go-Back-N (Error Treatment)
- Sliding Window: Selective Repeat (Error Treatment)
- Channel Utilization
- Comparing Protocols

6.1 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 packet) and rate of data
- end-to-end delay on L1
  - e.g. satellite
- window size

Operating resources and quality of service

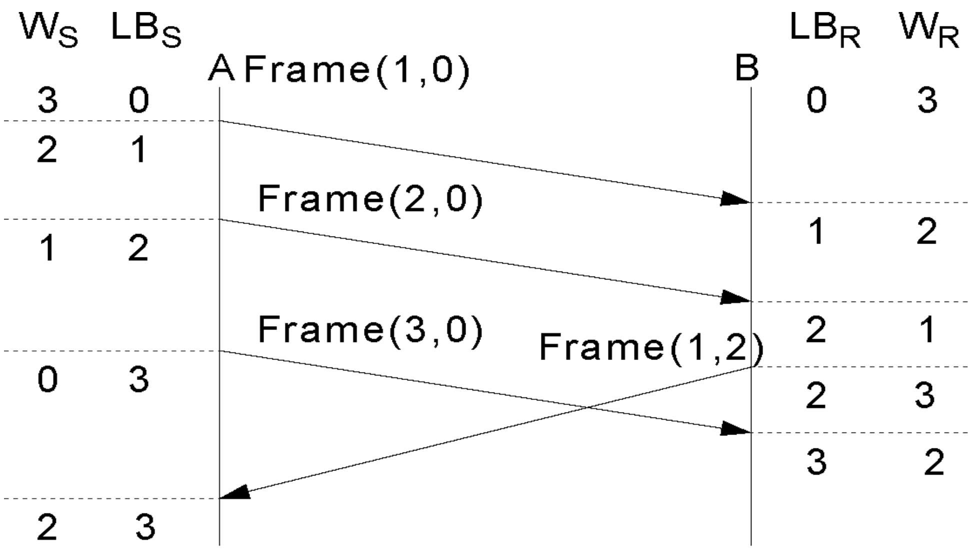
- if the window size is small
  - generally shorter end-to-end delays at the L2 service interface
  - less memory needs to be kept available
    - per L2 communication relation

6.2 Sliding Window: Piggybacking

Frames may contain implicit ACKs

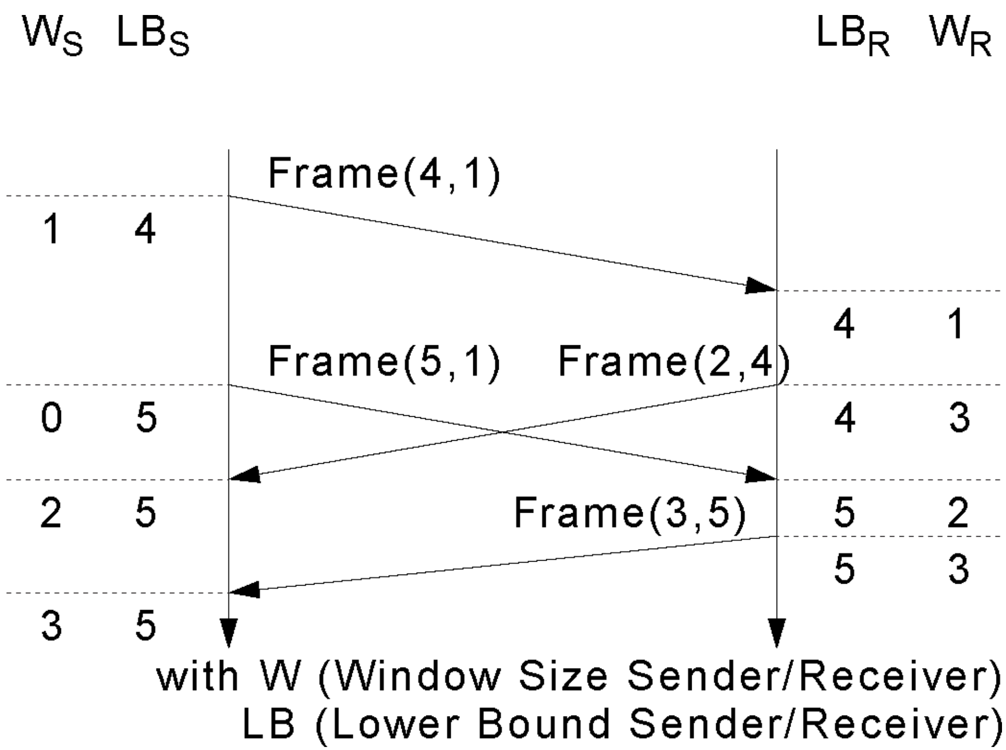
- duplex operation

- assumptions in this example
  - the initial SeqNo. is 0
  - the next SeqNo. and the next ACK-SeqNo to be expected is given



with W (Window Size Sender/Receiver)
LB (Lower Bound Sender/Receiver)

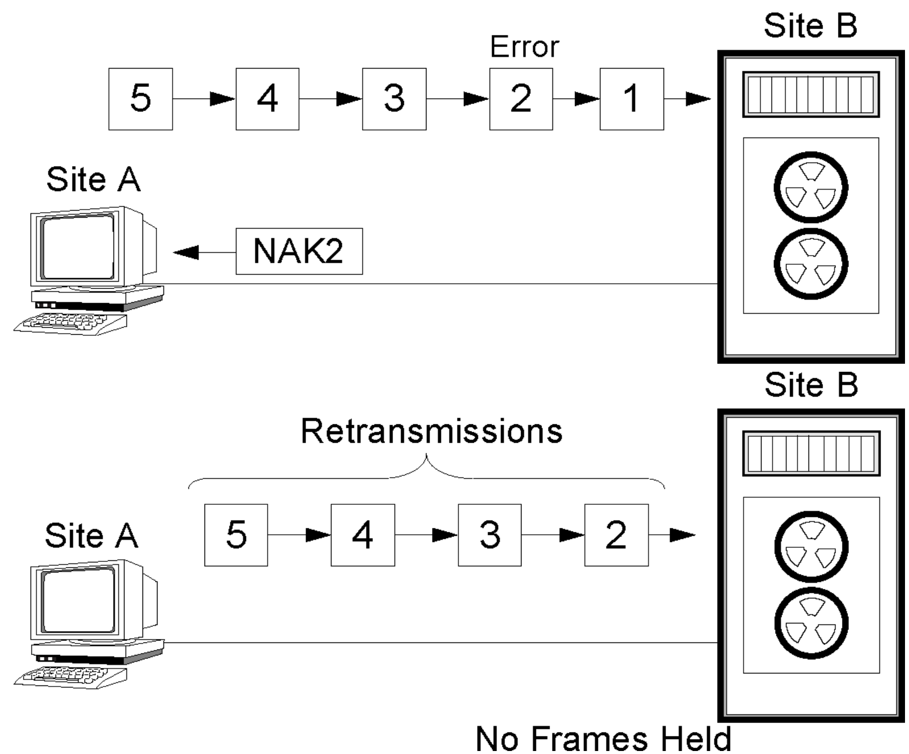
Sliding Window: Piggybacking



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

Procedure

- after a FAULTY FRAME has been received
  - receiver DROPS all FURTHER FRAMES
    until
    - correct frame has been received

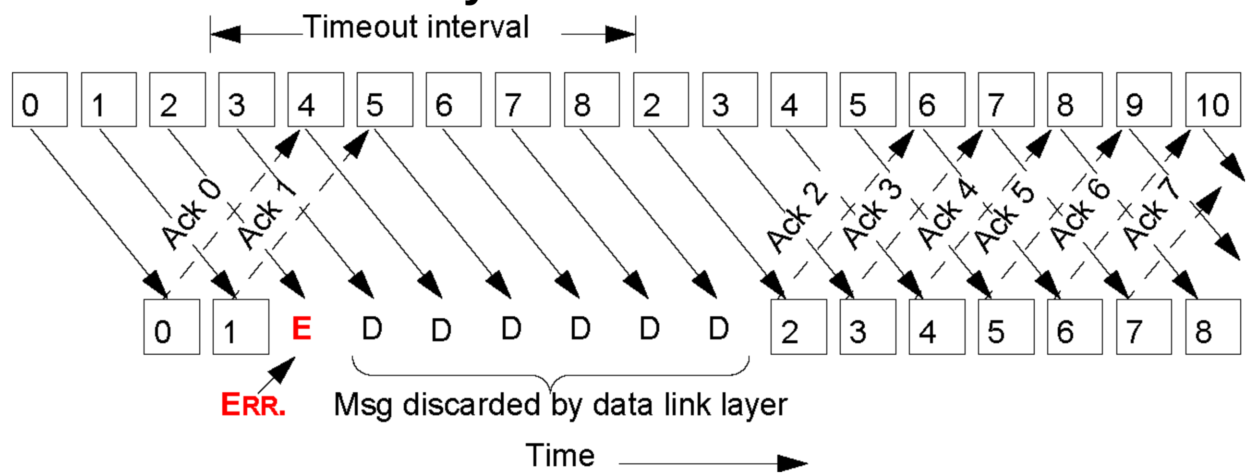


Evaluation

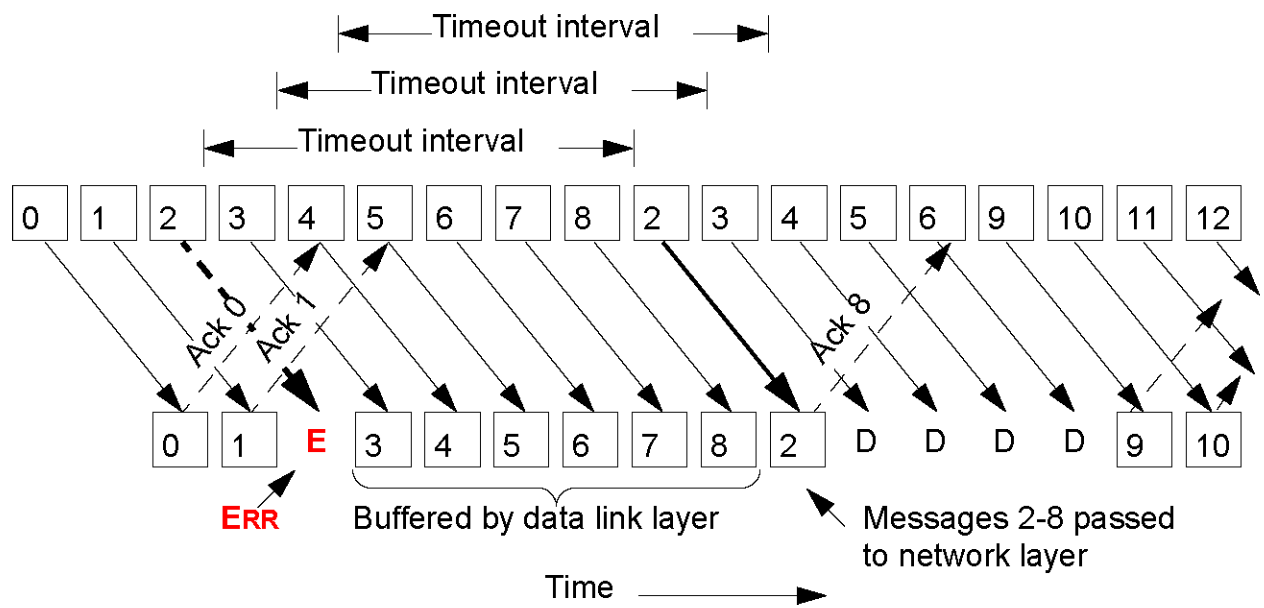
- simple
- no buffering of
  - “out-of sequence” frames necessary
    - (only for optimization purposes)
- poor throughput

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

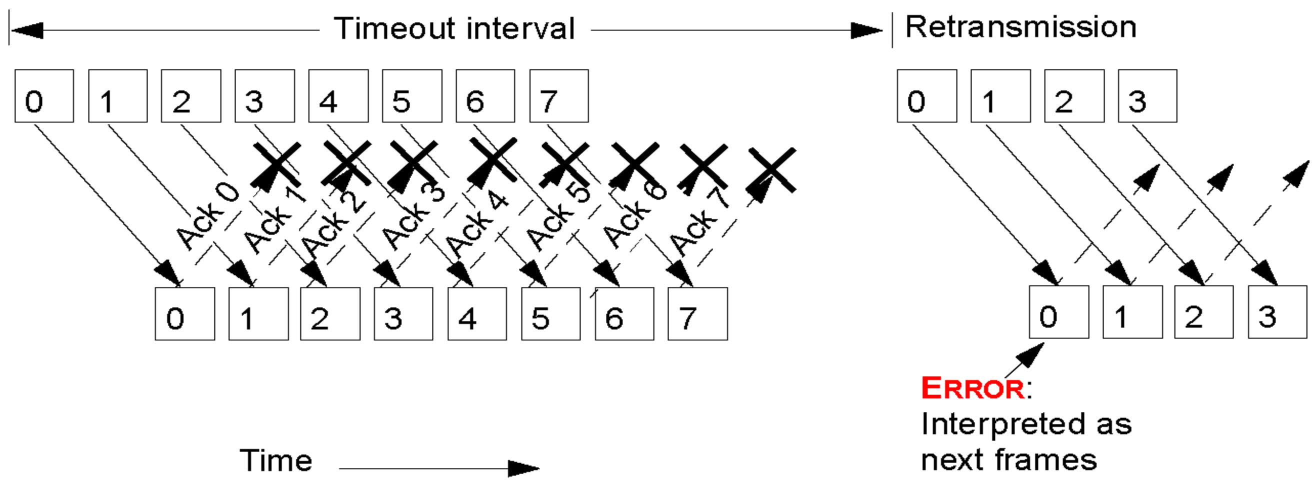
Example: sender: error detection by timeout



Optimization



Sliding Window: Maximum Window Size



Example:

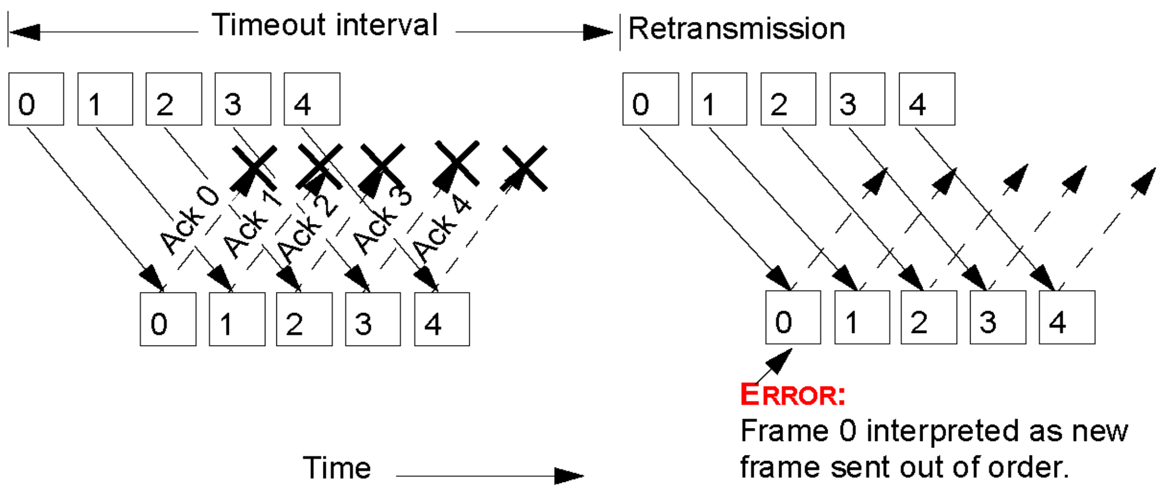
- amount of sequence numbers 8
- window size 8
- all ACKs lost

Correlation between

- window size and
- number of possible sequence numbers:

→ **at least max. window size < range of sequence numbers**

Sliding Window: Maximum Window Size and Frame Sequence



If the sequence is arbitrary
the following situation may for example occur:

- 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

Correlation between window size and number of possible sequence numbers:

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

- during Go back N (otherwise possibly different)

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 has a lot of frames in its buffer
  - and transfers L2 to L3 in correct sequence

Comments

- corresponds to window size > 1
- formation of bursts at data link service interface

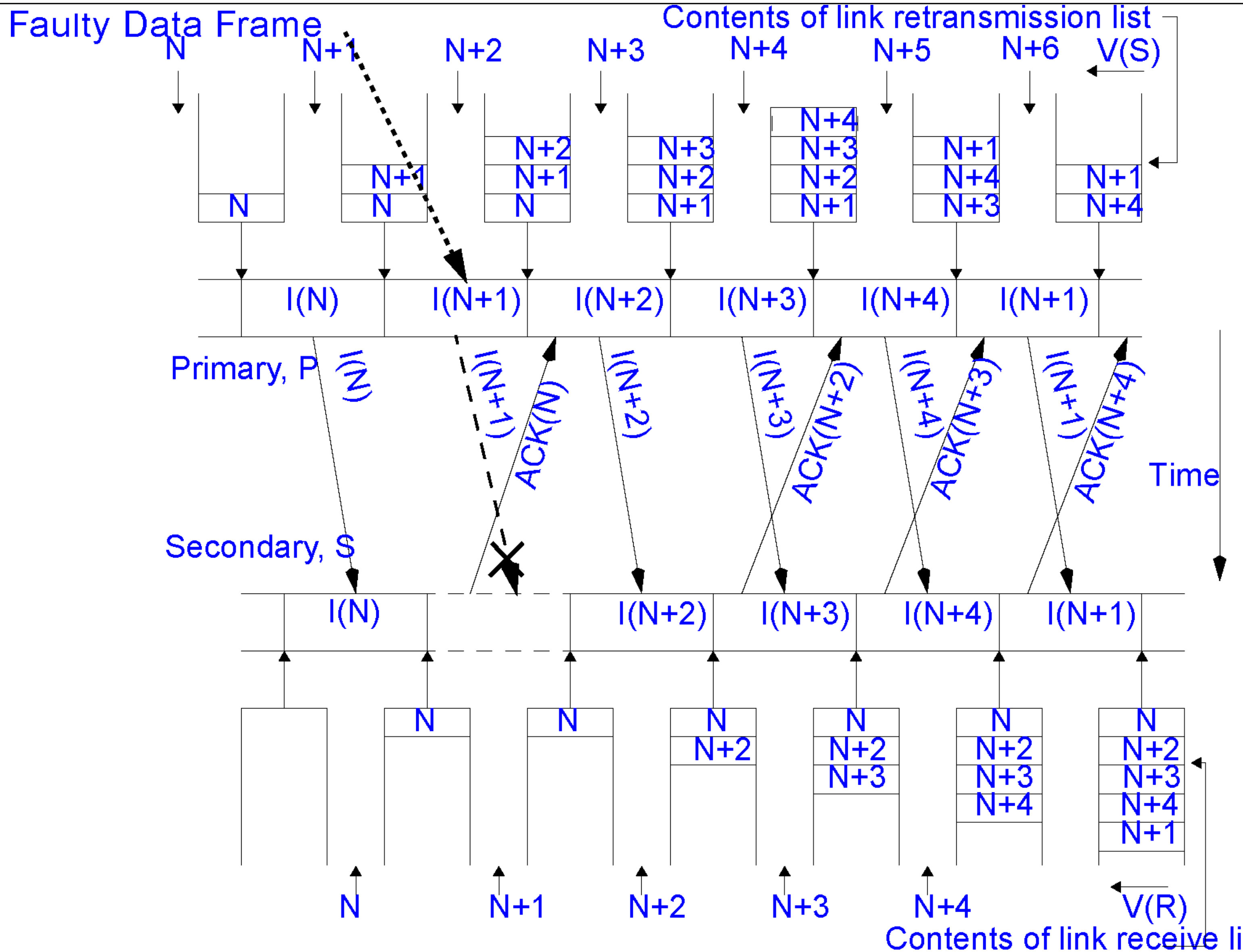
Sliding Window: Selective Repeat

Features

- more complex
- buffering of “out of sequence” frames
- increased throughput
- acknowledgements not included

Example: ....

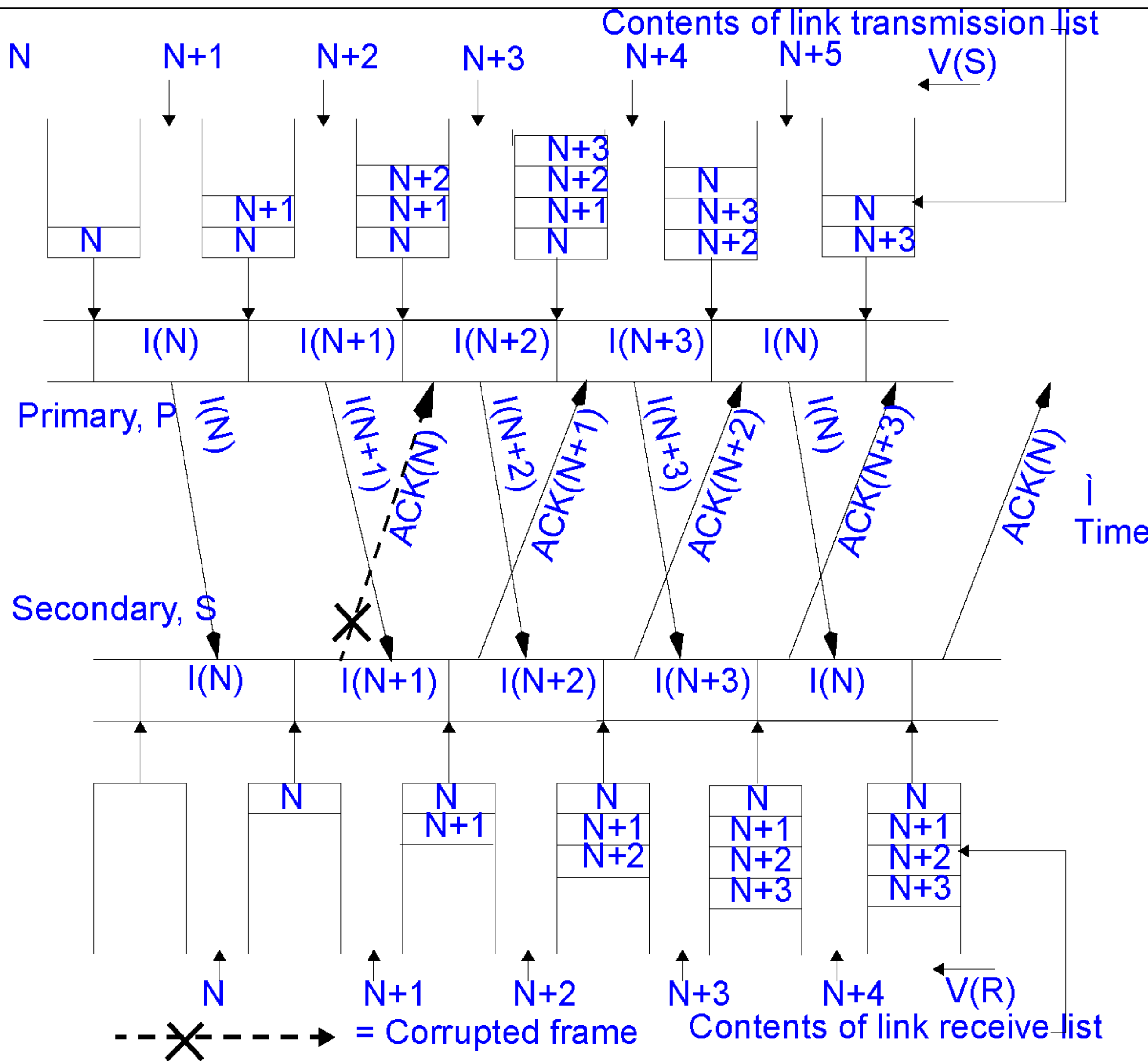
Faulty Data Frame



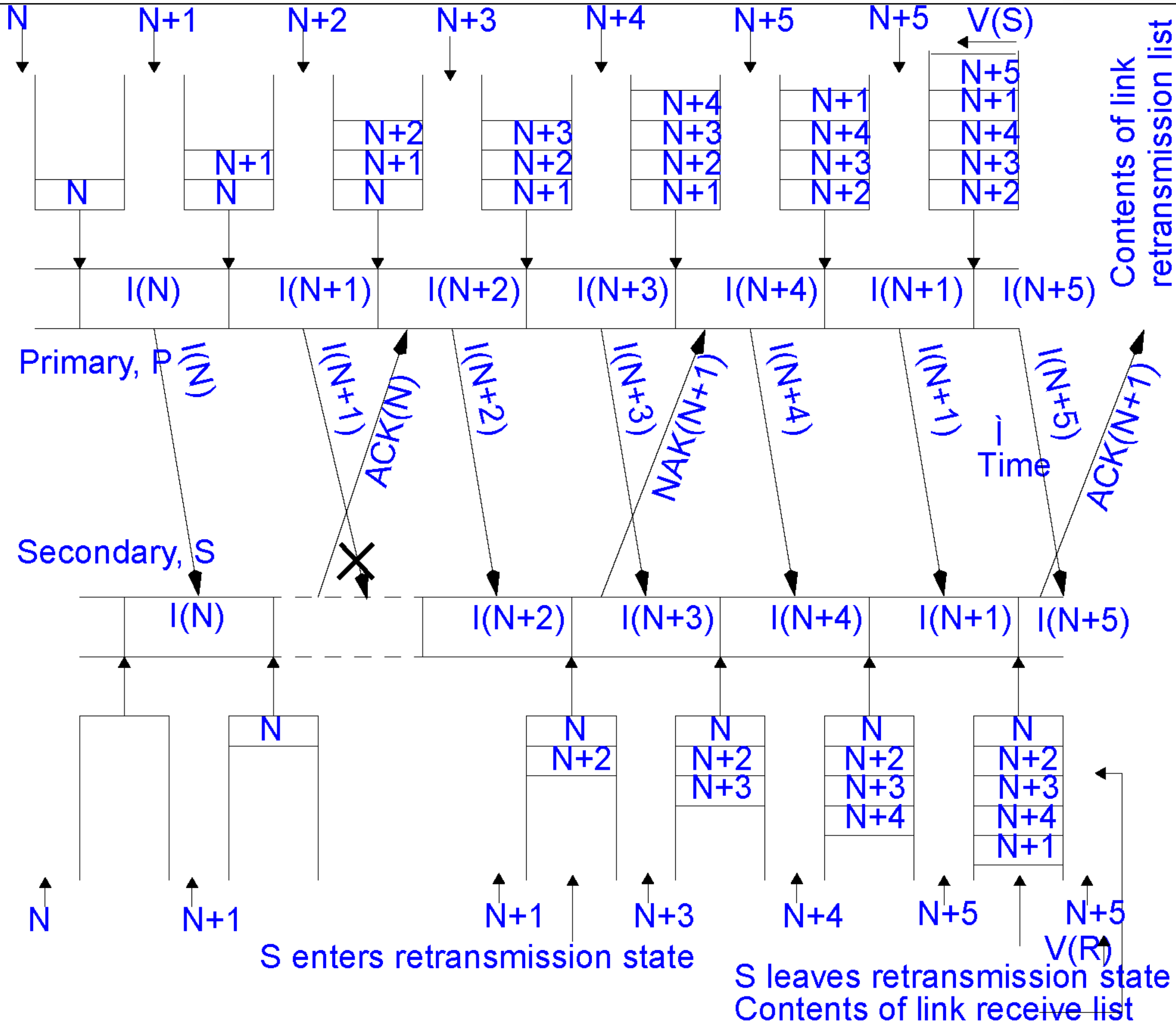
Sliding Window: Selective Repeat

Example: …

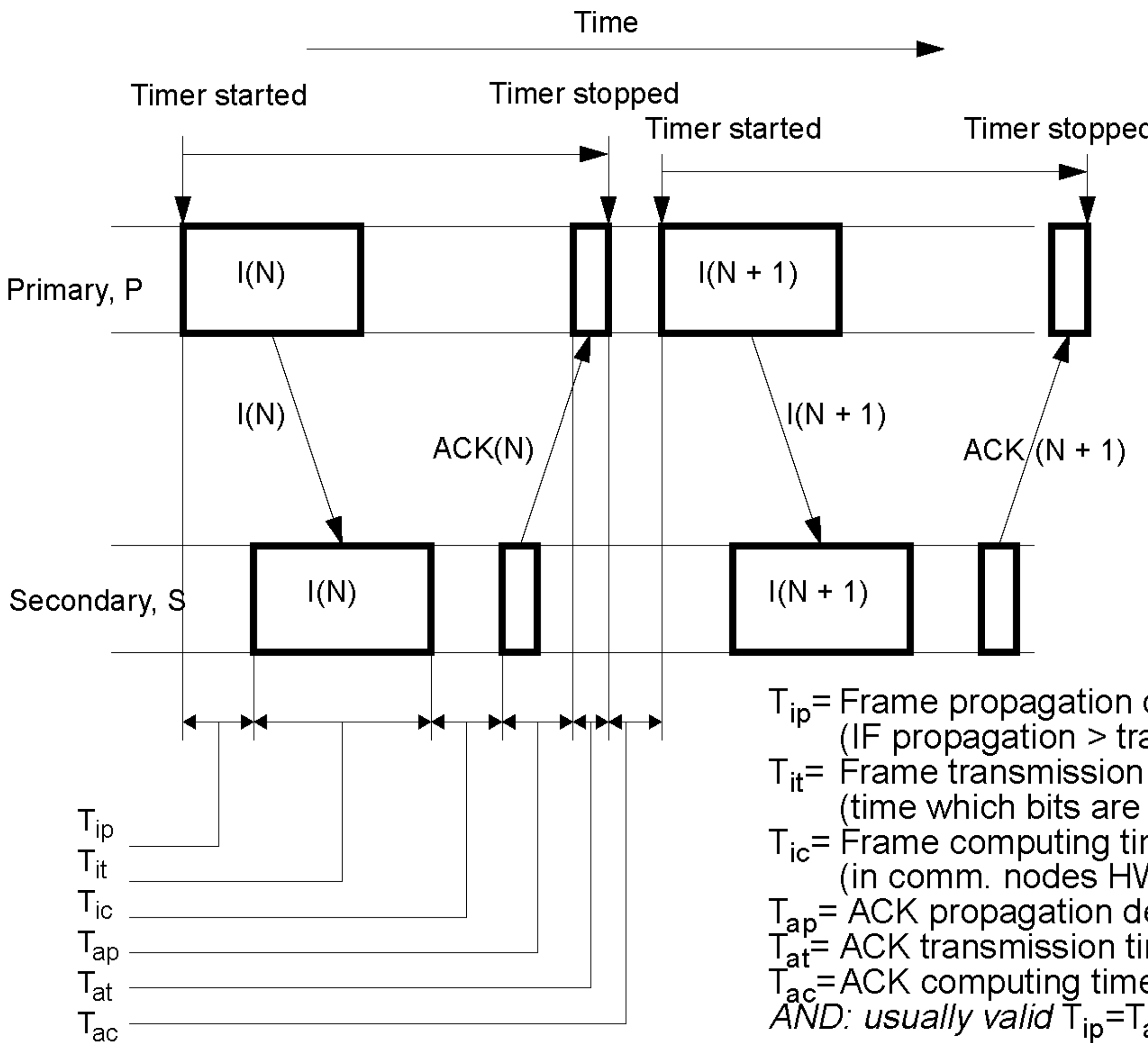
Faulty Acknowledge Frame



Active Error Control



6.5 Channel Utilization



T_{ip}= Frame propagation delay ($\mathbf{P} \rightarrow \mathbf{S}$)
(IF propagation > transmission)
T_{it}= Frame transmission time ($\mathbf{P} \rightarrow \mathbf{S}$)
(time which bits are on channel)
T_{ic}= Frame computing time in *S*
(in comm. nodes HW & SW)
T_{ap}= ACK propagation delay($\mathbf{S} \rightarrow \mathbf{P}$)
T_{at}= ACK transmission time($\mathbf{S} \rightarrow \mathbf{P}$)
T_{ac}=ACK computing time in *P*
AND: usually valid T_{ip}=T_{ap} = T_p

Channel Utilization and Propagation Delay: Recapitulation without Sliding Window

exact formula (note: some values based on assumptions):

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

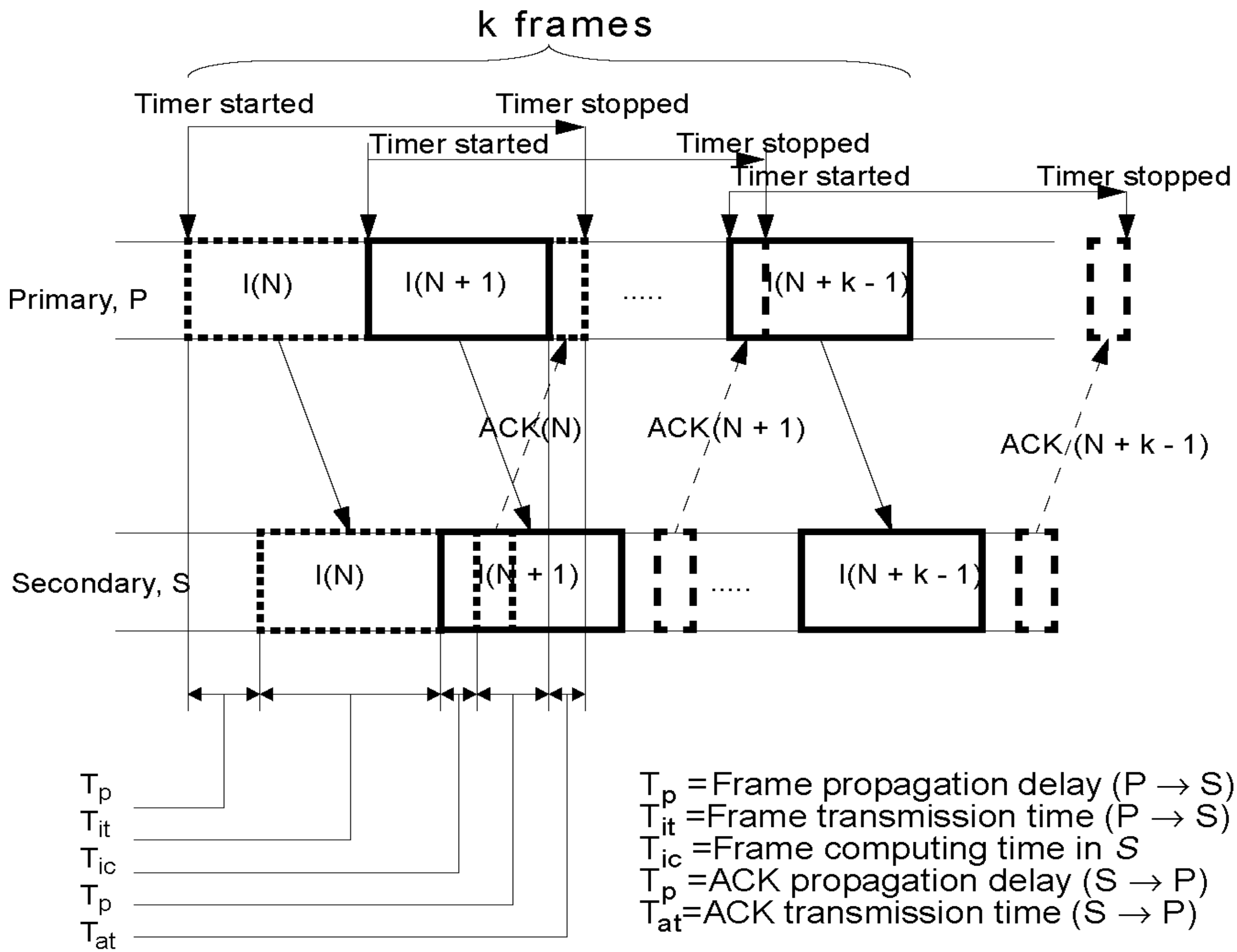
approximated formula:

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

- AND: usually valid
  - $T_{ip} = T_{ap} = T_p$
  - T_{ic}, ac computing << T_{ip}, ap propagation delay
  - T_{it} information frame transm. >> T_{at} ack information frame transm.

T_{ip} = Frame propagation delay (IF propagation > transmission)
T_{it} = Frame transmission time (time which bits are on channel)
T_{ic} = Frame computing time in S (in comm. nodes HW & SW)
T_{ap} = ACK propagation delay
T_{at} = ACK transmission time
T_{ac} = ACK computing time in P

Channel Utilization with Sliding Window



T_p = Frame propagation delay ($P \rightarrow S$)
T_{it} = Frame transmission time ($P \rightarrow S$)
T_{ic} = Frame computing time in *S*
T_p = ACK propagation delay ($S \rightarrow P$)
T_{at} = ACK transmission time ($S \rightarrow P$)

T_{ac} may be neglected if window size > 1

Channel Utilization

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

Comment

- k specifies
  - how many frames are transmitted simultaneously (sequentially) on the L1 channel
  - i.e. k is the window size

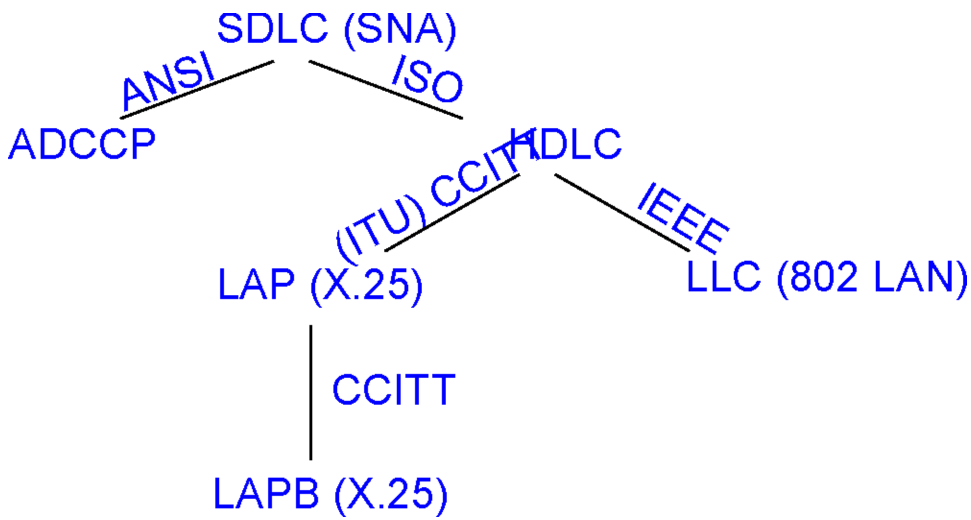
6.6 Comparing Protocols

Stop-and-Wait

+ little demand for buffering
+ well-suited for less complex end devices
- poor channel utilization
- low throughput

Sliding Window

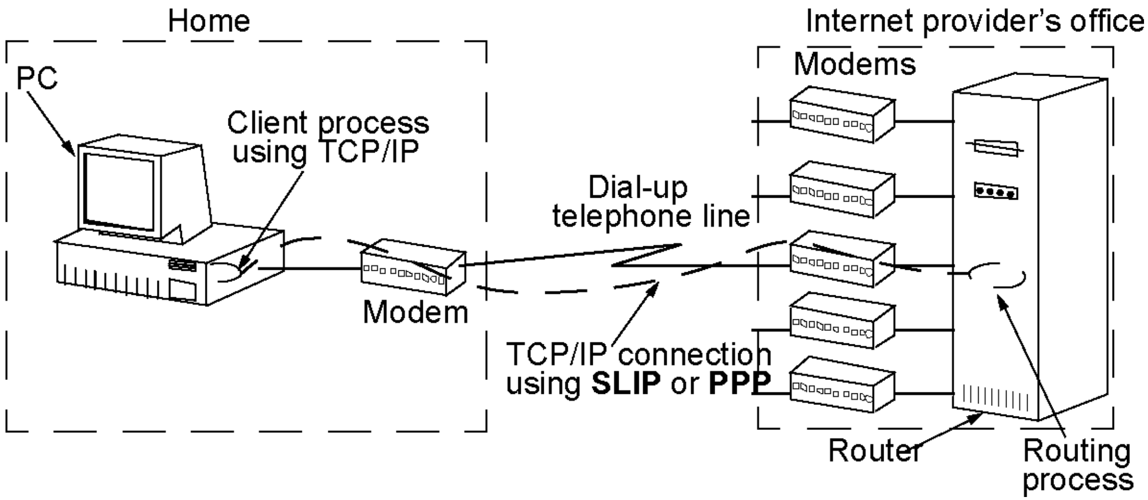
+ good channel utilization
+ good throughput
+ consideration of the current system state by adjusting the window size
- increased buffer demand
- more complex protocol



SDLC:	**Synchronous Data Link Control (derived from IBM System Network Architecture SNA)**
ADCCP:	**Advanced Data Communication Control Procedure**
HDLC:	**High-Level Data Link Control**
LAP:	**Link Access Procedure**
LAPB:	**Link Access Procedure, Balanced, exple.L2 from OSI L3: X.25**
LLC:	**Logical Link Control**
others:	**Kermit, XMODEM (for modems between PCs)**

→ "The nice thing about standards is that you have so many to choose from" [Tanenbaum]

8 Protocols: at Internet Layer 2



Internet Connections

- end devices to "network"
  - approx. 10m - 10 km: LAN & MAN: many connected to each other
  - more than approx. 10-100 km: WAN
    - point-to-point (considered here)
    - an Internet provider (AOL, T-Online, Univ.,...)
    - usually via phone line and modem
    - usually TCP/IP over SLIP or PPP
- connection between network nodes
  - point-to-point: (reviewed herein)
  - usually over dedicated line with router or bridge
  - often IP over SLIP or PPP

8.1 Internet: Serial Line IP (SLIP)

History

- 1984: Rick Adams connects Sun computers via modem to the Internet
- description in RFC 1055

Protocol (very simple)

- data (payload)
  - L3 packets (here only IP packets)
- framing:
  - add flag byte (0xC0) at the end of the packet
  - character stuffing: if 0xC0 part of the data, replace this by 0xDB, 0xDC, etc.
  - note: some implementations insert these also as headers

Internet: Serial Line IP (SLIP)

Properties

- no error detection or error correction
- only IP is supported
  - IP addresses of communicating entities have to be known in advance
  - (i.e. cannot be allocated dynamically)
  - i.e. each user would have to have his own IP address on the host -> too many addresses
- no authentication
  - (problem for switched line, not dedicated line)
- many different implementations exist because no Internet standard
- widely used

Optimizations:

- RFC 1144
- properties:
  - successive packets often have the same header
- use:
  - header compression, if successive packets are the same

8.2 Internet: Point-To-Point Protocol (PPP)

History

- IETF initiated group to
  - replace SLIP (not a standard) by the Internet standard
  - improve the data link protocol
- application: login connections and dedicated lines
- RFC 1661 (others: RFC 1662, RFC 1663)
- will replace SLIP

Protocol: character oriented (SLIP bit oriented) with

- FRAMING: frame identification and error treatment
- Link Control Protocol LCP (PHASE 1)
  - establish, test, release L2 connection
  - authentication: L2 entities can determine each other's identities
  - negotiate options with L2 partner entity (e.g. coordinate payload size, select NCP protocol)
- Network Control Protocol NCP (PHASE 2)
  - one NCP per each L3 protocol
  - NCP for IP: selects for example IP address
- actual data transfer (PHASE 3)

Internet: Point-To-Point Protocol (PPP)

Bits (of SDLC to compare)

8	**8**	**8**	**no**	**≥0**	**16**	**8**

Bytes PPP here:

1	**1**	**1**	**1 or 2**	**≥0**	**2 or 4**	**1**
Flag **01111110**	**Addr.** **11111111**	**Ctrl.** **00000011**	**Protocol**	**Payload**	**Checksum**	**Flag** **01111110**

Frame Format (similar to or derived from HDLC):

- (HDLC) flag: identifying characters for L2 frame
- address: always addressing of all stations
- control: default: unnumbered frame (see above) without ACK, without error treatment
- otherwise: numbered mode

(Commt.: Address & Control fields can be left out depending on the setup)

- protocol: designates the protocol, usually 2 byte
  - L2: LCP, NCP
  - L3: IP, OSI CLNP, Appletalk, ..
- payload/data: any user data
  - length: negotiated, otherwise max. 1500 byte
- checksum: CRC variation, usually 2 byte
- (HDLC) flag: bound of the L2 frame

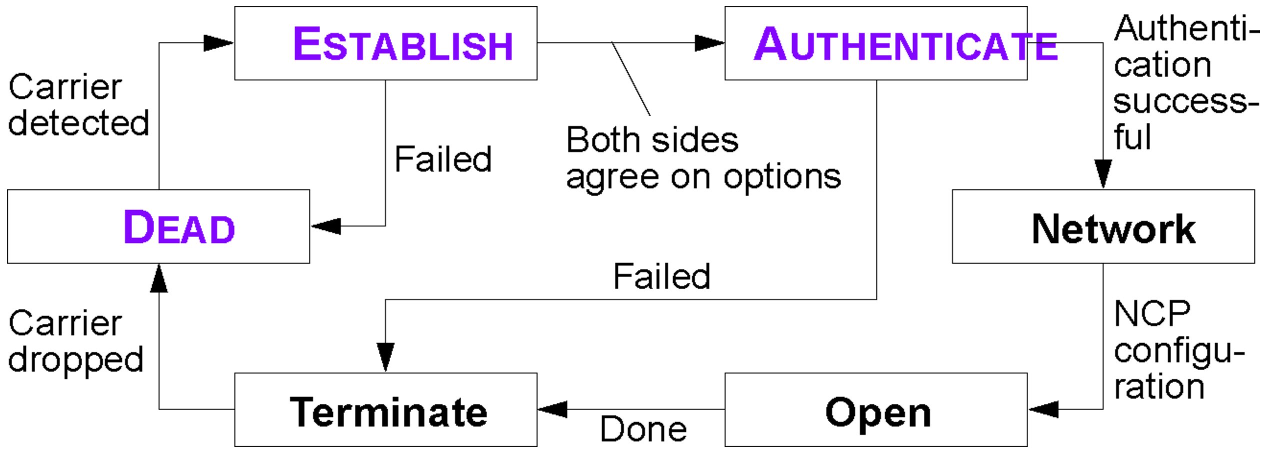
Internet: Point-To-Point Protocol (PPP)

Features

- error treatment
- supports several L3 protocols/services
- IP addresses are determined dynamically
- authentication

Example: ....

Internet: Point-To-Point Protocol (PPP)



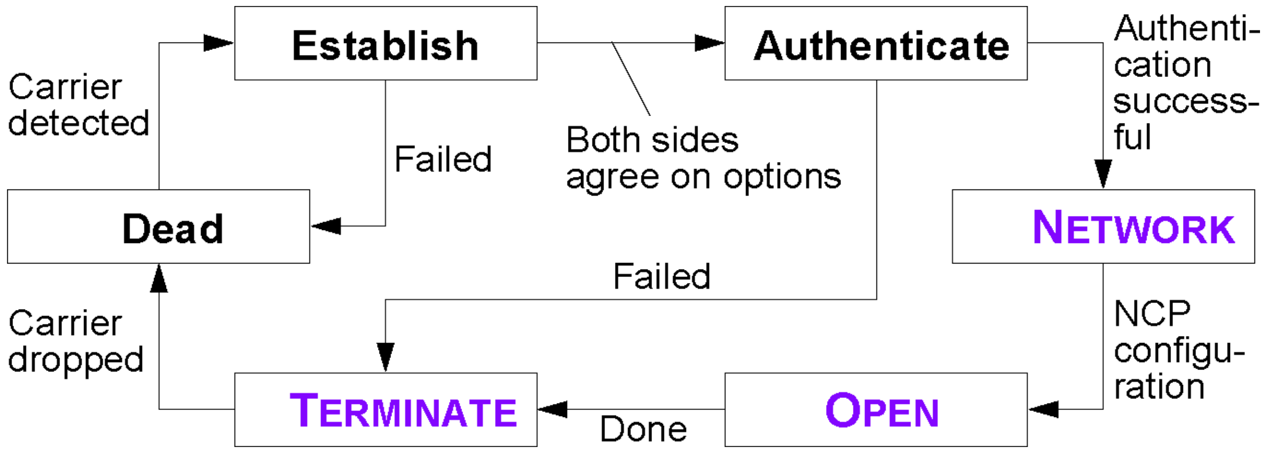
Dead
- L1 connection does not existent

Establish
- after L1 connect
- negotiation of LCP options

Authenticate
- authentication of both parties

...

Internet: Point-To-Point Protocol (PPP)



...

Network

- call of the desired NCP protocol
- configuration of the network layer

Open

- data transfer may begin
- e.g. transmission of IP packets in the payload field of PPP frames

Terminate

- disconnect

Service and protocol design include among others questions about:

Specification method

- type (Petri-Net, SDL, ESTELLE, LOTOS,..)
- utilization range (up to generating programs)
- deadlocks may be recognized (depending on method)

Implementation

- HW vs. SW
- SW environment (C, C++, class library,...)
- buffer management (logical copying)
- process segmentation

Service selection

- connection oriented vs. confirmed connectionless vs. connectionless

Configuration/parameter selection

- sliding window size (among others depends on L1 error properties, L1 transmission time)
- error correction procedure