

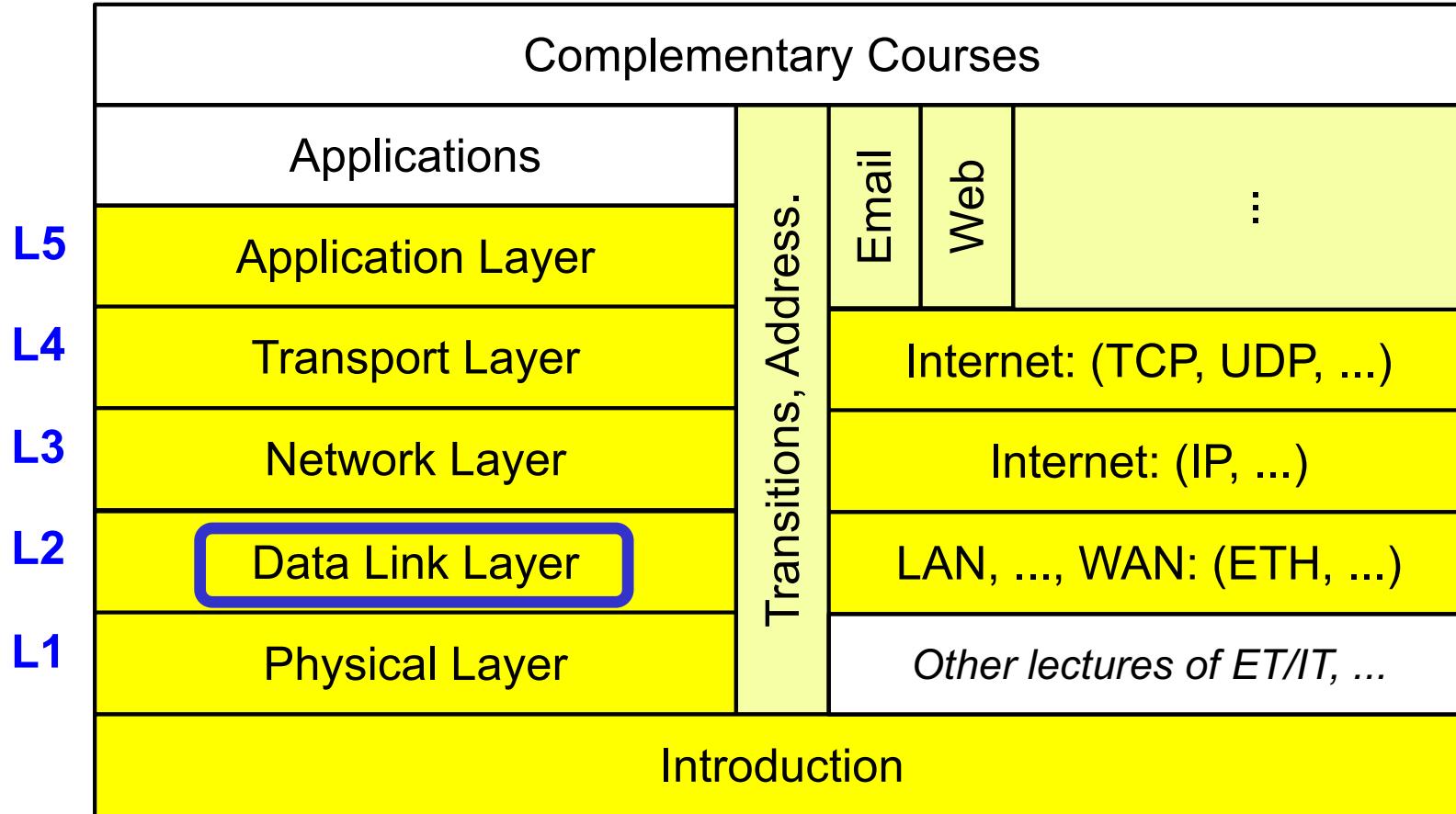
Computer Networks I

Data Link Layer

Prof. Dr.-Ing. **Lars Wolf**

IBR, TU Braunschweig
Mühlenpfordtstr. 23, D-38106 Braunschweig, Germany,
Email: wolf@ibr.cs.tu-bs.de

Scope



Overview

- 1 Scope of the Link Layer
- 2 Link Control / Framing
- 3 Error Detection and Correction
- 4 Flow Control and Error Treatment
- 5 Sliding Window – Flow Control & Error Treatment
- 6 Sliding Window: Remarks & Refinement
- 7 Example: HDLC Family
- 8 Example: Internet-related Layer 2

1 Scope of the Link Layer (L2)

Overall goal:

- provide data transfer between adjacent stations
- over one (or more) links
- provide (potentially) reliable data transfer
- for larger data units (frames):
not just bits, but of limited size
- using L1 service → on top of physical layer

What is provided by L1?

What is provided by L1?

L1 Service: transmission of a **bit stream** ("unreliable bit pipe")

- without sequence errors
- but other errors are possible: loss, insertion, bit flips

'malign' features of L1 service

- caused by communication channel
- finite propagation speed between sender and receiver
- limited data rate

→ Improve this L1 service

L2 Service & Function

L2 Service:

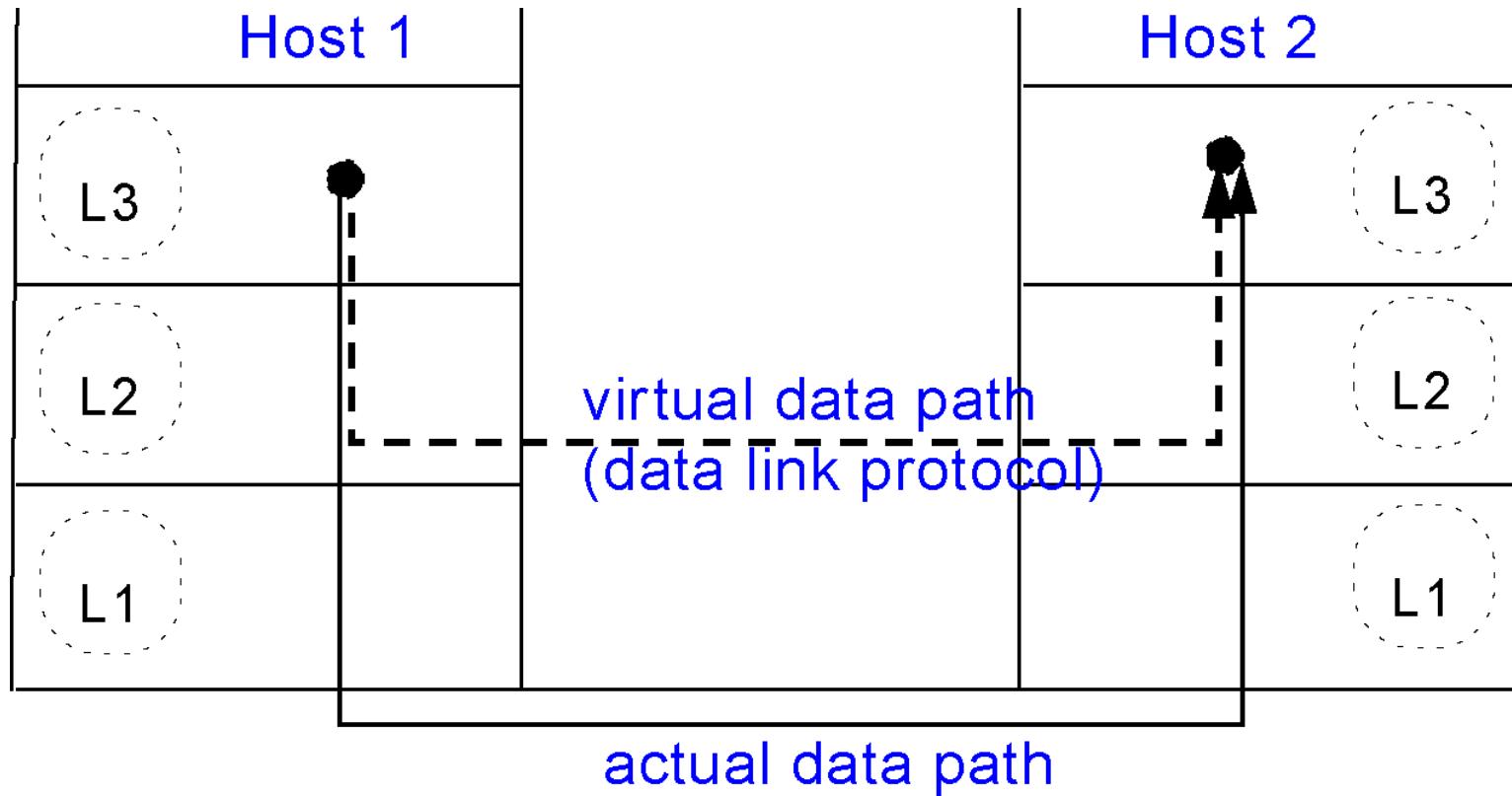
- (reliable), efficient data transfer between adjacent stations
 - may be between more than 2 stations
 - adjacent = connected by one physical channel
 - wireless, coax, optical fiber,...

L2 Functions:

- data transmission as **FRAMES**
- **ERROR** control and correction
- **FLOW CONTROL** of the frames
- configuration management

Services

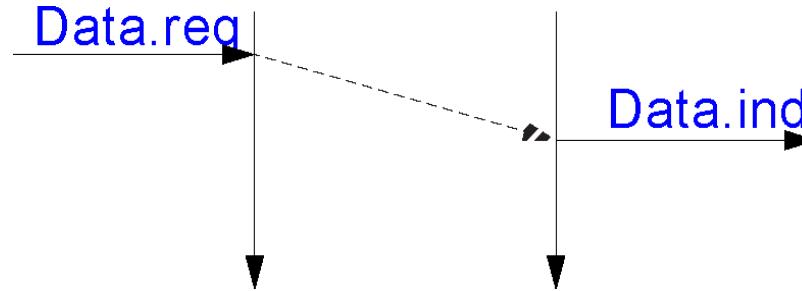
Actual data path and virtual data path:



L2 service classes:

- unconfirmed connection-less service
- confirmed connection-less service
- connection-oriented service

L2 Service Class: Unconfirmed Connection-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

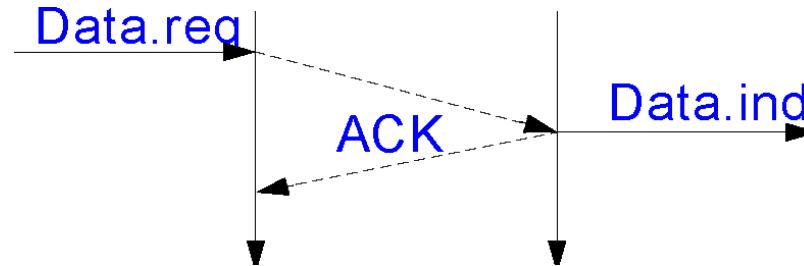
Features

- no flow control
- no connection setup (and no release)

Usage

- on L1 communication channels with VERY LOW ERROR RATE
 - corrections will possibly be done at a higher level
- For real time data transfer like interactive voice communication
 - timing errors probably more critical than errors in the voice data
- often used in LANs

L2 Service Class: Confirmed Connection-less Service



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)

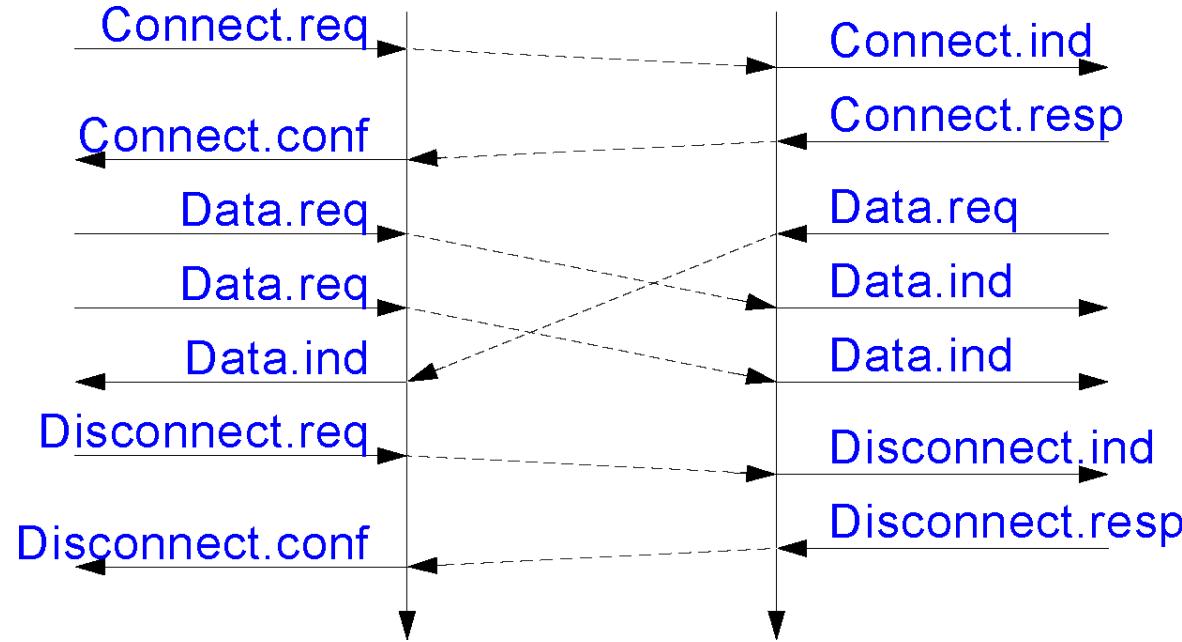
Features

- no flow control, no connection setup (and no release)
- duplicates and sequence errors may happen due to “retransmit”

Usage

- L1 communication channel with high error rate,
e.g. wireless channel

L2 Service Class: Connection-Oriented Service



Connection: idea is to offer error free channel with

- no loss, no duplication, no sequencing error
- flow control

3-phases

1. Connection setup
 - initialize the counters/variables of the sender and receiver
2. transfer data
3. disconnect

L2 Services: Comments

Acknowledgments on L2:

- is only for optimization but is not indispensable
- because this can also be done at a higher level (L4)

However

- L4 message may consist out of n (e.g. 10) L2 frames
 - if there is an error in a frame
→ the **whole** message will be retransmitted
 - that means loss of time and efficiency

End-to-end argument:

- J.H. Saltzer, D.P. Reed, D.D. Clark:
"End-to-End Arguments in System Design",
ACM Transactions on Computer Systems,
Vol. 2, No. 4, November 1984, pp. 277-288

2 Link Control / Framing

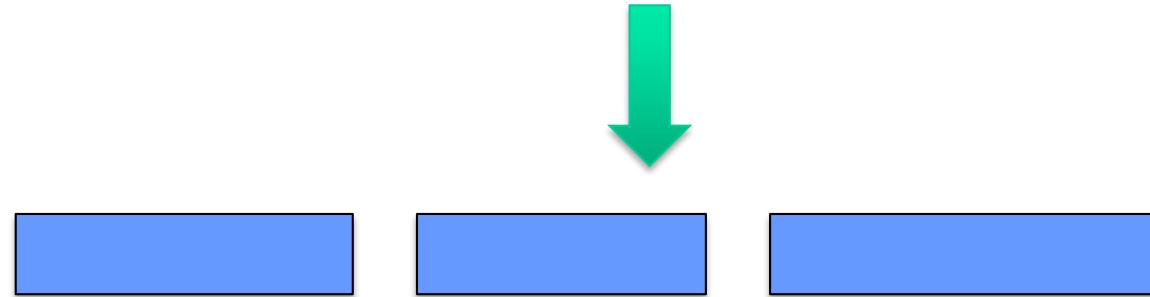
As said before:

L1 Service: transmission of a **bit stream** ("unreliable bit pipe")

→ L2 should improve this L1 service

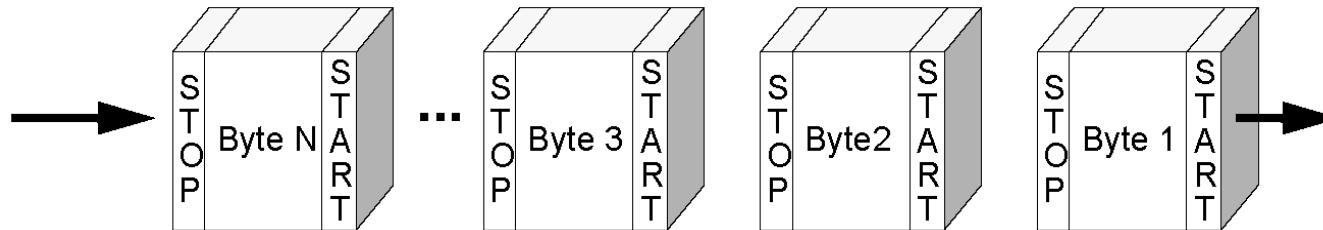
Provide data units / **frames** instead of bit stream

```
0011001010010001101001001100100110010101001100101000100100110010011001001100101  
010011001010010010001010010001101001001100100110010011001010010011010010010011  
0100100011010010010011
```



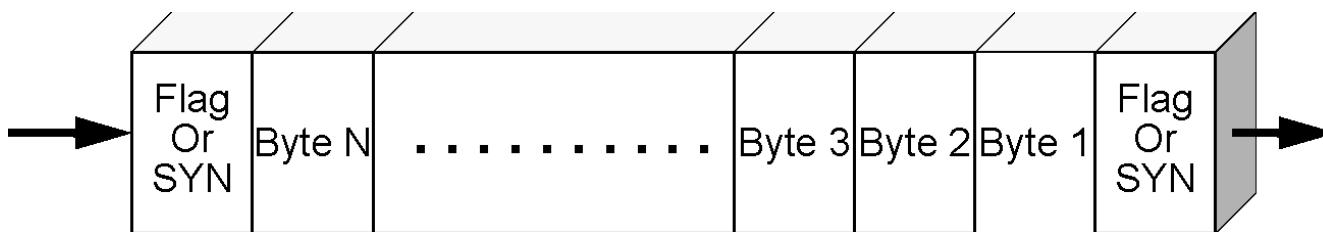
Asynchronous and Synchronous

Asynchronous transmission



- each character is bound by a start bit and a stop bit
- simple + inexpensive, but low transmission rates

Synchronous transmission (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

- Error correction is applied to whole frame (as a unit)

Possibilities for definition and recognition of frame boundaries

0. bound / guard frames by idle times

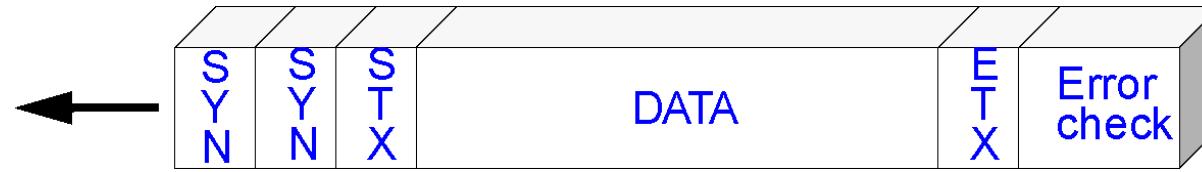
- problems
 - L1 might not have a suitable notion of time
 - possibly loss of efficiency

1. character oriented
2. count oriented
3. bit oriented
4. using invalid characters of the physical layer

Comment

- Combinations may be used in L2, e.g.
 - count oriented and bit oriented
 - the transmission is error free only if both match

2.1 Character-Oriented Protocols



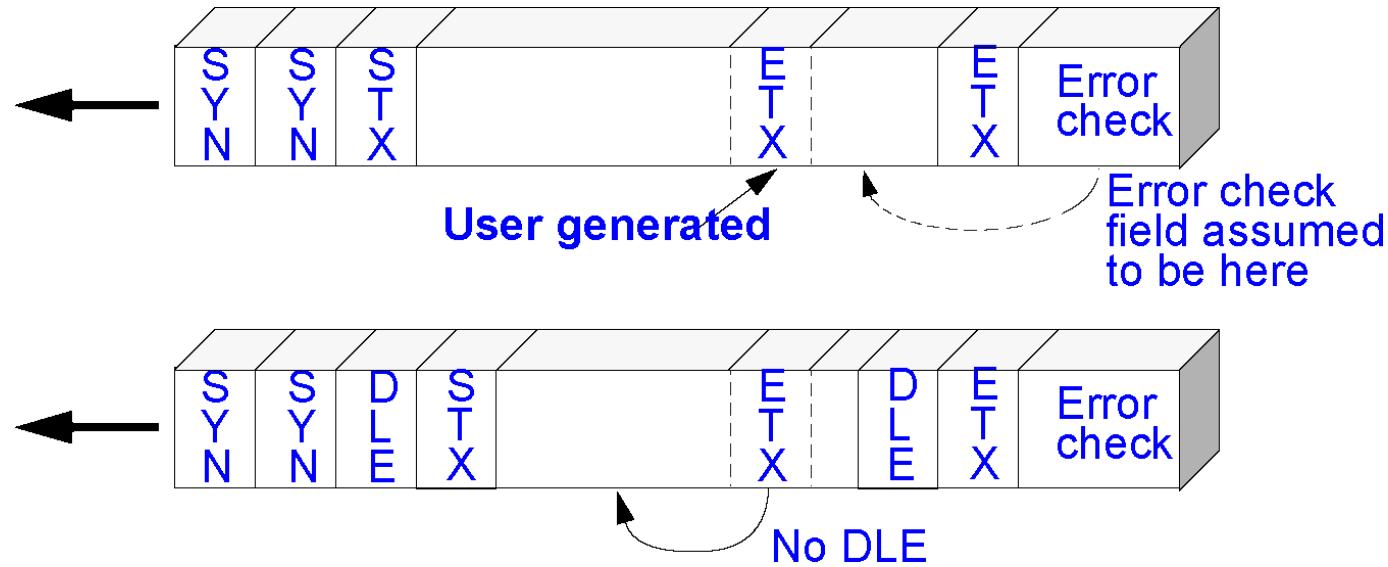
Control Fields

- flag frame areas
- depend on encoding (e.g. ASCII, EBCDIC)

Character-Oriented Protocols

Problem: user data may contain “control characters”

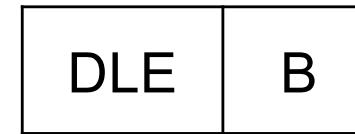
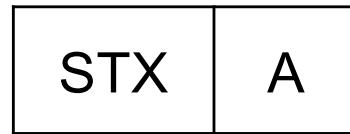
Solution: CHARACTER STUFFING



- **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

(a)

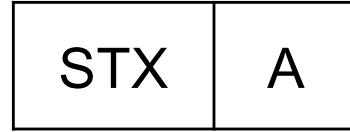


(b)



Stuffed
DLE

(c)



Problem: user generated data may contain DLE

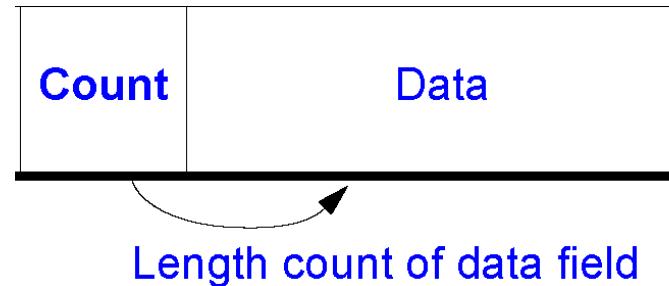
Solution:

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

Disadvantages:

- DLE insertion requires additional effort/time

2.2 Count-Oriented Protocol



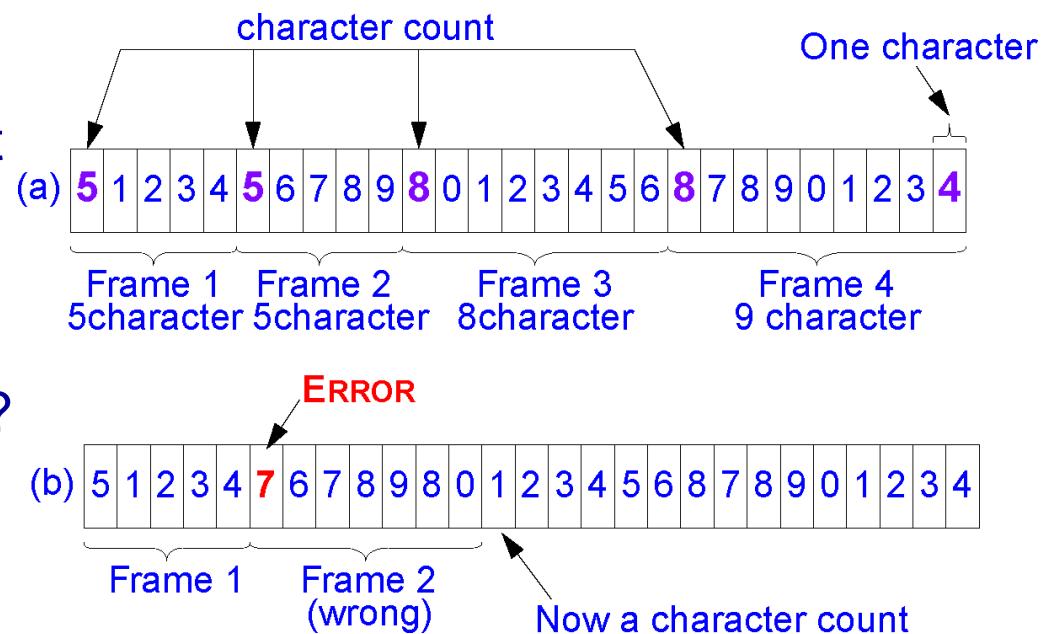
Frame contains LENGTH COUNT FIELD

Problem: Transmission error may
destroy length count

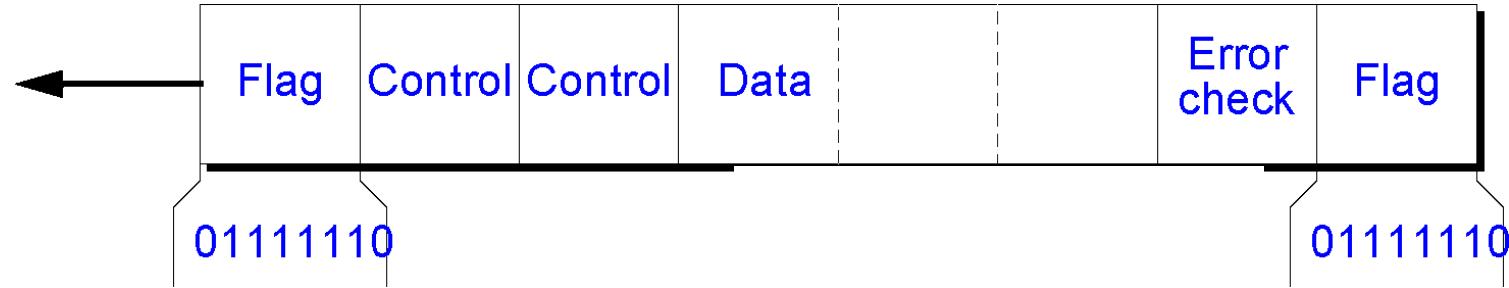
→ sender and receiver are not
synchronized anymore

that means it is unclear

- where does next frame start?



2.3 Bit-Oriented Protocols



Use a special bit pattern as flag

- independent from encoding
- block definition

flag (0111110)

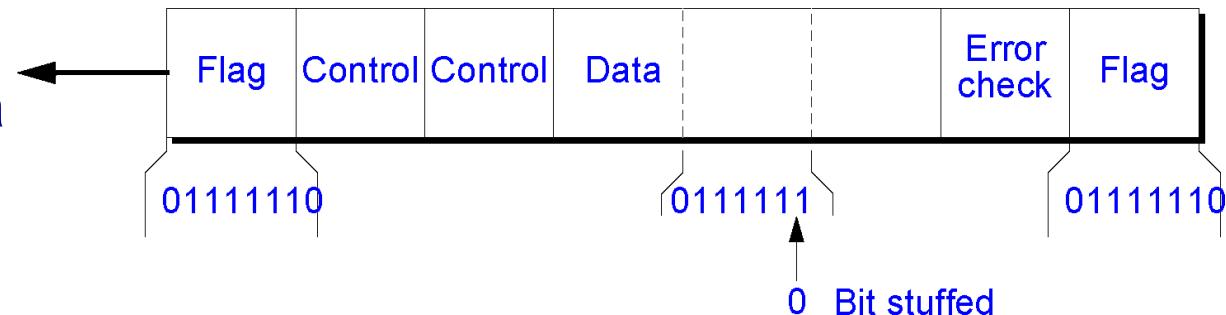
Start / end

- may be different flags
- but are typically identical

Bit-Oriented Protocols

PROBLEM:

- "flag" in user data
(e.g. 01111110)



SOLUTION:

- bit stuffing

(a) 0110111111 111111 11111 10010

(b) 01101111011111011111010010

(c) 0110111111111111111110010

SENDER

- inserts a "0" bit after 5 successive "1"s
(only in the user data stream)

RECEIVER

- suppresses “0” after 5 successive “1”

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

3 Error Detection and Correction

L1 is not perfect → errors may occur during transmission

Causes for errors:

- thermic noise:
 - electron movement generates background noise
- impulse disruptions (often last for 10 msec):
 - cause: glitches in electric lines, thunderstorms, etc.
- crosstalk in adjacent wires
- echo
- interference, e.g., other wireless / radio transmissions
- signal distortion (dampening depends on frequency)

Error Types:

- BIT ERROR: Modification of a single bit
- BURST ERROR: Modification of a sequence of bits

→ errors usually occur in bundles: BURST ERROR

Error Detection and Correction

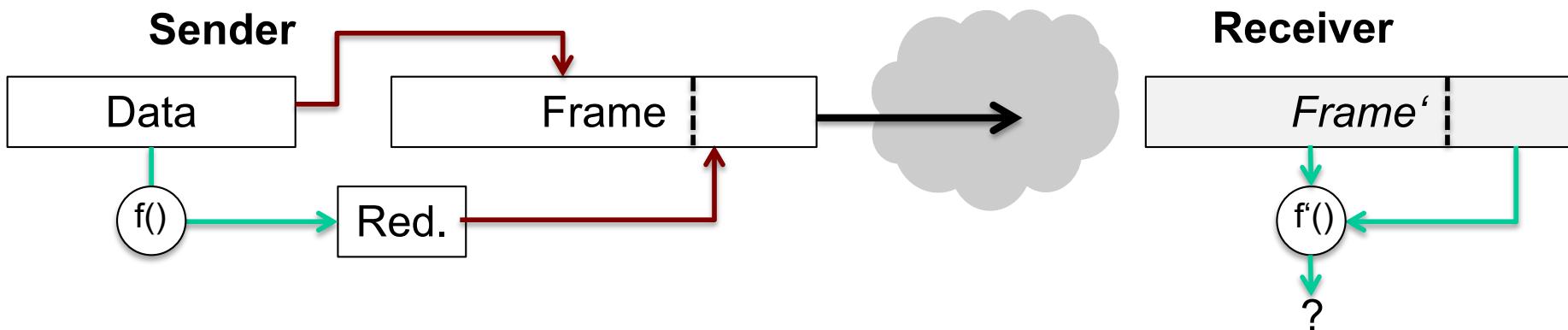
Two basic approaches

Error detection:

- insert redundancies so that receiver is able to **detect** an error
- **no** error correction: usage of separate method, if needed
 - e.g., retransmission

Error correction:

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



3.1 Basics: Code Word, Hamming Distance

Frame (= code word) contains

- data
- check information

Code = set of all valid code words

Hamming distance **d** of **two words** w1 and w2:

- number of bit positions by which w1 and w2 differ
- example:

$$\begin{array}{r} \text{w1} & 10001001 \\ \text{w2} & 10110001 \\ \text{XOR} & \\ & = 00111000 \end{array} \rightarrow d = 3$$

Hamming distance of a **code**:

- minimum Hamming distance **between any two words** of a code
- example:

$$\begin{array}{r} \text{w1} & 10001001 \\ \text{w2} & 10110001 \\ \text{w3} & 10110011 \end{array}$$

the Hamming distance $w2 \text{ XOR } w3 = 1$ is the smallest,
therefore, the Hamming distance of the code is $\rightarrow d = 1$

3.2 Detection & Correction (according to Hamming)

Hamming Distance determines

- a code's error detection and correction properties

1. DETECTION of f 1-bit errors:

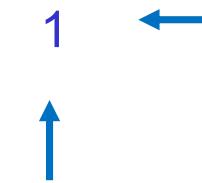
- example

1-dimensional

- example

2-dimensional

			parity bit p		parity bits last column and last row				
0	0	0	0	0	1	1	0	0	0
0	1	1	1	1	1	1	1	1	1
			1	0	0	0	0	0	1
			1	1	0	0	0	0	0
			0	0	0	0	1	1	1
			1	0	0	0	0	1	1



Detection and Correction (according to Hamming)

DETECTION of f 1-bit errors:

- if the Hamming distance d of code
 - $d \geq f + 1$
 - then f and less errors generate an invalid code word
- example:

p (parity bit)			
0	0	0	
0	1	1	
1	0	1	
1	1	0	→ $d = 2$

- i.e. maximum value for f : $f=1$
 - detection of 1-bit error

Detection and Correction (according to Hamming)

2. CORRECTION of f 1-bit errors:

- if the Hamming distance of code d
$$d \geq 2*f + 1$$
- f and less errors transcribe word w into an invalid word,
 - which is "closer" to w than to any other word

example:

- $d=5 \rightarrow f = 0, 1 \text{ or } 2$

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

- with two 1-bit errors ($f=2$) the following word may occur:

0	0	0	0	0	0	0	1	1	1
---	---	---	---	---	---	---	---	---	---

- the NEXT POSSIBLE WORD is

0	0	0	0	0	1	1	1	1	1
---	---	---	---	---	---	---	---	---	---

Error Correction: Some Properties

CORRECTION of f 1-bit errors:

- if Hamming distance of code is given by $d \geq 2f + 1$

Lower bound for the number of check-bits for correcting

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

m: #data bits; r: #check-bits

e. g.

$$\begin{array}{lll} m = 8 & \xrightarrow{\quad} & r = 4 \\ m = 1000 & \xrightarrow{\quad} & r = 10 \end{array}$$

Correction of burst errors (up to length k)

- treat k consecutive codewords as matrix
- transmission by columns

Properties:

- can be useful if only simplex communication is possible
- but high redundancy in each block

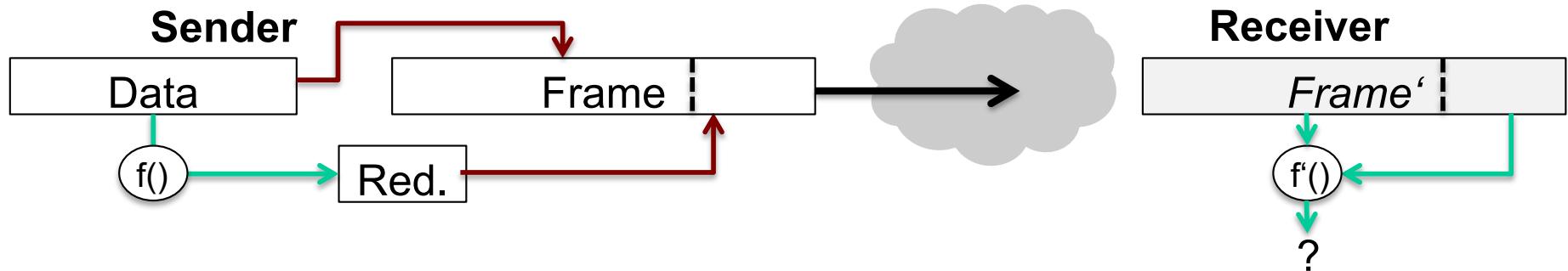
→ usually less efficient than error detection only
(depending on error probabilities and distributions)

Various codes have been developed

3.3 Error Detection

General Idea:

- Generate a code from user data on sender side
- Transmit user data plus code to receiver side
- Check on receiver side whether received data (user data and code) matches, i.e.,
locally calculated code == received code?



What type of code and how to generate it?

- Parity
- Checksum, e.g., used in IP, UDP, TCP
- Cyclic Redundancy Check (CRC)

Error Detection using CRCs

CYCLIC REDUNDANCY CHECK (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: $k_{n-1} \cdot x^{n-1} + k_{n-2} \cdot x^{n-2} + \dots + k_1 \cdot x^1 + k_0$
whereas $k_i = [0, 1]$

Example:

$$1 \ 1 \ 0 \ 0 \ 0 \ 1 \rightarrow x^5 + x^4 + 1$$

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

Error Detection

Algorithm

$B(x)$... 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)

Transmit the result.

Error Detection

www.ibr.cs.tu-bs.de

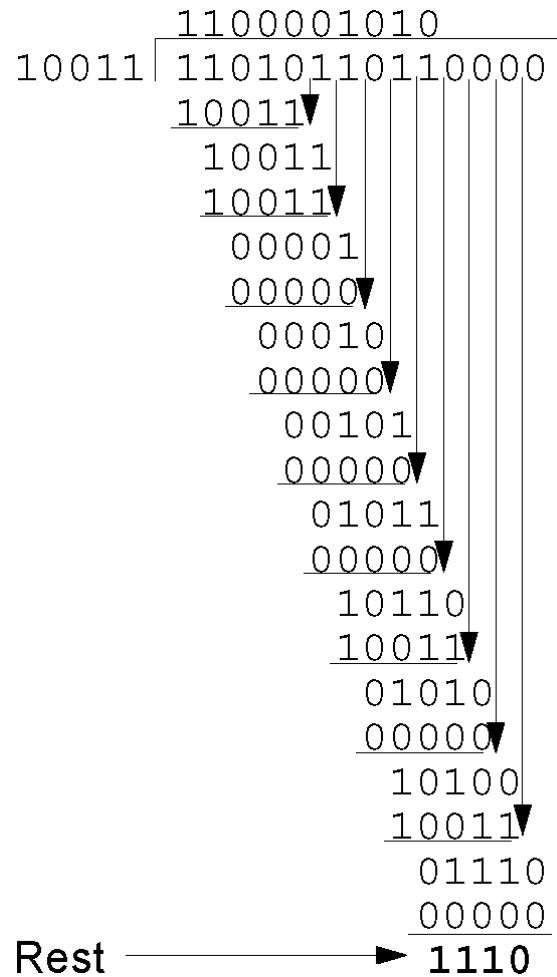
Sender	Receiver
	/*send block B*/ $B(x)/G(x) = Q(x) + R(x);$
	(B, R)
send ----->	receive;
	$(B(x) - R(x))/G(x) = Q(x) + R'(x)$
	if $R'(x) = 0$
	then Accept B
	else Reject B

Error Detection

Example: frame: 1101011011

Generator G(x), degree 4: 10011

Frame with 4 attached 0-bits: 11010110110000



Error Detection

Standardized polynomials:

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

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

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

CRC - CCITT recognizes

- all simplex 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

Implementation

- looks complicated but can be done using simple shift register in HW
- nearly all LANs use it

Assumptions & analyses

- have been made for long time assuming frames contain random bits
- recent work inspecting real data showed this to be wrong
 - undetected errors are more common than previously assumed

4 Flow Control and Error Treatment

Basic problem statement:

- sender can send faster than receiver can receive

→ Design mechanisms which can prevent this problem:

→ flow control

WITHOUT FLOW CONTROL:

- If sender can send faster than receiver can receive
- Then 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

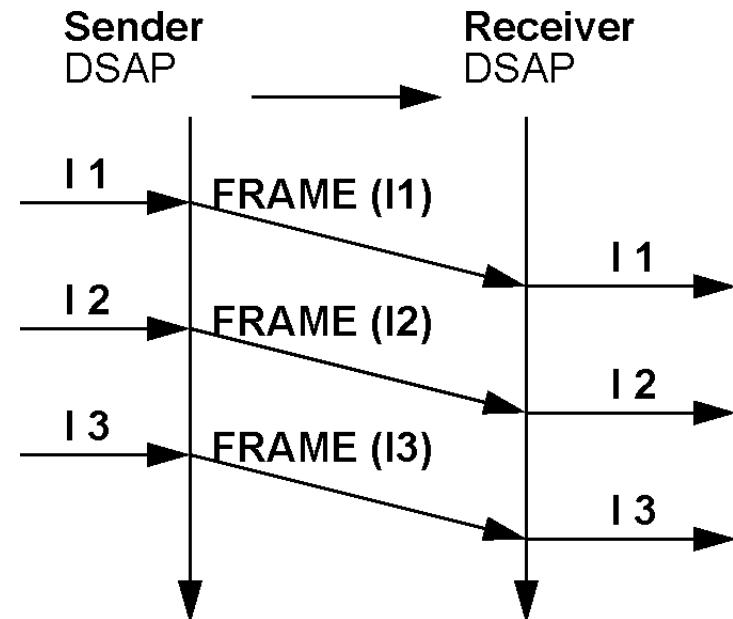
4.1 Protocol 1: Utopia

Assumptions:

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

DSAP: Data link (layer)
Service Access Point

- but: finite buffer, finite processor speed ...
→ sender floods receiver with data faster
than the receiver is able to process



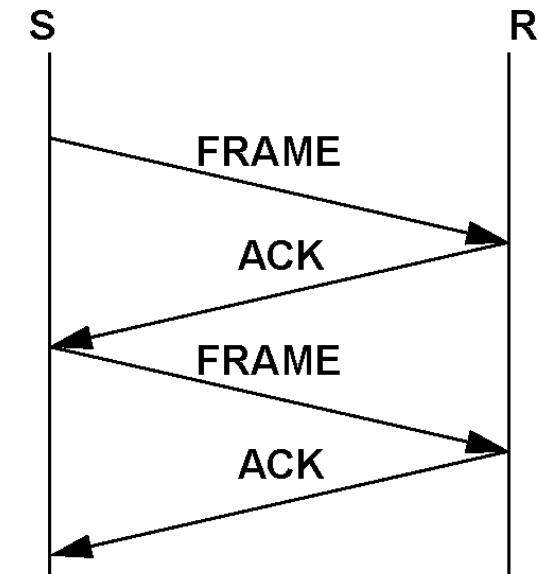
4.2 Protocol 2: Stop-and-Wait

Assumptions:

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

Flow control necessary: STOP-AND-WAIT

- receiver has buffer for one frame
- communication in both directions (frames, ACKs)



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

Problem:

- protocol 2 locks down during loss of either frames or ACKs

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

Relaxing assumptions further:

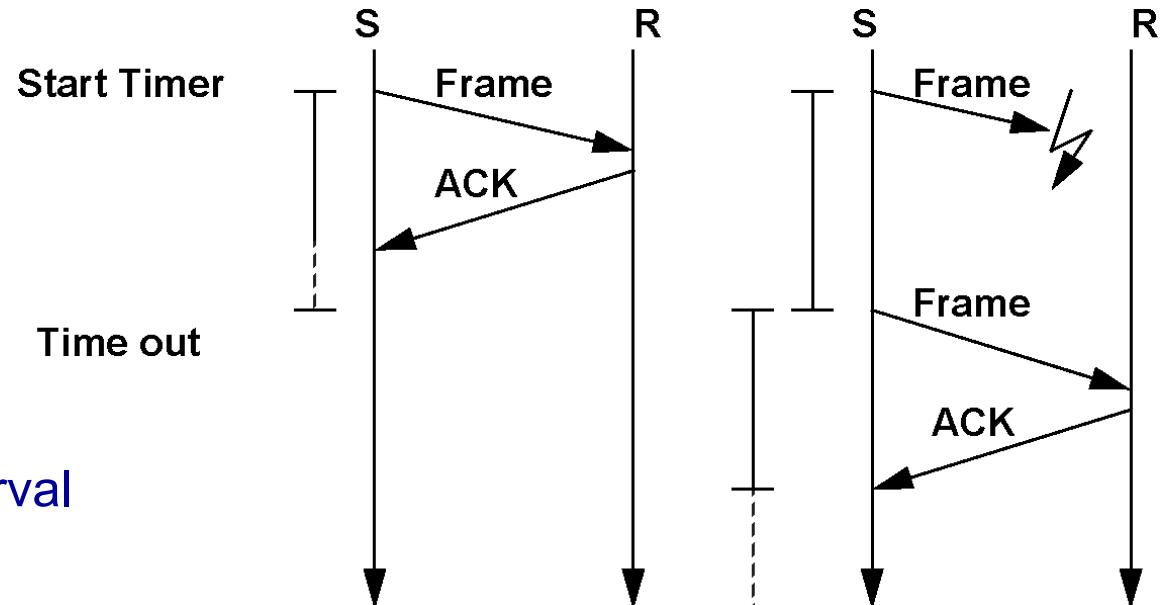
- NOT [error-free communication channel]
- NOT [infinitely large receiving buffer]
- NOT [receiving process infinitely fast]

Problem: protocol 2 locks down during loss of both frames and ACKs

Solution:

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

→ automatic retransmit
if no ACK is received
within certain time interval



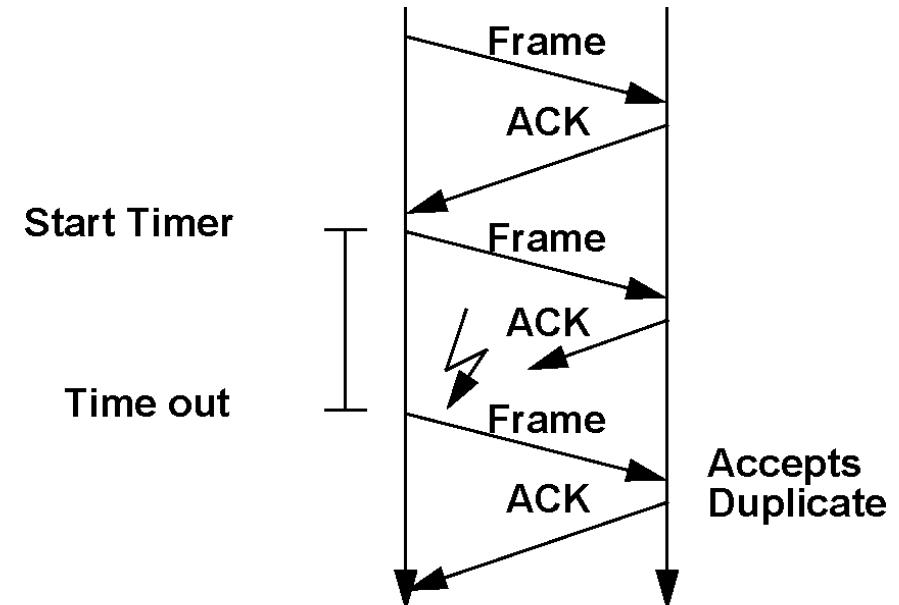
Timeout interval:

- If too short: unnecessary retransmission of frames
- If too long: unnecessary / too long waiting in case of error

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

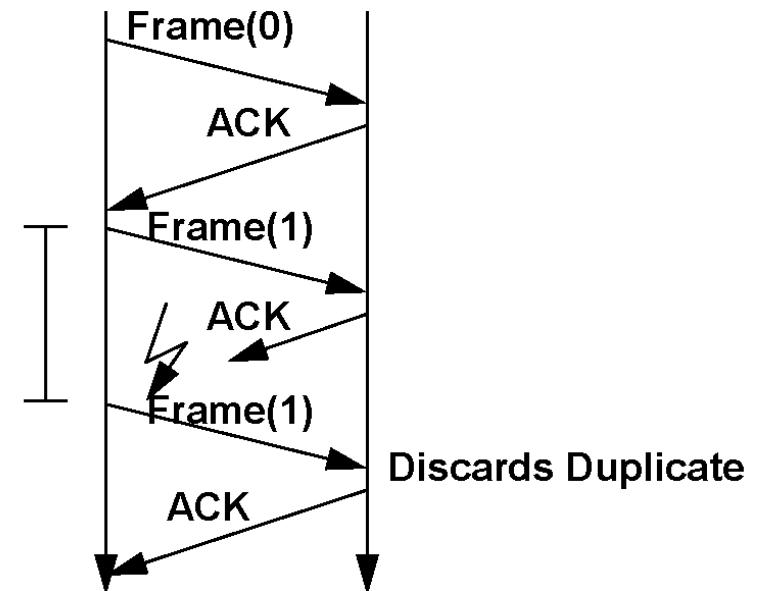
Problem:

- loss of ACKs leads to duplication



Solution: sequence numbers

- each block contains a sequence number
- sequence number is kept during retransmissions
- range
 - in general: $[0, \dots, k]$, $k=2^n-1$
 - Stop-and-Wait: 0, 1
 - n: window size



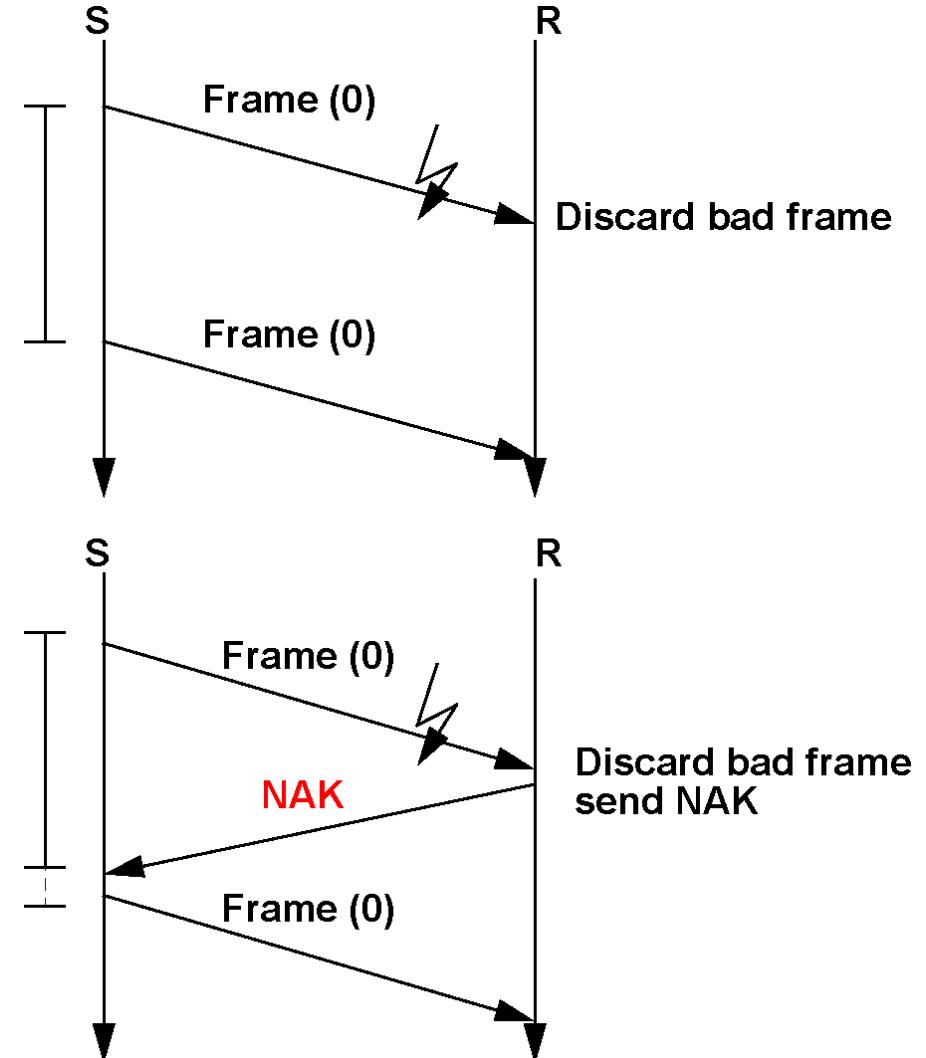
4.5 Protocol 3c: Stop-and-Wait / NAK+ACK / SeqNo

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 / NAK+ACK / SeqNo

1. Situation: OK

Frame correctly transmitted

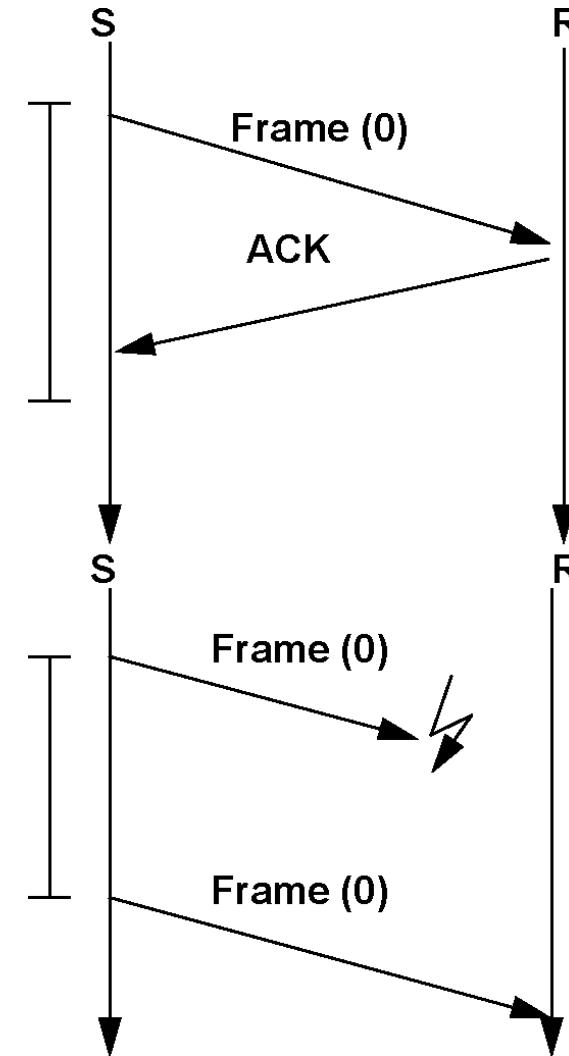
→ ACK sent

2. Situation:

Frame loss on path from
sender to receiver

- frame did not arrive

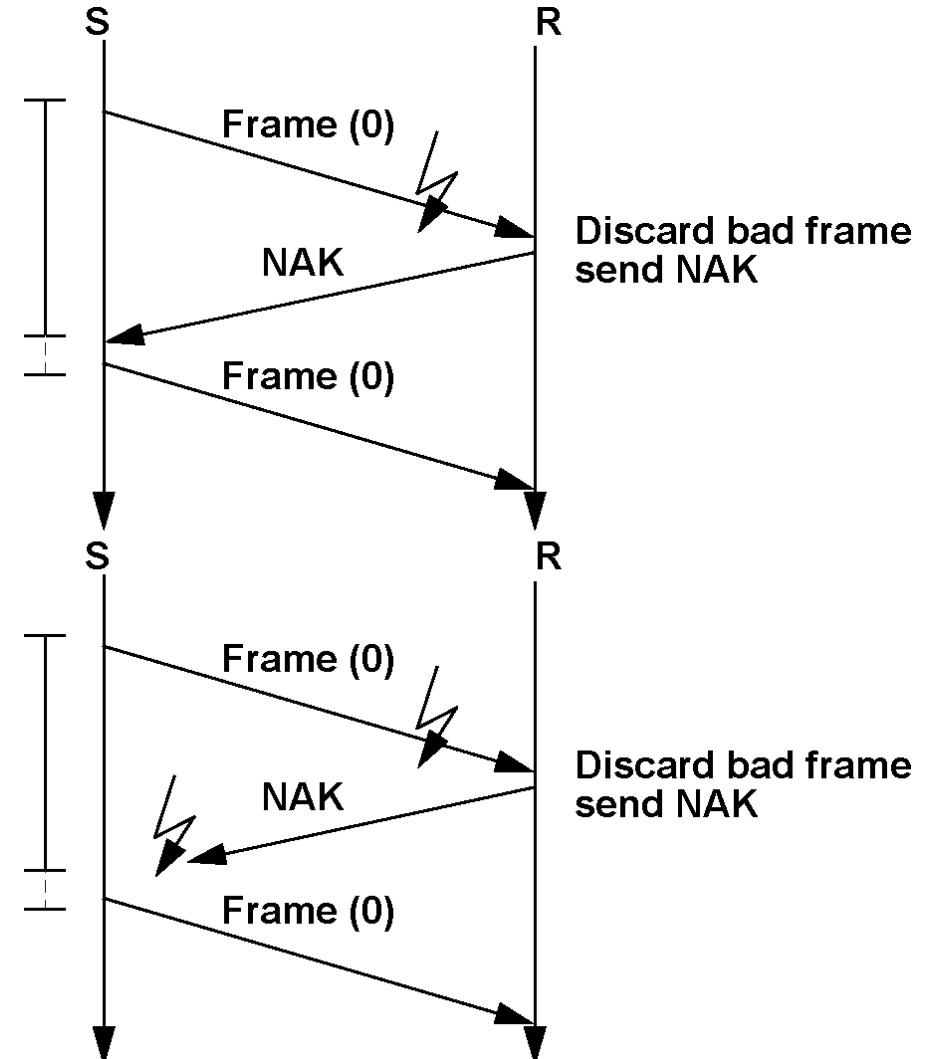
→ timer issues retransmission



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

3. Situation:

- Frame gets corrupted on path from sender to receiver
- faulty frame arrives
→ NAK issued

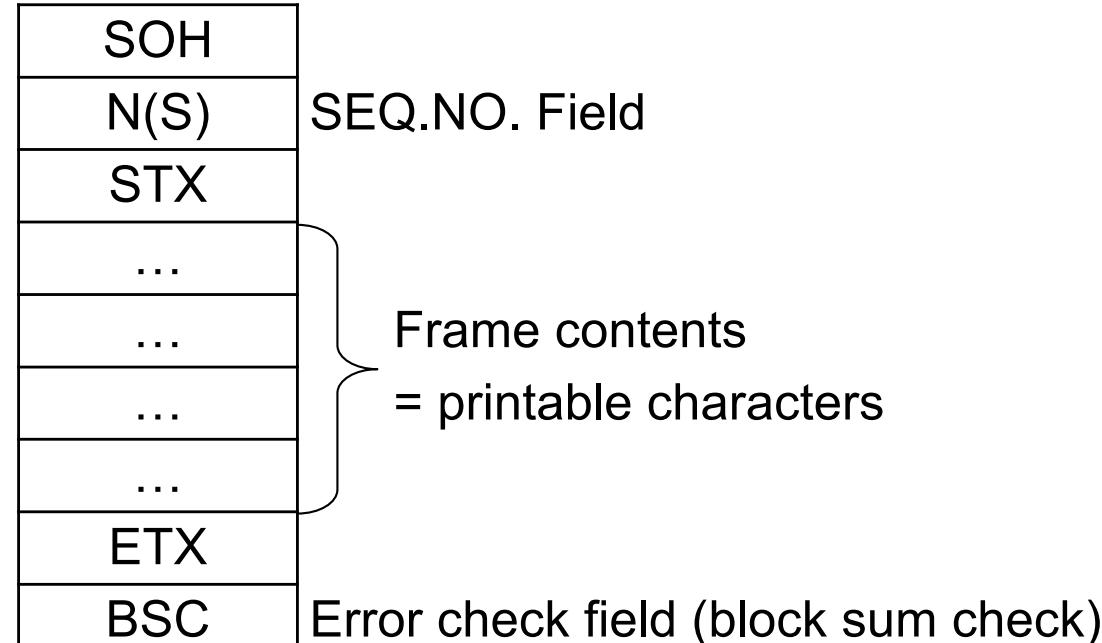


4. Situation:

- on path receiver to sender
- NAK issued
 - but,
 - NAK does not arrive
 - or
 - NAK arrives damaged
→ timer issues retransmission

4.6 Example of Frame Formats

Example: using a character oriented protocol



I-frame format (information, data frame)



ACK-frame format



NAK-frame format

5 Sliding Window – Flow Control & Error Treatment

5.1 Channel Utilization and Propagation Delay

5.2 Sliding Window: Concept

5.1 Channel Utilization and Propagation Delay

Stop-and-Wait:

- Basic approach: send one frame and wait
→ poor utilization of the channel

Example: satellite channel

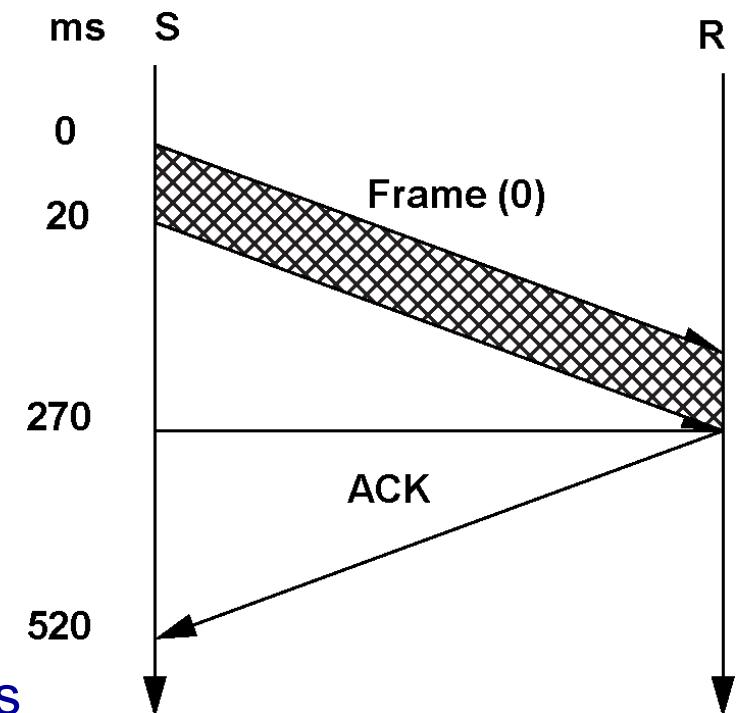
- transmission rate: 50 kbps
- roundtrip delay 500 ms ($2 \cdot 250$ ms)
- frame size: 1000 bit
- in comparison to that,
ACK is short and negligible

this means

- Sending one frame needs
 $1000 \text{ bit} / 50000 \text{ bps} = 20 \text{ ms}$
- sender is blocked for 500 ms of 520 ms

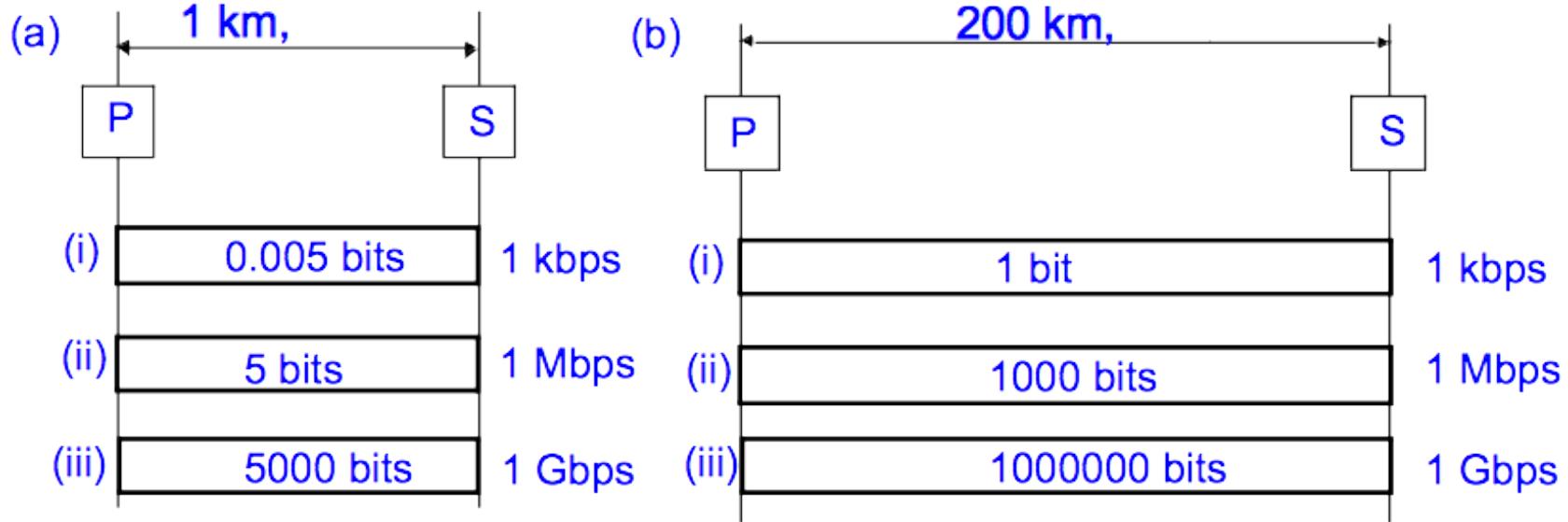
→ Channel utilization < 4%

- channel ‘effectively used’: 20 ms
- channel busy: 520 ms

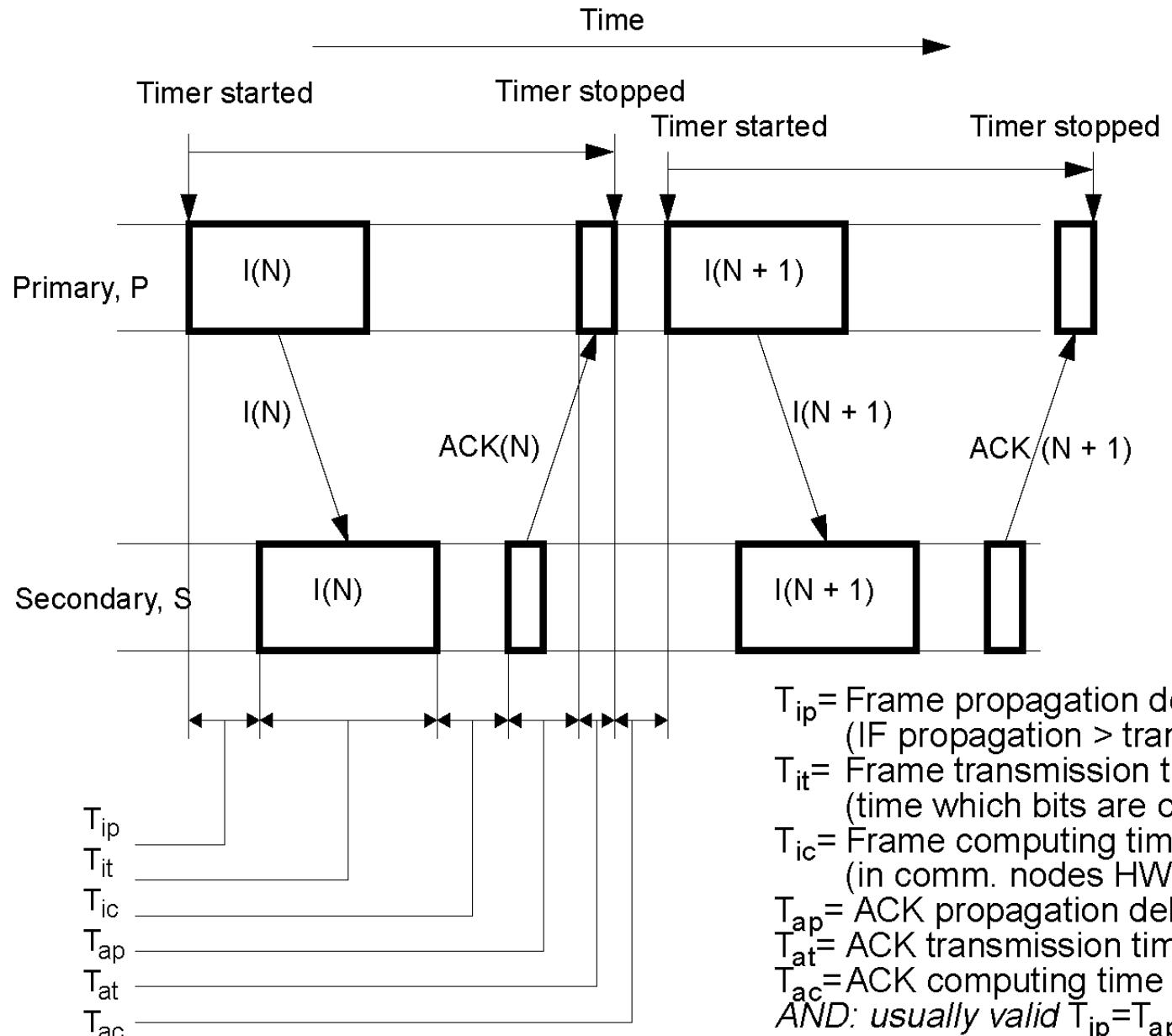


Channel Utilization and Propagation Delay

Storage capacity of a network link:



Channel Utilization and Propagation Delay



Channel Utilization and Propagation Delay

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:

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

- with the assumption

$T_{ip} = T_{ap} = T_p$

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 in which bits are on channel)

T_{ic} = Frame computing time in S (in communication nodes HW & SW)

T_{ap} = ACK propagation delay

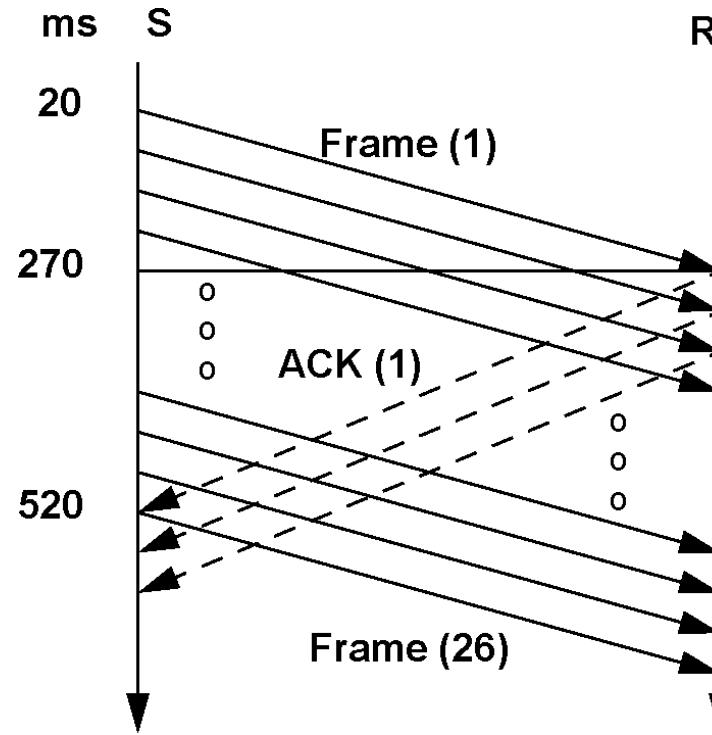
T_{at} = ACK transmission time

T_{ac} = ACK computing time in P

Channel utilization

Stop-and-Wait leads to poor utilization of the channel!
→ Better approach needed

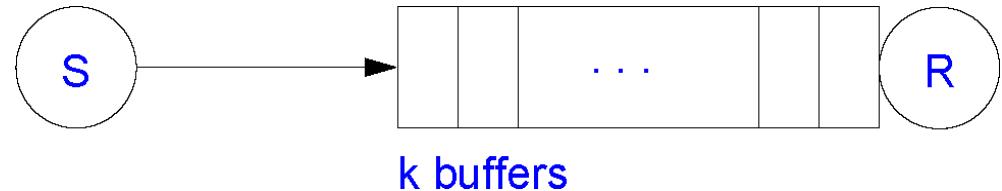
Solution: pipelining



Flow control: sliding window mechanism

5.2 Sliding Window: Concept

Flow control:
receiving buffer
must not be flooded



Sender and receiver window per connection

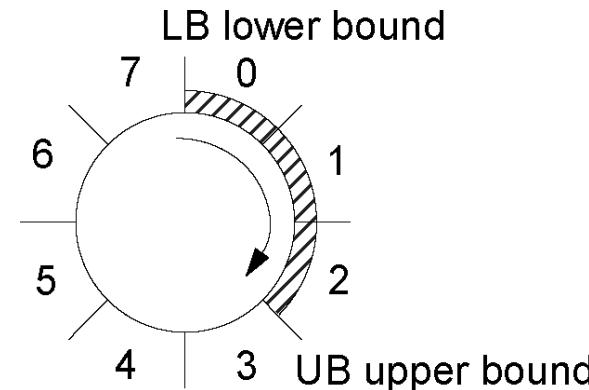
R-WINDOW:

- sequence numbers, which can be accepted

S-WINDOW:

- sequence numbers, which were sent but not yet acknowledged

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

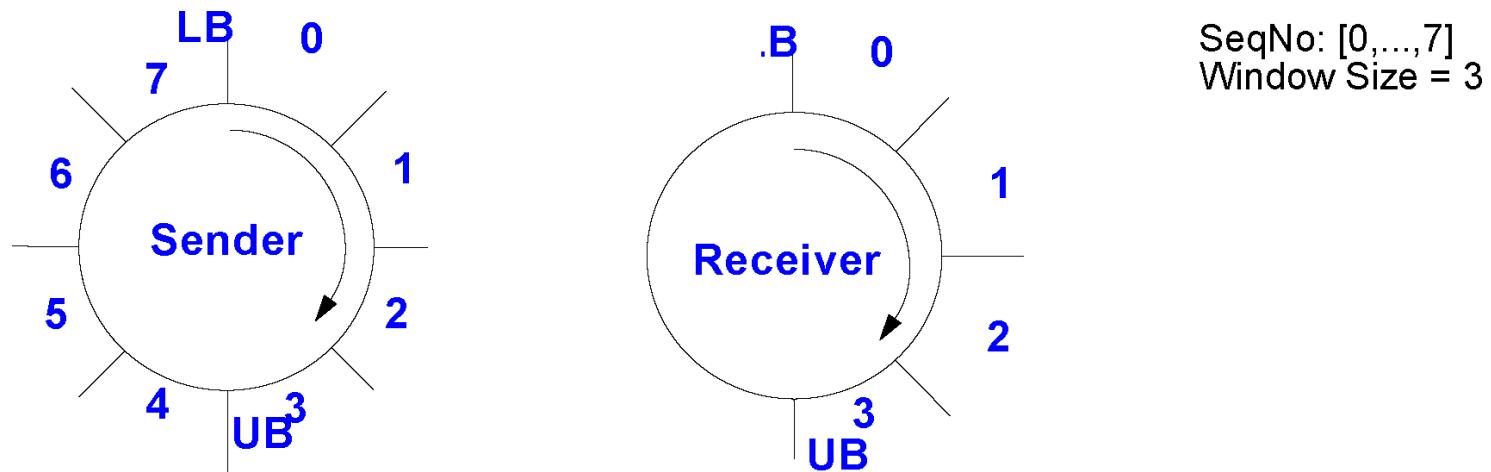


Initial window size:

- R-Window:
- S-Window:

number of buffers reserved
maximum number of blocks, which may still
be open for acknowledgement

Sliding Window: Concept



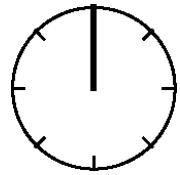
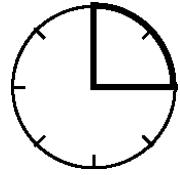
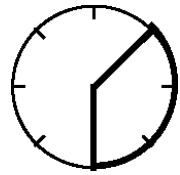
Lower Bound & Upper Bound

	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	on reception of an ACK	on reception of a frame
UB	when sending of a frame	when sending of an ACK

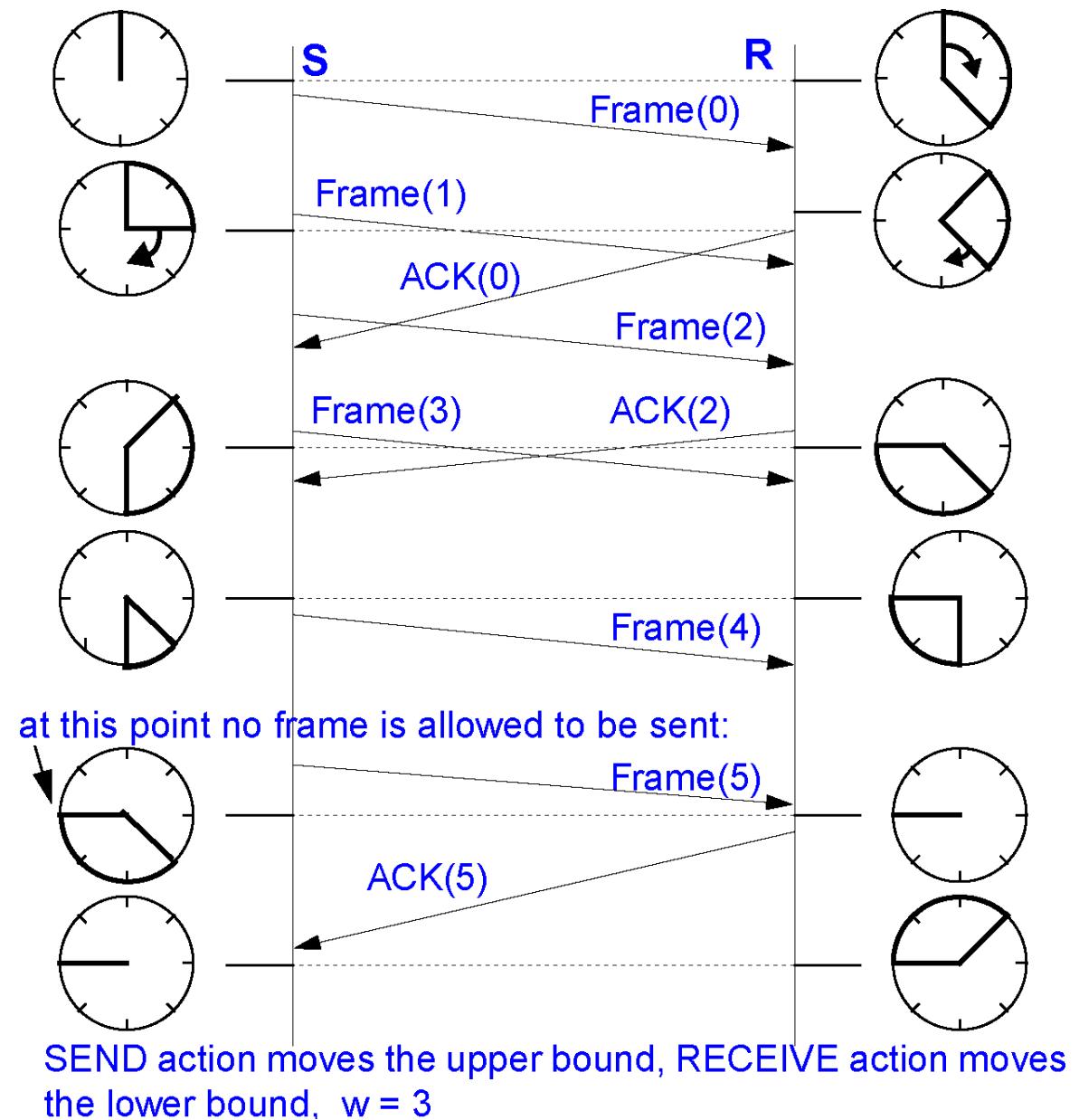
Sliding Window: Example (window size=3)

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

Example Including Acknowledgement

Including Acknowledgement

- ACKs contain SeqNo
- that means $\text{ACK}(\text{SeqNo})$ confirms all frames(SeqNo') with
 - $\text{SeqNo}' = \text{SeqNo}$ and
 - $\text{SeqNo}' < \text{SeqNo}$



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 ($n > 1$)
 - sequence not necessarily correct
 - but, amount of "displacement" 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
- window size

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

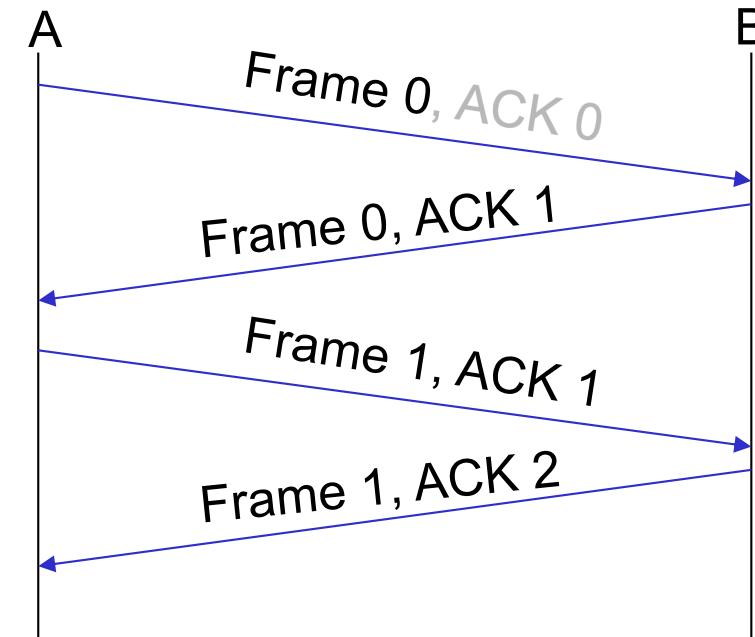
6.2 Sliding Window: Piggybacking

Idea:

- if both stations have data to exchange (full-duplex operation) then ACK can be included into data frame to avoid separate frame → reduce number of transmitted frames

Frames may contain data and ACKs, i.e., separate seq.no.

Frame (SeqNo, ACK-SeqNo, ... Data ...)

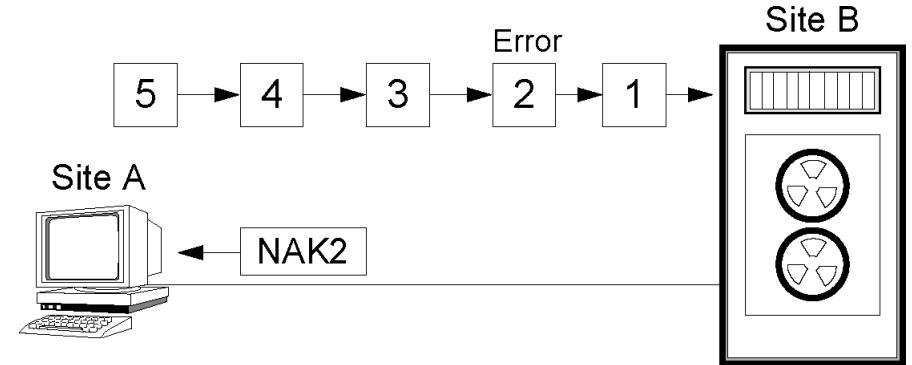


- assumptions in this example
 - the initial SeqNo. is 0
 - the next expected number is given as ACK-SeqNo

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

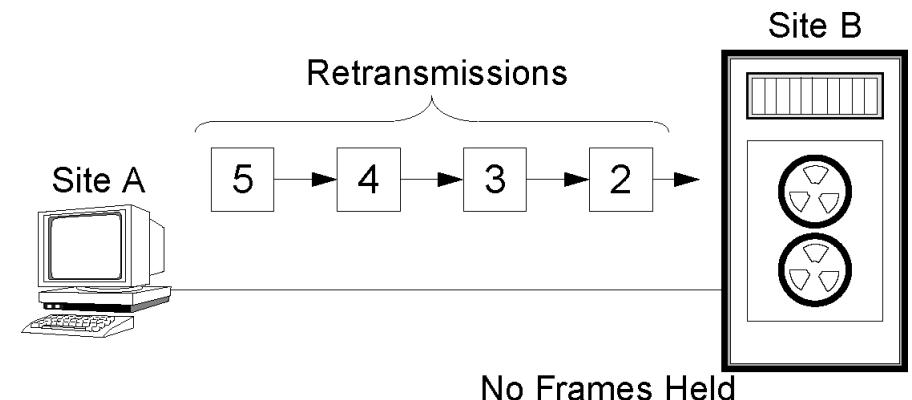
Procedure

- after a **faulty frame** has been received, receiver **drops all** further frames until **correct frame** has been received



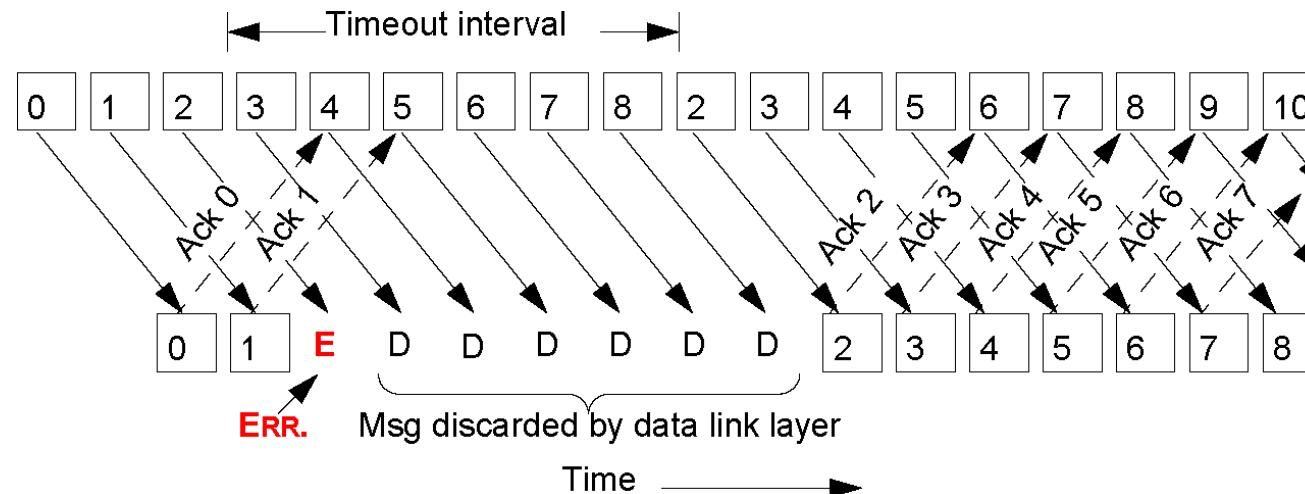
Evaluation

- Simple
- no buffering of “out-of sequence” frames needed
 - only for optimization purposes
- poor throughput

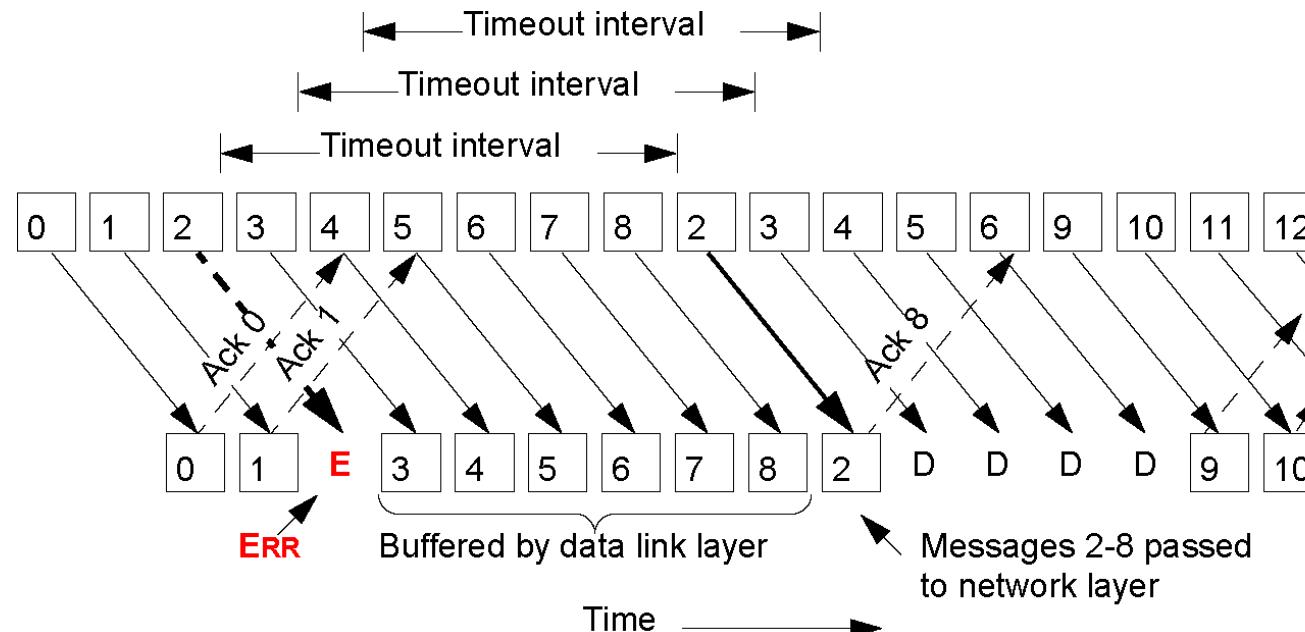


Sliding Window: Go-Back-N (Error Treatment)

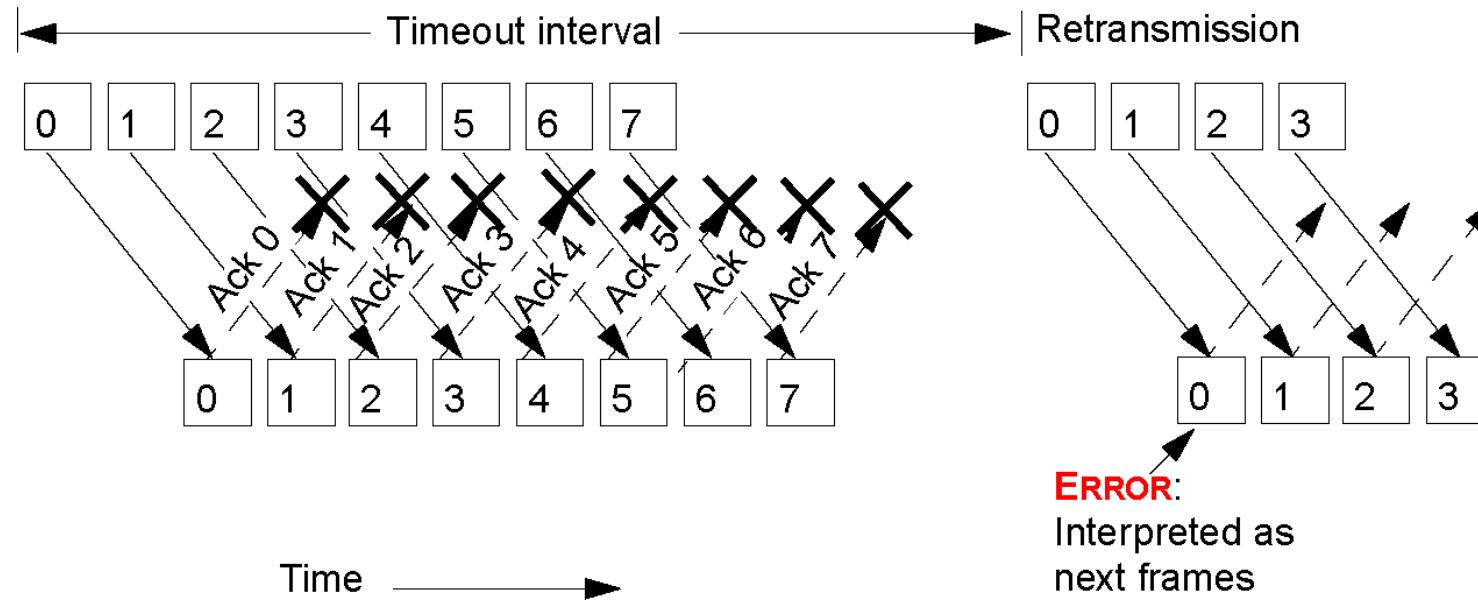
Example: sender: error detection by timeout



Optimization



Sliding Window: Maximum Window Size



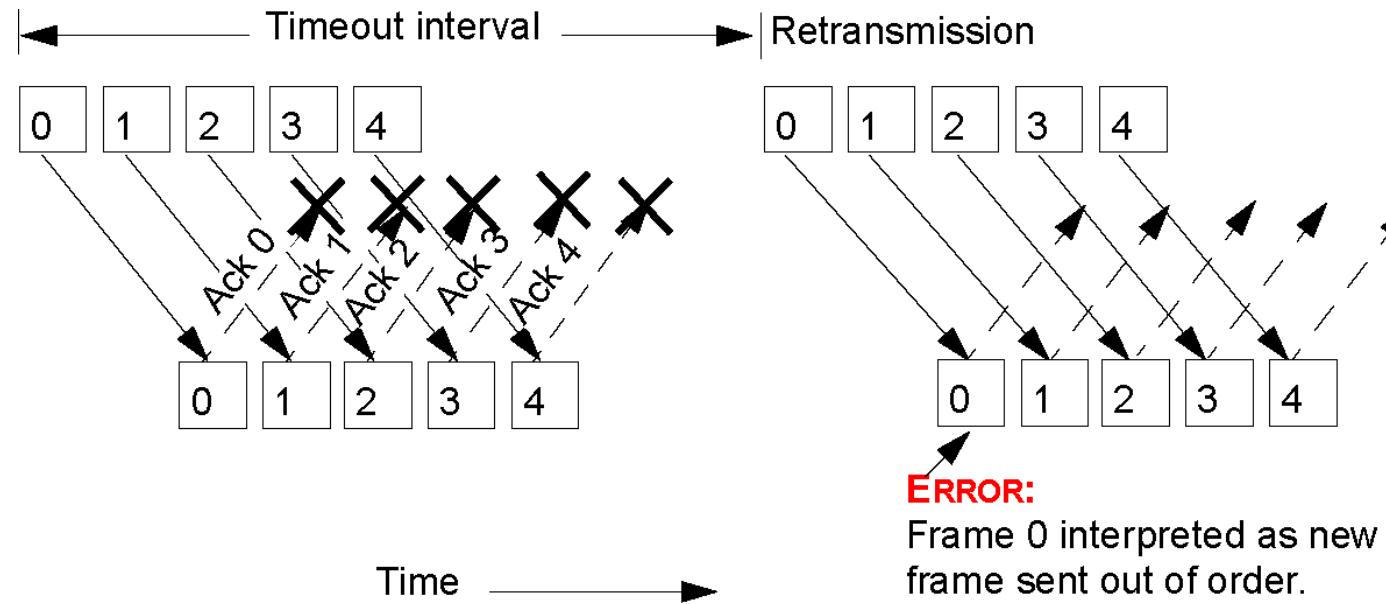
Example:

- 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 \leq range of sequence numbers

Sliding Window: Maximum Window



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

- 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 $\leq 1/2$ range of sequence numbers

6.4 Sliding Wind.: 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 has a lot of frames in its buffer
 - and transfers frames in correct sequence from L2 to L3

Comments

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

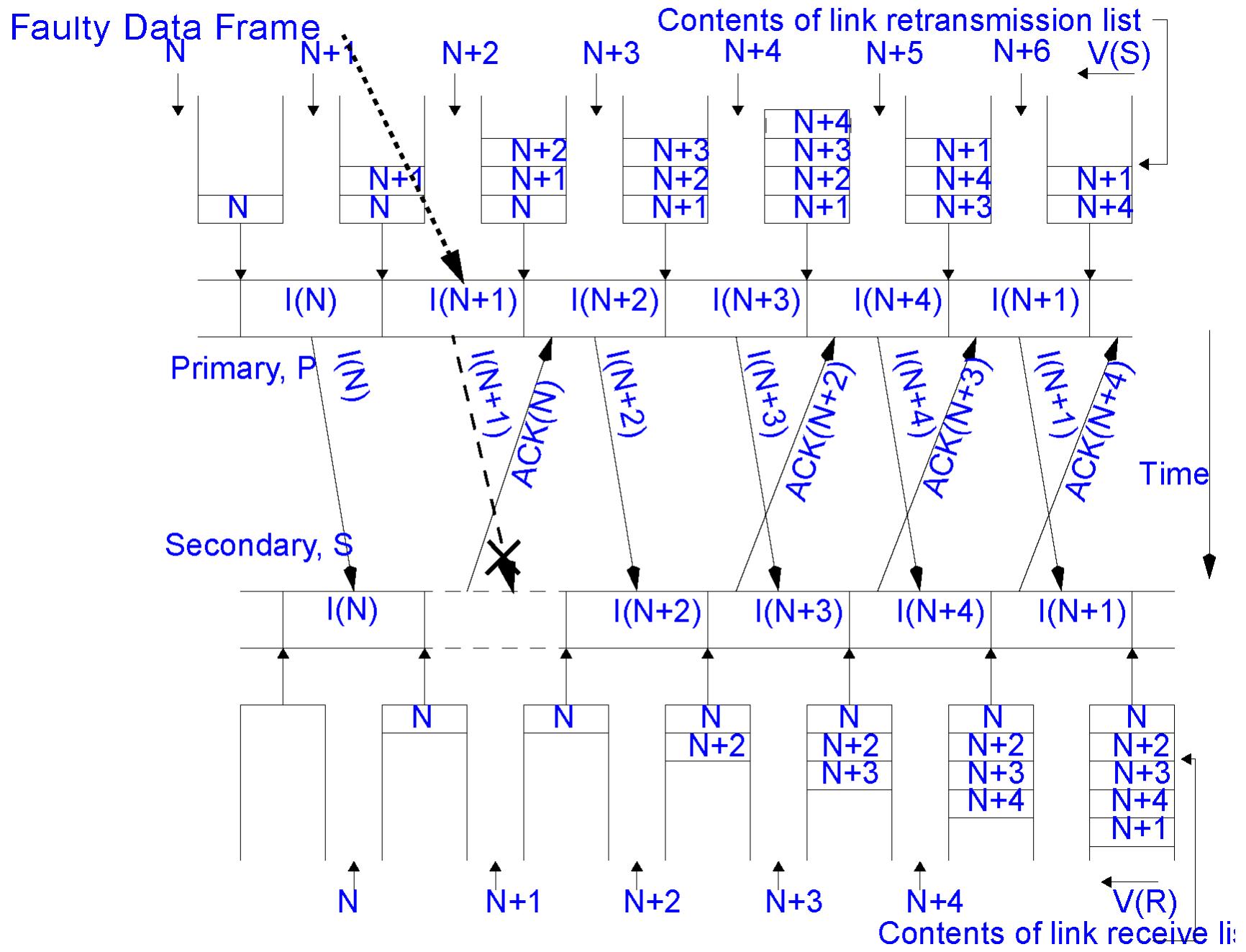
Sliding Window: Selective Repeat

Features

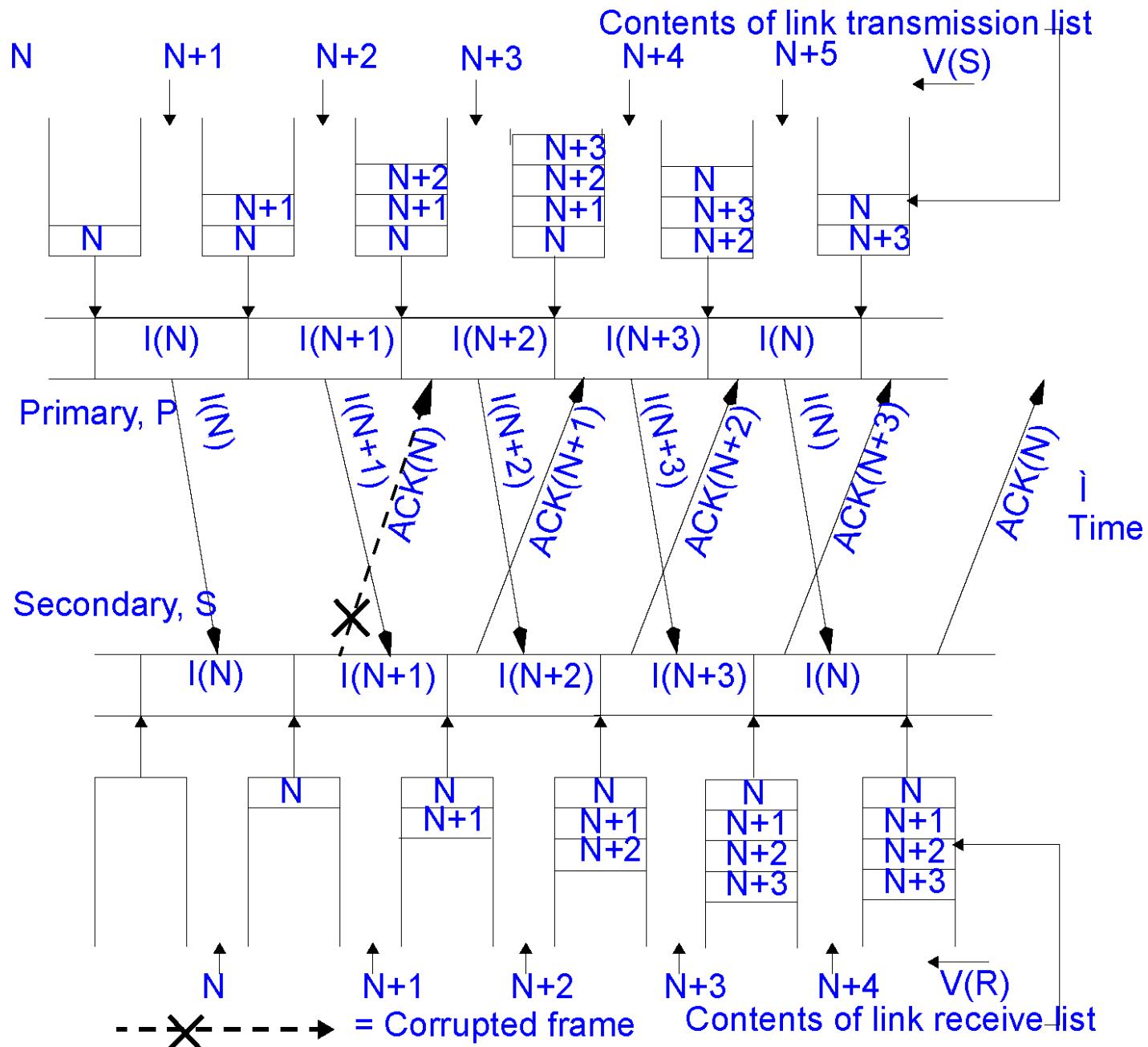
- more complex
- buffering of “out of sequence” frames
- increased throughput
- no cumulative acknowledgements

Example:

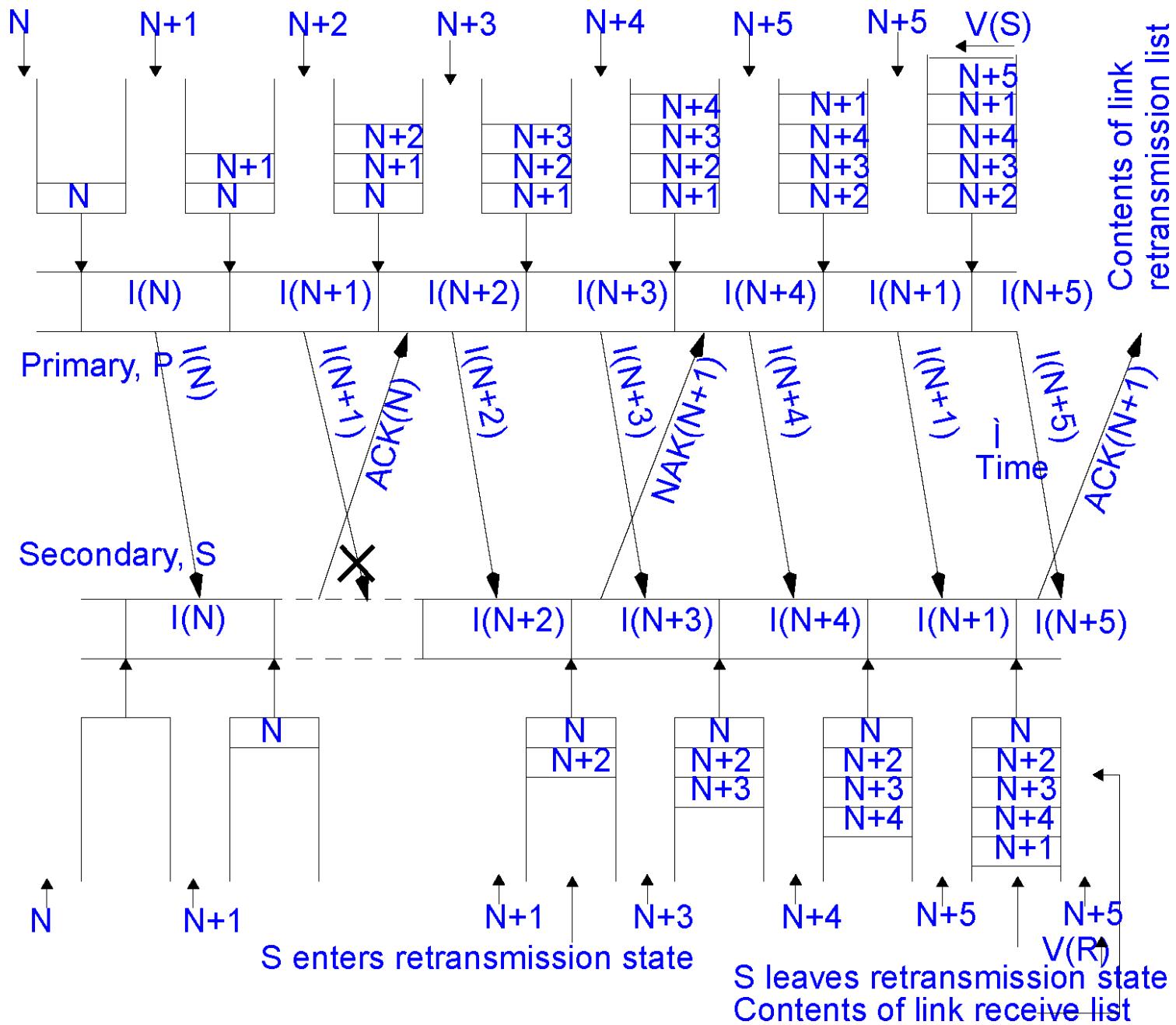
Faulty Data Frame



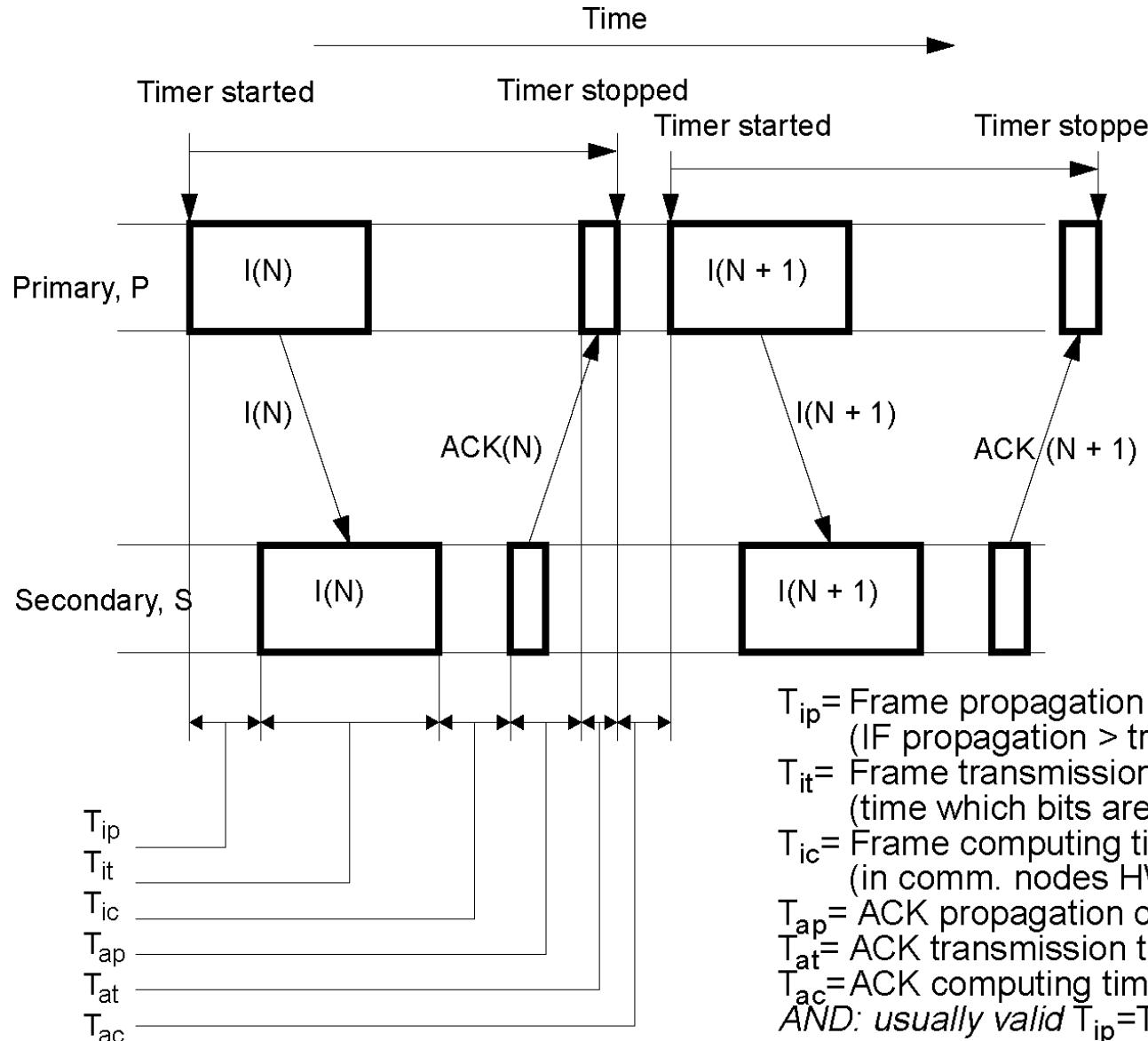
Faulty Acknowledge Frame



Active Error Control



6.5 Channel Utilization



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:

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

- with the assumption

$$T_{ip} = T_{ap} = T_p$$

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 in which bits are on channel)

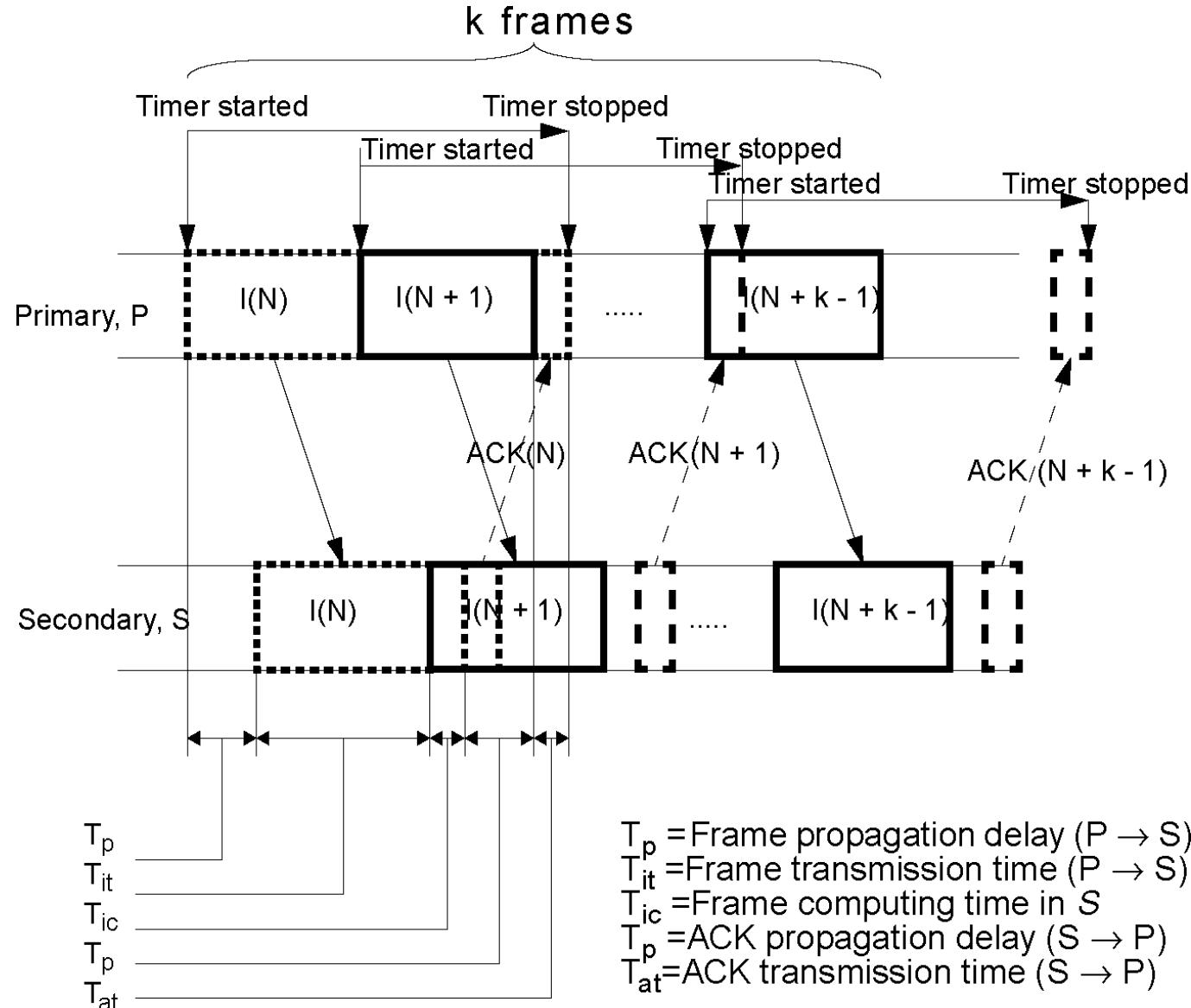
T_{ic} = Frame computing time in S (in comm. nodes HW & SW)

T_{ap} = ACK propagation delay

T_{at} = ACK transmission time

T_{ac} = ACK computing time in P

Channel Utilization with Sliding Window



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

Channel Utilization

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

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

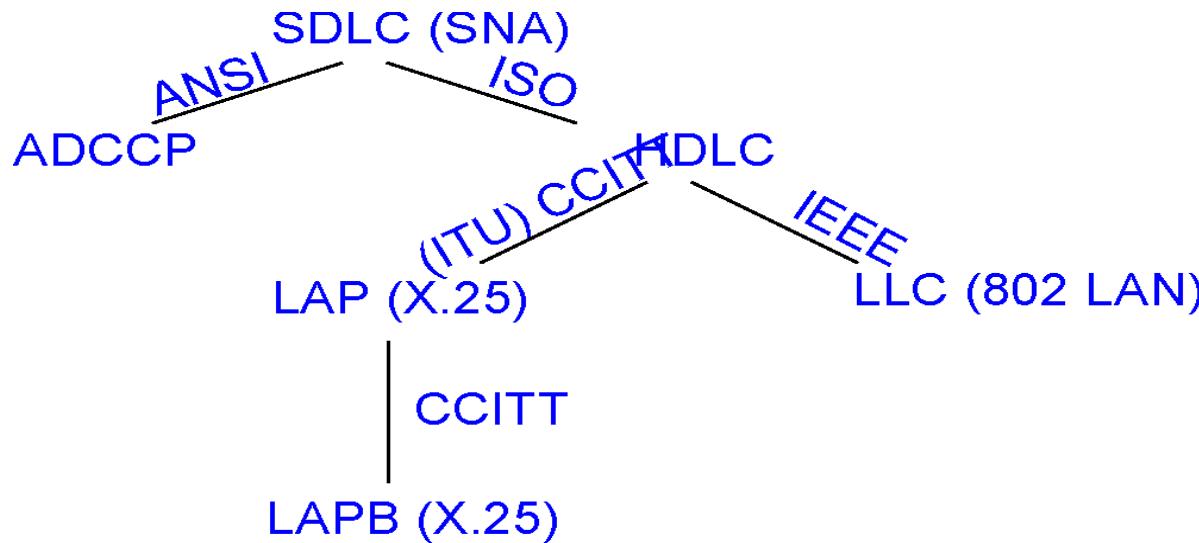
Stop-and-Wait

- + 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

7 Example: HDLC Family



- 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]

7.1 HDLC: Principle

Detailed description: standards, [black]

- a.o. ISO 3309 HDLC Procedures - Frame Structure

Protocol: bit oriented, full duplex

Data transparency: bit stuffing (5* “1” always followed by “0”)

Frame format (L2-PDU):

- address: addressing of stations:
 - important if several nodes on level L1
 - if point-to-point - L1:
 - sometimes used to differentiate between command and response
- control: sequence numbers, ACKs, ... (see below)
- data: any user data of any length
- checksum: Frame Check Sequence (FCS), variation of CRC

Bits	8	8	8	#(n*8)	16	8
	01111110	Address	Control	Data	Checksum	01111110

7.2 Three Types of frames

(they differ in the control field)

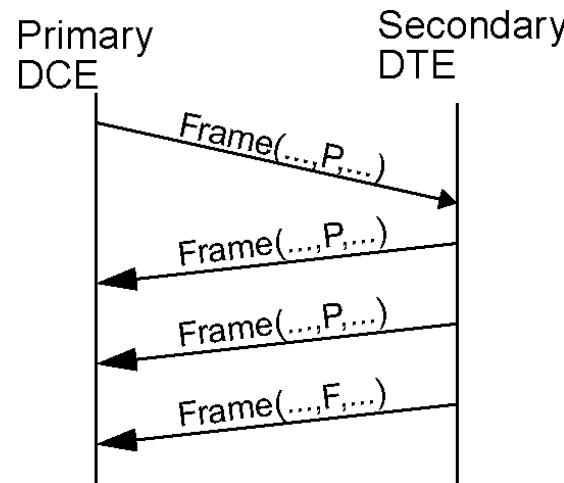
1. Information for data transmission
2. Supervisory for control management during data transfer
3. Unnumbered for connection management

HDLC: Information Frame

Control field:	Bits	1	3	1	3
		0	Seq	P/F	Next

Sliding window, 3-bit sequence number

- Seq: sequence number of the frame
- Next: sequence number of the frame to be expected next
(instead of the last correctly received frame as described earlier)
 - in form of “Piggybacking”
- P/F: Poll/Final, mainly used for polling
 - DCE (computer) requests DTEs to transmit data
 - DTE transmits (P-bit setting, last frame with F-bit)



HDLC: Supervisory Frame

Control field:

1	0	TYPE	P/F	Next
---	---	------	-----	------

TYPES:

- Type 00: RECEIVE READY (actually ACK)
 - explicit acknowledgement
 - NEXT: next expected frame
- Type 01: REJECT (actually NAK)
 - negative acknowledgement if transmission error
 - NEXT: first frame to be retransmitted
 - “Go-back-N” method (retransmit all up to faulty frame)
- Type 10: RECEIVE NOT READY (actually ACK & STOP)
 - reports that receiver has a temporary problem
 - cease acknowledgement and transmission
 - next: frame to be expected
 - re-activate transmission: RECEIVE READY, REJECT,...
- Type 11: SELECTIVE REJECT
 - retransmission of a frame (but not all previous frames)
 - next: frame to be retransmitted
 - “Selective Repeat” method

HDLC: Unnumbered Frame

Control field:

1	1	TYPE	P/F	MODIFIER
---	---	------	-----	----------

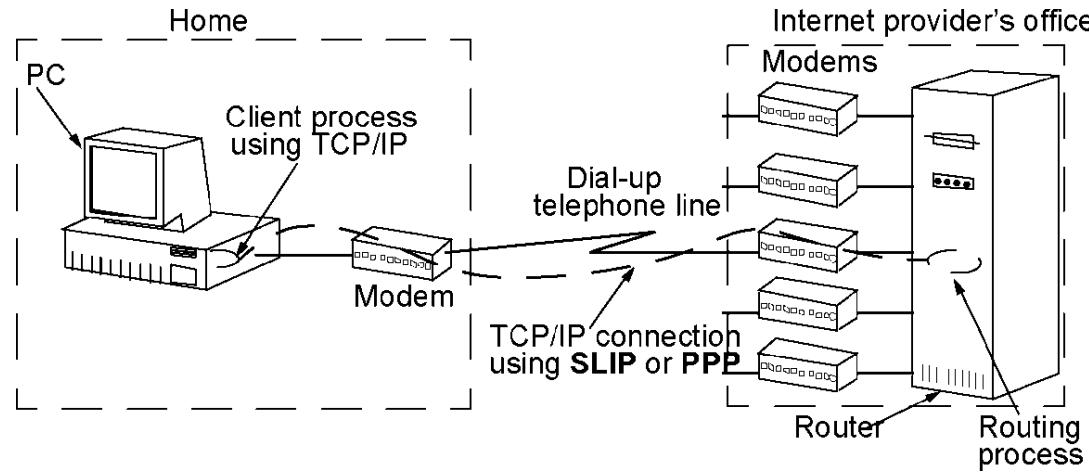
Application:

- control data
- unsecured connectionless service

TYPES (type, modifier) a.o.

- DISC (Disconnect):
 - reports inavailability (e.g. during preventive maintenance)
- SNRM (Set Normal Response Mode):
 - reports availability (Set SeqNo. = 0)
(Seq.No's. are reset to 0)
 - unbalanced (primary/secondary master/slave) from “old” times
- SABM (Set Asynchronous Balanced Mode):
 - reports availability (Set SeqNo. = 0) balanced (peer-to-peer) from “more recent” times
- FRMR (Frame Reject):
 - frame drop if protocol infraction
e.g. frame < 32 bit, ACK from frames with invalid SeqNo's
- UA (Unnumbered Acknowledgement):
 - safeguards against loss/error of unnumbered frames
 - ACK, acknowledgement for unnumbered frames

8 Example: Internet-related Layer 2



Classical 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.,...)
 - 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)
- replaces 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)

Features

- packet framing
 - supports several L3 protocols/services, not just IP
- bit transparency:
 - carry any bit pattern in data field
- error detection
 - not correction
- connection liveness
- network layer address negotiation
 - e.g., IP addresses determined dynamically
- authentication

Internet: Point-To-Point Protocol (PPP)

Bytes:

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

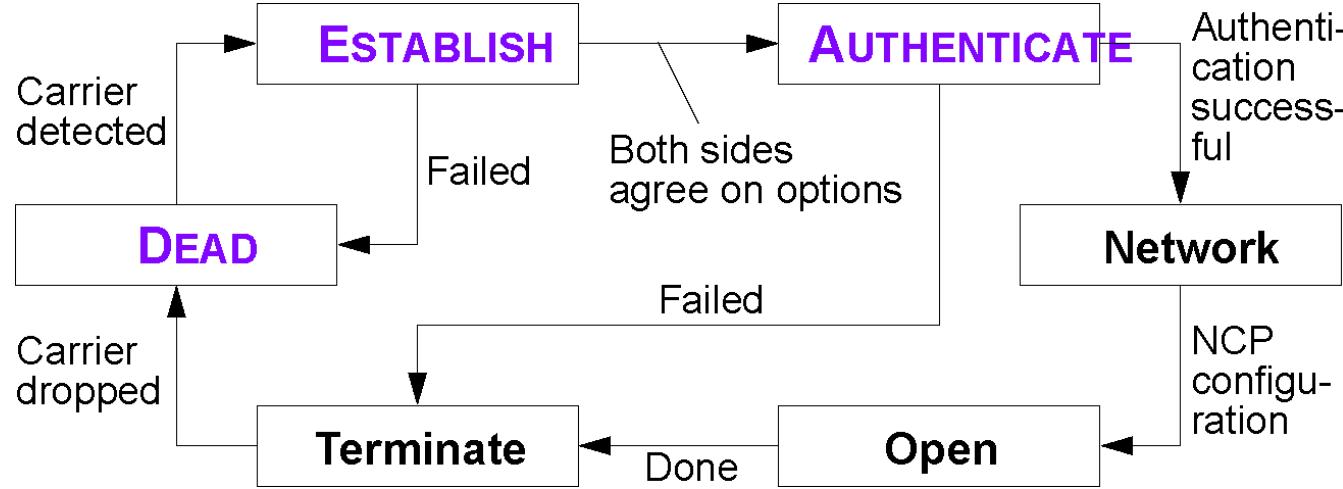
Frame Format (similar to or derived from HDLC):

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

(Comment: 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: boundary of the L2 frame

Internet: Point-To-Point Protocol (PPP)



Dead

- L1 connection does not existent

Establish

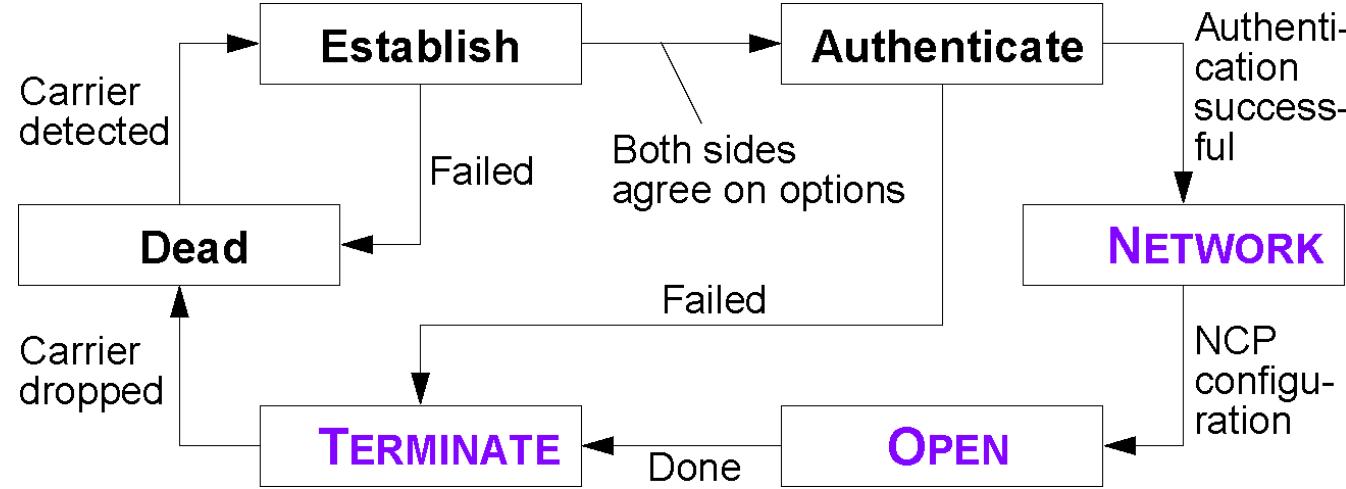
- after L1 connect
- negotiation of LCP options

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