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
  - ’malign’ features of the L1 service (& the communication channel)
    - finite propagation speed
      - between sending and receiving operations at L2
    - limited data rate

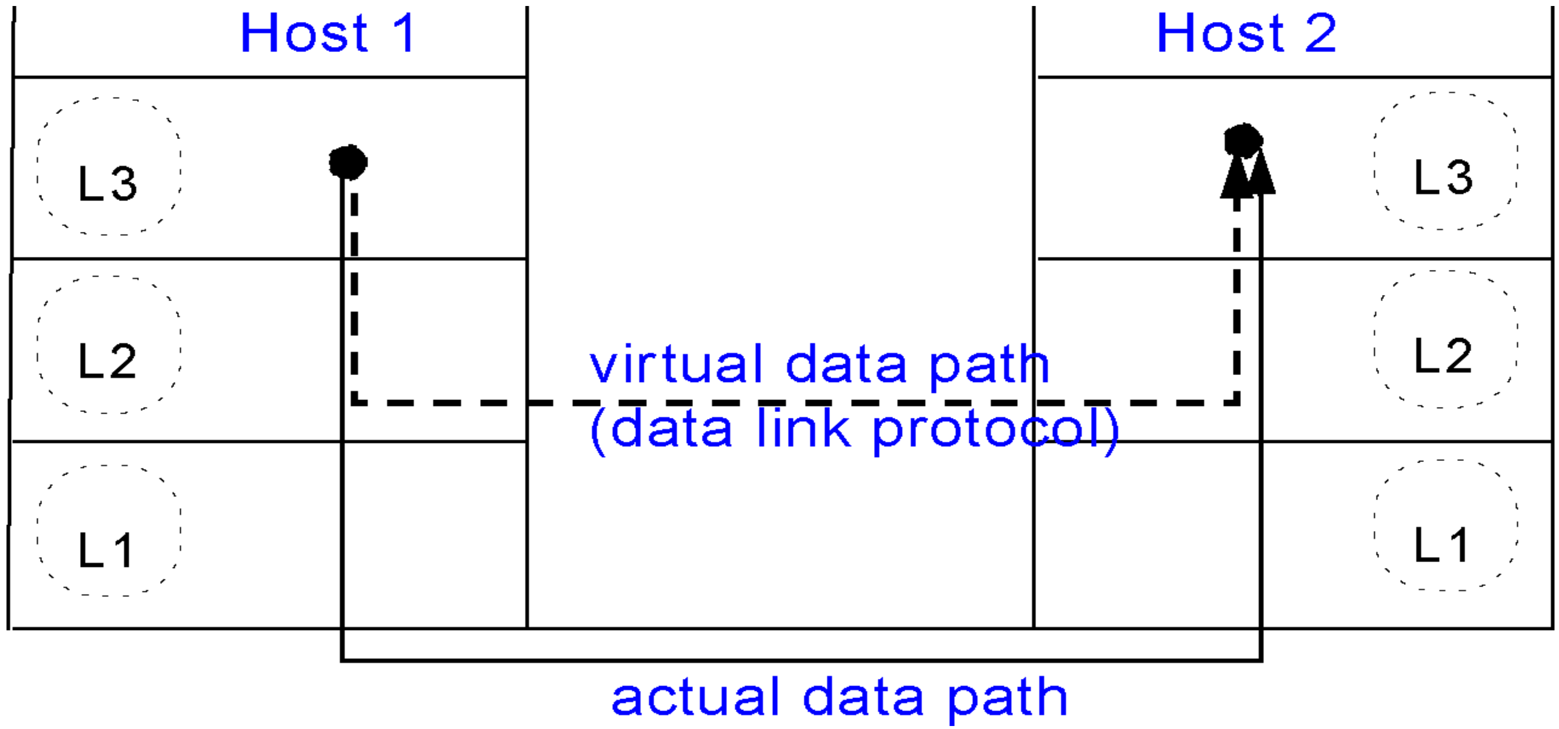
  → i. e. loss, insertion and changing of bits possible

- L2 Service:
  - (reliable), efficient data transfer between ADJACENT stations
    - may be between more than 2 stations
    - adjacent = connected by one physical channel

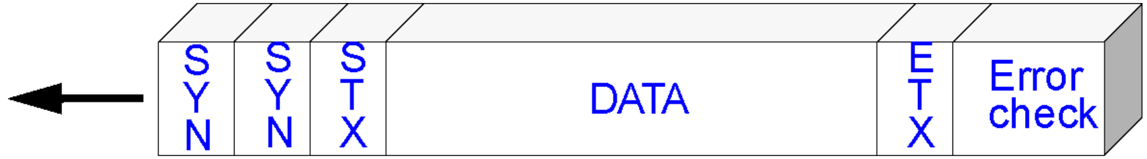
- L2 Functions:
  - data transmission as FRAMES
  - ERROR control and correction
  - FLOW CONTROL of the frames
  - configuration management

Services

Actual data path and virtual data path:



Character Oriented Protocols

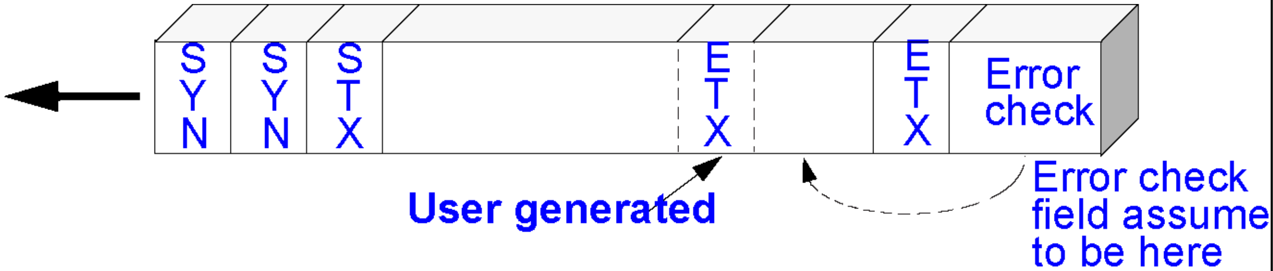


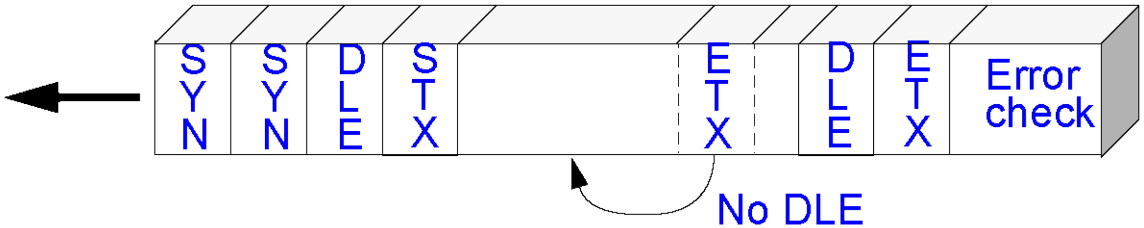
Control Fields

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

Problem: user data may contain “control characters”

Solution: CHARACTERSTUFFING

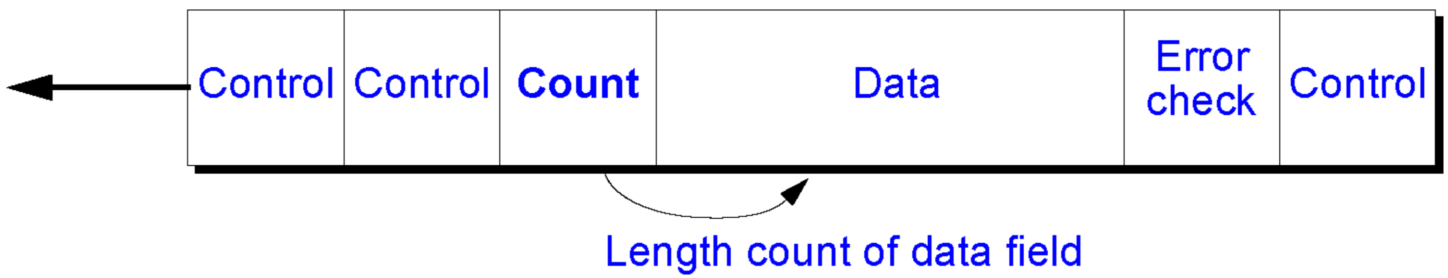




SENDER: each control character is preceded by a DLE (Data Link Escape), (but not in user generated data)

RECEIVER: only control characters preceded by DLEs are interpreted as such

Count Oriented Protocol



Frame contains LENGTH COUNT FIELD

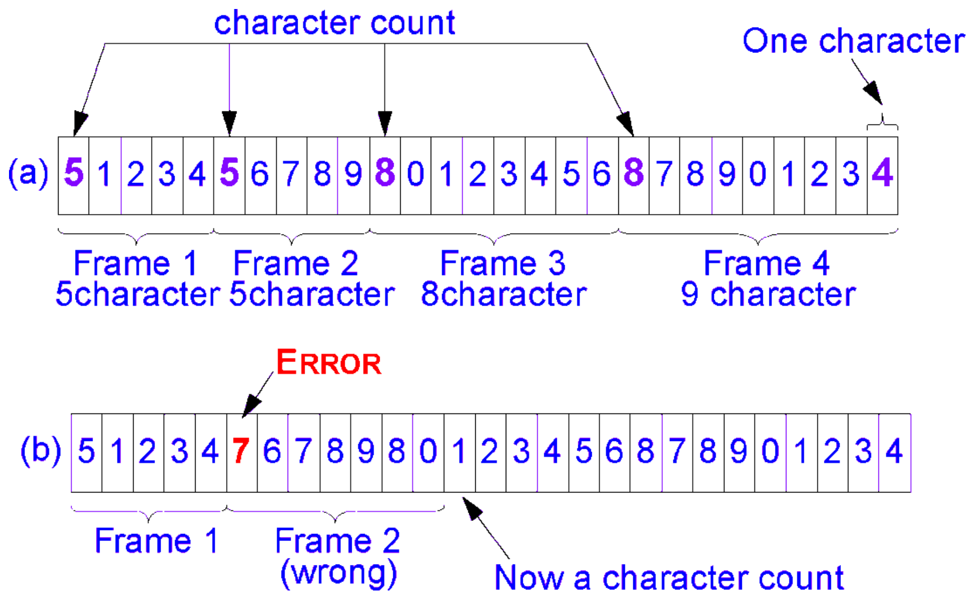
PROBLEM: Transmission error destroys length count

- sender and receiver are not synchronized anymore

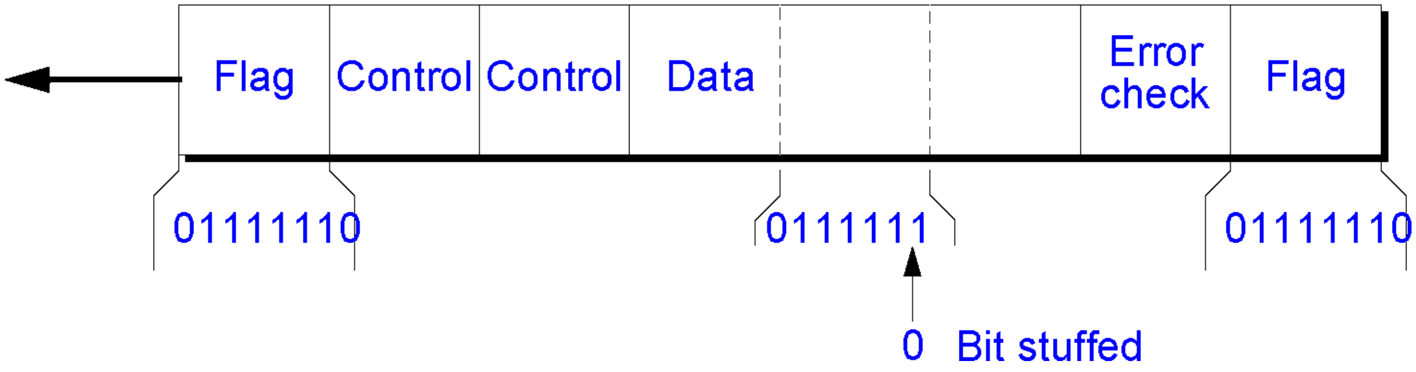
that means

- where does the next frame start?
- Where do retransmitted frames start?

→ Therefore not widely spread!



Bit Oriented Protocols



In use at most of today's protocols

- independent from encoding
- block definition

  flag (01111110)

Bit Oriented Protocols

(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 1 0 0 1 0

PROBLEM:

- "flag" in user data

  `e.g. flag 01111110`

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"

Error dectection

Error Detection

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

Code Word, Hamming Distance

Frame (= code word) contains

- data
- checking information

Code = set of all valid code words

Hamming distance of two words w1 and w2:

- number of bits that differ between two words

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

Hamming distance of a code:

- minimal Hamming distance of all pairs of words

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

Error Detection (according to Hamming)

DETECTION of f 1-bit errors:

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

  d ≥ f + 1

- **f** and less errors generate an invalid code word and are detected
- 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 one 1-bit error
```

Cyclic Redundancy Check (CRC)

Basic idea:

bit strings are treated as polynomials

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

where $k_i = [0,1]$

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

Polynomial arithmetic: modulo 2

```
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
```

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
Remainder ------> 1110
```

Transfered frame: 11010110111110

Error Detection

- Standardized polynomials:

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

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

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

- 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
  - if x+1 is a divider of the CRC, no odd bit error can escape

Flow control

Flow Control and Error Treatment

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

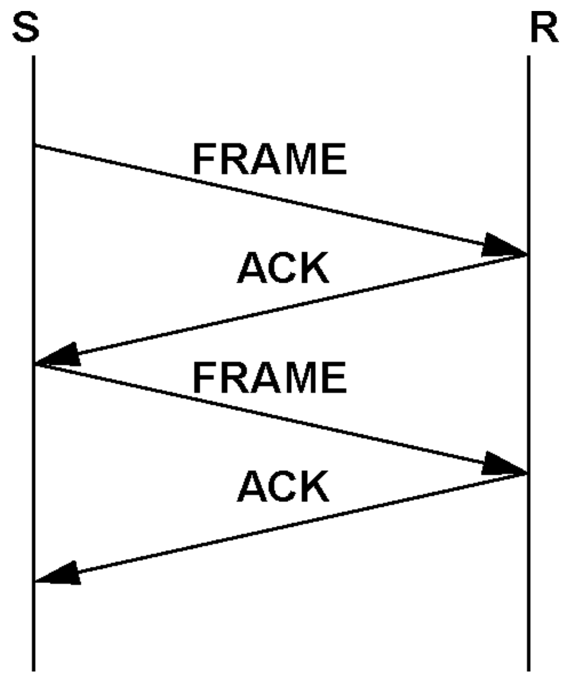
Comment

- 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)

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 (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)

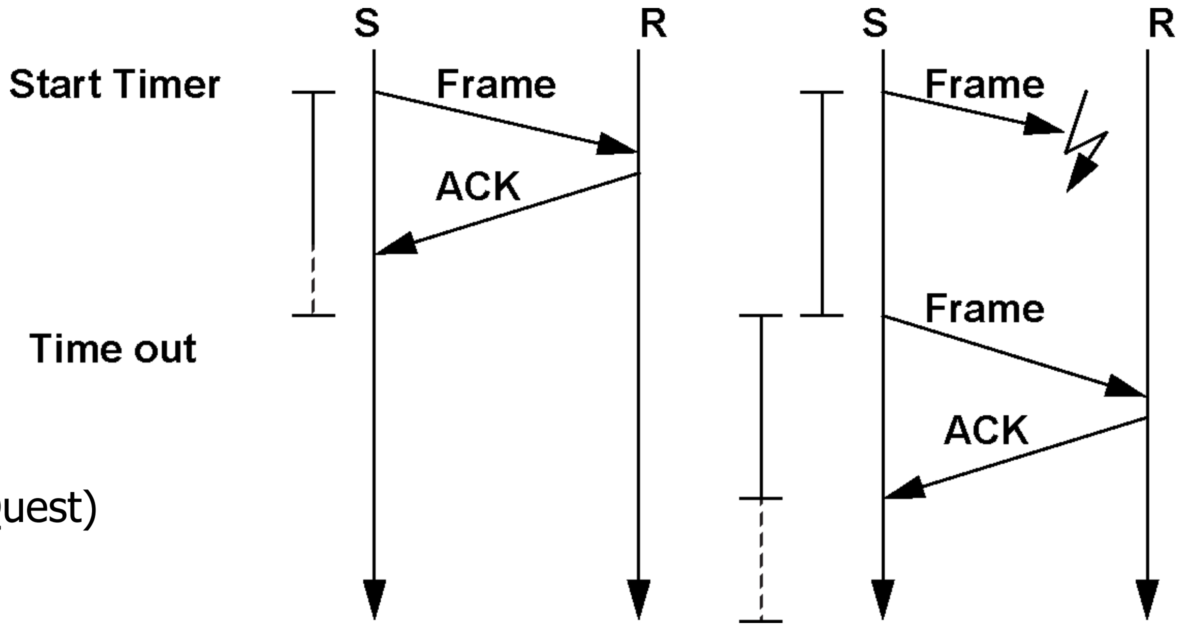
but: additionally, faulty communication channel (loss of frames)...

Protocol 3a: Stop-and-Wait / ARQ

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

Problem: protocol blocks at loss of frames

Solution:
- PAR (Positive-Acknowledgement with Retransmit)
- or
  ARQ (Automatic Repeat reQuest)

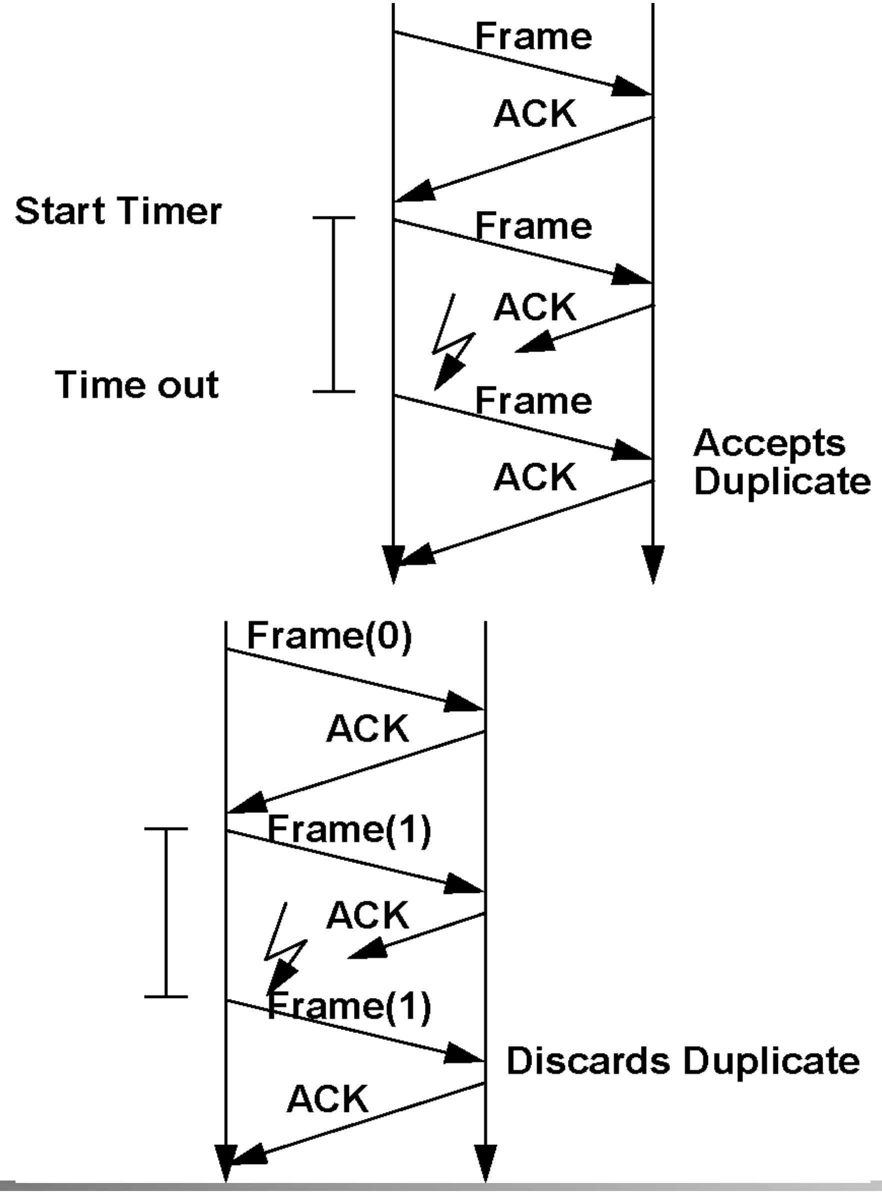


Timeout interval:
- TOO SHORT: unnecessary sending of frames
- TOO LONG: unnecessary long wait in case of error

Protocol 3b: Stop-and-Wait / 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

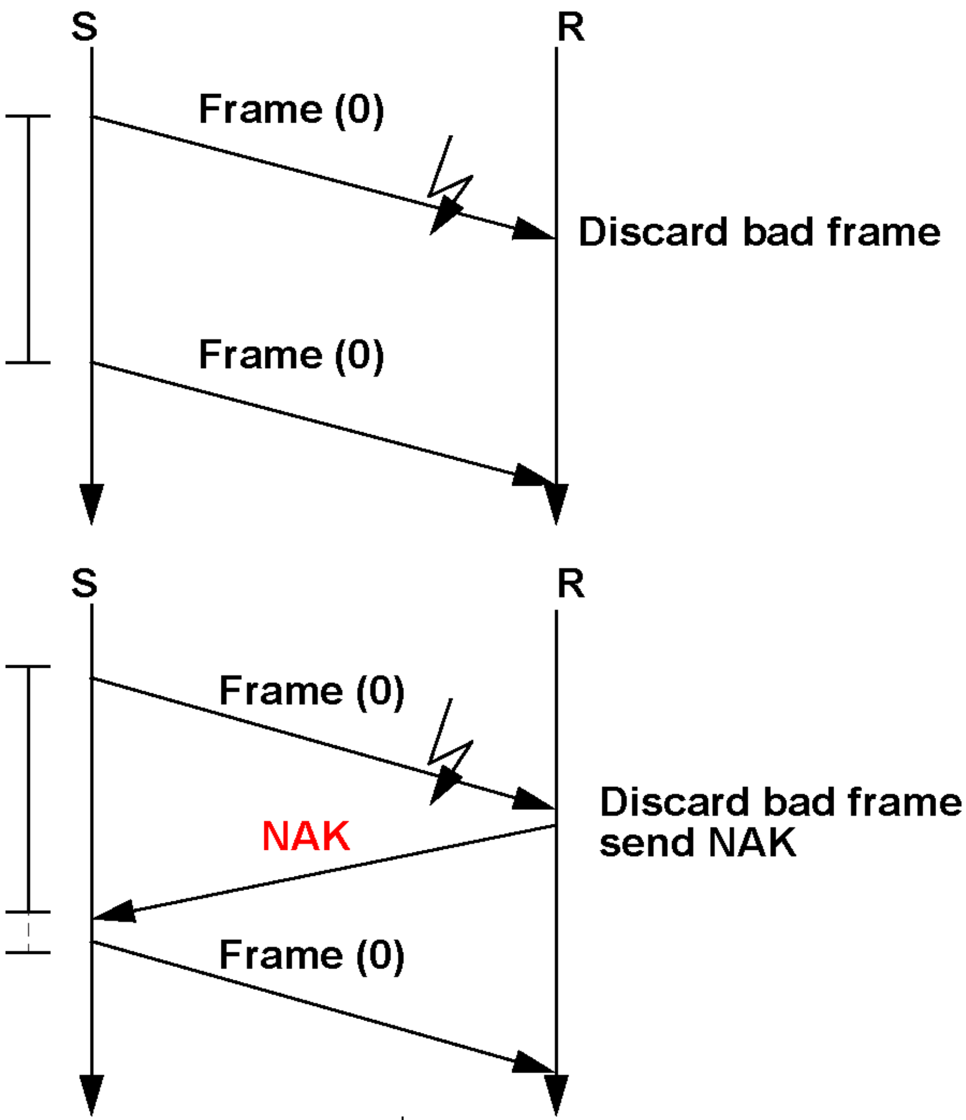
Protocol 3c: Stop-and-Wait / NAC+ACK / SeqNo

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

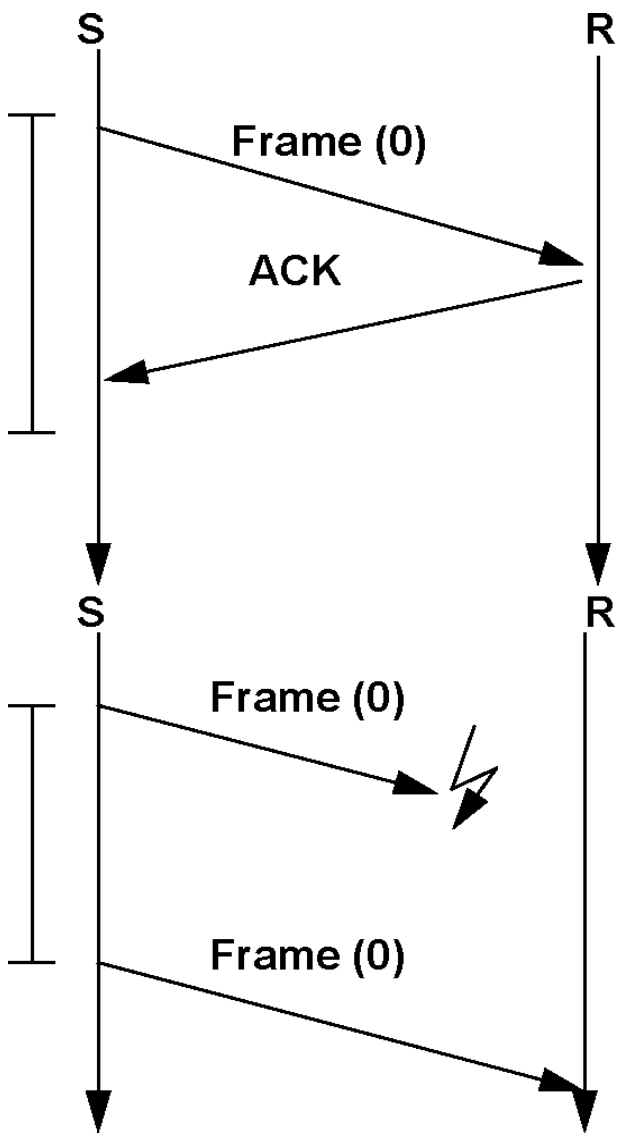
- Frame correctly transmitted

→ ACK sent

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

- frame did not arrive

→ timer issues retransmit

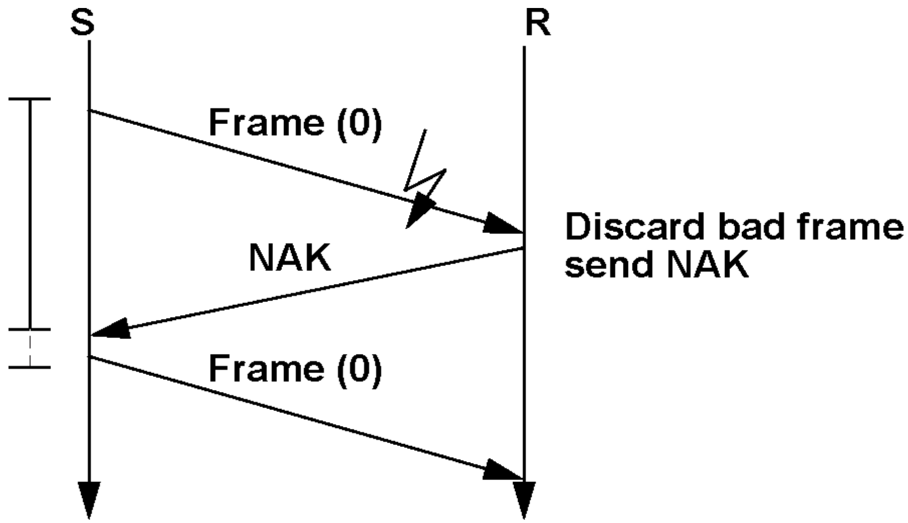


Protocol 3c: Stop-and-Wait / NAC+ACK / SeqNo

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

- faulty frame arrives

→ NAK issued



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

- NAK issued

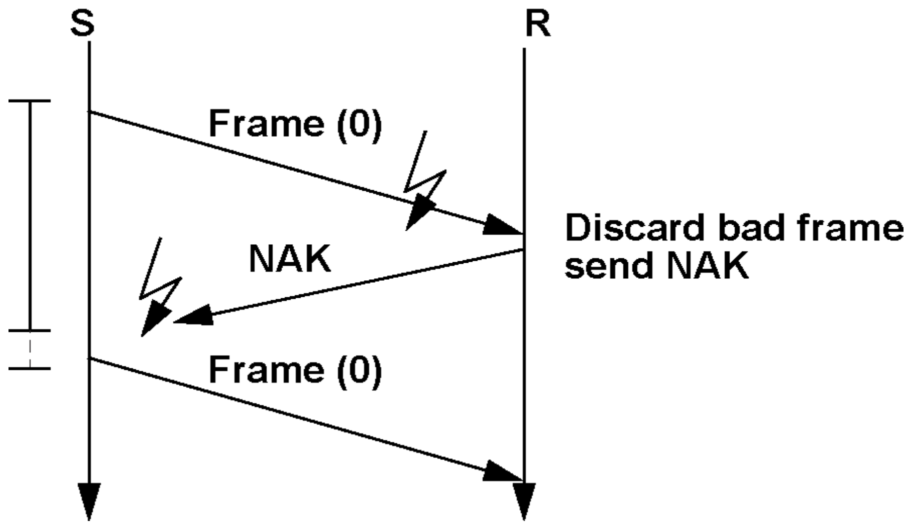
but,

- NAK does not arrive

or

- NAK arrives damaged

→ timer issues retransmit



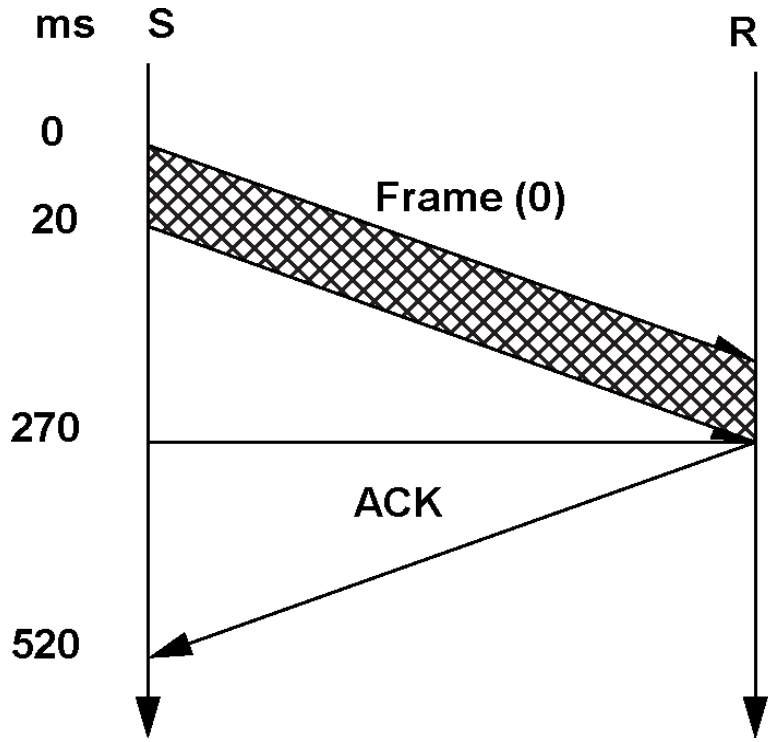
Channel Utilization and Propagation Delay

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

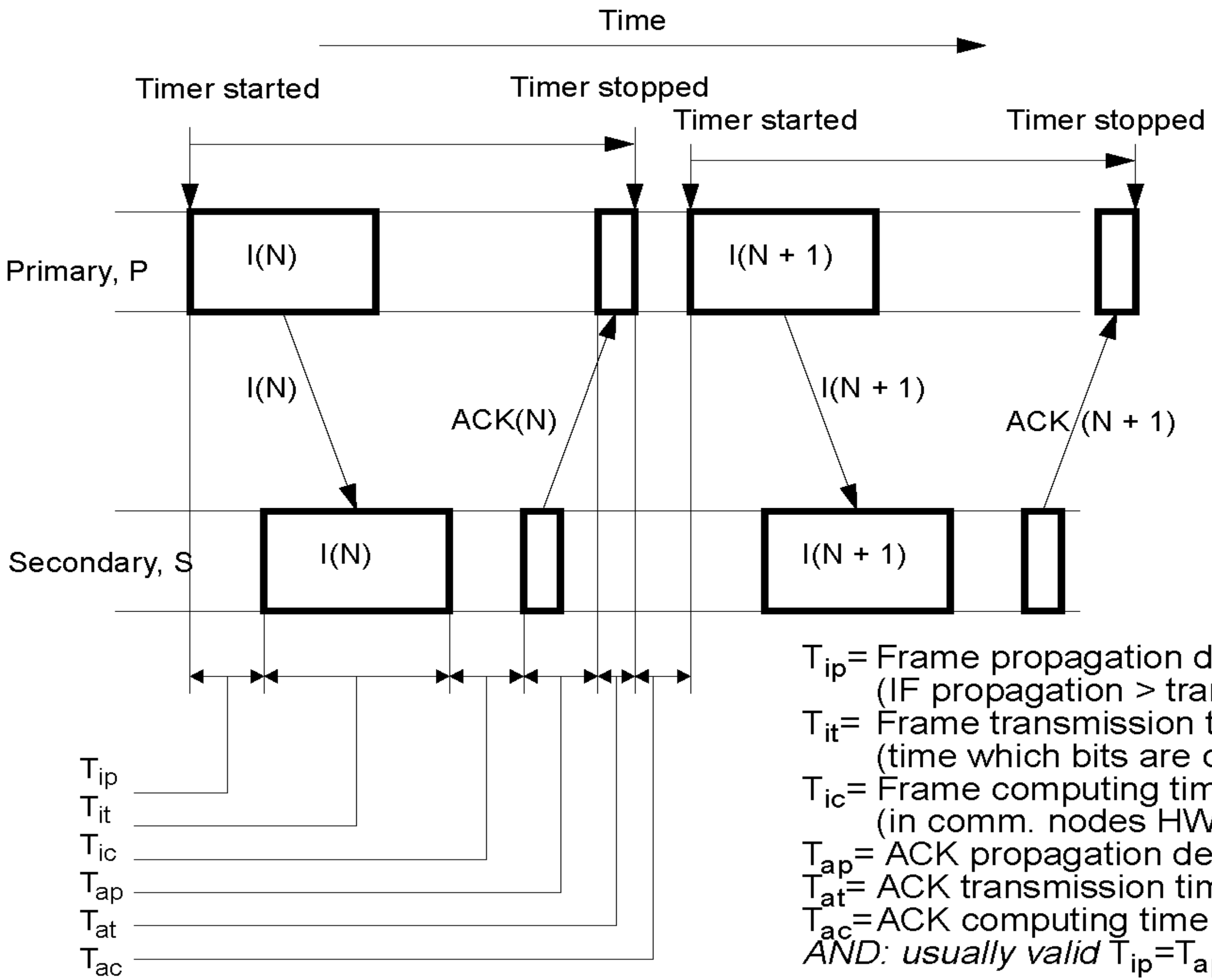


this means

- 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



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):

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

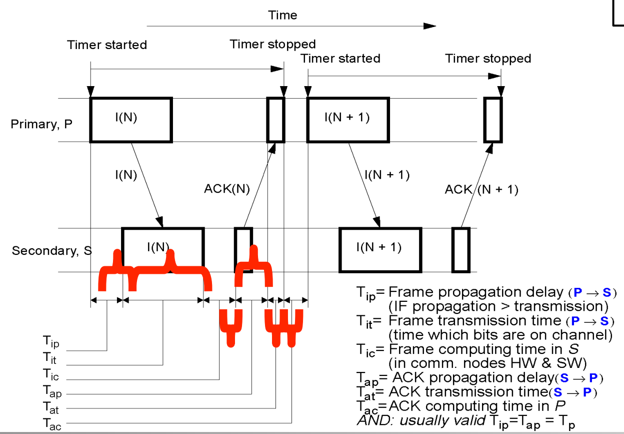
approximated formula:

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

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 transmission

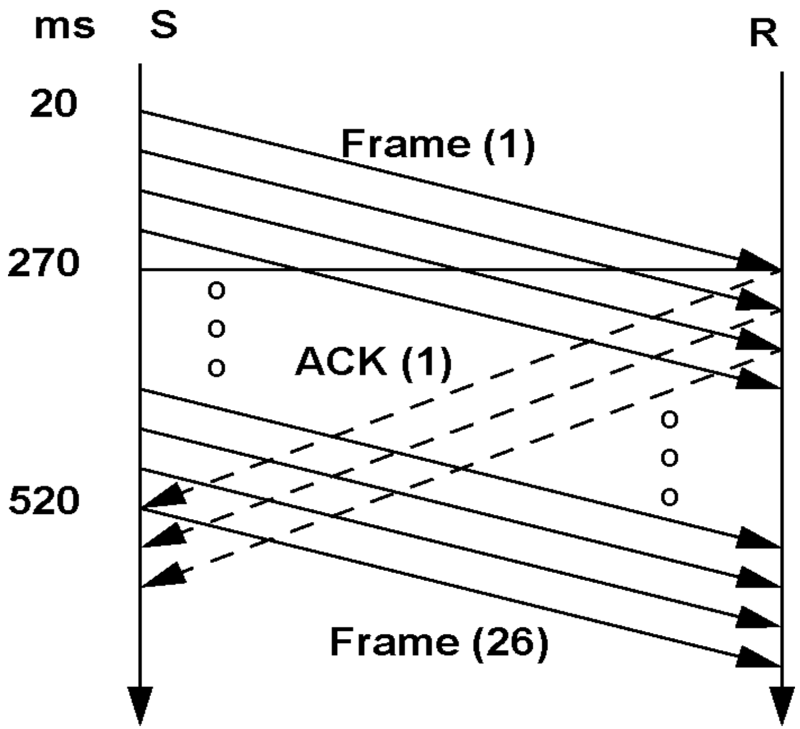
- T_{ip} = Frame propagation delay (IF propagation > transmission)
- T_{it} = Frame transmission time (time which bits are on channel)
- T_{ic} = Frame computing time in S (in comm. nodes HW & SW)
- T_{ap} = ACK propagation delay
- T_{at} = ACK transmission time
- T_{ac} = ACK computing time in P



Better: Pipeline ... Sliding Window

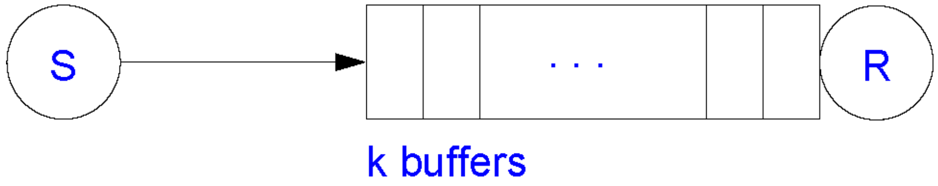
Solution: pipelining

Flow control: sliding window mechanism



Sliding Window: Concept

Flow control: receiving buffer must not flood



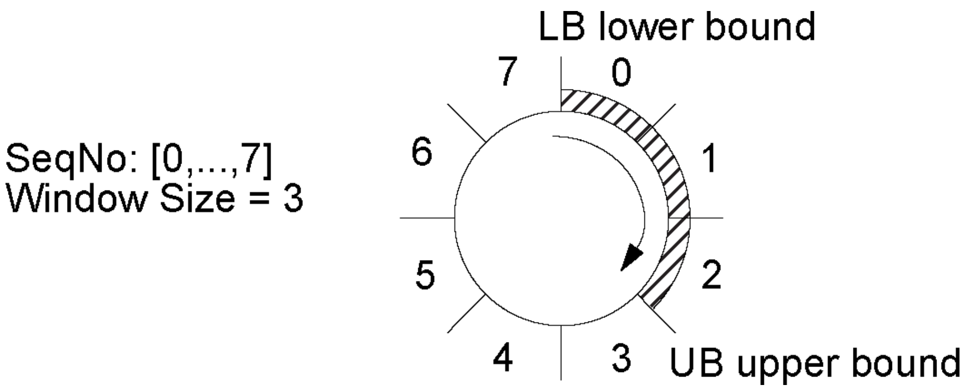
Sender and receiver window per connect/communication relationship

R-WINDOWS:

- sequence numbers, which can be accepted

S-WINDOW:

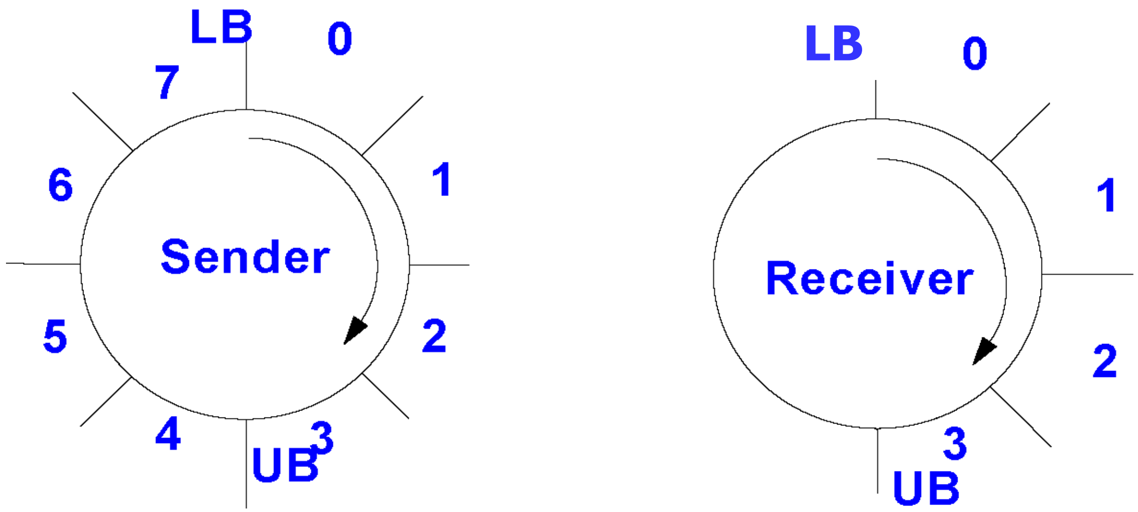
- sequence numbers, which were sent but not yet acknowledged



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

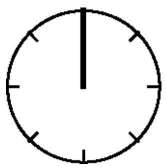
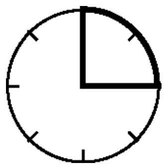
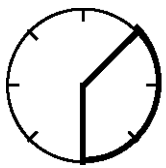
Lower Bound & Upper Bound

	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

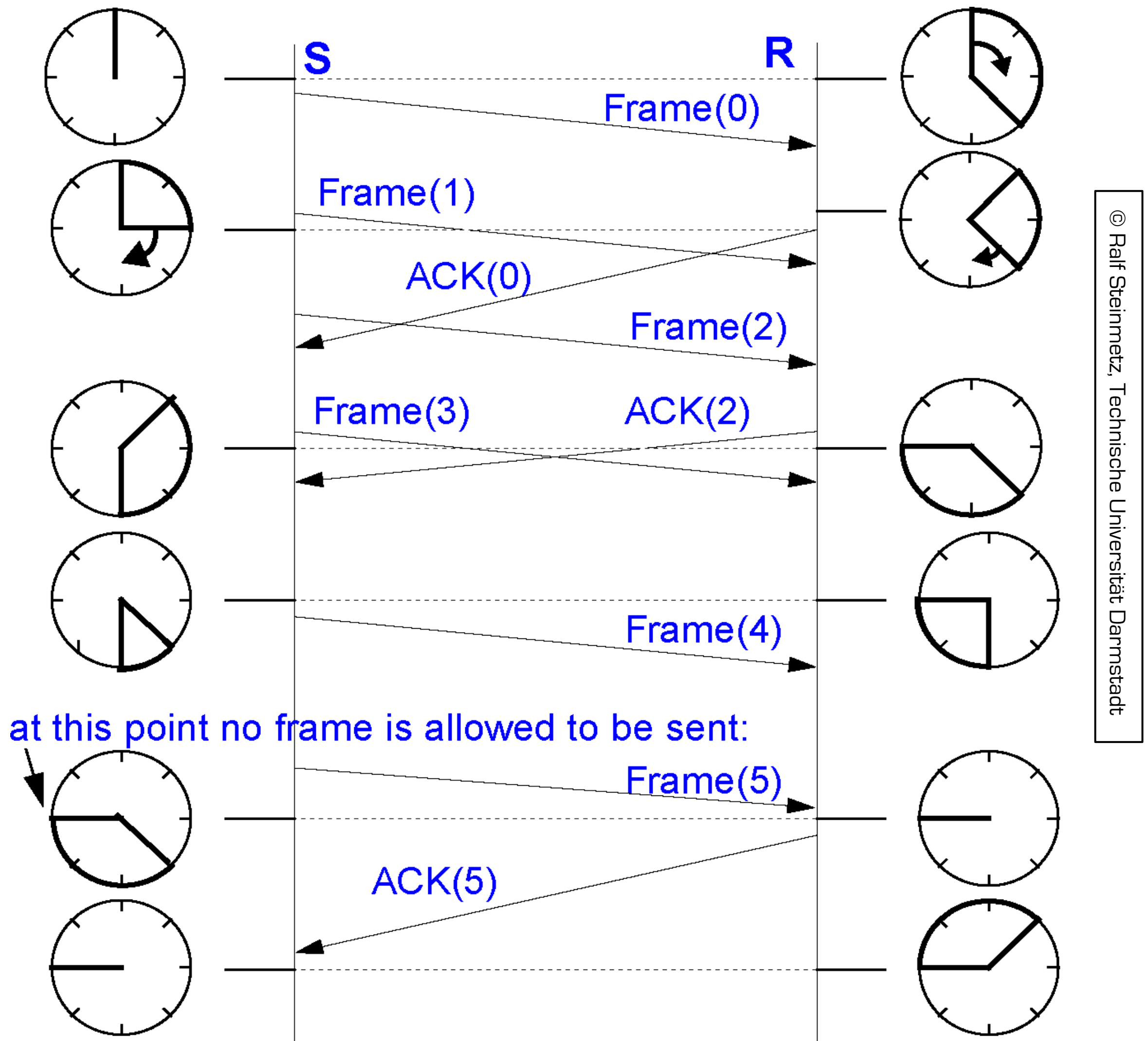
Sliding Window: Examples

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

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)

Sliding Window: Remarks & Refinement

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

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

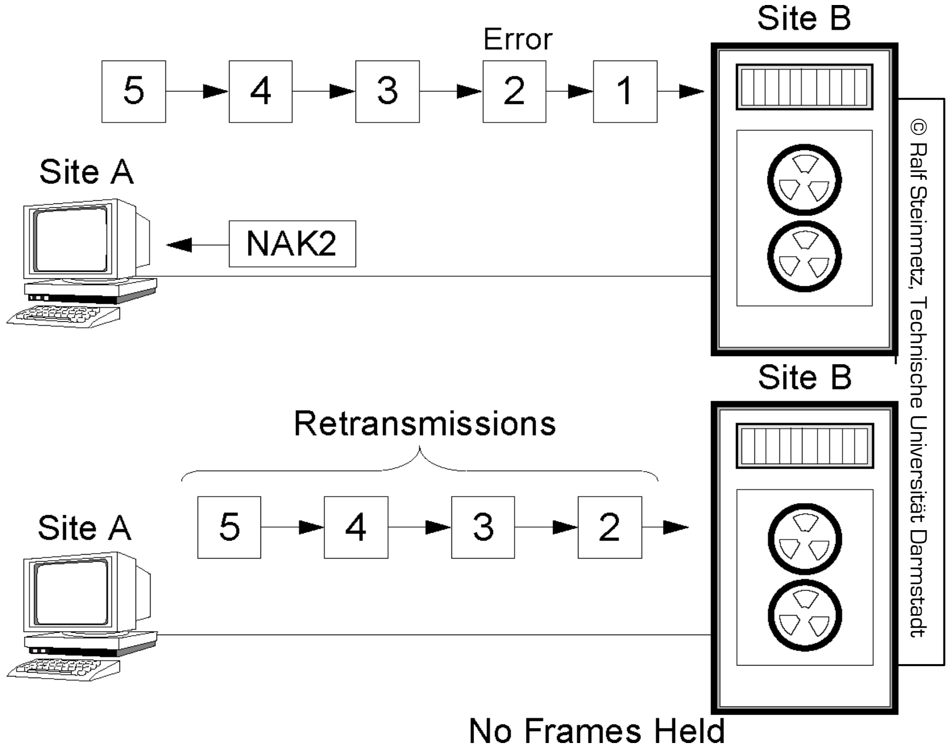
Sliding Window: Go-Back-N (Error Treatment)

Procedure

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

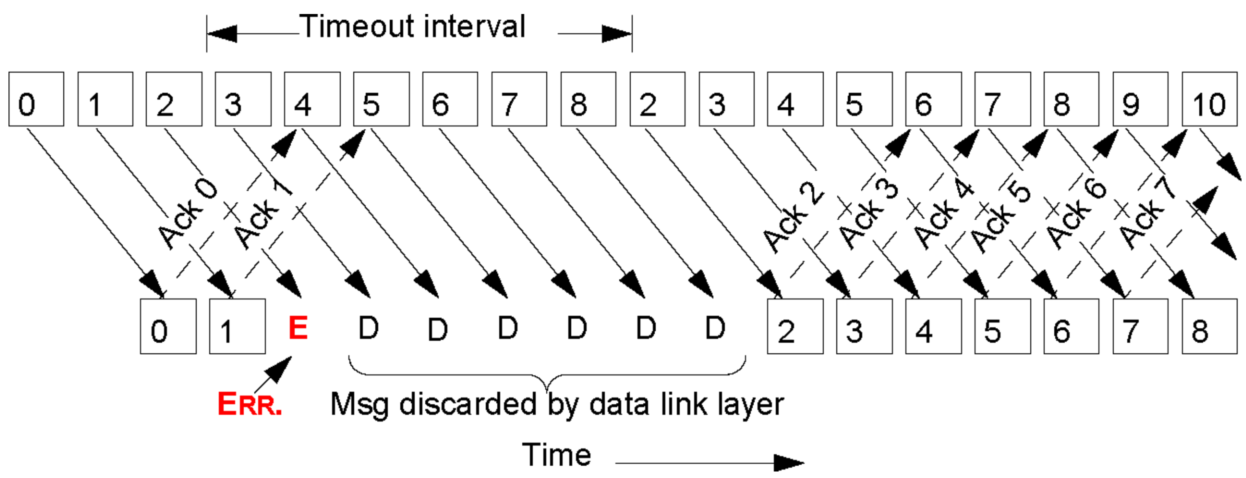
Evaluation

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

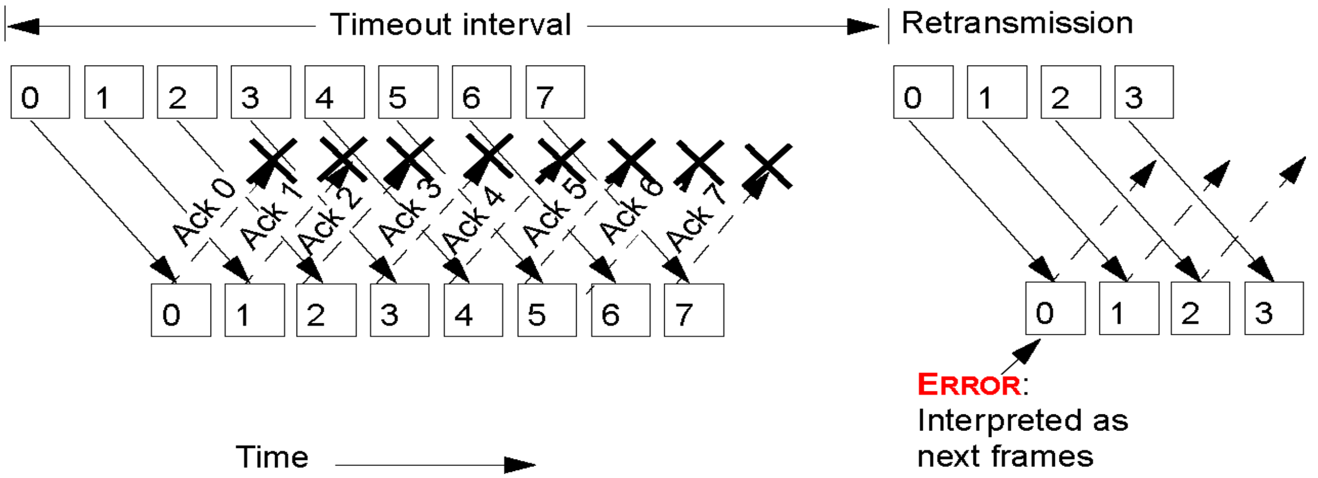


Sliding Window: Go-Back-N (Error Treatment)

Example: sender: error detection by timeout



Sliding Window: Maximum Window Size



Example:

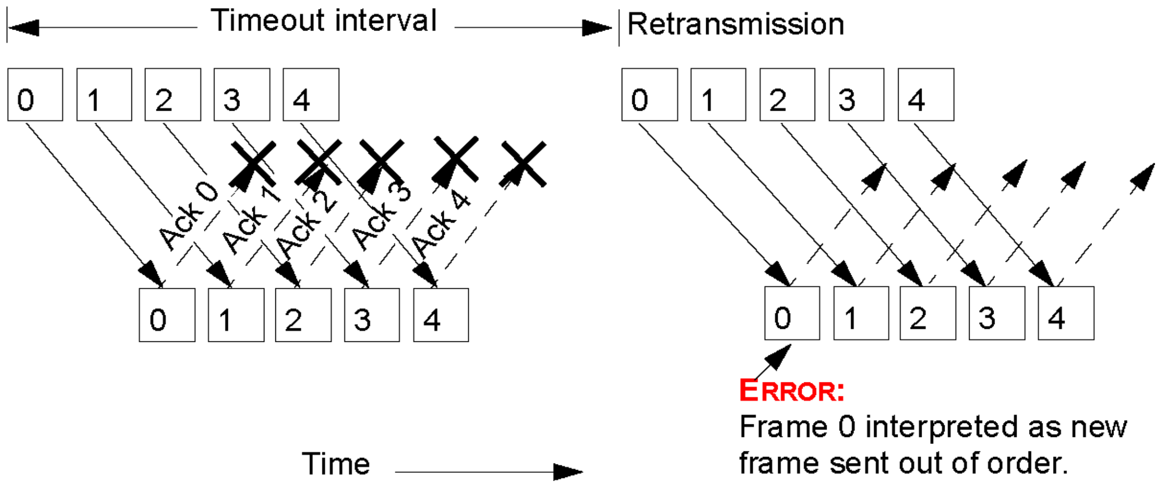
- amount of sequence numbers 8
- window size 8
- all ACKs lost

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

for Go back N (otherwise possibly different)

Sliding Window: Selective Repeat (Error Treatment)

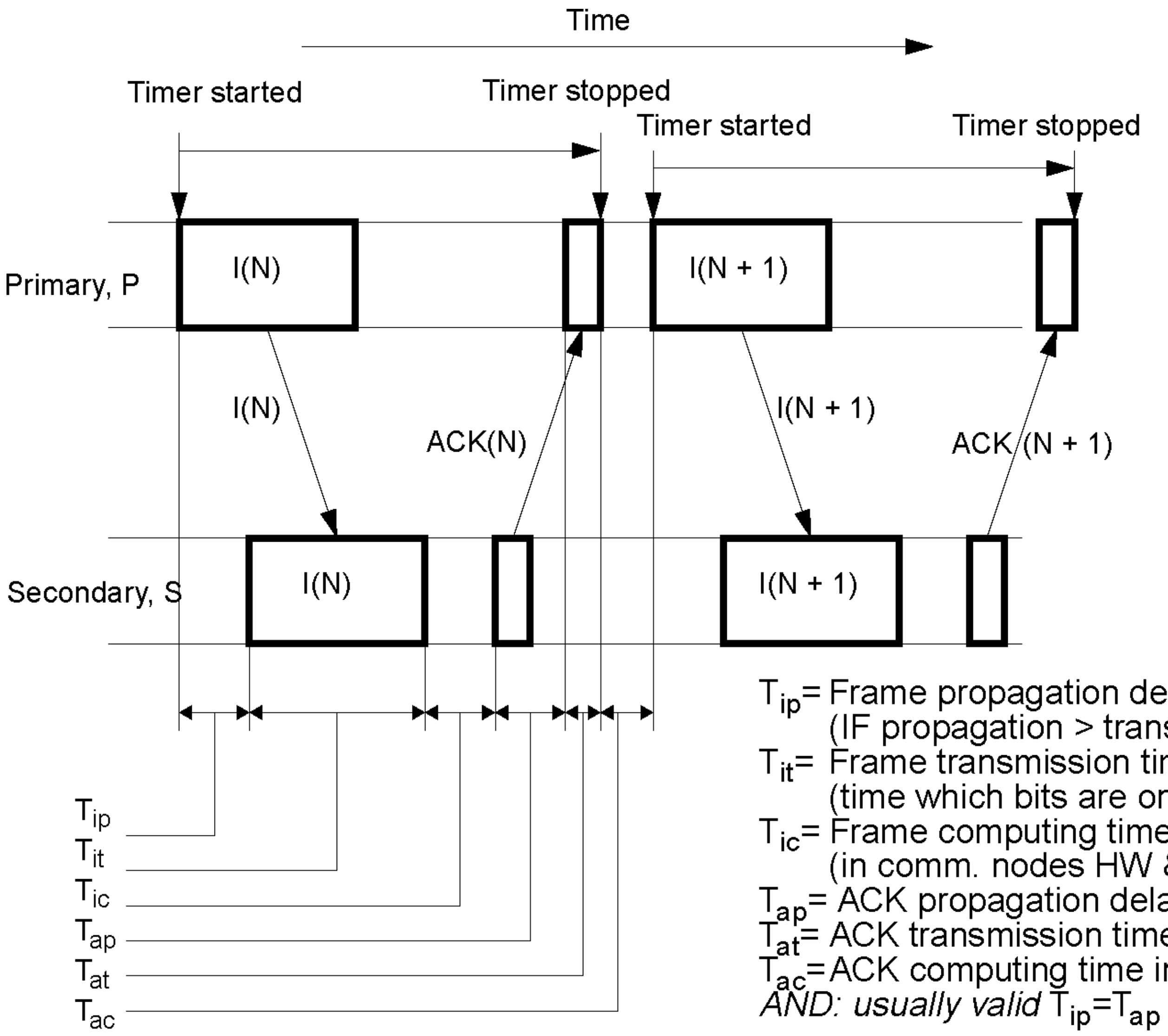
Procedure

- 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 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

Channel Utilization



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: Recapitulation without Sliding Window

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

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

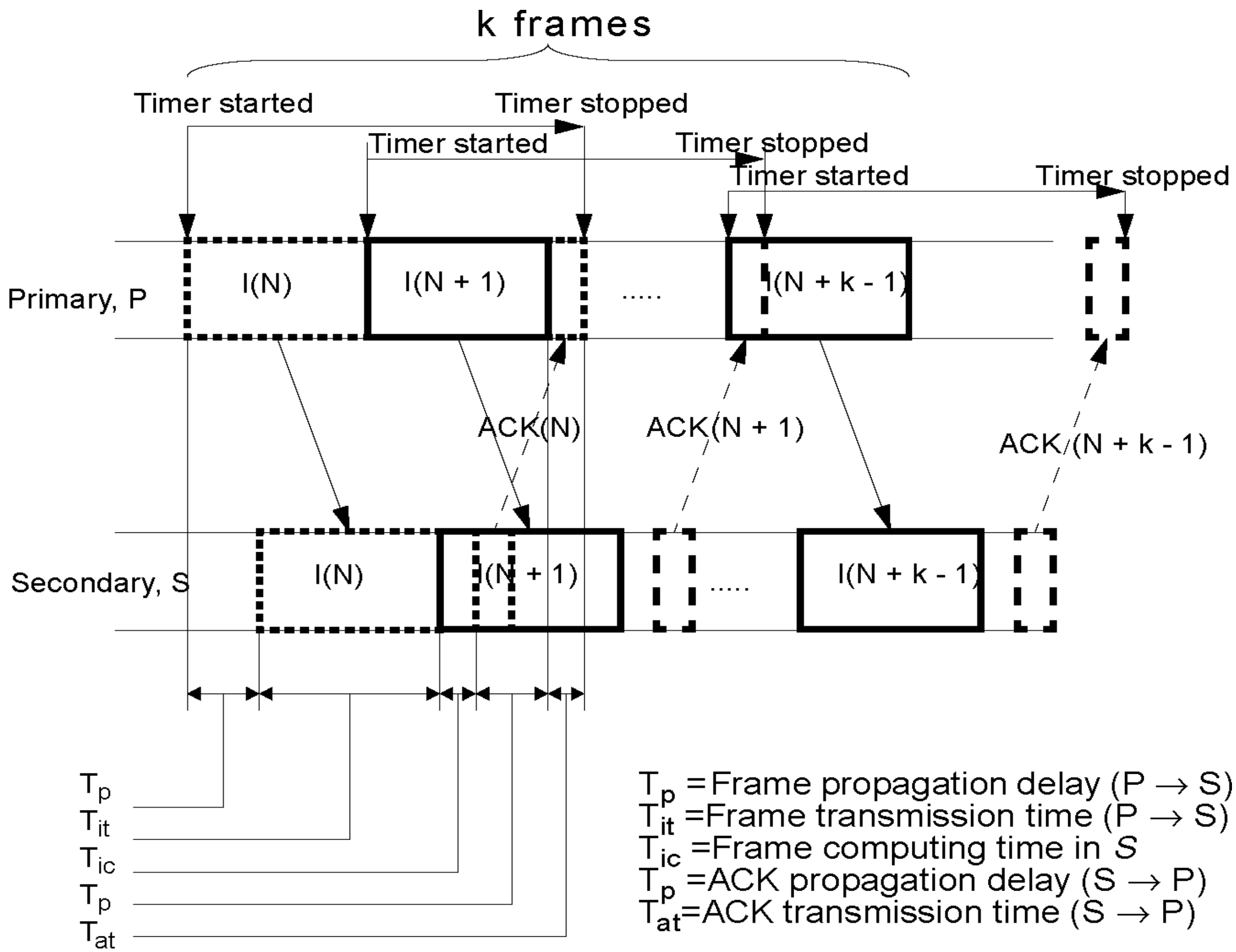
approximated formula:

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

- 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 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

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