Protocol Principles

Layering, CL vs. CO Protocols, Best-Effort vs. Reliable Services
ARQ Techniques, Sequence Numbers, Windowing,
Bandwidth-Delay Product, Flow Control, HDLC

© 2012, D.I. Lindner / D.I. Haas

Agenda

- **Introduction**
- **ARQ Techniques**
  - Introduction
  - Idle RQ
  - Continuous RQ
    - Selective Acknowledgement
    - GoBackN
    - Positive Acknowledgement
    - Selective Reject
  - Sequence Numbers and Windowing
  - Bandwidth-Delay Product
  - Flow Control
  - HDLC Overview

© 2012, D.I. Lindner / D.I. Haas

© 2012, D.I. Lindner / D.I. Haas

Line Protocols

- **Line protocols regulate and control communication between two devices over a point-to-point line**
- **Basic elements**
  - Frame synchronization
  - Frame protection
  - Error detection
  - Usually implemented in hardware
- **Optional elements**
  - Connection and line management
  - Addressing
  - Error recovery
  - Flow control
  - Usually implemented in software

© 2012, D.I. Lindner / D.I. Haas Protocol Principles, v5.1 3

We will start discussing protocol principles by first looking to line protocols. Later we will see that line protocols are corresponding to the Data-link layer (layer 2) of the OSI model. While the first layer of the OSI model (Physical-link layer) performs function like encoding, modulation, bit synchronization and describes the physical aspects (voltage levels, connector type) the Data-link layer typically performs the following logical tasks.

Framing is the task of packing the information of higher layers to provide for example start and end of packet detection plus some optional features.

Frame protection is used to detect possible errors during data transmission.

Addressing is optional and is normally only used in point-to-multipoint Data-link technologies.

Error recovery can be used to allow packet retransmissions if data errors are detected by the frame protection mechanism

Flow control can be used to prevent buffer overflow situations on the receiver side

The Data-link can be driven in connection-oriented mode or connection-less mode. Newer technologies mainly use the connection-less mode because the connection-oriented functionality is typically provided by higher OSI layers e.g. TCP on OSI layer 4. Error recovery and flow control are only possible in connection-oriented mode

Three Important Principles for Data Communication

- **Layering**
  - Structuring the complex task of data communication into smaller pieces by usage of “layers”
- **Services**
  - Are provided by a layer to the upper layer and describe what is exchanged between layers within a system
- **Protocols**
  - Are used for communication between the systems (peers) within a layer

© 2012, D.I. Lindner / D.I. Haas Protocol Principles, v5.1 4

A layer is built by the resources of the corresponding protocol peer entities and by the protocol procedures performed between them.

Protocol defines fields of the control field of a frame (bits seen on the wire) and the communication behavior of the peers receiving and sending frames.

A layer is using the services of the lower layers to provide an enhanced service to the upper layer. Communication between layers are internal to the system.

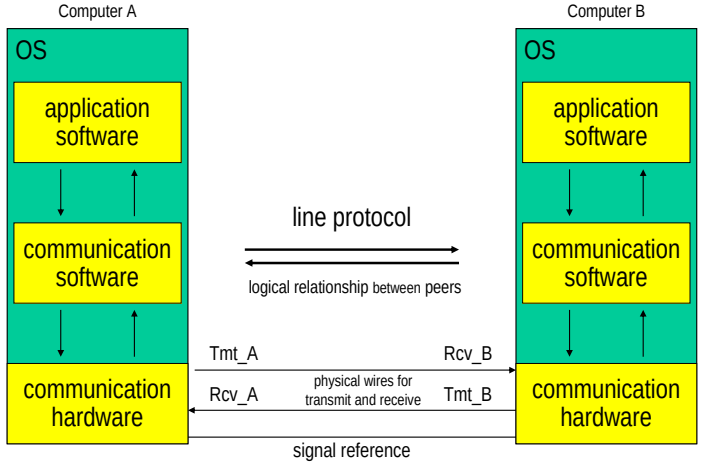
For example the application layer can access the lower layer (the protocol stack) via API (application programming interface. The communication layer can access the lower layer via network-card.

Application software uses the communication software (normally part of an operating system , OS) in order to exchange data. Mailbox or queuing techniques allow cooperation of application and communication software within a computer system.

The communication software uses a line protocol for peer to peer communication (logical horizontal communication relationship on a given layer) and hides the details of actual used line protocol and other related tasks from the application software .

Overall an procedural approach.

Point-to-Point Communication 3 Layer Model



The line protocol is implemented on transit by using the control field / protocol header of a frame. The horizontal arrows describe this logical cooperation between systems. The actual physical transport of bits (frames) is performed by the communication hardware (encoding, bit synchronization, frame synchronization and so on).

The vertical arrows describe the communication between layers within a system. It may be implemented by some kind of mailbox or queuing technique. An example for that is the already mentioned API of the TCP/IP protocol stack.

Cooperation of Software Layers:

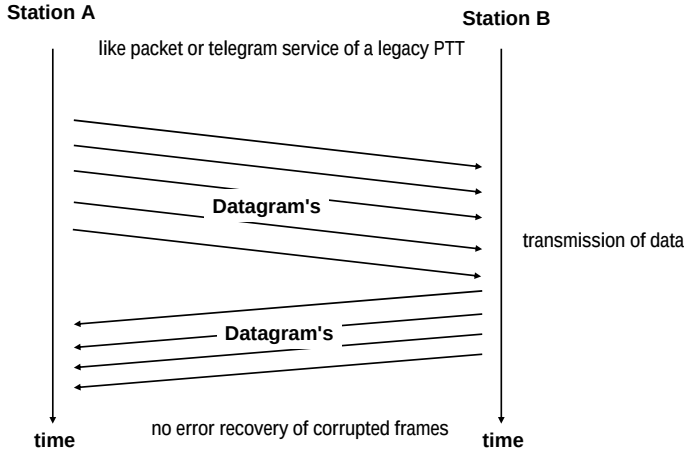
If information has to be transmitted from A to B the application SW of device A forwards some data blocks to the communication SW. The communication SW transmits the data using the communication hardware and the line protocol. The communication SW of device B receives the data and forwards it to the application SW.

That means, the communication SW – communication layer provides a service for the application SW.

This service type can be “Best-Effort” or “Reliable”

© 2012, D.I. Lindner / D.I. Haas

Best-Effort Service: Connection-less Protocol



Best-Effort Service:

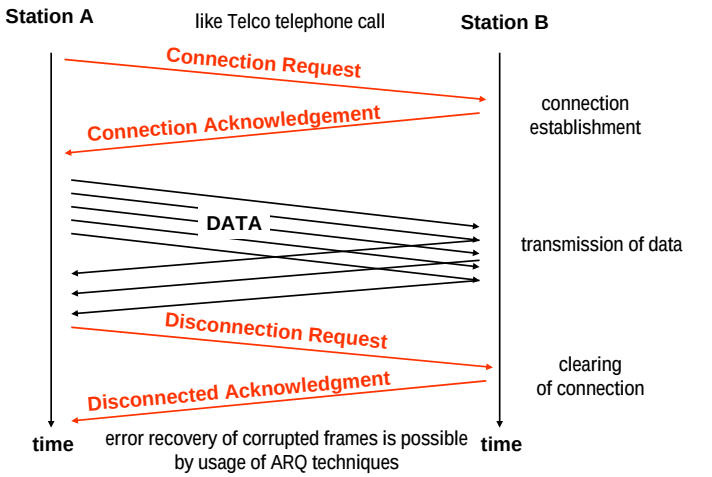
Communication SW uses only basic elements (frame synchronization, frame protection, error detection) of a line protocol to transmit data blocks.

No special frame types are necessary to implement this protocol strategy hence requirements for implementation of communication SW are low (no state machine, limited buffer memory, no variables). The protocol type is called connection-less.

But transmission errors cause receiver to discard data blocks. Error recovery (correction of errors) - if necessary - must be done by the application or better application protocol procedures themselves.

© 2012, D.I. Lindner / D.I. Haas

Reliable Service: Connection-oriented Protocol



© 2012, D.I. Lindner / D.I. Haas

Reliable Services can be implemented only by connection-oriented protocols.

In connection-oriented protocols a (logical) connection is established before data is allowed to flow.

The connection establishment is done by special control frames like connection request and connection established.

Then we find the data exchange phase which typically may use error recovery and flow control. When the data transmission is finished we have special control frames like disconnect request and disconnected to tear down the connection again.

Transmission errors will be detected and corrected by the communication SW using feedback error control

(retransmission of corrupted data blocks). We call that **Automatic Repeat reQuest (ARQ)**.

Hence a reliable transmission service for application SW can be achieved done by error recovery technique of the communication SW.

A more sophisticated communication SW with more resources is necessary in order to implement an ARQ strategy.

© 2012, D.I. Lindner / D.I. Haas

Connection-oriented Protocols

- **Different definitions**
  - Some say *"protocols without addressing information"* and think of circuit-switched technologies
  - Some say *"protocols that do error recovery"*

  - **Correct: "protocols that require a connection establishment before sending data and a disconnection procedure when finished"**

© 2012, D.I. Lindner / D.I. Haas

In the past a connection-oriented protocol was very often seen as a protocol that performs a connection setup procedure and supports error recovery and flow control. For the legacy X.25 packet-switching network technology that is quite correct.

But newer packet-switching network technologies like frame relay and ATM do not support error recovery and end-to-end flow control. But they are still connection oriented. In case of a temporary connection (SVC) we need a connection setup procedure but in case of permanent connection (PVC) a connection setup procedure is not even necessary.

© 2012, D.I. Lindner / D.I. Haas

Agenda

- **Introduction**
- **ARQ Techniques**
  - Introduction
  - Idle RQ
  - Continuous RQ
    - Selective Acknowledgement
    - GoBackN
    - Positive Acknowledgement
    - Selective Reject
  - Sequence Numbers and Windowing
  - Bandwidth Delay Product
  - Flow Control
  - HDLC Overview

ARQ Techniques Overview

- **ARQ protocols guarantee correct delivery of data**
  - Error recovery by usage of feedback error control
  - Retransmission of data (information) frames after errors are detected by the receiver
- **Basic Method:**
  - Receiver acknowledges correct receipt of data frame by sending special control frames (ACK) in opposite direction
  - Acknowledgements refer to identifiers (sequence numbers) carried in the protocol header of the original data frame (I)

ARQ techniques are used to allow data retransmissions in the case of transmission errors and packet loss.

With the help of sequence numbers (serial number of a data packet), applied by the sender, the receiver is able to detect packet loss and is further able to acknowledge properly received frames.

Frame buffering: Each data frame transmitted is stored in a retransmission buffer until receipt of the corresponding acknowledgement, also every received data frame is stored in a receive buffer at the receiver side.

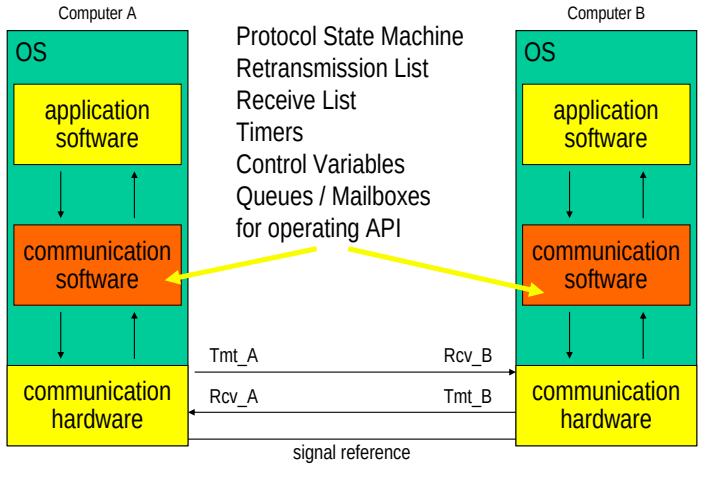
Frame retransmission: Either timeout driven (no acknowledgement is received within a timeout) or protocol driven

Identifiers (N, N+1, ...) are necessary to mark the sequence of data frames and to recognize duplicate frames. Identifiers are implemented by so called sequence numbers.

Reliable transmission techniques are mainly used for data traffic e.g. SMTP, HTTP, FTP etc, because we don t want to receive corrupted mails or html pages.

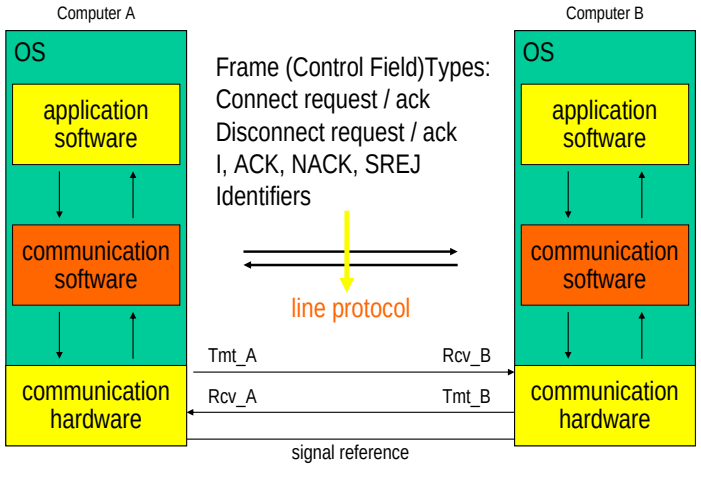
For real time traffic like Voice over IP systems we prefer unreliable transmission techniques, because it makes no sense to retransmit a lost word a few milliseconds later again. This would destroy the harmony in the speech even more than the lost word. For real time systems “Forward Error Correction” systems would be much more useful.

Necessary Resources for Com-SW Layer



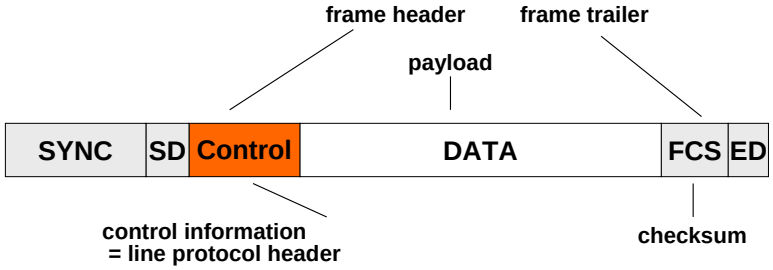
© 2012, D.I. Lindner / D.I. Haas Protocol Principles, v5.1 11

Necessary Resources for Com-SW Layer (cont.)



© 2012, D.I. Lindner / D.I. Haas Protocol Principles, v5.1 12

Remember Control Field (Generic Frame Format)



Frame Type Field: Connect Request, I, ACK, NACK, …
Identifier Field: N, N+1, …

ARQ Variants

Idle-RQ

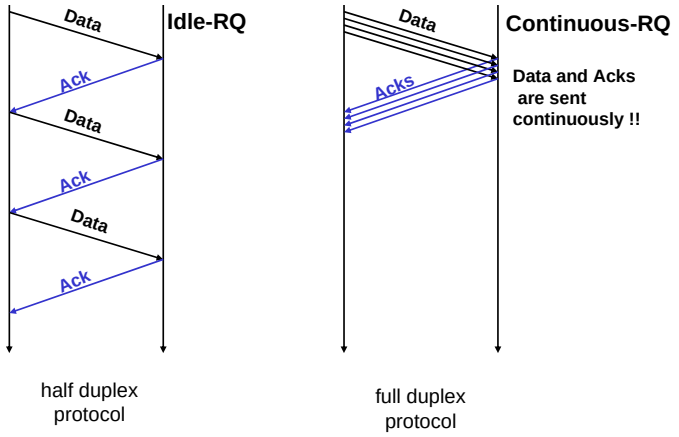
Continuous-RQ

- **Selective ACK (SACK)**
- **GoBackN**
- **Positive ACK**
- **Selective Reject (SREJ)**

There are two major families of ARQ requests the idle-RQ system and the continuous-RQ system.

The continuous-RQ system consists of four different flavors. The Selective ACK, Positive ACK, GoBackN, the Selective Reject (SREJ) method and sometimes even a mix out of these basic methods.

Idle-RQ versus Continuous-RQ



Idle-RQ: After sending a data frame the transmitter must wait for the acknowledgment before next data frame is allowed to send (half duplex protocol).

Old and slow method, but results in small code and only little resources are necessary. Even used today e.g. TFTP (Trivial File Transfer Protocol). Fits well to half-duplex-lines, but is inefficient concerning usage of bandwidth in case of full-duplex line. The only real advantage of the Idle-RQ technique compared to the more sophisticated Continuous-RQ techniques is the little amount of memory and processor resources that is needed. Therefore it is very easy to implement them in ROM based systems.

Continuous-RQ: Data frames and their according acknowledgements are sent continuously (full duplex protocol).The sender is allowed to put a certain amount of frames into the send buffer and transmit them all in one go. The amount of frames the sender is allowed to send is either negotiated during the connection establishment phase or dynamically adjusted by max window size announcements of the received acknowledgements. Continuous-RQ requires dramatically more resources than Idle-RQ or simple connectionless protocols (e.g. retransmission timers, retransmission buffers, receive buffers) and might result in high CPU loads. Would suffer from half-duplex-lines hence needs full-duplex lines in order to perform well.

Half-duplex line: Transmission path can be used only in one direction at one time. Used by modems where direction of transmission is switched in a controlled way (e.g. direction is agreed by line protocol control frames and activation of transmission signalled by physical line management techniques (e.g. V.24 RTS and CTS signals).

Full-duplex line: Transmission path can be used in both directions at the same time.

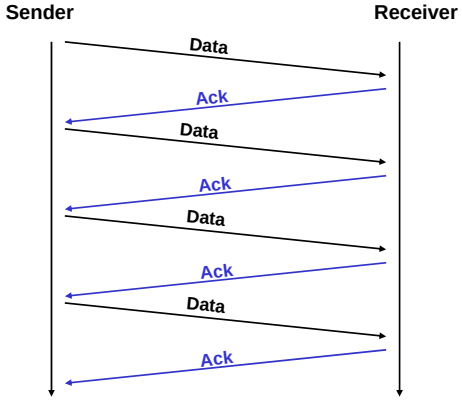
© 2012, D.I. Lindner / D.I. Haas

Agenda

- **Introduction**
- **ARQ Techniques**
  - Introduction
  - Idle RQ
  - Continuous RQ
    - Selective Acknowledgement
    - GoBackN
    - Positive Acknowledgement
    - Selective Reject
  - Sequence Numbers and Windowing
  - Bandwidth-Delay Product
  - Flow Control
  - HDLC Overview

© 2012, D.I. Lindner / D.I. Haas

Idle-RQ

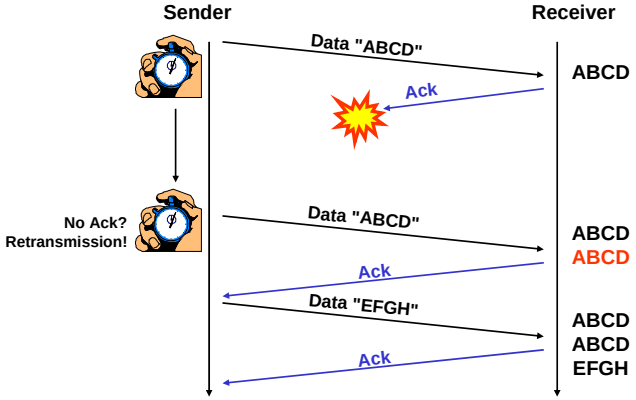


The idle-RQ technique is a stop and go protocol, this means when the sender has sent out one data frame he must wait for the according acknowledgement before he is allowed to sent the next frame.

The idle-RQ protocol operates in a half duplex mode and is typically used in master – slave environments. So the master sends a frame and must wait for the response of the slave whether the frame was properly received or not.

In today's networks the idle-RQ technique is seen very rarely, because it introduces large delays and is not able to fill data pipes we are currently used to.

Why Identifiers or Sequence Numbers?



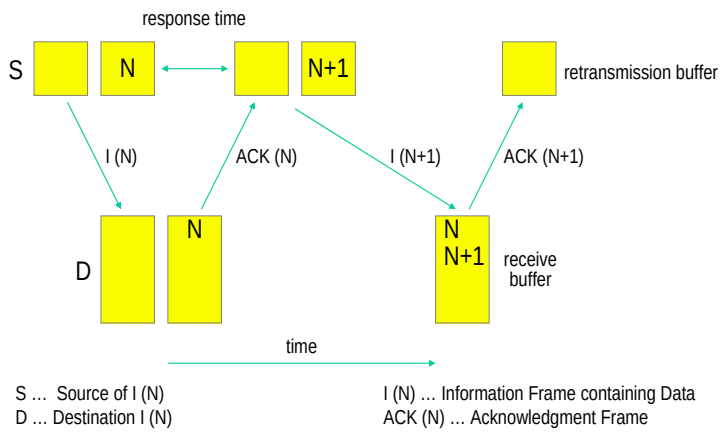
There are two different ways how idle-RQ technique might be implemented.

In this graphic an idle-RQ system without the use of sequence numbers is shown. The sender sends out a data frame, starts an retransmission timer and waits for the receive of an acknowledgement.

An already sent data frame remains in the send buffer and may only be deleted if an proper acknowledgement is received.

A data frame retransmission will happen, if the retransmission timer times out before an acknowledgement is received. Even if the transmission itself was successful and only the acknowledgement was lost. This could lead to an transport of duplicate data frames.

Basic Sequence of Idle-RQ



© 2012, D.I. Lindner / D.I. Haas

Simple ARQ implementation
- stop & wait protocol
- device waits for the acknowledgement (ACK) before sending the next data frame
- basic method can be improved by NACK

Two identifiers are necessary (0, 1)
- distinction between new data frame or duplicate frame

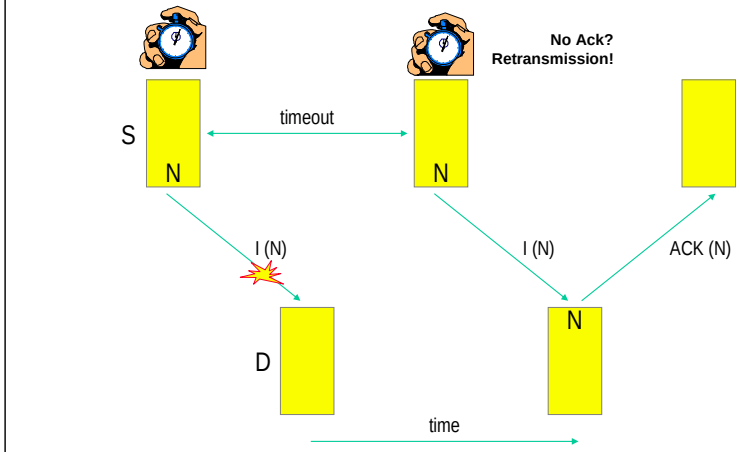
Numbering of data frames
- modulo 2

Half duplex protocol

Full duplex lines can not be efficiently used

© 2012, D.I. Lindner / D.I. Haas

Idle-RQ Retransmission 1

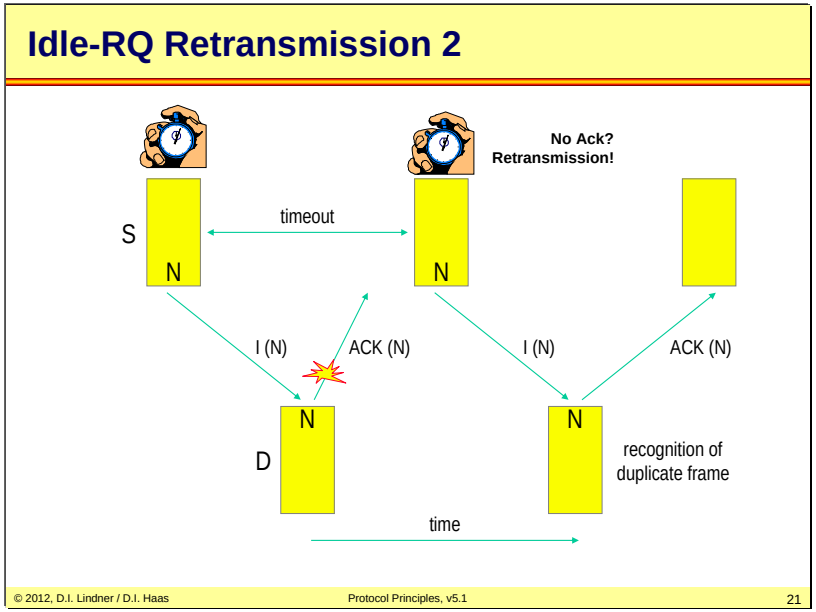


© 2012, D.I. Lindner / D.I. Haas

Retransmission caused by loss of a data (information) frame (I).

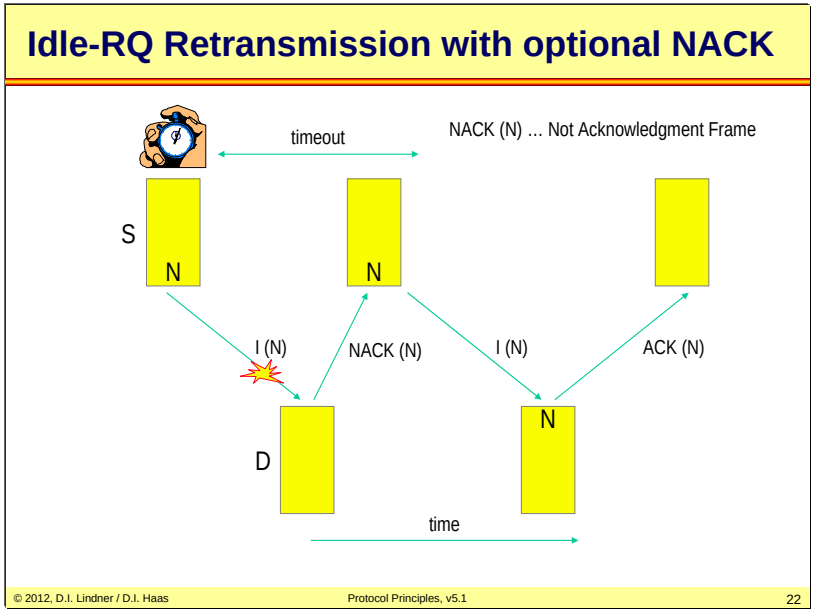
© 2012, D.I. Lindner / D.I. Haas

Idle-RQ Retransmission 2



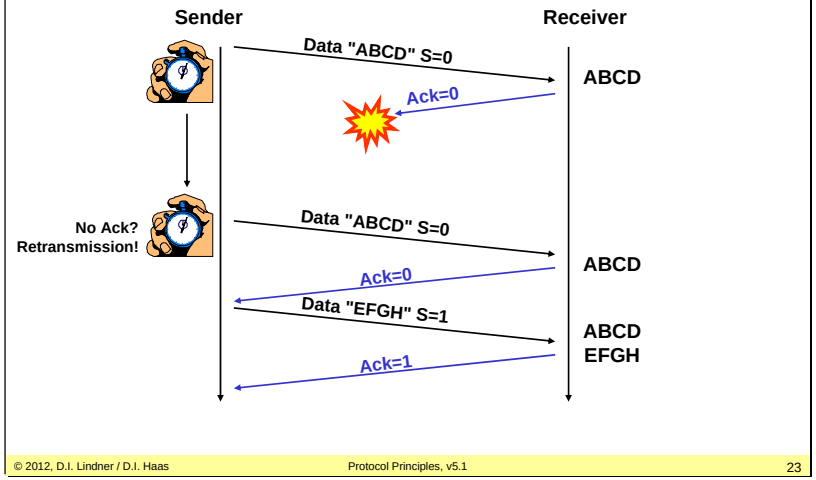
Retransmission caused by loss of a acknowledgement frame (ACK). Receiver needs receive buffer/ receive list to store last correct frame in order to recognize data frame duplicates.

Idle-RQ Retransmission with optional NACK



With NACK we need not to wait for the timeout to expire, hence the receiver can trigger the retransmission earlier. The problem for the receiver: How to get noticed about a corrupted frame (usually physical hardware will silently discard a received frame with an CRC-FCS error indicated).

Modulo 2 Numbering for Identifiers Sequence Numbers 0 and 1



© 2012, D.I. Lindner / D.I. Haas

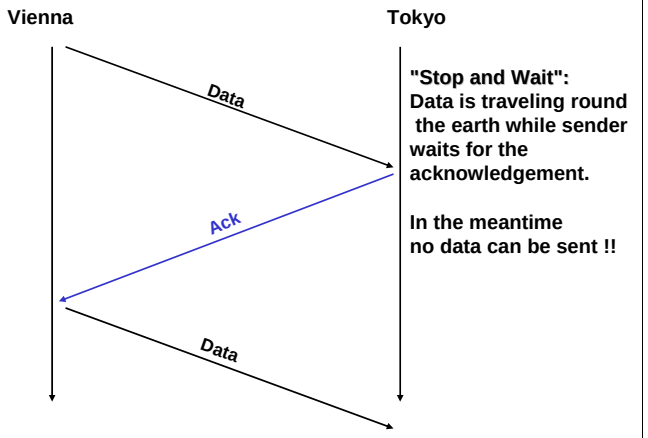
In the second scenario of idle-RQ systems sequence numbers are used.

In this case the sender sends out a data frame with a valid sequence number. Keeps the data frame in the send buffer and starts the retransmission timer. The acknowledgement frame is lost again and the data frame is retransmitted.

But now the receiver is able to recognize, by the help of the sequence number, that the received data frame is a duplicate and is able to discard it.

© 2012, D.I. Lindner / D.I. Haas

Slow !

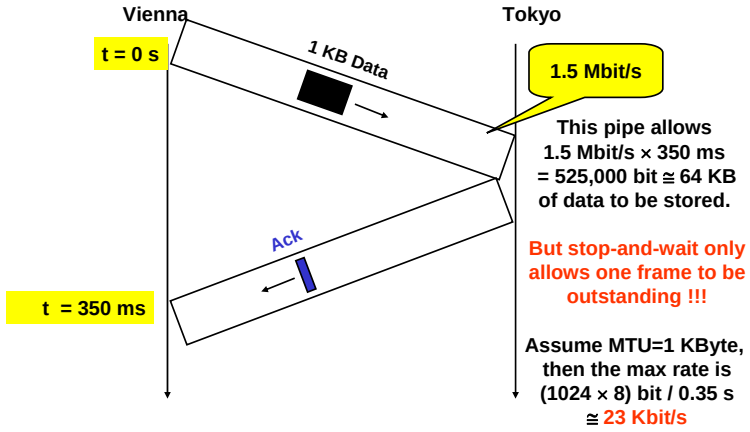


© 2012, D.I. Lindner / D.I. Haas

As already mentioned before the idle-RQ technique is not suited for today's data transport systems. The stop and wait procedure would introduce a lot of delay, especially on long distance connections, and would no be able to efficiently use the network infrastructure.

© 2012, D.I. Lindner / D.I. Haas

Nearly Empty Pipe !



In this scenario we have a 1.5 Mbit/s connection between Vienna and Tokyo.

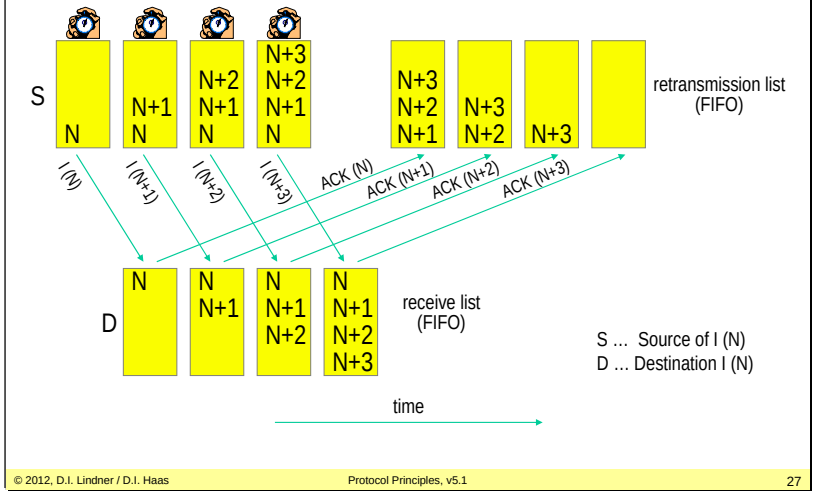
A 1 KByte data frame is sent from Vienna to Tokyo with a transport delay of 175 milliseconds in one direction. With the use of idle-RQ technique it would take at least 350 milliseconds before an acknowledgement is received and the next data frame is sent.

In this case a maximum throughput of only 23 Kbit/s can be achieved if the maximum allowed transmission unit (MTU) = frame length is 1Kbyte.

Agenda

- **Introduction**
- **ARQ Techniques**
  - Introduction
  - Idle RQ
  - Continuous RQ
    - Selective Acknowledgement
    - GoBackN
    - Positive Acknowledgement
    - Selective Reject
  - Sequence Numbers and Windowing
  - Bandwidth-Delay Product
  - Flow Control
  - HDLC Overview

Continuous-RQ Principle



Continuous-RQ protocols were developed for more efficient usage of full duplex lines. Device does not wait for acknowledgements for frames already sent. Full duplex protocol.

Until receipt of acknowledgments, data frames are buffered in a retransmission list. Each incoming acknowledgment removes the corresponding data frame from that list. With a every frame a timer is started to recover from mission acknowledgements.

Receiver stores data frames in receive list in order to detect duplicates and to reorder the sequence. The above pictures shows the principle in case of error-free transmission.

Several variants possible for error-recovery:

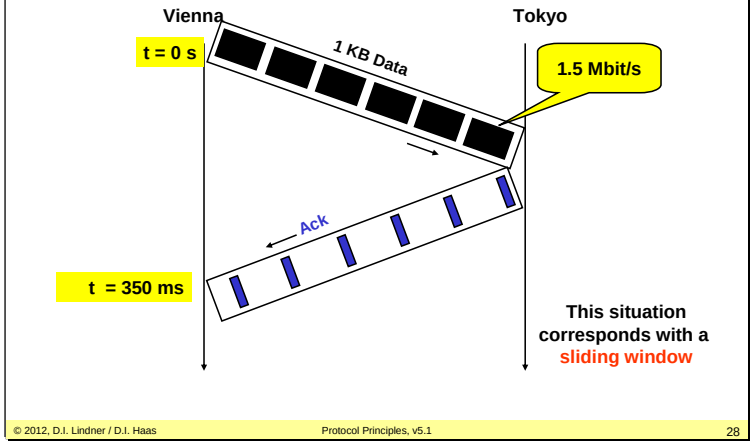
1) Selective acknowledgement (SACK)
   - selective retransmission done implicitly
   - order of frames is not maintained by the procedure
   - e.g. TCP (Transmission Control Protocol) SACK option

2) Multiple and negative acknowledgement
   - also known as GoBackN
   - order of frames is maintained by the procedure
   - e.g. HDLC (High Level Data Link Control) check pointing technique and REJ option
   - e.g. DDCMP (Digital Data Link Control Management Protocol)

3) Positive acknowledgement based on timeout
   - order of frames is not maintained by the procedure
   - e.g. TCP´s original procedure (early TCP)
   - note: today's TCP uses additionally duplicate ACKs to signal the rate of frames leaving the network to the sender

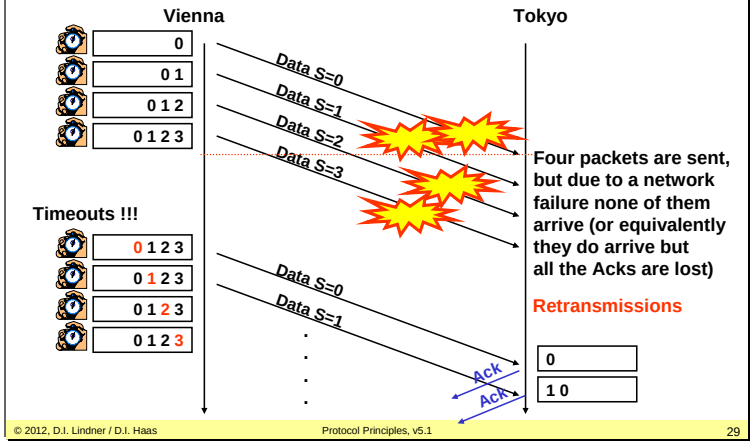
4) Selective reject
   - selective retransmission done explicitly
   - order of frames is not maintained by the procedure
   - e.g. HDLC´s SREJ option

Full Pipe !



By the use of continuous-RQ technique and an proper adjusted send window size it would now be possible to use the complete capacity of our Vienna – Tokyo connection.

Why we need a Retransmission Buffer?



© 2012, D.I. Lindner / D.I. Haas Protocol Principles, v5.1 29

In this example four data frames are sent from Vienna to Tokyo and all of them are lost or the according acknowledgements do not arrive.

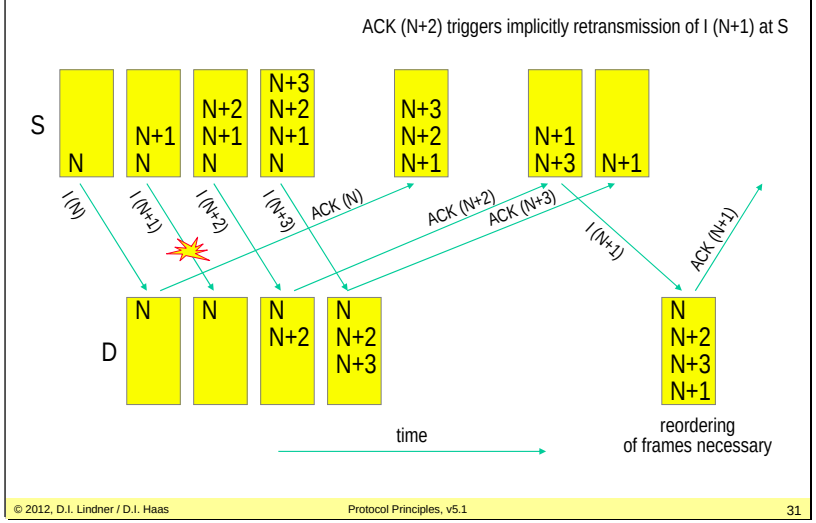
This situation leads to an timeout of the retransmission timer in the Vienna location and causes a retransmission of all four data frames.

Agenda

- **Introduction**
- **ARQ Techniques**
  - Introduction
  - Idle RQ
  - Continuous RQ
    - Selective Acknowledgement
    - GoBackN
    - Positive Acknowledgement
    - Selective Reject
  - Sequence Numbers and Windowing
  - Bandwidth-Delay Product
  - Flow Control
  - HDLC Overview

© 2012, D.I. Lindner / D.I. Haas Protocol Principles, v5.1 30

Selective Acknowledgement (Scenario 1)



Principle of Selective Acknowledgement (SACK):

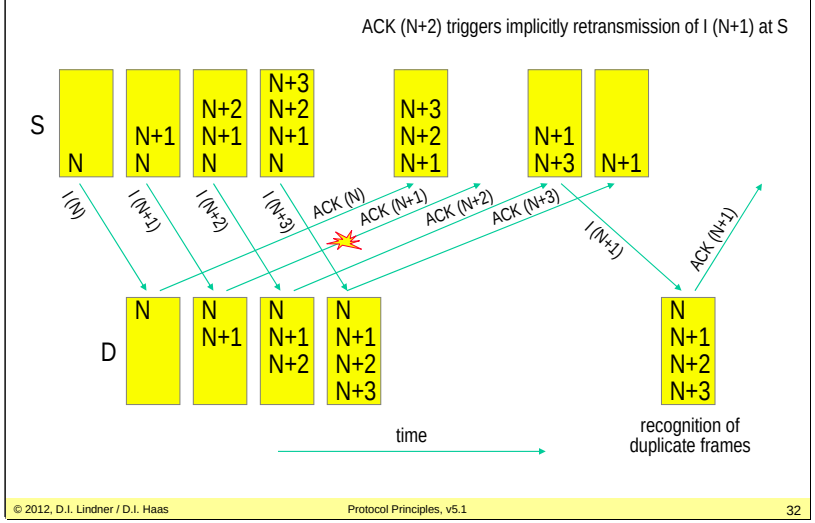
Every single frame that is sent out needs to be acknowledged. As soon as an acknowledgement arrives the according data frame may be deleted out of the retransmission buffer / retransmission list. If a data frame is lost no acknowledgement will be sent for this frame only. This causes again either a retransmission timeout (scenario 4,5) at the sender or may be recognized by receiving ACKs for other successfully delivered data frames (scenario 1 and 2). All three cases lead to an retransmission of the data frame which again is stored at the end of the current retransmission list.

In case of retransmission data frames may not remain in sequence at the receiver (scenario 1, 4), hence reordering is necessary at the receiver and therefore receive buffer and receive list are needed. The receiver is now able to reorder the data frames by the help of the identifiers (sequence numbers before they are handed over to the next higher level software process.

Also receiver must recognize duplicate frames and discard them (scenario 2, 5). Each transmitted data frame starts an individual timer, which will be reset, if acknowledgement is received. If timeout occurs data frame is sent once again (scenario 3).

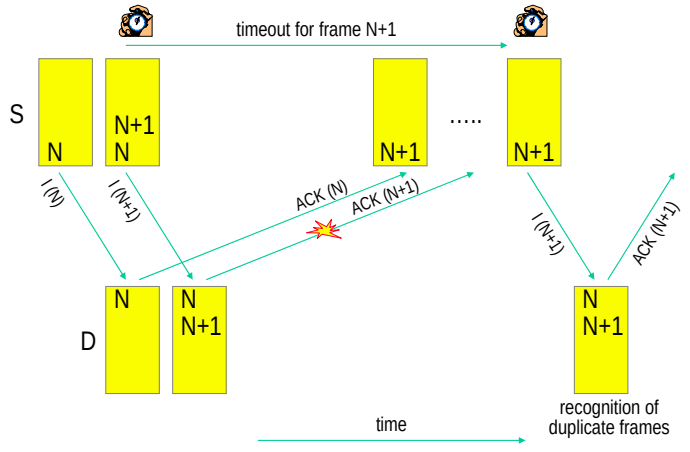
© 2012, D.I. Lindner / D.I. Haas

Selective Acknowledgement (Scenario 2)

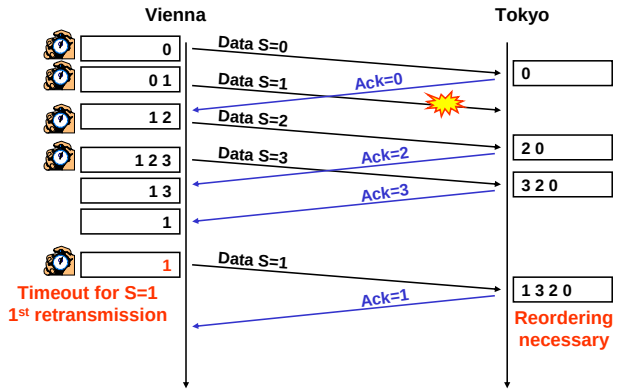


© 2012, D.I. Lindner / D.I. Haas

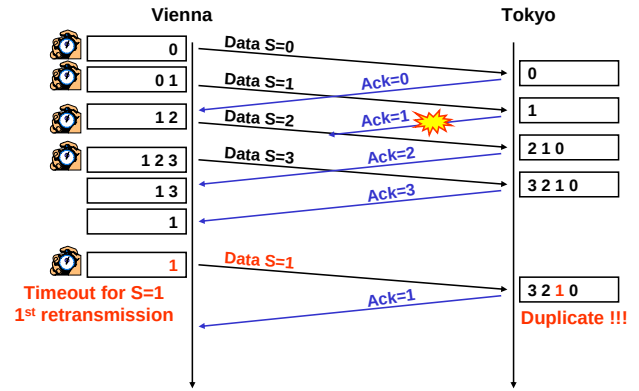
Selective Acknowledgement (Scenario 3)



Selective Acknowledgement (Scenario 4)



Selective Acknowledgement (Scenario 5)



The Selective Acknowledgement technique causes retransmissions even in the case when only the acknowledgement frame is lost. Because if the Ack frame is missing the sender is not allowed to remove the according data frame from the send buffer.

But nevertheless duplicate data frames are recognized by the receiver with the help of the sequence numbers.

SACK in TCP

- **Application:**
  - New option for TCP to accommodate to long fat pipes with high BER (Bit Error Rate)
  - Part of modern TCP
- **Optionally, retransmissions might be sent immediately when unexpected (the next but one) ACK occurs**
- **Opposite idea: Cumulative ACK**

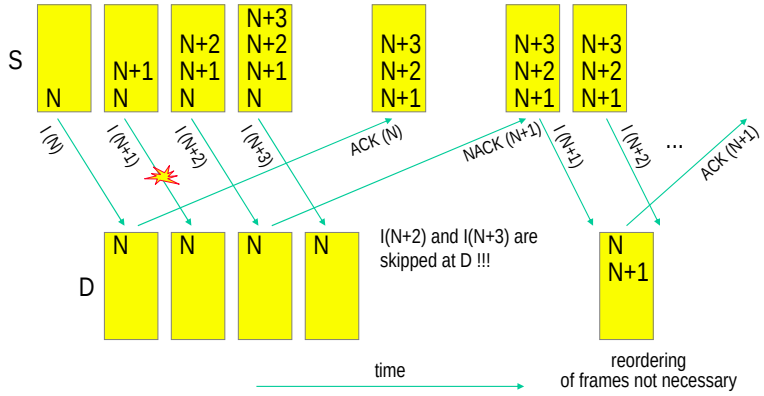
This SACK technology is nowadays also supported by modern TCP protocol stacks and its use is negotiated during connection setup.

Optionally retransmissions may occur immediately when an unexpected ACK is received.

Agenda

- **Introduction**
- **ARQ Techniques**
  - Introduction
  - Idle RQ
  - Continuous RQ
    - Selective Acknowledgement
    - GoBackN
    - Positive Acknowledgement
    - Selective Reject
  - Sequence Numbers and Windowing
  - Bandwidth-Delay Product
  - Flow Control
  - HDLC Overview

GoBackN (Scenario 1)



Principle of GoBackN:

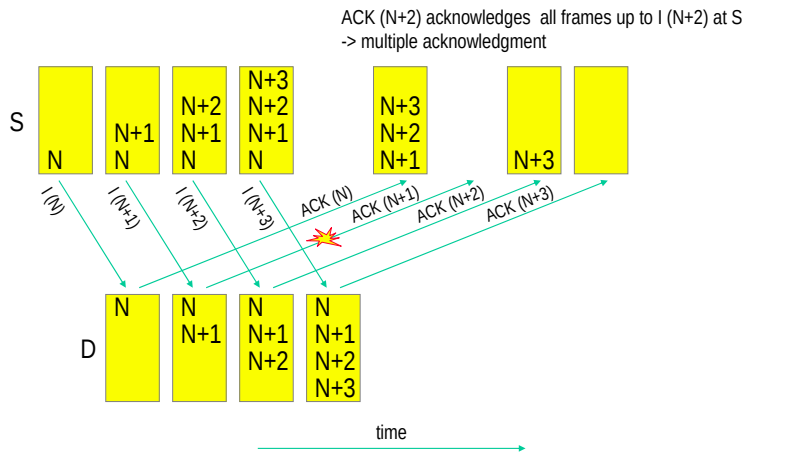
In case of errors, all data frames since “N” will be requested again by NACK(N) (Negative Acknowledgement) -> scenario 1.

All following frames are discarded by the receiver until frame with correct sequence number arrives. Therefore reordering is not necessary and SW at receiver could be kept more simple.

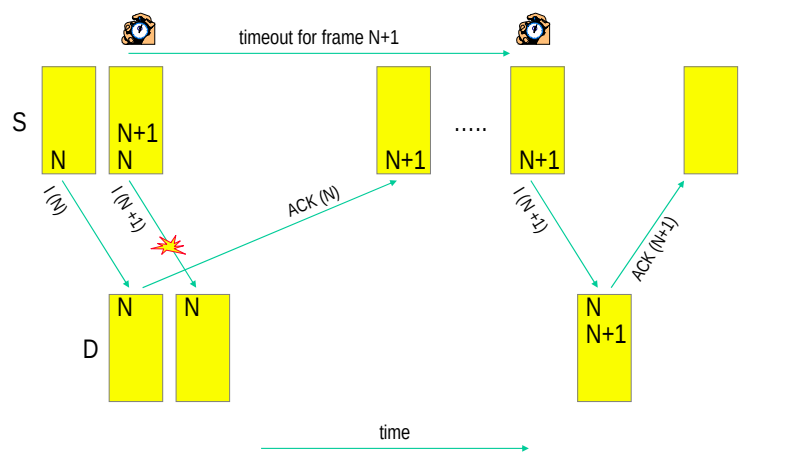
A single acknowledgments could confirm multiple data frames (multiple acknowledgement) which is often used to spare number of Acks in opposite direction -> scenario 2.

Each transmitted data frame starts an individual timer, which will be reset, if acknowledgement is received. If a timeout occurs data frame is sent once again -> scenario 3, 4.

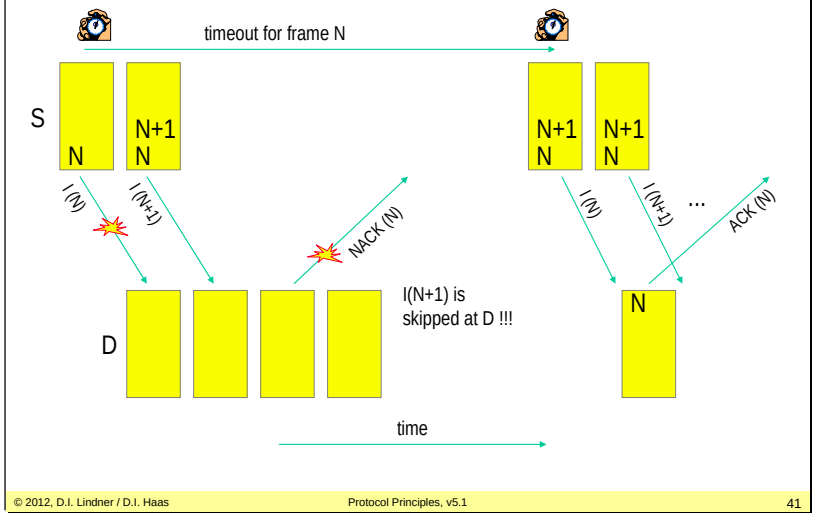
GoBackN (Scenario 2)



GoBackN (Scenario 3)

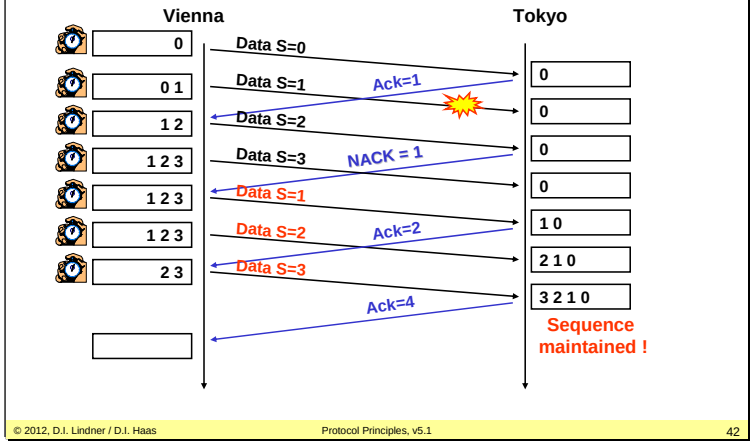


GoBackN (Scenario 4)



© 2012, D.I. Lindner / D.I. Haas Protocol Principles, v5.1 41

GoBackN with Sequence Numbers



© 2012, D.I. Lindner / D.I. Haas Protocol Principles, v5.1 42

In GoBackN ARQ technique data frames are only acknowledged when the series of received sequence numbers is continuous.

In case sequence numbers are used as identifiers the following principle is applied in all GoBaCKN implementation:

ACK number contains the number of the next to be expected frame at the receiver side. ACK 1 means: Receiver acknowledged frame 0 and waits for frame 1.

In the above example the data frame with the sequence number S = 1 is lost. The receiver recognizes that a data frame is missing when he receives the frame with the sequence number S = 2. Now an so called Not Acknowledgement (NACK) data frame is sent back to the sender. The sender recognizes which frame was lost by the sequence number carried in the NACK data frame.

Starting with this sequence number all data frames are retransmitted. This causes more overhead than by the use of SACK but makes sure that the order of the data frames is maintained.

GoBackN - Facts

- **Maintains order at receiver-buffer**
  - Reordering was too much time-consuming in earlier days
- **Still used by**
  - HDLC and clones ("REJECT")
  - TCP
    - Variant known as "fast retransmit"
    - Uses duplicate Acks as NACK

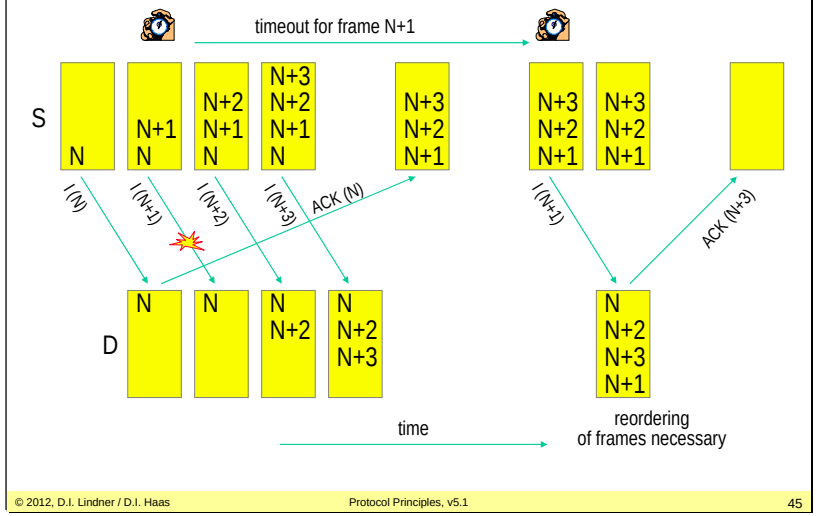
The GoBackN procedure is used by the quite old HDLC protocol because reordering of data packets was a too much time and memory consuming task for processors in the early days of data communication.

Also in today's TCP implementation a variant of the NACK procedure is used, the duplicate ACK. As soon as an TCP Stack recognizes that a data frame is missing it sends out an duplicate acknowledgement. So the sender recognizes that a data frame was lost and retransmits the according data frame. This speeds up the error recovery procedure because the retransmission timeout is omitted in his case.

Agenda

- **Introduction**
- **ARQ Techniques**
  - Introduction
  - Idle RQ
  - Continuous RQ
    - Selective Acknowledgement
    - GoBackN
    - Positive Acknowledgement
    - Selective Reject
  - Sequence Numbers and Windowing
  - Bandwidth-Delay Product
  - Flow Control
  - HDLC Overview

Positive Acknowledgement (Scenario 1)

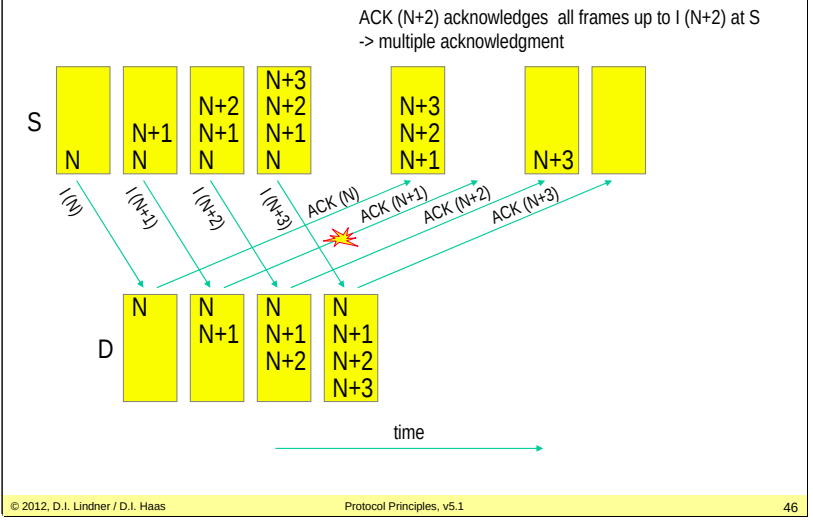


Principle of Positive Acknowledgement:

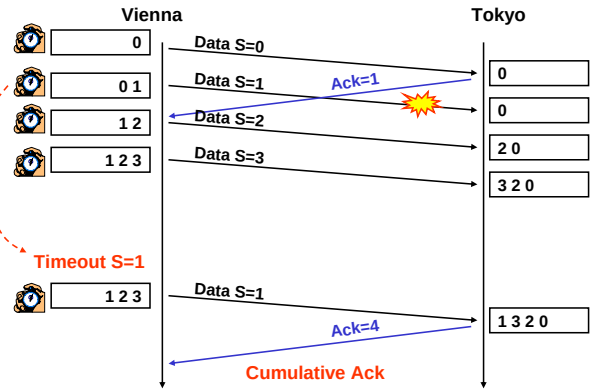
Data frames will be confirmed as long as frames arrives in sequence. Multiple acknowledgement can be used by receiver. If an ACK is lost the transmitter can use receiving of the next ACK as confirmation that everything moved on well until this frame (scenario 2).

If data frames get out of sequence, confirmation is stopped. Nevertheless, all following data frames will be stored. Each transmitted data frame starts an individual timer, which will be reset if acknowledgement is received. If timeout occurs data frame is sent once again. Timeouts of following unacknowledged frames are freezed by the sender to first probe what is going on. When the missing data frame arrives the receiver confirmed with multiple acknowledgements the already frames (scenario 1).

Positive Acknowledgement (Scenario 2)



Positive ACK with Sequence Numbers



© 2012, D.I. Lindner / D.I. Haas

In the positive ARQ technique an ACK is only sent when the order of the arriving data frames is correct.

In case sequence numbers are used as identifiers the following principle is applied in all Positive ACK implementation:

ACK number contains the number of the next to be expected frame at the receiver side. ACK 1 means: Receiver acknowledged frame 0 and waits for frame 1.

In the example above we find that the data frame with the sequence number S = 1 gets lost. The next frame that arrives is the data frame with S = 2. The receiver recognizes discontinuous sequence numbers and stops to transmit ACK frames. In the meantime all other frames are received and stored in the receive buffer but are not acknowledged.

When the data frame S = 1 falls into timeout, it is retransmitted. The receiver recognizes that the missing data frame has arrived and launches an ACK = 4 frame to acknowledge all till now received frames.

Obviously it may happen that another data frame is retransmitted depending on the remaining timeout period and the RTT of the connection.

© 2012, D.I. Lindner / D.I. Haas

Positive ACK - Facts

- **Always together with cumulative ACKs**
  - Any frame received is buffered
  - Receiver must be able to reorder
- **Problem:**
  - Only timeouts trigger retransmission
- **Application:**
  - Early (Original) TCP

© 2012, D.I. Lindner / D.I. Haas

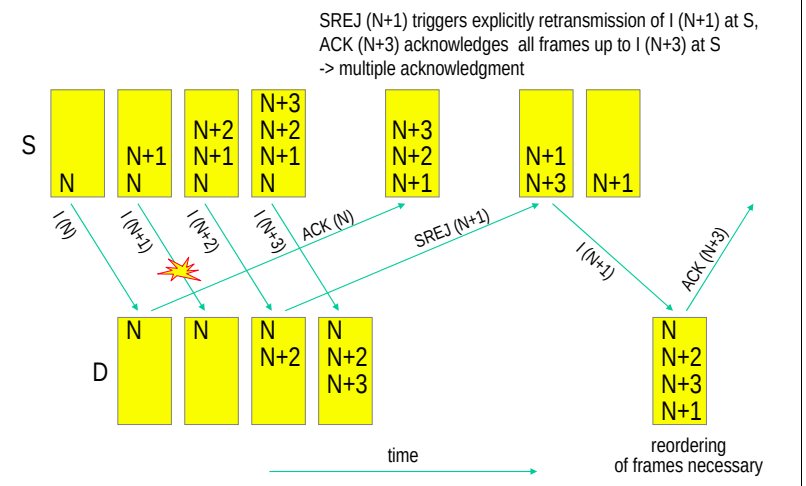
The positive ACK procedure is always used together with cumulative acknowledgement technology. Disordering of data frames may occur and must be fixed. Only timeouts trigger retransmissions of lost data frames.

© 2012, D.I. Lindner / D.I. Haas

Agenda

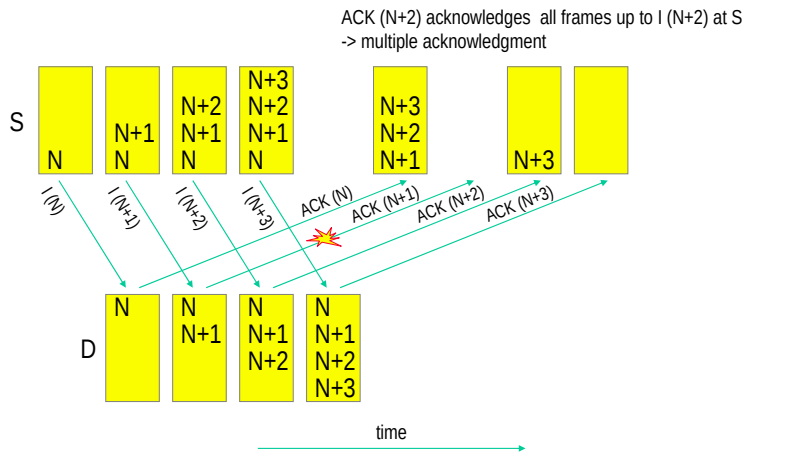
- **Introduction**
- **ARQ Techniques**
  - Introduction
  - Idle RQ
  - Continuous RQ
    - Selective Acknowledgement
    - GoBackN
    - Positive Acknowledgement
    - Selective Reject
  - Sequence Numbers and Windowing
  - Bandwidth-Delay Product
  - Flow Control
  - HDLC Overview

Selective Reject (Scenario 1)

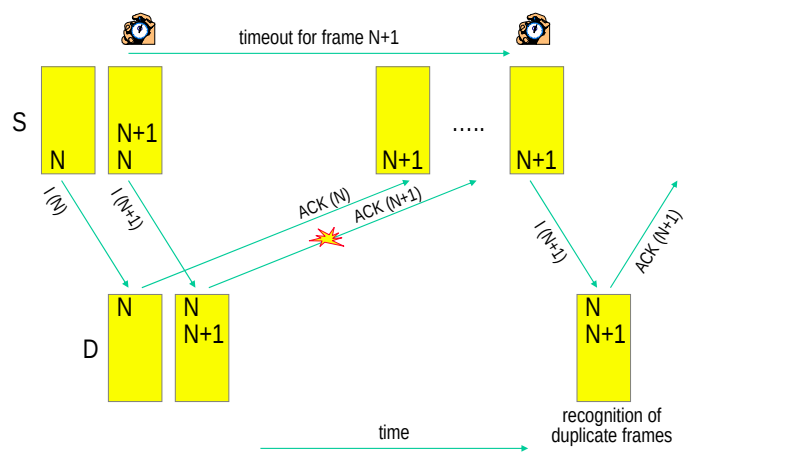


Principle of Selective Reject: Data frames will be confirmed as long as frames arrives in sequence and multiple acknowledgement can be used by receiver (scenario 2). In case of an error, only the data frame causing the error will be requested explicitly through SREJ(N) by the receiver. In case of retransmission data frames may not remain in sequence (scenario 1). Each transmitted data frame starts an individual timer which will be reset, if acknowledgement is received. If a timeout occurs data frame is sent once again (scenario 3)

Selective Reject (Scenario 2)



Selective Reject (Scenario 3)



Selective Reject - Facts

- **Modern modification of GoBackN**
- **Only those frames are retransmitted that receive a SREJ**
  - Or those that time out
- **Receiver must be able to reorder frames**
- **Application:**
  - Optional for modern HDLC clones
  - SREJ control frame

A combination of different ARQ procedures is the SACK procedure with the use of NACK frames. In this technique only data frames for which an NACK frame is received or those that time out will be retransmitted.

The correct order of the data frames might get lost.

Agenda

- **Introduction**
- **ARQ Techniques**
  - Introduction
  - Idle RQ
  - Continuous RQ
    - Selective Acknowledgement
    - GoBackN
    - Positive Acknowledgement
    - Selective Reject
  - Sequence Numbers and Windowing
  - Bandwidth-Delay Product
  - Flow Control
  - HDLC Overview

Sequence Number

- **Identifiers are implemented by increasing numbers**
- **The number used in I-frames**
  - Send sequence number N(S)
- **The number used in ACK/NACK/SREJ-frames**
  - Receive sequence number N(R)
  - Shows number of the frame which is expected next by the receiver !!!
- **Register variables are necessary for protocol implementation in the finite state machine of sender and receiver**
  - V(S), V(R)
  - Must be initialized (set to 0) by connection setup

© 2012, D.I. Lindner / D.I. Haas Protocol Principles, v5.1 55

Elements, handling and meaning of sequence number techniques for GoBackN:

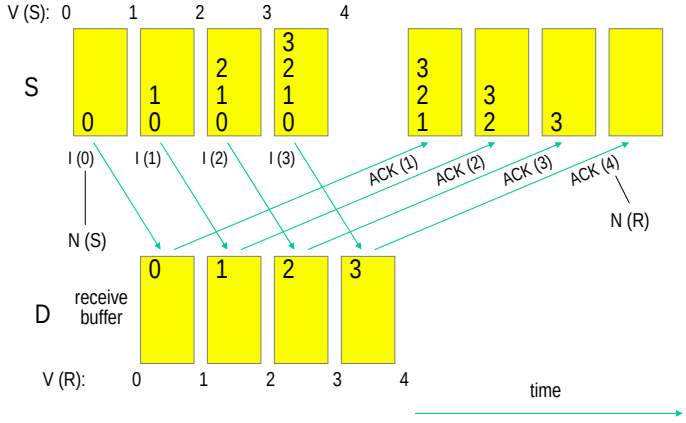
V(S) indicates the sequence number of the next I-frame that will be sent

V(R) indicates the expected sequence number of the next in-sequence I-frame to be received -> this value will be seen in N(R)

Prior to sending an I-frame, the value of N(S) is set to the value of V(S) -> afterwards V(S) is increased by one

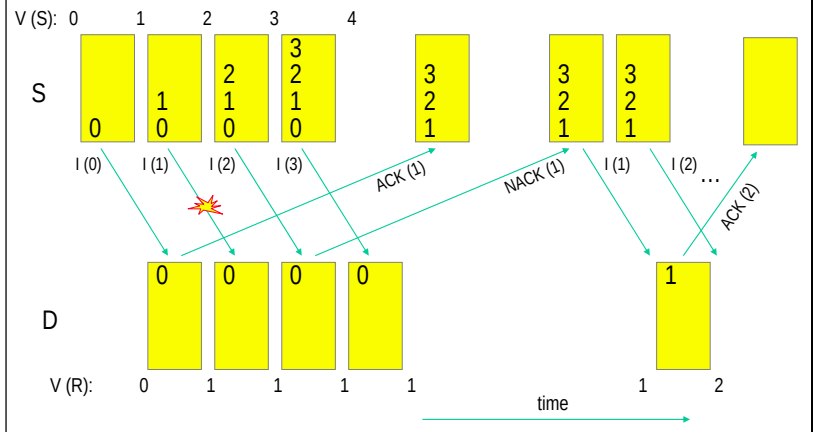
Receiver only accepts I-frames with N(S) = V(R) -> after successful receipt of a frame V(R) will first be increased by one and then acknowledgment with N(R) = V(R) will be sent -> therefore receipt of ACK with N(R) = x means that all I-frames until x-1 are confirmed

GoBackN 1

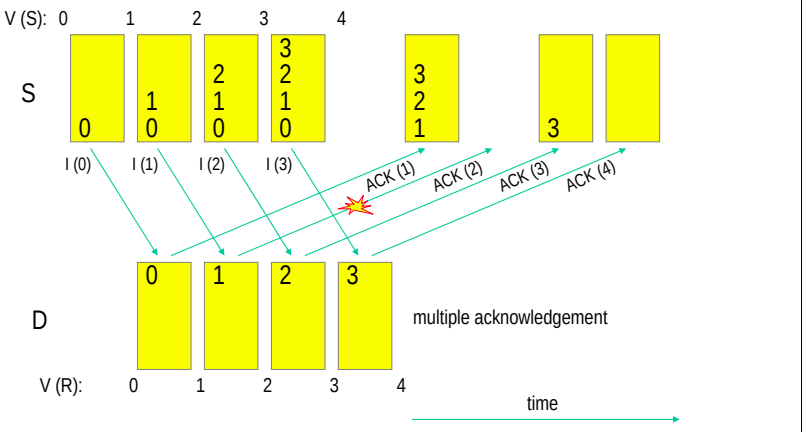


© 2012, D.I. Lindner / D.I. Haas Protocol Principles, v5.1 56

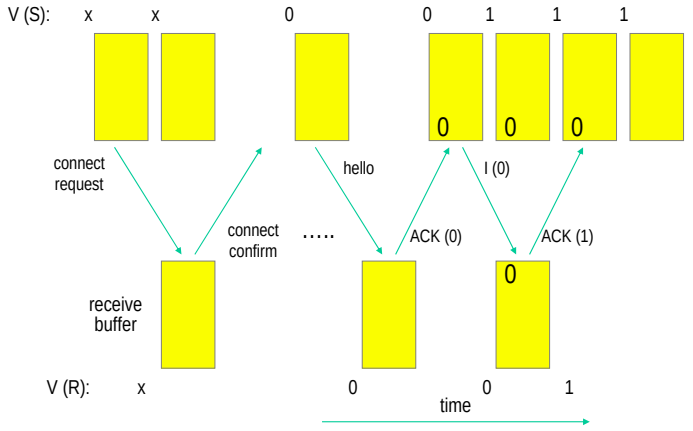
GoBackN 2



GoBackN 3



Connection Initializing / Keepalive at the Beginning Reason why Nr(1) acknowledges frame I(0) !

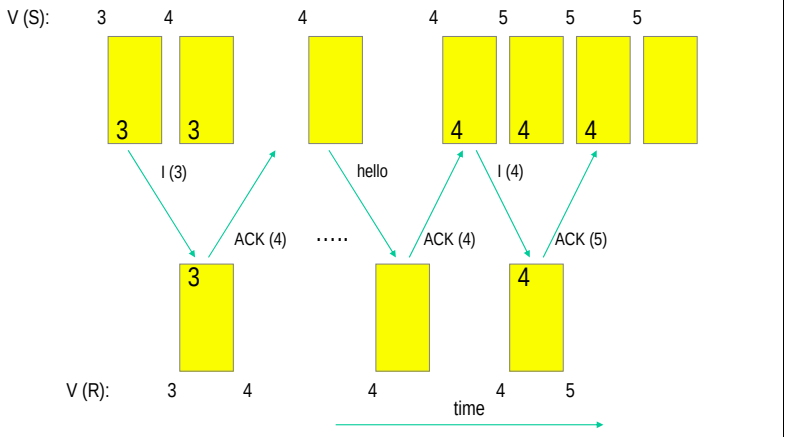


Connection establishment initializes V(S) and V(R) to zero.

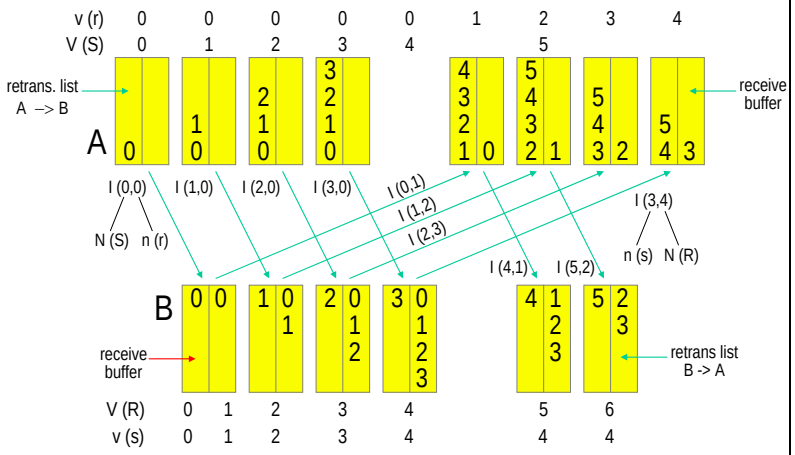
After connection establishment connection is maintained by keepalive messages during transmission pauses.

Example for such keepalive technique: Exchange of HELLO (request / poll) and ACK (response). But now ACK is used for acknowledgment as well as for connection maintenance. In order to differentiate between the ACK of an already successfully delivered data frame and a keepalive frame -> an ACK(N) only confirms all frames up to N-1 and not up to N. Sender can now distinguish between a keepalive response and acknowledgement of a data frame.

Keepalive in Between



Piggyback Acknowledgement



Until now ACKs control frames for received data frame were only seen in one direction (data flow streams in the opposite direction). If there is data flow in both directions dedicated ACK frames produce unnecessary overhead. Therefore acknowledgments contained in data frames in opposite direction can avoid that overhead –> **Piggyback Acknowledgement**.

Of course if no backward data frames are waiting for transmission still dedicated ACK frames will be used.l

Picture shows data flow in both directions:

Data frames contain both, there own send sequence number and the receive sequence number for acknowledgment reasons of the backward direction data streams. Now both communication devices will have V(S)- and V(R)-registers, retransmission and receive lists.

N(S), N(R), V(S) und V(R) used for control of data transfer from A to B

n(s), n(r), v(s) und v(r) used for control of data transfer from B to A

Timers - Retransmission Timeout

- **The value for retransmission timeouts for line protocols can be easily calculated using the following parameters**
  - Bitrate
  - Maximum data frame size
  - Worst case time at receiver to generate an acknowledgment
  - Size of acknowledgment frame
- **Calculation for network protocols with varying transmission delays is more complex**
  - Adaptive process is necessary

Send Window and Sliding Window

- **Continuous-RQ techniques would require infinite sender/receiver buffer size (also infinite number of identifiers)**
  - If we do not restrict the number of unacknowledged frames
- **Send Window W**
  - Is the maximum allowed number of unacknowledged frames in the retransmission list
- **Necessary sender-buffer size is W * maximum frame size**
  - Also called the **Window Size**
- **Handling procedure**
  - Is called **(Sliding) Windowing**

© 2012, D.I. Lindner / D.I. Haas

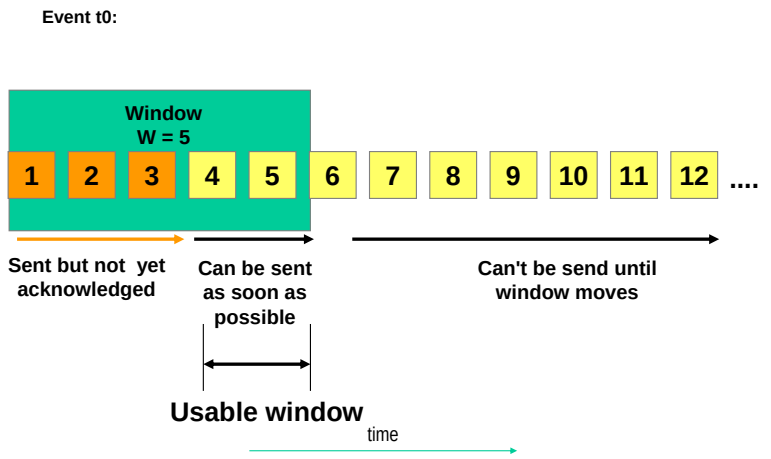
The sender needs to buffer all transmitted frames until the according acknowledgement arrives.

Windowing controls how many frames can be sent out in a sequence without waiting for the corresponding acknowledgements. Implemented by moving a “window” along the sequence number ray. If send window limit W is reached, sending of additional data frames is stopped until receipt of acknowledgement indicates that window is opened again (meaning the window moves on along the sequence number ray.

Elder protocols use a fixed send window or window size negotiated during connection setup. More modern protocols such as TCP use a method called adaptive windowing which allows to automatically adapt the window size to needs of the transport system.

© 2012, D.I. Lindner / D.I. Haas

Sliding Window Basics (1)

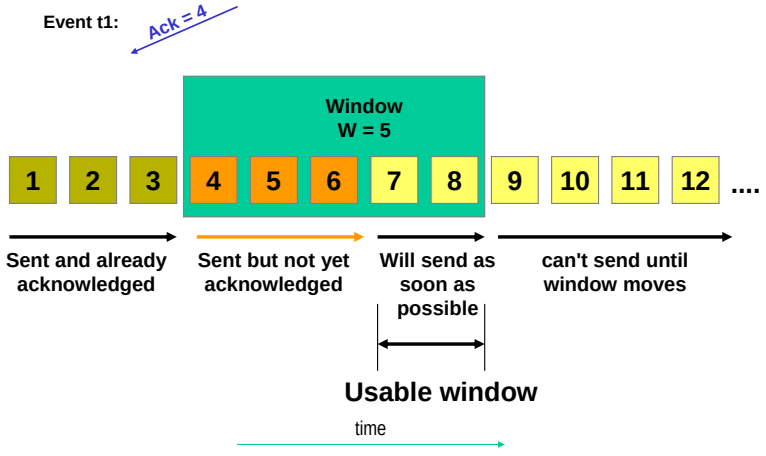


© 2012, D.I. Lindner / D.I. Haas

In this example a fixed window size of five is used. This means that up to five data frames without explicit acknowledgments may be sent. At the start time t0 we sent out three frames.

© 2012, D.I. Lindner / D.I. Haas

Sliding Window Basics (2)



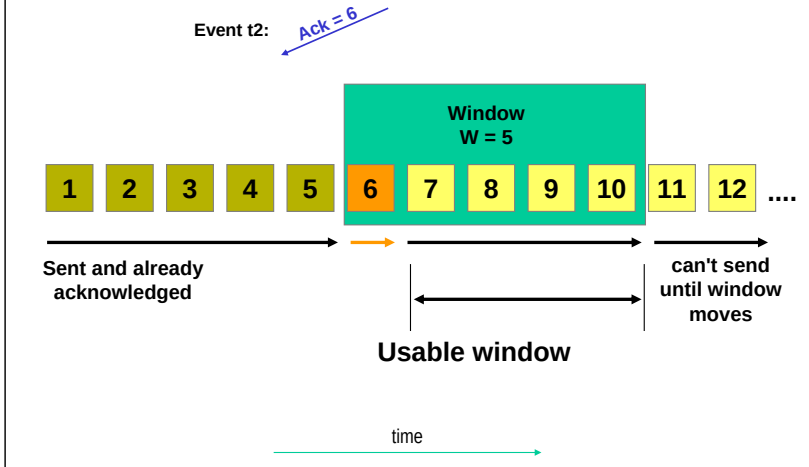
© 2012, D.I. Lindner / D.I. Haas

As soon as the first ACK = 4 (t1) arrives the sliding window moves to the left. This is possible because the data frames 1,2 and 3 are now removed out of the send buffer. Now the data frames 4, 5 and 6 are sent and kept in the send buffer until their acknowledgements arrive.

On the right side of the window we find the data frames that will be sent in the future, while on the left side of the sliding window already sent and acknowledged frames can be found. So the window moves continuously from right to left therefore the name sliding window.

© 2012, D.I. Lindner / D.I. Haas

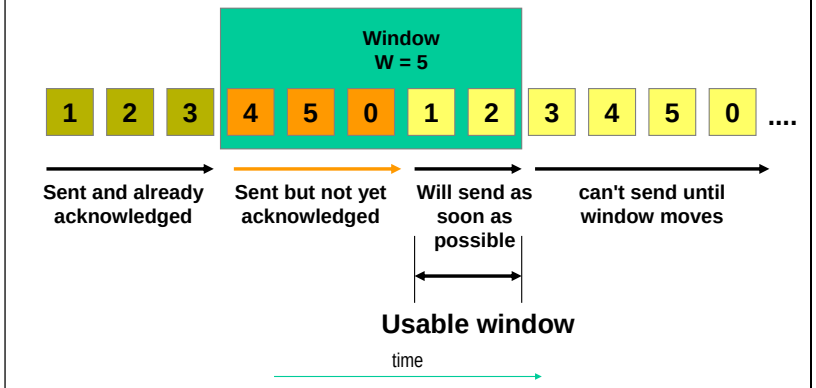
Sliding Window Basics (3)



© 2012, D.I. Lindner / D.I. Haas

© 2012, D.I. Lindner / D.I. Haas

Windowing with Numbering Modulo W+1



Additional Consequences of Windowing:

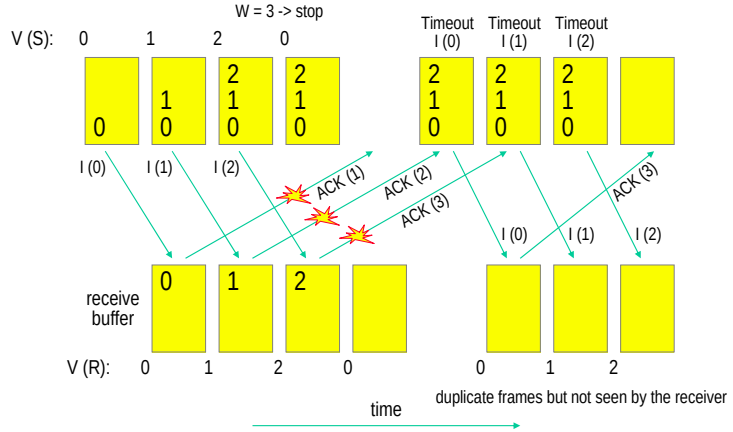
Windowing also reduces the buffer memory for the receive buffer -> Buffer size of receive list = W * maximum frame size.

Windowing reduces the number of necessary identifiers / sequence number range -> Numbering of data frames can be done by a modulo operation.

e.g. GoBackN with send window = W needs W+1 identifiers because of worst case scenario shown next slide and hence data frames can be numbered modulo W+1

e.g. Selective Acknowledgement with send window = W needs 2W identifiers because of worst case scenario and hence data frames can be numbered modulo 2W

Worst Case Scenario with GoBackN and W=3



Required minimum number of identifiers for W = 3:

Assume we have a window size W=3 and the same number of identifiers (sequence numbers 0..2). When we send three frames at once the send window is full after sending of I-frames 0, 1 and 2. Now unfortunately all corresponding acknowledgements get lost, our timers expire and we must retransmit those frames. Now we again send frames 0..2 but the receiver cannot recognize them as the second incarnation. So the duplicates can not be detected by the receiver Thus, the first real new frame must have a sequence number of 3, that is the next frames have identifiers 3, 0, 1, 2. The result is that we need at least W+1 identifiers.

Worst case scenario for Selective Acknowledgement method is based on another consideration:

Another tricky scenario might happen. Assume we have a sequence number space 0..7 (8 identifiers) and W=7. We send frames 0,1,2,3,4,5,6 and get all acknowledgements. Then we send 7, 0, 1, 2, 3, 4, 5 but now frame 7 get lost. How can the receiver be sure that frames 0,1,2,3,4,5 are not just retransmitted frames from the first burst? In GoBackN that is not a problem, because if frame 7 does not arrive at the receiver, the receiver sends a NACK 7 and ignores everything until 7 is successfully retransmitted. You remember GoBackN is avoiding reordering at the receiver by forcing frames to come in sequence. In SACK techniques there is an ambiguity in such situations. Therefore we have to use a smaller window size W=4. Now we send frames 0,1,2,3 and get all acknowledgements. Then we send 4,5,6,7. The receiver is not confused. So the final rule in these case requires W to be smaller than half the number of identifiers.

Therefore for Selective Acknowledgement with send window = W it needs 2W identifiers and data frames can be numbered modulo 2W.

Agenda

- **Introduction**
- **ARQ Techniques**
  - Introduction
  - Idle RQ
  - Continuous RQ
    - Selective Acknowledgement
    - GoBackN
    - Positive Acknowledgement
    - Selective Reject
  - Sequence Numbers and Windowing
  - Bandwidth-Delay Product
  - Flow Control
  - HDLC Overview

How Large should be the Window Size?

- **Window size depends on**
  - Bandwidth (bit rate) of the communication channel
  - Round-Trip-Time (RTT)
    - = 2 x (propagation + serialization) delay plus response delay of partner
  - (Available buffer size transmitter/receiver)
- **The optimum window size**
  - must be big enough so that the sender can fully utilize the channel volume which is given by the

    Bandwidth-Delay Product
  - Window size in Bytes >= BW × RTT
  - If smaller: jumping window
  - Extreme case: Idle-RQ with W=1

The optimum window size directly depends on the bandwidth of the channel as well as on the Round-Trip-Time.

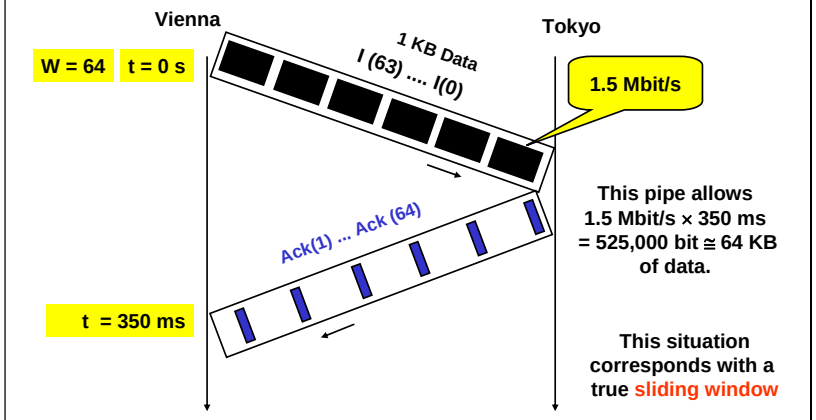
Optimal window size for Continuous-RQ will allow acknowledgments to arrive just in time to keep the window always open hence using sliding window all the time.

If the window size is smaller then the optimum the transmission will be stopped until acknowledgments arrive -> jumping window.

Idle RQ behaviour is the worst case with W = 1.

On the other side if window size is too large (larger as determined by the bandwidth-delay product) in case of errors many good frames may be retransmitted (see GoBackN)

Full Pipe with Continuous-RQ and W = 64



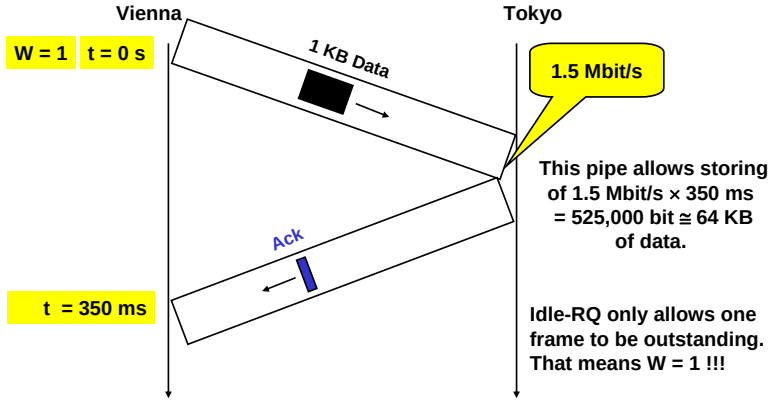
Continuous-RQ (C-RQ) allows several frames to be outstanding. With 1 KByte maximum Frame Size optimum would be W = 64 !!!

© 2012, D.I. Lindner / D.I. Haas

By the use of continuous-RQ technique and an proper adjusted send window size it would now be possible to use the complete capacity of our Vienna – Tokyo connection.

© 2012, D.I. Lindner / D.I. Haas

Nearly Empty Pipe with Idle-RQ and W = 1



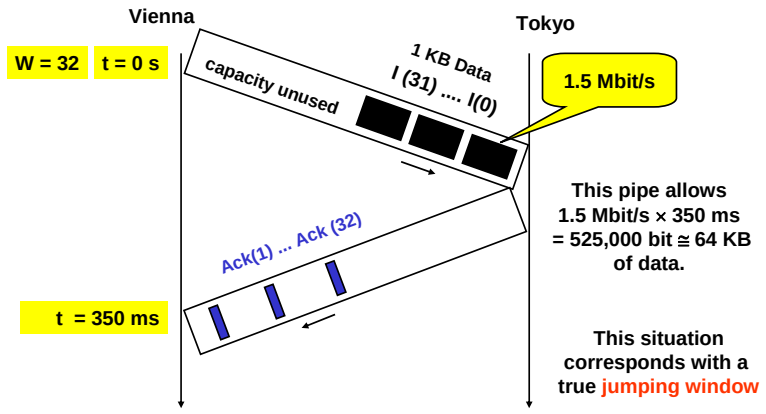
Assume 1 KByte (1024 byte) maximum frame size, then the maximal achievable rate is (1024 × 8) bit / 0.35 s ≅ 23 kbit/s

© 2012, D.I. Lindner / D.I. Haas

.

© 2012, D.I. Lindner / D.I. Haas

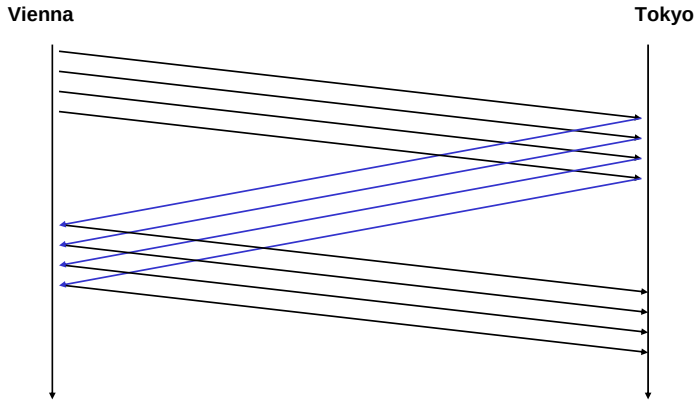
Only Half Pipe Used with C-RQ and W = 32



With 1 KByte maximum Frame Size and W = 32
the pipe could be used only half the time !!!

.

Jumping Window



In this example we find what will happen if the window size is to small.

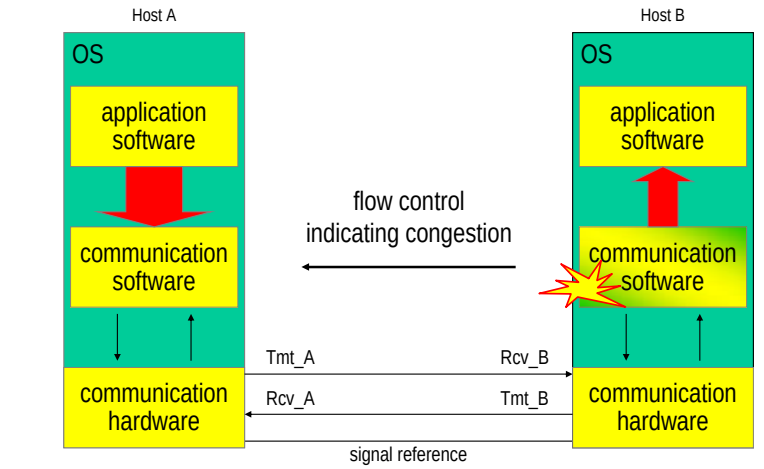
The sender in Vienna has sent out all data frames until he reached the max window size. Now the sender in Vienna has to wait for acknowledgments to arrive.

The incoming acknowledgements free up buffer space and the sender may continue to send. In this scenario the chosen window size is obviously to small leading to an insufficient use of the transport capacity.

Agenda

- **Introduction**
- **ARQ Techniques**
  - Introduction
  - Idle RQ
  - Continuous RQ
    - Selective Acknowledgement
    - GoBackN
    - Positive Acknowledgement
    - Selective Reject
  - Sequence Numbers and Windowing
  - Bandwidth-Delay Product
  - Flow Control
  - HDLC Overview

Flow Control



If data frames arrive faster than application is able to process,

Receiver runs out of available buffer storage and good frames must be discarded by the receiver

Discarded data frames will cause retransmission but they will be still discarded because of lack of buffers

Therefore receiver should control the rate of transmission of data frames

Flow control

Overload/congestion situation indicated to the sender using flow control messages

Sender stops and waits until receiver is able to process frames again

Flow Control Possibilities

- **Misuse of traditional windowing**
  - Constant send window during session time
  - Receiver stops acknowledgements in case of congestion
  - Not a good approach
- **Separate flow control frames**
  - Stop and Go indications
- **Adaptive windowing**
  - Variable send window during session time
  - Implicitly contains Stop and Go indication

© 2012, D.I. Lindner / D.I. Haas

Static windowing with constant/fixed window size may be used to implement flow control in such a way that receiver does not generate acknowledgements in case of congestion and therefore sender will stop transmission after send window is closed.. Problem with static windowing: After timeout unconfirmed frames will be retransmitted but after a defined amount of unsuccessful retransmissions, the connection is considered to be broken and will be closed by the protocol.

Flow control to prevent buffer overflow conditions at the receiver side can be implemented either by the use of some kind of stop and go procedure or by the use of adaptive windowing.

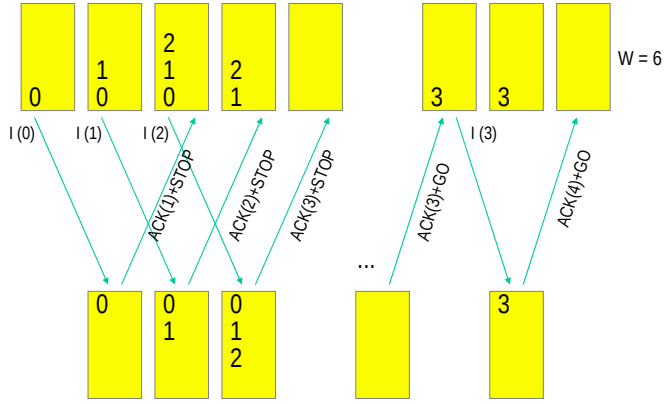
The Stop and Go technique, which is mainly used by some elder protocols like HDLC and its derivates, uses some special control frames Receiver Ready (RR) and Receiver Not Ready (RNR). These protocols typically are using a fixed maximum window size of 7 or 127 frames.

There is also a new approach 802.3x to implement flow control in Ethernet technology which allows an congested device to specify transmission pauses in terms of bit times.

Adaptive windowing technique, which is used by the TCP protocol stack, announces the receivers available buffer size within the acknowledgements that are send back to the sender side. So the window size changes dynamically dependent on the buffer conditions at the receiver.

© 2012, D.I. Lindner / D.I. Haas

Flow Control: Stop and Go



© 2012, D.I. Lindner / D.I. Haas

Typically used flow control messages are

STOP, e.g. HDLC´s Receiver Not Ready (RNR)

GO, e.g. HDLC´s Receiver Ready (RR)

In case of congestion receiver signals STOP. On receipt of STOP sender will suspend transmitting. In the worst case, sender will use the send window before stopping transmission. Receiver signals GO when data flow can resume. STOP and GO may contain N(R) and hence can be used for acknowledgement too.

In case of full duplex data communication STOP and GO control frames are used – of course - for flow control in both directions. In such cases STOP and GO frames may further be used for connection management in order to implement keepalive procedures. If no data frames are waiting for transmission a periodic GO frame can signal keepalive to the partner. If traffic was suspended by STOP a periodic repetition of STOP can still signal keepalive to the partner. In both cases these STOP and Go maintain the connection state.

© 2012, D.I. Lindner / D.I. Haas

Flow Control: Adaptive Windowing

- **Window size could be**
  - Constant or dynamic during lifetime of a connection
    - Constant window size is used e.g. by HDLC, X.25
- **If window size is dynamic**
  - A start value is negotiated during connection establishment
  - Actual window size will be dynamically adjusted to an optimal value
    - Receiver continuously advertises optimal value (e.g. based on availability of free buffer memory)
    - Advertised window size = 0 -> STOP
    - Advertised window size > 0 -> GO
  - Adaptive windowing (e.g used by TCP)

Agenda

- **Introduction**
- **ARQ Techniques**
  - Introduction
  - Idle RQ
  - Continuous RQ
    - Selective Acknowledgement
    - GoBackN
    - Positive Acknowledgement
    - Selective Reject
  - Sequence Numbers and Windowing
  - Bandwidth-Delay Product
  - Flow Control
  - HDLC Overview

What is HDLC ?

- **High-Level Data Link Control**
- **Early link layer protocol**
- **Based on SDLC (Synchronous-DLC, IBM)**
  - Access control on half-duplex modem-lines
  - Connection-oriented or connectionless
  - Framing
  - Frame Protection
- **Building elements**
  - Synchronous transmission
  - Bit-oriented line protocol using bit-stuffing
  - Continuous RQ with GoBackN, piggybacked ACK
  - P/F procedure for access control and check-pointing
  - Flow Control based on STOP and GO
- **Mother of many LAN and WAN protocols**

© 2012, D.I. Lindner / D.I. Haas

SDLC was created in the mid-1970s to carry SNA (Systems Network Architecture) traffic and supports line speeds up to 64 kbit/s. It was the first bit-oriented synchronous link-layer protocol. SDLC is used for DLSw and Advanced Peer-to-Peer Networking (APPN). ISO adopted and modified SDLC and called it HDLC.

HDLC is the mother of the most LAN and WAN protocols !

HDLC standardization was done by ISO. HDLC covers a broad range of applications. Therefore HDLC has been used as a basis for a number of other data link layer protocols.

Relevant standards are:

ISO 3309 – HDLC Frame Structure
ISO 4335 – HDLC Elements of Procedure
ISO 7478 – HDLC Multilink Procedures (MLP)
ISO 7809 – HDLC Class of Procedures
ISO 8885 – HDLC Exchange Data Link Identification (XID)

Family protocols:

ISO 7776 LAPB

ISO 8471 LAPB address information

ISO 8802/2 LLC

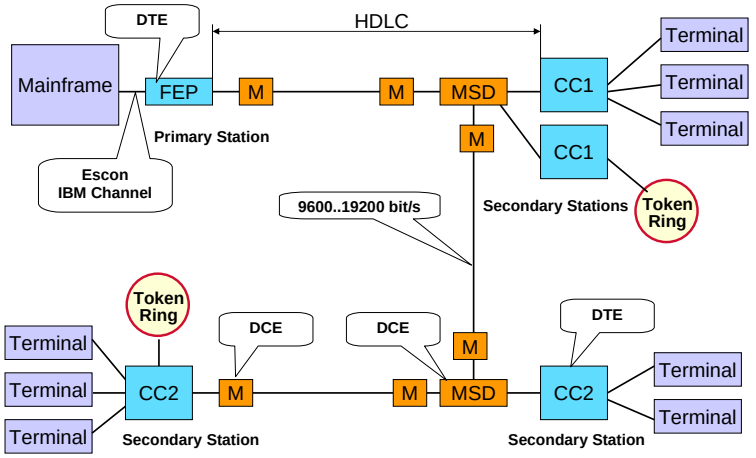
ITU-T I.441 LAPD

© 2012, D.I. Lindner / D.I. Haas

Datenkommunikation 384.081 - SS 2012

L02 - Protocol Principles (v5.1)

Early HDLC Example



© 2012, D.I. Lindner / D.I. Haas

The slide above shows a typical HDLC application. Many terminals which are connected to Cluster Concentrator.

Legend:

M = Modem

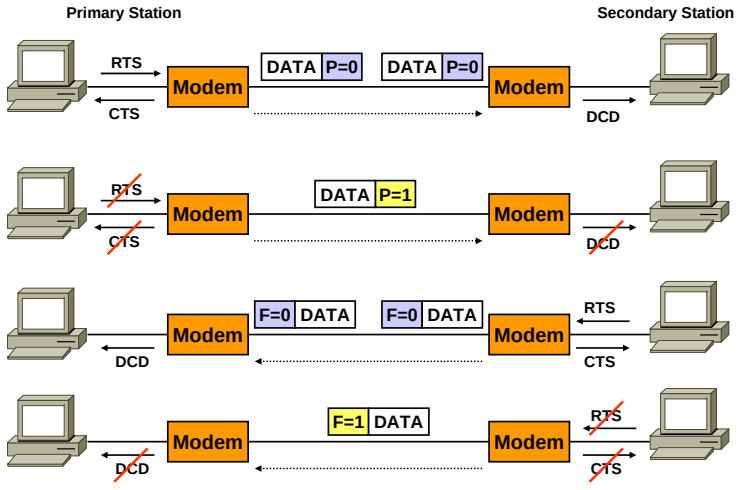
CC = Cluster Concentrator (today also called Establishment Controller)

MSD = Modem Sharing Device

FEP = Front-End Processor

© 2012, D.I. Lindner / D.I. Haas

Half-Duplex Management



© 2012, D.I. Lindner / D.I. Haas

HDLC was created to work in Half-Duplex mode only. The most important thing in the earlier days of HDLC was the P/F bit. This bit was used to hand-over the send-rights.

The server started to send data with P=0 to the host. After the server is finished sending data he sets the P-Bit to 1. Now the host knows that he can send data now, with F=0. When the host sets the F-bit to 1 its time for the server to talk again.

RS-232 Specification:

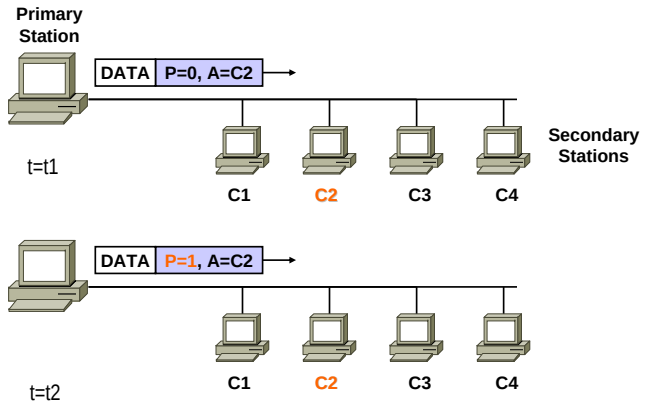
RTS = Request To Send

CTS = Clear To Send

DCD = Data Carrier Detected

© 2012, D.I. Lindner / D.I. Haas

Same on Multipoint Lines (1)

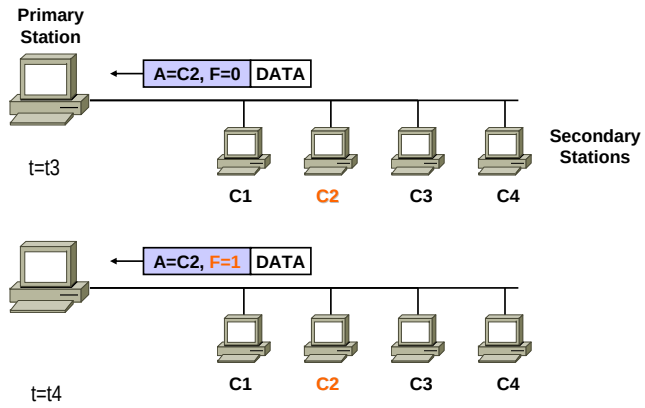


© 2012, D.I. Lindner / D.I. Haas

The same principle also works on multipoint lines. In the picture above you see a primary station (server) and 4 secondary stations (hosts) on a multipoint line. The server sends out his date with P=0 to host 2 (C2). When the server is finished with sending data, and when he wants to receive data from C2 he sets the P-bit to 1.

© 2012, D.I. Lindner / D.I. Haas

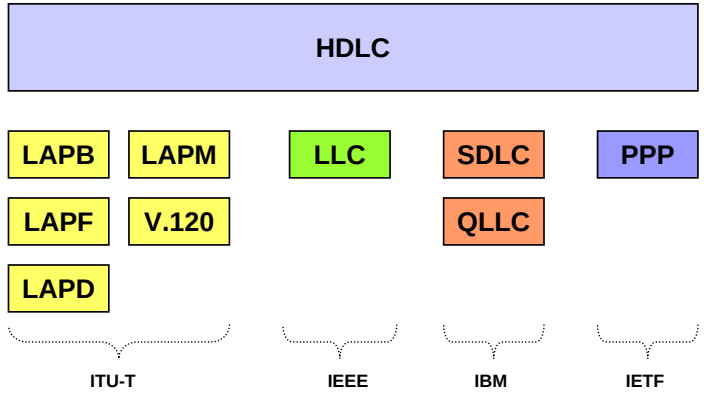
Same on Multipoint Lines (2)



Now C2 is sending data to the primary station, and when he sets F=1, its time for the server to send again.

Only the primary station can talk with all secondary stations. The hosts can only talk to the server.

HDLC Family



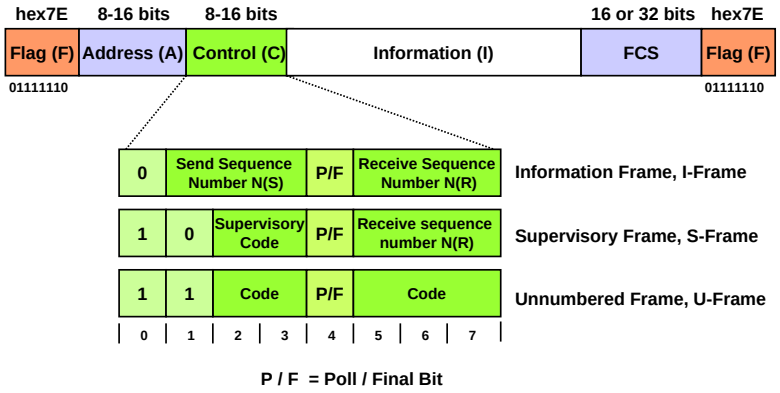
As mentioned, ISO adopted SDLC and standardized it as an extensible set called HDLC. The ITU-T versions are called LAPs (Link Access Procedures). The IEEE variant is called LLC (Logical-link control) also known as IEEE 802.2.

LAPB is the standard link layer for X.25, LAPF is also known as Frame Relay, LAPD is the ISDN link layer for the D channel, LAPM (aka V.42) is the invisible friendly ghost in modems who transports PPP frames from modem to modem, V.120 is used by ISDN on TAs (Terminal Adapters) to multiplex multiple users across a single link, PPP was designed to transport layer-3 datagram's (such as IP) over dial-up lines, LLC is the most famous LAN link-layer (aka IEEE 802.2) and is used by Ethernet, Token Ring, FDDI etc.

QLLC was created to transmit SNA data over an X.25 network. Both X.25 and QLLC replace SDLC in the SNA-stack.

Note: Most HDLC-related standards were standardized by ISO, for example: ISO 7776 LAPB
ISO 8471 LAPB address information
ISO 8802/2 LLC
ITU-T I.441 LAPD

HDLC Frame Format



The framing pattern is 0x7E or 01111110 in binary notation. Typically the address field is 8 or 16 bits and also the control field is 8 or 16 bits.

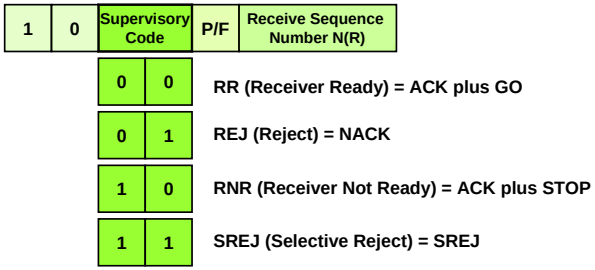
Information frames transport data. **Supervisory frames** are used for ACK and NACK when no data field is appended (and therefore no send sequence number is needed) and flow control (RR and RNR). **Unnumbered frames** are used for connectionless transmissions, connection establishment and exchange-identifier (XID) messages.

Note that every frame type contains the Poll/Final bit, which is used to hand-over the sending permission and to obtain a response from the other station ("check-pointing").

Sequence number space is either 3 bits (0..7) in the standard modes or 7 bits (0..127) in the so-called extended modes.

© 2012, D.I. Lindner / D.I. Haas

Supervisory Frames



The Supervisory Frames are also called "S-Frames" and are used for error recovery and flow control.

© 2012, D.I. Lindner / D.I. Haas

Unnumbered Frames

1	1	Code	P/F	Code

Code	Code	Primary Station: Commands	Secondary Station: Responses
0 0	0 0 0	UI	UI
0 0	0 0 1	SNRM	
0 0	0 1 0	DISC	RD
0 0	1 0 0	UP	
0 0	1 1 0		UA
0 1	0 0 0	NR0	NR0
0 1	0 0 1	NR1	NR1
0 1	0 1 0	NR2	NR2
0 1	0 1 1	NR3	NR3
1 0	0 0 0	SIM	RIM
1 0	0 0 1		FRMR
1 1	0 0 0	SARM	DM
1 1	0 0 1	RSET	
1 1	0 1 0	SARME	
1 1	0 1 1	SNRME	
1 1	1 0 0	SABM	
1 1	1 0 1	XID	XID
1 1	1 1 0	SABME	

© 2012, D.I. Lindner / D.I. Haas Protocol Principles, v5.1 89

The Unnumbered Frames are also called "U-Frames".

Commands are sent by the primary station only. Responses are sent by secondary stations. In later implementation of HDLC like LLC or LAPD,LAPF we will see an additional C/R bit in the protocol header. That is because of backward compatibility to the old days of HDLC.

Legend:

DISC – Disconnect
DM – Disconnect Mode
FRMR – Frame Reject
NR0 – Non-reserved 0
RD – Request Disconnect
RIM – Request Initialization Mode
RSET – Reset
SABM – Set ABM
SABME – Set ABM Extended
SARM – Set ARM
SARME – Set ARM Extended
SIM – Set Initialization Mode
SNRM – Set NRM
SNRME – Set NRM Extended
UI – Unnumbered Information
UA – Unnumbered Acknowledgement
UP – Unnumbered Poll
XID – Exchange Identification

© 2012, D.I. Lindner / D.I. Haas

ARQ Principles in HDLC

- **Default: GoBackN without dedicated NACK frame (!)**
  - Receive-Sequence Number indicates next frame expected
- **"Check-pointing"**
  - Sender triggers (N)ACK information with P/F bit
- **Optional: Reject (REJ)**
  - Dedicated NACK frame
  - Can be sent at any time (no check-pointing)
- **Optional: Selective Reject (SREJ)**
  - Requests retransmission of single frame
- **Flow control with RR and RNR**

© 2012, D.I. Lindner / D.I. Haas Protocol Principles, v5.1 90

HDLC utilizes the GoBackN method of error-recovery because this method maintains the sequence of the packets and there is no need for any reordering. HDLC is a rather old protocol and reordering was a comparatively time-consuming process in these days. Originally. a dedicated NACK frame was not necessary because the sequence number carried in each received frame indicates the number of the next frame expected. If this sequence number is below the actual senders sequence number, than the sender must retransmit all frames starting with the given number.

Each time a P/F=1 event occurs, both peers must check whether they received all data, otherwise error recovery is performed. This method is called "check-pointing" because the events P/F=1 determine specific moments in time where both peers synchronize with each other. The idea of check-pointing is generally used with redundant and failure-tolerant systems.

An optional feature is the usage of so-called "reject" (REJ) frames. These dedicated frames are interpreted as explicit negative acknowledgements (NACKs). So as soon as a receiving peer notices missing data he can issue a REJ frame immediately without waiting for the next checkpoint. Another optional feature is the use of "selective" rejects (SREJ) which are used to request the retransmission of single packets within a stream.

Both REJ and SREJ enhance the performance of HDLC.

© 2012, D.I. Lindner / D.I. Haas

HDLC Data Link Services

- **HDLC can provide connection-oriented service**
  - Setup of connection done by U-frames
    - SNRM, SARM, SABM, UA
  - I-frames and S-frame can be used only after connection setup
    - I, RR, RNR, REJ, SREJ
  - Clearing of a connection done by U-frames
    - DISC, UA
- **HDLC can provide connectionless service**
  - only U-frames can be used
    - UI for data transport

Frame-Types

Connection-Oriented

I	**Information**
RR	**Receiver Ready**
REJ	**Reject**
RNR	**Receiver Not Ready**
SREJ	**Selective Reject**
SNRM	**Set Normal Response Mode**
SABM	**Set Async Balanced Mode**
SARM	**Set Async Response Mode**
SNRME	**Set NRM Extended Mode**
SABME	**Set ABM Extended Mode**
SARME	**Set ARM Extended Mode**
DISC	**Disconnect**
UA	**Unnumbered Acknowledge**
RSET	**Reset**
FRMR	**Frame Reject**
RD	**Request Disconnect**
DM	**Disconnect Mode**

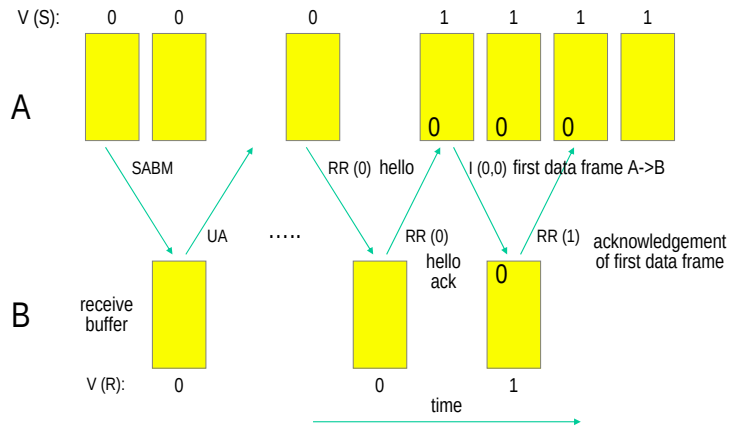
Connection-Less

UI	**Unnumbered Information**

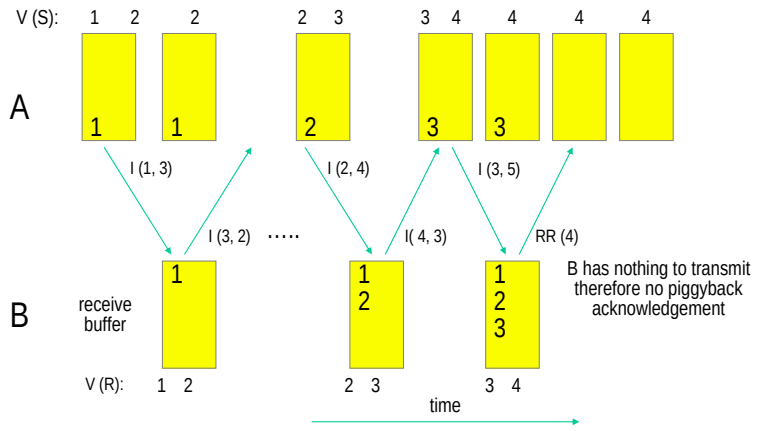
Miscellaneous

XID	**Exchange Identification**
UP	**Unnumbered Poll**
SIM	**Set Initialization Mode**
RIM	**Request Initialization Mode**
NR0-3	**Non-Reserved 0**

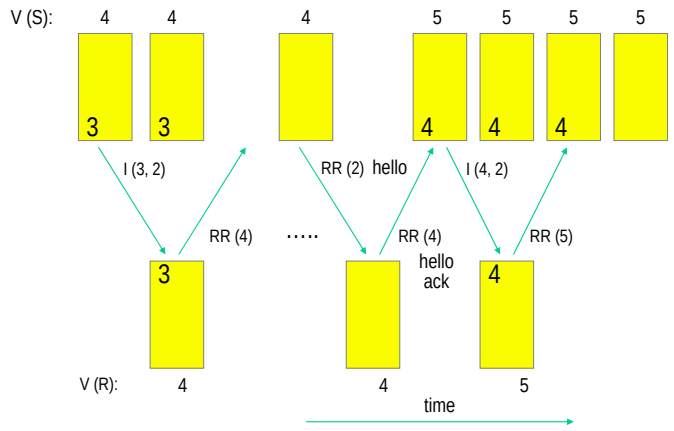
HDLC Example: Initializing / Keepalive / First Data Frame A -> B



HDLC Example: Data Frames A->B , B->A with piggyback ACK



HDLC Example: Keepalive



HDLC Example: Flow Control