LEC. 4

B.1

IU transfer models

- Connectionless (CL)
  - a single time step (no negotiation, independence and self-consistency of IUs)
  - agreement only between N-user and (N-1)-supplier

N entity

(N-1) entity

- Connection-Oriented (CO)
  - ① connection creation
  - three time steps (agreement between the ends of the connection and transfer of the IUs) ② ③ Tear-down connection
  - the transfer of the IUs occurs as through a "pipe"
    - IUs are sent from the source and extracted in an orderly manner from the receiver
    - logical relationship between "segments" the way IUs are typically indicated at Transport layer

CL Transfer

Transport layer

- Also called a datagram (this is how the IUs are called)
- The transfer of IUs occurs without ascertaining the availability of the recipient and/or network resources
- There are no establishment and tear-down phases of a call
- Each IU is handled by the network independently of the others, even if they are part of the same communication
- If switching nodes are present, they operate the routing function only on the basis of individual IUs

routers

- It is possible for packets to be delivered out of sequence

CL Transfer: an Example

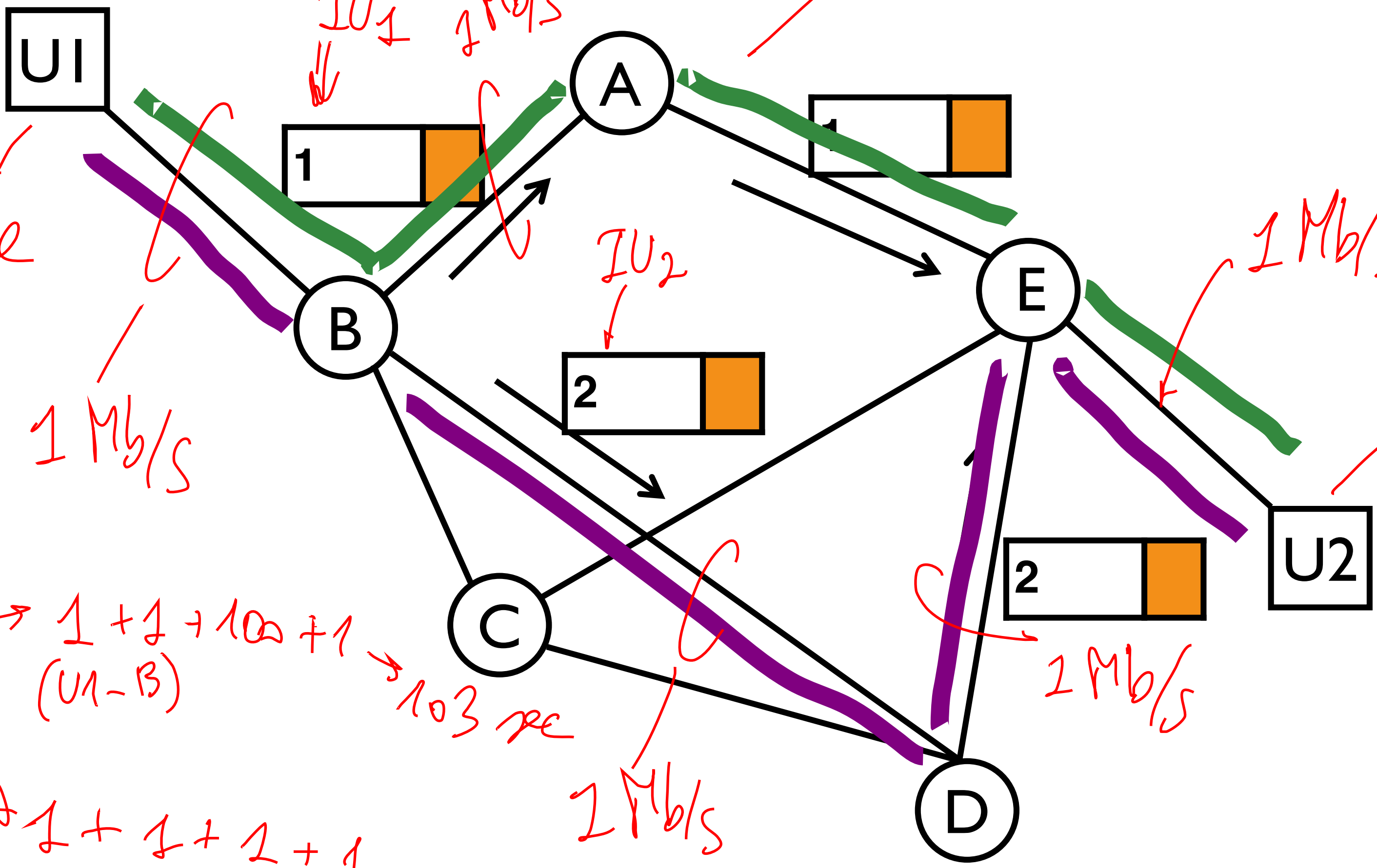
A-E → 10 Kb/s

IU1 1 Mb/s

Source

IU2

Destination



1 Mb/s

1 Mb/s

1 Mb/s

1 Mb/s

IU1 → 1 Mb → 1 + 1 + 100 + 1 (U1-B) → 103 sec

IU2 → 1 Mb → 1 + 1 + 1 + 1 → 4 sec

CO Transfer (1)

1. Connection establishment phase
   - call acceptance check and possible logical allocation of necessary resources → Presence of destination
   - assignment of appropriate ~~call~~ identifiers used by all IUs belonging to the same connection – Connection identifier of a transfer
   - if crossing multiple nodes, one determines the path which will be followed by the packets

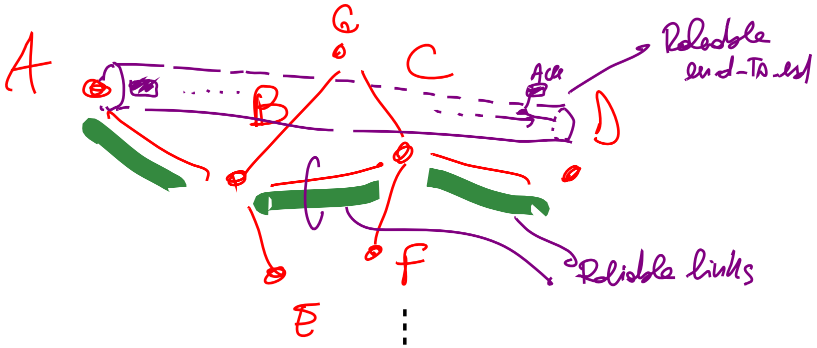
This is NOT what happens in Internet

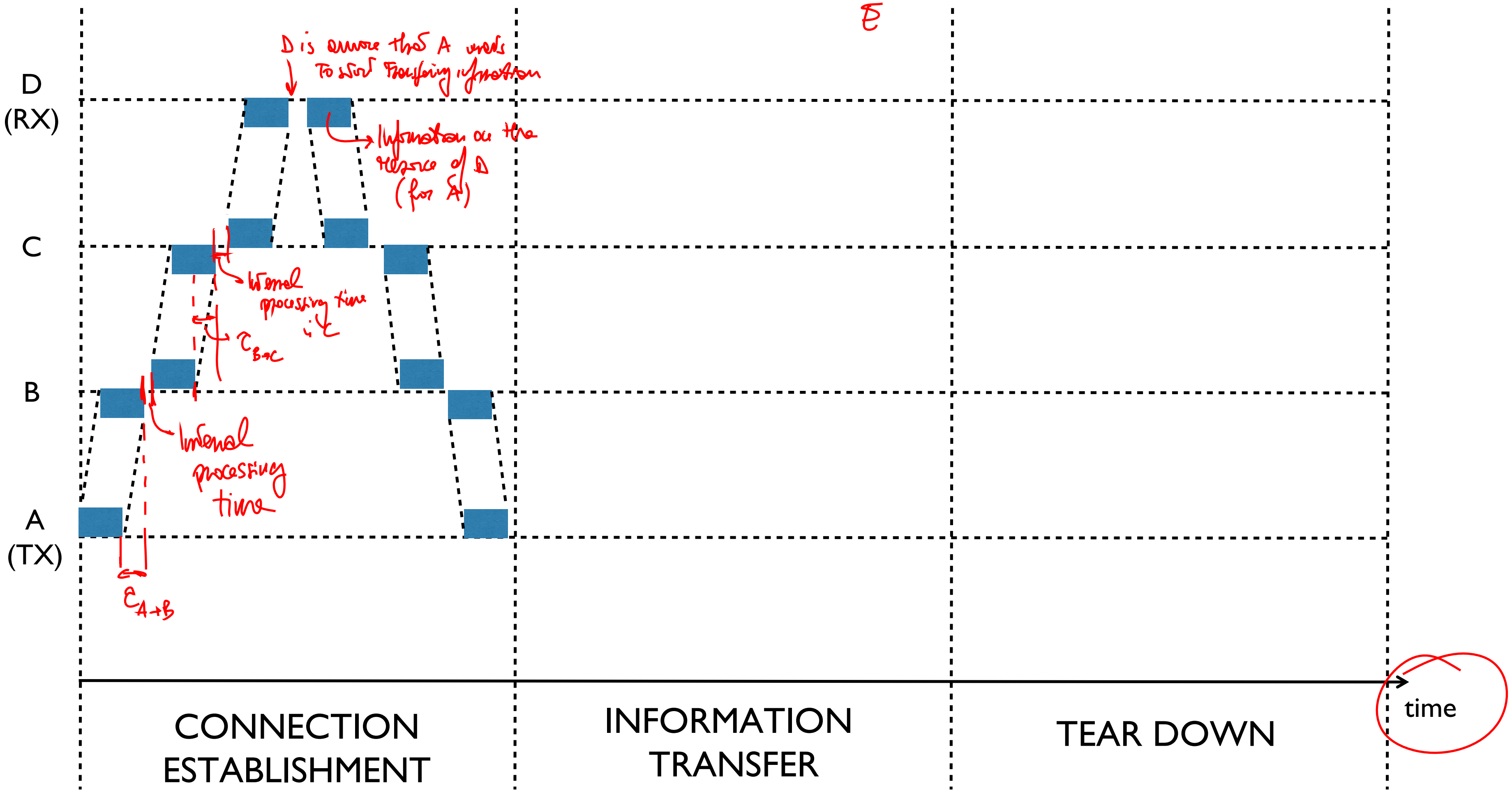
In this way you automatically guarantee that all packets are received in order

CO Transfer (2)

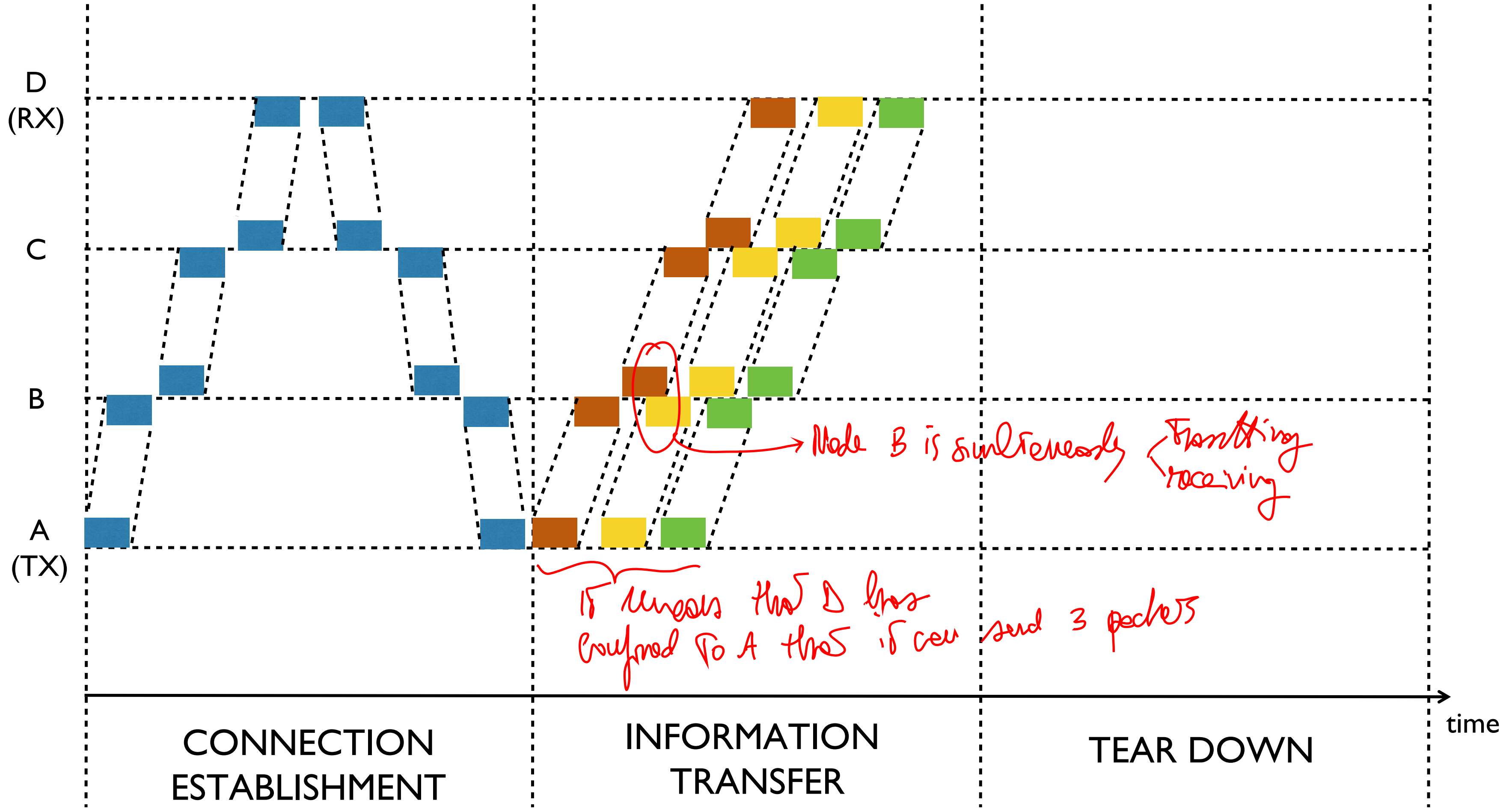
2. Information Transfer phase
   - IUs are processed by terminals and, possibly, network nodes on the basis of whether those IUs belong to a specific connection (previously established)
   - logical identifiers associated with individual IUs are considered for this purpose
   - such identifiers are sometimes called logical channel number or virtual circuit identifier (VCI) → Used for routing and related to the information flow identifiers mentioned before
3. Connection tear-down phase
   - previously allocated resources are released

Time Diagram (1)





Time Diagram (2)

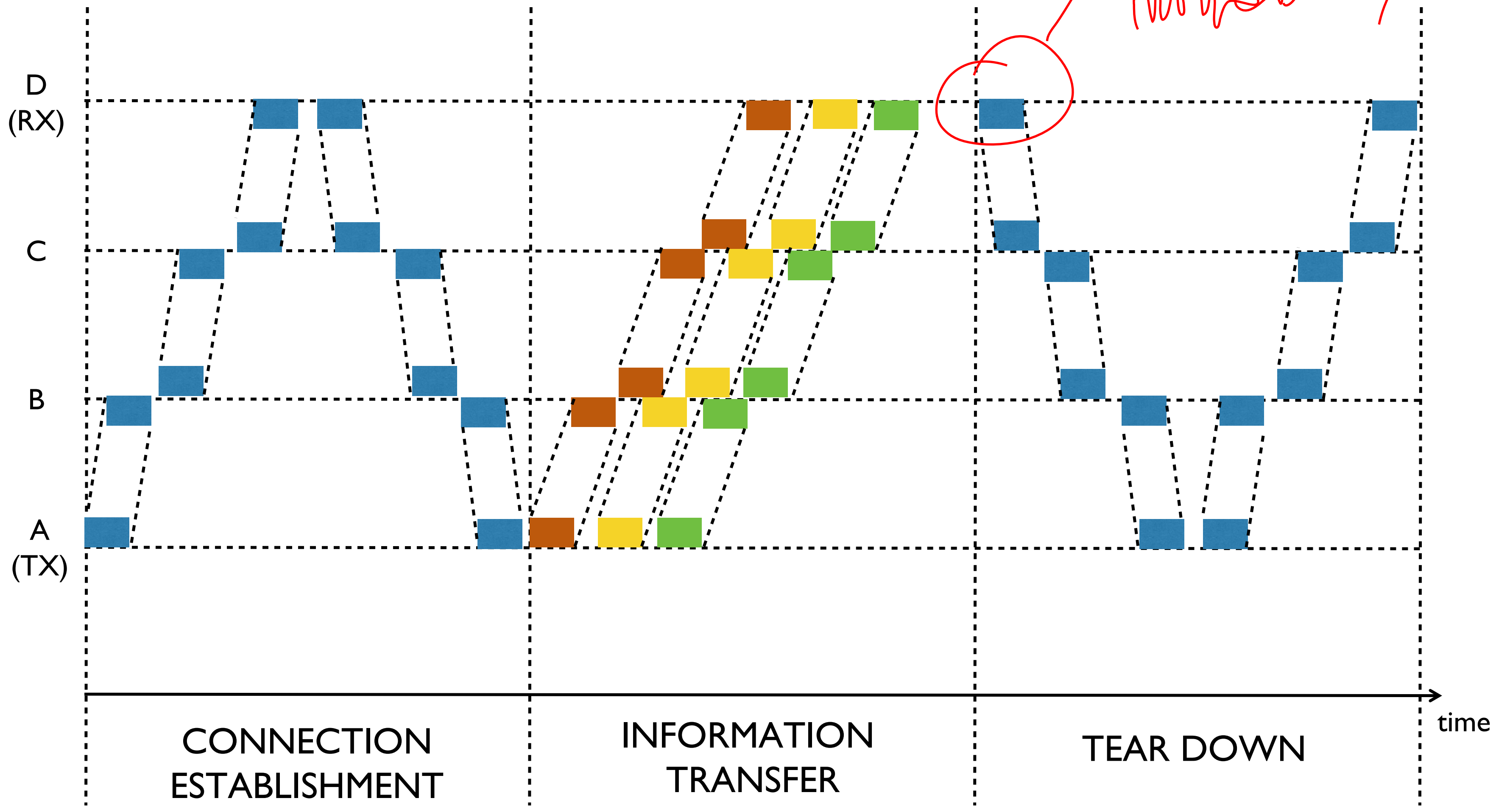


Node B is simultaneously transmitting / receiving

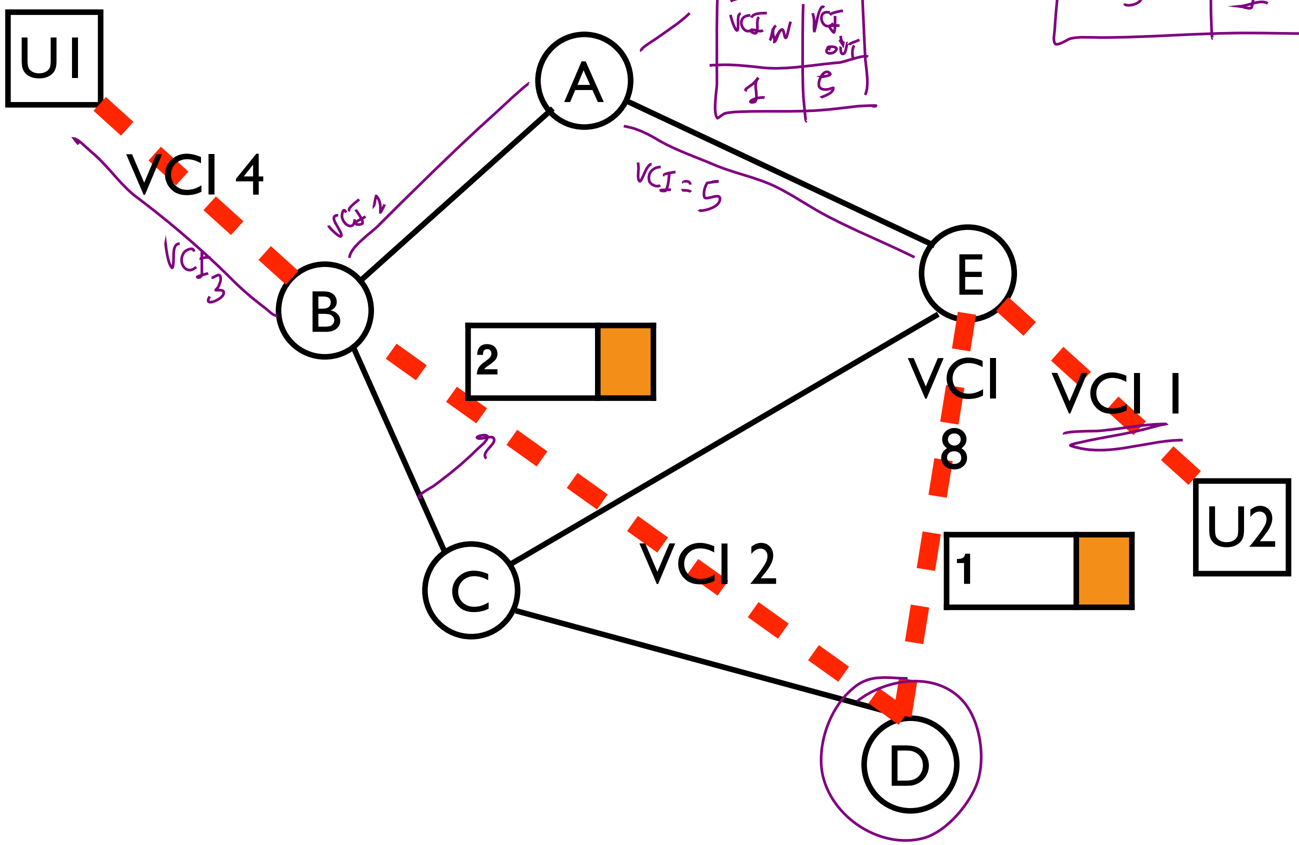
It means that D has confirmed to A that it can send 3 packets

Time Diagram (3)

In this example Tearing down is initiated by D



CO Transfer: an Example



NODE A	
VCI IN	VCI OUT
1	5

NODE B	
VCI IN	VCI OUT
4	2
3	1

NODE B	
VCI in	VCI out
4	2

"Arbitrary"

NODE D	
VCI in	VCI out
2	8

NODE E	
VCI in	VCI out
8	1

B.2

QoS-based Models (1)

- A communication is said to be reliable if it ensures the information integrity of the data transferred
  - individual IUs
  - sorting of the IUs
- Protocol characterization
  - reliable service = reliable protocol (TCP, HTTP, SMTP)
  - untrusted service = unreliable protocol (Ethernet, IP, UDP)

Transport layer

Application layer

Network access layer | Link & Phy layer

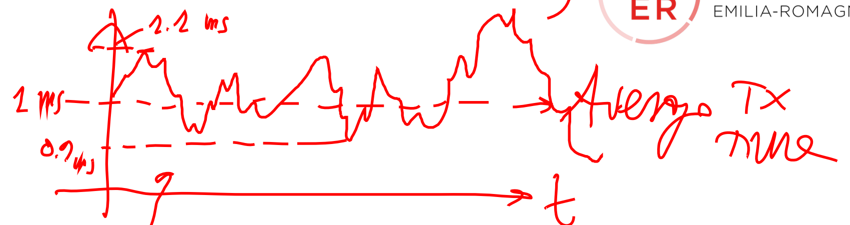
Network layer

Transport layer

Jitter → Transmission time (device-dependent)

QoS-based Models (2)

(latency)



- A communication is said to be **real-time** if it ensures temporal transparency of transferred data
- Maintenance/reconstruction in reception of the same time cadence of the sent IUs
  - Equalization in case of low jitter (variability) of end-to-end delays
- Low end-to-end delay (average)
  - requires specific transfer characteristics in the network
  - there are protocols that are more or less suitable for real-time communications → UDP

B.3

Message- vs stream-oriented

- Message-oriented
  - specific blocks of data are sent and received
  - on the receiving end, the data is guaranteed to belong to a specific block (message)
- Stream-oriented (flow-oriented)
  - data in the upper layer is handled as a continuous stream (not necessarily temporally) of bits or bytes
  - at reception side, the same sequence of data is guaranteed and delivered back to the upper layer in order
  - data are delivered to the upper layer as a single stream
  - requires the establishment of CO-type communication (the reverse is not true)

→ not necessarily reliable

Information-Centric Networking (ICN)

- Traditional architectures offer a service focused on communication between end nodes
- ICN: new communication paradigm that puts the information (data) to be exchanged at the center
    - flipping the functions offered by a network and redesigning the protocols
    - *content* (the data) is addressed, not the terminals
    - communication takes place according to a *request/response paradigm* (interest/data or publish/subscribe)
- Advantages
    - network nodes can become sources for data (caching) Reminiscent of P2P networking
    - security is tied directly to the data and not to the nodes that handle it
- Also referred to as content-centric networking (CCN)

Outline

- The OSI and Internet models
- Communication models
- Delimitation
- Sequence control
- Error management

Delimitation Function

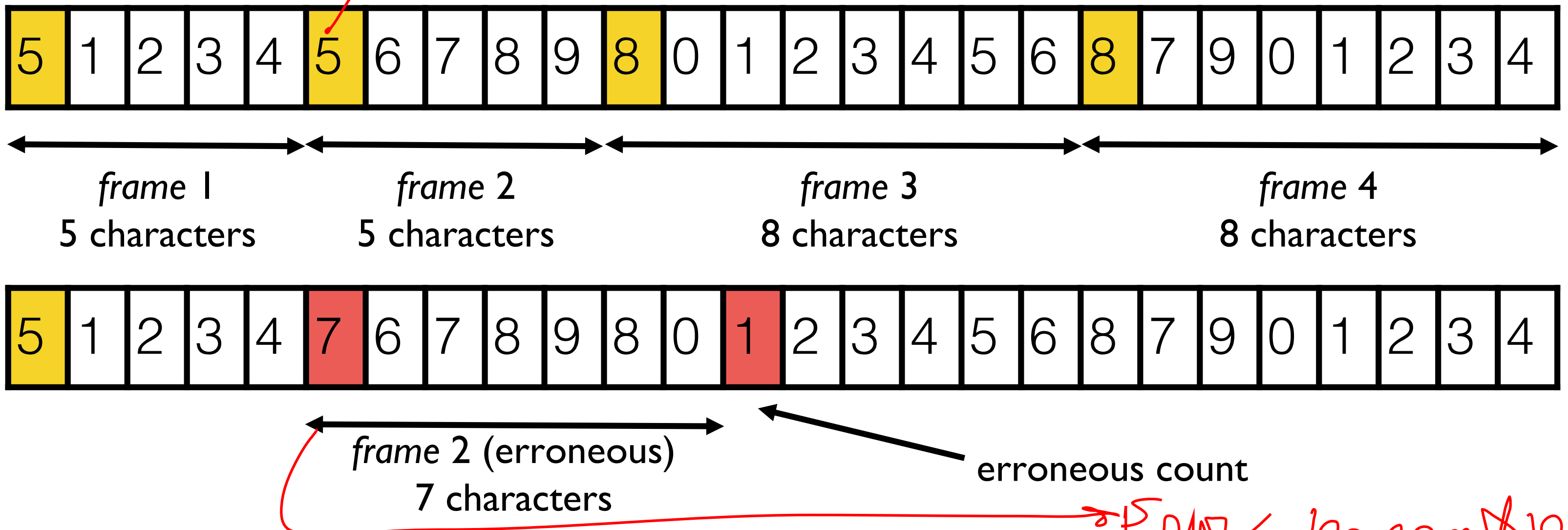
- Delimitation of IUs within a bit/byte stream/block to recognize the start/end of IUs
- Necessary if the underlying layer is stream-oriented (e.g., physical layer protocols)
- Implemented modes
  - bit/byte counting
  - insertion of start/end delimiters
  - combination of them (start delimiter + character count to distinguish end)

Character count

- Use of IU length indicator fields
- Use of fixed-length IUs
- Simple and low overhead
- Difficult synchronization management in case of errors
- Example: length of the IU as the first byte of the block

1

0000 01~~0~~1



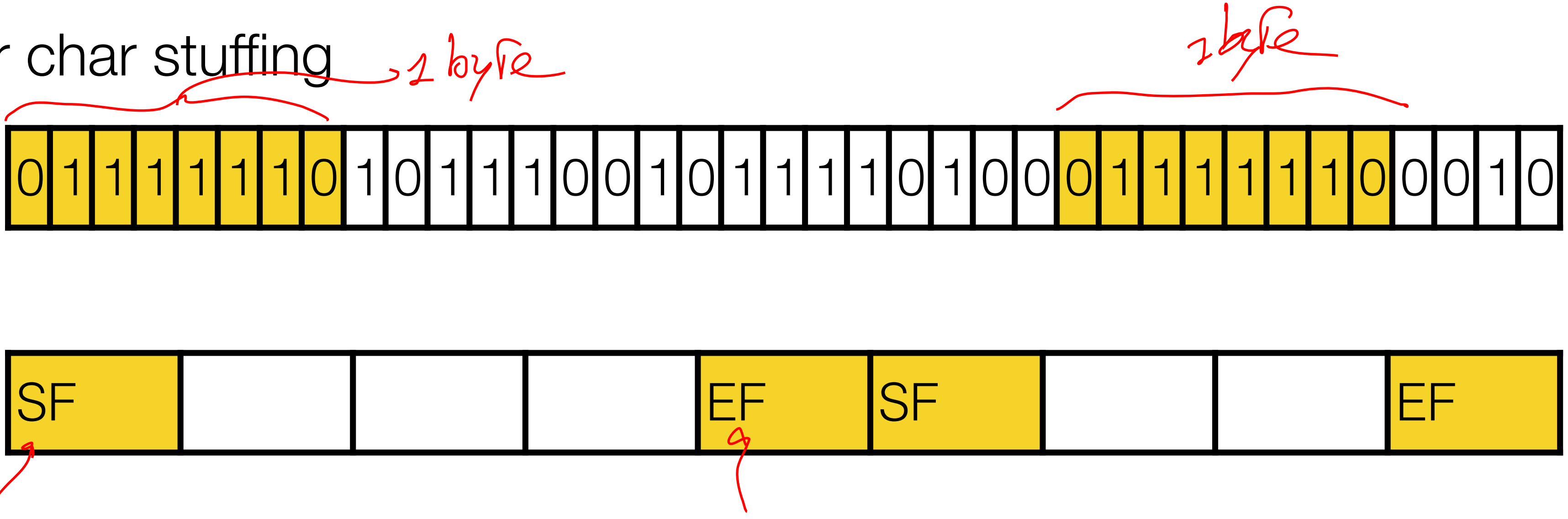
Error propagation

Insertion of delimiters

- Insertion of special bit/byte sequences
- Flags or control characters
- It is necessary that delimiters do not appear within plots
  - delimiters are characters that are not allowed
  - bit stuffing or char stuffing

delimination sequence *01111110*

start (*SF*) and end (*EF*) *flags*



Bit/Char Stuffing (1)

- Aim: To mask the possible presence of delimiter symbols within the IUs
- Sequential operation on component bit/byte blocks of IUs
- Algorithm of HDLC/LAPB/X.25
  - SF/EF equal to 01111110 → 0, 6 "1"s, 0
  - stuffing algorithm: after five consecutive "1's" add a 0
  - de-stuffing algorithm: after five consecutive "1's" remove the following 0

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

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

0 1 1 1 1 1 1 0 1 1 0 1 1 1 1 ~~1~~ 0 1 1 1 1 1 0 0 1 1 1 1 0 1 0 1 1 1 1 1 1 0

0

Bit error → consecutive bits are wrong

Bit/Char Stuffing (2)

Byte → binary → 8 bits
Byte → hex → 2 hex → 4 bits
Byte → decimal
0, ..., 9, A, ..., F
$0, \ldots, 2^8 - 1 = 255$

- SLIP algorithm
  - SF/EF equal to END (192 in decimal)
  - stuffing algorithm
    - if a byte of data is equal to the END character it is replaced by the two-character sequence ESC (219 in decimal) and 220
    - if the ESC character is present this is replaced by the two-character sequence ESC and 221

11	12	9	219	7	192	220	14

11	12	9	219	221	7	219	220	220	14

192	11	12	9	219	221	7	219	220	220	14	192

Advantages/disadvantages

- Advantages
  - increased robustness in the presence of errors
  - automatic synchronization to the receiver
- Disadvantages
  - need for redundancy if reserved symbols are used
  - greater overhead in the transmitted stream if bit or char stuffing is used

Outline

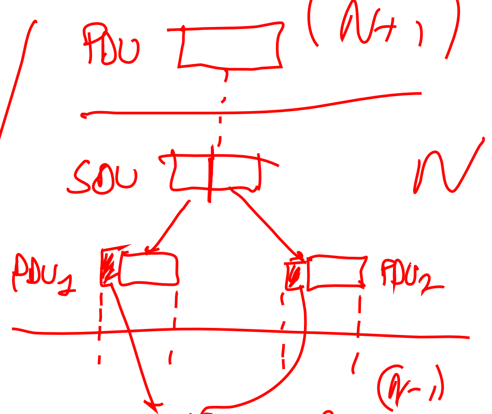
- The OSI and Internet models
- Communication models
- Delimitation
- Sequence control
- Error management

IPv4 → allows fragmentation at source node and relay nodes
IPv6 → allows fragmentation only at source node.

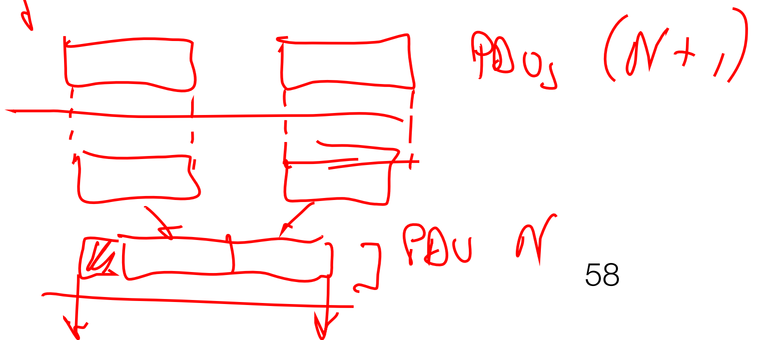
From p. 16

Fragmentation and aggregation

- Fragmentation: a function that allows a single SDU to be encapsulated by two or more PDUs
  - opposite reassembly at the receiver
  - can also be performed at intermediate nodes
- Aggregation: function that allows two or more SDUs to be encapsulated via a single PDU
  - opposite separation at the receiver
- Such functions are typically required due to physical constraints of the communication channel
- Discussion in detail: IP protocol (?)



In the control header → information on how to reassemble.



Risequentialization

- Necessary in the presence of fragmentation/aggregation (among several)
- Recovery at the receiver of the correct sequence of the sent IUs
- At the receiver, the IUs can be delivered in the correct order to the upper layer
- Achievable through use of sequence numbers
  - increasing order and modulo $N=2^k$ (using k bits)
  - IUs can be counted or bytes directly by entering the number of the first byte (e.g., in TCP)
- IUs out of sequence can be stored for reorder or discarded (error recovery is needed)

Outline

- The OSI and Internet models
- Communication models
- Delimitation
- Sequence control
- Error management

Error Control

- Detect errors incurred by the IUs during their transfer and, if necessary, restore the correct information flow
  - transmission or procedural errors
  - duplication
  - alteration of order
  - loss of IUs

very rare

channel-related problems (possible)

1. Error detection
   - Detect on the receiving end any errors (usually transmissive) in the received IUs
2. Error correction
   - Correct any erroneous bits or bytes in the IUs

   forward error correction (FEC)
3. Error recovery
   - Return to normal the flow of transferred IUs between two entities in the case of duplication, loss, or alteration of their order

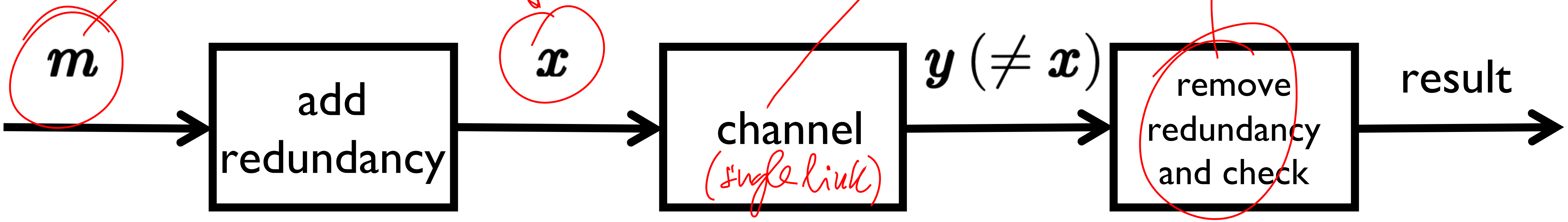
   Ask the transmitter to retransmit the IU x is an "extended" version of m

problems!

y → x → m

?

Error Detection



- It is normally based on the addition of redundancy in transmission
  - used in reception to detect (but not correct) errors
  - redundancy required for detection is much smaller than would be required for correction ~~(16-32 bits)~~
- May be the basis for possible correction/recovery
- Different mechanisms for generating error detection code
  - Parity check (blockwise), 1's complement sum (checksum), etc.
- Similar principles are used in security
  - an error detection code should detect only random changes

Parity Check

- A bit equal to 1 is added for each block of bits if the number of 1's in the block is odd, otherwise a 0 is added (even parity)

  # of 1's is even

  - the number of parity bits generated is equal to the number of blocks
  - these bits can be individually added successively to each block or all together at a specific point in the IU (e.g., at the end)
- The parity bit allows errors to be recognized in odd numbers

Example (1)

- Consider 8-bit blocks

$$\boldsymbol{m} = \boxed{10010010}\boxed{10100011}$$

$$\boldsymbol{m}_1 = 10010010 \rightarrow 3\ 1\text{'s}$$

ODD

$$\boldsymbol{m}_2 = 10100011 \rightarrow 4\ 1\text{'s}$$

EVEN

$$\boldsymbol{x}_1 = \boxed{10010010\color{red}{1}}$$

$$\boldsymbol{x}_2 = \boxed{10100011\color{red}{0}}$$

$$\boldsymbol{x} = \boxed{10010010\color{red}{1}}\boxed{10100011\color{red}{0}}$$

$$(\boldsymbol{x} = 1001001010100011\color{red}{10})$$

Example (2)

Problem!

1

- Consider 1 error in the first block

$$\boldsymbol{y} = 1000001010100011\boxed{10}$$

↳ 2 1's

- Assuming no errors in parity, we can say with certainty that there is an error in the first block
- Without more information, we cannot correct

discard

discard & ask for retransmission

- any sequence with an odd number of 1's can generate that parity bit
- What can we say about the following sequence?

0 1

$$\boldsymbol{y}_1 = 101000101010001110$$

↳ we do not detect any error!

LEC. 4