UNIVERSITÀ
DI PARMA

MOTORVEHICLE
UNIVERSITY OF
EMILIA-ROMAGNA

Vehicular Communications

Prof. Gianluigi Ferrari, Prof. Luca Davoli

Internet of Things (IoT) Lab

Department of Engineering and Architecture

University of Parma

https://iotlab.unipr.it

UNIVERSITÀ DI PARMA

MOTORVEHICLE UNIVERSITY OF EMILIA-ROMAGNA

Lecture 2

Telecommunication networks basics

Outline

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

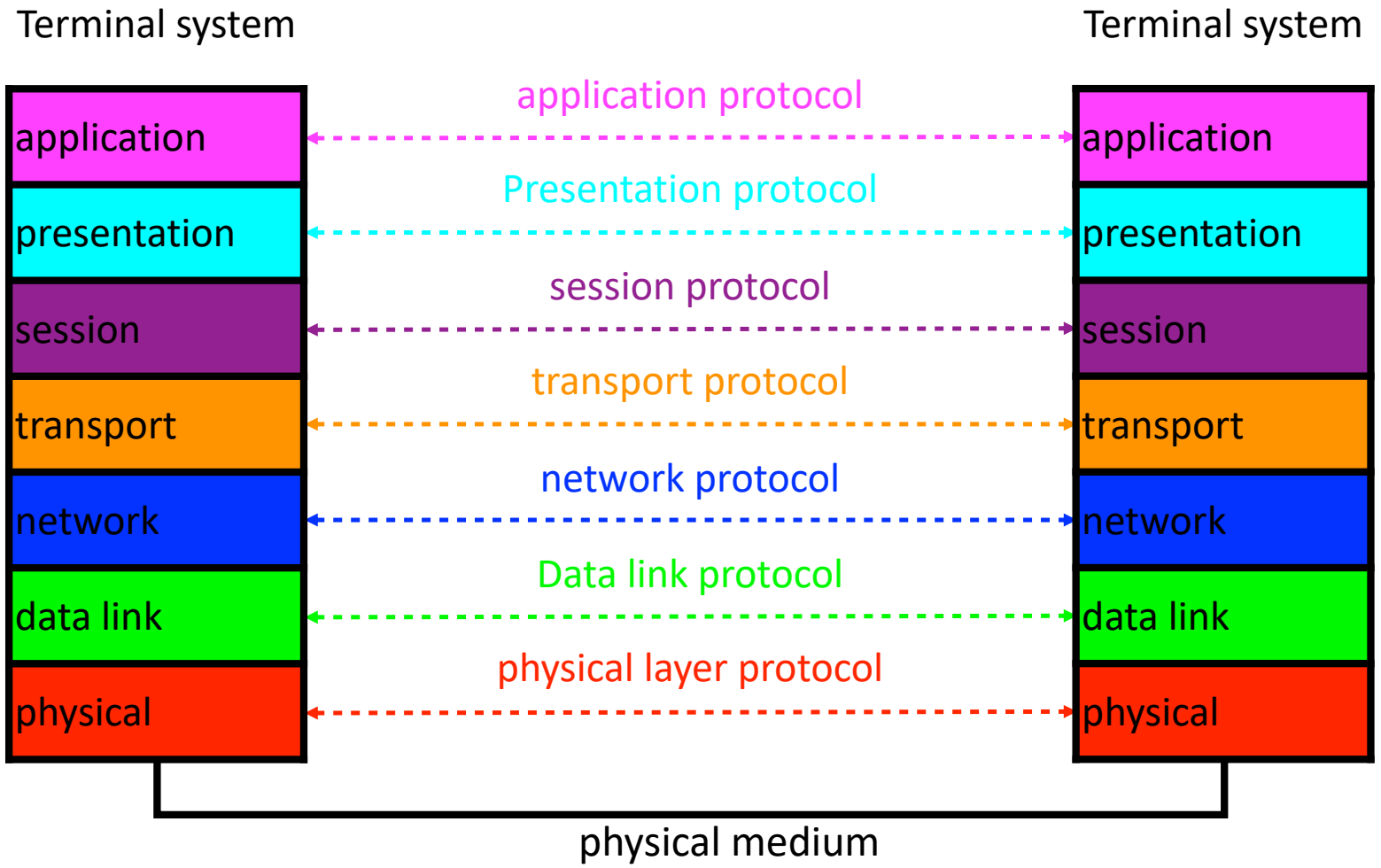
Outline

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

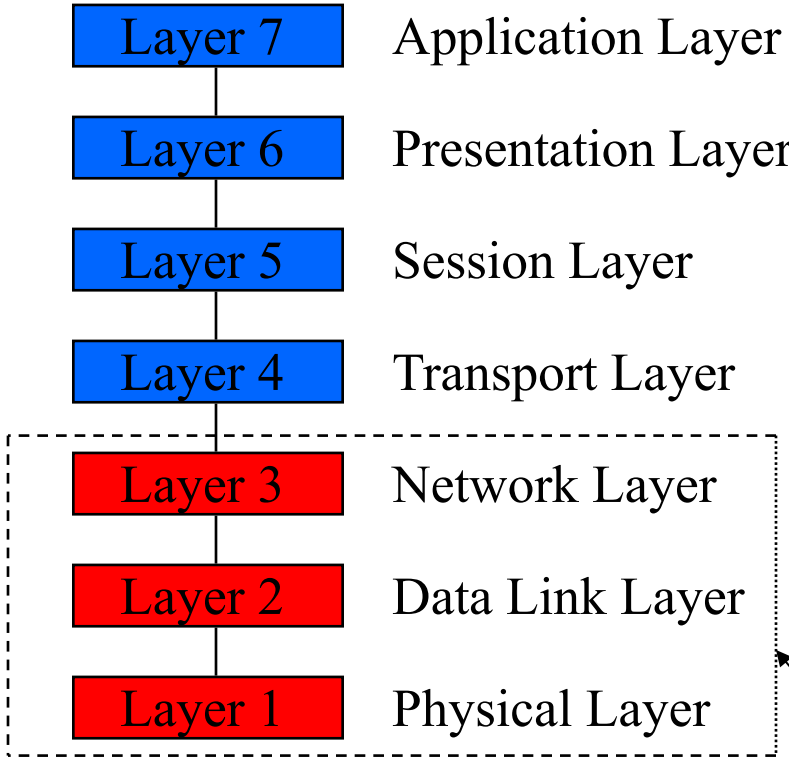
OSI Reference Model

- OSI Reference Model - internationally standardised network architecture
- OSI = *Open Systems Interconnection*: deals with *open systems*, i.e. systems open for communications with other systems
- Specified in ISO 7498
- Model has 7 layers

OSI Architectural model



7-Layer OSI Model



- Layers 1-4 relate to communications technology
- Layers 5-7 relate to user applications

Layer 7: Application Layer

- Level at which applications access network services
  - Represents services that directly support software applications for file transfers, database access, and electronic mail etc.

Layer 6: Presentation Layer

- Related to representation of transmitted data
  - Translates different data representations from the Application layer into uniform standard format
- Providing services for secure efficient data transmission
  - e.g., data encryption, and data compression

Layer 5: Session Layer

- Allows two applications on different computers to establish, use, and end a session
  - e.g., file transfer, remote login
- Establishes dialog control
  - Regulates which side transmits, plus when and how long it transmits
- Performs token management and synchronization

Layer 4: Transport Layer

- Manages transmission packets
  - Repackages long messages when necessary into small packets for transmission
  - Reassembles packets in correct order to get the original message
- Handles error recognition and recovery.
  - Transport layer at receiving acknowledges packet delivery
  - Resends missing packets

Layer 3: Network Layer

- Manages addressing/routing of data within the subnet
  - Addresses messages and translates logical addresses and names into physical addresses
  - Determines the route from the source to the destination computer
  - Manages traffic problems, such as switching, routing, and controlling the congestion of data packets
- Routing can be:
  - based on static tables
  - determined at start of each session
  - individually determined for each packet, reflecting the current network load

Layer 2: Data Link Layer

- Packages raw bits from the Physical layer into frames (logical, structured packets for data)
- Provides reliable transmission of frames
  - It waits for an acknowledgment from the receiving computer
  - Retransmits frames for which acknowledgement not received

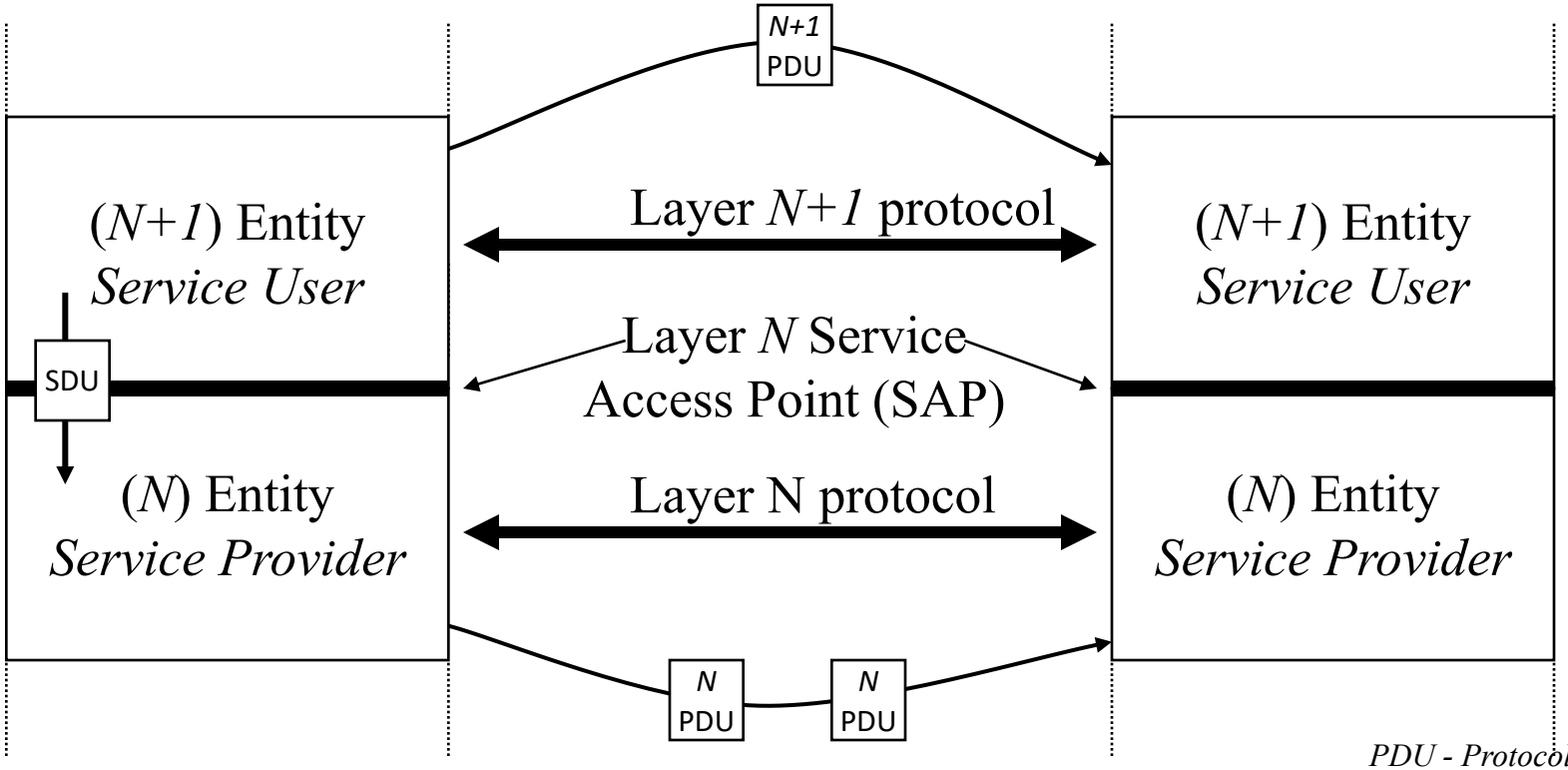
Layer 1: Physical Layer

- Transmits bits from one computer to another
- Regulates the transmission of a stream of bits over a physical medium
- Defines how the cable is attached to the network adapter and what transmission technique is used to send data over the cable. Deals with issues like
  - The definition of 0 and 1, e.g. how many volts represents a 1, and how long a bit lasts?
  - Whether the channel is simplex or duplex?
  - How many pins a connector has, and what the function of each pin is?

Services in the OSI Model

- In OSI model, each layer provide services to layer above, and ‘consumes’ services provided by layer below
- Active elements in a layer called *entities*
- Entities in same layer in different machines called *peer entities*

Layering Principles



PDU - Protocol Data Unit
SDU - Service Data Unit

- Layer *N* provides service to layer *N+1*

Connections

- Layers can offer *connection-oriented* or *connectionless* services
- Connection-oriented, like telephone system
- Connectionless, like postal system
- Each service has an associated *Quality-of-service* (e.g., reliable or unreliable)

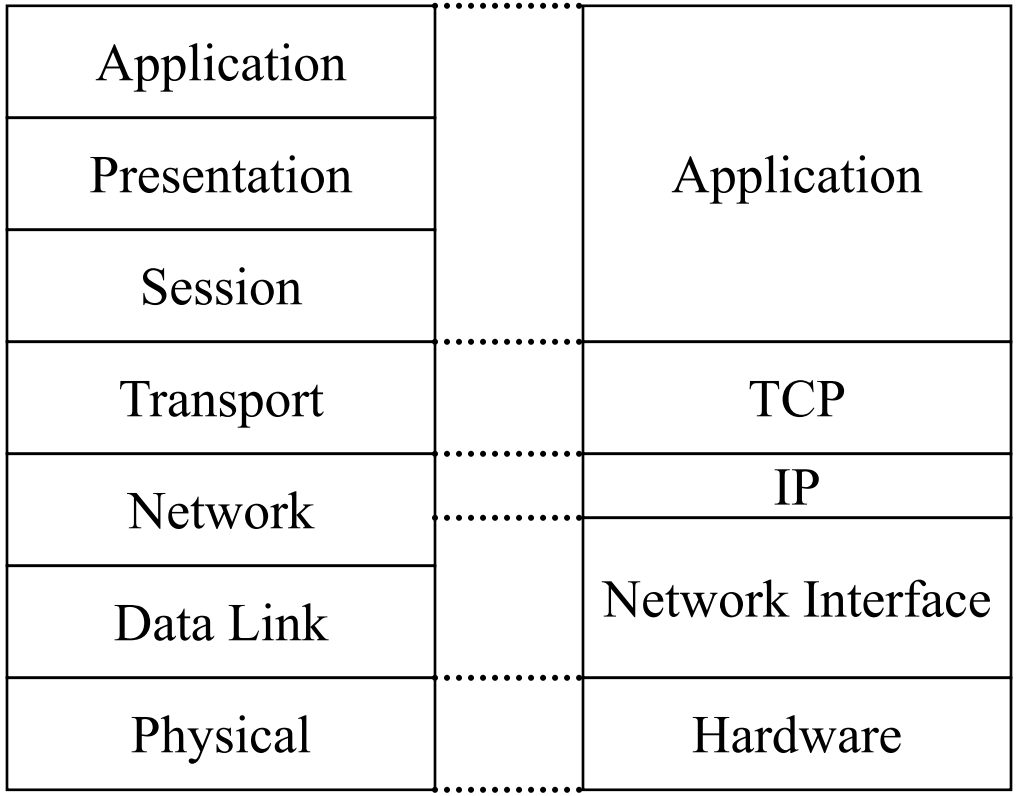
Reliability

- Reliable services never lose/corrupt data
- Reliable service costs more
- Typical application for reliable service is file transfer
- Typical application not needing reliable service is voice traffic
- Not all applications need connections

Topics

- *Service* = set of primitives provided by one layer to layer above
- Service defines what layer can do (but not how it does it)
- *Protocol* = set of rules governing data communication between peer entities, i.e. format and meaning of frames/packets
- Service/protocol decoupling very important

Internet Protocols vs OSI



- Explicit Presentation and session layers missing in Internet protocols
- Data Link and Network layers redesigned

Functionalities

- Network Access Layer
  - includes the functions that in the OSI model are included in the physical and link layers and the low network layer
  - the service offered at the upper layer can be connection-based or connectionless
- Internet layer
  - enables the interconnection of the various component subnets with functionality that in the OSI model is located in the network layer
  - provides a connectionless layer service
  - uses the Internet Protocol (IP)

Functionalities

- Transport layer
  - corresponds to the transport layer and part of the session layer in the OSI model
  - Two types of service: reliable with connection or simpler without connection
- Application layer
  - corresponds to part of the session layer and the presentation and application layers
  - encapsulates all application-type protocols

Outline

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

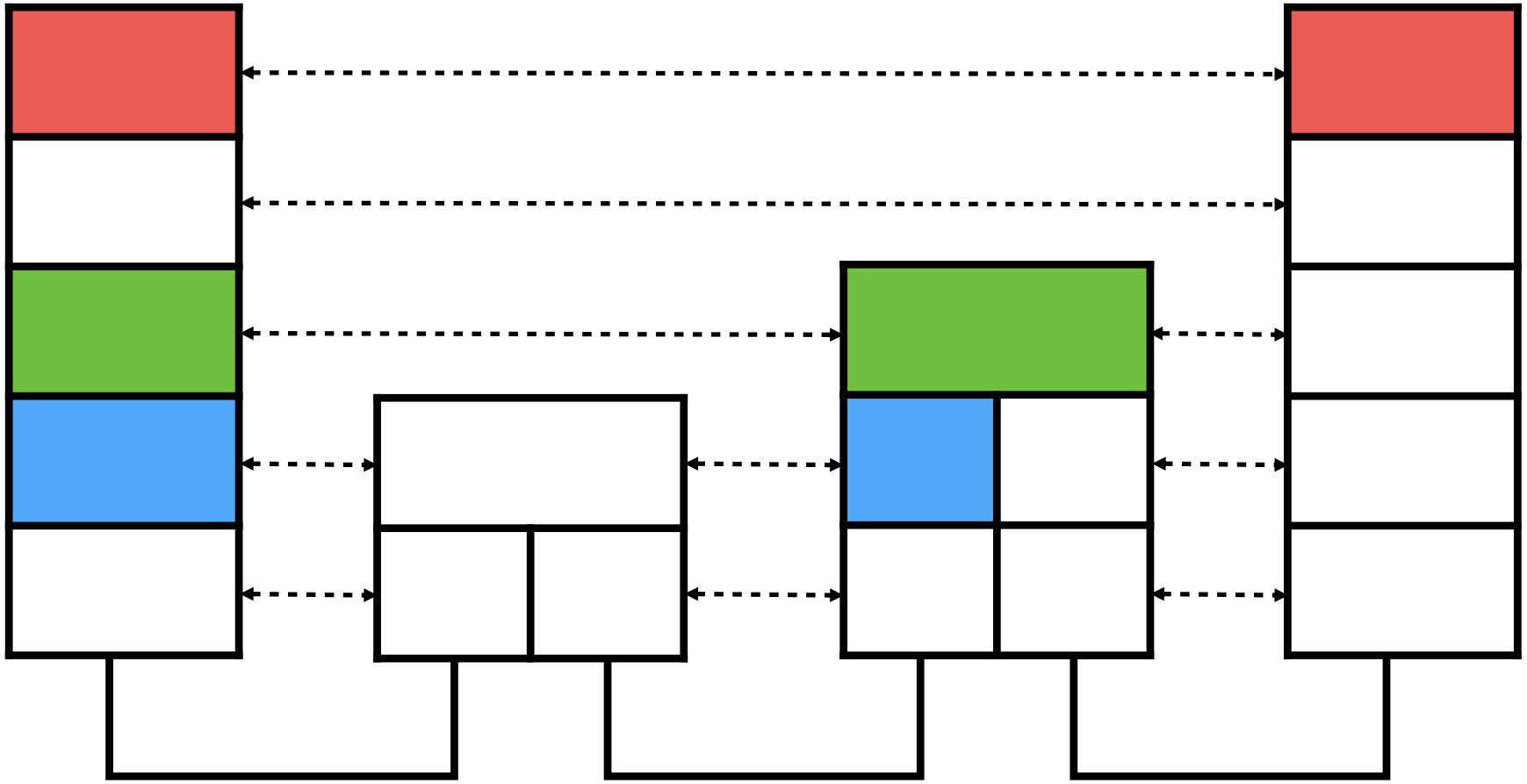
Communiction models

- Based on the entities involved in the communication
  - end-to-end and relayed (location)
  - unicast, multicast and broadcast (number)
  - client-server and peer-to-peer (role)
- Based on the mode of Information Unit (IU) transfer
  - with or without connection
  - reliable/real-time (QoS)
  - message-oriented and stream-oriented

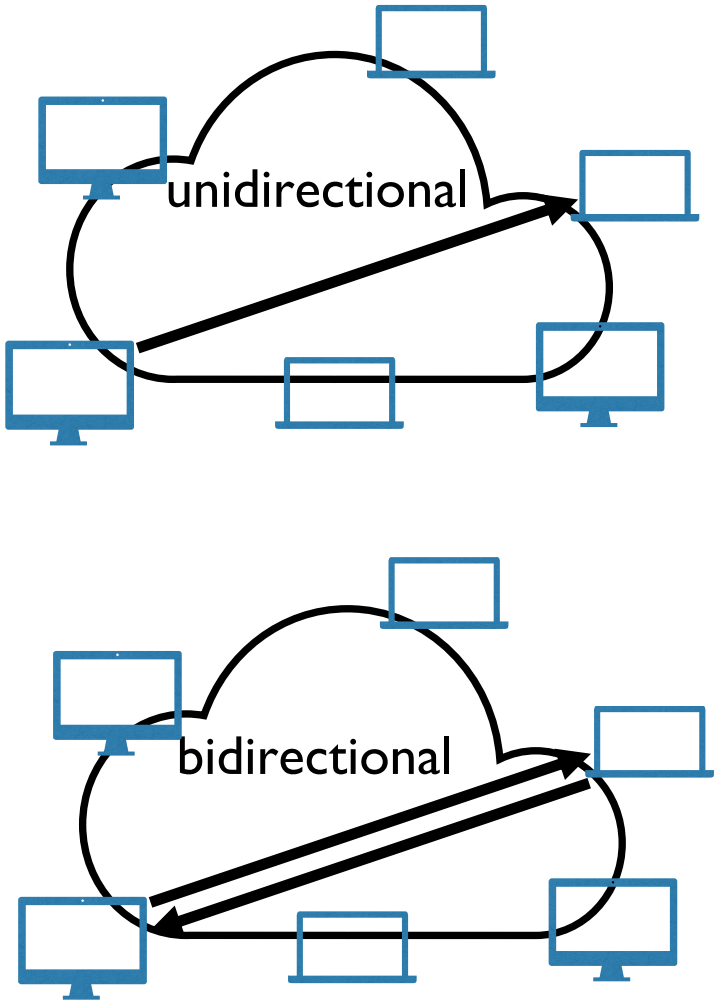
End-to-end vs relayed (1)

- End-to-end communication
  - directly between source and destination entities
  - does not require node addressing but only user addressing
- Relayed/switched communication (relayed)
  - occurs through the relaying of one or more intermediate nodes
- By extension a communication protocol may be end-to-end or relayed

End-to-end vs relayed (2)

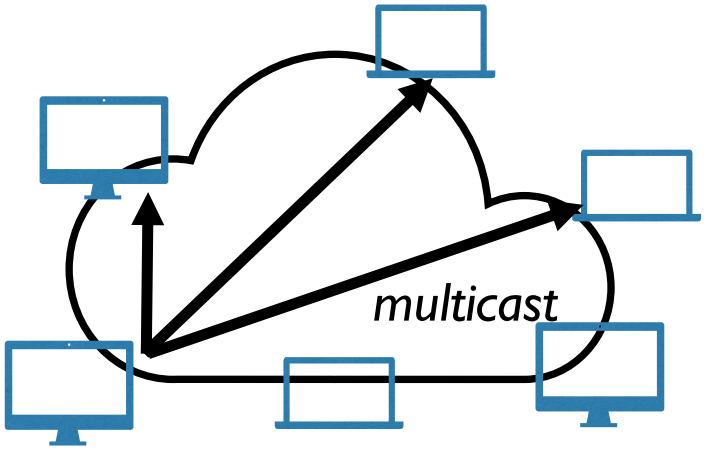


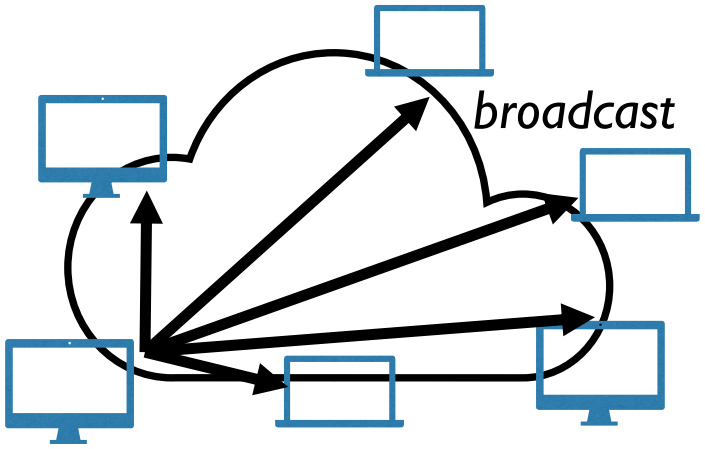
Unicast Communications



- 1 source and 1 destination
- Unidirectional or bidirectional
- Example: Classical telephony (fixed and mobile)

Multicast communications



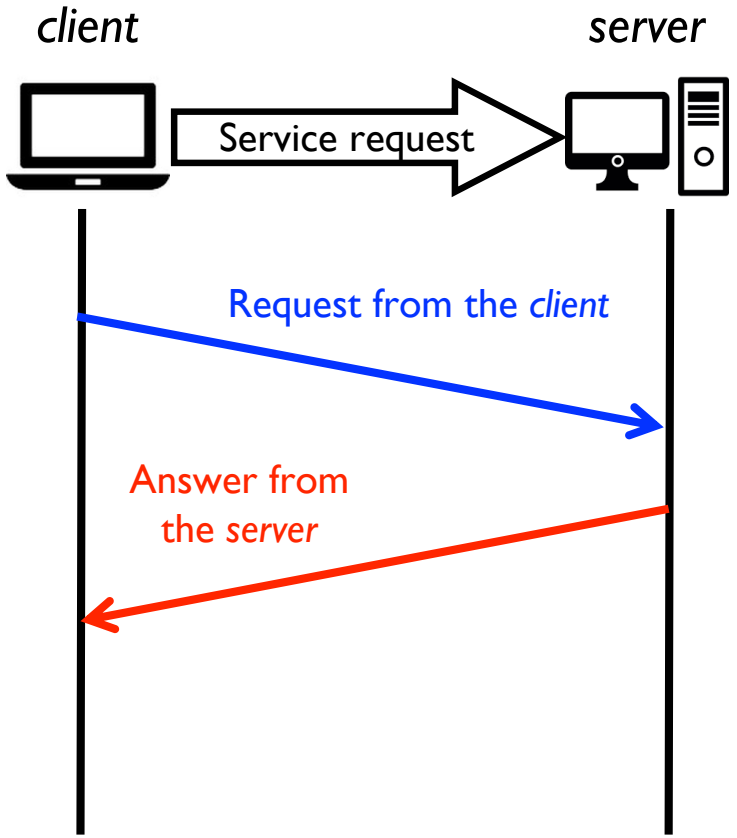


- Multicast: 1 source and multiple destinations
- Advanced telephone services (three-way calling)
- Broadcast: 1 source and all possible destinations
- TV, local networks

Protocols' Operations

- Some protocols support
  - only unicast mode
  - both unicast and broadcast
  - all three modes of communication (example: IP)
- Broadcast and multicast can be realized
  - taking advantage of any features of the underlying protocol
  - using multiple unicast communications at the underlying layer

Client-server model



- Direct communication between 2 entities with distinct roles
  - client: initiates communication and/or requests a service
  - server: opposite role
- The client forwards (service) requests to a server
- The server processes the requests and, if the request is accepted, responds to the client by providing the requested service
- Generally, the communication results in asymmetric type of communication

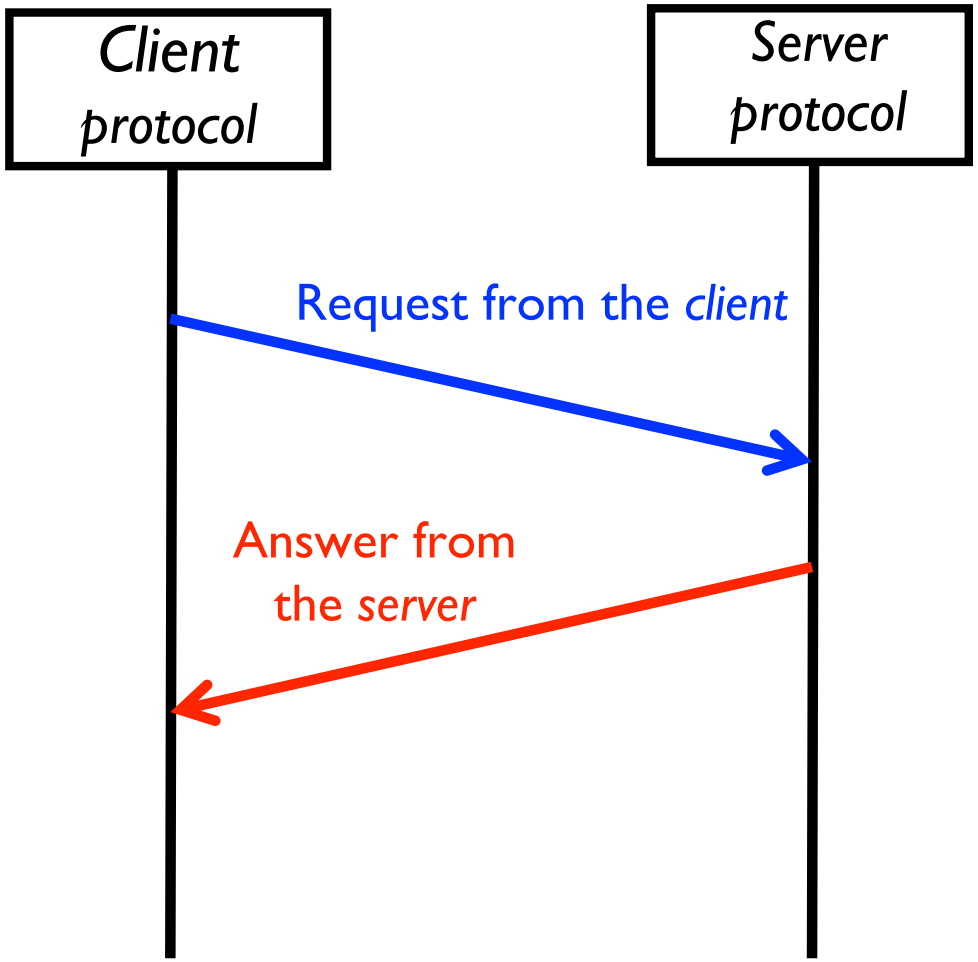
Client-server: definitions

- Client application or terminal process
  - calling party (*caller*, calling party) or requester
  - By extension terminal device that primarily hosts client-type applications
- Server application or terminal process
  - calling party (*callee*, called party) or respondent
  - by extension terminal device hosting primarily server-type applications

Client-server: examples

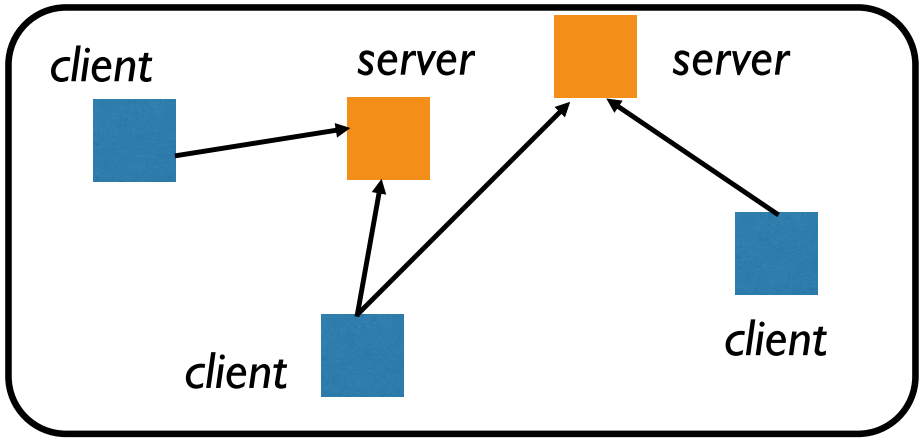
- Browser (client) vs web server (server)
- Email client (client) vs POP3/SMTP server (server)
- Some applications perform both the client and server sides of a communication
  - If hosted applications also have a server side, this side is often inhibited (by configuration, usage, or protection)

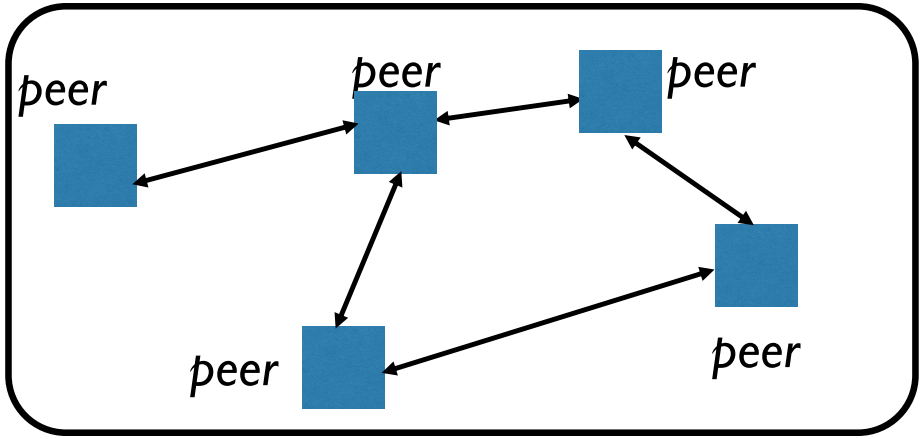
Client-server protocols



- A protocol is defined as client-server when the communication between the parties involved evolves in client-server mode
- Some protocols provide that the two parties can exchange the client or server function within the same communication
- In some cases the client or server mode characterizes only the first part of the communication, and the rest of the communication evolves symmetrically

Peer-to-Peer (P2P) Model





- Homogeneous nodes, generically referred to as peers
  - nodes act as both client and server
  - no distinction between intermediate (relay) and terminal nodes

P2P Network Overlays

- In the case of routing based on node identifiers, the mapping of the nodes to the underlying network addresses and/or any routing is done through a special network structure called a P2P overlay
- Peers cooperate together in maintaining the overlay
- In some cases, a distinction is made between different types of nodes depending on their role in the network architecture (bandwidth, processing/storage capacity, etc.)
- Classification
  - **Unstructured** networks: all peers are equal (no structure)
  - **Structured** networks: the overlay is organized with a specific topology

Characteristics of a P2P network

- Characteristics of network architecture
  - **dynamic** (concepts of join, leave, and churn)
  - **reliable** (redundancy of communication resources and services provided)
  - **scalable** as the number of peers increases
- Main advantages
  - reliability and fault-tolerance (fault-tolerance)
  - automatic distribution of functions and services (self-organization)

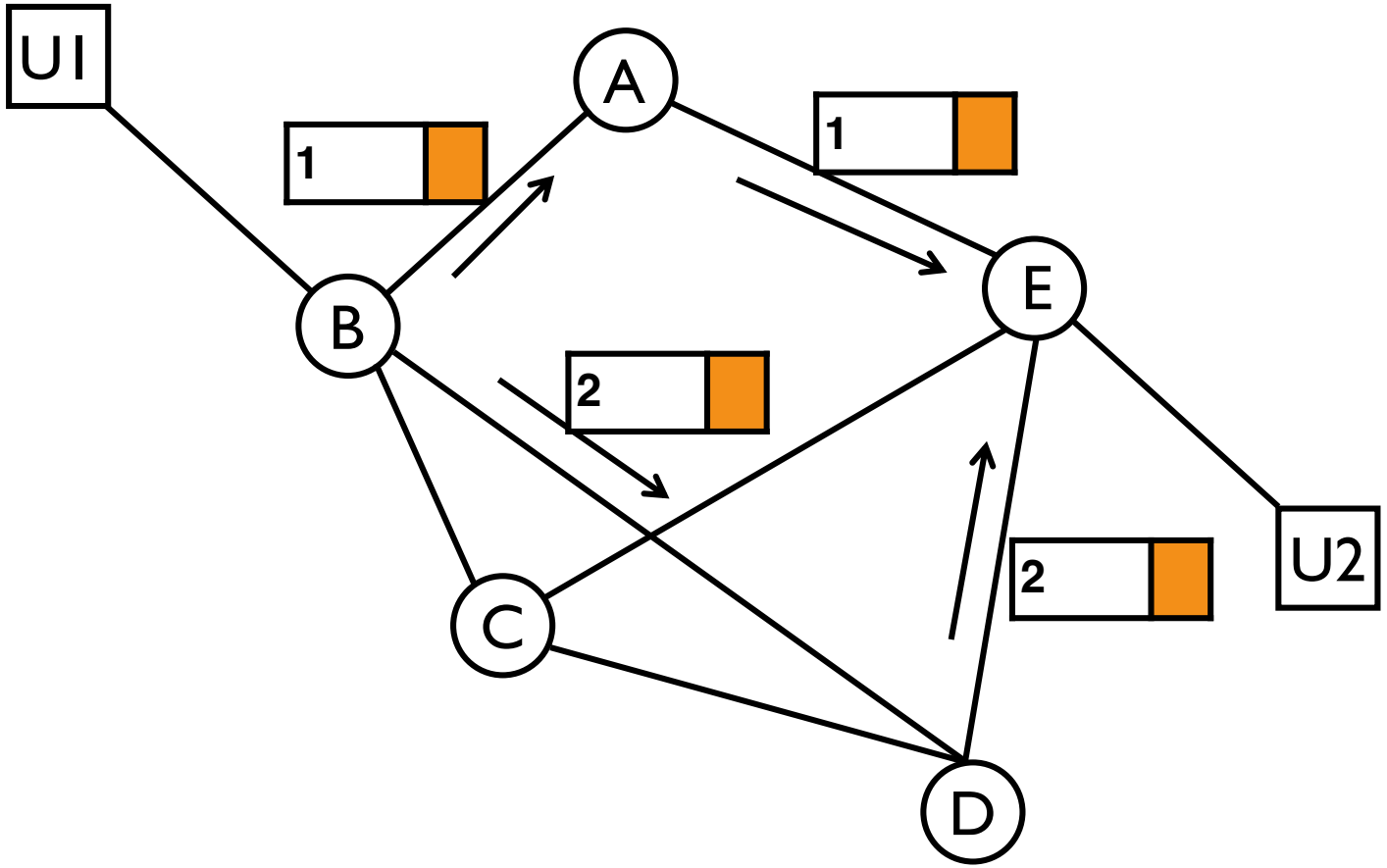
IU transfer models

- **Connectionless** (CL)
  - a single time step (no negotiation, independence and self-consistency of IUs)
  - agreement only between N-user and (N-1)-supplier
- **Connection-Oriented** (CO)
  - three time steps (agreement between the ends of the connection and transfer of the IUs)
  - 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

CL Transfer

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

CL Transfer: an Example



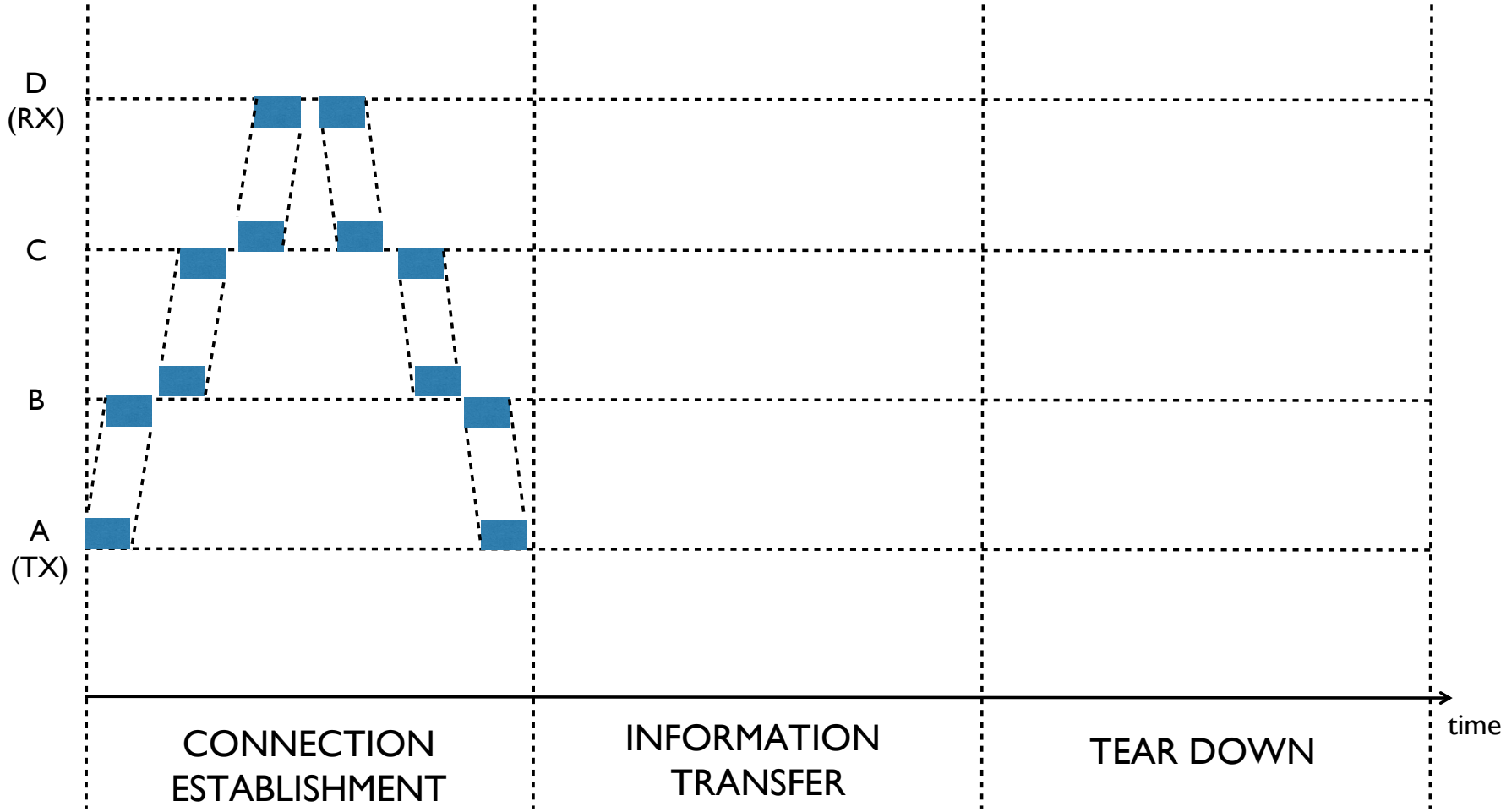
CO Transfer (1)

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

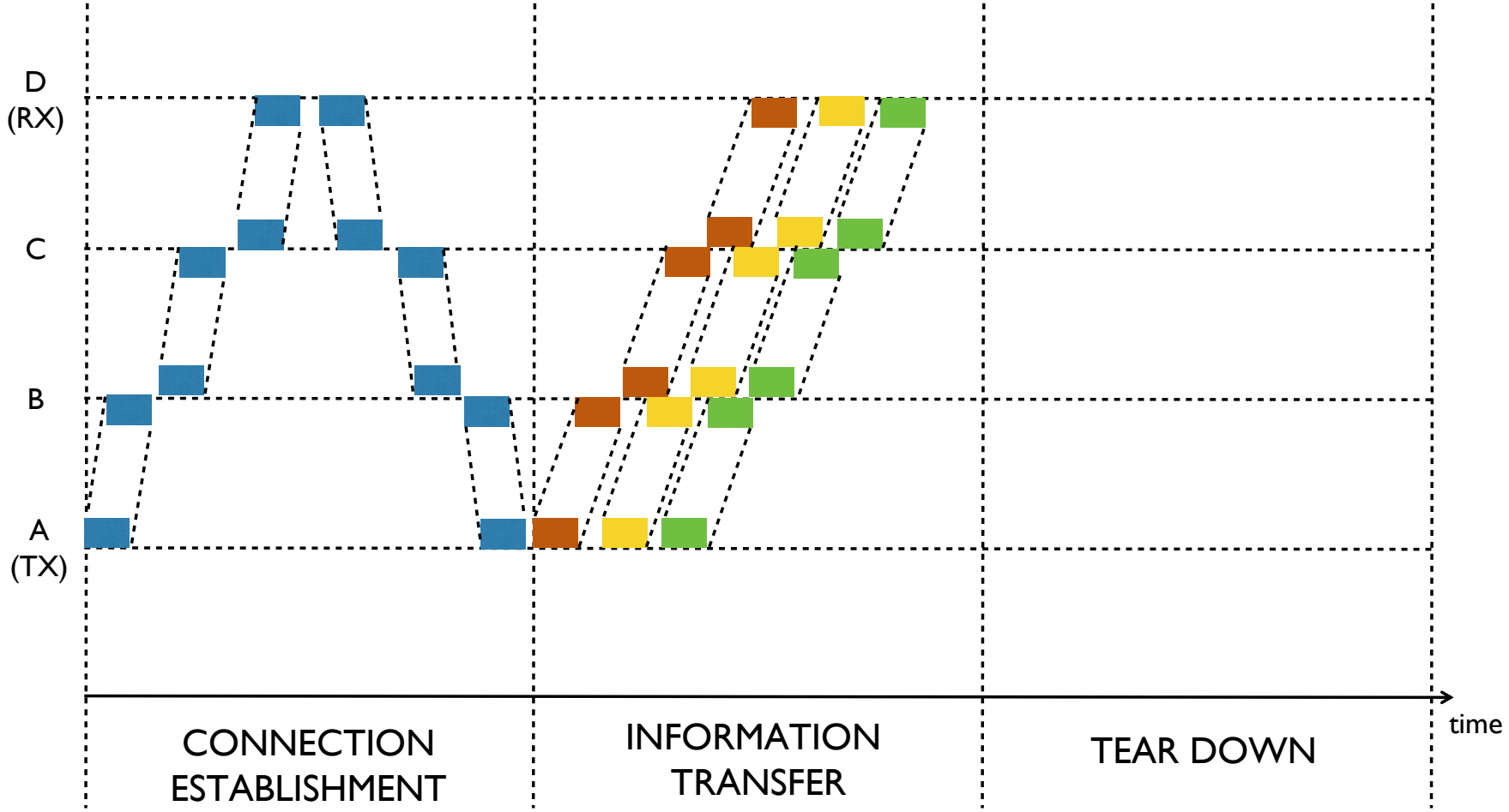
CO Transfer (2)

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)
3. Connection tear-down phase
   - previously allocated resources are released

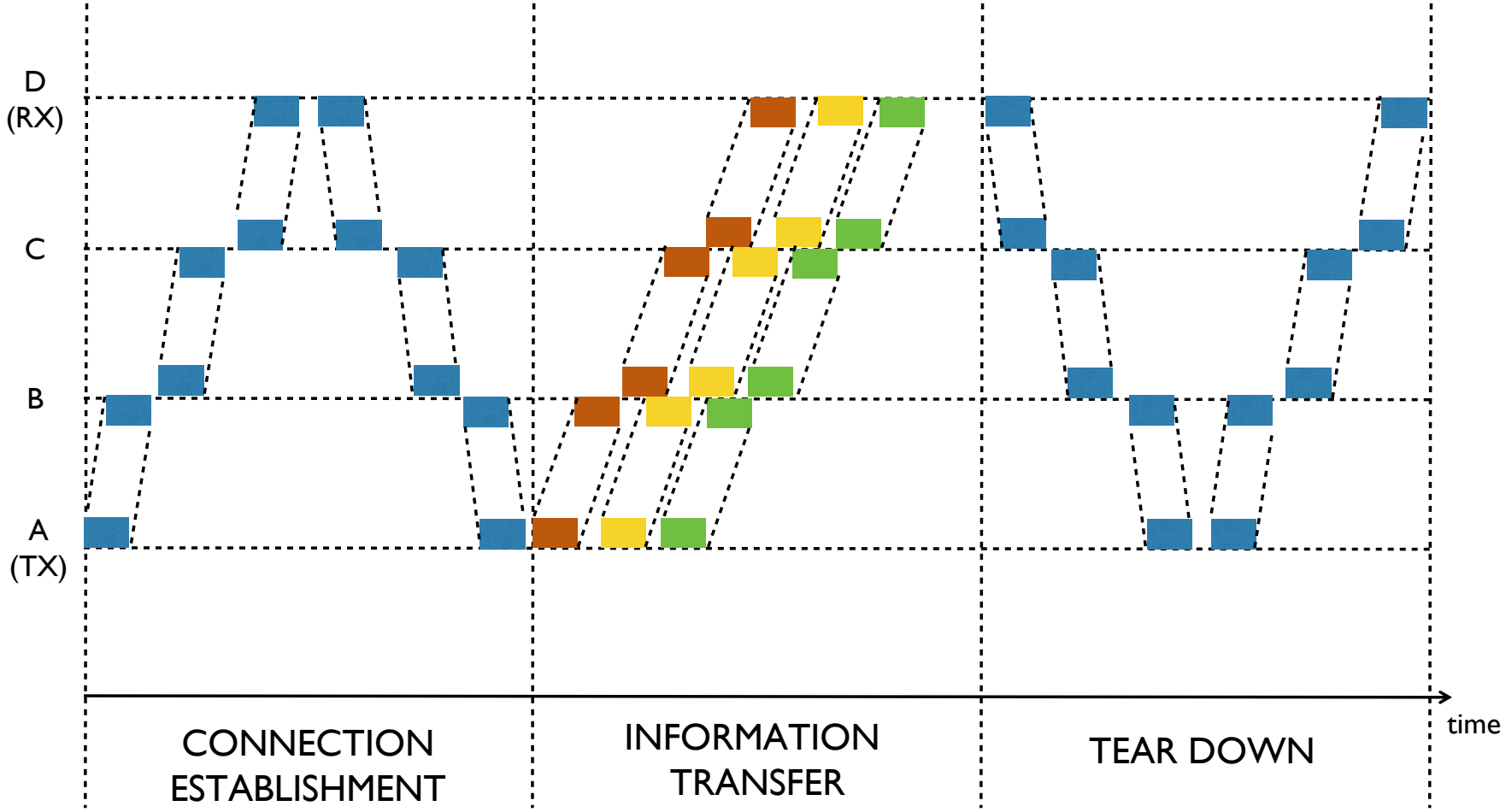
Time Diagram (1)



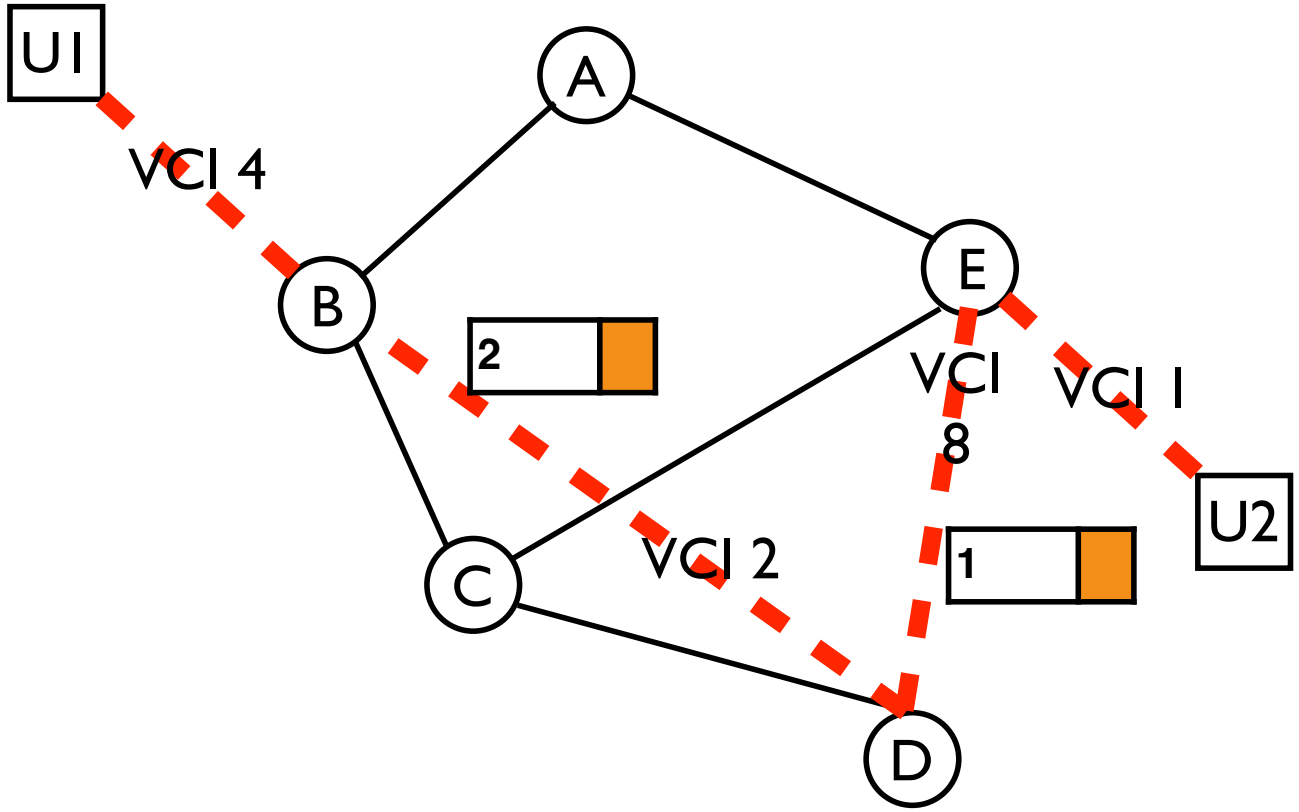
Time Diagram (2)



Time Diagram (3)



CO Transfer: an Example



NODE B	
VCI in	VCI out
4	2

NODE D	
VCI in	VCI out
2	8

NODE E	
VCI in	VCI out
8	1

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)

QoS-based Models (2)

- 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
  - requires specific transfer characteristics in the network
  - there are protocols that are more or less suitable for real-time communications

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)

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)
  - security is tied directly to the data and not to the nodes that handle it
- Also referred to as content-centric networking (CCN)

Outline

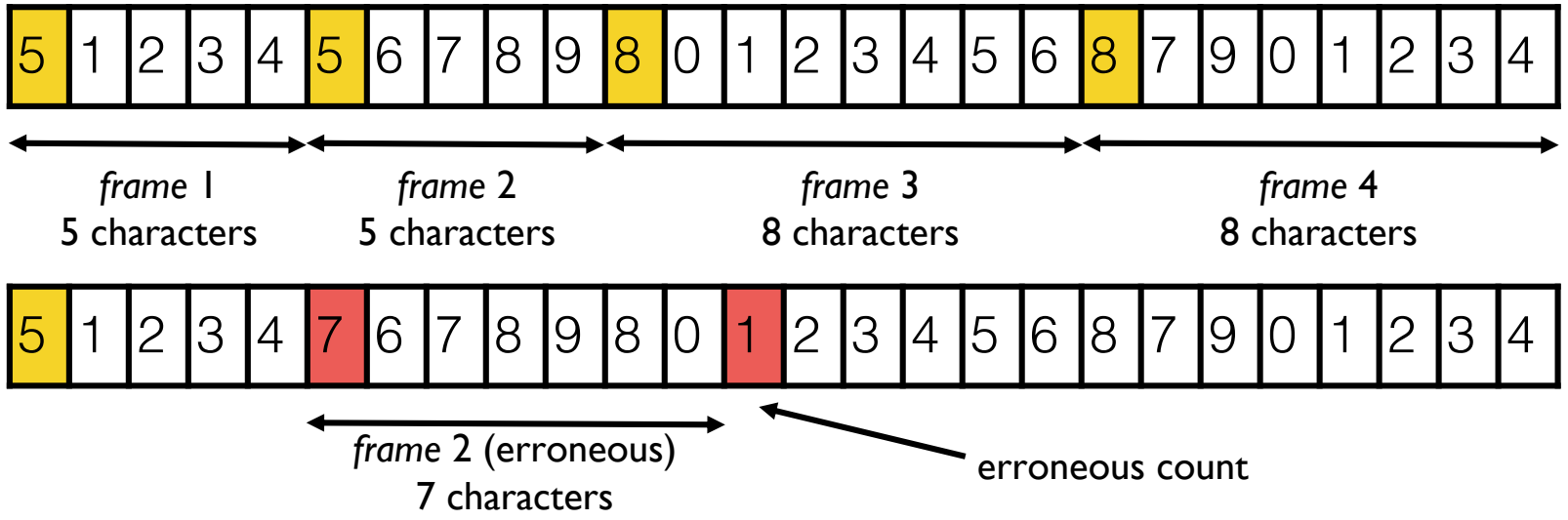
- The OSI and Internet models
- Communication models
- Delimitation
- Sequence control
- Error management

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)

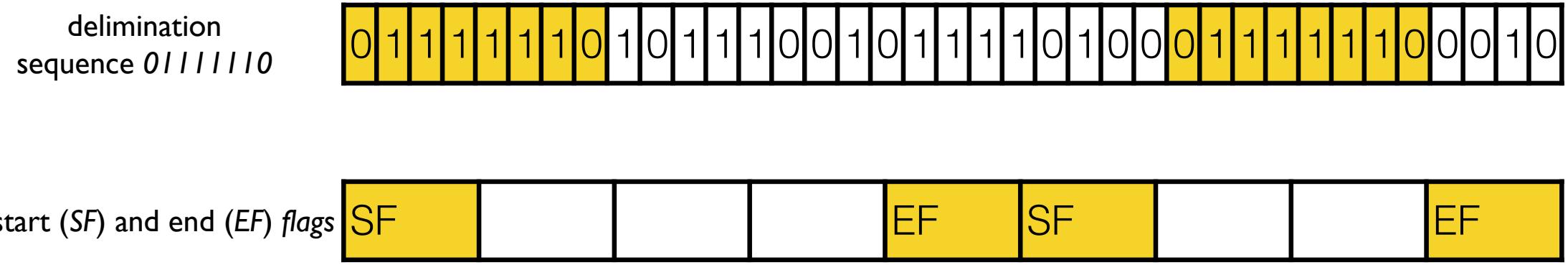
Character count

- 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



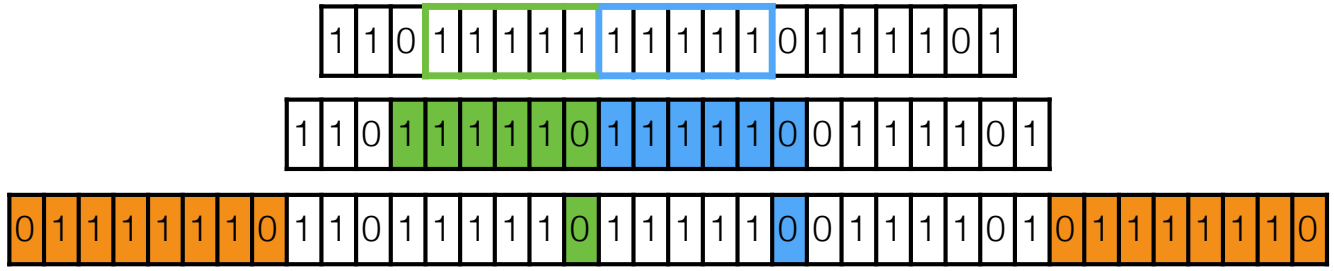
Insertion of delimiters

- 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



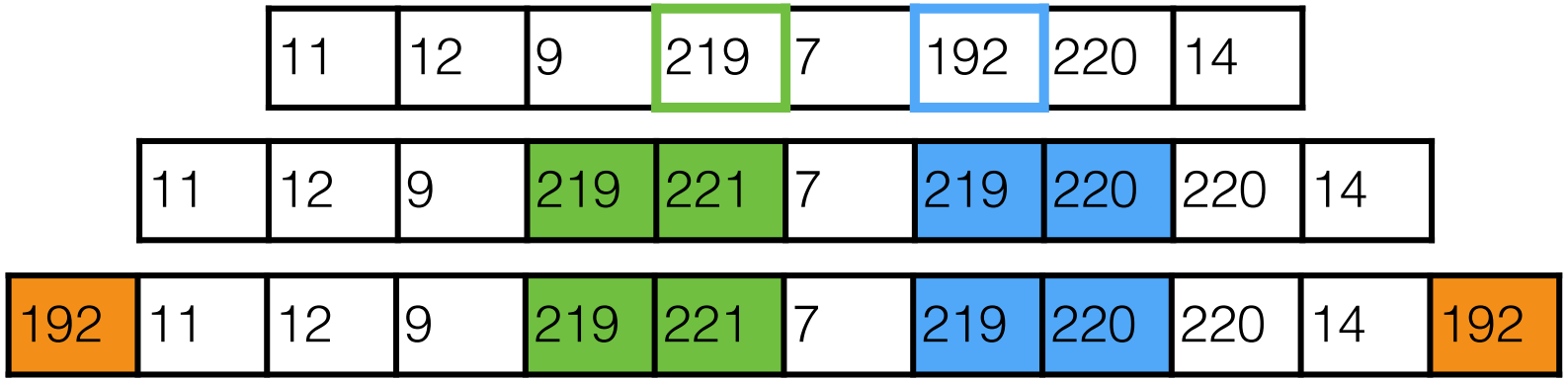
Bit/Char Stuffing (1)

- 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
  - stuffing algorithm: after five consecutive "1's" add a 0
  - de-stuffing algorithm: after five consecutive "1's" remove the following 0



Bit/Char Stuffing (2)

- 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



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

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

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

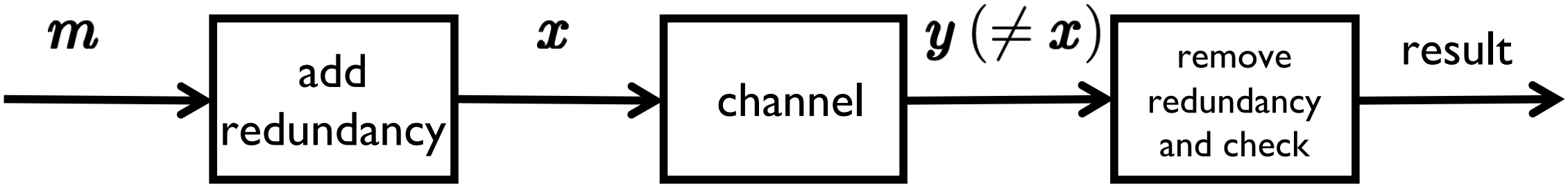
- The OSI and Internet models
- Communication models
- Delimitation
- Sequence control
- Error management

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

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
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

Error Detection



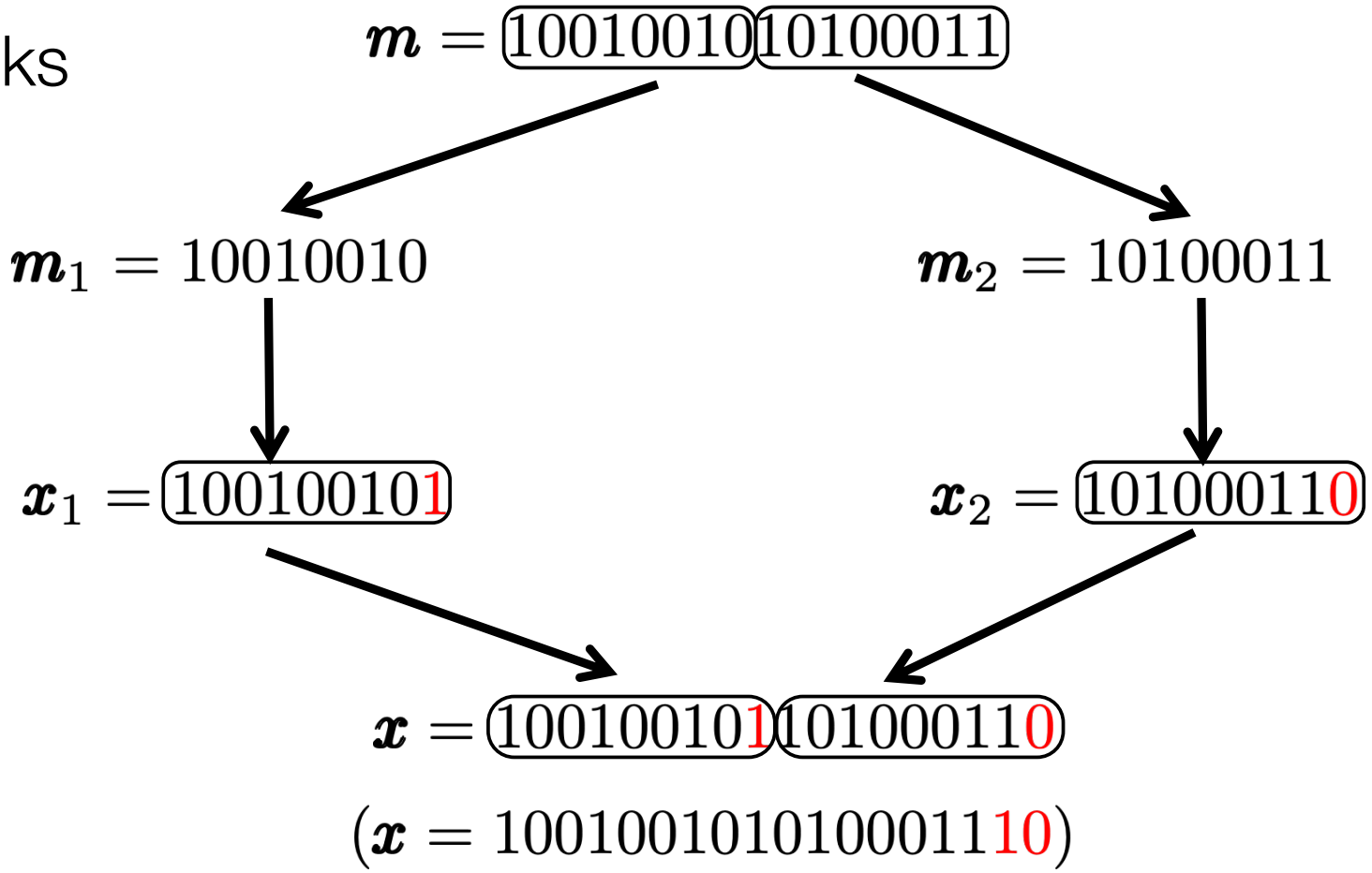
- It is normally based on the addition of redundancy in transmission
  - used in reception to detect (but not correct) errors
  - redundancy required for detection is much smaller than would be required for correction (16-32 bits)
- May be the basis for possible correction/recovery
- Different mechanisms 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)
  - 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



Example (2)

- Consider 1 error in the first block

$$\boldsymbol{y} = 10000010101000111\,0$$

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

$$\boldsymbol{y}_1 = 10100010101000111\,0$$

Block parity check

- To reveal burst errors (in the same block), one can break down the bit sequence into words and organize them into a matrix

k words, *N* bit/word, *k x N* matrix

- Parity bits are calculated on both rows and columns
- A *burst* of less length than the number of columns can be revealed
- Lower efficiency, higher effectiveness
- Alternative: *interleaving*

Example (1)

$$\boldsymbol{m} = \underbrace{01010010}\ \underbrace{00001100}\ \underbrace{00111010}\ \underbrace{11000100}\ \underbrace{00010110}\ \underbrace{01101010}\ \underbrace{10001100}$$

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

$$k = 7$$
$$N = 8$$

$$\boldsymbol{x} = \boldsymbol{m}\ 1001101\ 01010000$$

Example (2)

1 0 0 1 1 0 1 | 0 1 0 1 0 0 0 0

Detection: possible

0	1	0	1	0	0	1	0	1
0	**1**	0	0	1	**0**	0	0	**0**
0	0	1	1	1	0	1	0	0
1	1	0	0	0	1	0	0	1
0	0	0	1	0	1	1	0	1
0	1	1	0	1	0	1	0	0
1	0	0	0	1	1	0	0	1
0	**0**	0	1	0	**1**	0	0	0

Detection: impossible

0	1	0	1	0	0	1	0	1
0	**1**	0	0	1	**0**	0	0	**0**
0	0	1	1	1	0	1	0	0
1	1	0	0	0	1	0	0	1
0	**1**	0	1	0	**0**	1	0	**1**
0	1	1	0	1	0	1	0	0
1	0	0	0	1	1	0	0	1
0	**1**	0	1	0	**0**	0	0	0

1's Complement Sum

- Same matrix organization used in block parity check
- Rows are summed together with 1-complement arithmetic
- The sum, possibly complemented, is the error detection code (checksum)
  - The possible complement is dictated by computational complexity issues
- Used by IP, UDP, TCP and other Internet protocols

Example

0	1	0	0	0	1	0	1	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0
1	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	1	1	0	0	1	1	0	0	0
0	0	0	1	0	1	0	1	1	1	1	0	1	0	1	1
1	0	1	0	0	0	0	0	0	1	0	0	1	1	1	0
0	0	0	1	1	1	0	1	0	0	1	1	1	1	1	0
1	**0**	**0**	**1**	**1**	**0**	**1**	**1**	**0**	**1**	**0**	**0**	**0**	**1**	**0**	**1**
														1	**0**
1	**0**	**0**	**1**	**1**	**0**	**1**	**1**	**0**	**1**	**0**	**0**	**0**	**1**	**1**	**1**
0	**1**	**1**	**0**	**0**	**1**	**0**	**0**	**1**	**0**	**1**	**1**	**1**	**0**	**0**	**0**

partial sum

carry over

1's complement

checksum

Other Codes

- Polynomial codes, also called Cyclic Redundancy Check (CRC).
- The individual binary digits to be transmitted are treated as coefficients (of value "0" or "1") of a given polynomial
- The sending and receiving entities use a common polynomial, called the generator polynomial
- The parity bits and their checks are obtained from polynomial divisions in algebra modulo 2

Error Correction

- Forward Error Correction (FEC) techniques
- Redundancy is used to remedy erroneous bits (if small in number)
- No acknowledgement messages of correct reception are required
  - unidirectional communication
  - there is no need for buffering of sent IUs
- Coding theory
  - added redundancy is generally greater than that used for detection alone
  - redundancy introduced to correct error is constant (fixed)

Example: repetition code

message

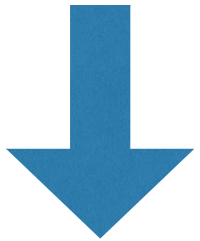
1 0 0 1 1 0 1

coded message (3 repeat code)

111 000 000 111 111 000 111

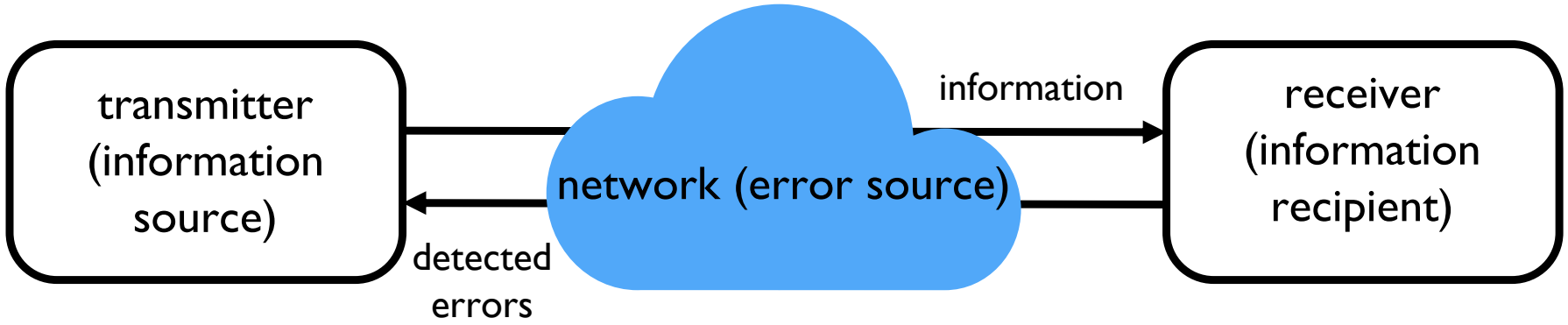
received message

1**0**1 000 0**11** 11 **0** 111 **1**00 111



1 can correct 1 error
or detect 2

Error recovery

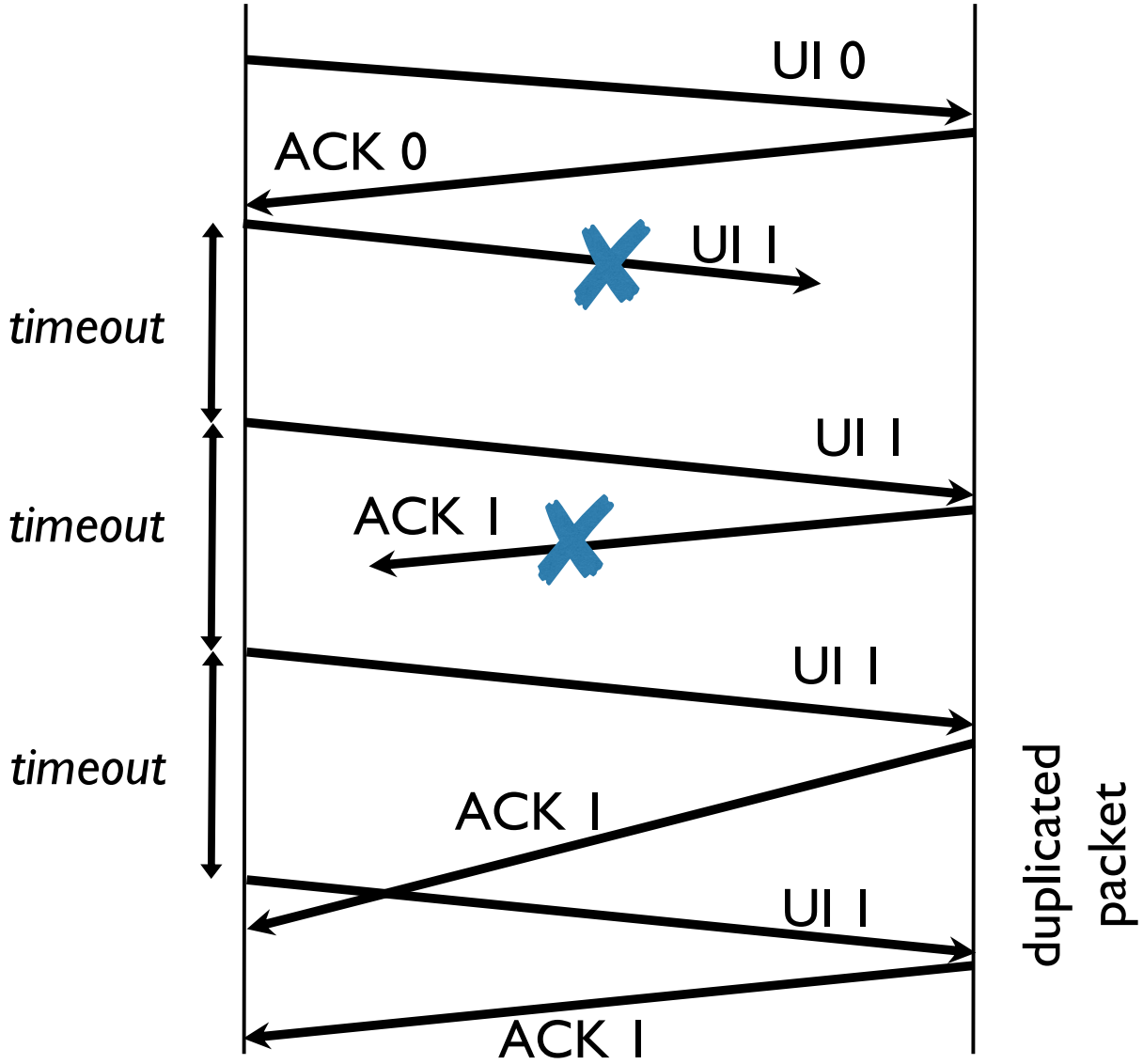


- Error control by automatic retransmission (Automatic Repeat Request, ARQ)
- Different mechanisms used for retransmission (error detection, acknowledgements, timers, IU identifiers)
- ARQ procedures (differ in the size of the windows)
  1. stop and wait: positive acknowledgement mode with reissue
  2. sliding window, go-back-N: variable window mode with non-selective reissue
  3. sliding window, selective repeat: variable window mode with selective reissue

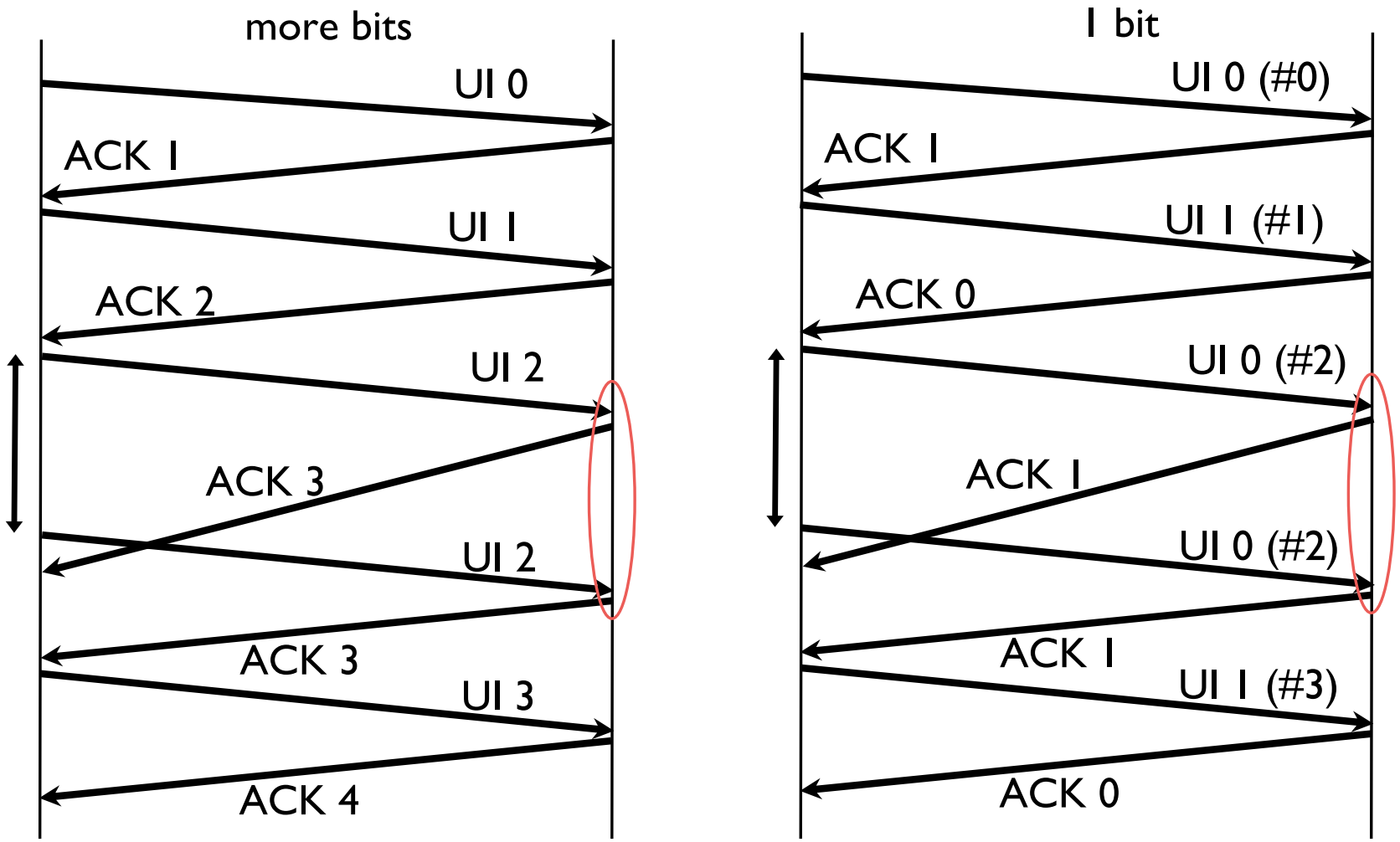
Stop and Wait

- Data IUs are sent individually
- An acknowledgement (acknowledgment, ACK) is expected before sending the next IU
  - Empty IU or with data directed in the opposite direction
  - Acknowledgements are also subject to loss/errors
- It is necessary to activate for each sent IU a retransmission timeout
  - if no ACK has been received when the timeout expires, the IU is retransmitted
  - sufficiently large but not too large
- To avoid duplication, sequence numbers (SeQuence Number, SQN) are used to identify the IUs
  - entered into the PCI with a fixed number of bits, modulo numbering
- Cumulative ACKs: the SQN following the one encountered is reported
- If the sequentiality of the IUs is maintained during the transfer (no out-of-sequence), only one bit is needed to count the SQN

Example


UI 0
ACK 0
UI 1
timeout
UI 1
ACK 1
timeout
UI 1
timeout
ACK 1
UI 1
ACK 1
duplicated packet

Example with SQN



Stop and wait: performance (1)

No error

- Total time required to send an IU and receive feedback

$$T_1 = T_{\mathrm{U}} + T_{\mathrm{A}} + 2T_{\mathrm{p}}$$

- Maximum degree of channel utilization

$$\rho_0 = \frac{T_{\mathrm{U}}}{T_1} = \frac{T_{\mathrm{U}}}{T_{\mathrm{U}} + T_{\mathrm{A}} + 2T_{\mathrm{p}}}$$

$$= \begin{cases} \dfrac{1}{2 + 2T_{\mathrm{p}}/T_{\mathrm{U}}} & \text{se } T_{\mathrm{U}} = T_{\mathrm{A}} \\ \dfrac{1}{2T_{\mathrm{p}}/T_{\mathrm{U}}} & \text{se } T_{\mathrm{U}} \gg T_{\mathrm{A}} \\ 0 & \text{se } T_{\mathrm{p}} \gg T_{\mathrm{U}} \end{cases}$$

Stop and wait: performance (2)

With errors

- Source does not receive ACKs (IU or incorrect/lost ACKs)
- Statistical independence of IU and lost ACK events
- Probability of error p
- Geometric distribution of the number of attempts

$$P\{\text{tx ok con } k \text{ tentativi}\} = p^{k-1}(1-p)$$

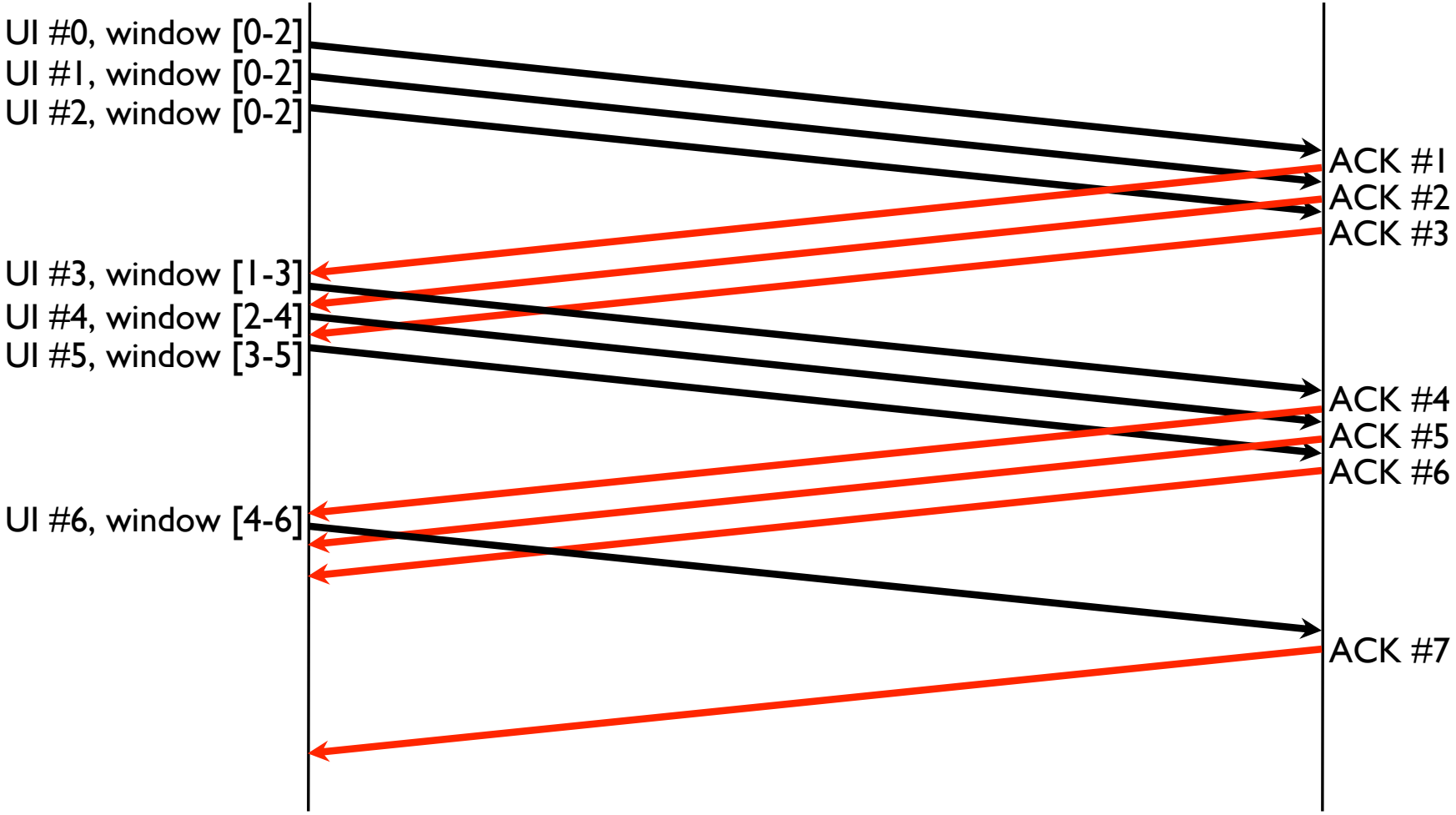
$$N_t = \sum_{k=1}^{\infty} k p^{k-1}(1-p) = \frac{1}{1-p}$$

$$\begin{aligned} \overline{T}_1 &= (N_t - 1)T_O + T_1 \\ &= \frac{p}{1-p} T_O + T_1 \\ &\simeq \frac{T_1}{1-p} \\ \rho &= (1-p)\rho_0 \end{aligned}$$

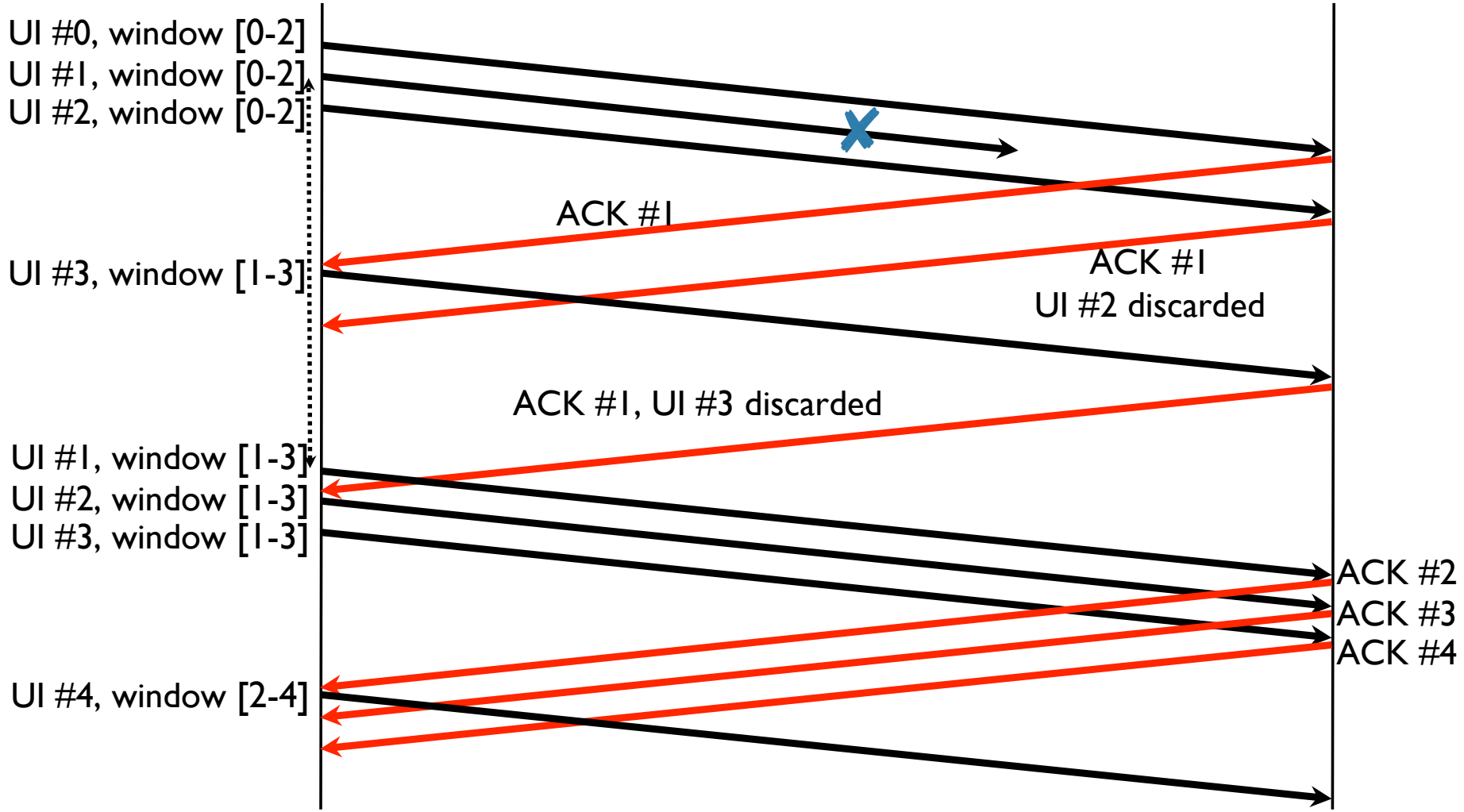
Sliding window

- Throughput can be increased by allowing the transmitter to send subsequent IUs without waiting for the corresponding acknowledgements
  - A copy of the IUs must be maintained until a positive ACK is received
- SQNs of consecutive IUs that can be issued without receiving acknowledgement can be considered a "window" that flows
- If the transmitter receives an ACK greater than the lowest position of the window it can advance it to that position, ignoring the fact that some ACKs have not yet been received
- Receive window: range of IUs that can also be received non-sequentially
- Protocols can be differentiated according to the ability of the receiver to keep received IUs out of sequence
  - *go-back-N* (unitary window) or *selective repeat* (larger window)

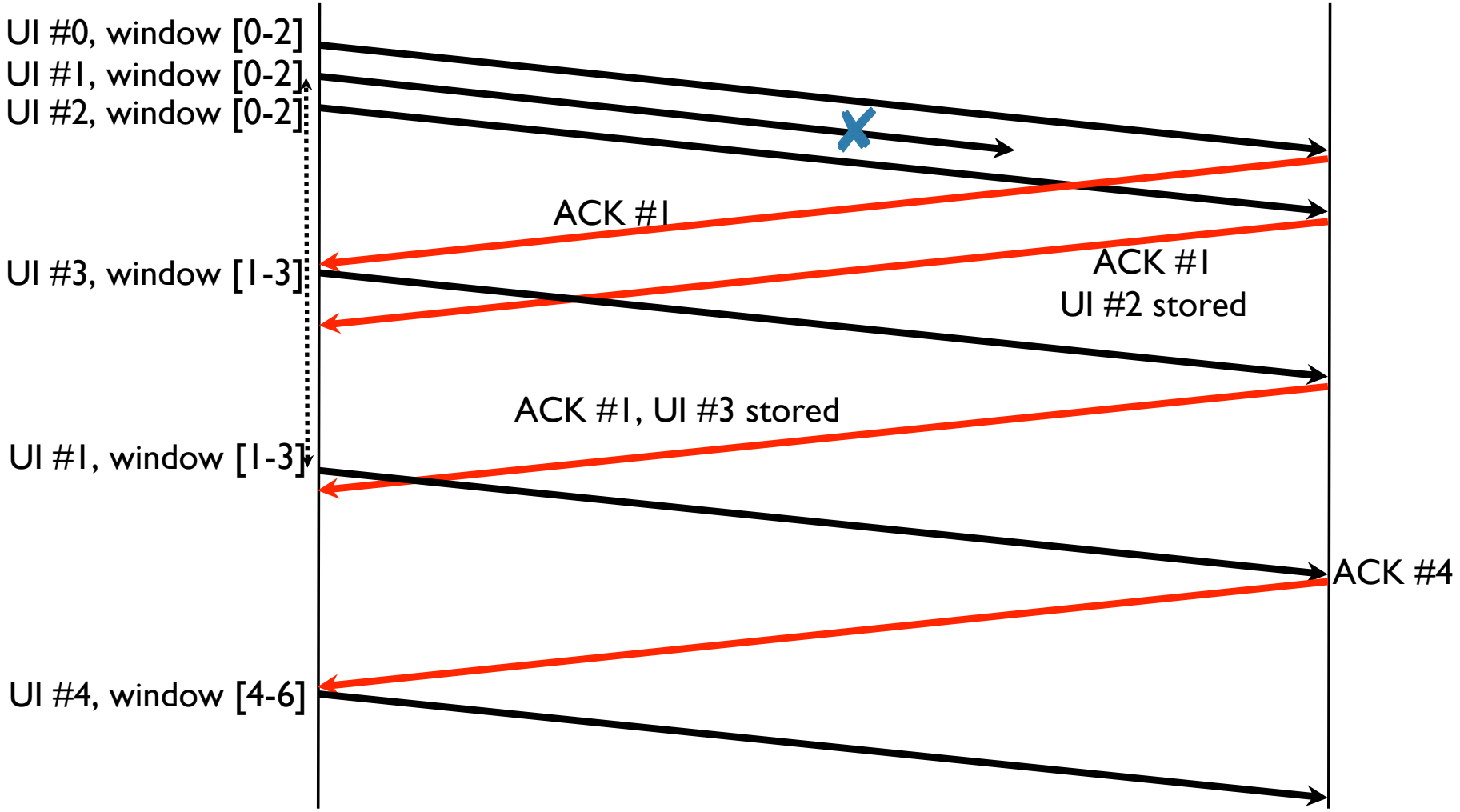
Example: basic operations



Example: *go-back-N*



Example: *selective repeat*




thank you
danke
謝謝
ngiyabonga
рахмат
teşekkür ederim
Баярлалаа
спасибо
faafetai lava
dank je
gracias
tapadh leat
bedankt
dziękuję
obrigado
sukriya
kop khun krap
go raibh maith agat
merci
terima kasih
dakujem
arigatô
takk
grazie
tack
barka
mersi
welalin
spas
kia ora
vinaka
спасибі
blagodaram
misaotra
matondo
paldies
grazzi
mahalo
kiitos
dankie
dhanyavad
nandri
nanni
maruuru
köszönöm
hvala
bayarlalaa
gràcie
enkosi
akun
dankon
aciu
djiere dieuf
tau
mochchakkeram
хвала
asante
manana
obrigada
murakoze
chokrane
tenki
mamnun
дякую
chnorakaloutioun
gratias ago
gràcies
sulpáy
taiku
sobodi
dēkuji
sagolun
mèsi
didi madloba
najis tuke
ありがとう
tanemirt
rahmet
kam sah hamnida
rahmat
diolch
dhanyavadagalu
shukriya
merci
мерси
trugarez
তোমাকে ধন্যবাদ
감사합니다
xièxie
ευχαριστώ