INSTITUT
POLYTECHNIQUE
DE PARIS

MQTT Packet Format

Ada Diaconescu

ada.diaconescu@telecom-paris.fr

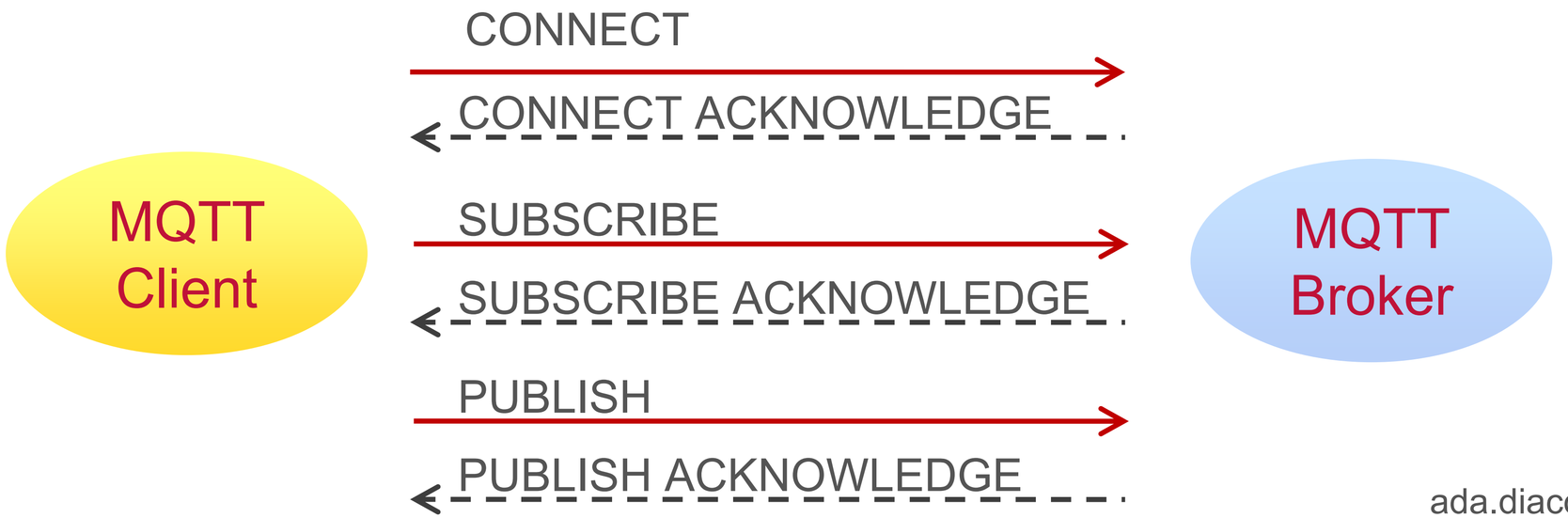
Main Topics

- MQTT message format.
- MQTT message header
- Message fields & coding
- Control Message coding example

ada.diaconescu@telecom-paris.fr

MQTT Overview – Reminder

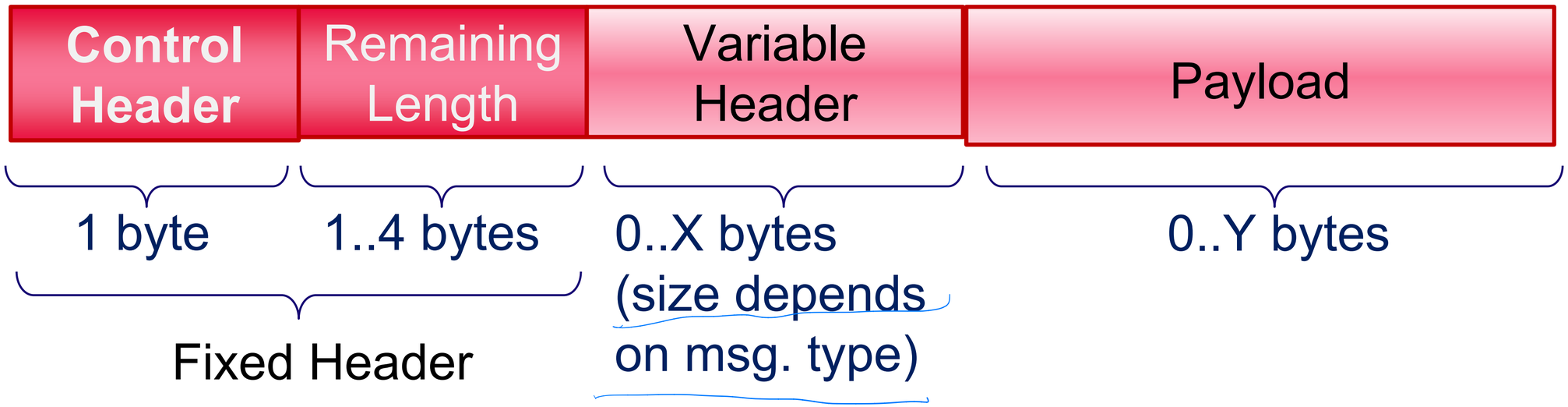
- Binary protocol
  - Control elements are binary bytes (rather than text or other formats)
  - Topic names, Client ID, Usr & Psw are encoded as UTF-8 Strings
  - Payload is binary, with application-specific format & content
- Protocol based on general format: command & command acknowledgement
  - Each command type has a corresponding acknowledgement command



ada.diaconescu@telecom-paris.fr

MQTT Packet Structure

- Fixed Header (compulsory): 2 bytes minimum (5 bytes max)
- Variable Header (optional): variable length
- Payload (optional): variable length



ada.diaconescu@telecom-paris.fr

MQTT Packet → Possible Formats & Length

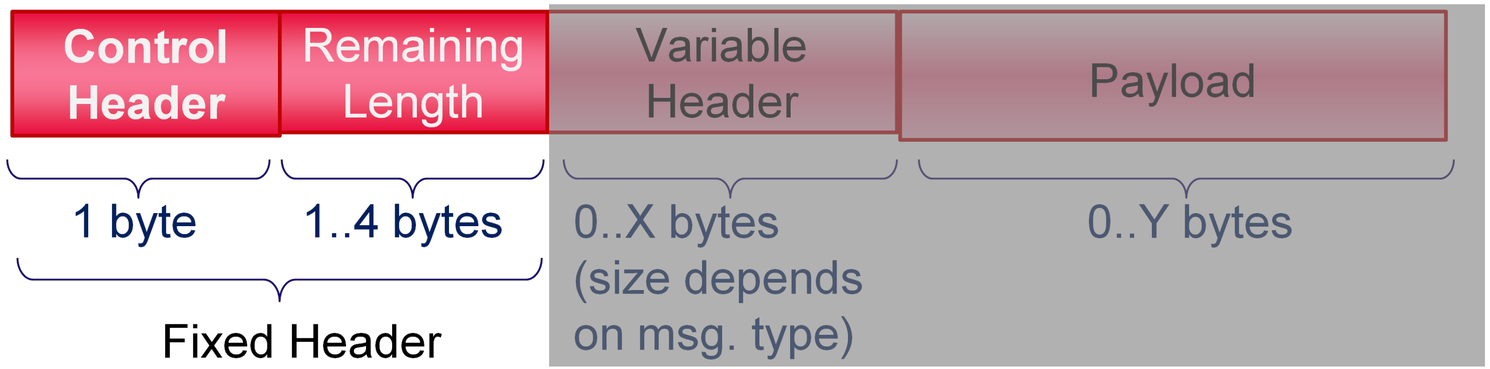
- Fixed Header – e.g. CONNACK → minimum 2 bytes
- Fixed Header & Variable Header – e.g. PUBACK
- Fixed Header & Variable Header & Payload – e.g. CONNECT, PUBLISH, ..

ada.diaconescu@telecom-paris.fr

Fixed Header

bit	7	6	5	4	3	2	1	0
byte 1	Message Type				DUP flag	QoS level		RETAIN
byte 2	Remaining Length							

- Control field (1 byte)
  - 4 most significant bits: message type
  - 4 least significant bits: message flags (duplicate, QoS, retain)
- Remaining Length field (between 1 & 4 bytes)
  - For each byte: 1 bit as continuation flag & 7 bits package length (base 128)
    - E.g. One byte needed to code the length of packages smaller than 128 bytes
  - Min packet size: 2bytes (1byte control field + 1byte packet length field)
  - Max package size: 256 MB

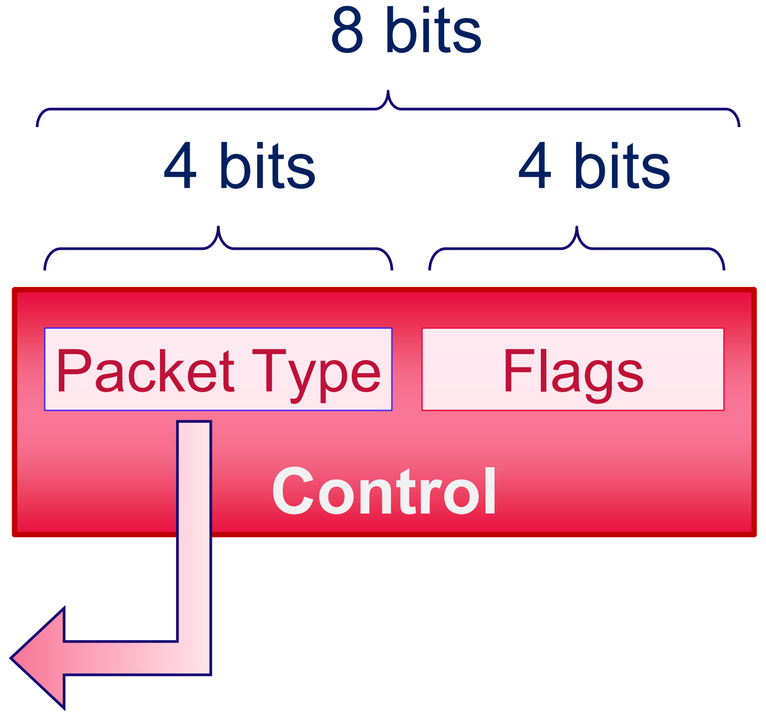


Control field

bit	7	6	5	4	3	2	1	0
byte 1	Message Type				DUP flag	QoS level		RETAIN
byte 2	Remaining Length							

- Represents all protocol commands & responses
- 1st packet byte : 2 x 4bit fields
  - First 4 bits: command / message / packet type
  - Last 4 bits: control flags
- Examples of Message / Packet Types:

Name	Value (decimal)	Value (binary)
Reserved	0	0000
CONNECT	1	0001
CONNACK	2	0010
PUBLISH	3	0011
PUBACK	4	0100
SUBSCRIBE	8	1000
SUBACK	9	1001
DISCONNECT	14	1110
Reserved	15	1111



Name	Value	Direction of flow	Description
Reserved	0	Forbidden	Reserved
CONNECT	1	Client to Server	Client request to connect to Server
CONNACK	2	Server to Client	Connect acknowledgment
PUBLISH	3	Client to Server or Server to Client	Publish message
PUBACK	4	Client to Server or Server to Client	Publish acknowledgment
PUBREC	5	Client to Server or Server to Client	Publish received (assured delivery part 1)
PUBREL	6	Client to Server or Server to Client	Publish release (assured delivery part 2)
PUBCOMP	7	Client to Server or Server to Client	Publish complete (assured delivery part 3)
SUBSCRIBE	8	Client to Server	Client subscribe request
SUBACK	9	Server to Client	Subscribe acknowledgment
UNSUBSCRIBE	10	Client to Server	Unsubscribe request
UNSUBACK	11	Server to Client	Unsubscribe acknowledgment
PINGREQ	12	Client to Server	PING request
PINGRESP	13	Server to Client	PING response
DISCONNECT	14	Client to Server	Client is disconnecting
Reserved	15	Forbidden	Reserved

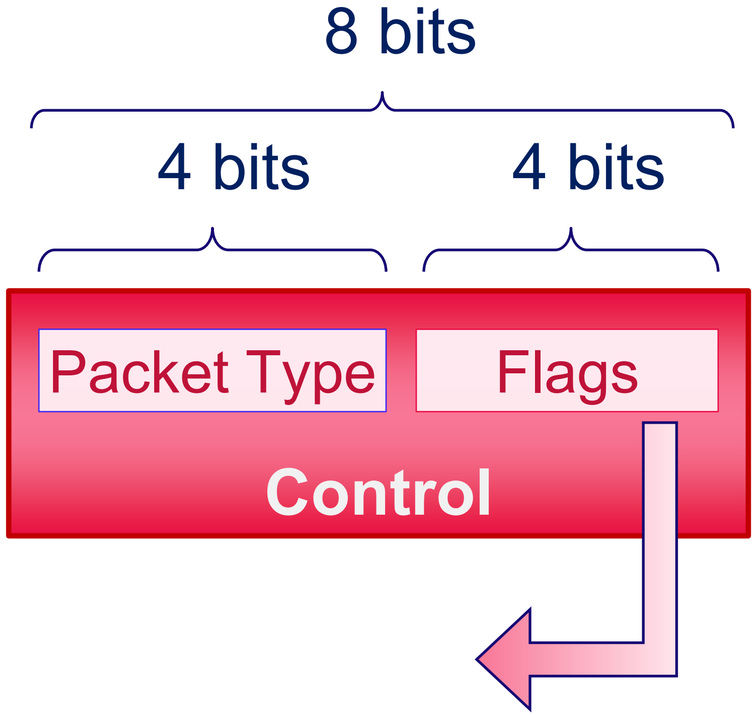
Package Types

MQTT v3.1.1

← Full List

http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html

Control field → flags



- **Last 4 bits of the fixed header**
- E.g. Flags for PUBLISH messages (MQTT v3.1.1):
  - Bit3 → **DUP:** Duplicate delivery of a PUBLISH message (with QoS 2 or 1)
  - Bit2 → **QoS:** PUBLISH Quality of Service (10 → Qos2)
  - Bit1 → **QoS:** PUBLISH Quality of Service (01 → QoS1 ; 00 → QoS0)
  - Bit0 → **RETAIN:** PUBLISH Retain flag

bit	7	6	5	4	3	2	1	0
byte 1	Message Type				DUP flag	QoS level		RETAIN
byte 2	Remaining Length							

Control Packet	Fixed header flags	Bit 3	Bit 2	Bit 1	Bit 0
CONNECT	Reserved	0	0	0	0
CONNACK	Reserved	0	0	0	0
PUBLISH	Used in MQTT 3.1.1	DUP[1]	QoS[2]	QoS[2]	RETAIN[3]
PUBACK	Reserved	0	0	0	0
PUBREC	Reserved	0	0	0	0
PUBREL	Reserved	0	0	1	0
PUBCOMP	Reserved	0	0	0	0
SUBSCRIBE	Reserved	0	0	1	0
SUBACK	Reserved	0	0	0	0
UNSUBSCRIBE	Reserved	0	0	1	0
UNSUBACK	Reserved	0	0	0	0
PINGREQ	Reserved	0	0	0	0
PINGRESP	Reserved	0	0	0	0
DISCONNECT	Reserved	0	0	0	0

Fixed Header Flags

MQTT v3.1.1

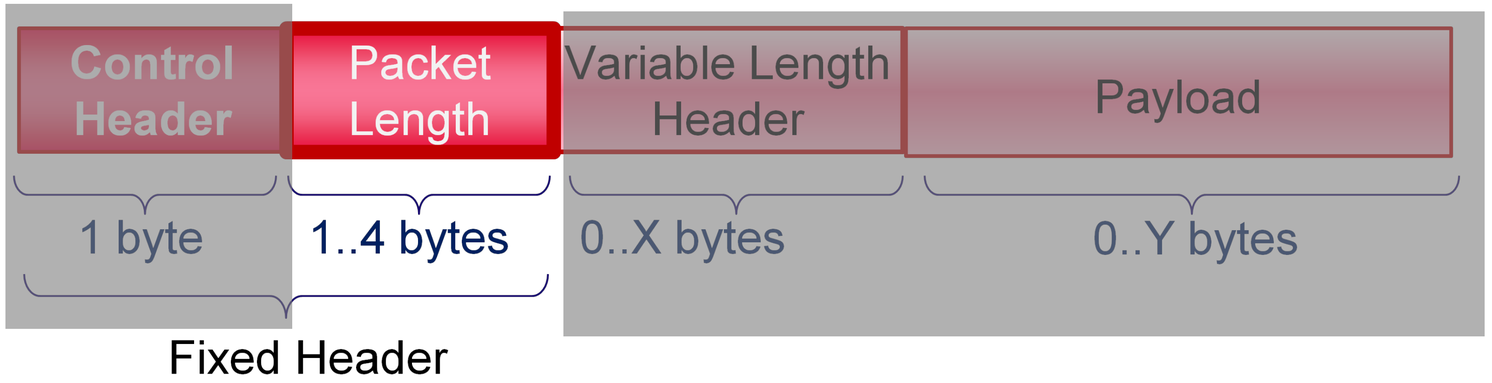
← Full Listing

http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html

Remaining Length

bit	7	6	5	4	3	2	1	0
byte 1	Message Type				DUP flag	QoS level		RETAIN
byte 2	Remaining Length							

- Starts at the packet's 2nd byte; consists of 1 to 4 bytes; least significant byte first !
- The number of bytes remaining within the current packet
  - Including data in the Variable Header and the Payload
  - Excluding the bytes used to encode the Remaining Length.
- Each byte
  - 1st most significant bit: used as a continuation flag → are there following bytes?
  - 7 least significant bits: used to code for packet length → 128 values



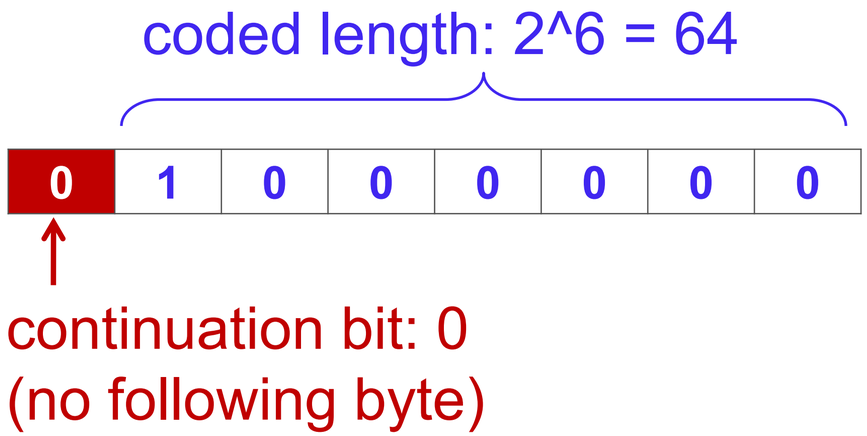
Remaining Length (2)

- Consider encoding in base 128 (2^7)
  - 1 byte → packets up to 127 bits in length (2^7 -1)
  - 2 bytes → packets up to 16,383 bits in length (2^14 -1)
  - 3 bytes → packets up to 2,097,151 bits in length (2^21 -1)
  - 4 bytes → packets up to 268,435,455 bits in length (2^28 -1) ; 256 MB
- Range of packet sizes:

Digits	From	To
1	0 (0x00)	127 (0x7F)
2	128 (0x80, 0x01)	16 383 (0xFF, 0x7F)
3	16 384 (0x80, 0x80, 0x01)	2 097 151 (0xFF, 0xFF, 0x7F)
4	2 097 152 (0x80, 0x80, 0x80, 0x01)	268 435 455 (0xFF, 0xFF, 0xFF, 0x7F)

Remaining Length → Example 1

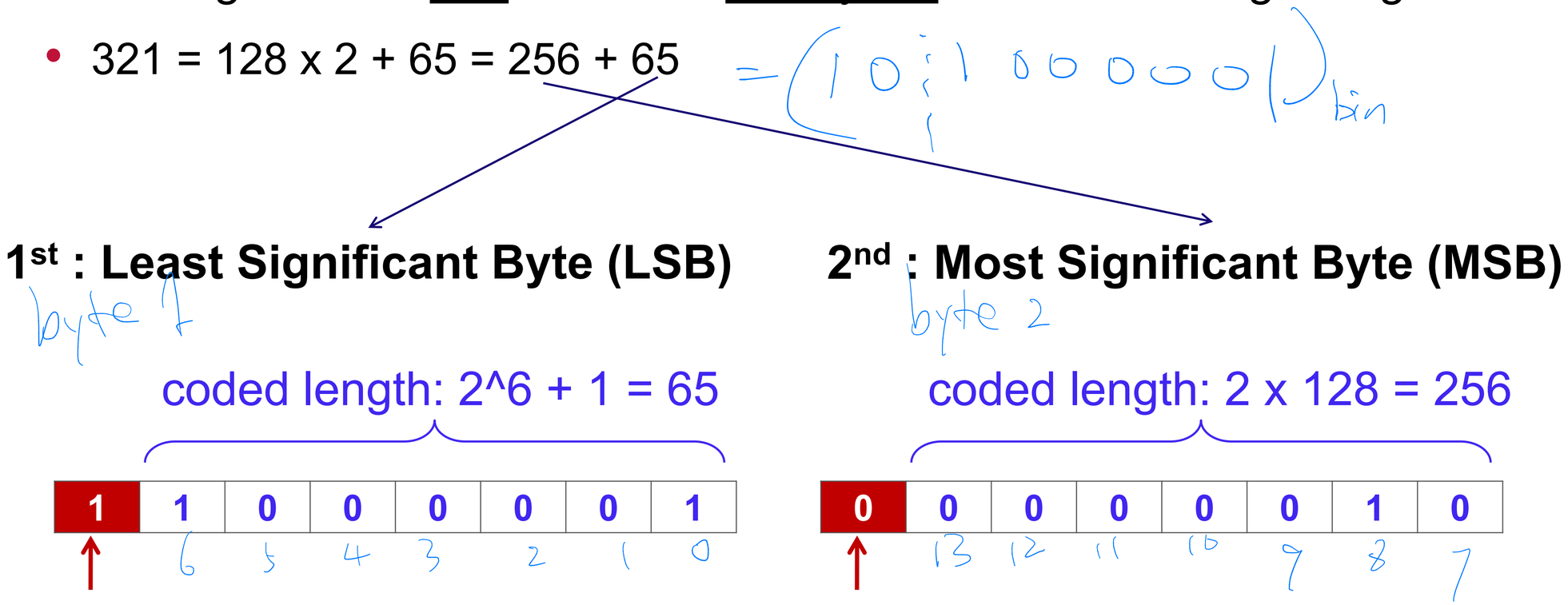
- Encoding decimal 64 → need one byte in the Remaining Length



ada.diaconescu@telecom-paris.fr

Remaining Length → Example 2

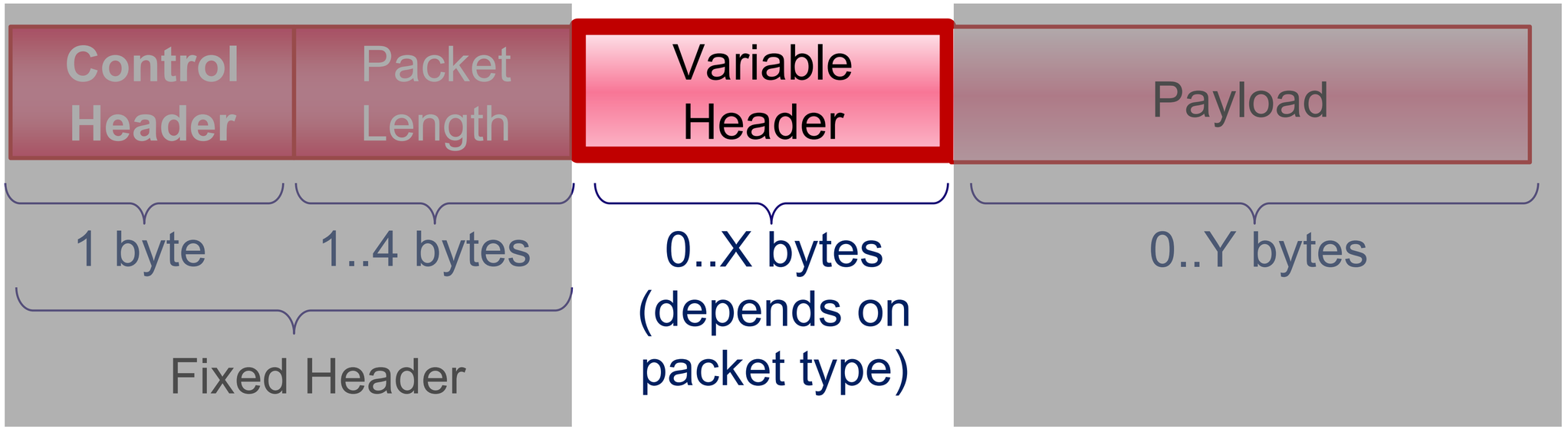
- Encoding decimal 321 → need two bytes in the remaining Length
  - 321 = 128 x 2 + 65 = 256 + 65



ada.diaconescu@telecom-paris.fr

Variable Header - only required for some packet types

- Carry additional control information
- Length & content varies with the packet type
- E.g. Pub & Sub exchanges with QoS > 0 : 2 byte Packet Identifier



ada.diaconescu@telecom-paris.fr

Variable Header → CONNECT message

- Protocol name: e.g. “MQTT” string → UTF-encoding (hex) → 0x4d 0x51 0x54 0x54
- Protocol version (8 bit): e.g., MQTT version 3 → 00000011
- Connect Flags (1 byte)
  1. Clean Session: 0 → server stores the client’s session ; 1 → server discards client info
  2. Will: 1 → server sends Message to subscribers when the client fails (also needs to set Will QoS & Will Retain flags ; & Will Topic & Will Message in the payload).
  3. & 4. Will QoS: the QoS of the Will Message (set via the Will flag & Message in payload)
  5. Will Retain: the Retain setting for the Will Message
  6. & 7. Psw & Usr: 1 → a password / username, repsectively, is included in the payload

bit	7	6	5	4	3	2	1	0
	User Name Flag	Password Flag	Will Retain	Will QoS		Will Flag	Clean Session	Reserved

- Keep Alive (2 bytes): number of seconds to wait between PINGREQ & PINGRESP

Variable Header → CONNACK message

- Connect return code (1 byte): unsigned return code → 0 typically indicates success

Enumeration	HEX	Meaning
0	0x00	Connection Accepted
1	0x01	Connection Refused: unacceptable protocol version
2	0x02	Connection Refused: identifier rejected
3	0x03	Connection Refused: server unavailable
4	0x04	Connection Refused: bad user name or password
5	0x05	Connection Refused: not authorized
6-255		Reserved for future use

ada.diaconescu@telecom-paris.fr

Variable Header → PUBLISH message

- Topic name:
  - The key that identifies the channel to with the message payload data is published
  - UTF-8 encoded String
  - Max length: 32,767 characters
- Subscribers use the same key to identify the channels from which they wish to receives information

ada.diaconescu@telecom-paris.fr

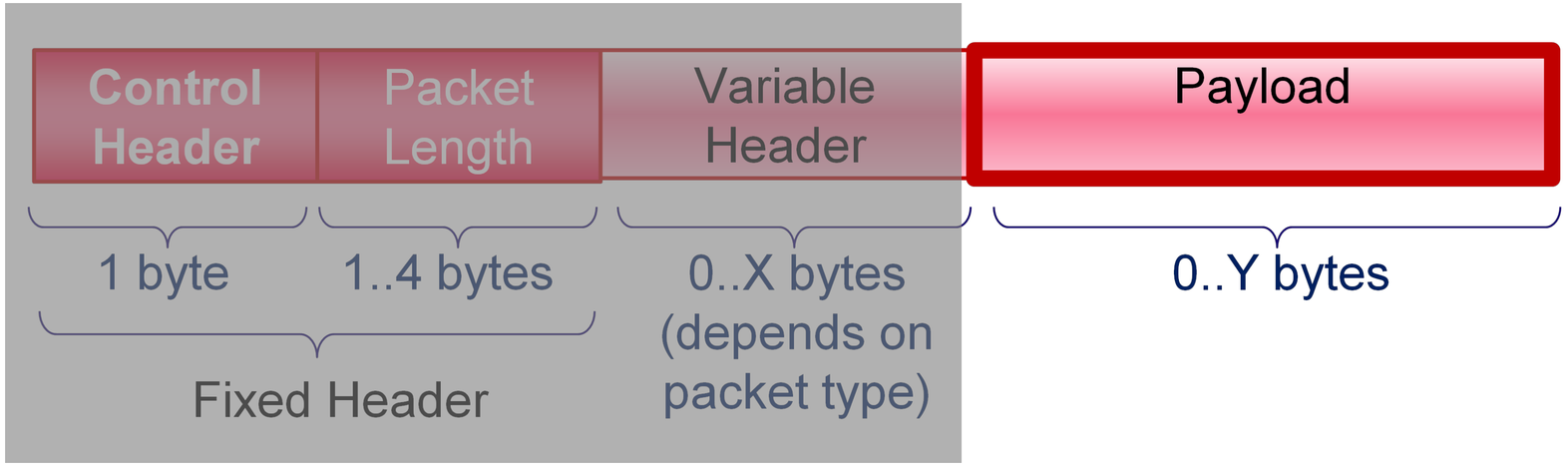
Variable Header → Message Identifier

- For message types:
  - PUBLISH, PUBACK, PUBREC, PUBREL, PUBCOMP, SUBSCRIBE, SUBACK, UNSUBSCRIBE, UNSUBACK
- Only for message with QoS 1 & 2 (in the fixed header)
- Unique among all communications in a given direction
- 16 bit unsigned integer
- Do *not* use 0 as message id (invalid reserved value)

ada.diaconescu@telecom-paris.fr

Payload – varies with packet type

- CONNECT: 1..* UTF-8 encoded strings
  - Client Id (compulsory); Will topic & msg; usr & psw
- SUBSCRIBE: list of topic names to subscribe to & QoS level (UTF encoded strings)
- SUBACK: list of granted QoS levels; same order as in the SUBSCRIBE msg.
- PUBLISH: application-specific data only



ada.diaconescu@telecom-paris.fr

UTF-8 encoding for strings in MQTT

- Efficient encoding of Unicode character strings (e.g. for text communication)
- MQTT strings: first two bytes used for the length of the string encoding (in bytes)
- E.g. UTF-8 encoding of string OTWP

bit	7	6	5	4	3	2	1	0
byte 1	Message Length MSB (0x00)							
	0	0	0	0	0	0	0	0
byte 2	Message Length LSB (0x04)							
	0	0	0	0	0	1	0	0
byte 3	'O' (0x4F)							
	0	1	0	0	1	1	1	1
byte 4	'T' (0x54)							
	0	1	0	1	0	1	0	0
byte 5	'W' (0x57)							
	0	1	0	1	0	1	1	1
byte 6	'P' (0x50)							
	0	1	0	1	0	0	0	0

ada.diaconescu@telecom-paris.fr

CONNECT Packet example

- CONNECT, Clean Session = True (1), Client ID = Phyton1

Byte	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
Meaning	Header	Remaining Length	Length of prtocol name		Protocol Name +Version					Connect Flags	Keep Alive		Length								
Hex	0x10	0x13	0x0	0x4	0x4d	0x51	0x54	0x54	0x4	0x2	0x0	0x3c	0x0	0x7	0x70	0x79	0x74	0x68	0x6F	6E	0x31
Ascii		19		4	M	Q	T	T	4			60		7	P	Y	T	H	O	N	1

- Byte 1: control code for CONNECTION type: 0001 0000 → 0x10
- Byte 2: remaining length: 19bytes → 0x13
- Bytes 3..4: Length of protocol name: 0000 0000 0000 0100 → 0x0 0x4
- Bytes 5..9: protocl name & version: MQTT 4 → 0x4d 0x51 0x54 0x54 0x4
- Byte 10: connect flags: 0000 0010 → 0x2 (clean session flag = 1)
- Bytes 11..12: keep alive: 60 secs → 0000 0000 0011 1100 → 0x0 0x3c
- Bytes 13..14: Length of Client ID: 7 → 0x7
- Bytes 15..21: Client ID: PHYTON1 → 0x70 0x79 0x74 0x68 0x6F 0x6E 0x31

MQTT Resources

- MQTT homepage: https://mqtt.org
- MQTT message format:
  - https://docs.oasis-open.org/mqtt/mqtt/v3.1.1/mqtt-v3.1.1.pdf
  - http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html
  - http://www.steves-internet-guide.com/mqtt-protocol-messages-overview/

ada.diaconescu@telecom-paris.fr