

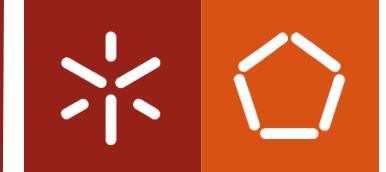
IoT – Internet of Things

Novos Paradigmas de Rede

2º Semestre

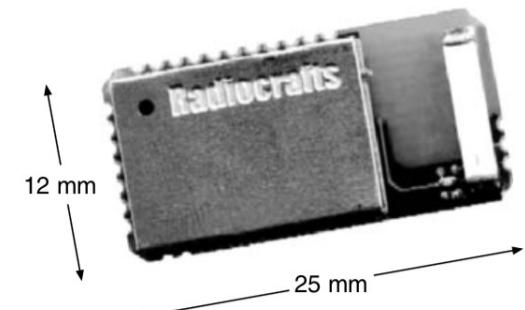
2023/2024



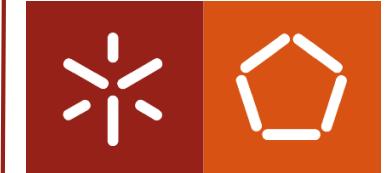


Introdução

- **Visão IoT:** centenas milhões de dispositivos embebidos, designados por *smart objects*, tornam-se verdadeiros dispositivos IP e consequentemente parte da Internet
 - Coisas com *chips* anexados, que podem determinar o seu estado e enviar para a *cloud*, ou receber ordens para atuar sobre a *coisa*
 - Serviços que controlam as coisas são normalmente *Serviços Web*
 - IP significa apenas IPv6 dado o número de dispositivos (endereçamento 2^{128} e não apenas 2^{32})
- **Nos anos 1990 a 2000** surgiram muitos dispositivos embebidos *proprietários* de baixo custo, muito baixo consumo energético
- **Necessidade de normas:**
 - **IEEE 802.15.4, Bluetooth LE (Low Energy), ...**



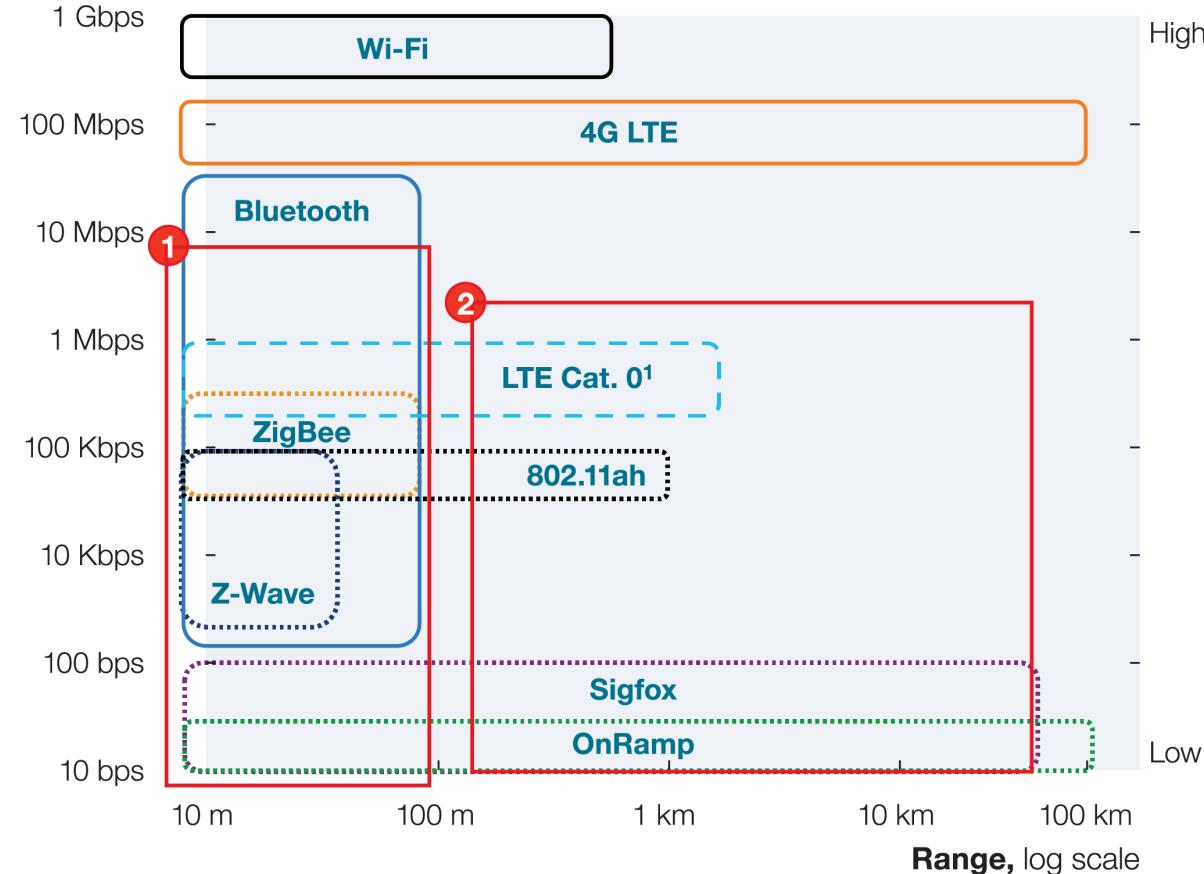
Z. Shelby and C. Bormann, 6LoWPAN: The Wireless Embedded Internet,



Introdução

— Widely adopted ······ New standard - - - Established, adoption ongoing

Data rate, log scale

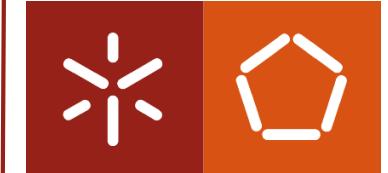


- ① Many competing standards for low-range, medium-low data rate hinder growth for many IoT applications
 - Interoperability missing
 - Consortia wars might be emerging
 - Additional incompatibilities in higher communication layers, eg, 6LoWPAN vs ZigBee

- ② Standard white space for low-data-rate, low-power, high-range applications such as smart grid
 - Wi-Fi and LTE have high power consumption
 - Alternatives with low power and wide range (eg, LTE Cat. 0, 802.11ah, Sigfox, and OnRamp) are in very early stages and compete against each other

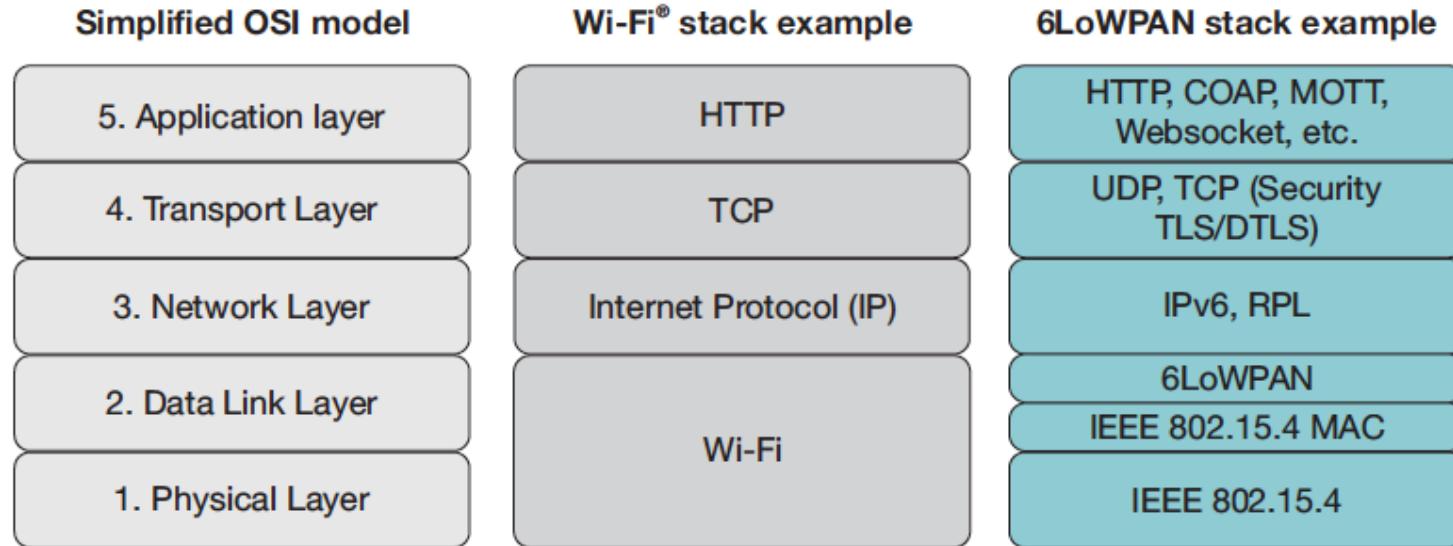
IoT Standards and Protocols, <https://www.postscapes.com/internet-of-things-protocols/#protocols>

(Source: <https://www.mckinsey.com/industries/semiconductors/our-insights/internet-of-things-opportunities-and-challenges-for-semiconductor-companies>)

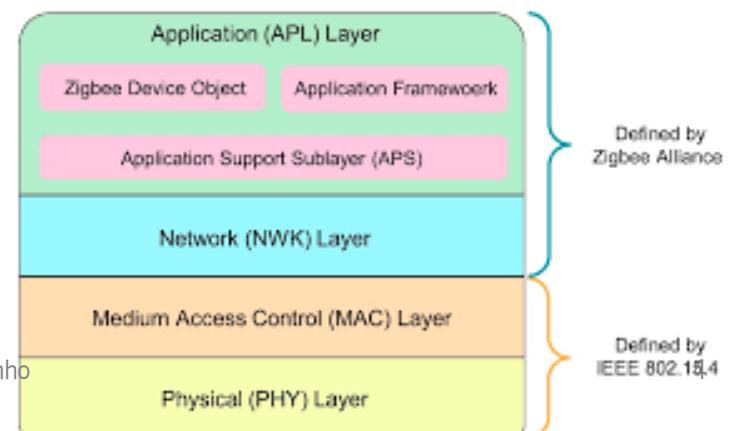


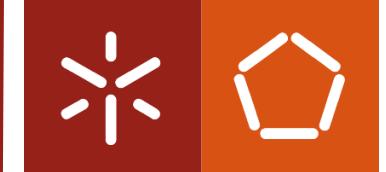
Introdução

- **Foco na pilha protocolar: CoAP / 6LoWPAN / IEEE802.15.4**



- Embora existam outras (ex: Zigbee)





Introdução

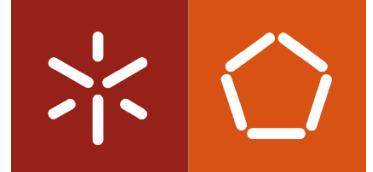
*IPv6 over low-power wireless personal area networks (**6LoWPAN**)*

● Principais objetivos:

- Poder ligar os dispositivos sem fios facilmente à Internet sem necessidade de Gateways ou Proxies de conversão
- O uso do IP permite tirar partido da infraestrutura de rede já existente
- Standards abertos e disponíveis para todos

● Aplicações:

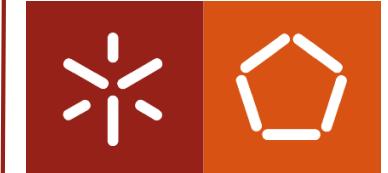
- Domótica e automação
- Cuidados de Saúde
- Logística
- Automação industrial
- Monitorização ambiental em tempo real
- Infraestruturas de *Smart meetering* e *smart grid*
- Etc..



Introdução

*IPv6 over low-power wireless personal area networks (**6LoWPAN**)*

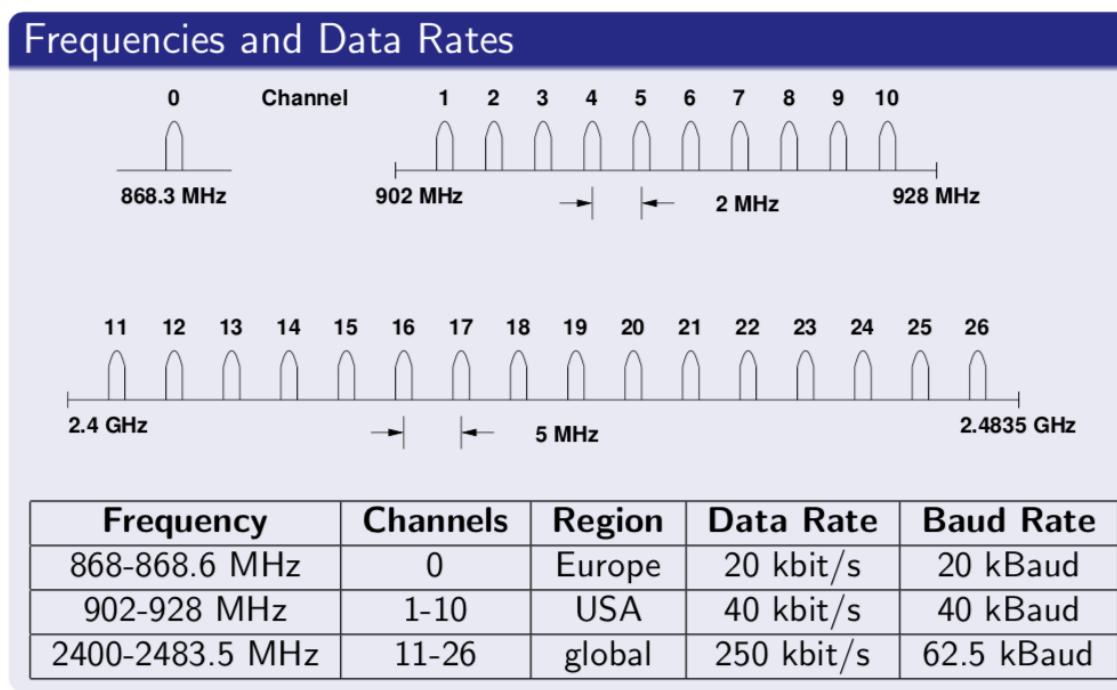
- O primeiro grupo de trabalho focou-se apenas na adaptação do IPv6 ao IEEE 802.15.4
- **6LoWPAN foi desenhado para o IEEE 802.15.4**
- Depois, seguiu-se IPv6 sobre BLE (*Bluetooth Low Energy*)
- E outras tecnologias dos níveis físico e MAC:
 - IEEE 802.11ah, MS/TP Networks , DECT *Ultra Low Energy*, NFC *Near Field Communication*, ITU-T G.9959 Networks
- As propostas de compressão dos cabeçalhos são as mesmas!

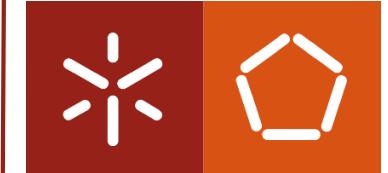


IEEE 802.15.4

● Características do IEEE 802.15.4

- Menos de **1%** do consumo de energia do WiFi (802.11)
- Baixa potência radio: inferior **a 1mW**
- Baixa largura de banda: **250 Kbps**
- Robustez ao nível físico; Camada MAC simples: pode ser usada pelo **IP** ou pelo **ZigBee**



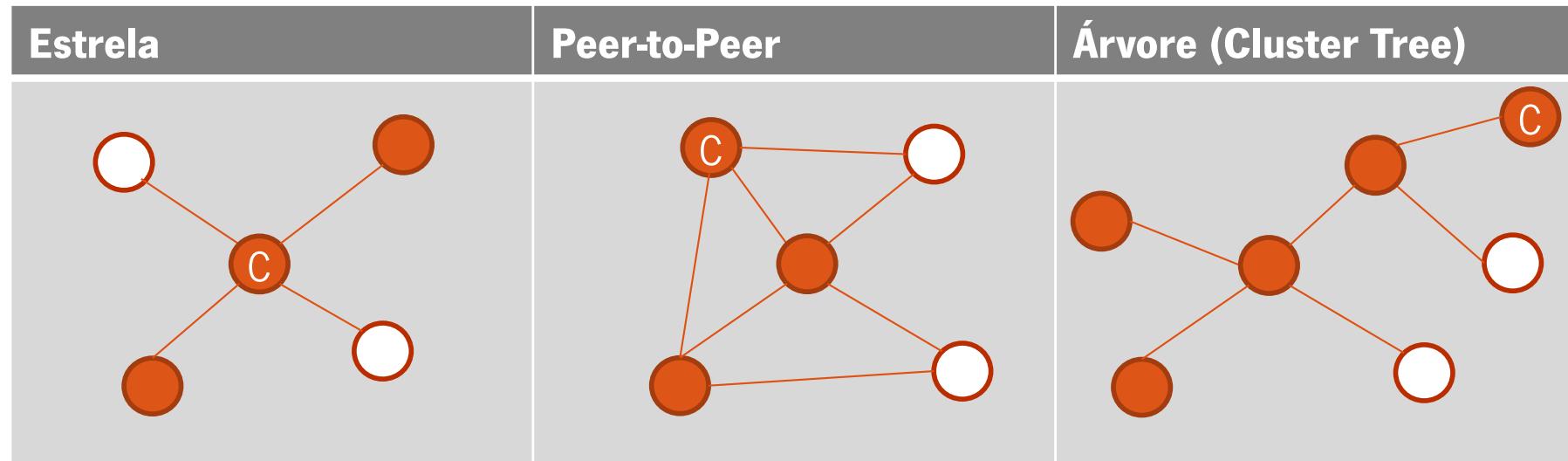


IEEE 802.15.4

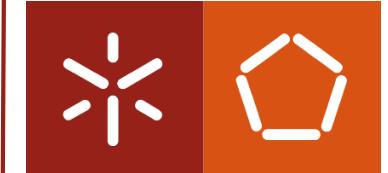
- **Classes de dispositivos IEEE 802.15.4**

- **FFD** (*Full Function Device*) – Implementam todo o conjunto de protocolos (podem assumir qualquer papel na topologia)
- **RFD** (*Reduced Function Device*) – Implementação simplificada e reduzida (só podem ser folhas nas topologias)

- **Topologias típicas de uma PAN (Personal Area Network) IEEE 802.15.4**



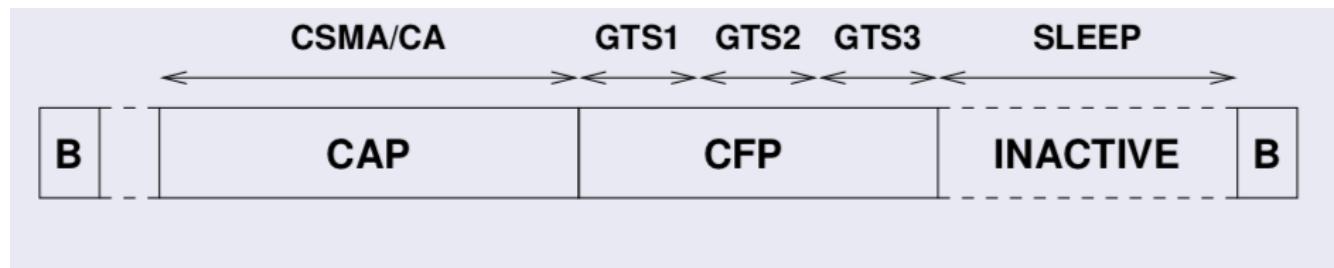
- **Funções na Topologia:** Nós **folha**, Nós **Coordenador** e Nós **Coordenador da PAN** (um e um só por topologia, que é o nó central na topologia em estrela)

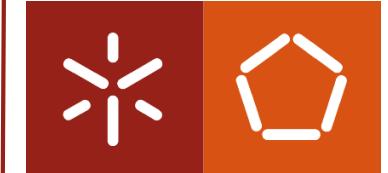


IEEE 802.15.4

- **Protocolos de acesso ao meio**

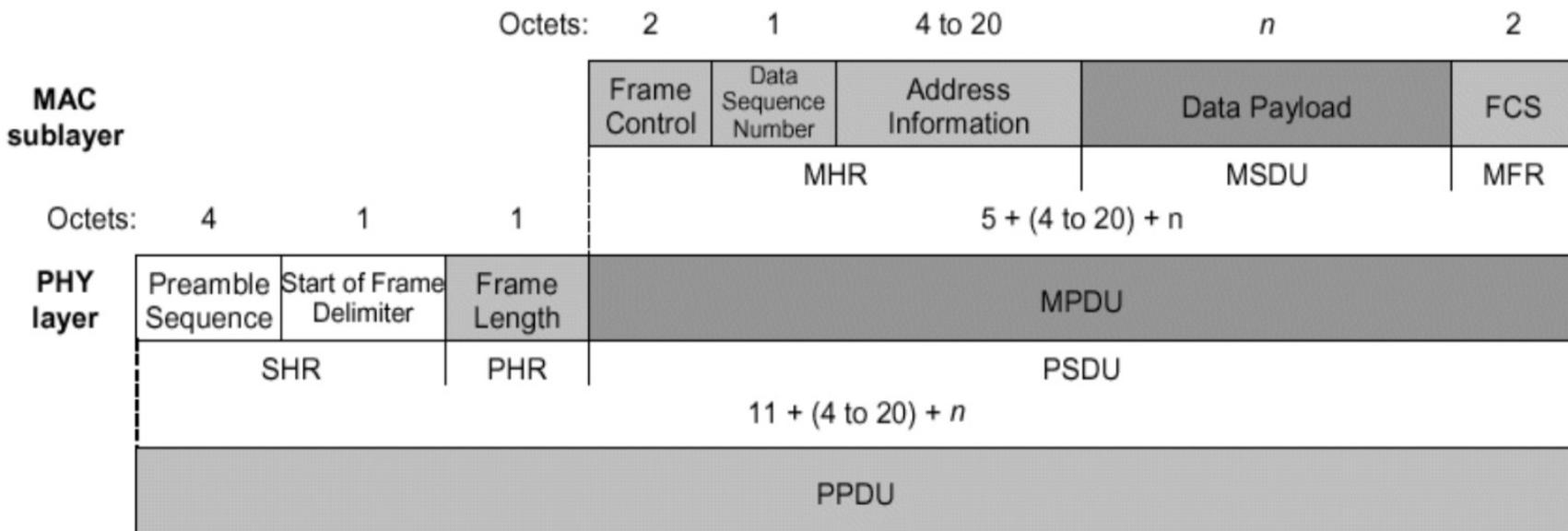
- **CSMA-CA** (*Carrier Sense Multiple Access – Collision Avoidance*)
- **Slotted Mode** – Divisão em superframes, delimitadas com Beacons, com três períodos:
 - **CAP** (*Contention Access Period*) – canal será acedido com CSMA-CA
 - **CFP** (*Contention Free Period*) – canal alocado em *intervalos de tempo* fixos por nó
 - **IP** (*Inactive Period*) – Período de inatividade em que os nós se podem desligar





IEEE 802.15.4

- **Formato das tramas: MAC e PHY**



- **Tramas são pequenas:** dispositivos enviam poucos dados de cada vez
- **Tamanho total da trama MAC:** **0 - 127 bytes**
- **Carga útil sem segurança:** **102 byte < n < 118 byte (melhor caso)**
- **Pior caso (com header de segurança):** **máximo de 81 byte !!**



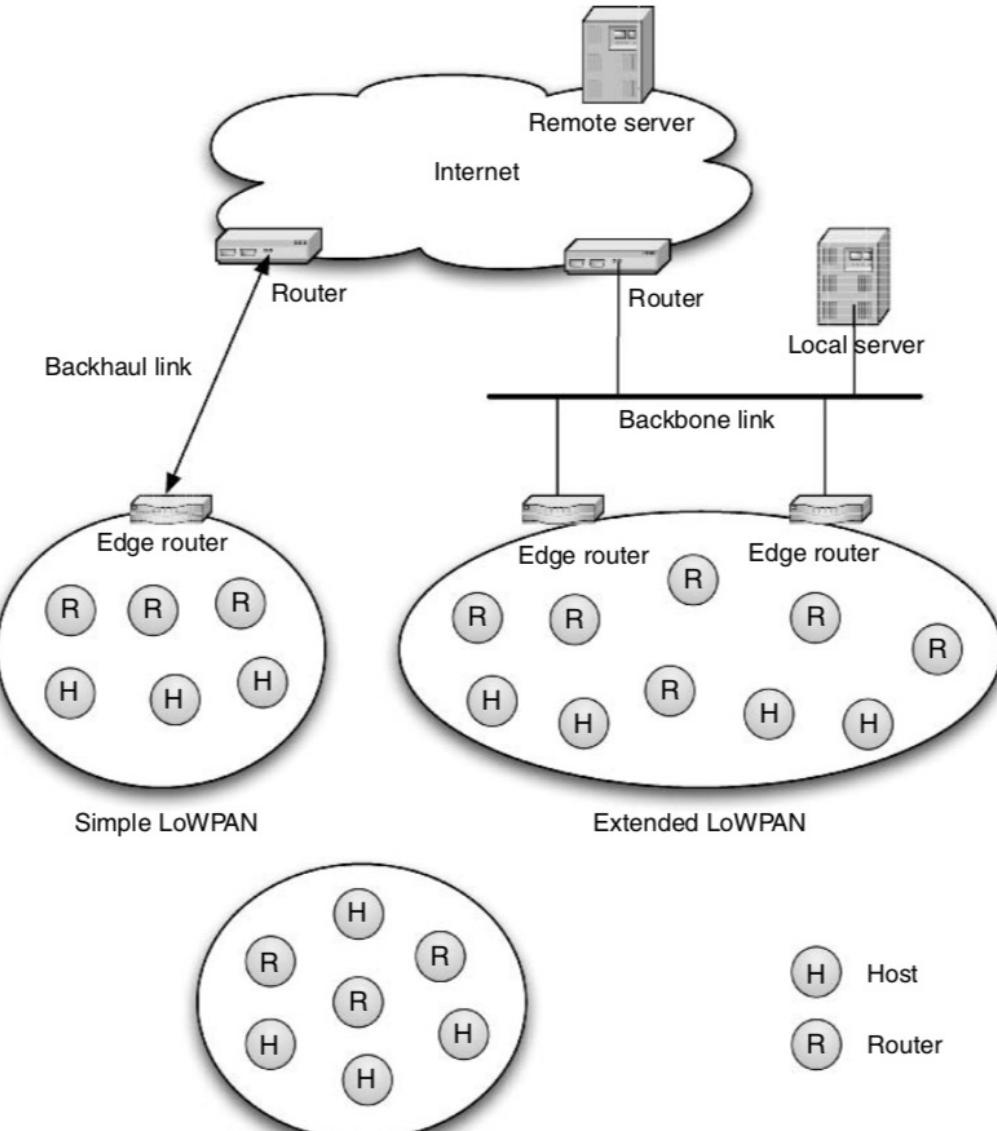
6LoWPAN – Visão geral

● Arquitetura genérica

- Interligar dispositivos em IPv6
- Incluindo sensores

● Tipos:

- LoWPAN Simples
- LoWPAN AdHoc
- LoWPAN Expandida

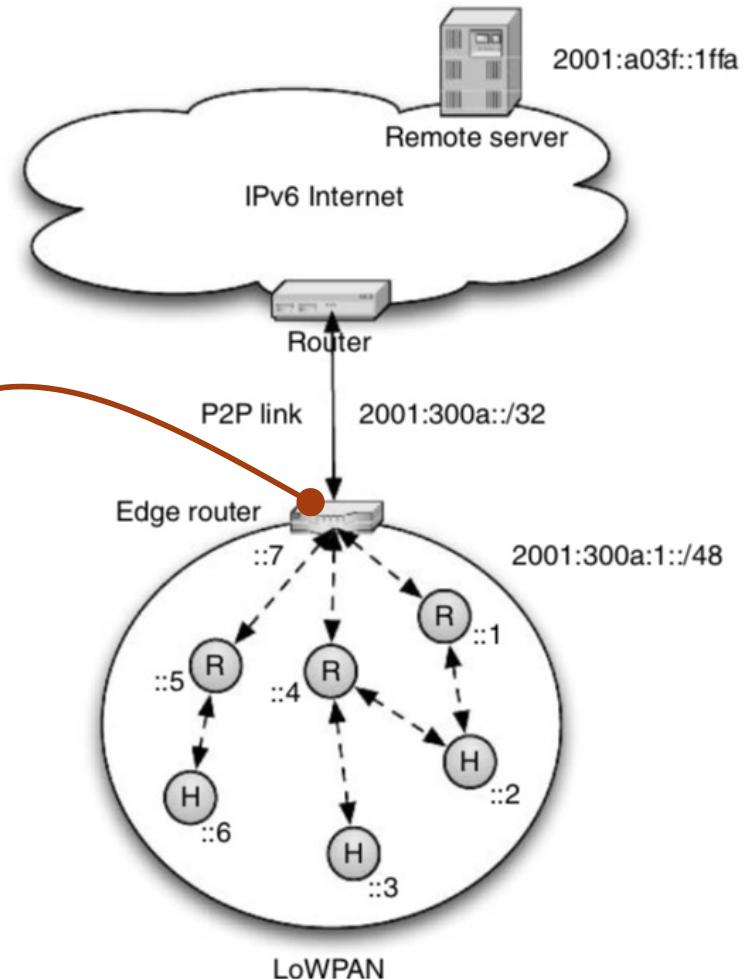
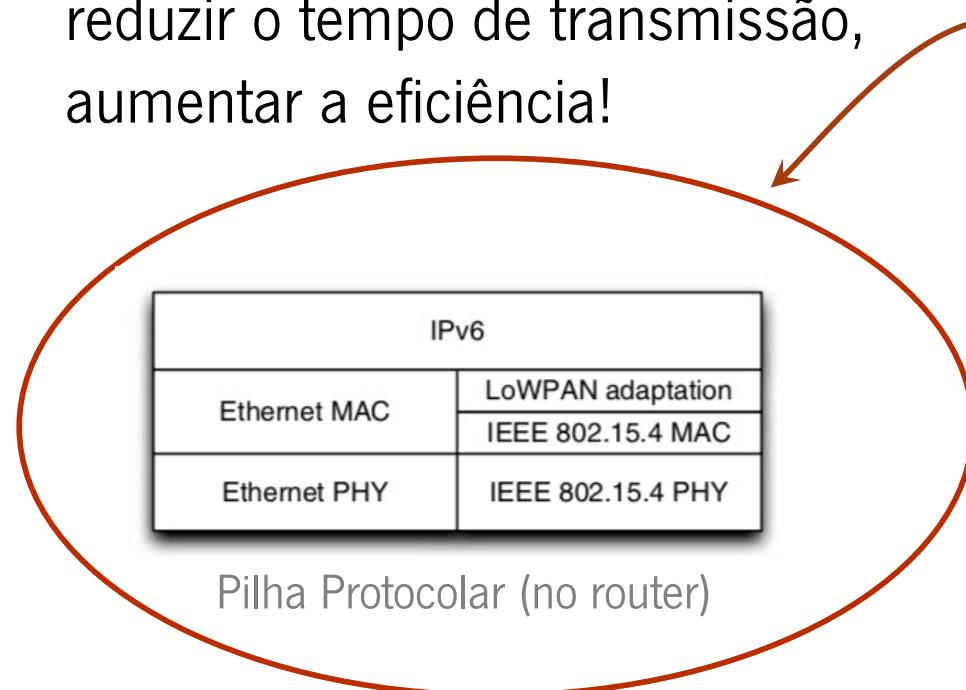


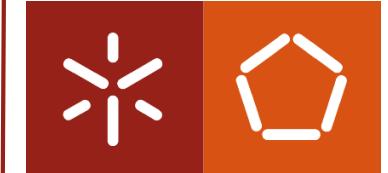


6LoWPAN – Exemplo

● 6LoWPAN: camada de adaptação

- Routers interligam ilhas 6LoWPAN com rede IPv6
- Objetivo é reduzir o *overhead*, reduzir o tempo de transmissão, aumentar a eficiência!





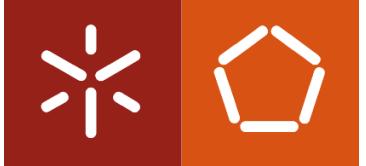
6LoWPAN

● Motivação

- Os dispositivos IoT enviam pacotes muito pequenos (poucos bytes)
- IPv6 quer MTU de 1280 bytes: reduzir o overhead em todas as camadas!
- Há normas, como **IEEE 802.11ah** que procuram reduzir o overhead na camada 2
- Podemos reduzir o **overhead do IPv6?**
 - cabeçalhos de 40 bytes consomem metade da carga útil! Reduzir a 2 ou 3!
- Podemos reduzir o **overhead de transporte (UDP)?** A 1 byte?

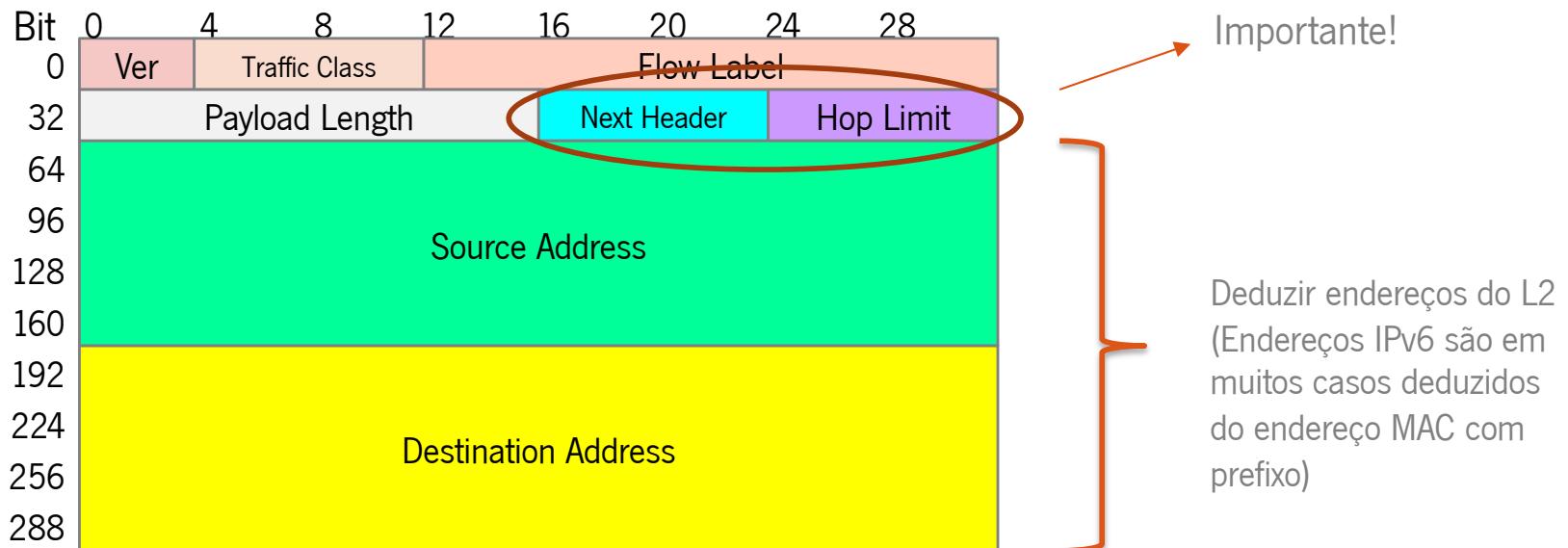
● Revendo o IPV6

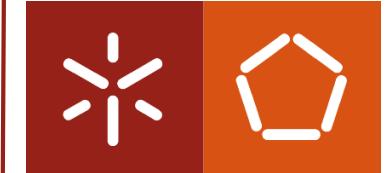
- Aumento do espaço de endereçamento: endereços de 32 bit para **128 bit**
- Autoconfiguração *stateless* e *statefull* (baseada no DHCPv6)
- Cabeçalho de tamanho fixo: 40 bytes (maior parte é endereços)
- Usa uma estratégia de **encadeamento de cabeçalhos opcionais** (NEXT_HEADER)
- Segurança é obrigatória (cabeçalhos opcionais)



● Princípios

- Muita informação pode ser **deduzida do nível 2**: nomeadamente os endereços!
- Tamanho do pacote IP também pode ser deduzido do nível 1 e 2
- Ignorar a Versão, a Classe de Tráfego e a Identificação de Fluxo
- No HOP LOMIT usar apenas alguns valores predefinidos

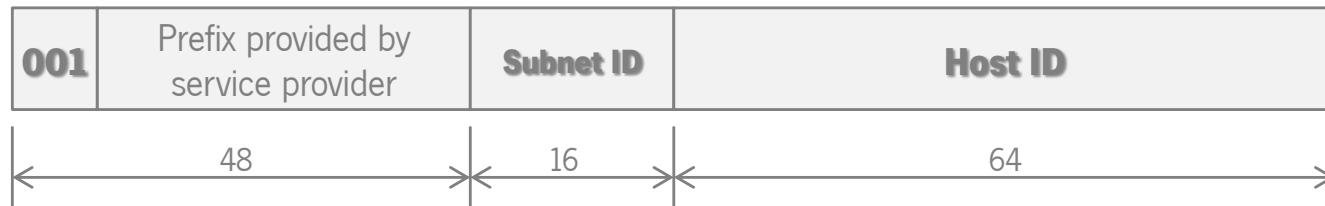




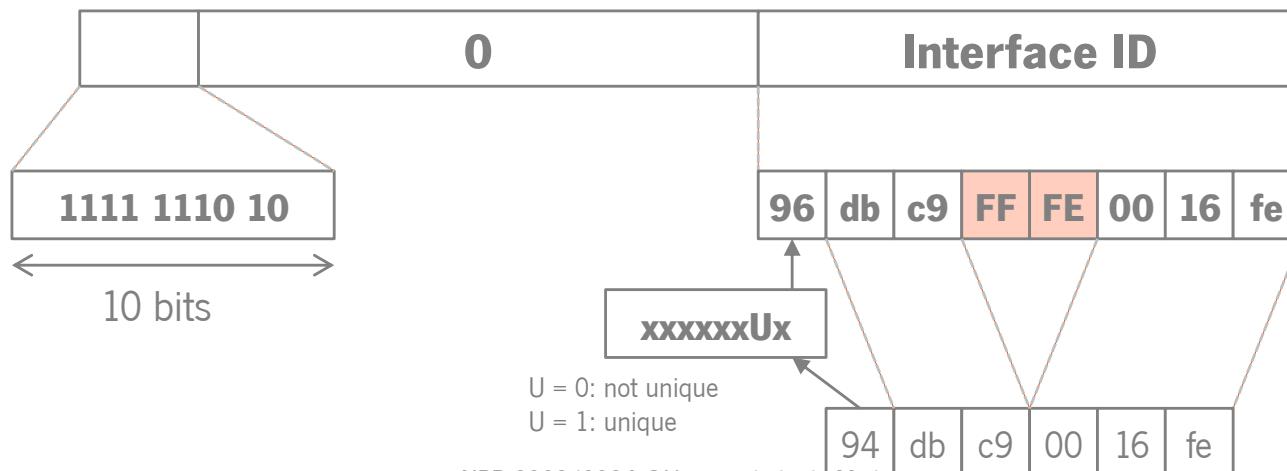
6LoWPAN

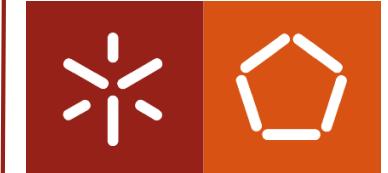
● Endereços IPv6

- Globais: começam com 001 e costumam **incorporar o MAC** no Host ID:



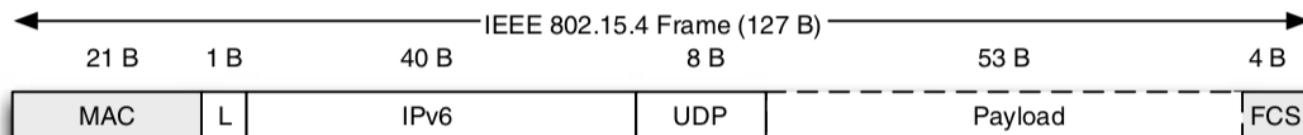
- Locais: começam com FE80::/10 e **também derivados do endereço L2**



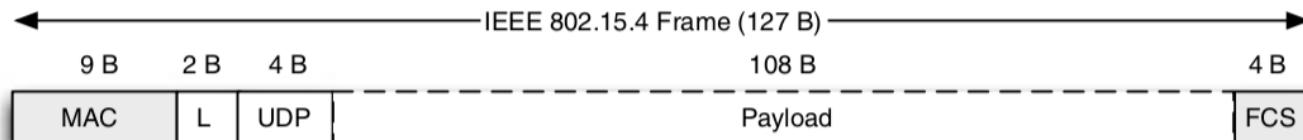


6LoWPAN

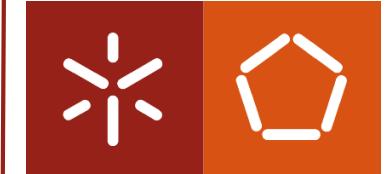
- Camada de adaptação 6LoWPAN disponibiliza três serviços
 - Fragmentação e reagrupamento
 - Compressão do cabeçalho
 - Reenvio nível 2
- Baseado num campo DISPATCH: **L = 1byte** (RFC 6282)
 - Identifica o tipo de cabeçalho que se segue:



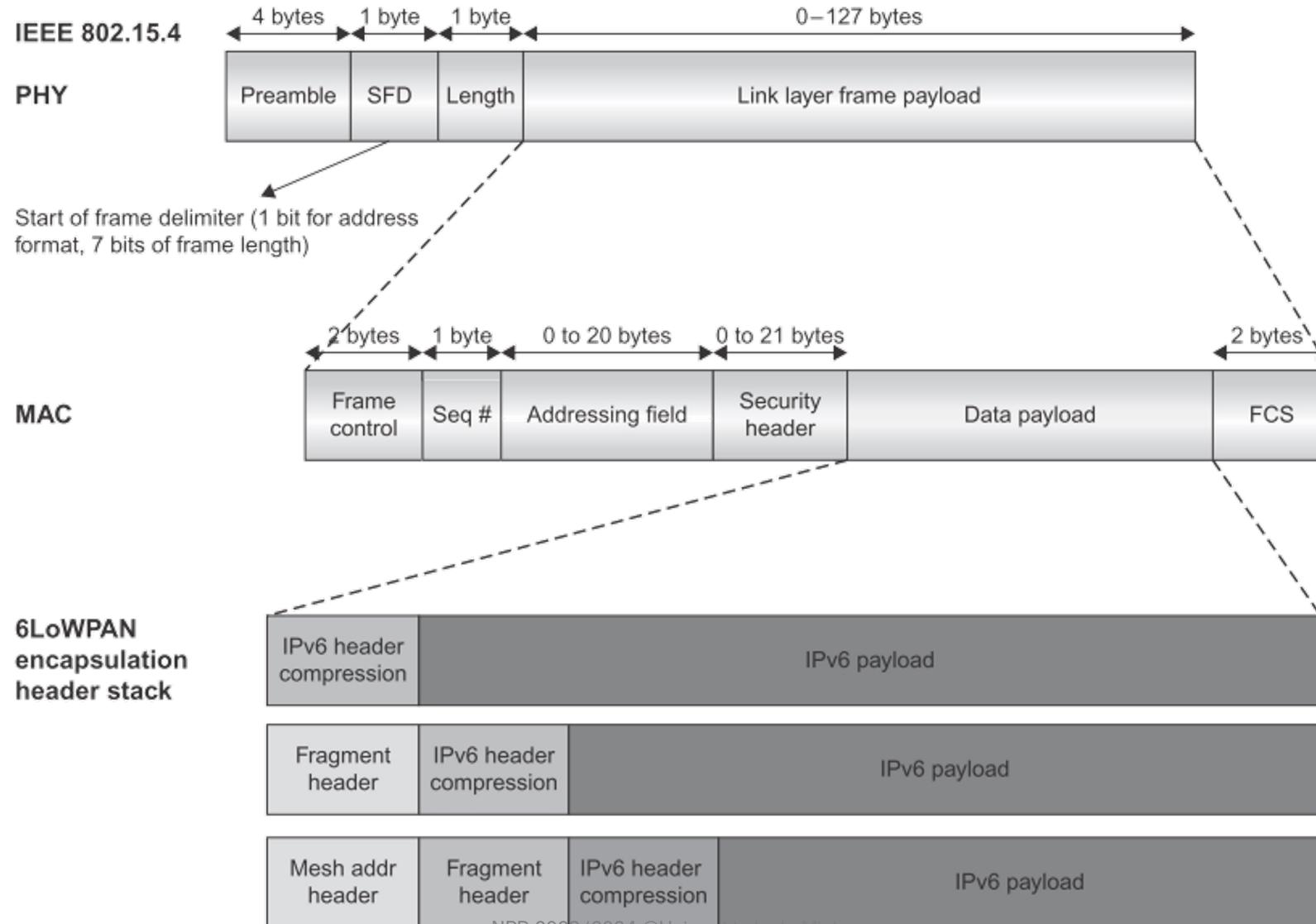
Full UDP/IPv6 (64-bit addressing)



NPR 2023/2024 @Universidade do Minho
Minimal UDP/6LoWPAN (16-bit addressing)



6LowPAN Header Stack

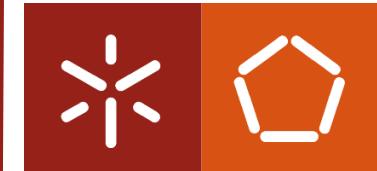




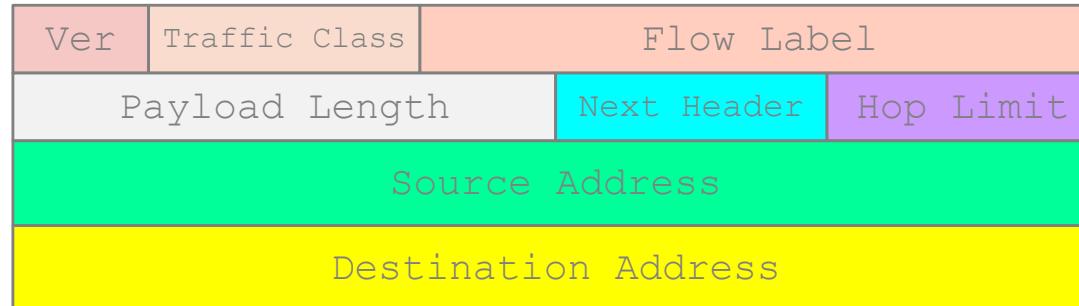
6LoWPAN: Compressão

- Originalmente foram definidos dois métodos: HC1 e HC2 (RFC4944, de 2007)
 - HC1 para compressão do cabeçalho IPv6
 - HC2 para compressão do cabeçalho UDP
 - Otimização para endereços de Link-Local
 - Endereços globais tem de ser usados de forma não comprimida!
- Novo método IPHC (RFC 6282, de 2011)
 - É o que deve ser usado... Independentemente
 - Aplica os mesmos princípios de compressão, mas prevê contextos

6LoWPAN: Compressão do IPv6

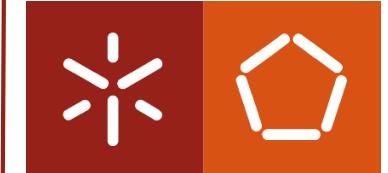


- Otimizado para endereços de link local



- Compressão baseia-se nas seguintes observações:

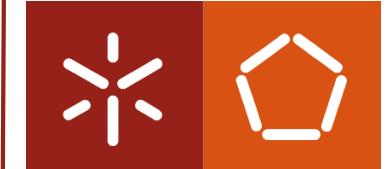
- Versão **é sempre** 6 e pode ser omitido
- Endereço IPv6 do interface é normalmente **deduzido** do MAC e pode ser compactado ou omitido
- Tamanho é **deduzido** do tamanho da *frame* e pode ser omitido
- Classe de Tráfego e Etiqueta de Fluxo são quase **sempre 0**
- O *Next Header* é **quase sempre** TCP, UDP, ou ICMP



6LoWPAN: Compressão do UDP

● Compressão do cabeçalho UDP

- Retirar o *tamanho* que pode ser **deduzido** do tamanho da frame
- Retirar o *checksum*, que muitas vezes já não é usado
- Encurtar o número de bits para endereçar portas apenas a 4 bit
 - Dispositivos são de pouca capacidade...
 - Assumimos que todas as portas a usar estão no intervalo entre 61616 e 61631 ($2^4 = 16$ valores)



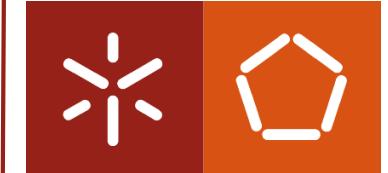
6LoWPAN – Compressão IPHC

- Após DISPATCH vem compressão do cabeçalho LOWPAN_IPHC
- Valores definidos pela IANA para o campo DISPATCH

Bytes: 1 2 or 3



Bit Pattern	Header Type	Reference
00 xxxx xx	NALP - Not a LoWPAN frame	[RFC4944]
01 000000	Reserved as a replacement value for ESC	[RFC6282]
01 000001	IPv6 - uncompressed IPv6 Addresses	[RFC4944]
01 000010	LOWPAN_HC1 - LOWPAN_HC1 compressed IPv6	[RFC4944]
01 000011	LOWPAN_DFF	[RFC6971]
01 000100 through 01 001111	reserved for future use	
01 010000	LOWPAN_BC0 - LOWPAN_BC0 broadcast	[RFC4944]
01 010001 through 01 011111	reserved for future use	
01 1xxxxx	LOWPAN_IPHC	[RFC6282]
10 xxxx xx	MESH - Mesh header	[RFC4944]
11 000xxx	FRAG1 -- Fragmentation Header (first)	[RFC4944]
11 001000 through 11 011111	reserved for future use	
11 100xxx	FRAGN -- Fragmentation Header (subsequent)	[RFC4944]
11 101000 through 11 111111	reserved for future use	



6LoWPAN – Compressão IPHC

● Cabeçalho IPHC (2 ou 3 bytes)

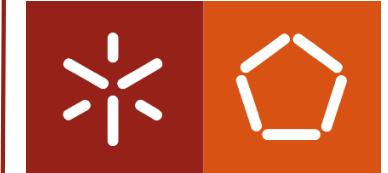
Octet-0								Octet-1								Octet-3 (opcional)							
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
0	1	1	TF	NH	HLIM	CID	SAC	SAM	M	DAC	DAM					SCI		DCI					
# bit								Description															
011XXXXX								LOWPAN_IPHC Dispatch code															
TF								Traffic class, Flow label															
NH								Next Header															
HLIM								Hop LIMit															
CID								Context Identifier															
SAC								Source Address Compression															
SAM								Source Address Mode															
M								Multicast Compression															
DAC								Destination Address Compression															
DAM								Destination Address Mode															

● Terceiro byte só existe se CID = 1

- Id de Informação de Contexto 4 bits (*Source*) +4 bits (*Destination*)

NPR2023/2024 @Universidade do Minho

28



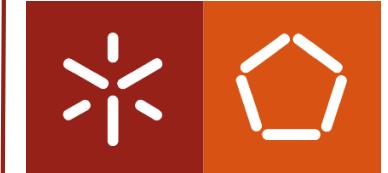
6LoWPAN – Compressão IPHC

● Compressão de endereços:

SAC DAC	SAM DAM	Descrição
0	00	Não é usada compressão do endereço
0	01	Só 64 bit (prefixo). Omitidos os últimos 64 bit que se deduzem do Layer 2.
0	10	Só 16 bit (prefixo=0000:00ff.fe00:XXXX). Restantes deduzem-se do Layer 2.
0	11	Todo o endereço omitido. Deve ser deduzido totalmente do Layer 2
1	00	Endereço é o “::” (UNDEFINED)
1	01	Só 64 bit. Restantes derivam-se da informação de contexto SCI/DCI
1	10	Só 16 bit (prefixo=0000:00ff.fe00:XXXX). Restantes do contexto SCI/DCI.
1	11	Todo o endereço omitido. Deve ser deduzido totalmente do contexto SCI/DCI

● Compressão TF e Flow Label:

- 00: ECN + DSCP + 4-bit Pad + Flow Label (4 bytes)
- 01: ECN + 2-bit Pad + Flow Label (3 bytes), DSCP é omitido.
- 10: ECN + DSCP (1 byte), Flow Label é omitido.
- 11: Traffic Class e Flow Label ambos omitidos .



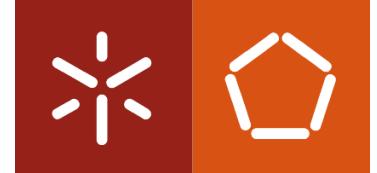
6LoWPAN – Compressão IPHC

● Compressão IPHC máxima → 2 byte (em vez de 40)

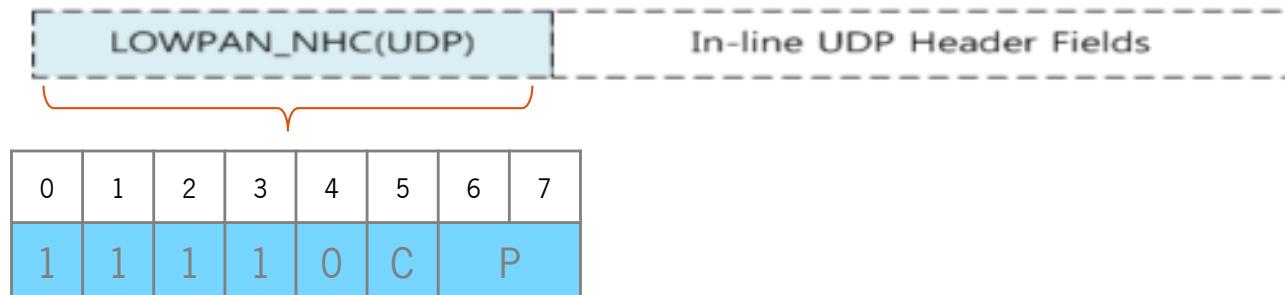
	Octet 0								Octet 1							
	DSP			TF		NH	HLIM		CID	SAC	SAM		M	DAC	DAM	
2 Byte	0	1	1	1	1	1	0	1	0	0	1	1	0	0	1	1

- [DSP] 001 – LOWPAN_IPHC IPv6 Header encoding
- [TF] 11 – IPv6 *Traffic class* e *Flow Label* omitidos e iguais a 0
- [NH] 1 – Segue-se um *Next Header* também compactado (ex: UDP)
- [HLIM] 01 – *Hop Limit* é 1 (apenas um salto)
- [CID] 0 – Não existe informação adicional de contexto
- [SAC] 0 – *Source Address* utiliza compressão *stateless*
- [SAM] 11 – IPv6 de origem é derivado diretamente do *link address*
- [M] 0 – Não é Multicast
- [DAC] 0 – *Destination Address* utiliza compressão *stateless*
- [DAM] 11 – IPv6 de destino é derivado diretamente do *link address*

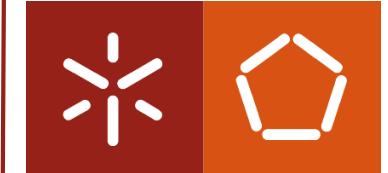
6LoWPAN – Compressão UDP



- **Cabeçalho UDP LOWPAN_NHC (2 ou 4 bytes)**



- [C] – Checksum: 0 – omitido; 1 – incluído em linha;
- [P] – Bits que definem a compressão das portas UDP
 - 00 – Portas Origem e Destino incluídas em linha: $Dst = 16$ bit; $Src = 16$ bit;
 - 01 – Omitidos os primeiros 8 bit da porta de destino: $Dst = 8$ bit; $Src = 16$ bit;
 - 10 – Omitidos os primeiros 8 bit da porta de origem: $Dst = 16$ bit; $Src = 8$ bit;
 - 11 – Assume-se que primeiros 12 bit das portas são 0xf0b: $Dst = 4$ bit; $Src = 4$ bit;



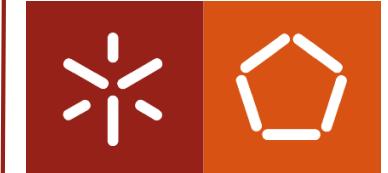
6LoWPAN – Compressão UDP

- Exemplo de compressão UDP máxima (com ou sem Checksum)

	Octet 0	Octet 1	Octet 2	Octet 3
4 byte		SRC Port		DST Port
8 byte	UDP Length		UDP Checksum	

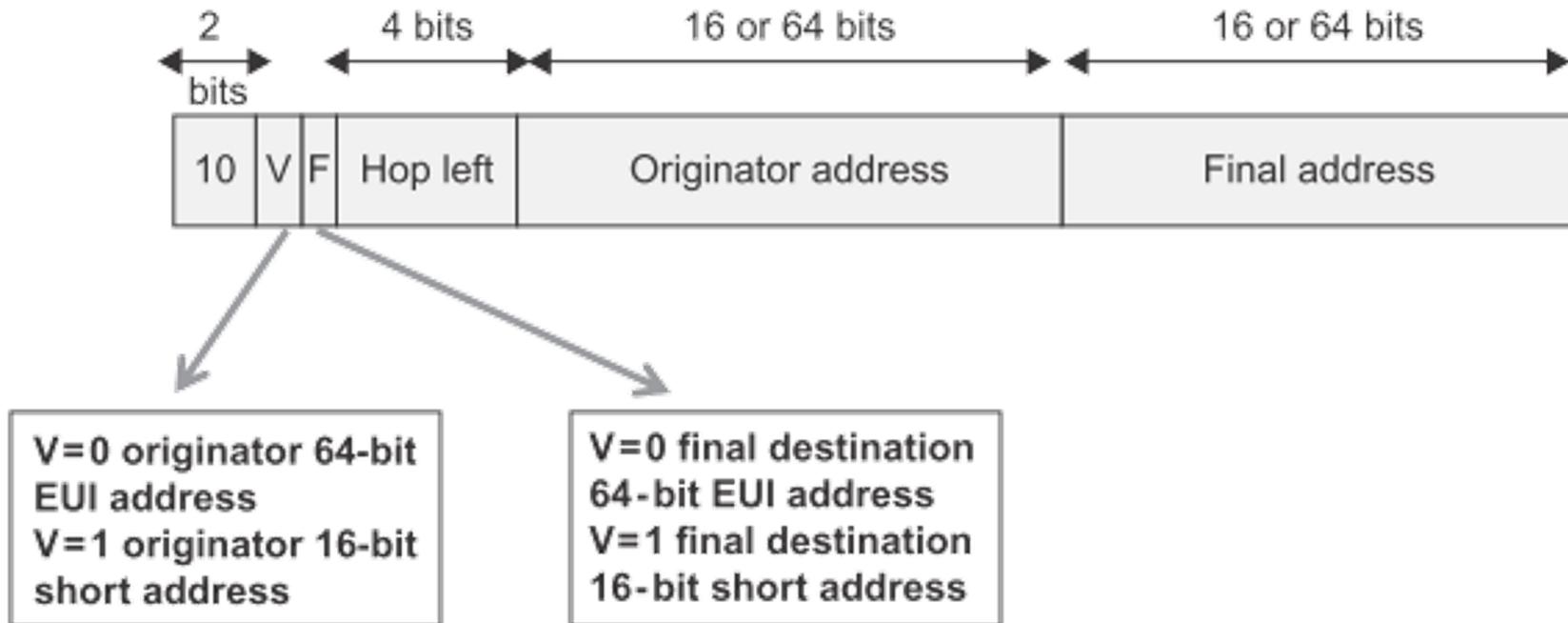


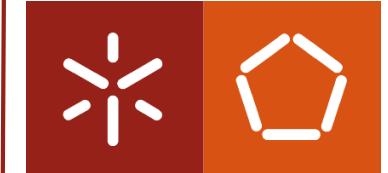
	Octet								Octet	
2 byte	1	1	1	1	0	0	1	1	Last 4-bit of SRC port	Last 4-bit of DST port
4 byte	UDP Checksum									



6LoWPAN: Cabeçalho Mesh

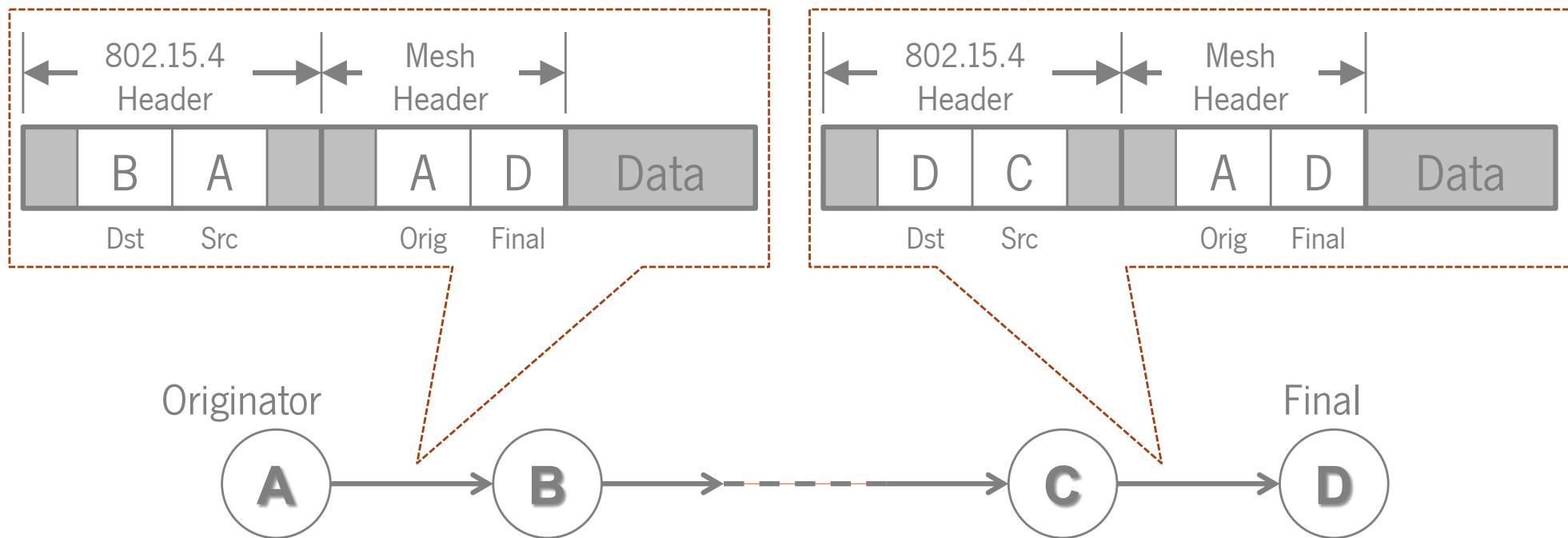
- Usar para routing MESH controlado pela camada de adaptação 6LoWPAN (*mesh-under routing*)
 - Só para nós do tipo FFD (*Full Function Devices*)

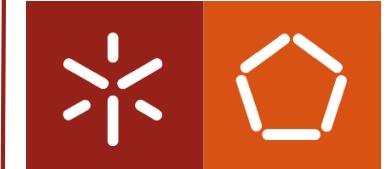




6LoWPAN: Cabeçalho Mesh

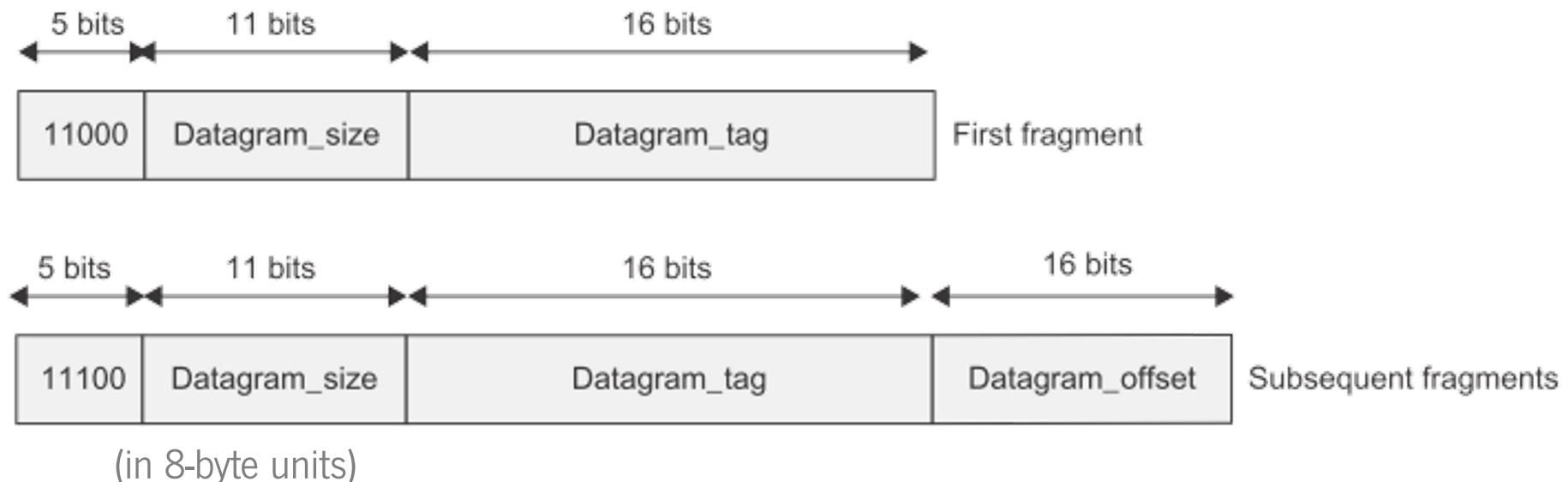
- **Hop left :** é decrementado uma unidade a cada salto
 - Frame é descartada quando valor chega a zero
- Os campos de endereço não mudam ao longo do percurso

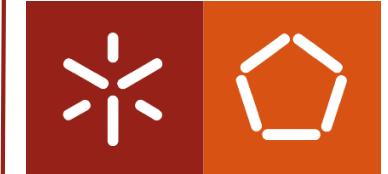




6LoWPAN: Cabeçalho de Fragmentação

- Fragmentação ocorre sempre que o *payload* do pacote IPV6 excede o *payload* do IEEE 802.15.4



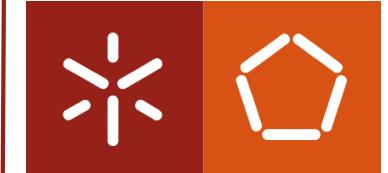


CoAP – Constrained Application Protocol

- **CoAP**

- Um standard aberto definido pelo IETF ([RFC 7252](#))
- Cabeçalho compacto de apenas **4 bytes!**
- Tipicamente para ser transportado por **UDP ou DTLS** (UDP com segurança equivalente ao TLS do TCP)
- Mas também pode ser transportado sobre TCP ou mesmo SMS!
- Integra nativamente a **descoberta de recursos/serviços**
- Integra nativamente a **subscrição assíncrona**

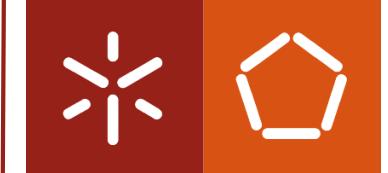




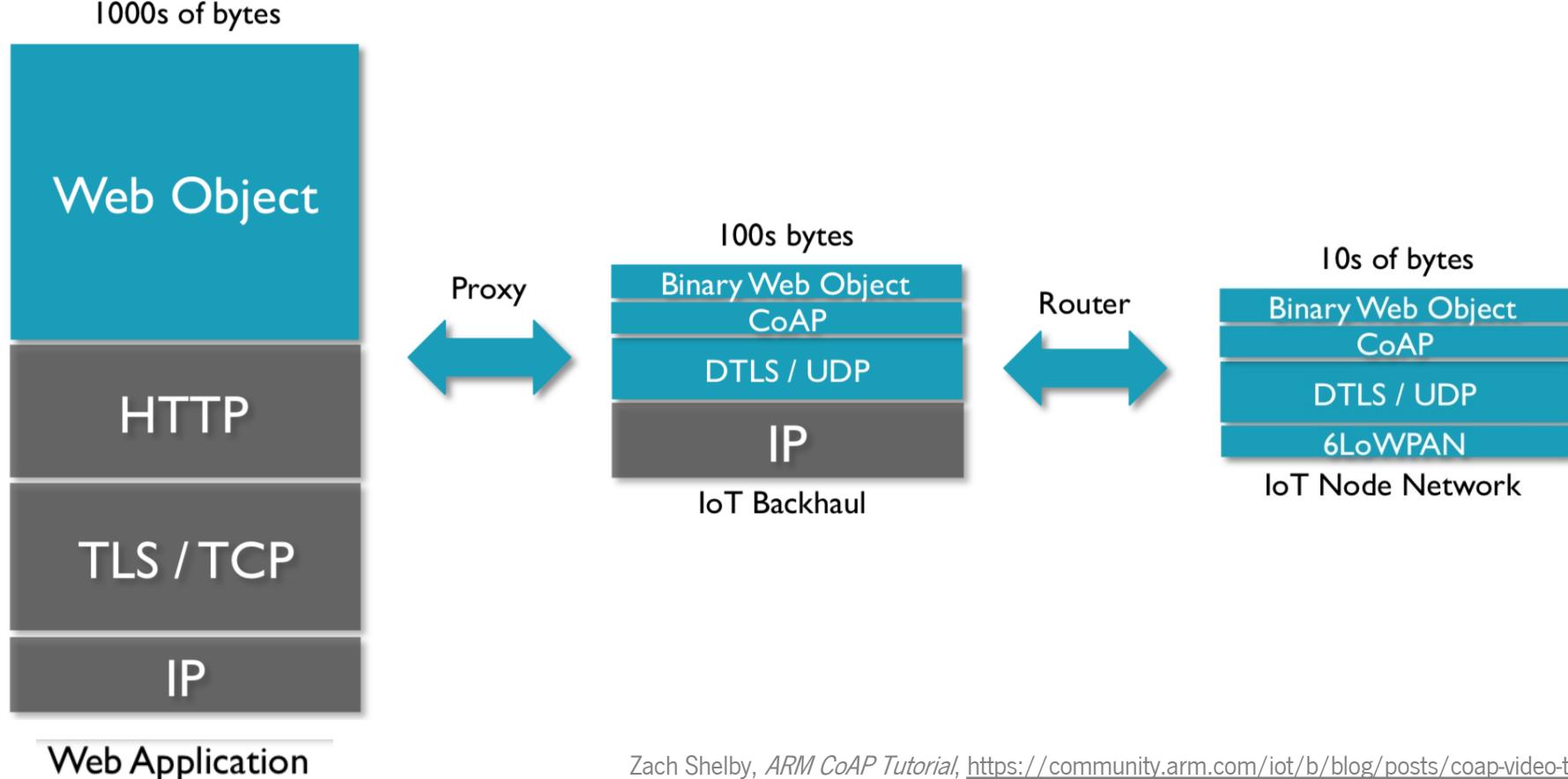
CoAP – Constrained Application Protocol

- **Características:**

- Um **protocolo Web** capaz de cumprir os requisitos de comunicação **M2M** (*Machine-to-Machine*)
- Mapeamento em **UDP** (*unicast e multicast*) opcionalmente confiável
- Troca **assíncrona** de mensagens
- **Baixa sobrecarga** nos cabeçalhos e baixa complexidade de *parsing*
- Suporte para URI e para Content-Type
- *Proxy* simplificado (muito importante)
- *Caching* simplificado (muito importante)
- Que possa ser mapeado no protocolo *stateless* HTTP usado na Internet
- Comunicação segura com DTLS (RFC 6347)



CoAP – Constrained Application Protocol



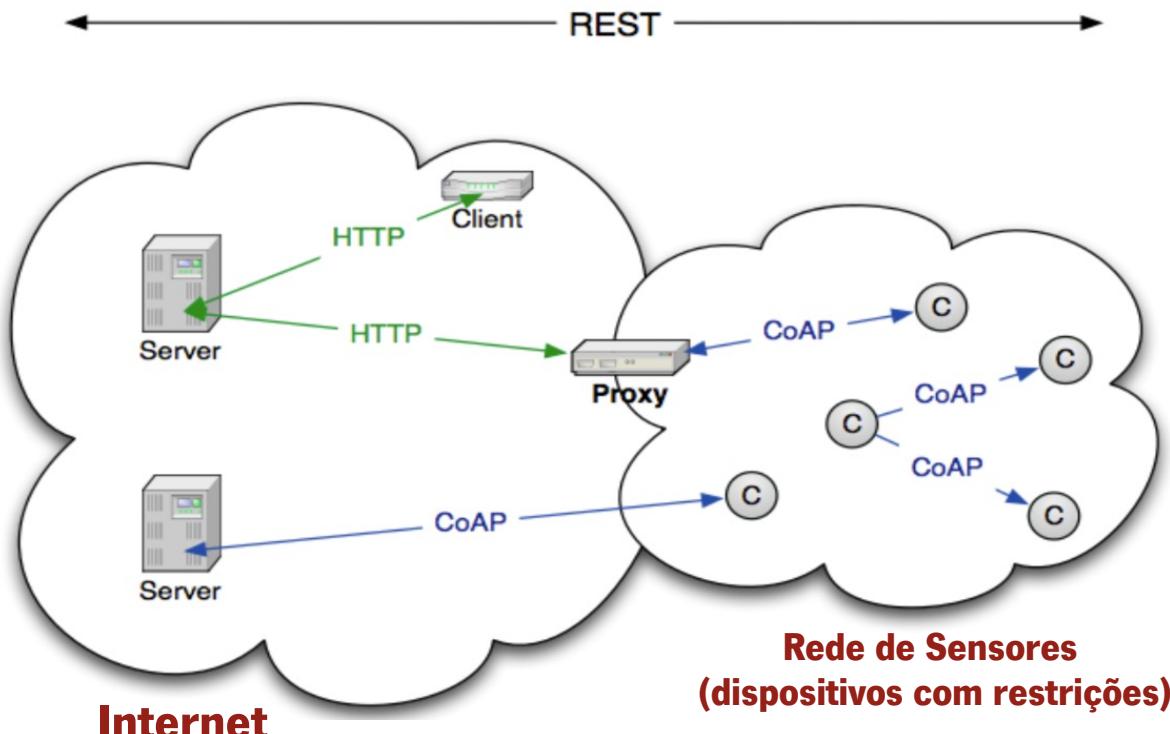
- As aplicações Web podem interrogar dispositivos IoT através de um proxy
- Dum lado milhares de bytes... do outro dezenas de bytes...

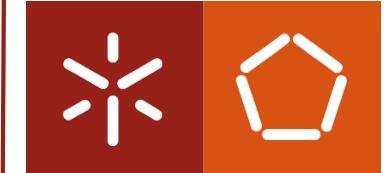


CoAP – Constrained Application Protocol

- **Objetivo principal: Compatibilidade REST**

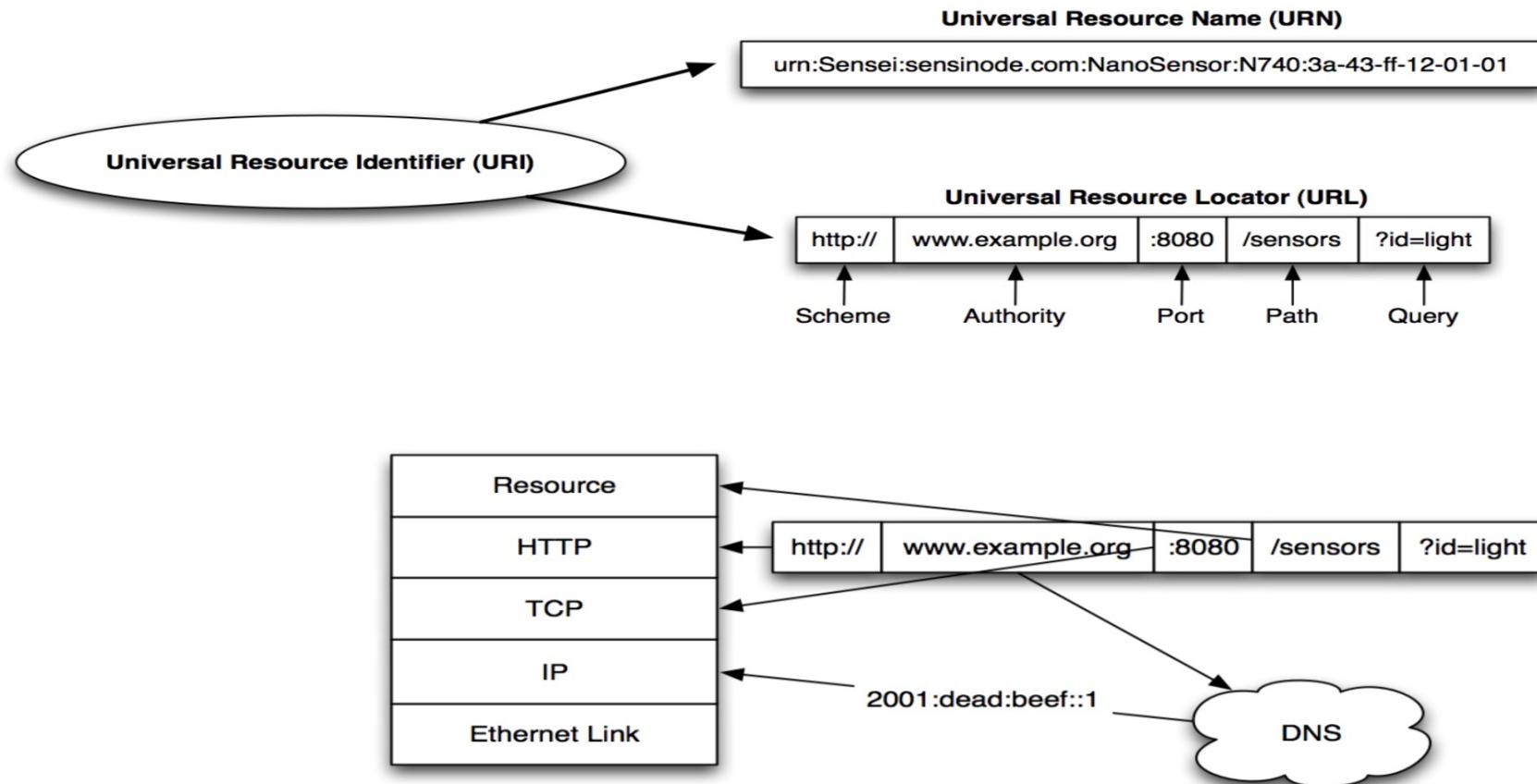
- Concretizar uma arquitetura *Representational State Transfer* [REST] que permita integrar dispositivos limitados e com restrições energéticas
- Não pretende ser um HTTP compactado! Mas sim um protocolo Web especializado!





CoAP – Constrained Application Protocol

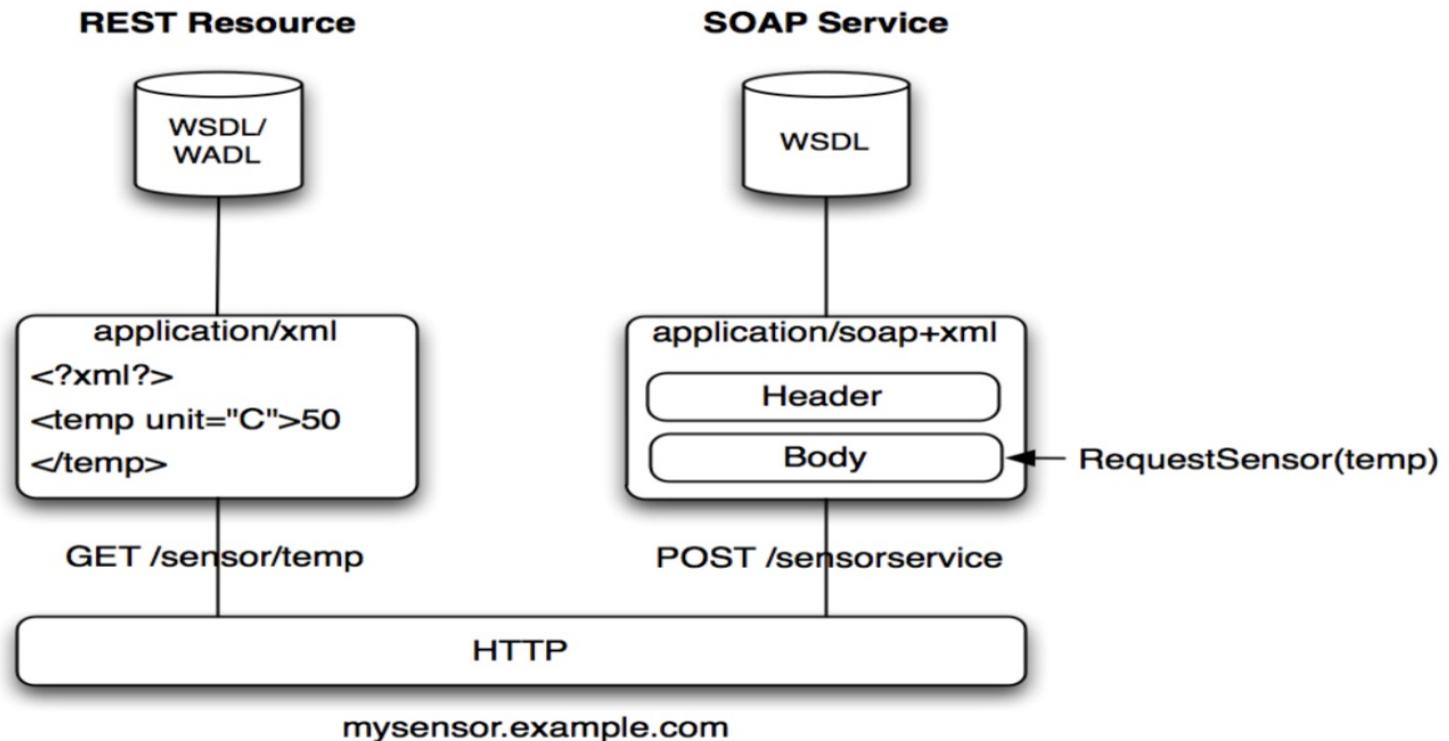
- Conceitos chave do Web: nomeação (URI) e tipos (MIME Content-Type)





CoAP – Constrained Application Protocol

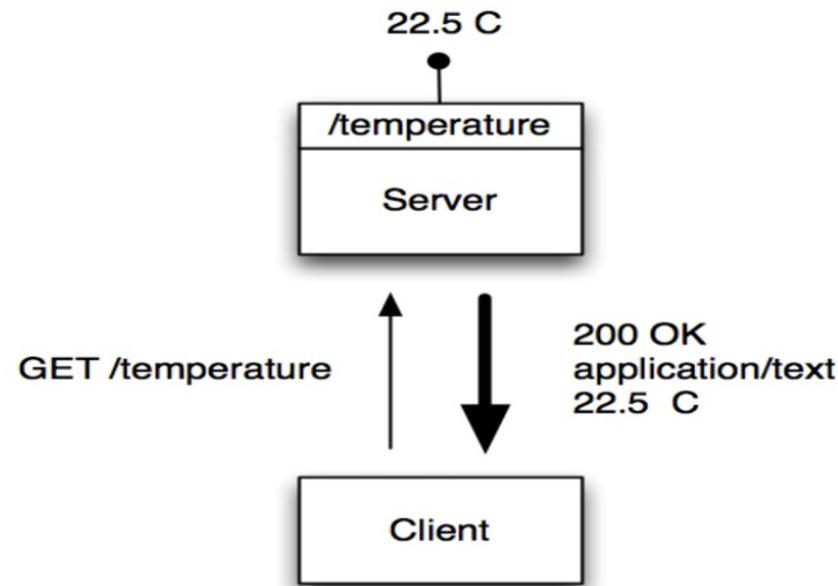
- Paradigmas Web: Web Services REST e Web Services SOAP

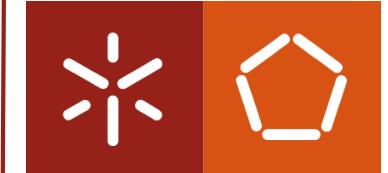




CoAP – Constrained Application Protocol

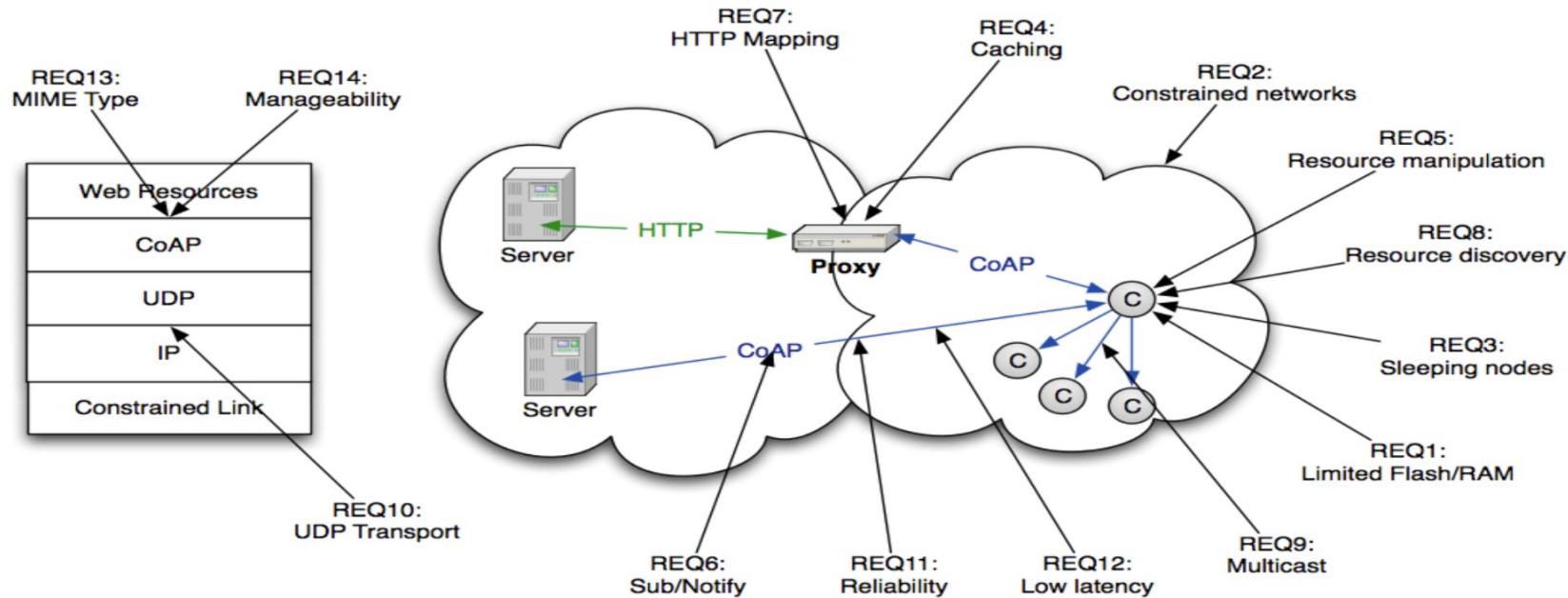
- Adotar o paradigma REST que é mais adequado... para **RECURSOS**





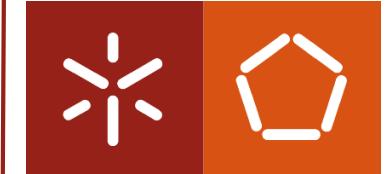
CoAP – Constrained Application Protocol

● Requisitos do CoAP



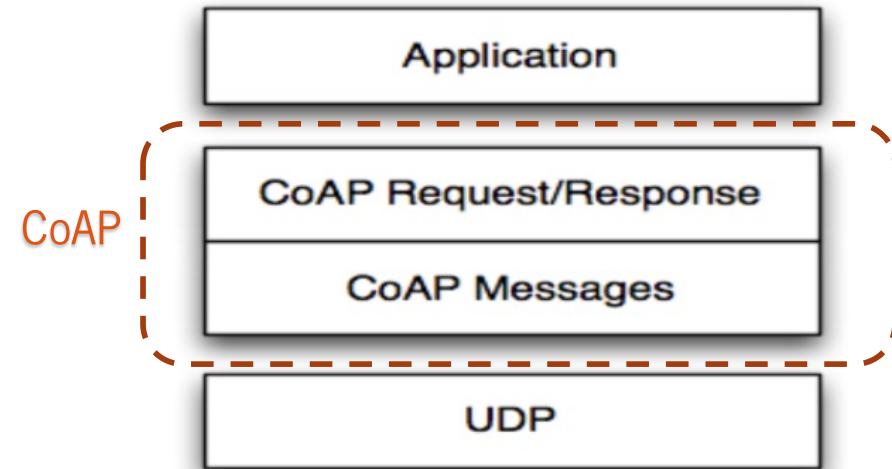
Zach Shelby, *ARM CoAP Tutorial*, <https://community.arm.com/iot/b/blog/posts/coap-video-tutorial>

- (1) dispositivos limitados em memória RAM; (2) acessíveis em redes com restrições; (3) nós que adormecem; (4) *Caching*; (5) manipulação de recursos; (6) subscrições e notificações; (7) mapeamento HTTP; (8) descoberta de recursos; (9) Multicast; (10) Sobre UDP; (11) fiabilidade; (12) baixa latência; (13) MIME Types; (14) suporte para gestão



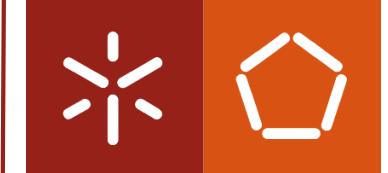
CoAP – Constrained Application Protocol

- Interação cliente/servidor
- Troca de mensagens simples
- Pedido CoAP / Resposta CoAP
- Como se fossem dois protocolos distintos mas é só um
- 4 Tipos de mensagens:
 - **Confirmable (CON)**
 - **Non-Confirmable (NON)**
 - **Acknowledgement (ACK)**
 - **Reset (RST)**



	CON	NON	ACK	RST
Request	X	X	-	-
Response	X	X	X	-
Empty	*	-	X	X

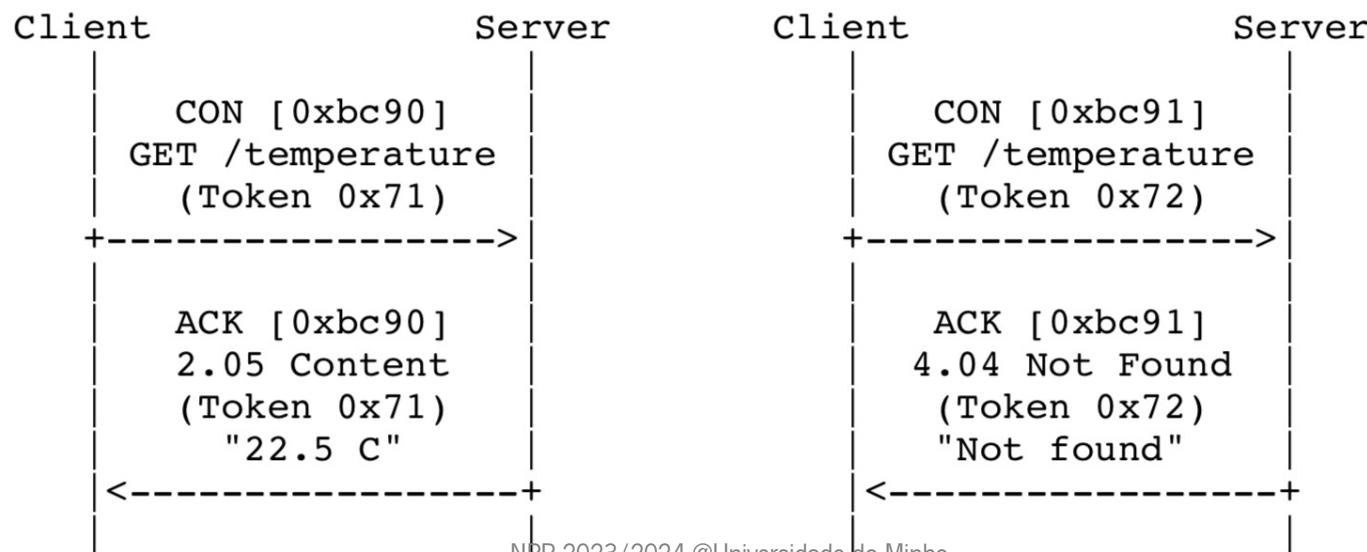
Utilização das mensagens

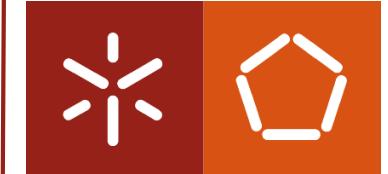


CoAP – Constrained Application Protocol

● Mensagens

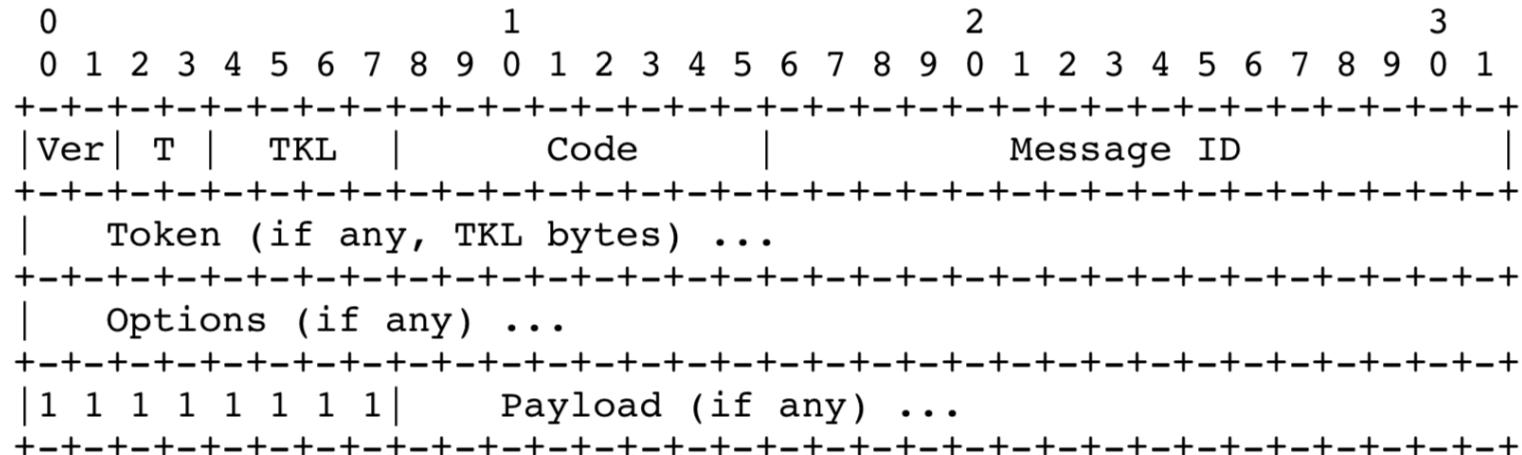
- Relacionamento *pedido/resposta* com **Token ID**: Inteiro de 16 bit
 - Uma resposta usa o mesmo Token ID que o pedido respetivo
- Todas as mensagens possuem um **Message ID**: Inteiro de 16 bit
 - Um dispositivo pode receber a mesma mensagem com o mesmo *message-id* mais que uma vez (duplicação fácil de detetar e neutralizar)



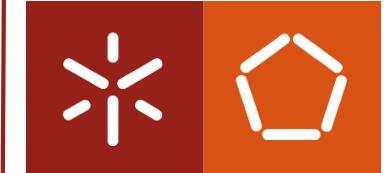


CoAP – Constrained Application Protocol

● Cabeçalho CoAP (4 byte)



- **Ver**: Versão (1)
- **T**: Tipo de mensagem (0) CON; (1) NON; (2) ACK; (3) RST;
- **TKL**: 4 bits com tamanho do token (se existir)
- **Code**: Request Method (1-10) ou Response Code (40-255) conforme o caso; 3 bit código e 5 bit detalhe (normalmente $c.dd$, c 0 a 7 e dd 00 a 32)
- **Message ID**: inteiro sem sinal de 16 bit (identificador de mensagem)



CoAP – Constrained Application Protocol

● Códigos Request / Response

Code	Name	Reference
0.01	GET	[RFC7252]
0.02	POST	[RFC7252]
0.03	PUT	[RFC7252]
0.04	DELETE	[RFC7252]

Assegurar
Compatibilidade REST
(CRUD)

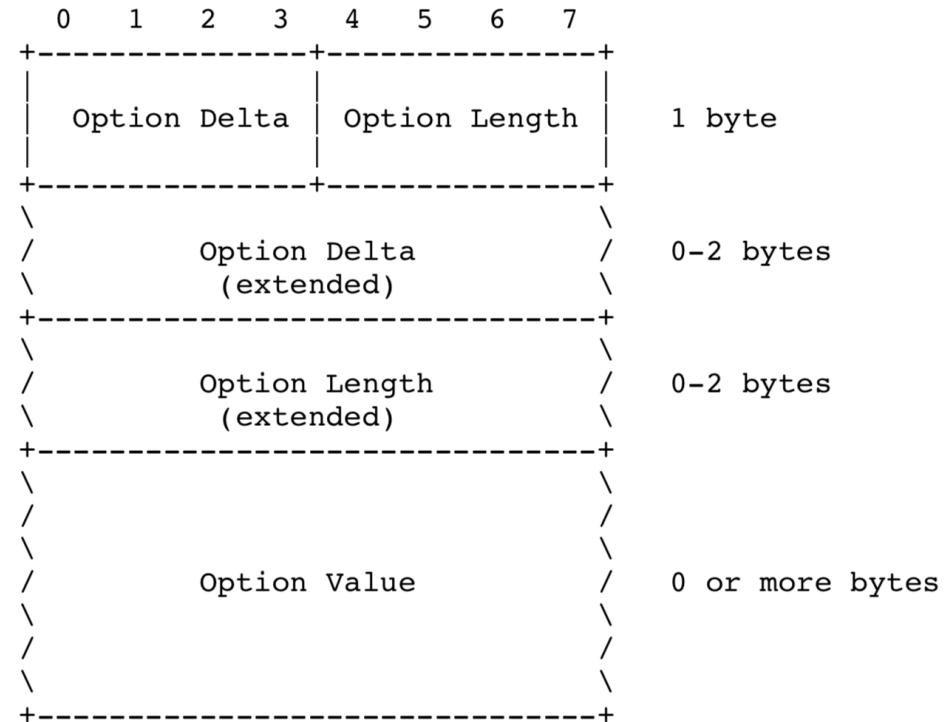
Code	Description	Reference
2.01	Created	[RFC7252]
2.02	Deleted	[RFC7252]
2.03	Valid	[RFC7252]
2.04	Changed	[RFC7252]
2.05	Content	[RFC7252]
4.00	Bad Request	[RFC7252]
4.01	Unauthorized	[RFC7252]
4.02	Bad Option	[RFC7252]
4.03	Forbidden	[RFC7252]
4.04	Not Found	[RFC7252]
4.05	Method Not Allowed	[RFC7252]
4.06	Not Acceptable	[RFC7252]
4.12	Precondition Failed	[RFC7252]
4.13	Request Entity Too Large	[RFC7252]
4.15	Unsupported Content-Format	[RFC7252]
5.00	Internal Server Error	[RFC7252]
5.01	Not Implemented	[RFC7252]
5.02	Bad Gateway	[RFC7252]
5.03	Service Unavailable	[RFC7252]
5.04	Gateway Timeout	[RFC7252]
5.05	Proxying Not Supported	[RFC7252]

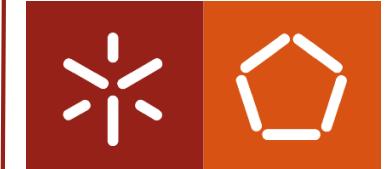


CoAP – Constrained Application Protocol

● Campo de Opções (que podem ou não existir)

- *Option Delta*: diferença para a opção anterior anterior
- *Option Length*: tamanho da opção
- *Option Value*: valor da opção logo a seguir ao tamanho





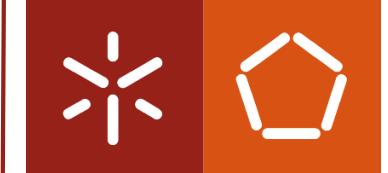
CoAP – Constrained Application Protocol

● Campo de Opções

No.	C	U	N	R	Name	Format	Length	Default
1	x			x	If-Match	opaque	0-8	(none)
3	x	x	-		Uri-Host	string	1-255	(see below)
4				x	ETag	opaque	1-8	(none)
5	x				If-None-Match	empty	0	(none)
7	x	x	-		Uri-Port	uint	0-2	(see below)
8				x	Location-Path	string	0-255	(none)
11	x	x	-	x	Uri-Path	string	0-255	(none)
12					Content-Format	uint	0-2	(none)
14		x	-		Max-Age	uint	0-4	60
15	x	x	-	x	Uri-Query	string	0-255	(none)
17	x				Accept	uint	0-2	(none)
20				x	Location-Query	string	0-255	(none)
35	x	x	-		Proxy-Uri	string	1-1034	(none)
39	x	x	-		Proxy-Scheme	string	1-255	(none)
60			x		Size1	uint	0-4	(none)

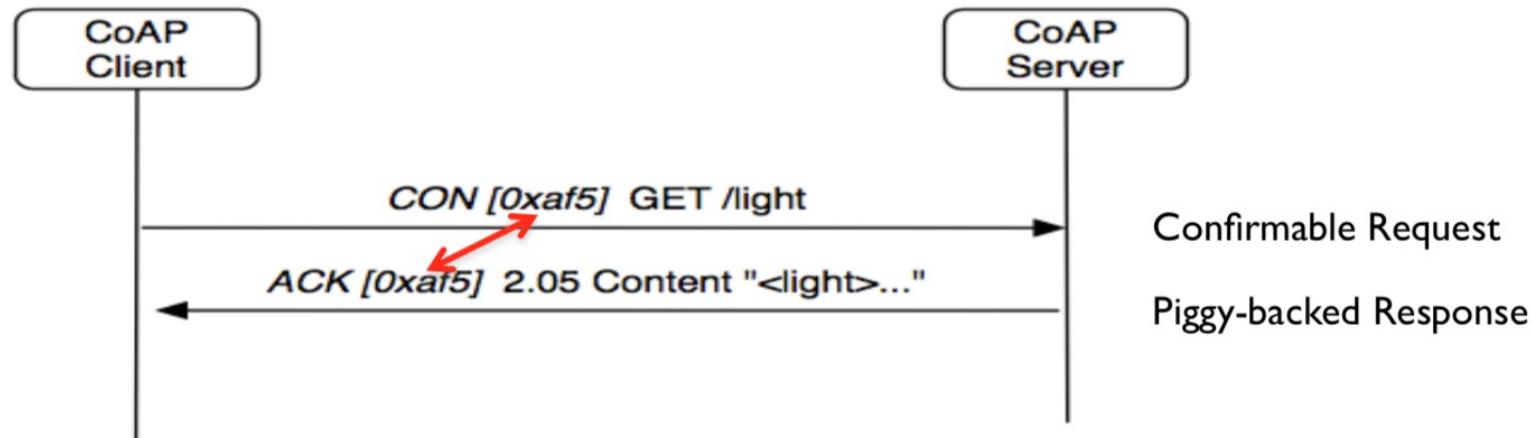
C=Critical, U=Unsafe, N=NoCacheKey, R=Repeatable

(As opções críticas devem ser reconhecidas pelos dispositivos; as repetíveis podem aparecer mais que uma vez; são indicadas as opções que pode não ser seguro fazer o forward num proxy, pois pode não fazer sentido)

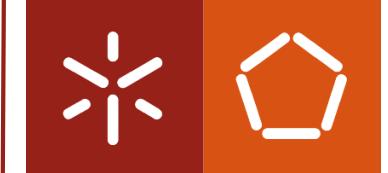


CoAP – Constrained Application Protocol

- Pedido com resposta imediata (*piggy-backed*)

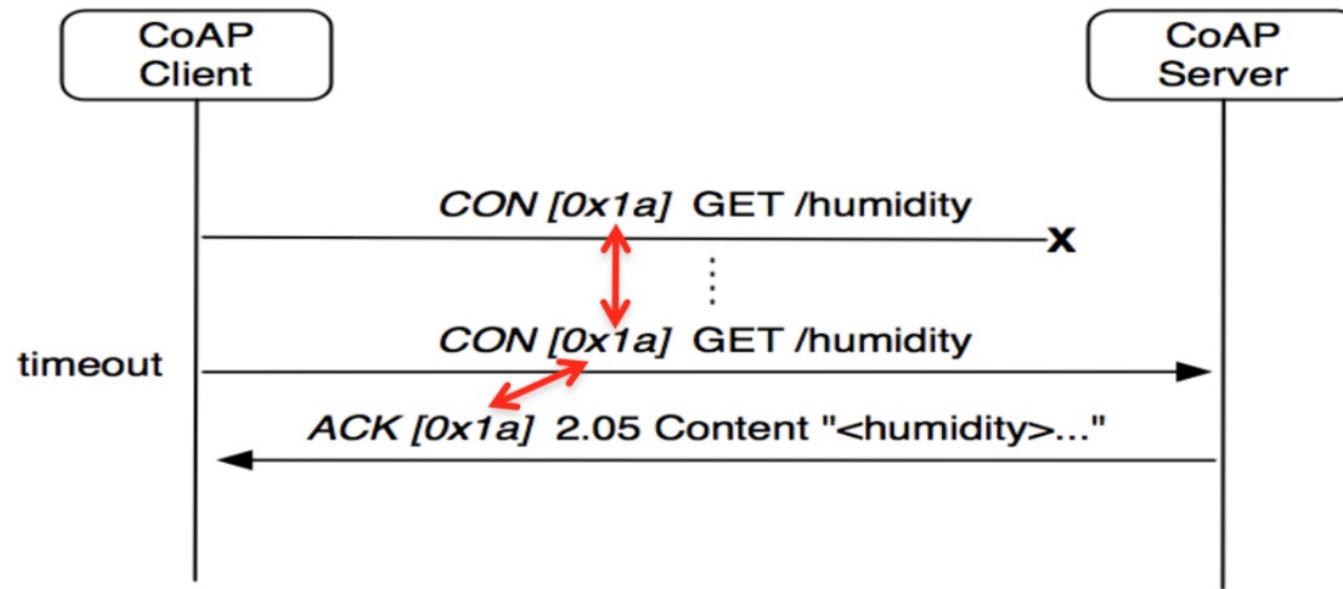


- O pedido é confirmado, exige resposta ACK, mas o ACK trás “às cavalitas” já a resposta pretendida



CoAP – Constrained Application Protocol

- Exemplo em que as mensagens se perdem...

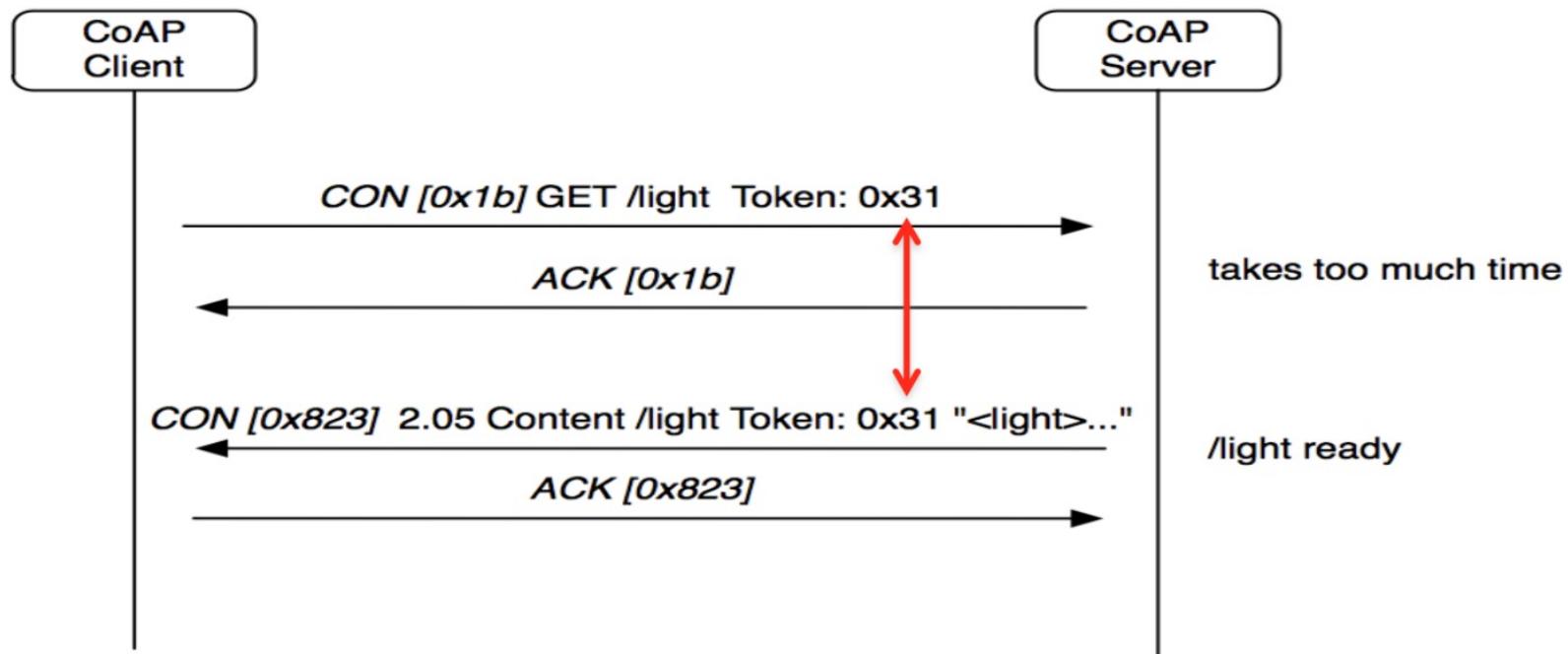


- Após *timeout* o pedido é reenviado com o mesmo Message-ID, permitindo ao destinatário descobrir eventuais pedidos repetidos

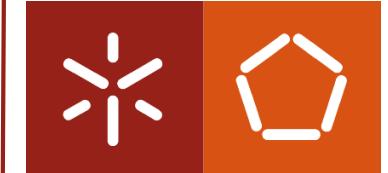


CoAP – Constrained Application Protocol

- Pedido cuja resposta vem mais tarde, quando disponível...

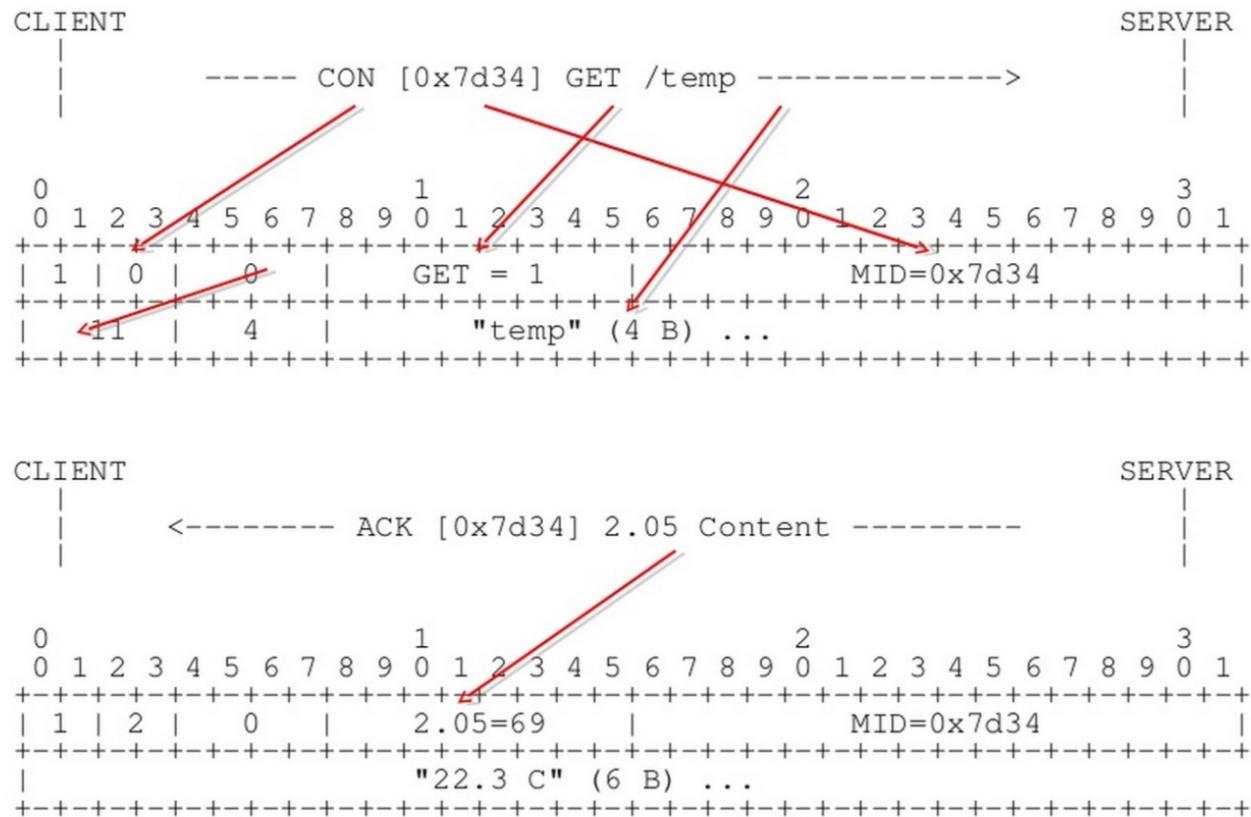


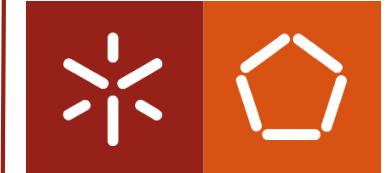
- Como o pedido é confirmado, tem logo uma resposta ACK, depois é o servidor que envida um "pedido" confirmado ao cliente para lhe dar o resultado



CoAP – Constrained Application Protocol

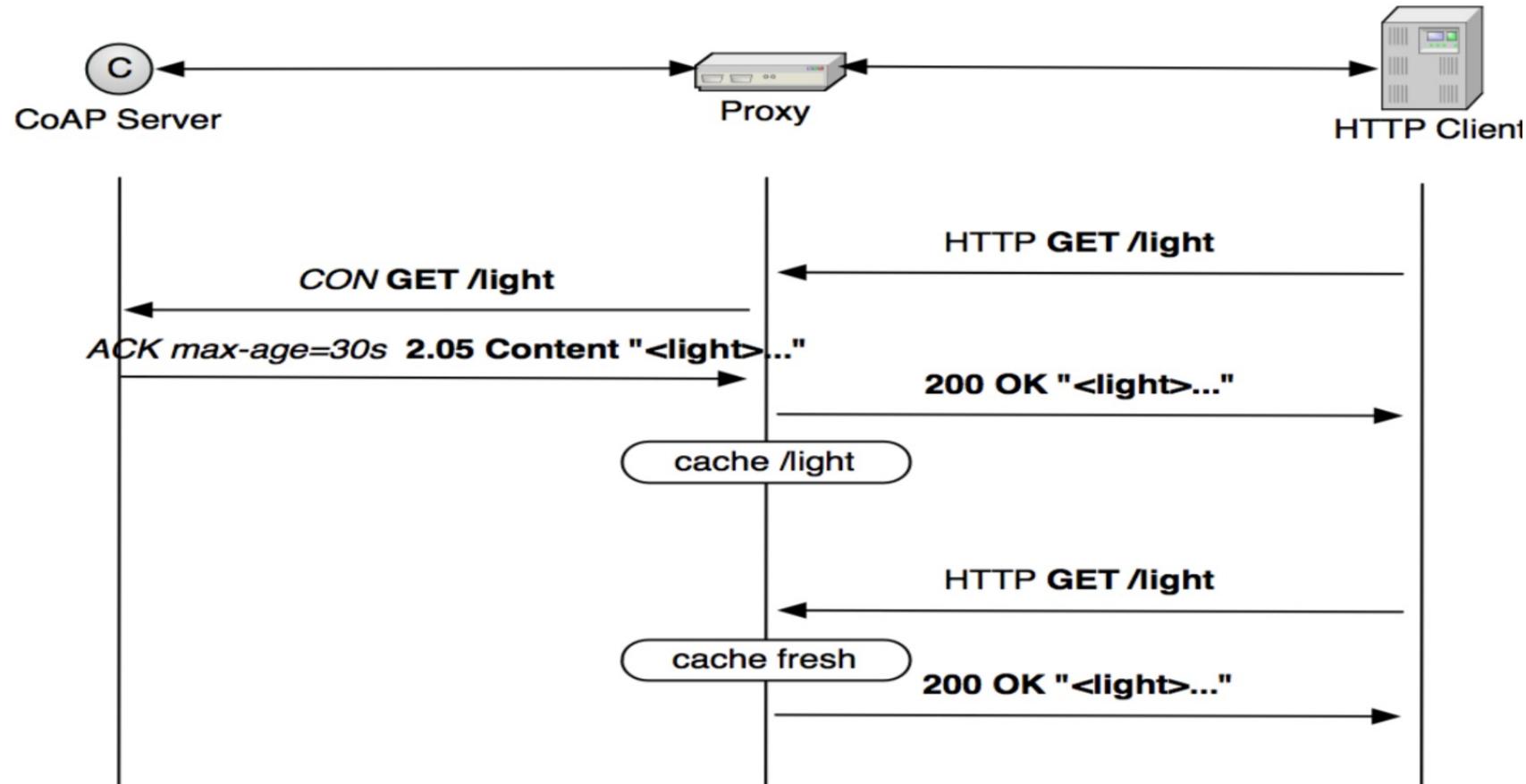
- Serialização de um pedido e de uma resposta

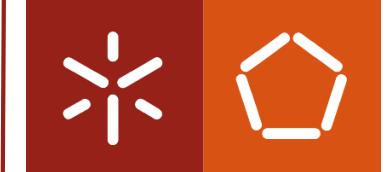




CoAP – Constrained Application Protocol

- **Proxying e Caching:** vital para os nós com restrições
 - Controlado pelos campos **Max-Age** e **ETag**

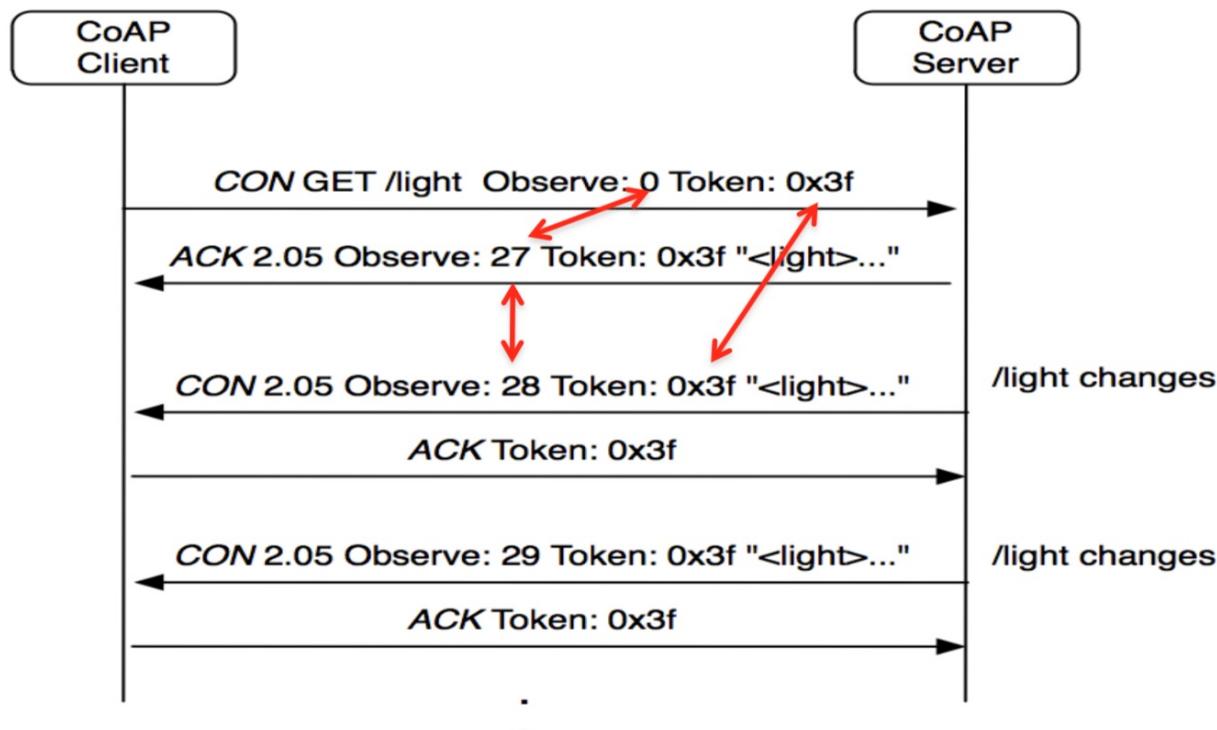


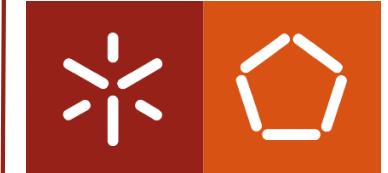


CoAP – Constrained Application Protocol

- **Subscrição e Notificação: Observe**

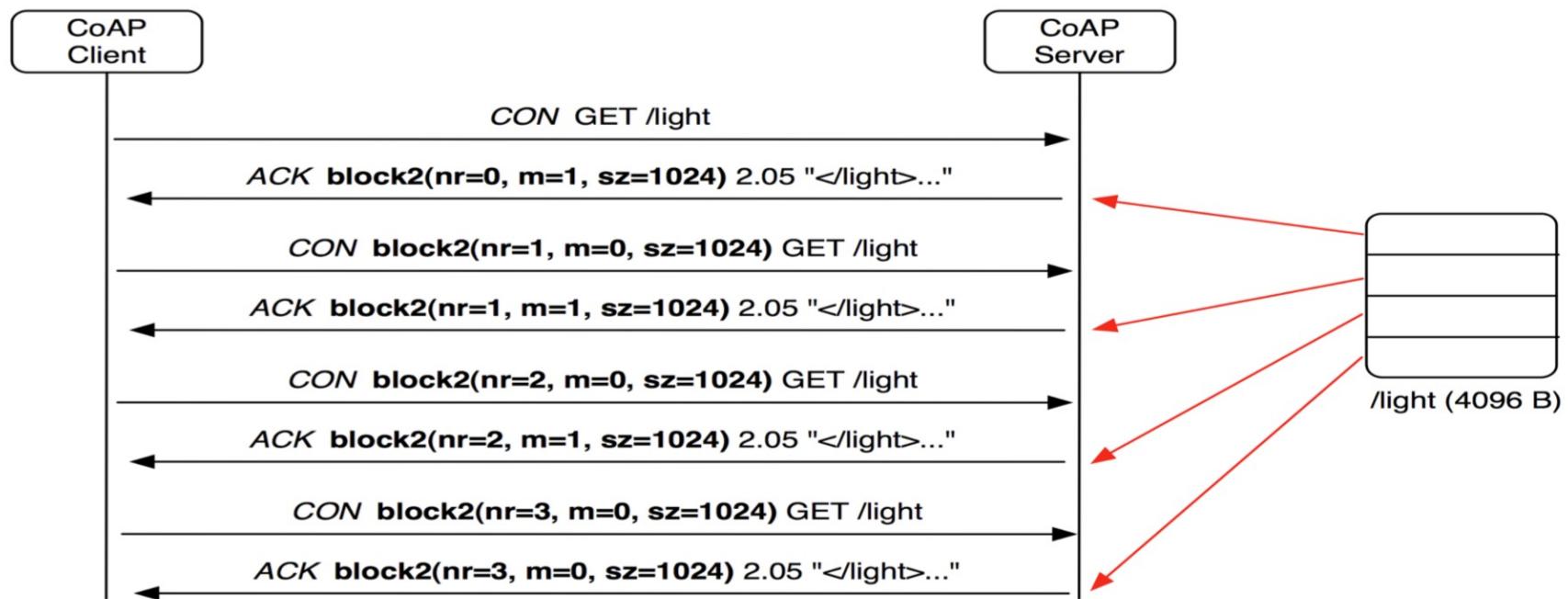
- *Um conceito muito importante que não é nada claro no modelo Web HTTP*

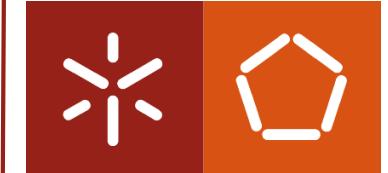




CoAP – Constrained Application Protocol

- Transferência fragmentada de ficheiros grandes (grandes blocos de dados)
 - Exemplos: atualização de *firmware* de um sensor, descarga dos seus *logs*
 - A resposta inclui numero do fragmento e flag a dizer se há mais blocos m=1 (o último m=0)





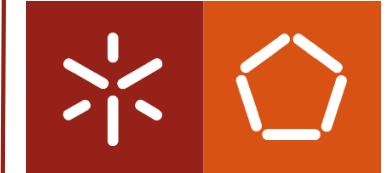
CoAP – Constrained Application Protocol

- **Implementações existentes:**

- Lista de implementações disponíveis
- Firefox Plugin: **Copper** (<https://github.com/mkovatsc/Copper>)
- Chrome Plugin: **Copper4Cr** (<https://github.com/mkovatsc/Copper4Cr>)
- CoAP em Java: **Californium** (<http://www.eclipse.org/californium/>)
- CoAP em Java: **nCoAP** (<https://github.com/okleine/nCoAP>)
- CoAP em Java: **Leshan** (<https://github.com/eclipse/leshan>)
- **Libcoap** em C: <https://libcoap.net>
- Etc... Muitas e boas...

- **Implementações comerciais**

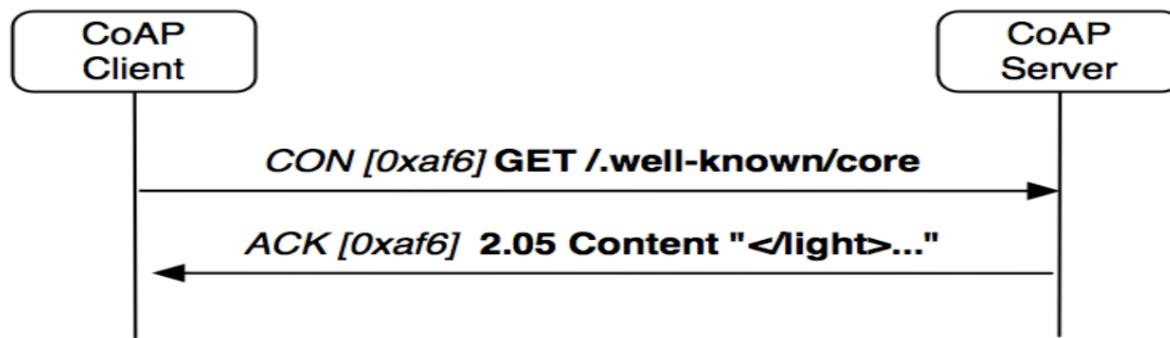
- ARM mbed: <https://docs.mbed.com/docs/mbed-client-guide/en/latest/>
- oneMPOWER (InterDigital)
- Thethings.io



CoAP – Constrained Application Protocol

- Descoberta de recursos é nativa!

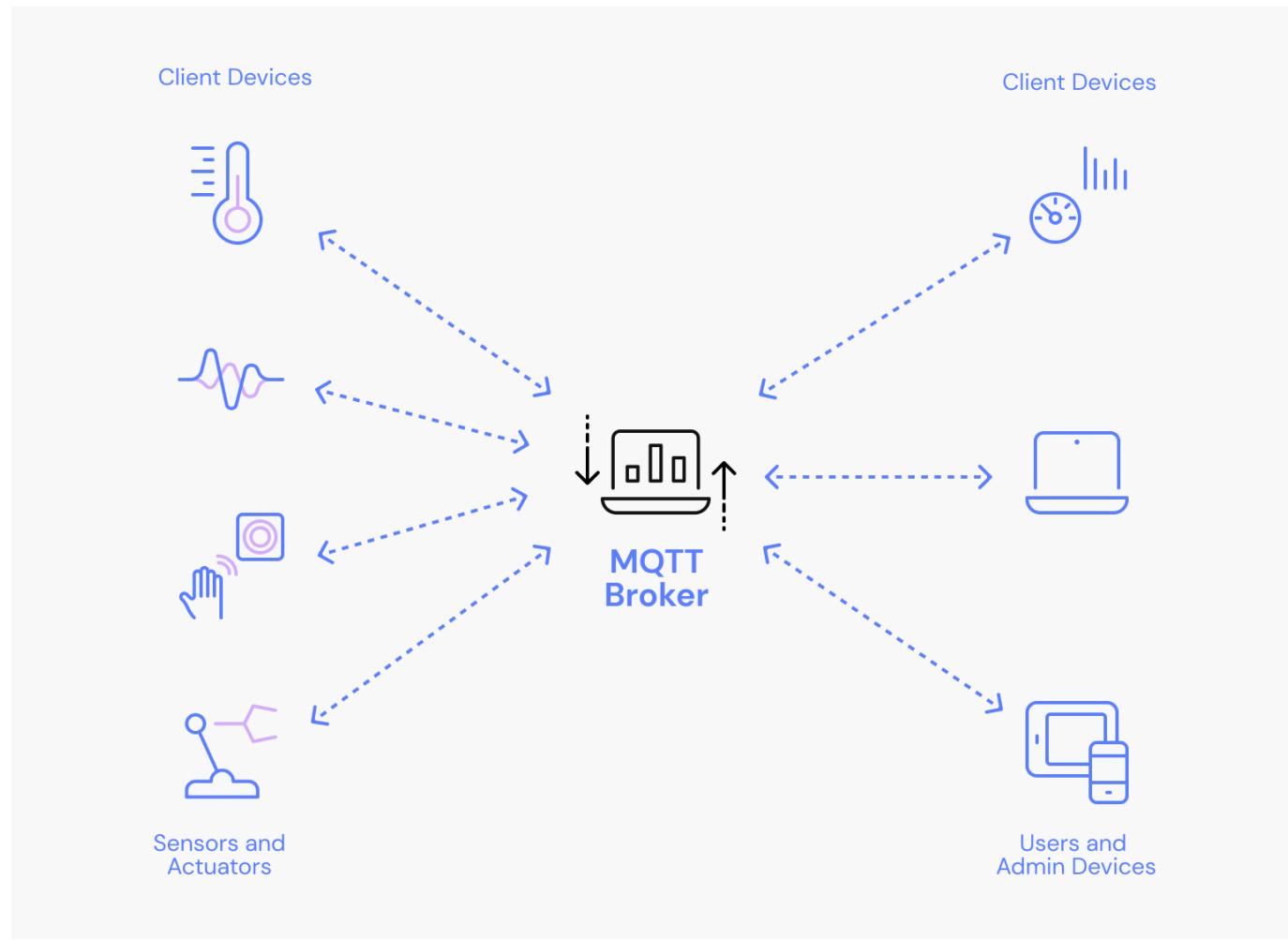
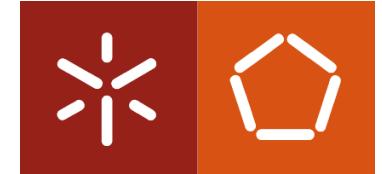
- Utiliza o conceito de **Links** (estende a definição) já existente e em uso no Web



```
</dev/bat>;obs;rt="ipso:dev-bat";ct="0",
</dev/mdl>;rt="ipso:dev-mdl";ct="0",
</dev/mfg>;rt="ipso:dev-mfg";ct="0",
</pwr/0/rel>;obs;rt="ipso:pwr-rel";ct="0",
</pwr/0/w>;obs;rt="ipso:pwr-w";ct="0",
</sen/temp>;obs;rt="ucum:Cel";ct="0"
```

RFC 6690:
Atributos:
rt= resource type
if= interface descr.
sz= size
ct= content-type
0 text/plain
50 application/json

MQTT





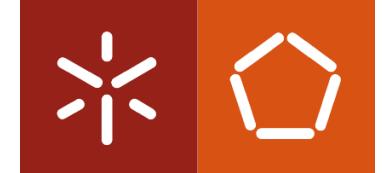
● O que é o MQTT?

- *Message Queue Telemetry Transport*

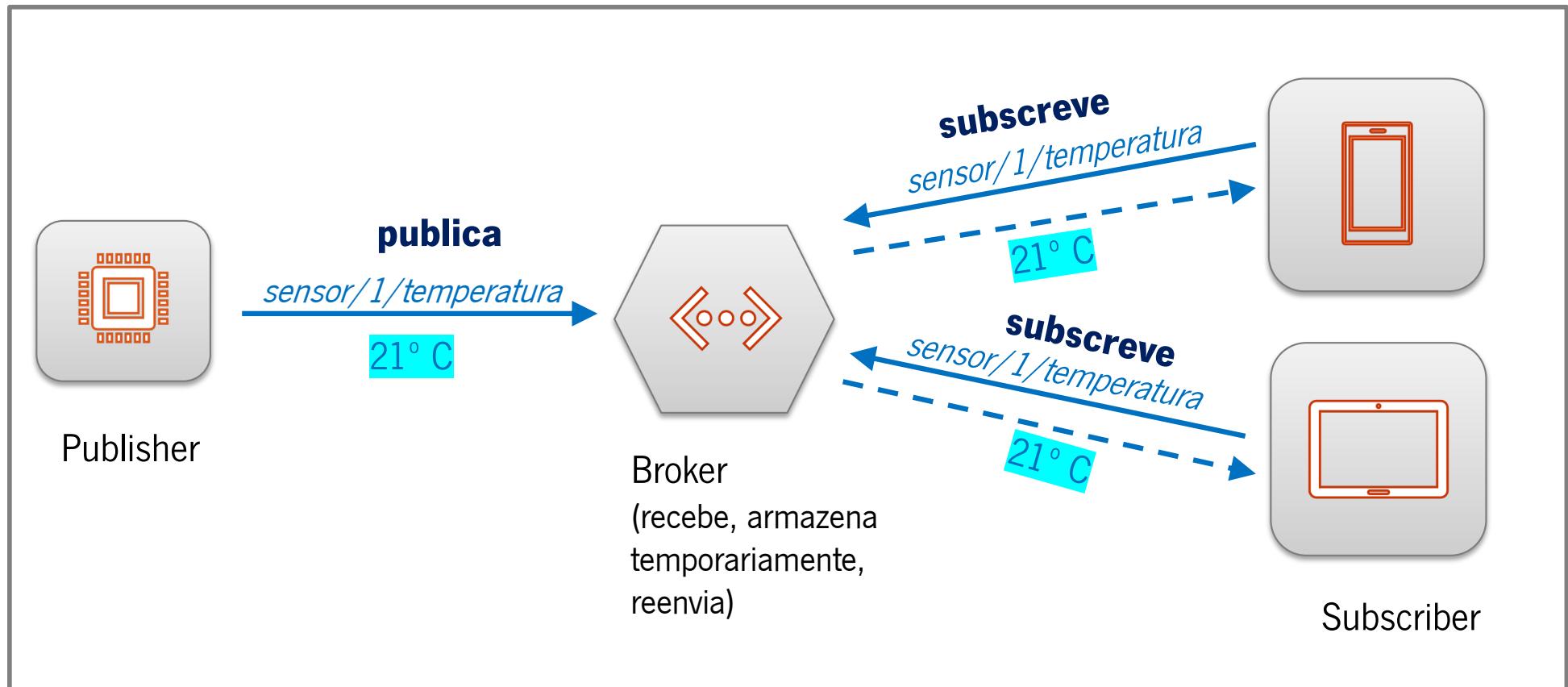
- Protocolo “leve” para ser usado na Internet das Coisas (IoT)
- Os dispositivos (sensores e atuadores) possuem restrições:
 - Energia: bateria pode ser limitada
 - Largura de banda: em dispositivos remotos, dependendo da tecnologia usada nas comunicações, também pode ser limitada

- Por isso o protocolo deve ser:

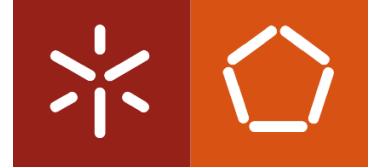
- Simples de implementar
- Com pouca sobrecarga de controlo: pacotes com cabeçalhos pequenos!
- Poder transportar qualquer tipo de dados: significa ser *agnóstico* aos dados
- Fornecer alguma qualidade de serviço: garantias de entrega dos dados
- Suportar sessões contínuas ao longo do tempo



● Paradigma Publica-Subscreve (*Publish-Subscribe*)



- Padrão baseado em mensagens, que desacopola os clientes que enviam as mensagens dos clientes que recebem as mensagens!



● Clientes MQTT

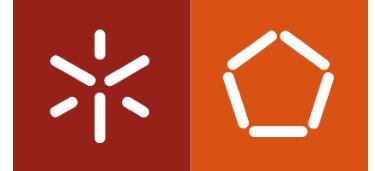
- Nenhum cliente tem de divulgar o seu endereço ou conhecer o endereço dos outros clientes!
- Ligam-se ao broker apenas, ou para **publicar, subscrever**, ou ambos!

Modelo adequado à **Internet das Coisas (IoT)**

Não é preciso conhecer os endereços dos sensores ou atuadores!

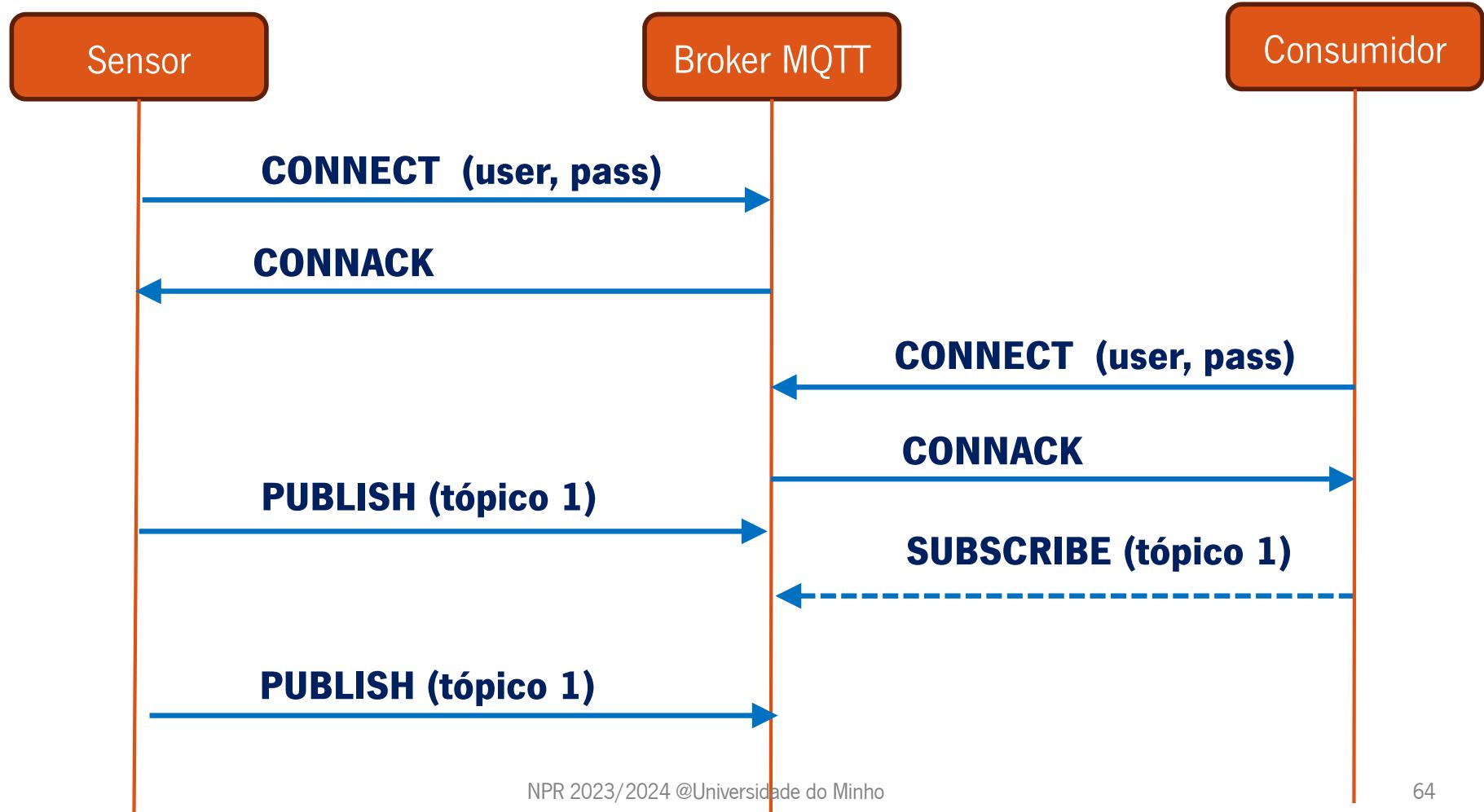
O broker serve de intermediário entre clientes

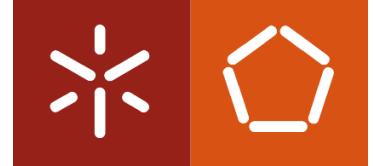
- Pode existir desfasamento temporal: cliente publica e subscriptores só ler e responder algum tempo depois (depende da conexão)



- Clientes conectam-se ao broker MQTT

- Podem publicar e subscrever





● Tópicos organizados de forma hierárquica

- Os clientes publicam os dados em **tópicos** organizados hierarquicamente:

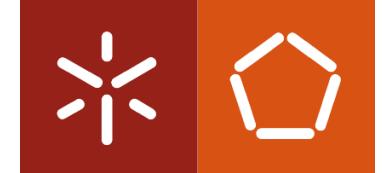
Um cliente publica a temperatura do sensor do laboratório da escola:
escola/lab/sensor/temperatura

- Subscritores podem usar caracteres *joker (wildcard)* único nível ou multinível: : ‘+’ e ‘#’

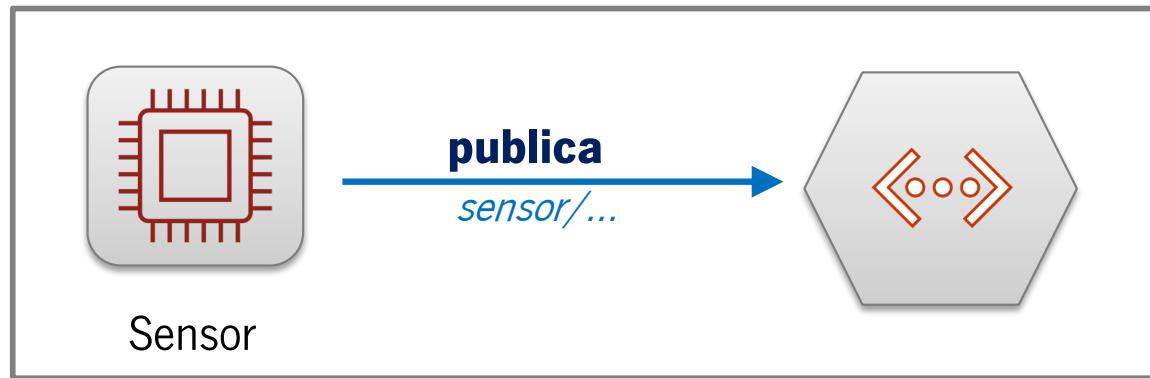
Um subscritor pode subscrever só a temperatura desse sensor:
escola/lab/sensor/temperatura

Ou de todos os tópicos de temperatura do lab (coloca um + no terceiro nível):
escola/lab/+/temperatura

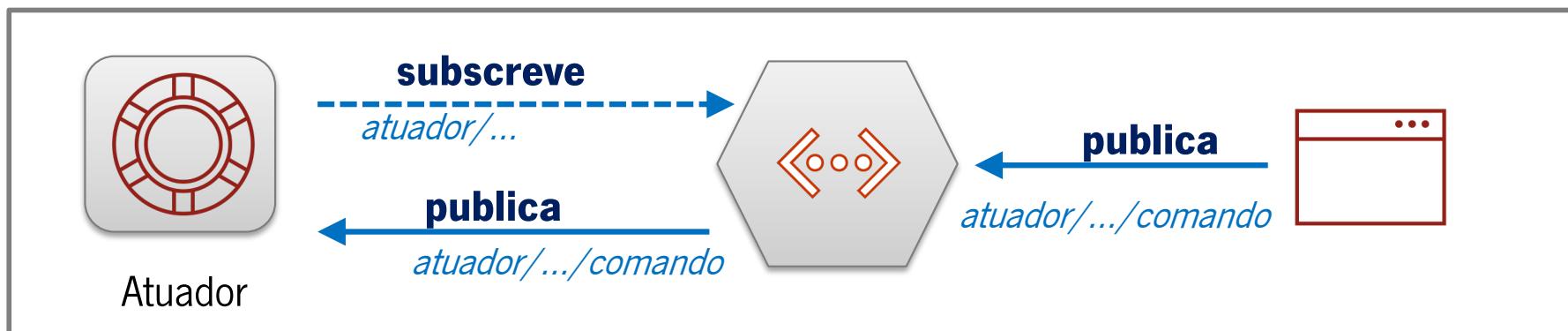
Ou todos os dados do lab da escola (coloca um # no final do segundo nível):
escola/lab/#

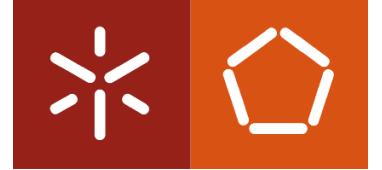


- Cliente pode ser um sensor, que publica dados:



- Ou um atuador, que subscreve dados que são comandos:

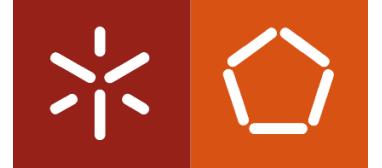




● Broker MQTT

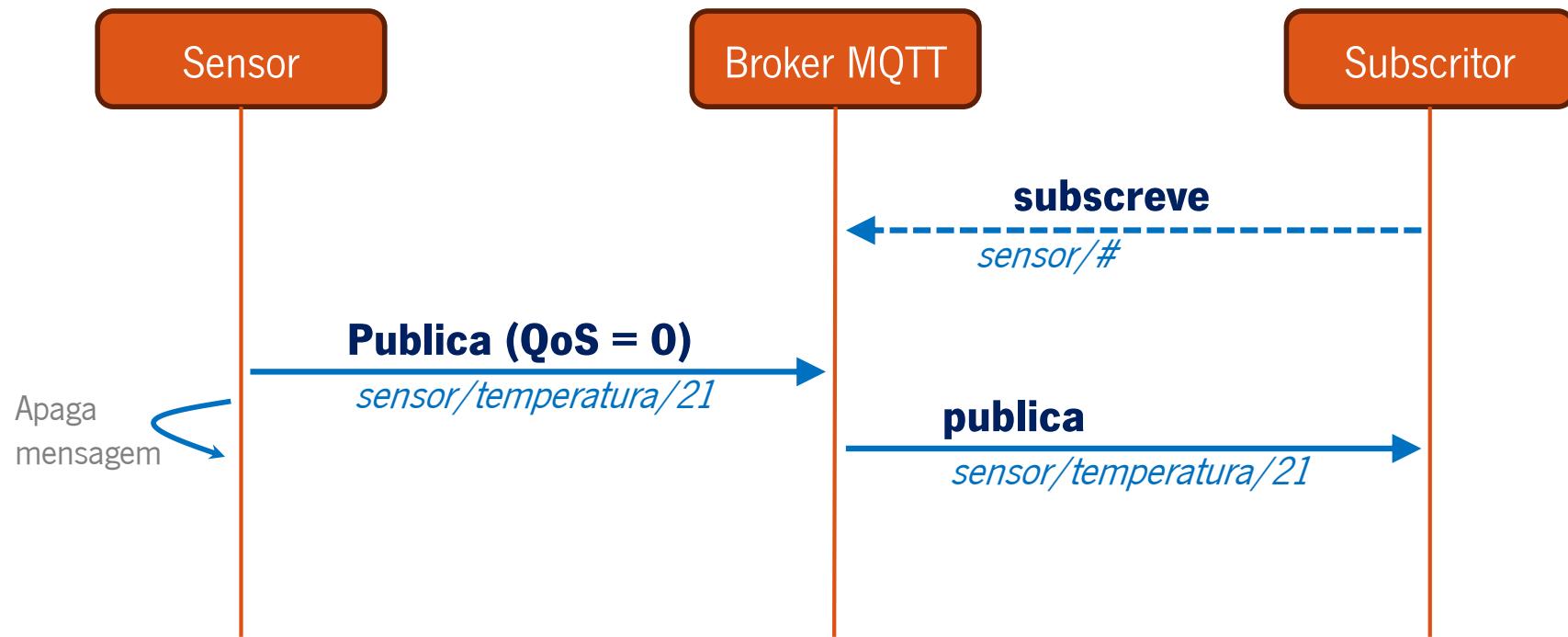
- Conecta os clientes e **filtre** os dados
 - Filtragem por assunto de acordo com os tópicos: o broker só envia os dados aos clientes que subscreveram os tópicos!:
 - Os clientes subscritores também podem filtrar os conteúdos
- Mantém sessões com os clientes e gere as mensagens entregues e por entregar
- Pode aceitar clientes anónimos ou apenas clientes autenticados e autorizar o acesso aos tópicos

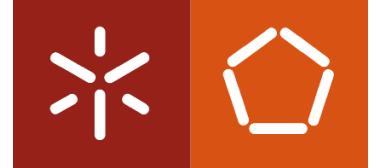
Se o broker receber uma mensagem num tópico para o qual não tem subscritores, pode simplesmente apagar a mensagem, a menos que o cliente que a publica tenha ativado a opção "RETAIN" que força o broker a guardá-la para entregar a futuros subscritores



● Qualidade de Serviço

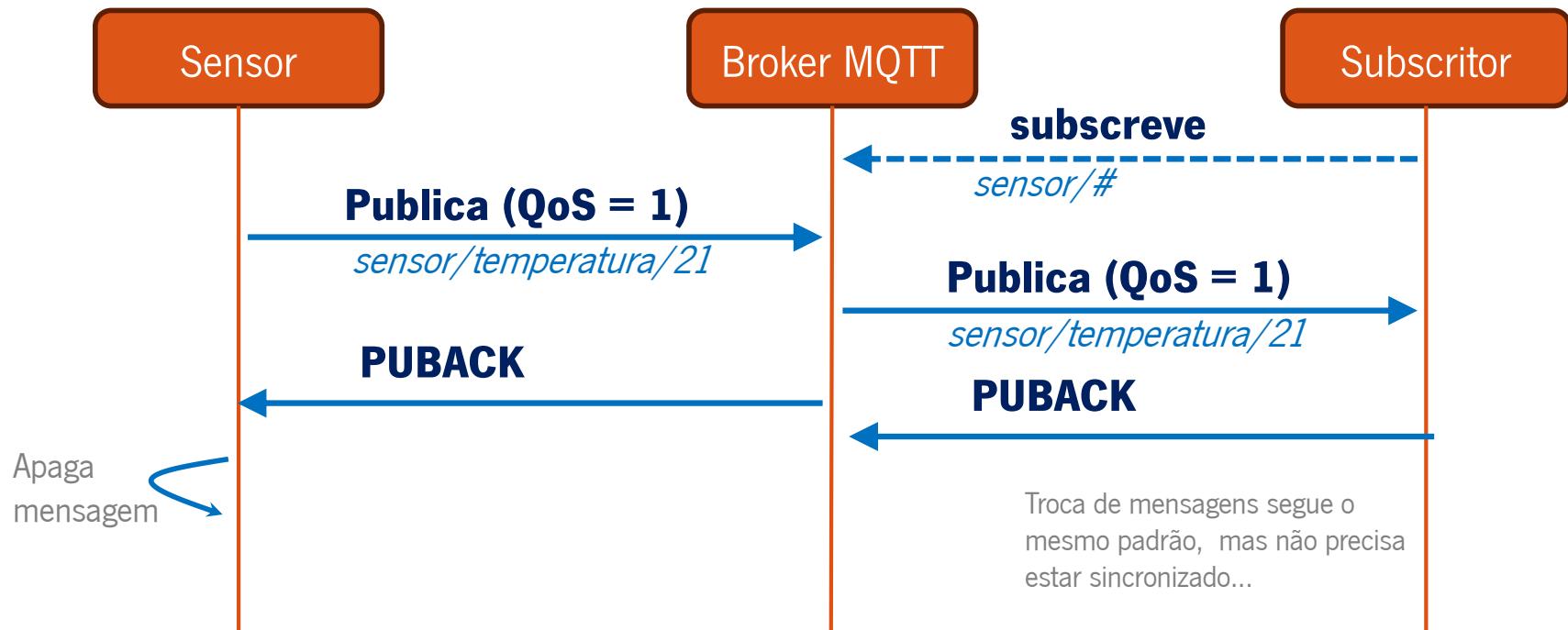
- QoS 0 (*At Most Once*) – No máximo uma vez. Também conhecido como “envia e esquece”. Envio das mensagens sem garantias.

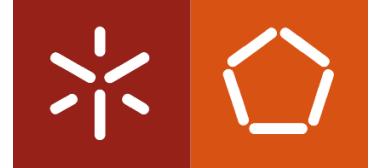




● Qualidade de Serviço

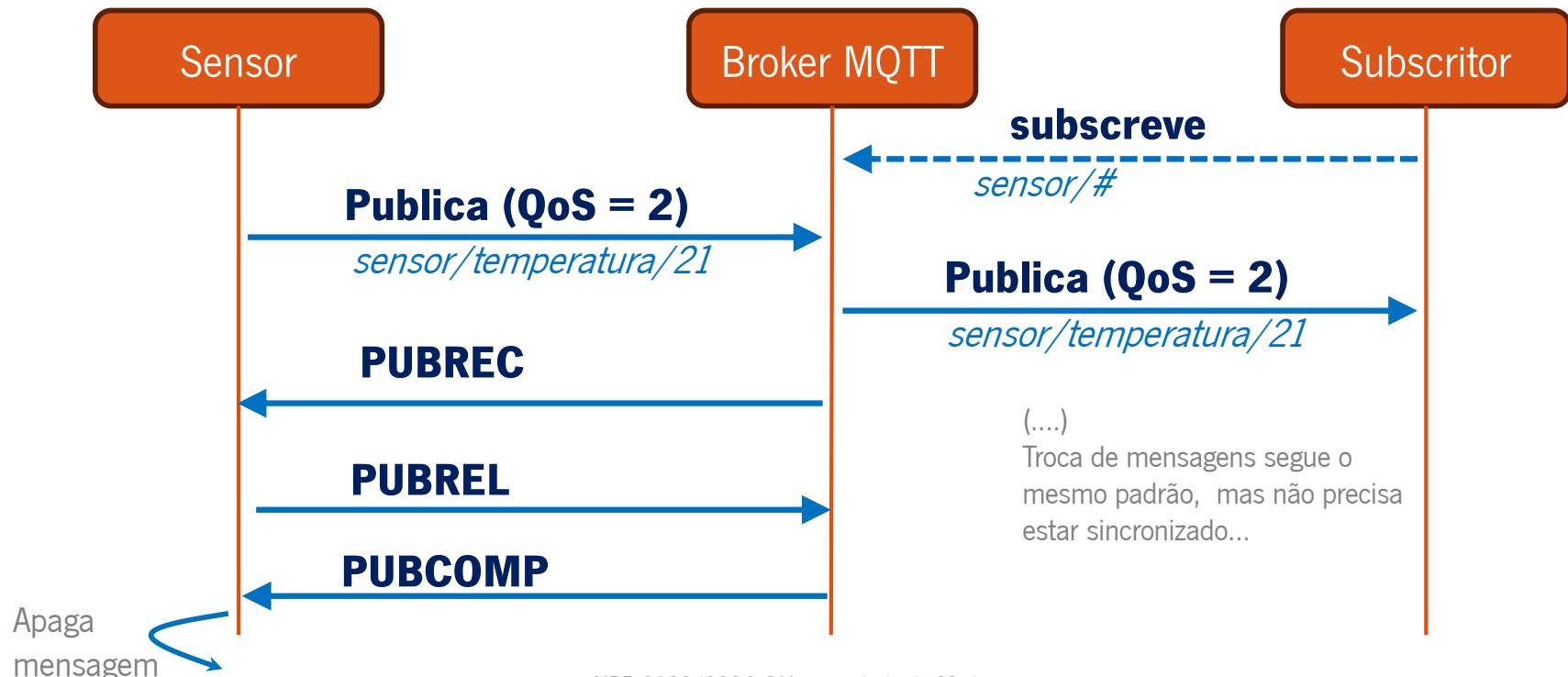
- QoS 1 (*At Least Once*) – Pelo menos uma vez. A mensagem é enviada e espera-se uma confirmação da receção. Se não houver confirmação retransmite-se. O que pode originar duplicados.

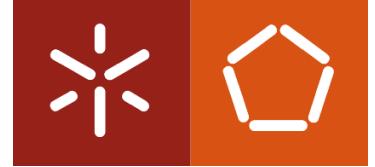




● Qualidade de Serviço

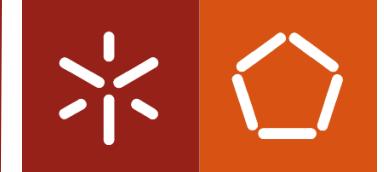
- QoS 2 (*Exactly Once*) – Exatamente uma vez. Nível mais elevado de garantias. Garante que a mensagem é entregue uma e uma só vez. Exige confirmação dupla.





● Última vontade (*Last Will*)

- O broker MQTT monitoriza todas as conexões com os seus clientes, e deteta quando eles se desligam
- Normalmente os subscriptores não são notificados quando um cliente que publica dados se desconecta!
 - *Exemplo: sensor de temperatura perde conexão e deixa de enviar dados*
- No entanto os clientes podem deixar um "testamento" (última vontade) que deve ser enviada quando perder a conexão:
 - A mensagem é definida inicialmente no estabelecimento da conexão
 - Só é enviada aos subscriptores, se perder a conexão anormalmente!

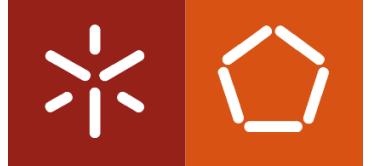


● Como escolher um broker MQTT?

- Versões do protocolo que suporta: MQTT v3.1.1, MQTT v5.0
- Níveis de Qualidade de Serviço suportados: QoS 0, QoS 1, QoS 2
- Suporte de segurança: TLS/SSL
- Desempenho: suporte para grande número de clientes
- Integração com outras plataformas de IoT
- Facilidade de instalação, configuração e gestão
- Documentação

● Exemplos?

- Eclipse Mosquitto, EMQX, RabbitMQ, Apache ActiveMQ, HiveMQ



● Instalação do Eclipse Mosquitto (raspberry pi)

1. Abrir um terminal e atualizar o sistema:

```
$ sudo apt update && sudo apt upgrade
```

2. Instalar o software (broker e cliente):

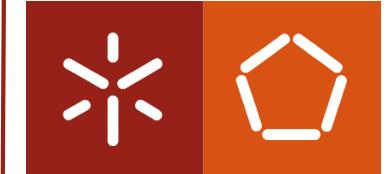
```
$ sudo apt install -y mosquitto mosquitto-clients
```

3. Testar a instalação:

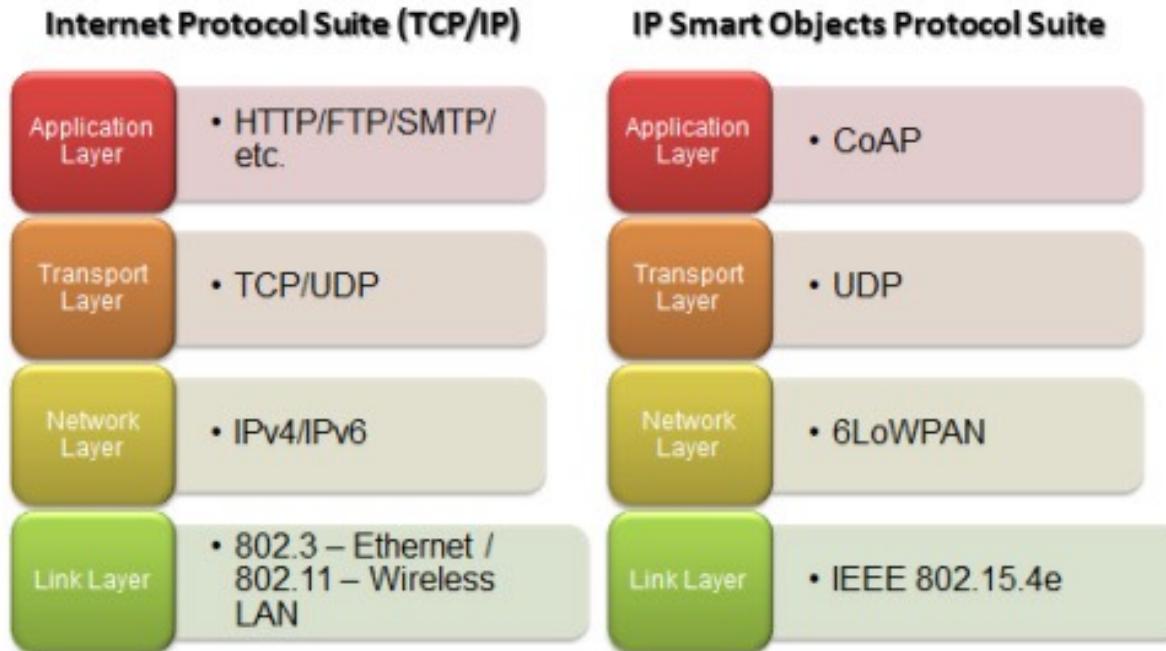
```
$ mosquitto -v
```

4. Ativar o broker automaticamente:

```
$ sudo systemctl enable mosquitto.service
```

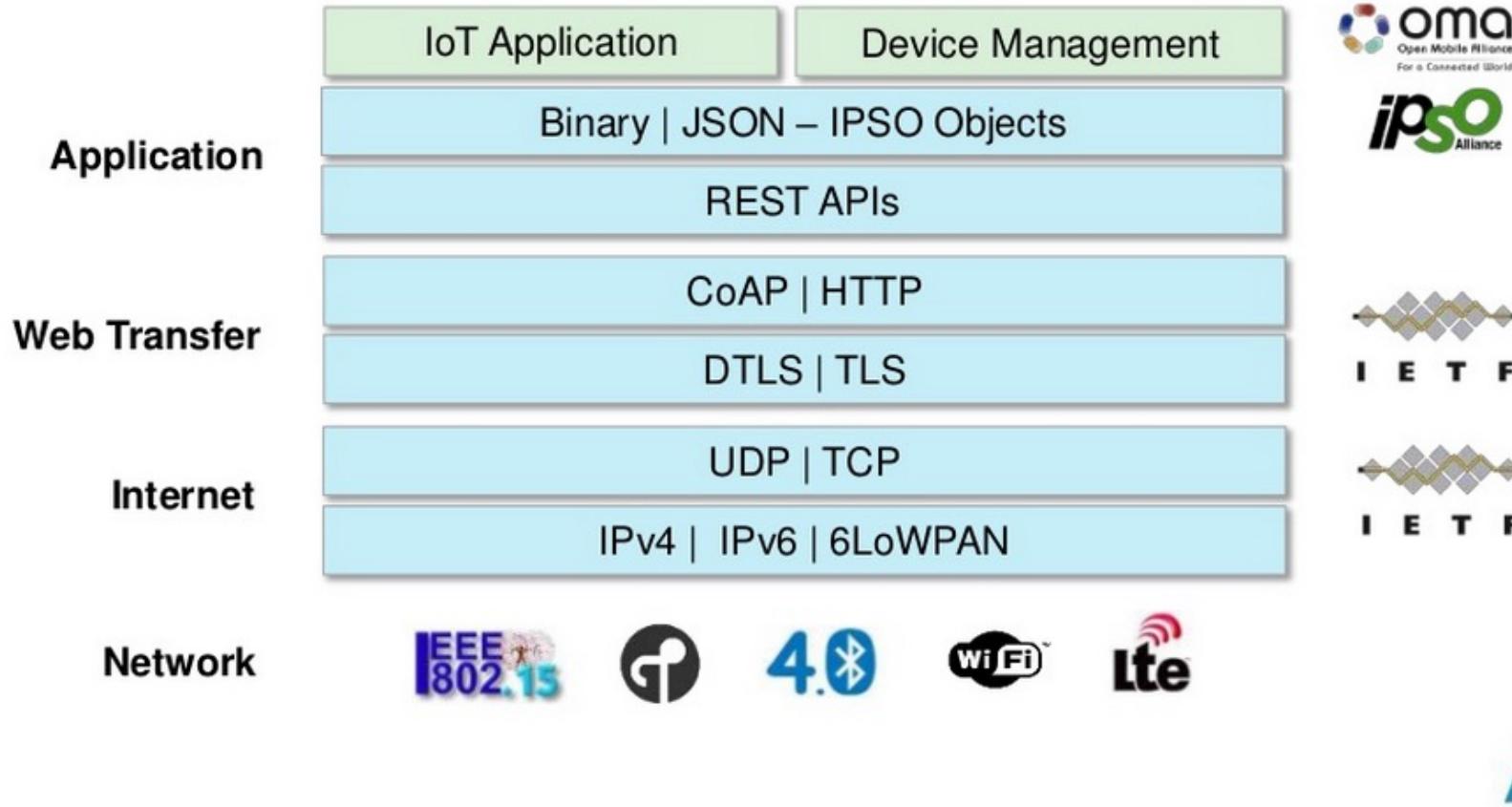
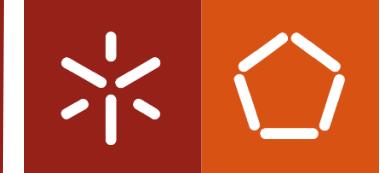


IoT: Visão geral (Perspetiva 1)



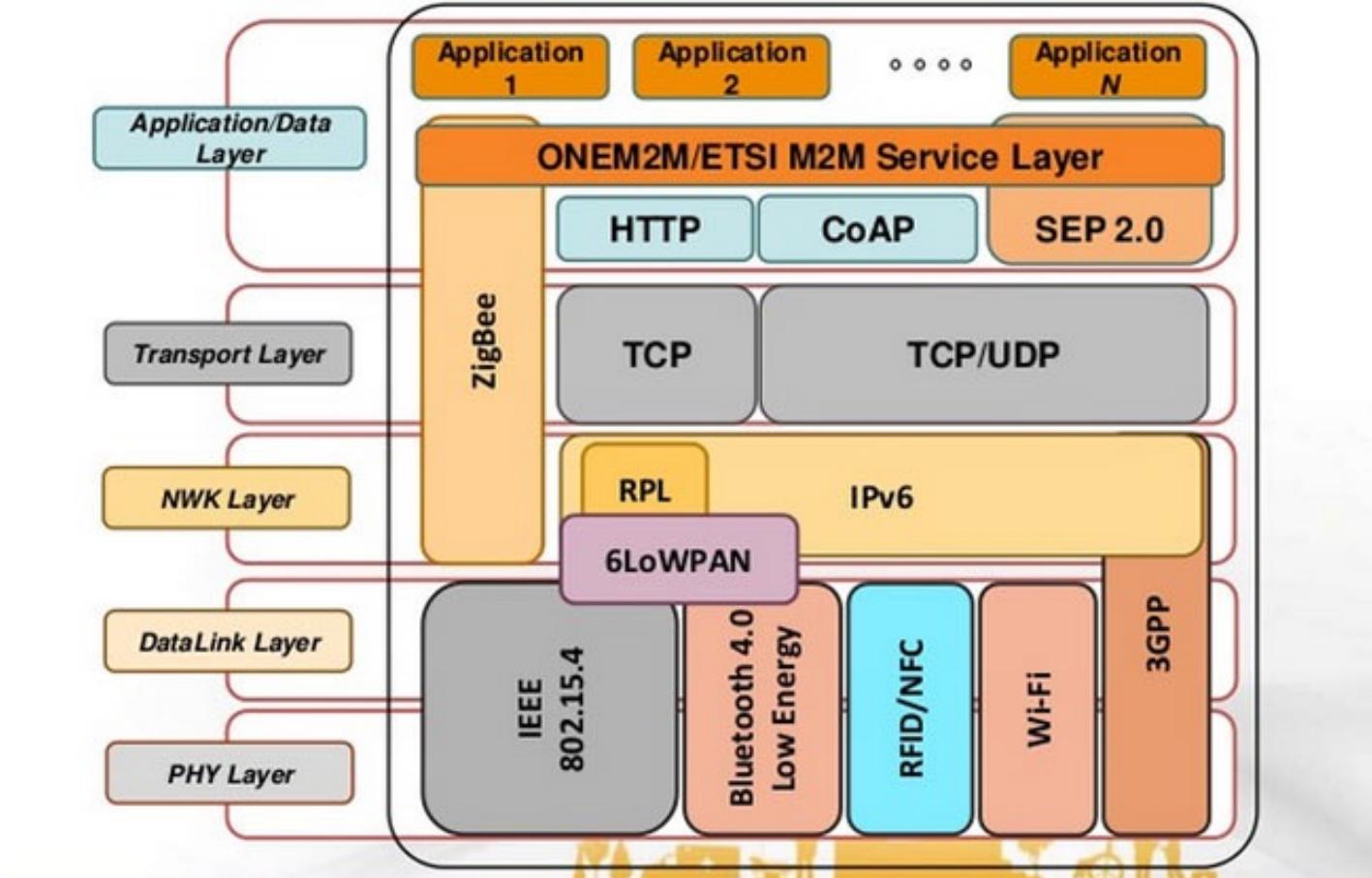
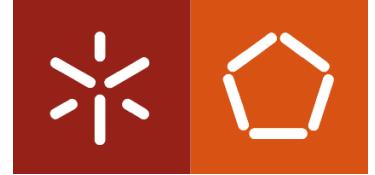
*IoT Standards and Protocols, <https://www.postscapes.com/internet-of-things-protocols/#protocols>
(Graphic via Ronak Sutaria and Raghunath Govindachari from Mindtree Labs in "Making sense of interoperability:Protocols and Standardization initiatives in IOT")*

IoT: Visão geral (Perspetiva 2)

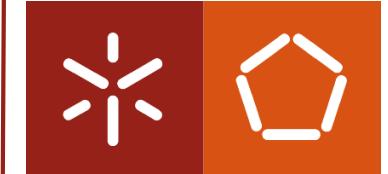


IoT Standards and Protocols, <https://www.postscapes.com/internet-of-things-protocols/#protocols>
(Credit: Simon Ford - Director of IoT Platforms ARM)

IoT: Visão geral (Perspetiva 3)

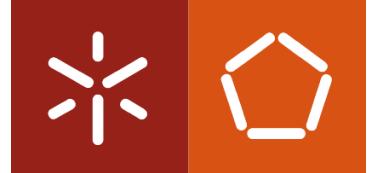


IoT Standards and Protocols, <https://www.postscapes.com/internet-of-things-protocols/#protocols>
(De: <http://www.slideshare.net/butler-iot/butler-project-overview-13603599>)



Bibliografia

- **6LoWPAN WG** <http://datatracker.ietf.org/wg/6LowPan/charter/> (já encerrado)
- **6Io WG:** <https://datatracker.ietf.org/wg/6lo/charter/> (ainda bem activo!)
- **IEEE802.15.4**, IEEE std. 802.15.4, Part. 15.4: *Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks(LR-WPANs)*, Standard for Information Technology Std., 2011.
- G. Montenegro, N. Kushalnagar, J. Hui, and D. Culler, *Transmission of IPv6 Packets over IEEE 802.15.4 Networks*, **RFC 4944**, Internet Engineering Task Force RFC 4944, September 2007. - Updated by: 6282, 6775
- J. Hui and P. Thubert, *Compression Format for IPv6 Datagrams over IEEE 802.15.4 Based Networks*, **RFC 6282**, Internet Engineering Task Force RFC 6282, September 2011.
- Nieminen J., Savolainen T., Isomaki M., Patil B., Shelby Z., Gomez C. *IPv6 over BLUETOOTH(R) Low Energy*. IETF; Fremont, CA, USA: 2015. **RFC 7668**.
- Z. Shelby and C. Bormann, *6LoWPAN: The Wireless Embedded Internet*, ser. Wiley Series on Communications Networking & Distributed Systems. John Wiley & Sons, 2010.
- *IoT Standards and Protocols*, <https://www.postscapes.com/internet-of-things-protocols/#protocols>



Bibliografia

- **CoAP Technology:** <http://coap.technology/>
- Zach Shelby, *ARM CoAP Tutorial*,
<https://community.arm.com/iot/b/blog/posts/coap-video-tutorial>(visitado em 05.05.2023)