O CoAP não procura ser um HTTP compactado, mas sim um protocolo Web especializado, com vantagens sobre o HTTP

Carlos Alves PG41840

Universidade do Minho, Portugal

Arquitetura Emergentes de Redes

2019/2020

Resumo. Essencialmente, o **COAP** (protocolo de Aplicação Restrita) é um protocolo de transferência de web especializado para uso com nós limitados e redes limitadas. Geralmente os nós/nodes tem microcontroladores de 8 bits com pequenas quantidades de ROM e RAM, enquanto que redes limitadas como IPv6/(6LoWPANs) geralmente apresentam altas taxas de erro de pacotes e uma taxa de transferência típica de 10s kbit/s. Este protocolo foi projetado para aplicações *machine-to-machine*(M2M) como energia inteligente e automação predial (*building automation*). Além disto, o protocolo **COAP** fornece um modelo de interação de solicitação/resposta entre os terminais de aplicativos, suporta a descoberta interna de serviços e recursos e inclui os principais conceitos da Web, como URIs. Interage facilmente com HTTP para integração com a Web e atende a requisitos especializados como *multicast*, sobrecargas muito baixas ou simplesmente para ambientes limitadas. Porém o **CoAP** não procura ser um HTTP compactado, mas sim um protocolo Web especializado com vantagens sobre o HTTP.

ÍNDICE

INTRODUÇÃO

Este trabalho foi proposto realizar na unidade curricular de Arquitetura Emergente de Redes tem como objetivo escrever um artigo sobre o protocolo **CoAP** e ainda o fato do mesmo não ser um HTTP compactado, mas sim um protocolo Web especializado com vantagens sobre o HTTP. Portanto ao longo deste documento escrito será apresentado o protocolo CoAP e as razões pela qual a primeira afirmação é incorreta.

THE CONSTRAINED APPLICATION PROTOCOL (COAP)

Primeiramente, o uso de serviços da Web (API's) na internet tornou-se "omnipresente" na grande parte das aplicações e depende da Transferência Representacional de Estado (REST) da Web. Um dos objetivos primários do protocolo CoAP é projetar um protocolo Web genérico para os requisitos especiais desses ambiente limitados. Como já referido, o CoAp não é um HTTP compacto, mas sim realiza um subconjunto de REST comum com HTTP, otimizado para aplicações *machine to machine*. Além disso o CoAP pode ser usado para remodelar interfaces simples de HTTP num protocolo mais compacto, pois este oferece recursos para *machine to machine*, como descoberta integrada, suporte a *multicast* e troca de mensagens assíncronas.

MECÂNICAS

- Ligação UDP com confiabilidade opcional que suporta solicitações de *unicast* e *multicast;*
- Protocolo Web que atende aos requisitos de M2M em ambientes limitados;
- Trocas de mensagens assíncronas;
- Baixa sobrecarga de cabeçalho e complexidade de análise;
- URI e suporte ao tipo de conteúdo;
- Proxy simples e recursos de cache;
- Um mapeamento HTTP sem estado, permitindo a construção de proxies, fornecendo acesso aos recursos do CoAP via HTTP de maneira uniforme ou para que interfaces simples HTTP sejam realizadas alternativamente através do CoAP.
- Ligação de segurança ao Datagram Transport Layer Security(DTLS)

O modelo de interação do protocolo CoAP é semelhante ao modelo cliente/servidor do protocolo HTTP, mas as implementações CoAP de forma geral devem suportar os dois papeis, funcionando como cliente e servidor sempre que necessário. Os pedidos CoAP são equivalentes aos do HTTP sendo enviados por um cliente que solicita uma ação ao servidor, o servidor responde a esse mesmo pedido com um código de resposta.

Contrariamente ao HTTP, o CoAP lida com essas trocas de forma assíncrona num transporte orientado a Datagramas, como o UDP. Este processo é feito logicamente usando uma camada de mensagens que suporta confiabilidade opcional (com retirada exponencial).

ARQUITETURA

Este protocolo define quatro tipos de mensagens:

- *Confirmable(CON):* Quando uma mensagem CON é recebida, é devolvida uma mensagem do tipo *Acknowledgement(ACK)* ou *Reset(RST).* Este tipo de mensagem nunca deve estar vazia.
- *Non-Confirmable*(*NOS*):Este tipo é para mensagens que são repetidas regularmente, no caso de ocorrer percas de certas mensagens não relevantes.
- *Acknowledgement(ACK)*: ACK corresponde ao tipo de mensagem de resposta à mensagem CON, não indicando o estado do pedido(sucesso ou falhanço).
- *Reset(RST*): RST corresponde ao tipo de mensagem de resposta à mensagem CON, apenas informa que o pedido não foi realizado.

Além destes 4 tipos de mensagens a arquitetura do protocolo ainda define quatro tipos de métodos parecidos com os de HTTP:

- GET
- PUT
- POST
- DELETE

MODELO DE MENSAGENS

O modelo de mensagens do CoAP é baseado na troca de mensagens pelo UDP entre *endpoints*. São usados cabeçalhos binário de comprimento fixo de 4 bytes. O formato das mensagens CoAP contém mensagem ID usado para detetar duplicações e para uma garantir fiabilidade. Estas mensagens são usadas essencialmente por pedidos e respostas. Em relação á fiabilidade, esta é conseguida marcando a mensagem como *Confirmable(CON)* e ainda usando um *timeout* padrão entre retransmissões até que o recetor envie de volta um *Acknowledgement(ACK)* com o mesmo ID da mensagem de origem.

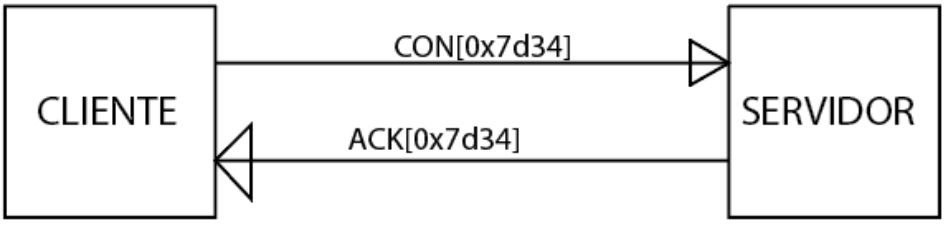


Figura 1. Transmissão confiável de mensagens

(Imagem feita com base na do RFC 7252)

No caso de se vir a constatar que uma mensagem do tipo *Confirmable(CON)* não é processada, o recetor emite uma mensagem do tipo *Reset(RST)* em vez da *Acknowledgement(ACK).*

Porem para mensagens que não requerem confirmação na transmissão, por exemplo as aplicações de medição de *stream* de dados de sensores, basta então ser enviadas mensagens do tipo *Non-Confirmable(NON).* Estas mensagens também contem um Message ID, tendo a mesma função que no tipo *Confirmable(CON)* de detetar duplicações. No próprio RFC do Protocolo é ainda referido que quando um destinatário não é capaz de processar uma mensagem Non-*Confirmable(NON),* este pode responder com um mensagem do tipo *Reset(RST).*

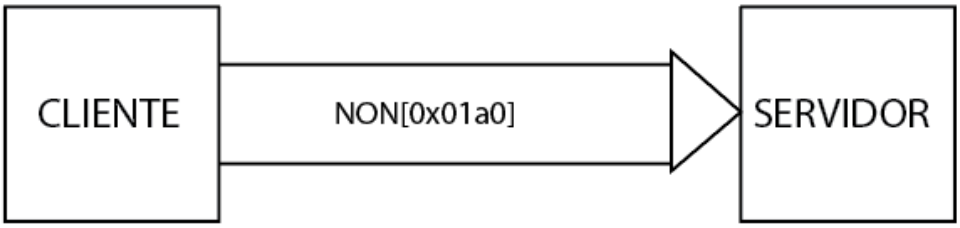


Figura 2. Transmissão não-confiável de mensagens

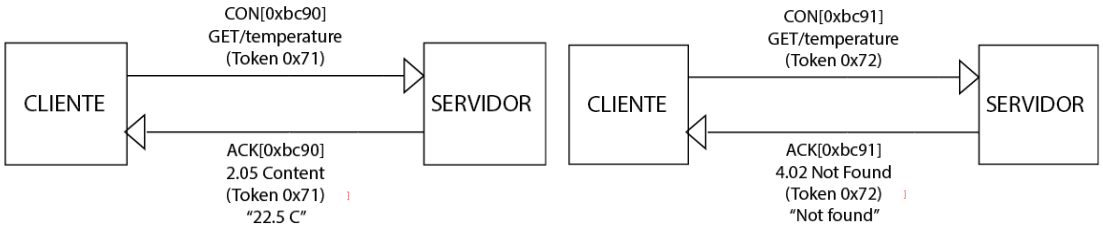
(Imagem feita com base na do RFC 7252)

Como o CoAP corre sobre UDP, este também suporta o uso de IP Multicast em endereços de destino, permitindo solicitações CoAP multicast.

MODELO PEDIDO E RESPOSTA

Neste quesito, a semântica de solicitações e de respostadas do CoAP é "transportada" nas mensagens CoAP, onde é incluído um Código do Método ou Codigo da Resposta, ainda opcionalmente a informação dos pedidos e respostas como o URI, e o tipo de informação podem ser incluídos nas opções do CoAP. Um token é usado para corresponder as respostas às solicitações independentemente das mensagens subjacentes.

O request é carregado em mensagens Confirmable(CON) ou Non-Confirmable(NON), e consequentemente a resposta a esse request resulta numa mensagem Acknowledgement(ACK). – Piggybacked response



Se o servidor não é capaz de responder imediatamente a esse pedido responde com uma mensagem ACK vazia, para que o cliente não retransmita o pedido. Quando a resposta está pronta, o servidor envia o pedido numa nova mensagem Confirmable. Se num pedido é enviada uma mensagem Non-Confirmable então a resposta é normalmente enviada através de uma nova mensagem Non-Confirmable, embora esta mensagem enviada pelo servidor possa ser igualmente uma mensagem Confirmable.

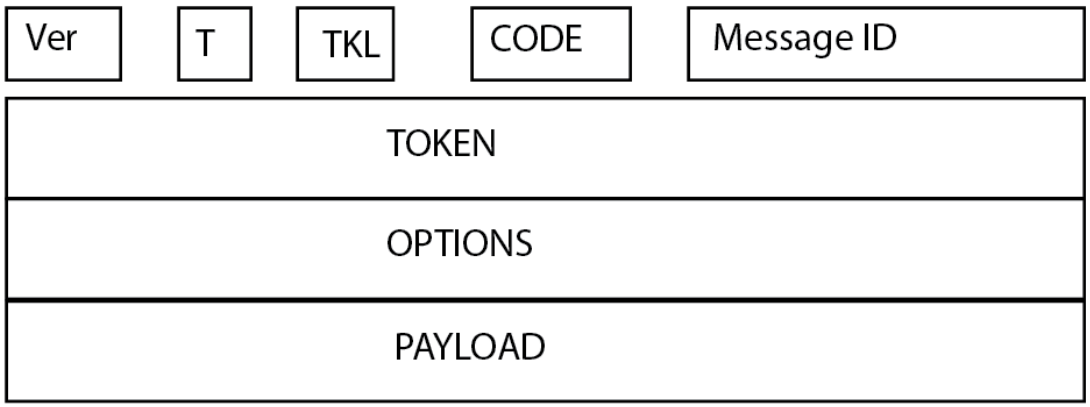
INTERMEDIÁRIOS E CACHE

Como referido no próprio RFC do protocolo, o CoAP suporta o armazenamento em cache para atender pedidos com bastante eficiência. Uma cache simple é ativado usando informações de validade transportadas com respostas CoAP.

O protocolo CoAP suporta também o pedido a um proxy em nome de outro terminal CoAP, usando o URI do recurso para solicitar o pedido, enquanto o endereço IP de destino é o endereço da proxy. Alem disso, com proxy é possível converter o método ou codigo de resposta.

FORMATO DE MENSAGENS

O CoAP possui mensagens codificadas num formato binário simples, com cabeçalhos de tamanho fixo seguido por opções no formato "Type-Lengh-Value"(TLV) e do payload utilizado pela aplicação para transportar os dados. O numero de opções é determinado pelo cabeçalho.



Os campos do cabeçalho são definidos:

- ***Versão (Ver):*** 2-bit inteiros. Indica a versão do CoAP.
- ***Tipo(T):*** 2-bit inteiros. Indica o tipo da mensagem (CON, NON,ACK,RST).
- ***Token Length (TKL***):4-bit inteiros. Indica o número de opções do cabeçalho. MUST NOT é enviada, e MUST é processada como uma mensagem de erro.
- ***Code***: 8-bit inteiros, indica se a mensagem transporta um pedido, resposta ou se está vazia.
- ***Message ID***:16-bits inteiros, usados para deteção de duplicações e combinar com mensagens ACK ou RST.
- O ***token*** pode ter 0 a 8 bytes, utilizado para correlacionar pedidos e respostas. Alem disto, o Cabeçalho e o Token podem ser seguidos por mais opções, sendo estas opções seguidas até ao final da mensagem, por outras opções.
- ***Payload***: Se o tamanho do payload for diferente de zero, este tem que ser precedido por um marcador fixo de um byte (com o valor 0xFF) que indica o fim de opções e o início da carga útil. O *payload* estende-se até ao final do datagrama UDP, ou seja, o comprimento da carga útil é calculada a partir do tamanho do datagrama. A ausência do marcador do *payload* indica uma carga útil com comprimento zero, indicando que deve ter existido um erro é por isso deve ser processada como um erro da mensagem.
-

TRANSMISSÃO DE MENSAGENS

As mensagens CoAP são trocadas de forma assíncrona entre os endpoints CoAp. São usados para transportar solicitações e respostas CoAP, como o CoAP está vinculado ao UDP – "transportes pouco confiáveis", pode acontecer que as mensagens cheguem fora de ordem, apareçam duplicadas ou até mesmo desapareçam. Entao o CoAP para minimizar tais eventos, implementa um mecanismo de confiabilidade, sem tentar recriar o conjunto completo de recursos de protocolo de transporte como o TCP. Este mecanismo engloba as seguintes funcionalidades:

- Simples retransmissão stop-and-wait com back-off exponencial para mensagens Confirmable(CON).
- Deteção de Duplicações para mensagens Confirmable(CON) e Non-Confirmable(NON).

Com toda esta contextualização do protocolo CoAP, já é possível responder ao tema a que este artigo se propõe.

Portanto o CoAP é um protocolo de transferência web especializado para uso com nós restritos/limitados em redes. O protocolo foi projetado para M2M(máquina-to-máquina). O modelo de interação do CoAP é semelhante ao modelo cliente/servido do protocolo HTTP. No entanto, as iterações M2M(máquina-to-máquina) normalmente resultam numa implementação do CoAP que atua nas funções do cliente e do servido, como já demonstrado na contextualização do protocolo.

Além disto, um pedido (GET) CoAP é semelhante ou até mesmo equivalente ao do HTTP e é enviado por um cliente com a finalidade de pedir uma ação, usando um Codigo de Metodo num URI num servidor. O servidor envia uma resposta com um codigo, sendo que esta resposta pode incluir uma representação de recursos.

Mas então em que diverge este protocolo do protocolo HTTP, bem ao contrário do HTTP, o CoAP lida com essas transmissões de forma assíncrona ao longo dum transporte orientado a datagramas, como já referido o UDP. Isso é feito logicamente usando uma camada de mensagens que suporta uma confiabilidade opcional (back-off exponencial). O protocolo CoAP na arquitetura TCP/IP, opera entre as camadas de **Aplicação** e **Transporte**, criando ali duas subcamadas:

- Pedidos
- Respostas
- Mensagem

DIFERENCIAÇÃO ENTRE O PROTOCOLO COAP E O PROTOCOLO HTTP

Essencialmente o HTTP é principalmente usado para visualizar páginas Web. Já o CoAP é uma versão simplificada do HTTP para **IoT** ou **WSNs**. Embora o CoAP seja baseado no UDP, ele deve ter mensagens ACK para emular o TCP. Visto que o CoAP é mais simples do que o protocolo HTTP, este terá uma menor latência e consumirá menos energia. (Isto pode mudar com a próximas versões HTTP, que terão uma compactação de cabeçalhos, trazendo assim semelhanças benéficas ao HTTP, relativamente ao CoAP).

Alem disso o CoAP nunca teve a intenção de substituir o HTTP, mesmo que pareça, tais semelhanças devem-se ao fato do CoAP e do HTTP seguirem a arquitetura REST (Transferência Representacional de Estado), o que facilita a operação de *proxies* por meio de HTTP. O CoAP destina-se a uma camada de aplicativo para o dispositivo e, mais especificamente, foi projetado para dispositivos Limitados.

Como já referido o CoAP foi projetado para ambientes IoT e M2M, isto é, para enviar mensagens curtas usando UDP. Por exemplo:

Numa típica troca do protocolo CoAp consiste basicamente em duas mensagens, um pedido (*Request*) e uma resposta (*Response*). Por outro lado, uma solicitação HTTP primeiro exige que o cliente estabeleça uma conexão TCP e depois a encerre. Isso resulta minimamente em 9 mensagens para apenas uma solicitação. Claro que isto pode não ser verdadeiro, quando se

trata de cargas uteis grandes. Após a fase de início lento do TCP, ele é capaz de enviar vários pacotes ao mesmo tempo e reconhecer todos eles com um único reconhecimento. A transferência em bloco do CoAP, no entanto, requer um reconhecimento para cada bloco e leva mais mensagens e maior tempo de transferência. À prior são esperadas mensagens curtas do CoAP, por isso não é de grande interesse. No entanto o mecanismo *blockwise* do CoAP permite que um servidor limitado/restrito não receba apenas, mas também processe uma grande solicitação, bloco por bloco. Isso não seria possível se usássemos HTTP e TCP.

Enquanto que o protocolo HTTP foi projetado principalmente para dispositivos da Internet em que o poder e outras limitações não são algo a ter em conta. Além disso, é correto afirmar que o HTTP acaba por ser mais confiável que o protocolo em estudo, pois, como já referido enumeras vezes, este usa TCP/IP.

CONCLUSÃO

Após a leitura de grande parte do RFC 7252, referente ao protocolo CoAP foi possível ter uma ideia geral do mesmo, como ele funcionada, como são as suas características e ainda o que difere do protocolo HTTP.

Portanto, de uma forma geral podemos concluir que este se parece bastante com o HTTP. Principalmente no tipo de operações possíveis de serem efetuadas, isto penso que se deva ao fato de ambos compartilharem o modelo REST, facilmente se pode conectar ambos através de proxys.

Por fim o CoAP difere do HTTP nos seguintes princípios:

- Comunicação – usa UDP
- Camada de rede – Usa IPV6 com 6LoWPAN.
- Suporta *Multicast*
- Modelo de arquitetura – usa tanto modelo cliente-servidor como *publisher-subscribe*;
- Não necessita de comunicação síncrona
- *Overhead* – menos *Overhead* e mais simples
- Desenhado para redes limitadas, como WSN/IoT/M2M.

BIBLIOGRAFIA

Lanter, M. (2013). *Scalability for IoT Cloud Services.* Zurich: Distributed Systems.

Z.Shelby, K.Hartke, & Bormann, C. (Julho de 2014). Internet Engineering Task Force (IETF). *The Constrained Application Protocol (CoAP)*, p. 112.