

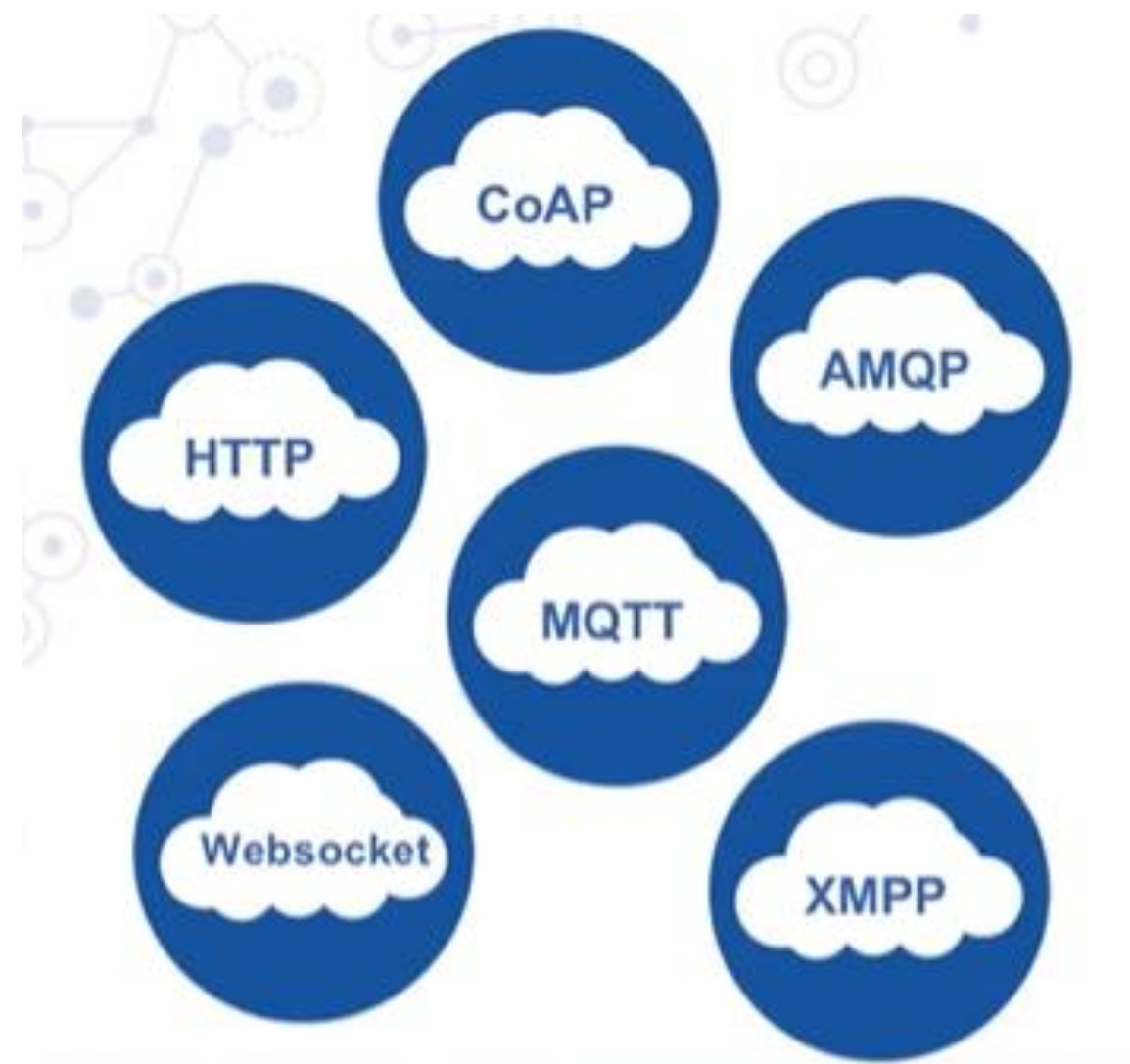
An aerial photograph of the Chicago skyline, showing a dense cluster of skyscrapers. The Lake Michigan is visible on the left side of the image. The sky is blue with some light clouds. The text "Internet Of Things" and "IoT" is overlaid in white on the right side of the image.

Internet Of Things IoT

UNIPLAC
Fernanda Forbici

Protocolos de Dados

- Assim como as tecnologias, não existe um protocolo de dados que atenda a todos os requerimentos possíveis em projetos IoT.
- Principais opções para o envio e recebimento de dados
 - HTTP
 - CoAP
 - Websocket
 - MQTT
 - AMQP, XMPP, STOMP, SSL, etc



Protocolos de Dados Modelo cliente/servidor

- A popularização da Internet está relacionada com a adoção do modelo cliente/servidor em aplicações.
- O padrão Cliente/Servidor aproveita a maior capacidade computacional dos servidores e busca reduzir a carga de trabalho dos clientes. Para isso, esse modelo estabelece uma relação desigual entre dispositivos, em que o cliente pode ter menor poder de processamento e o servidor fica encarregado de processar uma quantidade maior de informações, além de ser responsável por interligar outros dispositivos e serviços.

Protocolos de Dados

Modelo cliente/servidor

Vantagens para IoT


- Esse modelo se encaixa bem às restrições computacionais das “Coisas” e direciona o processamento das informações para um servidor central, que pode estar na nuvem. Além disso, cada dispositivo conecta-se diretamente com um ou mais servidores de confiança.

Desvantagens para IoT


- Uma arquitetura IoT deve ter a capacidade de escalar para centenas e até milhares de dispositivos. Para que isso seja possível, o servidor deve ser capaz de estabelecer muitas conexões simultaneamente.


Protocolos de Dados Modelo cliente/servidor

- Na Internet, é muito comum utilizar APIs (*Application Programming Interfaces*) REST como ponto de conexão entre clientes e servidores. APIs funcionam como uma camada de abstração para acessar um serviço em um servidor. Para facilitar a comunicação entre dispositivos, uma API define um conjunto de parâmetros que serão interpretados posteriormente.
- Exemplos: HTTP, HTTPS e CoAP


A large orange circle on the left side of the slide, partially cut off by the edge.


Protocolo de Dados Modelo Publish/Subscribe

- A maioria dos protocolos de dados funcionam no padrão publish/subscribe.
 - Este padrão estabelece uma entidade central chamada Broker que recebe mensagens dos Publishers e as reencaminha para os Subscribers cadastrados.
- 
- A blue dashed line in the bottom right corner, consisting of several short, curved segments.

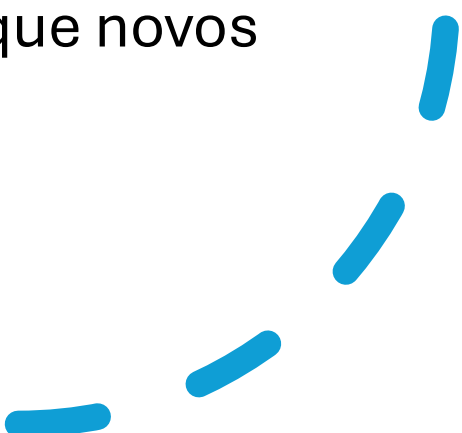
A large orange circle on the left side of the slide, partially cut off by the edge.

Protocolo de Dados Modelo Publish/Subscribe

- Um dispositivo que envia dados não precisa estabelecer uma conexão direta com os dispositivos, aplicativos ou serviços que irão utilizar essas informações posteriormente.
 - O Broker fica responsável por estabelecer a conexão com todos que irão gerar dados e/ou consumi-los.
 - Graças a heterogeneidade dos elementos a independência entre os Publishers e Subscribers favorece a adoção desse padrão na IoT
 - Ex: Websocket, MQTT, AMQP
- 
- A decorative blue curved line in the bottom right corner of the slide.

A large orange circle on the left side of the slide, partially cut off by the edge.

Protocolo de Dados Modelo Publish/Subscribe

- Em um projeto IoT:
 - **Broker** pode ser uma das funções do Gateway ou do servidor na nuvem
 - **Publishers** são as Coisas que se conectam ao Broker para enviar seus dados
 - **Subscribers** são dispositivos, aplicativos ou serviços interessados em um determinado tipo de dado. Eles se conectam aos Brokers para serem informados sempre que novos dados são recebidos.
- 
- Four blue curved lines of varying lengths and orientations, located in the bottom right corner of the slide.

Exemplos de Protocolos

MQTT (Message Queuing Telemetry Transport)

- Protocolo leve baseado em Publish/Subscribe, projetado para comunicação eficiente entre dispositivos limitados e redes instáveis.
- **Características principais:**
 - Leve e baixo consumo de energia.
 - Baixa latência e pequeno tamanho dos pacotes.
 - Utiliza modelo Publish/Subscribe com um servidor central (Broker).
- **Exemplos de uso:**
 - Telemetria em sensores ambientais (temperatura, umidade, etc.).
 - Monitoramento de frotas (rastreamento GPS em tempo real).
- **Caso típico:**
 - Uma rede de sensores distribuídos em uma fazenda enviando dados periodicamente para um servidor central (broker MQTT), que redistribui esses dados a dashboards e sistemas de análise.
- Plataformas/Exemplos: Mosquitto, AWS IoT Core, IBM Watson IoT.

Exemplos de Protocolos

HTTP/HTTPS (Hypertext Transfer Protocol)

- Protocolo tradicional baseado no modelo Cliente/Servidor, amplamente utilizado para APIs RESTful.
- **Características principais:**
 - Facilidade de integração com plataformas web.
 - Protocolo baseado em requisições/respostas.
 - Maior consumo de banda e energia em comparação ao MQTT e CoAP.
- **Exemplos de uso:**
 - Comunicação de dispositivos IoT com APIs externas.
 - Integração de dados IoT em aplicações web ou mobile.
- **Caso típico:**
 - Um sensor de qualidade do ar envia dados periodicamente via HTTP POST para uma API REST, que armazena os dados em um banco de dados central.
- Plataformas/Exemplos: REST APIs, Flask, Node.js.

Exemplos de Protocolos

CoAP (Constrained Application Protocol)

- Protocolo projetado especificamente para dispositivos com restrições severas (energia, processamento, memória) utilizando o paradigma REST sobre UDP.
- **Características principais:**
 - Muito leve, projetado para redes restritas.
 - Comunicação assíncrona com baixa latência.
 - Baixo overhead na transmissão.
- **Exemplos de uso:**
 - Dispositivos alimentados por bateria e redes sensoriais.
 - Aplicações domésticas ou industriais onde o consumo energético é crítico.
- **Caso típico:**
 - Dispositivos vestíveis ou sensores médicos que precisam preservar ao máximo a bateria, comunicando pequenas quantidades de dados periodicamente.
- Plataformas/Exemplos: Implementações em Contiki OS, Californium.

Exemplos de Protocolos

AMQP (Advanced Message Queuing Protocol)

- Protocolo aberto que permite comunicação assíncrona e segura entre aplicações distribuídas.
- **Características principais:**
 - Modelo avançado de filas, tópicos e gerenciamento robusto de mensagens.
 - Alta confiabilidade, segurança e suporte a múltiplos modelos de comunicação.
- **Exemplos de uso:**
 - Aplicações IoT industriais complexas.
 - Comunicações críticas e sistemas financeiros conectados com IoT.
- **Caso típico:**
 - Uma linha de montagem industrial com diversos sensores e dispositivos conectados, enviando dados críticos continuamente para sistemas analíticos avançados e ERP.
- Plataformas/Exemplos: Apache Qpid, RabbitMQ.

Exemplos de Protocolos

WebSocket

- Protocolo que permite comunicação bidirecional em tempo real entre cliente e servidor através de uma única conexão TCP.
- **Características principais:**
 - Conexão persistente (reduz a latência de comunicação).
 - Comunicação em tempo real, ideal para monitoramento contínuo.
- **Exemplos de uso:**
 - Monitoramento em tempo real de dispositivos IoT em dashboards web.
 - Aplicações interativas e notificações instantâneas.
- **Caso típico:**
 - Monitoramento contínuo de dados ambientais em tempo real num painel de controle online.
- Plataformas/Exemplos: Socket.IO, Node.js.

Exemplos de Protocolos

XMPP (Extensible Messaging and Presence Protocol)

- Protocolo aberto baseado em XML que facilita comunicação em tempo real, originalmente desenvolvido para mensageria instantânea.
- **Características principais:**
 - Comunicação em tempo real com presença integrada.
 - Extensível, permitindo adaptações para necessidades específicas.
- **Exemplos de uso:**
 - Aplicações IoT com necessidade de comunicação instantânea.
 - Gestão de dispositivos inteligentes, onde a presença do dispositivo precisa ser conhecida imediatamente.
- **Caso típico:**
 - Automação residencial inteligente onde diversos dispositivos precisam enviar e receber comandos instantaneamente.
- Plataformas/Exemplos: Ejabberd, Openfire

Tabela Comparativa

Protocolo	Modelo	Uso principal	Eficiência energética	Latência
MQTT	Publish/Subscribe	Telemetria, Sensores	Muito Alta	Baixa
HTTP	Cliente/Servidor (REST)	Integrações Web e APIs	Moderada	Moderada
CoAP	Cliente/Servidor (REST)	Redes restritas, sensores limitados	Muito Alta	Muito Baixa
AMQP	Publish/Subscribe e filas	Comunicação crítica e robusta	Média	Baixa
WebSocket	Cliente/Servidor	Comunicação em tempo real	Média	Muito Baixa
XMPP	Cliente/Servidor	Comunicação em tempo real e presença	Média	Baixa

Tabela Comparativa

Característica	HTTP/HTTPS	MQTT	CoAP
Modelo	Cliente/Servidor	Publish/Subscribe	Cliente/Servidor (RESTful)
Consumo de energia	Alto	Baixo	Muito Baixo
Latência	Moderada/Alta	Baixa	Muito Baixa
Tamanho de pacote	Alto (texto)	Baixo (binário)	Muito baixo (binário)
Confiabilidade	Alta (TCP)	Alta (TCP)	Média (UDP)
Casos de Uso	Aplicações web/API	Telemetria, tempo real	Dispositivos com restrição energética

Boas Práticas de Segurança em Protocolos IoT

- **MQTT com TLS (Transport Layer Security):**
 - Sempre utilize comunicação criptografada com TLS para proteger dados sensíveis.
 - Autenticação baseada em usuário/senha e certificados digitais para autenticar dispositivos.
- **HTTP/HTTPS:**
 - Utilize HTTPS ao invés de HTTP, garantindo comunicação segura com APIs REST.
 - Implementação adequada de autenticação (OAuth2 ou tokens JWT).
- **Controle de Acesso:**
 - Restringir acesso ao Broker MQTT por endereços IP e credenciais específicas.
 - Definir tópicos MQTT específicos e restritos.
- **Atualização Contínua:**
 - Manter firmware e software atualizados para prevenir vulnerabilidades.

Pesquisa

- 
- Sigfox
 - ZigBee
 - STOMP