BITS Pilani presentation

BITS Pilani
Pilani Campus

Paramananda Barik
CS&IS Department

innovate achieve lead

BITS Pilani

Pilani Campus

SEZG586/SSZG586, Business Model and Security

Lecture No.11

Business Model – Cloud Computing

Business Model of Cloud computing – Simple, straight forward

- Users directly purchase service from the service provider
- Access it over Internet
- services could be IT infrastructure, software, and other resources

Business Model – Edge Computing

The business model of Edge computing

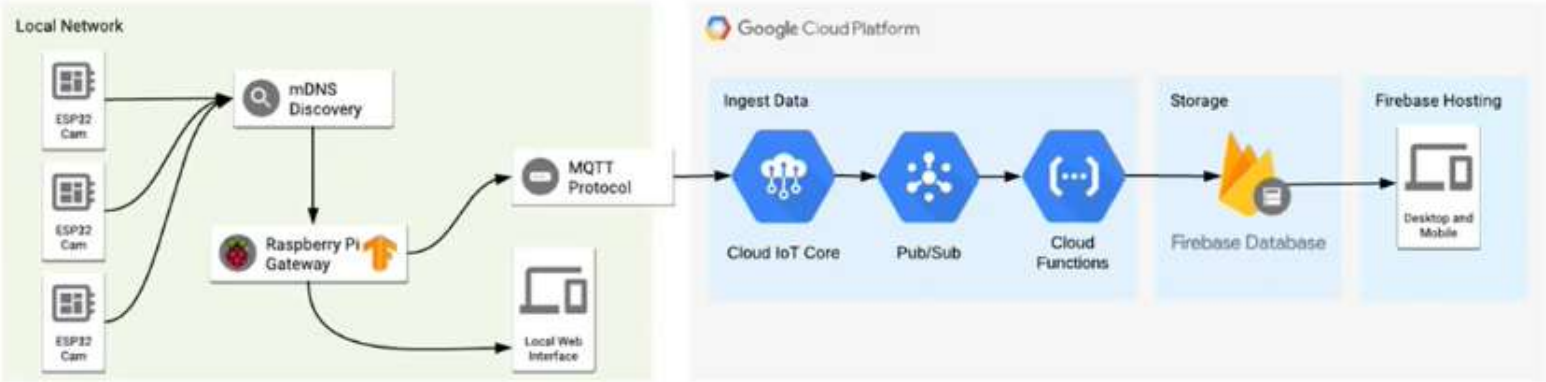
- -driven-by user demanding services
- -driven-by data

Edge computing is spread over multiple domains:

- Information Technology
- Communication Technology
- Industrial chain containing:
  - Software/hardware platform
  - Internet
  - Data aggregation
  - Chip, sensor, and industrial applications

Business Model of Edge Computing

Individual user will request data from the data owner, then cloud center or Edge data owner will feedback the processed result to the user



Optimization Metrics

To choose an optimal allocation strategy

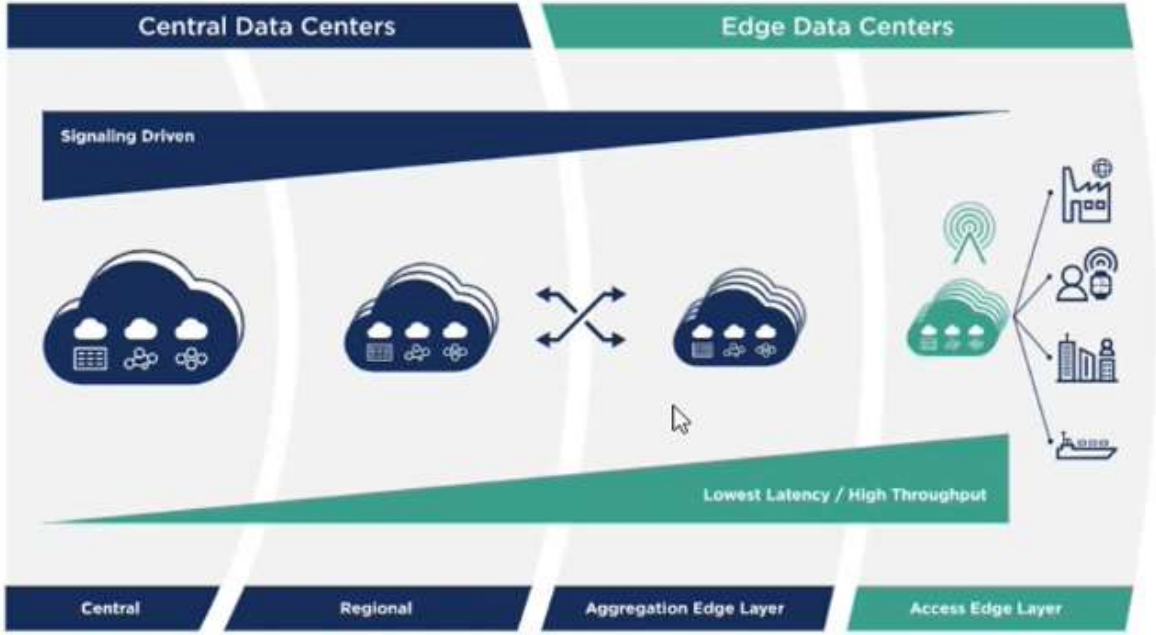
optimization metrics

- Latency
- Bandwidth
- Energy
- Cost

Optimization Metrics : Latency

Servers in Cloud computing provide ***high computation capability***

Is latency determined only by computation time??



Optimization Metrics : Bandwidth

How to reduce latency??

High bandwidth wireless access - to send data to the edge

Less use of bandwidth between edge and cloud

Ex: smart home

All the data handled in the home gateway through Wi-Fi

Provides transmission reliability

Ex: smart city

Can we handle all the data in the Edge?

Optimization Metrics : Energy

Battery is the most important resource for things at the Edge of the network.

For the endpoint layer, offloading workload to the edge can be treated as an energy free method

Is it energy efficient to offload the whole workload (or part of it) to the edge rather than compute locally?

(Will be discussed through a presentation of a published paper in the next webinar)

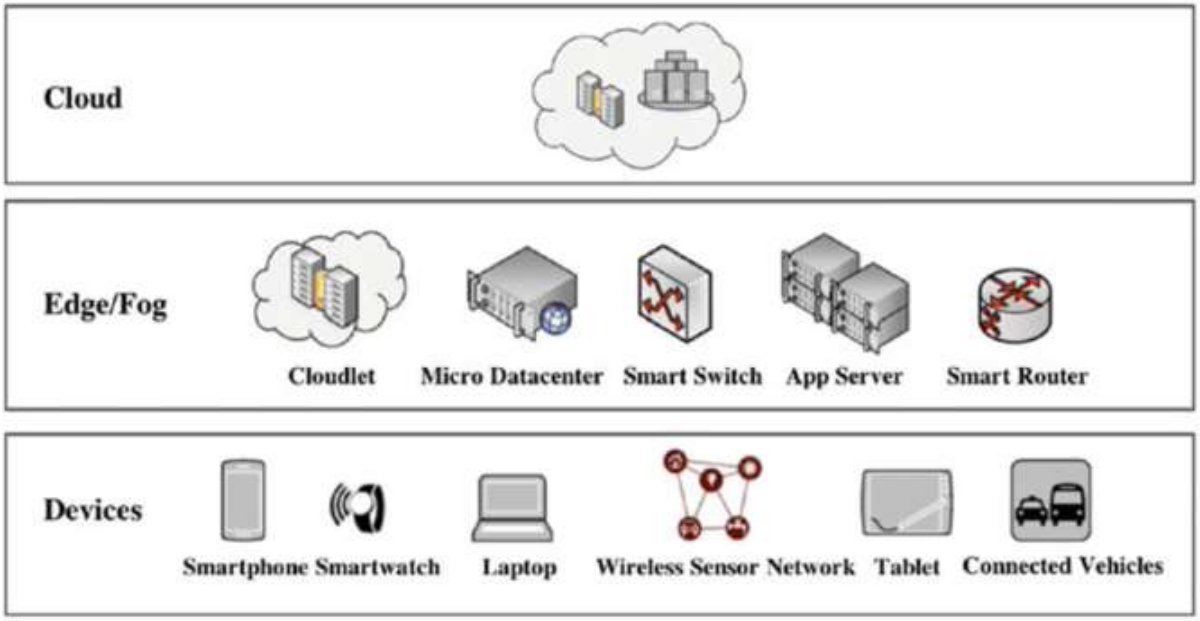
Optimization Metrics : Energy

New layered cost model approach is needed

Optimization Metrics : Tools

Two significant roles in edge computing:

- Provider
- Developer



Provider Vs. Developer

Responsibility of a provider:

- Resource virtualization
- Provisioning
- Managing resources: transparent to users

Developers are interested in:

- What kind of edge devices can be leveraged by their applications
- What tools or data processing platforms are the best fit to their applications
- How to develop, test, and deploy their applications

Security Threats

1) Weak Computation Power
2) Attack Unawareness
3) OS and Protocol Heterogeneities
4) Coarse-Grained Access Control

Mirai virus - 65 000 IoT devices - first 20 h - August 2016 - few days later - turned into botnets - launch distributed denial of service (DDoS) - shutting down over 178 000 domains

IoTReaper and Hajime – infect more than 378 million IoT devices in 2017

Attack Unawareness

General purpose computers have UI

IoT devices do not have user interfaces (UIs)

Therefore, a user may have limited knowledge about the running status of a device

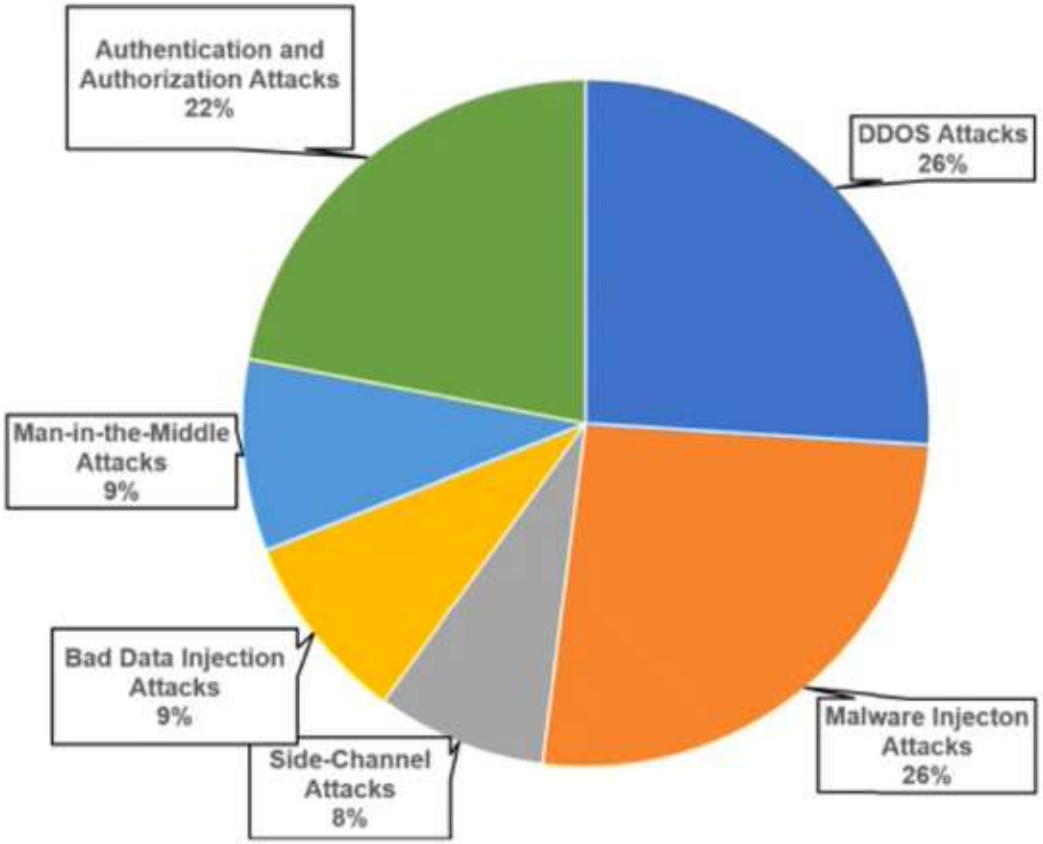
OS and Protocol Heterogeities

General purpose computers use:

- Standard Os
- Communication protocols such as POSIX

Edge devices have different OSes and protocols without a standardized regulation

Types of attacks targeting edge computing infrastructure



Security Threats of Edge Computing

Threats mainly due to
- the design flaws
- misconfigurations
- implementation bugs

Defense mechanisms
- Detection based
- Prevention based

DDoS Attacks and Defense Mechanism

DDoS - type of cyberattack in which attackers aim to disrupt normal services provided by one or more servers based on distributed resources such as a cluster of compromised edge devices

Edge servers are more susceptible to DDoS attacks since they are relatively computationally less powerful to maintain strong defense systems as cloud servers do

Flooding-based Attack Mechnism

UDP flooding attack, an attacker continuously sends a large amount of ***noisy UDP packets*** to a target edge server

ICMP flooding attack, an attacker exploits the ICMP protocol to attack by sending a large number of ICMP Echo Request packets to a target edge server as ***fast*** as possible ***without waiting for the replies***. This type of attack consumes both outgoing and incoming throughputs of the victim server

SYN flooding attack, an attacker exploits the three-way handshake of the transmission control protocol (TCP) by initiating a huge amount of SYN requests with a spoofed IP address to a target edge server

Flooding-based Attack Mechanism

PoD attack, an attacker creates an IP packet with malformed or malicious content that has a length greatly larger than the maximum frame size for a standard IP packet (65 535 bytes) and splits the long IP packet into multiple fragments and sends them to a target server.

HTTP flooding attack, an attacker simply sends a large amount of HTTP GET, POST, PUT, or other legitimate requests to an edge server

Slowloris attack, an attacker creates numerous partial HTTP connections which can be realized by only sending HTTP headers to a target server but never completing one

Zero-day DDoS Attack

A zero-day DDOS attack

advanced than flooding-based DDoS

but it is more difficult to implement

attacker must find an unknown vulnerability (i.e., zero-day vulnerability) in a piece of code running on the target edge server/device

which can cause memory corruption and finally result in a service shutdown

Common vulnerabilities and exposures (CVE)- 2010-3972

Heap-based overflow that can cause a DoS on Internet Information Services (IIS) 7.0 and IIS 7.5

Defense Solution

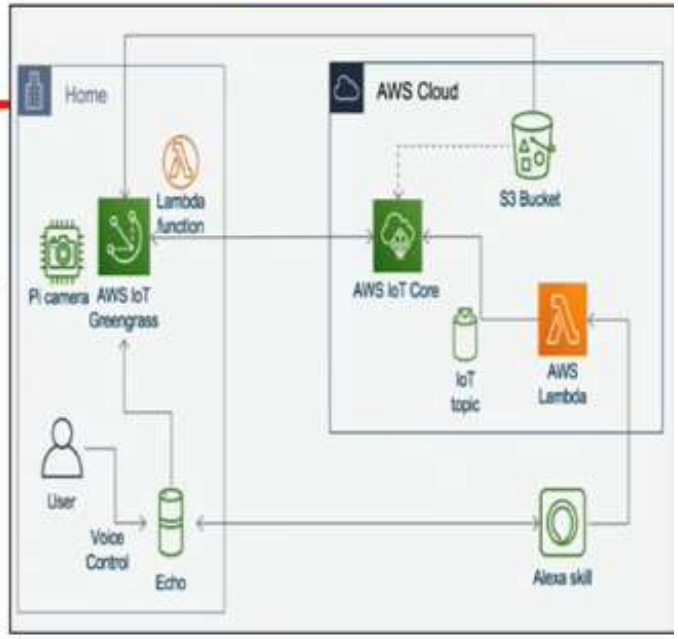
- Root cause
  - Flooding-based attacks
    - Protocol-level design flaws/vulnerabilities within the network communication protocols
  - Zero-day attacks
    - Code-level vulnerabilities

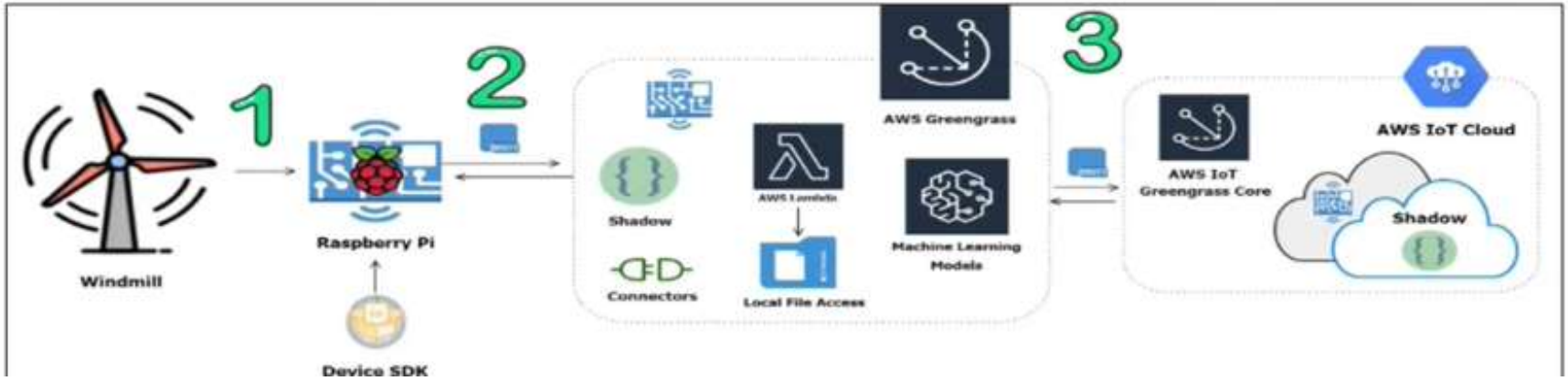
Features of Kubernetes

- Automated Scheduling
- Self-Healing Capabilities
- Automated rollouts & rollback
- Horizontal Scaling & Load Balancing
- Offers environment consistency for development, testing, and production
- Infrastructure is loosely coupled to each component can act as a separate unit
- Provides a higher density of resource utilization
- Offers enterprise-ready features
- Application-centric management
- Auto-scalable infrastructure

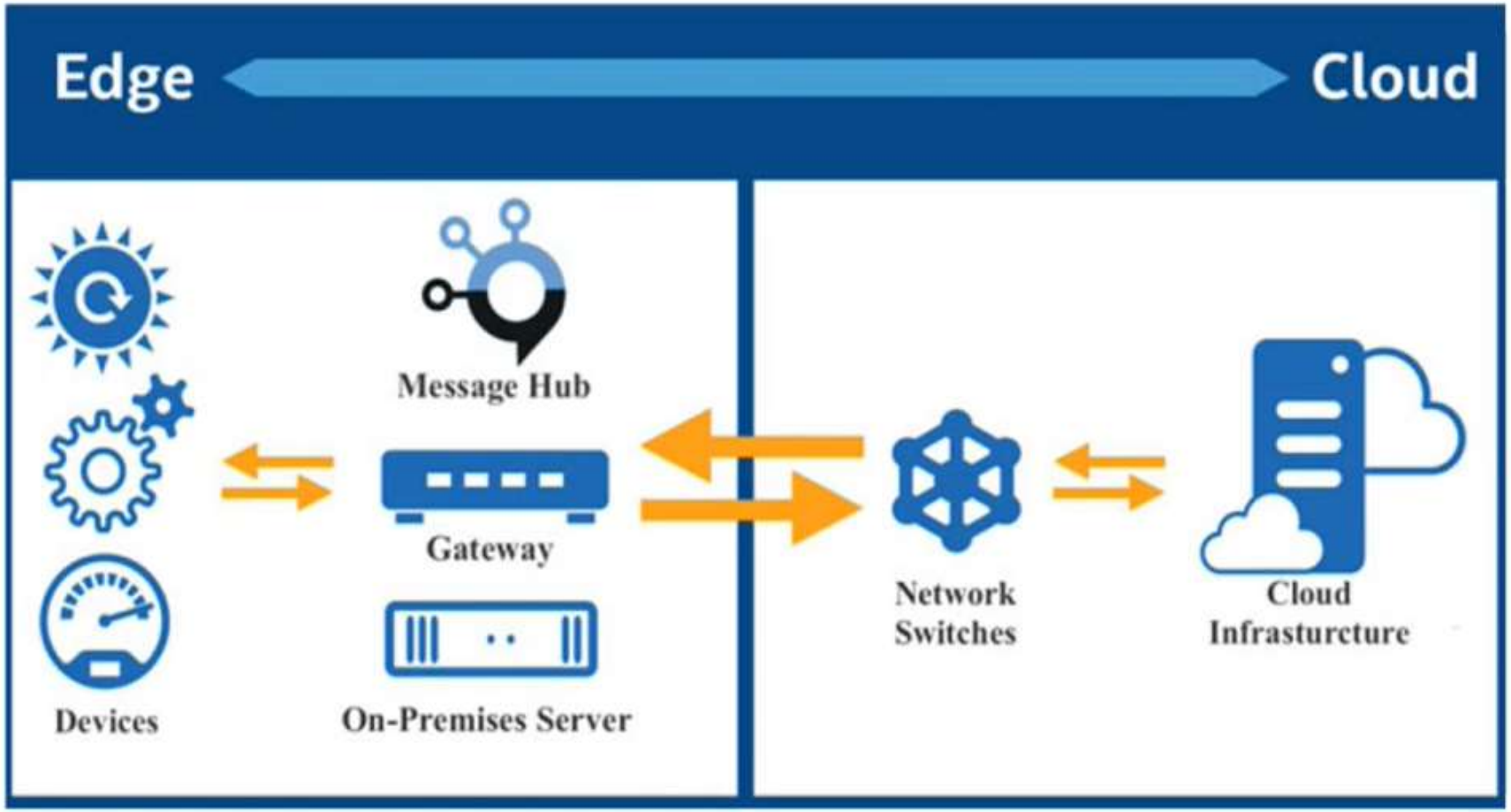
Developing Platforms for Edge Computing

Edge Analytics





IT Architecture for edge analytics



Messaging Protocols

Message Queuing Telemetry Transport (MQTT)

Constrained Application Protocol (CoAP)

Advanced Message Queuing Protocol (AMQP)

CoAP – Constrained Application Protocol

CoAP - designed to translate the HTTP model

- used in restrictive device and network environments
- relies on the User Datagram Protocol (UDP)
- allows broadcasting and multicasting
- supports a request/response interaction model

- QoS - messages sent with mark - 'confirmable' or 'nonconfirmable'
- indicates whether the recipient should return an 'ack' or not.

CoAP – Constrained Application Protocol

It is a specialized web transfer protocol developed for the following:

- Constrained nodes: The nodes often have 8-bit microcontrollers with small amounts of ROM and RAM
- Constrained (e.g., low-power, lossy) networks: such as IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs) often have high packet error rates and a typical throughput of 10s of kbit/s

The protocol is designed for machine- to-machine (M2M) applications such as smart energy and building automation.

It provides a request/response interaction model between application endpoints

It supports built-in discovery of services and resources

It is designed to easily interface with HTTP for integration with the Web

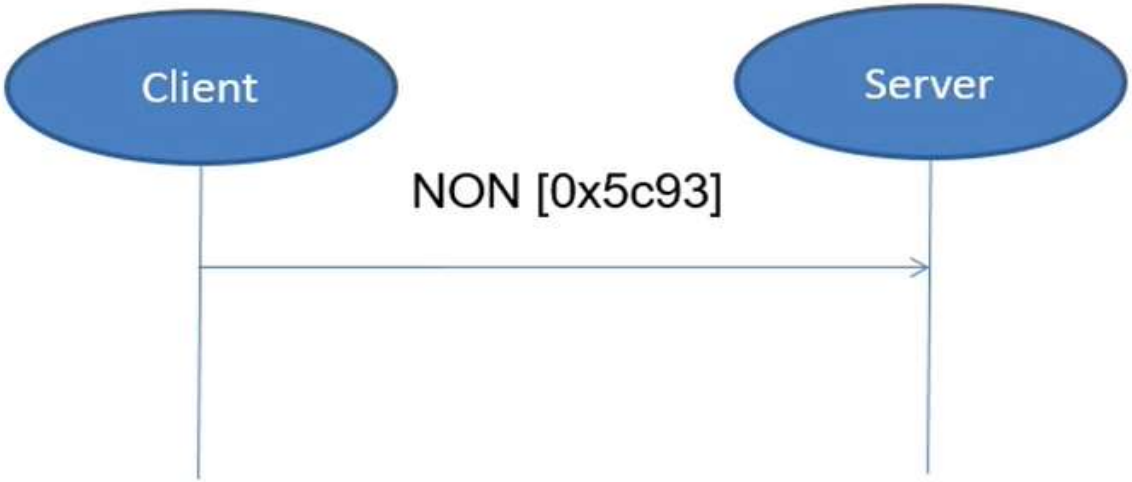
And meets the following requirements:

- Multicast support
- Very low overhead
- Simplicity in data transfer for constrained environments

Message Layer Model

Unreliable message transport:

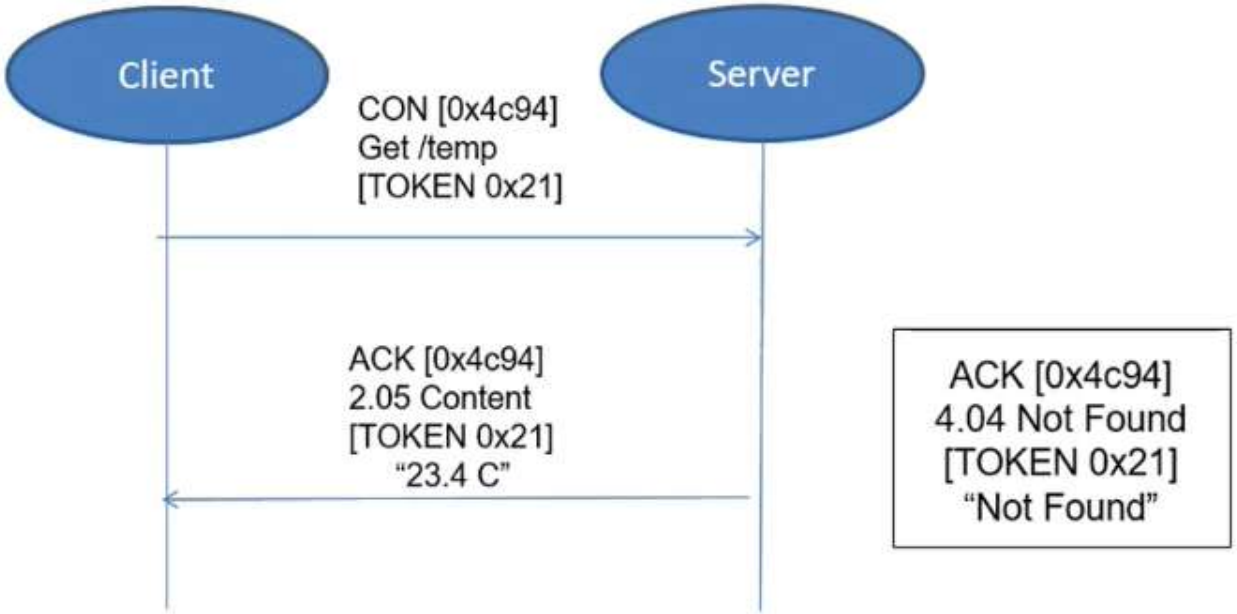
- Is used to transfer NON type message.
- It doesn't need to ACK, but has to contain message ID for supervising in case of retransmission.
- If recipient fail to process message, server replies RST



Request/Response Layered Model

Piggy-backed:

- Client sends request using CON type or NON type message and receives response ACK with confirmable message immediately.
- For successful response, ACK contain response message (identify by using token), for failure response, ACK contain failure response code.



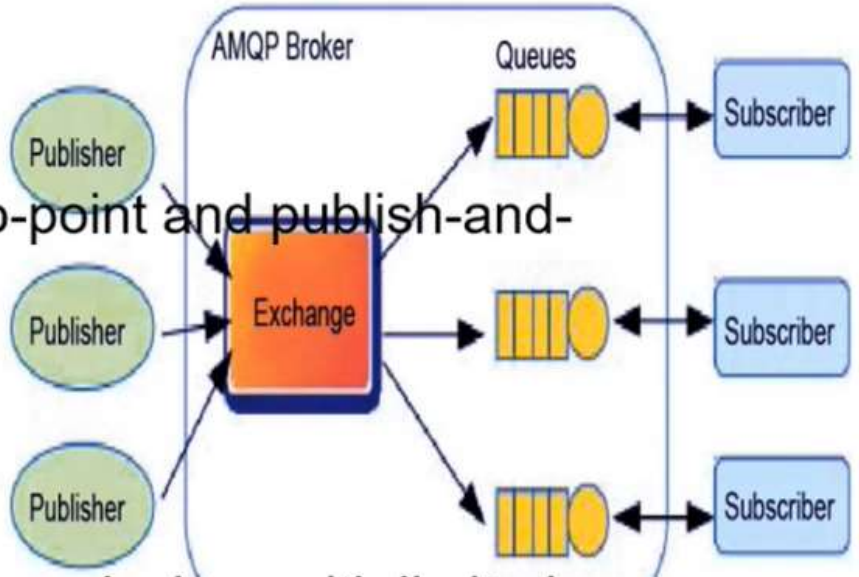
Advanced Message Queuing Protocol (AMQP)

AMQP is an open standard publish/subscribe type protocol

Developed mainly for financial services sector

features

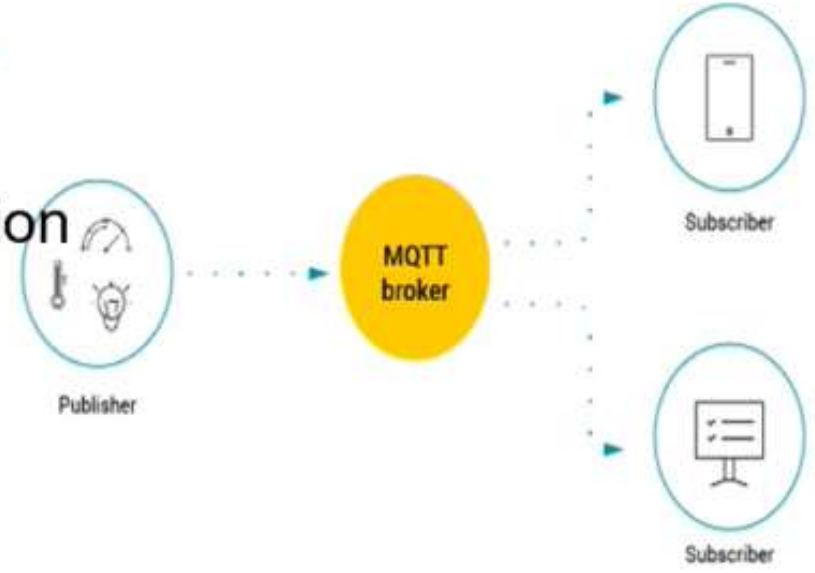
- message orientation
- queuing
- routing (including point-to-point and publish-and-subscribe)
- reliability
- secure

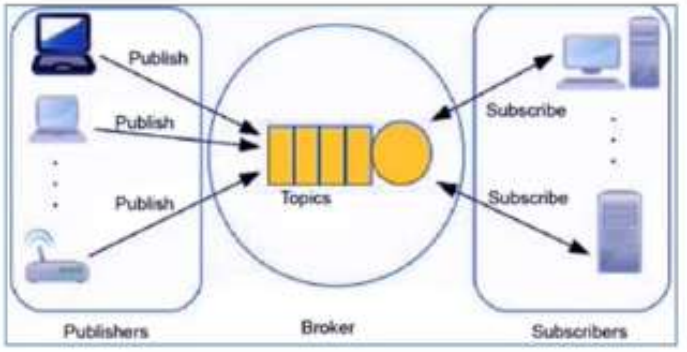


AMQP is not suitable for sensor devices with limited memory, power or network bandwidth - heavy

Message Queuing Telemetry Transport (MQTT)

- most widely adopted standard
- lightweight publication/subscription type (pub/sub)
- designed for battery-powered devices
- architecture is simple and lightweight
- consumes low power
- unreliable communication networks
- good for small-sized cheap low-power objects





Messaging Queueing Telemetry Protocol

MQTT is based on subscriber, publisher and broker model

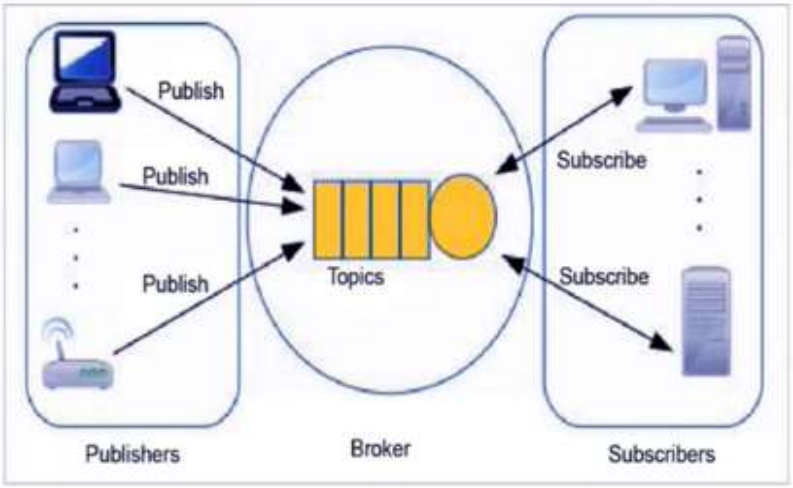
Broker - ensure security by cross-checking the authorization of publishers and subscribers

MQTT offers three modes:

QoS0 (At most once): The least reliable mode but also the fastest. The publication is sent but confirmation is not received.

QoS1 (At least once): Ensures that the message is delivered at least once, but duplicates may be received.

QoS2 (Exactly once): The most reliable mode while the most bandwidth-consuming. Duplicates are controlled to ensure that the message is delivered only once.



Sending Receiving Messages with MQTT

Option 1: Using mosquitto (Broker), mosquitto-clients (pub/sub)

Option 2: Using HBMQTT (consists of broker, subscriber, publisher)

Option 3: Using **Paho MQTT Python Client**

- Paho is an iot.eclipse.org project
- Eclipse Paho project provides open-source client implementations of MQTT and MQTT-SN messaging protocols
- http://www.eclipse.org/paho/
- http://www.eclipse.org/paho/downloads.php

Using Paho MQTT Python Client

Here python scripts will be used for implementation, so install the python library paho-mqtt.

'pip'/'pip3' is required to install this module

- $sudo apt-get install python-pip

Now you can install paho-mqtt:

- $sudo pip install paho-mqtt

Publisher example:

```
#!/usr/bin/env python
import paho.mqtt.client as mqtt

# This is the Publisher
client = mqtt.Client()
client.connect("localhost",1883,60)
client.publish("topic/test", "Hello world!");
client.disconnect();
```

Note: if an external broker is used, replace "localhost" with the IP address of the device that hosts the broker (iot.eclipse.org, AWS IOT)

Using Paho MQTT Python Client

Subscriber example:

```
#!/usr/bin/env python
import paho.mqtt.client as mqtt
# This is the Subscriber
def on_connect(client, userdata, flags, rc):
        print("Connected with result code "+str(rc))
        client.subscribe("topic/test")
def on_message(client, userdata, msg):
        if msg.payload.decode() == "Hello world!":
                print("Yes!")
                client.disconnect()
client = mqtt.Client()
client.connect("localhost",1883,60)
client.on_connect = on_connect
client.on_message = on_message
client.loop_forever()
```

Note: when the publisher sends a string as payload use decode() as in the example above. When the Publisher sends a number, you can use int(msg.payload).

Development Tools and Platforms

On-Device processing – What is the need?

TensorFlow

OpenCV

Apache Edgent

AWS Greengrass

Tensor Flow

- By Google
- *"TensorFlow Lite is an open-source deep learning framework for on-device inference."*
- Tensorflow Lite help developers to run TensorFlow models
  - Mobile
  - Embedded devices
  - IoT devices
  - It enables on-device machine learning inference with low latency and a small binary size.

Tensor Flow Lite

TensorFlow lite has two main components:

TensorFlow Lite Converter

convert the models into an efficient form for use by the interpreter

TensorFlow Lite Interpreter.

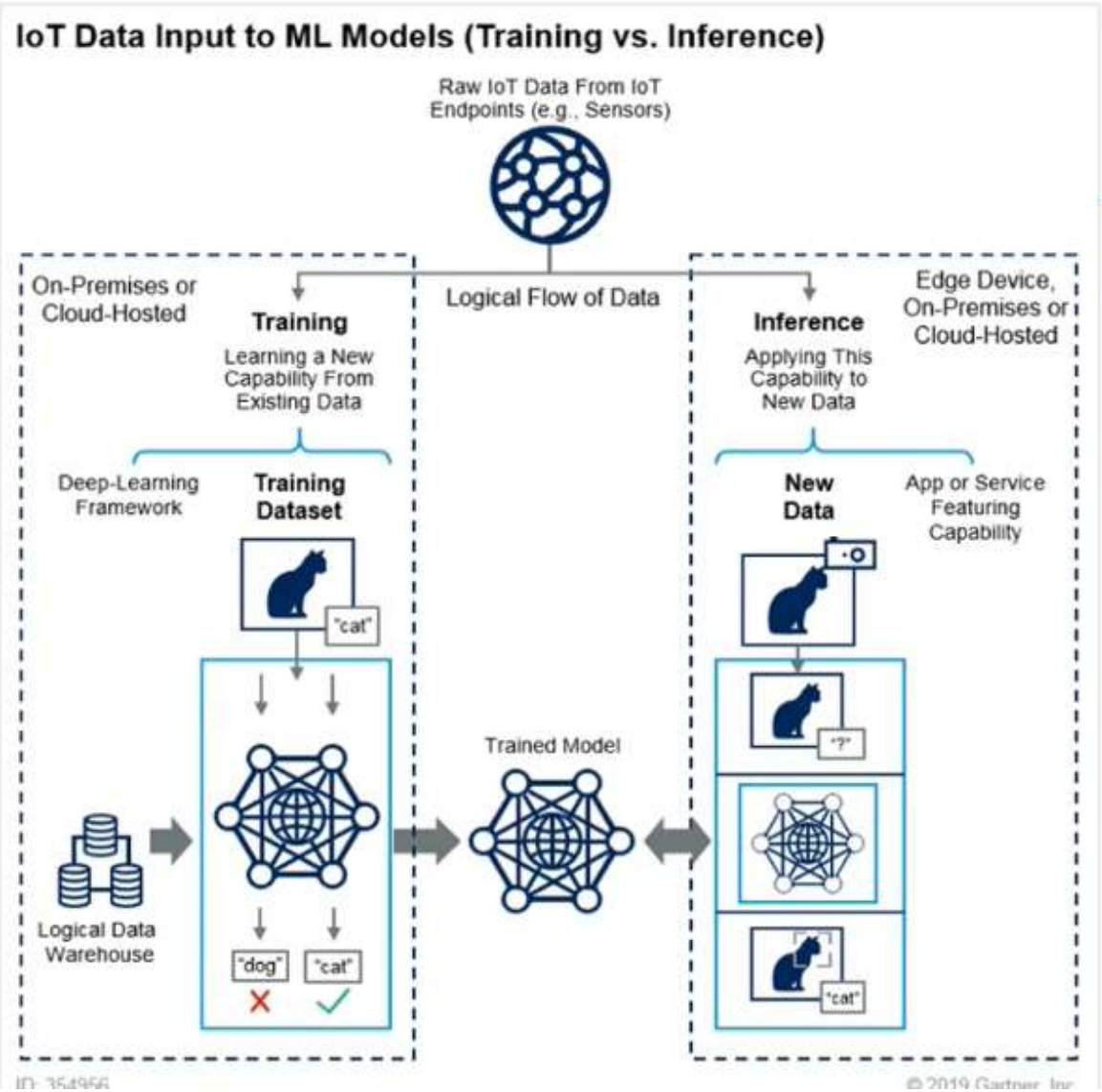
runs specially optimized models on many different hardware types

mobile phones

embedded Linux devices

microcontrollers.

IoT Data Input to ML Models (Training vs. Inference)



What is Micro Cloud?

Smaller

Managed clouds

Decentralized

Micro clouds are a new class of infrastructure for on-demand computing at the edge

A micro cloud is a small cluster of compute nodes with local storage and networking

How do I get a MicroCloud?

MicroK8s is the smallest, fastest, fully-conformant Kubernetes that tracks upstream releases and makes clustering trivial.

MicroK8s is great for offline development, prototyping, and testing.

Use it on a VM as a small, cheap, reliable k8s for CI/CD.

It's also the best production grade Kubernetes for appliances.

Develop IoT apps for k8s and deploy them to MicroK8s on Linux boxes.

Minikube Vs. K3s Vs. MocroK8s

	Minikube	K3s	MicroK8s
Developed by	Kubernetes project	Rancher	Canonical
Ease of installation	Very easy	Easy	Easy
Modularity	Low	Medium	High
Supports multi-node clusters	Yes, with ease	Yes, with effort	Yes, with ease
Supports production deployments	No	Yes	Yes
Supports IoT/edge	No	Yes	Yes
Supported operating systems	Linux, Windows, macOS	Linux, Windows, macOS	Linux, Windows, macOS

How do I get a MicroCloud?

What is LXD?

LXD is a next-generation system container and VM manager.

It gives a way to easily spin up and manage pre-made images for a range of Linux distributions over the network with a designed to be simple REST API.