

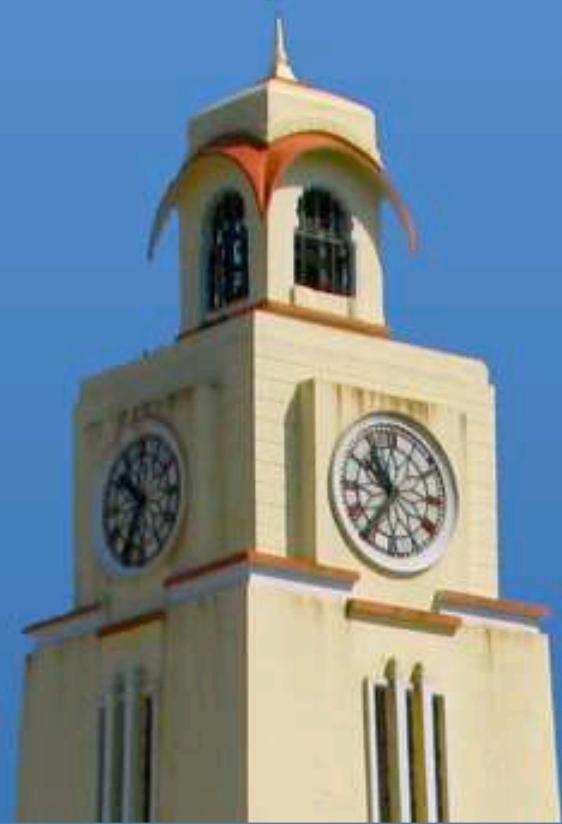


BITS Pilani
Pilani Campus

BITS Pilani presentation

Paramananda Barik
CS&IS Department





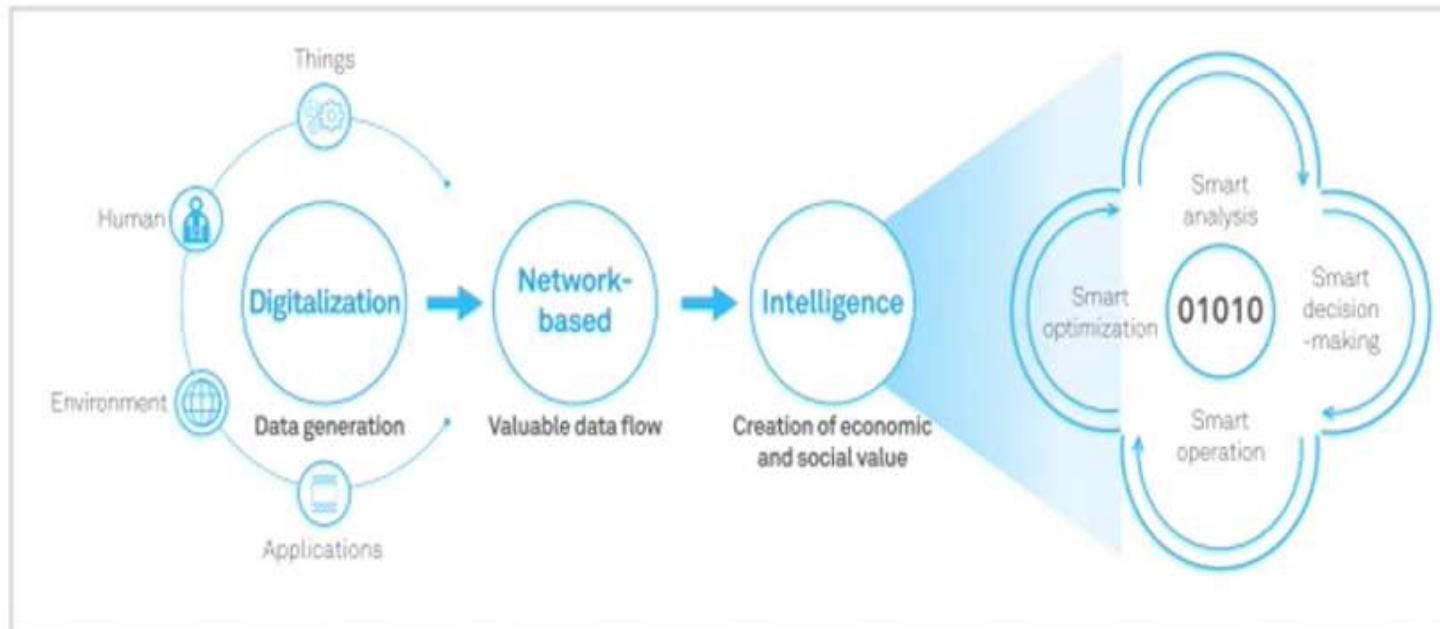
SEZG586/SSZG586, Edge Computing Lecture No.1

Evolution of Computing



Industry moves to use Intelligence?

Digital transformation of Industries





Digital Transformation

Intelligence Technologies

Big Data

Machine learning

Deep learning

Applications

Speech recognition – Smart assistance

Image recognition – AR/VR

User Profiling

Wearables

Connected Cars



Digital Transformation

Industries

Manufacturing

Power

Transportation

Healthcare

Agriculture



Intelligence in Industry

Industry intelligence is defined in two phases.

Phase 1: Oriented to the business process

Market leads

Marketing

Purchase

Logistics

After-sales



Intelligence in Industry

Technology support for Phase 1
Transformation and Communications Technology (ICT)

- Ubiquitous network connections
- Cloud computing
- Big Data mining and analytics



Intelligence in Industry

Phase II : Oriented to the production processing covering

- Product planning
- Designing
- Manufacturing
- Operation

Intelligence in Industry

Products, production equipment, and manufacturing process have already started to become digitalized and network-based.

This phase aims at:

- Improve agility and collaboration.
- Increase resources sharing and save energy.
- Reduce uncertainties in production and operation.

Edge Computing - definition

“Edge computing in telecom, often referred to as Mobile Edge Computing, MEC, or Multi-Access Edge Computing, provides execution resources (compute and storage) for applications with networking close to the end users, typically within or at the boundary of operator networks.” – **Ericsson**

“Edge computing is a distributed, open IT architecture that features de-centralised processing power, enabling mobile computing and Internet of Things (IoT) technologies. In edge computing, data is processed by the device itself or by a local computer or server, rather than being transmitted to a data centre.” – **HP**

“Edge computing is a distributed computing framework that brings enterprise applications closer to data sources such as IoT devices or local edge servers.” – **IBM**

“Edge computing is part of a distributed computing topology where information processing is located close to the edge, where things and people produce or consume that information.” – **Gartner**

“Edge computing is a distributed computing paradigm that brings computation and data storage closer to the sources of data. This is expected to improve response times and save bandwidth. "A common misconception is that edge and IoT are synonymous”” - Wikipedia

CLOUD



EDGE

Service delivery
Computing offload
IoT management
Storage & caching

Edge Node

Edge Node



Edge Computing



Cloud

Edge nodes

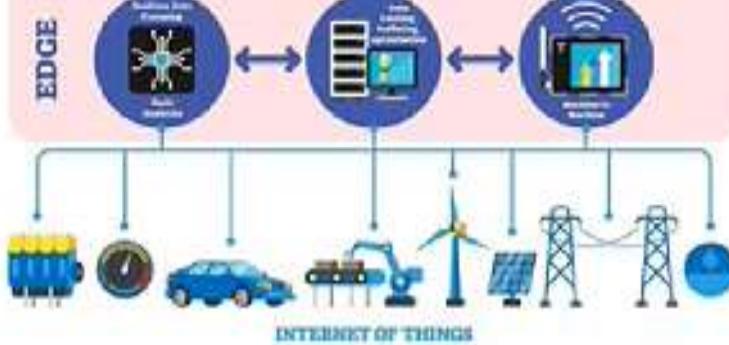
Edge devices



Edge Computing



EDGE



Before EDGE

Cloud Computing:

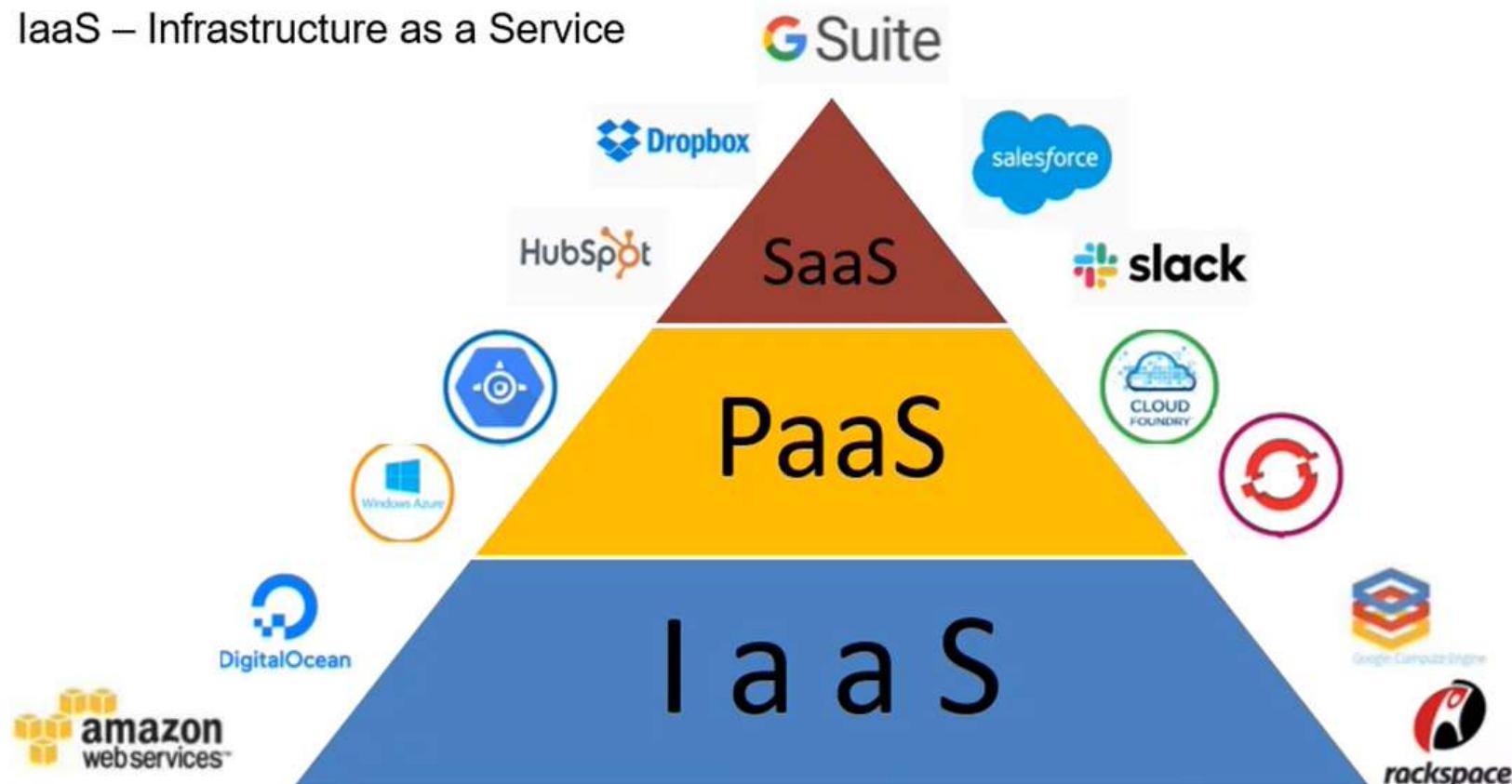
Cloud computing is relatively new business model in the computing world. According to the official NIST definition, “cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool configurable computing resources (e.g., network servers, storage, application and services) that can be rapidly provisioned and released with minimal management effort of service provider interactions.”

Cloud Service Models

SaaS – Software as a Service

PaaS – Platform as a Service

IaaS – Infrastructure as a Service



3-4-5 rule of Cloud Computing

3 – cloud service models or service types

4- deployment models

5 – essential characteristics of cloud computing
infrastructure

SaaS and Scalable Services

Software as a Service (SaaS)

Google Apps, Twitter, Facebook, and Flickr

Scalable infrastructures – processing engines

Google File System

MapReduce

Apache Hadoop

Apache Spark

Support cloud service

Internet of Things(IoT)

“making a computer sense information without the aid of human intervention”

Adapted by

Healthcare

Home

Environment

Transportation

Heavy Industry

Internet of Things(IoT)

Data produced by people, machines, and things

500 zettabytes

Global data center IP traffic

10.4 zettabytes

Things connected to the Internet by 2025

~100 billion and growing

What is IoT?

Internet connects people - “Internet of people”

IoT connects all things – “Internet of Things”



Interconnection of Machines or Devices or Things, and communicate with each other via Internet



Things are embedded with software, sensors, and network connectivity—that enables these objects to collect and exchange data.



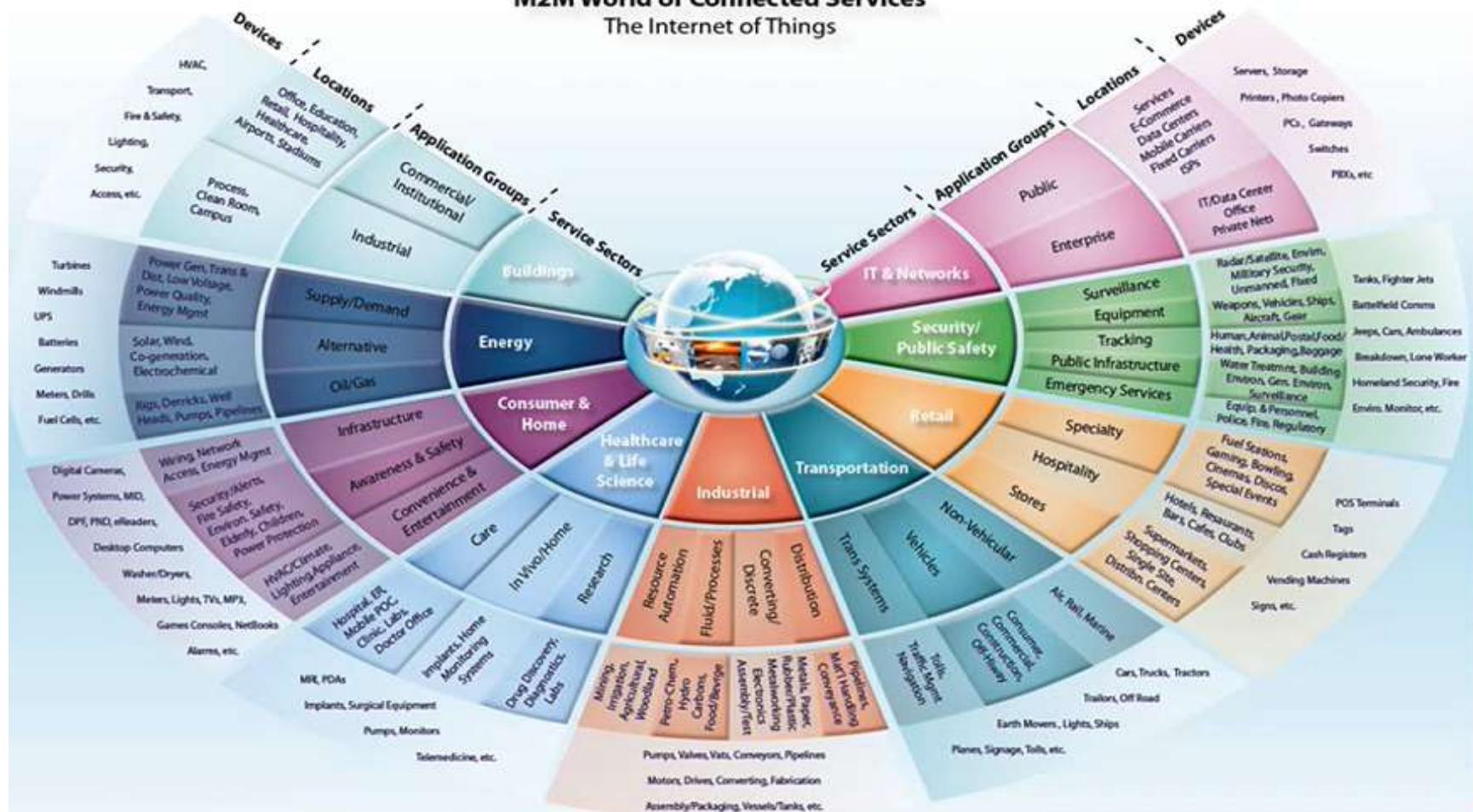
Internet of Things

innovate

achieve

lead

M2M World of Connected Services The Internet of Things



IoT: 4 Layer model



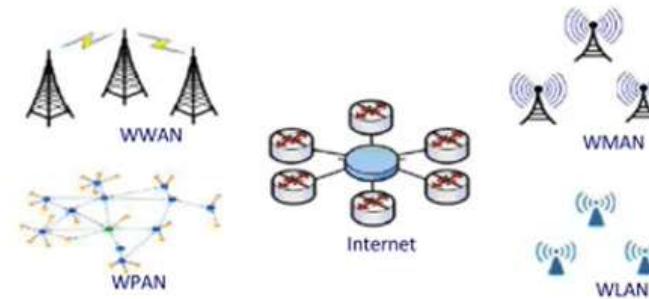
Application



Information Processing



Network



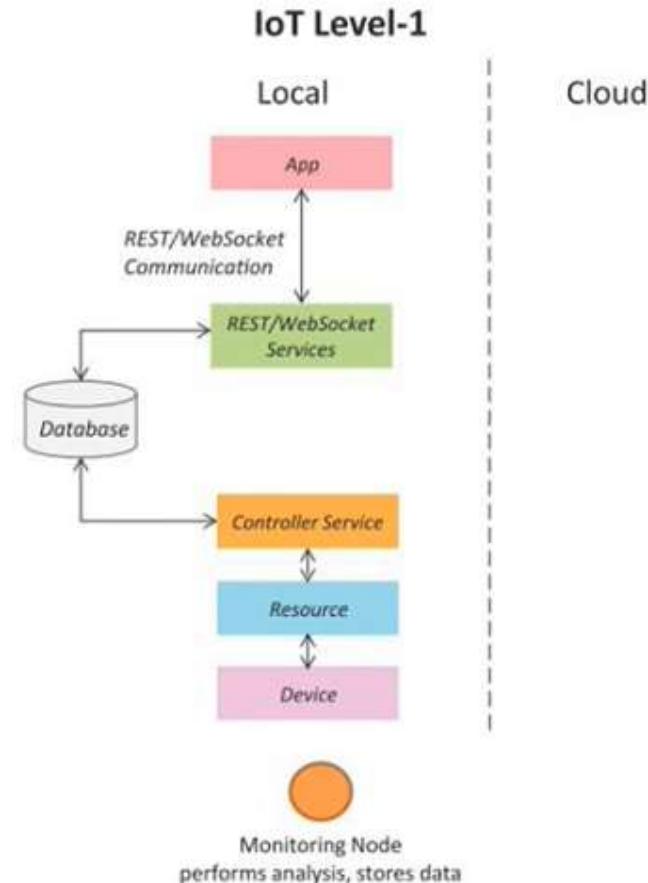
Sensors and Actuators



IoT Level 1

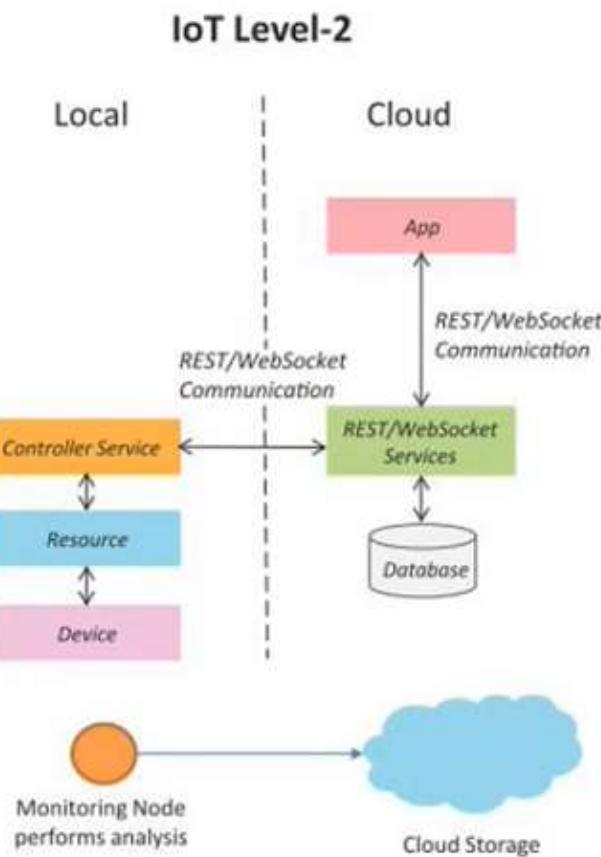
- A level-1 IoT system has a single node/device that performs sensing and/or actuation, stores data, performs analysis and hosts the application

Level-1 IoT systems are suitable for modeling low- cost and low-complexity solutions where the data involved is not big and the analysis requirements are not computationally intensive.



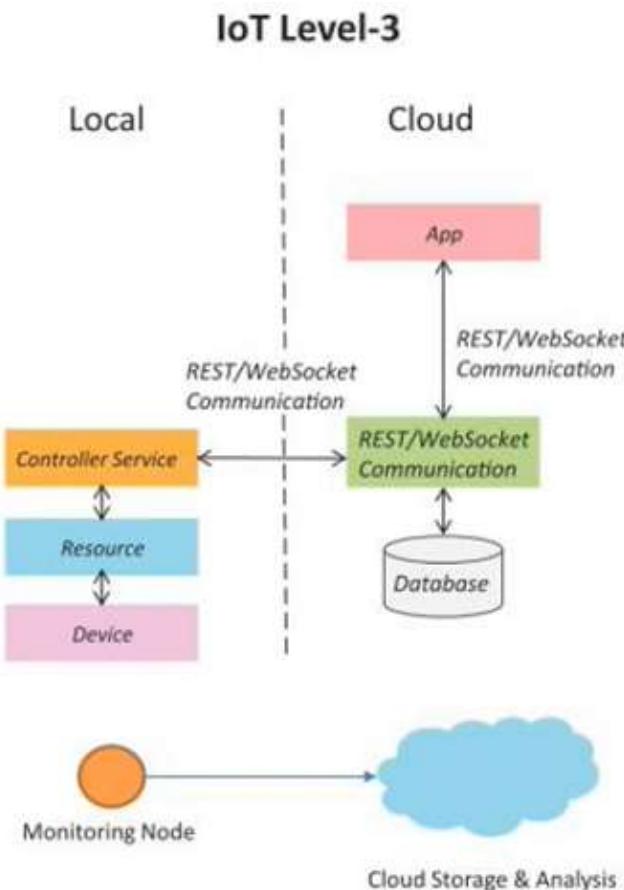
IoT Level 2

- A level-2 IoT system has a single node that performs sensing and/or actuation and local analysis.
- Data is stored in the cloud and application is usually cloud-based.
- Level-2 IoT systems are suitable for solutions where the data involved is big, however, the primary analysis requirement is **not computationally intensive and can be done locally itself**.



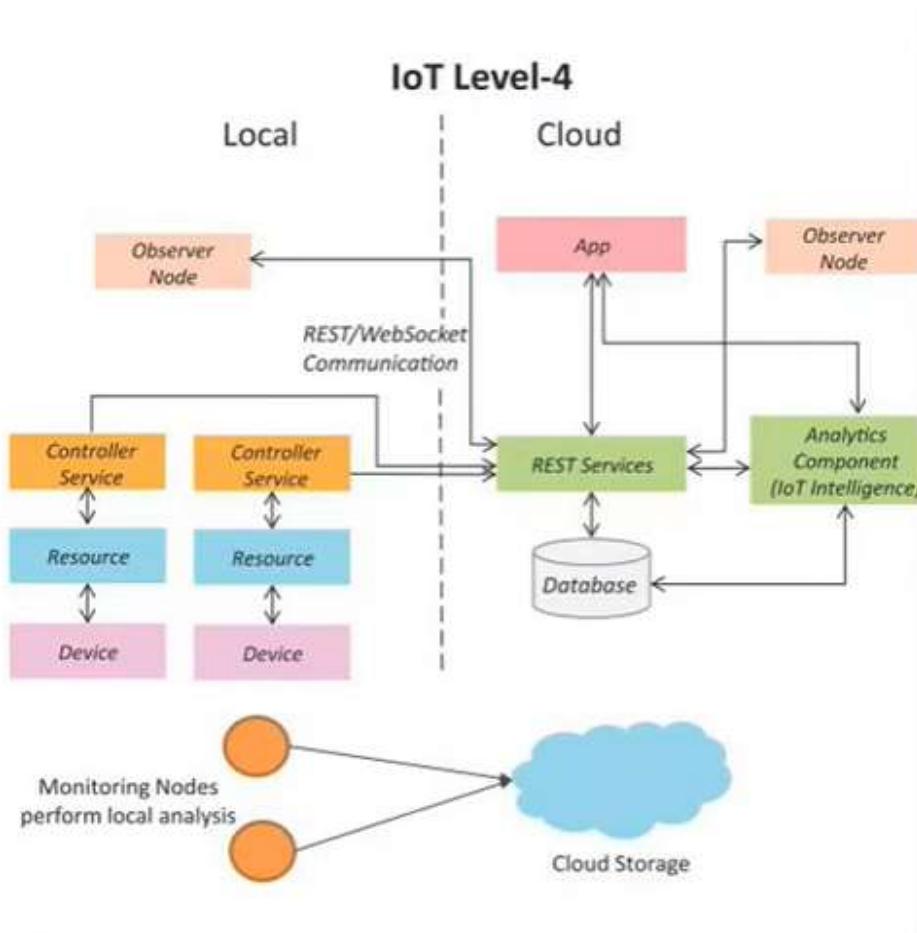
IoT Level 3

- A level-3 IoT system has a single node. Data is stored and analyzed in the cloud and application is cloud-based.
- Level-3 IoT systems are suitable for solutions where the data involved is big and the analysis requirements are **computationally intensive**.



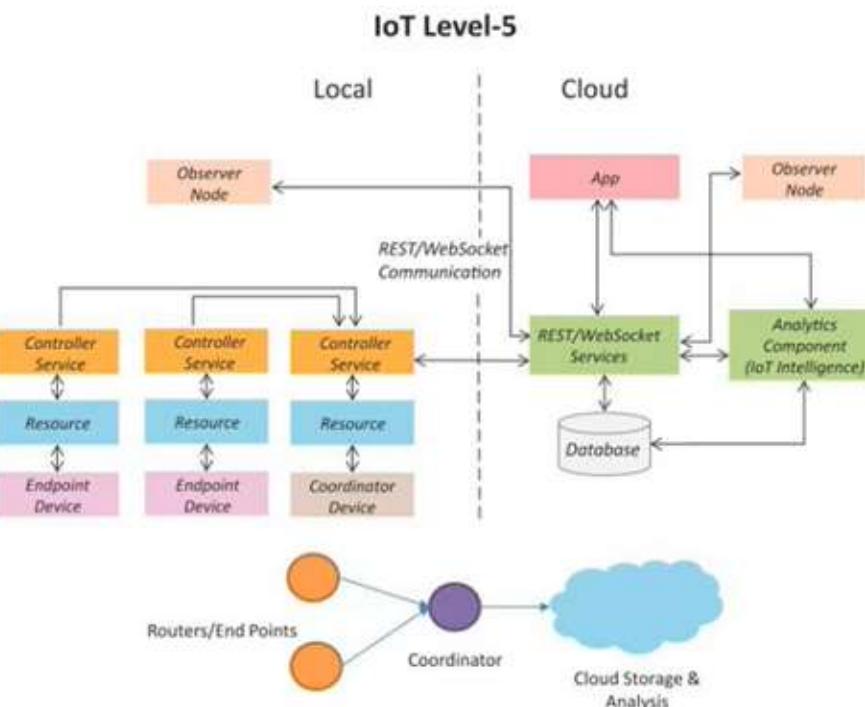
IoT Level 4

- A level-4 IoT system has multiple nodes that perform local analysis. Data is stored in the cloud and application is cloud-based.
- Level-4 contains local and cloud-based observer nodes which can subscribe to and receive information collected in the cloud from IoT devices.
- Level-4 IoT systems are suitable for solutions where multiple nodes are required, the data involved is big and the analysis requirements are computationally intensive.



IoT Level 5

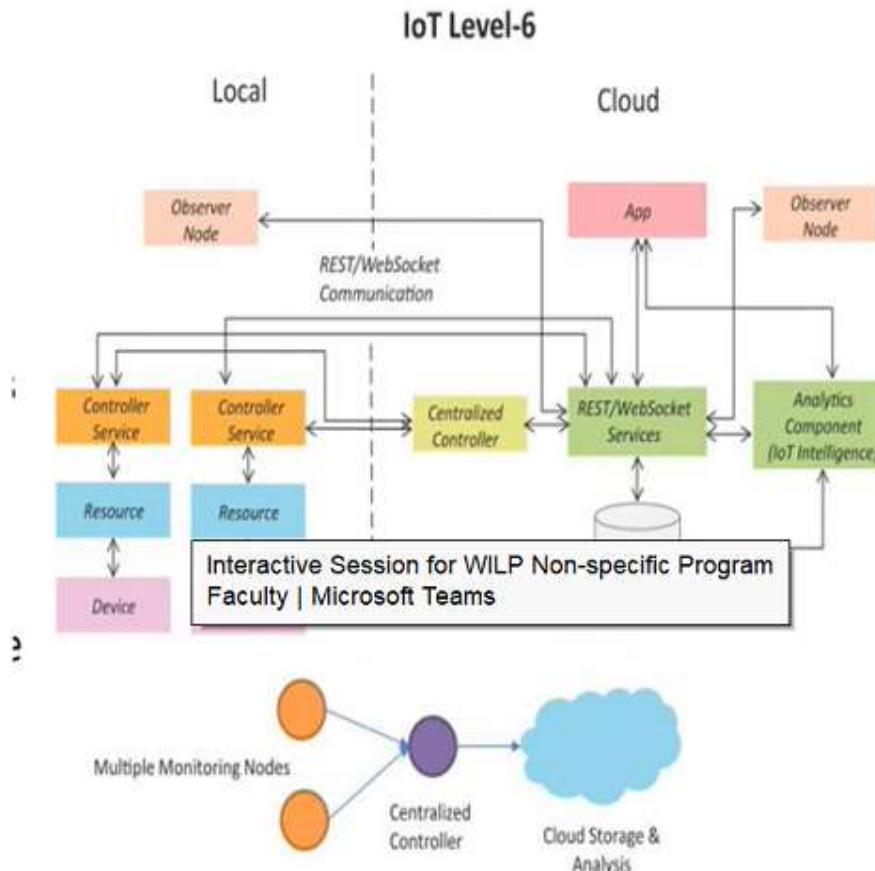
- A level-5 IoT system has multiple end nodes and one coordinator node.
- The end nodes that perform sensing and/or actuation.
- Coordinator node collects data from the end nodes and sends to the cloud.
- Data is stored and analyzed in the cloud and application is cloud-based.
- Level-5 IoT systems are suitable for solutions based on wireless sensor networks, in which the data involved is big and the analysis requirements are computationally intensive.



IoT Level 6

A level-6 IoT system has multiple independent end nodes that perform sensing and/or actuation and send data to the cloud.

- Data is stored in the cloud and application is cloud-based.
- The analytics component analyzes the data and stores the results in the cloud database.
- The results are visualized with the cloud-based application.
- The centralized controller is aware of the status of all the end nodes and sends control commands to the nodes.



Use Cases

- Farming and agriculture
 - Agribots
 - Farm automation
 - Disaster protection
- Healthcare
 - Patient monitoring that leverages medical devices such as insulin pumps, smart lenses and pacemakers
 - Rural Medicine
 - Wearables and connected apps that track various health metrics
 - A closed-loop system in an ICU that uses smart sensors

Use Cases

Boeing 787 – Generate 5GB of data every second

Autonomous Vehicles – Generate 1 GB of data by every car every second

Smart grid - Sensors and IoT devices in factories, plants and offices are being used to monitor energy use and analyze their consumption in real-time

In-hospital patient monitoring - monitoring devices (e.g. glucose monitors, health tools and other sensors) generate large amounts of unprocessed data from devices would need to be stored on a 3rd party cloud – Security concern

Content delivery - By caching content – e.g. music, video stream, web pages. Need reduced latency

Video surveillance - need strict real-time analytics to mine video content



BITS Pilani
Pilani Campus

BITS Pilani presentation

Paramananda Barik
CS&IS Department





SEZG586/SSZG586, Edge Computing Lecture No.3

Agenda

- Shortcomings of Cloud for IoT
 - Driving factors of Edge Computing
 - Why do we need Edge Computing?
 - Key Techniques that Enable Edge Computing
 - VMs and Containers
 - SDN
 - CDN
 - Cloudlets/Micro Datacenters
 - Basic Attributes of Edge
 - "CROSS" Value of Edge Computing
 - Edge Computing Enables Industry Intelligence
 - Edge Computing Benefits
 - Edge Computing Systems
-

Shortcomings of Cloud for IoT

- a safety critical control system operating an industrial machine might need to stop immediately if a human is too close – **Speed, Reliability**
- a temperature sensor reports a 20°C reading every second might not be interesting until the sensor reports a 40°C reading - **Cost**
- autonomous vehicles or augmented reality applications need a response time below 20ms – **Speed**

Need of the hour: IoT applications might require short response time, private data, and produce a large quantity of data needing large bandwidth

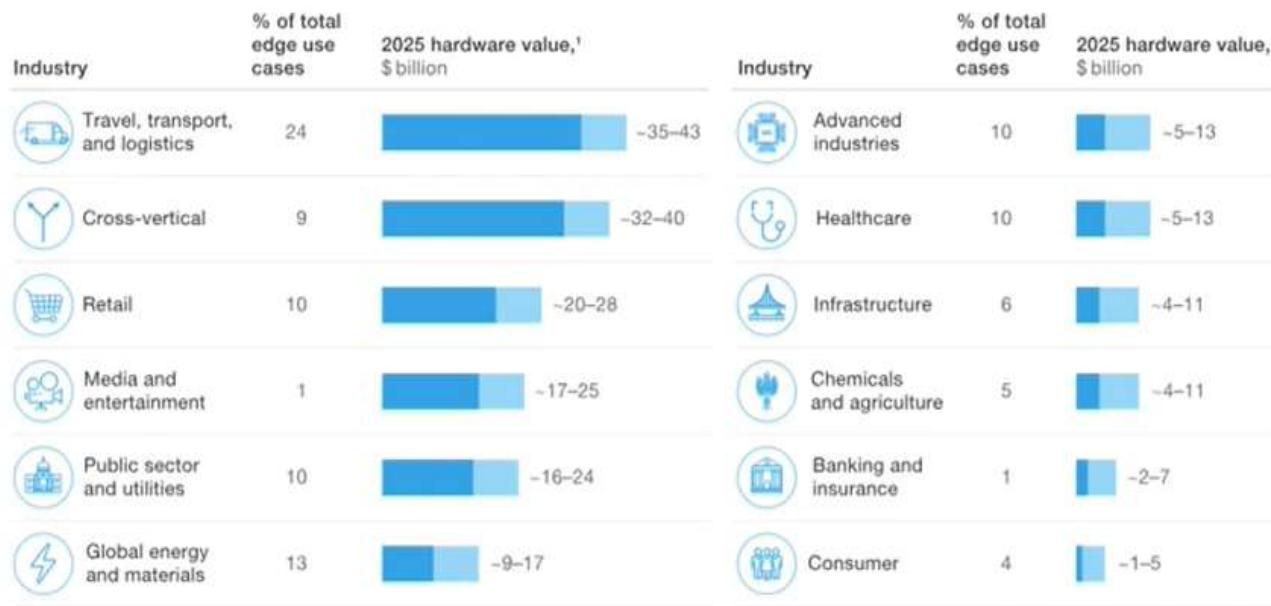
Study by McKinsey



Industries with **the most edge computing use cases** are

- Travel, transportation, and logistics
- Energy
- Retail
- Healthcare
- Utilities

Edge computing represents a potential value of \$175 billion to \$215 billion in hardware by 2025.



Total: ~\$175 billion-\$215 billion

Driving factors of Edge Computing



- Varied connectivity and data mobility
- Need for real-time decision making
- Localized compute power & storage

Characteristics of Edge Computing

- Proximity
- Ultra-low latency
- High bandwidth
- Reliability
- Real time access to radio network

Why do we need Edge Computing?



- Push(ed) from Cloud Services
 - Limited bandwidth over Internet
 - Response Time
 - Reliability of network
- Pull(ed) from Internet of Things
 - Enormous amount of Data generated by billions of devices
 - Leading to huge unnecessary bandwidth and computing resources usage
 - End devices in IoT are energy constraint things
 - Wireless communication module - drains battery

Why do we need Edge Computing?



- The end devices at the edge usually play as a data producer and consumer
- Example – Social networking platforms
 - Youtube – 72 hrs of content uploaded every single minute
 - Facebook – users share ~2.5 million pieces of content
 - Twitter – 300,000 tweets
 - Instagram – 220,000 new photos

Key Techniques that Enables Edge Computing



- VMs and Containers
 - Software Defined Networking (SDN)
 - Content Delivery/Distribution Network (CDN)
 - Cloudlets and Micro Data Centers (MDC)
-

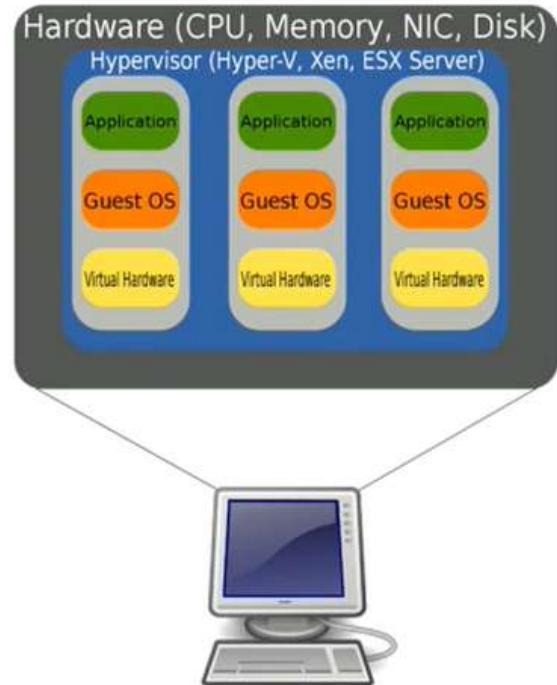
VMs and Containers

- What is Virtualization?
 - Virtualization is technology that lets users create useful IT services using resources that are **traditionally bound to hardware**.
 - It allows users to use a **physical machine's full capacity** by distributing its capabilities among many users or environments.
 - Virtualization and cloud computing are not interchangeable.
 - Virtualization is software that makes computing environments independent of physical infrastructure.



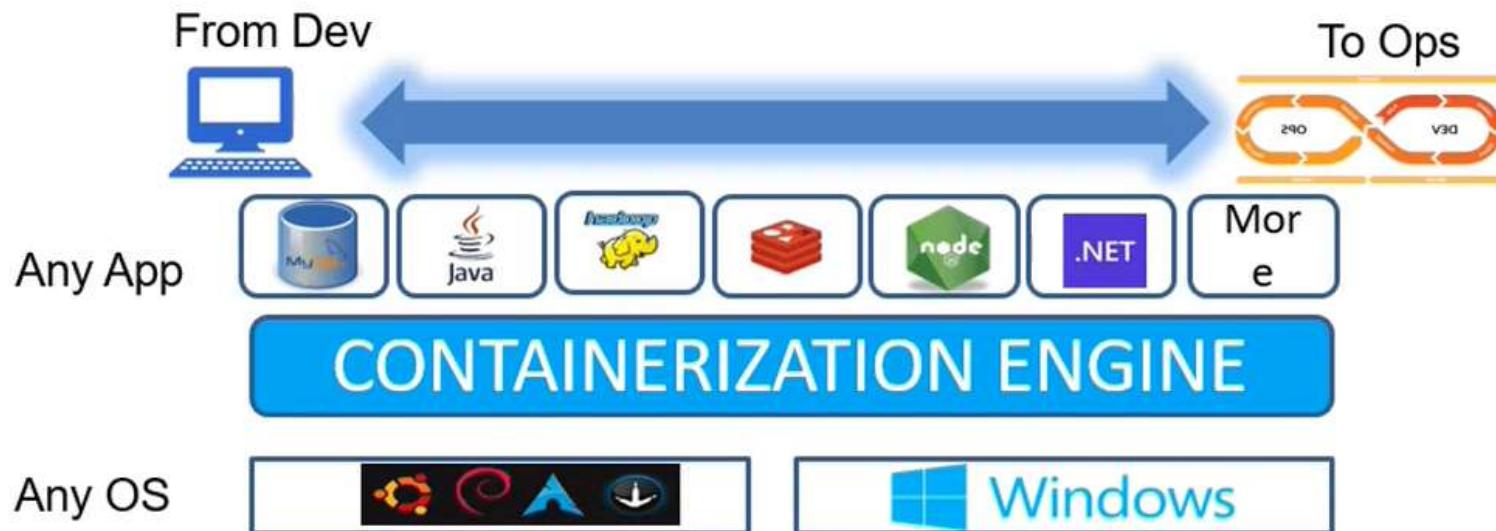
Virtual Machines

- A virtual computer system is known as a “virtual machine” (VM): a tightly isolated software container with an operating system and application inside.
- Each self-contained VM is completely independent.
- Putting multiple VMs on a single computer enables several operating systems and applications to run on just one physical server, or “host.”

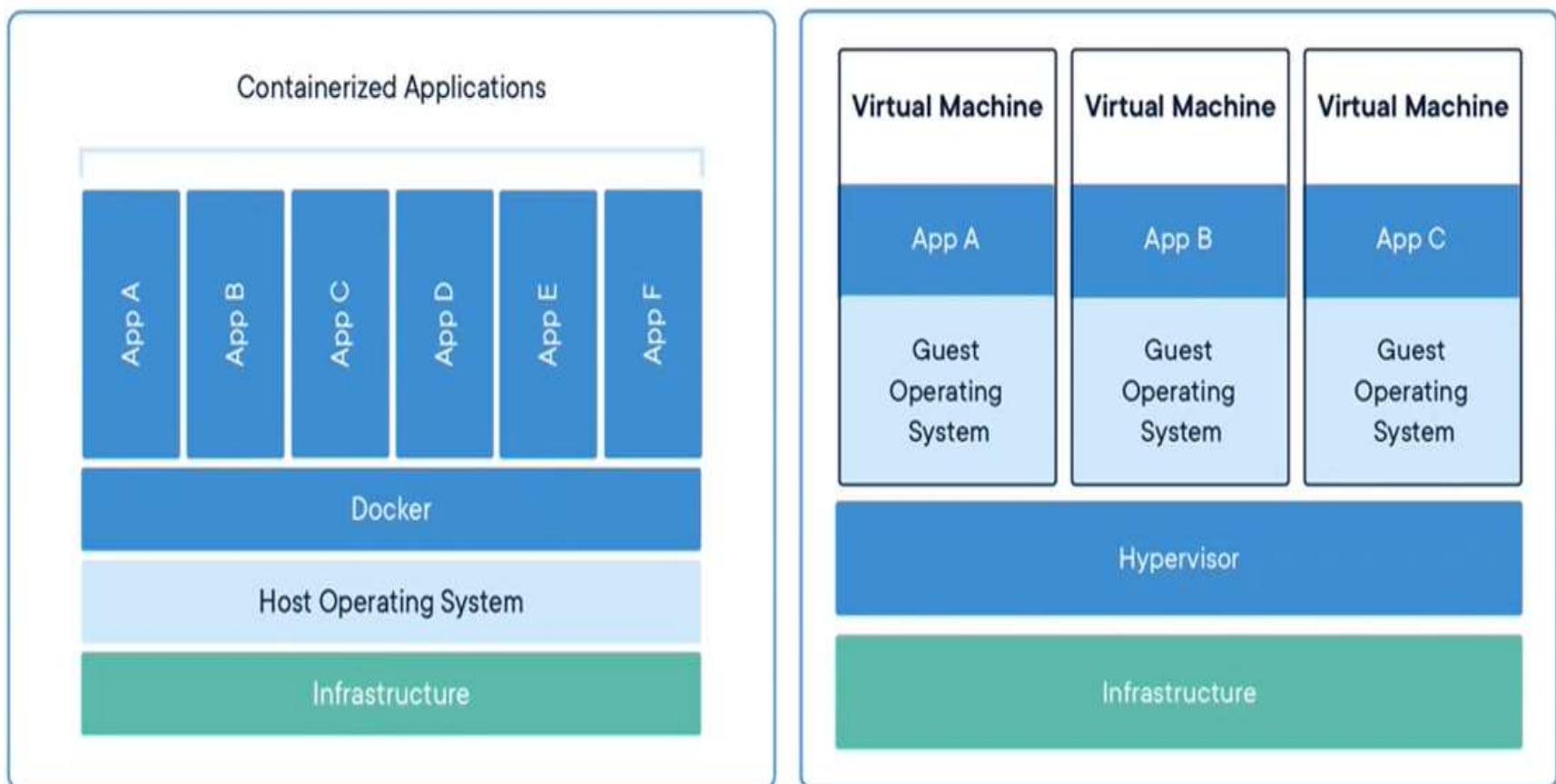


What are Containers?

- A software container is a standardized package of software.
- Everything needed for the software to run is inside the container.
- The software code, runtime, system tools, system libraries, and settings are all inside a single container



Virtual Machine vs Containers



Software Defined Networking (SDN)



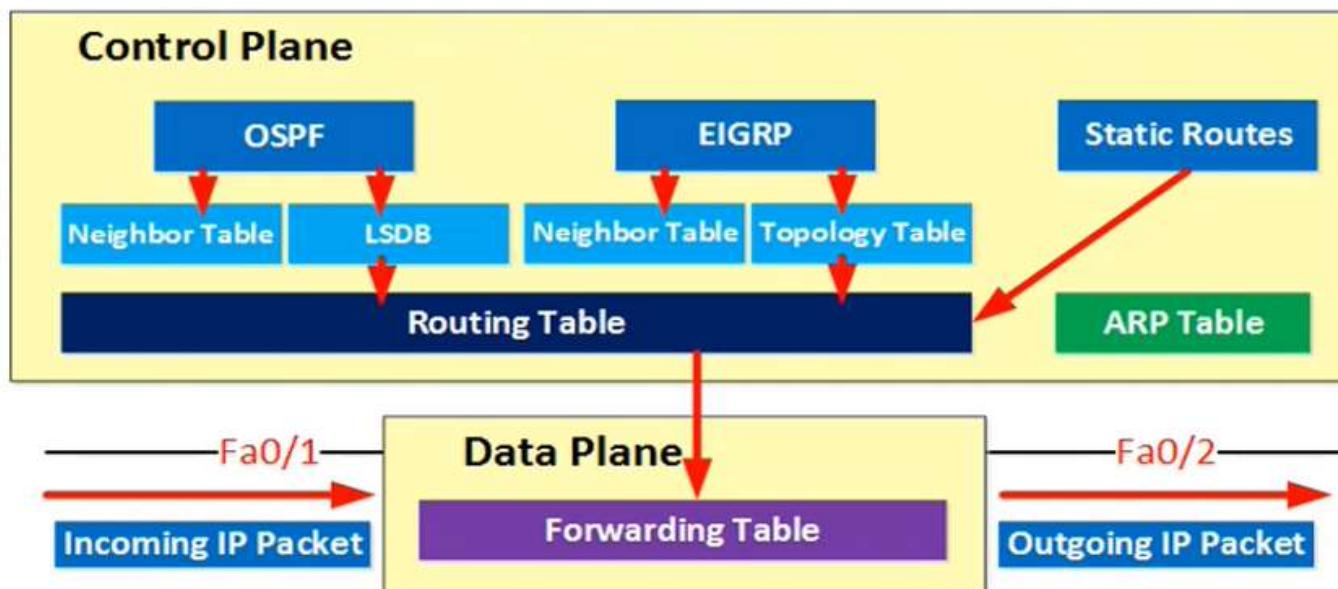
- Traditional Networking
 - Router functions
 - check the destination IP address in the routing table
 - Routing protocols like OSPF, EIGRP or BGP
 - use ARP to figure out the destination MAC address
 - Ethernet frame checksum recalculated

All these different tasks are separated by different **planes**. There are three planes:

- **control plane**
- **data plane**
- **management plane**

Different Planes

- Control Plane
- Data Plane
- Management Plane



Limitations of traditional networks



- Configuration and re-configuration of network is **SLOW, MANUAL** process
 - VLANs have to be created on all switches
 - configure a root bridge for the new VLANs
 - assign four new subnets, one for each VLAN
 - create new sub-interfaces with IP addresses on the switches
 - configure VRRP or HSRP on the switches for the new VLANs
 - configure the firewalls to permit access to the new applications / subnets
 - advertise the new subnets in a routing protocol on our switches, routers, and firewalls
- Might take hours to carry out these tasks in spite of having automation tools

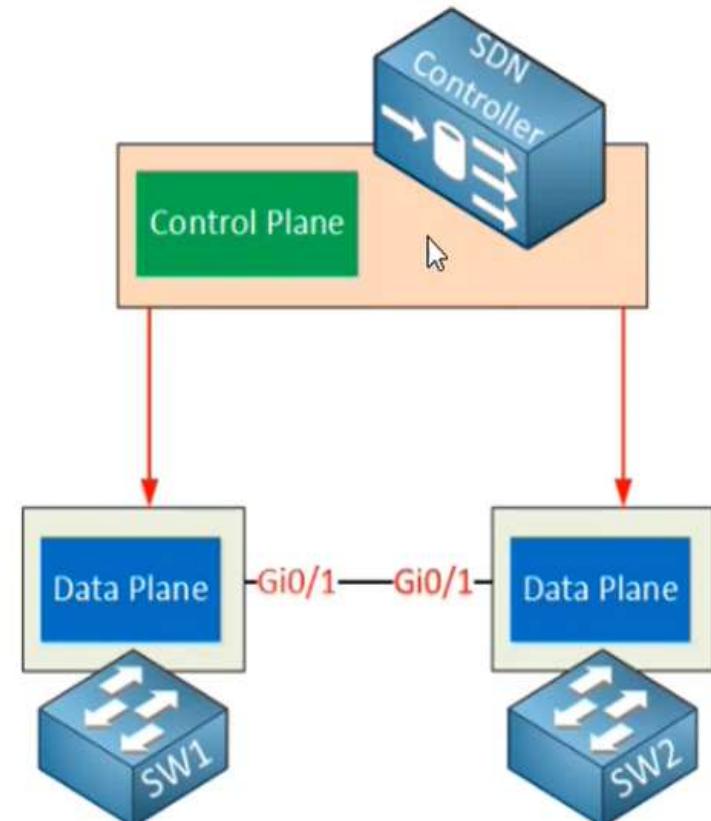
Software Defined Networking (SDN)



SDN controller - responsible for the activities done by control plane.

The switches are now just “dumb” devices that only have a data plane, no control plane.

The SDN controller is responsible for feeding the data plane of these switches with information from its control plane



How does it help Edge Computing?



Edge computing pushes the computational infrastructure to the proximity of the data source, and the computing complexity will also increase correspondingly.

SDN provides a cost-effective solution for Edge network virtualization

- Simplifies the network complexity by offering the automatic Edge device reconfiguration and bandwidth allocation.
- Edge devices could be set up and deployed in a plug-and-play manner enabled by SDN

Content Delivery/Distribution Network (CDN)



CDN is the concept of caching the content to the servers near the data consumers matches the system of Edge computing.

As the upstream server that delivers the content is becoming the bottleneck of the web due to the increasing web traffic, CDN can offer data caching at the Edge of the network with scalability and save both the bandwidth cost and page load time significantly.

Cloudlets and Micro Data Centers (MCD)



Cloudlets and Microdata centers are the small-scale cloud data centers with mobility enhancement. They can be used as the gateway between Edge/mobile devices and the cloud. The computing power on the Cloudlets or MDCs could be accessed with lower latency by the Edge devices due to the geographical proximity.

Essential computing tasks for Edge computing such as speech recognition, language processing, machine learning, image processing, and augmented reality could be deployed on the Cloudlets or MDCs to reduce the resource cost.

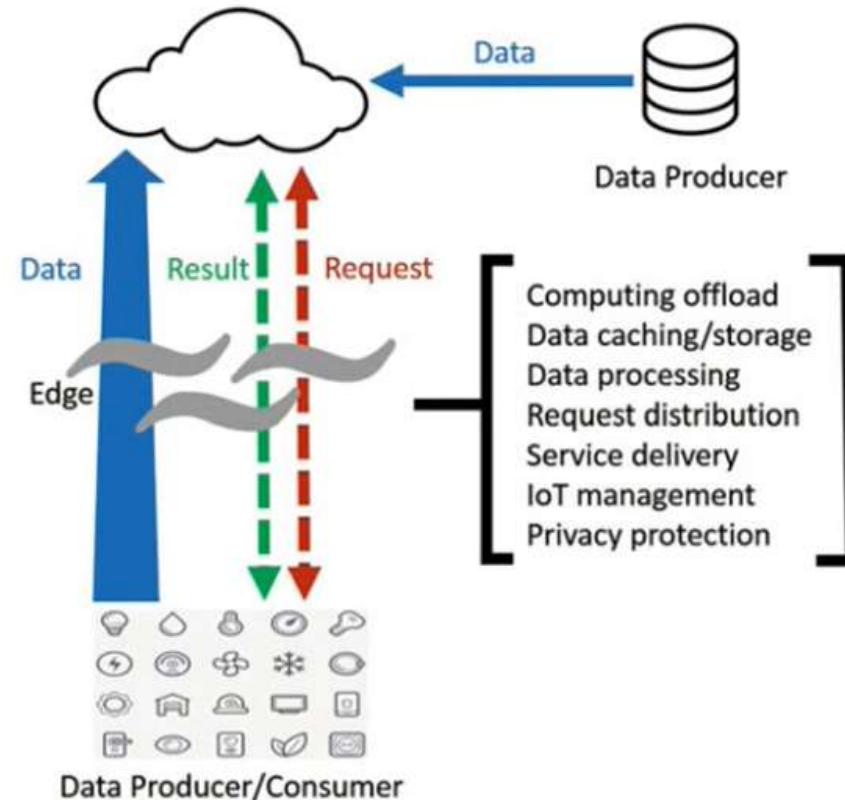
Edge Computing Definition

Edge computing refers to the enabling technologies allowing **computation** to be performed at the **edge of the network**, on *downstream data on behalf of cloud services* and *upstream data on behalf of IoT services*

- *computing should happen at the proximity of data sources*
-

Two-way computing streams

- Up stream
- Down stream



Basic Attributes of Edge

- Connectivity
 - First Entry of Data
 - Constraint
 - Distribution
 - Convergence
-

Basic Attributes of Edge

- **Connectivity**
 - Provide connection functions – Protocol support, deployment, management and maintenance
 - Research advancements - TSN, SDN, NFV, NaaS, WLAN, NB-IoT, and 5G
 - Interoperability with existing industrial buses
- **First Entry of Data**
 - Mass, real-time, diversity
 - Data management

Basic Attributes of Edge

- Constraint
 - Adaptability - harsh working conditions and operating environments
- Distribution
 - Support - distributed computing and storage, dynamic scheduling
- Convergence
 - Convergence of the Operational Technology (OT) and Information and Communications Technology (ICT)
 - Support collaboration in connection, data, management, control, application, and security.

“CROSS” Value of Edge Computing



Mass and Heterogeneous Connection

- large number of connected devices
- heterogeneous Bus connections

Real-Time Services

- 10 ms

Data Optimization

- large amount of heterogeneous data

“CROSS” Value of Edge Computing



Smart Applications

- intelligent applications

Security and Privacy Protection

- end-to-end protection
- data integrity and confidentiality

Edge Computing Enables Industry Intelligence – How?



- Connection – Physical and digital worlds
- Platform - Model driven, intelligent, distributed
- Collaborate - with cloud computing

Edge Computing Enables Industry Intelligence – How?



- Changes in the network field
 - Bandwidth increase – 1000 fold, Cost has decreased
- Changes in the computing field
 - Celeron – 6.40×10^3 MIPS, Xeon – 1.40×10^5 MIPS, A9 – 3.6×10^3 MIPS
- Changes in the storage field
 - Capacity increase – 10000 fold, cost has decreased
 - Speed also has increased

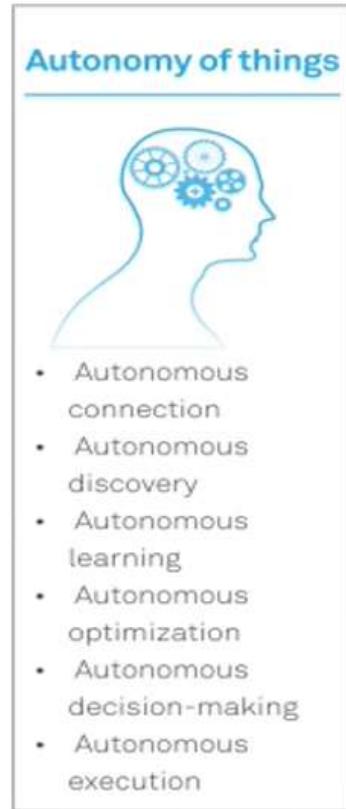
Connection – Physical and digital world



- Changes in the network field
 - Bandwidth increase – 1000 fold, Cost has decreased
- Changes in the computing field
 - Celeron – 6.40×10^3 MIPS, Xeon – 1.40×10^5 MIPS, A9 – 3.6×10^3 MIPS
- Changes in the storage field
 - Capacity increase – 10000 fold, cost has decreased
 - Speed also has increased

Platform – Model driven, intelligent, distributed

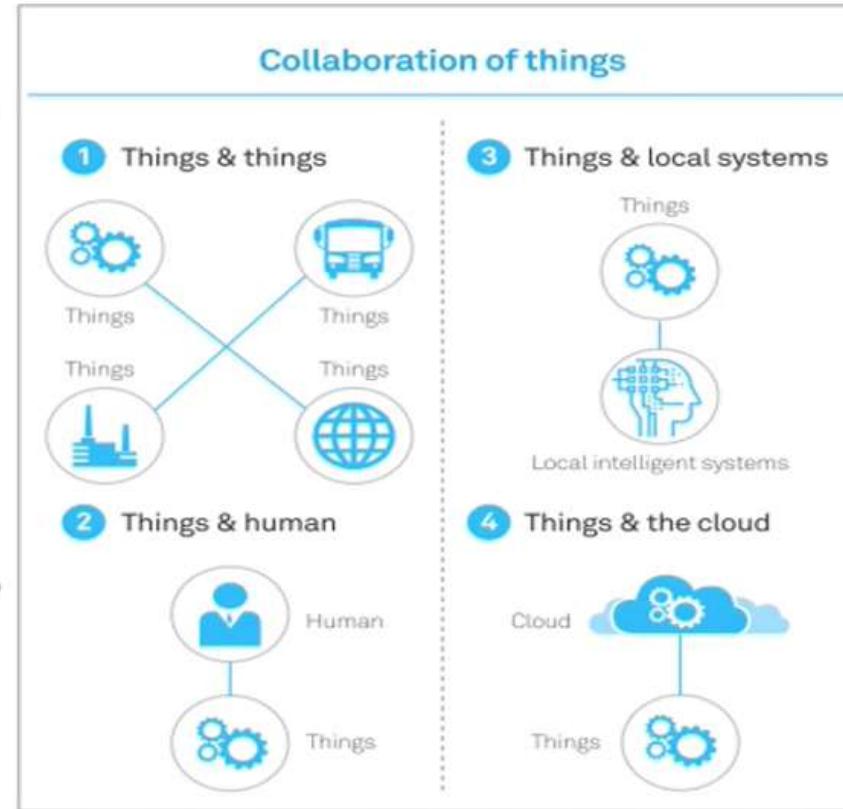
- Autonomy of things



Share data and knowledge to enhance collaboration

Learn data generated in collaboration to enhance autonomy

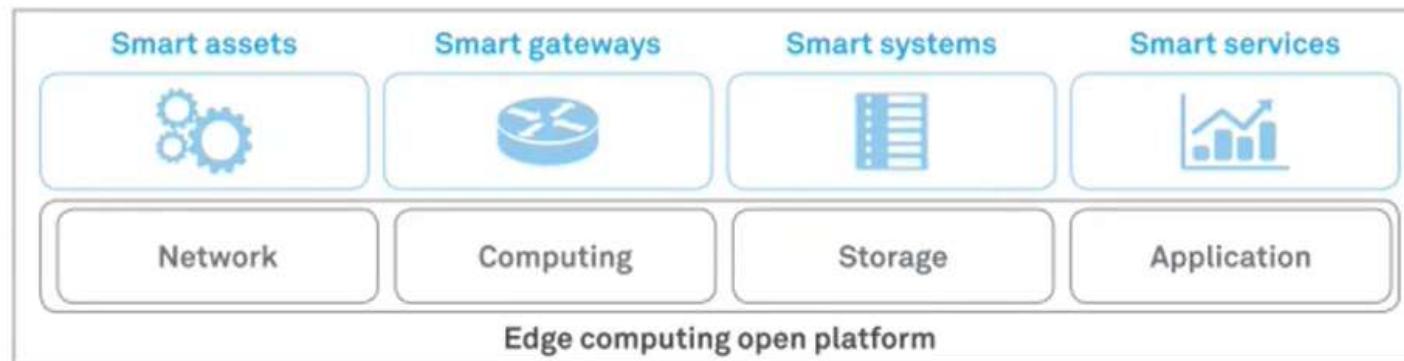
- Collaboration of things



Intelligent distributed architecture



- Smart assets
- Smart gateways
- Smart systems
- Smart services



Collaboration of Edge Computing and Cloud Computing



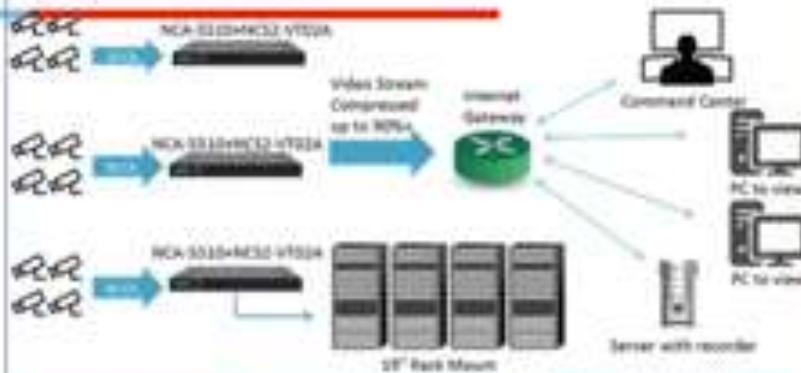
Point of Collaboration	Edge Computing	Cloud Computing
Network	Data aggregation (TSN + OPCUA)	Data analysis
Service	Agent	Service orchestration
Application	Micro applications	Lifecycle management of applications
Intelligence	Distributed reasoning	Centralized training

Edge Computing Benefits



- Speed and Latency
- Security
- Cost Savings
- Greater Reliability
- Scalability

Video Streaming Encoder Deployment



CURRENT: 4G

Only a few large centralized data centers



> 80 ms Latency

The vehicle moved over four feet by the time it received a response due to the large distance from the data center.

UPCOMING: 5G

Thousands of new micro data centers under cell towers



< 5 ms Latency

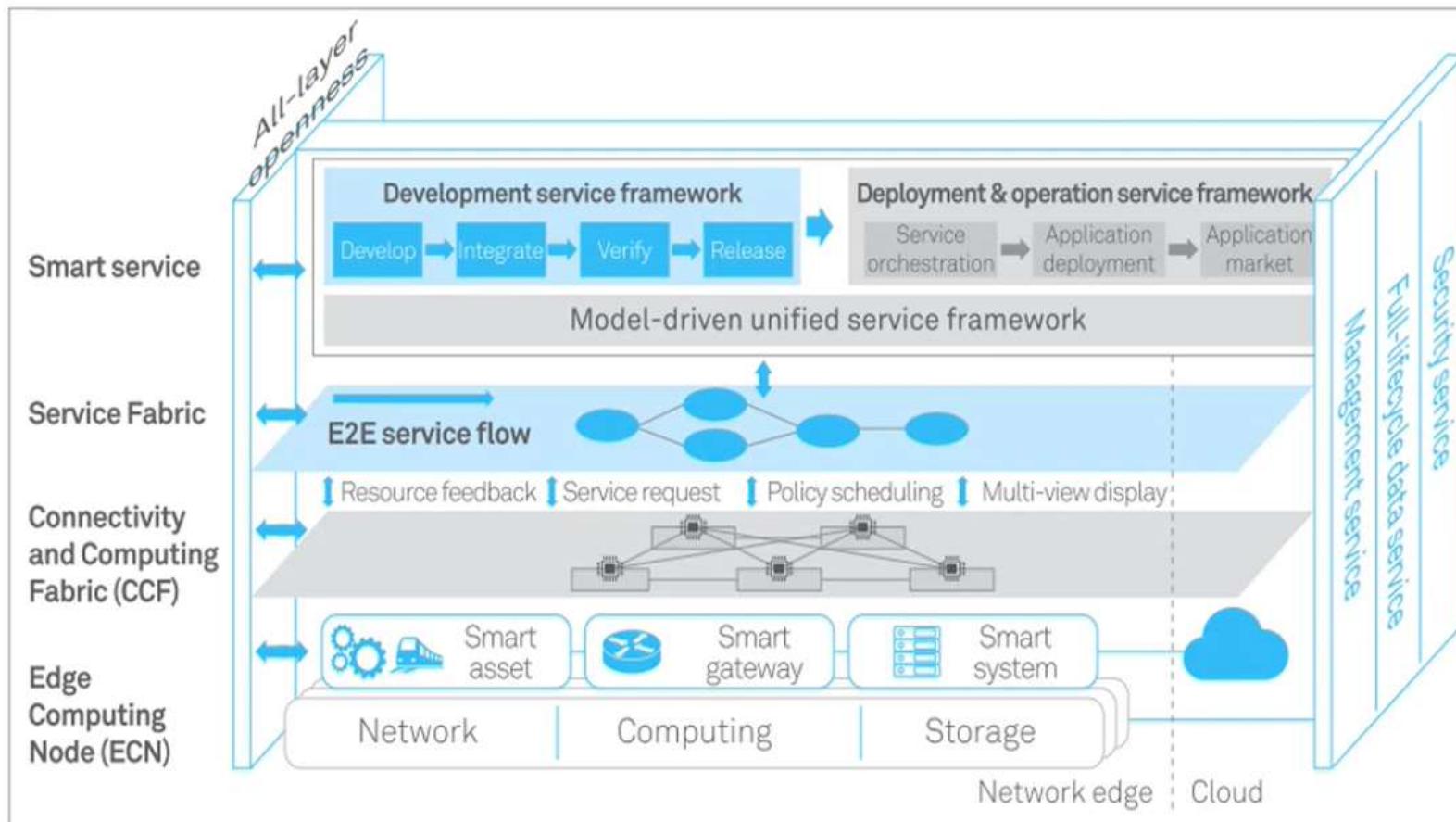
The vehicle moved less than four inches by the time it received a response, thanks to the close distance to the micro data center.

<https://www.verizon.com/business/solutions/5g/edge-computing/industry-use-cases-examples/>

Edge Computing Systems

- Apache Edgent
 - AWS Greengrass
 - AWS Wavelength
 - Azure IoT Edge
 - Bosch IoT Edge
 - EdgeX foundry
-

Edge Computing reference architecture 2.0



Edge Computing reference architecture 2.0



Model-Driven Engineering

Coordination Between the Physical and Digital Worlds

Cross-Industry Collaboration

Reduced System Heterogeneity and Simplified Cross-Platform Migration

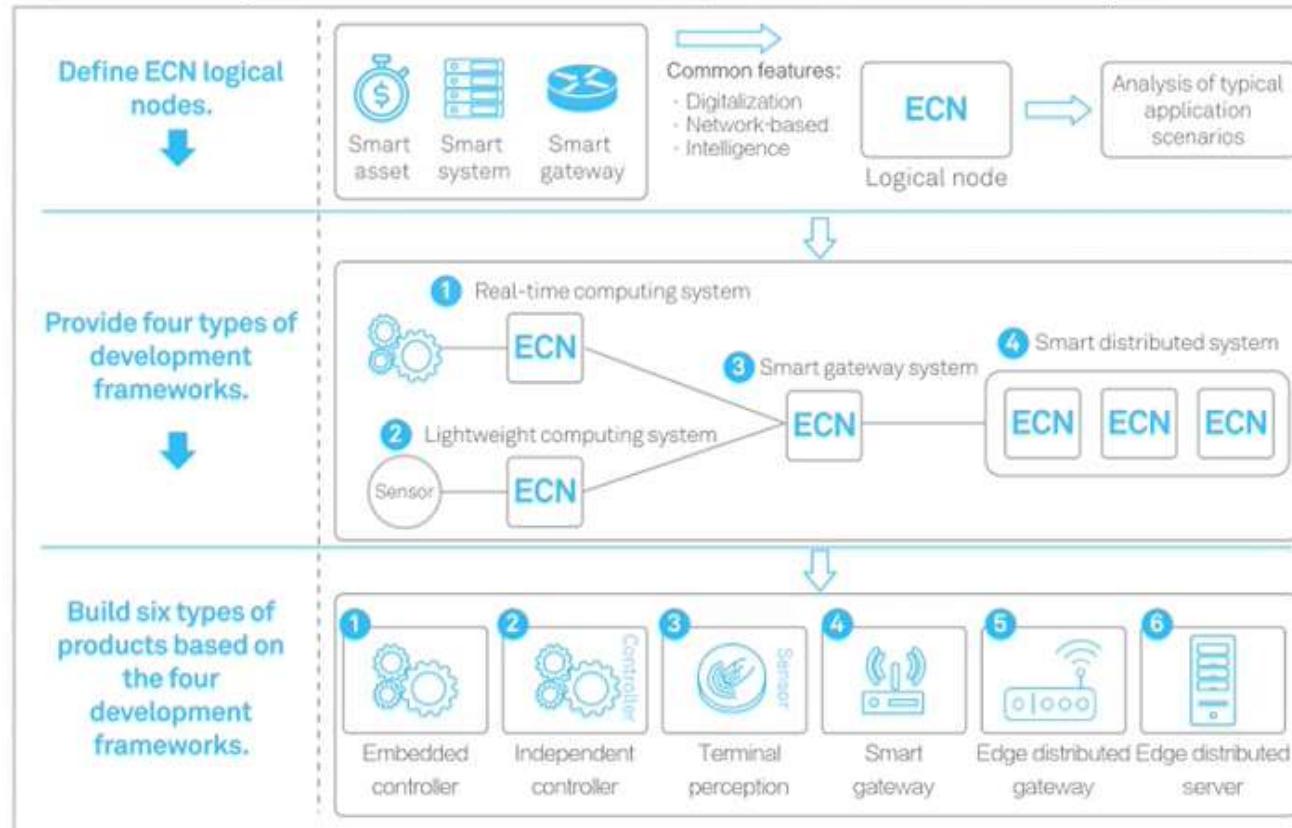
Effective Support for System Lifecycle Activities

Multi-View Display

- **Concept View**
 - Domain models and key concepts of edge computing
 - **Function View**
 - Functions and design concepts
 - **Deployment View**
 - System deployment process
-

Multi-View Display

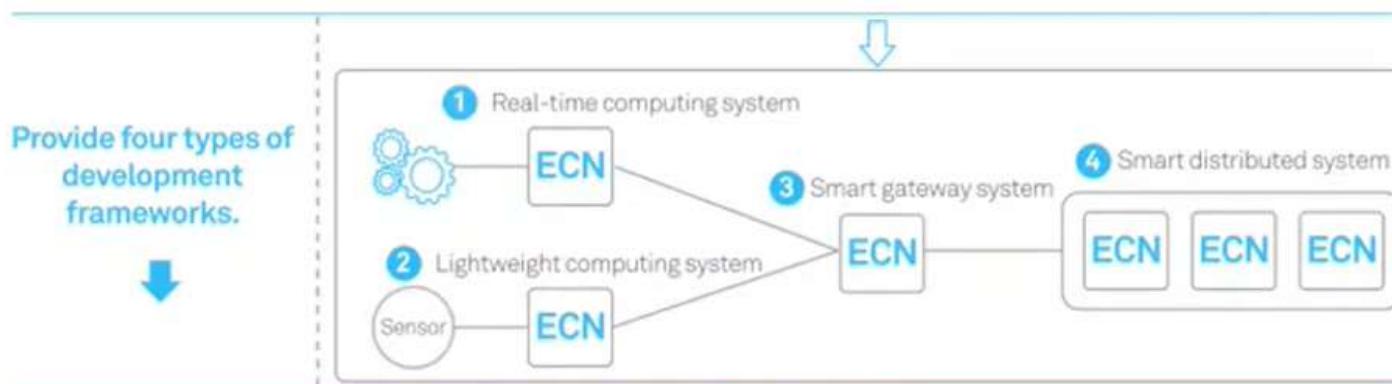
ECNs, Development Frameworks, and Product Implementation



Development frameworks of ECNs



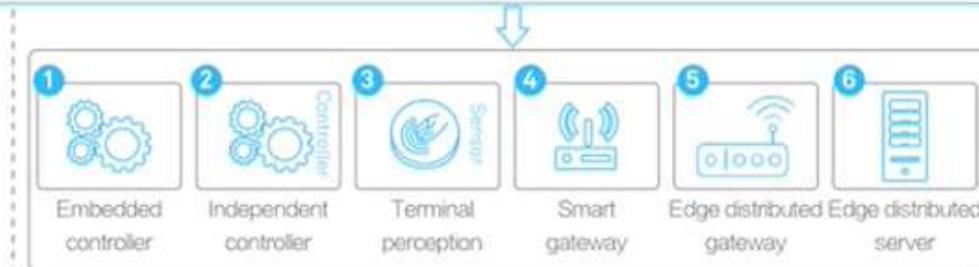
- Real-Time Computing System
- Lightweight Computing System
- Smart Gateway System
- Smart Distributed System



ECN product implementation

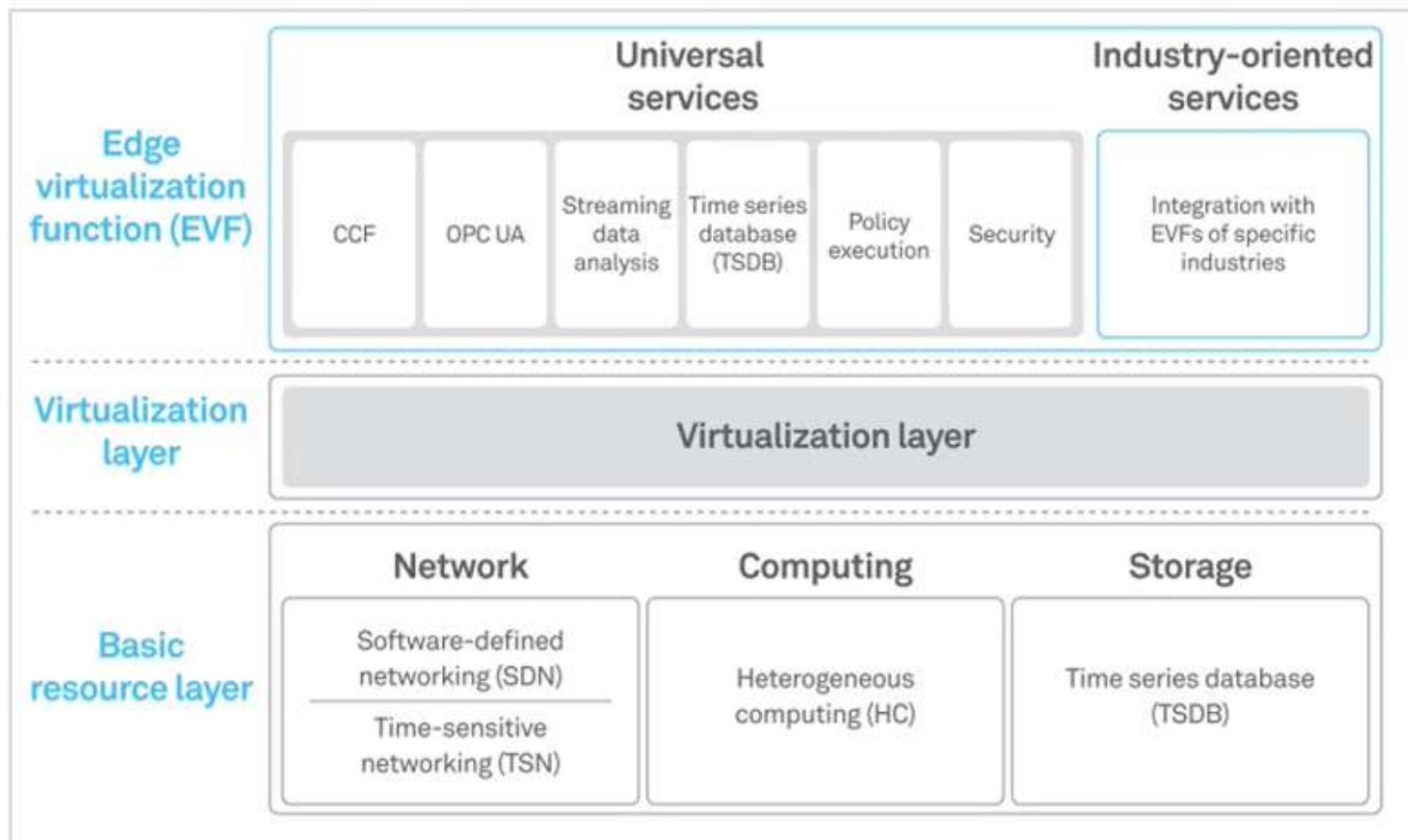
Product	Typical Scenario
ICT-converged gateway	Connection of elevators, smart street lamp
Independent controller	Industrial Programmable Logic Controller (PLC)
Embedded controller	Virtual Programmable Logic Controller (vPLC), robot
Sensing terminal	Computer Numerical Control (CNC), instrument
Distributed service gateway	Smart power distribution
Edge cluster (edge cloud)	Digital workshop

Build six types of products based on the four development frameworks.



1. Embedded controller (Icon: Gears) 2. Independent controller (Icon: Gears with 'Controller' text) 3. Terminal perception (Icon: Eye) 4. Smart gateway (Icon: Router) 5. Edge distributed gateway (Icon: Router with 'edge' text) 6. Edge distributed server (Icon: Server tower)

Function view : ECN functional layer



Basic Resource Layer

This layer includes the following modules:

- Network
 - Computing
 - Storage
-

Virtualization Layer

Virtualization technology

- reduces system development
- deployment costs
- Virtualization technologies
 - Bare metal architecture
 - Host architecture
- The bare metal architecture has better real-time performance and is generally used by smart assets and smart gateways.

Edge Virtualization Functions Layer



The EVF layer delivers the following basic services:

- Distributed CCF service
- OPC UA service
- Streaming real-time data analysis service
- TSDB service
- Policy execution service
- Security service

SDN

SDN's unique benefits:

- Mass Connections
- Model-Driven Policy Automation
- E2E Service Protection
- Lifecycle Management of Applications
- Architecture Openness

- Standard Ethernet technologies
 - High transmission
 - Speed
 - Flexible topology
 - Long transmission distance
 - Cost-effectiveness
- Constraints
 - Quality of Service (QoS) mechanism and
 - Carrier Sense Multiple Access with Collision Detection (CSMA/CD) mechanism
- Key industry requirements - timeliness and determinism

- ## Advantages

- Ensures μ s-level latency and jitter of less than 500 ns
- Large bandwidth requirements
- Reliable data transmission

Heterogeneous Computing

The heterogeneous computing architecture uses the following key technologies:

- Memory processing optimization
 - Task scheduling optimization
 - Tool chain for development
-

Time Series Database (TSDB)

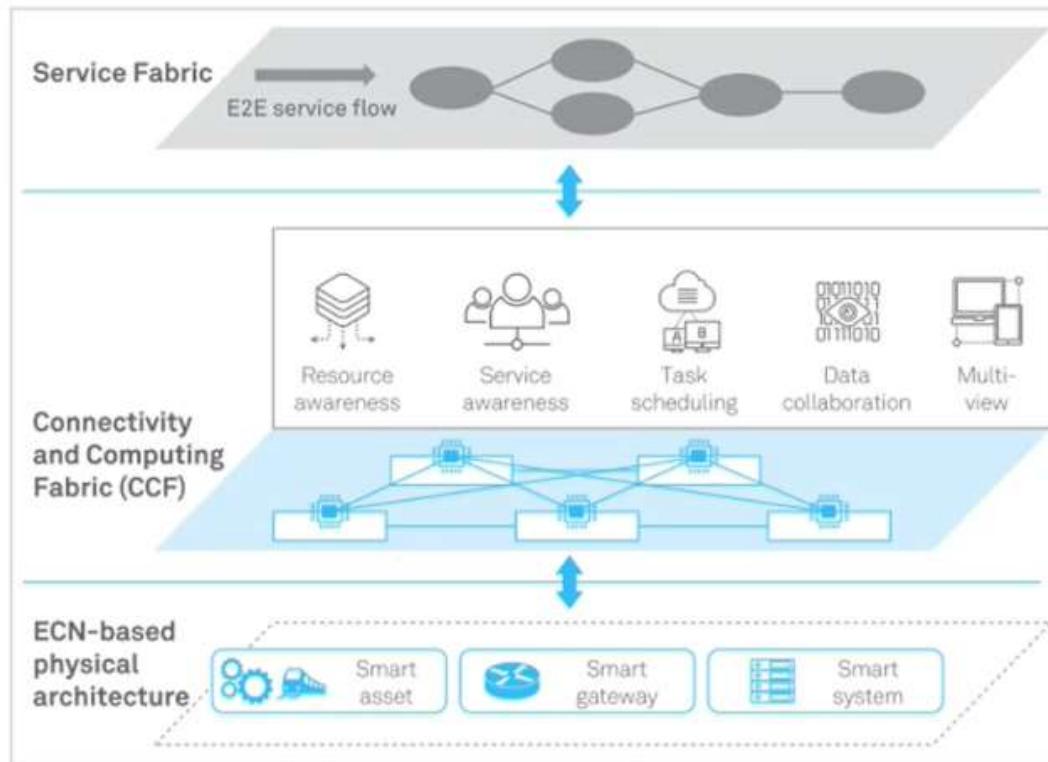
- Distributed storage
 - Data fragmentation
- Priority-based storage
 - Data processing
 - Data storage
- Fragment-based query optimization
 - Data segments are queried based on query conditions
- Open-source TSDBs, such as OpenTSDB, KairosDB, and InfluxDB

Functional View : Service Fabric

- The service model includes the following information:
 - Service name
 - Function to be executed or provided
 - Nesting, dependency, and inheritance relationships between services
 - Input and output of each service
- A service fabric provides the following functions:
 - Workflow and workload definition
 - Visualized display

Functional View : CCF

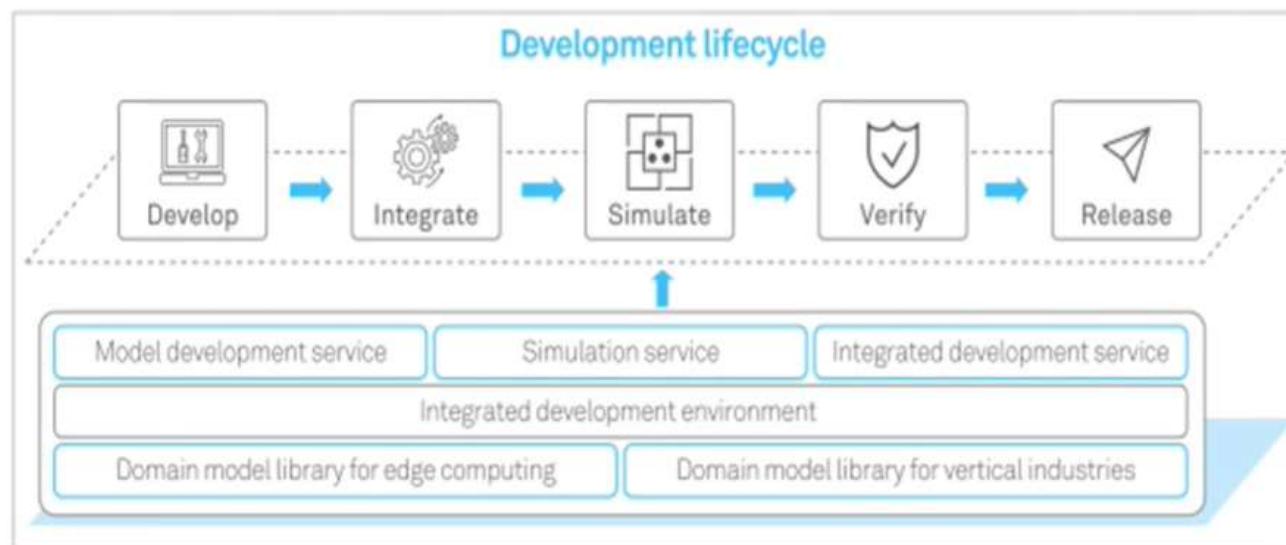
CCF is a virtualized connectivity and computing service layer



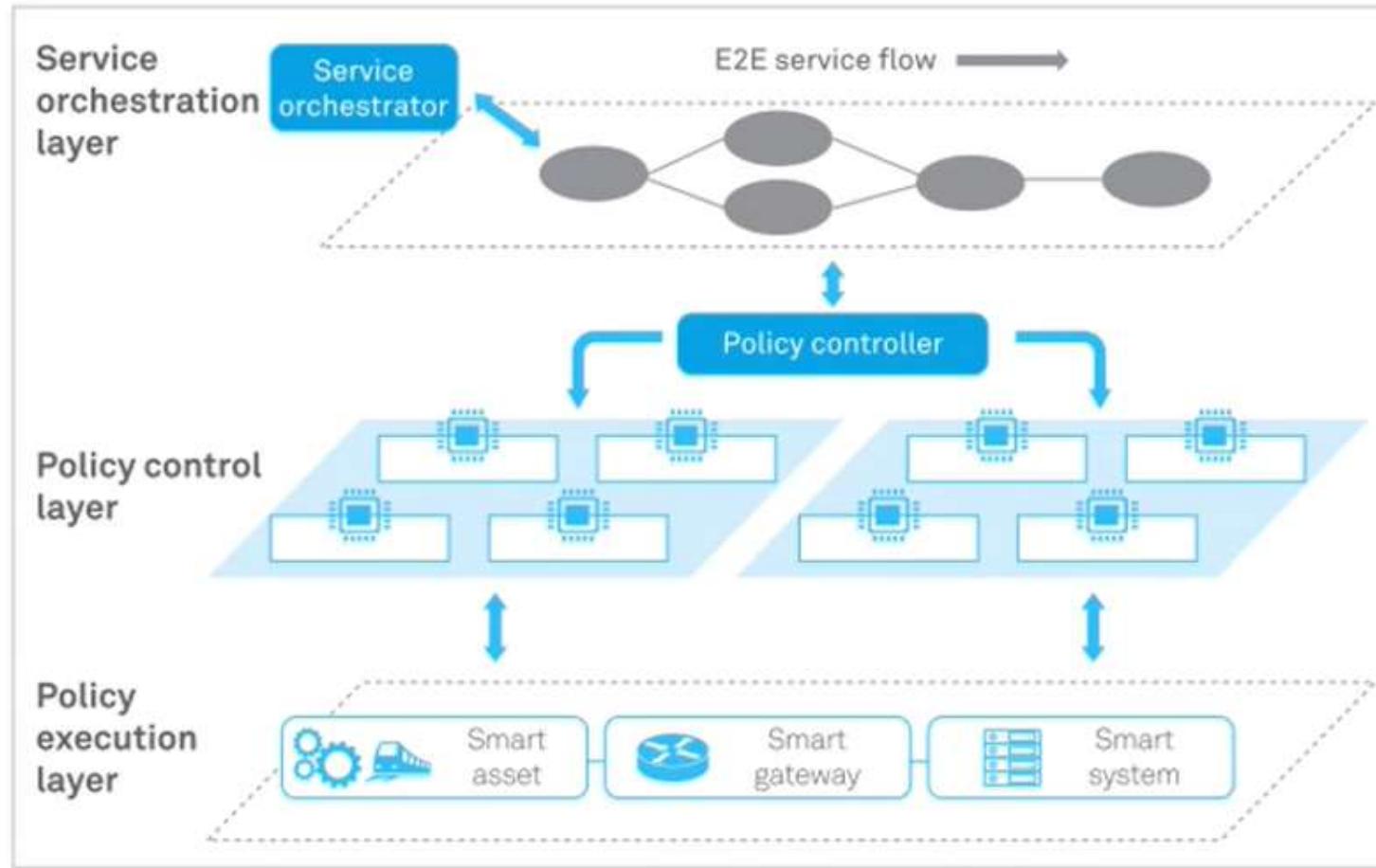
Function View : Development Service Framework



- The development service framework supports the following key services:
 - Model-based development service
 - Emulation service
 - Integrated release service



Function View : Development Operation Service Framework

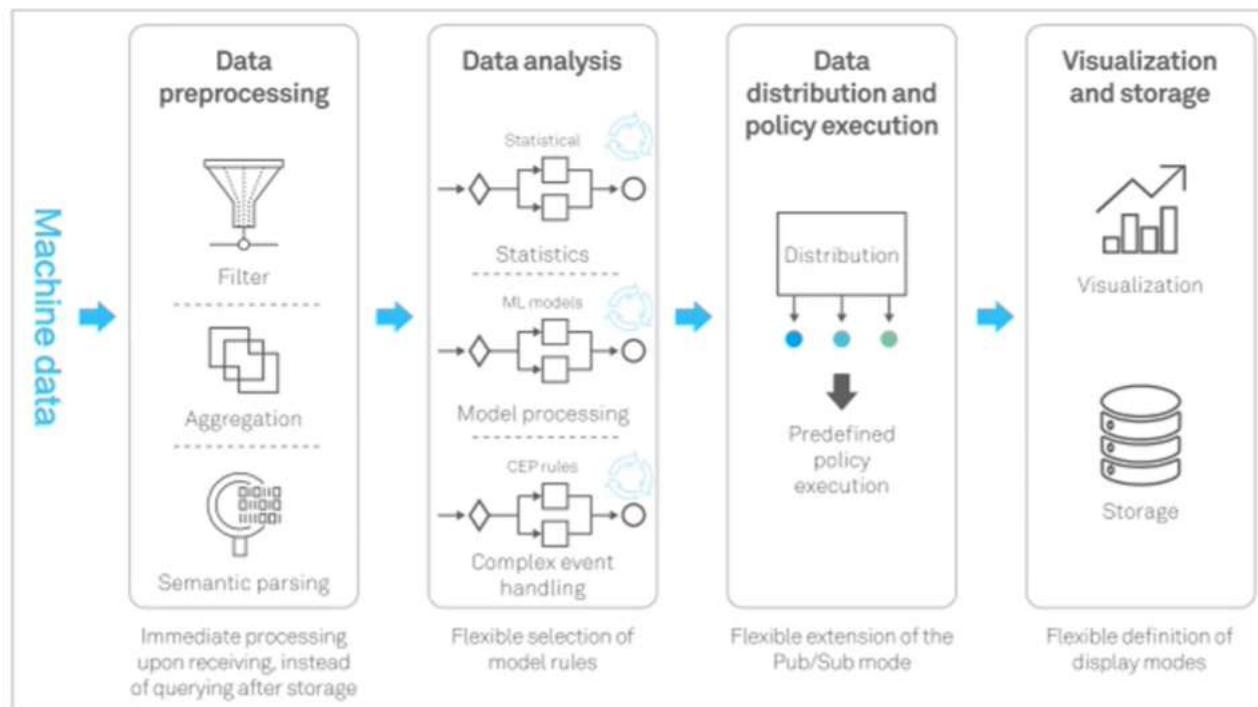


Function View : Development Service Framework



- Edge data characteristics
 - Causal relationship vs. association relationship
 - High reliability vs. low reliability
 - Small data vs. big data

Function View : Full-Life Service Framework



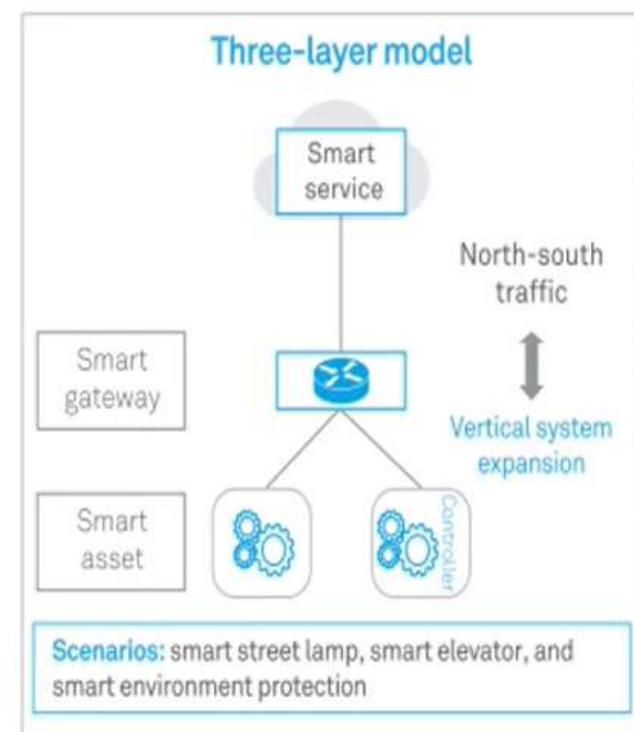
Function View : Security Service



- ECN security
- Network (fabric) security
- Data security
- Application security
- Identity and authentication management

Deployment View : Three-Layer model

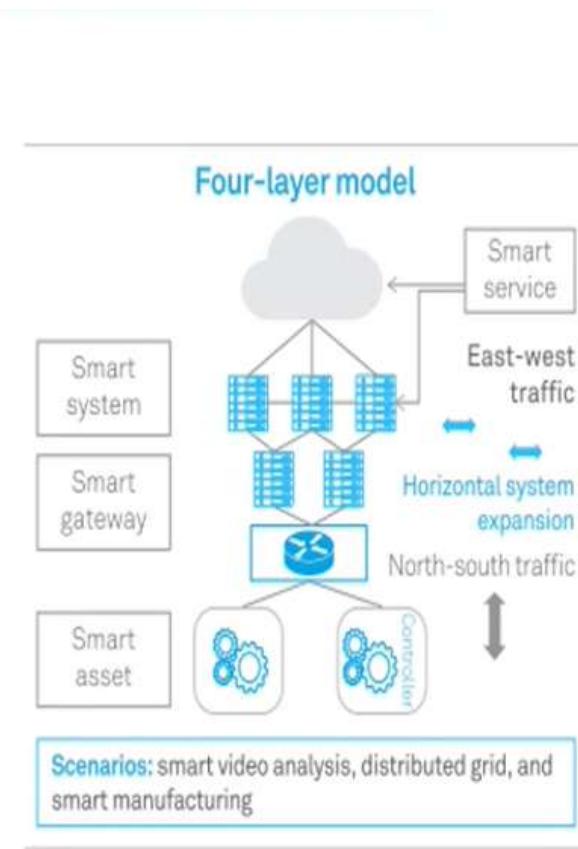
- This model is applicable to scenarios where services are deployed in one or more scattered areas, each with a low traffic volume.
- Scenarios include smart street lamps, smart elevators, and smart environmental protection.



Deployment View : Four-Layer model



- This model is applicable to scenarios where services are deployed centrally and the traffic volume is high
- Scenarios include smart video analysis, distributed grid, and smart manufacturing



Session 6

Edge Computing Reference Architecture

Edge computing reference architecture provides a framework for designing and implementing edge computing solutions.

It defines the components, relationships, and interactions necessary to deliver efficient and effective edge computing capabilities.

The architecture typically takes into account various considerations such as latency, bandwidth, data volume, security, and scalability. Keep in mind that edge computing is a rapidly evolving field, and reference architectures can vary based on specific use cases and technologies. Here's a general outline of an edge computing reference architecture:

Edge Devices or Sensors:

These are the devices that collect data at the edge of the network. Examples include IoT devices, sensors, cameras, and other data sources.

Edge Nodes/Gateways:

These are intermediate devices responsible for pre-processing and filtering data collected from edge devices. They can aggregate data, perform analytics, and make decisions locally before sending relevant information to the central cloud or data center.

Edge Computing Infrastructure:

This layer includes the computing resources (processors, memory, storage) deployed at the edge. It supports running applications, services, and analytics close to the data source, reducing latency and conserving network bandwidth.

Edge Management and Orchestration:

This component manages and orchestrates edge nodes and resources. It involves tasks such as provisioning, scaling, monitoring, and updating edge devices and applications remotely.

Edge Analytics and AI:

Running analytics and artificial intelligence (AI) algorithms at the edge can enable real-time insights and decision-making without relying heavily on the central cloud. This layer may include machine learning models, anomaly detection, and predictive maintenance.

Edge Connectivity:

This layer handles communication between edge devices, edge nodes, and potentially other edge computing elements. It may include protocols for device management, data synchronization, and communication.

Security and Privacy:

Security is crucial in edge computing due to the distributed nature of the architecture. This layer includes authentication, encryption, access control, and threat detection mechanisms to safeguard data and devices.

Data Management and Storage:

This component deals with storing and managing data generated at the edge. It might involve local storage solutions, data caching, and mechanisms to synchronize data with central repositories.

Interoperability Standards:

To ensure seamless integration between different components from various vendors, standards for communication, data formats, and APIs play a vital role.

Edge-to-Cloud Communication:

While the emphasis is on processing data locally, there's often a need to send aggregated or relevant data to the central cloud or data center for further analysis and storage.

Service Management and Orchestration (Optional):

In more complex scenarios, where multiple edge nodes and devices collaborate to provide a specific service, orchestration mechanisms manage the interaction between these entities.

Deployment Flexibility and Scalability:

The architecture should support flexible deployment options, enabling organizations to scale edge resources up or down as needed.

Remember, the specific components and their interactions can vary based on use cases such as industrial IoT, autonomous vehicles, smart cities, and more. It's essential to tailor the reference architecture to your organization's needs while considering factors like latency requirements, data volume, and the types of applications you're running at the edge.

Model-Driven Reference Architecture

A Model-Driven Reference Architecture (MDRA) is a framework that uses models and modeling techniques to guide the design, development, and implementation of complex systems or solutions. It provides a structured approach to designing systems based on well-defined models, ensuring consistency, traceability, and effective communication among stakeholders. MDRA is particularly useful for systems that are characterized by their complexity, variability, and the need for adaptability.

Here's a breakdown of the key components and concepts within a Model-Driven Reference Architecture:

Models:

Models are abstract representations of different aspects of a system. They can include conceptual, functional, structural, behavioral, and other types of models, depending on the characteristics of the system. Models provide a common language for communication and documentation.

Metamodels:

Metamodels define the structure, concepts, and relationships that can be used to create models. They establish a formal framework for constructing consistent and well-defined models. Metamodels often use modeling languages like UML (Unified Modeling Language) or SysML (Systems Modeling Language).

Model Transformation:

Model transformations are processes that take input models in one form and produce output models in another form. These transformations can be automated and are used to generate code, documentation, or other artifacts from the models.

Model-Based Development (MBD):

MBD involves creating and evolving software systems primarily through models and model transformations. Instead of coding directly, developers work with models that are then transformed into executable code.

Model-Driven Engineering (MDE):

MDE is a broader concept that encompasses MBD. It emphasizes the use of models throughout the entire software development lifecycle, from requirements analysis and design to implementation, testing, and maintenance.

Model-Driven Architecture (MDA):

MDA is a specific approach to MDE that's associated with the Object Management Group (OMG). It defines a set of standards for creating and using models to enable interoperability and portability of software across different platforms.

Model-Based Systems Engineering (MBSE):

MBSE applies model-driven techniques to systems engineering. It involves creating models that represent the various aspects of a system, including its requirements, functions, behaviors, and physical components.

Model Validation and Verification:

Ensuring the accuracy and reliability of models is crucial. Techniques for model validation and verification help identify inconsistencies, errors, or deviations from requirements within the models.

Traceability:

Traceability ensures that relationships between different model elements are well-defined and maintained. It allows stakeholders to track how requirements are satisfied by different parts of the system.

Reuse and Variability:

MDRA often promotes the concept of creating reusable models and components, which can be customized or combined to address different system variations.

Tool Support:

Various tools and platforms are available to support MDRA, ranging from modeling tools and model transformation engines to simulation and code generation tools.

MDRAs are valuable in various domains, including software engineering, systems engineering, and other complex domains where precise specification, analysis, and adaptation are required. They enable better collaboration among interdisciplinary teams, facilitate change management, and enhance the overall quality of the systems being developed.

Multi-View Display

A multi-view display refers to a technology that allows the simultaneous presentation of multiple visual perspectives or images on a single screen or surface. This technology is often used in various applications to provide users with a comprehensive view of different pieces of information, angles, or data sources at the same time. Multi-view displays are particularly beneficial when dealing with complex data, comparisons, or situations that require monitoring multiple sources of information.

Here are a few different aspects and applications of multi-view displays:

Multi-View Monitors:

These are computer monitors or displays designed to show multiple windows or applications side by side. They're commonly used in tasks that require multitasking, such as video editing, programming, financial analysis, and more.

3D Displays:

In the context of 3D technology, multi-view displays refer to screens that provide different perspectives of a 3D scene to each eye, creating a stereoscopic effect without the need for specialized glasses.

Video Walls:

Video walls consist of multiple displays arranged in a grid to create a larger combined display area. Each display can show different content or be part of a larger image or video.

Surveillance and Command Centers:

Multi-view displays are crucial in surveillance and command center environments, where operators need to monitor multiple security cameras, data feeds, and systems simultaneously.

Medical Imaging:

In medical applications, multi-view displays can show various medical images and data together, helping doctors analyze and diagnose patients more effectively.

Gaming and Entertainment:

Multi-view displays are used in gaming setups that require a panoramic or immersive view. They can also enable split-screen multiplayer gaming on a single screen.

Collaborative Workspaces:

Multi-view displays are useful in collaborative environments, allowing team members to share and compare data from different sources during meetings or brainstorming sessions.

Interactive Digital Signage:

Interactive displays in public spaces can offer multiple views or interactive elements to engage passersby with different types of content.

Automotive Displays:

Some modern vehicles are equipped with multi-view displays that provide drivers with different perspectives, such as rearview cameras, side-view cameras, and navigation information.

Simulation and Training:

Training simulators often use multi-view displays to replicate real-world scenarios, providing trainees with a comprehensive and immersive experience.

Multi-view displays can enhance productivity, decision-making, and user engagement by providing a more holistic view of information. They rely on technologies such as high-resolution screens, graphics processing, and advanced software to manage and present multiple views effectively. As technology continues to advance, multi-view displays are likely to become even more versatile and widespread in various industries.

Concept View

The term "Concept View" doesn't have a universally fixed definition but can refer to a perspective or representation that focuses on conceptual understanding and abstraction rather than concrete details. In various contexts, "concept view" might relate to different domains, such as software architecture, data modeling, or design. Here are a couple of potential interpretations:

Software Architecture:

In the context of software architecture, a concept view could refer to a high-level representation of the major components, modules, and interactions within a software system. It's an abstraction that helps stakeholders understand the system's structure and behavior without getting into technical details. Concept views are often used to communicate architecture decisions to non-technical audiences.

Data Modeling:

In data modeling, a concept view might involve creating an abstract representation of the relationships, entities, and attributes that define the data domain. It provides a conceptual understanding of the data without getting into the specifics of how it's stored or implemented.

User Interface Design:

In user interface design, a concept view could be a preliminary visual representation of a user interface. It's a rough sketch or mockup that conveys the overall layout, structure, and basic interactions of the user interface without delving into finer design details.

Education and Learning:

In education, a concept view could be a way of presenting complex ideas or theories in a simplified manner, focusing on key concepts and principles. This approach helps learners grasp the foundational aspects before diving into more intricate details.

Business Strategy:

In business strategy, a concept view might involve presenting high-level strategic ideas or plans without the detailed operational steps. It's a way to communicate the core concepts and goals of a strategy to stakeholders.

In essence, a concept view is about distilling complex ideas or systems into their essential elements to facilitate understanding, communication, and decision-making. It's a way of presenting information at a higher level of abstraction, making it accessible to a wider audience or for preliminary discussions before moving into more detailed perspectives. The specific interpretation of "concept view" can vary based on the context in which it's used.

ECNs, Development Frameworks, and Product Implementation

Engineering Change Notices (ECNs):

Engineering Change Notices (ECNs) are formal documents used in engineering and manufacturing to communicate changes to a product's design, specifications, or processes. ECNs are typically generated when modifications are needed for a product that is already in the development or manufacturing phase. This could involve changes to parts, materials, dimensions, manufacturing processes, or other aspects of a product. ECNs help ensure that all stakeholders, including design teams, manufacturing teams, and quality control, are informed about the changes and can implement them correctly. ECNs often go through an approval process before the changes are implemented.

Development Frameworks:

Development frameworks are structured environments or platforms that provide a set of tools, libraries, guidelines, and best practices to streamline the process of creating software applications. These frameworks offer a foundation for developers to build upon, saving time and effort by providing pre-built components, standardizing certain processes, and promoting a consistent structure. Development frameworks can be specific to various programming languages or application domains. Examples include web development frameworks like Ruby on Rails, frontend frameworks like React, and backend frameworks like Django.

Product Implementation:

Product implementation refers to the process of transforming a product concept or design into a tangible, functional product that can be manufactured, distributed, and used by customers. This involves taking the specifications and designs created during the development phase and executing the necessary steps to bring the product to market. Implementation includes tasks such as manufacturing, quality control, testing, packaging, distribution, and potentially post-launch support and maintenance.

These three concepts can be related in the context of product development:

ECNs can be relevant during the product implementation phase if changes are required to the design, manufacturing processes, or specifications of a product that is already in the implementation stage.

Development frameworks can provide a structured approach and tools to assist developers during the implementation of software products, helping them adhere to best practices and standards.

Product implementation is the overarching process that encompasses both physical product manufacturing and software development, and it may involve incorporating changes based on ECNs while adhering to the guidelines provided by development frameworks (in the case of software).

Managing ECNs effectively, choosing the appropriate development framework, and executing a successful product implementation are all crucial steps in the product development lifecycle, ensuring that products meet quality standards, customer requirements, and are delivered efficiently to the market.

Edge Computing Domain Models

Edge computing domain models are abstract representations that capture the essential components, relationships, and interactions within edge computing systems. These models help stakeholders, including architects, developers, and decision-makers, understand the structure and behavior of edge computing environments. Since edge computing spans various industries and use cases, domain models can vary based on specific contexts. Here's a general outline of key components often found in edge computing domain models:

Edge Devices:

These are the physical devices at the edge of the network, such as IoT sensors, cameras, drones, industrial machines, and mobile devices. They collect data and interact with the physical world.

Edge Nodes/Gateways:

Edge nodes or gateways serve as intermediate points between edge devices and central cloud or data centers. They can preprocess data, aggregate information, apply analytics, and make local decisions.

Edge Computing Infrastructure:

This includes computing resources like processors, memory, and storage deployed at the edge. These resources enable the execution of applications and services closer to the data source.

Edge Analytics and AI:

Edge computing often involves running analytics and AI algorithms on the edge devices or nodes. This allows real-time insights and decision-making without relying solely on centralized cloud processing.

Data Streams and Event Processing:

Edge environments generate a significant amount of data streams. Event processing mechanisms handle the ingestion, filtering, and transformation of data before it's forwarded for further processing.

Data Storage and Caching:

Edge devices or nodes might include local storage for caching frequently accessed data or storing critical information temporarily.

Connectivity and Protocols:

The domain model should represent the various communication protocols and technologies used for device-to-device, device-to-edge, and edge-to-cloud communication.

Security and Identity Management:

Security mechanisms, including authentication, encryption, access control, and device identity management, play a crucial role in edge computing domain models.

Orchestration and Management:

This aspect covers management tasks like provisioning, scaling, monitoring, updating, and configuring edge devices and resources.

Interoperability Standards:

Representing the standards and protocols that enable interoperability between different devices and systems at the edge.

Latency and Quality of Service (QoS):

Edge computing is often chosen to minimize latency. Models may depict how latency requirements are met and how QoS is ensured.

Edge-to-Cloud Communication:

Detailing how data flows between the edge and the central cloud, including data synchronization, aggregation, and offloading.

Deployment Flexibility and Scalability:

Models should capture how edge solutions can be scaled up or down based on demand while maintaining efficiency.

Application Lifecycle Management:

Addressing how applications are developed, deployed, and managed at the edge, including software updates and version control.

Use Case-Specific Components:

Depending on the domain (e.g., industrial, healthcare, smart cities), models might include domain-specific components like robotics, remote monitoring, smart grids, etc.

Creating effective edge computing domain models requires a deep understanding of the specific use case, industry, and technologies involved. These models serve as communication tools, guiding the design, development, and deployment of edge computing solutions.



BITS Pilani
Pilani Campus

BITS Pilani presentation

Paramananda Barik
CS&IS Department





SEZG586/SSZG586, Edge Computing Lecture No.7

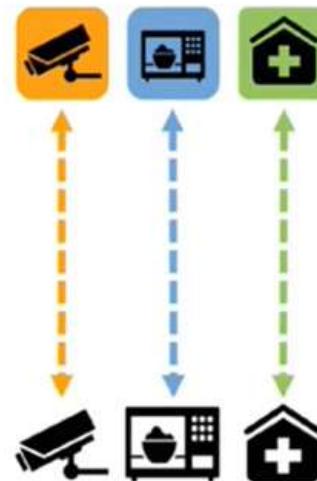
Case Study: EdgeOS_H

A Home Operating System for Internet of Everything

Smart home design

The lack of a home operating system makes it very difficult to manage devices, data, and services.

Systems work in a silo-based manner and cannot be connected or communicate with other systems

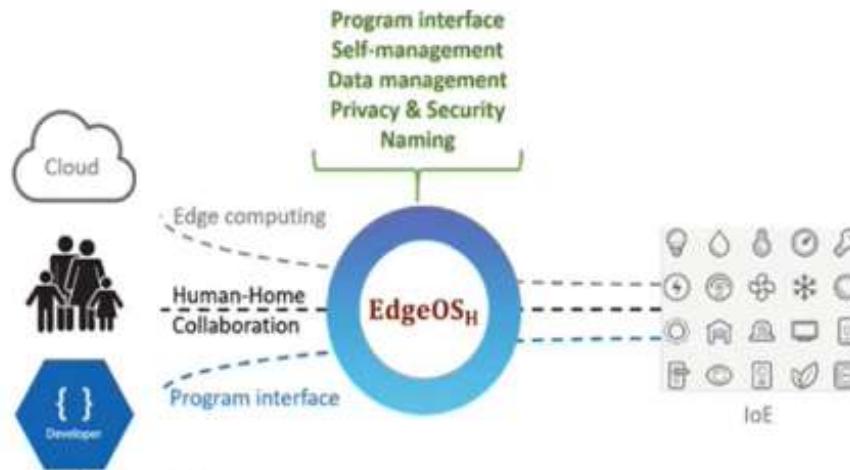


EdgeOS_H Overview and Design



Benefits

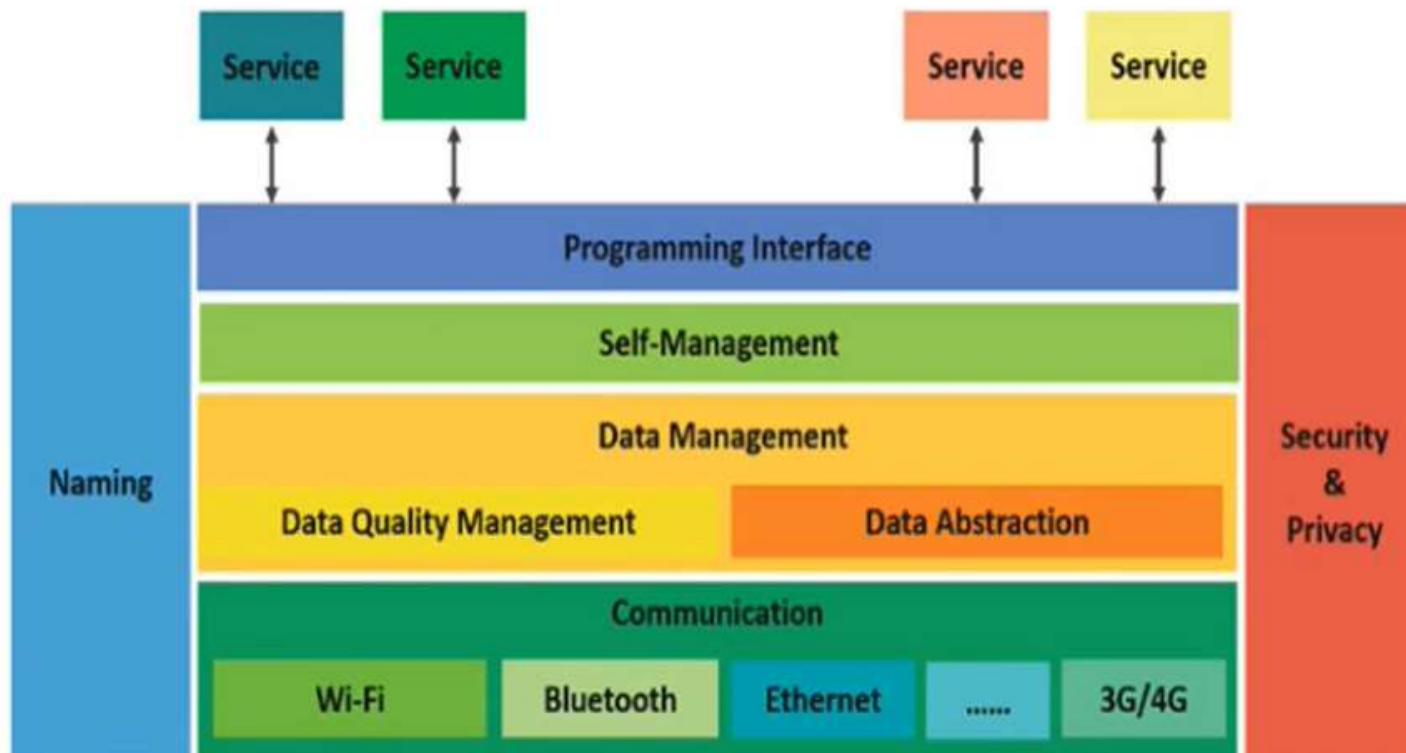
- Network load could be reduced
- Service response time could be decreased
- Data is better protected



Challenge

- Home environment is very dynamic - hardware provided by different manufacturers

Overview

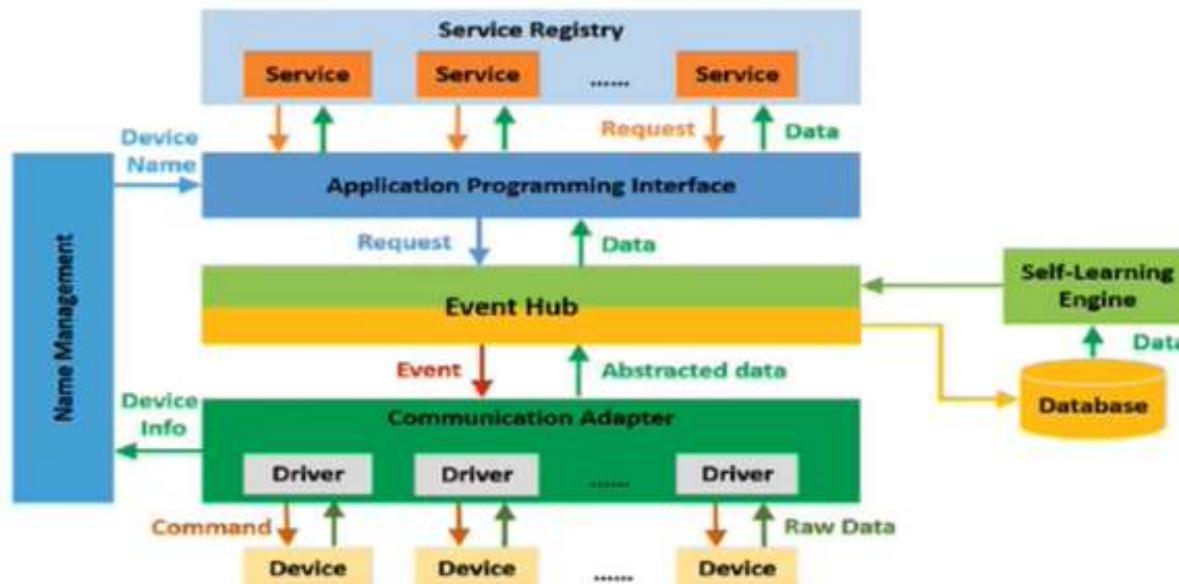


Layered approach

- **Communication**
 - Collected data needs transportation
- **Data Management**
 - Data fusion and massage
- **Self-Management**
 - Device registration, maintenance, and replacement
- **Programming Interface**
 - Provide performance for user applications
- **Naming**
- **Security & Privacy**
 - data security and privacy

The design consists of 7 components:

- Communication Adapter, Event Hub, Database, Self-Learning Engine
- Application Programming Interface, Service Registry, Name Management



7 Components



Communication Adapter gets access to devices by the embedded drivers

- Drivers - responsible for sending commands to devices and collecting state data
- Combines different communication methods and provide a uniform interface for upper layers

Event Hub maps two layers in the logical view:

- Data Management and Self-Management layers
- Responsible for capturing system events and sending instructions to lower levels



7 Components

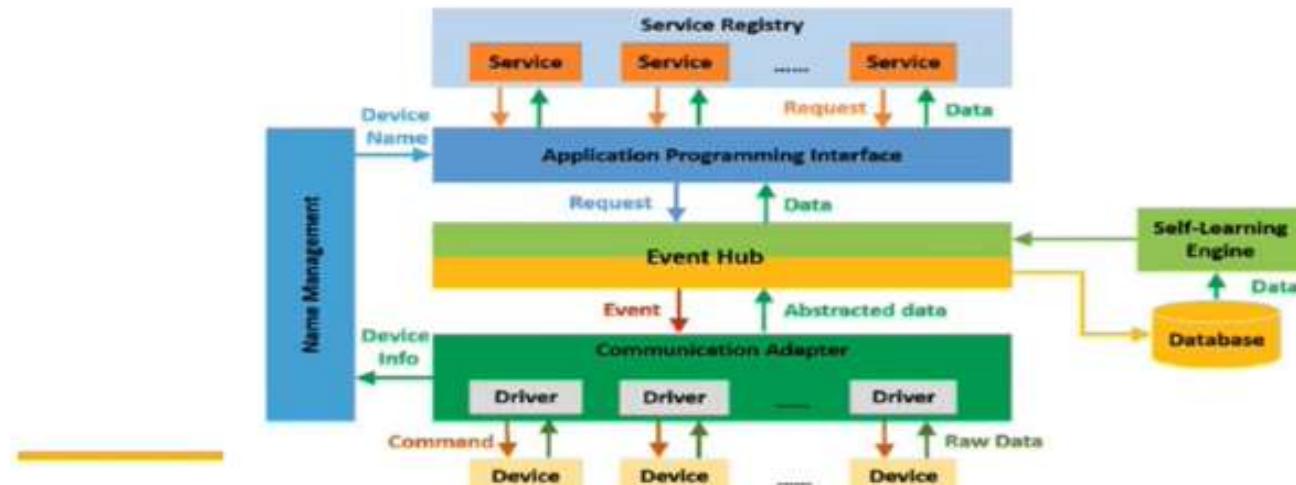


Database is another layer in Data management layer

- Event Hub stores data in the Database
- Stored data is utilized by the Self-Learning Engine

Self-Learning Engine

- creates a learning model
- analyze user behavior, generate the personal model for the user



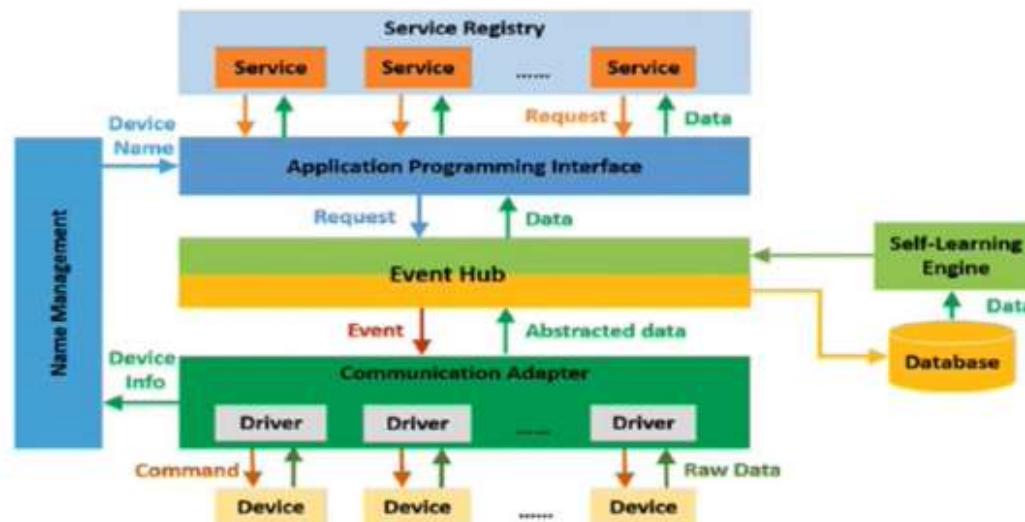
7 Components

Application Programming Interface (API) and Service Registry

- located in the upper layers of the system and are utilized for third-party services.

Name Management

- Helps the system keep devices organized
- A name for it using the following rule: location (where), role (who), and data description (what)



Challenges and Opportunities of Edge Computing



- Programmability
- Naming
- Data abstraction
- Service management
- Application distribution
- Scheduling Strategies
- Privacy and Security
- Business model
- Optimization metrics

Programmability



Programming on Cloud

Users program and deploy the code on the cloud

Who decides, where is it computed?

Program written in

- One programming language

- Compiled for a certain target platform

- Runs in the cloud



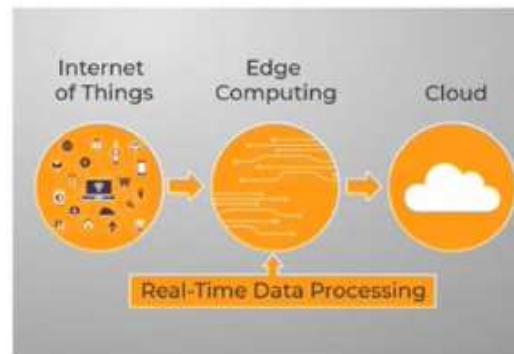
What happens in Edge?



Computation is offloaded from the cloud or ?? Devices

Edge nodes are most likely ?? Platforms

Runtime on these nodes might differ



What happens in Edge?

How is this addressed?

Computing Stream - a serial of functions/computing applied to the data

The function can be reallocated

The data and state along with the function should also be reallocated

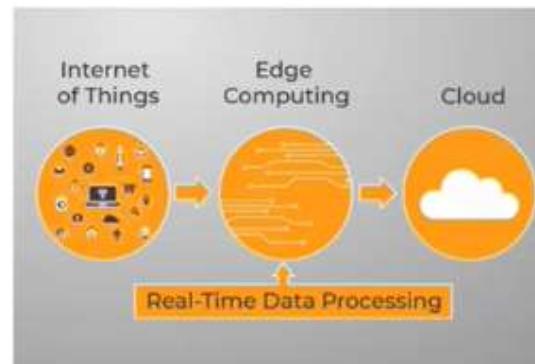
Software-defined computing flow

Data Processed

Data generating devices

Edge nodes

Cloud environment



Example -1

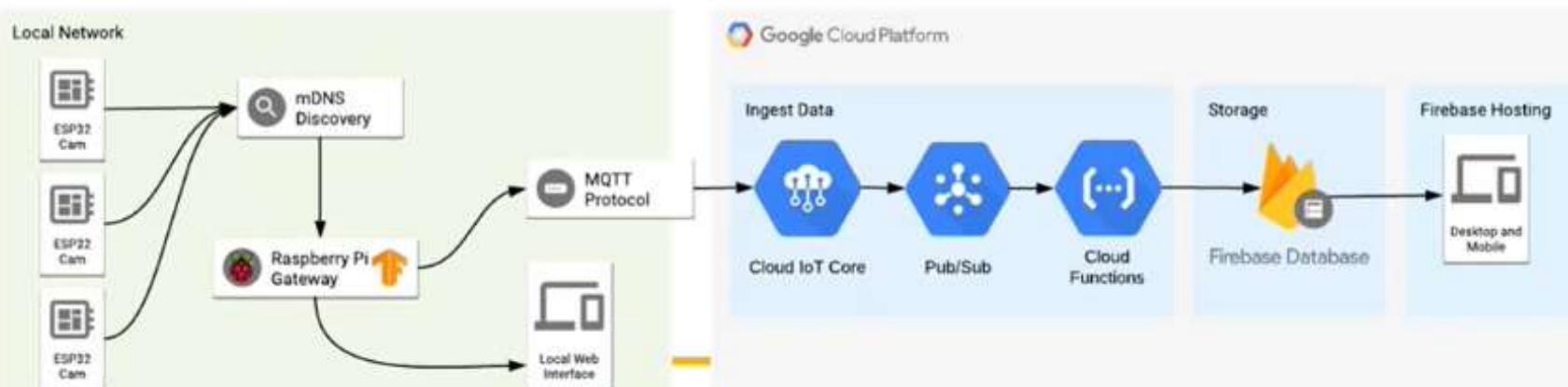
ESP32 module with an integrated camera

Raspberry Pi board as a local server

image classification using **Tensorflow**

Classified data is sent to the cloud securely using **Cloud IoT Core**

Data processed using **Firebase Cloud Functions**,
Data stored on **Firebase**



Example 2 – AWS IoT Greengrass for Windmill monitoring

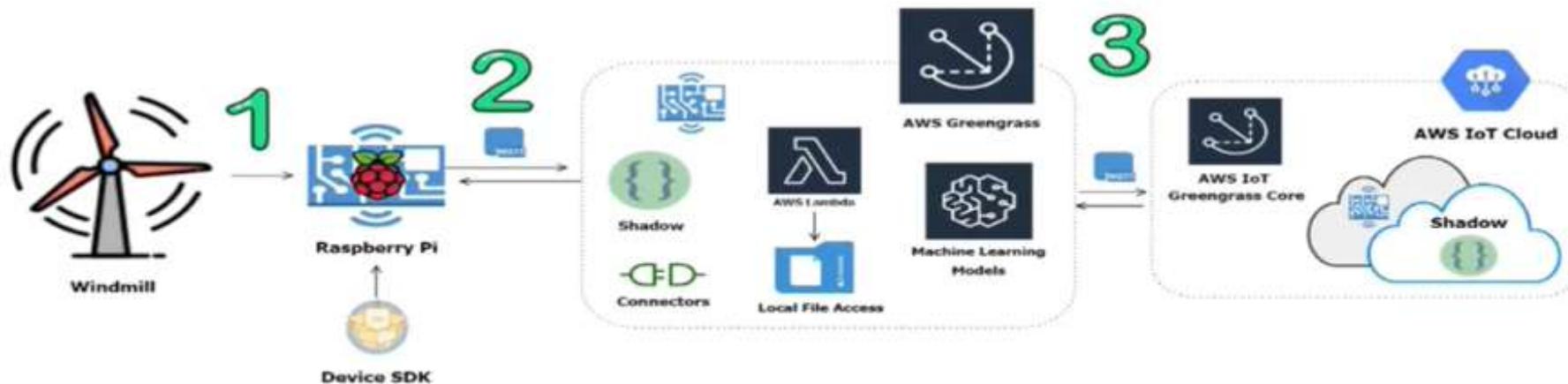


Local Greengrass Core Device: Sensor data (temperature, humidity and velocity) is collected and published using MQTT protocol.

A Lambda function which is running on Greengrass Core subscribes to that topic payload (Sensors Data)

Data stored it on local file storage

As data is being collected, it is analyzed for future Predictive Maintenance



Example 3: Face Recognition Model at the Edge with AWS IoT Greengrass



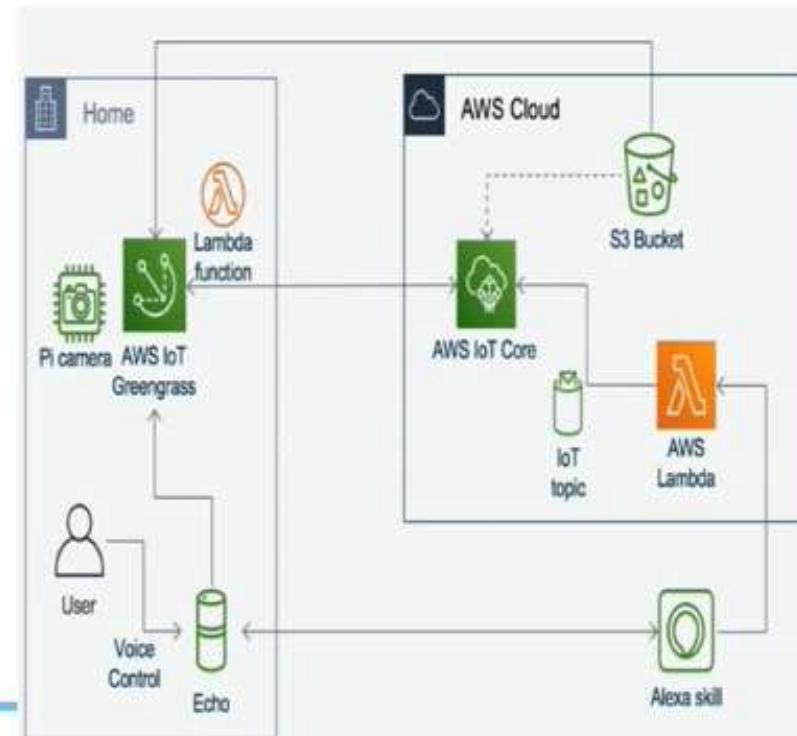
The facial recognition model and datasets uploaded to an Amazon S3 bucket

Used to create AWS Lambda function for recognition

AWS IoT Greengrass synchronizes the required files to the Raspberry Pi.

Echo Dot runs as a trigger. When Echo Dot listens to a command such as, "Alexa, open Monitor," it calls an Alexa skill to send a message to AWS IoT Core.

AWS IoT Core invokes the recognition Lambda function, which is deployed on Raspberry Pi local storage, and if the Lambda function recognizes the identity of the guest, the door opens.



Naming

Why is naming of things important?

- Addressing
- Things identification
- Data communication
- Programming

The naming scheme for Edge computing needs to handle:

- Mobility of things
- Highly dynamic network topology
- Privacy and security protection
- Scale

Naming : Edge OS

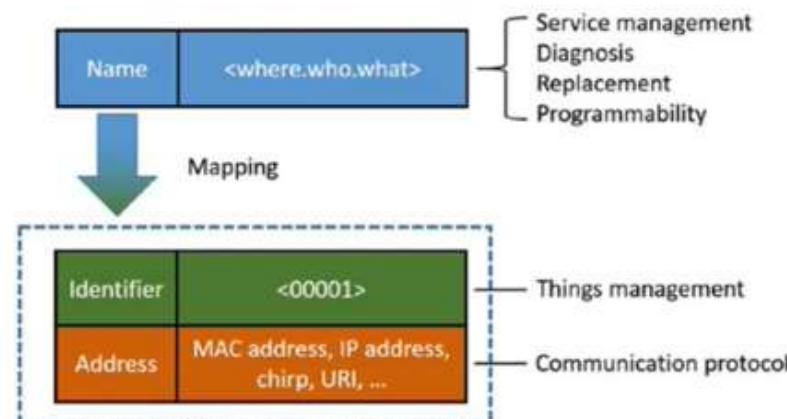
Naming mechanism in EdgeOS:

EdgeOS assign the network address to each thing

Human-friendly name which describes the following information: location (where), role (who), and data description (what)

Example: kitchen.oven2.temperature3

EdgeOS will assign identifier and network address to this thing

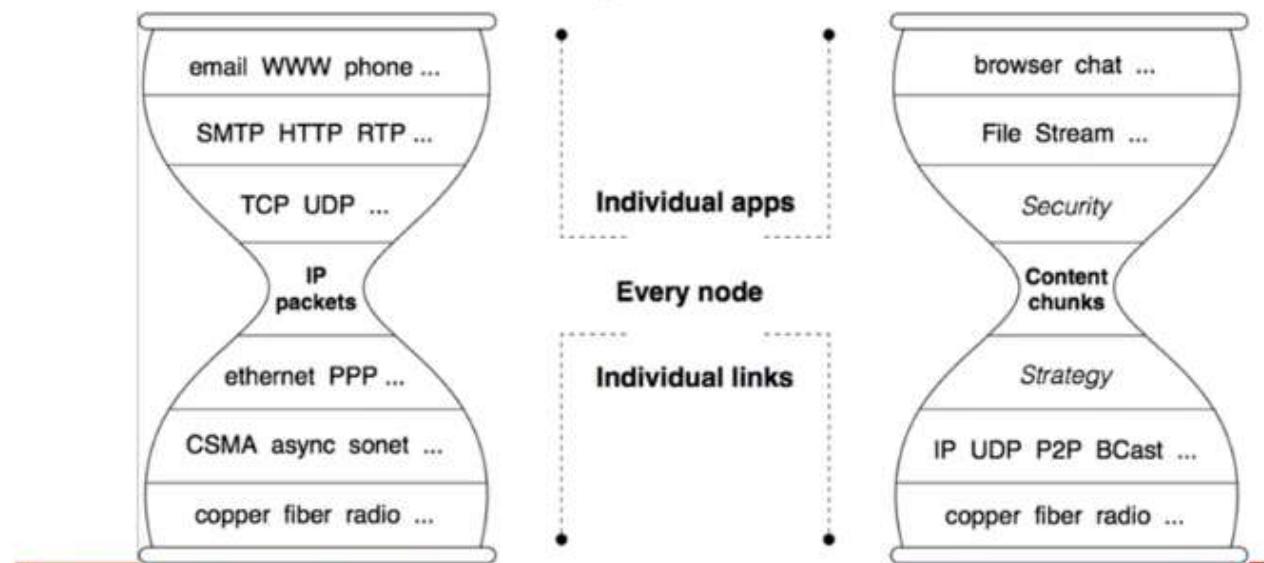


New naming mechanism

Named Data Networking

NDN - hierarchically structured name for content/data-centric network

Extra proxy required to fit into other communication protocols such as BlueTooth or Zigbee

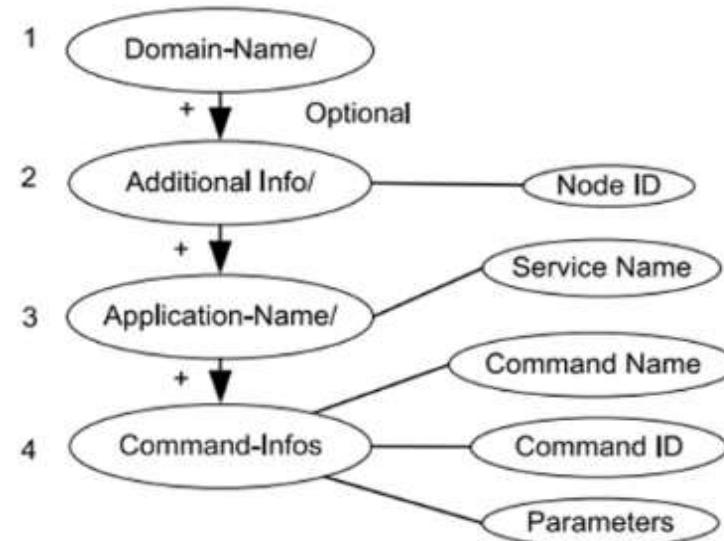


Naming : NDN (Named Data Networking)

The data in IoT-NDN are addressed by names.

Requesting data is based on hierarchically structured approach.

The names contain human-friendly components and are location independent



Data Abstraction



Large number of things report data

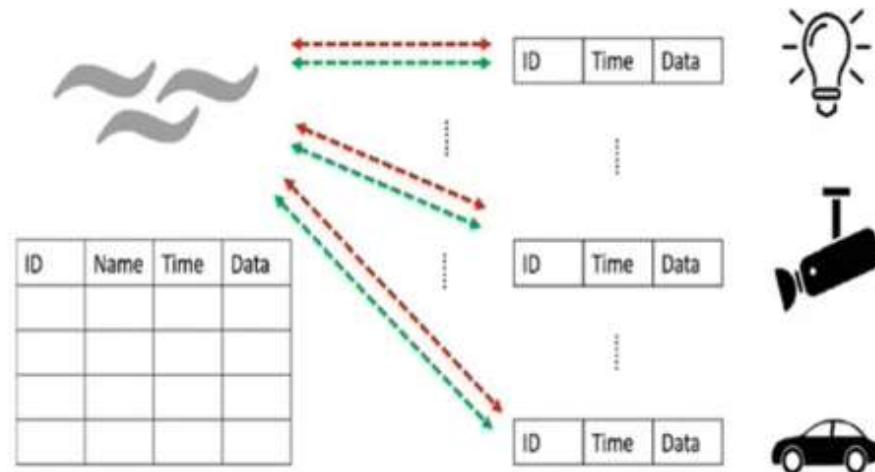
- Ex: a. Thermometer reports the temperature every minute, but this data will be consumed by the real user only few times a day
- b. Recording video using security cameras

Challenges in Edge: Edge node should consume/process all the data and interact with users proactively

- a. Data reported from different things comes with various formats
- b. Difficult to decide the degree of data abstraction

Data Abstraction : Challenges in Edge

- a. Data reported from different things comes with various formats
 - I. Gateway should not see raw data
 - II. Extract the knowledge from an integrated data table



Data Abstraction : Challenges in Edge



b. Difficult to decide the degree of data abstraction

- I. Too much raw data is filtered out – effects learning
- II. Keep a large quantity of raw data – effects storage
- III. Unreliable data - low precision sensor, hazard environment, weak wireless connection

How to abstract useful information from the unreliable data source is still a challenge for IoT developers?

Service Management: Differentiation



Services to have different priorities

Smart Home: critical services - things diagnosis and failure alarm should be processed earlier

Health-related service: fall detection or heart failure detection higher priority than other services

Service Management : Extensibility



When you buy a new mobile device and connection?

Can a new **Device** be easily added to the current service without any problem?

Solution: to design service management layer

- flexible and extensible

Service Management : Isolation



What if an application fails or crashes?

What should happen to the system?

EdgeOS:

If one application failed or was not responding?

User should still be able to control the lights

When a user removes the only application that controls lights from the system?

Lights should still work

How to isolate a user's private data from third-party applications?

Service Management : Reliability



From the service point of view

Sometimes difficult to identify the reason for a service failure accurately

Ex: if an air conditioner is not working

reasons:-

- power cord is cut
- compressor failure
- temperature controller has run out of battery
- sensor node lost connection to battery outage
- bad connection condition
- component wear out

Service Management : Reliability



From the data point of view

Reliability in

data sensing

low battery

physical damage

data in communication

unreliable communication protocols

using HTTP for communication

using Message Queuing Telemetry

transport (MQTT) - QoS0 (At most once)

Privacy and Security



User privacy and data security protection are the essential services

Private information can be learned from the sensed usage data

Computing at the edge of the data resource – decent method to protect privacy and data security

Challenges



- Awareness of privacy and security

WiFi networks security

49% of WiFi networks are unsecured

80% of routers set on default passwords

89% of public WiFi hotspots are unsecured

Devices like IP camera, health monitor, or even some WiFi enabled toys can be easily connected and misused

Challenges



- Ownership of the data collected from things @Edge
 - Data collected by wearables - stored and analyzed at the service provider side
 - Private photos and Videos

Storing data at the Edge device which is owned by the user provides better privacy protection

User should be able to control if service providers should use the data by process of authorization

Application Distribution



How to distribute the individual applications to various Edge nodes?

The current approaches for application distribution can be divided into two categories:

- dynamic: Hadoop distributed system
- static: Messaging Passing Interface (MPI)

Application distribution approaches for Edge computing:

Cloud-Edge

Edge-Edge

Scheduling Strategies



Scheduling Strategies help in the following:

- optimize the utilization of the resource
- reduce the response time
- improve energy efficiency
- improve the efficiency of task processing

Scheduling Strategies



“One size does not fit all”

Scheduling strategies of Edge computing need to be designed according to:

Different applications

Based on the heterogeneous of the resources like:

Data

Computing

Storage

Network

Demo: EdgeCloudSim or PureEdgeSim

Business Model



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

Optimization Metrics



To choose an optimal allocation strategy
optimization metrics

Latency

Bandwidth

Energy

Cost



BITS Pilani
Pilani Campus

BITS Pilani presentation

Paramananda Barik
CS&IS Department





SEZG586/SSZG586, Virtualization & Containers

Lecture No.8

VMs and Containers

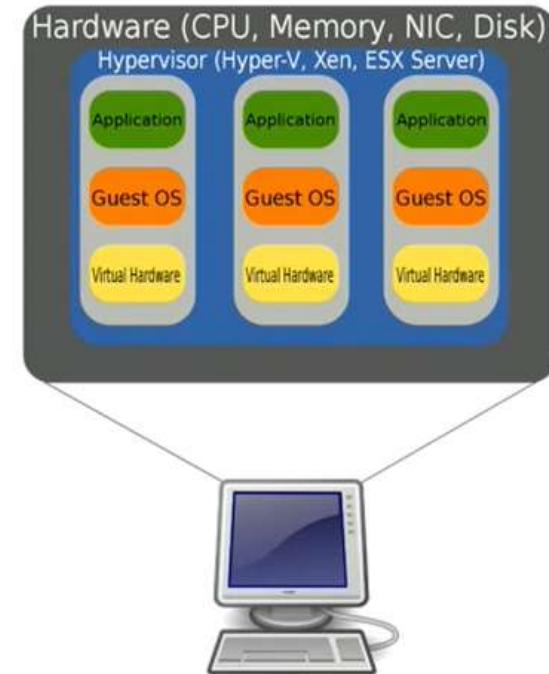
- **What is Virtualization?**

- Virtualization is technology that lets users create useful IT services using resources that are **traditionally bound to hardware**.
- It allows users to use a **physical machine's full capacity** by distributing its capabilities among many users or environments.
- Virtualization and cloud computing are not interchangeable.
- Virtualization is software that makes computing environments independent of physical infrastructure.



Virtual Machine

- A virtual computer system is known as a “virtual machine” (VM): a tightly isolated software container with an operating system and application inside.
- Each self-contained VM is completely independent.
- Putting multiple VMs on a single computer enables several operating systems and applications to run on just one physical server, or “host.”



What is a Hypervisor?

A hypervisor is the software layer that coordinates VMs.

Interface between Virtual Machine and physical hardware.

Virtual Machine isolation

Installation

- Directly on Hardware
- On a operating system

Examples – KVM (Kernel Based Virtual Machine), Xen, Hyper-V, ESXi, Virtual Box, Parallels for Mac

Types of Hypervisor?

Type 1

- Also known as “bare-metal”
- Interact directly with the hardware of the host machine
- Separate software tool is used to create and manipulate VM’s
- Used in Data Centres

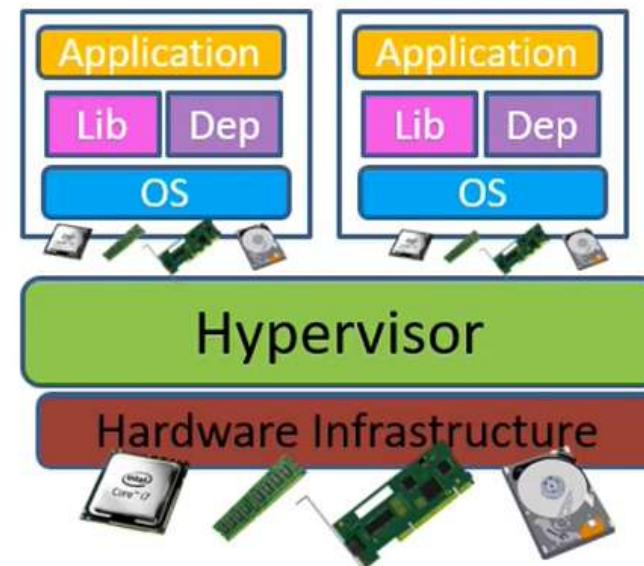
Pros:

- Highly efficient
- Increased Security

Cons:

- Need of separate management machine

Examples: Hyper-V, Xen, KVM, ESXi



Types of Hypervisor?

Type 2

- Also known as “Hosted”
- Doesn’t run directly on the underlying hardware
- Runs as an application in an OS
- Suitable for individual PC users

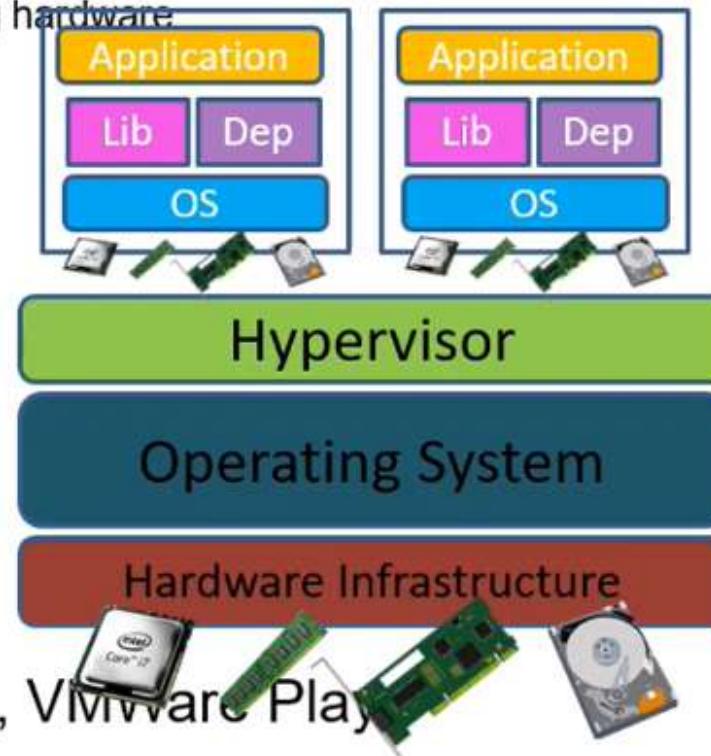
Pros:

- Quick alternate OS

Cons:

- Introduces latency
- Reduces performance

Examples: Virtual Box, QEMU, VMware Play



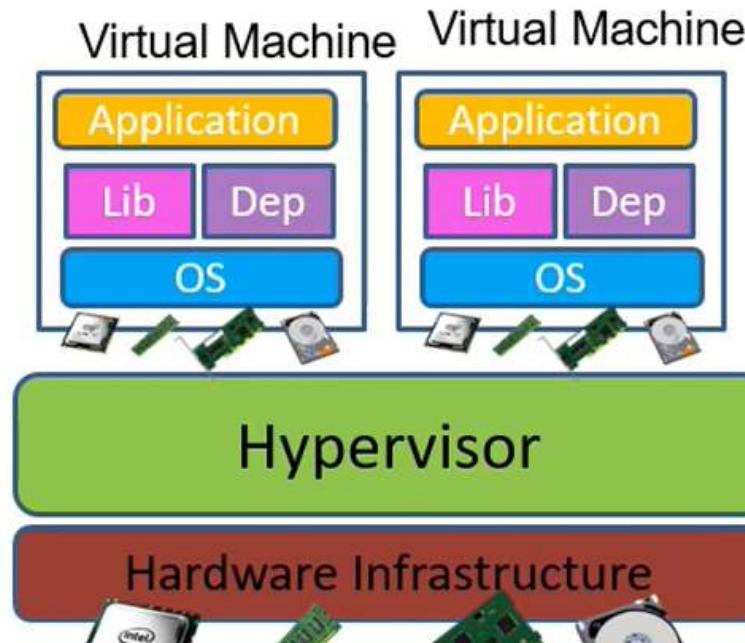
Properties of Virtual Machines

Partitioning

Isolation

Encapsulation

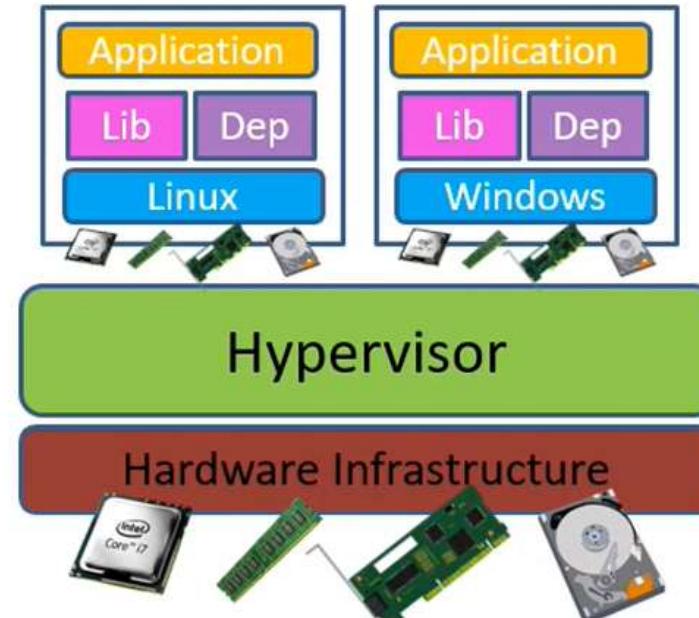
Hardware Independence



Properties of Virtual Machines

Partitioning

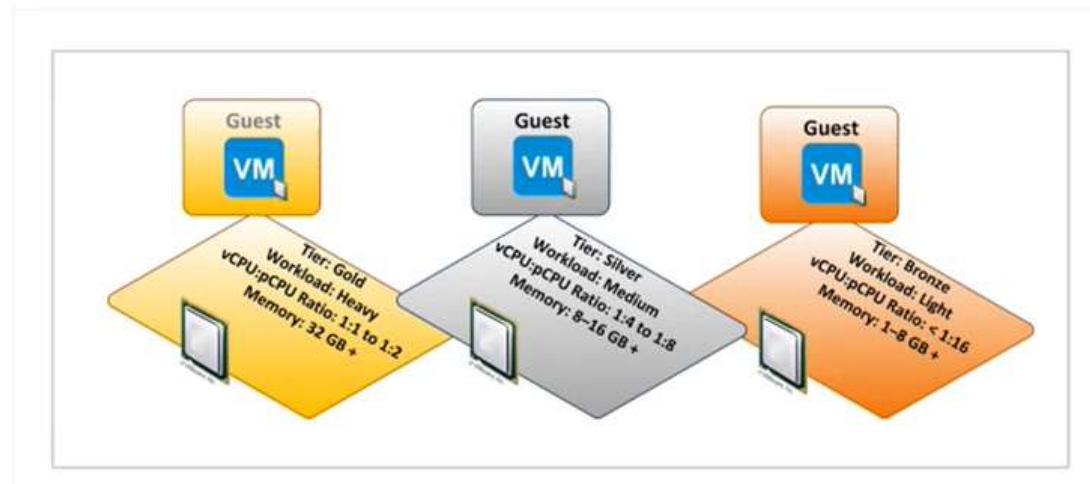
- Run multiple operating systems on one physical machine.
- Divide system resources between virtual machines.



Properties of Virtual Machines

Isolation

- Provide fault and security isolation at the hardware level.
- Preserve performance with advanced resource controls.



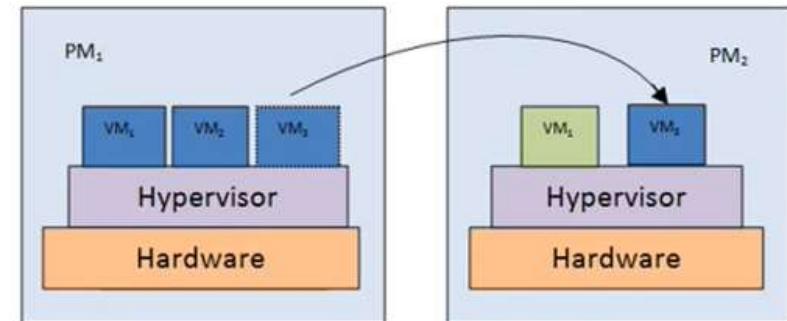
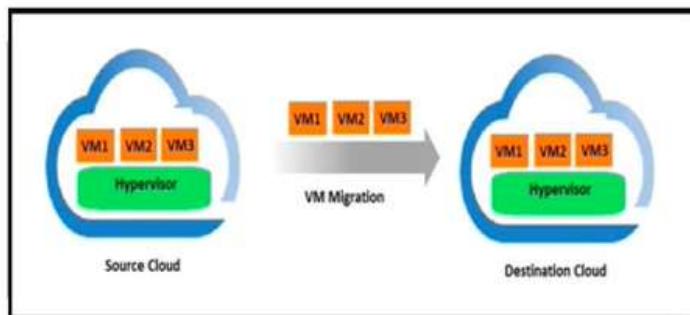
Properties of Virtual Machines

Encapsulation

- Save the entire state of a virtual machine to files.
- Move and copy virtual machines as easily as moving and copying files.

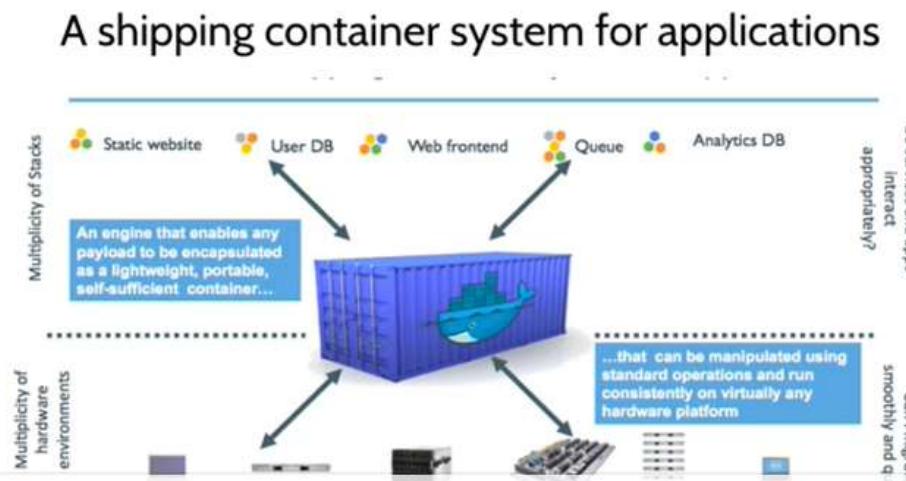
Hardware Independence

- Provision or migrate any virtual machine to any physical server.



What is Container?

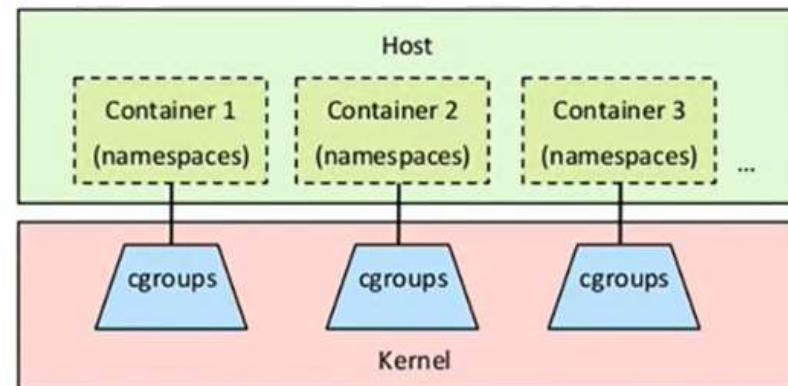
A software container is a standardized package of software. Everything needed for the software to run is inside the container. The software code, runtime, system tools, system libraries, and settings are all inside a single container



Container History

Linux Containers are powered by two underlying Linux Kernel technologies:

- cgroups
- namespaces



cgroups

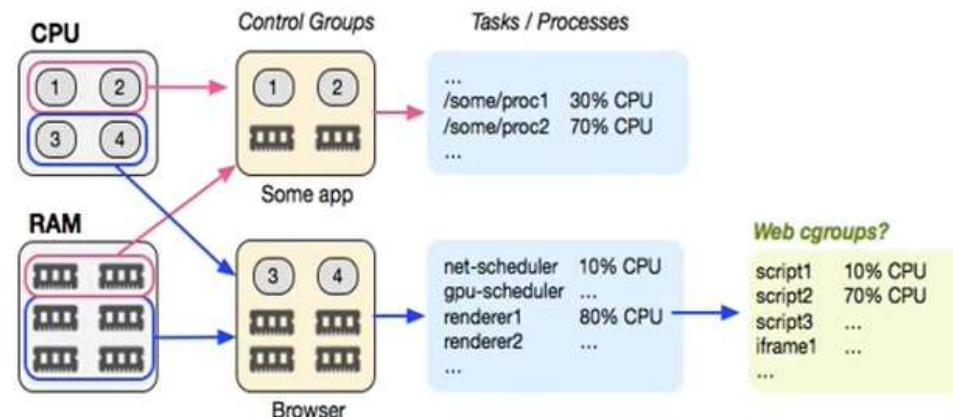
cgroups – Control groups

- A kernel mechanism for limiting and measuring the total resources used by a group of processes running on a system
- Processes can be applied with CPU, memory, network or IO quotas

Cgroup merged into Linux 2.6.24

Cgroups provides:

- **Resource limiting**
- **Prioritization**
- **Accounting**
- **Control**



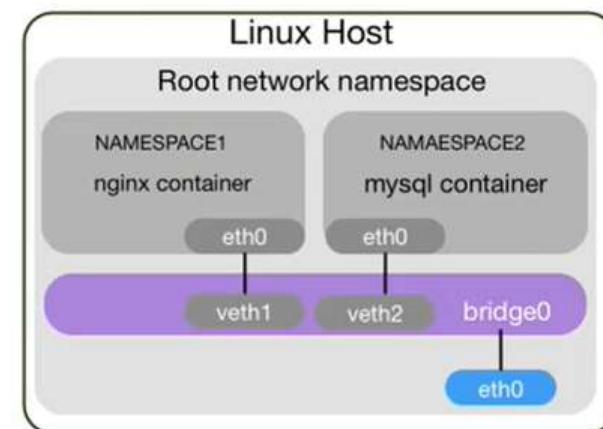
Namespaces

Namespaces - kernel mechanism for limiting the visibility that a group of processes has of the rest of a system

Limit visibility

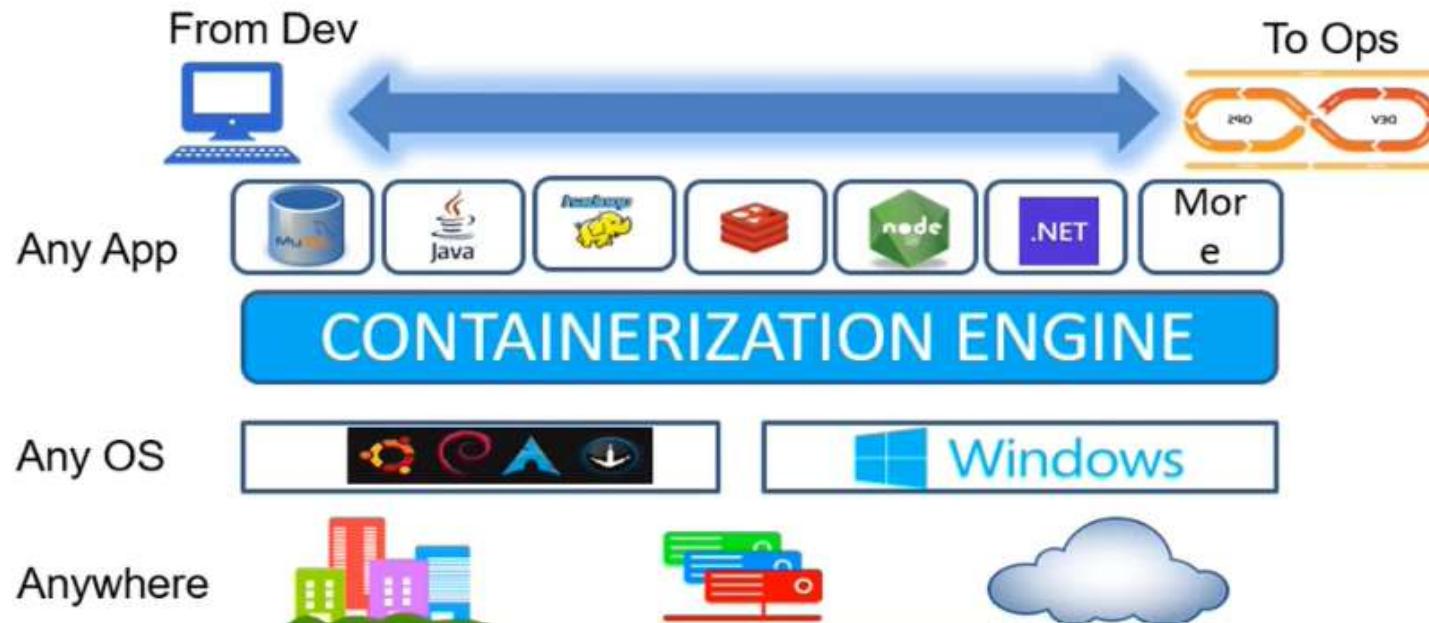
- Certain process trees
- Network interfaces
- User IDs
- Filesystem mounts

Namespace merged into Linux 3.8



What is Containers?

- A software container is a standardized package of software.
- Everything needed for the software to run is inside the container.
- The software code, runtime, system tools, system libraries, and settings are all inside a single container



The Docker Platform

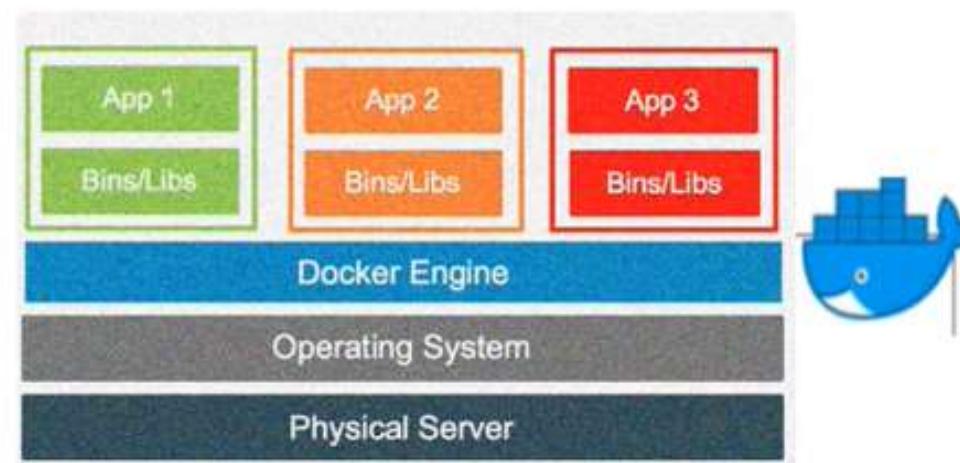
Docker is an open platform

Docker separates applications from hardware infrastructure

Containers are used to package and run an application

A single host can run many containers simultaneously

Containers are lightweight and contain everything needed
to run the application



The Docker Platform

Docker provides tooling and a platform to manage the lifecycle of your containers:

- Develop application using containers.
- Distributing and test application using containers.
- Deploy application into production environment, as a container or an orchestrated service.

Containers are great for continuous integration and continuous delivery (CI/CD) workflows.

The Docker Architecture

Docker uses a client-server architecture.

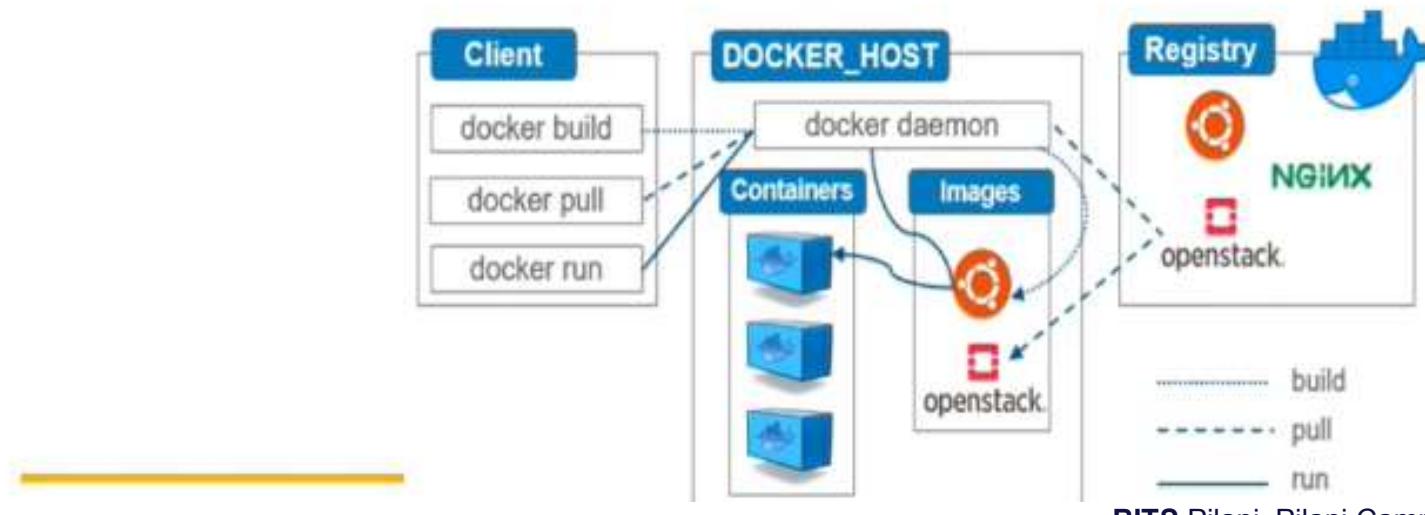
The Docker daemon

- The Docker daemon (`dockerd`) listens for Docker API requests
- Manages Docker objects such as images, containers, networks, and volumes.
- Builds, runs, and distributes containers

The Docker client

- The Docker *client* talks to the Docker *daemon*
- The Docker client and daemon *can* run on the same system
- The Docker client can communicate with more than one daemon..

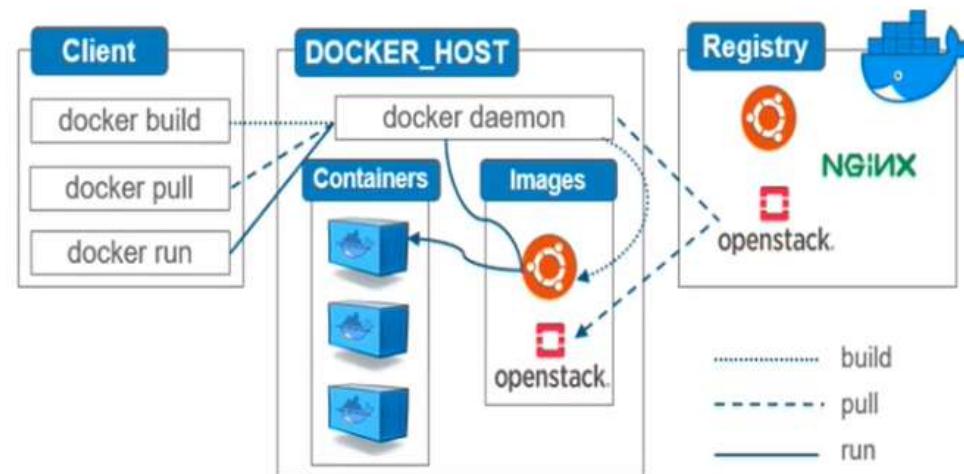
The Docker client and daemon communicate using a REST API, over UNIX sockets or a network interface.



The Docker Architecture

Docker registries

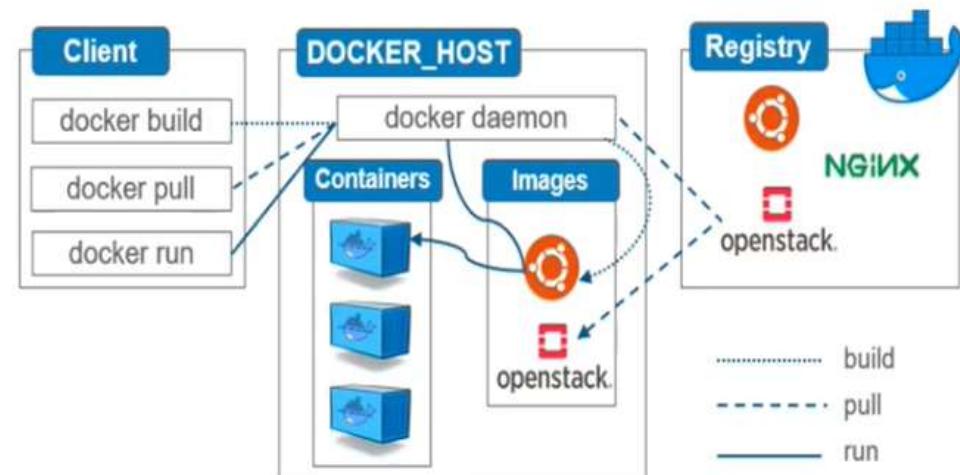
- A Docker *registry* stores Docker images.
- Docker Hub is a public registry that anyone can use.
- Docker is configured to look for images on Docker Hub by default



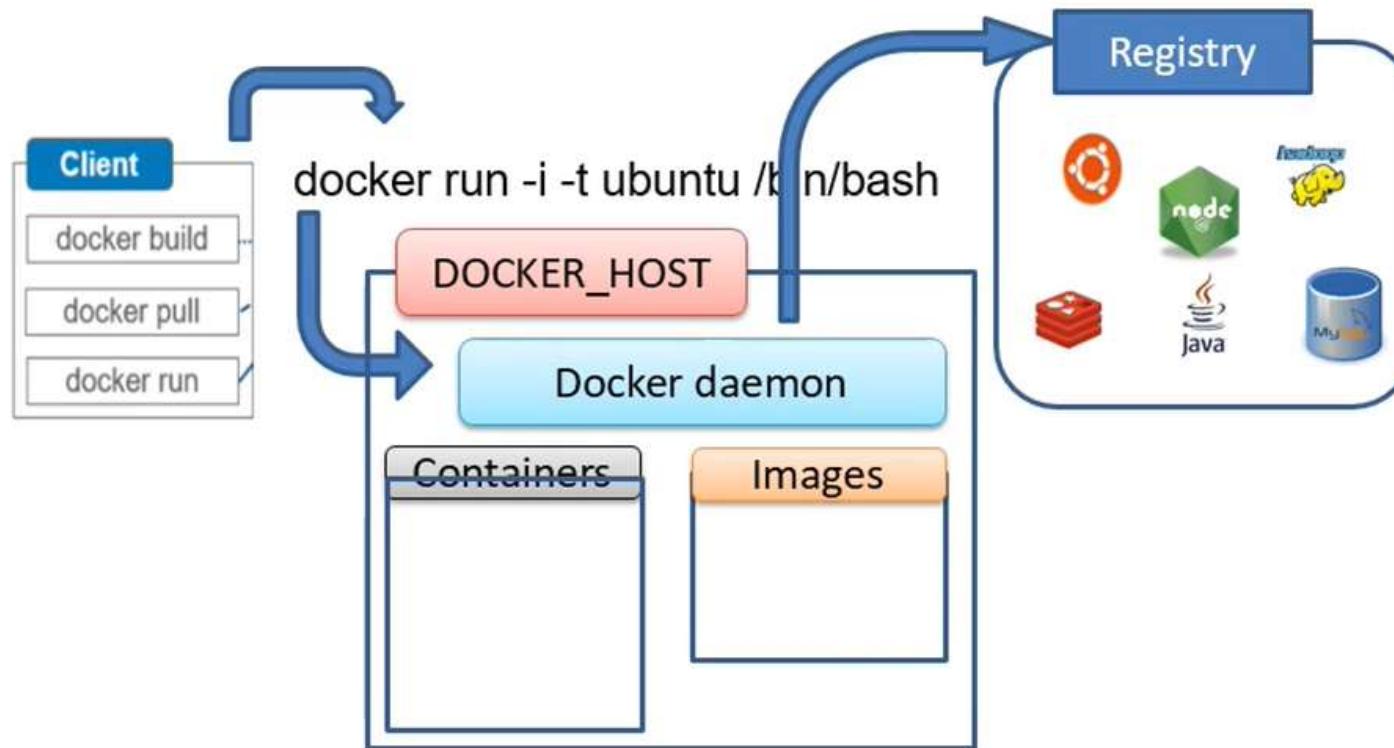
The Docker Architecture

Docker objects

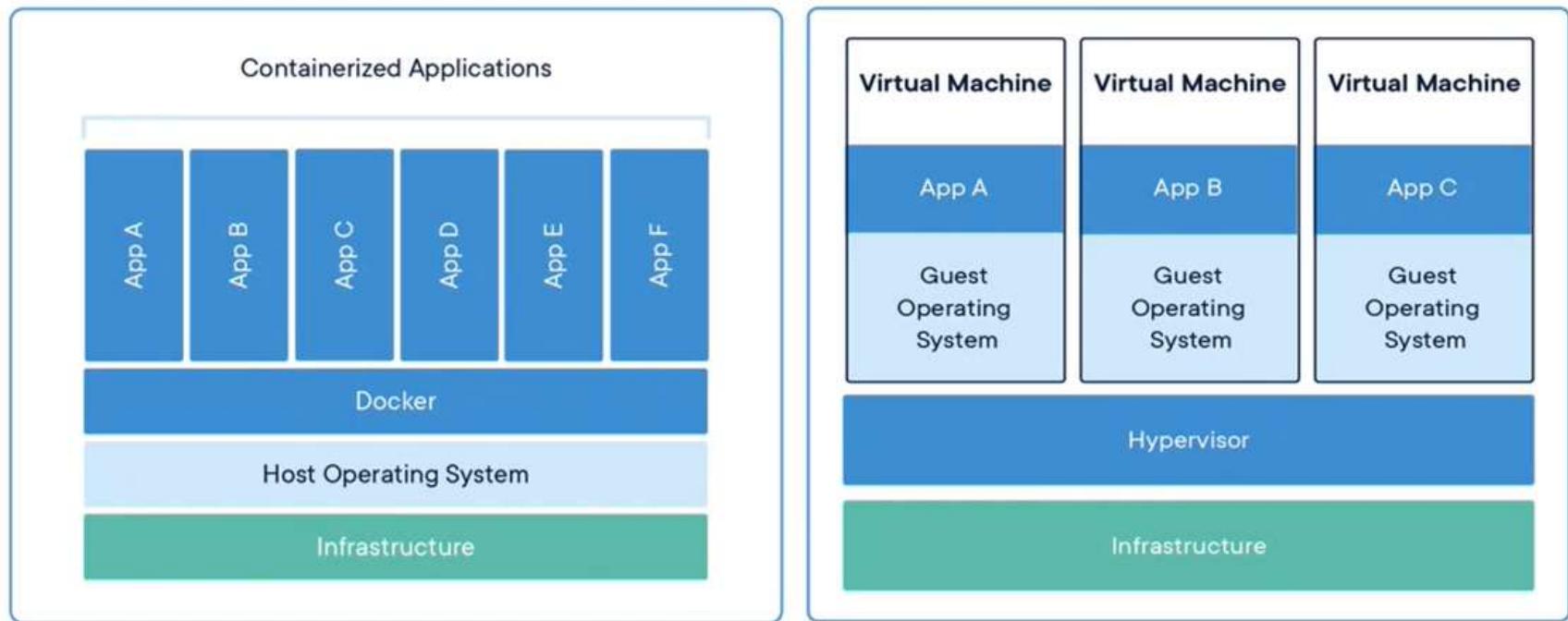
- IMAGES: An *image* is a read-only template with instructions for creating a Docker container.
- CONTAINERS: A container is a runnable instance of an image



Example for running a Docker container



Virtual Machine vs Container

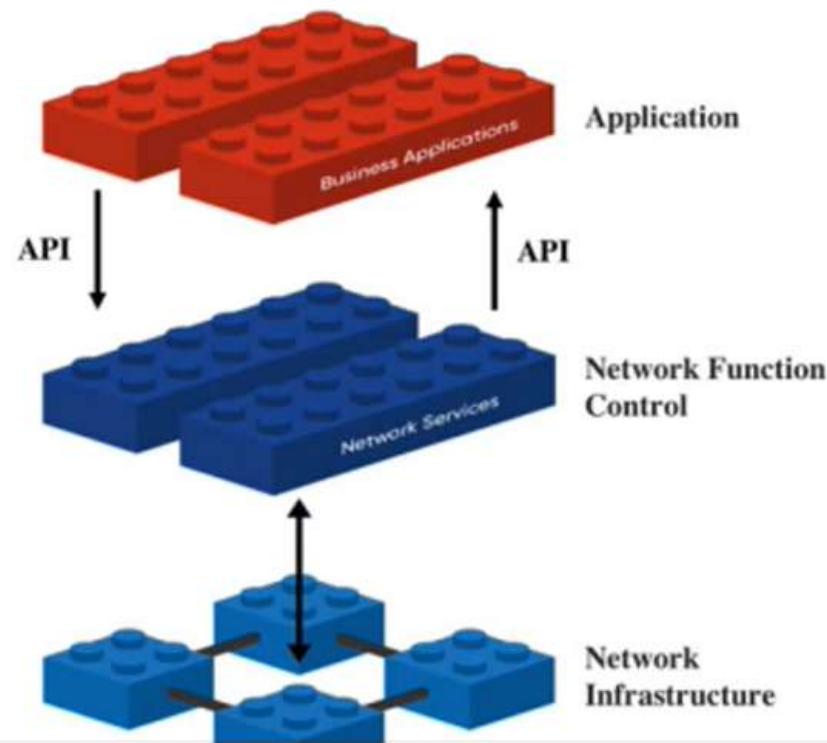


Virtual Machine vs Container

Factors	VM	Container
Start/stop time	~10s seconds	~10s milliseconds
Deployed services	Multiple services	Usually a single service
Guest OS	Independent	Depends on host OS
Failure pattern	Isolated between VMs	May affect other containers
Migration cost	More	Less but require the same host OS
Deployment density	1–10 instances	>10 instances
Security	Better	Good

Network Virtualization

High level architecture overview of SDN



SDN

Open source Projects:

Openflow - protocol that enables network controllers to determine the path of network packets across a network of switches

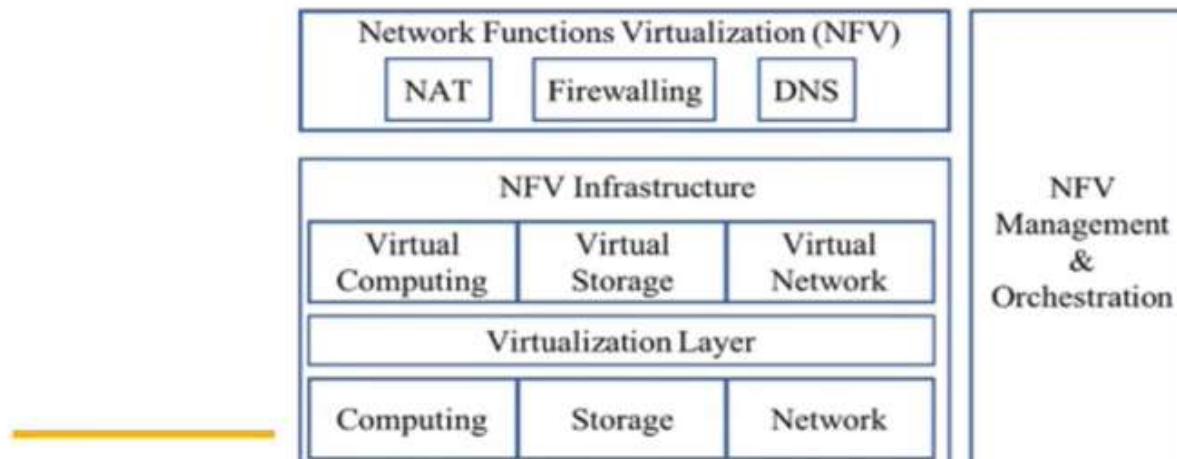
OpenvSwitch - multilayer virtual switch implemented in software

- ported to various hardware platforms

Network function virtualization

Network function virtualization (NFV)

- Network architecture concept similar to IT virtualization
- Decouples network services from proprietary hardware appliances.
- The software that provides these network services are known as virtual network functions (VNF)



SDN vs NFV

SDN forwards data packets from one network device to another

SDN's networking control functions for routing, policy definition, and applications run in a virtual machine

Therefore –

NFV provides essential networking functions

SDN controls and orchestrates them for specific uses

Resource Management

The resources in edge computing include:

- Computing resource – CPU
- Memory
- Storage
- Network
- Energy

Concerns of the resource management in edge computing:

- High degree of heterogeneity of hardware/software
- Dynamic availability and mobility of hardware/software
- Existing tools are relatively heavy

Docker:

It's a platform for building, running and shipping applications.

If your machine works on your development machine. It will work in the same way on the other machine.

Why it doesn't work in other machine??

- If one or more files are not deployed
- Software version mismatch
- Different configuration settings

That's the reason Docker comes into picture.

In docker we can package our application with everything it needs. It can be run in anywhere in any machine.



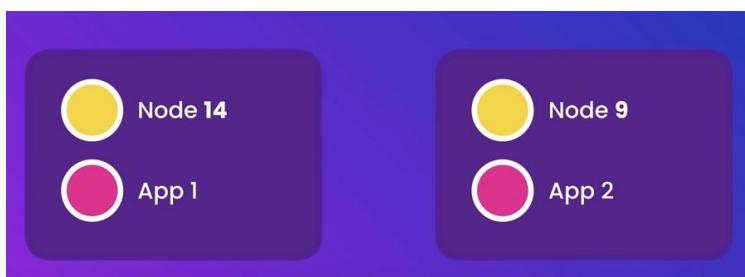
We can take this package and run on any machine.

If it works on your development machine, it is definitely going to work on your test machines.

For a new team member, they have to install the dependencies.

`docker-compose up`

isolated environment



If you don't need any docker you can remove still other docker will not be impacted.

```
docker-compose down -rmi all
```

Container

An isolated environment for running an application.

Virtual machine

An abstraction of a machine (physical hardware)



Hypervisors



Virtual machines



Both the VMs are running in isolated environments.

Problems in VMs

- Each VM needs a full copy of a full-blown OS
- Slow to start (Entire OS is loaded)
- Resource intensive (need all the required h/w and s/w)

Containers

Allows running multiple applications in isolation.

Are light weight.

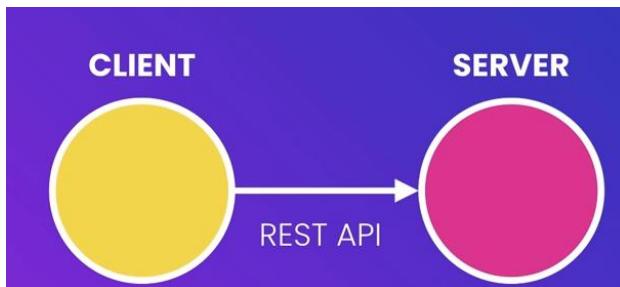
Use the OS of the host.

Start quickly.

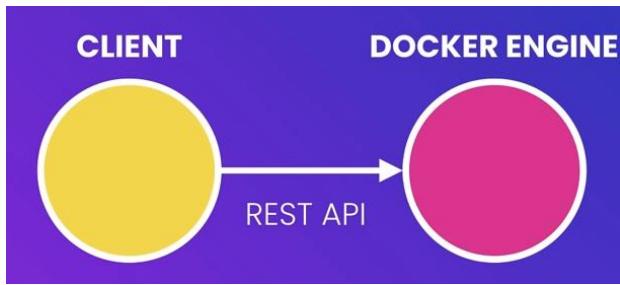
Need less hardware resources.

Docker Architecture

It's a client server architecture. A Client talks to the server over REST API.



A server is also called. Docker engine.



A docker is a process.



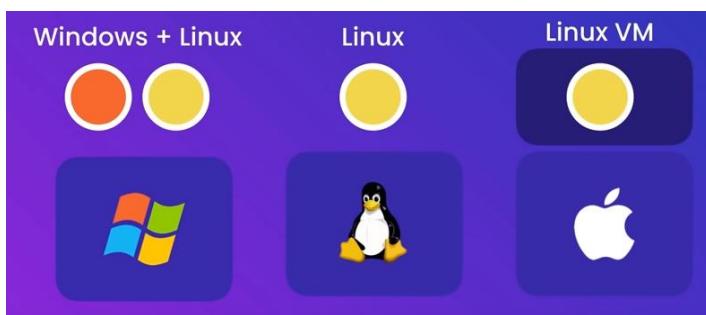
All the process uses the kernel which is the Windows OS here.

The kernel is the core of the OS, like the engine in a car.

A Kernel manages applications and hardware resources.

Every OS has its kernel or engine.

On a windows machine we can run both windows and Linux container.



Installing Docker

```
ubuntu@ip-172-31-14-85:~$ docker --version
Command 'docker' not found, but can be installed with:
sudo snap install docker      # version 20.10.24, or
sudo apt install podman-docker # version 3.4.4+ds1-lubuntul.22.04.2
sudo apt install docker.io     # version 24.0.5-0ubuntul~22.04.1
See 'snap info docker' for additional versions.
ubuntu@ip-172-31-14-85:~$
```

To install the docker

```
ubuntu@ip-172-31-14-85:~$  
ubuntu@ip-172-31-14-85:~$  
ubuntu@ip-172-31-14-85:~$ docker run ubuntu  
Command 'docker' not found, but can be installed with:  
sudo snap install docker      # version 20.10.24, or  
sudo apt install podman-docker # version 3.4.4+ds1-1ubuntu1.22.04.2  
sudo apt install docker.io     # version 24.0.5-0ubuntu1~22.04.1  
See 'snap info docker' for additional versions.  
ubuntu@ip-172-31-14-85:~$  
ubuntu@ip-172-31-14-85:~$  
ubuntu@ip-172-31-14-85:~$ sudo snap install docker  
docker 20.10.24 from Canonical✓ installed  
ubuntu@ip-172-31-14-85:~$
```



Check the docker version

```
ubuntu@ip-172-31-14-85:~$  
ubuntu@ip-172-31-14-85:~$ docker --version  
Docker version 20.10.24, build 297e128  
ubuntu@ip-172-31-14-85:~$  
ubuntu@ip-172-31-14-85:~$  
ubuntu@ip-172-31-14-85:~$
```

sudo apt-get update

```
ubuntu@ip-172-31-14-85:~$  
ubuntu@ip-172-31-14-85:~$ sudo apt-get update  
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy InRelease  
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates InRelease [119 kB]  
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-backports InRelease [109 kB]  
Get:4 http://security.ubuntu.com/ubuntu jammy-security InRelease [110 kB]  
Get:5 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 Packages [974 kB]  
Get:6 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/universe amd64 Packages [981 kB]  
Get:7 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/universe Translation-en [214 kB]  
Fetched 2506 kB in 1s (2062 kB/s)  
Reading package lists... Done  
ubuntu@ip-172-31-14-85:~$  
ubuntu@ip-172-31-14-85:~$  
ubuntu@ip-172-31-14-85:~$
```

sudo apt-get install ca-certificates curl gnupg

```
ubuntu@ip-172-31-14-85:~$ sudo apt-get install ca-certificates curl gnupg  
Reading package lists... Done  
Building dependency tree... Done  
Reading state information... Done  
ca-certificates is already the newest version (20230311ubuntu0.22.04.1).  
curl is already the newest version (7.81.0-1ubuntu1.13).  
gnupg is already the newest version (2.2.27-3ubuntu2.1).  
0 upgraded, 0 newly installed, 0 to remove and 124 not upgraded.  
ubuntu@ip-172-31-14-85:~$  
ubuntu@ip-172-31-14-85:~$
```



BITS Pilani
Pilani Campus

BITS Pilani presentation

Paramananda Barik
CS&IS Department





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

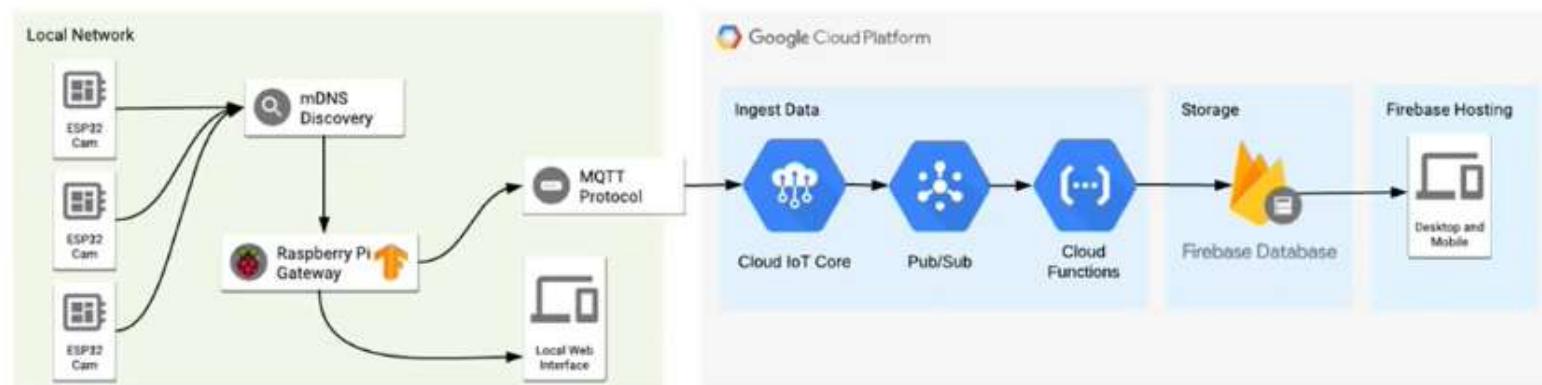
Data aggregation

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

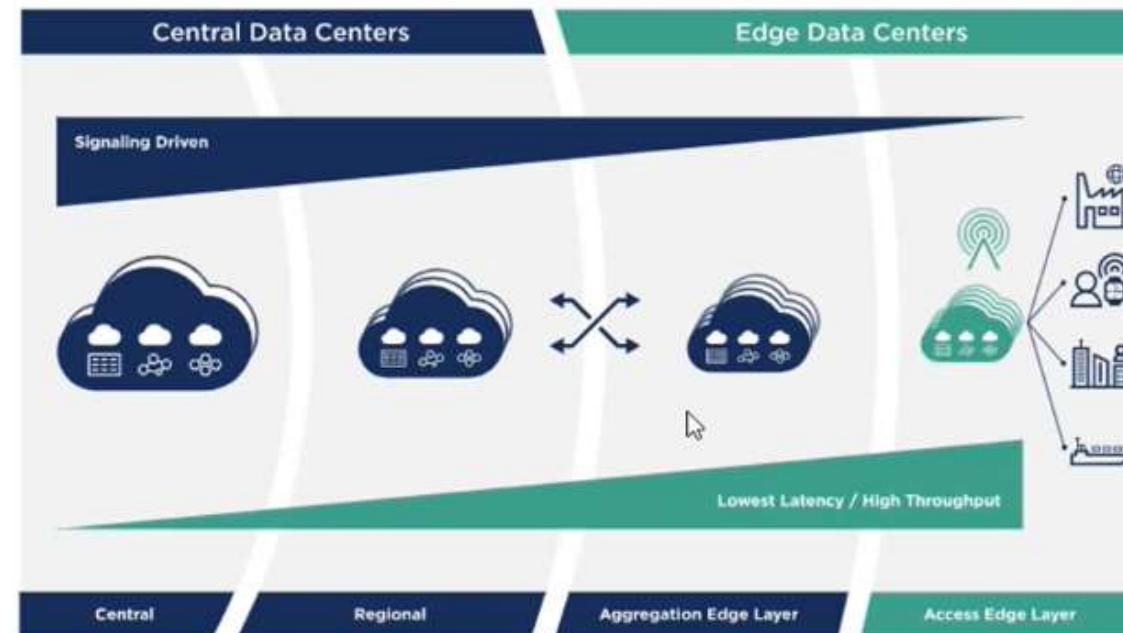
Energy

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

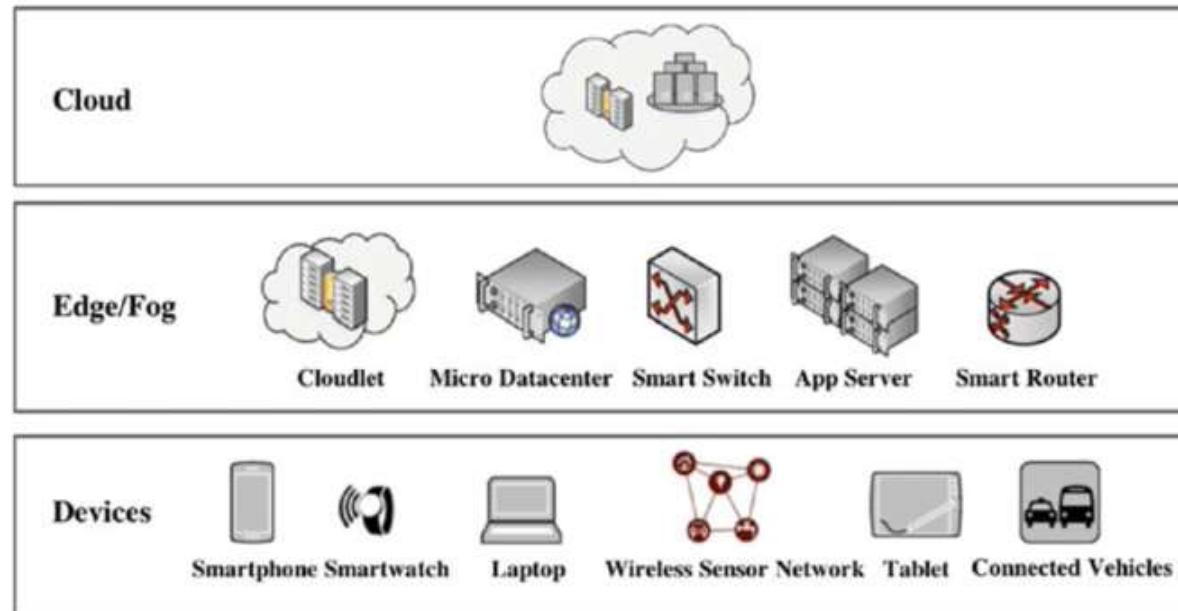
New layered cost model approach is needed

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 Heterogeneities

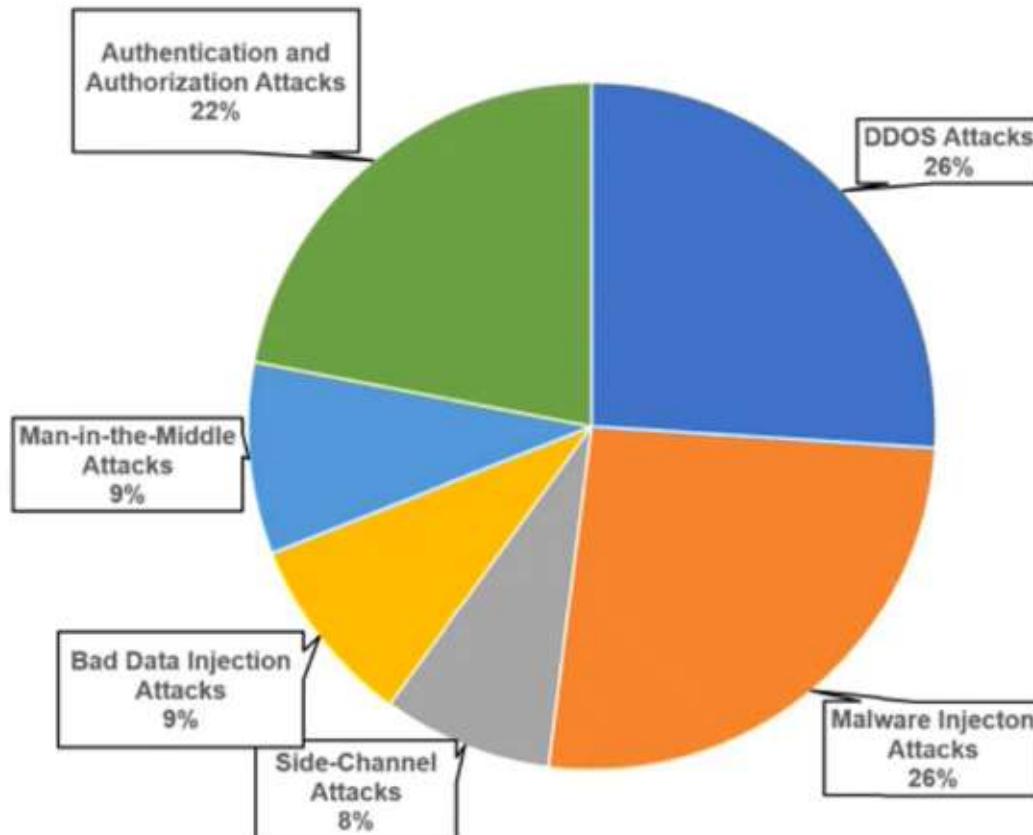
General purpose computers use:

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 Mechanism

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
is more 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

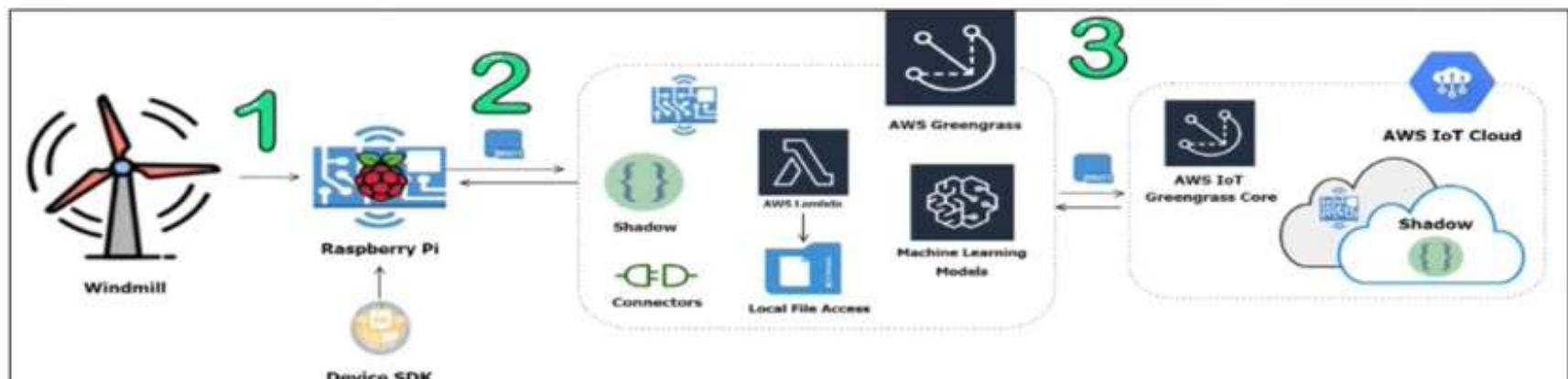
Code-level vulnerabilities

Features of Kubernetes

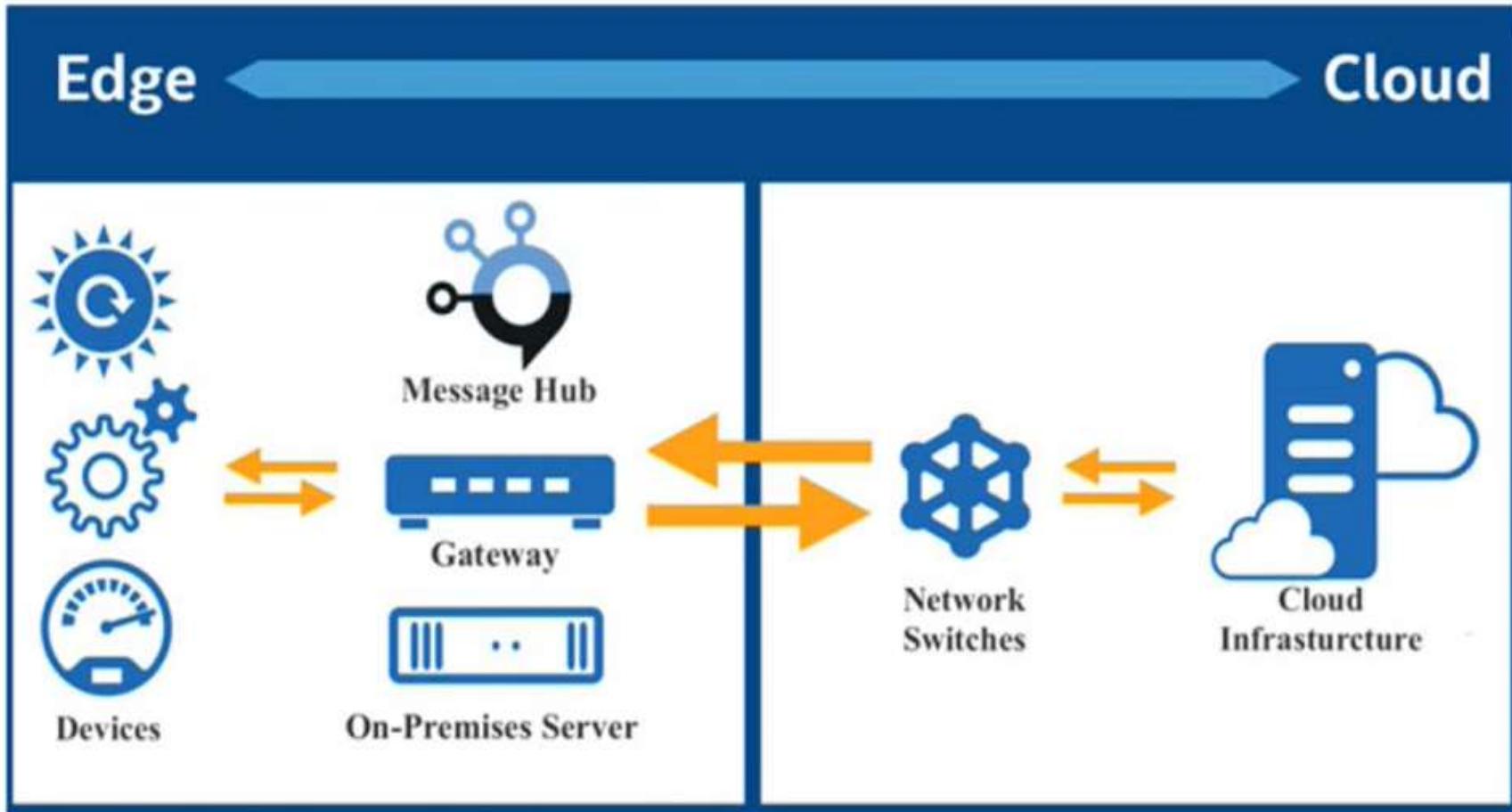
- Automated Scheduling
 - Self-Healing Capabilities
 - Automated rollouts & rollback
 - Horizontal Scaling & Load Balancing
 - Offers environment consistency for development, testing, and production
 - Infrastructure is loosely coupled to each component can act as a separate unit
 - Provides a higher 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)

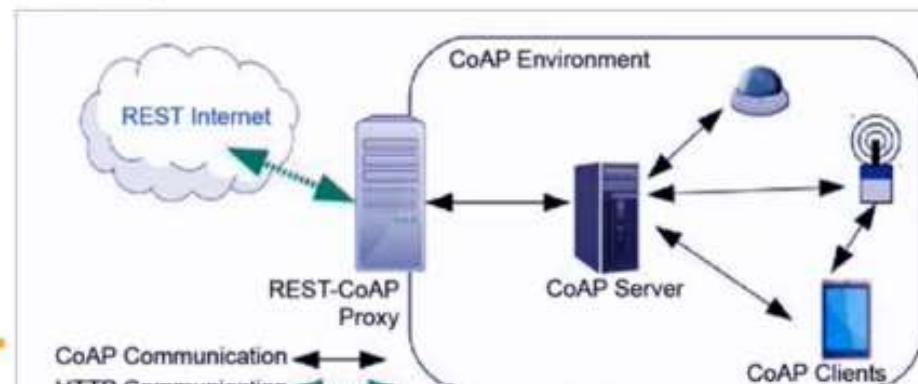
Advanced Message Queuing Protocol (AMQP)

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

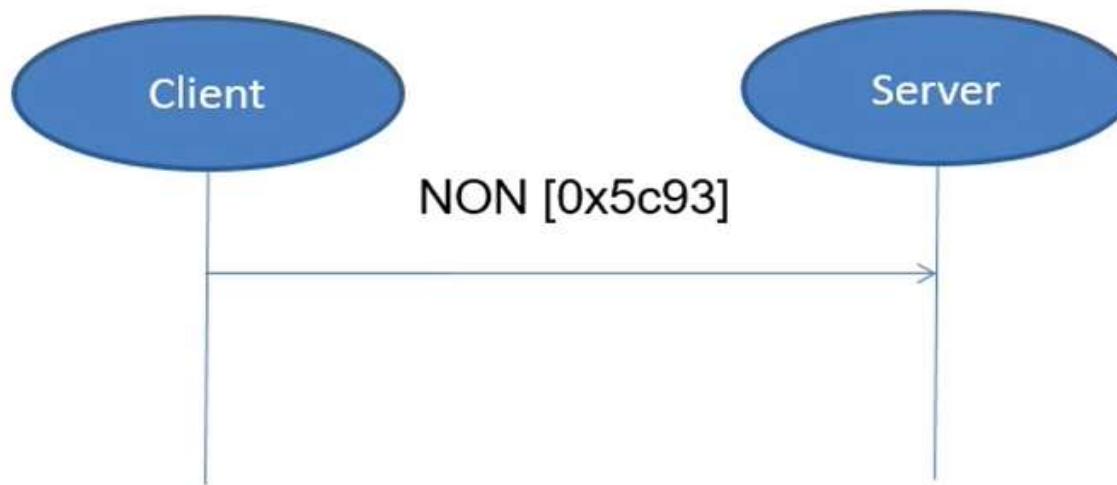
And meets the following requirements:

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

Message Layer Model

Unreliable message transport:

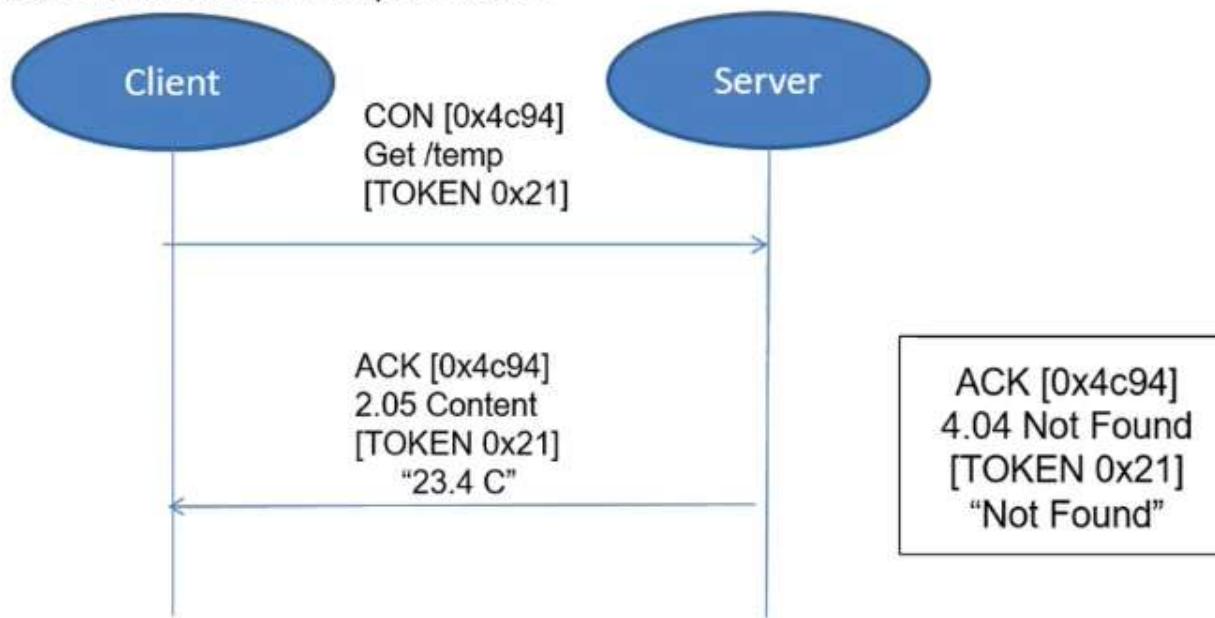
- Is used to transfer NON type message.
- It doesn't need to ACK, but has to contain message ID for supervising in case of retransmission.
- If recipient fail to process message, server replies RST



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.



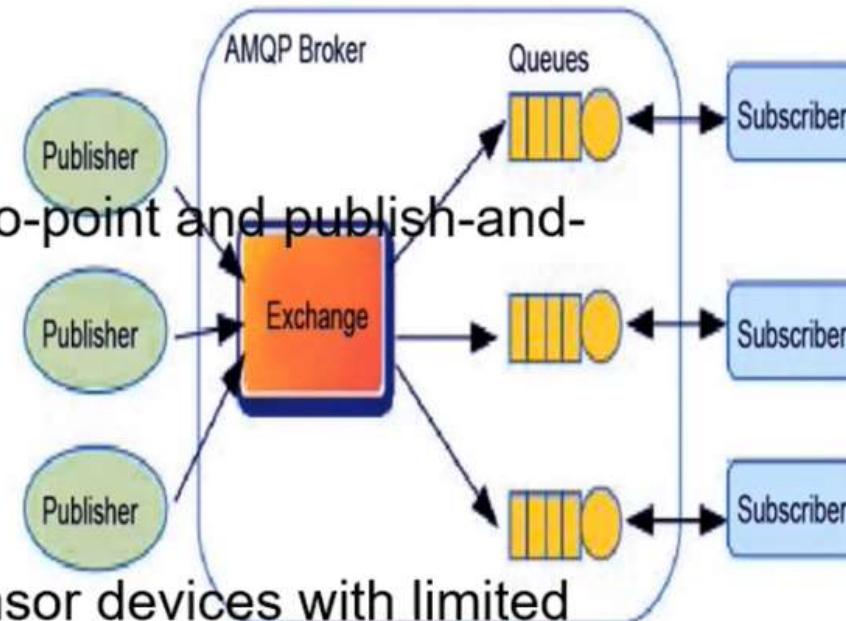
Advanced Message Queuing Protocol (AMQP)



AMQP is an open standard publish/subscribe type protocol

Developed mainly for financial services sector
features

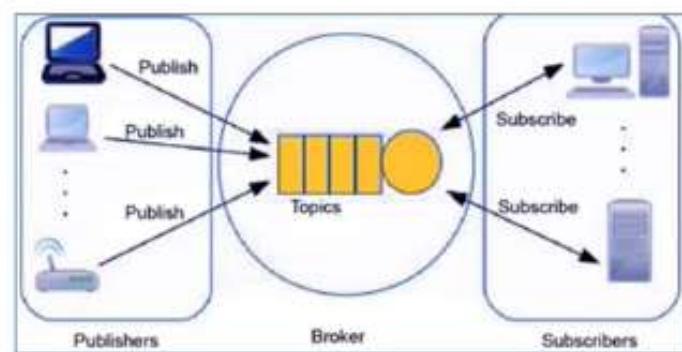
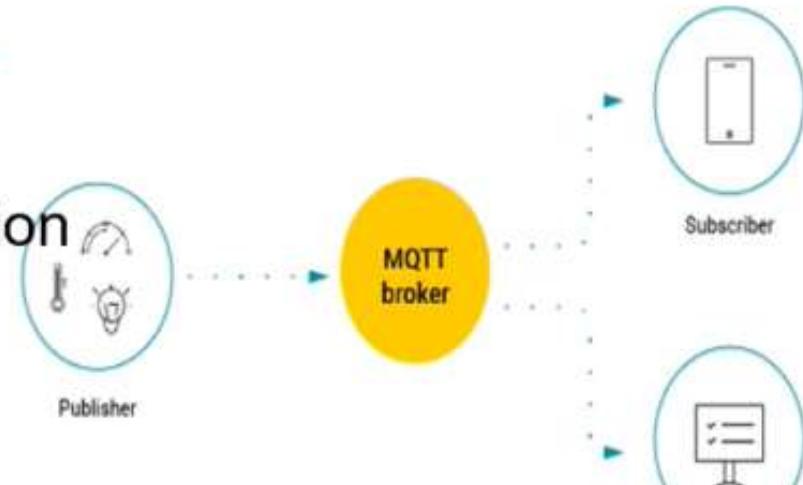
- message orientation
- queuing
- routing (including point-to-point and publish-and-subscribe)
- reliability
- secure



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

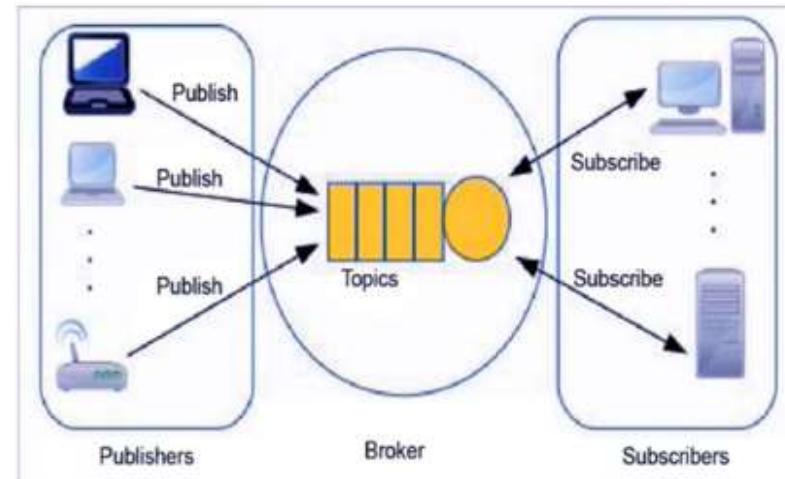
Broker - ensure security by cross-checking the authorization of publishers and subscribers

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.

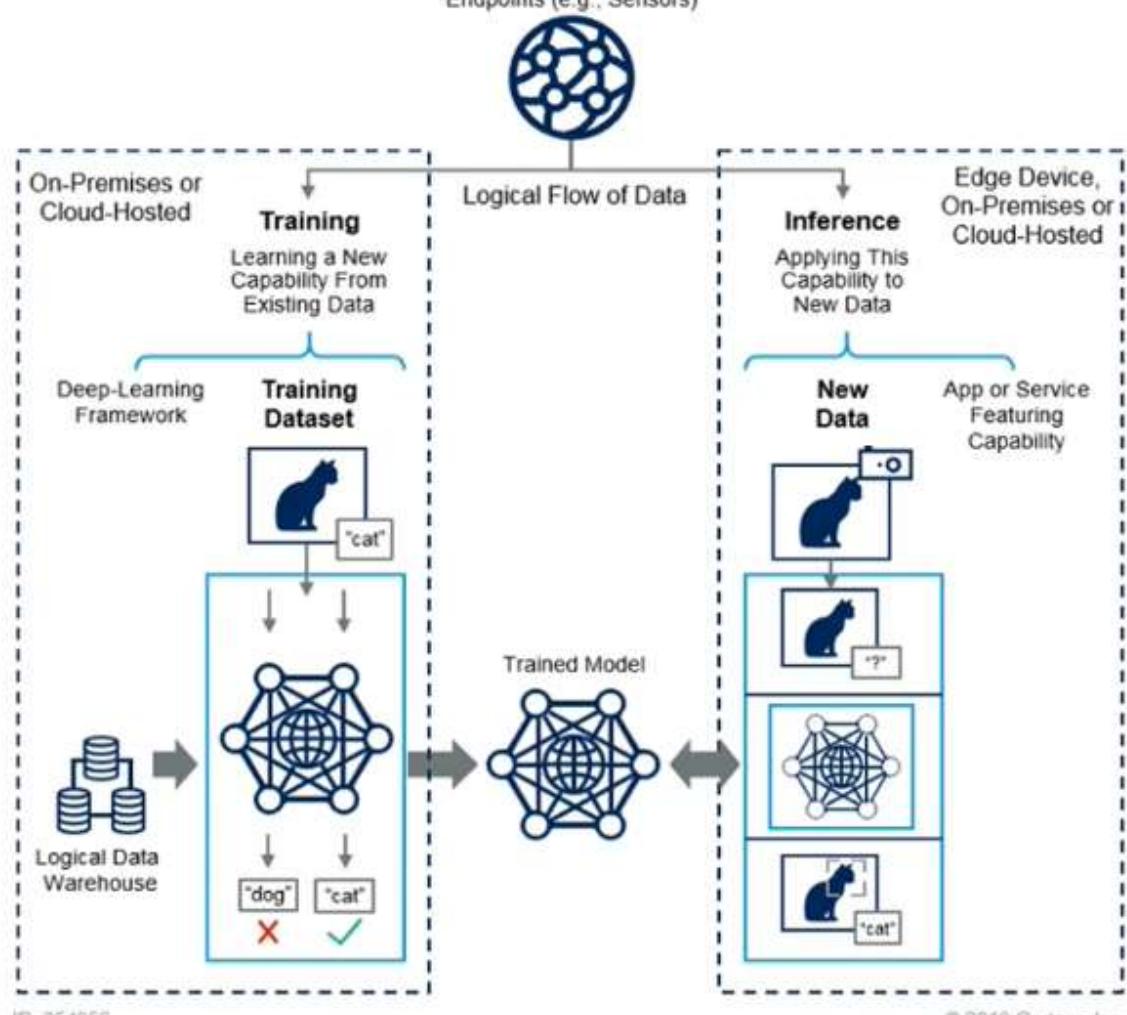
runs specially optimized models on many different
hardware types

mobile phones

embedded Linux devices

microcontrollers.

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



© 2019 Gartner, Inc.

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. MicroK8s

	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.

A Conceptual Framework for Security and Privacy in Edge Computing

A conceptual framework for security and privacy in edge computing is essential to address the unique challenges and requirements that arise in this emerging field.

Edge computing involves processing data closer to the source of data generation, such as IoT devices, rather than relying solely on centralized data centers or cloud computing.

This proximity to data sources introduces new security and privacy concerns that need to be carefully managed.

Here's a high-level conceptual framework to guide the design and implementation of security and privacy in edge computing:

1. Threat Landscape Analysis:

- Begin by assessing the specific threats and vulnerabilities that edge computing environments face. These can include physical threats, network attacks, device-level vulnerabilities, and more.

2. Access Control and Authentication:

- Implement robust access control mechanisms to ensure that only authorized entities can interact with edge devices and systems.
- Employ strong authentication methods, including multi-factor authentication, to verify the identity of users, devices, and applications.

3. Data Encryption:

- Encrypt data both in transit and at rest to protect it from interception or unauthorized access.

- Consider using end-to-end encryption to secure data from the edge device to the cloud or other endpoints.

4. Security at the Edge Devices:

- Hardening edge devices by regularly updating firmware, ensuring secure boot processes, and using trusted platform modules (TPMs) for hardware-level security.
- Employ containerization and isolation techniques to compartmentalize applications and data on edge devices.

5. Network Security:

- Implement robust network security measures, such as firewalls, intrusion detection and prevention systems, and virtual private networks (VPNs), to secure communication between edge devices and the central system.

6. Identity and Access Management (IAM):

- Define and manage the identities and permissions of users, devices, and applications to ensure that they have the appropriate level of access to edge resources.

7. Data Governance and Privacy Protection:

- Establish data governance policies that outline how data is collected, processed, stored, and shared at the edge.
- Comply with privacy regulations (e.g., GDPR, CCPA) by anonymizing or pseudonymizing sensitive data and obtaining user consent where necessary.

8. Security Monitoring and Incident Response:

- Implement continuous monitoring of edge devices and networks to detect anomalies and potential security breaches.
- Develop an incident response plan to react promptly to security incidents and minimize their impact.

9. Edge-to-Cloud Security Integration:

- Ensure a cohesive security strategy that spans from the edge to the cloud, addressing security and privacy concerns at every stage of data processing and transmission.

10. Compliance and Audit:

- Regularly audit and assess the security and privacy measures in place to ensure compliance with relevant standards and regulations.

11. Education and Training:

- Provide ongoing education and training for personnel involved in edge computing to raise awareness about security and privacy best practices.

12. **Vendor and Supply Chain Security:**

- Evaluate the security practices of third-party vendors and supply chain partners, as their products and services can impact the overall security of edge computing systems.

13. **Resilience and Redundancy:**

- Plan for redundancy and failover mechanisms to ensure continued operation in case of device failures or security incidents.

14. **User Awareness and Consent:**

- Educate end-users and IoT device owners about the data collection, processing, and sharing practices, and obtain informed consent where applicable.

15. **Ethical Considerations:**

- Consider the ethical implications of data collection and processing at the edge, addressing issues like bias and fairness in AI algorithms.

Similarities and Differences Between Edge Paradigms

Edge computing encompasses several different paradigms, each designed to address specific use cases and requirements. Here, I'll outline the similarities and differences between three prominent edge paradigms: Fog Computing, Edge Cloud, and Multi-Access Edge Computing (MEC).

Similarities:

1. **Proximity to Data Sources:** All three paradigms involve processing data closer to the source of data generation, which reduces latency and conserves network bandwidth.
2. **Low Latency:** They aim to achieve low latency for applications, making real-time processing feasible. This is crucial for applications like IoT, augmented reality, and autonomous vehicles.

3. Distributed Architecture: These paradigms adopt a distributed architecture, with resources distributed across various edge nodes or devices.
4. Scalability: They are designed to scale horizontally as the number of edge devices or the demand for edge computing resources increases.
5. Resource Optimization: They focus on efficient resource utilization, ensuring that computational resources at the edge are not wasted.

Differences:

1. Fog Computing:

- Processing Location: Fog computing typically places computational resources in proximity to the data source, such as at the network's edge (e.g., routers, switches).
- Scope: Fog computing is designed to provide computing services within a local network or region, making it suitable for use cases like smart cities or industrial automation.
- Resource Heterogeneity: Fog nodes can vary widely in terms of computational power and resources.
- Coordination: Coordination and management of fog nodes are typically performed by a fog controller.

2. Edge Cloud:

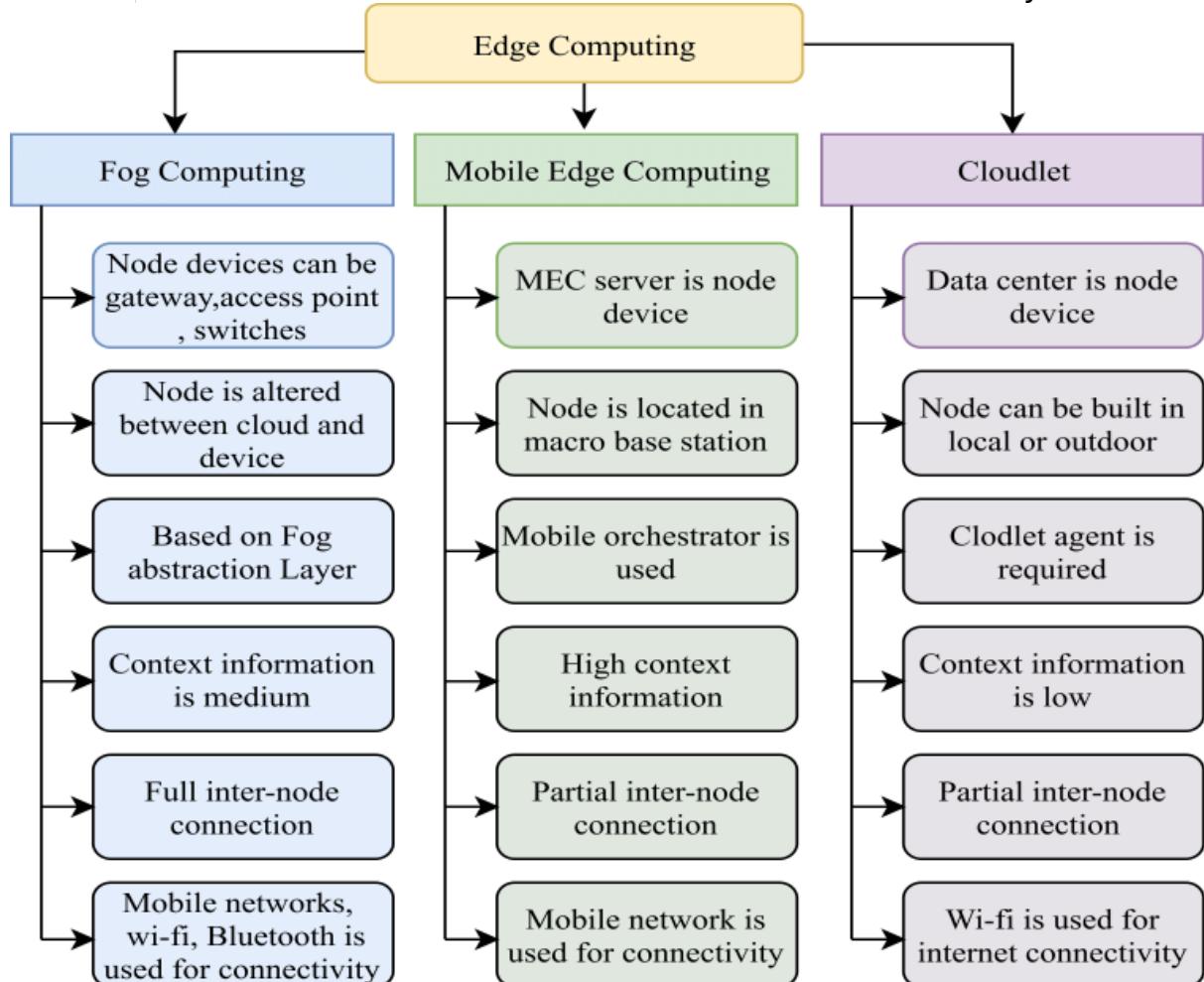
- Processing Location: Edge cloud solutions are characterized by deploying cloud-like resources at the edge, often in a centralized manner. This approach can include small-scale data centers or cloud infrastructure closer to the edge.
- Scope: Edge cloud extends the capabilities of traditional cloud computing to the edge, allowing for more powerful and flexible computing at the edge.
- Resource Heterogeneity: Edge cloud deployments tend to be more uniform and standardized than fog computing, with a focus on cloud infrastructure.

3. Multi-Access Edge Computing (MEC):

- Processing Location: MEC is a standards-based approach that places computing resources at the edge of the cellular network, typically in close proximity to base stations (eNodeBs or gNodeBs).
- Scope: MEC is primarily designed to enhance the capabilities of mobile networks (e.g., 4G and 5G) by enabling low-latency,

high-throughput services, such as augmented reality and network slicing for different applications.

- Resource Heterogeneity: MEC nodes are often homogeneous and standardized to facilitate uniform service delivery.



Cloud/Core Network



Overview of Security, Privacy, and Threats in Edge Computing

edge computing introduces unique challenges and opportunities related to security, privacy, and threats due to its distributed architecture, where data processing occurs closer to the data source. Here's an overview of security, privacy, and potential threats in edge computing:

Security in Edge Computing:

1. **Physical Security:** Edge devices are often located in less secure environments compared to data centers. Ensuring physical security at the edge is critical to prevent unauthorized access and tampering.
2. **Access Control:** Effective access control mechanisms are essential to restrict access to edge devices and systems. Role-based access control and authentication mechanisms are crucial.
3. **Data Encryption:** Data should be encrypted both in transit and at rest to protect it from eavesdropping and unauthorized access. Strong encryption standards are essential.
4. **Device Hardening:** Edge devices should be hardened to reduce vulnerabilities. Regular updates and patches are necessary to mitigate security risks.
5. **Network Security:** Robust network security measures, such as firewalls, intrusion detection and prevention systems, and virtual private networks (VPNs), are required to secure communication between edge devices and central systems.
6. **Identity and Access Management (IAM):** Effective IAM is crucial for managing identities, permissions, and roles of users, devices, and applications, ensuring appropriate access to edge resources.
7. **Security Monitoring:** Continuous monitoring of edge devices and networks for anomalies and potential security breaches is essential for early threat detection.
8. **Incident Response:** An incident response plan should be in place to react promptly to security incidents and minimize their impact.
9. **Compliance:** Compliance with relevant regulations and standards is a fundamental aspect of security in edge computing. This may include GDPR, HIPAA, or industry-specific standards.

10. **Vendor and Supply Chain Security:** Evaluating the security practices of third-party vendors and supply chain partners is vital, as they can impact the overall security of edge computing systems.

Privacy in Edge Computing:

1. **Data Minimization:** Collect and process only the data necessary for the intended purpose to reduce privacy risks. Minimizing the data footprint helps protect user privacy.
2. **Anonymization and Pseudonymization:** Use techniques like anonymization and pseudonymization to protect sensitive information while still enabling useful analysis and processing.
3. **Consent:** In cases where personal data is involved, obtain informed consent from users before collecting and processing their data.
4. **Data Governance:** Establish clear data governance policies that outline how data is collected, processed, stored, and shared at the edge. These policies should include privacy considerations.
5. **User Awareness:** Educate end-users and IoT device owners about data collection and processing practices and inform them about how their data is used.

Threats in Edge Computing:

1. **Data Breaches:** Unauthorized access to edge devices or networks can lead to data breaches. Data should be protected with encryption and access controls.
2. **Malware and Ransomware:** Edge devices may be vulnerable to malware and ransomware attacks, especially if they are not regularly updated and secured.
3. **Physical Attacks:** Edge devices are often deployed in physically accessible locations, making them susceptible to physical attacks, tampering, and theft.
4. **Denial of Service (DoS) Attacks:** Attackers may launch DoS attacks against edge devices or networks, causing service disruptions.

5. **Interception of Data:** Data in transit between edge devices and central systems is vulnerable to interception if not properly encrypted.
6. **IoT Device Vulnerabilities:** Many edge devices are IoT devices, which can have security vulnerabilities if not properly configured and updated.
7. **Insider Threats:** Insider threats can be a concern, especially in environments where multiple users and administrators have access to edge resources.
8. **Regulatory Non-Compliance:** Failure to comply with privacy regulations can lead to legal issues and penalties.

Framework for Security and Privacy in Edge Computing

Refer the research paper...

Building a comprehensive framework for security and privacy in edge computing is essential to protect data and systems in this distributed and decentralized environment. Below is a structured framework that combines both security and privacy considerations in the context of edge computing:

1. Threat Landscape Analysis:

- **Identification:** Analyze and identify potential threats and vulnerabilities specific to the edge computing environment.
- **Risk Assessment:** Evaluate the impact and likelihood of these threats to prioritize security and privacy measures.

2. Access Control and Authentication:

- **Role-Based Access Control (RBAC):** Implement RBAC to ensure that only authorized entities have access to edge devices and resources.

- **Strong Authentication:** Employ strong authentication mechanisms, such as multi-factor authentication, for user, device, and application identity verification.

3. Data Encryption:

- **Data in Transit:** Encrypt data in transit using secure protocols to protect it from interception.
- **Data at Rest:** Encrypt data stored on edge devices to prevent unauthorized access.

4. Secure Device Management:

- **Firmware Updates:** Regularly update device firmware and software to patch vulnerabilities and enhance security.
- **Trusted Platform Modules (TPMs):** Use TPMs for hardware-level security to ensure device integrity.

5. Network Security:

- **Firewalls:** Deploy firewalls to filter incoming and outgoing traffic and protect edge devices from network threats.
- **Intrusion Detection and Prevention Systems (IDPS):** Use IDPS to identify and mitigate network-based threats.
- **Virtual Private Networks (VPNs):** Employ VPNs to secure communications between edge devices and central systems.

6. Identity and Access Management (IAM):

- **User and Device Identity Management:** Define and manage user and device identities to ensure appropriate access.
- **Permission Management:** Define and enforce permissions and roles for users and applications.

7. Data Governance and Privacy Protection:

- **Data Classification:** Categorize data based on its sensitivity and privacy requirements.

- **Data Anonymization and Pseudonymization:** Use techniques like anonymization and pseudonymization to protect privacy while enabling analysis.
- **Privacy by Design:** Incorporate privacy principles into the design of edge applications and systems.

8. Security Monitoring and Incident Response:

- **Continuous Monitoring:** Implement continuous monitoring to detect anomalies and potential security breaches.
- **Incident Response Plan:** Develop and test an incident response plan to react promptly to security incidents and minimize their impact.

9. Edge-to-Cloud Security Integration:

- Ensure a cohesive security strategy that spans from the edge to the cloud, addressing security and privacy concerns at all stages of data processing and transmission.

10. Compliance and Audit:

- Regularly audit and assess security and privacy measures to ensure compliance with relevant regulations and standards.

11. User Awareness and Consent:

- Educate end-users and IoT device owners about data collection, processing, and sharing practices.
- Obtain informed consent where applicable, especially for the processing of personal data.

12. Ethical Considerations:

- Address ethical implications of data collection and processing, including issues like bias and fairness in AI algorithms.

13. Resilience and Redundancy:

- Plan for redundancy and failover mechanisms to ensure continued operation in case of device failures or security incidents.

14. Vendor and Supply Chain Security:

- Evaluate the security practices of third-party vendors and supply chain partners, as their products and services can impact the overall security of edge computing systems.



BITS Pilani
Pilani Campus

BITS Pilani presentation

Paramananda Barik
CS&IS Department





SEZG586/SSZG586, Security and Privacy in Edge Computing

Lecture No.13

A Conceptual Framework for Security and Privacy in Edge Computing



Edge computing devices - specific attributes such as:

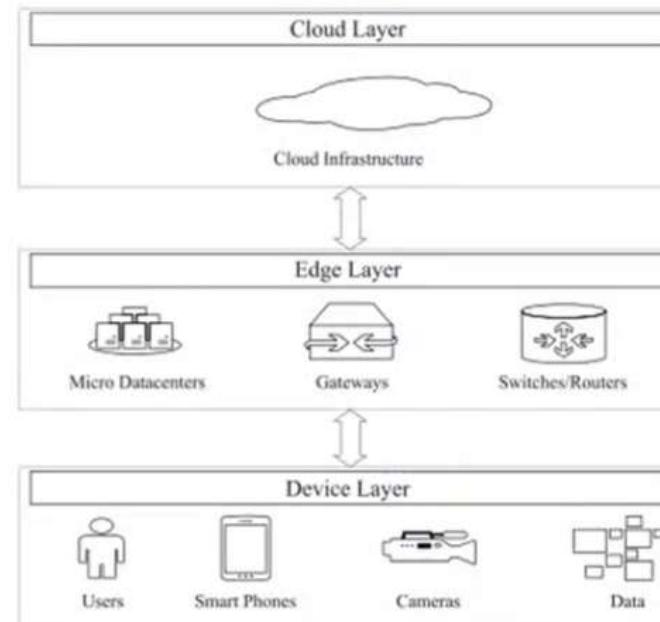
heterogeneity

low latency

location awareness

spatial distribution

mobility



Difference between edge computing and cloud computing

Features	Cloud computing	Edge computing
Computational capacity	High	Medium to low
Size and operating mode	Centralized large servers	Distributed smaller servers
Applications	Delay tolerant, computationally intensive	Low latency, real-time operation, high QoS
Fronthaul/backhaul communication overhead	High	Low
Mobility support	Low	High
Management	Service provider	Local business
Deployment	Requires complicated deployment planning	Ad hoc deployment/minimal or no planning
User device	Computers, limited mobile devices	Mobile smart wearable devices
Network access type	WAN	LAN/WLAN

Overview of Security, Privacy, and Threat in Edge Computing



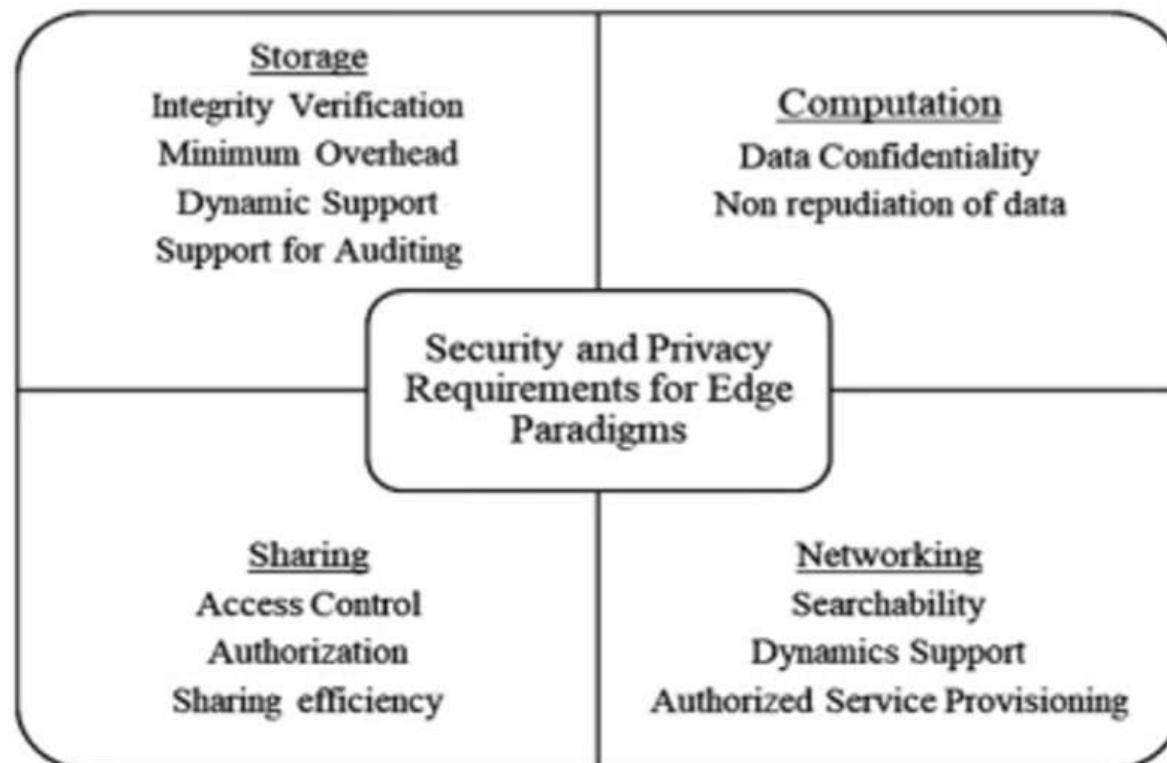
The principal services offered by edge computing

storage

computation

data sharing

networking



Network Security

Edge devices

distributed across the network

cost of maintenance is high

therefore

Configurations by the network administrator

Network management information

must be secured

isolated from the normal data flow

Data Storage Security

If the data storage in edge platform is outsourced
tough to ensure data integrity
possibility for data loss and data modifications

Data abuse by the attackers is easy

Hence,

data storage auditing is needed

Encryption techniques

ensure integrity, verifiability, and confidentiality of the data

Data Computation Security

Data computation performed by the edge servers and devices should be
secured
verifiable

Computations can be made secure by using data encryption techniques

Prevents data visibility to any hackers/attackers

Microdata centers have the provisions to off-load mechanism to verify the computation results establish trust between the two entities

End Devices Security

End devices can be easily tampered
less powerful
limited access

Potential harm that can be done by an attacker:
create rogue node to retrieve essential network
management data
change normal behavior by corrupting the device data
increase the frequency of data access by sending fake
information
propagate false data across the network

Access Control

Edge platform is distributed and decentralized
good access control policy acts as a defensive shield

Access control helps in the following:
realizing the interoperability
collaboration among microdata centers that are provided
by different service providers and are separated
across different geo-locations

Intrusion Detection

Intrusion detection techniques help in the following:

- identifying malicious data entries

- detect device anomalies

Used to investigate and analyze the behavior of devices in the network

Provide methods to perform packet inspection:

- early detection of denial-of-service attacks, integrity attacks, and data flooding

Privacy in Edge Computing

Privacy in edge computing:

user privacy

data privacy

location privacy

User privacy deals with protecting the privacy of user's credentials and their frequency of accessing the data.

Data privacy helps in avoiding unauthorized access of the network by hackers and preserve the integrity of the network.

Location privacy is required since the edge data centers may be spatially distributed

Ensuring location privacy without affecting the computational overhead is of primary importance

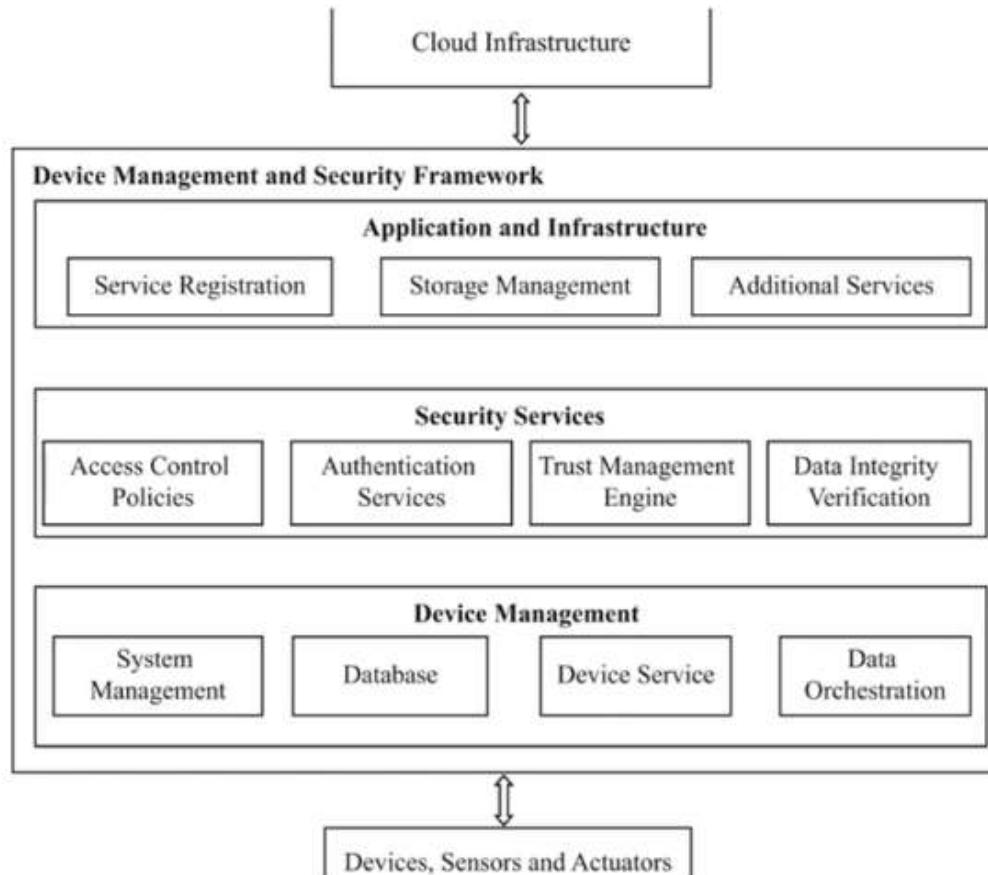
Nature of Threats in Edge Computing

Components	Possible threats
Data	Data replication
	Data manipulation
	Data deletion
	Illegal data access
	Eavesdropping

Network	Denial of service Man-in-the-middle Physical damage Service manipulation Rogue server Privacy leakage Jamming Congestion Black hole
---------	---

Communication	Data loss Data breach Sniffing Message replay Impersonation
Services	Service manipulation Rogue infrastructure Privacy leakage Insecure APIs
Devices	Physical damage Misuse of resources Flooding Power failure

Framework for Security and Privacy in Edge Computing



Device Management

Primary functionalities of the device management layer:

- manage the end devices
- create database
- manage the services provided to the end device
- perform data orchestration operations

Device management layer takes care of:

- status of the end devices
- access control
- maintaining integrity of the data stored
- provide the necessary data from service providers

Security Services

Security services layer performs:

creating, updating, and maintaining the access control policies.

ensure authentication of devices, service providers, and the data that is communicated.

makes use of a trust management engine

governs the privacy of the overall system

generates rules to implement intrusion detection mechanism



Application Infrastructure

Responsible for

keeping track of the network-wide application interactions

governs the authentication policies for accessing the network devices

service provisioning details of neighboring data centers
decision of whether cloud support is required or not

Security and Privacy Challenges in Edge Computing



- The added security mechanism should not increase the computation overhead and hamper the storage space required for other system operations.
 - Edge computing mechanisms use cache management techniques which are prone to side channel attacks; hence, care should be taken to prevent leakage of private and sensitive data.
-

Security and Privacy Challenges in Edge Computing



- For applications that stream data continuously for monitoring and management purposes, due to increased number of devices, the volume data to be processed is typically large. To detect the anomaly, packet filtering mechanisms are needed which might require additional memory and processing power.
 - Due to the increased number of devices, implementing a common access control policy becomes a tedious task. Accounting for mobility of devices is also important.
-

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

Weak Computation Power

Computation power of an edge server is relatively weaker.

Similarly, compared to general-purpose computers, edge devices have more fragile defense systems;

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 Heterogeneities

General purpose computers use:

- Standard Os

- Communication protocols such as POSIX

Edge devices have different OSes and protocols without a standardized regulation

Coarse-Grained Access Control

Access control model in cloud:

four types of permissions:

No Read & Write

Read Only

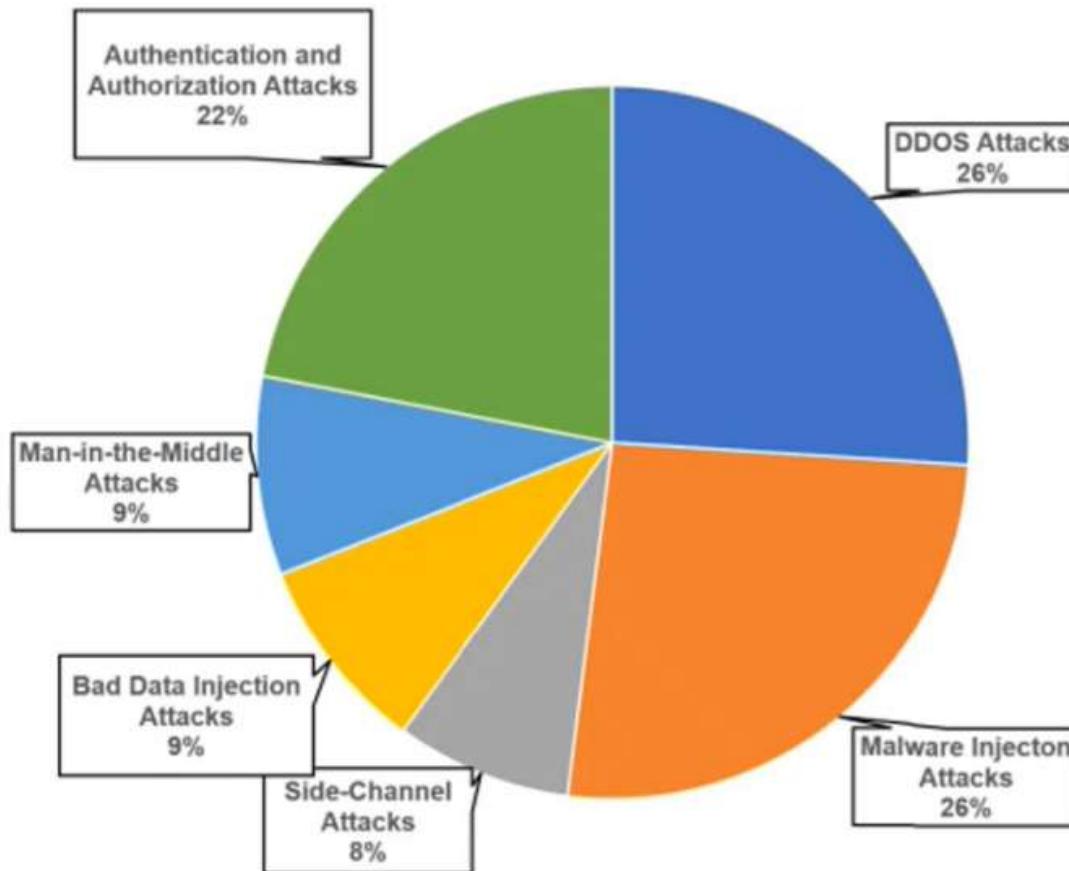
Write Only

Read & Write

Access control in edge should be able to answer:

such as “who can access which sensors by doing what at when and how”

Types of attacks targeting edge computing infrastructure



Security threats and attacks faced by edge computing



Threats mainly due to
the design flaws
misconfigurations
implementation bugs

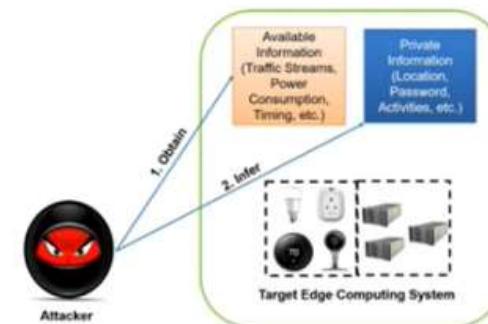
Defense mechanisms
Detection based
Prevention based

Types of attacks

DDoS Attacks



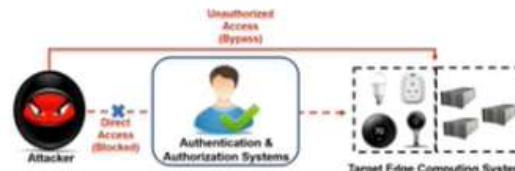
Side-Channel Attacks



Malware Injection Attacks



Authentication and Authorization Attacks



DDoS Attacks and Defense Mechanism



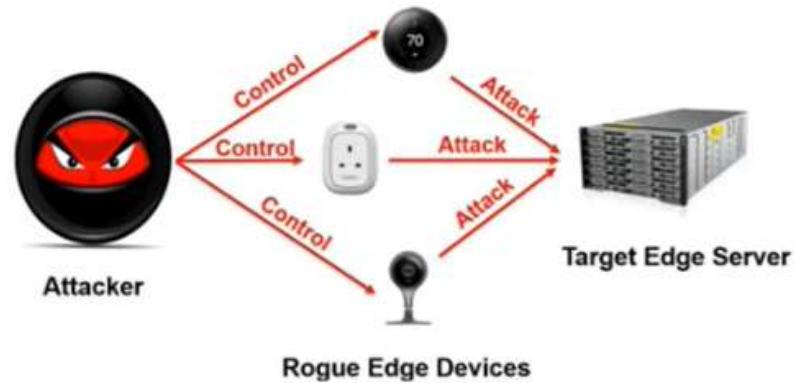
DDoS - type of cyberattack in which attackers aim to disrupt normal services provided by one or more servers based on distributed resources such as a cluster of compromised edge devices

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

DDoS attacks

DDoS attacks:

- Flooding-based attacks
- Zero-day DDoS attacks



Flooding-based attack techniques

- UDP flooding
- ICMP flooding
- SYN flooding
- Ping of death (PoD)
- HTTP flooding
- Slowloris

Flooding-based Attack Mechanism

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
is more 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

Defense Solution against flooding-based attacks



Two categories:

- per-packet-based detection
- statistics-based detection

Per-packet-based detection:

- detecting and filtering malicious packets
- integrating packet filtering mechanisms into congestion control frameworks

Proposed ways in per-packet-based detection

spot DDoS on a per-packet basis having the same identifier

changing the identifiers of the packets using tools such as hping3

negative selection algorithm: whether the IP address of a packet is legitimated based on the eigenvalue sets to resist this type of DDoS

server to maintain a list of legitimate IP addresses

Proposed ways in Statistics-based detection



Machine learning-based mechanisms to detect DDoS traffics

methods : J48, naïve Bayes, and Bayesian network classifiers to detect botnet DDoS

deep learning model using an autoencoder to detect encrypted DDoS traffics

neural networks to identify DDoS attacks in software-defined networks

Benefit: little human effort

But they may perform differently on different types of DDoS attacks

Defense solutions against zero-day attack

Mechanisms to spot possible memory leaks in a program :

- Pointer taintedness detection

- ECC-memory

But these methods require the original source codes, which are usually unavailable for edge devices

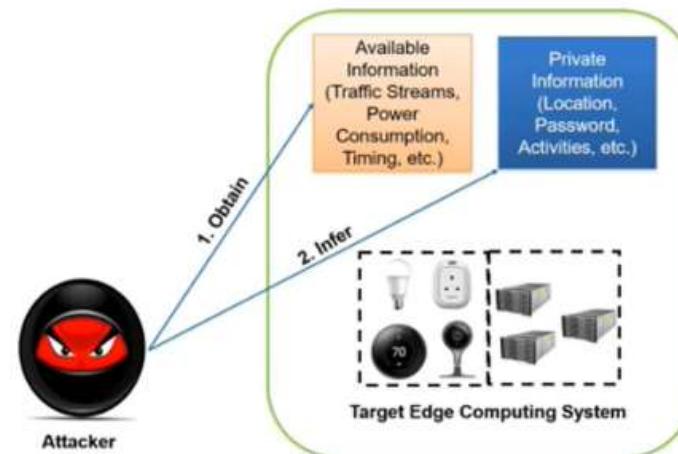
Deep learning models, e.g., recurrent neural networks (RNNs), graph neural networks (GNNs), and deep natural language processing (NLP), used to identify vulnerabilities in firmware with higher accuracy rates

But encryption and antidebug fuses are used in firmwares

An IoT firewall using software-defined networking (SDN) to reduce the attack surface of an exposed IoT device

Side-Channel Attacks and Defense Mechanisms

Definition: “A side-channel attack is **a security exploit that aims to gather information from or influence the program execution of a system** by measuring or exploiting indirect effects of the system or its hardware -- rather than targeting the program or its code directly.”



Attacks exploiting communication channel



Communication signals have a high potential to reveal sensitive information of a victim due to the rich channel information

Two types:

- exploiting packet streams
- exploiting wave signals

Exploiting packet stream

A packet is the atomic unit in most communication channels.

Different coding scheme employed by H.264 and MPEG-4 to reduce temporal redundancy in adjacent video frames can cause severe privacy leak in home surveillance

Simple machine learning algorithms such as k-nearest neighbors (k-NN) and density-based spatial clustering of applications with noise (DBSCAN) - accuracy of 95.8% to infer - four standard human daily activities
dressing, styling hair, moving, and eating

Exploiting packet stream

Attacks can be launched by exploiting the IoT traffic streams

Attack mechanism: a three-step attack

First separating the traffic into individual device flows using the IP addresses of the edge servers

Correlating flow with its responsible IoT device according to unique identifiers

Inferring user activities from the traffic rate changes

Exploiting wave signals

Using EMI (electromagnetic interference)

Attack to infer the video content playing in modern TVs through the discernible EMI signatures

Intentional EMI (IEMI), an attacker can manipulate the input and output signals of an IoT sensory device

Using Wi-Fi: as side channels

A malware-less side-channel attack by exploiting channel state information (CSI) to infer a victim's sensitive password input such as Alipay code based on the finger movements

Exploiting wave signals

Brain–computer interfaces (BCIs) - attacker can successfully capture the raw electroencephalography (EEG) data (i.e., human brain wave data), and combine with machine learning algorithms

- Boosted logistic regression

- Stepwise linear discriminant analysis (SWLDA)

- Fisher's linear discriminant analysis (FLDA)

Attacker can infer victim's banking information, month of birth, face, and geographic location with the accuracies of 15%–40% better than the random guessing attack

Attacks exploiting power consumption

Power consumption is an indicator of the electric usage of a system

Information related to:

- the device that consumes the energy
- the intensity of computations in a computing task

Two types of attacks:

- exploiting power consumption collected by meters
- exploiting power consumption collected by oscilloscopes

Exploiting power consumption collected by meters

Smart meters can accurately measure the electric power consumption of a household

Side-channel inference method named nonintrusive appliance load monitoring (NILM) to monitor simple device states, e.g., ON or OFF, based on the energy consumption of individual appliances

Revised NILM - inference attack to show household activities, such as cooking, washing, laundering, watching TV, gaming

inferred from the energy data available in a smart meter infrastructure

Exploiting power consumption collected by oscilloscopes



Oscilloscope is an instrument measuring the electronic information (e.g., voltage and current) of a hardware device

In embedded devices: chip can perform cryptographic algorithms such as AES-CCM, with a hardcoded secret key in the chip

Research has found that the power consumption of the hardware may be susceptible to leaking the key

Adopting correlation power analysis can completely reverse the AES-CCM master key used to encrypt/decrypt the firmware installed in the Philips hue smart lights to create any malicious firmware and install on any Philips hue smart light over-the-air

Attacks exploiting smartphone-based channels

Smartphones have
advanced OS
Rich system information

Two types of attacks:
exploiting the /proc filesystem
exploiting the smartphone embedded sensors

Exploiting the /proc filesystem

/proc is a system-level filesystem created by the kernel in Linux – used for side-channel attacks

Contains the system information such as interrupt and network data

It is readable by the user-level threads and applications

Exploiting the smartphone embedded sensors



Infer a user's keystroke by analyzing the acoustic sounds emitted from the physical keyboards

Cracked the pattern lock of a smartphone by leveraging the acoustic signals reflected by the fingertip captured through microphones

Tap keystrokes can be inferred using the smartphone accelerometer and gyroscope sensors

Victim's eye movements from a video secretly recorded by a smartphone camera, to infer the victim's keystrokes on a mobile device

Defense Solutions

Defenses against side-channel attacks can be performed from two directions:

- restricting the accesses to side-channel information
- protecting the sensitive data from inference attacks

no feasible defense mechanism that can restrict the access to uncontrollable side channels

well-researched technique for protecting sensitive data from inference attacks - Data perturbation

Malware Injection Attacks and Defense Mechanism

The action to effectively and stealthily inject/install malware into a computing system is called malware injection attack.

This type of attacks is one of the most dangerous ones malware is a significant threat to system security and data integrity.

Cloud has strong computational power to support high-performance firewall or other threat protection systems

But edge servers and devices are vulnerable to malware injection attacks



Malware injection attacks

Malware injection attacks in edge computing into two categories:

Server-side injections (injection attacks targeting edge servers)

Device-side injections (injection attacks targeting edge devices)

Server-side injections

There are mainly four types of injection attacks targeting edge servers

SQL injection

Cross-site scripting (XSS)

Cross-Site Request Forgery (CSRF)

Server-Side Request Forgery (SSRF)

Device-side injections

The most common approach to remotely inject malware is to exploit the zero-day vulnerabilities that can lead to remote code execution (RCE)

“IoT Reaper” virus captured in 2017

infected millions of IoT devices through the Internet protocol and Wi-Fi by exploiting at least 30 RCE vulnerabilities existing in 9 different IoT devices ranging from the network router to IP camera

Smart Nest Thermostat lacks proper protection for firmware update, allowing an attacker to update an arbitrary firmware using a USB connection



BITS Pilani
Pilani Campus

BITS Pilani presentation

Paramananda Barik
CS&IS Department



SEZG586/SSZG586, MEC – Multi-access Edge Computing

Lecture No.14

MEC – Multi-access Edge Computing



Mobile operators - advantage of low latency services that 5G promises

New range of services

faster gaming experience, Augmented/Virtual Reality, connected cars

Azure, AWS, Google

building their own MEC platforms

AWS Wavelength

MEC Architecture

MEC Architecture is based on ETSI model – the defacto body for MEC standards

By understanding the MEC architecture, you:

- As service provider, will know how to evolve to a standard based MEC platform/architecture.
- As vendor/developer, will enable you to tell your customer how your solution fits the ETSI MEC model

Definition of MEC

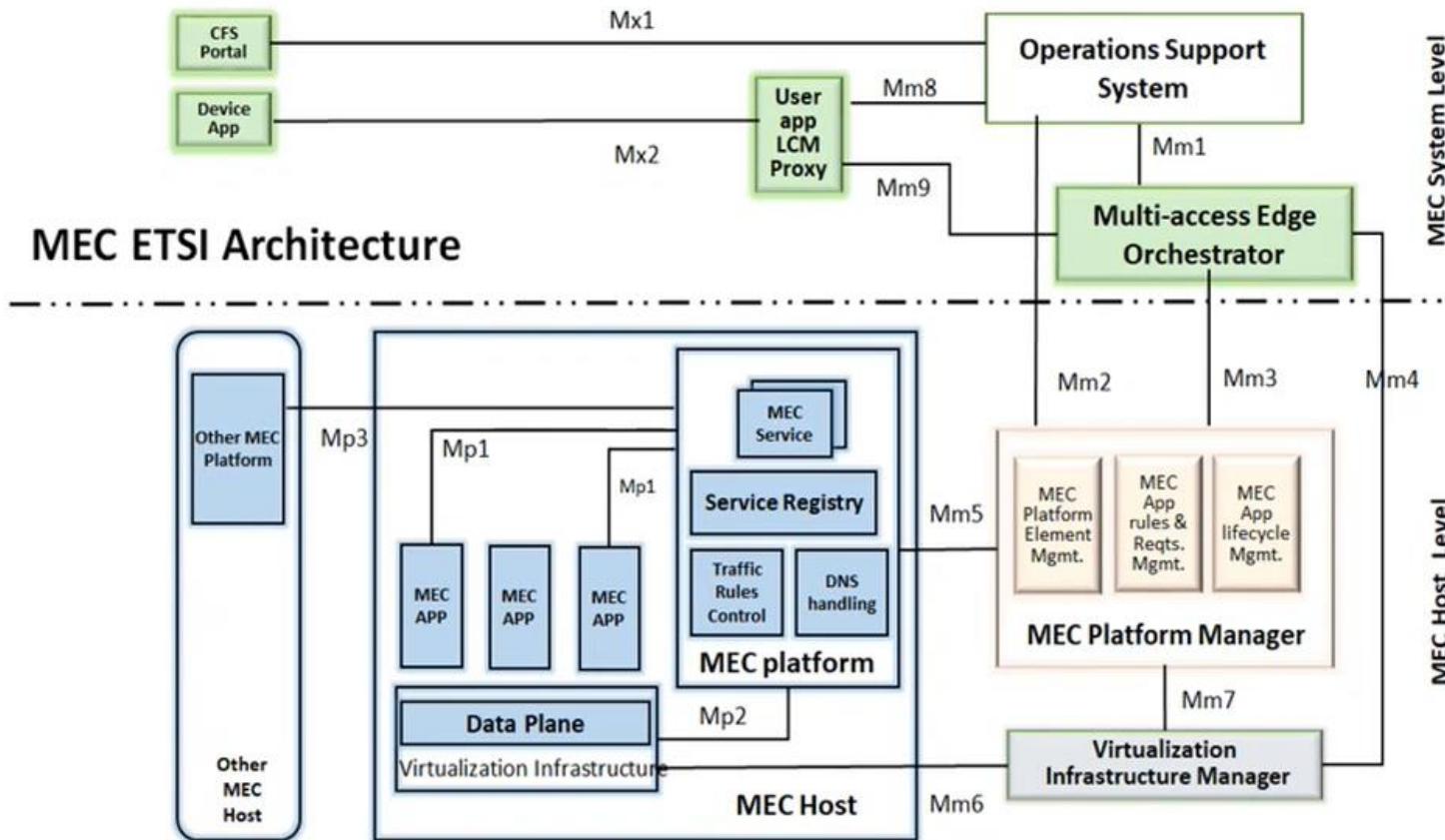
According to the ETSI, the MEC is defined as following

“Multi-access Edge Computing (MEC) offers application developers and content providers **cloud-computing capabilities** and an **IT service environment** at the **edge of the network**. This environment is characterized by **ultra-low latency** and **high bandwidth** as well as **real-time access to radio network information** that can be leveraged by applications.”

ETSI's MEC standards are guided by the following principles:

- Edge technology should have a virtualization platform to be considered MEC.
 - MEC can be deployed at radio nodes, aggregation points, and the edge of the core network.
 - APIs in a MEC environment should be simple, controllable, and if possible, reusable for other tasks.
 - Since the compute, storage, and network resources that a MEC application requires may not match what are available at a node, a MEC network needs a system-wide lifecycle management of applications to handle these variables correctly.
 - MEC systems must be able to relocate a mobile edge application running in an external cloud to a MEC host and back while fulfilling all of the application's requirements
-

MEC ETSI architecture



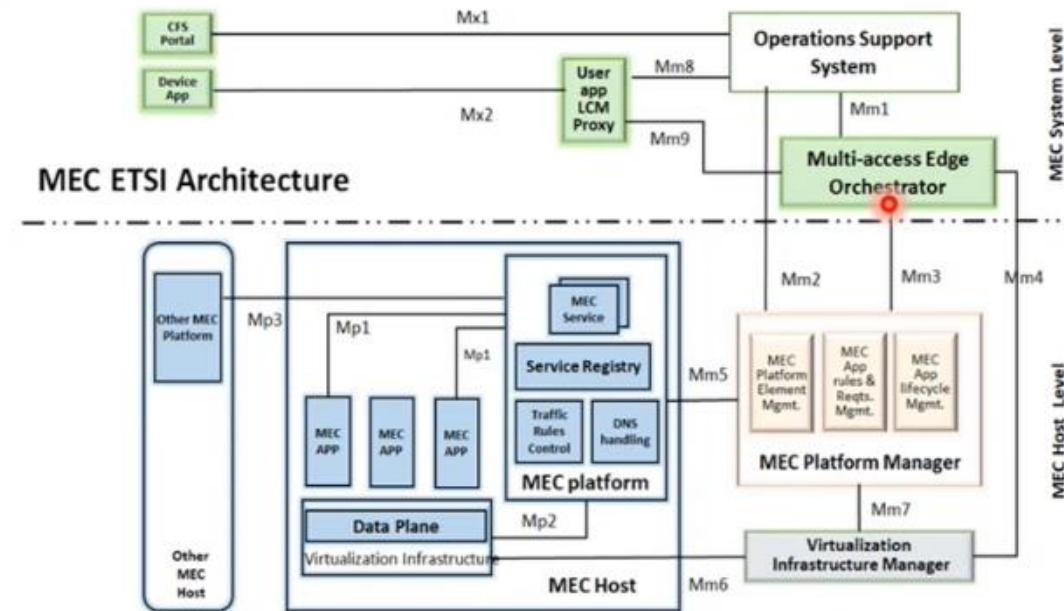
MEC ETSI architecture

Three distinct blocks:

MEC Host

MEC Platform Manager

MEC Orchestrator



Virtualization Infrastructure Manager(VIM)

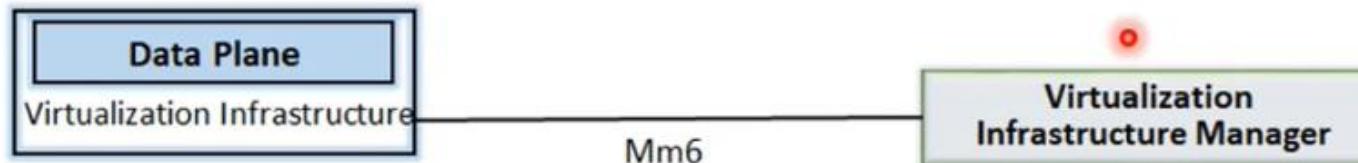


Manage the virtual machines (VMs) on top of physical infrastructure (compute, storage, and networking).

It is responsible for

- allocating
- maintaining
- releasing

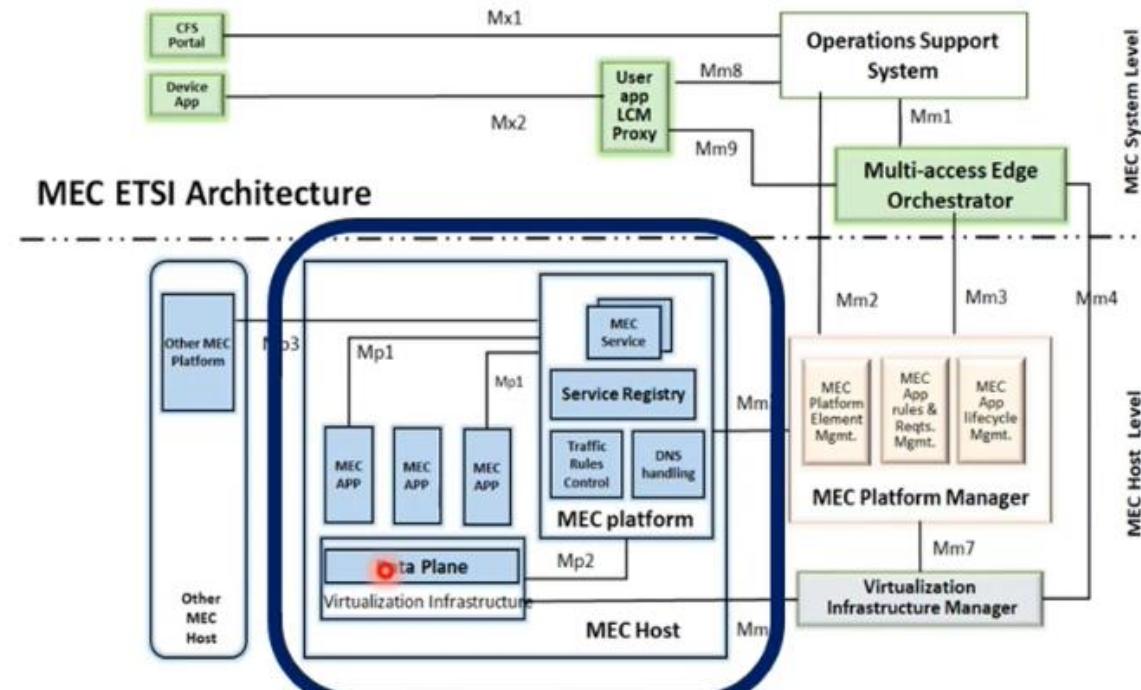
virtual resources of the “virtualization infrastructure”.



MEC Host

Virtualization Infrastructure

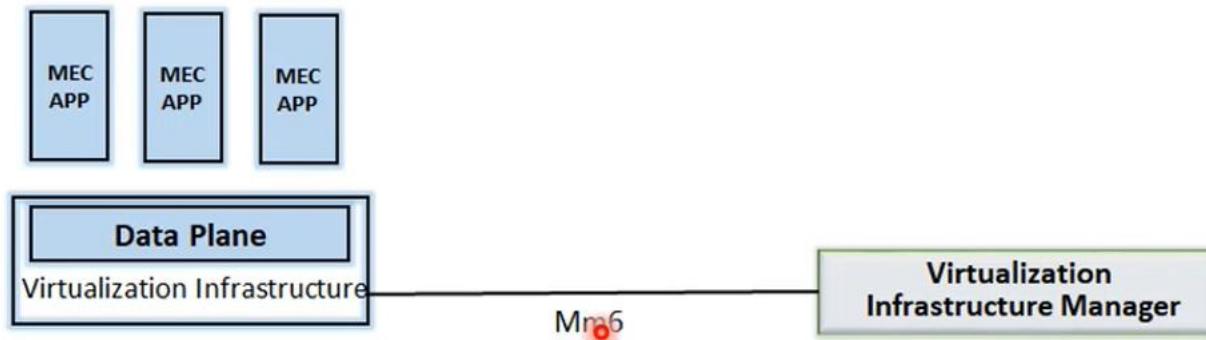
- MEC Apps
- MEC Platform



MEC Application

These are the actual applications that are run in MEC on top of VMs

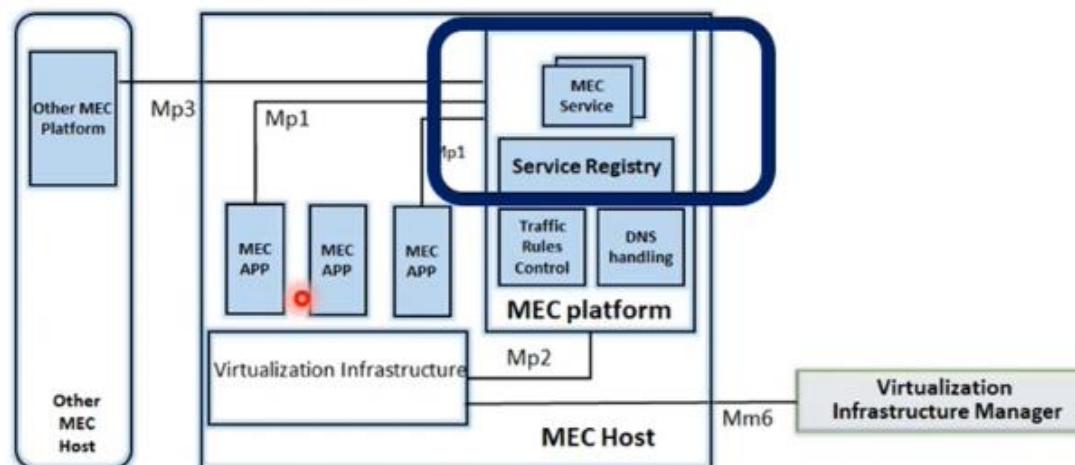
Actual apps in MEC like an application for gaming or Virtual Reality or Augmented Reality



MEC Service

“MEC Service” is an important block in MEC.

- The network-related APIs are exposed by MEC service to MEC Apps through reference point Mp1.
- Also, the MEC platform can consume these services.



MEC Service

MEC Apps are network-aware

MEC Service help by exposing the network information through APIs

At least three types of following services must be exposed by “MEC service”

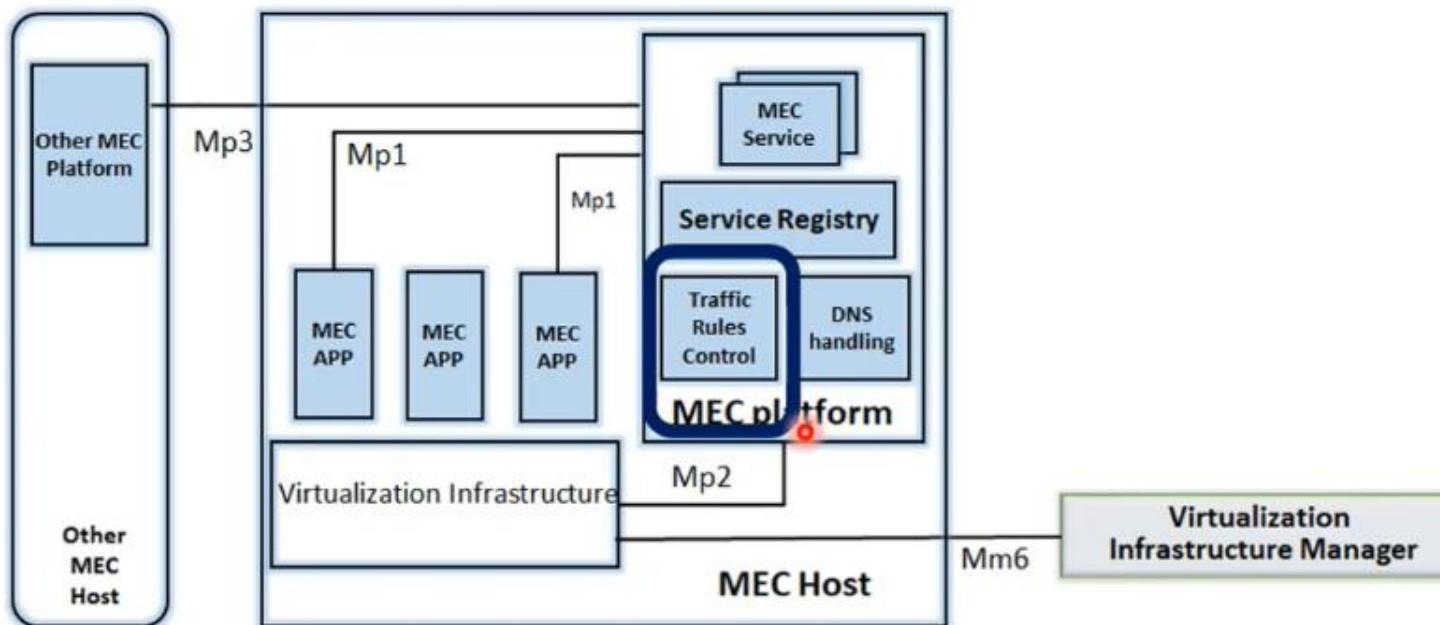
They are

- Radio Network conditions.
- Location information (for example, location of a User Equipment(UE))
- Bandwidth Manager- The Bandwidth Manager service, when available, allows allocation of bandwidth to certain traffic routed to and from MEC applications and its prioritization

They are part of service registry

Traffic Rules Control

As MEC Platform is serving multiple applications, simultaneously, It should be able to assign priorities through “Traffic rules control”

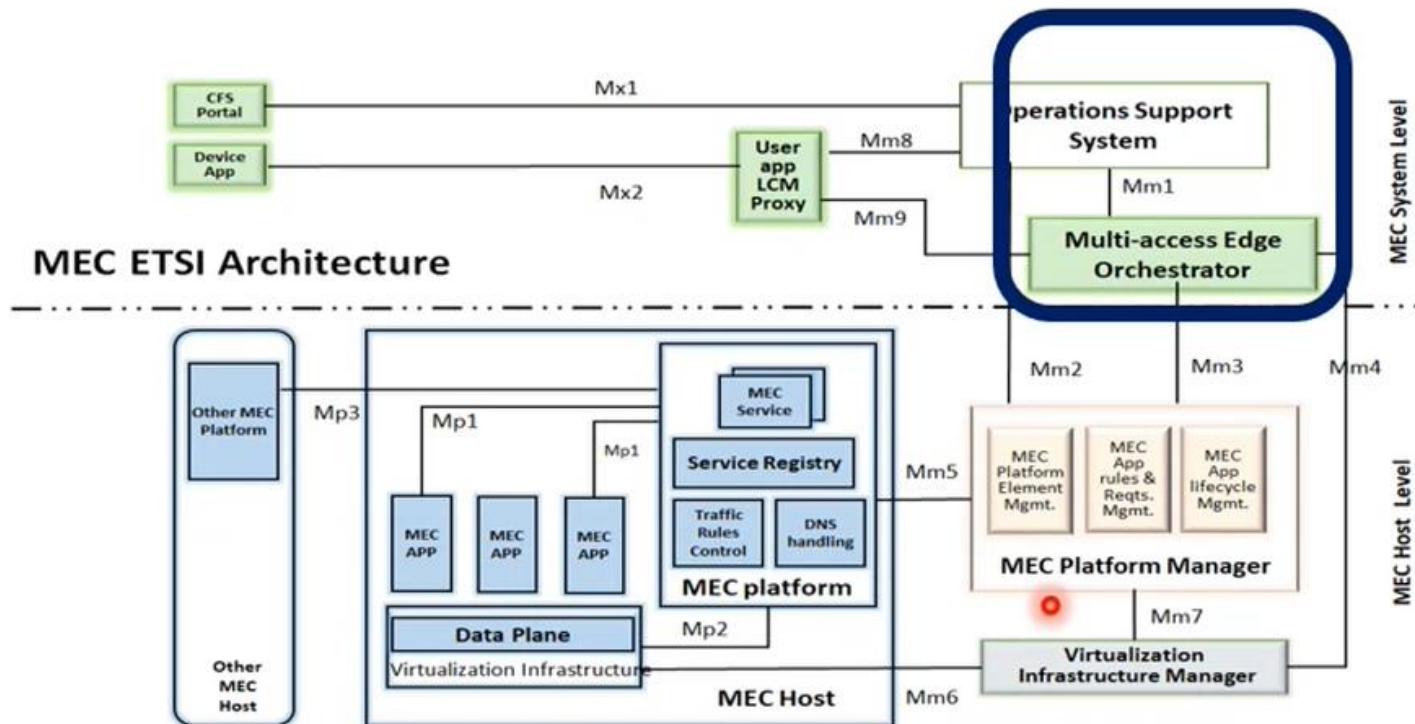


MEC Platform Manager

MEC Platform manager performs following functions:

- MEC Apps' Life-cycle management:
 - instantiating, maintaining and tearing down MEC Apps on VMs
- Element Management:
 - FCAPS (Fault, Configuration, Accounting, Performance, Security) management for the MEC platform
- Managing the application rules, traffic rules, DNS configuration

Multi-Access Edge Orchestration

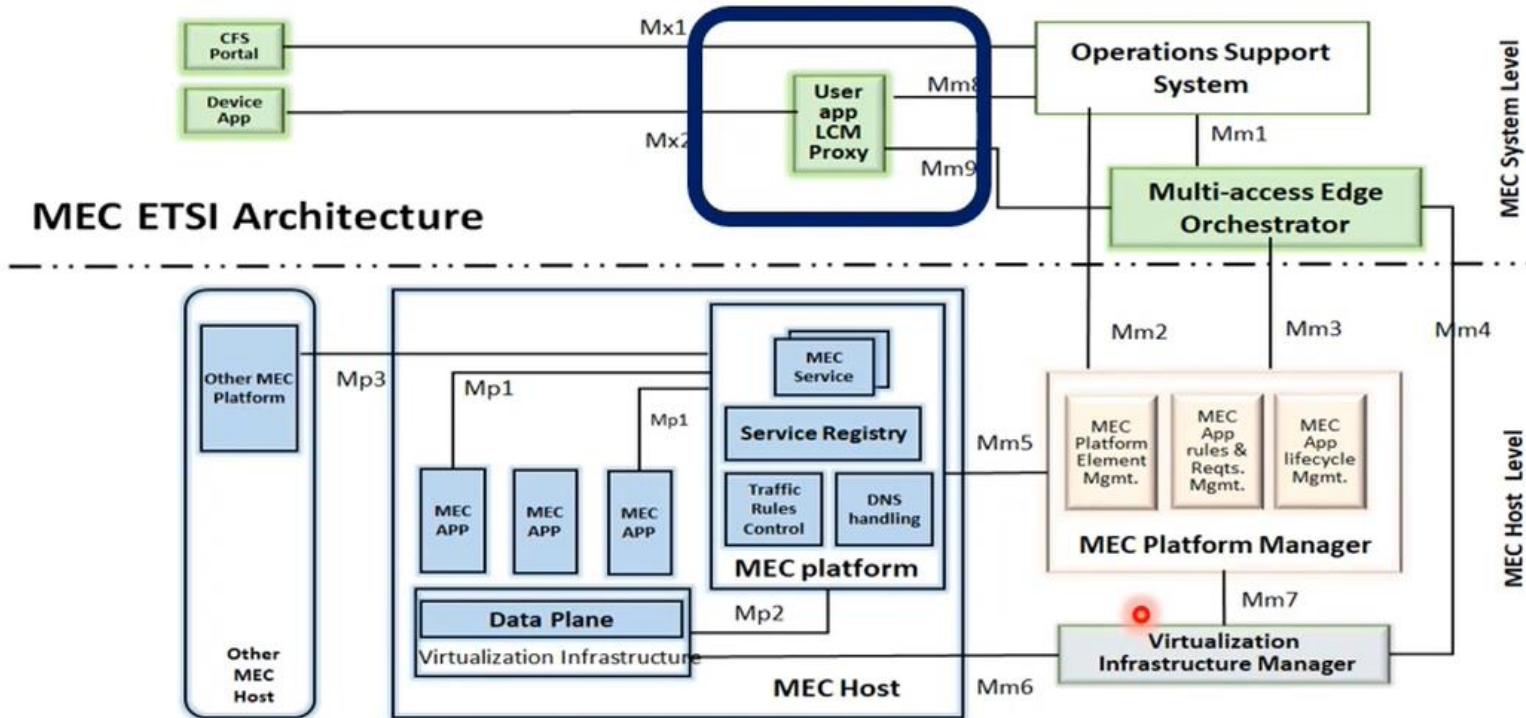


MEC Orchestration

It does the following functions

- Lifecycle management of MEC Apps (Compare this with MEC Platform manager, which can do a similar function). The Orchestrator achieves this function by talking to the application through the MEC platform manager.
- On-boarding of application packages, including checking the integrity and authenticity of the packages.
- Selecting appropriate MEC host(s) for application instantiation based on constraints, such as latency, available resources, and available services

User App LCM Proxy



Intelligent Video-Acceleration

Video Download Service

Media delivery transitioning to HTTP-based transport
uses TCP at L4

Performance issues of using TCP over wireless networks

Idea: To include a throughput guidance component at edge
Application can adapt its video codec and TCP parameters
to the conditions of a wireless network

Improving the performance of video transmission from
all perspectives
user experience
network load

Intelligent Video Acceleration

Options:

An edge-based video codec

Or

Just a “throughput guidance” service

Reads the radio network information

Converts the information provided into a “guidance report”

Send it to a video coder/transcoder elsewhere in the cloud

Optionally - throughput guidance component may also configure traffic rules

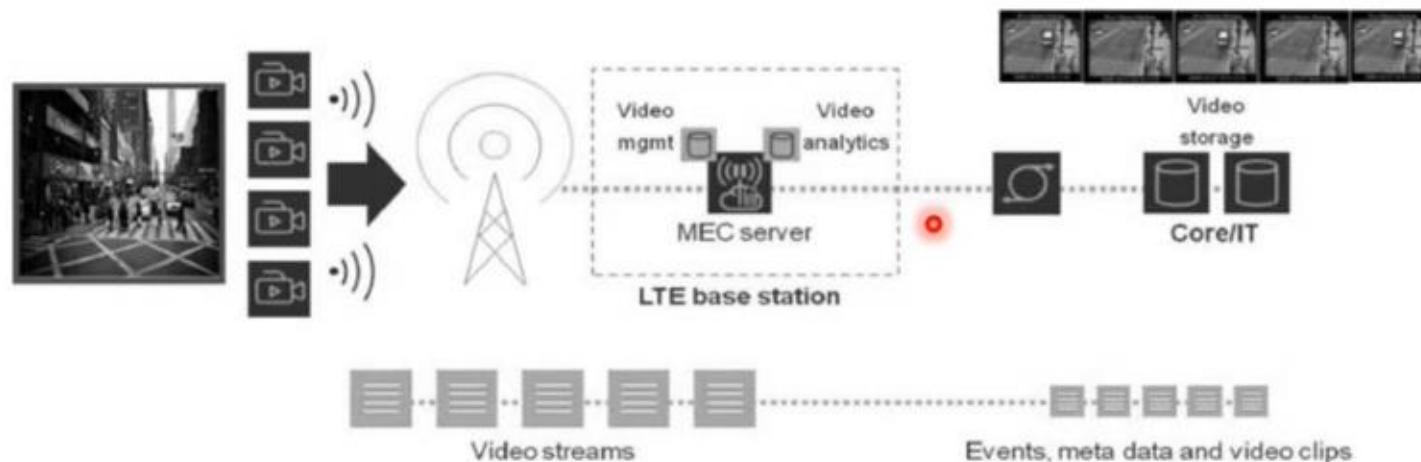
handle of the application’s traffic - it may decide between a mobile RAN- and a WLAN-based access

Video Stream Analysis

Video upload service

Surveillance system - consisting of a number of cameras with a cloud-based video-analytics system

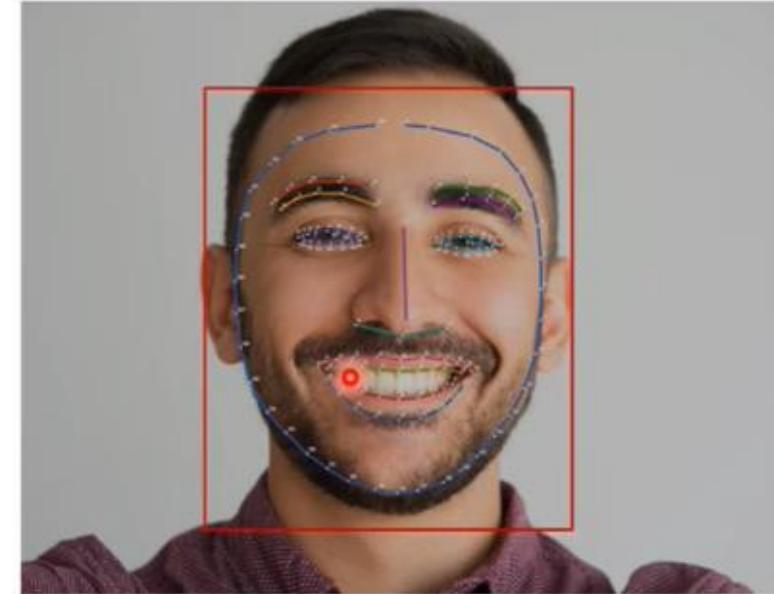
System - transmission of multiple video streams for feature extraction and analysis



Video Stream Analysis

Edge-based pre-processing

- At the edge, raw video images are processed extracting only the information containing relevant features.
- The extracted information is forwarded into the cloud, which performs further image processing for extractions and then performs analysis



Example: facial recognition system

Augmented Reality

A smart museum augmented reality application

A smart phone or a viewing device (smart glasses) is pointed at a museum object

Relevant information is augmented by local servers



Locally processing and delivery of such data makes sense

- end-user QoE perspective
- network performance perspective
- system design perspective

Connected Vehicles

MEC is vehicular automation
strict latency requirement
perform computation on an aggregate of multi-vehicle data



IoT Gateways

IoT Gateway is a key function – usually realized in a stand-alone device

- communication aggregation

- computational offload

- identity proxy



Public Edge Clouds – AWS Greengrass

Data Analytics in the Cloud and Edge



Analytics for the IoT segment deals with:

- **Structured data** (for example, SQL storage): A predictable format of data
- **Unstructured data** (for example, raw video data or signals): A high degree of randomness and variance
- **Semi-structured** (for example, Twitter feeds): Some degree of variance and randomness in form

Data Analytics

Data need to be interpreted and analyzed

- In real time as a streaming dataflow
- It may be archived and retrieved for deep analytics in the cloud.
- It is simply logged and dumped to a data lake like a Hadoop database.
- This is the **data ingest** phase.

Data Analytics

- Staging Phase
 - A messaging system like Kafka will route data to a
 - Stream processor
 - Batch processor, or
 - Both

Data Analytics – Staging Phase-Stream Processing



- Stream processing accepts a continuous stream of data.
- Processing is typically **constrained** and **very fast**, as the data is **processed in memory**.
Therefore, processing must be as fast, or faster, than the rate of data entering the system.
- While stream processing provides near-real-time processing in the cloud, when we consider industrial machinery and self-driving cars, **stream processing does not provide hard real-time operating characteristics**.

Data Analytics

Data analytics intends to find events, usually in a streaming series of data

Analytic functions:

- Preprocessing
- Alerting
- Windowing
- Joins
- Errors
- Databases
- Temporal events and patterns
- Tracking
- Trends
- Batch queries
- Deep analytics pathway
- Models and training
- Signaling
- Control

Analytic function: Processing

Preprocessing includes

Filtering out events of little interest

Denaturing

Feature extraction

Segmentation

Transforming data to a more suitable form (although data lakes prefer no immediate transformation), and

Adding attributes to data such as a tag

Analytic function: Alerting

Inspect data, and if it exceeds some boundary condition, then raise an alert.

The simplest example is when the temperature rises above a set limit on a sensor.

Analytic function: Windowing

A sliding window of events is created that only draws rules upon that window.

Windows can be based on time (for example, one hour), or length (2000 sensor samples).

Sliding windows

Ex: inspect only the 10 latest sensor events and produce a result whenever a new event arises

Batch windows

Ex: produce an event only at the end of the window

Windowing is good for rules and for counting events.

For instance, you could look for the number of temperature spikes in the last hour and resolve that a defect will occur on some machine.

Analytic function: Joins

Joins combine multiple data streams into a new single stream.

A scenario where this applies is a logistics example, say a shipping company tracks their shipments with asset-tracking beacons and their fleet of trucks, planes, and facilities have geolocation information streaming, as well.

There are initially two streams of data: one for the package and one for a given truck.

The two streams **join** when a truck picks up a package

Analytic function: Errors

Millions of sensors will generate

- Missing data

- Garbled data

- Data that is out of sequence.

This is important in IoT cases with multiple streams of asynchronous and independent data.

For example, data may be lost in a cellular WAN if a vehicle enters an underground parking garage.

Analytic function: Database

The analytics package will need to interact with some data warehouse.

For example, if data is streaming in from a number of sensor tags

An example: Bluetooth asset tags tracking whether an item is stolen or lost.

A database of missing tag IDs would be referenced from all the gateways streaming in Bluetooth tag IDs to the system.

Analytic function: Temporal events and patterns



Most often used with the window pattern

Here, a series or sequence of events constitutes a pattern of interest – a state machine.

Ex: monitoring the health of a machine based on temperature, vibrations, and noise.

A temporal event sequence could be as follows:

1. Detect whether the temperature exceeds 100° C.
2. Then detect whether vibrations exceed 1 m/s.
3. Next, detect whether the machine is emitting noise at 110 dB.
4. If those events take place in that sequence, only then raise an alert.

Analytic function: Tracking

Tracking involves

when or where something exists or an event has occurred, or
when something doesn't exist where it should.

A very basic example is geolocation of service trucks

This has application in agriculture, human movement, tracking patients, tracking high-value assets, luggage systems, smart city garbage, snow removal, etc.

Analytic function: Trends

This pattern is particularly useful for predictive maintenance.

Here, a rule is designed to detect an event based on time-correlated series data.

This is similar to temporal events but differs in the sense that temporal events have no notion of time, only sequence order. This model uses time as a dimension in the process.

Ex: A running history of time-correlated data could be used to find patterns like a livestock sensor does in farming.

Analytic function: Batch Queries

Batch processing is more comprehensive and deeper than real-time stream processing.

A well-designed streaming platform can fork analysis and call into a batch processing system.

Analytic function: Deep analytics pathway



In real-time processing,

Decisions are made on the fly that some event has occurred.

Whether that event really should signal an alarm may require further processing that will not operate in real time.

An example is a video surveillance system in smart city.

A smart city issues an amber alert for a lost child.

The smart city can issue a simple feature extraction and classification model for the real-time streaming engines.

The model would detect license plates for a vehicle the child may be in, or potentially a logo on the child's shirt.

The first step would be to image-capture vehicles' license plate numbers or logos on pedestrians and send them to the cloud. The analysis package may identify a plate of interest or a logo out of millions of image samples as a first-level pass.

Analytic function: Models and training

The first-level model described previously may, in fact, be an inference engine for a machine learning system.

These machine learning tools are built on trained models that can be used for in-flight, real-time analysis.

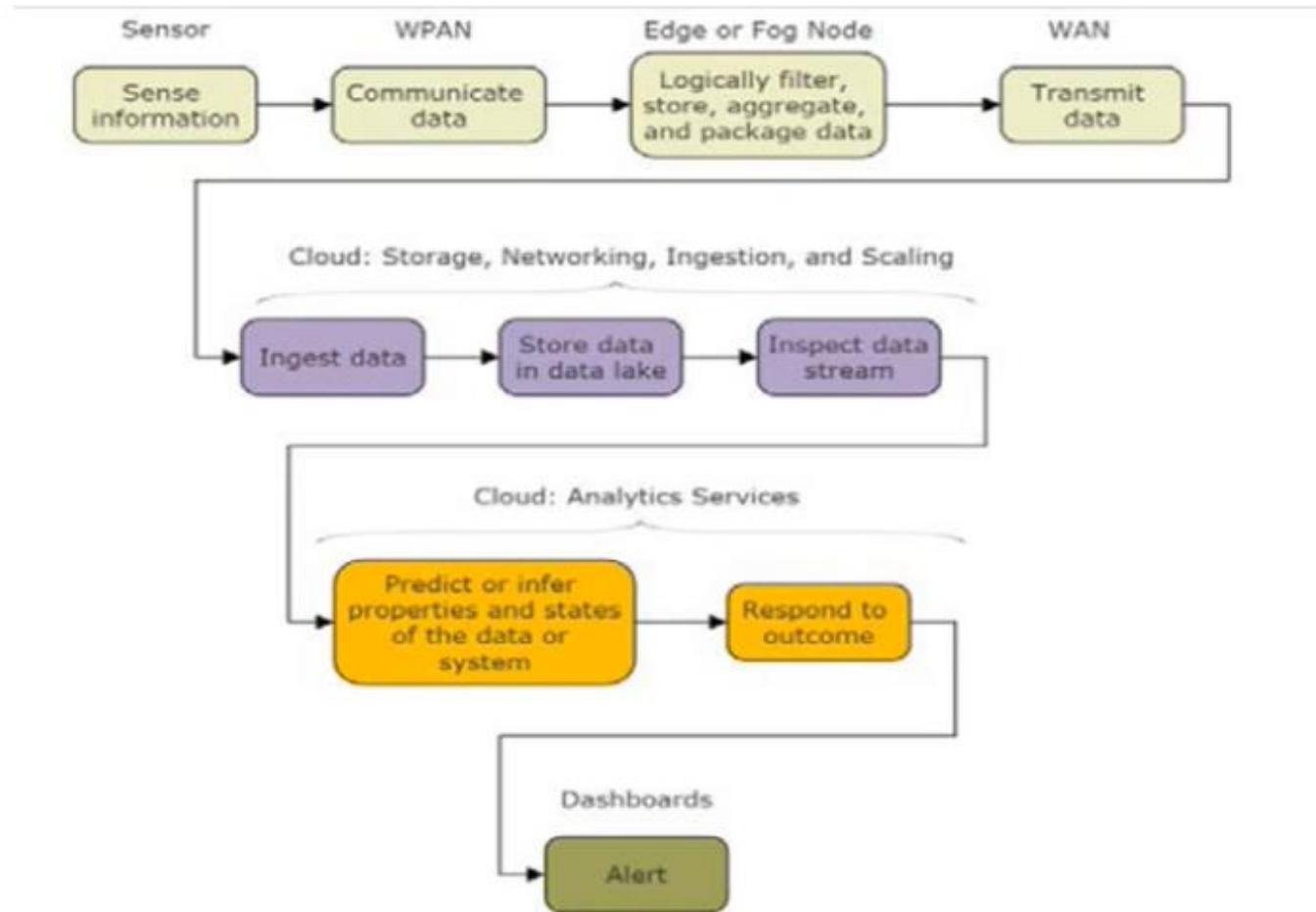
Analytic function: Control

Facilities need to be in place to manage this system

A way is needed to control these analysis tools:

- Starting
- Stopping
- Reporting
- Logging
- Debugging

Top-level cloud pipeline



Data Analytics

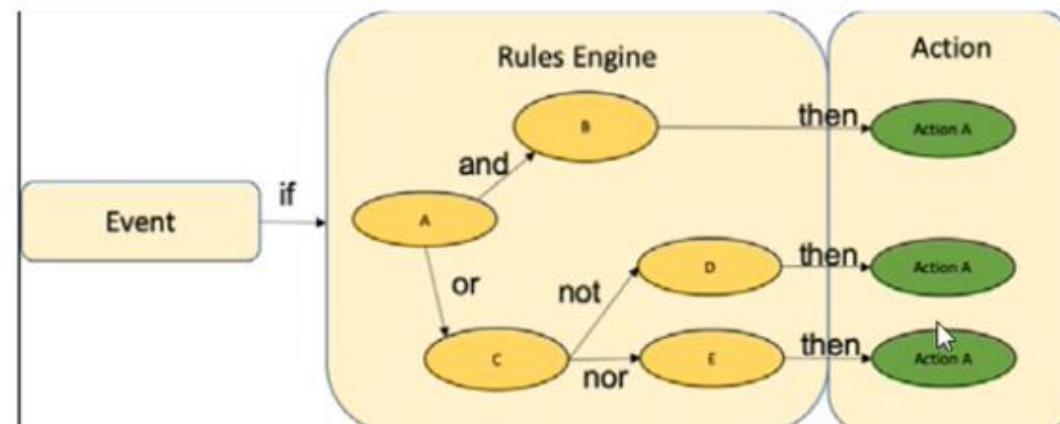
The analytics (predict-respond) portion of the cloud can take on several forms:

- **Rules engines:** These simply define an action and produce an outcome.
- **Stream processing:** These are where events like sensor readings are injected into the stream processor.
 - The processing path is a graph where nodes in the graph represent operators and send events to other operators.
 - The nodes contain the code for that portion of the processing and a path to connect to the next node in the graph.
 - This graph can be replicated and executed in parallel on a cluster so it is amenable to scaling up to hundreds of machines.

Rule engines

A rules engine is simply a software construct that executes actions on events. For example, if the humidity in a room exceeds 50%, send an SMS message to the owner. These are also called business rule management systems (BRMSs).

Rules engines may or may not have state and be called stateful.



Rule engines

Drools can support two forms of chaining:
forward and backward.

Pseudocode: Smoke Sensor = Smoke Detected Heat Sensor = Heat Detected

Forward

```
if (Smoke_Sensor == Smoke_Detected) && (Heat_Sensor == Heat_Detected)
then Fire

if (Smoke_Sensor == !Smoke_Detected) && (Heat_Sensor == Heat_Detected)
then Furnace_On

if (Smoke_Sensor == Smoke_Detected) && (Heat_Sensor == !Heat_Detected)
then Smoking

if (Fire) then Alarm

if (Furnace_On) then Log_Temperature

if (Smoking) then SMS_No_Smoking_Allowed
```

Let us assume:

Smoke_Sensor: Off
Heat Sensor: On

Rule engines

Backward:

1. Can we prove the temperatures are being logged? Take a look at this code:

```
if (Furnace_On) then Log_Temperature
```

2. Since the temperatures are being logged, the antecedent (Furnace_On) becomes the new goal:

```
if (Smoke_Sensor == !Smoke_Detected) && (Heat_Sensor == Heat_Detected) then Furnace_On
```

3. Since the furnace is proven to be on, the new antecedent comes in two parts: Smoke_Sensor and Heat_Sensor. The rules engine now breaks it up into two goals:

```
Smoke_Sensor off  
Heat_Sensor on
```

4. The rules engine now attempts to satisfy both the subgoals. Upon doing so, the inference is complete.

Stream Processing

Ingestion – streaming, processing, and data lakes

- An IoT device is usually associated with some sensor or a device whose purpose is to measure or monitor the physical world.
 - It does so asynchronously with respect to the rest of the IoT technology stack.
 - That is, a sensor is always attempting to broadcast data, whether or not a cloud or fog node is listening.
-

Stream Processing

The IoT stream from a sensor to a cloud is assumed to be:

- Constant and never-ending
 - Asynchronous
 - Unstructured, or structured
 - As close to real time as possible
-

Stream Processing

Streaming system must:

- Scale with event growth and spikes
- Provide a publish/subscribe API to interface
- Approach near-real-time latency
- Provide scaling of processing of rules
- Support data lakes and data warehousing

Ex: Apache ~~Spark~~ is a stream processing framework that processes data in small batches.

Stream Processing - Spark

Apache Spark is a stream processing framework that processes data in small batches.

It is particularly useful when memory size is constrained on a cluster in the cloud (for example, 1TB).

Spark is built on in-memory processing, which has the advantages of reducing filesystem dependency and latency

Use Cases

Industry	Use cases	Cloud services	Typical bandwidth	Real time	Analytics
Manufacturing	Operational technology Brownfield Asset tracking Factory automation	Dashboards Bulk storage Data lakes SDN Low latency	500 GB/day/factory part produced 2 TB/minute mining operations	Less than 1s	RNN Bayesian networks
Logistics and transport	Geolocation tracking Asset tracking Equipment sensing	Dashboards Logging Storage	Vehicles: 4 TB/day/vehicle (50 sensors) Aircraft: 2.5 to 10 TB/day (6000 sensors) Assets tracking: 1 MB/day/beacon	Less than 1s (real time) Daily (batch)	Rules engines
Healthcare	Asset tracking Patient tracking Home health monitoring Wireless health equipment	Reliability and HIPPA Private cloud option Storage and archival Load balancing	1 MB/day/sensor	Less than 1s: life critical Non-life critical: on each change	RNN Decision trees Rules engines

Use Cases

Running locally, deep learning models categorize packages,

- check if the postage matches a parcel's size, weight, and destination, and decipher barcodes, even the damaged ones.

- Edge intelligence also helps locate missing parcels: With AI, this takes a couple of people and just a few hours — instead of the previous several days and 8 to 10 people.

Kepler Night Nurse Edge Box increases patient safety



Kepler Vision, a Dutch medtech company, designed its Night Nurse Edge Box to keep elderly patients safe at night. The device runs Kepler software for detecting falls and physical distress and alerting staff when their help is required.

Instead of sending visual data to the cloud, Edge Boxes use computer vision locally and decide if nurses should intervene. This makes the system unaffected by the Internet connection breakdowns. Besides, it was estimated that replacing old sensors with Edge Boxes eliminates **99 percent** of false alarms.

Spanish connected smart tunnel offers assistance to drivers



The Cereixal tunnel in Spain's Galicia region leverages 5G and edge computing to capture and analyze data from tunnel sensors, cameras, and connected vehicles. Managers can remotely monitor what is happening inside the infrastructure via a dashboard.

Drivers who are moving through the tunnel receive notifications and alerts on the presence of pedestrians and emergency vehicles, possible delays because of traffic jams or accidents, weather conditions at the exit, and more. The project is supported by leading telecommunication companies Telefonica and Nokia.

