BITS Pilani presentation

BITS Pilani
Pilani Campus

Paramananda Barik
CS&IS Department

BITS Pilani
Pilani Campus

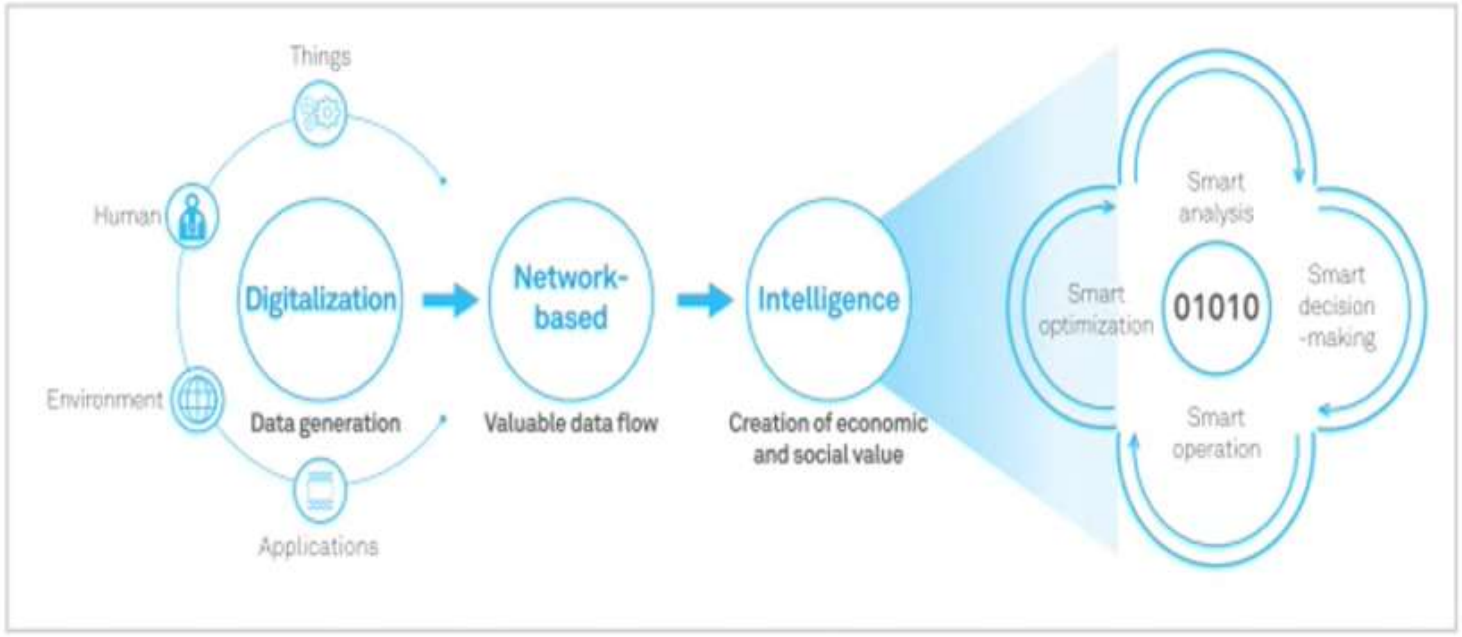
SEZG586/SSZG586, Edge Computing

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

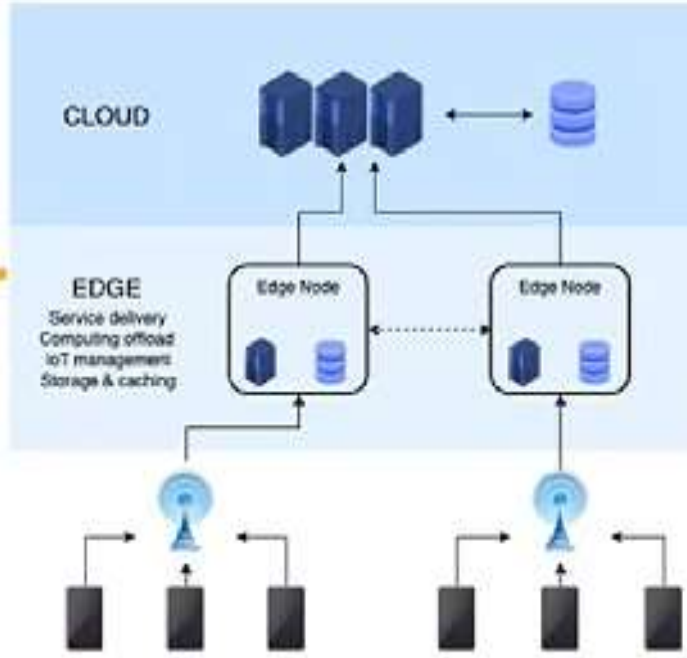
"Edge computing in telecom, often referred to as Mobile Edge Computing, MEC, or Multi-Access Edge Computing, provides execution resources (compute and storage) for applications with networking close to the end users, typically within or at the boundary of operator networks." – **Ericsson**

"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**

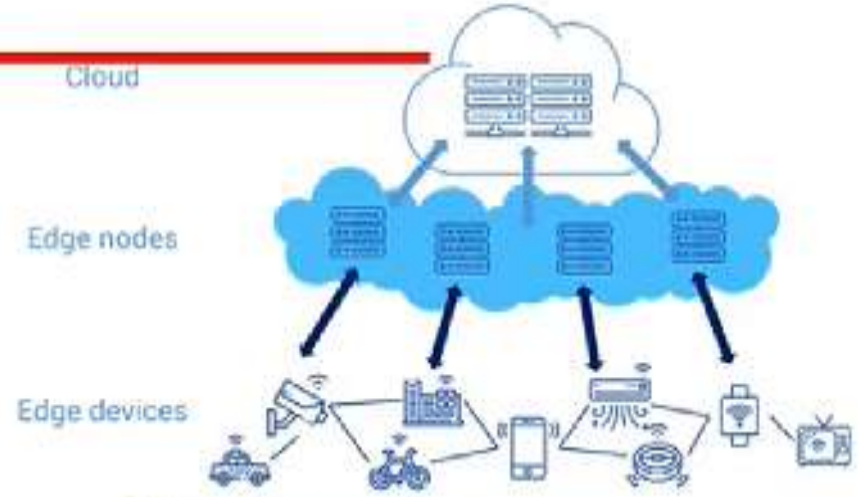
"Edge computing is a distributed computing framework that brings enterprise applications closer to data sources such as IoT devices or local edge servers." – **IBM**

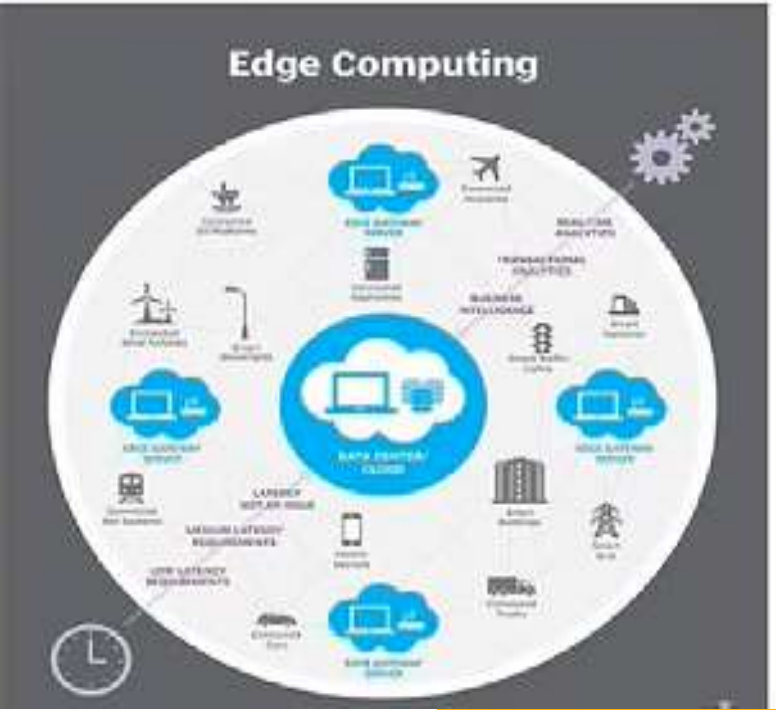
"Edge computing is part of a distributed computing topology where information processing is located close to the edge, where things and people produce or consume that information." – **Gartner**

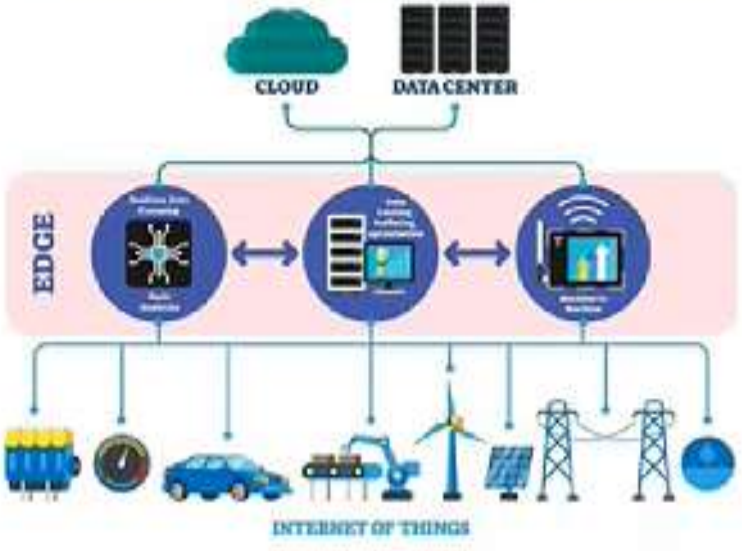
"Edge computing is a distributed computing paradigm that brings computation and data storage closer to the sources of data. This is expected to improve response times and save bandwidth. "A common misconception is that edge and IoT are synonymous"" - Wikipedia


CLOUD
EDGE
Service delivery
Computing offload
IoT management
Storage & caching
Edge Node
Edge Node


innovate
achieve
lead


Cloud
Edge nodes
Edge devices


Edge Computing


Edge Computing
CLOUD
DATA CENTER
EDGE
INTERNET OF THINGS

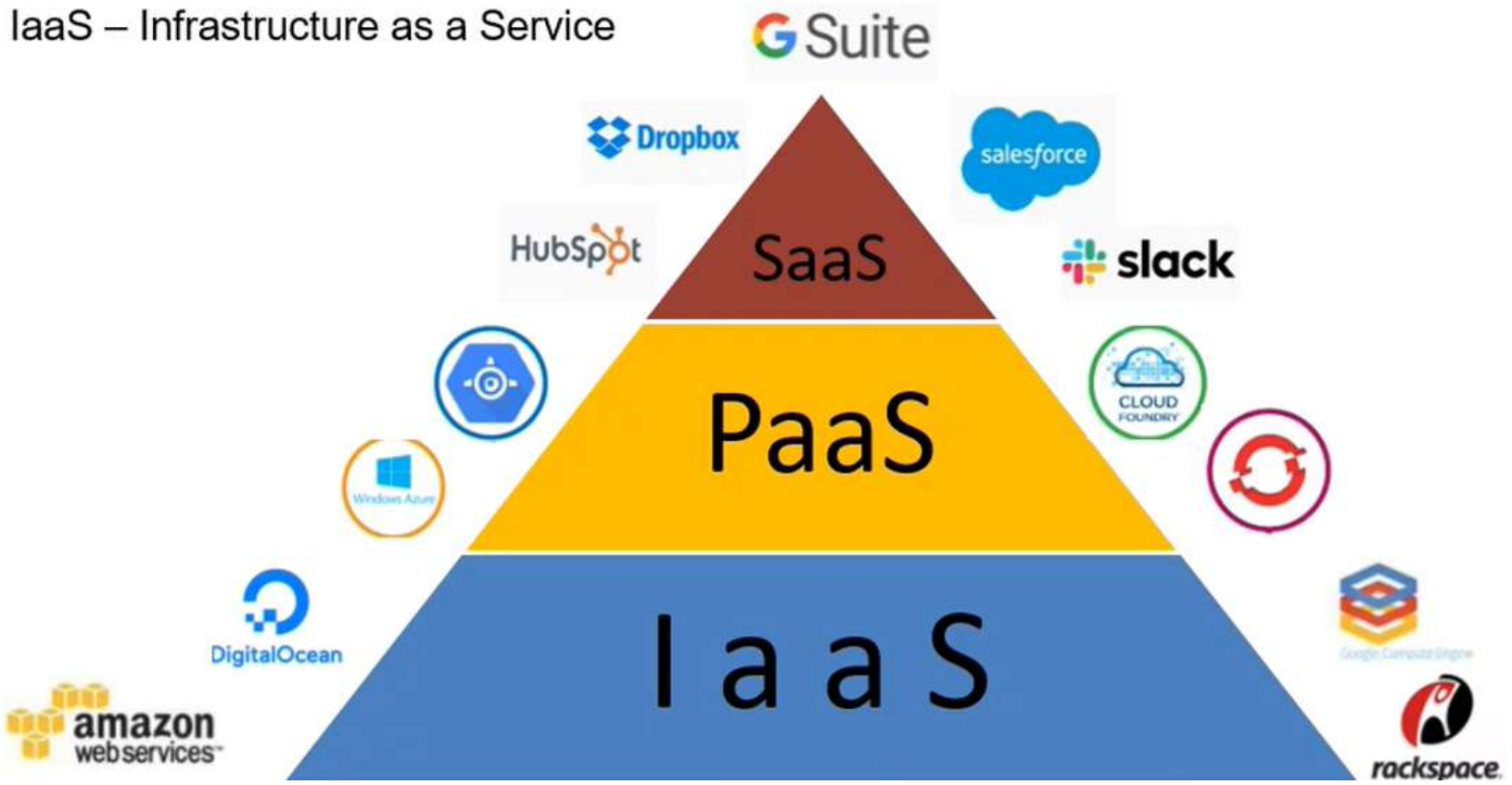
Before EDGE

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?

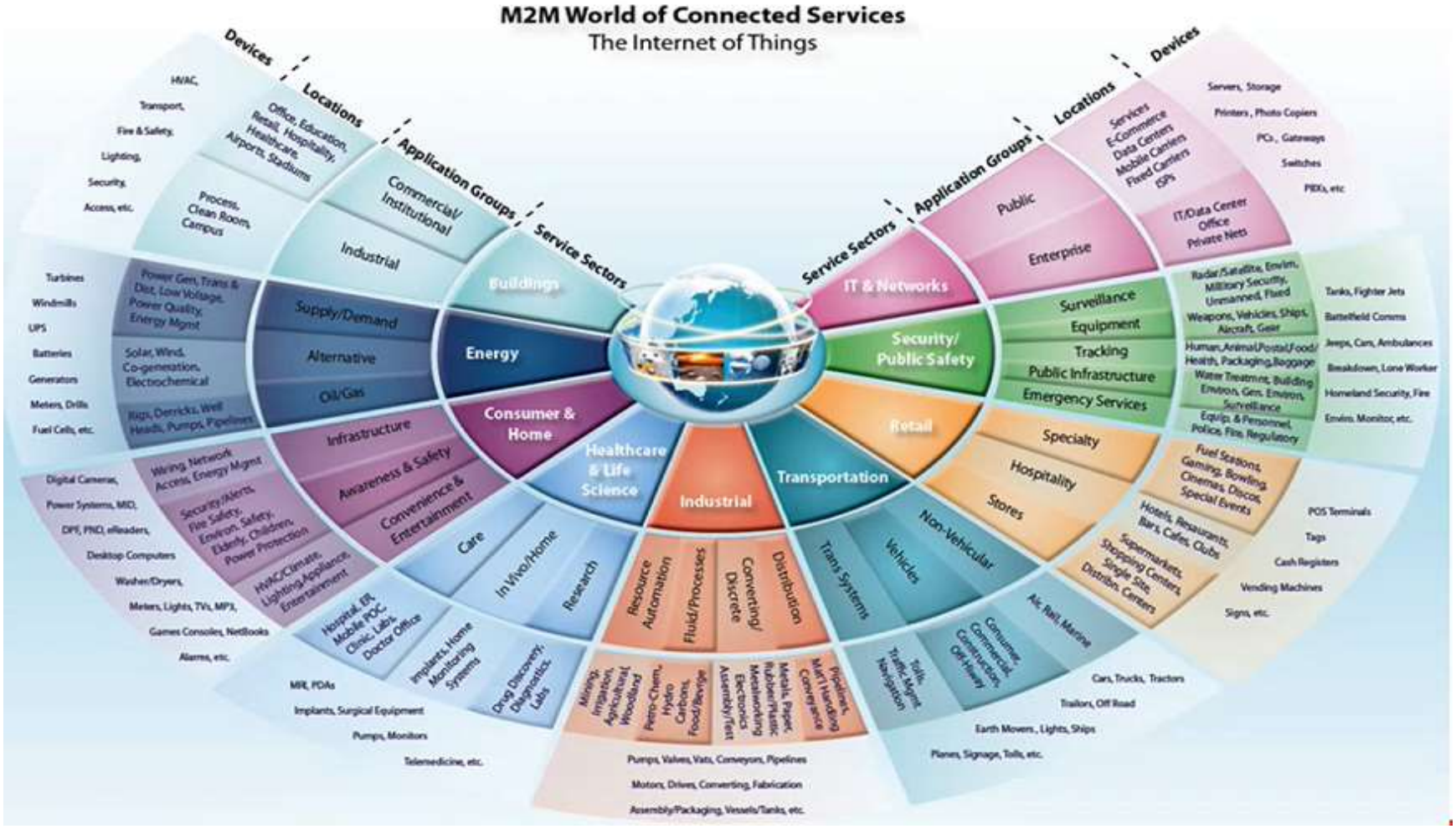
Internet connects people - "Internet of people"

IoT connects all things – "Internet of Things"

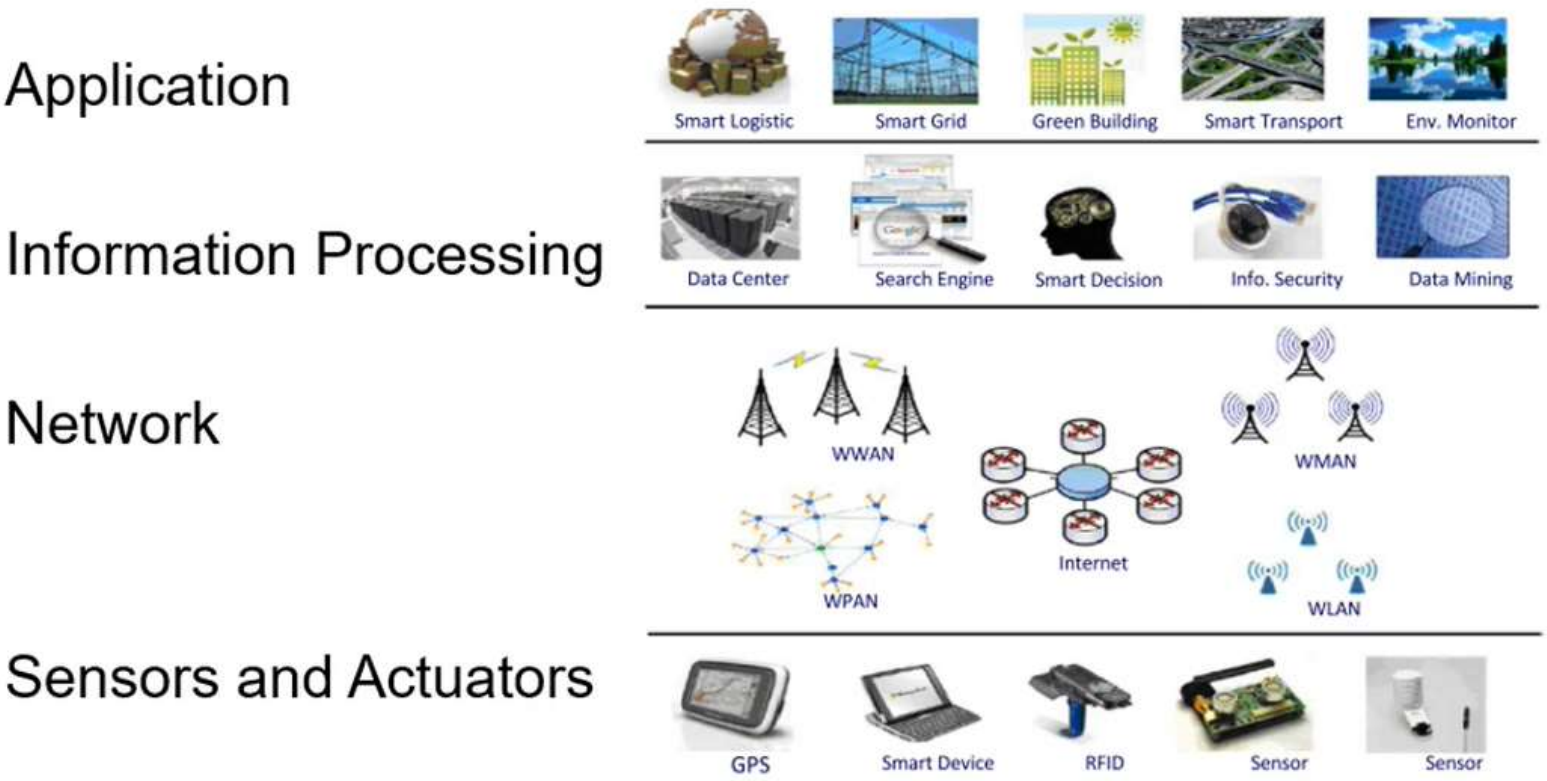
Interconnection of Machines or Devices or Things, and communicate with each other via Internet

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

Internet of Things



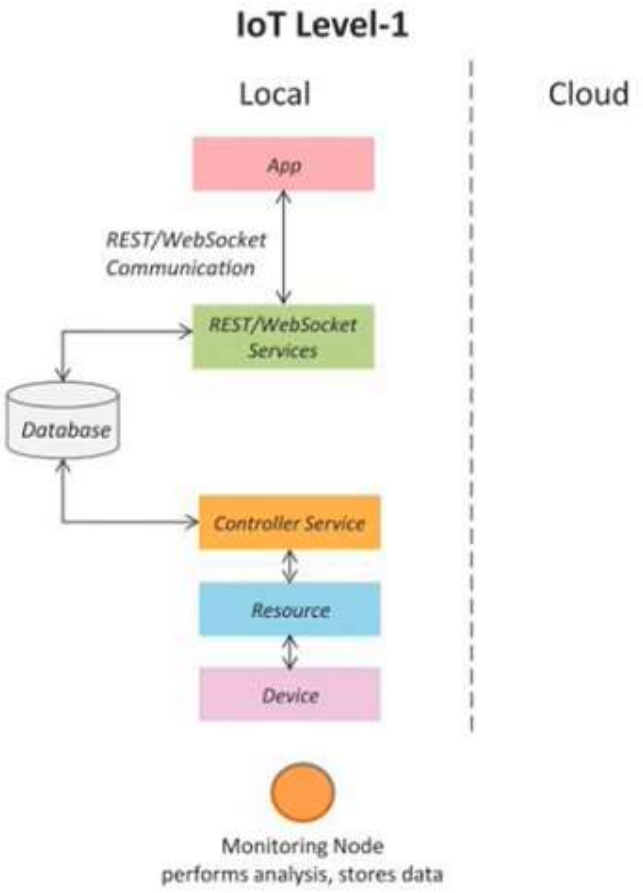
IoT: 4 Layer model



IoT Level 1

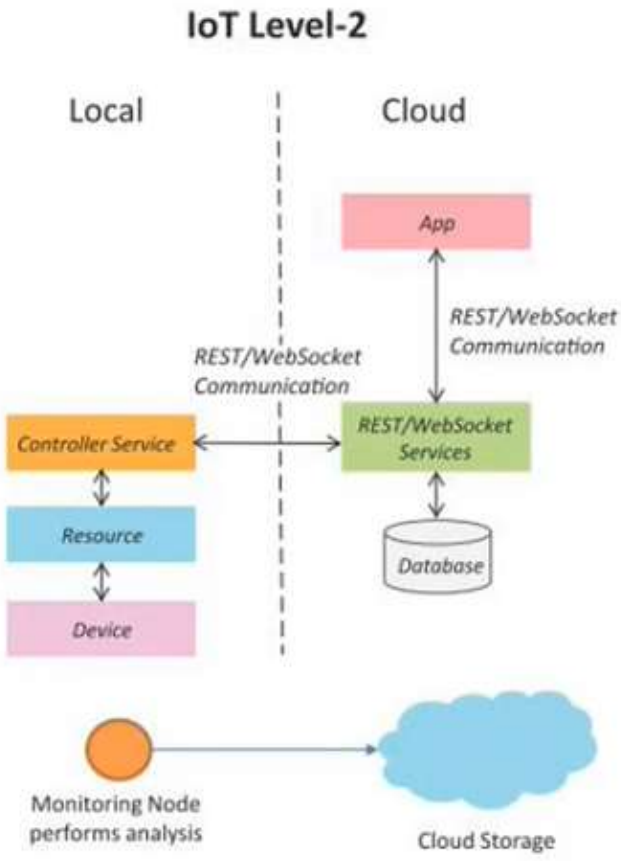
- A level-1 IoT system has a single node/device that performs sensing and/or actuation, stores data, performs analysis and hosts the application

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.



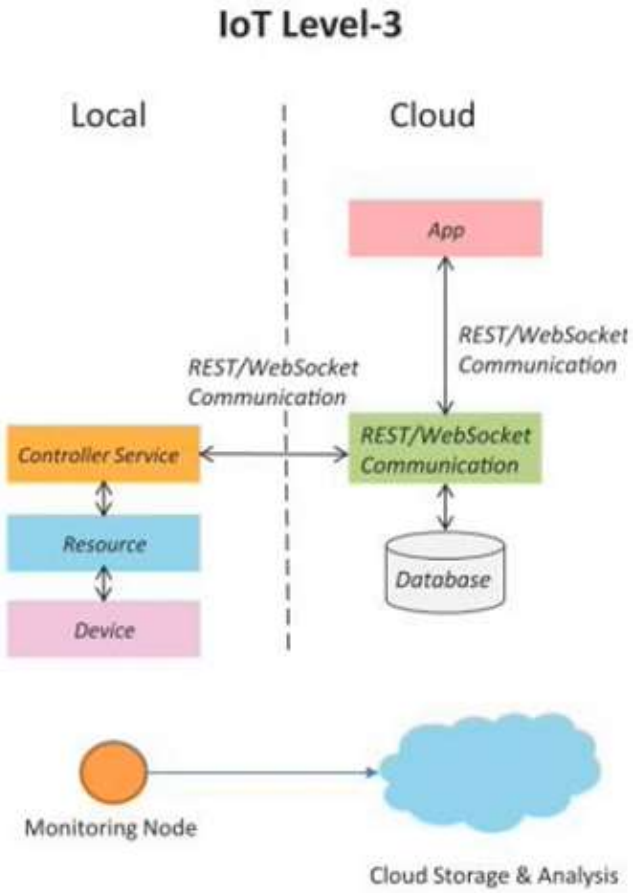
IoT Level 2

- 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**.



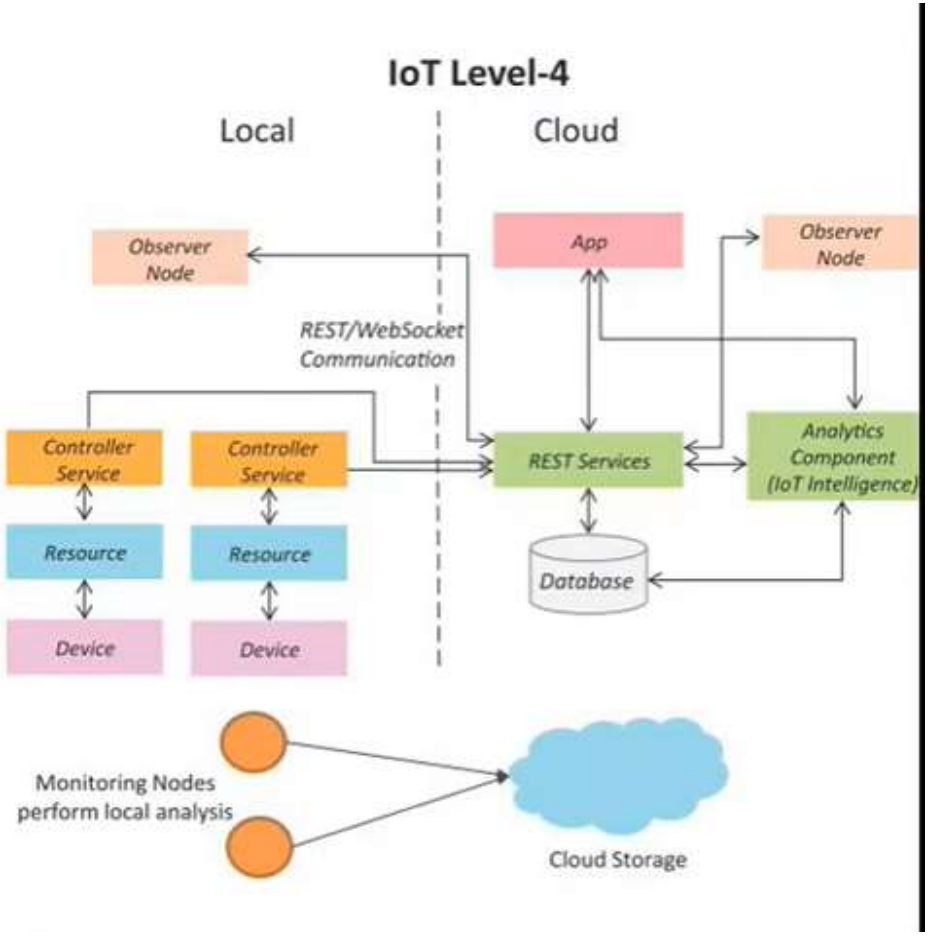
IoT Level 3

- A level-3 IoT system has a single node. Data is stored and analyzed i 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**.



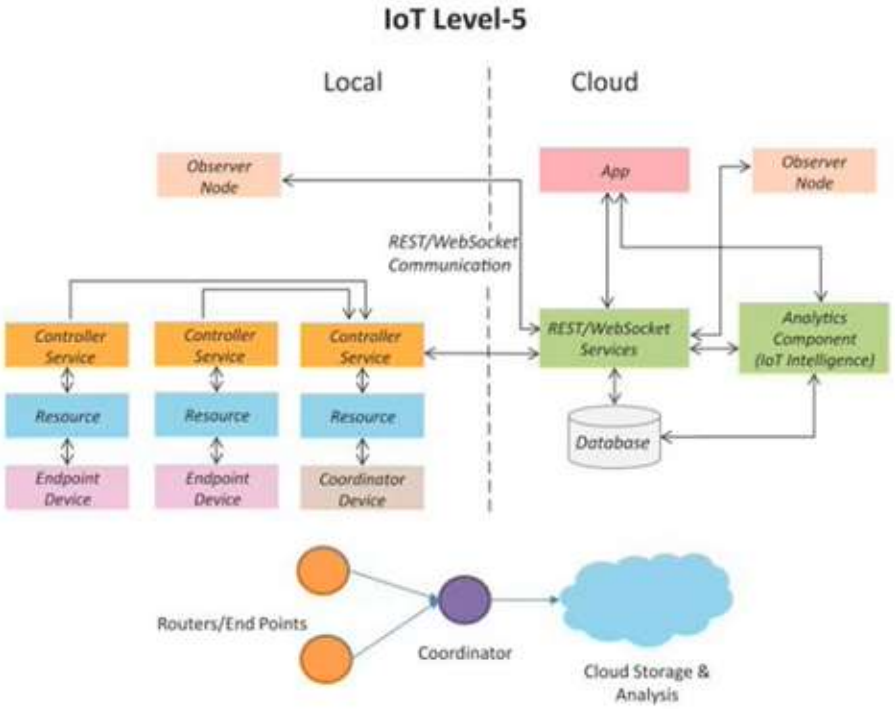
IoT Level 4

- 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

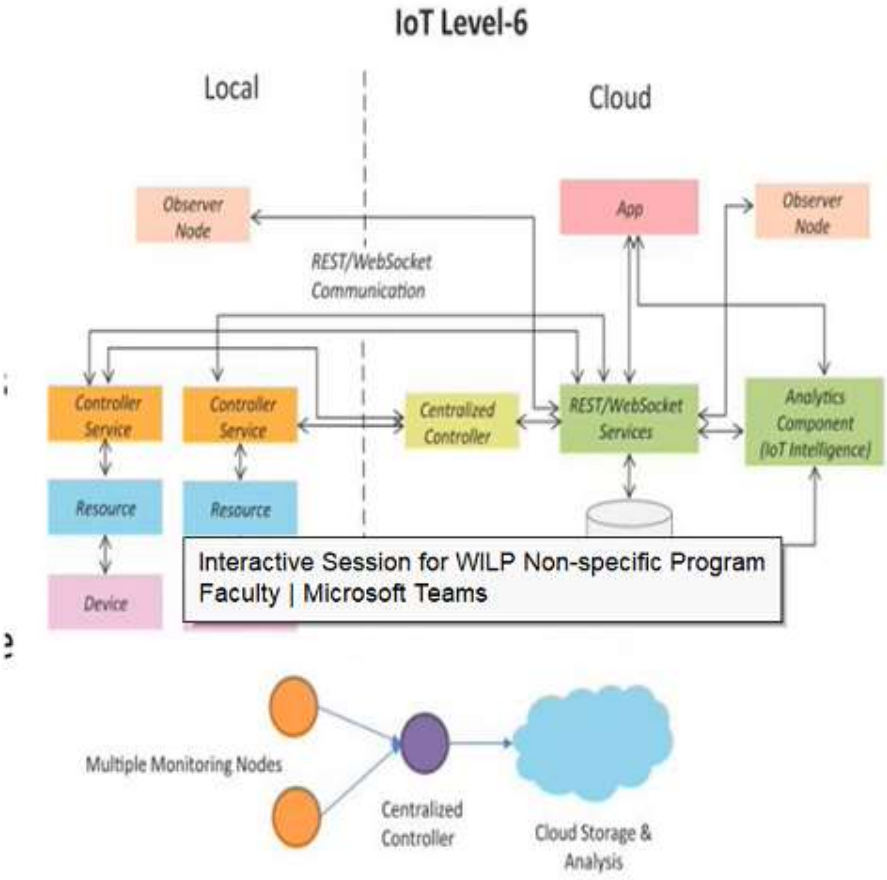
- A level-5 IoT system has multiple end nodes and one coordinator node.
- The end nodes that perform sensing and/or actuation.
- Coordinator node collects data from the end nodes and sends to the cloud.
- Data is stored and analyzed in the cloud and application is cloud-based.
- Level-5 IoT systems are 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 presentation

BITS Pilani
Pilani Campus

Paramananda Barik
CS&IS Department

BITS Pilani
Pilani Campus

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 20ºC reading every second might not be interesting until the sensor reports a 40ºC 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.

Industry	% of total edge use cases	2025 hardware value,[1] $ billion
Travel, transport, and logistics	24	~35–43
Cross-vertical	9	~32–40
Retail	10	~20–28
Media and entertainment	1	~17–25
Public sector and utilities	10	~16–24
Global energy and materials	13	~9–17
Advanced industries	10	~5–13
Healthcare	10	~5–13
Infrastructure	6	~4–11
Chemicals and agriculture	5	~4–11
Banking and insurance	1	~2–7
Consumer	4	~1–5

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

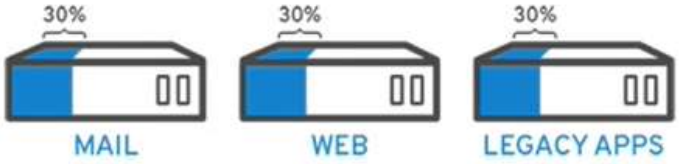
- VMs and Containers
- Software Defined Networking (SDN)
- Content Delivery/Distribution Network (CDN)
- Cloudlets and Micro Data Centers (MDC)

VMs and Containers

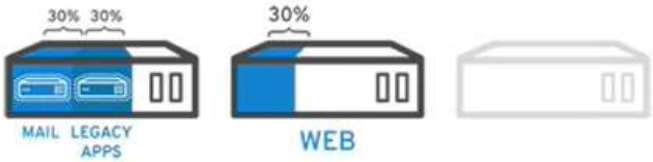
- What is Virtualization?
  - Virtualization is technology that lets users create useful IT services using resources that are **traditionally bound to hardware**.
  - It allows users to use a **physical machine's full capacity** by distributing its capabilities among many users or environments.

  - Virtualization and cloud computing are not interchangeable.
  - Virtualization is software that makes computing environments independent of physical infrastructure.

Before Virtualization

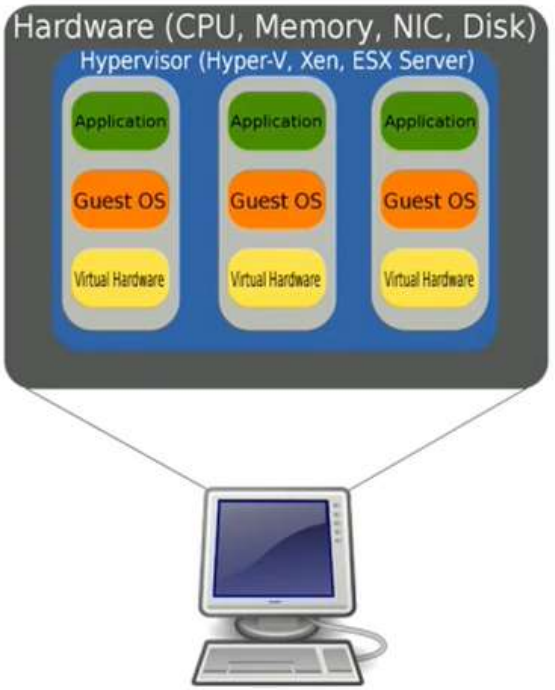


After Virtualization



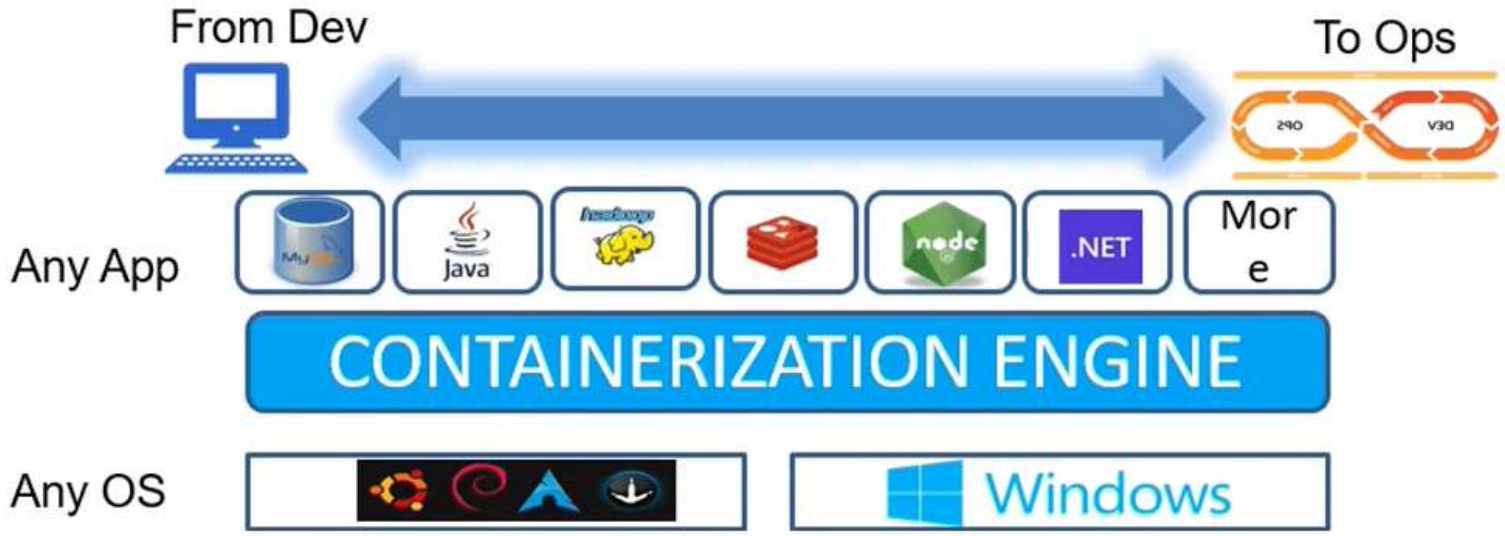
Virtual Machines

- 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.”

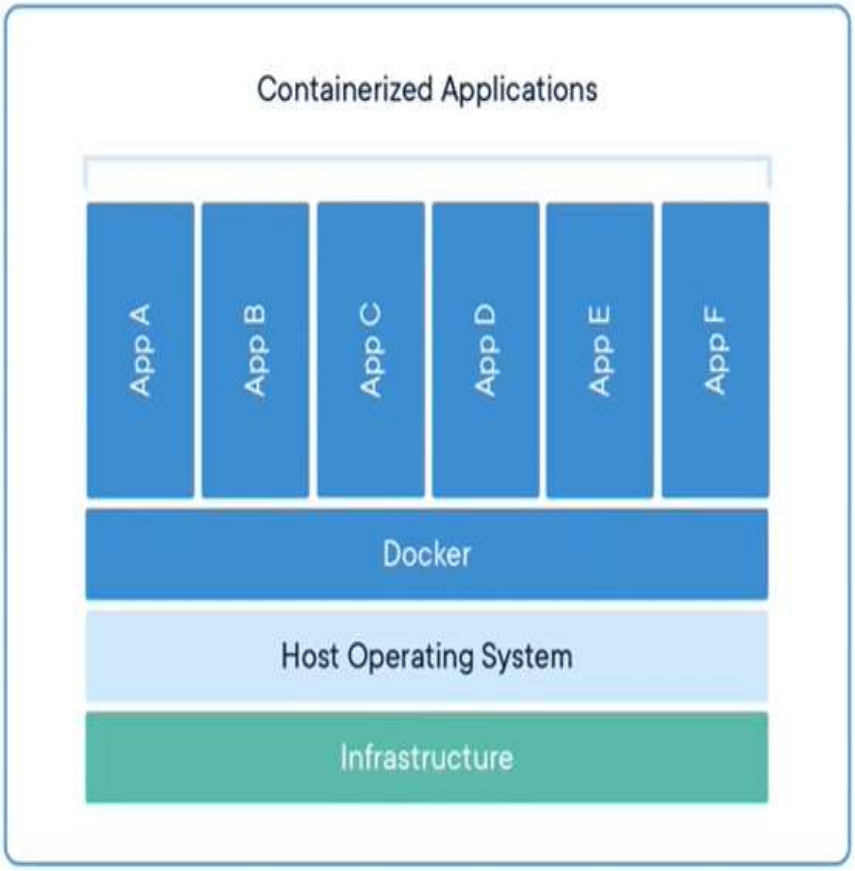


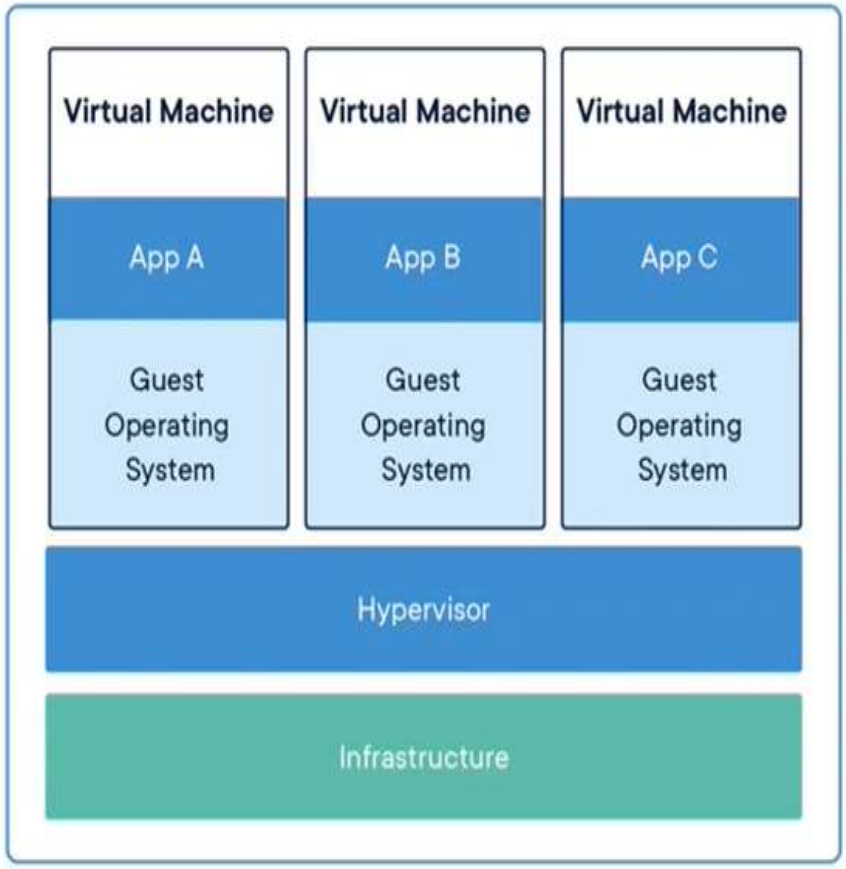
What are Containers?

- 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)

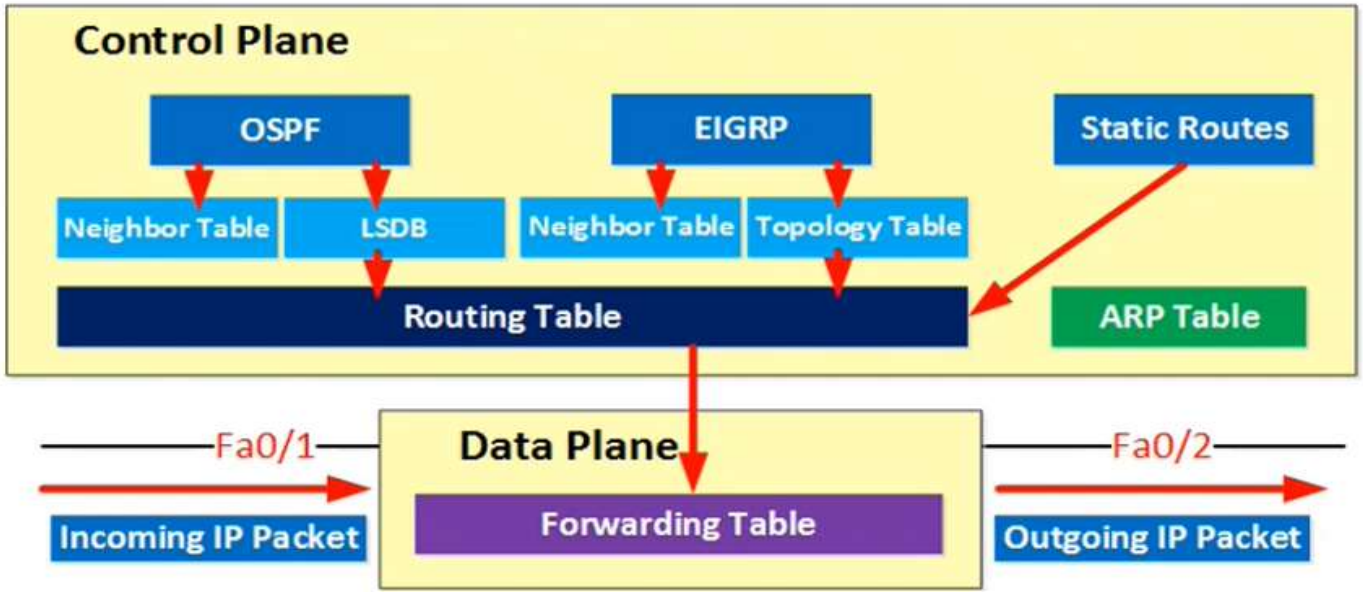
- Traditional Networking
  - Router functions
    - check the destination IP address in the routing table
    - Routing protocols like OSPF, EIGRP or BGP
    - use ARP to figure out the destination MAC address
    - Ethernet frame checksum recalculated

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



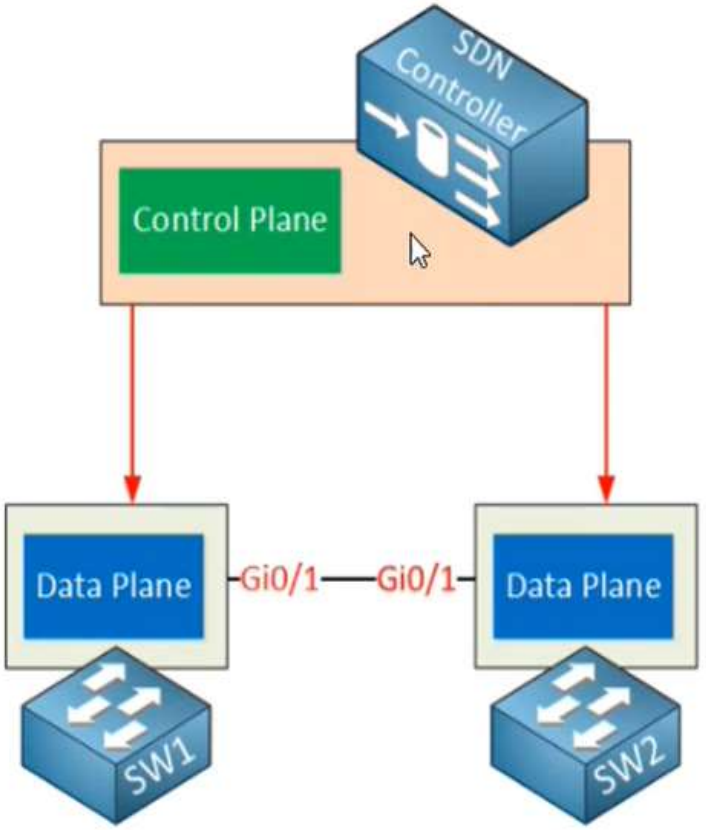
Limitations of traditional networks

- 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.

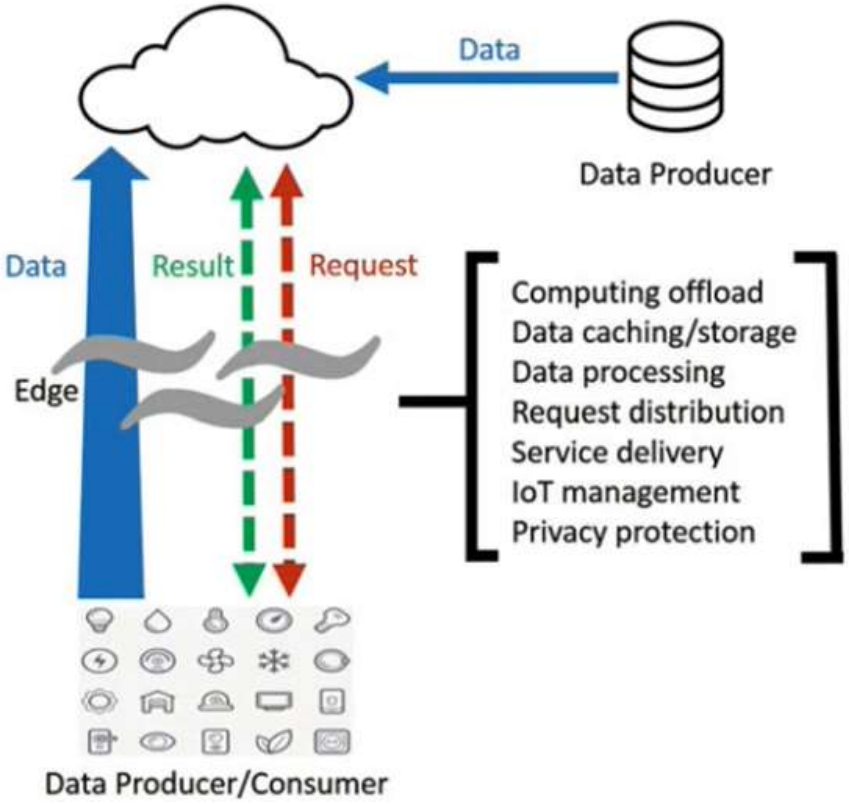
Edge Computing Definition

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 **C**onnection

- large number of connected devices
- heterogeneous Bus connections

Real-Time Services

- 10 ms

Data **O**ptimization

- 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 – 6.40x10^3 MIPS, Xeon – 1.40x10^5 MIPS, A9 – 3.6x10^3 MIPS
- Changes in the storage field
  - Capacity increase – 10000 fold, cost has decreased
  - Speed also has increased

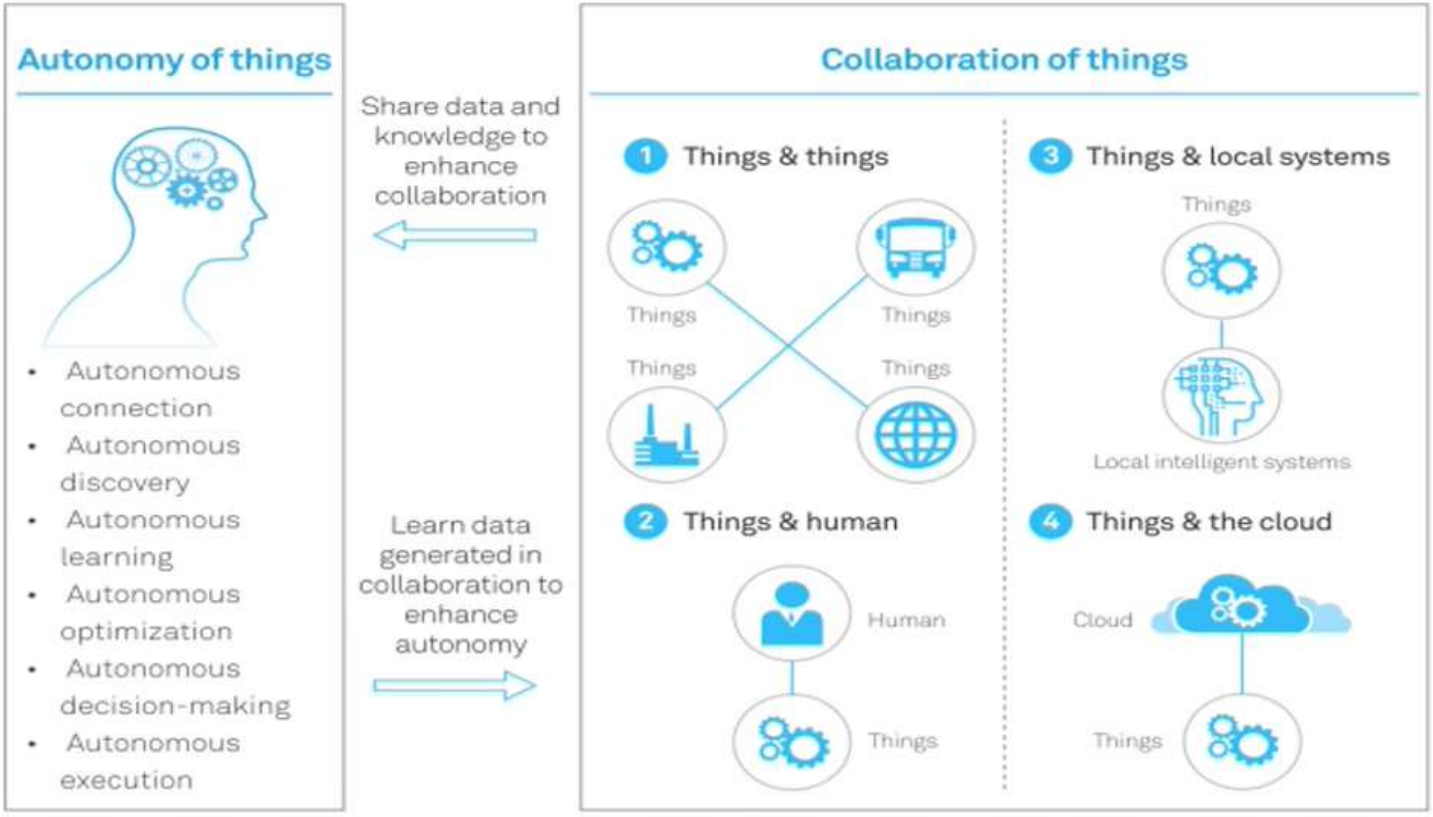
Connection – Physical and digital world

- Changes in the network field
  - Bandwidth increase – 1000 fold, Cost has decreased
- Changes in the computing field
  - Celeron – 6.40x10^3 MIPS, Xeon – 1.40x10^5 MIPS, A9 – 3.6x10^3 MIPS
- Changes in the storage field
  - Capacity increase – 10000 fold, cost has decreased
  - Speed also has increased

Platform – Model driven, intelligent, distributed

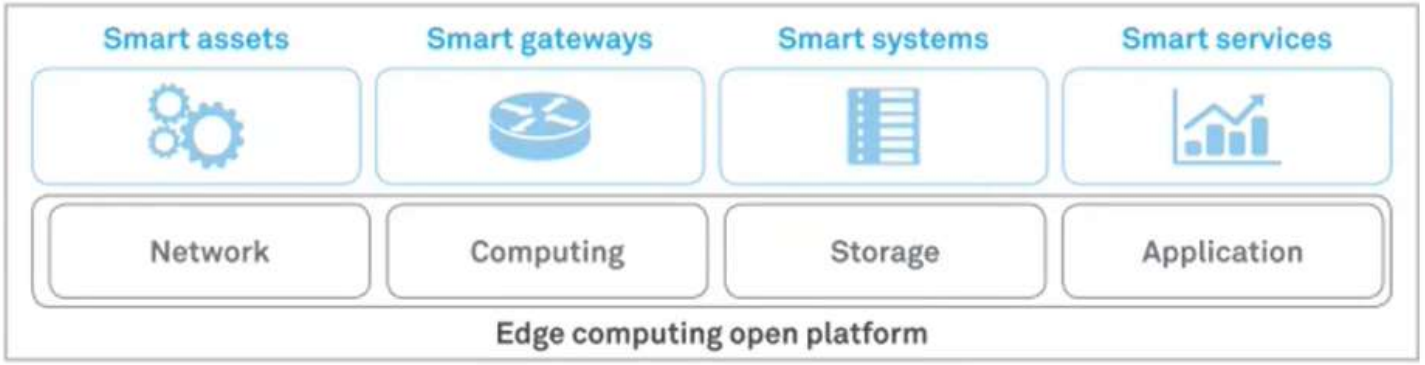
- Autonomy of things

- Collaboration of things



Intelligent distributed architecture

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

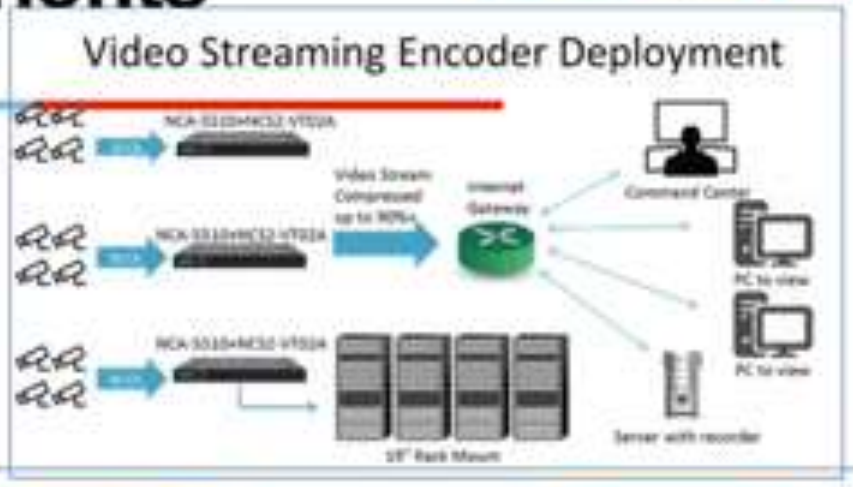


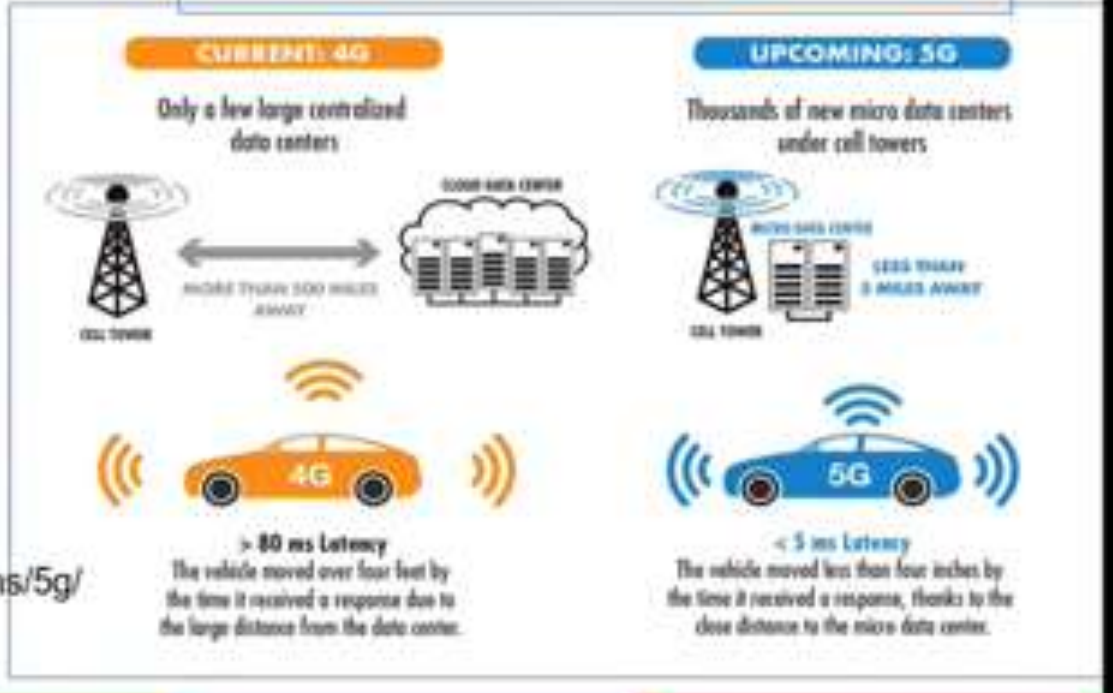
Collaboration of Edge Computing and Cloud Computing

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



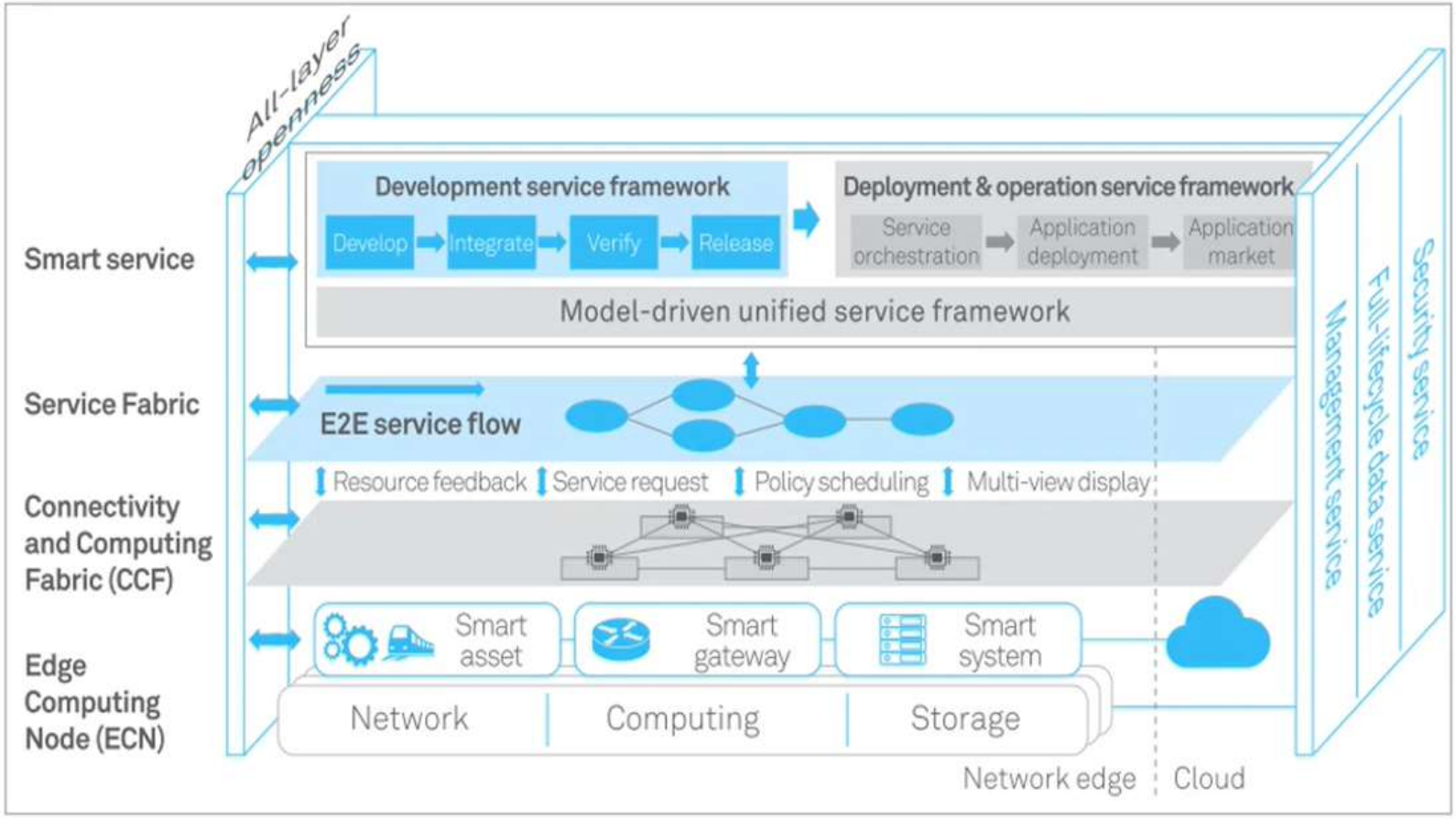


https://www.verizon.com/business/solutions/5g/edge-computing/industry-use-cases-examples/

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

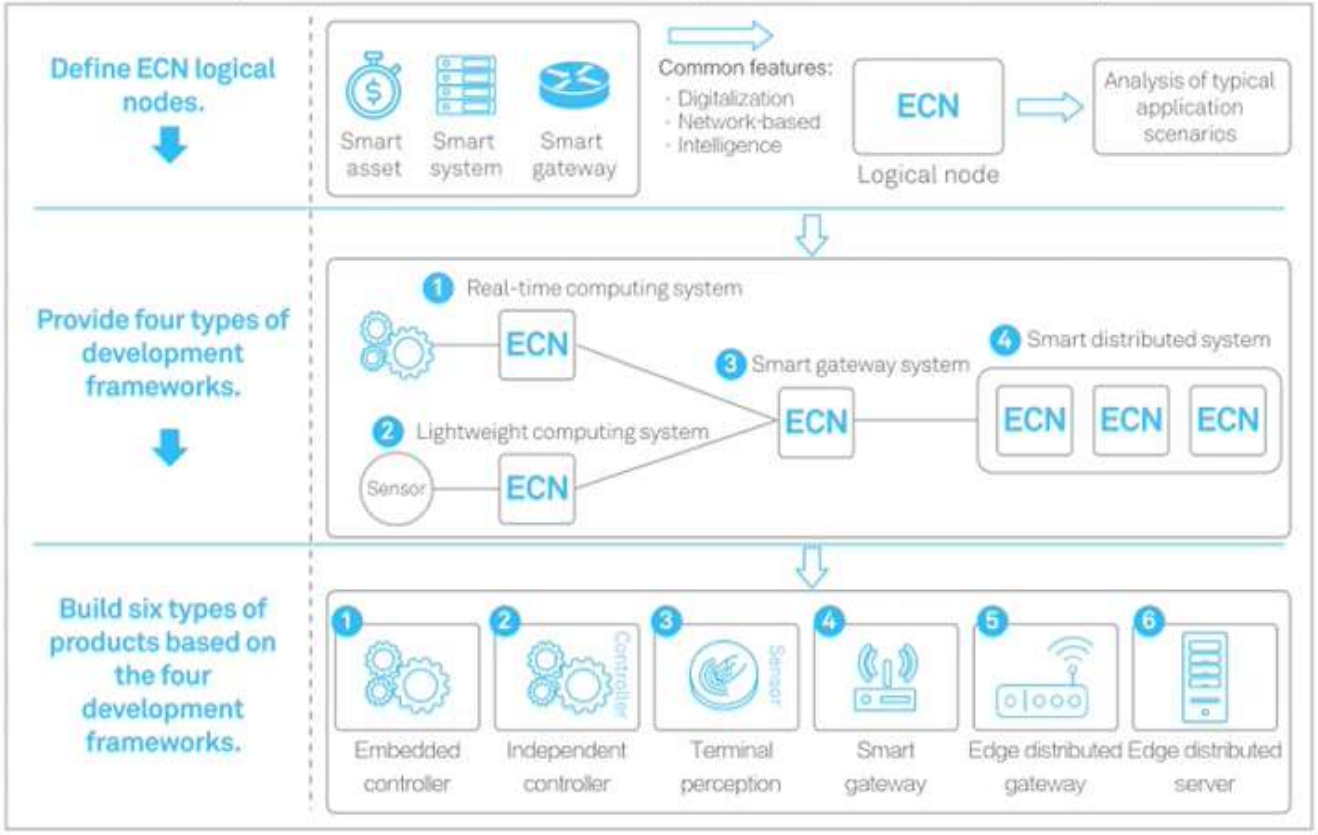
- Coordination Between the Physical and Digital Worlds
- Cross-Industry Collaboration
- Reduced System Heterogeneity and Simplified Cross-Platform Migration
- Effective Support for System Lifecycle Activities

Multi-View Display

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

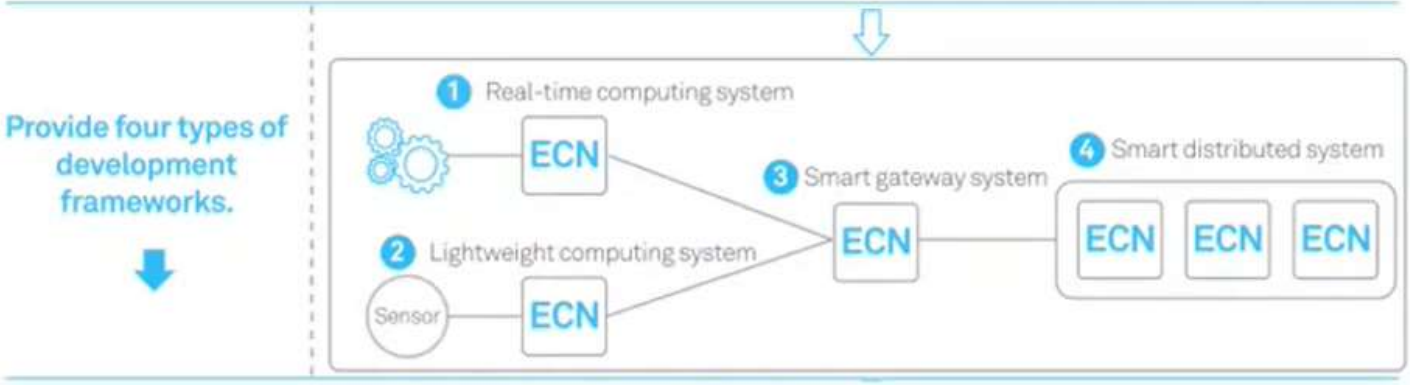
Multi-View Display

ECNs, Development Frameworks, and Product Implementation



Development frameworks of ECNs

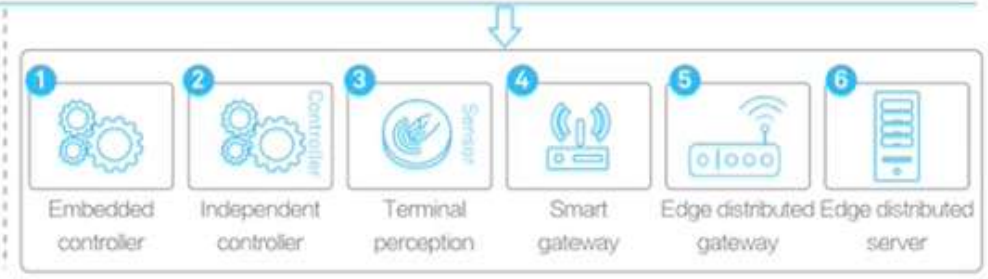
- Real-Time Computing System
- Lightweight Computing System
- Smart Gateway System
- Smart Distributed System



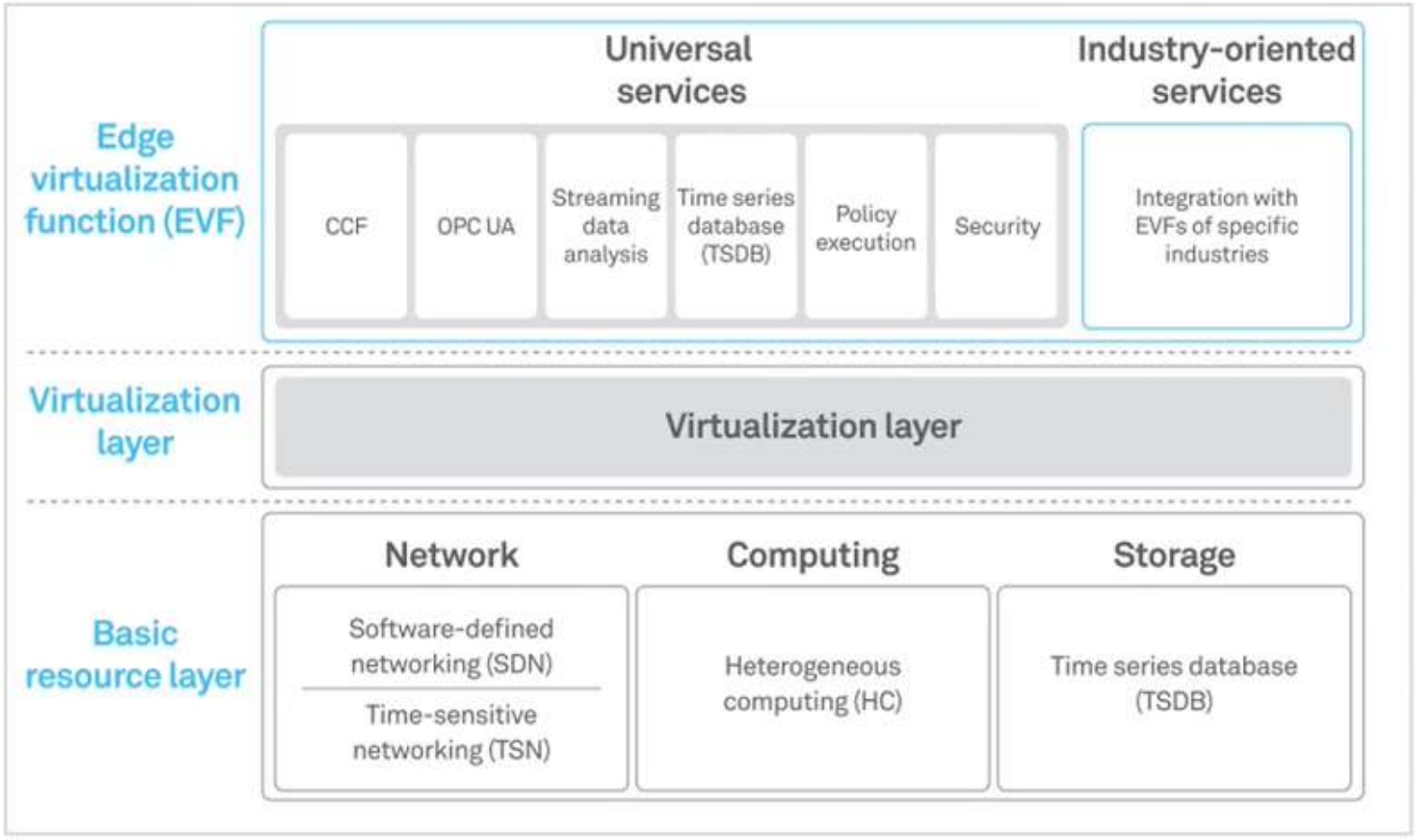
ECN product implementation

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.



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

TSN

- 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

TSN

- Advantages
  - Ensures µs-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

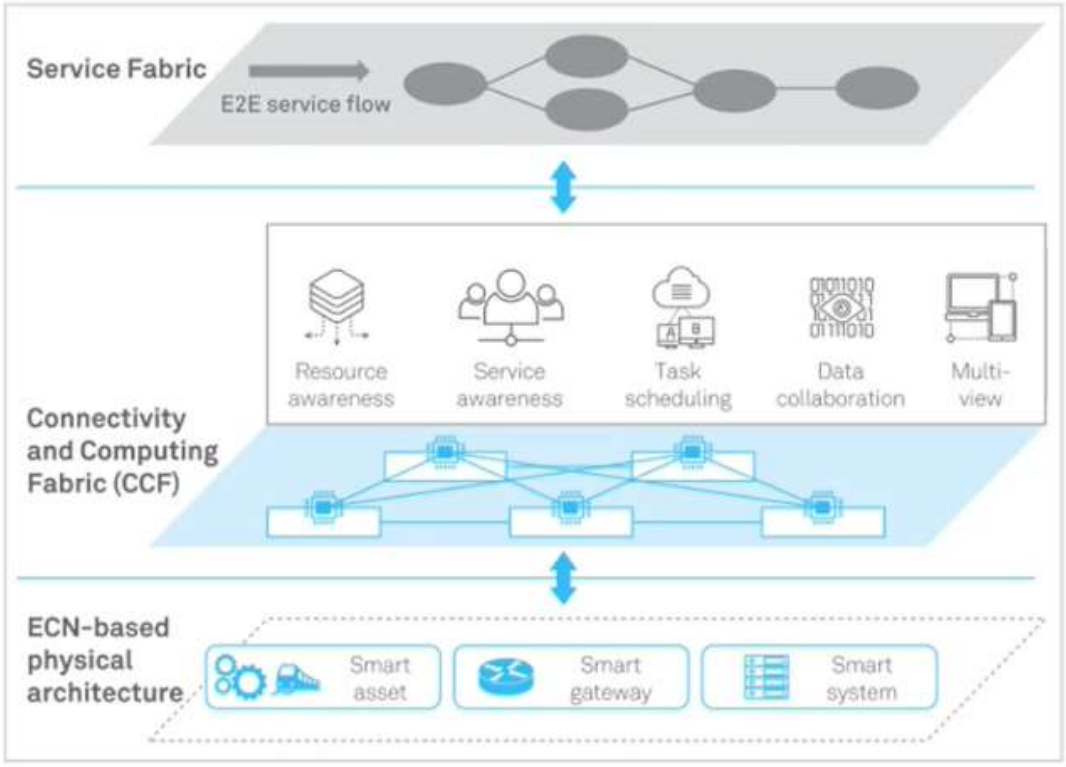
Functional View : Service Fabric

- 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

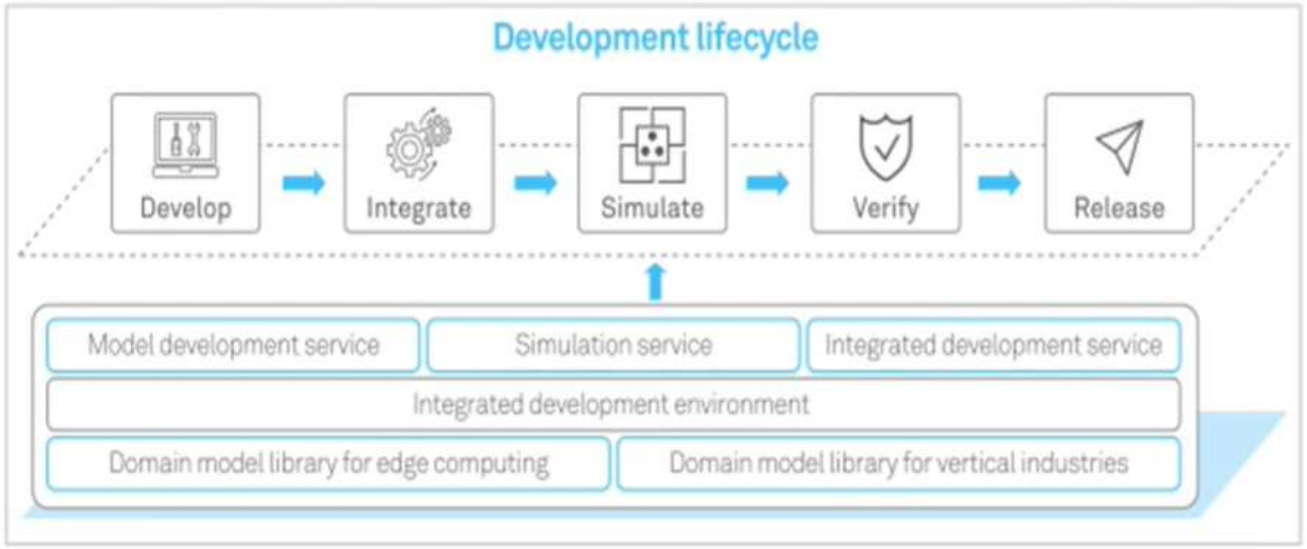
Functional View : CCF

CCF is a virtualized connectivity and computing service layer

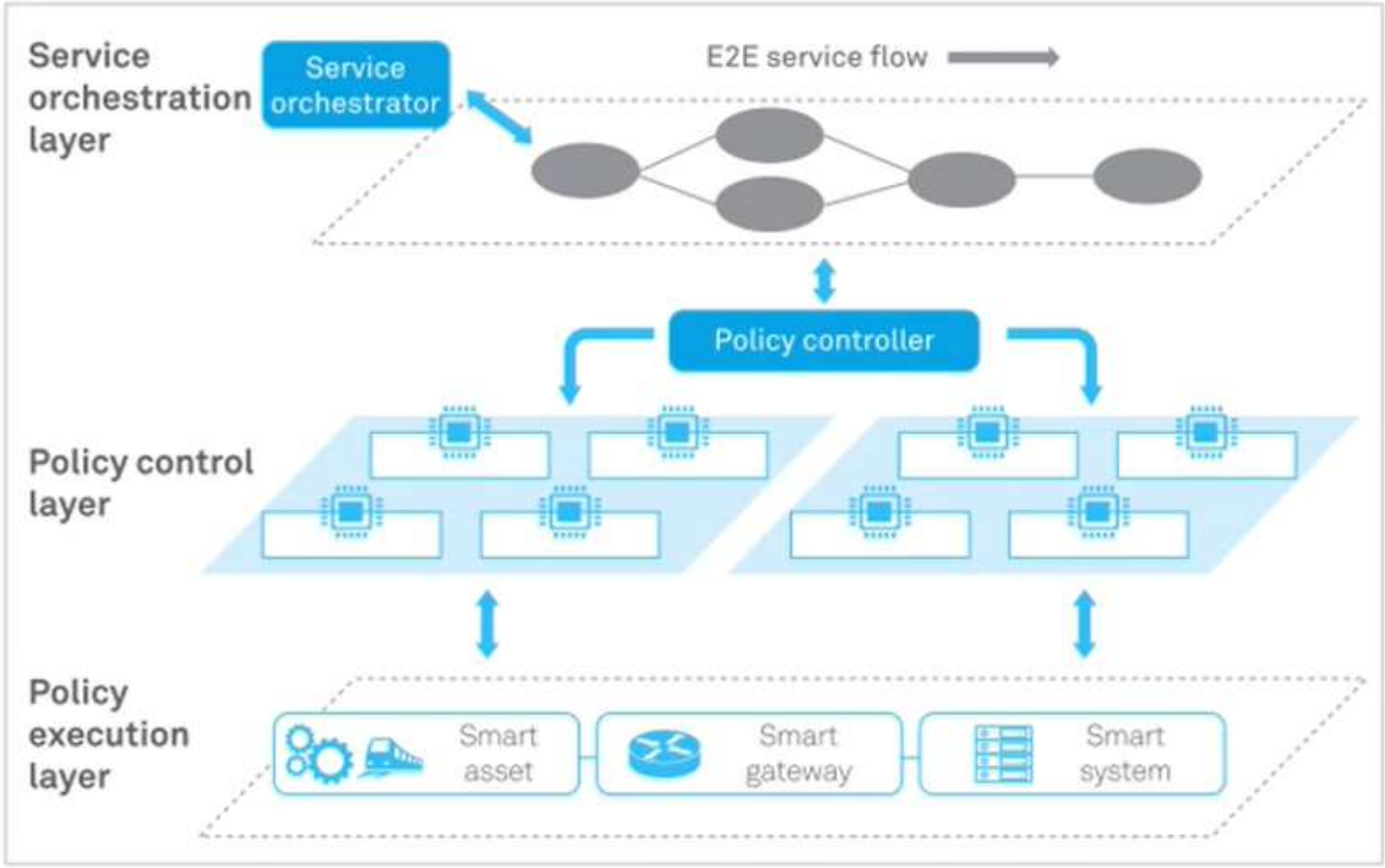


Function View : Development Service Framework

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



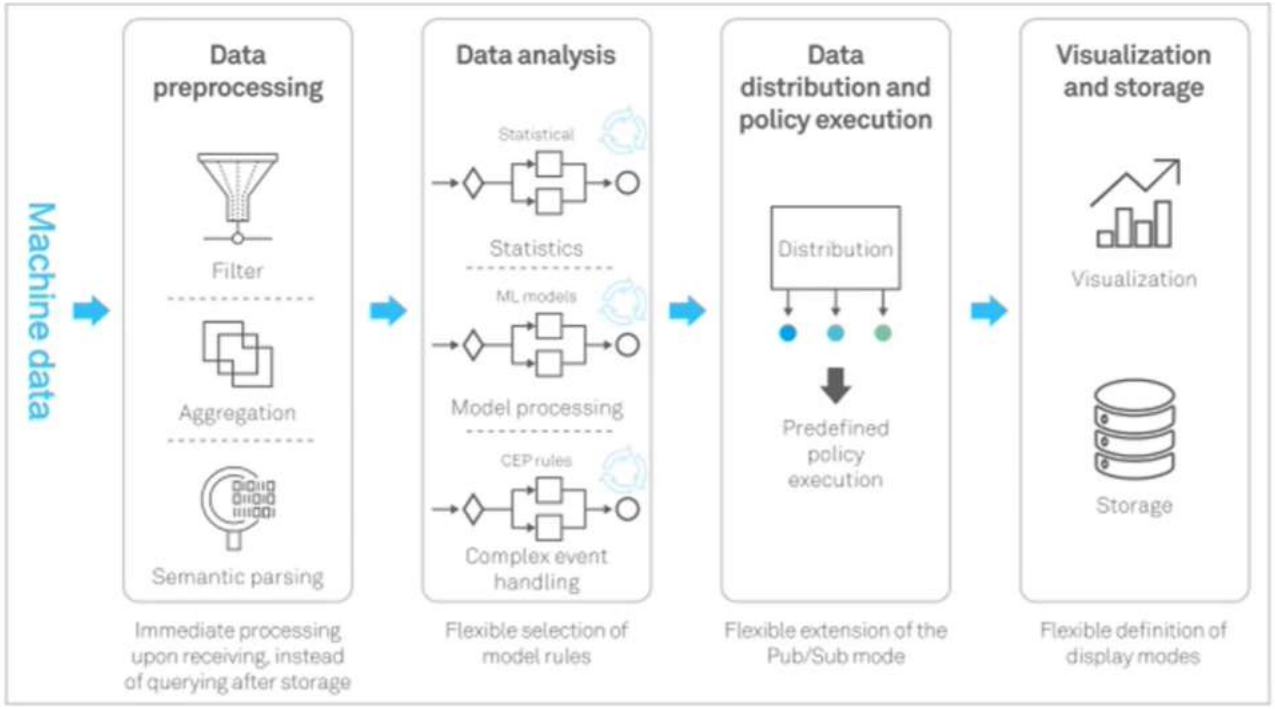
Function View : Development Operation Service Framework



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



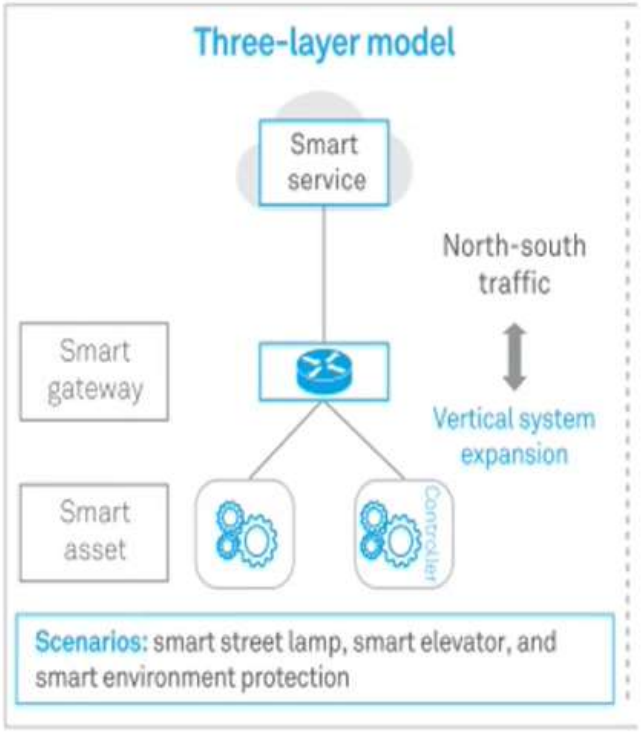
Function View : Security Service

- 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.

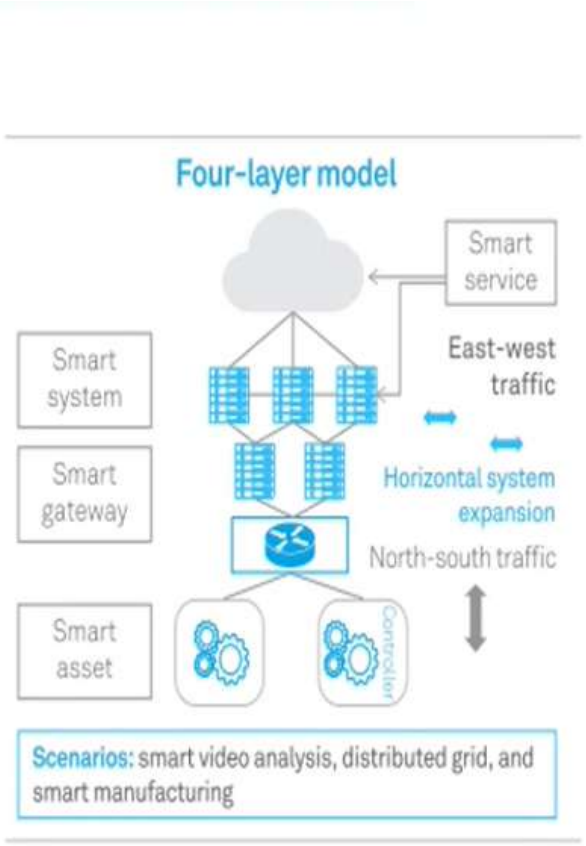
- Scenarios include smart street lamps, smart elevators, and smart environmental protection.



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



EDGE COMPUTING

Edge computing refers to the practice of processing data closer to its source or point of use, rather than sending all the data to a centralized data center or cloud for processing. In traditional cloud computing, data is sent over a network to remote data centers where it's processed, and then the results are sent back to the user. Edge computing, on the other hand, aims to perform computations and analysis at or near the data source, reducing latency, bandwidth usage, and dependence on a centralized location.

The "edge" in edge computing refers to the network's periphery, such as devices, sensors, routers, and other endpoints that are closer to the data source or user. Edge computing can take place in various forms:

Edge Devices: These are smart devices or sensors that collect and process data locally. Examples include smartphones, IoT devices, and industrial sensors.

Edge Servers: These are intermediary computing nodes placed closer to the data source. They can process and filter data before sending relevant information to the cloud. This helps in reducing the amount of data transmitted and the processing load on the central cloud.

Fog Computing: Similar to edge computing, fog computing involves deploying computing resources in between edge devices and the cloud, often within the local network infrastructure. It's particularly useful for applications requiring real-time analytics and low latency.

Mobile Edge Computing: This is focused on processing data at the cellular base station level, often in collaboration with telecom providers. It aims to reduce latency for applications that require real-time processing, such as augmented reality and online gaming.

Edge computing offers several benefits:

Reduced Latency: Processing data closer to its source reduces the time it takes for data to travel to a central server and back, which is crucial for real-time applications like video streaming, IoT, and autonomous vehicles.

Bandwidth Efficiency: Edge computing reduces the amount of data that needs to be sent to the cloud, leading to lower bandwidth requirements and cost savings.

Improved Privacy and Security: Local processing can enhance data privacy by keeping sensitive information on-site rather than transmitting it to a remote server. It also reduces the attack surface for potential security breaches.

Offline Operation: Some edge devices can continue processing data even when disconnected from the central network or cloud, ensuring uninterrupted operation.

Scalability: Distributing computing across edge devices can improve scalability as the network grows, without overburdening a centralized data center.

Regulatory Compliance: Some industries have strict regulations about where data can be processed or stored. Edge computing can help meet these requirements by keeping data within specific geographical boundaries.

However, edge computing also comes with challenges, including managing a distributed infrastructure, ensuring consistency across edge nodes, handling device heterogeneity, and addressing security concerns at the edge.

In summary, edge computing is an approach that decentralizes data processing and analysis, bringing computation closer to where data is generated or consumed, which can greatly benefit applications that require low latency, real-time processing, and efficient data usage.

Why Do We Need Edge Computing?

Edge computing addresses several key challenges and requirements that arise from the increasing complexity and demands of modern technology and applications. Here are some reasons why edge computing has become necessary:

Reduced Latency: Many applications require real-time or near-real-time processing, such as autonomous vehicles, industrial automation, and online gaming. Edge computing minimizes latency by processing data locally, improving the responsiveness of these applications.

Bandwidth Efficiency: Sending large amounts of data to a centralized cloud for processing can strain network bandwidth and lead to high costs. Edge computing reduces the need to transfer large datasets by processing data locally and only sending relevant information to the cloud.

Scalability: As the number of connected devices and sensors grows, centralized cloud infrastructure might struggle to handle the increasing load. Edge computing allows for distributed processing, making it easier to scale by adding more edge devices as needed.

Reliability in Unstable Networks: In scenarios where network connectivity is unreliable, such as remote locations or areas with poor internet coverage, edge computing ensures that local processing can continue even when the connection to the central cloud is lost.

Data Privacy and Compliance: Certain industries, like healthcare and finance, have strict regulations about data privacy and residency. Edge computing enables data to be processed locally, helping to comply with regional data protection laws.

Real-Time Decision Making: Edge computing supports applications that require rapid decision-making based on real-time data analysis, such as predictive maintenance in industrial equipment or processing sensor data in autonomous vehicles.

Security and Privacy: Processing data locally can enhance security by minimizing the exposure of sensitive information to the broader network. It can also help prevent data breaches and unauthorized access.

Cost Efficiency: Offloading some processing tasks to edge devices can reduce the computational load on central cloud servers, resulting in cost savings for cloud usage.

Offline Operations: Edge devices equipped with local processing capabilities can continue to function even when they lose connectivity to the central cloud, ensuring uninterrupted operation in critical situations.

Regulatory and Legal Requirements: In some cases, data must remain within specific geographical boundaries due to legal or regulatory requirements. Edge computing allows organizations to keep data locally and comply with these rules.

AI and Machine Learning at the Edge: Edge devices equipped with AI and machine learning capabilities can make real-time decisions without the need to transmit data to the cloud for analysis, enabling more responsive and intelligent applications.

Redundancy and Load Balancing: Distributing processing across edge devices can provide redundancy and improve fault tolerance in case one node fails. It can also balance the load on the network and prevent bottlenecks.

In essence, edge computing offers a solution to the limitations of traditional cloud computing in scenarios where low latency, real-time processing, privacy, reliability, and efficient data usage are crucial. It complements cloud computing by distributing processing tasks across the network's edge, enabling a more decentralized and responsive architecture for modern applications.

Key Techniques that Enable Edge Computing

Edge computing relies on several key techniques and technologies that enable the processing and analysis of data at or near the edge of the network. Here are some important techniques that play a role in making edge computing possible:

Edge Devices and Sensors: The foundation of edge computing is the proliferation of smart devices, sensors, and IoT devices. These devices collect data from their environment and often have processing capabilities to perform initial data filtering, preprocessing, and basic analysis before sending data further.

Edge Servers and Gateways: These are intermediate nodes placed between edge devices and the central cloud. They aggregate, filter, and process data from multiple edge devices, reducing the amount of data that needs to be transmitted to the cloud. Edge servers and gateways can also handle local storage and perform analytics.

Fog Computing: Fog computing extends edge computing by creating a hierarchical computing infrastructure that includes both edge devices and intermediate fog nodes. Fog nodes can perform more complex processing tasks compared to edge devices, allowing for more sophisticated analysis and decision-making closer to the data source.

Low-Power and High-Performance Processors: Edge devices and servers often use specialized processors that balance low power consumption with the ability to perform tasks efficiently. This is crucial for maintaining the operational lifespan of battery-powered devices while ensuring sufficient processing power.

Distributed Data Storage: Data storage at the edge is essential for caching frequently accessed data, enabling offline operation, and reducing the need for constant communication with the central cloud. Distributed storage solutions optimize data availability and latency.

Data Preprocessing: Edge devices and servers preprocess data before sending it to the central cloud. This includes data filtering, aggregation, transformation, and feature extraction, reducing the volume of data that needs to be transmitted and processed centrally.

Machine Learning and AI: Edge devices equipped with AI and machine learning capabilities can perform real-time analysis and decision-making without the need to send data to the cloud. This is especially useful for applications requiring quick responses and reducing latency.

Containerization and Virtualization: Technologies like Docker and Kubernetes enable the deployment and management of applications in containers or virtualized environments at the edge. This enhances flexibility, scalability, and portability of applications across different edge nodes.

Edge Analytics Platforms: These platforms provide tools and frameworks for developing and deploying edge applications. They often include libraries for data analysis, machine learning, and integration with various edge devices and sensors.

Security and Encryption: Edge computing requires robust security mechanisms to protect data, devices, and communications. Techniques include encryption, secure boot, identity and access management, and intrusion detection systems at the edge.

Real-Time Data Streaming: Technologies for real-time data streaming and processing, such as Apache Kafka and MQTT, enable edge devices and servers to handle streams of data in real-time, allowing for immediate analysis and response.

5G and Low-Latency Networks: The rollout of 5G networks enhances edge computing capabilities by providing higher bandwidth and lower latency, making it feasible to transmit data quickly between edge devices, servers, and the cloud.

These techniques collectively enable edge computing to address challenges related to latency, bandwidth, scalability, privacy, and real-time processing, making it possible to deploy applications that require rapid decision-making, efficient data usage, and improved user experiences.

Edge Computing Definition

Edge computing refers to a decentralized computing paradigm in which data processing and analysis are conducted closer to the data source, often at the edge of the network or near the point of data generation. Unlike traditional cloud computing, where data is sent to centralized data centers for processing, edge computing involves performing computations locally on devices or servers located in proximity to the data origin.

In edge computing, data is processed, filtered, and sometimes analyzed at the "edge" of the network, which can be devices like sensors, smartphones, gateways, or local servers. This approach reduces latency, conserves bandwidth, enhances real-time processing capabilities, and supports applications that require rapid decision-making, low-latency interactions, and efficient data usage.

Edge computing is particularly relevant in scenarios where immediate processing and response are critical, such as in industrial automation, IoT (Internet of Things), autonomous vehicles, remote monitoring, and applications involving real-time analytics. By distributing processing tasks across the network's edge, edge computing complements cloud computing and offers a more decentralized and responsive architecture to meet the demands of modern data-intensive applications.

Edge Computing Benefits

Edge computing offers a range of benefits that make it a valuable approach for various industries and applications. Some of the key benefits of edge computing include:

Reduced Latency: Processing data closer to its source minimizes the time it takes for data to travel to a central cloud and back. This is crucial for applications requiring real-time responses, such as IoT, autonomous vehicles, and industrial automation.

Improved Responsiveness: Edge computing enables quicker decision-making by analyzing data locally, leading to faster responses and actions. This is vital for applications like real-time monitoring and control systems.

Bandwidth Savings: By processing and filtering data at the edge, only relevant information needs to be sent to the central cloud, reducing the amount of data transmitted over the network and saving bandwidth costs.

Enhanced Data Privacy: Sensitive data can be processed locally, reducing the need to transmit it over external networks. This helps maintain data privacy and compliance with regulations in industries like healthcare and finance.

Reliable Operation in Unstable Networks: Edge devices can continue functioning even if network connectivity is lost, ensuring critical operations are maintained during network disruptions.

Offline Operation: Some edge devices have the ability to process data even when disconnected from the central network, allowing them to operate in remote or intermittent connectivity environments.

Scalability: Edge computing allows for distributed processing, making it easier to scale by adding more edge devices as needed, without overburdening a centralized cloud infrastructure.

Real-Time Analytics: Edge devices with local processing capabilities enable real-time data analysis and decision-making, which is essential for applications like predictive maintenance and anomaly detection.

Cost Efficiency: Offloading processing tasks to edge devices can reduce the computational load on central cloud servers, leading to cost savings in terms of cloud usage and network bandwidth.

Redundancy and Resilience: Distributing processing across edge nodes improves fault tolerance and redundancy. If one edge device fails, others can continue processing and maintaining operations.

Optimized Network Traffic: By processing data at the edge, network traffic to centralized data centers is reduced, preventing congestion and improving overall network performance.

AI and Machine Learning at the Edge: Edge devices equipped with AI and machine learning capabilities can make decisions locally, avoiding the latency associated with sending data to the cloud for analysis.

Customized Processing: Edge computing allows for tailored processing at different locations, adapting to the unique requirements of each application and location.

Real-Time Feedback and Interaction: Edge computing enables applications that require real-time feedback and interaction, such as augmented reality, virtual reality, and interactive gaming.

Compliance with Regulatory Requirements: Edge computing helps meet regulatory requirements that mandate data processing or storage within specific geographic boundaries.

In summary, edge computing offers a holistic approach that addresses challenges related to latency, bandwidth, privacy, scalability, and real-time processing. It brings computation closer to data sources, resulting in improved performance, responsiveness, and efficiency for a wide range of applications across various industries.

Edge Computing Systems

Edge computing systems consist of a combination of hardware, software, and network components that enable the implementation and operation of edge computing infrastructure. These systems are designed to process and analyze data at or near the edge of the network, providing benefits such as reduced latency, improved responsiveness, and efficient data usage. Here are the key components of edge computing systems:

Edge Devices: These are the endpoints in the edge computing ecosystem, such as sensors, IoT devices, smartphones, and industrial machinery. Edge devices collect data from the environment and often have processing capabilities to perform initial data filtering and preprocessing.

Edge Servers: These are intermediate computing nodes located closer to the edge devices. Edge servers can perform more complex processing tasks compared to edge devices and help aggregate, filter, and analyze data before transmitting relevant information to the central cloud.

Edge Gateways: Gateways serve as intermediaries between edge devices and the central cloud or data center. They can perform data aggregation, protocol translation, and security functions, facilitating communication between various types of edge devices and the cloud.

Fog Nodes: In fog computing architectures, fog nodes are a layer of intermediate nodes between edge devices and the cloud. They can perform more advanced processing, analytics, and data manipulation compared to edge devices, offering a higher level of computational capacity.

Distributed Data Storage: Edge computing systems include local storage capabilities for caching frequently accessed data, supporting offline operation, and reducing the need for constant communication with the central cloud.

Networking Infrastructure: Reliable and low-latency network connections are essential for edge computing. This includes both wired and wireless networks, with technologies like 5G and Wi-Fi 6 enhancing connectivity and data transmission.

Data Preprocessing and Analytics Tools: Software tools and frameworks are used to preprocess, analyze, and filter data at the edge. This might involve data filtering, aggregation, feature extraction, and even real-time analytics using machine learning algorithms.

Containerization and Virtualization: Containerization technologies like Docker and virtualization platforms help manage and deploy applications in isolated environments at the edge. This facilitates application portability and ensures consistent operation across different edge nodes.

Security Mechanisms: Edge computing systems require robust security measures to protect data, devices, and communication. This includes encryption, access control, secure boot, and intrusion detection systems.

Management and Orchestration: Tools for managing, monitoring, and orchestrating edge devices and servers are crucial for maintaining system performance, diagnosing issues, and ensuring seamless operation.

AI and Machine Learning Capabilities: Edge computing systems may incorporate AI and machine learning models to enable real-time decision-making and analysis at the edge, without the need to send data to the central cloud.

Edge Analytics Platforms: These platforms provide developers with the tools and frameworks needed to create and deploy edge applications, often including libraries for data analysis, machine learning, and integration with edge devices.

Protocol Support: Edge computing systems need to support various communication protocols to interact with different types of edge devices, servers, and cloud services.

Overall, edge computing systems are designed to bring the benefits of localized processing, reduced latency, and efficient data usage to a wide range of applications and industries, catering to the growing demand for real-time responsiveness and intelligent decision-making.

Challenges and Opportunities in Edge Computing

Edge computing presents both challenges and opportunities, reflecting its potential to transform various industries and applications. Here are some of the key challenges and opportunities associated with edge computing:

Challenges:

Resource Constraints: Many edge devices, such as sensors and IoT devices, have limited computational power, memory, and energy resources, which can make running complex applications challenging.

Heterogeneity: Edge computing environments often consist of diverse hardware and software platforms, leading to compatibility and interoperability challenges when deploying applications across different edge nodes.

Data Management: Distributing data across multiple edge devices and servers can lead to data inconsistency, synchronization issues, and difficulties in maintaining data integrity.

Security Concerns: Securing edge devices and nodes can be complex, as they are often physically distributed and have varying levels of security measures. This opens up potential vulnerabilities in the network.

Network Congestion: Processing data at the edge can lead to increased local network traffic, potentially causing congestion and affecting network performance.

Lifecycle Management: Managing and updating software, firmware, and configurations across distributed edge devices can be challenging, requiring effective remote management tools.

Data Privacy: Edge computing raises privacy concerns, especially when processing sensitive data locally. Ensuring compliance with data protection regulations becomes critical.

Scalability: Scaling edge computing systems can be challenging due to the distributed nature of the architecture. Ensuring seamless scaling while maintaining performance and reliability is a significant challenge.

Opportunities:

Latency Reduction: Edge computing significantly reduces latency, enabling real-time processing and decision-making for applications like IoT, gaming, and autonomous vehicles.

Improved Responsiveness: Edge computing enhances the responsiveness of applications by reducing the round-trip time to the central cloud, enabling quicker actions based on data analysis.

Efficient Data Usage: Edge computing minimizes the need to transmit large amounts of data to the cloud, leading to reduced bandwidth consumption and cost savings.

Real-Time Analytics: Edge computing allows for real-time data analysis and insights at the point of data generation, enabling applications that require immediate insights.

Offline Operation: Edge devices with local processing capabilities can operate offline, ensuring continuous functionality even in the absence of network connectivity.

AI and Machine Learning at the Edge: Edge computing enables AI and machine learning applications to operate in real time without relying on cloud processing, opening up new possibilities for intelligent applications.

Customization: Edge computing allows for tailored processing based on specific edge nodes, enabling customized applications that cater to local requirements.

Redundancy and Resilience: Distributed edge architectures offer redundancy and improved fault tolerance, ensuring continuous operation even if some edge nodes fail.

Industry-Specific Solutions: Edge computing can be tailored to meet the unique requirements of different industries, such as healthcare, manufacturing, agriculture, and transportation.

Edge Cloud Collaboration: Opportunities exist for collaboration between edge computing and cloud computing, creating hybrid architectures that optimize the benefits of both paradigms.

Energy Efficiency: Localized processing reduces the need to transmit data over long distances, potentially leading to energy savings in data transmission.

In conclusion, while edge computing introduces challenges related to resource constraints, security, and management complexity, it also offers opportunities for latency reduction, improved responsiveness, and real-time analytics. Its potential to transform industries, enable new applications, and optimize data processing makes it a key technology for the future of computing.

Programmability

Programmability in the context of edge computing refers to the ability to develop, deploy, and manage applications on edge devices and servers. It involves providing developers with the tools, frameworks, and interfaces necessary to create software solutions that leverage the capabilities of edge computing infrastructure. Programmability is crucial for enabling a wide range of applications that benefit from localized processing, real-time analytics, and efficient data usage. Here's how programmability plays a role in edge computing:

Application Development: Programmability allows developers to create applications that run on edge devices and servers. These applications can range from simple data preprocessing tasks to complex real-time analytics and machine learning algorithms.

APIs and Frameworks: Edge computing platforms provide programming interfaces (APIs) and software development frameworks that abstract the underlying hardware and communication

protocols. This makes it easier for developers to write applications without needing to be experts in the intricacies of the edge infrastructure.

Language Support: Programmability in edge computing includes support for various programming languages commonly used in software development, such as Python, Java, C++, and more. This flexibility allows developers to work with languages they are already familiar with.

Edge Analytics: Developers can leverage programmability to implement real-time analytics at the edge. They can write code that analyzes incoming data streams, detects patterns, and triggers actions based on predefined rules or machine learning models.

Customization: Edge computing systems offer programmability to enable customization. Developers can tailor applications to the specific requirements of their use cases, optimizing performance and efficiency.

Machine Learning at the Edge: Programmability enables the deployment of machine learning models on edge devices, allowing them to make real-time decisions without relying on cloud resources.

Orchestration and Management: Programmability includes tools for orchestrating and managing applications on edge devices. This involves tasks such as deploying, updating, and monitoring applications remotely.

Containerization and Virtualization: Technologies like containerization (e.g., Docker) and virtualization allow developers to package applications and their dependencies in isolated environments. This enhances portability and simplifies deployment across different edge nodes.

Integration with Cloud: Programmability allows applications to seamlessly interact with cloud services. Developers can build hybrid applications that utilize both edge and cloud resources for optimal performance and scalability.

Debugging and Testing: Programmability tools include debugging and testing capabilities that help developers identify and resolve issues in their edge applications.

Security: Edge computing programmability also involves implementing security measures, such as access controls and encryption, to protect applications and data at the edge.

Edge Analytics Platforms: These platforms provide a higher level of programmability, offering pre-built libraries and tools for data analysis, machine learning, and integration with edge devices.

Overall, programmability is a key enabler of edge computing's potential, empowering developers to create innovative applications that leverage the benefits of localized processing, reduced latency, and efficient data usage at the edge of the network.

2 Naming

In the context of edge computing, "naming" refers to the process of assigning unique identifiers or names to edge devices, services, applications, and resources within an edge computing environment. Naming plays a critical role in managing and identifying various components within the edge infrastructure. Effective naming conventions and systems help ensure proper communication, organization, and coordination in edge computing systems. Here's how naming is relevant in the context of edge computing:

Device Identification: Each edge device needs a unique name or identifier to distinguish it from other devices within the network. This enables accurate communication and data exchange between devices.

Service Discovery: Naming allows services and applications running on edge devices to be discovered by other devices or clients. This is crucial for establishing connections and interactions in a distributed edge environment.

Resource Allocation: Naming helps in allocating resources and managing resource utilization across edge devices. For example, naming conventions can be used to identify available processing capacity or storage on specific devices.

Routing and Communication: In edge networks, naming is used to route data and communication requests to the appropriate devices or services. Devices with specific names can be targeted for data transmission or remote commands.

Configuration and Management: Naming simplifies the configuration and management of edge devices and services. Administrators can easily identify and configure devices based on their names.

Orchestration: Naming conventions play a role in orchestrating applications and services across distributed edge nodes. Applications can be deployed and coordinated based on their assigned names.

Naming Services: Some edge computing platforms provide naming services that centrally manage and resolve names to corresponding network addresses. This simplifies the process of name resolution and ensures consistency.

Data Organization: Naming is used to organize and categorize data generated by edge devices. Proper naming conventions aid in efficient data storage, retrieval, and analysis.

Security and Access Control: Naming is linked to security and access control mechanisms. Names can be associated with access privileges, ensuring that only authorized devices or users can interact with specific resources.

Debugging and Troubleshooting: Clearly named devices and components simplify debugging and troubleshooting processes. Issues can be identified and resolved more easily when devices have meaningful names.

Scalability: Naming systems should be scalable to handle the growing number of edge devices and services as the network expands.

Naming Conventions: Establishing consistent naming conventions ensures that names are meaningful, standardized, and easy to understand. This improves overall system organization and communication.

In summary, naming is a fundamental aspect of edge computing that aids in identifying, discovering, managing, and coordinating various components within the edge infrastructure. A well-designed naming system enhances the efficiency, reliability, and scalability of edge computing systems.

3 Data Abstraction

Data abstraction in the context of edge computing refers to the process of simplifying complex data representations, structures, and interactions to provide a higher-level, user-friendly interface for working with data. It involves hiding the underlying complexities of data management, storage, and processing, allowing users and applications to interact with data in a more intuitive and efficient

manner. Data abstraction is crucial in edge computing to enable easier development, deployment, and management of applications while optimizing resource usage. Here's how data abstraction is relevant in edge computing:

Simplified Interaction: Data abstraction provides a simplified view of data, making it easier for developers and users to interact with edge devices, services, and applications without needing to understand the intricacies of the underlying hardware and protocols.

Application Development: Abstraction allows developers to work with higher-level data models and interfaces, enabling them to focus on application logic rather than low-level data management details.

Data Heterogeneity: Edge computing environments often consist of diverse edge devices and sensors with varying data formats. Data abstraction can unify these diverse data sources into a consistent format, making it easier to process and analyze the data.

Resource Optimization: Abstraction enables more efficient utilization of edge device resources by abstracting hardware-specific operations. This allows applications to adapt to different devices without requiring significant modifications.

Standardization: Data abstraction promotes the use of standardized data formats and APIs, making it easier to integrate various edge devices and services into a cohesive system.

Data Transformation: Abstraction allows data to be transformed and adapted as it moves between different layers of the edge computing architecture, facilitating interoperability.

Remote Management: Data abstraction simplifies remote management and monitoring of edge devices. Operators can interact with devices using higher-level commands without needing to understand device-specific protocols.

Efficient Data Usage: Abstraction can optimize data transmission by providing summarized or aggregated views of data. This reduces the amount of data transferred over the network, conserving bandwidth.

Security and Privacy: Abstraction can help manage security and privacy concerns by presenting only relevant data to users and applications while keeping sensitive information hidden.

Data Analytics: Abstraction can facilitate data analytics by providing aggregated, preprocessed data to analytics applications, enabling faster and more efficient analysis.

User Experience: Abstraction enhances the user experience by presenting data in a way that is meaningful and relevant to the user's needs, without overwhelming them with technical details.

Interoperability: Data abstraction can bridge the gap between different edge devices, protocols, and standards, facilitating communication and interoperability in heterogeneous edge environments.

Scalability: Abstraction allows applications to scale across different edge devices without requiring extensive modifications, promoting flexibility and ease of deployment.

Overall, data abstraction in edge computing contributes to simplifying development, improving resource efficiency, and providing a consistent and user-friendly way to interact with data. It aligns with the goal of making edge computing more accessible and effective for a wide range of applications and industries.

Service Management

Service management in the context of edge computing refers to the process of planning, deploying, monitoring, optimizing, and maintaining services and applications that run on edge devices, servers, and other components within an edge computing infrastructure. Effective service management ensures that edge computing systems operate efficiently, deliver the desired performance, and meet the requirements of various applications and industries. Here's how service management is relevant in edge computing:

Service Deployment: Service management involves deploying applications and services to edge devices, gateways, and servers. This includes configuring software, allocating resources, and ensuring proper integration with the edge environment.

Resource Allocation: Service management optimizes the allocation of computational resources such as processing power, memory, and storage across different edge devices based on the requirements of applications.

Scalability: Effective service management allows applications to scale seamlessly across multiple edge devices as demand fluctuates, ensuring consistent performance and responsiveness.

Monitoring and Health Checks: Service management involves continuous monitoring of edge devices and applications to detect issues, performance bottlenecks, and potential failures. Health checks help ensure the reliability of services.

Load Balancing: Service management can include load balancing mechanisms to distribute workloads across multiple edge devices, preventing overburdening of specific devices.

Fault Tolerance and Redundancy: Service management strategies can implement fault tolerance mechanisms to ensure that services remain operational even if individual edge devices or components fail.

Configuration Management: Service management involves maintaining consistent configurations across edge devices, ensuring that applications run as expected and reducing the risk of misconfigurations.

Remote Management: Service management tools enable administrators to remotely monitor and control edge devices, update software, and apply patches without needing physical access.

Service Discovery: Service management includes mechanisms for discovering available services and applications in the edge environment, enabling seamless communication and interaction.

Automated Orchestration: Service management systems can automate the orchestration of applications and services, coordinating their deployment and interactions across multiple edge devices.

Security Management: Service management incorporates security measures such as access controls, authentication, encryption, and intrusion detection to protect services and data at the edge.

Analytics and Reporting: Service management tools can provide insights into service performance, usage patterns, and resource utilization through analytics and reporting features.

Version Control and Updates: Service management ensures that applications and services are up to date with the latest versions, patches, and security fixes.

User Experience Optimization: Effective service management contributes to optimizing user experiences by maintaining consistent and responsive services.

Compliance and Governance: Service management strategies consider regulatory requirements and industry standards to ensure that services adhere to relevant rules and regulations.

Service Decommissioning: As services evolve or become obsolete, service management involves decommissioning or migrating services to maintain a streamlined and efficient edge computing environment.

In summary, service management in edge computing is crucial for maintaining reliable, scalable, and efficient operations. It encompasses various activities, tools, and strategies aimed at ensuring the optimal deployment, performance, and maintenance of applications and services within the edge environment.

Privacy and Security

Privacy and security are critical considerations in the design, implementation, and operation of edge computing systems. With data being processed and stored closer to the source at the edge of the network, it becomes essential to address potential vulnerabilities and protect sensitive information. Here's how privacy and security are relevant in the context of edge computing:

Privacy:

Local Data Processing: Edge computing allows data to be processed locally, reducing the need to transmit sensitive data to centralized cloud servers. This minimizes the exposure of sensitive information during transmission.

Data Minimization: Edge computing encourages the collection and processing of only the necessary data, reducing the risk of collecting and storing excessive or irrelevant personal information.

Data Anonymization and Aggregation: Techniques like data anonymization and aggregation can be employed to ensure that individual user identities are protected while still enabling meaningful analysis.

User Consent and Transparency: Clear communication with users about data collection, processing, and usage is essential. Users should be informed and given control over how their data is used.

Geographical Compliance: Edge computing can enable compliance with data protection regulations that require certain data to be processed within specific geographic boundaries.

Data Encryption: Encrypting data during transmission and storage at the edge ensures that even if data is intercepted, it remains unreadable without the appropriate decryption keys.

Security:

Device Security: Edge devices should be secured against unauthorized access and attacks. This involves implementing secure boot processes, access controls, and regular security updates.

Authentication and Authorization: Strong authentication mechanisms and role-based access controls prevent unauthorized users from accessing sensitive data or controlling edge devices.

Network Security: Edge networks need robust security measures to protect data in transit. This includes encryption, firewalls, intrusion detection systems, and other network security tools.

Secure Communication: Communication between edge devices, gateways, servers, and the cloud should be encrypted to prevent data interception and tampering.

Intrusion Detection: Intrusion detection and prevention systems can monitor edge devices and network traffic for signs of unauthorized access or malicious activities.

Security Updates: Regular security updates and patches should be applied to edge devices and software to address vulnerabilities and protect against emerging threats.

Container Security: If containers or virtualization are used, securing containerized applications and ensuring isolation between them is crucial to prevent breaches.

Edge Analytics Security: Implementing security measures in edge analytics processes prevents unauthorized access to processed data and ensures the integrity of analysis results.

Security Auditing: Regular security audits and assessments of edge computing systems help identify vulnerabilities and ensure compliance with security standards.

Incident Response: Having a well-defined incident response plan in place helps address security breaches effectively and minimize their impact.

Secure Updates: Ensuring that software updates and patches are obtained from trusted sources and are securely deployed prevents the exploitation of vulnerabilities.

Balancing privacy and security in edge computing involves a holistic approach that includes both technical measures and policy considerations. It's crucial to maintain the trust of users and stakeholders by implementing comprehensive measures to protect data and systems in the evolving landscape of edge computing.

Application Distribution

Application distribution in the context of edge computing refers to the process of deploying, installing, and managing software applications across various edge devices, servers, and nodes within an edge computing infrastructure. Effective application distribution ensures that applications are available and operational on the appropriate edge nodes, enabling them to provide the intended functionality while optimizing resource usage and performance. Here's how application distribution is relevant in the context of edge computing:

Edge Application Deployment: Application distribution involves deploying software applications to edge devices, gateways, and servers. This includes transferring application binaries, libraries, and configuration files to the target devices.

Resource Allocation: Applications need to be distributed in a way that optimally allocates computational resources such as processing power, memory, and storage across different edge devices based on application requirements.

Scalability: Application distribution strategies should allow for seamless scaling of applications across multiple edge nodes as the demand for services fluctuates.

Version Management: Ensuring that the correct and up-to-date versions of applications are distributed to edge devices helps maintain consistent and reliable operations.

Dependency Management: Applications often have dependencies on specific libraries or components. Effective distribution involves ensuring that all required dependencies are available on the target edge nodes.

Orchestration: Application distribution can involve orchestrating the deployment of multiple interdependent applications to ensure proper coordination and interaction between them.

Remote Deployment: Application distribution tools enable administrators to remotely deploy and manage applications on edge devices, minimizing the need for physical access.

Configuration Management: Distributing applications includes configuring software settings and parameters on edge devices to ensure that applications run as expected.

Load Balancing: Applications can be distributed to different edge nodes using load balancing mechanisms to evenly distribute workloads and prevent overloading specific devices.

Health Monitoring: Application distribution strategies should incorporate monitoring mechanisms to track the health and performance of distributed applications.

Security Considerations: Application distribution involves ensuring that applications are distributed securely, with proper access controls and encryption during transmission.

Edge Analytics: Distributing analytics applications to edge nodes enables real-time data analysis and insights at the source of data generation.

Efficient Data Usage: Applications can be distributed to minimize data transmission by processing data locally at the edge, reducing bandwidth consumption.

User Experience Optimization: Effective application distribution contributes to optimizing user experiences by ensuring that applications are readily available and responsive.

Failover and Redundancy: Application distribution strategies can include failover mechanisms to ensure uninterrupted operation if specific edge nodes become unavailable.

Decommissioning and Updates: Application distribution also involves updating and decommissioning applications as they evolve or become obsolete.

In summary, application distribution in edge computing is crucial for maintaining efficient, reliable, and scalable operations across distributed environments. It encompasses various activities, tools, and strategies aimed at ensuring the optimal deployment, performance, and maintenance of applications and services within the edge infrastructure.

Scheduling Strategies

Scheduling strategies in the context of edge computing refer to the methods and algorithms used to allocate computational resources, such as processing power, memory, and storage, to various tasks, applications, and services running on edge devices, servers, and nodes. Effective scheduling strategies are essential to optimize resource usage, improve performance, and ensure efficient operations in edge computing environments. Here are some common scheduling strategies used in edge computing:

Task Priority Scheduling: This strategy assigns higher priority to tasks that are more critical or time-sensitive. Tasks with lower priority are scheduled only when higher-priority tasks are not active, ensuring that critical tasks receive prompt attention.

Round-Robin Scheduling: In this strategy, tasks are assigned to edge devices in a cyclic manner. Each device gets a turn to execute tasks, ensuring fair resource distribution. This can be useful for tasks with similar priorities.

Deadline-Based Scheduling: Tasks with specific deadlines are given priority. The scheduling algorithm ensures that tasks are scheduled in a way that their deadlines are met, minimizing the risk of missing important time constraints.

Load Balancing Scheduling: This strategy aims to evenly distribute workloads across edge devices to avoid overloading any single device. It improves resource utilization and prevents performance bottlenecks.

Predictive Scheduling: Using historical data or predictive analytics, this strategy anticipates resource demands and schedules tasks accordingly. It's useful for applications with varying workloads.

Location-Aware Scheduling: Tasks are assigned to edge devices based on their physical proximity to data sources or users. This reduces data transmission time and improves response times.

Dynamic Scheduling: This strategy adjusts task assignments based on real-time conditions, such as changing workloads, device availability, and network conditions. It's flexible and adaptive to changing environments.

Energy-Efficient Scheduling: Tasks are scheduled to minimize energy consumption by considering factors like device power consumption, workload, and execution time.

Batch Scheduling: Tasks are grouped into batches and scheduled for execution together. This strategy can improve efficiency by reducing context-switching overhead.

Task Offloading: Some tasks can be offloaded from edge devices to more powerful cloud resources based on criteria like workload, device capabilities, and network conditions.

QoS-Aware Scheduling: Quality of Service (QoS) requirements, such as response time or throughput, are considered when scheduling tasks. It ensures that applications meet their performance targets.

Multi-Objective Scheduling: This approach considers multiple objectives, such as minimizing response time while maximizing resource utilization. It involves trade-offs between competing goals.

Container Orchestration Scheduling: When containers are used for application deployment, orchestration platforms like Kubernetes can automate the scheduling of containers on edge nodes based on resource availability and constraints.

Data-Driven Scheduling: Scheduling decisions are influenced by the characteristics of the data being processed. Data-intensive tasks may be scheduled to run on nodes where the required data is readily available.

The choice of scheduling strategy depends on the specific requirements of the edge computing environment, the nature of applications, resource constraints, and the desired performance goals. Effective scheduling strategies contribute to efficient resource utilization, improved application responsiveness, and optimized user experiences in edge computing systems.

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 presentation

BITS Pilani
Pilani Campus

Paramananda Barik
CS&IS Department

BITS Pilani
Pilani Campus

SEZG586/SSZG586, Edge Computing

Lecture No.7

Case Study: EdgeOS$_H$

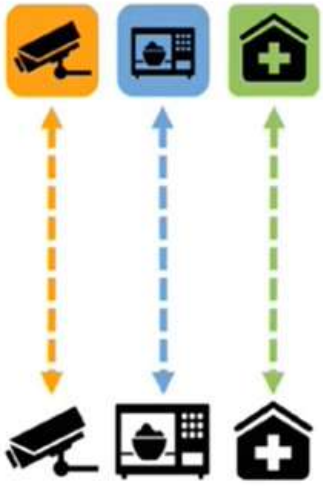
A Home Operating System for Internet of Everything

Smart home design

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

EdgeOSH

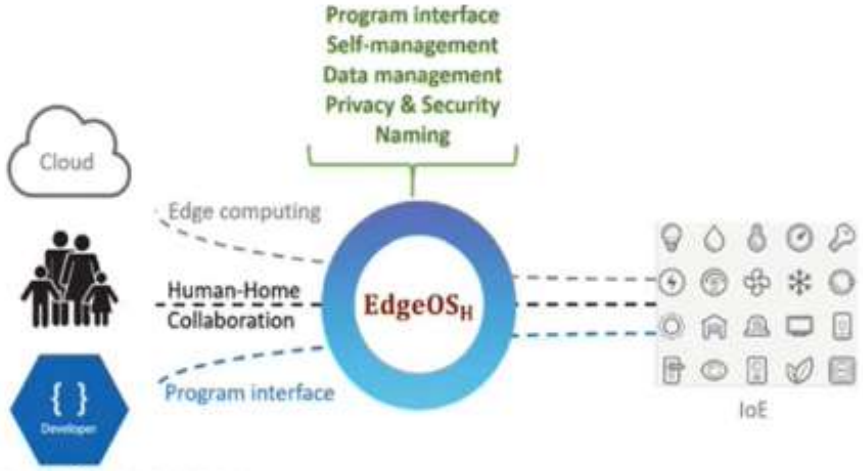
Systems work in a silo-based manner and cannot be connected or communicate with other systems



EdgeOSH Overview and Design

Benefits

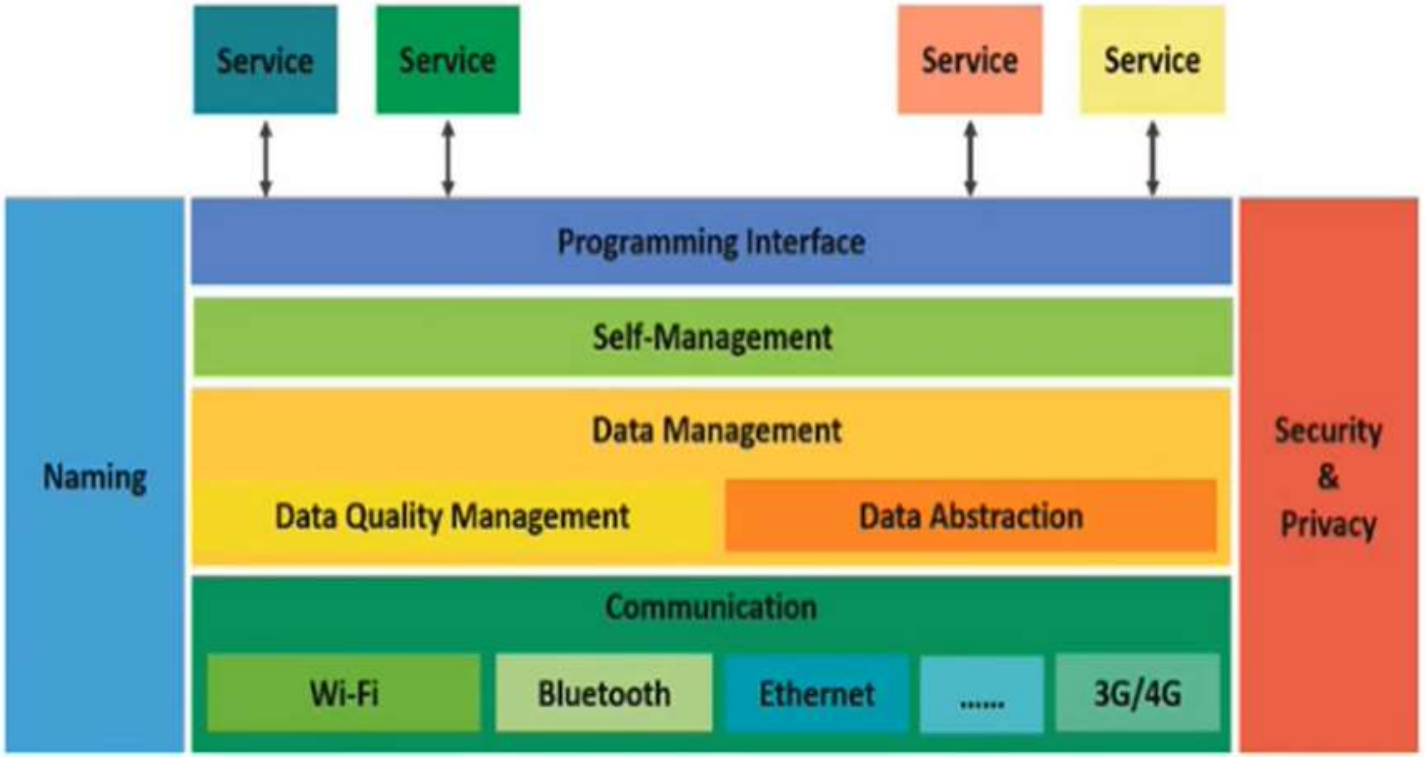
- Network load could be reduced
- Service response time could be decreased
- Data is better protected



Challenge

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

Overview


Service
Service
Service
Service
Programming Interface
Self-Management
Data Management
Data Quality Management
Data Abstraction
Communication
Wi-Fi
Bluetooth
Ethernet
.....
3G/4G
Naming
Security & Privacy

Overview

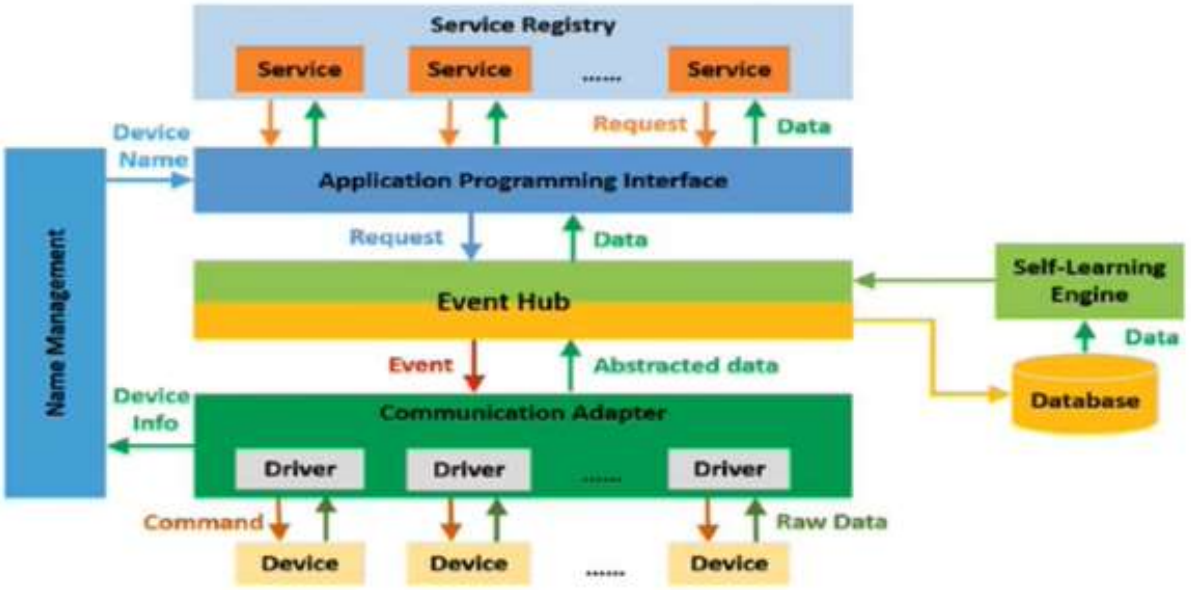
Layered approach

- 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

Design

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

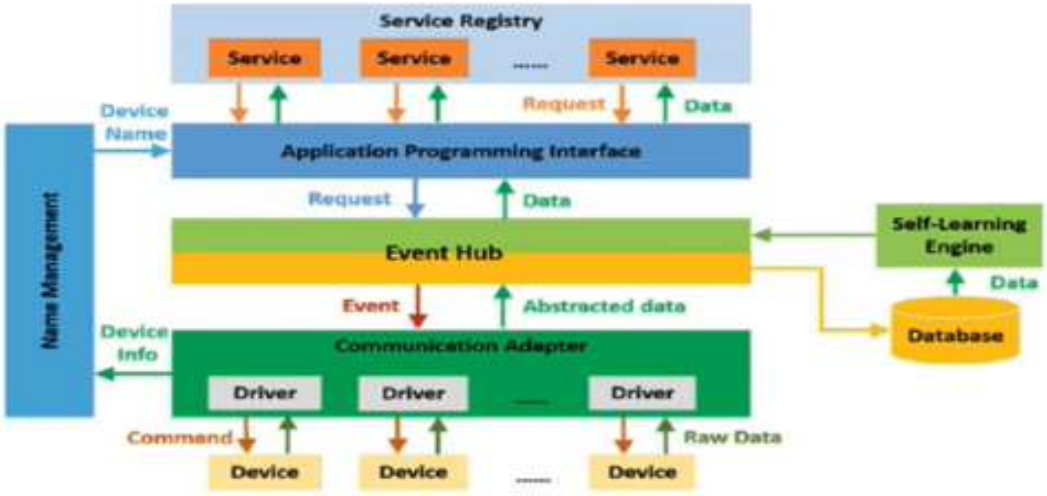
7 Components

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



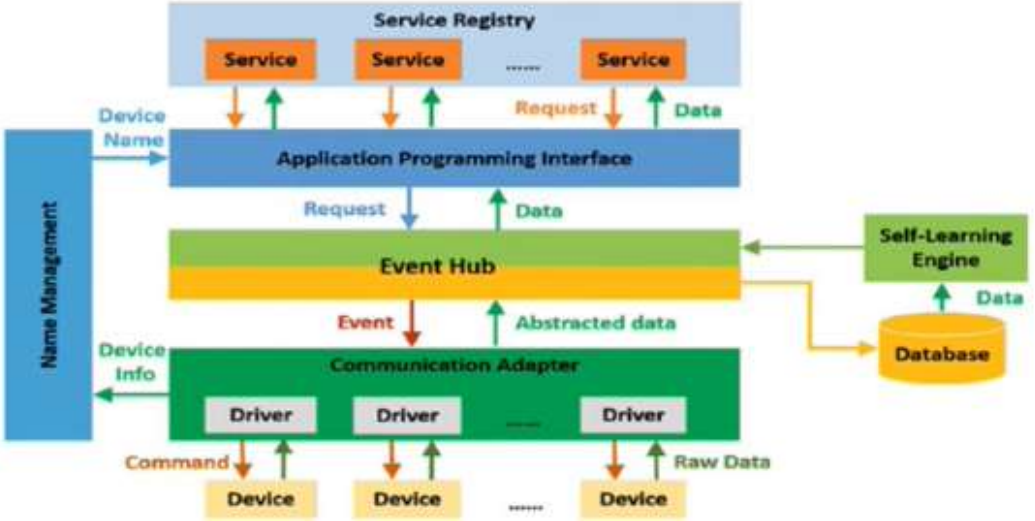
7 Components

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

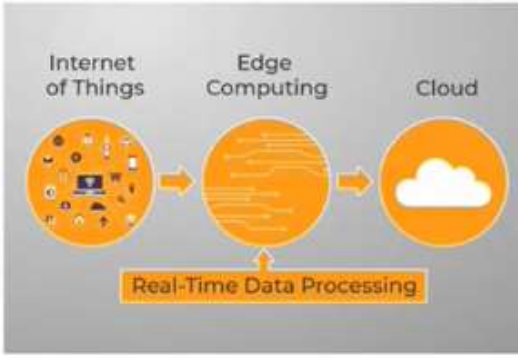
- One programming language
- Compiled for a certain target platform
- Runs in the cloud

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

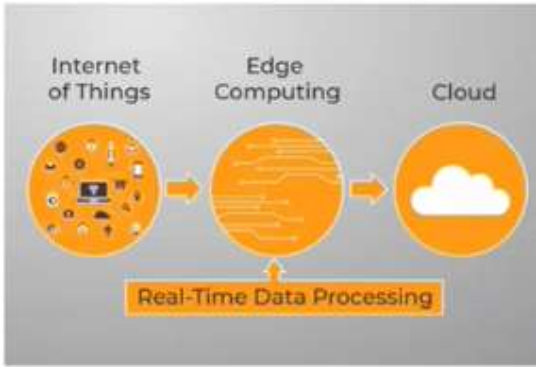
The function can be reallocated

- 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

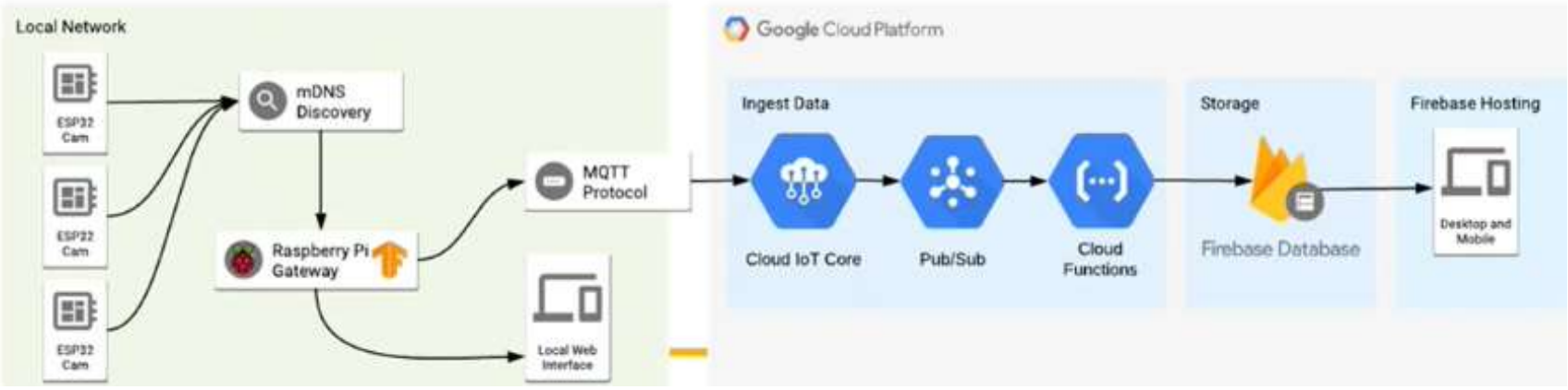
Raspberry Pi board as a local server

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**



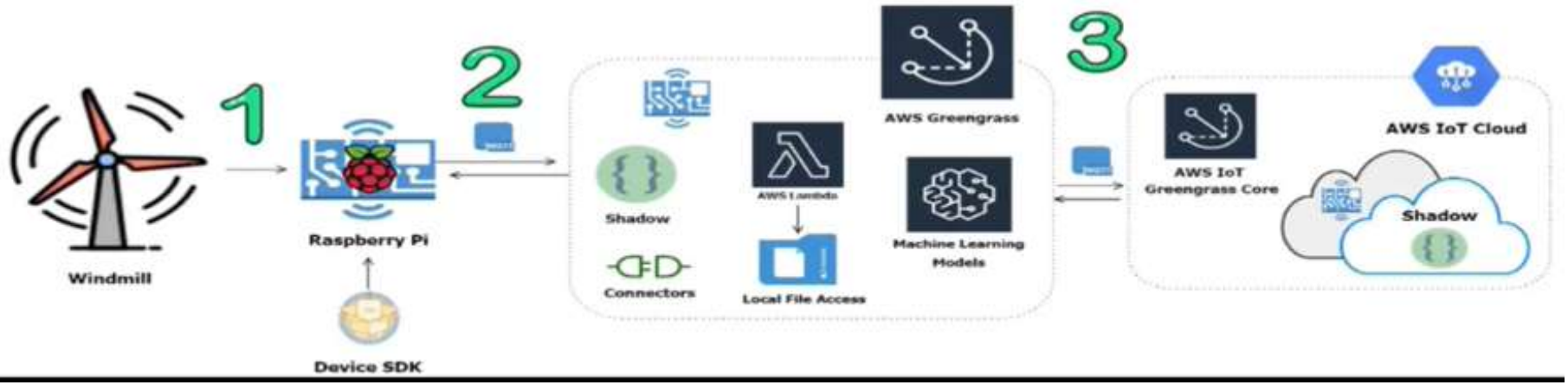
Example 2 – AWS IOT Greengrass for Windmail monitoring

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

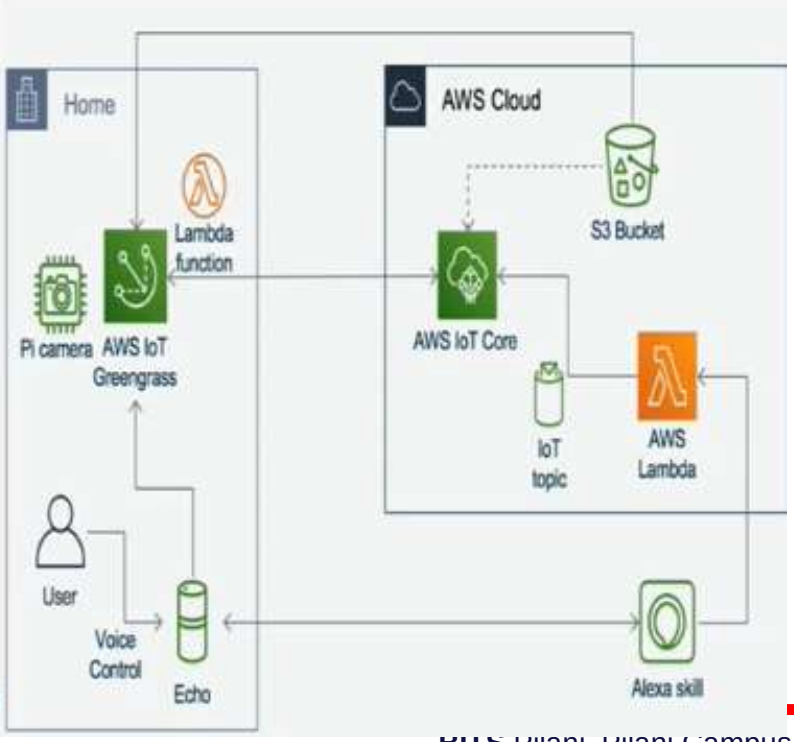
The facial recognition model and datasets uploaded to an Amazon S3 bucket

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

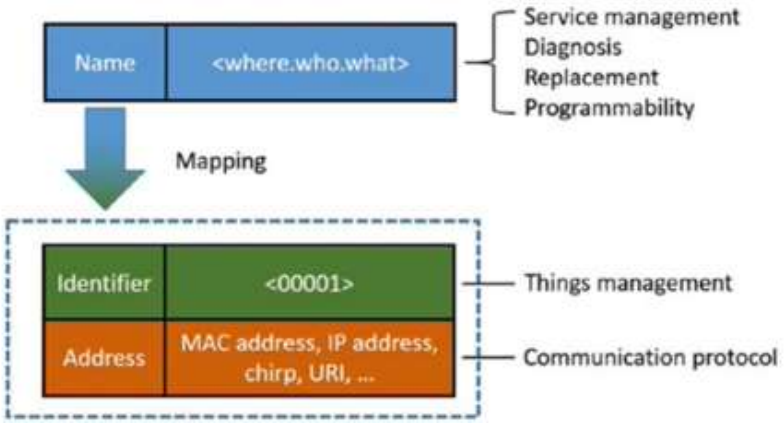
Naming : Edge OS

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



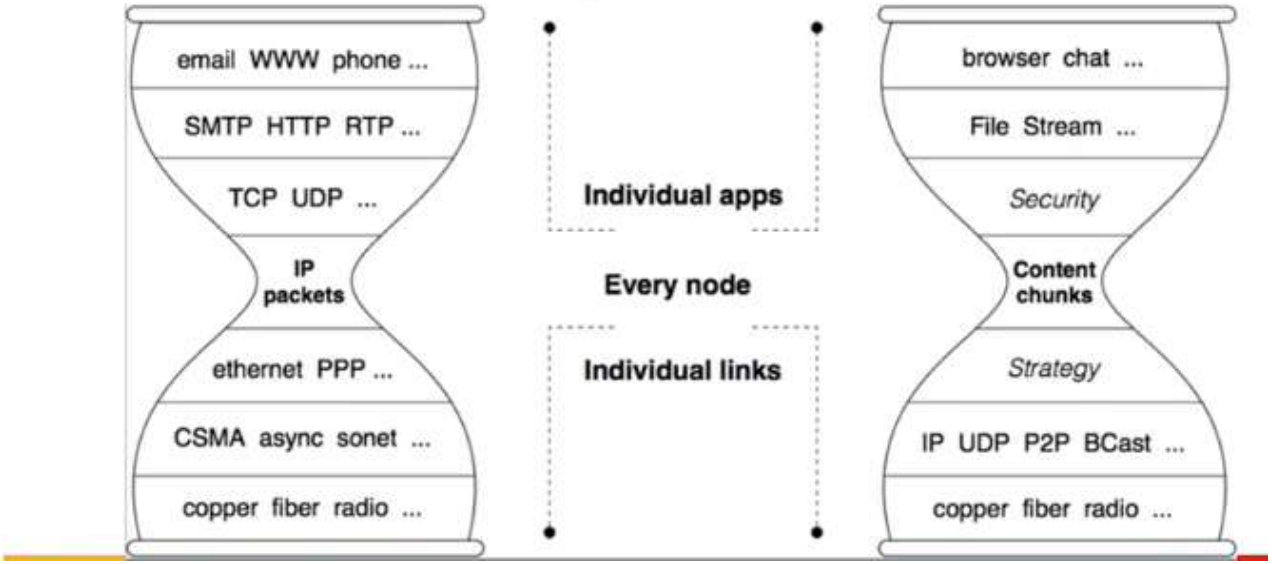
New naming mechanism

innovate achieve lead

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

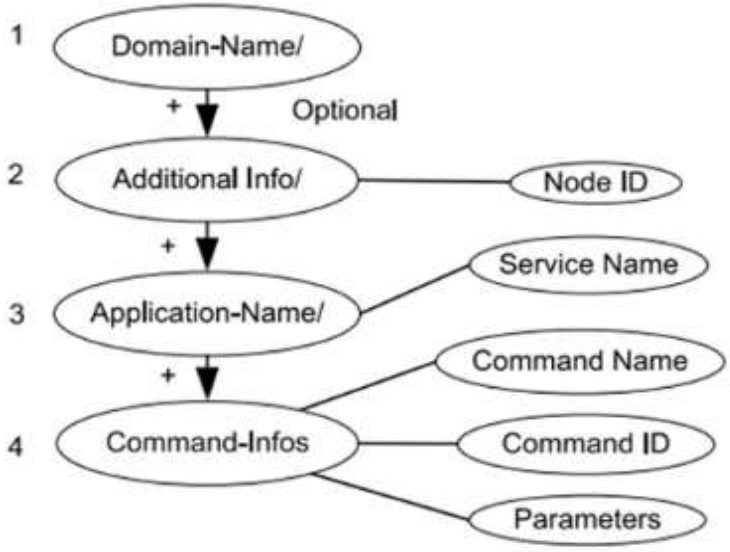


Naming : NDN (Named Data Networking)

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

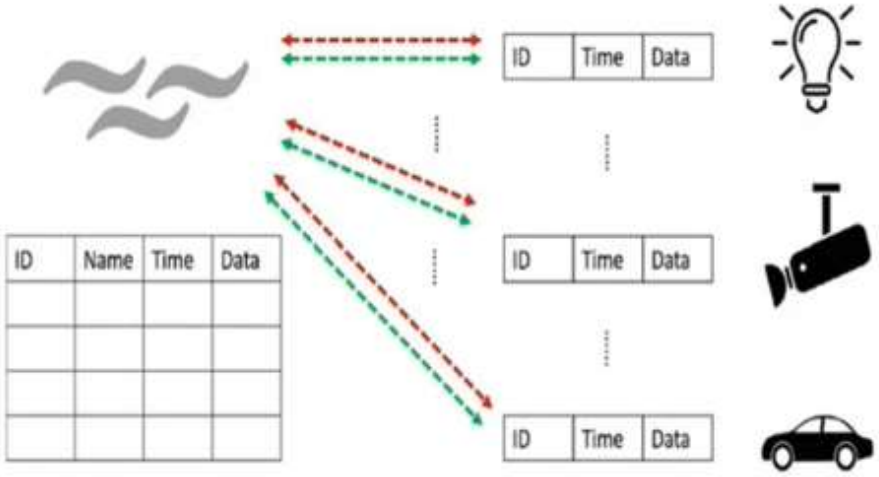
Challenges in Edge: Edge node should consume/process all the data and interact with users proactively

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: Messing 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