Computer Engineering for the Internet of Things

UNIVERSITÀ DELLA CALABRIA
DIPARTIMENTO DI INGEGNERIA INFORMATICA, MODELLISTICA, ELETTRONICA E SISTEMISTICA
DIMES

Distributed Systems and Cloud/Edge Computing for IoT

Practical Lesson 1

Riccardo Cantini

Mail: *rcantini@dimes.unical.it*
Website: *https://riccardo-cantini.netlify.app/courses/ds4iot/*

Academic year 2019/2020

EDGE AND FOG COMPUTING

- Main concept behind Edge and Fog:

  "Networking paradigm focused on bringing computing as close as possible to the edge of the network, where data are generated."

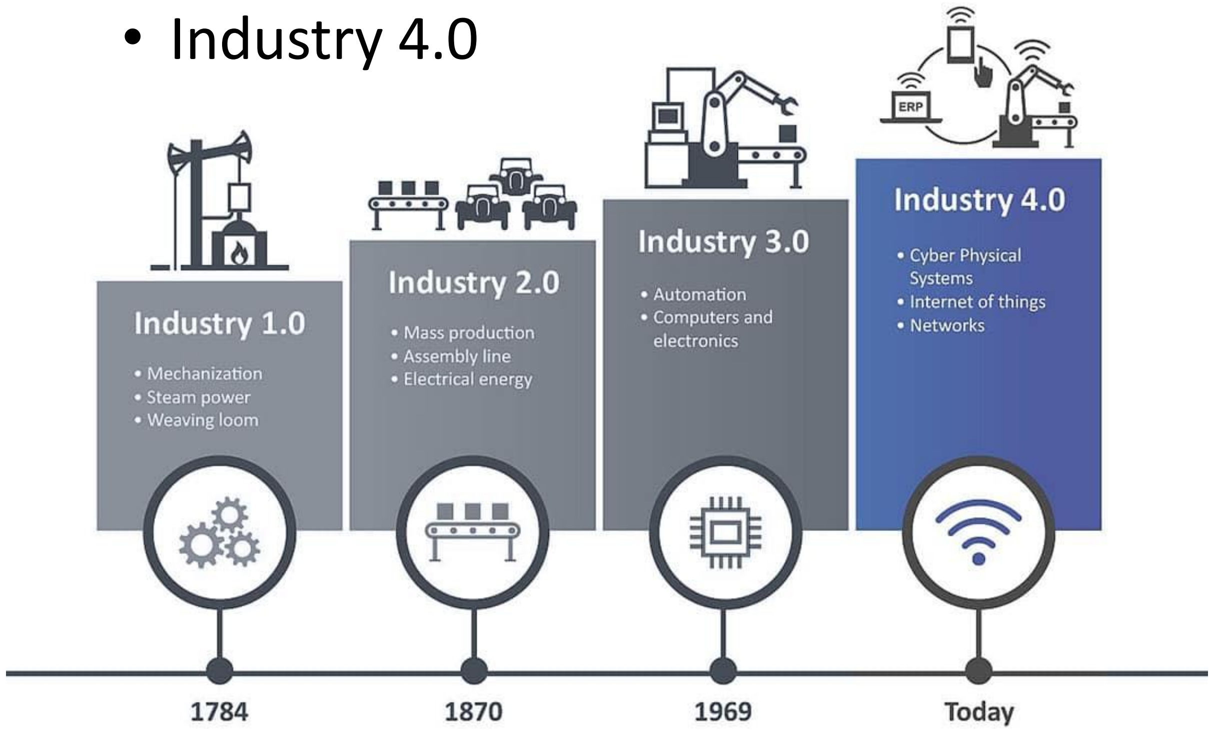
- What's the difference?
  - **Fog computing** pushes intelligence down to the local area network level of network architecture, processing data in a fog node or IoT gateway, plced between the cloud and the edge of the network.
  - **Edge computing** pushes the intelligence, processing power and communication capabilities directly into the device where data are originated.

WHY EDGE? – SOME BENEFITS

- **Low latency**
  - Operating at the source of data
  - Faster response time for triggers
- **Cost effectiveness**
  - No need to transport everything to the cloud
  - Reduce bandwith consumption
- **Reliability**
  - Can work fine also with a bad connectivity
- **Security**
  - Reduce the risk of tempering or eavesdropping as the data are stored locally

WHY EDGE? – APPLICATIVE SCENARIOS

- **Autonomous systems**
  - Industry 4.0
  - Self driving cars

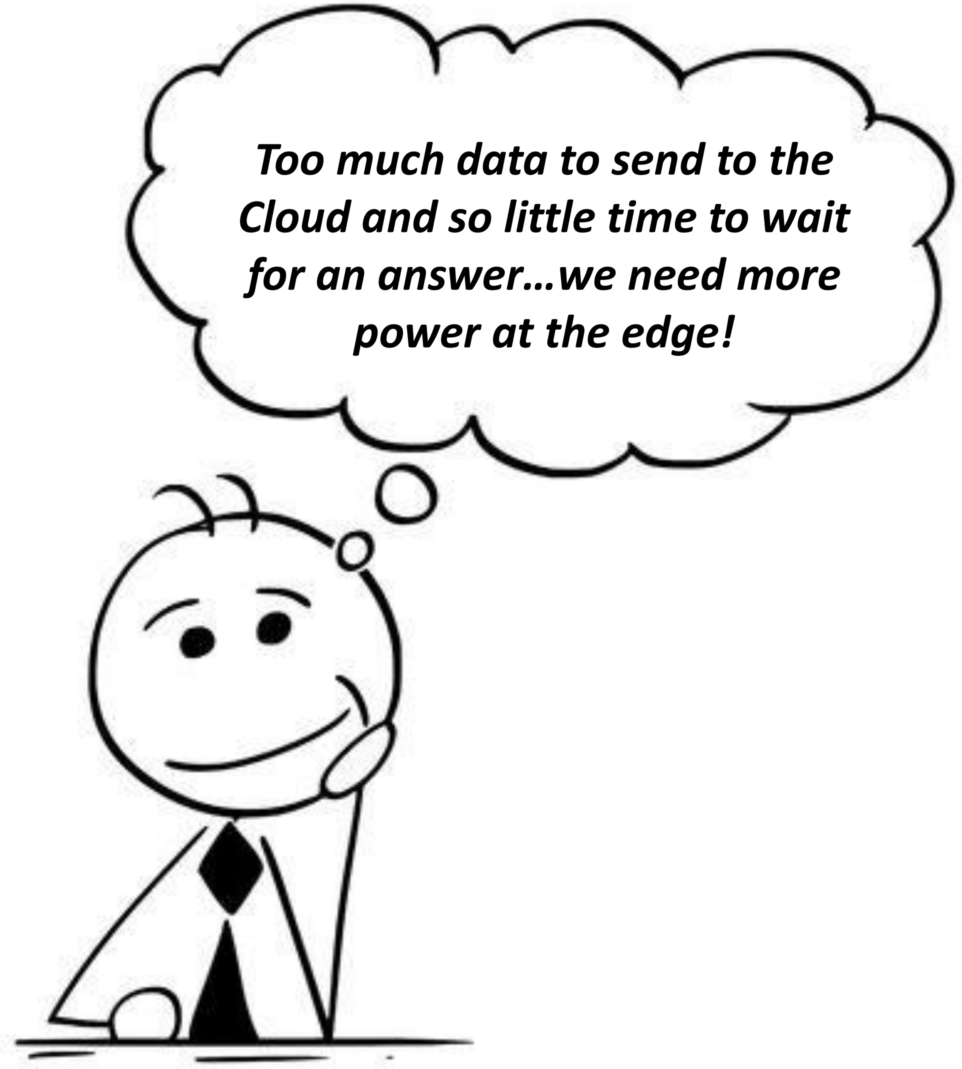


WHY EDGE? – APPLICATIVE SCENARIOS

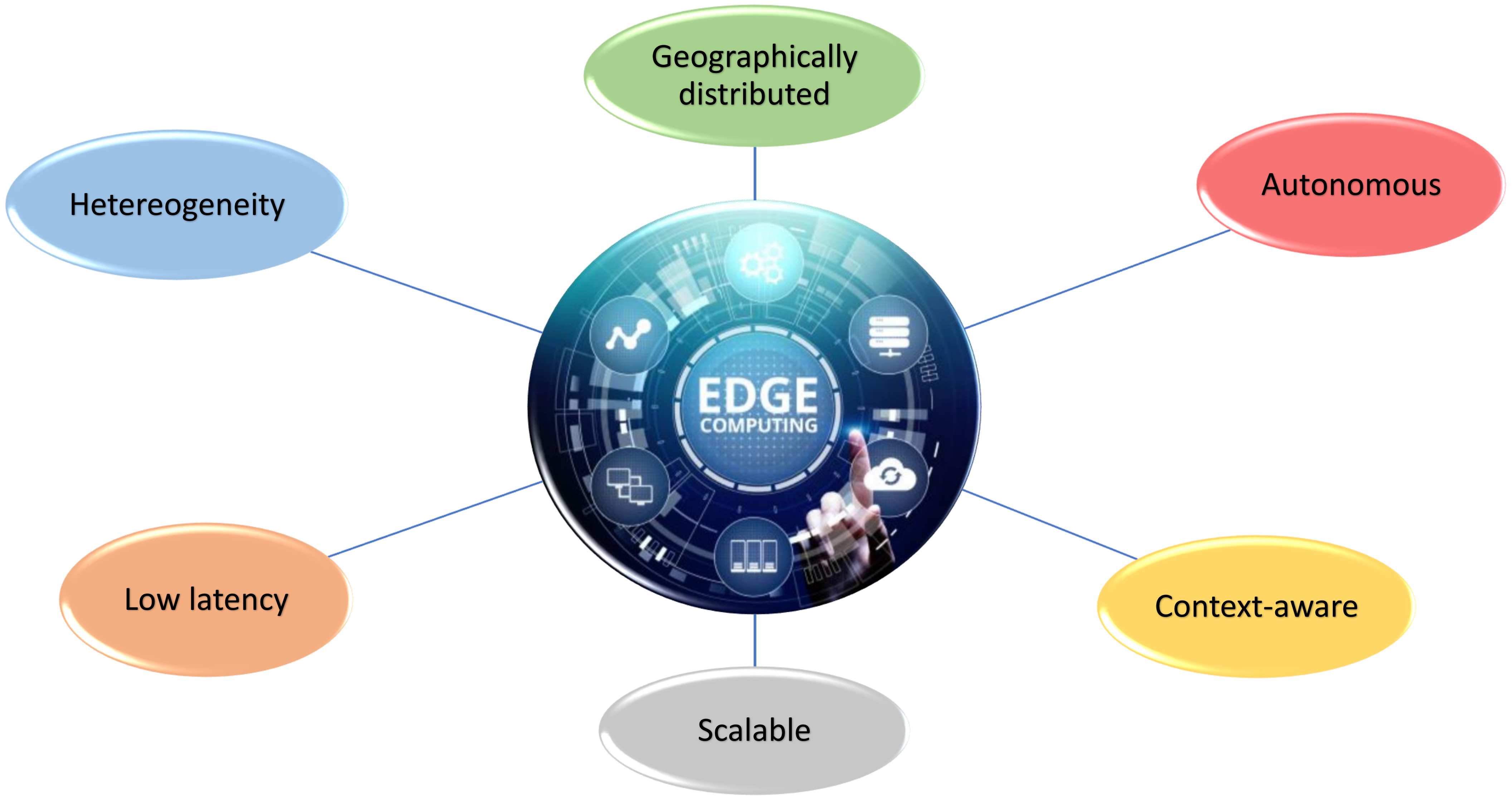
- **Low latency (near real-time) systems**
  - Health care
  - Financial transactions

WHY EDGE? – APPLICATIVE SCENARIOS

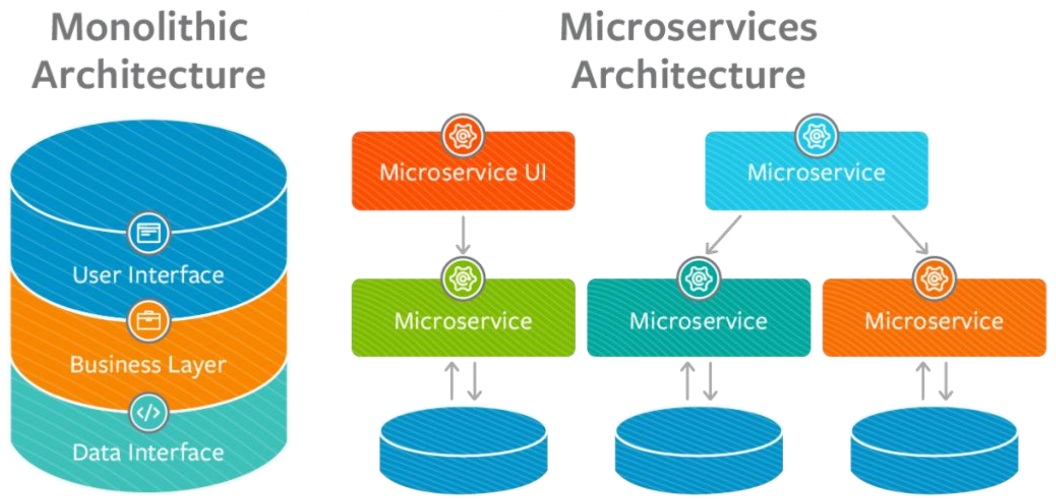
- **Bandwidth high consuming systems**
  - Smart surveillance systems


Too much data to send to the Cloud and so little time to wait for an answer…we need more power at the edge!

MAIN CONCEPTS



MICROSERVICES IN EDGE COMPUTING AND IOT



Main idea behind MSA

- Disaggregation of a general application across different services that operate together to perform an application function.
- Can apply appropriate compute, storage, and network capability to particular services without impacting every other microservice.

Applying it to Edge or IoT systems

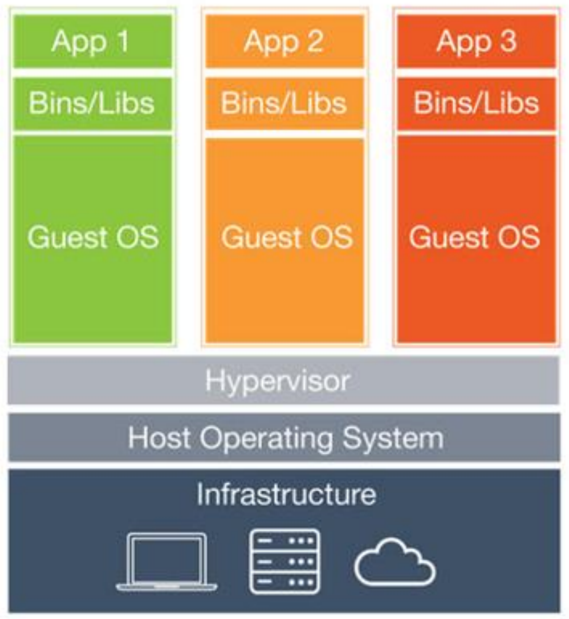
- ***Efficiency***
  - An IoT or edge microservices application can be orchestrated to use the appropriate edge hardware to run the appropriate function.
  - It's easier to share and reuse scarce edge resources in a virtualized, cloud-native fashion.
- ***Security***
  - Microservices provide an higher level of isolation for edge and IoT applications.
  - They can be designed to minimize the attack surface by running only specific functions and only when needed.

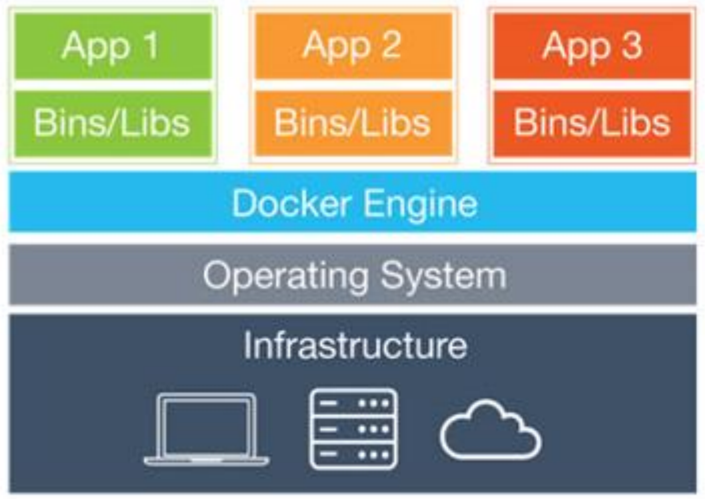
MICROSERVICE CONTAINERIZATION

- Microservices application architecture can be run using containerization.
- Containers offer:
  - A lightweight and finer-grained runtime environment, in contrast with classic virtualization.
  - Application isolation and security, which allows for component cohabitation and simultaneous execution of different containers on the same host.
  - Faster initialization and execution.

- Docker is an open source platform for development, distribution and execution of applications which uses a container-based virtualization technology.

CONTAINERS VS VIRTUAL MACHINES





Virtual machines

- Each virtual machine include the application, the necessary binaries and libraries and an entire guest operating system (all of them may be tens of GBs in size).

Containers

- Containers include the application and all of its dependencies, but share the kernel of the host with other containers. They run as an isolated process in a userspace on the host operating system and are not tied to any specific infrastructure.

CONTAINER ORCHESTRATION

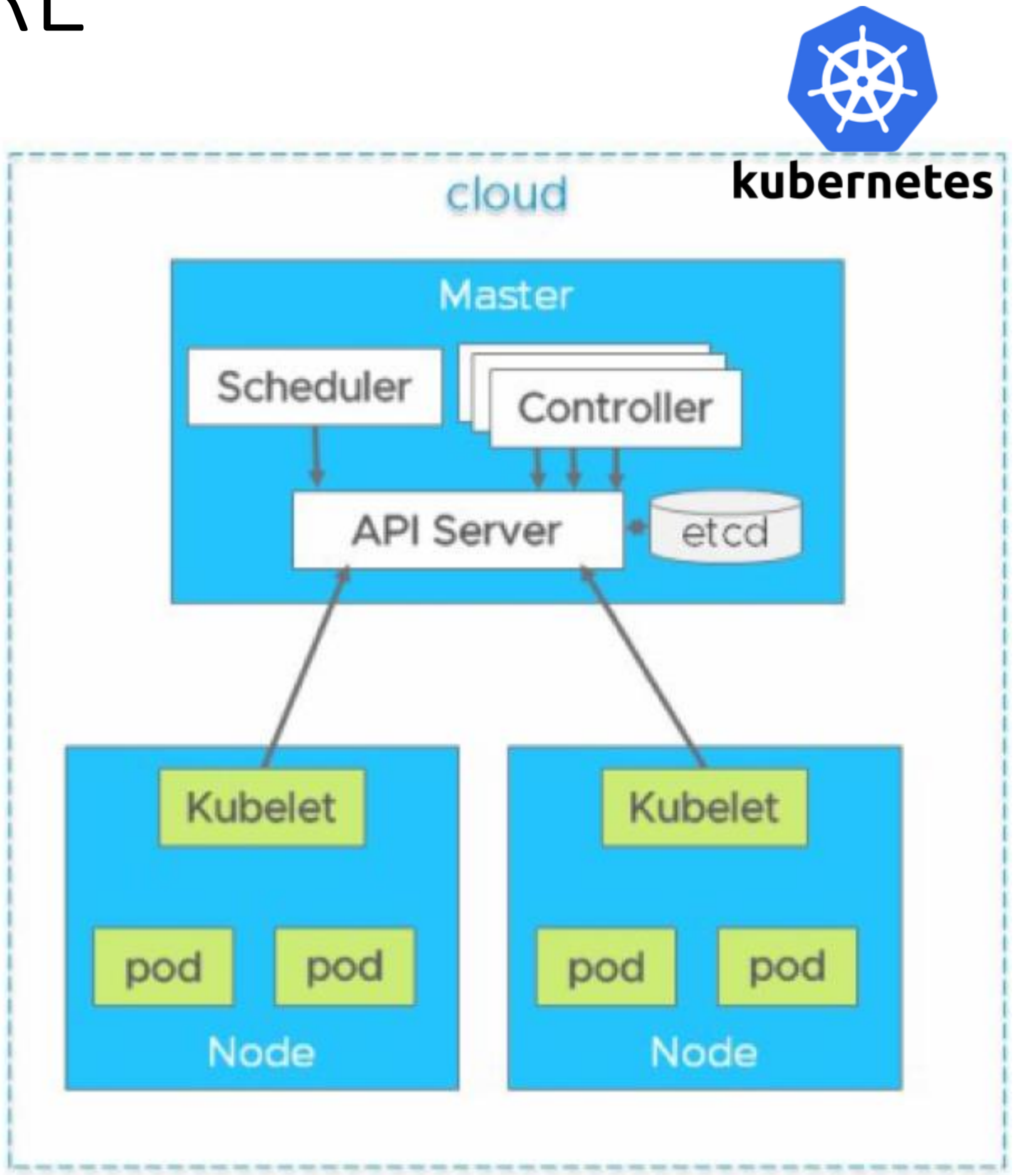
- Kubernetes is a Google open source project for container orchestration
- Main advantages are:
  - Automatization of the administration and distribution process of large-scale multi-container applications.
  - Faster deployment and update of multi-container applications.
  - Speed-up large-scale application development
  - Load balancing and efficient use of physical resources.
- Kubernates is not tied to a specif environment but it can be used everywhere containers are supported (private cloud, public cloud, physical or virtual hw).
- Kubernates is most of all used in association with Docker, but it supports every system compliant with OCI (Open Container Initiative) standard.

kubernetes

KUBERNATES ARCHITECTURE

A Kubernetes cluster has a master-node architecture:

- Master is responsible for exposing the API to developers and scheduling the deployment of all clusters, including nodes.
- Nodes (e.g. virtual machine in the cloud) contain three elements:
  - The container runtime environment (Docker)
  - An element called kubelet, which communicates with the master
  - Pods, which are collection of one or more containers



SOME KEYWORDS…

- **Desired state**: defines applications or other workloads to be run, container images to used, the number of replicas, available network and disk resources, and more.

- **Control Plane**: makes the cluster’s current state match the desired one, performing a variety of tasks automatically (container restarting, scaling of replicas…).

- **Master**: collection of processes responsible for maintaining the desired state for the cluster. When interacting with Kubernetes (e.g. using the *kubectl* command-line interface), you’re communicating with your cluster’s Kubernetes master.

SOME KEYWORDS…

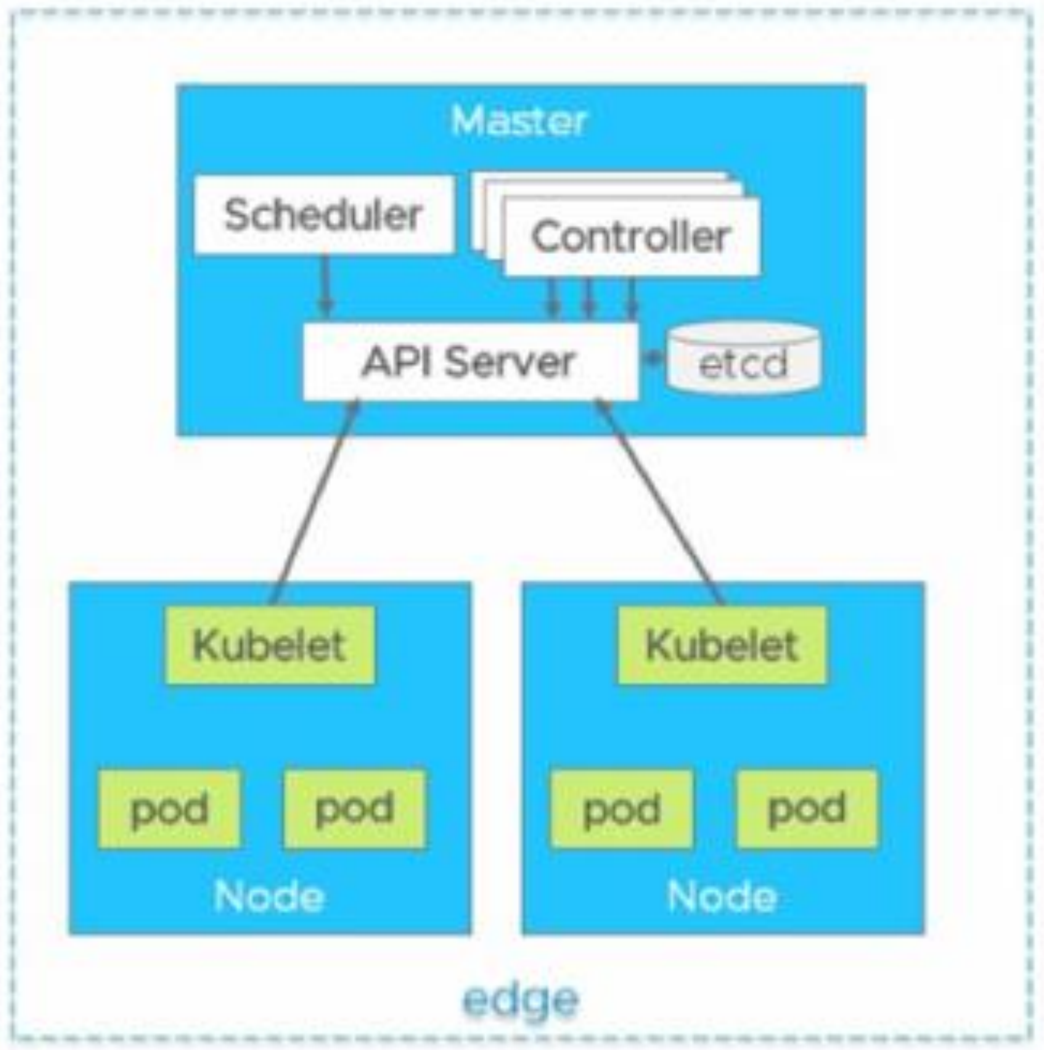
- **Nodes**: refers to the machines (e.g. VMs or physical servers) that run applications and cloud workflows. The Kubernetes master controls each non-master node, which runs two processes:
  - ***Kubelet***: communicates with the Kubernetes Master.
  - ***kube-proxy***: reflects Kubernetes networking services on each node.

- **Pod**: represents a unit of deployment (a single instance of an application in Kubernetes) and might consist of either a single container (Docker) or a small number of containers that are tightly coupled and that share resources.

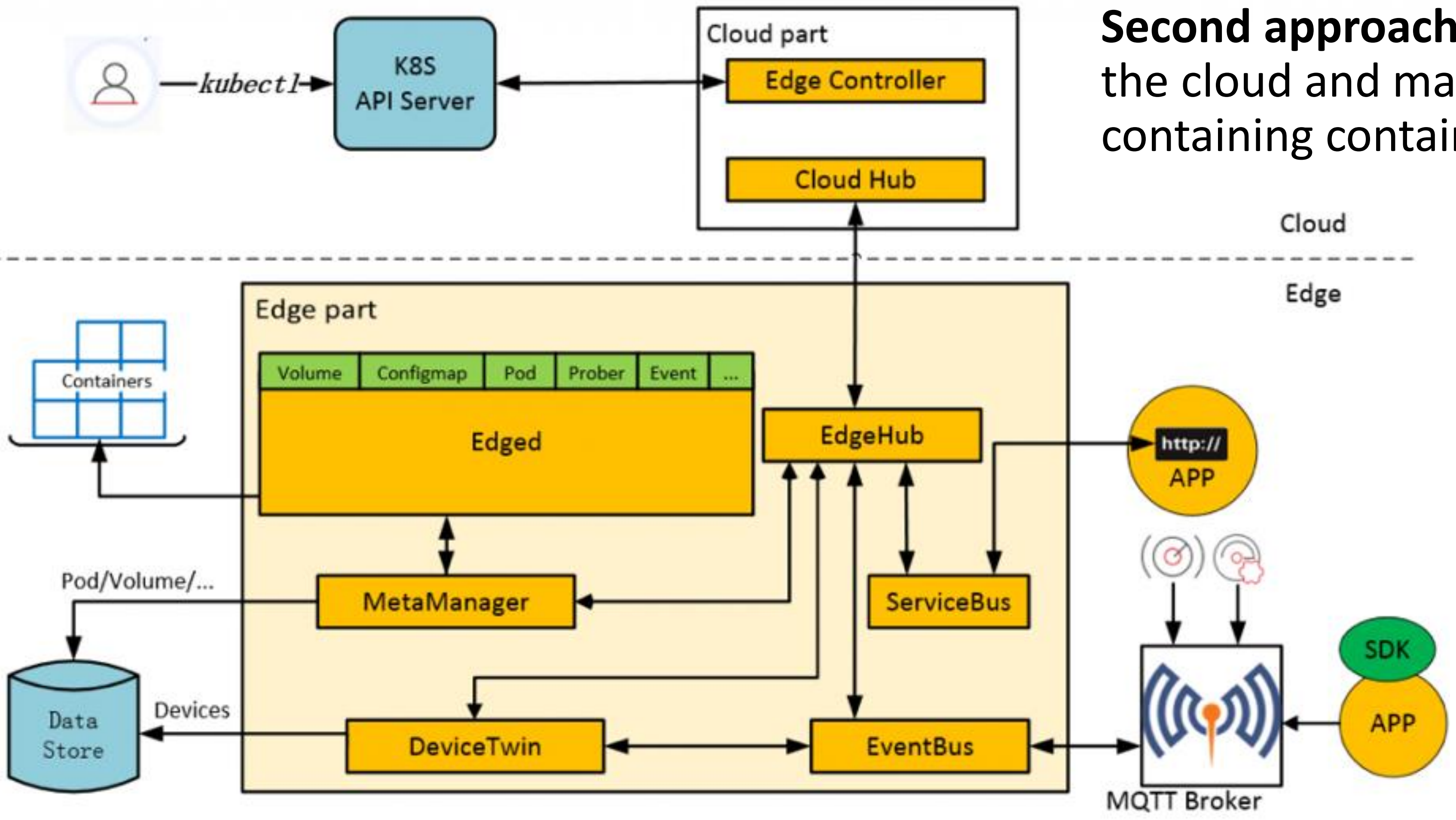
USING KUBERNATES IN AN EDGE ARCHITECTURE

First approach: the whole Kubernetes cluster is deployed within edge nodes.

- This option is a good choice when the edge node has fewer capacity resources or a single-server machine and does not want to consume more resources (e.g. for *control plane*).
- K3s is the reference architecture suited for this type of solution.
- Lightweight clusters well suited for running on edge nodes.



USING KUBERNATES IN AN EDGE ARCHITECTURE



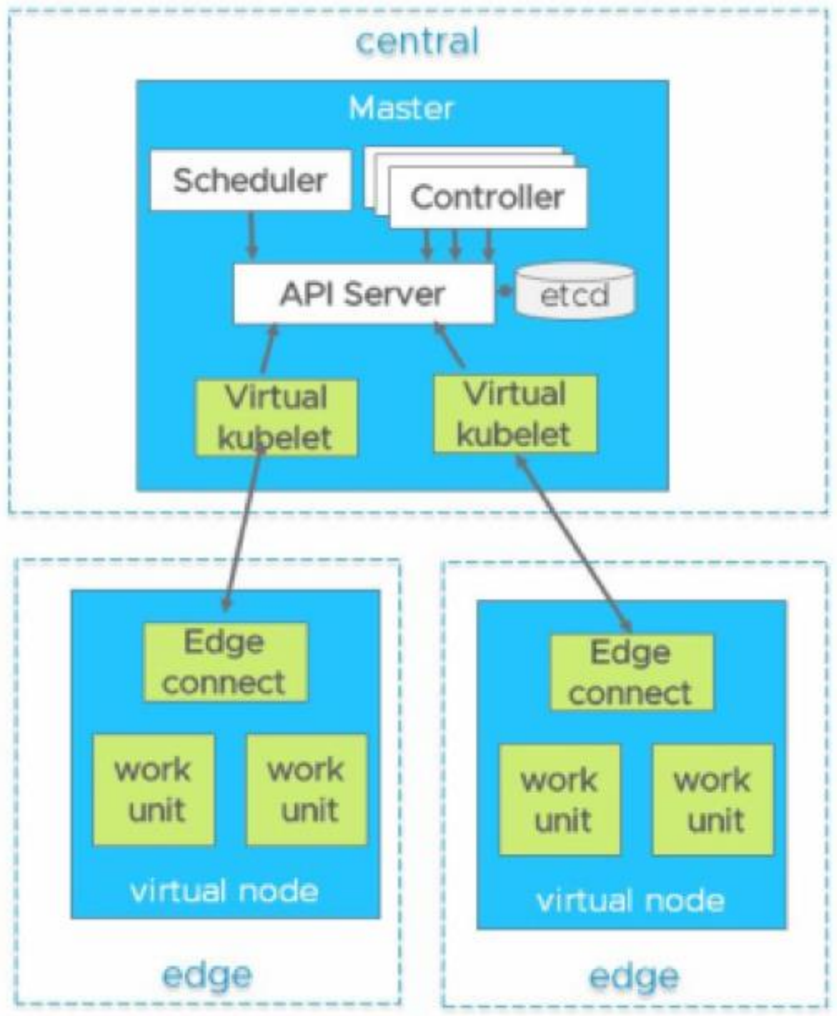
Second approach: the *control plane* resides in the cloud and manages the edge nodes containing containers and resources.

- Support for different hardware resources at the edge
- Optimization in edge resource utilization
- Significant reduction of setup and operational costs for edge cloud deployment.

USING KUBERNATES IN AN EDGE ARCHITECTURE

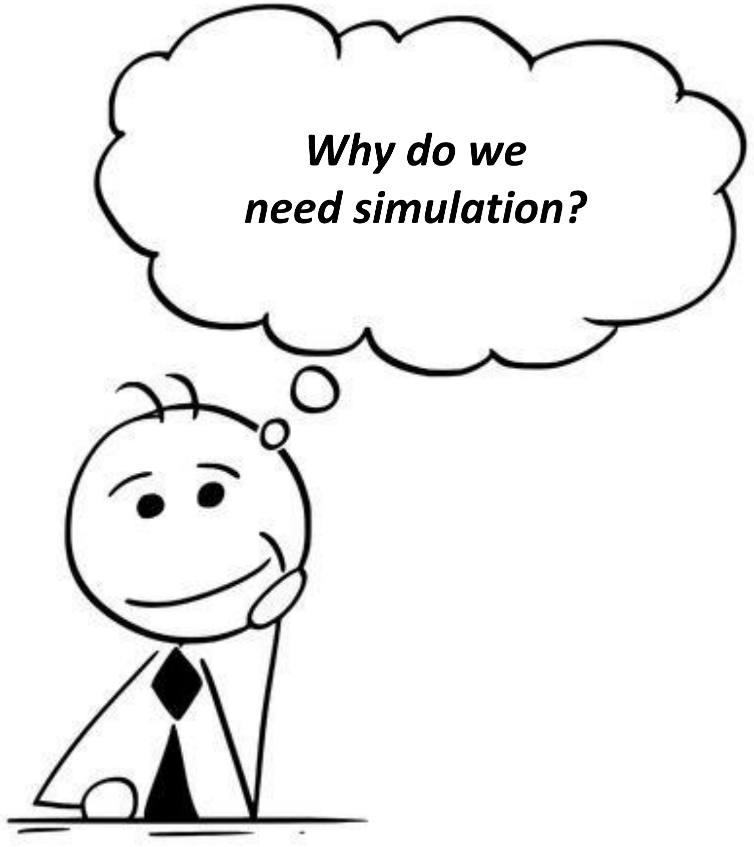
Third approach: hierarchical cloud plus edge.

- A virtual kubelet is used, which resides in the cloud and contain the abstract of nodes and pods deployed at the edge.
- Virtual kubelets get supervisory control for edge nodes containing containers.
- Using virtual kubelets enables flexibility in resource consumption for edge-based architecture.



SIMULATION IN FOG AND EDGE COMPUTING

- Simulation frameworks are used extensively in the modelling of cloud computing and IoT environments in order to test and validate technical solutions:
    - Service configuration and resource placement and management strategies can be simulated and validated prior to infrastructure deployment.
    - Performance can be optimized
    - Technical and commercial hypotheses can be tested
    - Research results can be reproduced and validated in a low cost, low risk and time-sensitive manner
- Simulation tools are at a nascent stage of development and adoption for fog and edge computing.



FOG AND EDGE COMPUTING: MODELLING AND SIMULATION CHALLENGES

Application-level modelling

- There’s a vary wide range of potential applications for fog and edge computing:
  - Simple-IoT based sensor monitoring
  - Data processing systems in Industry 4.0, e-health, smart cities…
- Underlying applications are developed according to the degree of:
  - Contextual location awareness
  - Geographic distribution, mobility and heterogeneity
  - Scale and coordination of end-point networks
  - Latency: real-time vs batch processing
  - Interplay between the edge, the fog and the cloud layers
- Modelling all of the applications deployed within a fog/edge network can be beneficial for infrastructure providers but constructing a simulation solution that can efficiently handle a set of such broad objectives is a hard challenge.

FOG AND EDGE COMPUTING: MODELLING AND SIMULATION CHALLENGES

Infrastructure and network-level modelling

- An automated approach is needed for model building:
  - Thousands of distributed site locations create a network of resources spanning multiple countries.
  - Each site can be comprised of computing and network equipment hosting multiple applications that can be accessed by edge service users.
- Integration with a monitoring data collection system can partially address the challenge by taking snapshot of an existing infrastructure state.
- In order to build meaningful system behaviour models the monitoring data has to be extended in the simulation domain in order to extract behaviour trends of workload and application resource demands:
  - Big data management
  - Big data processing

FOG AND EDGE COMPUTING: MODELLING AND SIMULATION CHALLENGES

Mobility

- 5G networks offer network improvements by optimizing mobile resource usage, large data pre-processing, and context-aware services.
- Fog and edge application may have different latency requirements and generate different types of data and network traffic: delay-sensitive flows differentiation is needed.
- Modelling user mobility aspects requires the implementation of geographic awareness logic (e.g. calculation of the nearest mobile access point based on user coordinates at each simulation timestep).
- Availability and access to real-world data on end user mobility is problematic both legally and technically.
- Intelligent model generators allows the creation of fog and edge infrastructure workload models based on 3rd party socio-demographic and geographic data, that can be used for simulation purposes.

FOG AND EDGE COMPUTING: MODELLING AND SIMULATION CHALLENGES

Resource management

- Edge devices are very energy constrained, but also hungry energy consumer.
- Simulation can help to find the best configuration in terms of resource allocation:
  - Computational and storage capacity
  - Battery life
  - Mobility
  - Communication interface
- Simulation can also be used as part of resource management algorithms, within optimization techniques like simulated annealing.
  - Simulated annealing is an optimization algorithm that uses local search approach of moving around the neighboring values in a defined search space until the optimal solution is found.
  - It uses a decaying probability of moving with a random step , in order to avoid local minima.

FOG AND EDGE COMPUTING: MODELLING AND SIMULATION CHALLENGES

Scalability

- The choice of a simulation tool depends on:
  - The type of applications and desired granularity of the simulation (e.g., macroscopic phenomena, such as routing strategies, can be studied by packet-level, using the DES approach).
  - The generality of the range of phenomena and applications that can be simulated.
- Simulating approaches:
  - DES (Discrete Event Simulation) is the most popular but the sequential nature of the event queue is difficult to parallelize as each event can change the state of the system.
  - DTS (Discrete Time Simulation) overcomes this parallelization issue. It uses the concept of time-step to update the state of the system components, avoiding the need for pre-computation and storage of future events.
- Pro & Cons:
  - DES is more suitable when applied to a problem requiring a more granular modelling approach which is more difficult to scale.
  - DTS enables the modelling of large scale systems with relatively less effort, but with the possibility of a higher degree of inaccuracy.

FOG AND EDGE COMPUTING: MODELLING AND SIMULATION TOOLS

FogNetSim++

- Is a fog simulator tool that provides users with detailed configuration options to simulate a large fog network.
- It is designed on the top of OMNeT++ (an open source tool for network characteristics simulation based on DES).
- Allows the definition of customized mobility models and fog node scheduling algorithms as well as managing handover mechanisms.
- Allows the analysis of different network parameters, such as execution delay, packet error rate, handovers, and latency.
- Does not yet support VM migration among fog nodes.

FOG AND EDGE COMPUTING: MODELLING AND SIMULATION TOOLS

iFogSim

- Is a fog computing simulation toolkit, based on CloudSim, used for the modelling and simulation of fog computing infrastructures and execute simulated applications.
- It allows:
  - The measurement of performance, in terms of latency, energy consumption and network usage.
  - The evaluation of resource-management and scheduling policies.
  - The simulation of edge devices, cloud data-centres, sensors, network links, data streams, and stream-processing applications.
- In addition, iFogSim integrates simulated services for power monitoring and resource management at two separate levels:
  - Application placement
  - Application scheduling

FOG AND EDGE COMPUTING: MODELLING AND SIMULATION TOOLS

iFogSim

- Supports multiple deployment scenarios through two application module placement strategies:
  - Cloud-only placement, where all applications modules run in data centres
  - Edge-ward placement, where application modules run on fog nodes close to edge devices.
- Extensions are available to support the design of data placement strategies according to specific objectives such as minimization of:
  - Service latency
  - Network congestion
  - Energy consumption.
- iFogSim does not support mobility and has a limited scalability, as it is based on DES.

FOG AND EDGE COMPUTING: MODELLING AND SIMULATION TOOLS

EdgeCloudSim

- Is a simulation tool specifically designed for the evaluation of the computational and networking needs of edge computing.
- Unlike iFogSim, EdgeCloudSim supports mobility. It provides:
    - Mobility model
    - Network link model
    - Edge server model
- It is relatively user-friendly providing a mechanism to obtain the configuration of devices and applications from the XML files instead of defining them programmatically.
- It presents the same scalability and DES limitations as iFogSim.

FOG AND EDGE COMPUTING: MODELLING AND SIMULATION TOOLS

IOTSim

- Is a simulation tool specifically designed for the simulation of edge computing environments where large data volumes are sent to a big data processing system by the IoT application.
- It adds into CloudSim a big data storage and processing layer:
  - In the storage layer, the network and storage delays are simulated for IoT applications.
  - The big data processing layer simulates MapReduce to support the batch-oriented data processing paradigm.
- It presents the same scalability and DES limitations as iFogSim and does not support mobility.

FOG AND EDGE COMPUTING: MODELLING AND SIMULATION TOOLS

FogTorchII

- It is an open source simulator developed in Java, which allows:
  - The evaluation of fog computing infrastructure deployments
  - The modeling of software capabilities (O.S., programming languages, frameworks, …)
  - The modeling of hardware capabilities (CPU cores, RAM and storage)
  - The modeling of QoS attributes (latency and bandwidth)
- It uses Monte Carlo simulations to implement variations in communications links used as inputs.
- The final output consists of the aggregated results in terms of QoS-assurance and fog resource consumption through an indicator of consumed RAM and storage percentages.
- The main limitation of FogTorchII is scalability.

FOG AND EDGE COMPUTING: MODELLING AND SIMULATION TOOLS

EmuFog

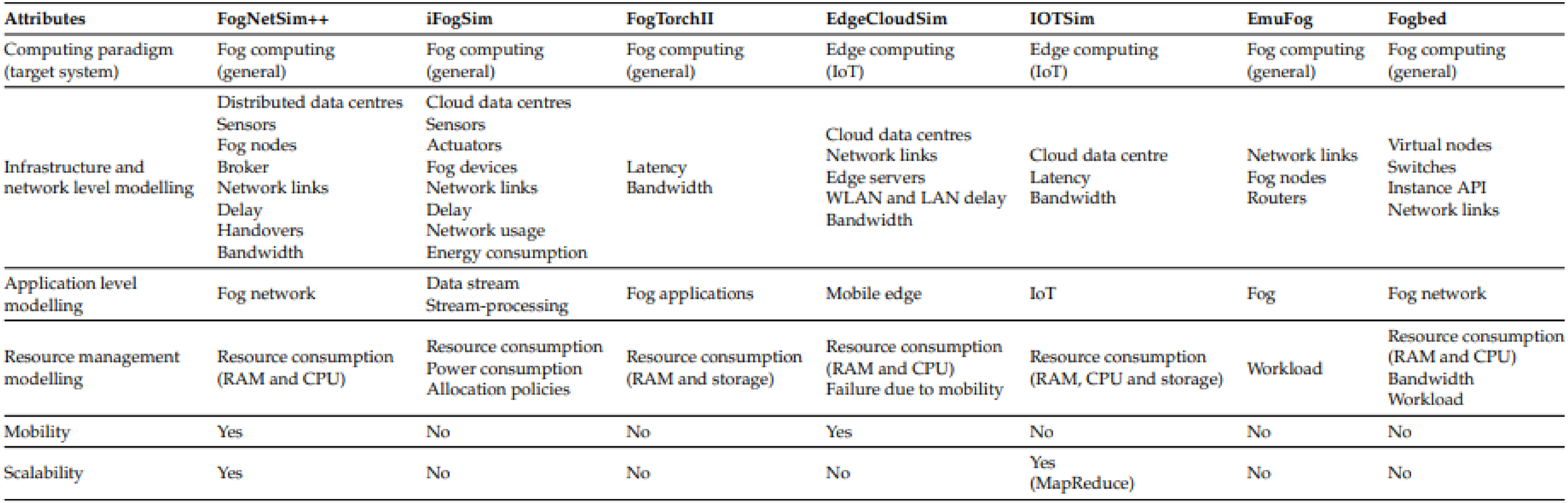
- It enables the design of fog computing infrastructures and the emulation of real large scale applications and workloads.
- The implementation process in EmuFog consists of four stages:
    1. A network topology is either generated or loaded from a file, supporting thus real-world topology datasets.
    2. The network topology is converted in an undirected graph, where nodes represent network devices (e.g., routers) and links correspond to the connections between them.
    3. The edge devices are determined and the fog nodes are placed according to a placement policy. Users are able to define the computational capabilities of fog nodes as well as the number of clients expected to be served by each node.
    4. Fog nodes are emulated from the network emulated environment, while the applications in any individual fog node are running under Docker containers.
- EmuFog does not support hierarchical fog infrastructures and mobility.

FOG AND EDGE COMPUTING: MODELLING AND SIMULATION TOOLS

Fogbed

- It is an emulator which extends the Mininet framework to allow the use of Docker containers as virtual nodes.
- It provides capabilities to build cloud and fog testbeds.
- The Fogbed API enables adding, connecting and removing containers dynamically from the network topology, allowing for the emulation of real-world cloud and fog infrastructures.
- It is possible to change the run-time resource limitations for a container, such as CPU time and memory available.
- Fogbed does not yet support key aspects of fog computing including security, fault tolerance, scalability and reliability management.

OVERVIEW



Attributes	FogNetSim++	iFogSim	FogTorchII	EdgeCloudSim	IOTSim	EmuFog	Fogbed
Computing paradigm (target system)	Fog computing (general)	Fog computing (general)	Fog computing (general)	Edge computing (IoT)	Edge computing (IoT)	Fog computing (general)	Fog computing (general)
Infrastructure and network level modelling	Distributed data centres Sensors Fog nodes Broker Network links Delay Handovers Bandwidth	Cloud data centres Sensors Actuators Fog devices Network links Delay Network usage Energy consumption	Latency Bandwidth	Cloud data centres Network links Edge servers WLAN and LAN delay Bandwidth	Cloud data centre Latency Bandwidth	Network links Fog nodes Routers	Virtual nodes Switches Instance API Network links
Application level modelling	Fog network	Data stream Stream-processing	Fog applications	Mobile edge	IoT	Fog	Fog network
Resource management modelling	Resource consumption (RAM and CPU)	Resource consumption Power consumption Allocation policies	Resource consumption (RAM and storage)	Resource consumption (RAM and CPU) Failure due to mobility	Resource consumption (RAM, CPU and storage)	Workload	Resource consumption (RAM and CPU) Bandwidth Workload
Mobility	Yes	No	No	Yes	No	No	No
Scalability	Yes	No	No	No	Yes (MapReduce)	No	No