

Osmotic Computing: a step over Cloud and Edge Computing leveraging Microservices

PROF. MASSIMO VILLARI

HEAD OF FUTURE COMPUTING RES LAB (FCRLAB) UNIVERSITÀ DI MESSINA

<http://fcrlab.unime.it/>



Our Present is ALREADY Part of Our Future

Our past determines our present, and our present is what shapes our future.

- unknown





Outline: Part I

Where we are:

- Cloud Computing
- FOG Computing
- Edge Computing
- IoT
-
- ***Big Data: it is the driver***



Outline: Part I

Where we are, in particular:

- MicroServices and Containerization
- Social Platforms
- Serverless
- APIs for any taste:
 - RESTFul
 - CoAP



Outline: Part II

Consolidated Activities:

- Cloud Computing:
- FOG Computing
- IoT
- Big Data



Outline: Part II

Where are we going??:

- Current Trends
- New Devices
- Needs
-

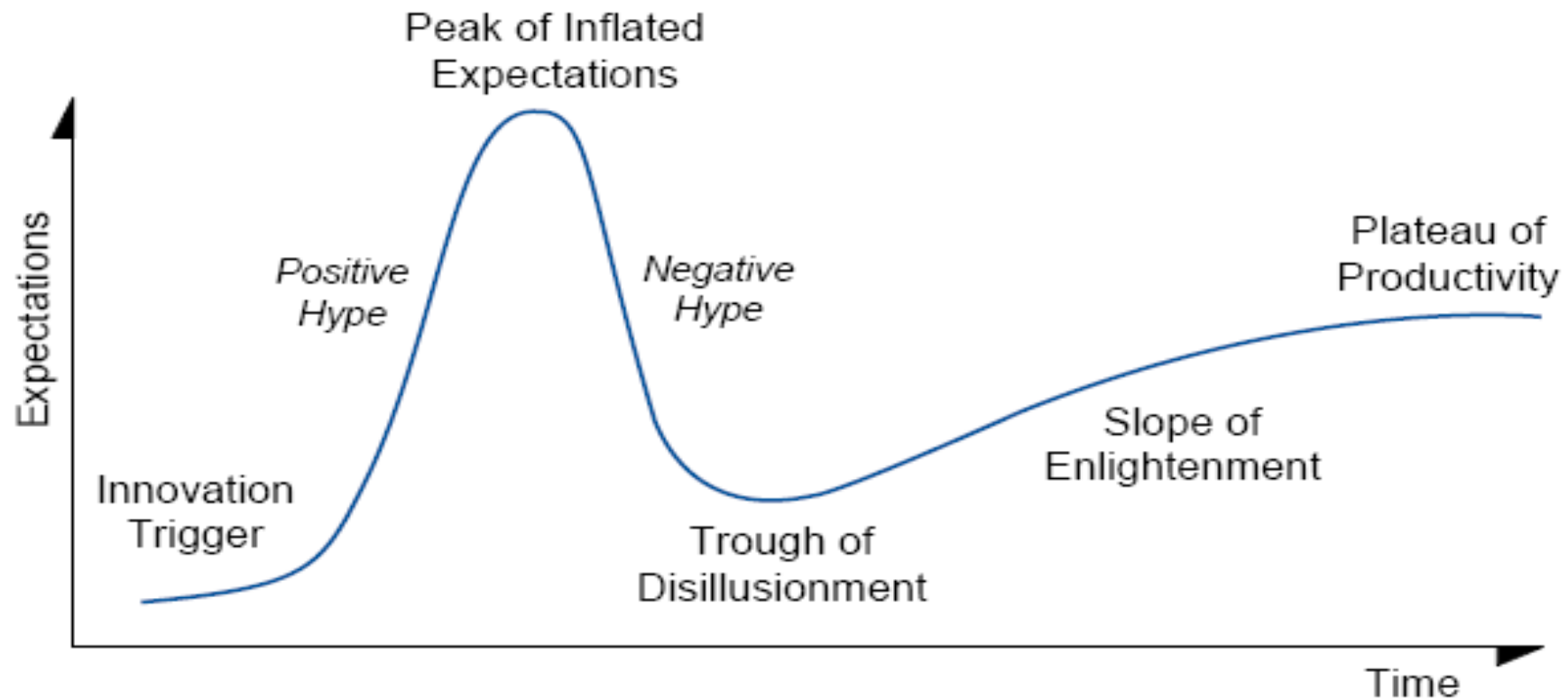
- ***A new concept: Osmotic Computing***



PART I

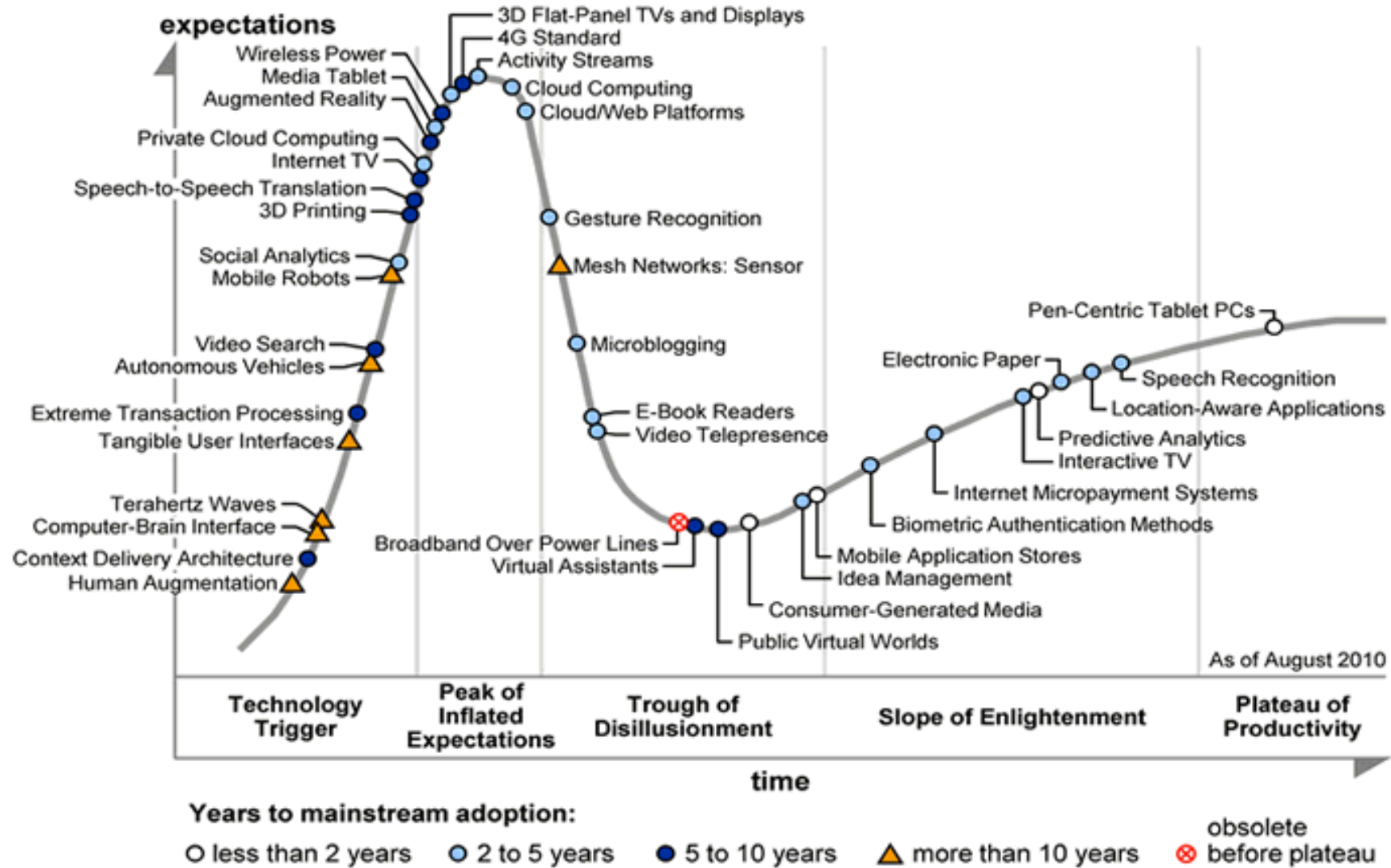


Simple graphical approach to get a better understanding: *Gartner hype cycle*





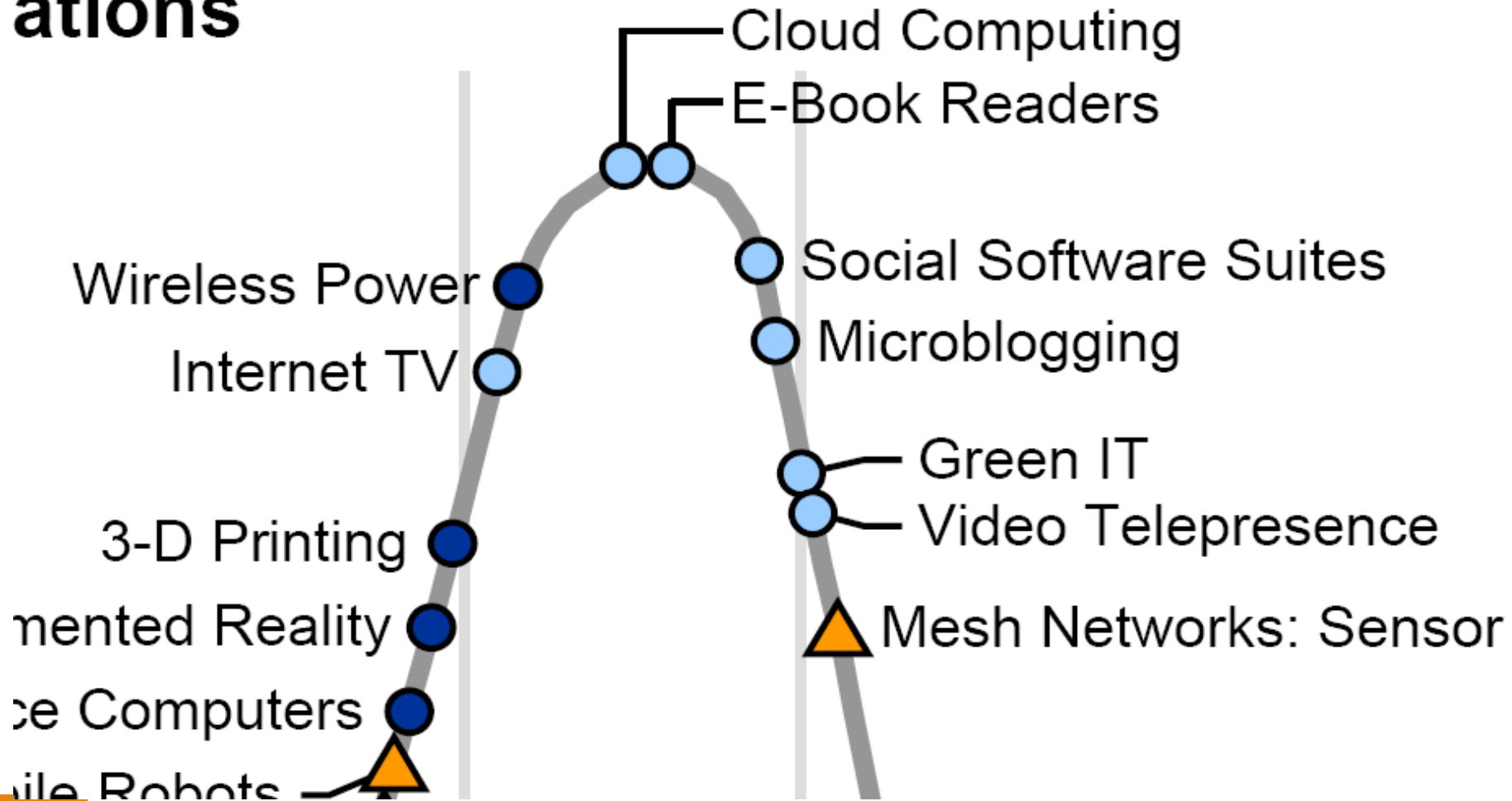
Hype Cycle: 2010





Inside the Hype Cycle: What's Hot and What's Not in 2009

ations

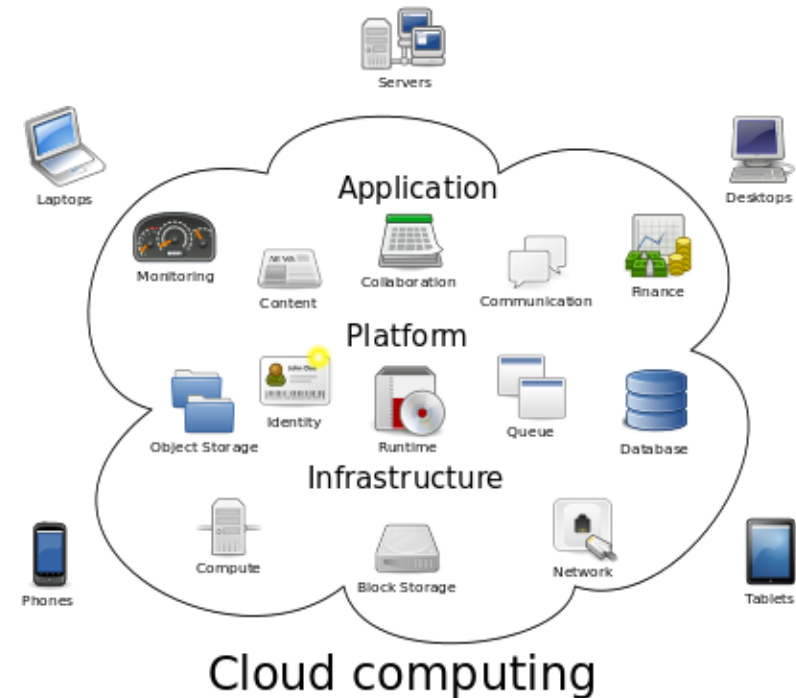




Cloud Computing on the wiki

Cloud computing

is a type of [Internet](#)-based computing that provides shared computer processing resources and data to computers and other devices on demand. It is a model for enabling ubiquitous, **on-demand access to a shared pool** of configurable computing resources (e.g., computer networks, servers, storage, applications and services), which can be rapidly provisioned and released with minimal management effort. Cloud **computing and storage** solutions provide users and enterprises with various capabilities to store and process their data in either privately owned, or third-party [data centers](#) that may be located far from the user—ranging in distance from across a city to across the world. Cloud computing relies on sharing of resources to achieve coherence and [economy of scale](#), similar to a utility (like the [electricity grid](#)) over an electricity network.





Why Cloud?

European official says new actions needed to “shock” digital world

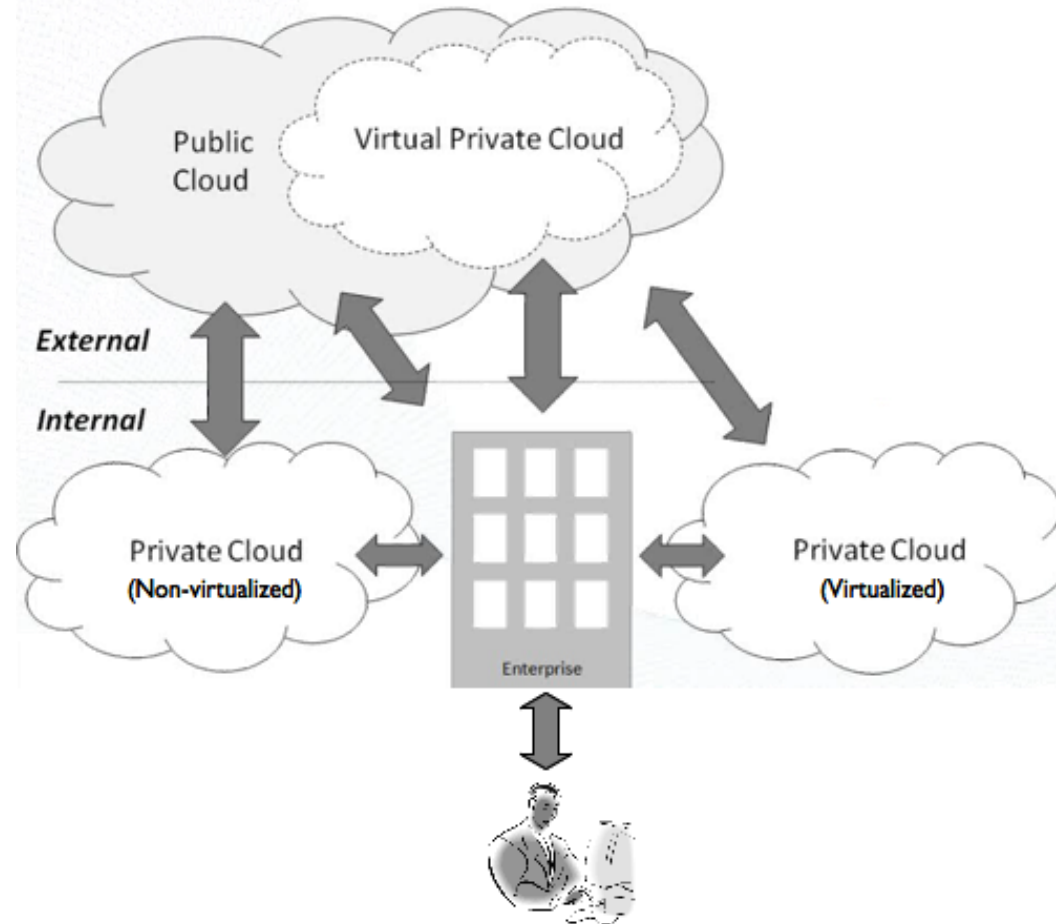
In France: “Les Gafa”
Google, Apple, Facebook and Amazon

These Silicon Valley companies have hide revenue and global reach, but they pay little corporate income taxes





Cloud computing





Cloud computing

Pay per Use (pay as you go)

Computing as an Utility

Click and Go



Explosive Combination





Explosive Combination

Smart Things

Humans





A few Examples

Uber

AirB&B

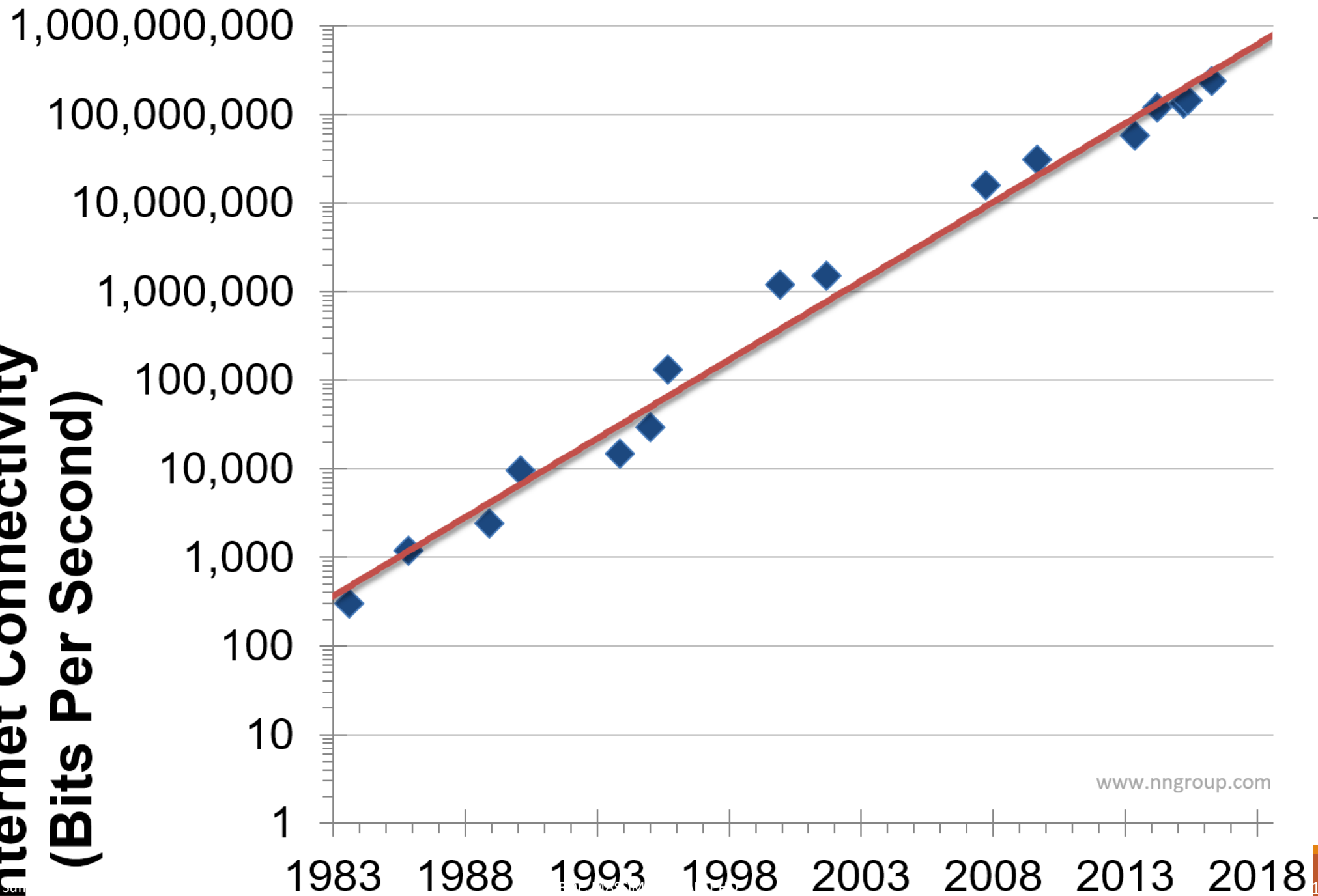
TripAdvisor

Facebook





Internet Connectivity (Bits Per Second)



www.nngroup.com



FOG Computing on the Wiki

Fog computing can be perceived both in large cloud systems and big data structures, making reference to the growing difficulties in accessing information objectively.

This results in a lack of quality of the obtained content.

The effects of fog computing on **cloud computing and big data systems may vary; yet, a common aspect that can be extracted is a limitation in accurate content distribution, an issue that has been tackled with the creation of metrics that attempt to improve accuracy.**

Fog networking consists of a **control plane and a data plane**. For example, on the data plane, fog computing enables computing services to reside at the edge of the network as opposed to servers in a data-center.



FOG Computing on the Wiki

Compared to cloud computing, fog computing emphasizes proximity to end-users and client objectives, dense geographical distribution and local resource pooling, latency reduction for quality of service (QoS) and edge analytics/stream mining, resulting in superior user-experience and redundancy in case of failure.

Fog networking supports the Internet of Things (IoT) concept, in which most of the devices used by humans on a daily basis will be connected to each other. Examples include phones, wearable health monitoring devices, connected vehicle and augmented reality using devices such as the Google Glass.



Open FOG Architecture:

1. Security;
2. Scalability;
3. Open - Autonomy;
4. Programmability;
5. RAS (Reliability, Availability, and Serviceability);
6. Agility; and
7. Hierarchy.



Edge Computing on the Wiki

Edge computing pushes applications, data and computing power (services) away from centralized points to the logical extremes of a network. Edge computing replicates fragments of information across distributed networks of web servers, which may be vast. As a topological paradigm, edge computing is also referred to as **mesh computing**, **peer-to-peer computing**, **[autonomic \(self-healing\) computing](#)**, **[grid computing](#)**, and other names implying non-centralized, nodeless availability.

To ensure acceptable performance of widely dispersed distributed services, large organizations typically implement edge computing by deploying Web [server farms](#) with [clustering](#). Previously available only to very large corporate and government organizations, technology advancement and cost reduction for large-scale implementations have made the technology available to small and medium-sized businesses.

The target end-user is any Internet client making use of commercial Internet application services.

Edge computing imposes certain limitations on the choices of technology platforms, applications or services, all of which need to be specifically developed or configured for edge computing.



Edge Computing on the Wiki

Edge computing has many advantages:

Edge application services significantly decrease the data volume that must be moved, the consequent traffic, and the distance the data must go, thereby reducing transmission costs, shrinking latency, and improving [quality of service](#) (QoS).

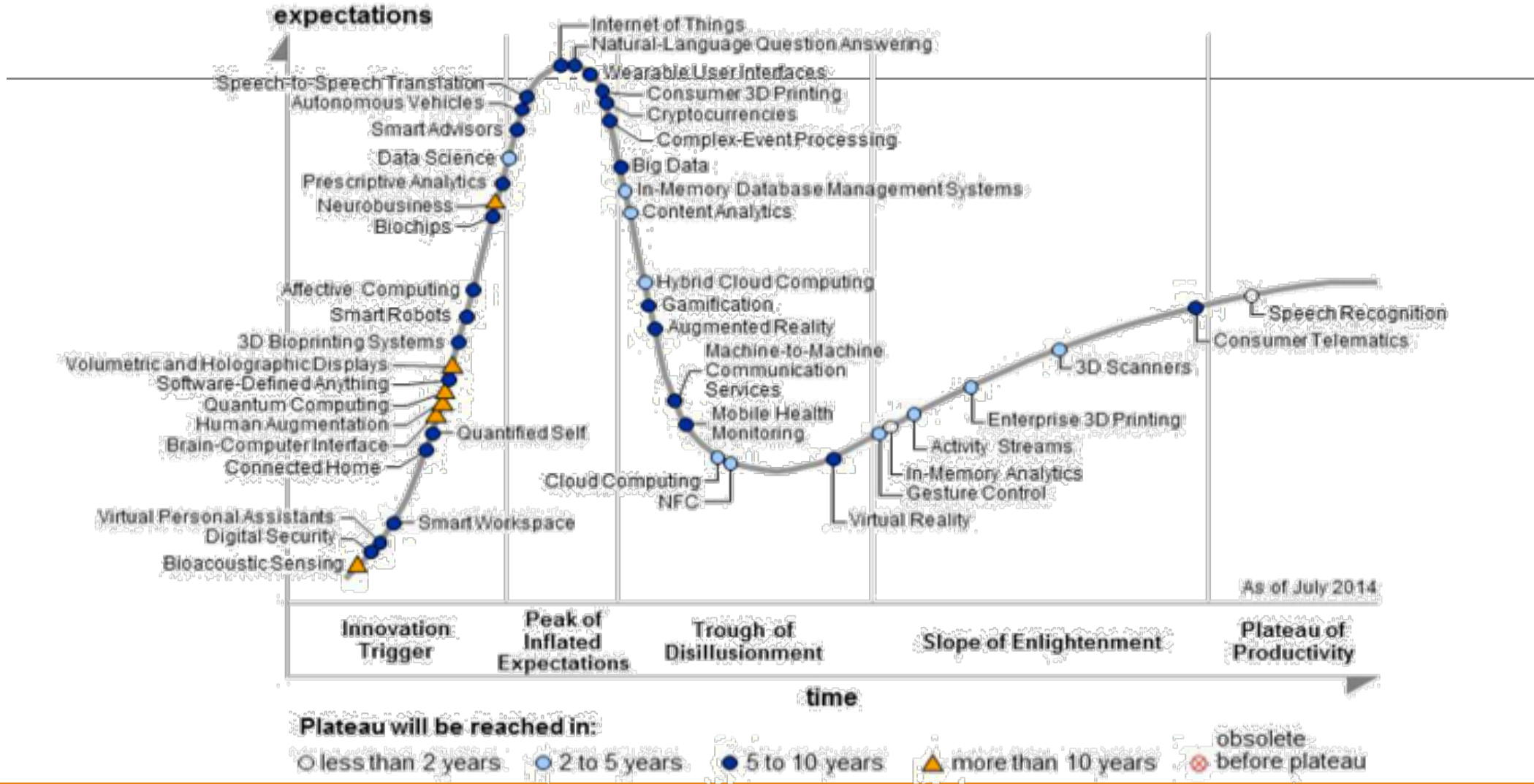
Edge computing eliminates, or at least de-emphasizes, the core computing environment, limiting or removing a major bottleneck and a potential point of failure.

Security is also improved as encrypted data moves further in, toward the network core. As it approaches the enterprise, the data is checked as it passes through protected firewalls and other security points, where viruses, compromised data, and active hackers can be caught early on.

Finally, the ability to "virtualize" (i.e., logically group CPU capabilities on an as-needed, real-time basis) extends [scalability](#). The edge computing market is generally based on a "**charge for network services**" **model**, and it could be argued that typical customers for edge services are organizations desiring linear scale of business application performance to the growth of, e.g., a subscriber base.

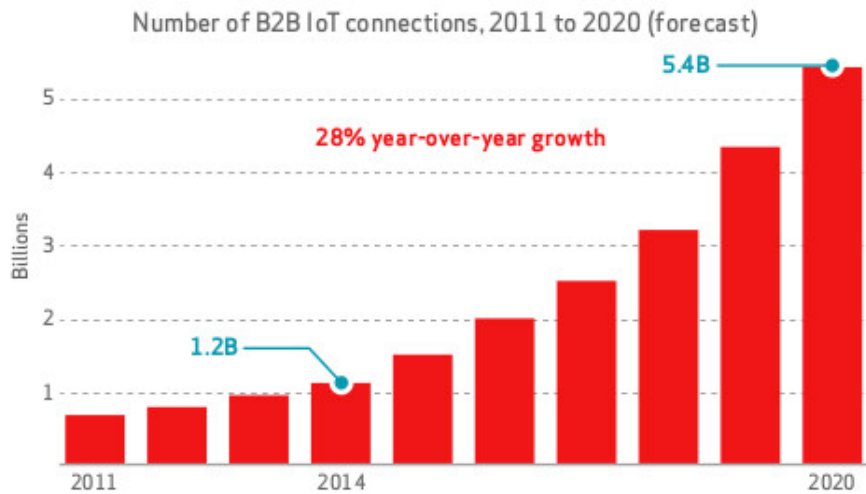


Hype Cycle: 2014





Internet of Things

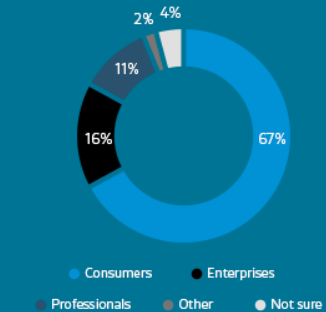


IIoT DEVELOPERS WANTED

Digital Trends Daily

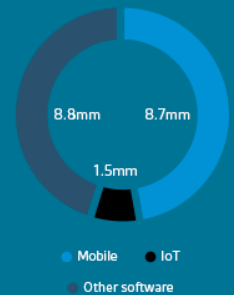
Experts have been throwing out numbers and estimates for awhile now about the future of the Industrial Internet of Things, usually focusing on projected total revenue or total number of connected devices. VisionMobile has released a report that estimates the actual people that will get involved in the massive growth of Internet of Things. This latest statistic details how there the workforce could sport up to 1.5 million IoT-specific developers by 2016, and an estimated 4.5 million by 2020. That's a huge influx of professionals into a relatively new industry, and there's no doubt that companies will have to adapt and nurture the tech-centric communities that currently exist within their walls to keep up with the growing demands and advancing technology of the IoT.

Developers Split by Primary Audience



Source: VisionMobile

Worldwide Developer Population

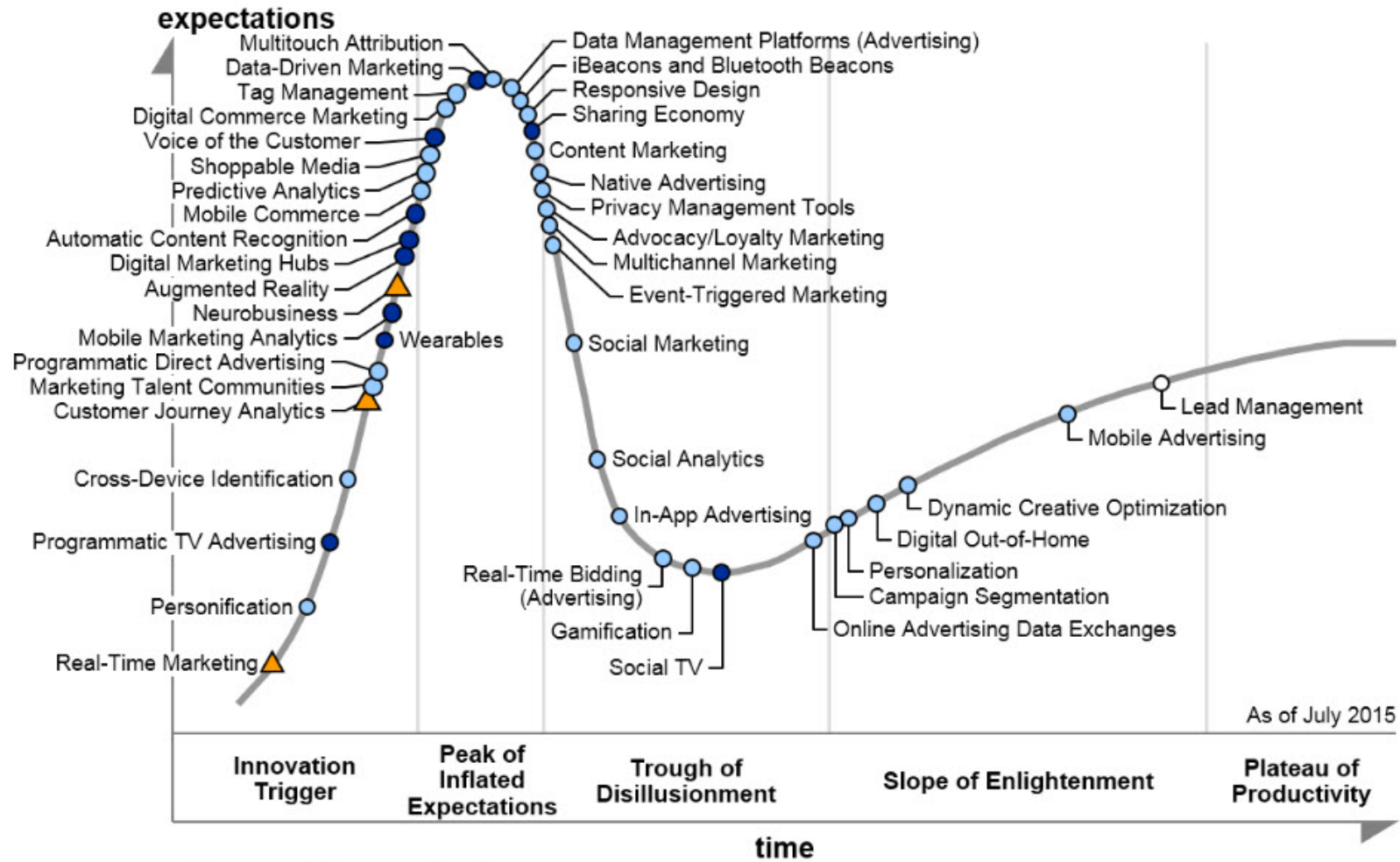


Source: Evans Data



© Centric Digital LLC 2015

Hype Cycle for Digital Marketing, 2015

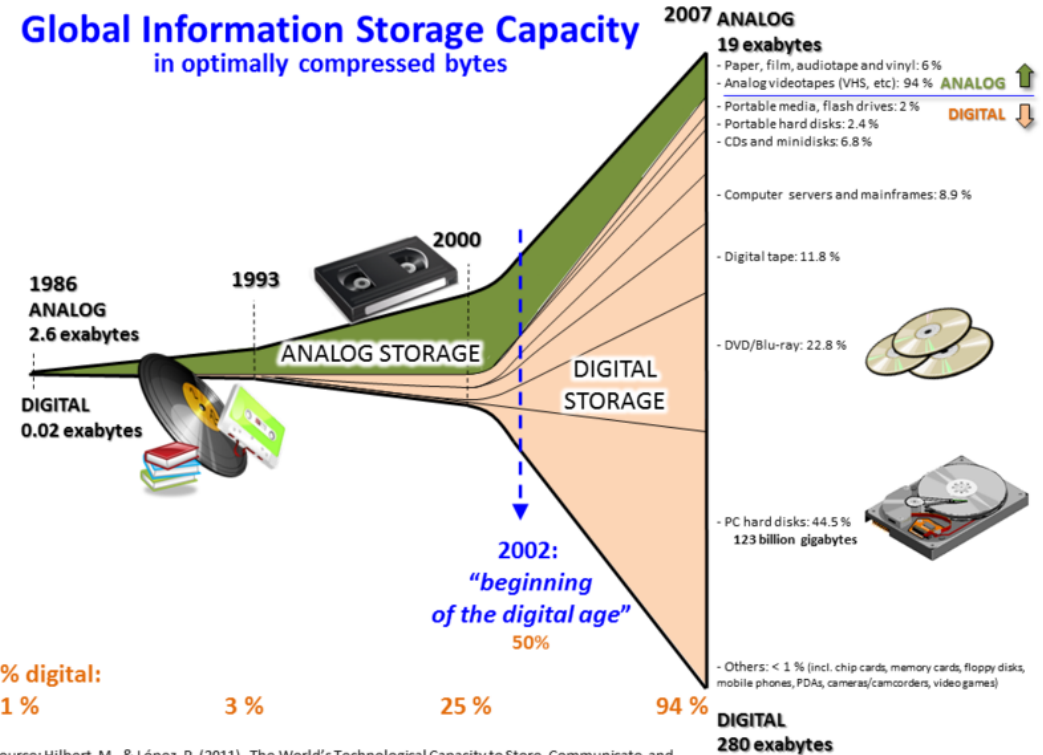




Data

Big Data Challenges:

include [capture](#), [storage](#), [analysis](#), [data curation](#), [search](#), [sharing](#), [transfer](#), [visualization](#), [querying](#), [updating](#) and [information privacy](#). The term "big data" often refers simply to the use of [predictive analytics](#), [user behavior analytics](#), or certain other advanced data analytics methods that extract value from data, and seldom to a particular size of data set.





Microservice on the Wiki

In a ***microservices*** architecture, **services should have a small [granularity](#) and the protocols should be lightweight.** A central microservices property that appears in multiple definitions is that services should be independently deployable. **The benefit of distributing different responsibilities of the system into different smaller services is that it enhances the [cohesion](#) and decreases the [coupling](#).** This makes it easier to change and add functions and qualities to the system at any time. It also allows the architecture of an individual service to emerge through continuous [refactoring](#), and hence reduces the need for a big up-front design and allows for releasing software early and continuously.



A Service Computing Manifesto: The Next 10 Years

Service computing is a key paradigm that offers cross-disciplinary computational abstractions, architectures, and technologies to support business services.



A Service Computing Manifesto: The Next 10 Years

Service computing has not yet realized its potential, because it has fallen short in addressing the challenges facing business services that go beyond technical aspects, especially in:

1. incorporating human concerns;
2. incorporating recent technological advances; and
3. addressing the effect of confusing standards.



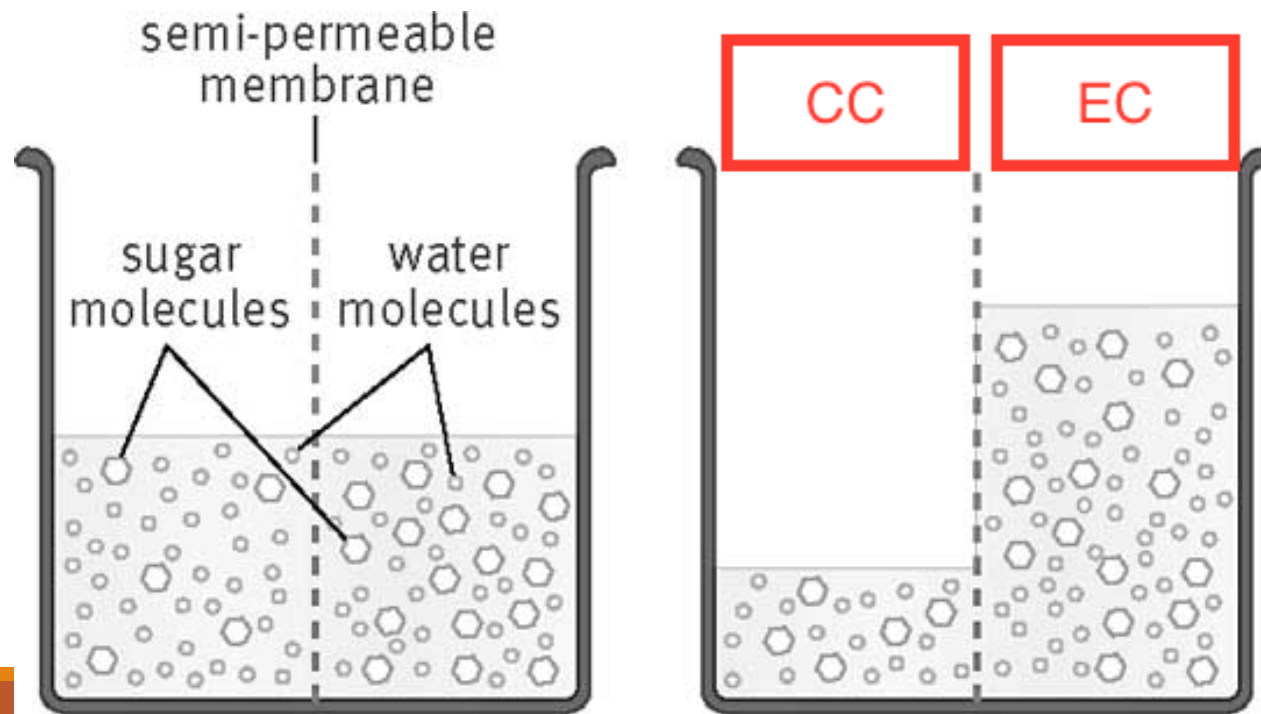
A Service Computing Manifesto: The Next 10 Years

A reboot of service computing is essential for it to play its crucial role in the era of cloud computing, big data, the Internet of Things (IoT), and social and mobile computing. It will require rethinking the service life cycle to better incorporate data and interaction semantics and elements of crowdsourcing reputation and trust to provide personalized, explicit value to stakeholders.



And the reboot is...

The Osmosis Process: Osmotic Computing

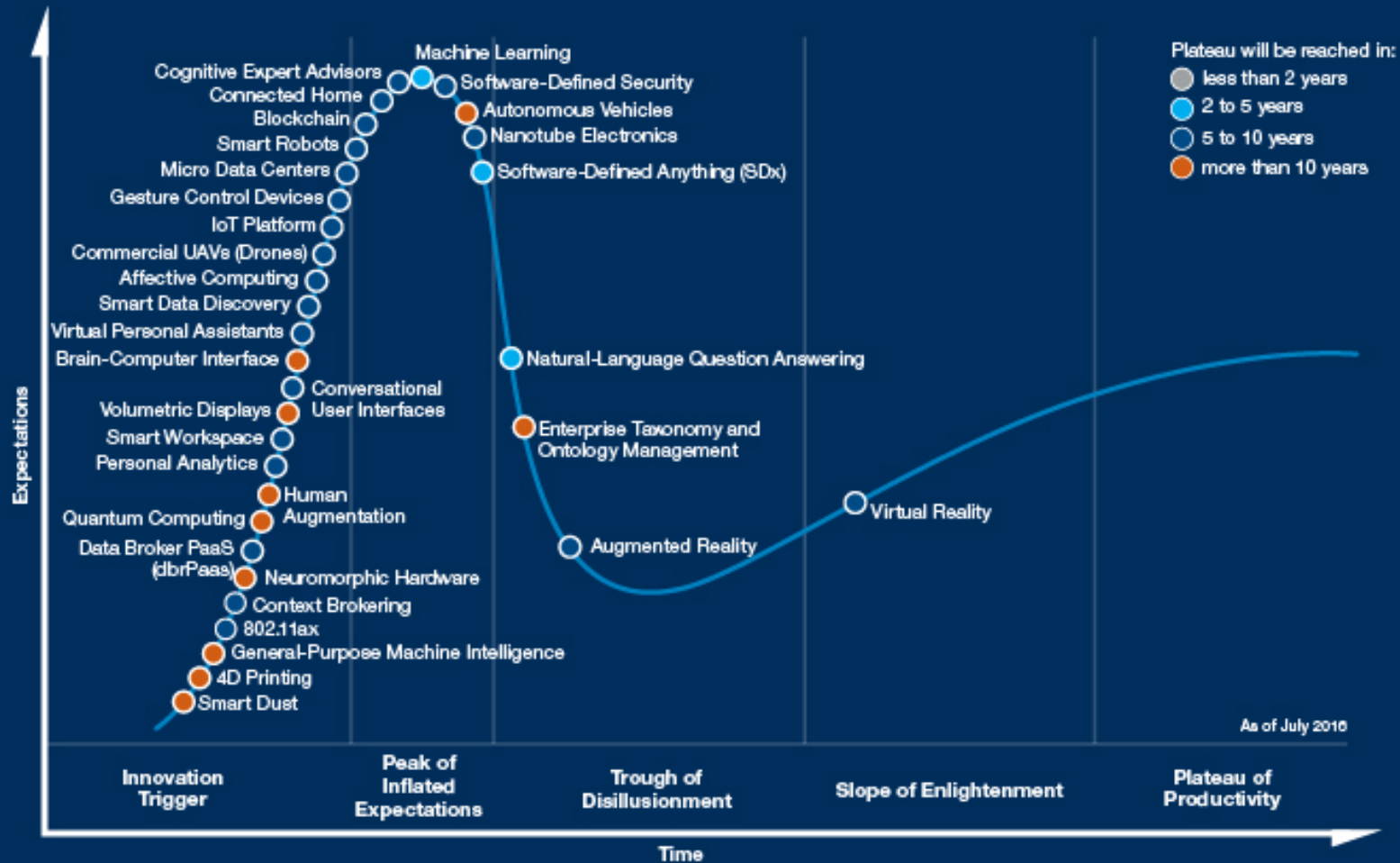




PART II



Gartner Hype Cycle for Emerging Technologies, 2016



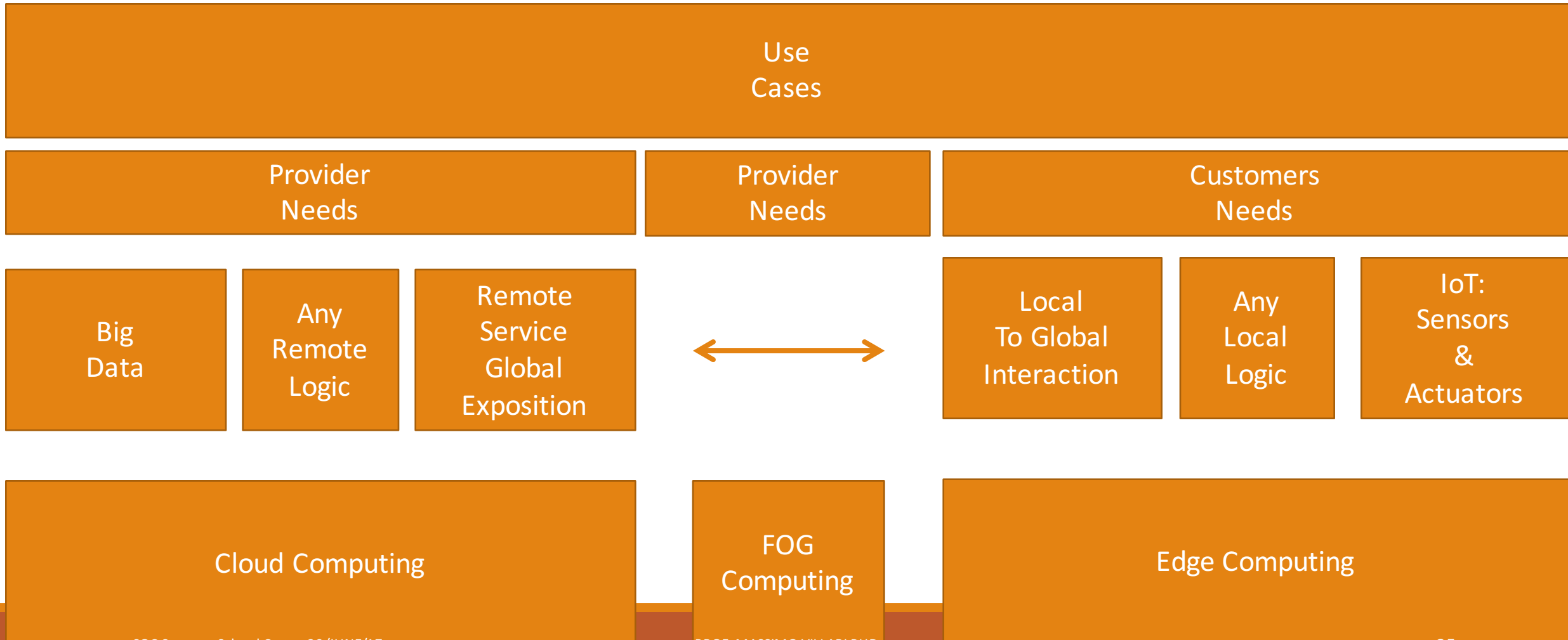
gartner.com/SmarterWithGartner

Source: Gartner
© 2016 Gartner, Inc. and/or its affiliates. All rights reserved.

Gartner.

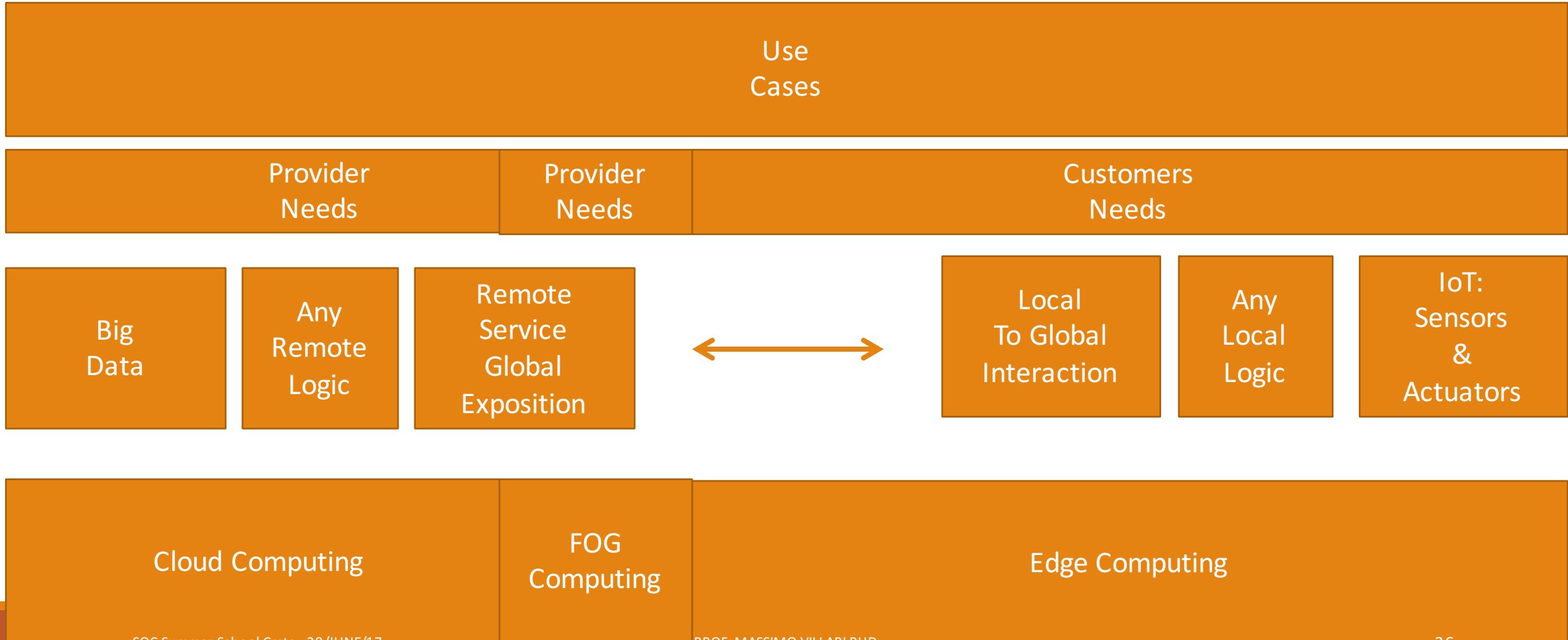


A Common BED



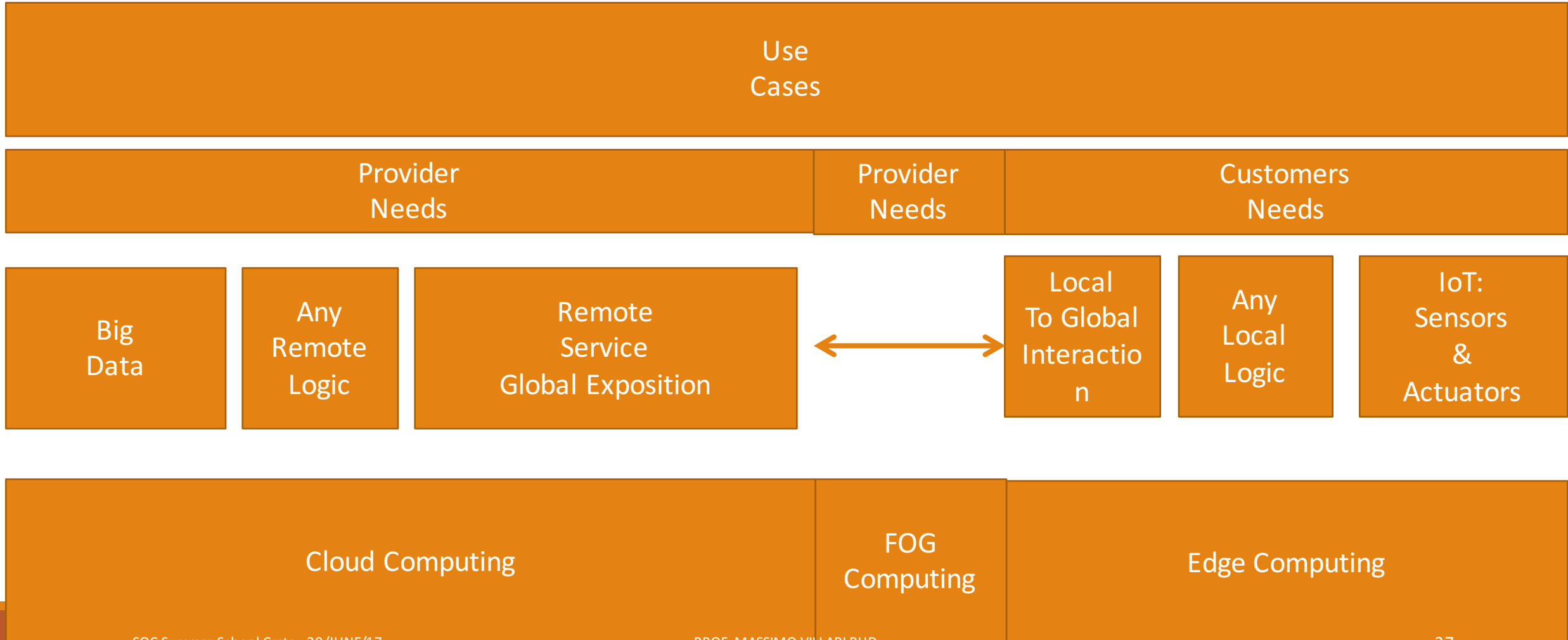


A Common BED



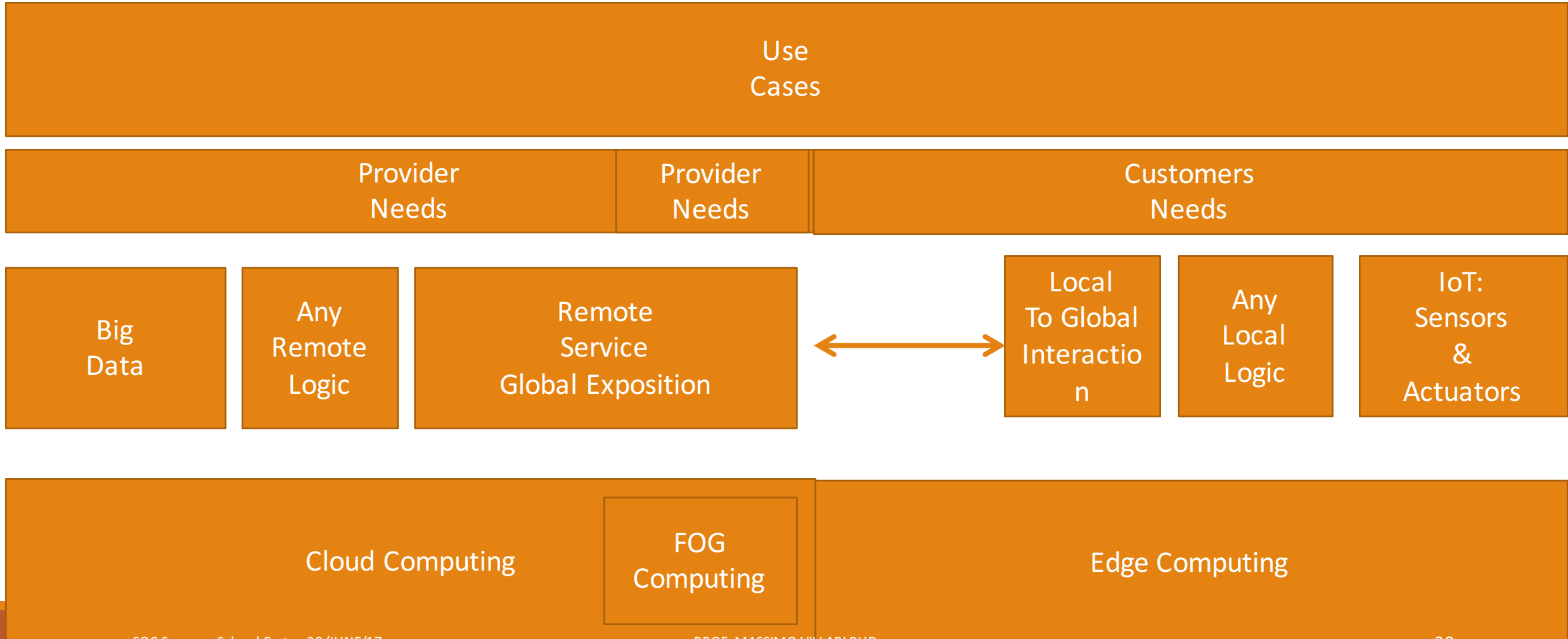


A Common BED



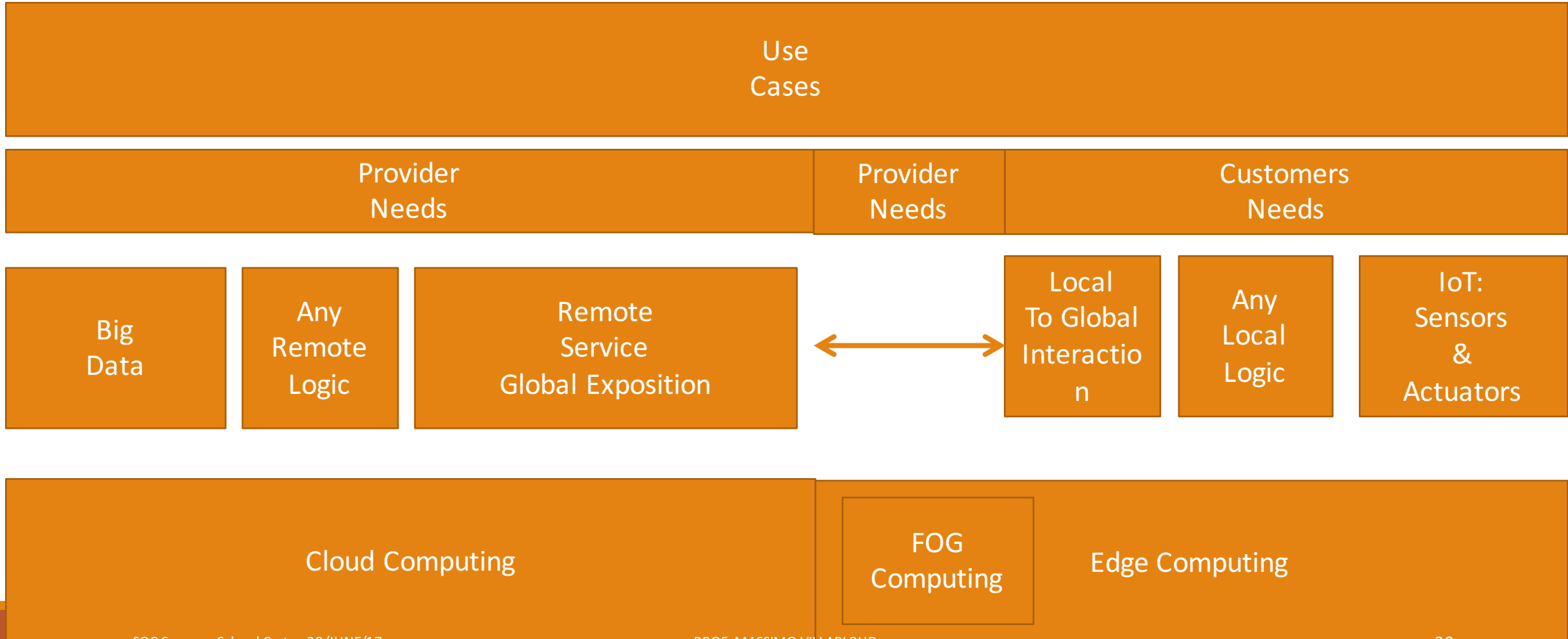


A Common BED



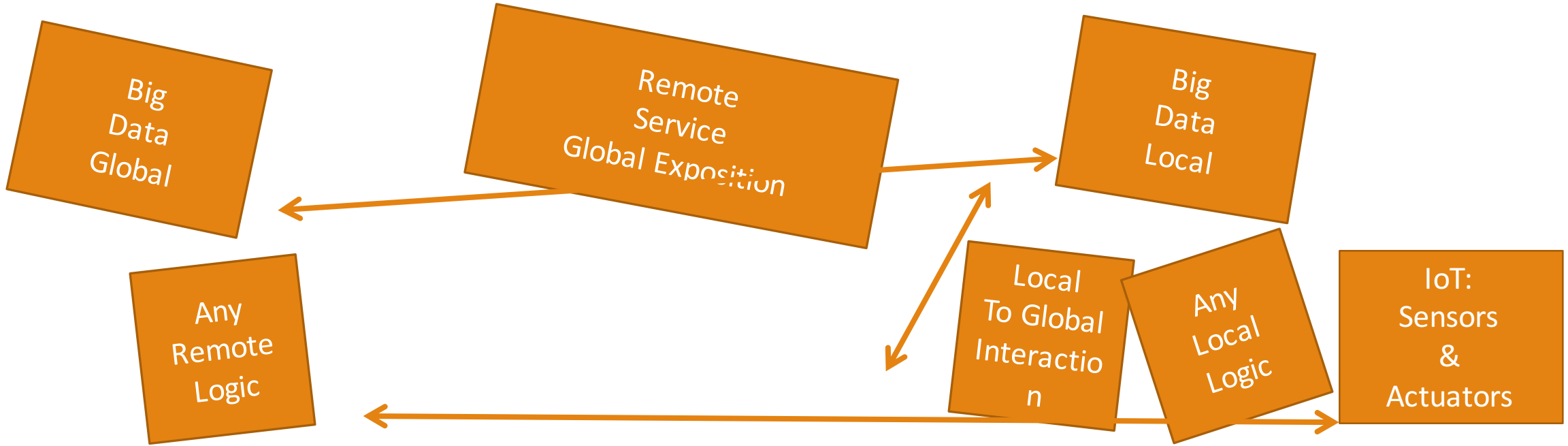


A Common BED



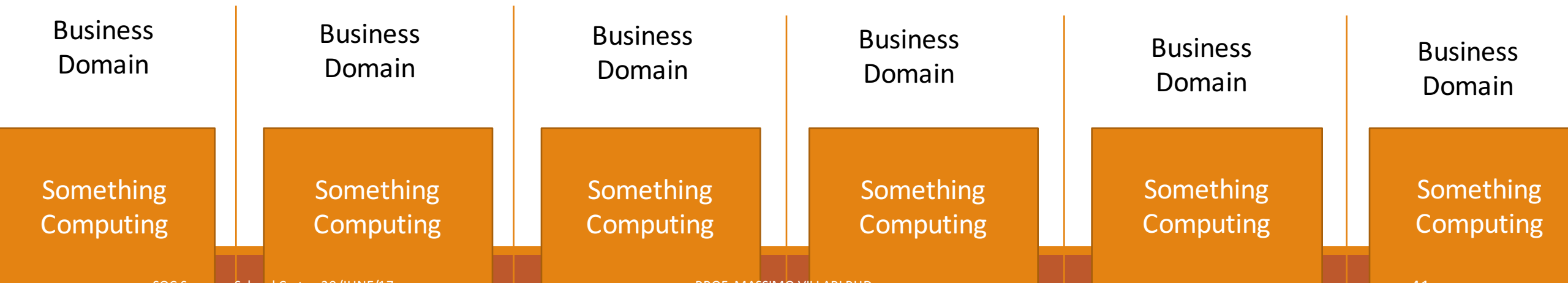


Use Cases



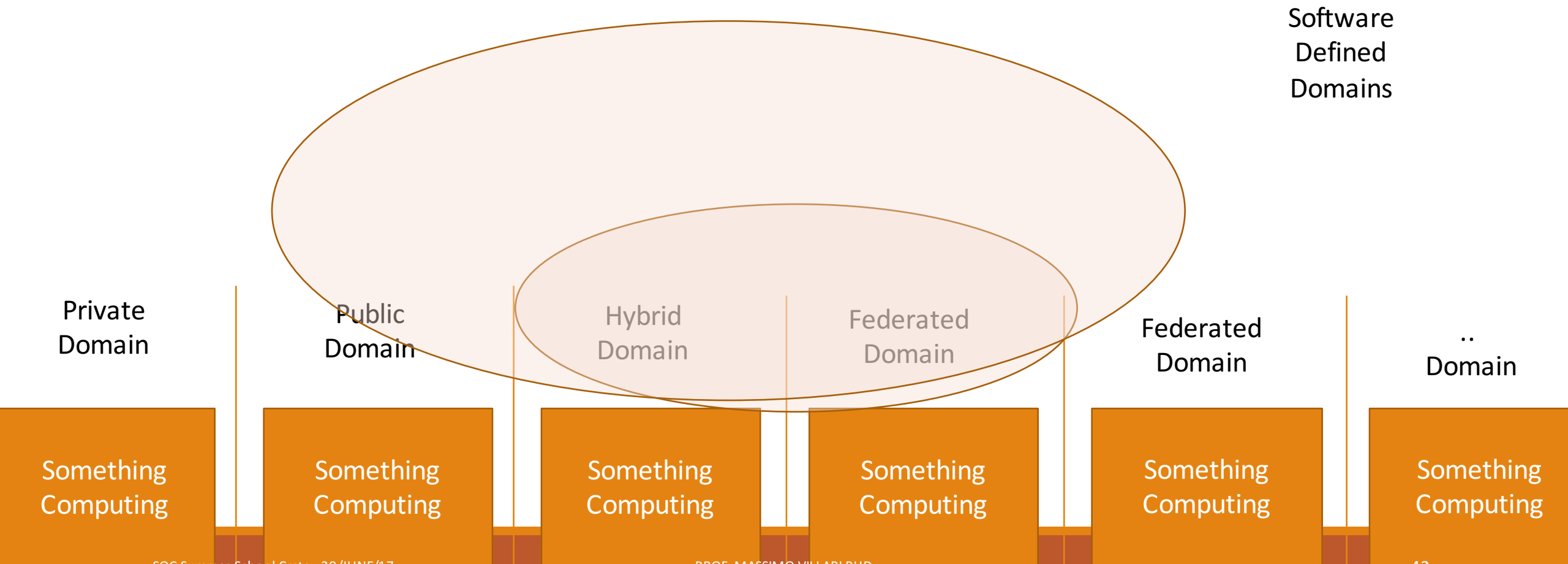


Something Computing -> Software Defined Anything



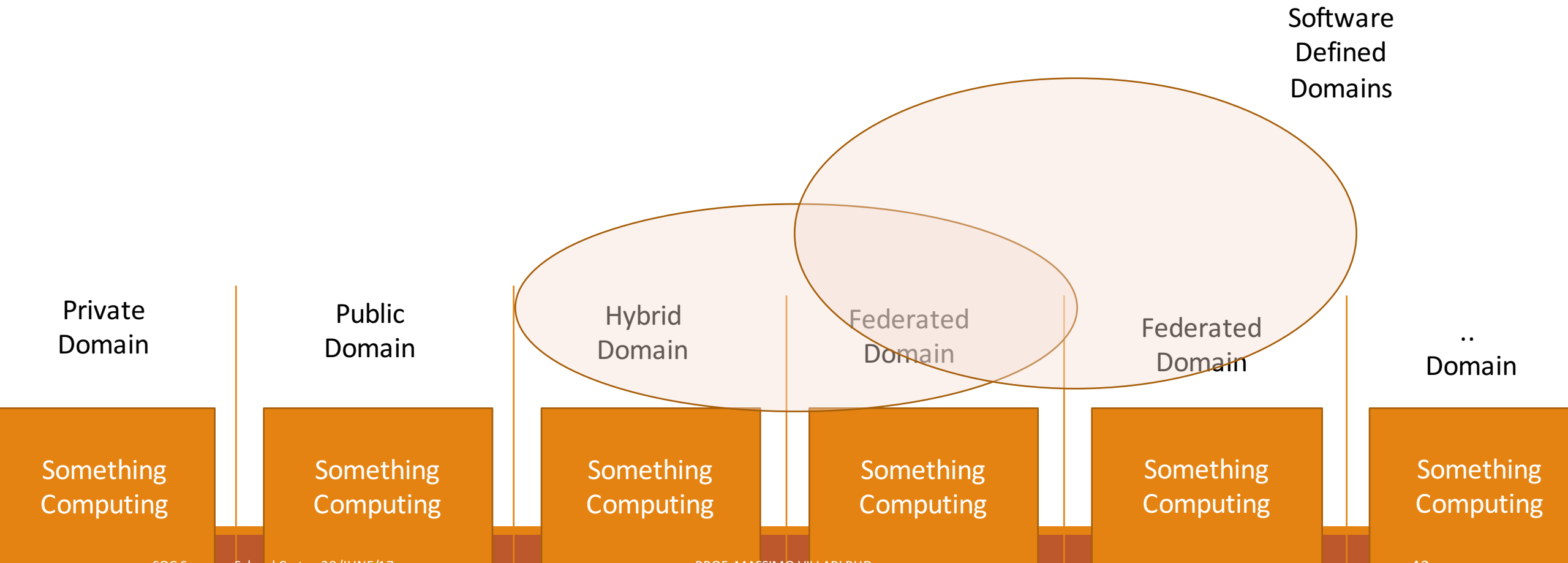


Something Computing -> Software Defined Domains





Something Computing -> Software Defined Domains





Something Computing -> Software Defined Anything

In concrete:

- Define a Set of APIs
- Migrability of Services (Micro is better)
- Bounds for Security (No Breaches-> if [yes] delete service)
- A priory agreements



Looking at Devices

Smarter?:

- Even more Abstracted
- Even more Capable (Proc. / Storage / Net.)
- Even more Multiple Cores



What devices?: MPU or MCU

MPUs offer more functionality and faster time to market, while MCUs provide a smaller, more cost-efficient solution.

MPU:

MPUs generally use **open-source operating systems such as Linux and Android**, although high-reliability applications may require proprietary operating systems like those from Green Hills Software and Wind River. These **operating systems include a library of drivers, for example for codecs, Ethernet, USB, etc.**

MCU:

An MCU delivers the right balance of cost, size, efficiency, and reliability. **Vendors leverage modular platforms and value pricing to create multiple part numbers from one die.** As a result, designers have a range of options in terms of memory size, pin count, and peripherals to develop the most cost-effective solution. **MCUs are sometimes referred to as SoCs because of the large amount of built-in functionality.** They typically integrate reset functionality, low-voltage inhibit, clock sources, interrupts, and on-chip regulators, to name a few.



What devices?: MPU or MCU

MPUs offer more functionality and faster time to market, while MCUs provide a smaller, more cost-efficient solution.

MPU:

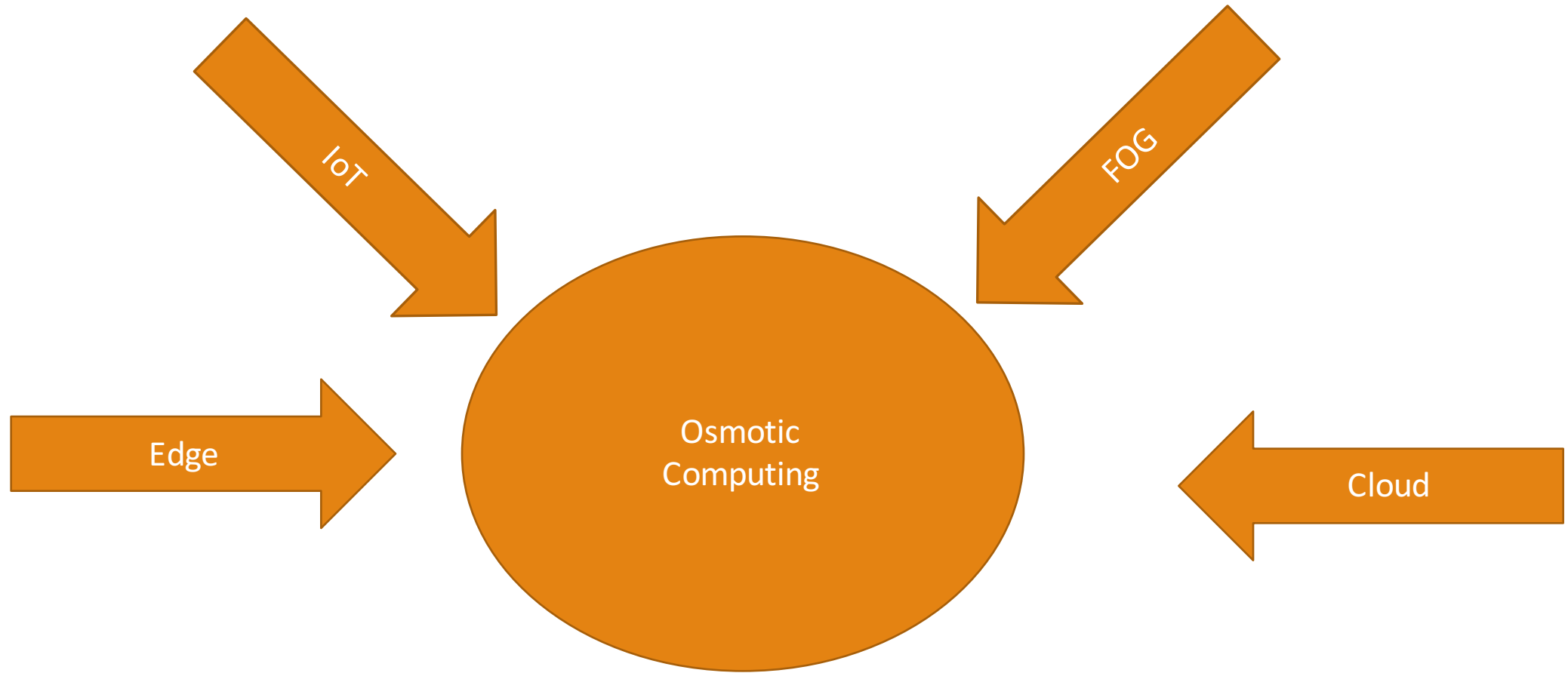
Most MPU providers create and maintain both Linux and Android releases for their evaluation boards, often supporting them with large software teams. **Full-fledged operating systems make the development of software much more simple and lend themselves to “plug and play” hardware.**

MCU:

The CPU cores used in **MCUs are designed explicitly to deliver the very low interrupt latency and deterministic code execution required for motor control in real-time applications like fans, compressors, washing machines, etc.** The Cortex-M4, for example, has a worst-case interrupt latency of 12 cycles and uses a three-stage pipeline.



Looking at the Overall Picture: E2E



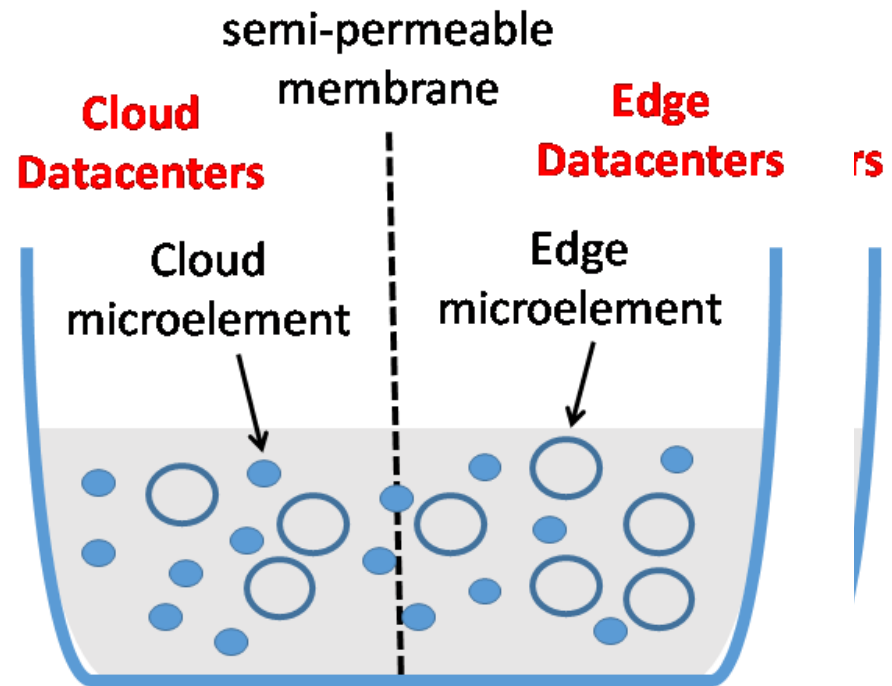
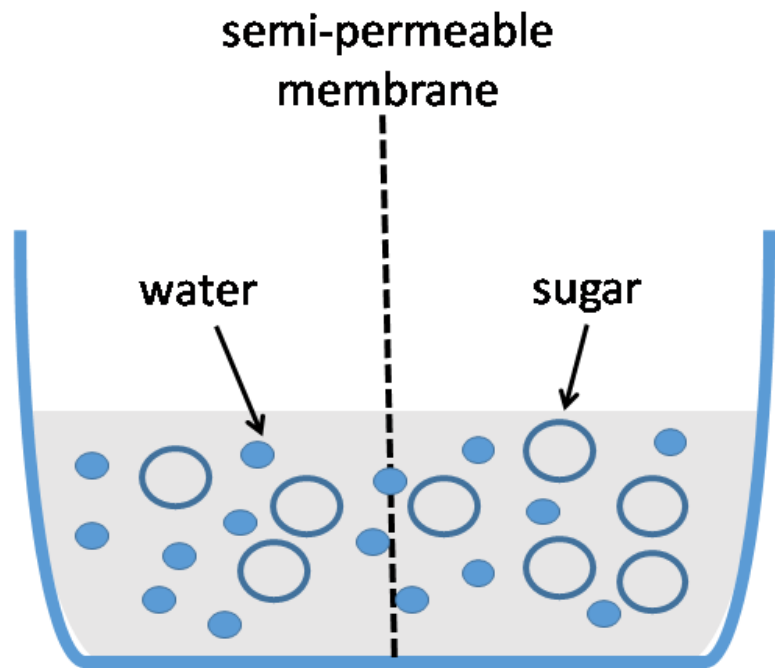


Time for Cooking all Ingredients Together



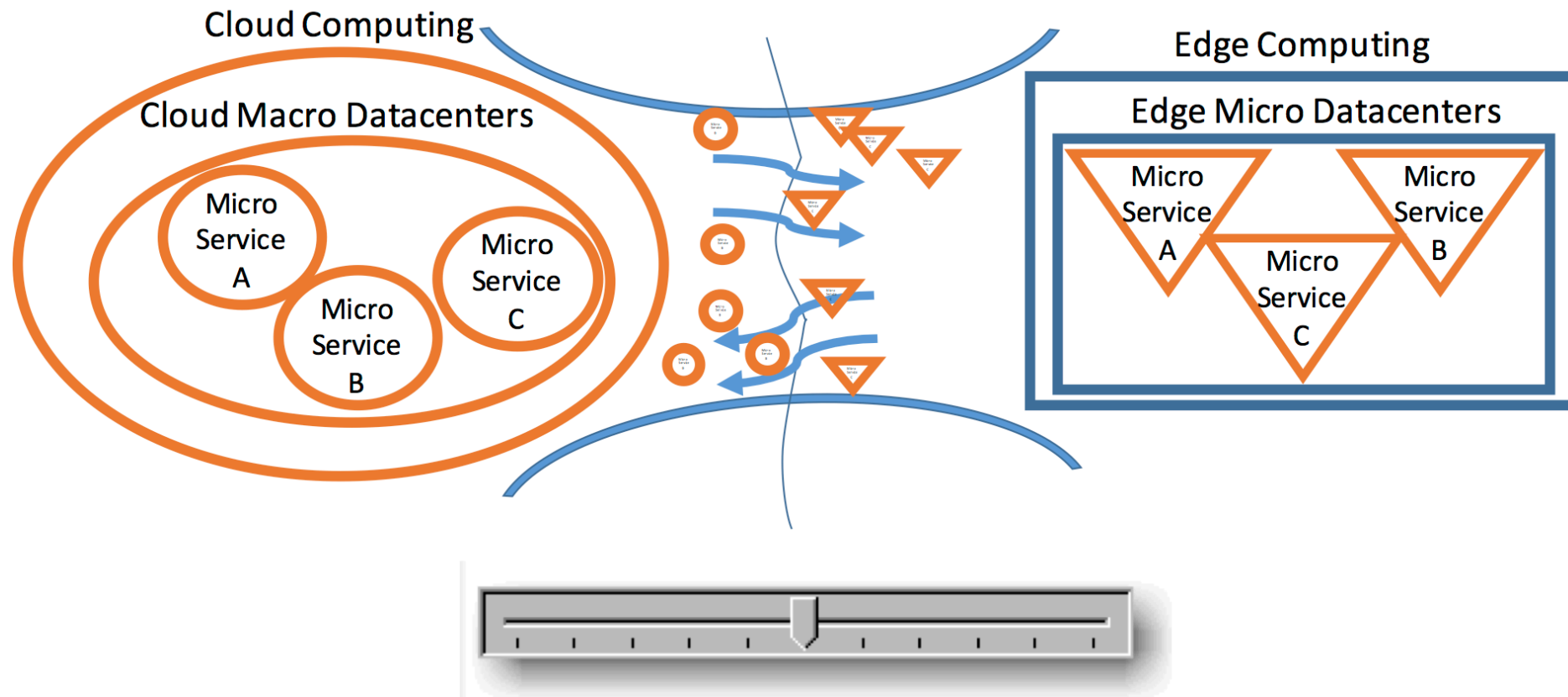


Osmosis Process



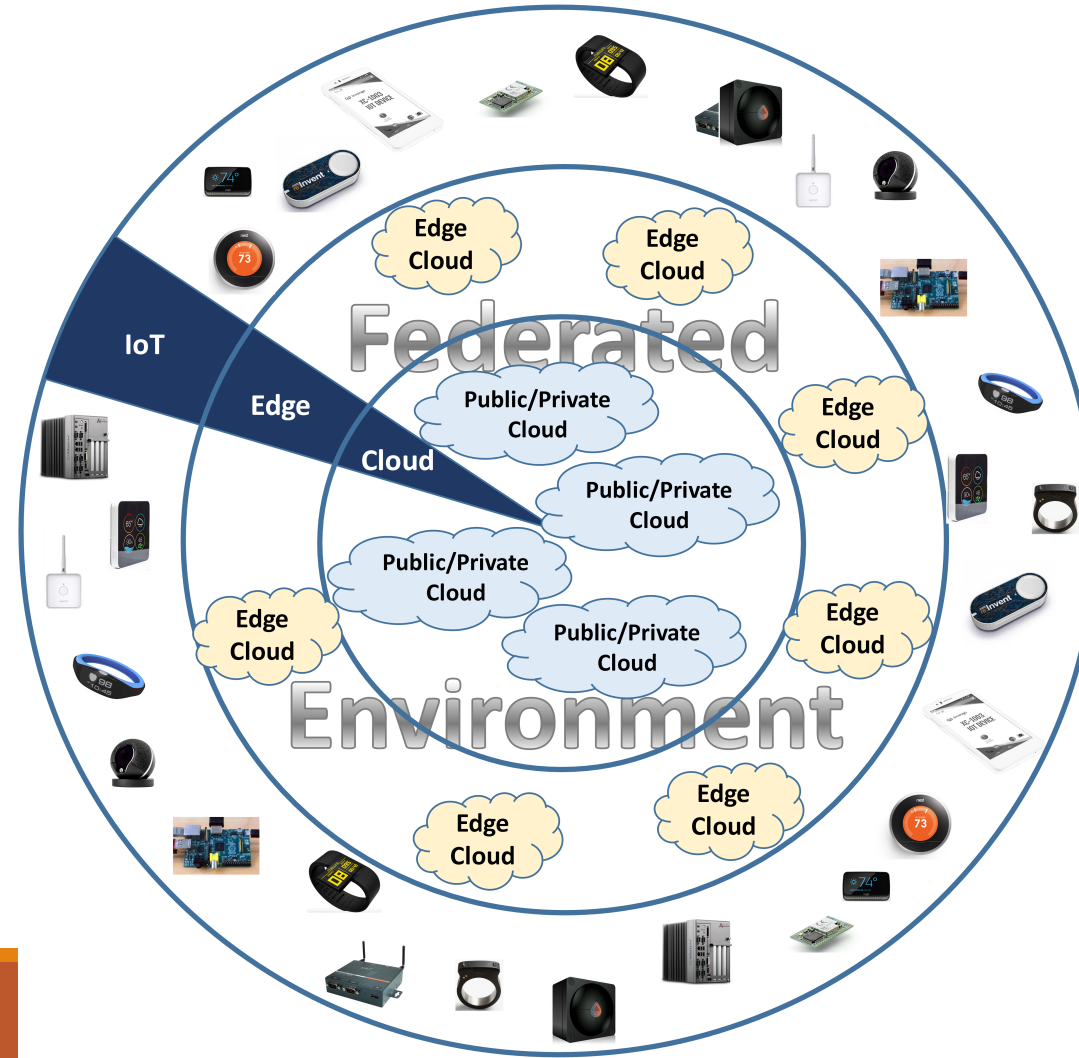


Osmotic Computing Concept



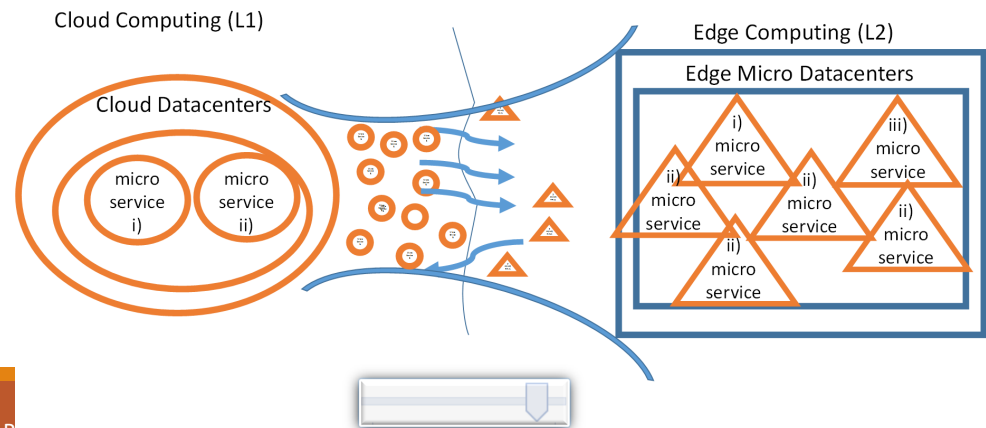
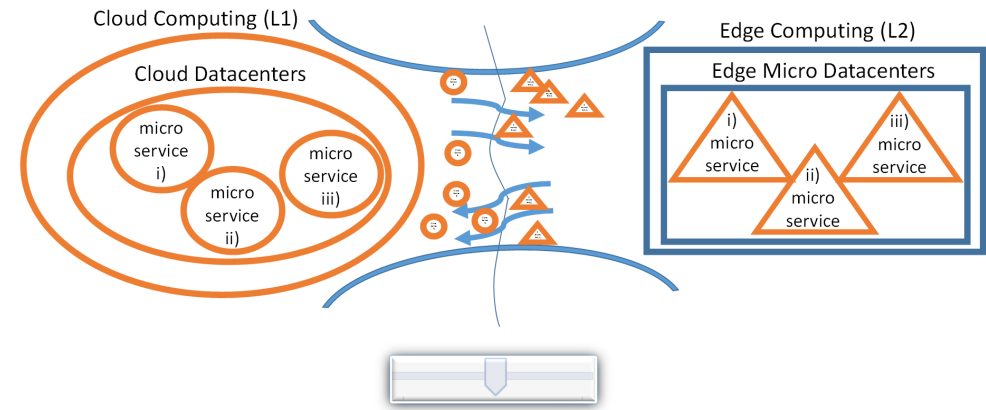
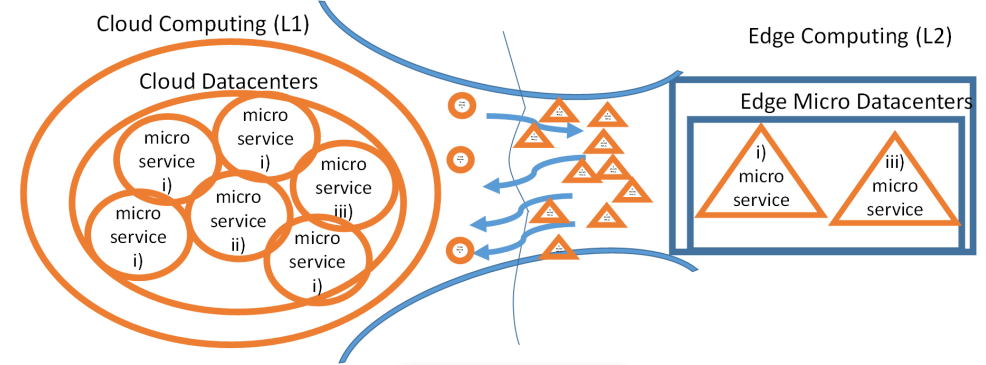


Osmotic Computing Federated Scenario



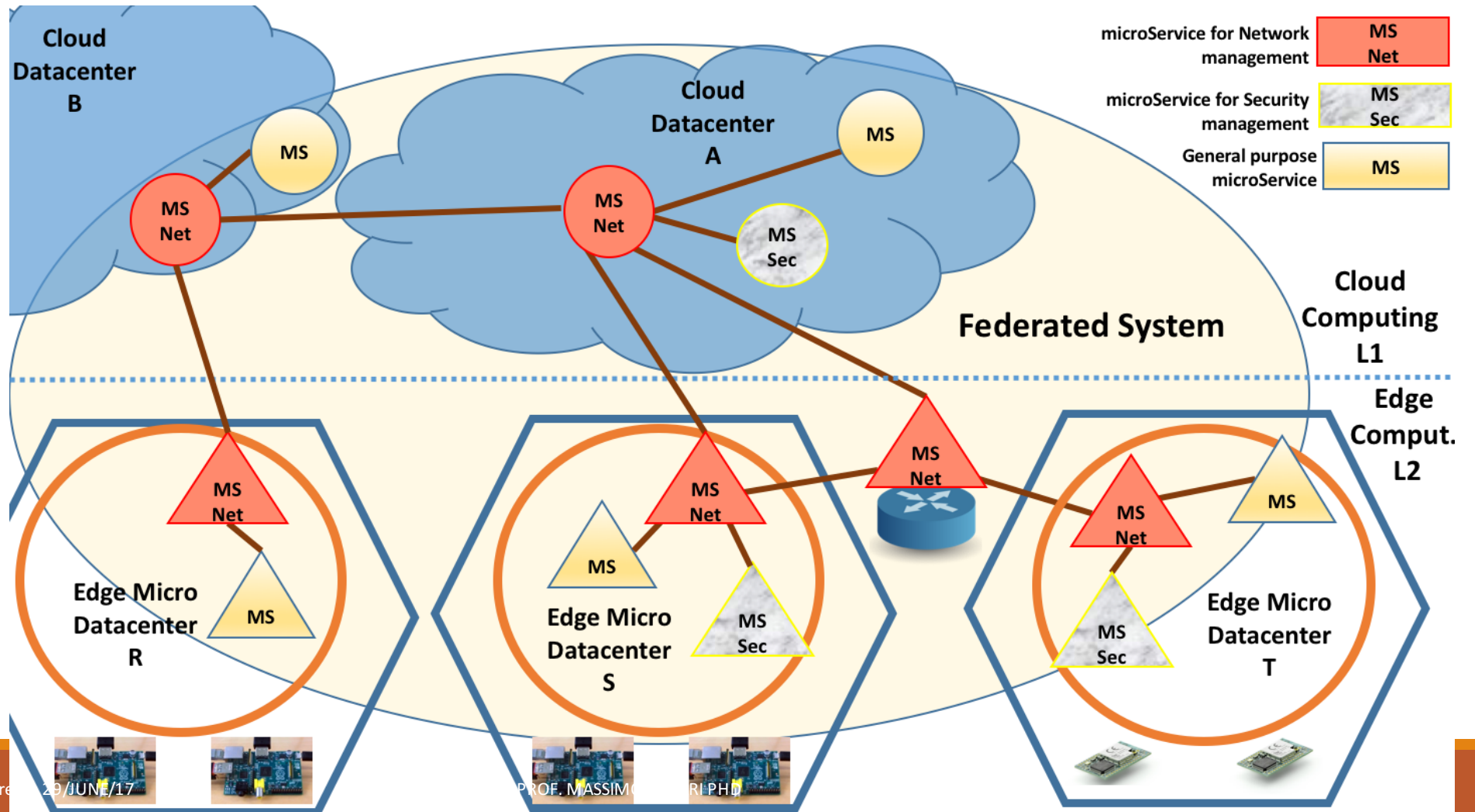


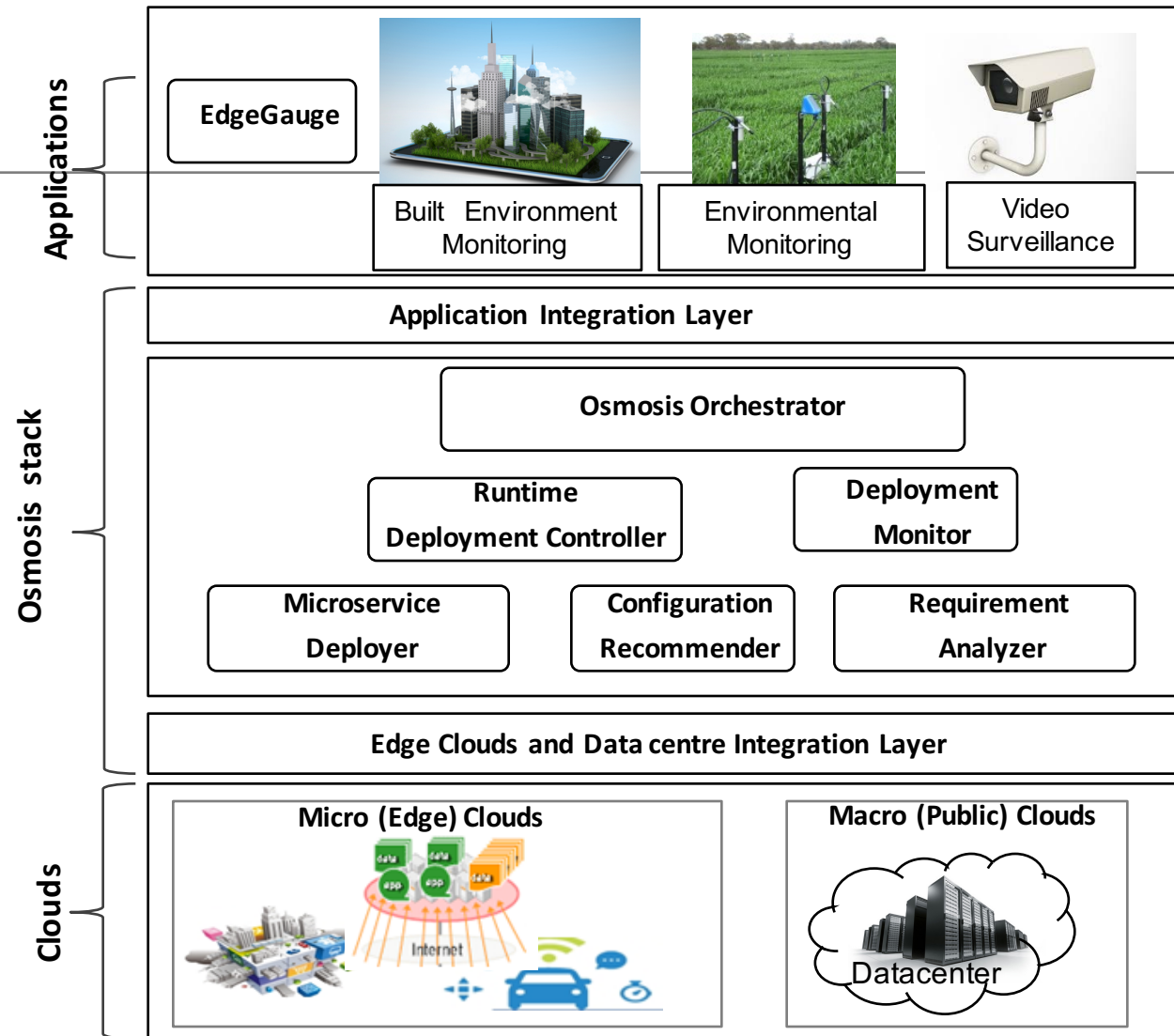
Osmotic Computing





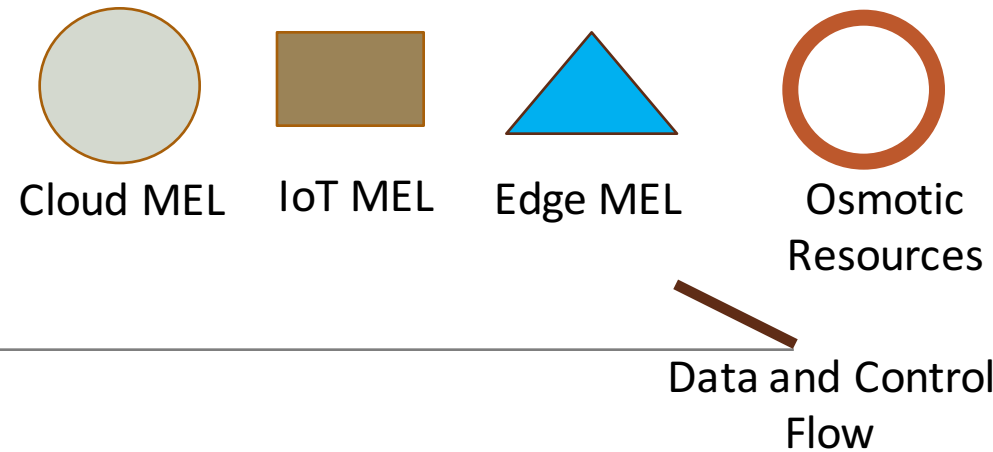
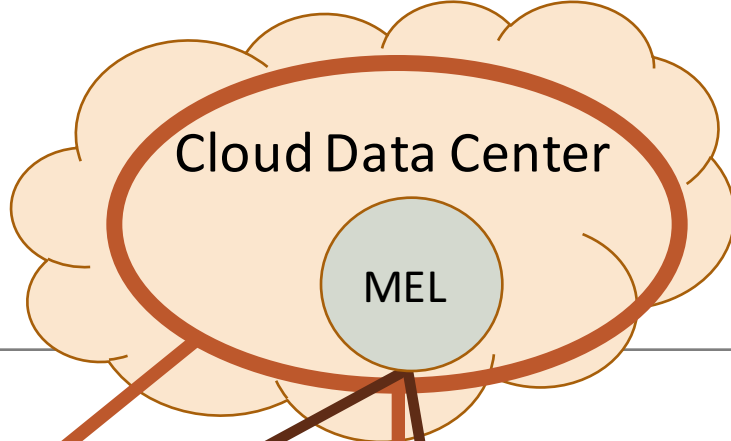
Osmotic Computing



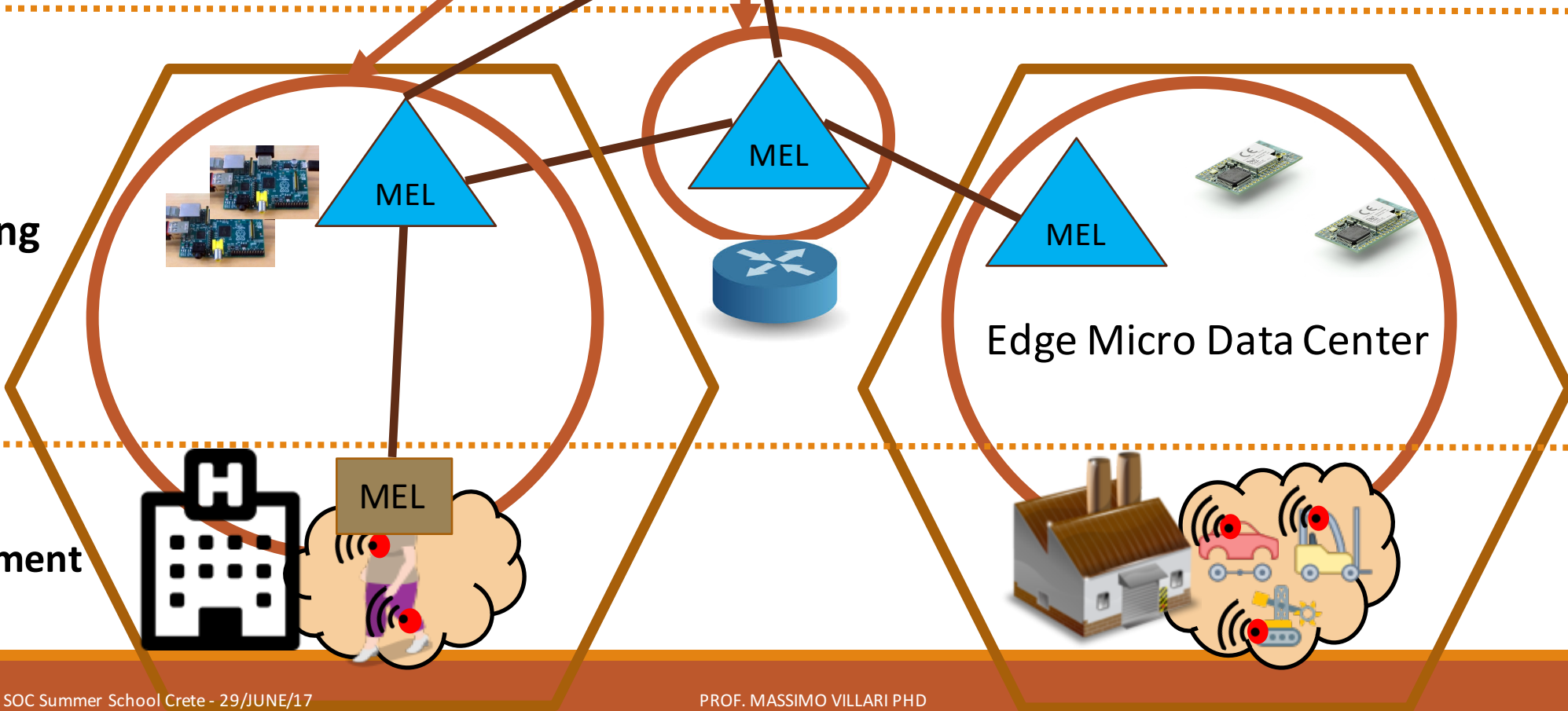




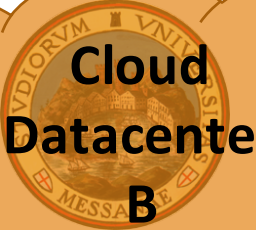
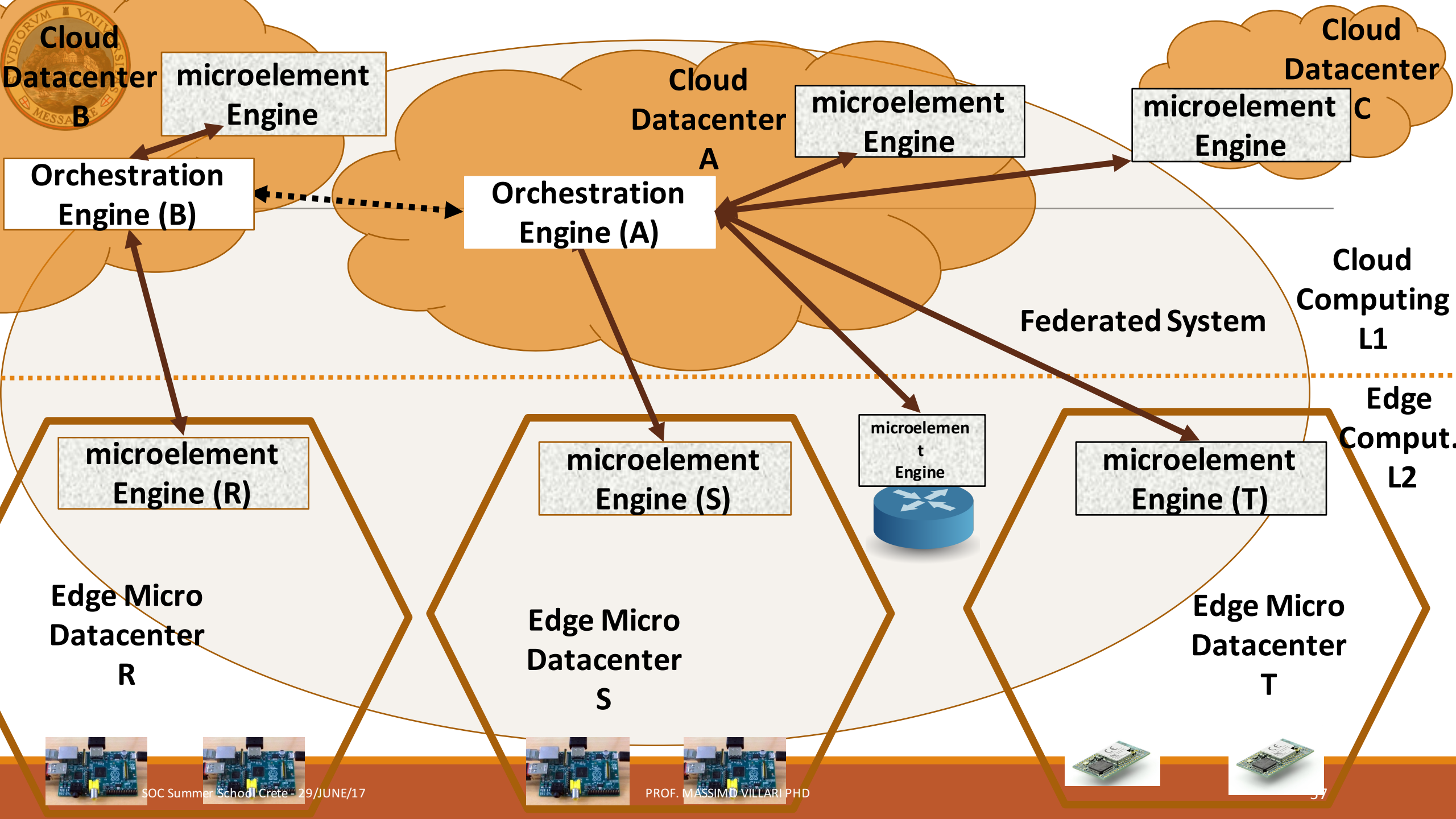
Cloud Computing L1



Edge Computing L2



IoT environment L3





Cloud Computing (L1)

Cloud Datacenters

micro element i)

micro element ii)

micro element iii)

Micro element b

Edge Computing (L2)

Edge Micro Datacenters

i) micro element

ii) micro element

iii) micro element

IoT (L3)

i) micro element

ii) micro element

iii) micro element



Cloud Computing (L1)

Cloud Datacenters

micro element i)

micro element ii)

micro element iii)

Micro element B

Software Defined MEMbrane (B)

Edge Computing (L2)

Edge Micro Datacenters

i) micro element

ii) micro element

iii) micro element

IoT (L3)

i) micro element

ii) micro element

iii) micro element

IoT (L3)

i) micro element

ii) micro element

iii) micro element

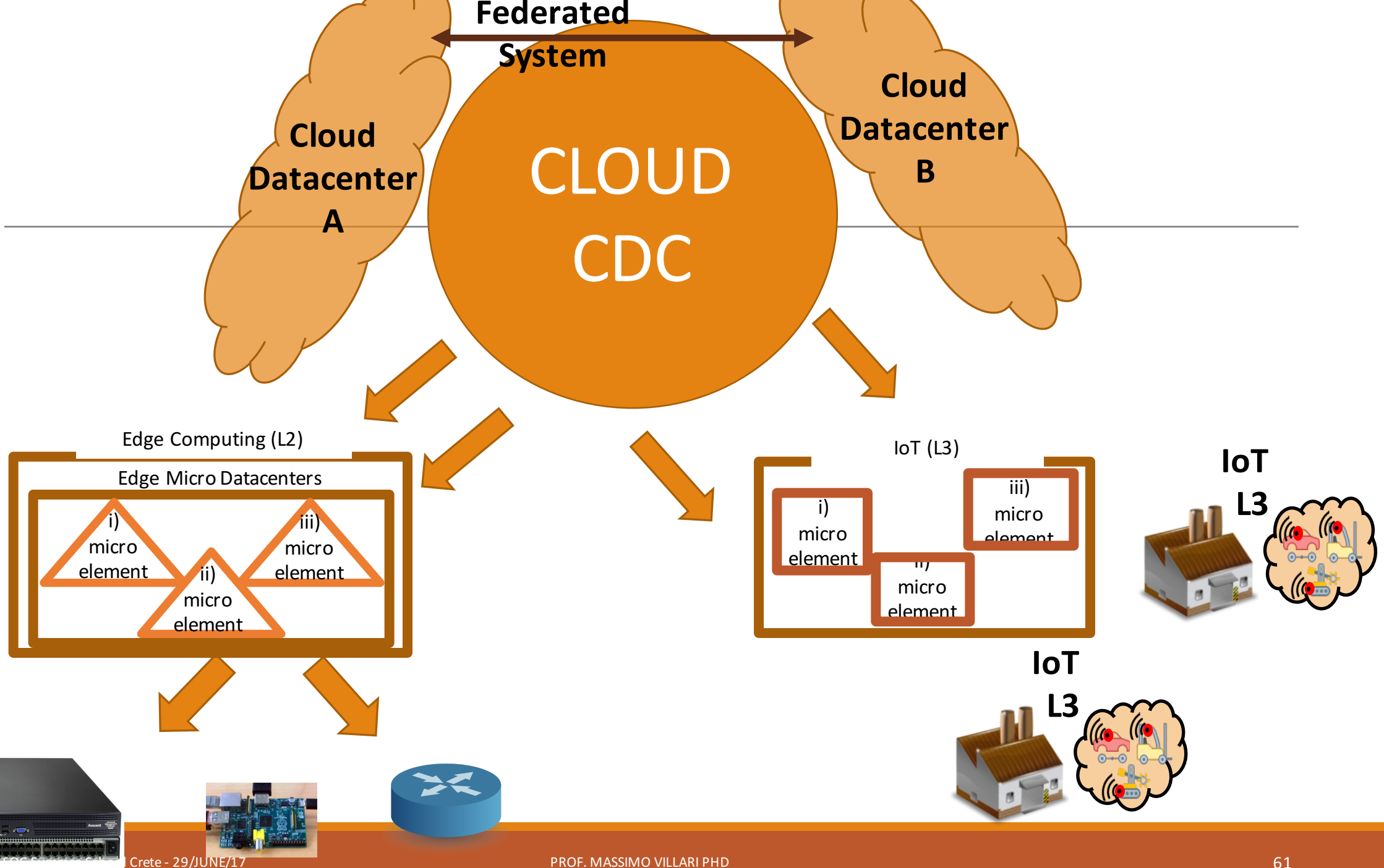
Software Defined MEMbrane (A)

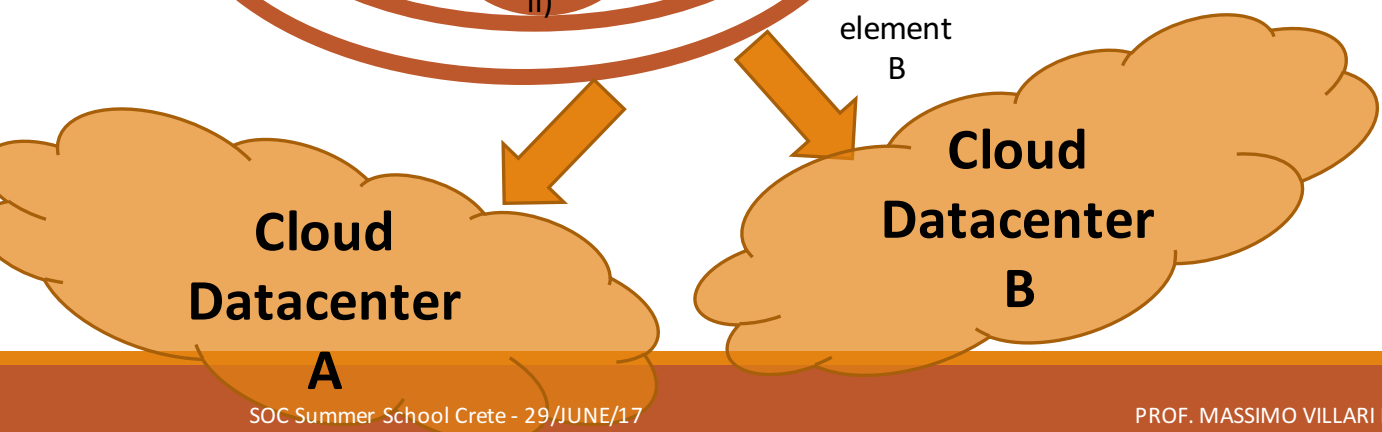
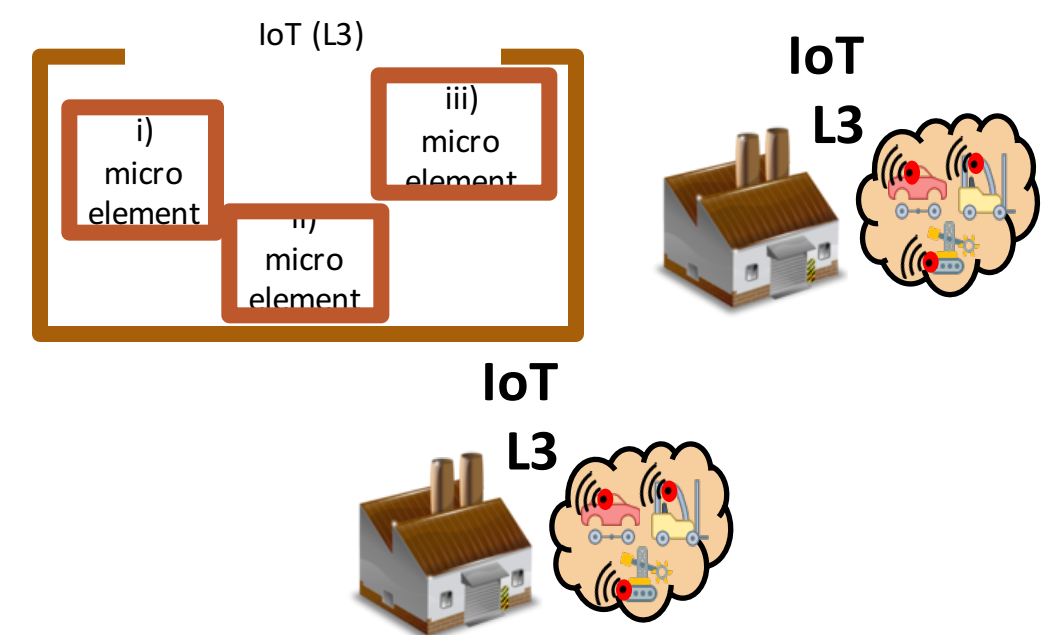
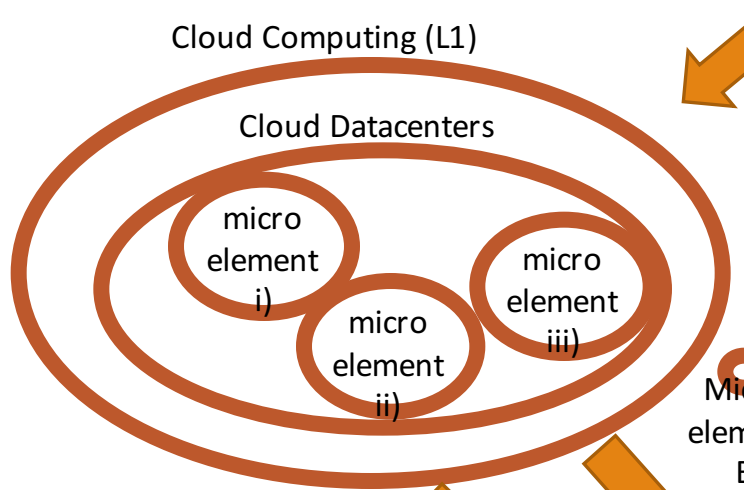
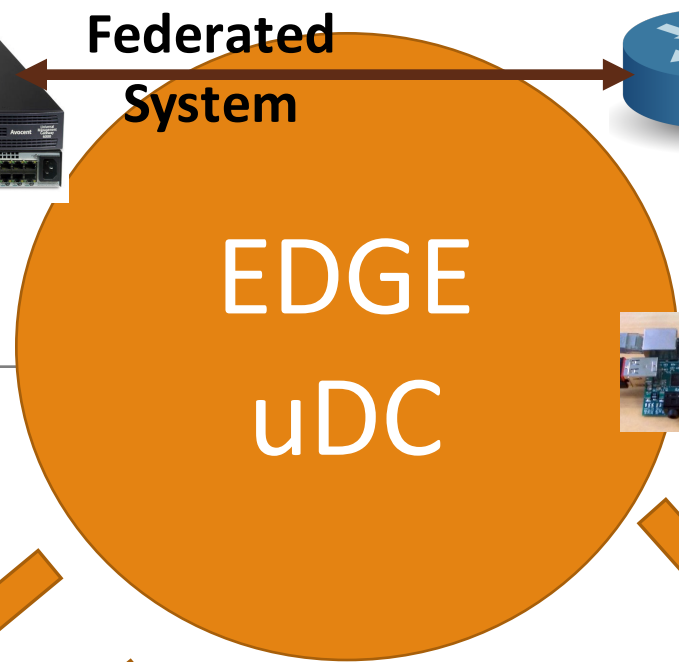
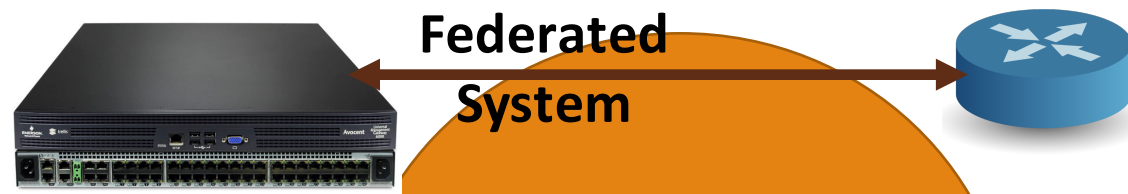


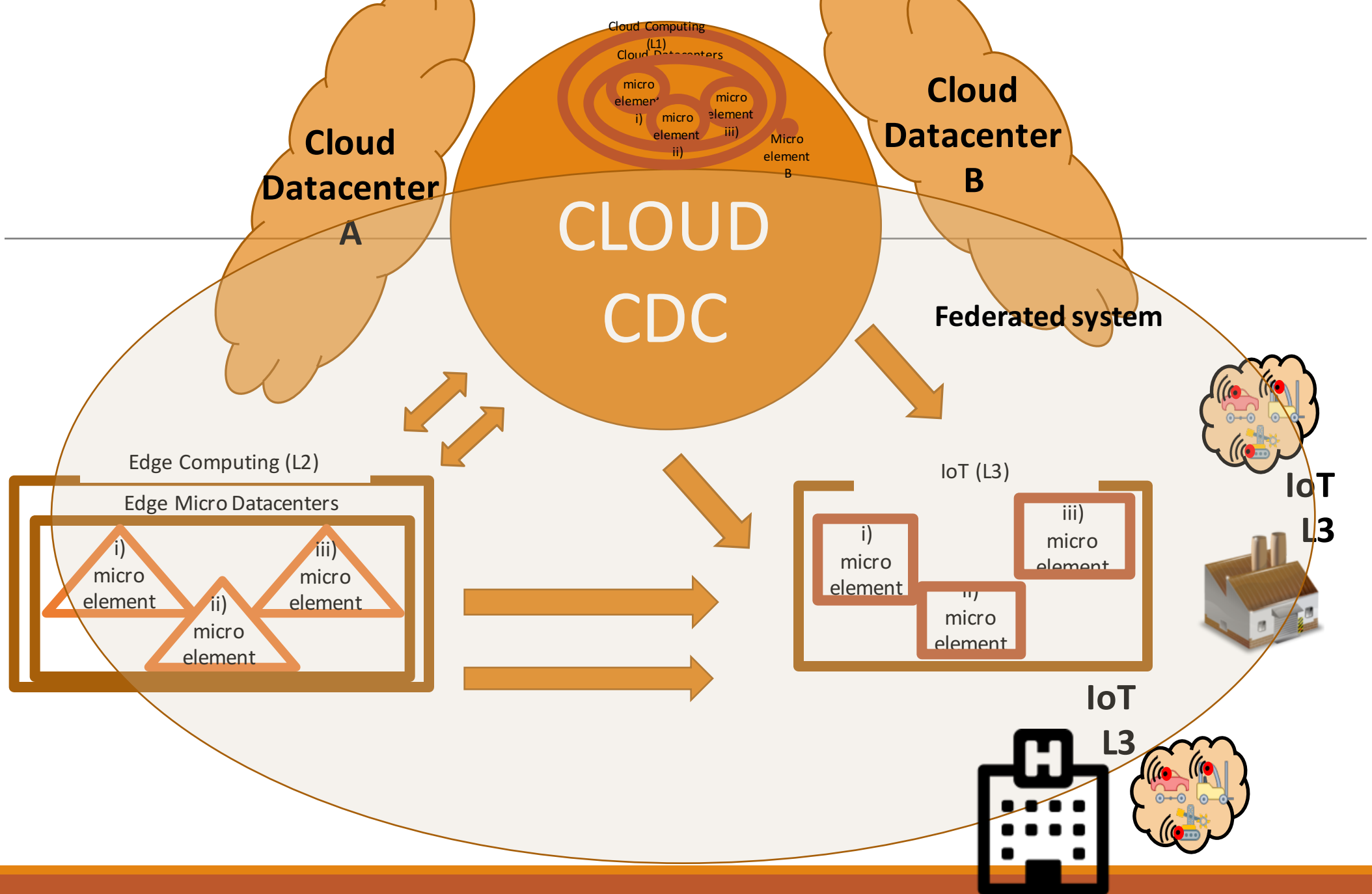
Many models: In OSMOTIC two models.

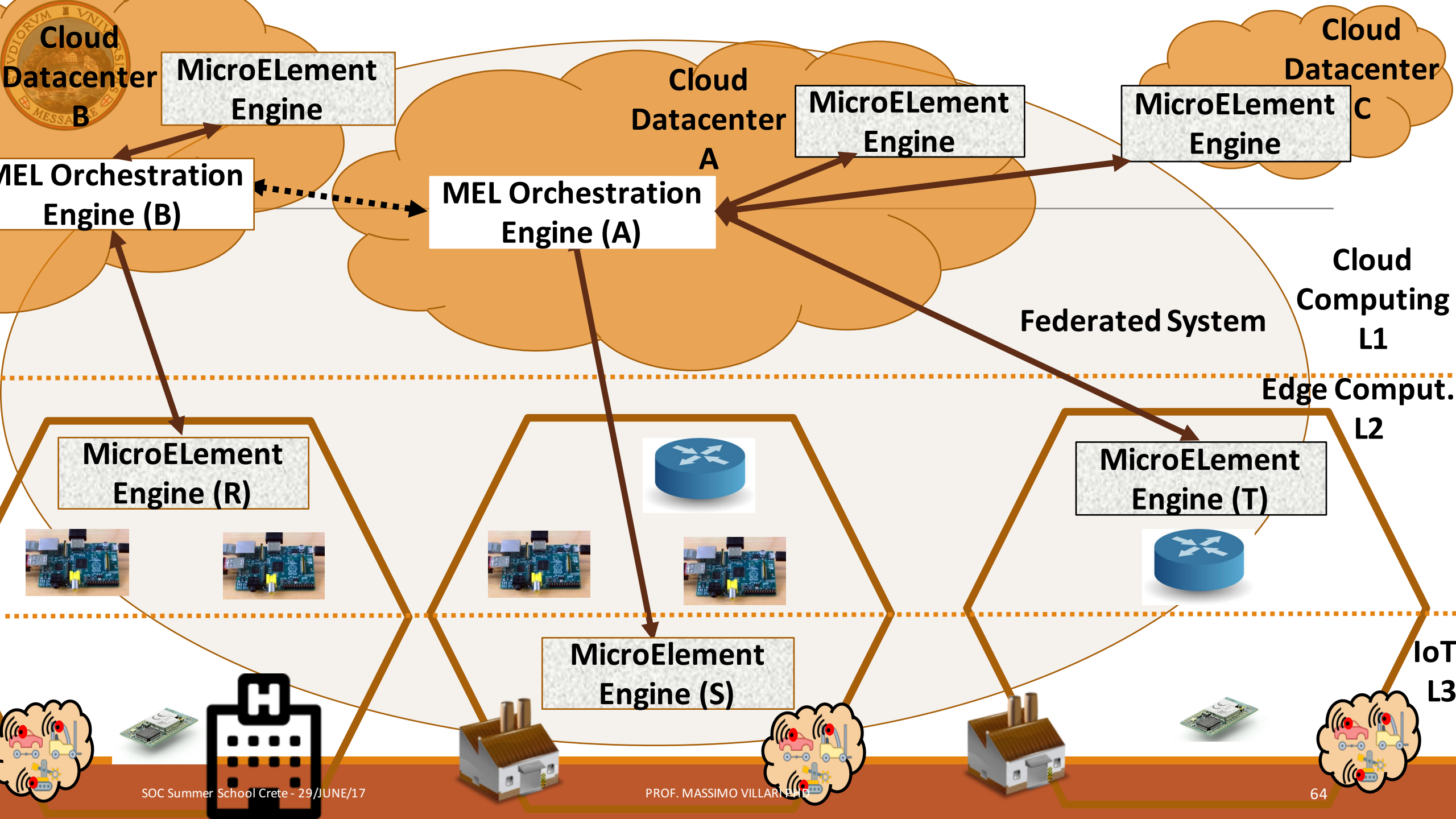
MEL Life Cycle Management:

- From the Cloud to Edge/IoT
- From the Edge to Cloud/IoT











Osmotic Computing *Challenges*

[IEEE Cloud Computing](#) > [Volume: 3 Issue: 6](#) (Dec 2016): Osmotic Computing:
[Massimo Villari](#); [Maria Fazio](#); [Schahram Dustdar](#); [Omer Rana](#); [Rajiv Ranjan](#) A New Paradigm for Edge/Cloud
Integration

- A. Runtime Microservice Deployment
- B. Microservice configuration
- C. Microservice Networking
- D. Microservice Security
- E. Edge Computing
- F. Microservice workload contention and interference evaluation
- G. Monitoring
- H. Microservice orchestration and elasticity control



Osmotic Computing *Challenges*

Runtime Microservice Deployment:

An Osmotic Computing framework should provide a **microservice Engine**, allowing users and developers to deploy containers running microservices on IoT and Edge devices, enabling microservice execution and deployment. The innovation delivered by Osmotic Computing will **facilitate the creation of a market of virtual IoT based applications**. Software adaptation and versioning mechanisms will allow Edge Cloud providers to deploy microservices consisting of a **heterogeneous pool of physical devices**. Benefits of Osmotic Computing include deployment of **distributed IoT oriented microservices, software consolidation, and service optimization**.



Osmotic Computing *Challenges*

Microservice configuration:

In Osmotic Computing developing holistic decision-making frameworks that automate configuration selection across microservices and resources in Cloud and Edge datacenters to meet QoS constraints is necessary. To this end, novel decision-making techniques based on multi-criteria optimization (e.g., Genetic Algorithms) and multi-criteria decision making (e.g., Analytic Network Process) techniques should be investigated.



Osmotic Computing *Challenges*

Microservice Networking:

Osmotic Computing is based on an abstraction of networks that spawn from Cloud to Edge and vice versa for improving the performance of the communication among microservices. The network here represents an enabler that allows us to **dynamically adjust the overall microservices behavior according to user requirements**. The network management advances in Osmotic Computing should include the development of an **interoperability layer** for remote orchestration of heterogeneous Edge devices, for example, exploiting **Software Defined Networking (SDN) and Network Function Virtualization (NFV) capabilities**, accessible through an API.



Osmotic Computing *Challenges*

Edge Computing

The approach suggests the need to combine **“mobile offloading”** with **“data centre offloading”** i.e., we off-load computation initially carried out within a datacenter to a mobile device. **This “reverse” off-loading** enables computation to be undertaken closer to the phenomenon being measured (overcoming latency and data transfer costs). The Osmotic Computing approach is therefore focused on **understanding the types of microservices which would be more relevant to execute at the Edge**, rather than within a datacenter environment, and vice versa.



Osmotic Computing *Challenges*

Microservice workload contention and interference evaluation

The co-deployed microservices on Cloud or Edge datacenters can lead to **contention problems which will affect QoS**. During deployment of microservices, orchestration techniques must consider which microservices should be combined on a datacenter resource, to minimize resource contention due to workload interference. **Workload resource consumption and QoS are not additive**, so understanding the nature of their composition is critical to deciding which microservices can be deployed together.

Research in Osmotic Computing should be focus on novel **microservice consolidation techniques** that can **dynamically detect and resolve resource contention** via microservice performance characterization, workload prioritization and coordinated deployment.



Osmotic Computing *Challenges*

Monitoring

(i) monitor and instrument data (workload input and QoS metrics, disruptive event) across microservices, Cloud datacenter, intransit network, and Edge datacenter

(ii) detect root causes of QoS violations and failures across the infrastructure based on workload and QoS metrics logs.

Researchers should investigate scalable methods (based on self-balanced trees) to

monitor QoS and security metrics across multiple-levels of Osmotic Computing including microservices, Cloud datacenters and Edge micro-datacenters.



Osmotic Computing *Challenges*

Microservice orchestration and elasticity control

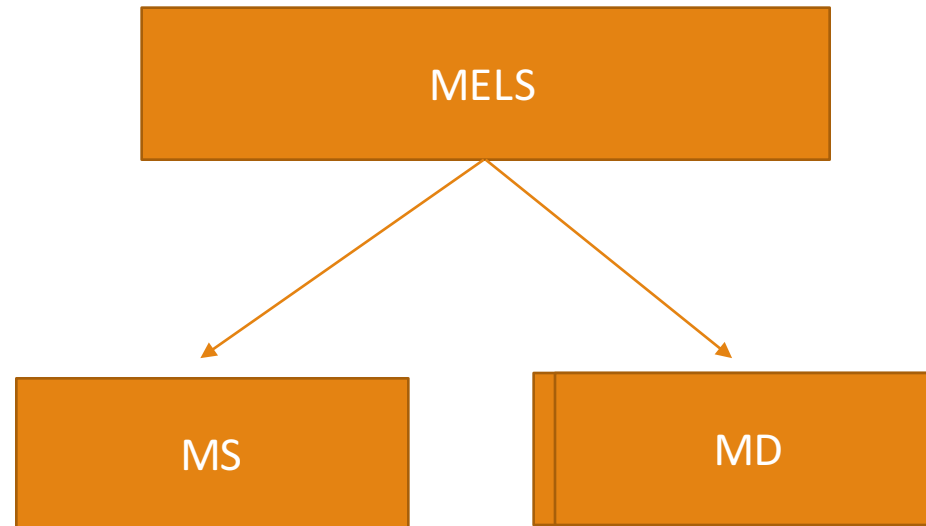
In Osmotic Computing , the traditional **notion of run-time control and reconfiguration** which only considers resources hosted in Cloud datacenters, to resources that are deployed and available at the Edge, should be **extended**. **Machine learning techniques for developing predictive models to forecast workload** input and performance metrics across multiple, colocated microservices on Cloud and Edge datacenter resources should be investigated. Additionally, **intelligent, QoS-aware, and contention-aware resource orchestration algorithms** should be developed based on the above models, **monitoring systems, and configuration selection** techniques.



Osmotic Computing *into details*

MicroElementS (MELS) split into two main abstracted components:

- MS(MicroServices) and
- MD(MicroData).





Osmotic Computing *into details*

Main root of the hierarchy is represented from MicroElementS, whereas underneath there are MSs and MDs.

The leaf of hierarchy is represented by:

- MUS
- MOS

- MUD
- MOD

U for User and O for Operational.



Osmotic Computing *into details*

Main root of the hierarchy is represented from MicroElementS, whereas underneath there are MSs and MDs.

The leaf of hierarchy is represented by:

- MicroOperationalService (like an Operating System) and
- MicroUserService(like a user application on OS).

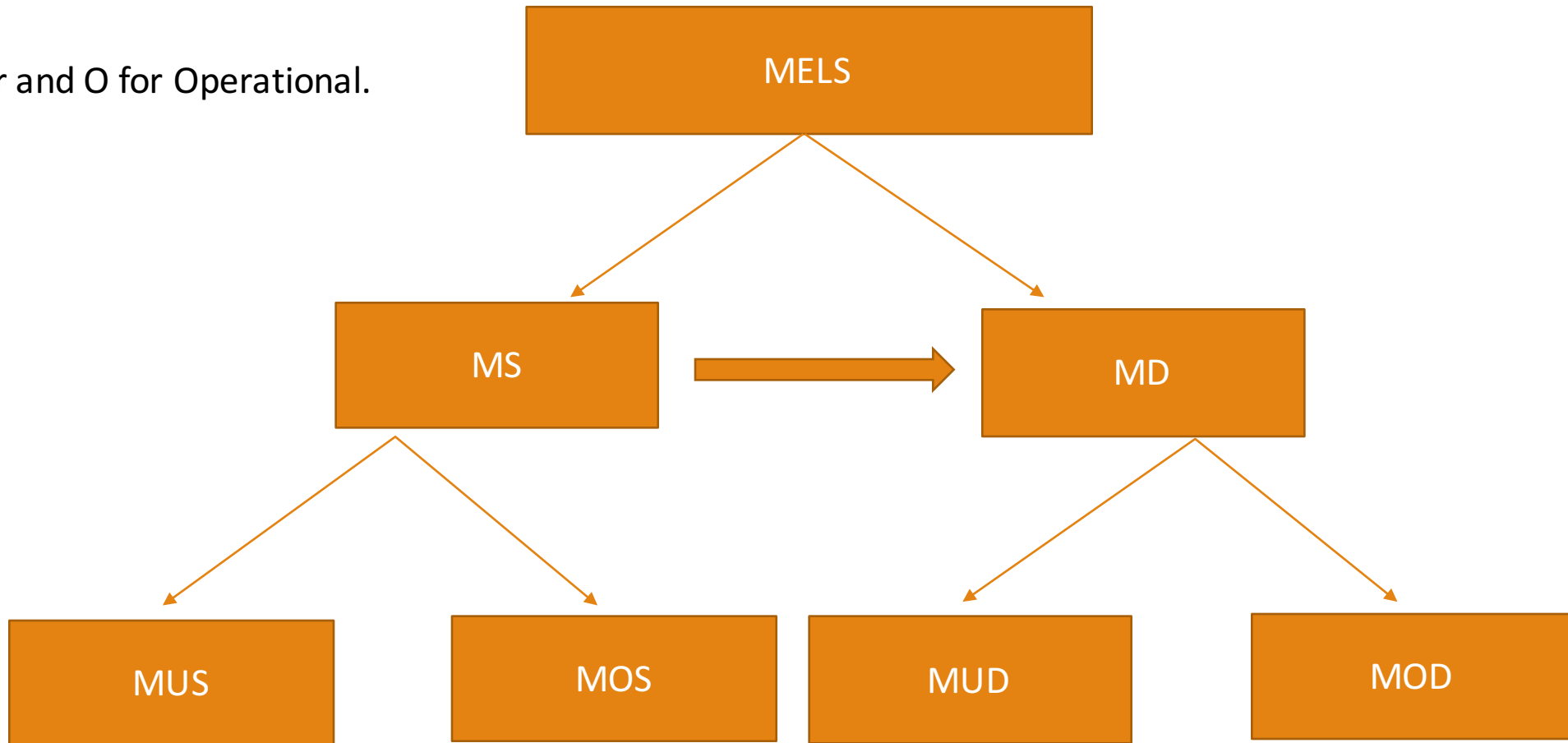
- MicroOperationalData(-> MS configuration; <- MS monitoring) and
- MicroUserData(-> User Data; <- User Data from IoT).

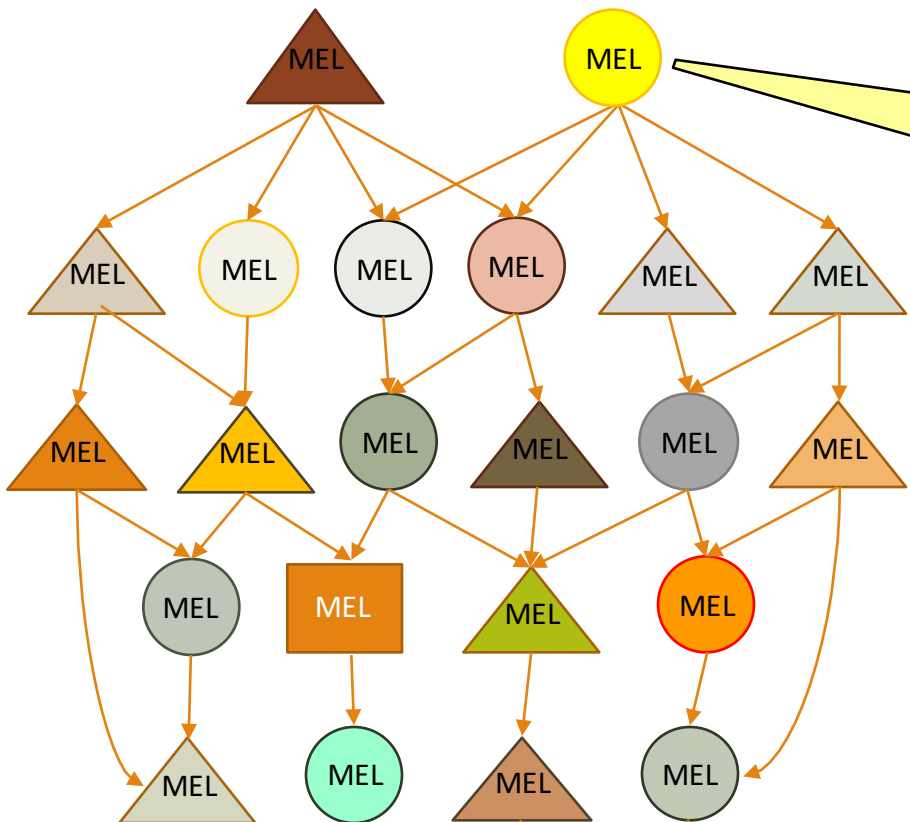
MD and MS are mobile, can be portable and cross-platform.



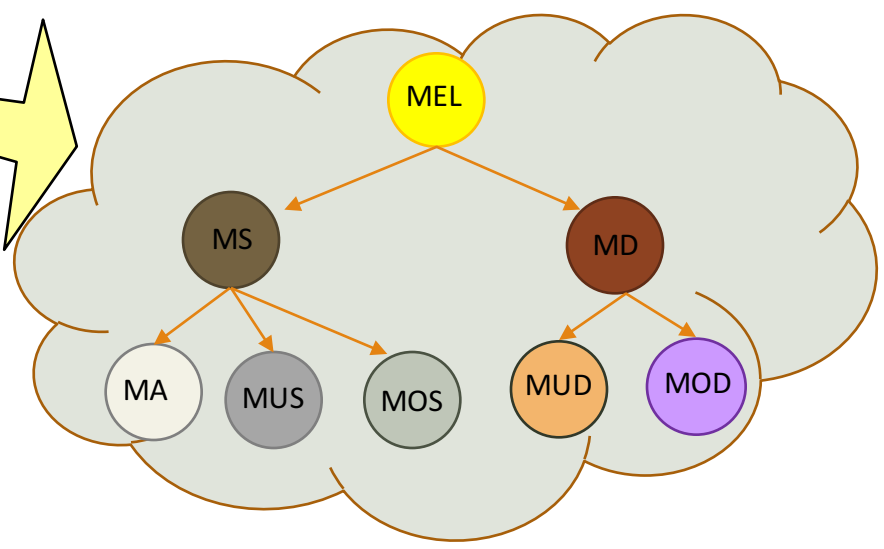
Osmotic Computing *into details*

U for User and O for Operational.

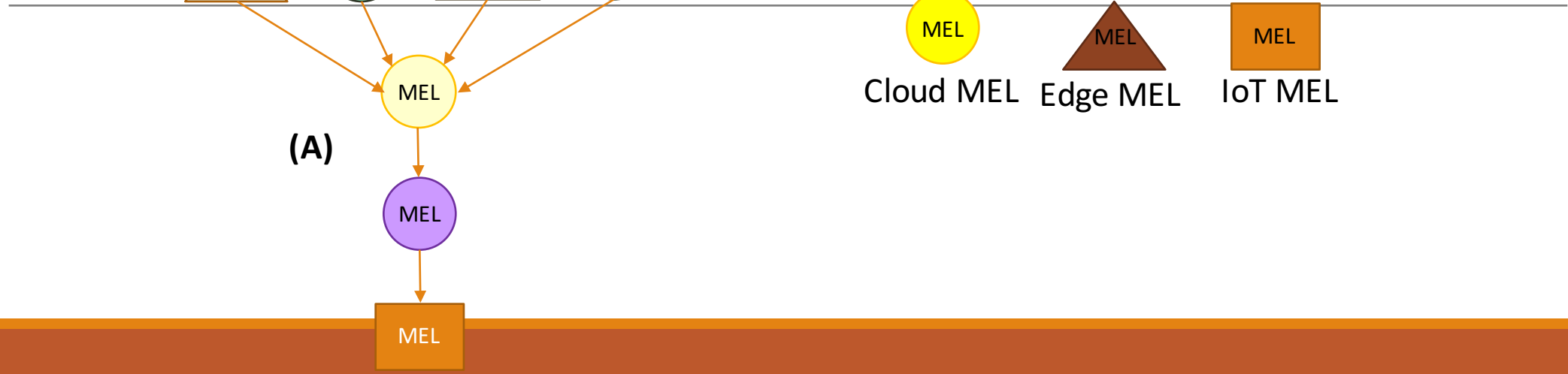
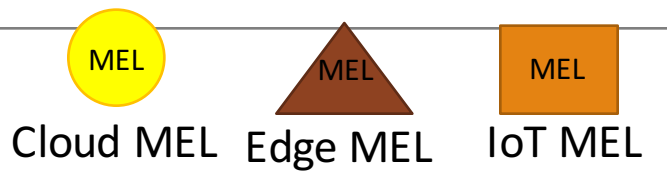




(A)



(B)





Osmosis equivalences:

- Membranes (semipermeable and impermeable),
 - Software Defined Membranes



OC: Software Defined Membrane (permeability)

- Membranes As A Filter
- Membranes assembled from MS
- Membranes Assimilated at Gateways and/or Proxys
- Able to be permeable or impermeable

Software
Defined
Membranes



Big Security Issues in the EDGE ☹️

In November 2013, the owner of a smart TV made by LG Electronics discovered that [the device was collecting information](#) about his viewing habits, even when the "collection of watching info" feature was turned off. Worse, the TV also sent back to LG's servers the names of files stored on external media devices and even network shares.

2 more wireless baby monitors hacked: Hackers remotely spied on babies and parents

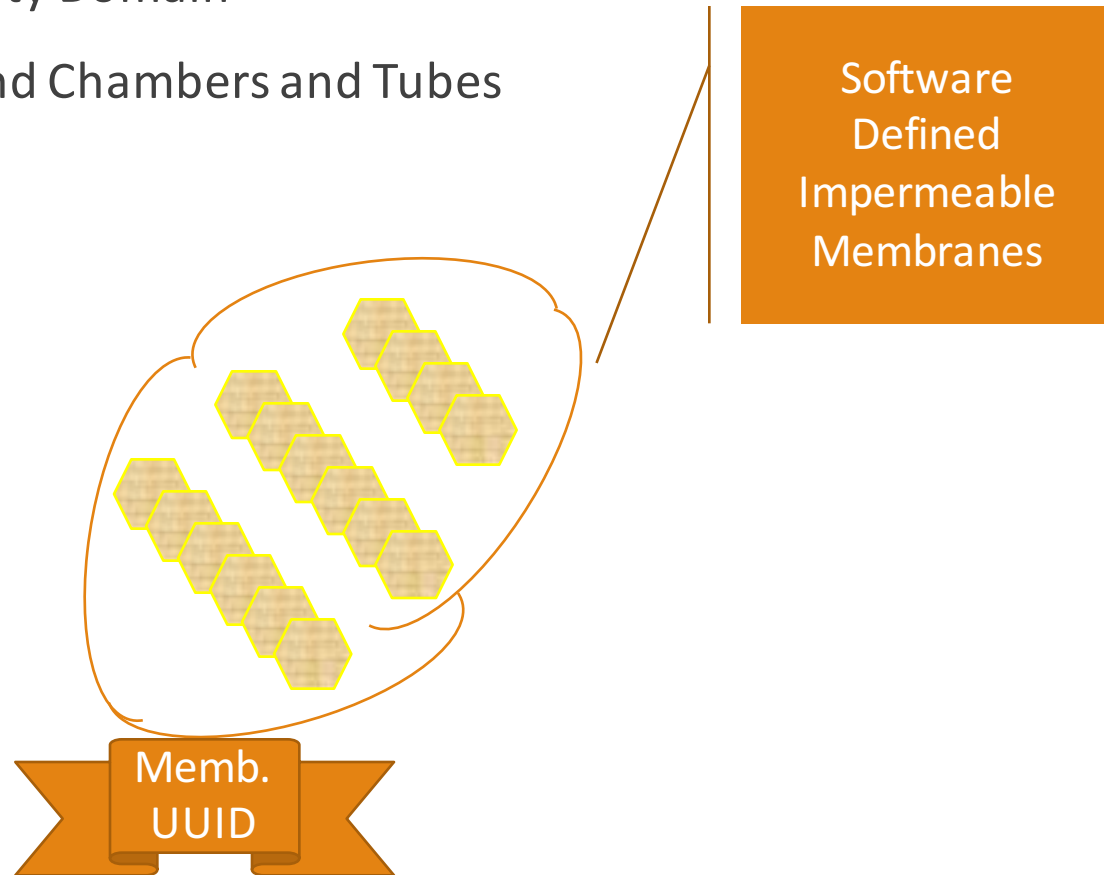
Two more wireless baby monitors were hacked. One family heard voices as the camera followed them about the room; the second mom was freaked out and scared as a hacker remotely controlled the camera to follow her movements.



OC: Software Defined Pressure and Imperm.

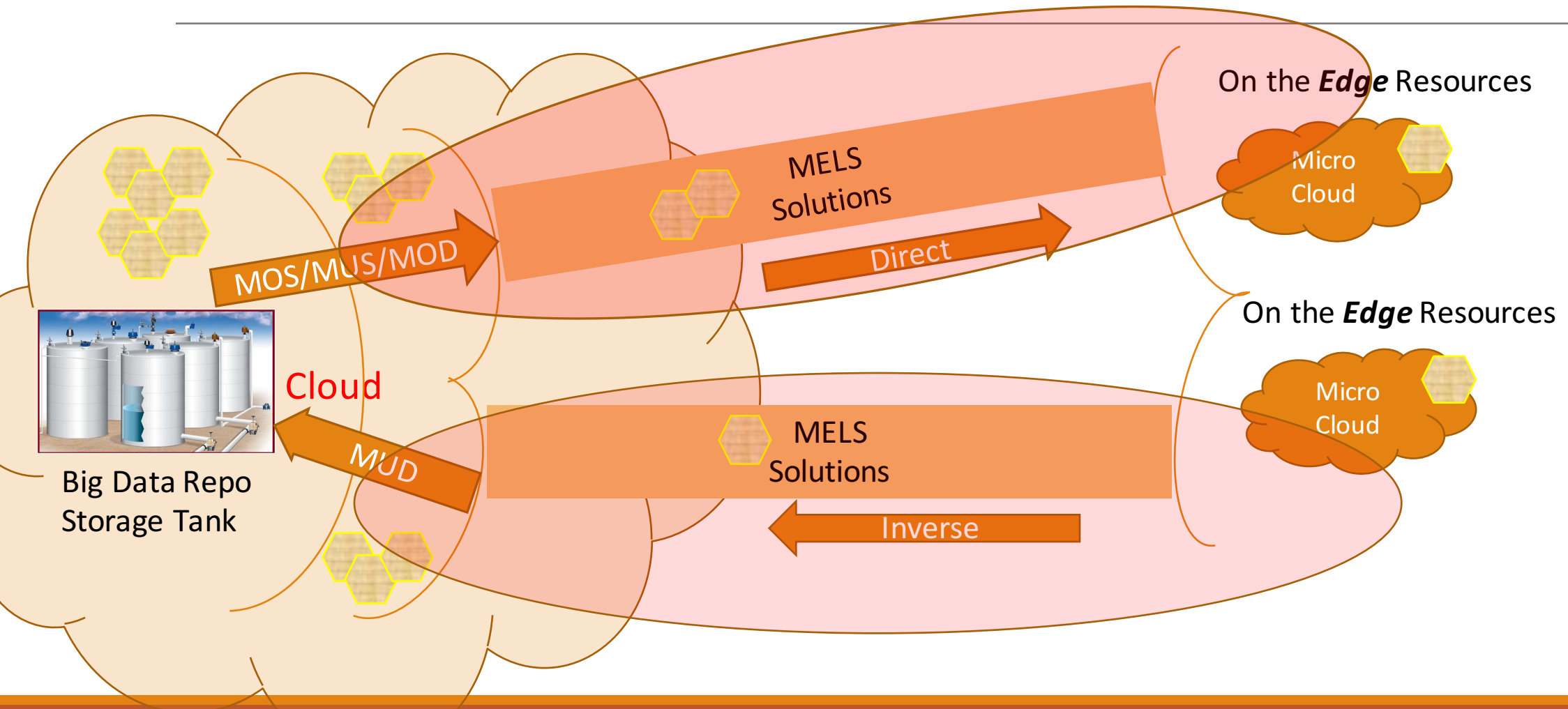
Security Domain

Around Chambers and Tubes





Osmosis: Membranes



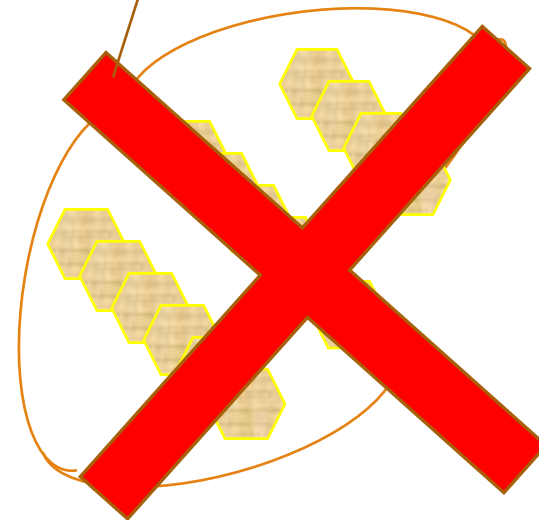
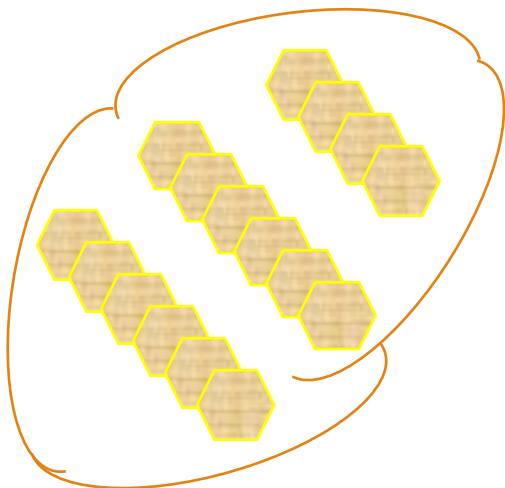


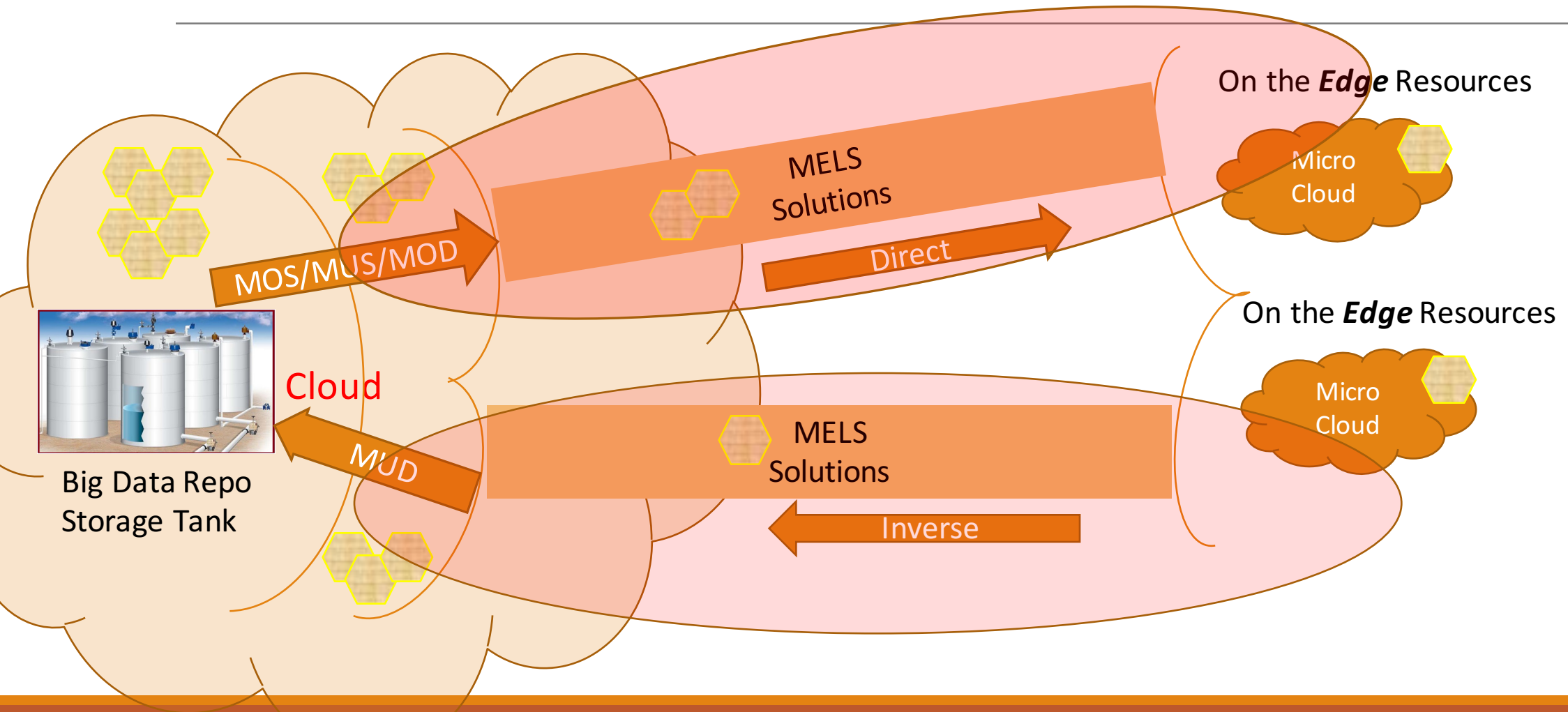
OC: Software Defined Pressure in

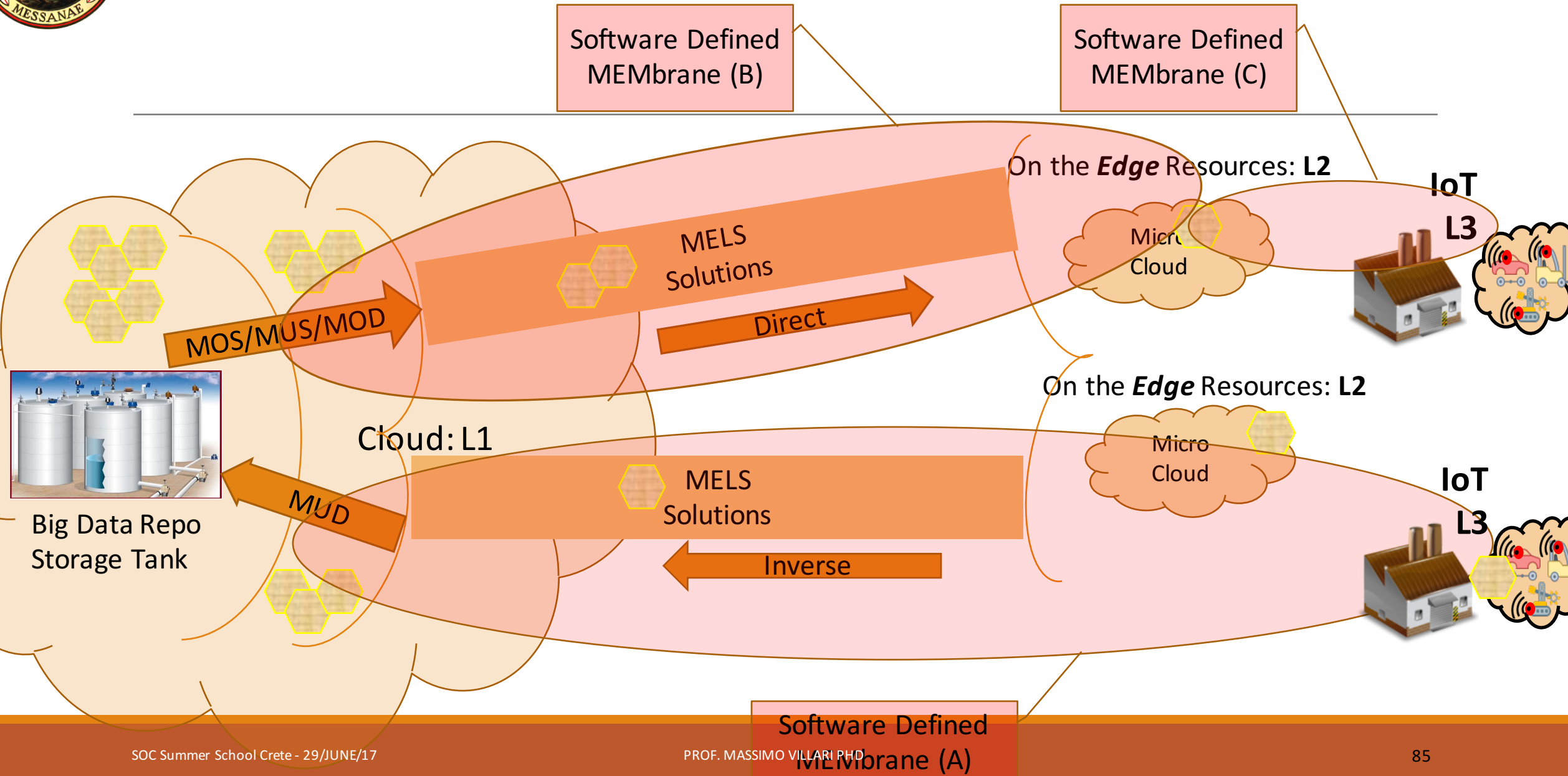
Security Domain

Software
Defined
Impermeable
Membranes

Software
Defined
Security
And Disruption of it

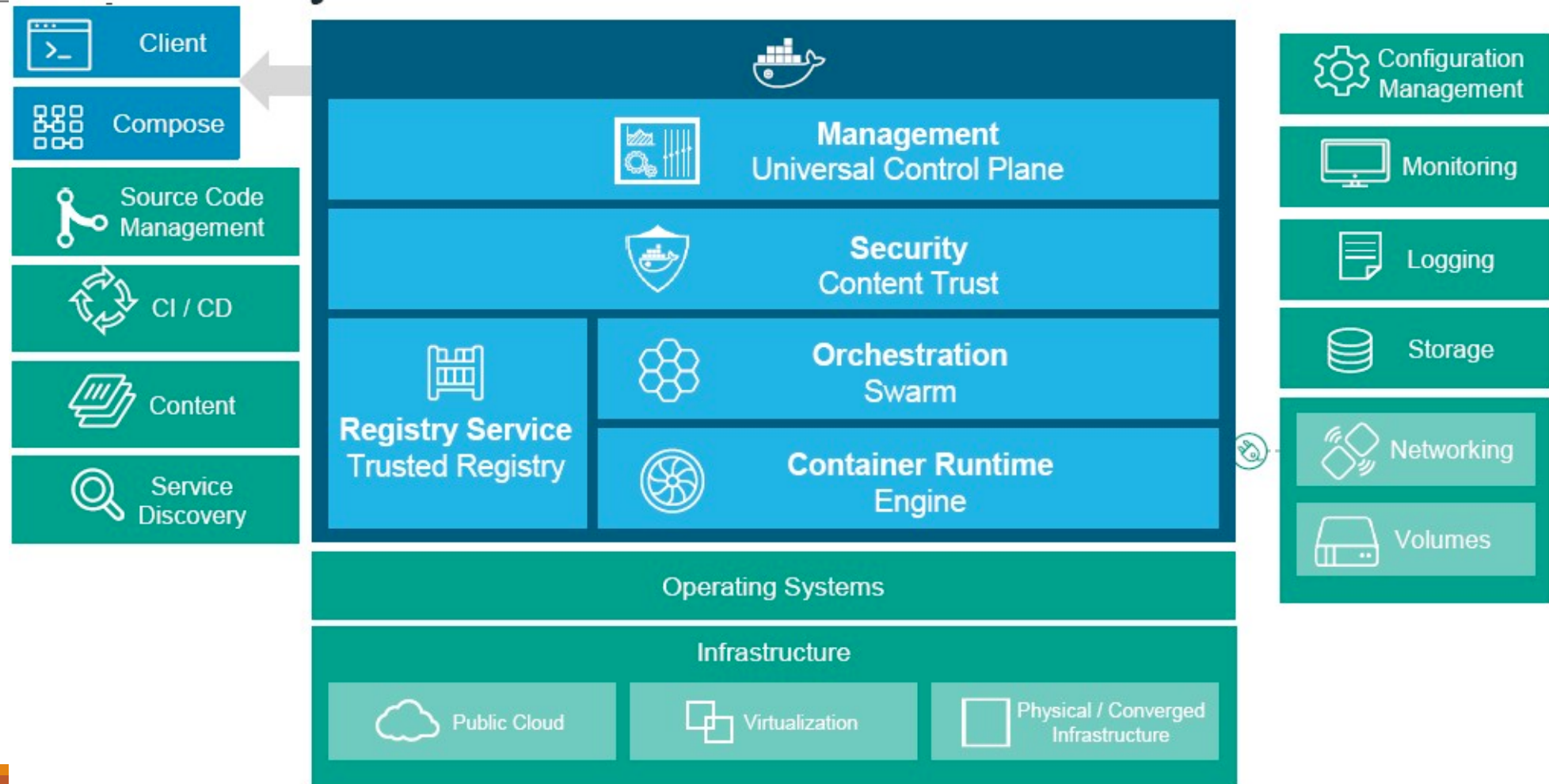








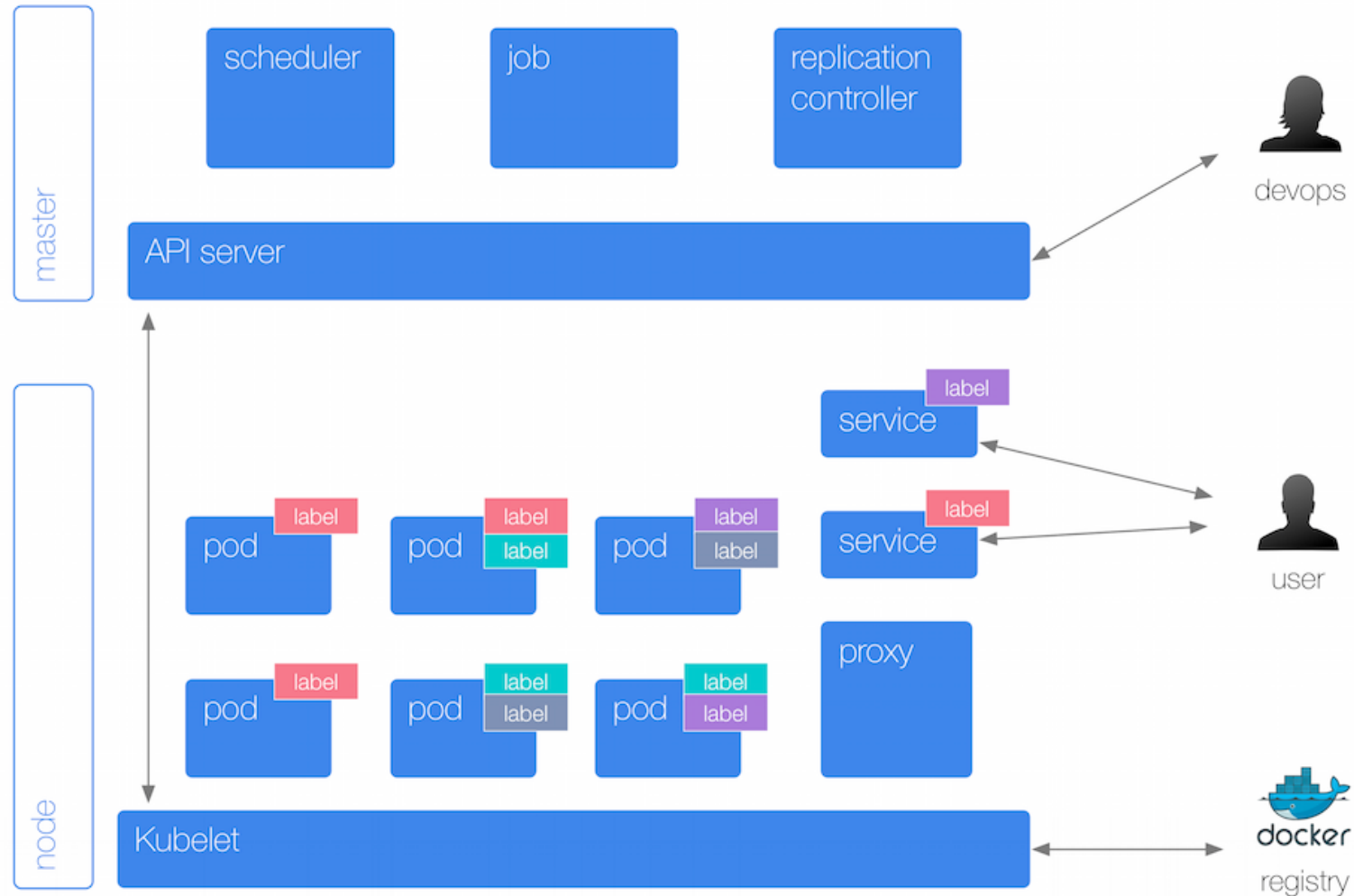
Osmotic NOT from Scratch: eg. containerization





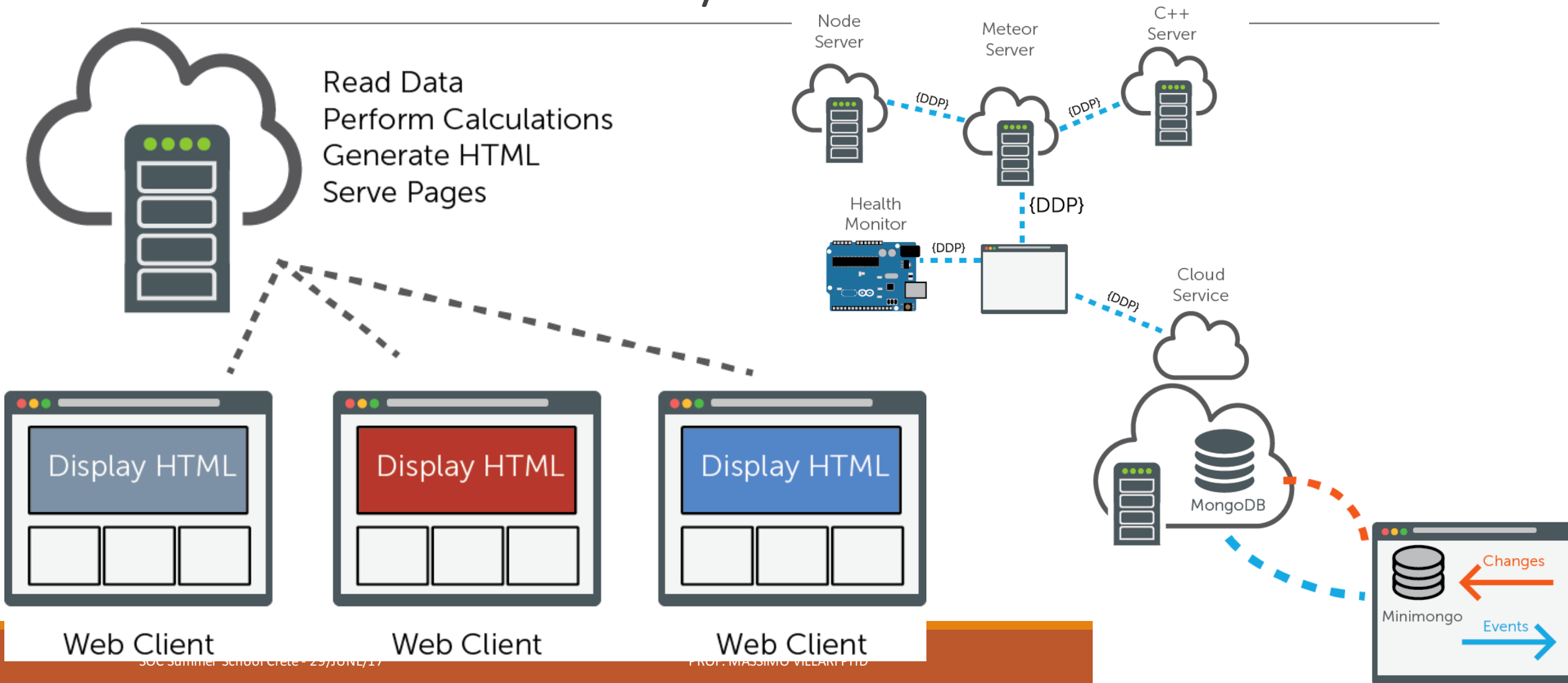
Osmotic NOT from Scratch: eg. containerization

Here
PODS are equals to MELS





Osmotic NOT from Scratch: eg. Javascript MeteorJS on MPU/MCU





Osmotic Computing: in Docker and JS Meteor or..

In 1- MSs (MOS and MUS) are equals to Container

In 1 – MD can be:

- MOD in YAML for deploying Services
- MUD in Json for charactering Filesystem layers in AUFS and LayaoutFS

i.e., Kubernetes leverages the approach of more MOS for deploying the Agents

In 2- MS (MOS and MUS) are equals to Javascript code:

In 2 – MD can be:

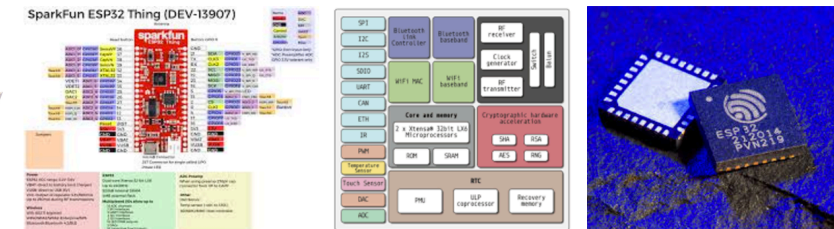
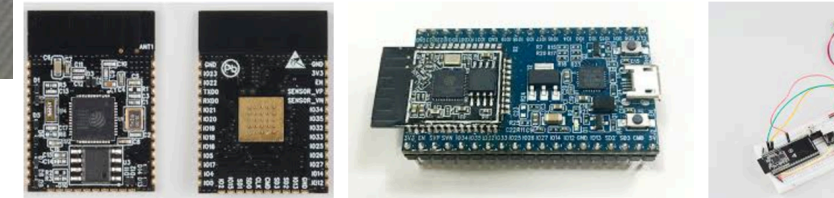
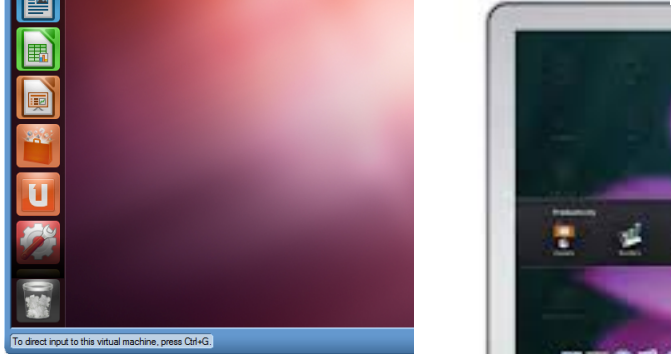
- Json

*** The TOSCA Simple Profile in YAML V1.1 is now out for a public review.**

For details and links to the specification, see the announcement at <https://www.oasis-open.org/news/announcements/15-day-public-review-for-tosca-simple-profile-in-yaml-version-1-1-ends-march-2nd>



Osmotic Computing *which devices: mpu/mcu*





With Raspberry



A new Prototype with:

- Archlinux Read Only Mode
- Docker
- Kubernetes and Hypecube
- Fl..

For the Future (looking at life cycles of Microservices) :

- PXE in Raspberry
- Filesystem and with Snapshot, TAGs and Versioning
-



With ESP 8266/32: looking at platforms and protocols

A new Prototype with:

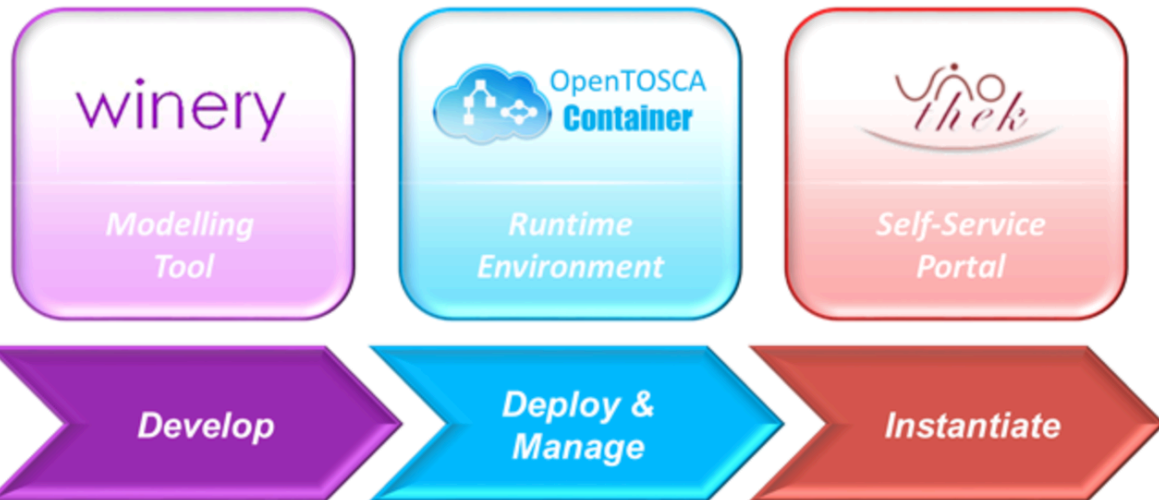
- LUA and Python Architectures
- Simple Code Injection
- CoAP Client and Server in Python



OpenTOSCA - Open Source TOSCA Ecosystem helpful in Osmotic??

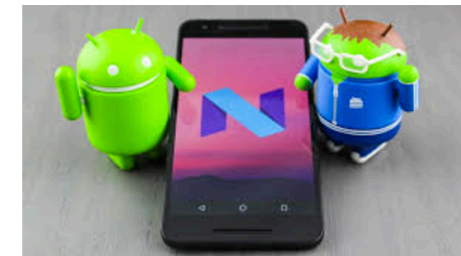
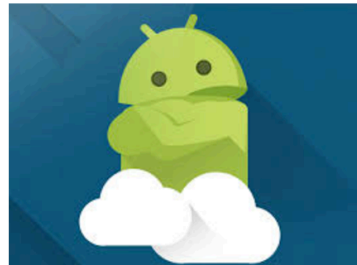
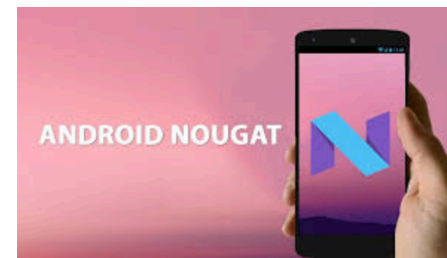


OpenTOSCA Ecosystem Overview





Osmotic Computing *which oth. devices??*



ANDROID





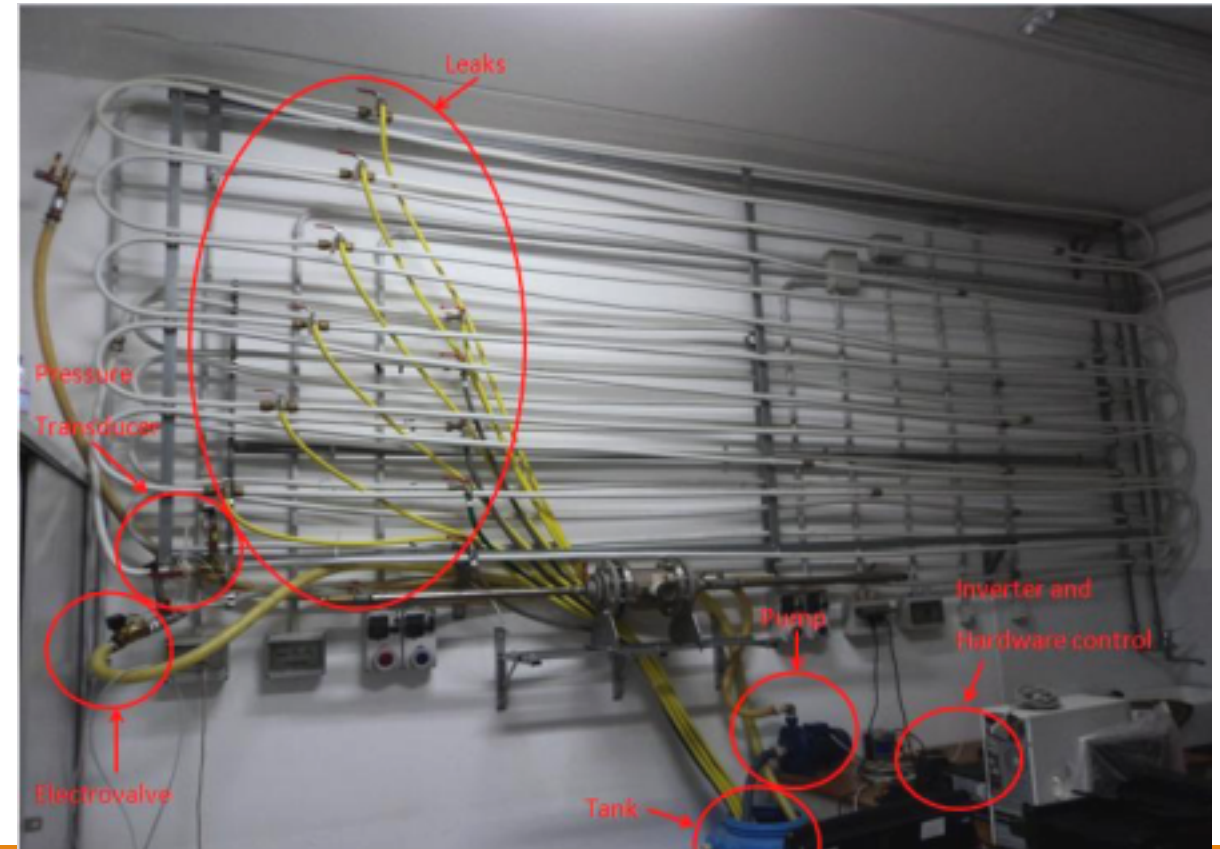
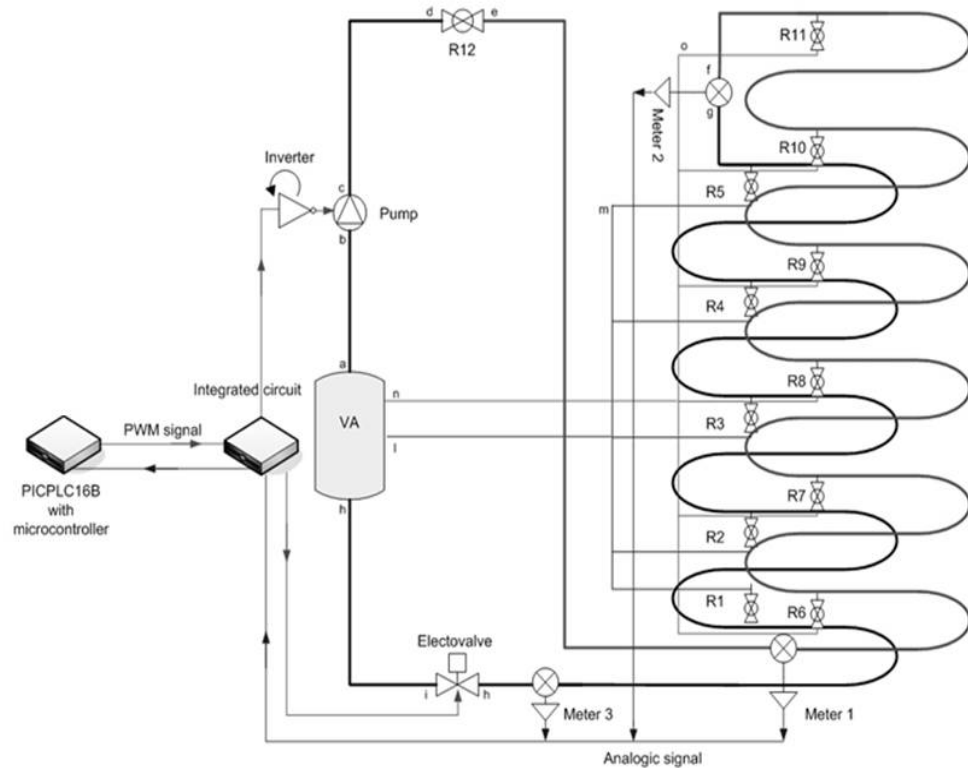
Osmotic Computing On the Cloud

Reusing the existing cloud infrastructure:

- Cloud Storage
- Cloud Processing
- NFV-SFC
- IAM
- NoSQL DB: eg., MongoDB
- APIs RESTFul
- ...
- VMs and Containers

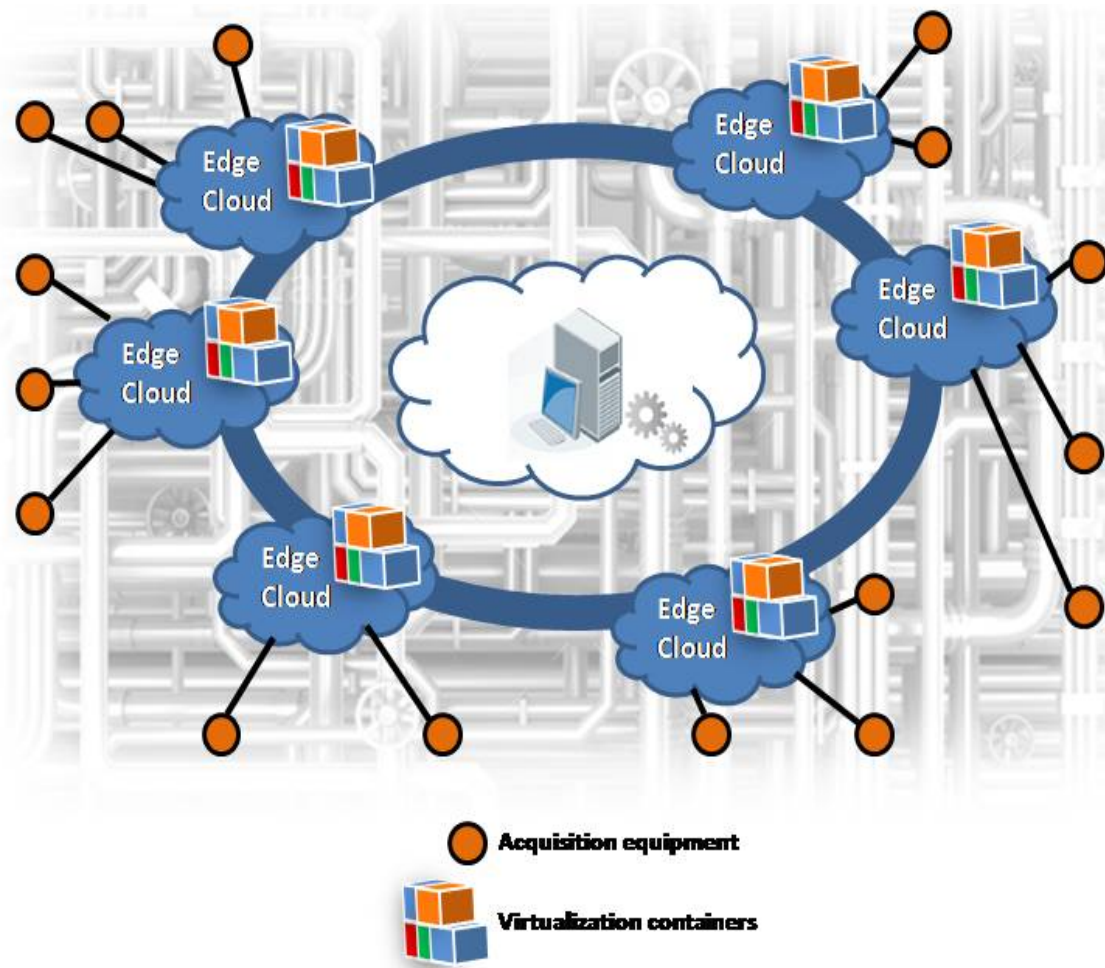


Osmotic Computing Use-Cases: Acquedotto



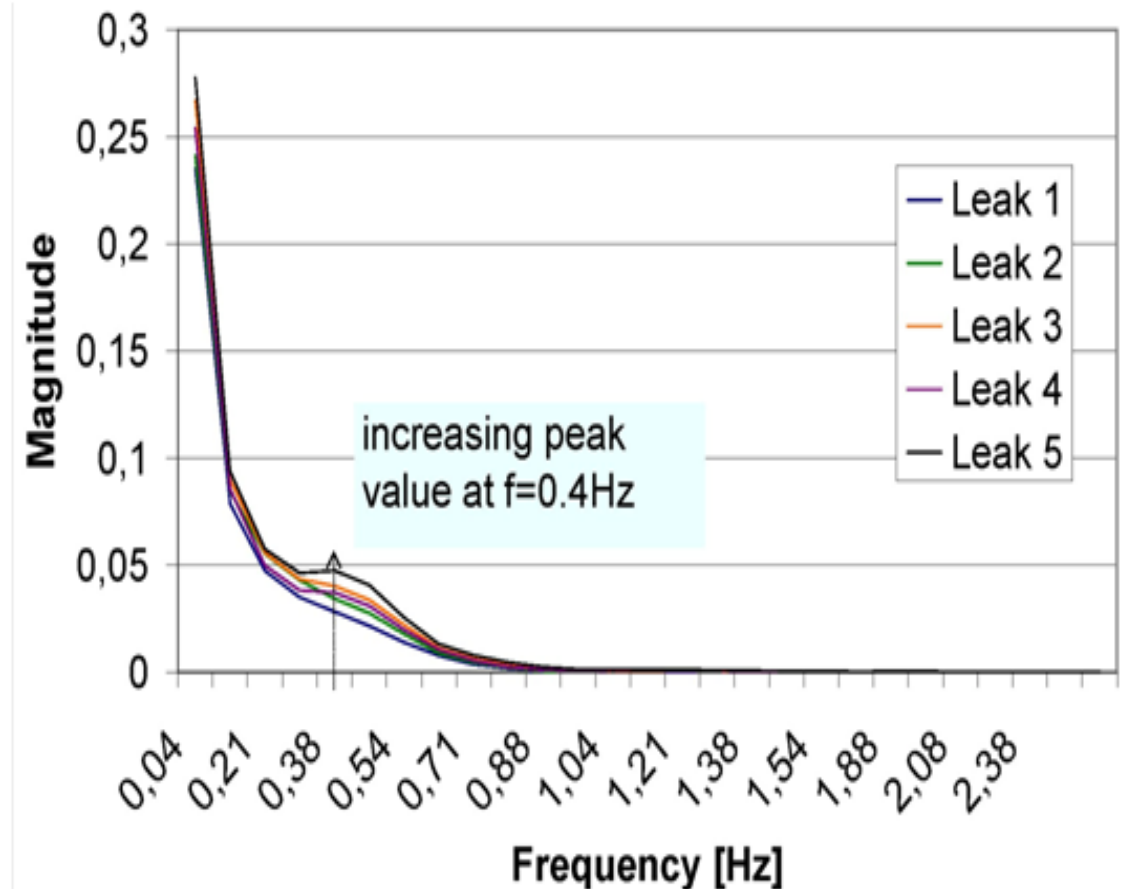
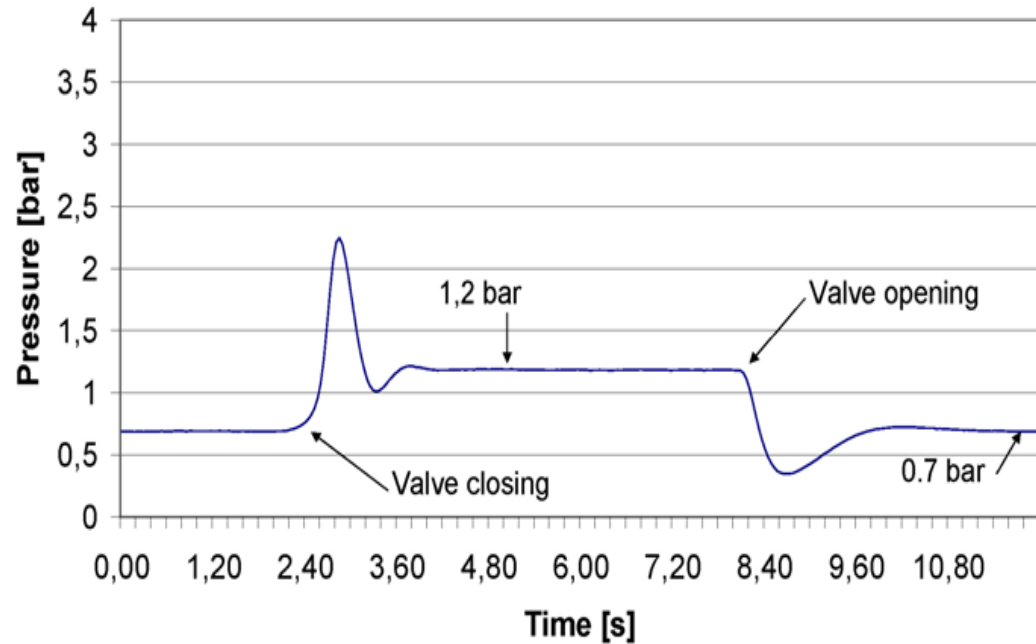


Osmotic Computing Use-Cases





Osmotic Computing Use-Cases: FFT



Osmotic Computing

some thoughts



MovieMaker: Giuseppe Tornatore

Doctorate Honoris Causa in Scienze Storiche,
Archeologiche e Filologiche





WIRED: The War for Digital Talent Has Begun

Markus Nordlin, the CIO of Zurich Insurance, sums it up perfectly:

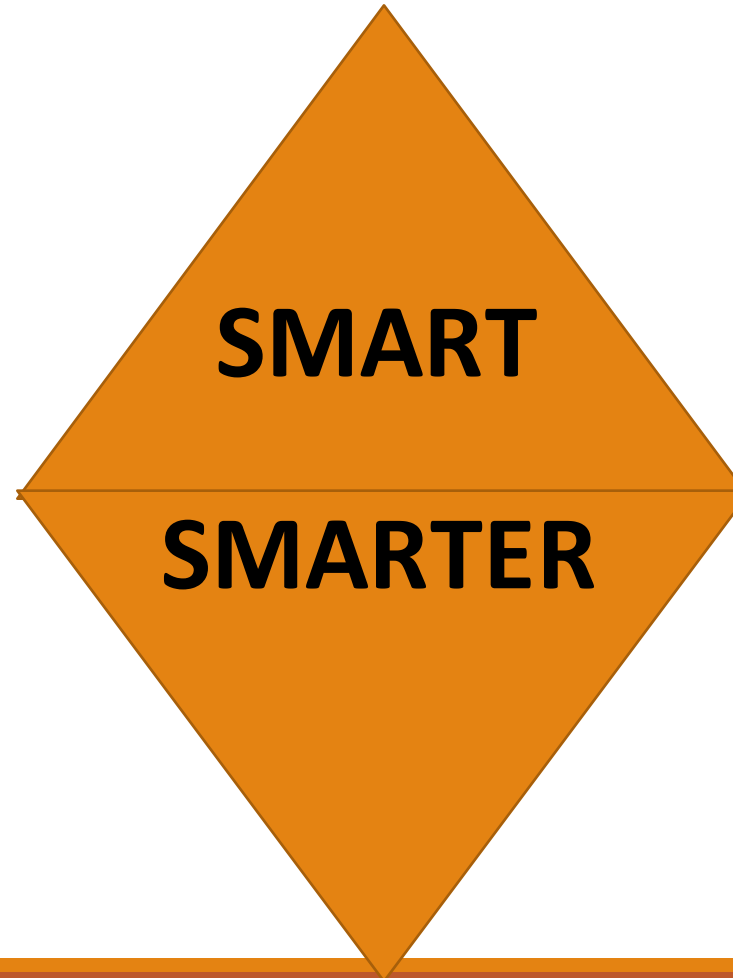
“I believe that the successful leaders of tomorrow, in any business or industry, are going to be true hybrid professionals who have spent some time in IT but have shifted to operations and vice-versa.”



Enablers

Sustainability

Optimization



Adaptation



Digital life: the new *mindset*

DIY

Community

Virtual Stuff:

- Rooms
- Meetings
- Networking
- Tasks
- Social

VS

• Real Stuff:

- Rooms
- Meetings
- Networking
- Tasks
- ...



Challenges: pros and cons

Real vs Virtual

True vs Fake

Social vs Isolation

Global vs Local

Openness vs Closeness

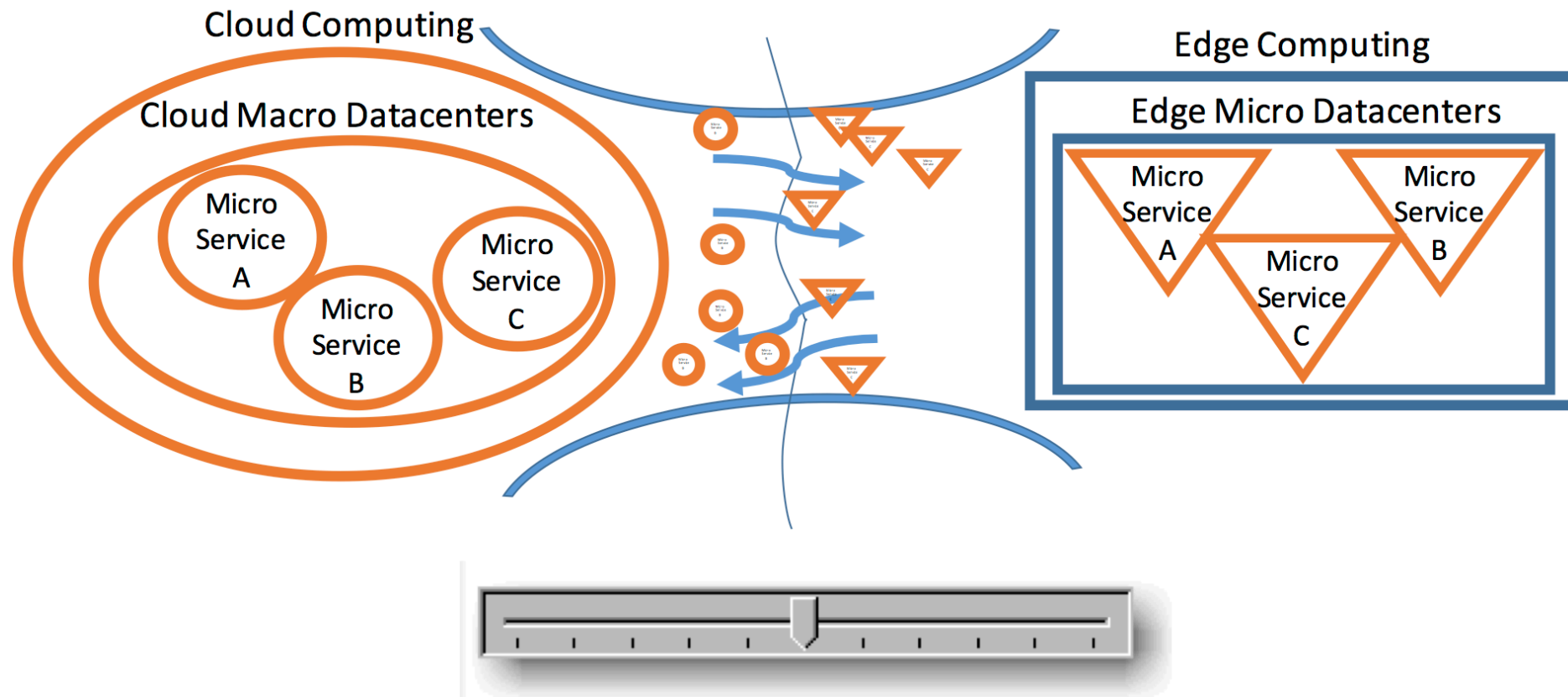
Static vs Dynamic



SMOOTH
AND
NO DISRUPTIVE ANY MORE
ICT of the future....



Osmotic Computing Concept in Dig-Life





Hazard the applicability

User Interfaces (UI):

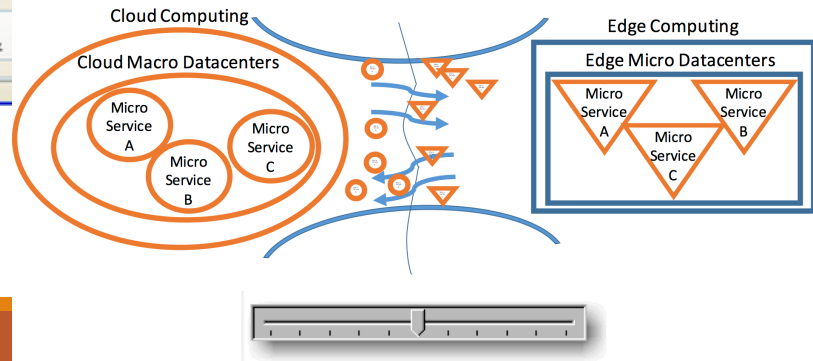
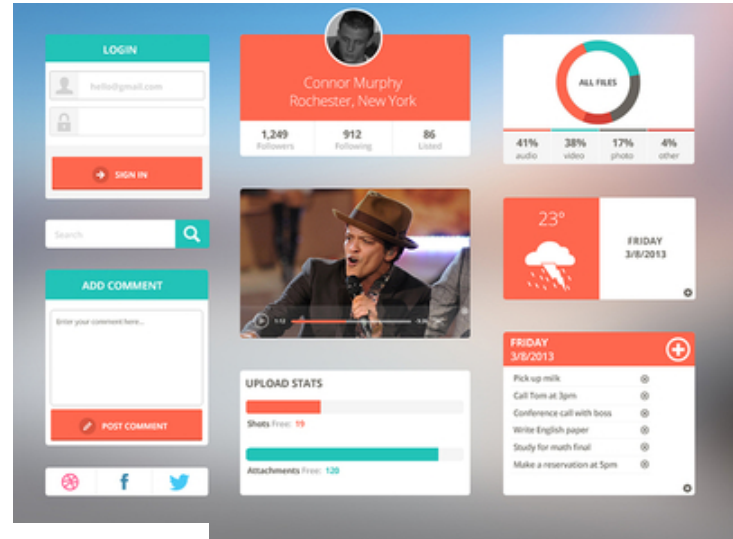
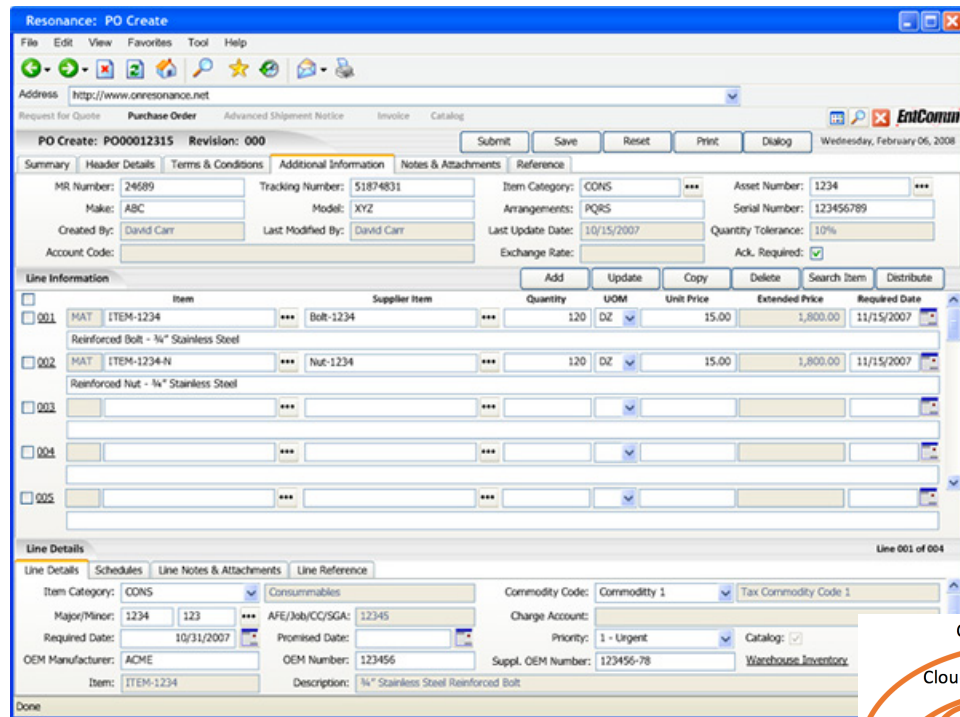
- Social ??
- Serverless ??

Software Defined Osmosis for People
From Smart to Osmotic Cities



Osmotic for UI

Software Defined Membranes





Software Defined Osmosis for People

Multidisciplinary Boards of People

(different knowledge and IT Skills)

- Osmotic Collaboration Virtual IT Tools

Software
Defined
Membranes

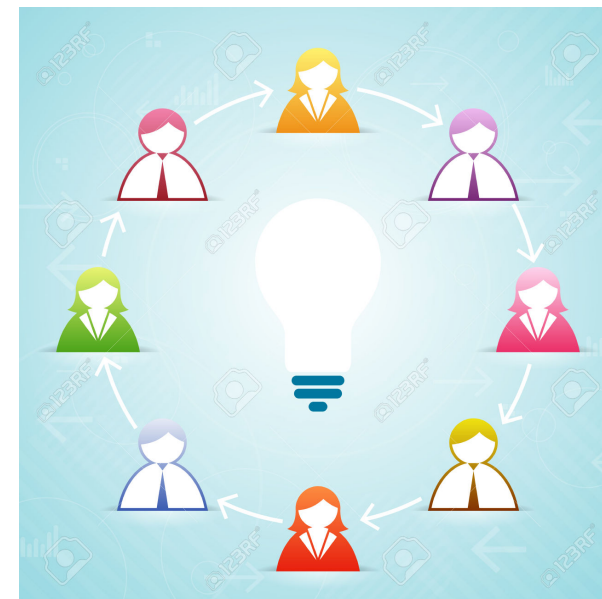
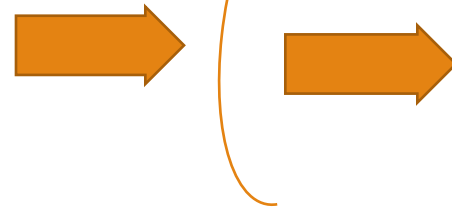
Lorem ipsum

Vivendo contentiones ea nec.
ex per, vix nemore nusquam an.



Download from
Dreamstime.com

58211246
Mast3r | Dreamstime.com



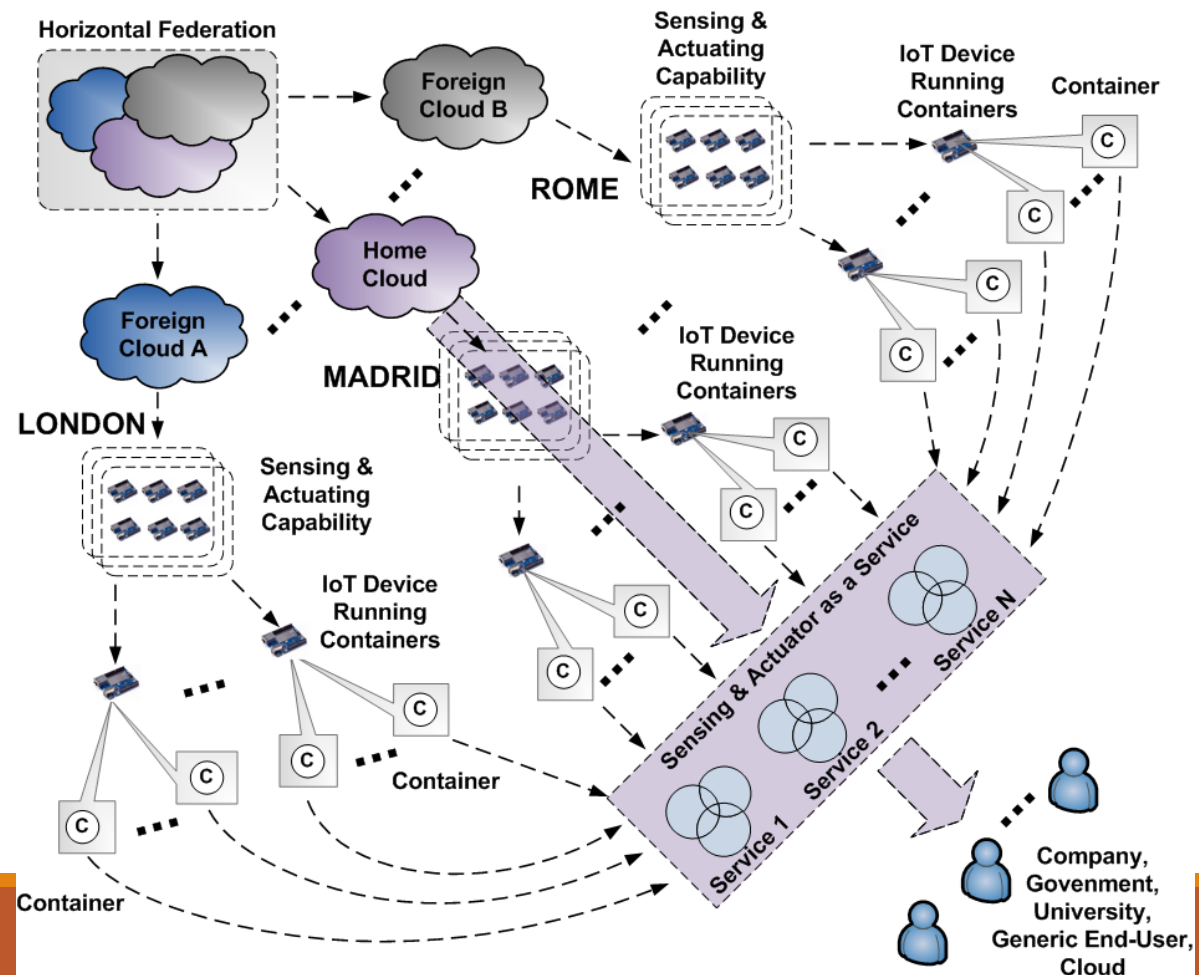


From Smart to Osmotic Cities

Natural Extension of **Federation** among Users and Service - Utility Providers

MicroElements Oriented

Osmotic Based



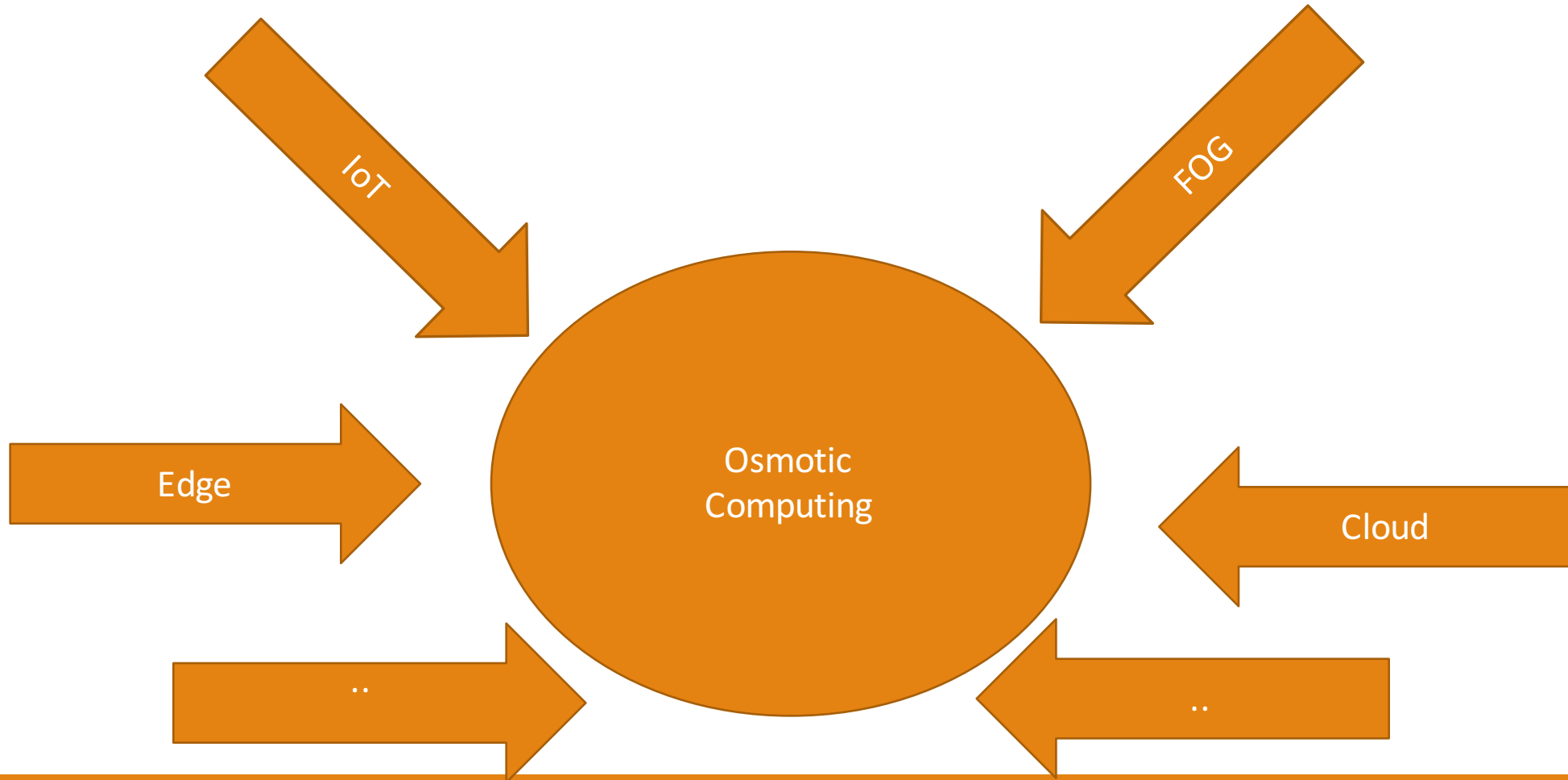


From Smart to Osmotic Cities

Where People and Devices Osmotically Behave

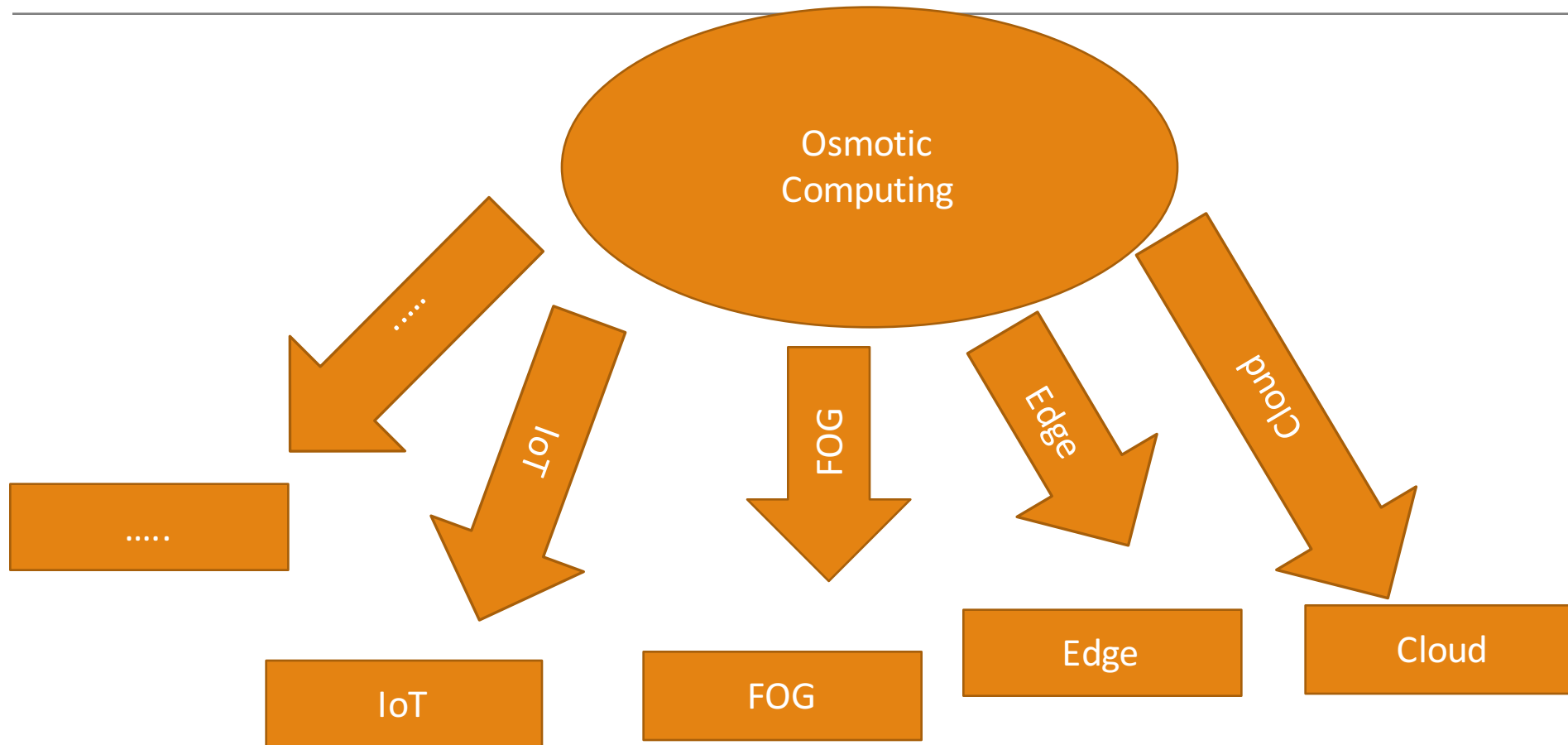


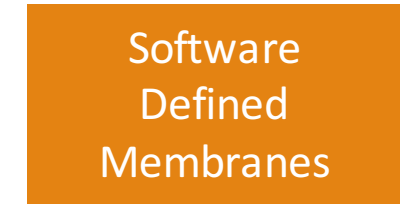
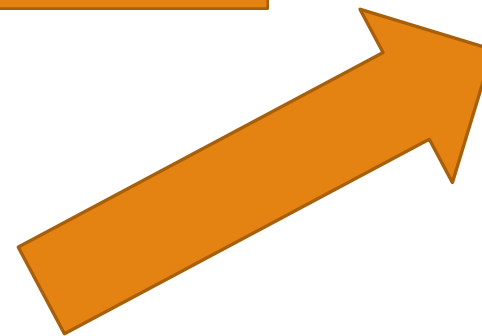
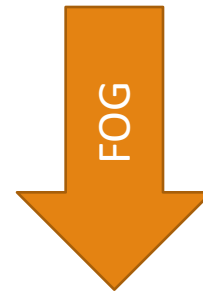
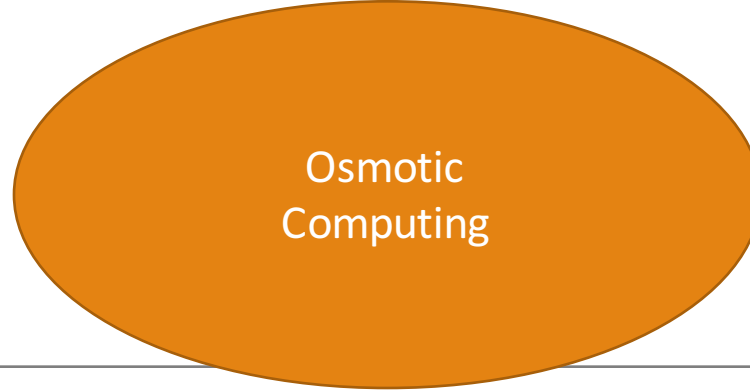
In Conclusion we can...





But we also can ...





In Conclusion
we can state
Osmotic Computing
is an
Abstraction of
.. existing Something Computing
And more

PROF. MASSIMO VILLARI

FCRLAB - UNIVERSITÀ DI MESSINA, ITALIA

MVILLARI@UNIME.IT

