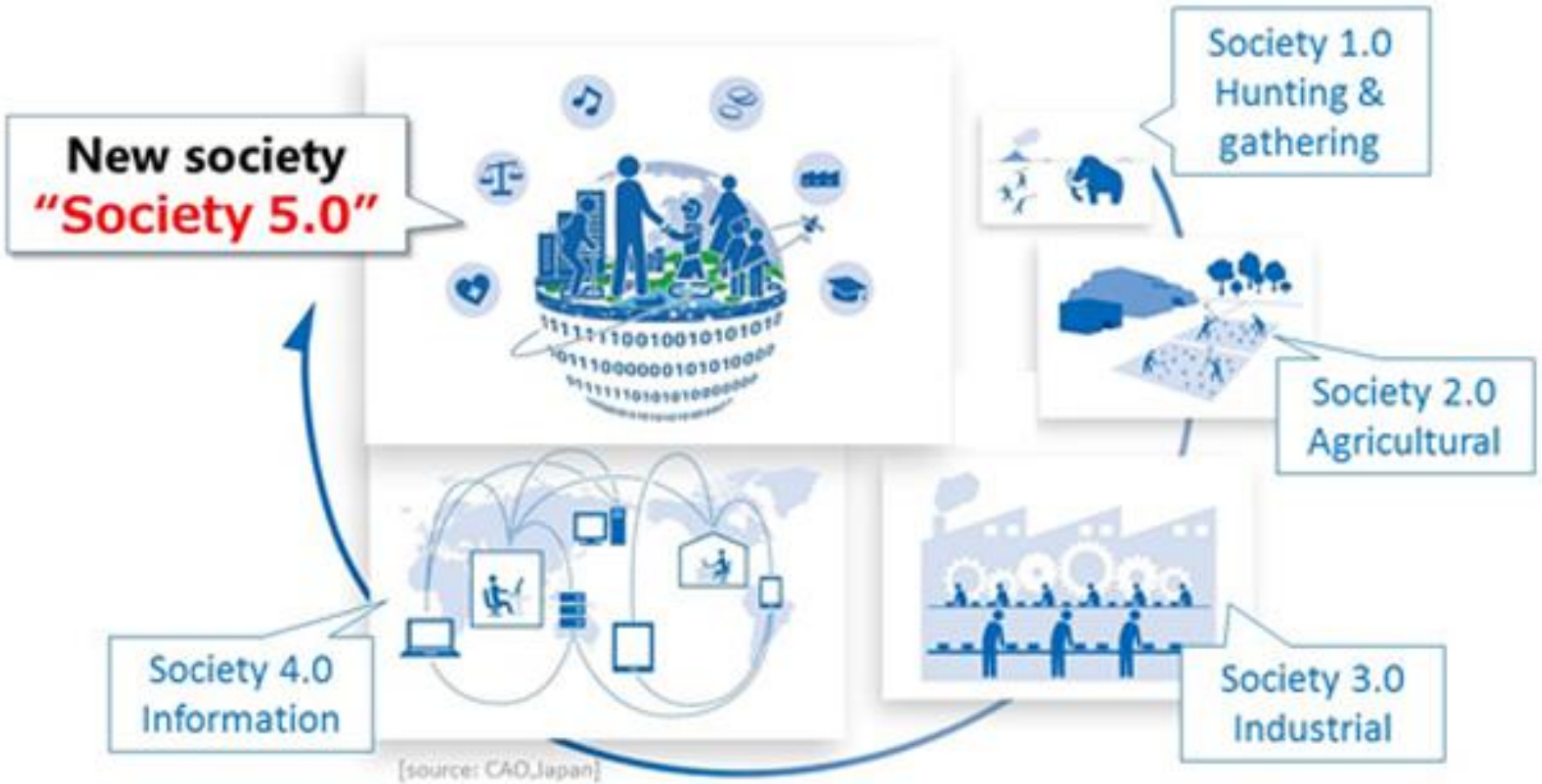
New Cloud Computing models Supporting Society 5.0 Service Integration

Andrea Sabbioni

Department of Computer Science and Engineering, DISI
University of Bologna, Italy

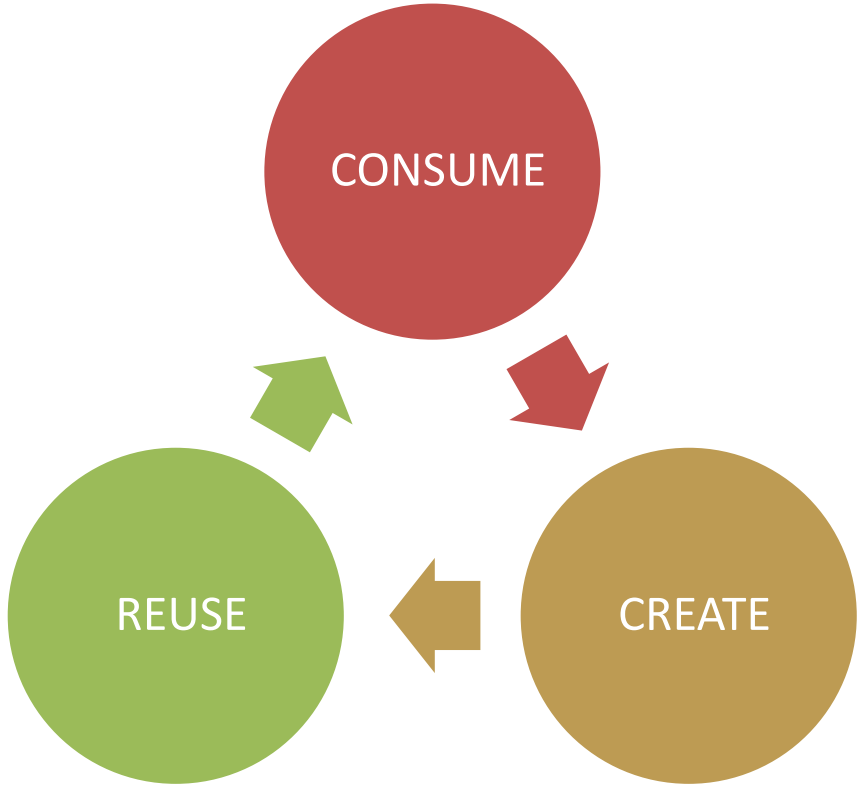
Society 5.0



"A human-centered society that balances economic advancement with the resolution of social problems by a system that highly **integrates** cyberspace and physical space."

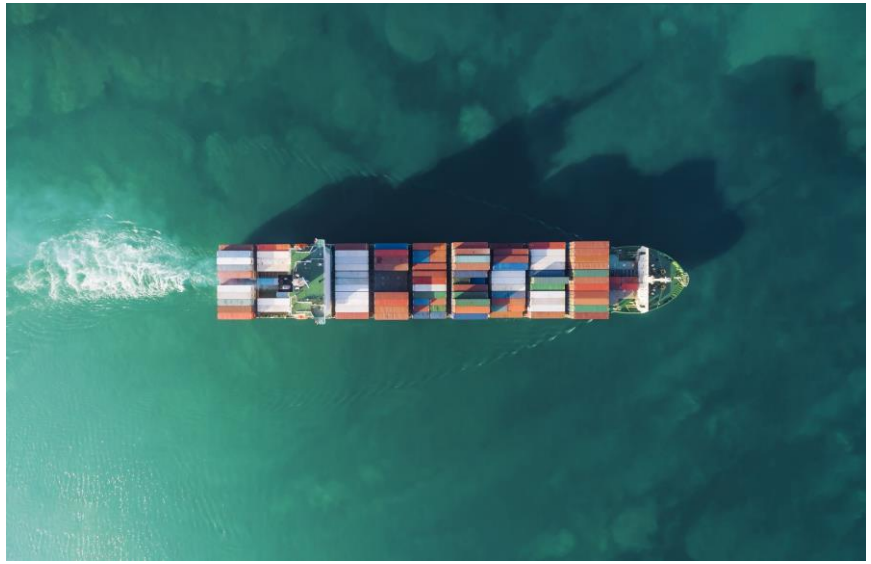
Sustainable

- Integration of regenerative processes
  - Sharing
  - Reuse
  - Repair
  - Recycling
- Active in the development of circular processes
- Pay attention to resource consumption



Resilient

- Decentralized and Distributed
- Sustainability and circular economy
- Adaptability
- Emergencies
- Evolution of technologies

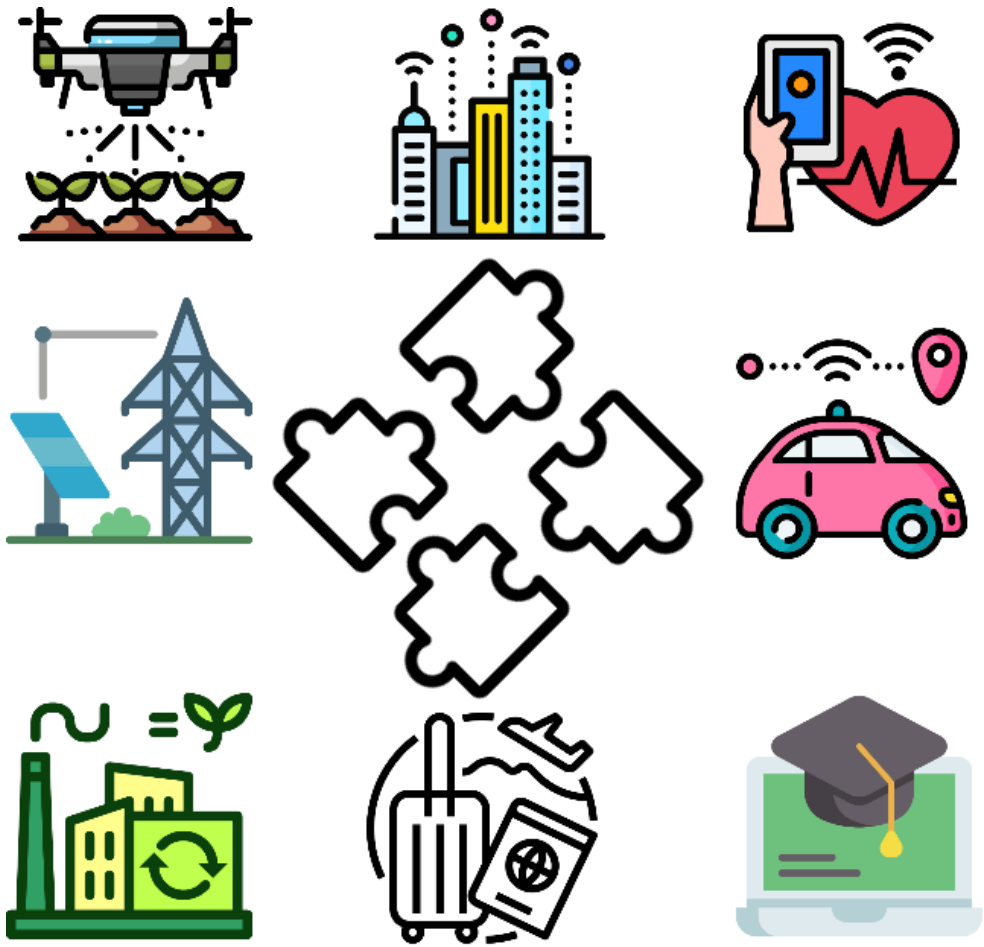


Human factor

- Training investment asset
- Technology that adapts to the individual
- Hyper-personalized services and goods
- Safety
- Cultural and generational inclusiveness

Cross Sector Integration in Society 5.0

- Interconnect Businesses
- Seamless Inclusive Experience
- Efficiency
- Enhanced Data Driven Processes
- Collaborative Scaling
- Resource Wise
- Feedback

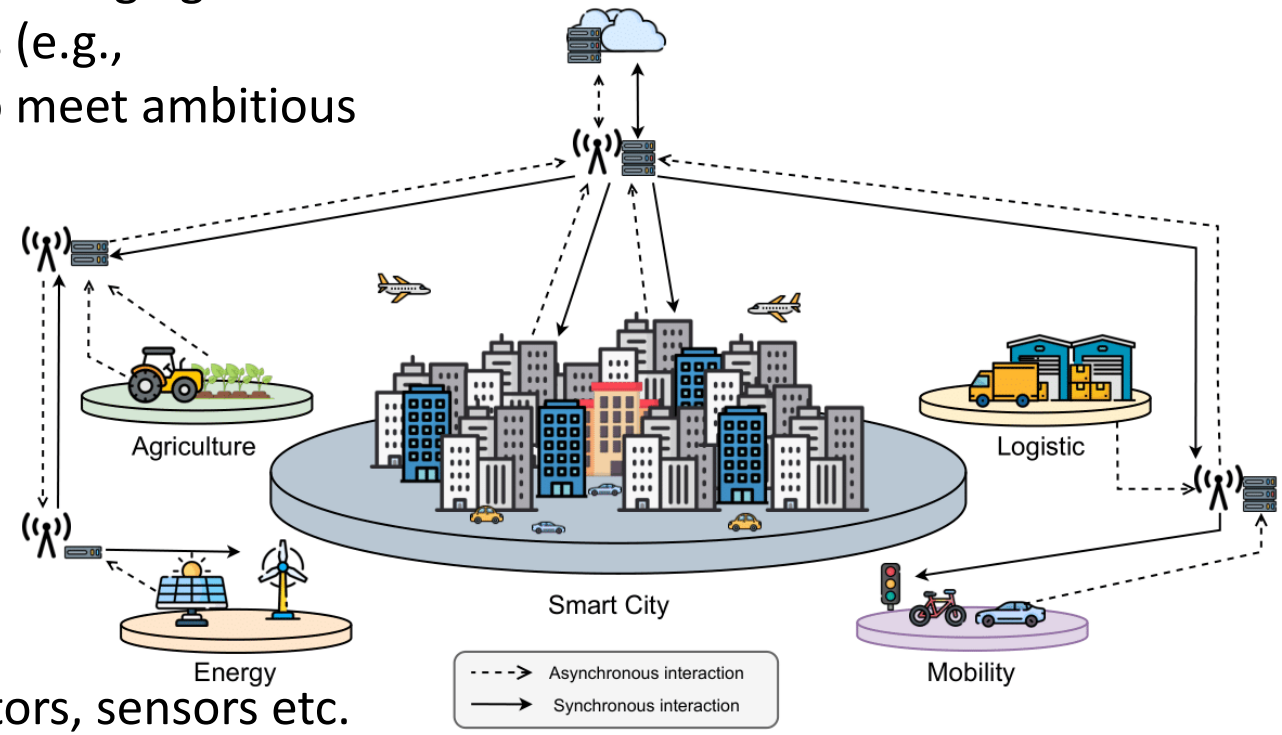


Industry 5.0

- Symbiotic Relationship
- Workforce upskilling
- Workforce well being
- Technology Transfer and Cooperation
- Enviromental Responsability

Society 5.0 enabled Smart Cities

Society 5.0 evolves the Smart City concept by encouraging cooperation among different sector stakeholders (e.g., Industry4.0, Smart Agriculture, Smart Grids…) to meet ambitious goals of **Sustainability** and **Resiliency**.



Some challenges:

- A multitude of business case
- Heterogeneity in terms of requirements
- Multi-region and geo-distributed
- Integration with (resource constrained) actuators, sensors etc.
- Scale up and down depending on demand

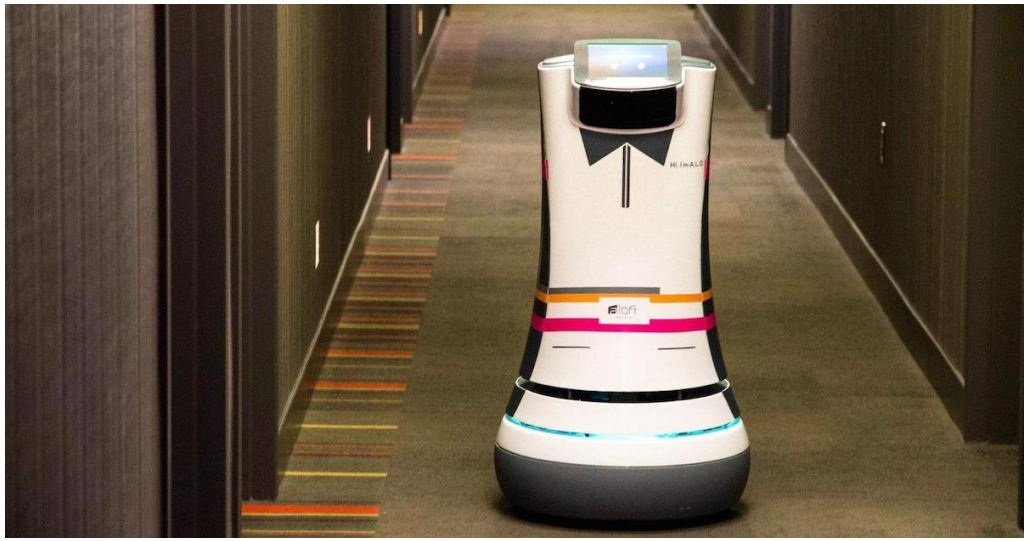
Smart Tourism

* https://smarttourismcapital.eu/

Heterogeneous Data and Service Integration: The use case of Smart Tourism

Smart Tourism challenges for **service and data Integration**:

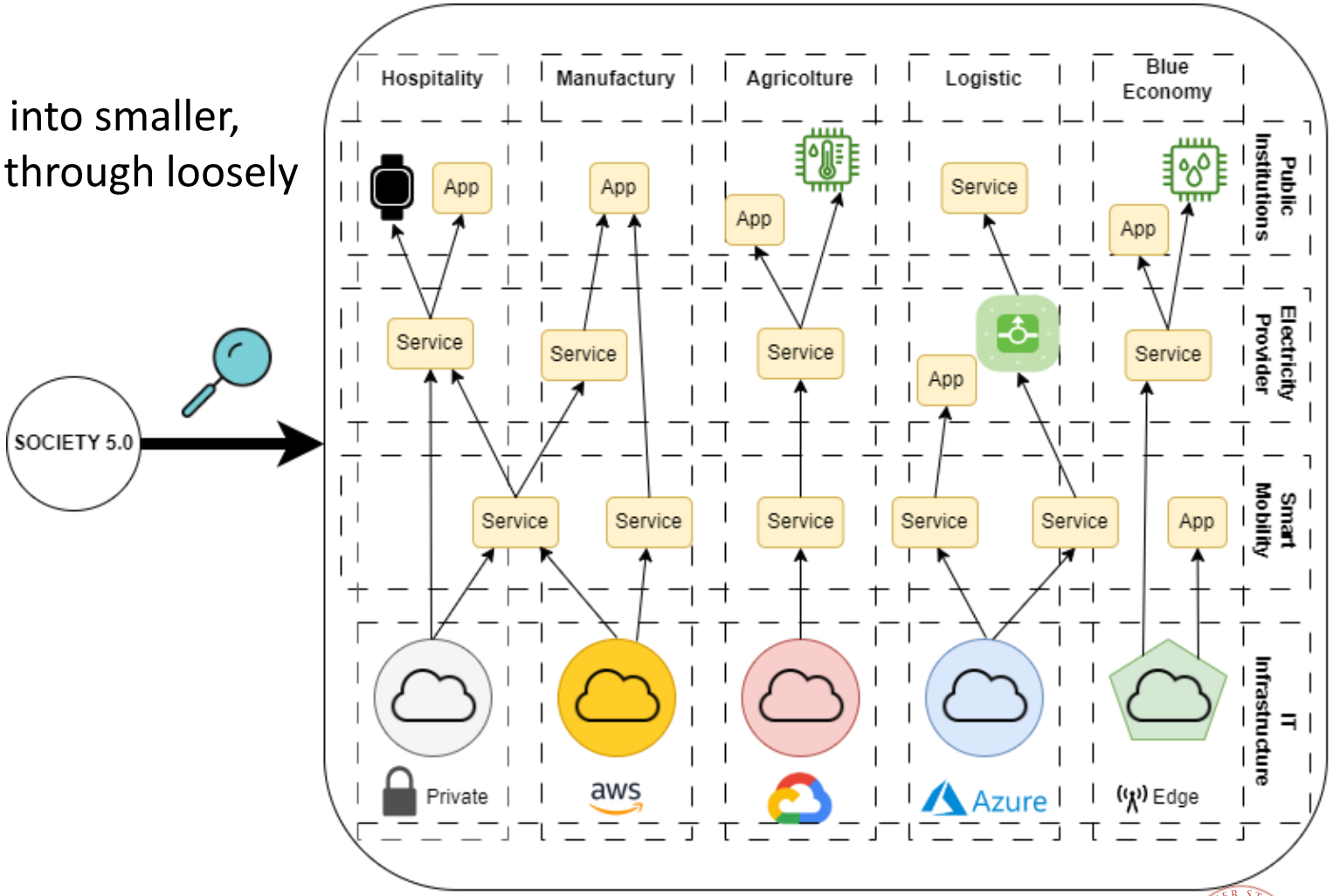
- A multitude of business case
- Heterogeneity in terms of requirements
- Multi-region and geo-distributed
- Integration with actuators, sensors etc.
- Scale up and down depending on demand



Microservice Integration in Society 5.0

Microservices decompose application into smaller, services that interact with each other through loosely coupled interfaces or APIs.

- Faster development
- Scaling
- Modularity
- Fault Isolation
- Technology flexibility



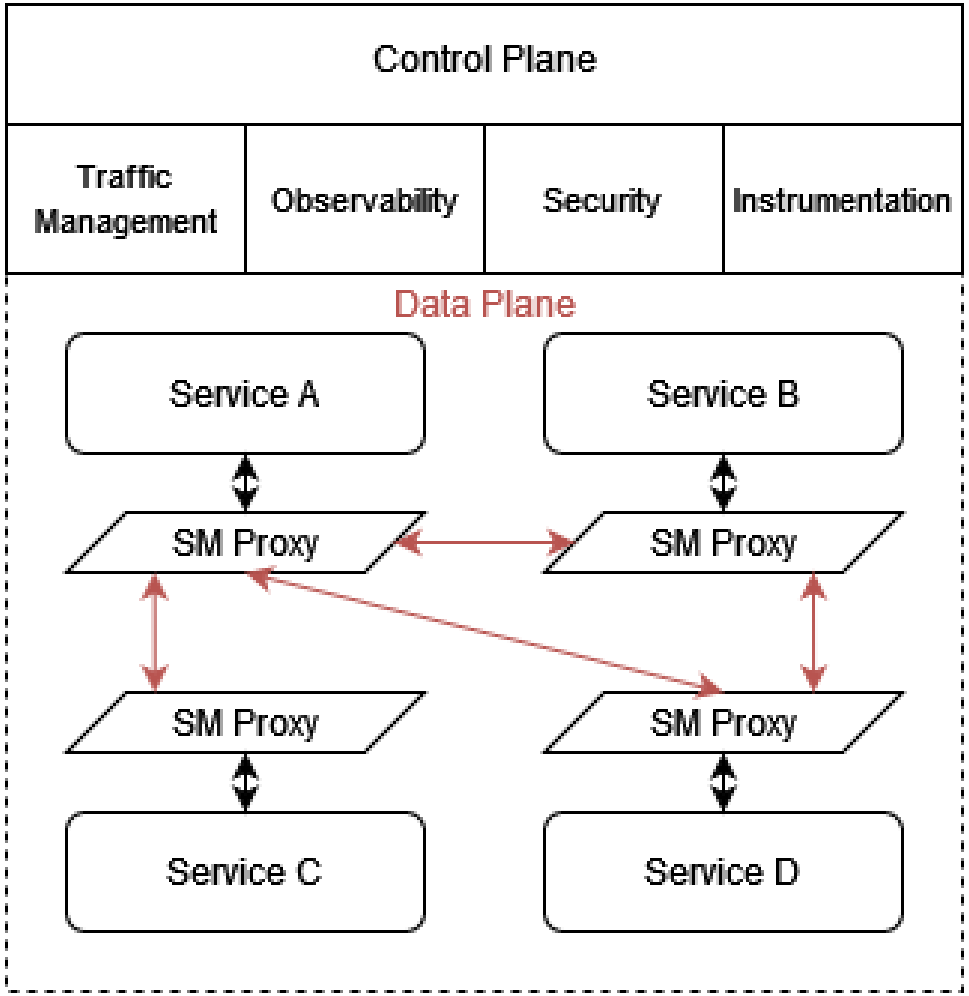
Distributed System Challenges

- Network reliability and capacity
- Dynamic topologies
- System as-a-whole observability
- Security concerns

Service Mesh

Service Mesh exploits a Proxy-based communication to provide as an infrastructural layer:

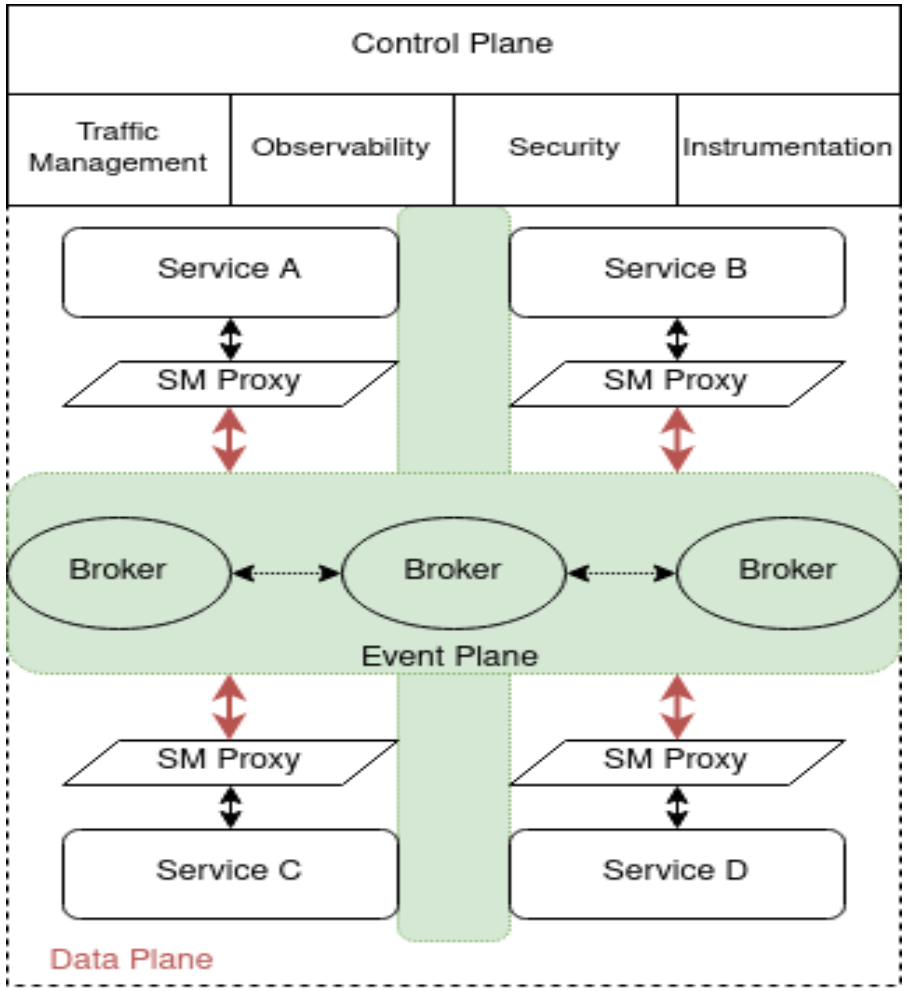
- Same-Host Communication abstraction
- Traffic management
    - Discovery
    - Load Balancing
- Observability
    - Metrics & Logs aggregation
    - Service Interaction monitoring
- **Resiliency**
    - Fault Recovery
    - Circuit braking
- Security



Event Mesh

Event Mesh exploits a broker mediated message exchange for Proxy-to-Proxy interactions

- Asynchronous service invocation
- Native Event Driven computation
- Location Transparent invocation
- Uncoupled interactions
- Different QoS
- Many-to-Many interaction



Serverless: some definitions

«Serverless computing is a method of providing backend services on an as-used basis. Servers are still used, but a company that gets backend services from a serverless vendor is charged based on usage, not a fixed amount of bandwidth or number of servers..»

[Cloudflare]

«A serverless architecture is a way to build and run applications and services without having to manage infrastructure. Your application still runs on servers, but all the server management is done by AWS. You no longer have to provision, scale, and maintain servers to run your applications, databases, and storage systems.»

[Amazon AWS team]

«A Serverless solution is one that costs you nothing to run if nobody is using it.»

[Paul Johnston, AWS]

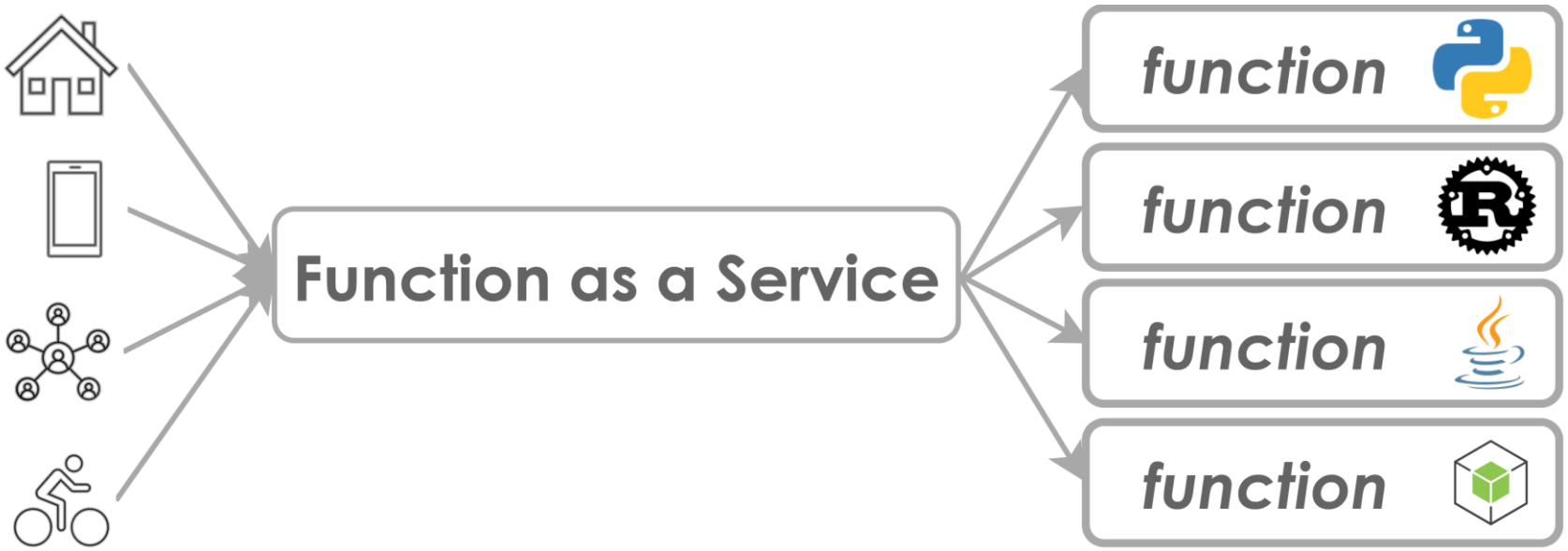
ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Key concepts

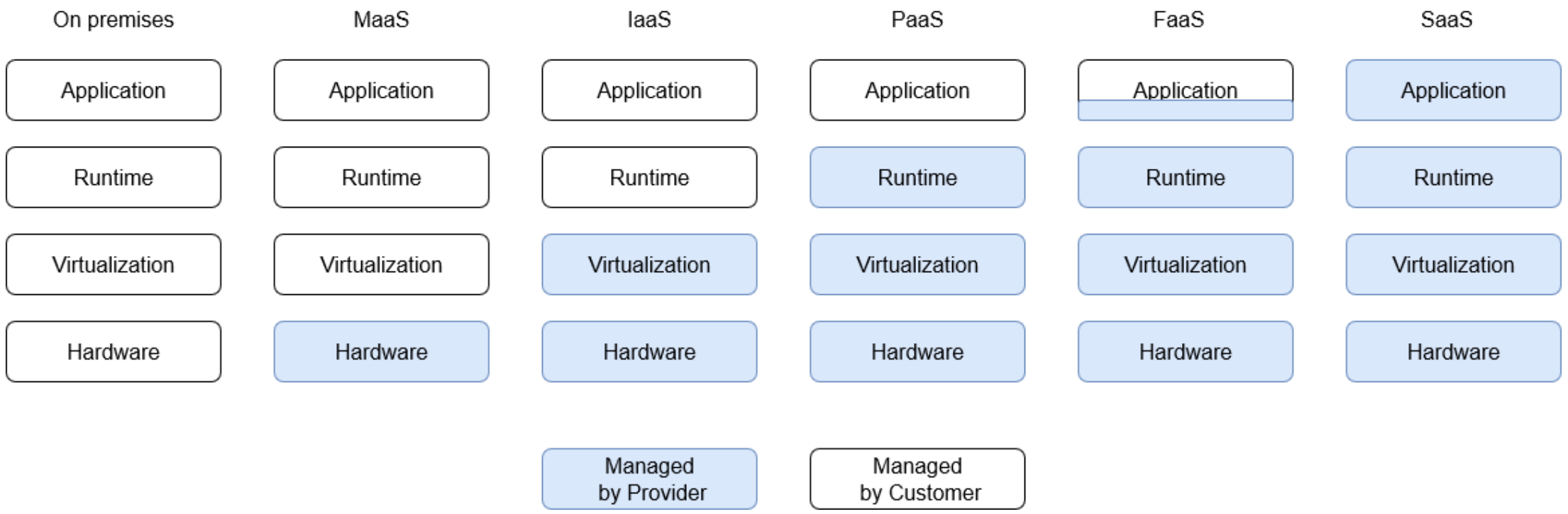
- Absence of control on scheduling and scaling logics
- No control on resources deployments
- Zero-Scaling (cost based on number of activations)
- Developer focuses only on business logic development

Function as a Service(FaaS)

Event-centric computing model where user-defined business logic (functions) is triggered and dynamically instantiated by incoming events.



Cloud computing models



Serverless Computing: Function as a Service (FaaS)

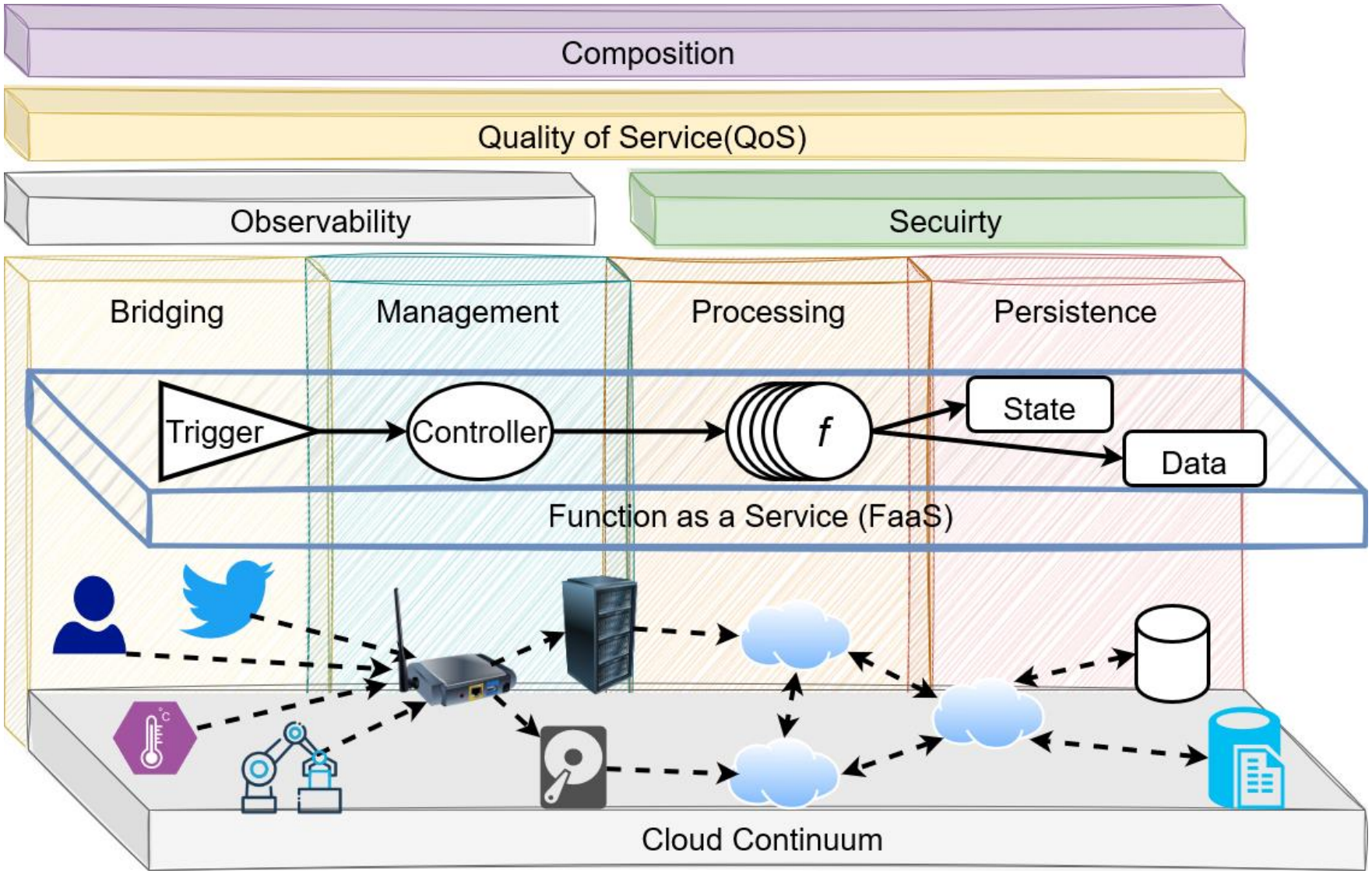
Serverless Computing is a family of cloud computing models providing total abstraction over running Infrastructure

- Abstract from Infrastructure heterogeneity
- Managed Lifecycle
- Customer focuses on business logics

Function as a Service (FaaS) is a Serverless Cloud Computing model in which a *function,* representing the customer defined logic, is automatically executed at the triggering of a specific event:

- General purpose model
- Event-driven
- Fine-grained scaling
- Built-in support to advanced features:
    - Composition
    - Event bridge pattern

Serverless Computing: Function as a Service (FaaS)



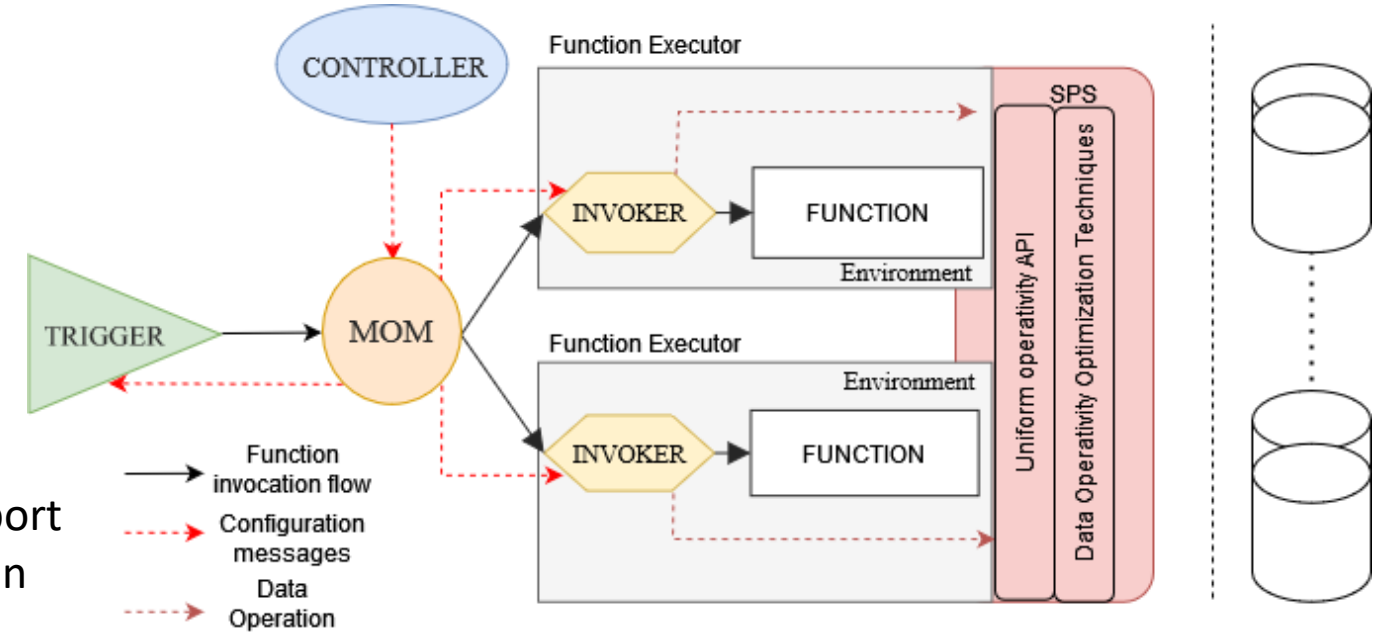
Data Persistence Layer [7]

Challenges:

- Many data sources and representations
- Different data storage technology coexisting
- Data ecosystem(s) integrating private/public stakeholders to leverage networking effects
- Function **ephemerality** hinders standard optimization mechanisms

Architecture principles:

- Message Oriented Middleware (MOM) to support **decoupled** and **asynchronous** event distribution among components
- Controller outside of function invocation chain
- Separates business logics from protocol-specific implementations adopted by data storages
- Optimizations realized at the **infrastructural layer**



Data Persistence: Experimental Validation

Comparison between SPS supported FaaS environment vs. data operations executed **Natively** at the business logic layer

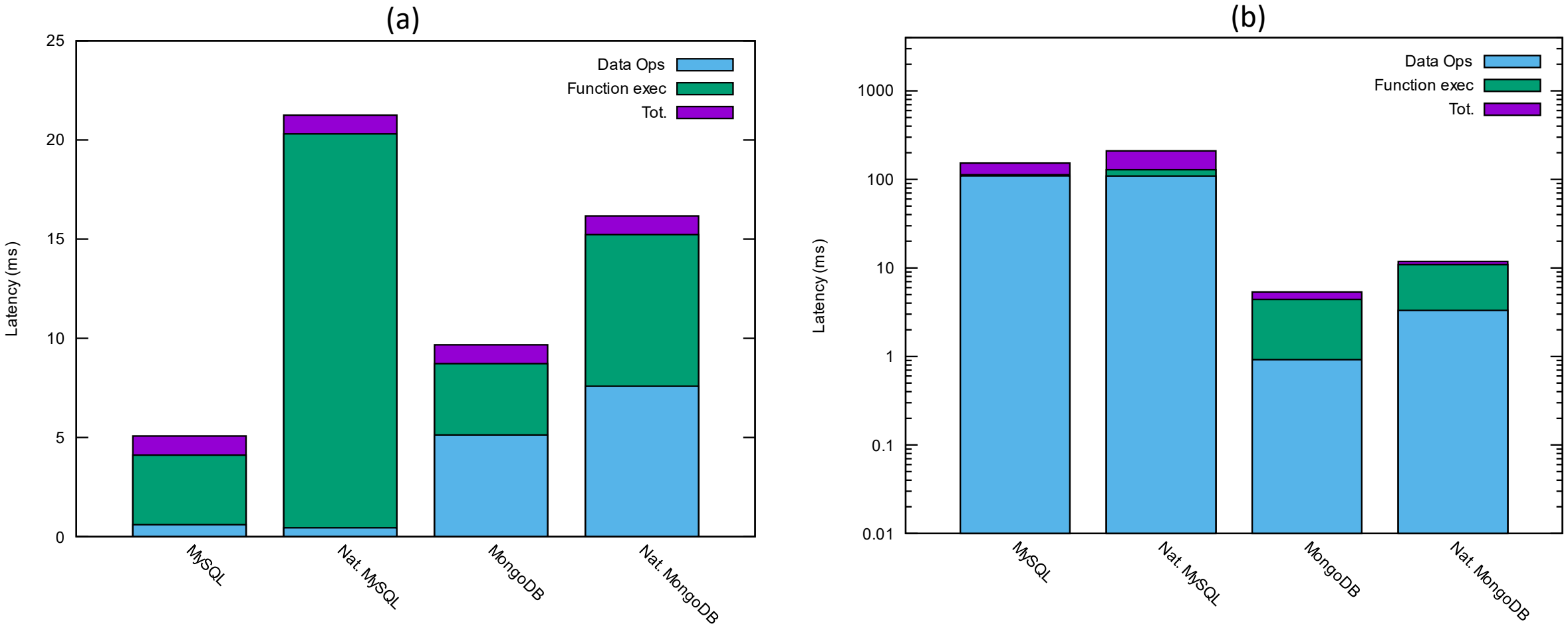
Evaluation scenarios:

Constant-rate stream of requests for different chain lengths

Evaluate performance when operating on different data persistence technologies

N. Nodes	Role	CPU	MEM	O.S.
1	Traffic Generator	4-core i5-3470	10 GB	Ubuntu 20.04
4	FaaS platform	4-core i5-3470	10 GB	Ubuntu 20.04

Data Persistence: Test Results



Average end-to-end response latency, function execution time, and database operation latency of the various configurations. The system is subjected to a constant load of 20 requests/second, assessing the performance of an (a) read operation and a (b) create operation (log scale).

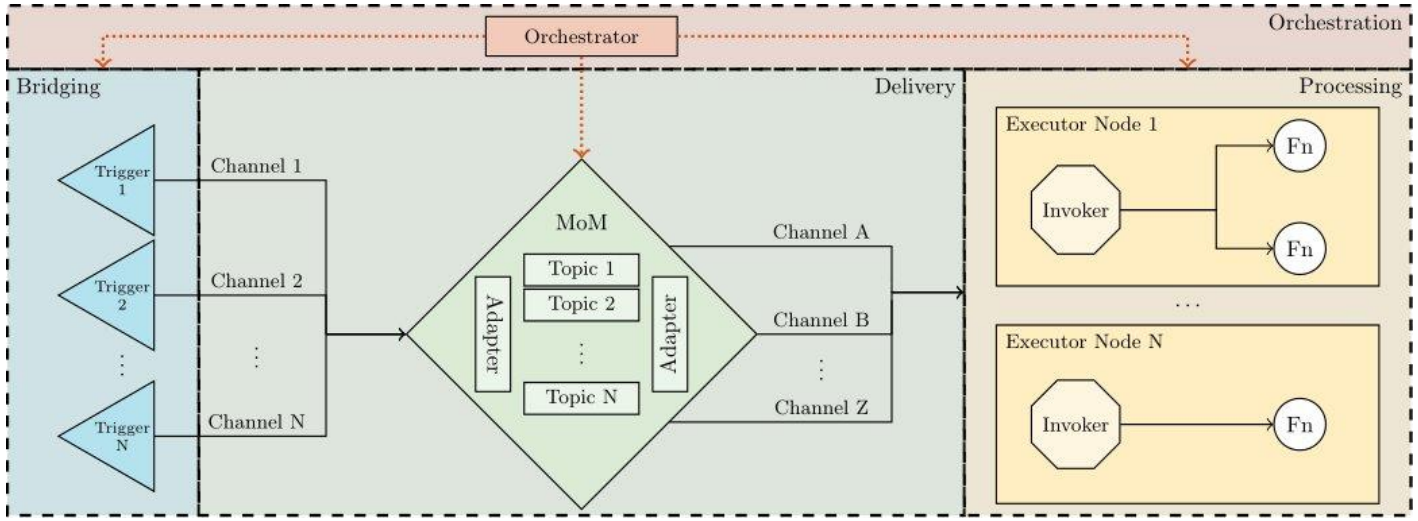
End-to-end QoS Service Differentiation [2]

Challenges:

- CC includes regions with **limited computational resource** availability
- **Ephemerality** of functions prevents classical prioritization mechanisms
- Heterogeneous mechanisms exists across the resources available in the CC

Architecture principles:

- End-to-end QoS **Orchestration**
- Abstracts from heterogeneous QoS mechanisms available in the CC
- Abstracts complexities by offering abstract quality levels FaaS workflows



End-to-end QoS Service Differentiation: Experimental Validation

Test capability of QoS layer in prioritize workloads execution on heterogeneous hardware

- In a regime condition
- Under heavy loads conditions

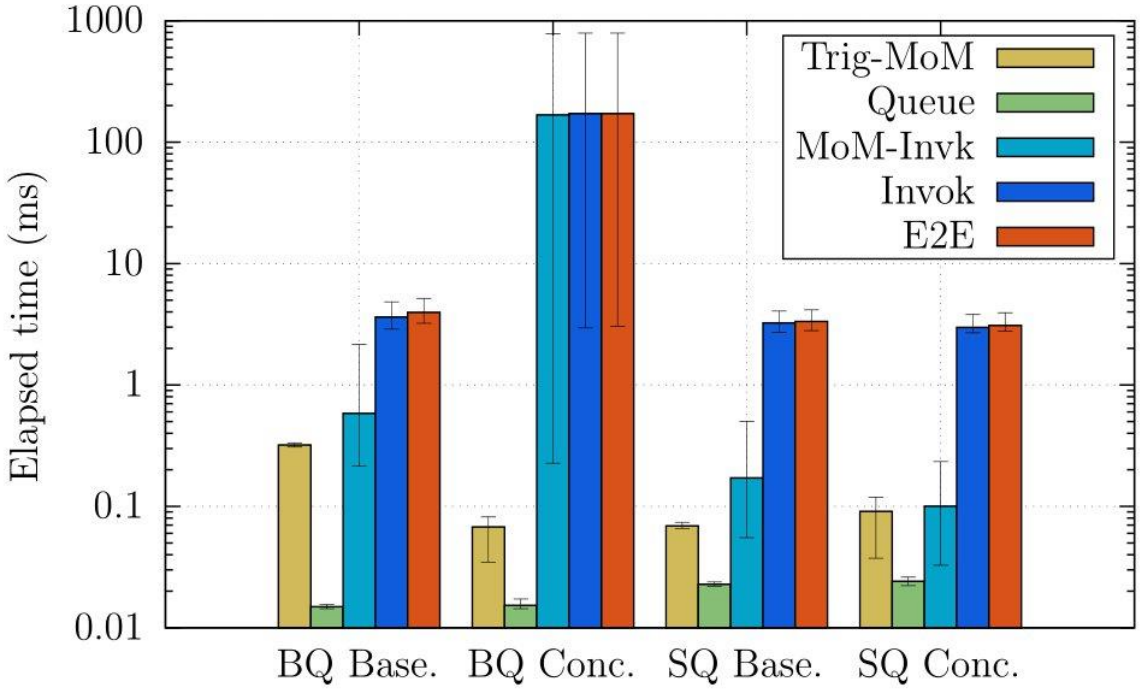
Prioritization mechanisms coordinated:

- Different *Trigger* instance in the **Bridging Layer**
- Time Sensitive Networking (TSN) and MOM queue prioritization in the **Delivery Layer**
- Separate *Invoker* instances and Linux Real Time Scheduler in the **Processing Layer**

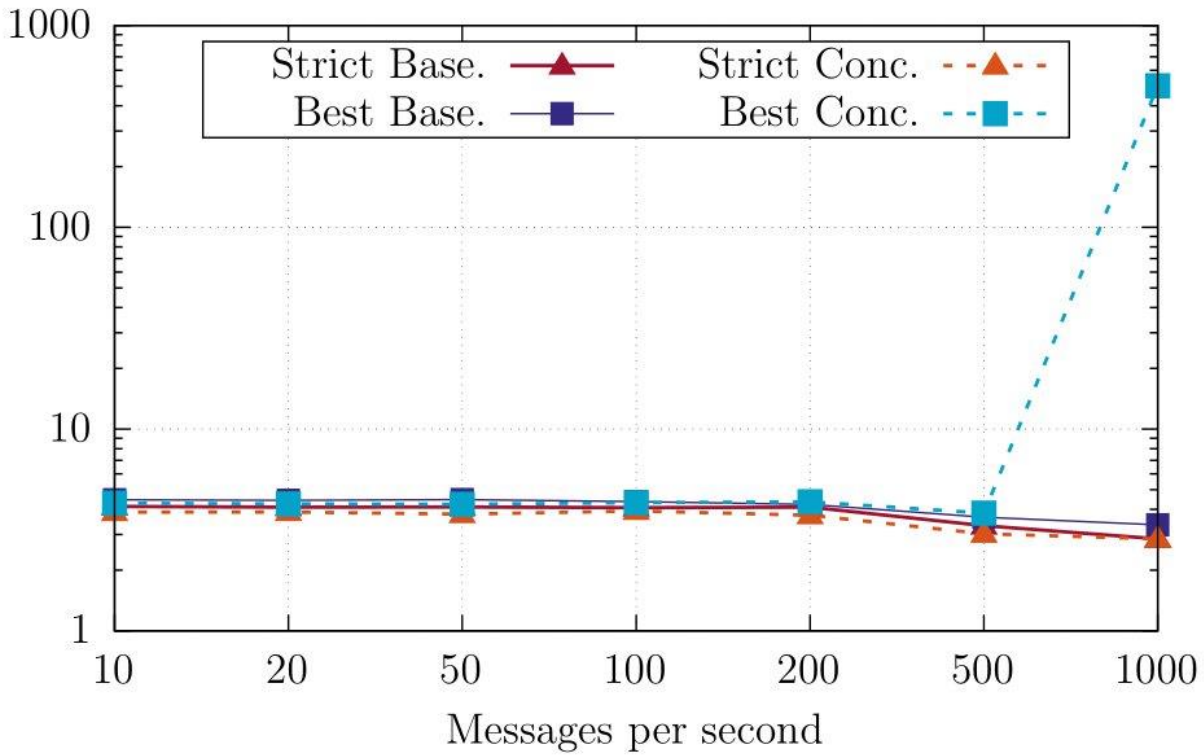
Node Tag	Model	CPU	Memory	TSN driver
A	Custom Workstation	AMD Ryzen 3700X 8/16 Core	32 GB	Intel I211
B	Dell Optiplex	Intel Core i5-3470 4/4 Core	10 GB	-
C	UP Core Plu()s	Intel Atom E3950 4/4 Core	8 GB	Intel I210
D,E	UP Xtreme	Intel Core i3-8145UE 2/4 Core	8 GB	Intel I210

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

End-to-end QoS Service Differentiation: Experimental Reults



Zoom-in on end-to-end latency results showing single contributions of TEMPOS components (execution time) to the overall response times



Comparison of end-to-end latency averages for BQ and SQ traffic executed when an increasing number of messages/seconds (from 10 to 1000) are submitted.

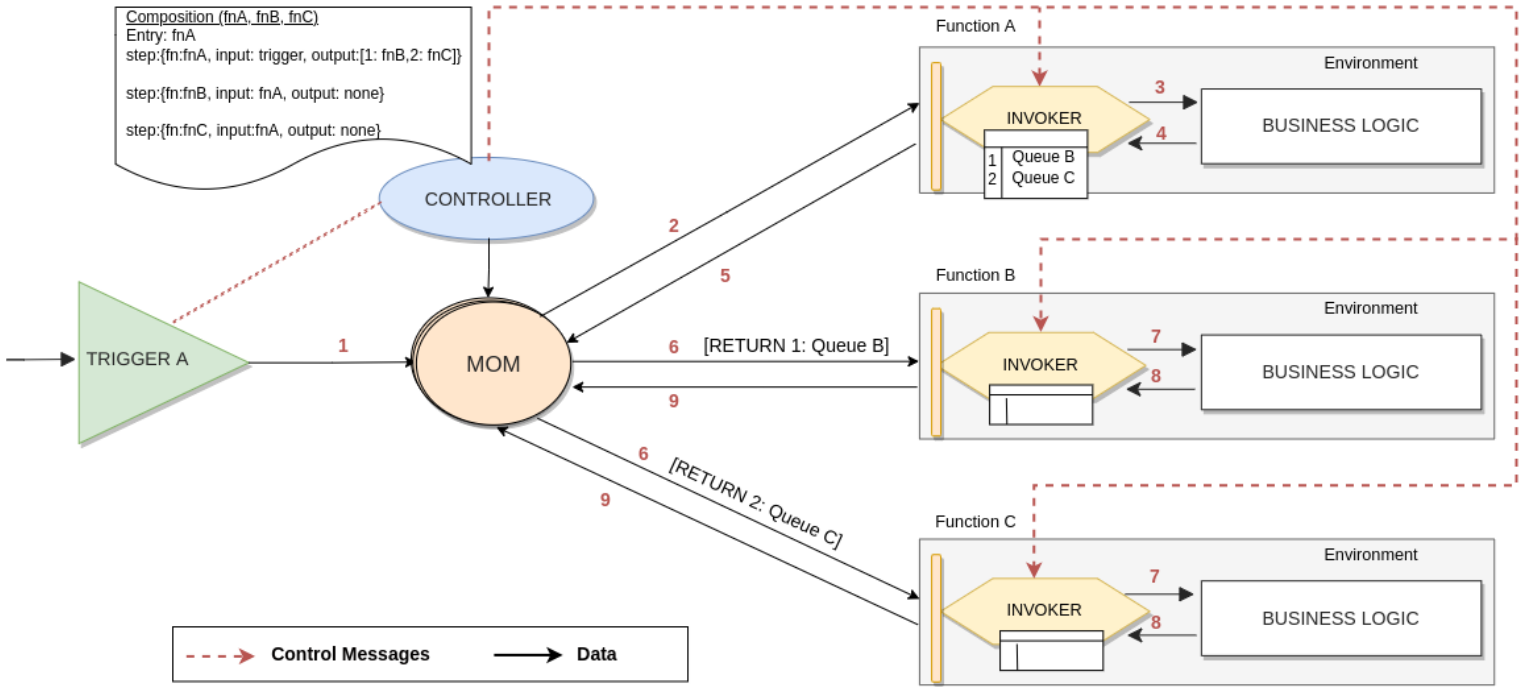
Distributed Function Composition [1]

Challenges:

- Composition coupled with business logic in function
- Centralized coordination of compositions
- Single Public cloud

Architecture principles:

- Distributed logic based on MOM primitives
- Can exploit multiple MOMs deployed in the CC
- Dynamic chains based on function outputs



Distributed Function Composition: Experimental Validation

Test behavior of different MOMs in supporting event distribution and function composition under different workloads:

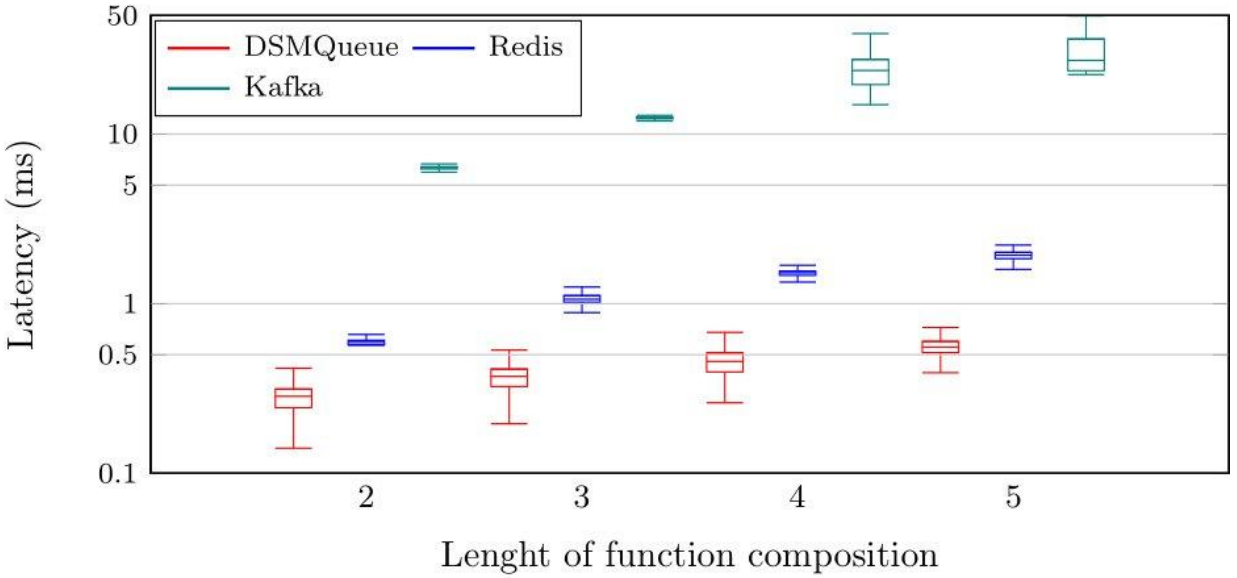
- End-to-End latency perceived
- System Throughput

Message Oriented Middleware considered:

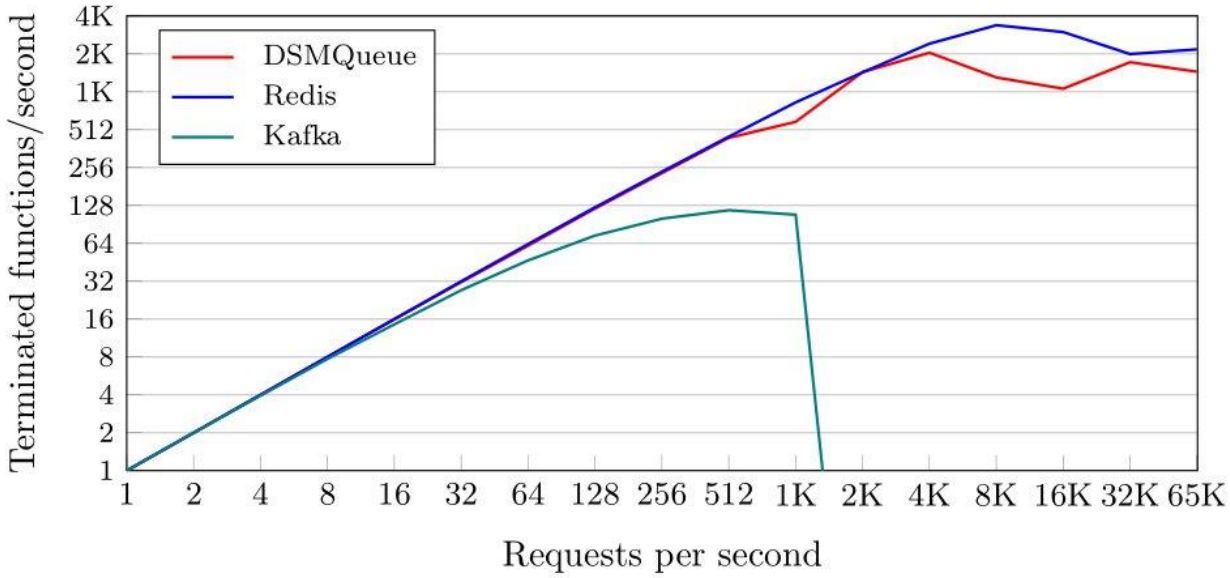
- Apache Kafka
- Redis Stream
- Distributed Shared Memory Queue (DSMQueue)

Test Bed Characteristics: The experiments are conducted on two identical machines, each equipped with a 4-core i5-3470 CPU @ 3.20GHz, 10GB RAM, running Ubuntu 20.04. The two nodes are directly interconnected by a 100Gbps Mellanox ConnectX-6 DX NIC, which supports both standard Ethernet traffic and **RDMA** network.

Distributed Function Composition: Experimental Results



End-to-end function composition latency at a steady regime.



End-to-end latency under a linearly increasing workload.

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Q&A

Andrea Sabbioni

Department of Computer Science and Engineering, DISI
University of Bologna, Italy

andrea.sabbioni5@unibo.it

www.unibo.it