Edge Computing for IoT Application Scenarios

RESCOM Summer School, Le Croisic, France
June 23, 2017

Paolo Bellavista
Mobile Middleware Research Group – http://middleware.unibo.it
Dept. Computer Science&Engineering (DISI)
University of Bologna, Italy
(currently visiting Professor at UPMC, France)

Credits to Giuseppe Carella, Luca Foschini, Carlo Giannelli, and Alessandro Zanni
Edge Computing for IoT Applications: Motivations

Number of connected devices worldwide continues to grow (triple by the end of 2019, *from 15 to 50 billions*)

Deep transformation of how we organize, manage, and access *virtualized distributed resources*

Is it reasonable that we continue to identify them with the *global location-transparent cloud*?

In particular, in many IoT application scenarios:

- strict *latency* requirements
- strict *reliability* requirements
  - For instance, *prompt actuation of control loops*
  - Also associated with *overall stability and overall emerging behavior*
Edge Computing: Definition (to be discussed…)

Edge computing = *optimization of “cloud computing systems”* by performing data processing (only?) at the edge of the network, near datasources. *Possibility of intermittent connectivity*

Edge computing can include technologies such as wireless sensor networks, mobile data acquisition, mobile signature analysis, cooperative distributed peer-to-peer ad hoc networking and processing, distributed data storage and retrieval, autonomic self-healing networks, remote cloud services, …
Edge computing = **optimization of “cloud computing systems”** by performing data processing (only?) at **the edge of the network**, near datasources. *Possibility of intermittent connectivity*

IMHO, crucial to have **virtualization techniques at edge nodes**

Synonyms = mobile edge computing, fog computing, cloudlets, …
MEC is bringing computing close to the devices (in the base stations or aggregation points)

- **On-Premises:** the edge can be completely isolated from the rest of the network
- **Proximity:** capturing key information for analytics and big data
- **Lower Latency:** considerable latency reduction is possible
- **Location awareness:** for location-based services and for local targeted services
- **Network Information Context:** real time network data can be used by applications to differentiate experience

Notable example: ETSI Mobile Edge Computing (MEC)
Local vs Global: the MEC Use Cases

Depending on the integration with the core network three types of use cases are defined:

- **Private Network Communication** (factory and enterprise communication)
  - Providing support for on-premises low-delay private communication
  - Providing secure interconnection with external entities

- **Localized Communication** (traffic information and advertisements)
  - Providing support for localized services (executed for a specific area)
  - Specific ultra-flat service architectures

- **Distributed Functionality** (content caching, data aggregation)
  - Providing extra-functionality in specific network areas
Also directions of ongoing research towards the merging of:

- **Mobile Edge Computing (MEC)**  
  e.g., ETSI standardization
- **and fog computing approaches**  
  e.g., Foud for V2G or MEFC (see reference section)

“Only” stronger accent on standard protocols (MEC), content caching (MEC), data aggregation (fog), distributed control (fog), orchestration of virtualized resources (both), mobile offloading (?)
Edge Computing for IoT Apps: Quality Requirements
Edge Computing for IoT Apps: Quality Requirements

Towards the vision of **efficient edge computing support** for “**industrial-grade**” IoT applications

- Latency constraints
- Reliability
- Decentralized control
- Safe operational areas
- Scalability
Edge Computing for IoT Apps: Research Directions

1. Architecture modeling
2. **Quality support even in virtualized envs**
3. **Scalability via hierarchical locality management**
4. **Distributed monitoring/control functions** at both cloud and edge nodes to ensure safe operational areas

But also:
- Data aggregation
- Control triggering and operations
- Mgmt policies and their enforcement
- ...
1) Architecture Modeling

**Dynamic distribution of storage/processing (network resource allocation?) functions in all the three layers** of a node-edge-cloud IoT deployment environment

Different and richer concept of **mobile offloading**
1) Architecture Modeling

Need for new models
Need for new models for richer mobile offloading:
- From sensors/actuators to the cloud (traditional)
- **From sensors/actuators to the edge**
- From the edge to the cloud

But also:
- **From the cloud to the edge**
- From the edge to sensors/actuators

*Growing overall status visibility vs. growing decentralization and autonomy*
For example, Network Function Placement

Through **edge cloud computing**:

- Network functions can be deployed in both *edge nodes and central node*
- Edge controller has to be very simple to manage a limited set of devices (energy efficiency, compute limitations)
  - **Dynamic decisions** about where to execute functionalities, depending on
    - state of subscribers
    - network congestion
    - single device/group) mobility pattern
  - **Autonomic functioning of edge nodes**
    when no backhaul is available / backhaul communication is interrupted
- Policy-based functioning of edge networking for making decisions when edge routing is used
Open source software automating the deployment of applications on top of containers:

- uses the isolation mechanisms provided by the Linux kernel like cgroups and namespaces allowing multiple “containers” to execute on the same physical host without having to use virtualization techniques
- can be integrated within OpenStack as a different type of hypervisor

Edge computing empowered by containerization
Edge computing empowered by containerization
Edge computing empowered by \textit{containerization}

Someway similar approach, but \textit{more lightweight for resource-limited IoT gateways:}

- \textit{Kura Gateways with Docker containerization} and our simplified orchestrator
- Experimentation with \textit{Raspberry PI nodes and Docker containerization}
- Efficient, flexible, and incremental usage of Docker images, layers, … via ad hoc repositories

For additional details, please see our papers (refs section)
2) Quality Support even in Virtualized Envs

But definitely, here we are not starting from scratch…

*Notable experience of mobile cloud networking for telco services with quality requirements*

- *Carrier-grade industrial usage of elastic* distributed cloud resources for telco support infrastructures
- *Quality constraints of typical telco providers*
  - *Latency*
  - *Scalability*
  - *Reliability*
First lesson learnt: sufficient quality levels?

In the last years, growing industrial interest in Mobile Cloud Networking (MCN) as the opportunity to exploit the cloud computing paradigm through Network Function Virtualization (NFV)

- primarily with the goal to reduce CAPEX/OPEX for future mobile networks deployment and operation

Risk/skepticism:

a virtualized infrastructure could not reach the levels of service reliability, availability, and quality usual for mobile telcos

EU MCN project – http://www.mobile-cloud-networking.eu
First lesson learnt: sufficient quality levels?


Large experimental campaigns and results from **wide-scale industrial testbeds** have demonstrated that it is possible via the adoption of advanced techniques for:

- **lazy coordination** of distributed cloud resources
- **standardized** virtualization of network functions
- **proactive mobility-aware resource management**, including load balancing, handovers, …
- **interoperable orchestration** of **infrastructure+service** components
EU Mobile Cloud Networking Project: Network Functions as a Service

FP7 Integrated Project (2013-2016) targeted to bringing cloud computing features to mobile operator core networks (e.g., EPCaaS):

- Virtualization of components
- Software defined networking
- Elasticity
- Infrastructure sharing
- Redefining roaming

End user → RAN → Mobile Core → Data Centre

"as a service"

Cloud Computing - service enabler

Service provider

Mobile Cloud Networking

End user → RAN → Mobile Core → Data Centre

"as a service"

Cloud Computing - service enabler

Service provider
Motivations: Why NFV is needed?

① **Virtualization**: use network resource without worrying about where it is physically located, how much it is, how it is organized, etc

② **Orchestration & Automation**: configuration through complied global policies versus the current manual translation and per device download

③ **Programmability & Openness**: modular design allows evolvability and customization to own choices

④ **Dynamic Scaling**

⑤ **Visibility**: Monitor resources, connectivity

⑥ **Performance**: Optimize network device utilization

⑦ **Multi-tenancy**: Should be able to serve new business models

⑧ **Service Integration**: seamlessly integrating interdependent services

Source: [www.cse.wustl.edu](http://www.cse.wustl.edu)
The objective of NFV is to translate the classic network appliances to software modules

- Running on high volume servers with high volume storage
- Interconnected by generic high volume switches
- Automatically orchestrated and remotely installed

NFV is a novel paradigm that presumes that the network functions

- Are implemented only as software (programs)
- Can run on top of common servers

NFV has to fix the following main issues:

- **Performance**
- **Co-existence, portability, and interoperability**
- **Automation**
- **Scalability**
NFV and SDN as the support technologies for 5G

5G will be based on slices on top of same infrastructure

NFV and SDN as the main enablers for:

- **business agility** – with its capabilities for on-demand, fast deployments
- **network adaptability and flexibility** – requires redesign of network functions (to cloud native), support for functions variance, flexible function allocation, etc.
- **composition** – putting multiple services together in a slice – end-to-end management
- **slicing** – separation at network level
- **programmability** – software-only network functions and their interaction with physical systems

➔ **Orchestration** is the cornerstone for all of these features
NFV and SDN

- NFV requires network functions to be implemented as software on top of common hardware
- SDN brings remote programmability of the network
- NFV/SDN platform acts as an end-to-end middleware between:
  - A distributed heterogeneous infrastructure for compute and storage
  - Interconnected through a controlled network
  - Generic network functions implemented in software running in isolated containers/virtual machines
    - VPNs, NATs, DNSs, IMSs, EPCs, Application Servers, etc.

The main value added differentiator between different solutions is the quality of the software
- how well it can solve the specific service needs
NFV Architecture Blue print is ready since Nov. 2012…
“…The business case for this emerging technology has yet to be proven for data centre CIOs and CSP CTOs. They require de facto standards, full interoperability, and use cases that are proven in the field. However, operational efficiencies, quicker time to market for newer applications, and newer revenue-share business models may result from this technology.”

Source: Gartner 2015
The worldwide NFV market will grow from USD181 million in 2013 to USD2.4 billion in 2018.

Spending on this technology is more likely to increase significantly after 2018, if the following issues are addressed:

- OSS
- Security
- Pricing
NFV Architecture Blueprint is ready since Nov. 2012…
The NFV Ecosystem
Virtualized Infrastructure Manager (VIM)

- Responsible for the *lifecycle management* of compute, storage and network resources provided by the NFVI

- It is basically a **Cloud Management System** which exposes an API for standard CRUD operations on those resources

- **OpenStack** is the *de facto standard implementation* of this functional block
VNF Manager (VNFM)

- Responsible for the lifecycle management of Virtual Network Function instances
  - One per NF
  - One per multiple VNF instances even of different type
- It has to support the:
  - VNF instantiation
  - VNF configuration
  - VNF update
  - VNF scaling in / out
  - VNF instance termination
Network Function Virtualization Orchestration (NFVO)

- Responsible for the **lifecycle management of Network Services**:
  - In a single domain
  - Over multiple datacenters
- Applies policies for resource utilization
- Requests the instantiation of VNFs via the VNF Managers
The OASIS TOSCA Technical Committee works to enhance the portability of cloud applications and services.

TOSCA will enable the *interoperable description of application and infrastructure cloud services*, the relationships between parts of the service, and the operational behavior of these services (e.g., deploy, patch, shutdown) - independent of the supplier creating the service, and any particular cloud provider or hosting technology.

TOSCA will also make it possible for higher-level operational behavior to be associated with cloud infrastructure management.
Two approaches in regard to orchestration were taken:

1) Orchestrating from the infrastructure perspective

   *Extending VIM towards service orchestration.* Missing:
   - Adaptation to complex network services requirements, e.g. fault management, scaling, network function placement, virtual network configuration, information flow paths, security, reliability

2) Orchestrating from the network service perspective

   *Extending the Network Management System to handle orchestration.* Missing:
   - Capitalize through native components on cloud opportunities: scaling, dynamic resource allocation
   - Define the appropriate network service KPIs, end-to-end fault management, end-to-end reliability insurance, etc.
What is OpenBaton?

OpenBaton is Open Source implementation of the ETSI MANO specification

OpenBaton aims to foster, within the NFV framework, the integration between:

- **Virtual Network Function providers**
- **Cloud Infrastructure providers**

**Functionality:**

- Installation, deployment, and config. network services
- Runs on top of multi-site OpenStack
- Provides independent infrastructure slices
- Support for generic or specific VNF management

Designed for answering R&D requirements

- Easy to configure and to deploy
- Providing a centralized view of the testbed

**github:** [https://github.com/openbaton](https://github.com/openbaton)
What OpenBaton stands for

- **No vendor lock-in**: OpenBaton does not contain any vendor specific features. It follows open specifications and it is open to the community

- Built from scratch following the **ETSI MANO specification**
  - The NFVO uses the ETSI NFV data model internally for the definition of the Network Service and Virtual Network Descriptors

- **Allows interoperability**
  - Being interoperable is one of the challenges brought by the fragmented ecosystem in the management and orchestration area. It requires a lot of work to make two different vendors solution working together → need of a single vendor-independent platform
OpenBaton

OpenBaton is based on the ETSI NFV MANO v1.1.1 (2014-12) specification. It provides:

- A **NFV Orchestrator** managing the lifecycle of Network Service Descriptors (NSD) and interfacing with one or more VNF Manager(s) (VNFM)
- A **generic VNF Manager**, which can be easily extended for supporting different type of VNFs
- A **set of libraries** which could be used for building your own VNFMs (vnfm-sdk)
- A **dashboard** for easily managing all the VNFs

It currently integrates with OpenStack as main VIM implementation
The Fraunhofer FOKUS 5G Playground

5G Playground: A comprehensive testbed environment for prototyping 5G-ready VNFs using OpenBaton orchestration

- Open5GCore providing the next wireless system beyond LTE/EPC with more efficient communication for the subscribers and improved automation/reliability (applying SDN and NFV principles)
- Open5GMTC enabling connectivity management and end-to-end service establishment for a huge number of connected devices
- OpenSDNCore enabling SDN experimentation for data path, backhaul networks or customized network environments

- All those are software components and can be customized, deployed and configured on demand via OpenBaton enabling automatic just-in-time test environment creation, experimentation and demonstration
3) Scalability via Hierarchical Locality Management

For IoT applications in particular, to achieve scalability but not only…

**Need for additional scalability based on:**

- **Locality identification**
- **Locality autonomy** (partial)
- **Locality coordination**
  - *Hierarchical organization as simple tradeoff of practical usage*

Still quite uncovered research area, in particular with no industry-grade implementations, also in “more traditional” IoT gateways
3) Scalability via Hierarchical Locality Management

*Multi-layers hierarchy* (each node specifies domain/group to limit the interest towards external resources):

- **Level0** includes the root node, enables inter-localities communications
- **Level1** includes all the nodes belonging to a specific domain, updates level2 about hierarchy modification (quicker update)
- **Level2** includes all the nodes of a given domain and (sub)group, receives periodically updates by level1
Industry-grade Integration with IoT Gateways: Eurotech Kura

- **Kura framework** for building gateways for IoT applications
- Design/implementation **abstraction** of real-world scenarios complexity (heterogeneous hardware/network devices)
- Large set of network protocols to communicate with lower-layer
- **Java OSGi** for dynamic management of software components via self-contained pluggable packages (i.e., bundles)
- Support for **VPN, NAT, and firewalls**
- **Open-source** with a fervent community
Traditional Usage of Kura
MQTT-CoAP Interworking in our Extended Kura Gateway

- Design and implementation of a **scalable distributed architecture** for the dynamic management of IoT resources **via hierarchical localities**
- Gateway coordination via integration of emerging standard protocols, i.e., **MQTT** and **CoAP**:
  - MQTT natively integrated into Kura
  - MQTT non-negligible limitations in terms of scalability
  - Introduction of more lightweight CoAP-based functionality, thus achieving scalable interactions
  - Improvement for system dynamic management (e.g., resource/device discoverability, resilience to disconnections, dynamic reconfiguration)

- **What about virtualization support in Eurotech Kura/Kapua?**
4) Distributed monitoring/control for *autonomous and safe operational areas*

Starting from the idea of *geometric monitoring* by A. Schuster, Technion, Israel

Definition of safe operational areas in *different flavors*:
- **No need of monitoring updates**
  (reduction of monitoring overhead)
- **Triggering of cloud-assisted update** of overall global monitoring status
- …
4) Distributed monitoring/control for autonomous and safe operational areas

Definition of safe operational areas in *different flavors*:

- …

- **Self-adaptive control functions** that can operate autonomously over a locality

- **Triggering of cloud-assisted coordination** of autonomous control functions

- Triggering of richer forms of **mobile offloading**? …
4) Distributed monitoring/control for autonomous and safe operational areas

Other open directions for research work:
• Transformation of any geometric shape in the monitoring space of interest into a \textit{linear combination of convex spaces}

• Usage of \textit{safe operational areas as autonomously verifiable regions for self-adaptation} with no need of coordination

• \textit{Safe constraints in IoT} application domains and their \textit{static/dynamic verifiability} before the enforcement of adaptation actions

• Local control/stability vs. global system control/status
Another lesson learnt from experience: Innovative Business Models

Suitability of *coupling efficient technical solutions with sound and effective business models*

- capable of turning the advantages of cloudification/virtualization into market competitive pros

In the MCN project:

- Different **RANs**
- **Macro DataCenters (DCs)** as standard large-scale computing farms deployed and operated at strategically selected locations
- **Micro DCs** are medium- to small-scale deployments of server clusters (*edge nodes?*) across a certain geographic area, for instance covering a city or a certain rural area and as part of a mobile network infrastructure

With **different possible ownerships and operation ways** of these infrastructure elements
Need for Innovative Business Models

Examples:
1. A mobile network operator may operate several RANs, mobile core networks, as well as DCs, and thus enjoy full control of all technology domains.
2. More advanced - a company (e.g., a mobile network operator, a DC provider, or any other enterprise) acts as end-to-end MCN provider without owning and operating any physical infrastructure, by signing wholesale agreements with, for example, mobile network carriers. The same would be for contracting DC operators in strategic locations to complete a full MCN offering (RAN, Mobile Core, DC).

The distinctive aspect of the latter is that a MCN provider exploits the MCN architecture to compose and operate a virtual end-to-end infrastructure and platform layer on top of a set of fragmented physical infrastructure pieces provided by different mobile network and DC owners/operators, thus providing a differentiated end-to-end MCN service (mobile network+compute+storage).
As a consequence, in the MCN project…

_Lowering expenses_ is a common industry practice but ultimately only _novel services and business models_, with clearly perceivable added value, will sustain healthy Average Revenue Per Unit (ARPU)

**End-to-End (E2E) MCN services:**

on-demand, elastic, and metered mobile network + compute + storage services (*-as-a-Service - *aaS)

- **Wireless-aaS**, enabled by Remote Access Network virtualization, with Base Band Units deployed on-demand on elastic IaaS running on top of micro DC close to antennas
- **Evolved Packet Core (EPC)-aaS**, i.e., on-demand deployment of distributed EPC instances on top of elastic IaaS on micro and/or macro DC
- **IP Multimedia Subsystem (IMS)-aaS**, i.e., on-demand deployment of IMS instances for complementing voice/video services on top of elastic IaaS on micro and macro DC
E2E MCN Services

- On-demand and elastic content/storage/application distribution services, on top of IaaS on micro and macro DC and exploiting cloud storage services (e.g., the Follow-Me cloud solution – CDN-aaS);
- End-to-end MCN service orchestration
- MCN Authentication Authorization Accounting, Service Level Agreements, Monitoring, Rating, and Charging
MCN E2E Services

MCN Services
- RANaaS
- EPCaaS
- IMSaaS
- DSSaaS
- CDN/ICNaaS

Atomic Services
- IaaS
  - Compute
  - Storage
  - Network

Support Services
- LBaaS
- DNSaaS
- AaaS
- MaaS
- MOBaaS
- DBaaS
- RCBaaS
- SLaaS
- AAAaaS

Business
- Design
- Agreement

Technical
- Design
- Implementation
- Deployment
- Provisioning
- Runtime & Mgt
- Termination
MCN Key Architecture Elements (1)

Service Manager
- Provides an external interface to the user
- Business dimension: encodes agreements
- Technical dimension: manages Service Orchestrators of a particular tenant

Service Orchestrator
- Oversees (E2E) orchestration of a service instance
- Domain specific component
- Manages service instance
- 'Runtime & Management' step of the Lifecycle
- One SO is instantiated per each tenant within the domain
- SO is associated with a Service Manager
- Monitors application specific metrics and scales
CloudController

- Supports the deployment, provisioning, and disposal of services
- Access to atomic services
- Access to support services
- Configures atomic services (IaaS)
MCN Key Arch Elements Overview

All are used throughout MCN
MCN Services and Arch Elements

MCN Services
- SM RANaaS
- SO EPCaaS
- SO IMSaaS
- SM DSSaaS
- SO ICN/CDNaaS

Cloud Controller

Atomic Services
- SM
- SO IaaS
  - Compute
  - Storage
  - Network

Support Services
- SM LBaaS
- SO DNSaaS
- AaaS
- SM MaaS
- SO MOBaaS
- SM DBaaS
- SM SLAaaS
- SO RCBaaS
- SM AAaaS
How is an E2E MCN Service Instance Created?

Deployment phase
Each required service provider’s service manager creates a service orchestrator
How is an E2E MCN Service Instance Created?

Deployment phase
Service orchestrators requiring services from CloudController request them
Key Enabling Framework Technologies

Service Manager
- Python, Pyssf, OCCI

Service Orchestrator
- Python, Pyssf, OCCI

Cloud Controller
- OpenShift, OpenStack, Pyssf, OCCI
For instance, requesting SO submits a request for a service instance (direct, user interface or CLI)
Service Manager

contains a list of the available services offered by the provider
Service Manager

deploys the SO bundle to the CC
provisioning of the service instance incl. all SICs
Service Manager

- Tracks all provisioned SOs (service instance)
- Also contains info on all mgmt interfaces
Service Manager

Deletes the complete service instance
Service Orchestrator

All requests by SM to SO goes through here
Service Orchestrator

Takes decisions on the run-time management of the SICs (e.g. based on monitoring data)
Service Orchestrator

Responsible for enforcing the decisions towards the CC
Service Orchestrator

- Which services are required to support SO implementation
- How they are configured
- Model defined by CC
Which services are required to support SO implementation. How they are configured

*Diff - live information from CC*
CloudController

Providers a Frontend and exposes an API which can be used to interface with CC
CloudController

Allows the listing of capabilities which CC offers
CloudController

Will enable the deployment of the SO and its individual SIC
CloudController

Will enable the configuration of the SIC
CloudController

- Takes care of runtime operations such as scaling requests

Diagram:
- Cloud Controller
  - Northbound Frontend
  - Southbound Backend
  - Design
  - Deployment
  - Provisioning
  - Runtime
  - Disposal

Services of Category
  - Atomic
  - Support
CloudController

Interface with other services, requested by higher layers

Cloud Controller

Northbound Frontend

Design
Deployment
Provisioning
Runtime
Disposal

Southbound Backend

Services of Category Atomic

Services of Category Support
To conclude:
Open Research Directions (1)

- **Fog-enabled federated management** - efficiently deploying and federatedly managing densely inter-connected and decentralized cloud infrastructures, by dynamically moving (partial) MCN functions to the edge of the network to take local decisions and optimizations.

- **Edge computing for extremely high availability** - How to exploit mobile edge computing towards disaster resilient and emergency robust MCN solutions? How should it be efficiently combined with DC networking virtualization?

- **Scalability and quality for data-intensive applications** - Effective and efficient solutions for scale, quality, and privacy/security, in particular in data-intensive applications deployed over federated environments, such as in the case of MCN for smart cities or wide-scale IoT with dominant M2M communications.
To conclude:
Open Research Directions (2)

- **Locality-based resource efficiency and decentralized orchestration** - Novel algorithms and techniques for resource efficiency and composition, e.g., taking into consideration dynamically changing patterns for service demand and mobility, application confidentiality levels.

- **State, state, state**... - efficient state migration, replication, eventual consistency, proactive state management, etc.
To conclude: Open Innovation Challenges for Industrial Exploitation

About immediate industrial applicability of solutions in the field, in several sub-areas with specific performance/functional constraints we are far from ready-to-deploy frameworks:

- **high-availability by design**, in particular in the case of federated infrastructures
- **cost-efficient scalability**
- **QoS differentiation** with reasonable guarantees under dynamically changing (in both time and space) load profiles
- Prototyping and demonstrating **wide-scale pilots** that show the advantages of edge computing techniques in “hard” application scenarios, such as **federated mobile public safety networks**, with specific challenges in terms of reliability and privacy
Conclusions?

Still a lot of **research & innovation work to complete** to make edge computing solutions applicable in **different application domains** (e.g., data intensive apps) and **economically sustainable to leverage new business models** (e.g., need for portable orchestration solutions for federated environments, especially container-based)

**Opportunities** for both academia & industries
Some Related Recent and Ongoing Research Projects

**EU projects**

**National projects**
- Regional projects funded by POR-FESR, 2015-2018
- Industry 4.0 and national competence center in Bologna, 2017-2020, many collaborations within companies of the “packaging” valley
Related Ongoing International Collaborations and Opportunities

- IBM Dublin, IBM Haifa, IBM T.J. Watson
- Engineering
- Eurotech, Siemens
- Italian automation industries for Industry 4.0
- Fraunhofer FOKUS, TU Berlin, UPMC, Missouri UST, Technion Israel, UCLA, U.Ottawa, Concordia U.
Some Primary References (1)

- ETSI’s Mobile Edge Computing initiative


- Open Edge Computing (OEC) open source project,
  http://openedgecomputing.org/

- Cisco IOx, https://developer.cisco.com/site/iox/

Some Primary References (2)


Some Secondary 😊 References (3)

Some Secondary 😊 References (4)

- … and several 😊 others under review…
Questions?

Thanks again and keep in touch!

Contact info and related publications:
Paolo Bellavista, Ph.D., Associate Professor
EiC of MDPI Computers
EB Member of IEEE TNSM, IEEE TSC, Elsevier PMC, Elsevier JNCA, Springer JNSM,
ACM/Springer WINET
DISI - University of Bologna
Viale del Risorgimento, 2 - 40136 Bologna - ITALY
Email: paolo.bellavista@unibo.it
Web: http://lia.disi.unibo.it/Staff/PaoloBellavista/