IoT Architecture

Fundamentals of Internet of Things

Andrea Omicini
andrea.omicini@unibo.it

Bologna Business School, Università di Bologna

Master in Digital Technology Management 2020/2021

Disclaimer

- most of the ideas here are extracted from [Tanenbaum and van Steen, 2007], [Jabraeil Jamali et al., 2020], and [Serpanos and Wolf, 2018]
- however, any mistake should be attributed to the author of these slides

Next in Line...

Focus on...

Software Architectures to Handle Complexity I

Distributed systems are complex

- in order to manage their intrinsic complexity, distributed systems should be properly *organised*
- organisation of a distributed system is mostly expressed in terms of its *software components*

Representing complex distributed systems

- representing distributed systems is then difficult
- their organisation should be represented at the *right level of abstraction*
- which means, with all what is needed to understand, and nothing more

Software Architectures to Handle Complexity II

Software architectures expresses component organisation

- there are many ways to organise components of a distributed system, classified and represented as software architectures
- there are many possible *instantiations* of a software architecture, where components have their actual *place* in the distributed system—often called system architectures

What is a Software Architecture?

Software architecture

A software architecture is an abstraction of the run-time elements of a software system during some phase of its operation. A system may be composed of many levels of abstraction and many phases of operation, each with its own software architecture. [Fielding, 2000]

Architectural elements

A software architecture is defined by a configuration of architectural elements – components, connectors, and data – constrained in their relationships in order to achieve a desired set of architectural properties. [Fielding, 2000]

Architectural Elements

Components

A component is an abstract unit of software instructions and internal state that provides a transformation of data via its interface

Connectors

A connector is an abstract mechanism that mediates communication, coordination, or cooperation among components

Data

A datum is an element of information that is transferred from a component, or received by a component, via a connector

Focus on...

Architectural Styles to Classify Systems

An *architectural style* is formulated in terms of...

- *components*
- the way in which components are *connected* to each other
- the *data* flowing through the components
- the way in which all the above things are *configured* altogether to build the system

The notion of architectural style...

- encompasses a way to cluster and classify groups of similar systems—that is, having the *same* sort of *organisation*
- allow distributed systems to be *compared*
- but also provide general *patterns* for their overall *design*

Architectural Properties & Constraints

Architectural properties

The set of architectural properties of a software architecture is derived from the *selection* and *arrangement* of components, connectors, and data within a system

- *functional properties*
- *quality attributes* such as ease of evolution, reusability of components, efficiency, and dynamic extensibility

Properties are induced by the *set of constraints* within an architecture

Architectural constraints

Architectural constraints are often motivated by the application of a software engineering principle to an aspect of the architectural elements

What is an Architectural Style?

Architectural Style

An architectural style is a coordinated set of architectural constraints that restricts the roles/features of architectural elements and the allowed relationships among those elements within any architecture that conforms to that style [Fielding, 2000]

Architectural styles

- are a mechanism for categorising architectures and defining their common characteristics
- provide an abstraction for the interactions of components, capturing the essence of a pattern of interaction by ignoring the accidental details of the rest of the architecture

Main Examples of Architectural Styles

Identification of architectural styles

- architectural styles – like patterns in software engineering – are to be *devised out* rather than invented
- today, four different architectural styles have been identified as the main ones for distributed systems

Main architectural styles for distributed systems

- layered architectures
- object-based architectures
- data-centered architectures
- event-based architectures
- \+ shared data-space architectures

Layered Architectures I

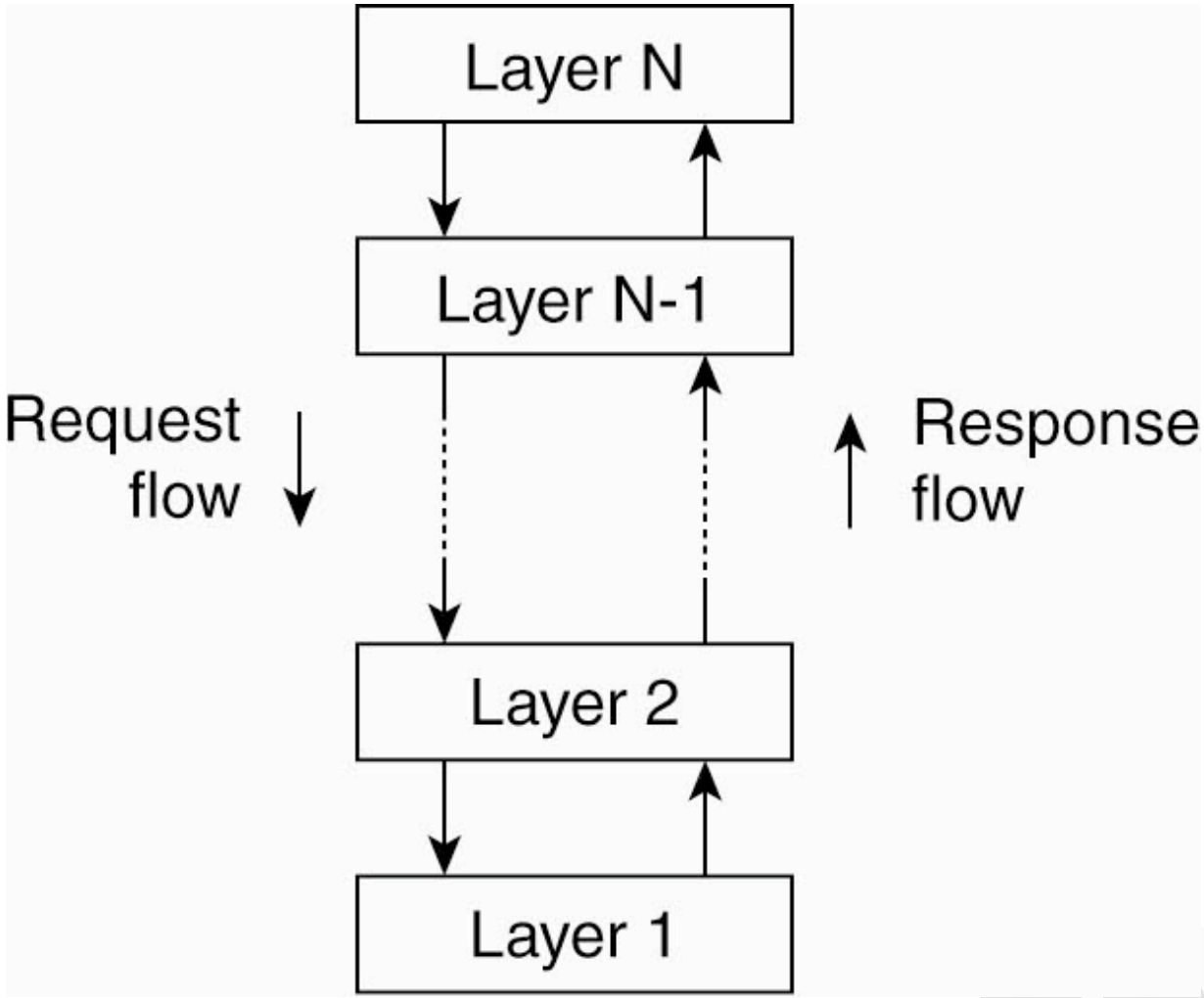
Basic idea

- components are organised in a *layered fashion*
- where components of a layer *only* call components of the layer below, and are *only* called by the components of the layer above

Data flow

- the request-response flow is always top-down / bottom-up
- control flow follows the same pattern along with data

Layered Architectures II



[Tanenbaum and van Steen, 2007]

Object-based Architectures I

Basic idea

- components are *objects*
- components are connected through a *RPC mechanism*

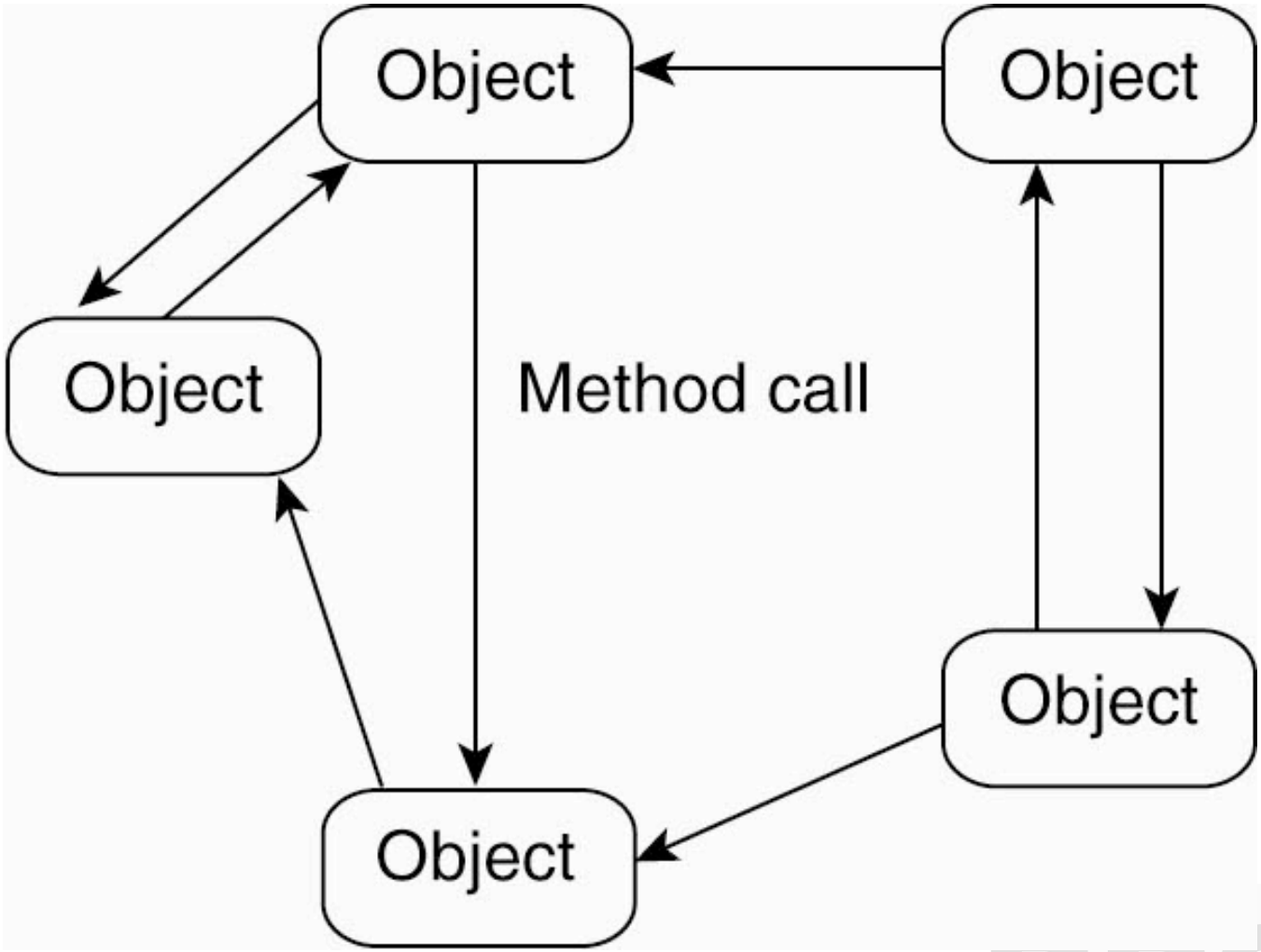
Client-server architectures

- . . . are built out of this style

Layered and object-based architectures

- . . . are the most important styles for distributed systems today
- however, a lot of things are going to happen in the future, which may change such an overall picture

Object-based Architectures II



[Tanenbaum and van Steen, 2007]

Data-centred Architectures I

Basic idea

- communication among processes occurs through a shared repository
- the repository might be either passive (reactive) or (pro)active

Main features

- ... depends on the choice made for the shared repository
  - how *information* is represented
  - how *events* are handled
  - how the *shared repository* behave in response to interaction
  - how *processes* interact with / through the shared repository

Data-centred Architectures II

Examples are everywhere

- web-based systems, for instance, are largely data-centric
- also, many distributed applications still work by sharing files around the network

Event-based Architectures I

Basic idea

- processes communicate through an event bus
- through which *events* are propagated
- possibly carrying *data* along

Main example: Publish/subscribe systems

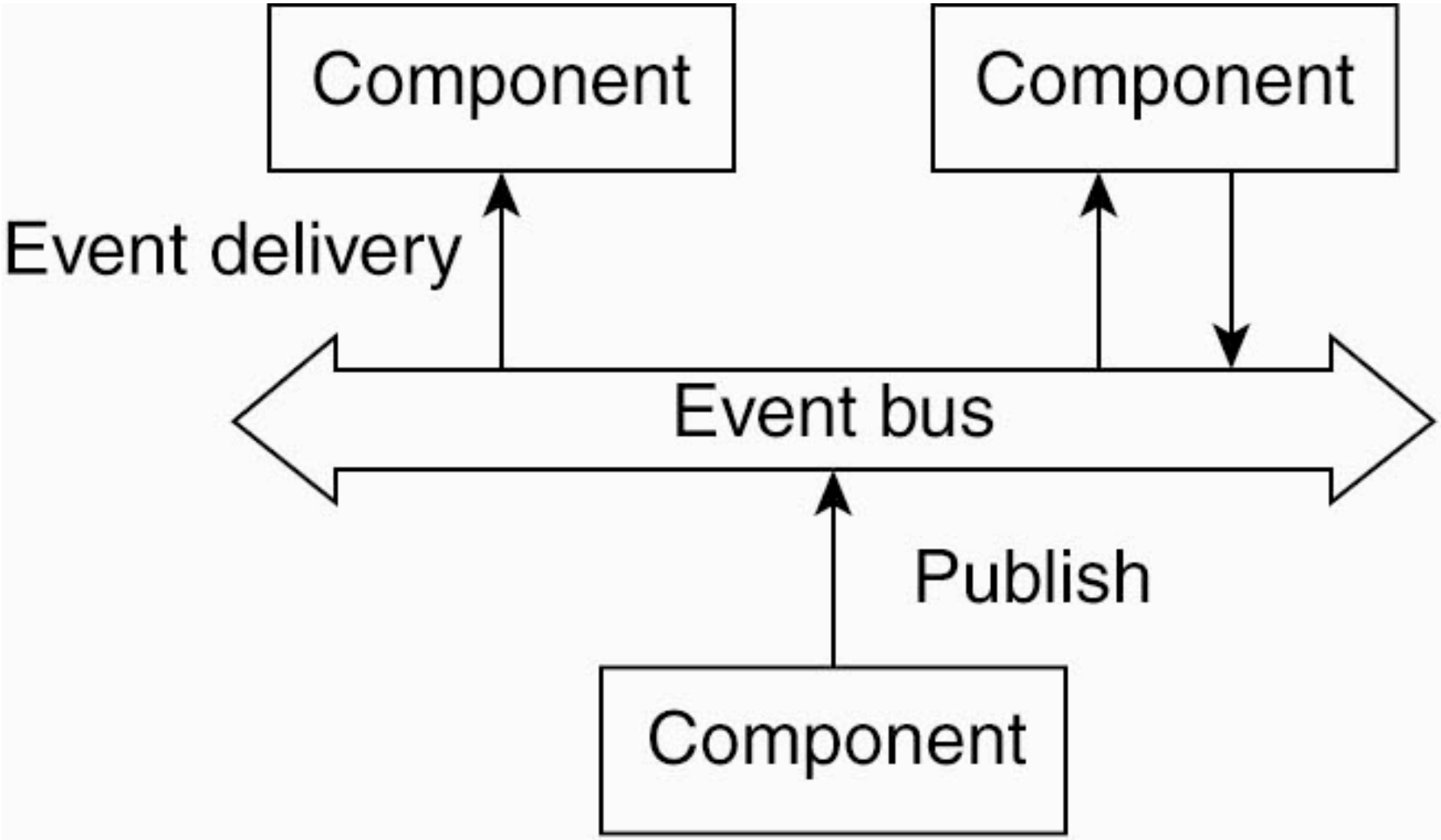
- *publishers* publish events through the middleware
- *subscribers* receive events to which they have subscribed

Event-based Architectures II

Main feature

- processes can communicate with no need to *reference* each other / to know each other—they are *referentially uncoupled*
- processes can communicate with no need to *share* the same space—they are *uncoupled in space*

Event-based Architectures III



[Tanenbaum and van Steen, 2007]

Shared Data-space Architectures I

Basic idea

- putting together Data-centric and Event-based architectures
- the shared repository is a shared persistent data-space, and also an event bus
- where data is stored and accessed
- along with related events

Main example: Blackboard systems

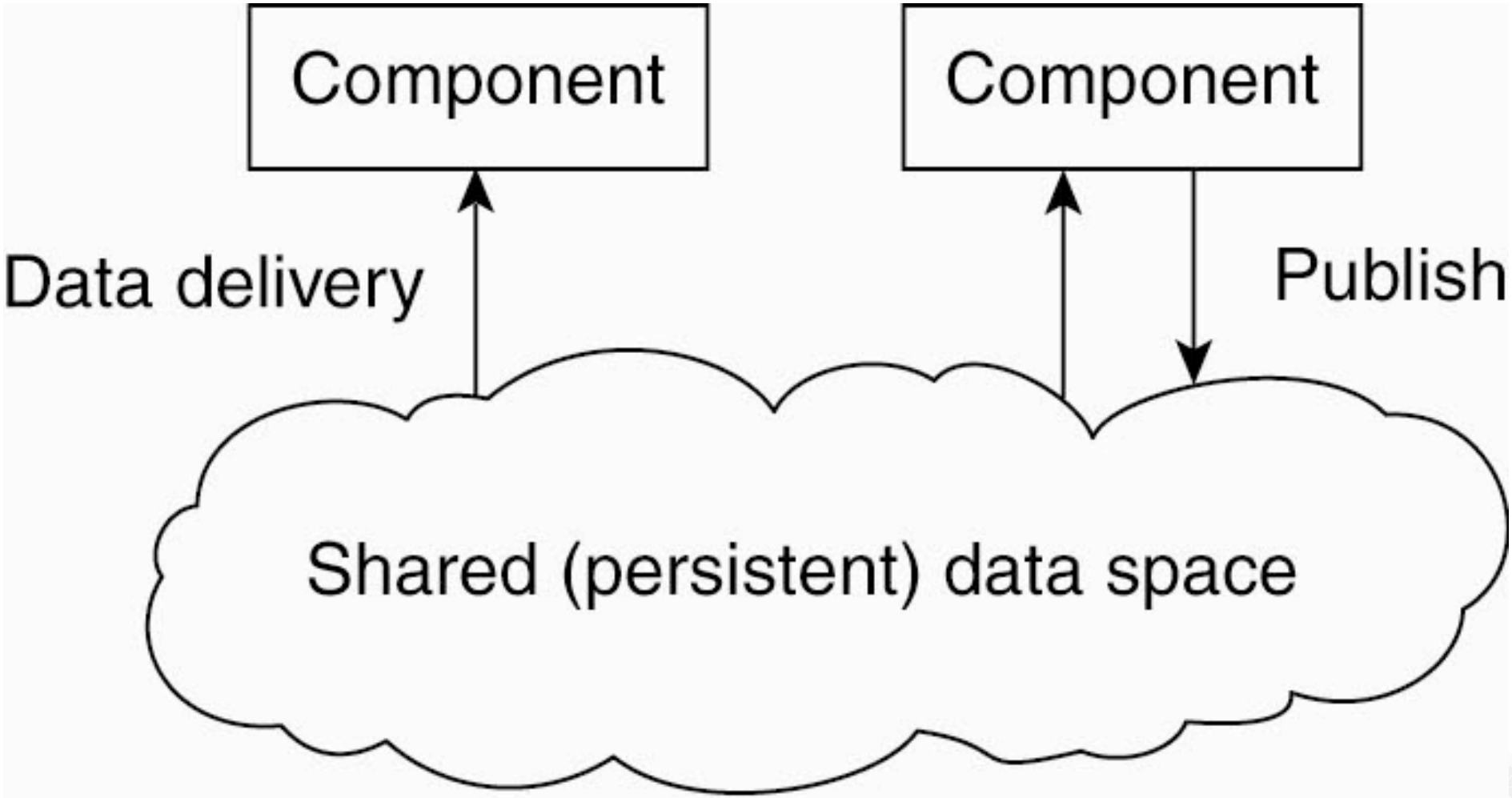
- processes put data in the blackboard
- the blackboard aggregates *knowledge*, implements *policies*, and drive the *coordination* of processes

Shared Data-space Architectures II

Main feature

- processes can communicate with no need of compresence
- processes are also *uncoupled in time*

Shared Data-space Architectures III



[Tanenbaum and van Steen, 2007]

Next in Line...

Are Software Architecture Suitable for IoT Systems?

- technically, no—at least not straightforwardly
  - the level of abstraction is too low, the level of detail too fine-grained, ...
- yet, they point out some of the the main issues here
  - *abstraction* is the only way out complexity
  - the main *components* and their mutual *interaction* have to be pointed out in order to suitably *represent* IoT systems
- so, they show us how to proceed in our attempt to describe the articulation of IoT systems beyond their intrinsic *complexity*

Are Architectural Styles Suitable for IoT Systems?

- yes, in principle—not the simple styles shown before, yet
  - maybe they are applicable, but just too simple to represent something meaningful about IoT systems in general
  - in order to use them we should *abstract too much*
- yet, they point out another main issue
  - there are common ways of *representing* diverse IoT systems despite their technical differences
- and, those ways are useful to understand them, and manage their huge *heterogeneity*

Architectural Views

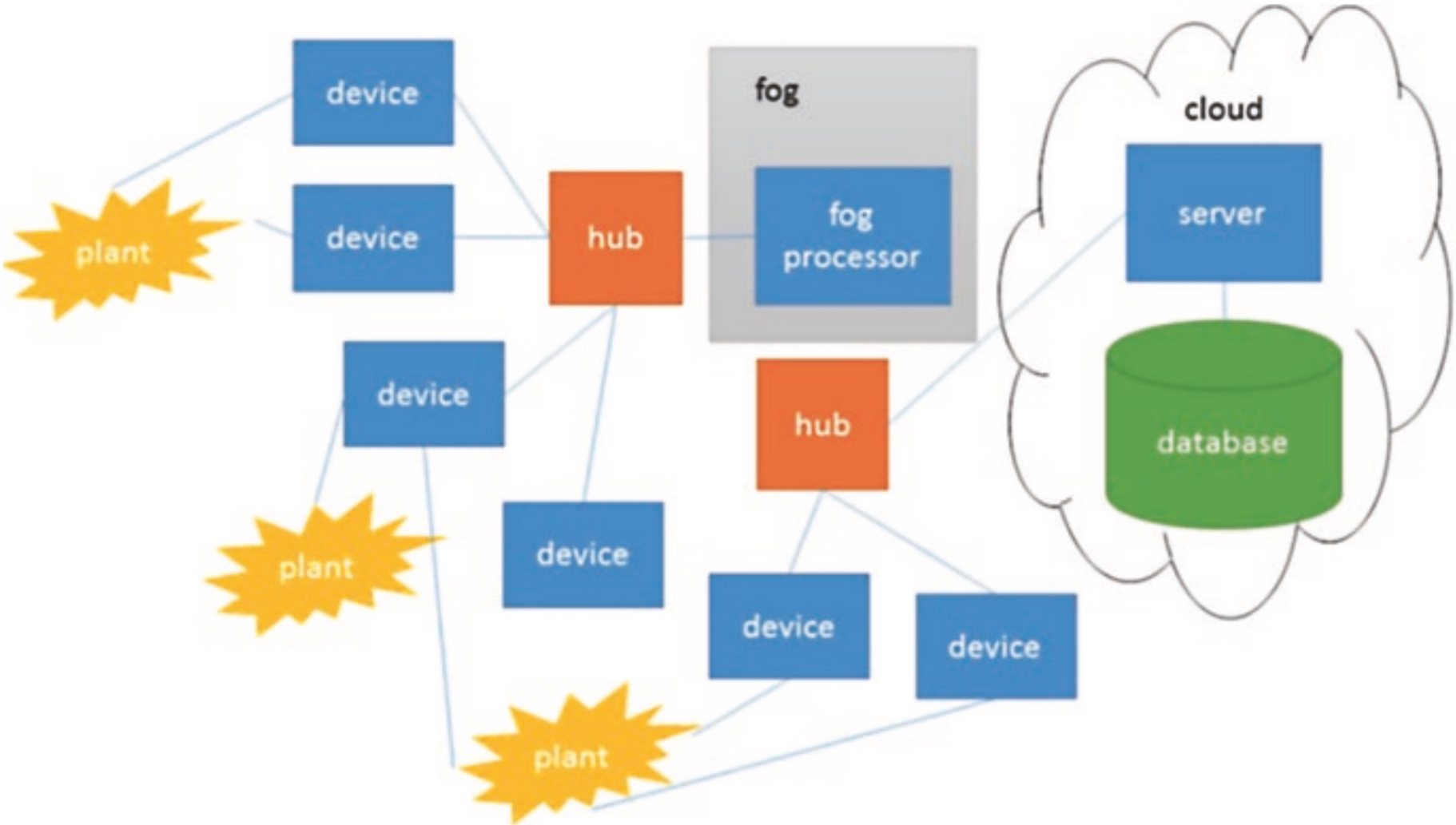
- so, in the next slides we will discuss some *architectural views* of the IoT
- which differs one from another quite apparently
  - yet they are mostly compatible with each other
- overall, serving the purpose to point out different essential properties that all IoT systems share

Focus on...

Event-based Architectural View [Serpanos and Wolf, 2018] I

- a key aspect of IoT is *event-driven* or *aperiodic sampling*
  - traditional digital signal processing and control assume periodic samples resulting in *time-series data*
  - however, time series consume too much power at the nodes and too much bandwidth on the network
- constraints on power and bandwidth also encourage *distributed computing over sensor events*
- relatively small processors can process many data streams
- recognising interesting events with *edge processing* reduces both network bandwidth and power consumption
  - *cloud* or *fog computing* for processing interesting events

Event-based Architectural View [Serpanos and Wolf, 2018] II

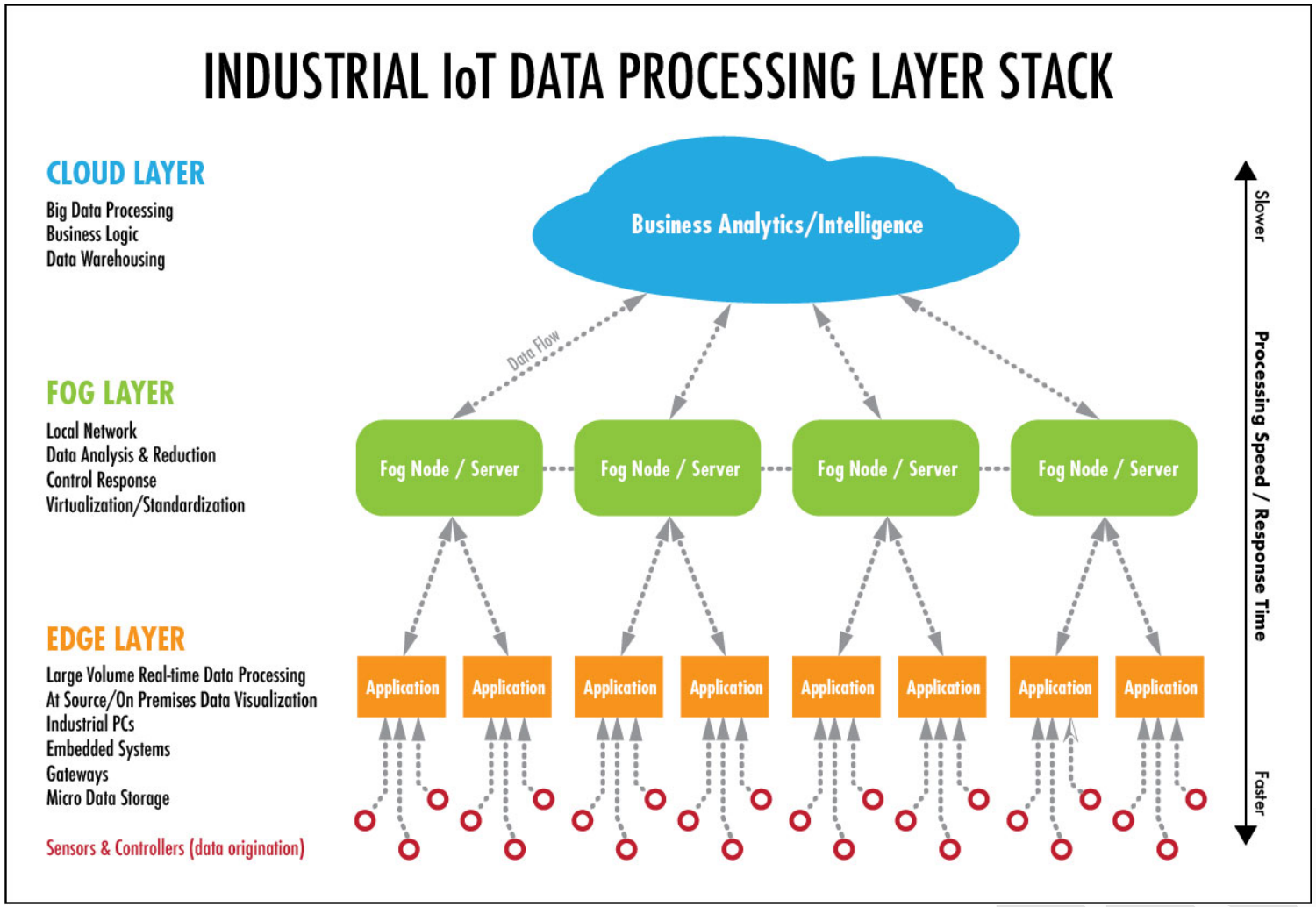


A view over the organisation of an IoT system [Serpanos and Wolf, 2018]

Event-based Architectural View [Serpanos and Wolf, 2018] III

- the *plant* or *environment* is the physical system interacting with the IoT system
- a set of *devices* form the leaves of the network
  - typically including sensors and/or actuators, processors, and memory
  - with a network interface
  - possibly running the Internet Protocol (IP)
- *hubs* for first-level connectivity between nodes and (rest of) network
  - typically running the IP
- *fog processors* perform operations on local sets of nodes and hubs
  - servers near nodes reduces latency
  - yet, fog devices have not the same computational power as cloud servers
  - fog devices also introduce system management issues
- *cloud servers* provide computational services for the IoT system
  - databases store data and computational results
  - the cloud may provide services mediating between nodes and users

Cloud, Fog, Edge: A Layered View



https://www.winsystems.com/cloud-fog-and-edge-computing-whats-the-difference/

What About Classical Software Architecture?

- the first and most classical distributed system architecture is a *centralised* one
  - with one central server and many distributed clients—initially, just dumb *terminals*

? how does that compare with the cloud/fog/edge layered view just discussed?

Focus on...

Different Views

- different researchers propose different architectures
- typically depending on their main focus of attention
  - their specific competence, their interest
  - the different application scenarios taken into account
- yet, some more or less shared views over IoT architecture can be devised out

What is an architecture here?

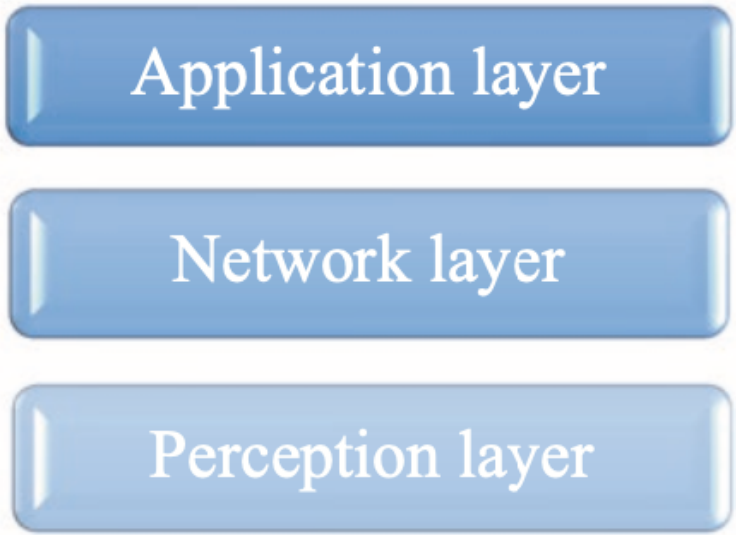
In this context, an architecture of an IoT system is basically a *framework* to define

- the *physical components* of a network
- its functional *organisation* and *configuration*
- its *operating principles* and *procedures*
- the *data formats* used in its operation

Three-Layer Conventional IoT Architecture I

Three layers

- perception layer
- network layer
- application layer



[Jabraeil Jamali et al., 2020]

Three-Layer Conventional IoT Architecture II

Perception layer

- . . . also known as *recognition layer*
- the lowest layer of conventional IoT architecture
- primarily responsible for *collecting* and *transforming* useful information/data from things or environment
  - e.g., wireless sensor networks, heterogeneous devices, real-world "enhanced" objects, environmental sensors
- short-range technologies are the most relevant here
  - e.g., RFID, Bluetooth, Near-Field Communication (NFC), 6LoWPAN (Low Power Personal Area Network)

Three-Layer Conventional IoT Architecture III

Network layer

- works as the brain of the architecture
- the core layer of IoT *systems*
- primarily responsible to help secure the transmission of data between the application and the perception layer
- it ensures unique addressing and routing capabilities for the integration of countless devices into a single cooperative network
- it involves various sorts of networking technologies
  - such as wired, wireless, and satellite networking technologies

Three-Layer Conventional IoT Architecture IV

Application layer

- top layer of the conventional IoT architecture
- providing specific services based on the user's need
- primary responsible for bridging the gap between users and the IoT technological layers
- high-level intelligent applications
  - such as, e.g., disaster monitoring, health monitoring, intelligent manufacturing

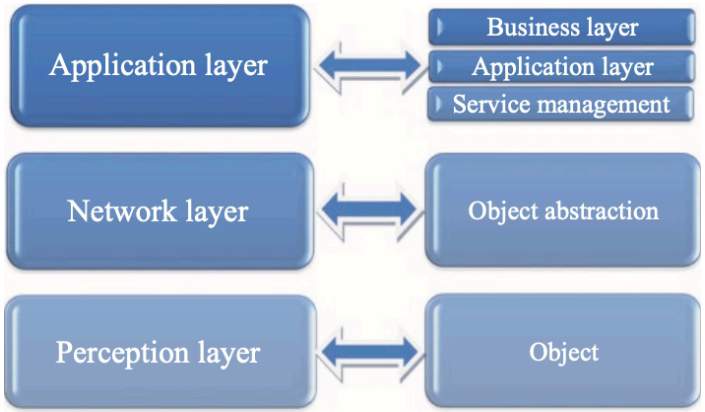
What is the problem here?

- this architecture is too simple
- too few relevant features are highlighted
- more layers?

Five-Layer Conventional IoT Architecture I

Five layers

- object layer
- object abstraction layer
- service management layer
- application layer
- business layer



[Jabraeil Jamali et al., 2020]

Five-Layer Conventional IoT Architecture II

Object layer

- lowest layer here
- primarily responsible of *collecting data* from many heterogeneous devices
  - to be processed and digitised
  - then transferred to the upper layers

Object abstraction layer

- primarily responsible of providing a uniform and overall view over the heterogeneous objects in the system
- basically, the same of the network layer in the three-layer architecture discussed before

Five-Layer Conventional IoT Architecture III

Service management layer

- used to facilitate the processing of information, decision-making, and controlling the processing of pairing requestor information for relevant tasks

Application layer

- provides customers with high-quality, intelligent facilities that are required by the specific application scenarios

Business layer

- defines and implements the specific business model

Five-Layer Conventional IoT Architecture IV

What is the problem here?

- this architecture is still too generic
- application scenarios of IoT are actually too heterogeneous to be meaningfully captured by a mere layered architecture
- so what?
  - how does the industry look at IoT systems?
  - are there any specific architectures for specific scenarios?

Cisco IoT Reference Model I

Levels

7 **Collaboration & Processes**
(Involving People & Business Processes)

6 **Application**
(Reporting, Analytics, Control)

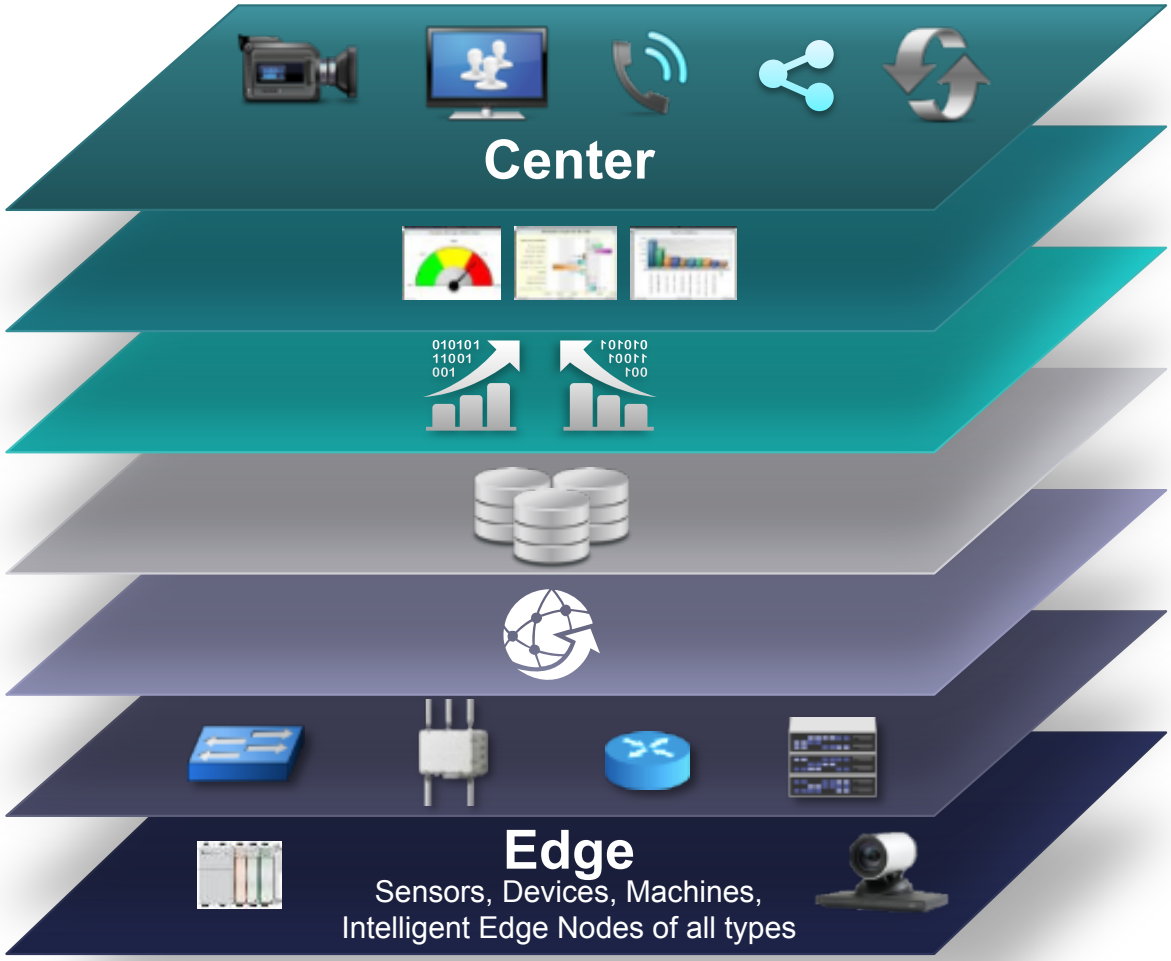
5 **Data Abstraction**
(Aggregation & Access)

4 **Data Accumulation**
(Storage)

3 **Edge Computing**
(Data Element Analysis & Transformation)

2 **Connectivity**
(Communication & Processing Units)

1 **Physical Devices & Controllers**
(The "Things" in IoT)



Cisco Internet of Things reference model [Cisco, 2014]

Cisco IoT Reference Model II

Level 1 [Cisco, 2014]

- Physical devices and controllers that might control multiple device. These are the "things" in the IoT, and they include a wide range of endpoint devices that send and receive information. Today, the list of devices is already extensive. It will become almost unlimited as more equipment is added to the IoT over time.
- Devices are diverse, and there are no rules about size, location, form factor, or origin. Some devices will be the size of a silicon chip. Some will be as large as vehicles. The IoT must support the entire range. Dozens or hundreds of equipment manufacturers will produce IoT devices. To simplify compatibility and support manufacturability, the IoT Reference Model generally describes the level of processing needed from Level 1 devices.

Cisco IoT Reference Model III

1 **Physical Devices & Device Controllers**
(The "Things" in IoT)

IoT "devices" are capable of:

- Analog to digital conversion, as required
- Generating data
- Being queried / controlled over-the-net

Level 1: Physical devices and controllers [Cisco, 2014]

Cisco IoT Reference Model IV

Level 2 [Cisco, 2014]

- Communications and connectivity are concentrated in one level—Level 2. The most important function of Level 2 is reliable, timely information transmission. This includes transmissions:
  - Between devices (Level 1) and the network
  - Across networks (east-west)
  - Between the network (Level 2) and low-level information processing occurring at Level 3
- Traditional data communication networks have multiple functions, as evidenced by the International Organization for Standardization (ISO) 7-layer reference model. However, a complete IoT system contains many levels in addition to the communications network.

Cisco IoT Reference Model V

2 **Connectivity**
(Communication & Processing Units)

Level 2 functionality focuses on East-West communications

Connectivity includes:

- Communicating with and between the Level 1 devices
- Reliable delivery across the network(s)
- Implementation of various protocols
- Switching and routing
- Translation between protocols
- Security at the network level
- (Self Learning) Networking Analytics

Level 2: Connectivity [Cisco, 2014]

Cisco IoT Reference Model VI

Level 3 [Cisco, 2014]

- The functions of Level 3 are driven by the need to convert network data flows into information that is suitable for storage and higher level processing at Level 4 (data accumulation).
- This means that Level 3 activities focus on high-volume data analysis and transformation.
- For example, a Level 1 sensor device might generate data samples multiple times per second, 24 hours a day, 365 days a year.
- A basic tenet of the IoT Reference Model is that the most intelligent system initiates information processing as early and as close to the edge of the network as possible.
- This is sometimes referred to as fog computing. Level 3 is where this occurs.

Cisco IoT Reference Model VII

Level 3 (contd.) [Cisco, 2014]

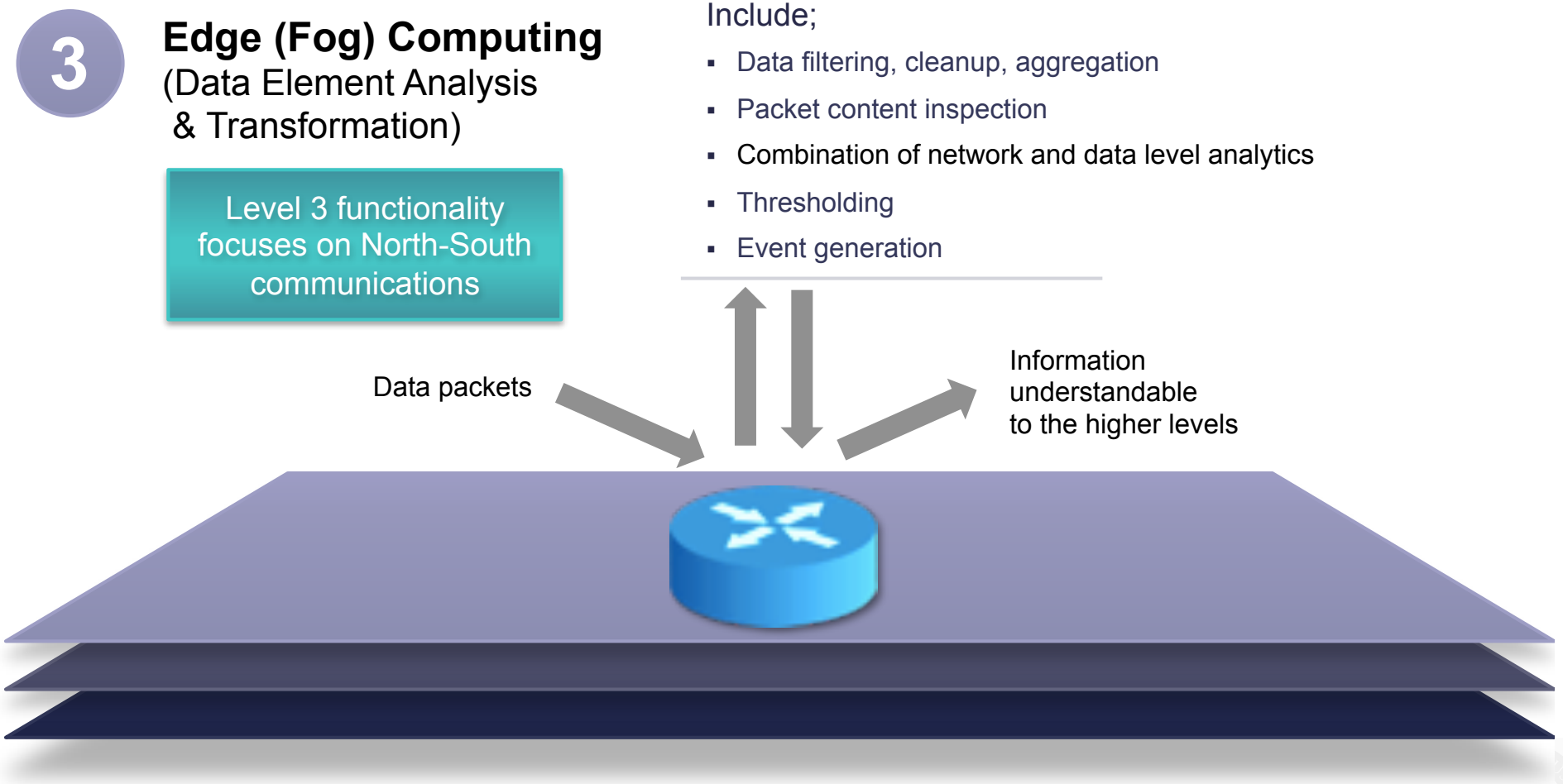
- Given that data is usually submitted to the connectivity level (Level 2) networking equipment by devices in small units, Level 3 processing is performed on a packet-by-packet basis.
- This processing is limited, because there is only awareness of data units—not "sessions" or "transactions".

Cisco IoT Reference Model VIII

Level 3 (contd.) [Cisco, 2014]

- Level 3 processing can encompass many examples, such as:
  - Evaluation: Evaluating data for criteria as to whether it should be processed at a higher level
  - Formatting: Reformatting data for consistent higher-level processing
  - Expanding/decoding: Handling cryptic data with additional context (such as the origin)
  - Distillation/reduction: Reducing and/or summarizing data to minimize the impact of data and traffic on the network and higher-level processing systems
  - Assessment: Determining whether data represents a threshold or alert; this could include redirecting data to additional destinations

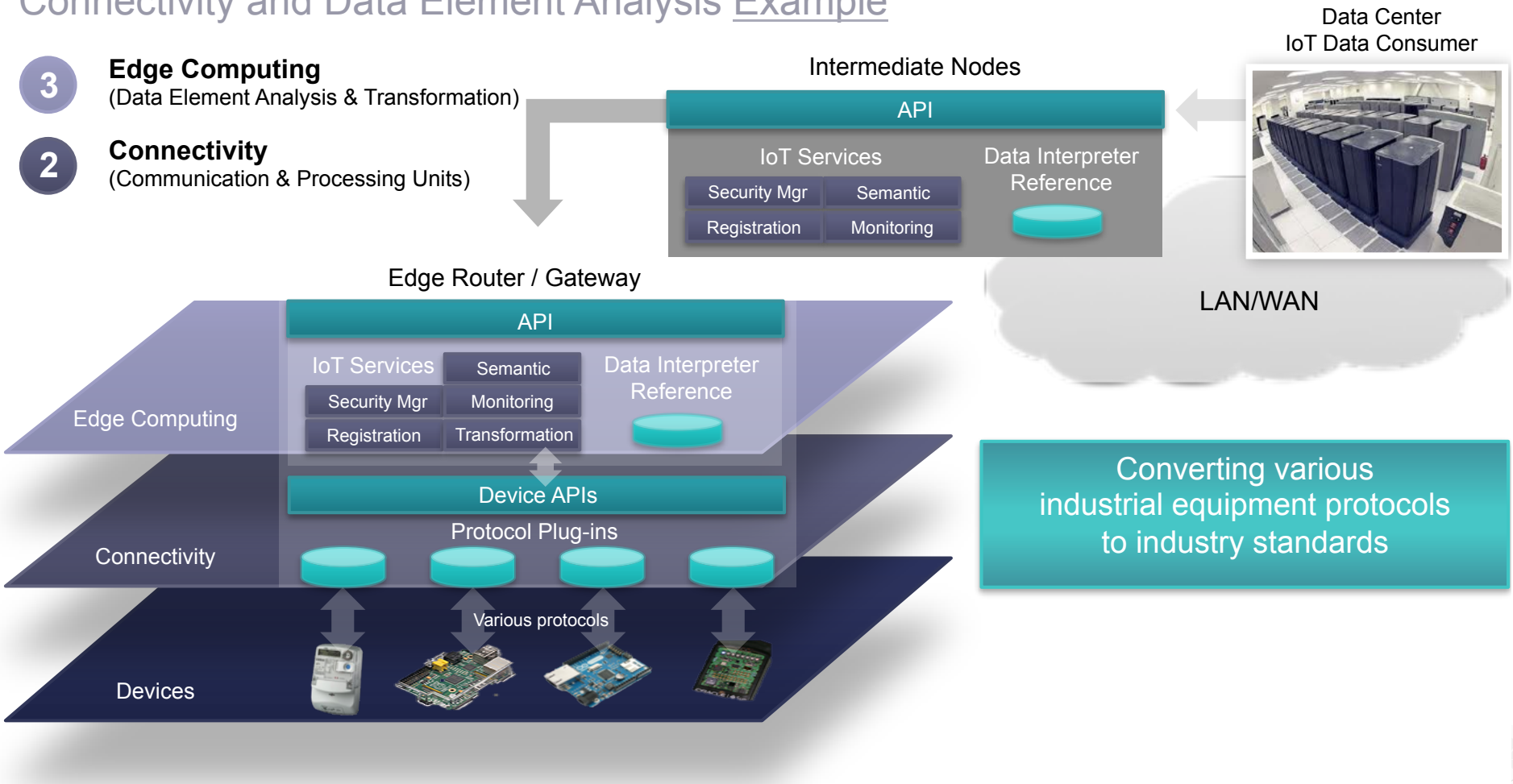
Cisco IoT Reference Model IX



Level 3: Edge (fog) computing [Cisco, 2014]

Cisco IoT Reference Model X

Connectivity and Data Element Analysis Example



Level 2 & 3: Connectivity and Data Element Analysis Example [Cisco, 2014]

Cisco IoT Reference Model XI

Level 4 [Cisco, 2014]

- Networking systems are built to reliably move data. The data is "in motion". Prior to Level 4, data is moving through the network at the rate and organization determined by the devices generating the data. The model is event driven.
- Most applications cannot, or do not need to, process data at network wire speed. Applications typically assume that data is "at rest"—or unchanging—in memory or on disk.
- At Level 4, Data Accumulation, data in motion is converted to data at rest.

Cisco IoT Reference Model XII

Level 4 (contd.) [Cisco, 2014]

- Level 4 determines:
  - If data is of interest to higher levels: If so, Level 4 processing is the first level that is configured to serve the specific needs of a higher level.
  - If data must be persisted: Should data be kept on disk in a non-volatile state or accumulated in memory for short-term use?
  - The type of storage needed: Does persistency require a file system, big data system, or relational database?
  - If data is organized properly: Is the data appropriately organized for the required storage system?
  - If data must be recombined or recomputed: Data might be combined, recomputed, or aggregated with previously stored information, some of which may have come from non-IoT sources.

Cisco IoT Reference Model XIII

Level 4 (contd.) [Cisco, 2014]

- As Level 4 captures data and puts it at rest, it is now usable by applications on a non-real-time basis. Applications access the data when necessary.
- In short, Level 4 converts event-based data to query-based processing.
- This is a crucial step in bridging the differences between the real-time networking world and the non-real-time application world.

Cisco IoT Reference Model XIV

4 **Data Accumulation** (Storage)

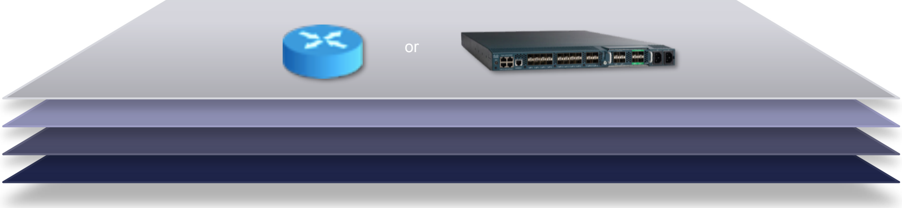
- Event filtering/sampling
- Event comparison
- Event joining for CEP
- Event based rule evaluation
- Event aggregation
- Northbound/southbound alerting
- Event persistence in storage

Query Based Data Consumption

Event Based Data Generation

Making network data usable by applications

1. Converts data-in-motion to data-at-rest
2. Converts format from network packets to database relational tables
3. Achieves transition from 'Event based' to 'Query based' computing
4. Dramatically reduces data through filtering and selective storing



Level 4: Data accumulation [Cisco, 2014]

Cisco IoT Reference Model XV

Level 5 [Cisco, 2014]

- IoT systems will need to scale to a corporate – or even global – level and will require multiple storage systems to accommodate IoT device data and data from traditional enterprise ERP, HRMS, CRM, and other systems.
- The data abstraction functions of Level 5 are focused on rendering data and its storage in ways that enable developing simpler, performance-enhanced applications.

Cisco IoT Reference Model XVI

Level 5 (contd.) [Cisco, 2014]

- With multiple devices generating data, there are many reasons why this data may not land in the same data storage:
  - There might be too much data to put in one place.
  - Moving data into a database might consume too much processing power, so that retrieving it must be separated from the data generation process. This is done today with online transaction processing (OLTP) databases and data warehouses.
  - Devices might be geographically separated, and processing is optimized locally.
  - Levels 3 and 4 might separate "continuous streams of raw data" from "data that represents an event." Data storage for streaming data may be a big data system, such as Hadoop. Storage for event data may be a relational database management system (RDBMS) with faster query times.
  - Different kinds of data processing might be required. For example, in-store processing will focus on different things than across-all-stores summary processing.

Cisco IoT Reference Model XVII

Level 5 (contd.) [Cisco, 2014]

- For these reasons, the data abstraction level must process many different things. These include:
  - Reconciling multiple data formats from different sources
  - Assuring consistent semantics of data across sources
  - Confirming that data is complete to the higher-level application
  - Consolidating data into one place (with ETL, ELT, or data replication) or providing access to multiple data stores through data virtualization
  - Protecting data with appropriate authentication and authorization
  - Normalizing or denormalizing and indexing data to provide fast application access

Cisco IoT Reference Model XVIII

5 **Data Abstraction**
(Aggregation & Access)

Abstracting the data interface for applications

Information Integration

1. Creates schemas and views of data in the manner that applications want
2. Combines data from multiple sources, simplifying the application
3. Filtering, selecting, projecting, and reformatting the data to serve the client applications
4. Reconciles differences in data shape, format, semantics, access protocol, and security

Level 5: Data abstraction [Cisco, 2014]

Cisco IoT Reference Model XIX

Level 6 [Cisco, 2014]

- Level 6 is the application level, where information interpretation occurs. Software at this level interacts with Level 5 and data at rest, so it does not have to operate at network speeds.
- The IoT Reference Model does not strictly define an application. Applications vary based on vertical markets, the nature of device data, and business needs. For example, some applications will focus on monitoring device data. Some will focus on controlling devices. Some will combine device and non-device data. Monitoring and control applications represent many different application models, programming patterns, and software stacks, leading to discussions of operating systems, mobility, application servers, hypervisors, multi-threading, multi-tenancy, etc.

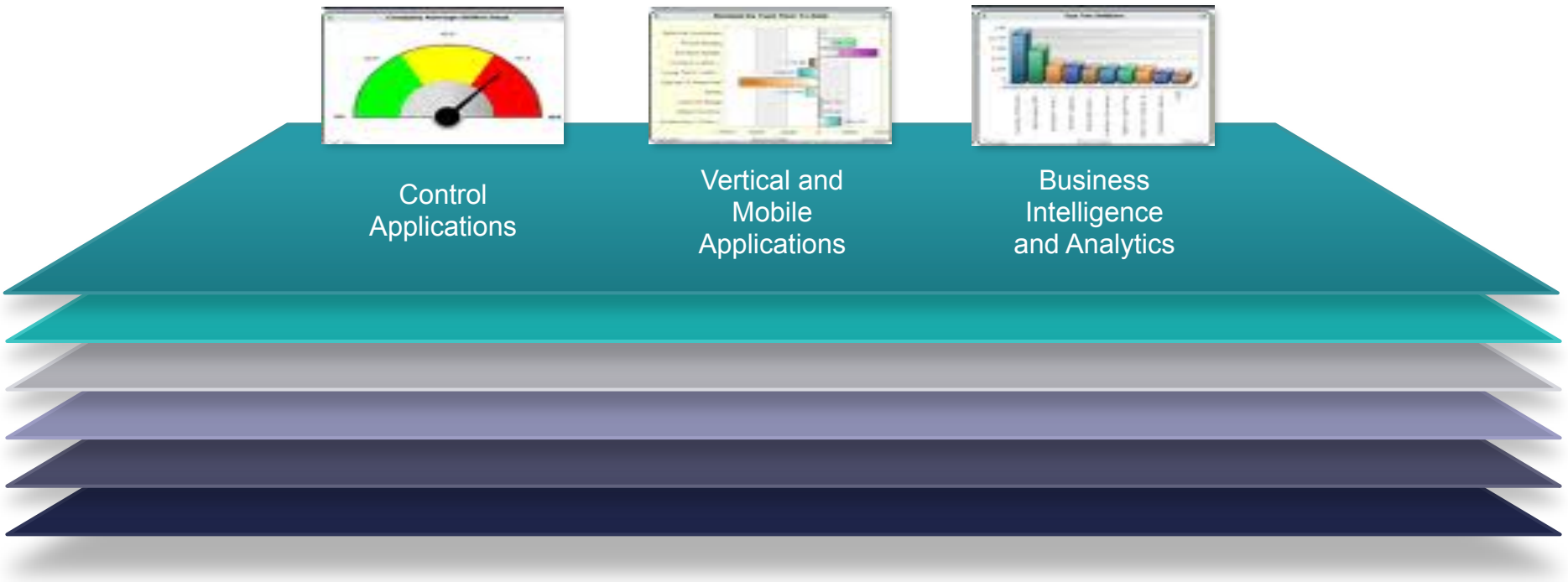
Cisco IoT Reference Model XX

Level 6 (contd.) [Cisco, 2014]

- Examples include:
  - Mission-critical business applications, such as generalized ERP or specialized industry solutions
  - Mobile applications that handle simple interactions
  - Business intelligence reports, where the application is the BI server
  - Analytic applications that interpret data for business decisions
  - System management/control center applications that control the IoT system itself and don't act on the data produced by it
- If Levels 1-5 are architected properly, the amount of work required by Level 6 will be reduced.
- If Level 6 is designed properly, users will be able to do their jobs better

Cisco IoT Reference Model XXI

6 **Application**
(Reporting, Analytics, Control)



Level 6: Application [Cisco, 2014]

Cisco IoT Reference Model XXII

Level 7 [Cisco, 2014]

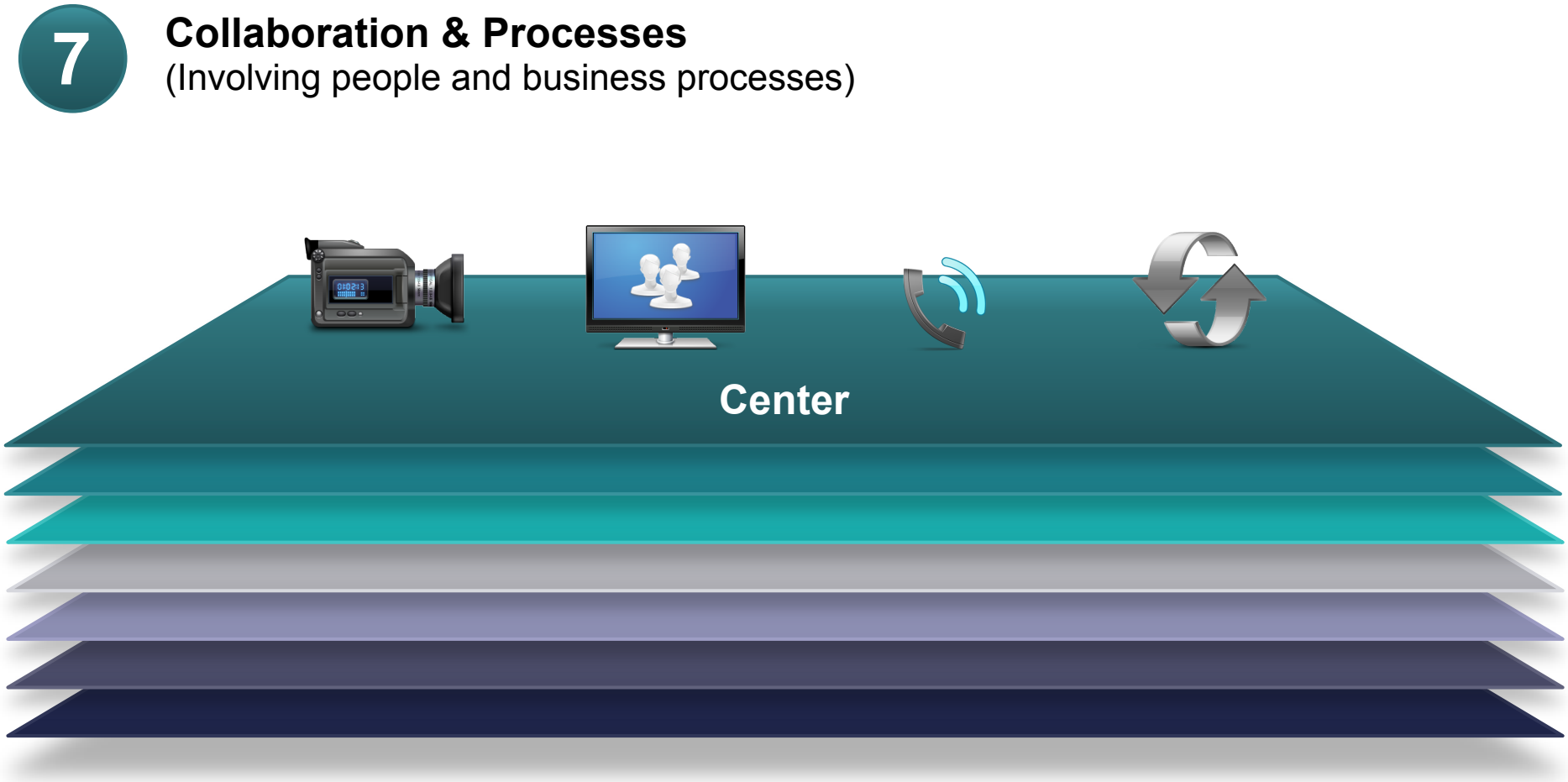
- One of the main distinctions between the Internet of Things (IoT) and the Internet is that IoT includes people and processes.
- This difference becomes particularly clear at Level 7: Collaboration and Processes. The IoT system, and the information it creates, is of little value unless it yields action, which often requires people and processes.
- Applications execute business logic to empower people. People use applications and associated data for their specific needs. Often, multiple people use the same application for a range of different purposes.

Cisco IoT Reference Model XXIII

Level 7 (contd.) [Cisco, 2014]

- So the objective is not the application—it is to empower people to do their work better. Applications (Level 6) give business people the right data, at the right time, so they can do the right thing.
- But frequently, the action needed requires more than one person. People must be able to communicate and collaborate, sometimes using the traditional Internet, to make the IoT useful.
- Communication and collaboration often requires multiple steps. And it usually transcends multiple applications.
- This is why Level 7 represents a higher level than a single application.

Cisco IoT Reference Model XXIV



Level 7: Collaboration and processes [Cisco, 2014]

IoT Architecture Scenario: Agricultural Monitoring I

Wireless sensor network (WSN)

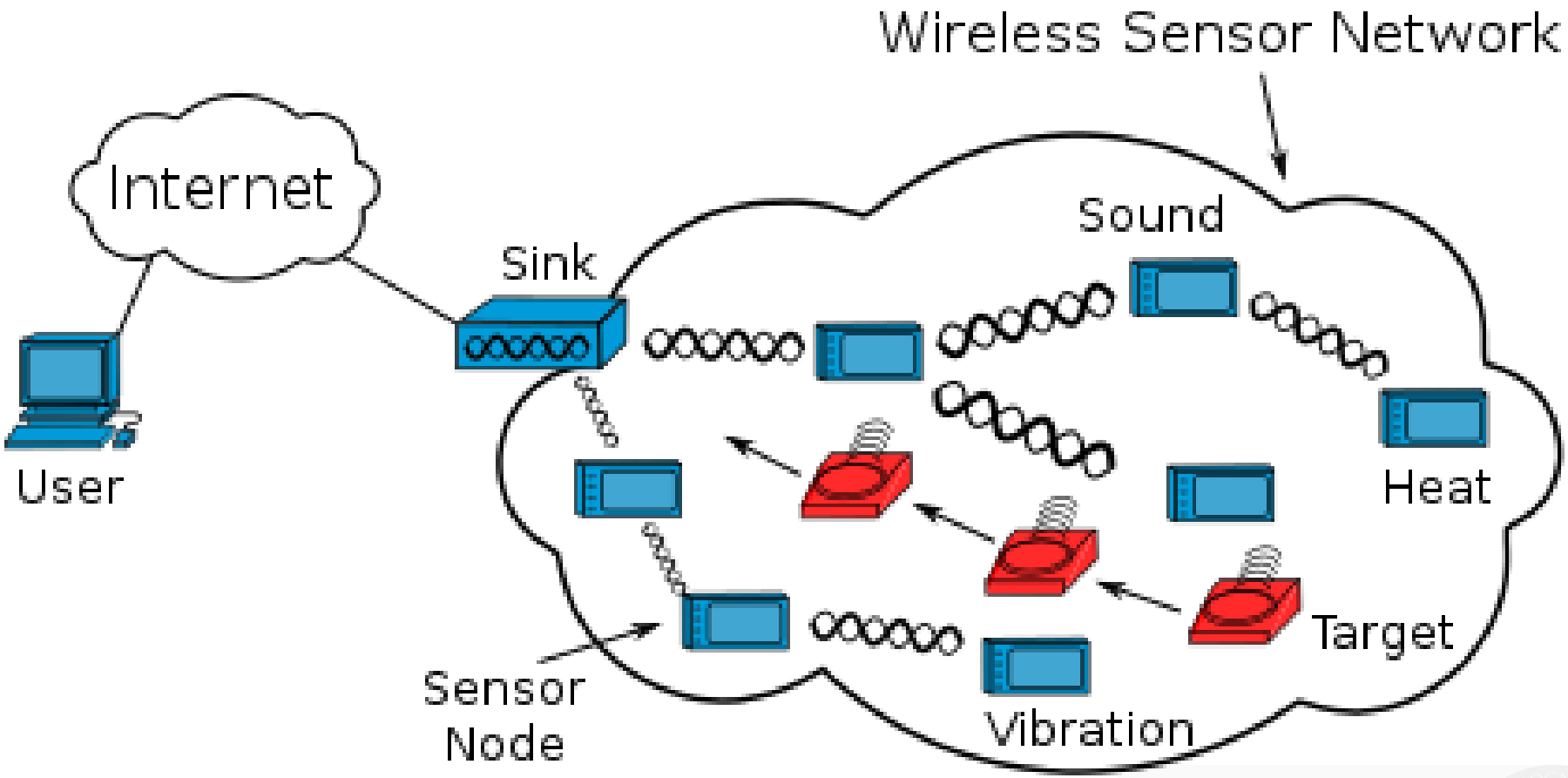
- the main technology here is WSN
  - as it determines most of the new applications, shapes the architecture, and defines the IoT systems in this application field
- a WSN is a network of sensor nodes where each node is equipped with a sensor to detect physical phenomena—e.g., light, heat, pressure
- it consists of a limited number of sensor nodes mastered using a multi-layered protocol organisation using a special purpose node (*sink*)
- w.r.t. wired solutions, it provides for more energy efficiency, scalability, reliability, robustness, and flexibility

IoT Architecture Scenario: Agricultural Monitoring II

Wireless personal area networks (WPAN)

- the most important standard for WSN
  - IEEE 802.15.4 https://www.ieee802.org/15/pub/TG4.html
- it defines the physical and connecting layer for wireless short-range transmission, low power consumption, low complexity, low cost
- it is the basis for the Zigbee, ISA100.11a, WirelessHART, MiWi, 6LoWPAN, Thread (network protocol), and SNAP specifications

IoT Architecture Scenario: Agricultural Monitoring III



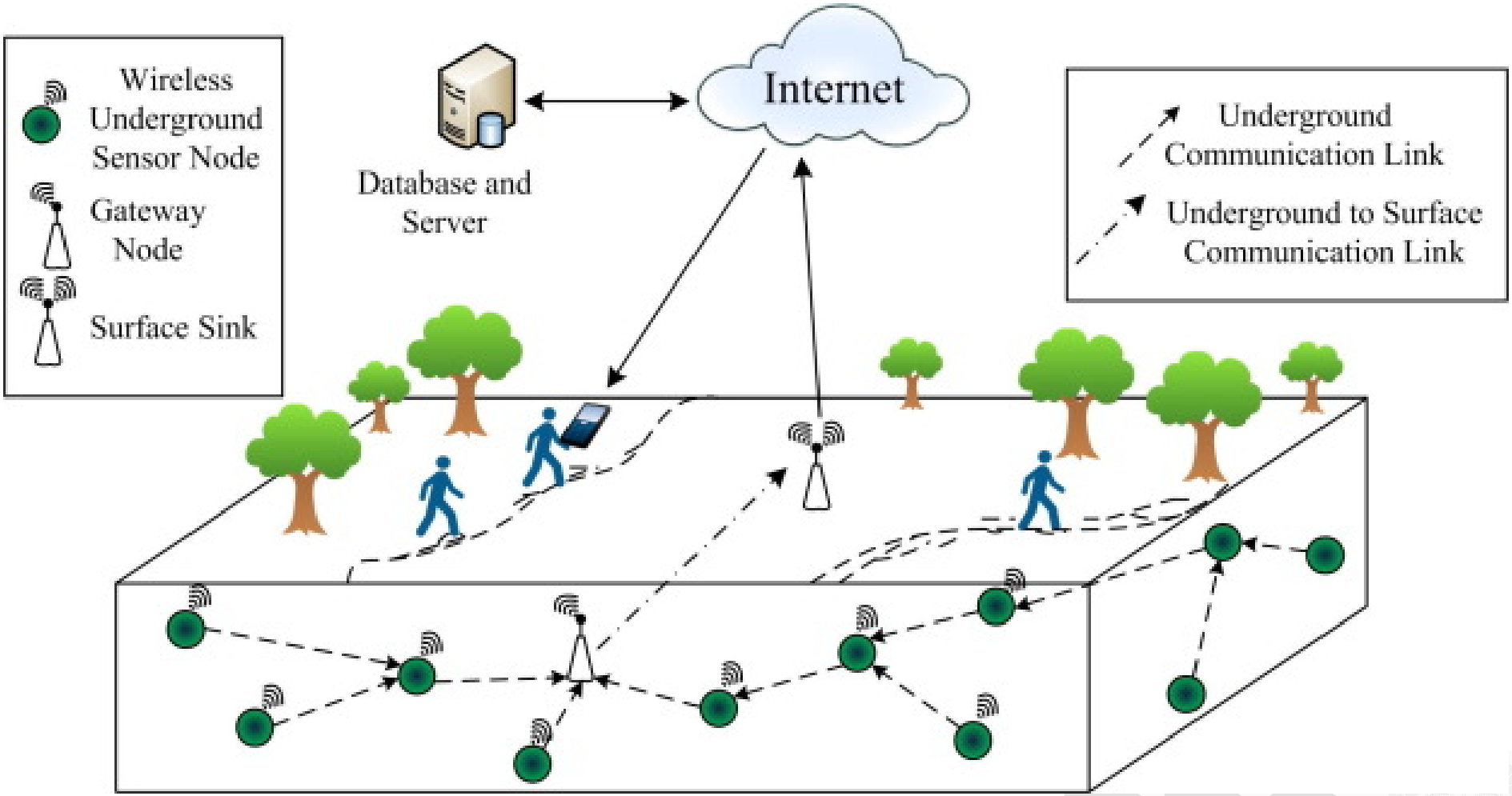
A typical WSN architecture

IoT Architecture Scenario: Agricultural Monitoring IV

Why agriculture?

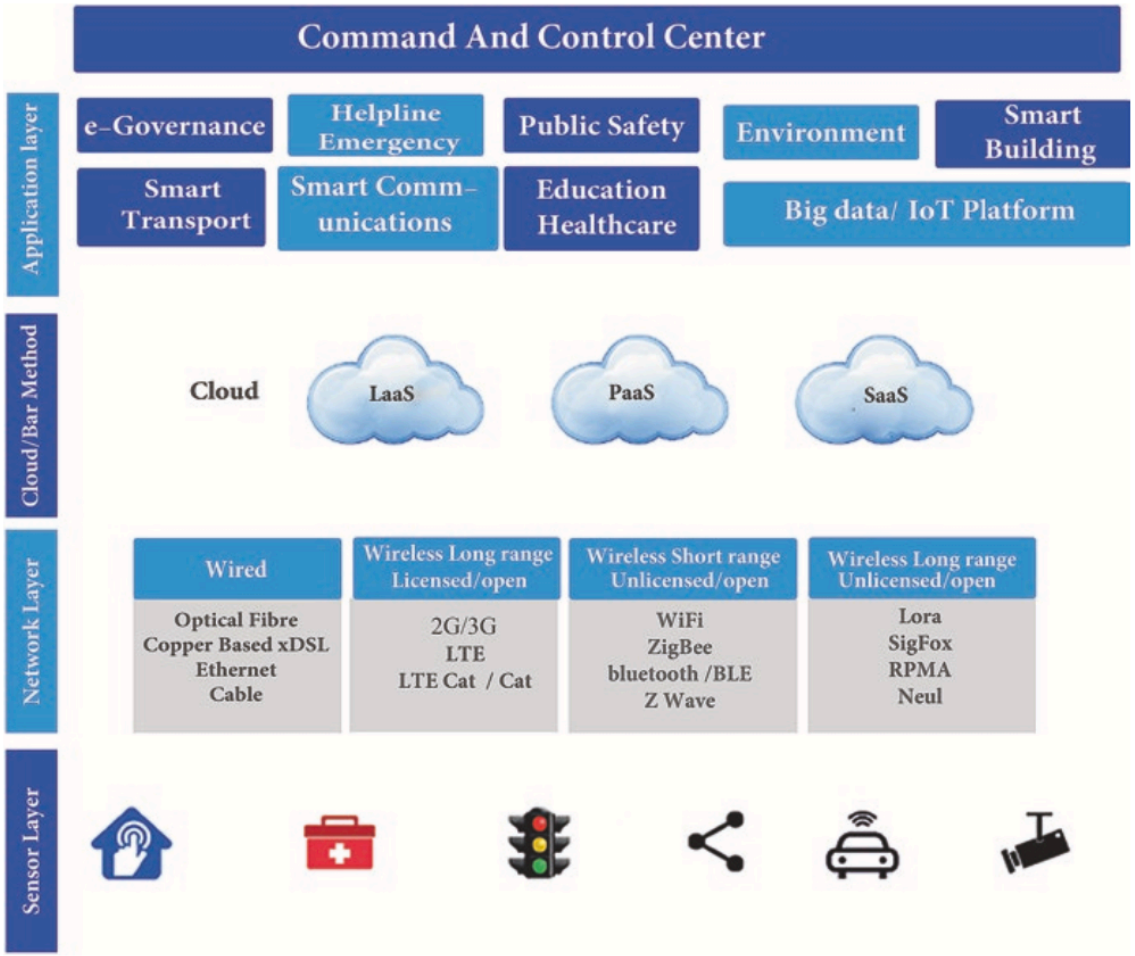
- people abandoning rural areas—still, we need products from there
- smart agriculture is required, IoT a huge opportunity
- specific needs: remote controlled monitoring, moisture and temperature sensing, intruders scaring, security, leaf wetness, proper irrigation, ...
- sensors are deployed to get data from the environment
- WSN allows for collecting data from sensors, upon which specific agricultural applications can be built

IoT Architecture Scenario: Agricultural Monitoring V



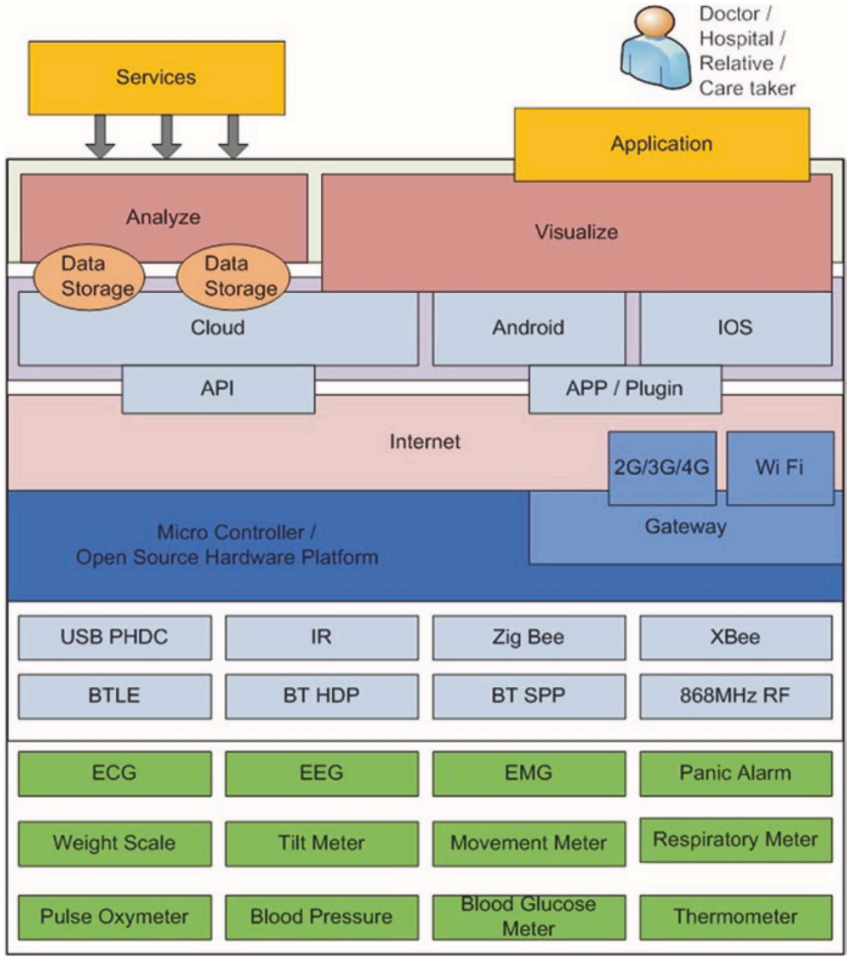
Wireless sensor networks for agriculture [Ojha et al., 2015]

IoT Architecture Scenario: Smart Cities



A generic IoT architecture for a smart city [Jabraeil Jamali et al., 2020]

IoT Architecture Scenario: Healthcare



An IoT architecture for health care [Jabraeil Jamali et al., 2020]

Next in Line...

Overall

- IoT architecture includes a collection of physical objects, sensors, cloud services, developers, actuators, communication layers, users, business layers, and protocols
  - a wildly heterogeneous range of components
- given the wide domain of IoT components (and their possible organisation), there is no clear consensus on a single IoT reference architecture
- so that different researchers typically propose different architectures for IoT systems
  - focussing on the diverse aspects that they consider the most relevant features of IoT systems
- this is why we discussed more than one *architectural view* over IoT systems
  - each one providing a different viewpoint over IoT
  - by pointing out different fundamental features of IoT systems

IoT Architecture

Fundamentals of Internet of Things

Andrea Omicini
andrea.omicini@unibo.it

Bologna Business School, Università di Bologna

Master in Digital Technology Management 2020/2021

References

[Cisco, 2014] Cisco (2014).
The Internet of Things reference model.
White paper (Draft).

[Fielding, 2000] Fielding, R. T. (2000).
Architectural Styles and the Design of Network-based Software Architectures.
PhD thesis, University of California, Irvine, CA, USA.

[Jabraeil Jamali et al., 2020] Jabraeil Jamali, M. A., Bahrami, B., Heidari, A., Allahverdizadeh, P., and Norouzi, F. (2020).
Towards the Internet of Things. Architectures, Security, and Applications.
EAI/Springer Innovations in Communication and Computing (EAISICC). Springer International Publishing.

[Ojha et al., 2015] Ojha, T., Misra, S., and Raghuwanshi, N. S. (2015).
Wireless sensor networks for agriculture: The state-of-the-art in practice and future challenges.
Computers and Electronics in Agriculture, 118:66–84.

[Serpanos and Wolf, 2018] Serpanos, D. and Wolf, M. (2018).
Internet-of-Things (IoT) Systems. Architectures, Algorithms, Methodologies.
Springer.

[Tanenbaum and van Steen, 2007] Tanenbaum, A. S. and van Steen, M. (2007).
Distributed Systems. Principles and Paradigms.
Pearson Prentice Hall, Upper Saddle River, NJ, USA, 2nd edition.