Distributed Systems
From Middleware to the IoT
via Cloud and Blockchain

Digital Culture

Andrea Omicini
andrea.omicini@unibo.it

Master in Gestione d'Impresa / Bologna Business School

Academic Year 2018/2019

1. Distributed Systems & Middleware
2. Blockchain & Smart Contracts
3. The Cloud
4. The Internet of Things

Next in Line...

1. Distributed Systems & Middleware
2. Blockchain & Smart Contracts
3. The Cloud
4. The Internet of Things

Focus on...

1. Distributed Systems & Middleware
   - Definitions
   - Middleware
2. Blockchain & Smart Contracts
   - Basics
   - Smart Contracts
3. The Cloud
   - Premises
   - Definitions
   - Background Technologies
   - Cloud Architecture
   - Cloud Management
4. The Internet of Things
   - Premises
   - Basics

Distributed System: Classic Definitions I

A distributed system can be defined as...

... a collection of *independent* computers that *appears* to its users as a *single coherent system* [Tanenbaum and van Steen, 2007]

User's view

- a possible definition, accounting for an *observational*, user-oriented view
- we may also call it the computer scientist definition of a distributed system
- from an engineering viewpoint, it is a sort of *a posteriori* definition

Distributed System: Classic Definitions II

A distributed system can be defined as...

... a collection of *autonomous* computational entities *conceived* as a *single coherent system* by its designer

Engineer's view

- another possible definition, accounting for a *constructive*, design-oriented view
- we may also call it the computer engineer definition of a distributed system
- from an engineering viewpoint, it is a sort of *a priori* definition

Distributed System: Classic Definitions III

Remarks

- a distributed system is made of a multiplicity of *components*
  - independent / autonomous computational entities (computers, microprocessors, . . .)
  - ! no definition focus specifically on either computational processes or computing devices, here
  - *no assumptions* on their individual nature, structure, behaviour, . . .
  - heterogeneity
- a distributed system can be seen as a single coherent system
  - according to either the user's view or the engineer's view—or both
  - need for *coherence* over multiplicity and heterogeneity

Distributed System: Another Definition

A distributed system can be defined as...

... one in which components located at *networked* computers *communicate* and *coordinate* their actions only by passing *messages*.

[Coulouris et al., 2012]

Remarks

This definition focusses on the following features of distributed systems:

- physical *distribution* of components
- the role of *interaction*—network-based communication and coordination
- uncoupling in terms of *control*

Distributed System: Working as One, Looking as One

Issues: coherence & uniformity

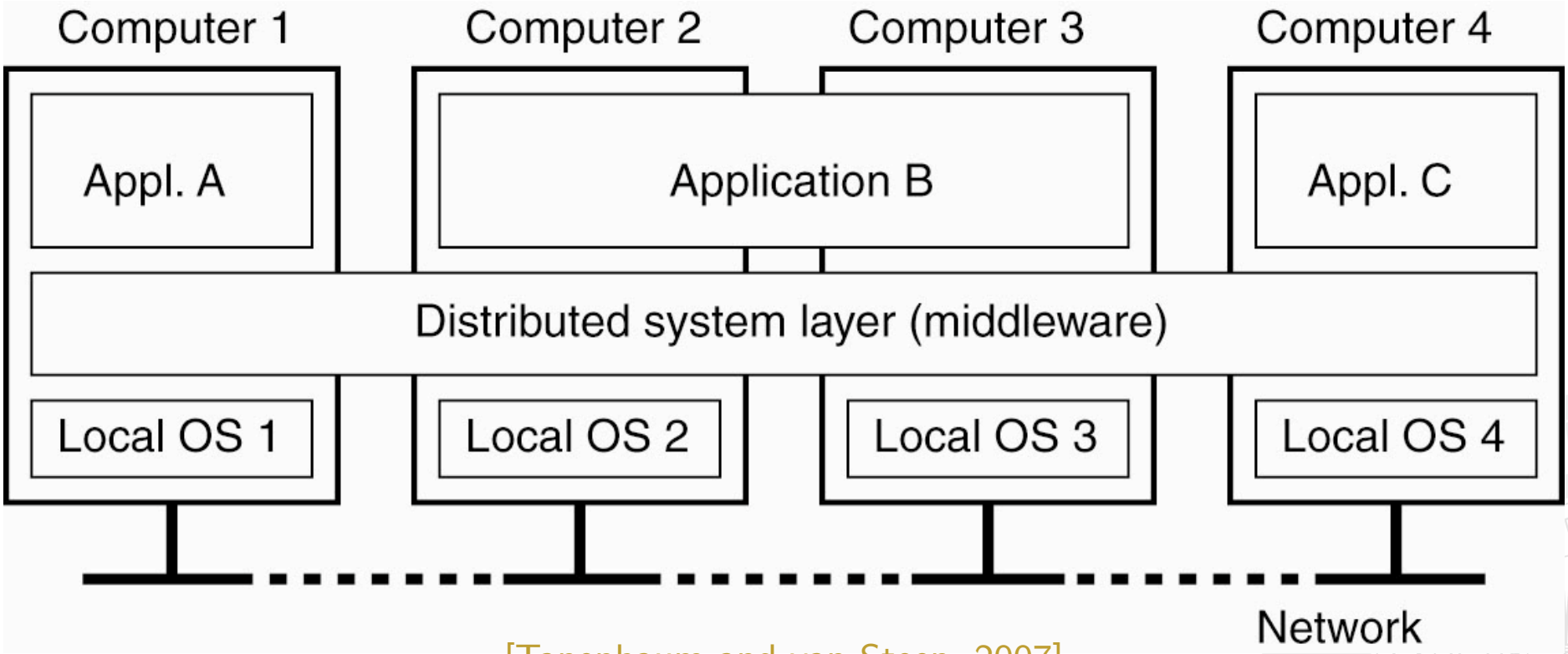
collaboration in order to achieve coherence, many *autonomous* entities should *collaborate*—that is, work altogether as a single (coherent) system

amalgamation in order to achieve uniformity, many *heterogeneous* entities should *amalgamate*—that is, look altogether as a single (uniform) system

Focus on...

1. Distributed Systems & Middleware
   - Definitions
   - Middleware
2. Blockchain & Smart Contracts
   - Basics
   - Smart Contracts
3. The Cloud
   - Premises
   - Definitions
   - Background Technologies
   - Cloud Architecture
   - Cloud Management
4. The Internet of Things
   - Premises
   - Basics

An Architectural View of Distributed Systems: Middleware



[Tanenbaum and van Steen, 2007]

- a distributed system organised as middleware
- the *middleware layer* extends over multiple machines, and offers each application the *same interface*

An Architectural View of Distributed Systems: Old vs. New

Moving from a view of non-distributed computational systems...

- ... by trying to extend the same old interpretation of systems
- good for preserving good habits
- bad when looking for new ideas and new problems
- ! *middleware* has obviously a role to play here

Middleware: A Principled Solution

Middleware for collaboration & amalgamation

Through *separation*, the middleware layer...

- ... enables *meaningful interaction* between autonomous distributed components
  - communication issues like the language for messages and its semantic interpretation
- ... hides differences in technology, structure, behaviour
  - providing both applications and components with a common shared interface—in the same way as operating systems

Next in Line...

1 Distributed Systems & Middleware

2 Blockchain & Smart Contracts

3 The Cloud

4 The Internet of Things

Focus on...

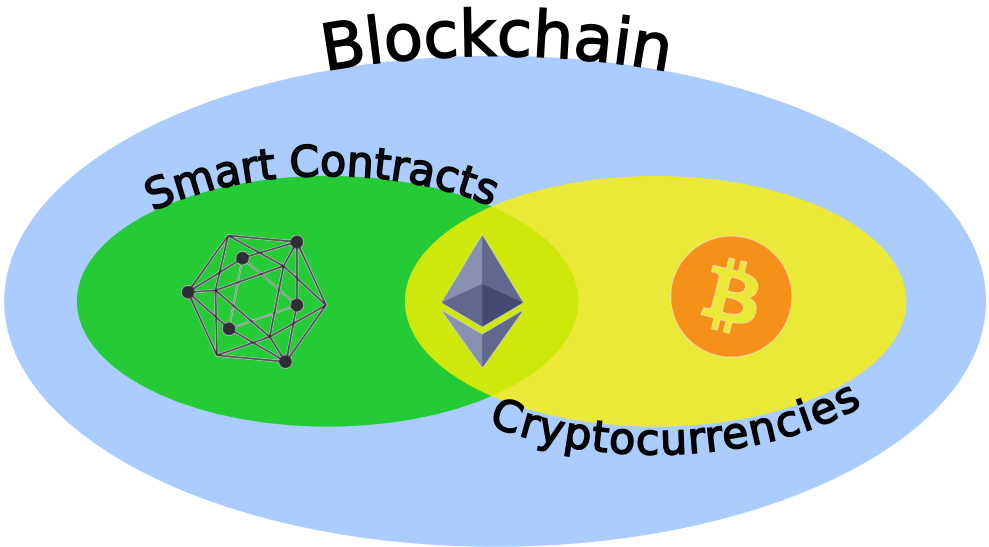
1. Distributed Systems & Middleware
   - Definitions
   - Middleware
2. Blockchain & Smart Contracts
   - Basics
   - Smart Contracts
3. The Cloud
   - Premises
   - Definitions
   - Background Technologies
   - Cloud Architecture
   - Cloud Management
4. The Internet of Things
   - Premises
   - Basics

"Blockchain, the technology behind Bitcoin"

Blockchain ≠ Bitcoin

More precisely:

$$\text{blockchain} \supset \text{cryptocurrencies} \supset \text{Bitcon}$$



HyperLedger Fabric

Ethereum

Bitcoin

First of All... I

The blockchain is ...

... a middleware

... a general model for distributed & decentralised computation

- may be implemented in several ways,
- within heterogeneous application domains,
- and with disparate goals

Bitcoin [Nakamoto, 2008] is ...

... one specific blockchain implementation

- in one specific application domain (cryptocurrencies)
- with a specific goal (digitalize & decentralize fiat currency)

First of All... II

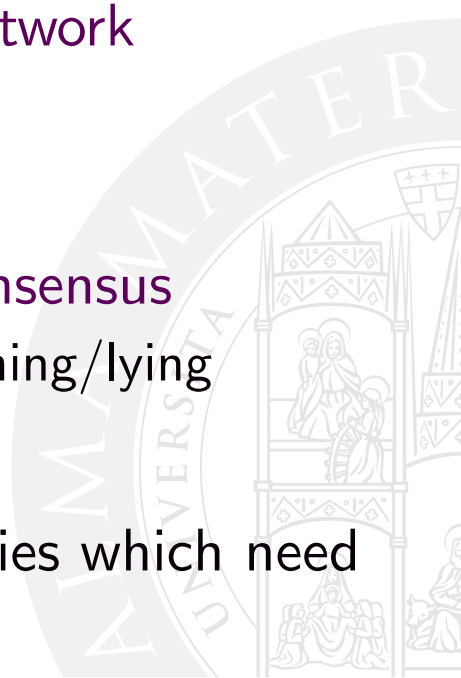
Smart-contracts-enabled blockchains are ...

... a particular sort of blockchain

- exposing (some declination of) the notion of smart contract [Szabo, 1997]
- making them interesting in even more application domains

The Blockchain at a Glance

- shared, *append-only*, transparent, and distributed ledger
  - i.e., a register keeping track of the *order & timing* of transactions
    - i.e., changes to some common data
- secured using cryptographic schemes
  - e.g. 1-way hash functions, digital signatures and certificates, encryption
- the ledger is replicated on multiple nodes of a P2P network
  - removing need for a *trusted* centralized intermediary
  - making the ledger immutable
- transactions are approved and propagated through consensus
  - preventing data loss/corruption due to machines crashing/lying
  - i.e., providing *robustness* against Byzantine failures
- ideal use case: secure hub for mutually untrusted parties which need to interact



BCT as an Heterogeneous Technology

- most of blockchain-related works describe a *specific* blockchain technology (BCT henceforth) using a bottom-up approach
- BCT are a growingly-heterogeneous technology
- in the following we sketch the blockchain in a top-down way, getting informations from a number of sources, being [Nakamoto, 2008, Wood, 2014, Androulaki et al., 2018] the most prominent ones.
- the following description of the blockchain architecture and functioning is strongly inspired to Ethereum[1], being the most mature, studied, and documented smart-contracts-enabled BCT

[1] https://github.com/ethereum/wiki/wiki/White-Paper

Blockchain Overview I

Generally speaking, a *Blockchain Technology* (BCT) is . . .

. . . a clever implementation of a SMR (state machine replication) system keeping track of which users own some assets (representations), by means of a *replicated* ledger, updated through consensus

Blockchain Overview II

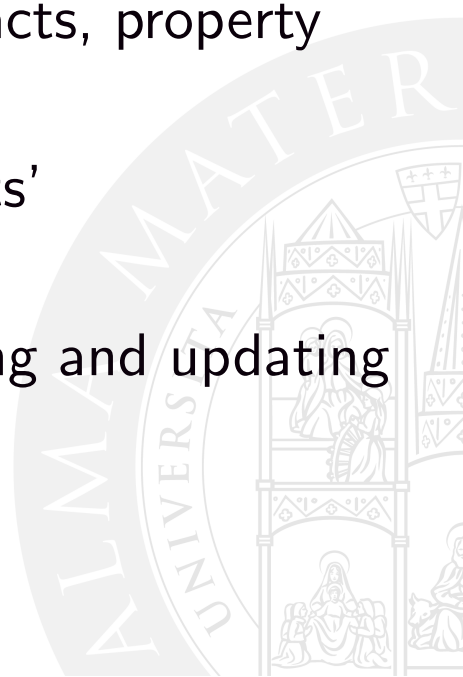
users — are identified though public keys and must sign their transactions with them

assets — may be anything having some value, an owner, and, possibly, some mutable state to be tracked

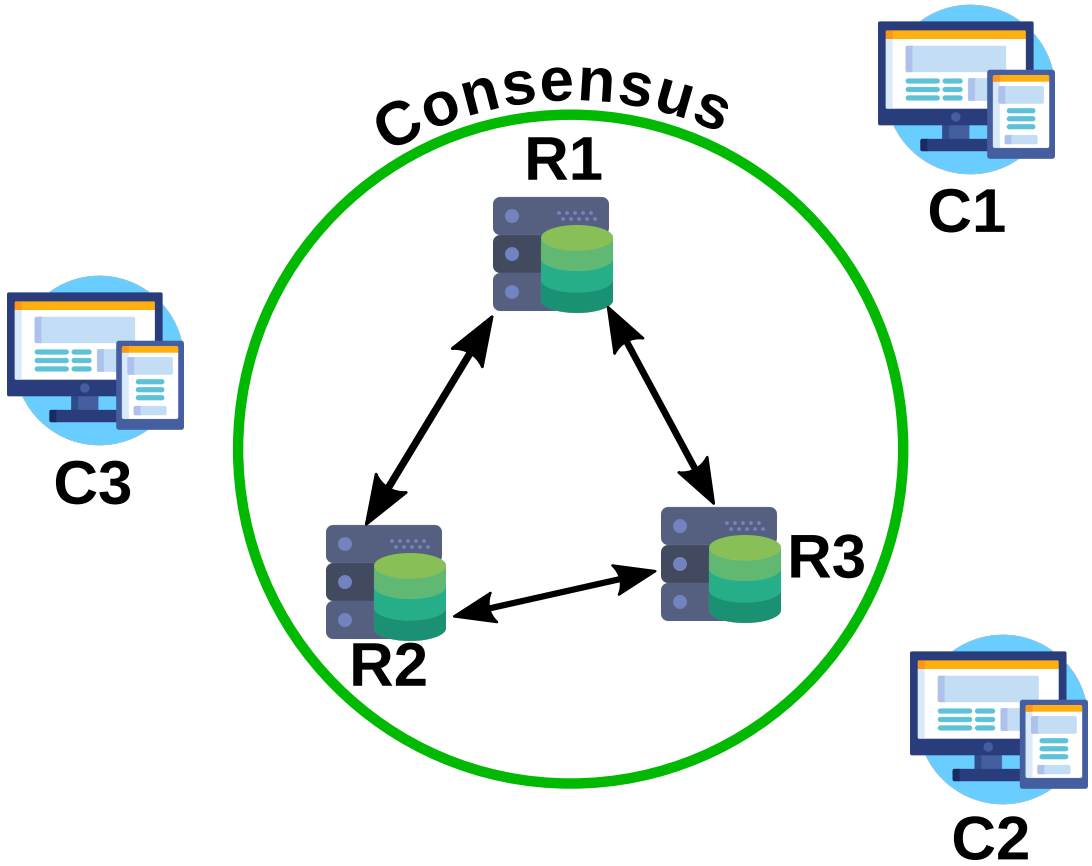
- e.g. events, money, documents, notary acts, property acts, contracts, etc

ledger — is a replicated notary service holding assets' property/state evolution

consensus — a distributed protocol aimed at maintaining and updating the ledger replicas in sync



The Architecture of a BCT



- replicas $R_1, \ldots, R_N$ actually store a copy of the whole ledger
- clients $C_1, \ldots, C_M$ may propose some asset updates to any R_i
  - they need not to store the whole ledger

Blocks and Block Chains

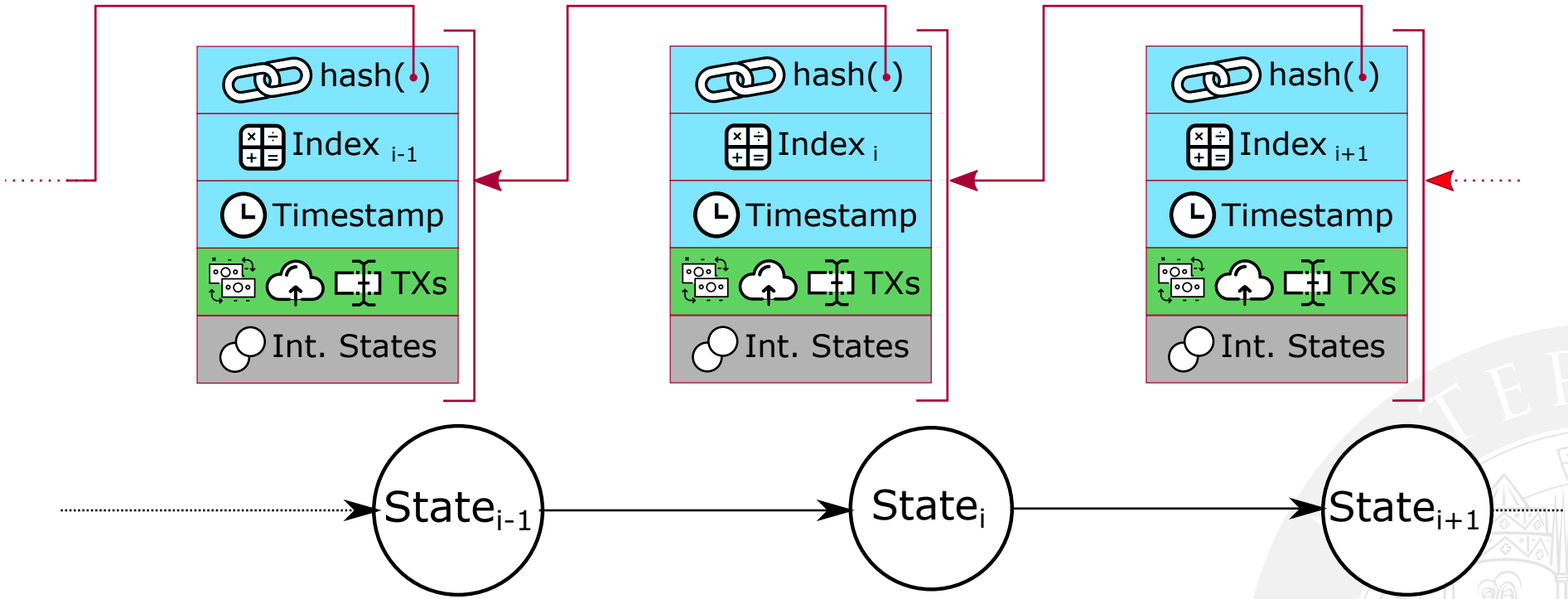


Figure: Graphical representation of the blockchain from the global viewpoint

Blocks Features

- replication + hash-chaining $\rightsquigarrow$ untamperability of the past
- hash-chain + time + ordering $\rightsquigarrow$ timestamping/notary service
  [Haber and Stornetta, 1991]
- cash-chain + cryptographic signatures $\rightsquigarrow$ $\begin{cases} \text{accountability} \\ \text{non-repudiation} \end{cases}$
- supposedly published (almost) periodically
- eventually consistent

Focus on...

1. Distributed Systems & Middleware
   - Definitions
   - Middleware
2. Blockchain & Smart Contracts
   - Basics
   - Smart Contracts
3. The Cloud
   - Premises
   - Definitions
   - Background Technologies
   - Cloud Architecture
   - Cloud Management
4. The Internet of Things
   - Premises
   - Basics

Smart Contracts

Smart contracts (SC) [Szabo, 1997]

Smart contracts are *stateful*, *user-defined*, *reactive*, *immutable* – therefore trustable –, and deterministic processes executing *replicated* and *expressive* computations on the blockchain

stateful — they encapsulate their own state, like OOP objects

reactive — they can only be triggered by some off-chain client

user-defined — users can deploy their own smart contracts implementing any business logic

immutable — their code cannot be altered after deployment

replicated — the blockchain acts as a replicated interpreter for such processes

expressive — they are expressed with a Turing-complete language

SC Features

- SC can be used to automate workflows involving ≥ 2 parties
  - they can act as intermediaries between the parties
  - reducing failures due to corrupted/selfish *human* operators
  - making such workflow (more) trustable
- immutability + inspectability + accountability + replication

  $\Longrightarrow$ can be trusted in handling critical assets
  - (even easier with native cryptocurrency)
- the code is always right, the true history is on the blockchain
  - reducing disputes
  - removing the need for arbitration

! lack of situatedness: totally disembodied [Mariani and Omicini, 2013] data & computation
  - like the cloud, but with no single point of trust

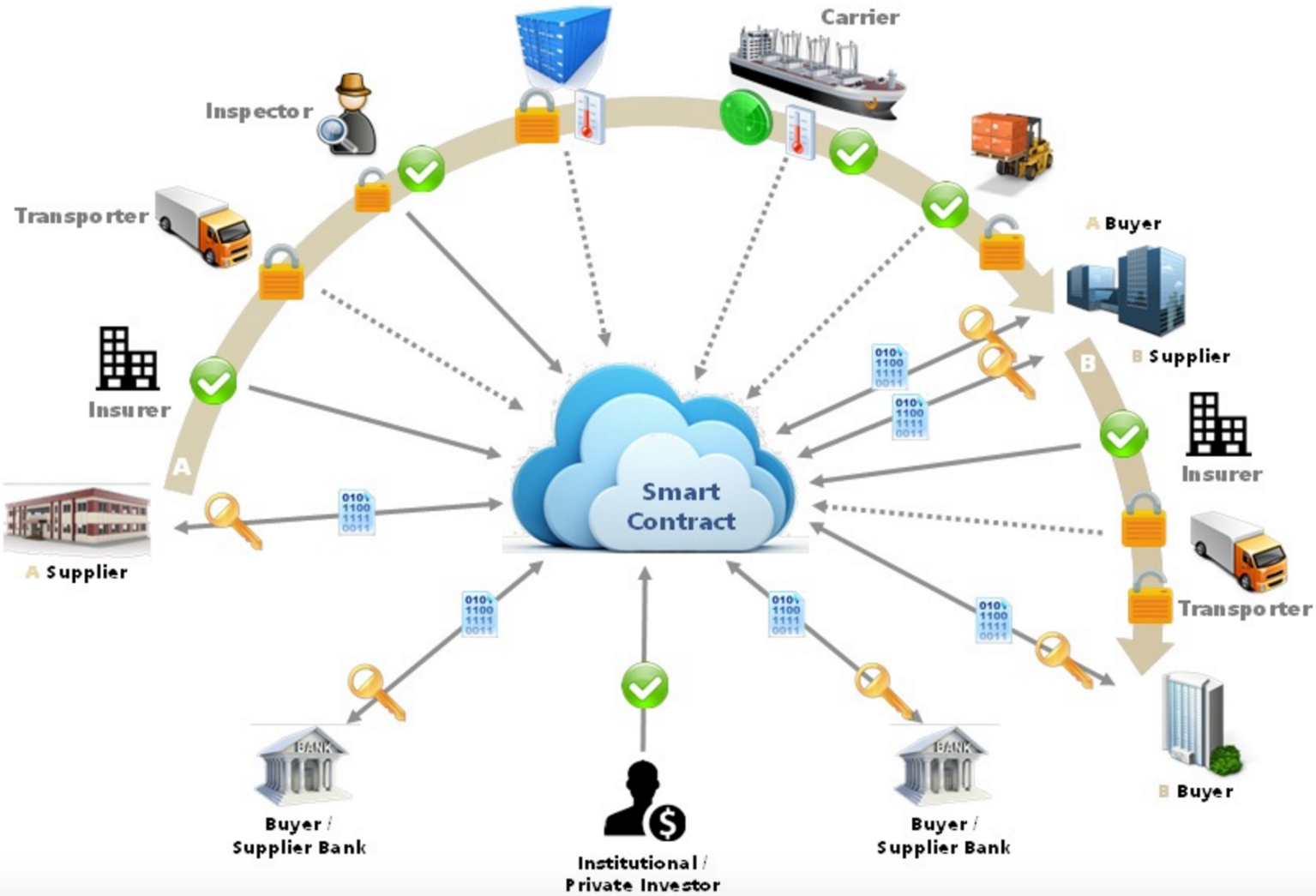
- in a given moment, *N* copies exists of a given SC, one for each replica
  - but they always compute as one
  - "smart contract instance" = "the many copies of a SC instance"

Use Cases & Expectations

Smart contracts are expected to act as intermediaries in

- automatic money management or payments
  - e.g. bills
  - e.g. insurances
  - e.g. shares distribution
- shared governance
  - e.g. Distributed Autonomous Organizations (DAOs)
- automatic asset/goods/supply-chain tracking
  - e.g. in combination with IoT

Smart Contracts + Internet of Things (IoT) I



Smart Contracts + Internet of Things (IoT) II

1. the possibly-many movements of goods could be tracked automatically
   - by one or more interacting SC
2. departures and arrivals of goods may be detected by IoT sensors
   - no way for the many involved actors to lie about quantities, or timings
3. SC could automatically perform payments
   - as soon as movements are detected
   - or refunds in case they are not
4. funders, creditors, inspectors, and end-users can always know the distribution path of products
   - they just have to agree on the policy enforced by the SCs code

The main idea

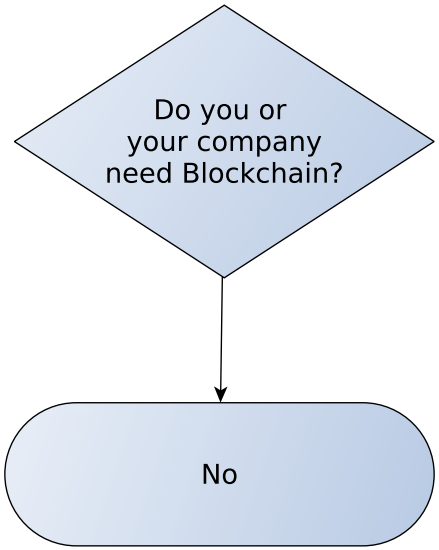
Delegating trust-critical decisions to transparent, and trustable SW agents

Issues

Smart contracts are quite immature

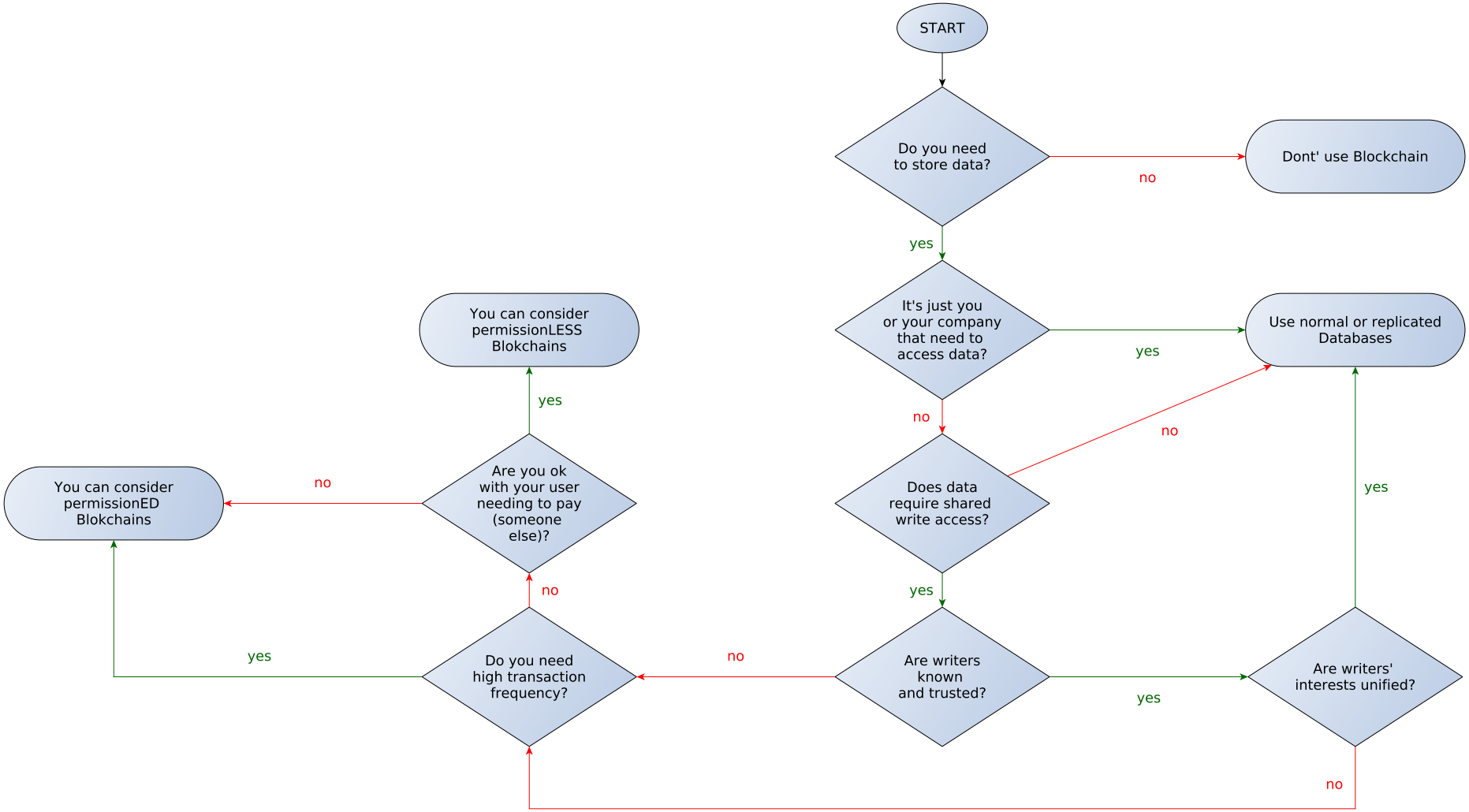
- no privacy or secrets
- no randomness
- inter-SC communication & re-entrancy
- immutability
- control flow & pure reactiveness
- disembodiment & lack of concurrency
- granularity of the computational cost

Do *I / My Company* Need the Blockchain? I



- seriously?
- yes, if its just you or your company
  - you may better choose to use a replicated DB

Do *I / My Company* Need the Blockchain? II



Next in Line...

1 Distributed Systems & Middleware

2 Blockchain & Smart Contracts

3 The Cloud

4 The Internet of Things

Focus on...

1. Distributed Systems & Middleware
   - Definitions
   - Middleware
2. Blockchain & Smart Contracts
   - Basics
   - Smart Contracts
3. The Cloud
   - Premises
   - Definitions
   - Background Technologies
   - Cloud Architecture
   - Cloud Management
4. The Internet of Things
   - Premises
   - Basics

Towards Cloud Computing [Coulouris et al., 2012] I

Resources as commodities

- with the increasing maturity of distributed systems infrastructure, a number of companies are promoting the view of distributed resources as a commodity or utility, drawing the analogy between distributed resources and other utilities such as water or electricity.
- with this model, resources are provided by appropriate service suppliers and effectively *rented rather than owned* by the end user.
- this model applies to both physical resources and more logical services

Towards Cloud Computing [Coulouris et al., 2012] II

Physical resources

- *physical resources* such as storage and processing can be made available to networked computers, removing the need to own such resources on their own
- at one end of the spectrum, a user may opt for a remote storage facility for file storage requirements
- at the other end of the spectrum, users can access sophisticated data centres or computational infrastructure using the sort of services now provided by companies such as Amazon and Google
- operating system virtualisation is a key enabling technology for this approach, implying that users may actually be provided with services by a virtual rather than a physical node, thus offering greater flexibility to the service supplier in terms of resource management

Towards Cloud Computing [Coulouris et al., 2012] III

Logical resources

- *software services* can also be made available across the global Internet using this approach
- many companies offer a comprehensive range of services for effective rental, including services such as email and distributed calendars—e.g., Google Apps
- this development is enabled by agreed standards for software services, for example as provided by *web services*

Towards Cloud Computing [Coulouris et al., 2012] IV

Cloud computing

- the term cloud computing is used to capture this vision of *computing as a utility*
- a cloud is defined as a set of Internet-based application, storage and computing services sufficient to support most users' needs, thus enabling them to largely or totally dispense with local data storage and application software
- the term also promotes a view of *everything as a service*, from physical or virtual infrastructure through to software, often paid for on a per-usage basis rather than purchased
- note that cloud computing reduces requirements on users' devices, allowing very simple desktop or portable devices to access a potentially wide range of resources and services

Focus on...

1. Distributed Systems & Middleware
   - Definitions
   - Middleware
2. Blockchain & Smart Contracts
   - Basics
   - Smart Contracts
3. The Cloud
   - Premises
   - Definitions
   - Background Technologies
   - Cloud Architecture
   - Cloud Management
4. The Internet of Things
   - Premises
   - Basics

Definitions & Remarks I

Definition [Baun et al., 2011]

By using virtualized computing and storage resources and modern Web technologies, cloud computing provides scalable, network-centric, abstracted IT infrastructures, platforms, and applications as on-demand services. These services are billed on a usage basis.

Definitions & Remarks II

Remarks

- even though cloud services usually rely on *distributed systems*, the definition do not specify this
  - in principle, it might be provided also by a single, high-performance server, such as a mainframe
  - also, cloud management is typically determined in a central (and proprietary) manner by a single provider
  - unlike grid computing, where distributed nodes are usually autonomous
- also, a fundamental issue for cloud computing is its *economic impact*
  - being strictly service-oriented and Web-standard-based, cloud computing is an amenable choice fur a huge number of application scenarios
  - many IT approaches and solutions might exploit cloud services and get economical benefits

Definitions & Remarks III

NIST

Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. This cloud model is composed of five essential characteristics, three service models, and four deployment models.

NIST, US Department of Commerce
`http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf`

Definitions & Remarks IV

Five essential features according to NIST

on-demand self-service — Services can be provided on demand to consumers with no humans in the loop

broad network access — Services are available over the network in real-time through standard mechanisms

resource pooling — Resources are pooled to enable concurrent service provision to multiple users (multi-tenant model), while being adjusted to the actual demand of each user

(rapid) elasticity — Requests for extra resource are self-managed and automatic in relation to demand; from the consumer's perspective, the supply of resources has no limit

measured (quality of) service — Services leverage a quantitative and qualitative metering capability making usage-based billing and validation of the service quality available

Focus on...

1. Distributed Systems & Middleware
   - Definitions
   - Middleware
2. Blockchain & Smart Contracts
   - Basics
   - Smart Contracts
3. The Cloud
   - Premises
   - Definitions
   - Background Technologies
   - Cloud Architecture
   - Cloud Management
4. The Internet of Things
   - Premises
   - Basics

Background

Technologies behind the Cloud

There are some fundamental technologies the Cloud depends upon

- virtualisation
- service-oriented architectures (SOA)
- Web Services

Virtualisation I

Resources in the cloud

- resources can be detached from their physical nature by means of *virtualisation*
- virtualised resources for the cloud include servers, data stores, networks, and software
- the basic idea is to *pool* physical resources, make them transparent, and manage them as a whole
- pools provide transparent access to resources

Virtualisation II

Benefits for providers

resource usage — Load balancing towards full capacity of physical servers

management — Resource pool management can be automated

consolidation — Reducing the number of physical servers, increasing efficiency and cost effectiveness

energy consumption — From consolidation

less space required — From consolidation

emergency planning — Moving virtual machines from one resource pool to another

Virtualisation III

Benefits for customers

dynamic behaviour — Resource pools can easily satisfy almost all requests dynamically—even when requests are peaking

availability — Automated fault tolerance procedures make services highly available

access — Isolation of virtual machines make it possible to safely delegate management functionality to the customer

emancipation — Customers can purchase IT capabilities from a self-service portal, without humans in the loop

Virtualisation IV

Drawbacks

cost of abstraction — the abstraction layer requires resources; however, modern virtualisation techniques make such a cost more and more negligible

system management — the physical infrastructure grows in complexity; automation of procedures and availability of management tools requires however much less staff in the overall

Virtualisation V

Concepts

OS virtualisation — applications run within *containers* layered upon the host OS

platform virtualisation — OS & applications run in a virtual environment, based on a VM monitor or *hypervisor*

storage virtualisation — dynamically scalable storage space as a service

network virtualisation — virtual IP addresses, virtual LAN (VLAN)

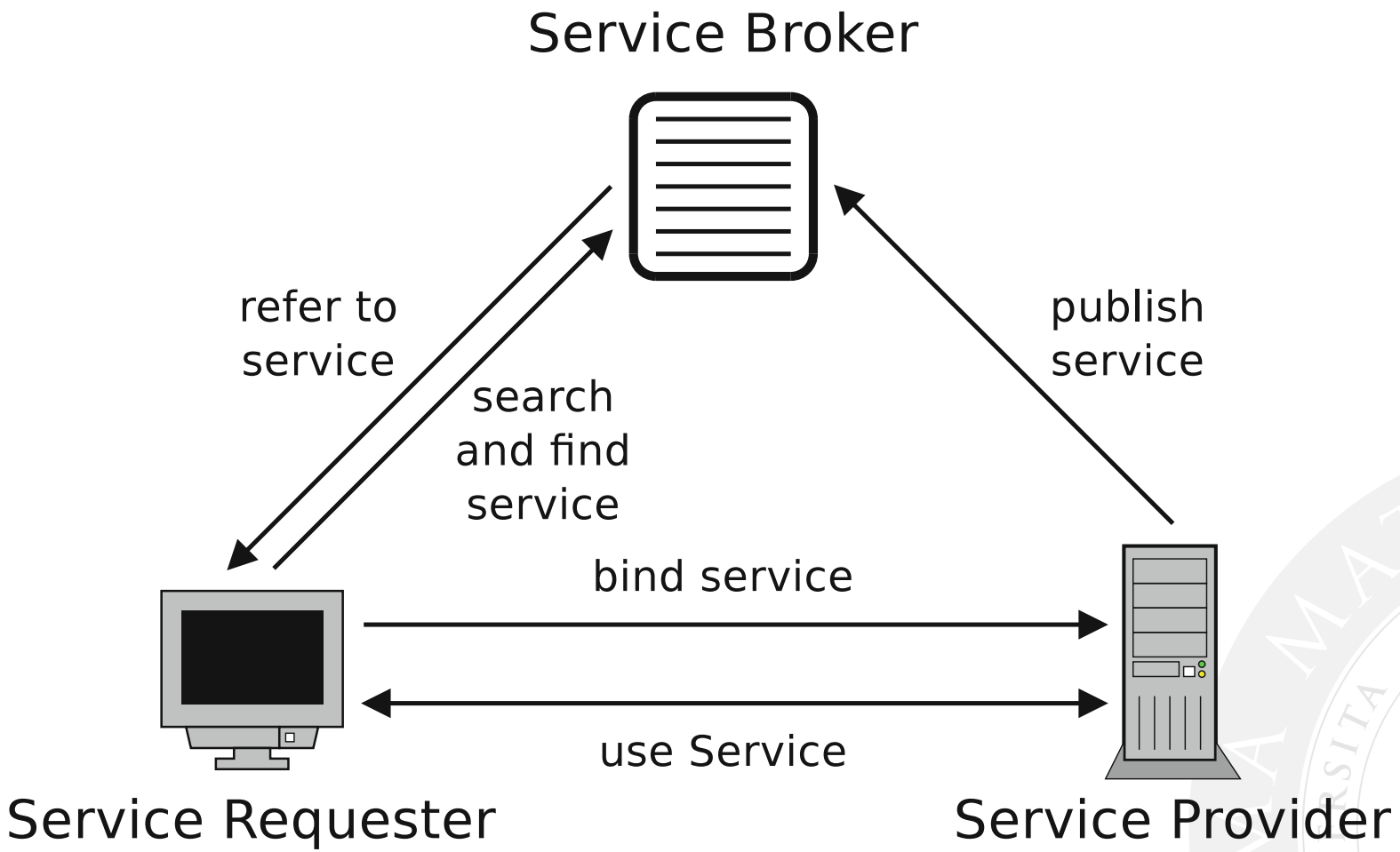
application virtualisation — software sales model for centralised management and distribution of applications through the network

Service-Oriented Architectures (SOA) I

Another prerequisite for cloud computing

- service-oriented architectures (SOA) are architectures whose components are implemented as independent services
- services interact via messages, and coordinate together (orchestration)
- cloud makes virtualised IT infrastructures, platforms, and whole applications available for use

Service-Oriented Architectures (SOA) II



[Baun et al., 2011]

Web Services I

Internet resources as WS

- distributed cloud computing systems integrate heterogeneous resources
- unpredicability of Internet connection demands for loosely coupled, asynchronous, message-based communication

→ Web Services

Web Services II

Web Services Architecture Working Group

A Web Service is a software application identified by a URI, whose interfaces and binding are capable of being defined, described and discovered by XML artifacts and supports direct interactions with other software applications using XML based messages via internet-based protocols.

`http://www.w3.org/TR/2002/WD-wsa-reqs-20020429`

Focus on...

1. Distributed Systems & Middleware
   - Definitions
   - Middleware
2. Blockchain & Smart Contracts
   - Basics
   - Smart Contracts
3. The Cloud
   - Premises
   - Definitions
   - Background Technologies
   - Cloud Architecture
   - Cloud Management
4. The Internet of Things
   - Premises
   - Basics

Perspectives on Cloud

Technical vs. organisational viewpoints

- a *technical* viewpoint focuses on the functional features of cloud architectures
- however, this is not the only one available and meaningful perspective: the way in which the cloud is deployed is also relevant
- in particular, the different roles played by providers and customers is essential for the *organisational* viewpoint

About...

1. Distributed Systems & Middleware
   - Definitions
   - Middleware
2. Blockchain & Smart Contracts
   - Basics
   - Smart Contracts
3. The Cloud
   - Premises
   - Definitions
   - Background Technologies
   - Cloud Architecture
     - Organisational View
     - Technical View
   - Cloud Management
4. The Internet of Things
   - Premises
   - Basics

Public Cloud

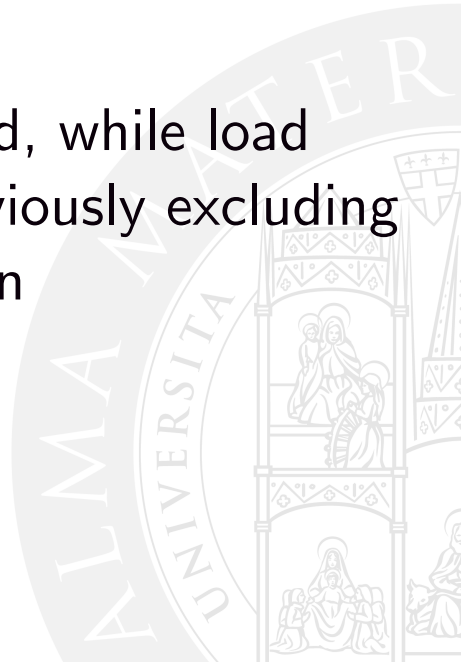
- a *public cloud* (or, *external* cloud) is offered by a provider to users outside its organisation boundaries
- the cloud service is made publicly available through an automated service (typically, a Web portal) with no humans in the loop
- obligations are pre-defined in terms of the service, and automatically negotiated
- billing based on actual usage

Private Cloud

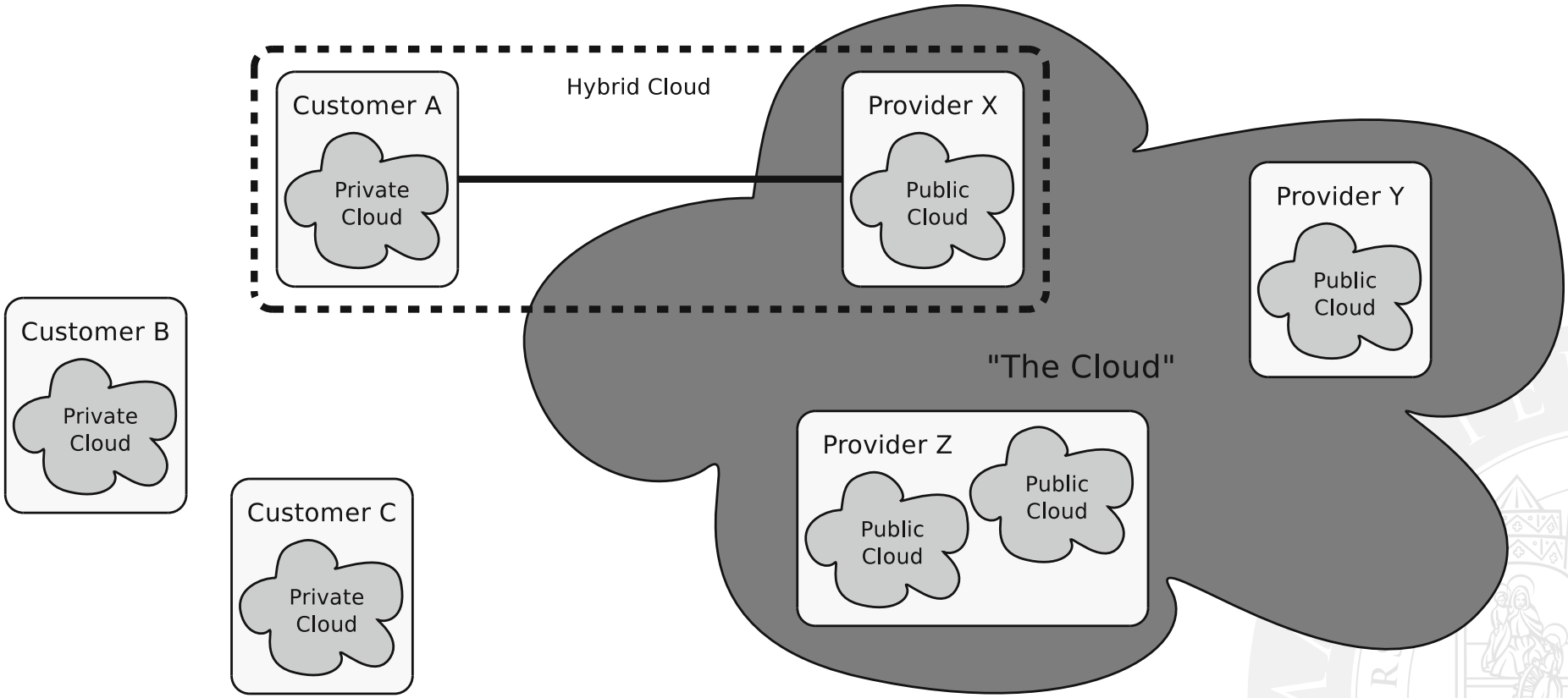
- providers of a *private cloud* (or, *internal* cloud) belong to the same organisation as the users
- the cloud service is made publicly available through an automated service (typically, a Web portal) with no humans in the loop
- the main aim of private cloud is security, by keeping control of organisation data

Hybrid Cloud

- in order to exploit the same tech & tools of public clouds
- and to scale automatically when private cloud is not enough
- *hybrid clouds* are built by bringing together resources from both private and public clouds
- there, normal operation is handled by the private cloud, while load peaks are typically transferred to the public cloud, obviously excluding data and resources that are critical for the organisation



Public / Private / Hybrid Clouds



[Baun et al., 2011]

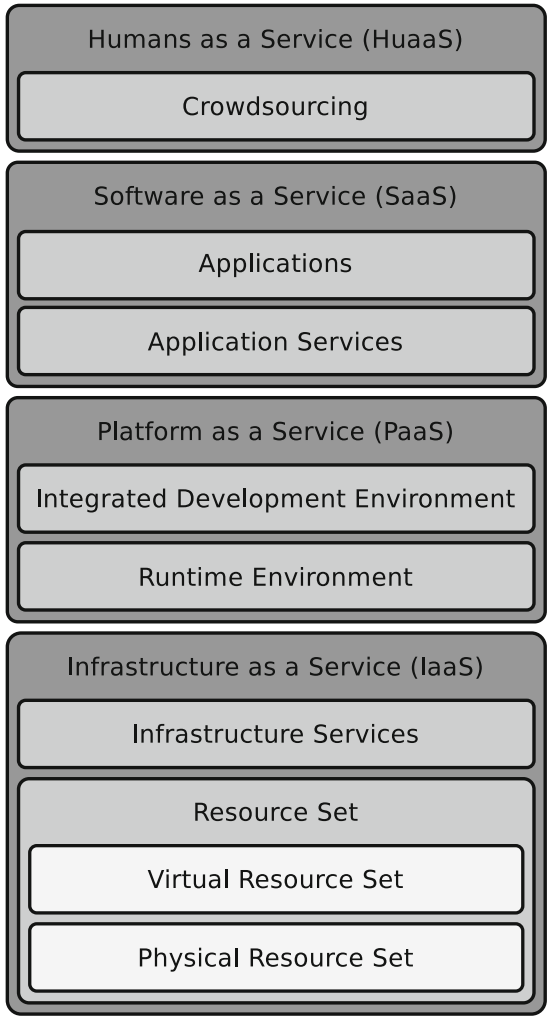
About...

1. Distributed Systems & Middleware
   - Definitions
   - Middleware
2. Blockchain & Smart Contracts
   - Basics
   - Smart Contracts
3. The Cloud
   - Premises
   - Definitions
   - Background Technologies
   - Cloud Architecture
     - Organisational View
     - Technical View
   - Cloud Management
4. The Internet of Things
   - Premises
   - Basics

Everything as a Service (XaaS)

- through the cloud, so many diverse infrastructures, services, and applications are made available as Web services
- as a result, cloud services present with diverse functions and goals
- the cloud offering can be then depicted as an architecture layered according to different layers of abstractions
- the three main layers follow the Everything-as-a-Service paradigm (XaaS)

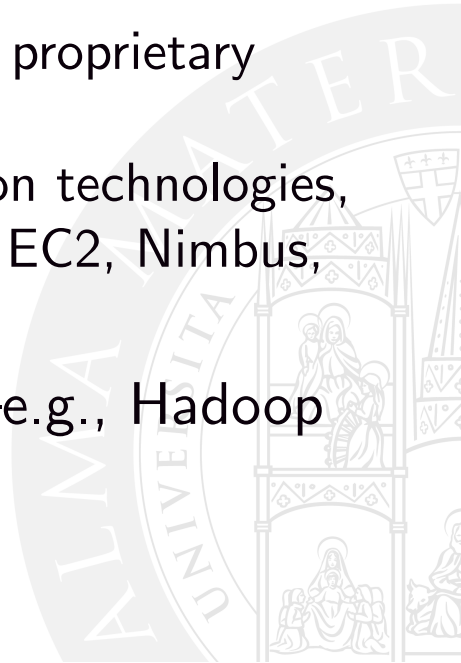
Cloud Stack



[Baun et al., 2011]

Infrastructure as a Service (IaaS)

- the *IaaS layer* provides an abstract view on the physical resources — computers, mass storage systems, networks, etc.
- a user interface is provided to manage resources in the resource set sub-layer (RS), allowing users allocate and use them
- items in the resource set can be subdivided into
  - physical resource set (PRS), representing and offering proprietary physical hardware—e.g., Emulab, iLO
  - virtual resource set (VRS), built on top of virtualisation technologies, and making virtual instances available—e.g., Amazon EC2, Nimbus, OpenNebula
- infrastructure services, too, belong to the IaaS layer—e.g., Hadoop MapReduce, Amazon S3, Dropbox



Platform as a Service (PaaS)

- the *PaaS layer* provides developers with tools for the cloud
- programming environments (PE) and execution environments (EE) for programming the cloud

PE e.g., Django Framework, Sun Caroline

EE e.g., Google App Engine (Python, Java), Azure by Microsoft (multilanguage)

Software as a Service (Saas)

- the *SaaS layer* directly provides users with software via cloud
- applications are no longer installed locally, but are instead stored and managed by the providers, and downloaded when needed
- application services, such as Google Maps
- full-fledged applications, such as Google Docs, Microsoft Windows Live

Humans as a Service (Huaas)

- the *HuaaS layer* is the top layer of the cloud computing stack, showing that the cloud paradigm can be extended to include services provided by *human beings as resources*
- until humans exhibit capabilities that outperform computer systems, their technical integration as resources is potentially useful
- e.g., *crowdsourcing*—such as Amazon Mechanical Turk

Focus on...

1. Distributed Systems & Middleware
   - Definitions
   - Middleware
2. Blockchain & Smart Contracts
   - Basics
   - Smart Contracts
3. The Cloud
   - Premises
   - Definitions
   - Background Technologies
   - Cloud Architecture
   - Cloud Management
4. The Internet of Things
   - Premises
   - Basics

Service Level Agreements (SLAs)

- a service level agreement is an agreement between a service provider and a service consumer related to the service level (quality of service, QoS)
- the SLA implies a mutual agreement with respect to security, priorities, responsibilities, guarantees, and billing modalities
- furthermore, SLA specifies metrics such as availability, throughput, response times, and others
- two phases for service level management
  - agreement on the quality of service
  - service monitoring at runtime

Lifecycle & Automation

- cloud services have a lifecycle: made available by the provider, selected and used by the customer, closed and detached at the end, finally accounted for
- groups of similar resources are managed automatically as an *ensambles*, stepping through *monitoring*, *analysis*, *scheduling*, *execution*

Management Services & Tools

- many tools are made available by providers, either via CLI or via Web
- for
  - *monitoring*
  - *control*
  - *development*

Security Management

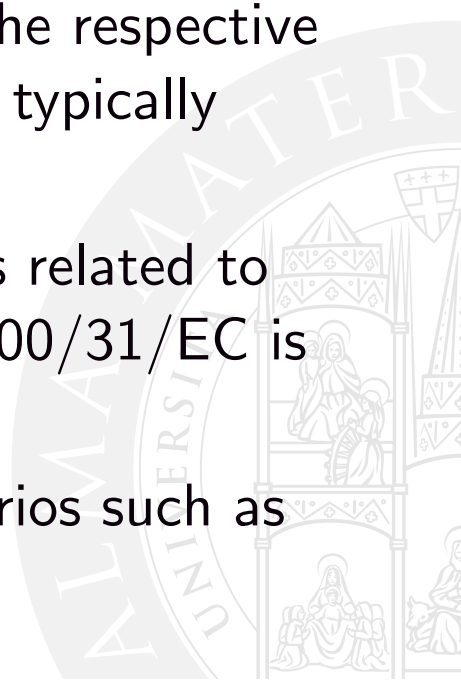
- nothing new here, safety goals are the same as a local data center
  - confidentiality
  - integrity
  - availability
  - authenticity
  - accountability
  - pseudonymity

Risk Management

- again, cloud does not add much in the risk management perspective
- risks are more or less the same of any outsourcing process, and linked to data transfer, service disruption, provider bankruptcy, vendor lock-in

Legal Compliance

- compliance ensures that laws are applicable to cloud computing as to renting (in case of paid services) or lending (with respect to unpaid services)
- geographical location of providers affects applicable law
- e.g., according to article 4 of the European Data Protection Directive, the pertaining national data protection legislation of the respective country is applicable—which is why cloud services are typically provided region-based
- particular attention should be devoted to legal aspects related to personal data processing—e.g., in Europe directive 2000/31/EC is applicable
- special legislation takes care of sensitive data in scenarios such as healthcare and finance



Next in Line...

1 Distributed Systems & Middleware

2 Blockchain & Smart Contracts

3 The Cloud

4 The Internet of Things

Focus on...

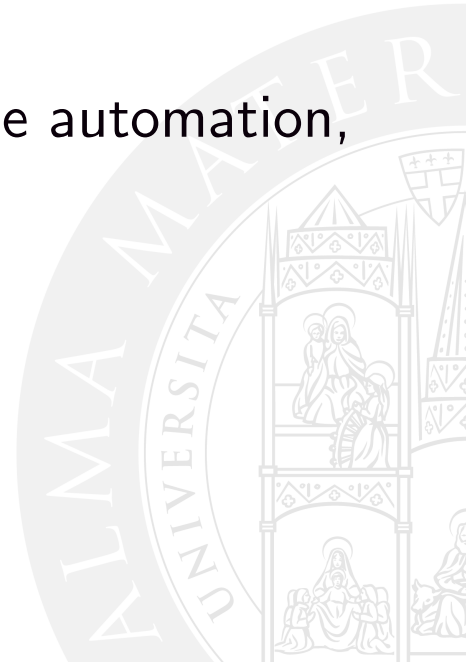
1. Distributed Systems & Middleware
   - Definitions
   - Middleware
2. Blockchain & Smart Contracts
   - Basics
   - Smart Contracts
3. The Cloud
   - Premises
   - Definitions
   - Background Technologies
   - Cloud Architecture
   - Cloud Management
4. The Internet of Things
   - Premises
   - Basics

Embedded Systems

- special-purpose computational systems
- devoted to a specific *function*
- typically associated to / depending on the original function of the hosting object
- interacting with physical world through sensors and actuators
- HW and SW part, typically
- software is *embedded*

Pervasive Systems

- when embedded systems are everywhere
- and networked
- computational systems become *pervasive*
- examples: home appliance, consumer electronics, office automation, vehicles, ...

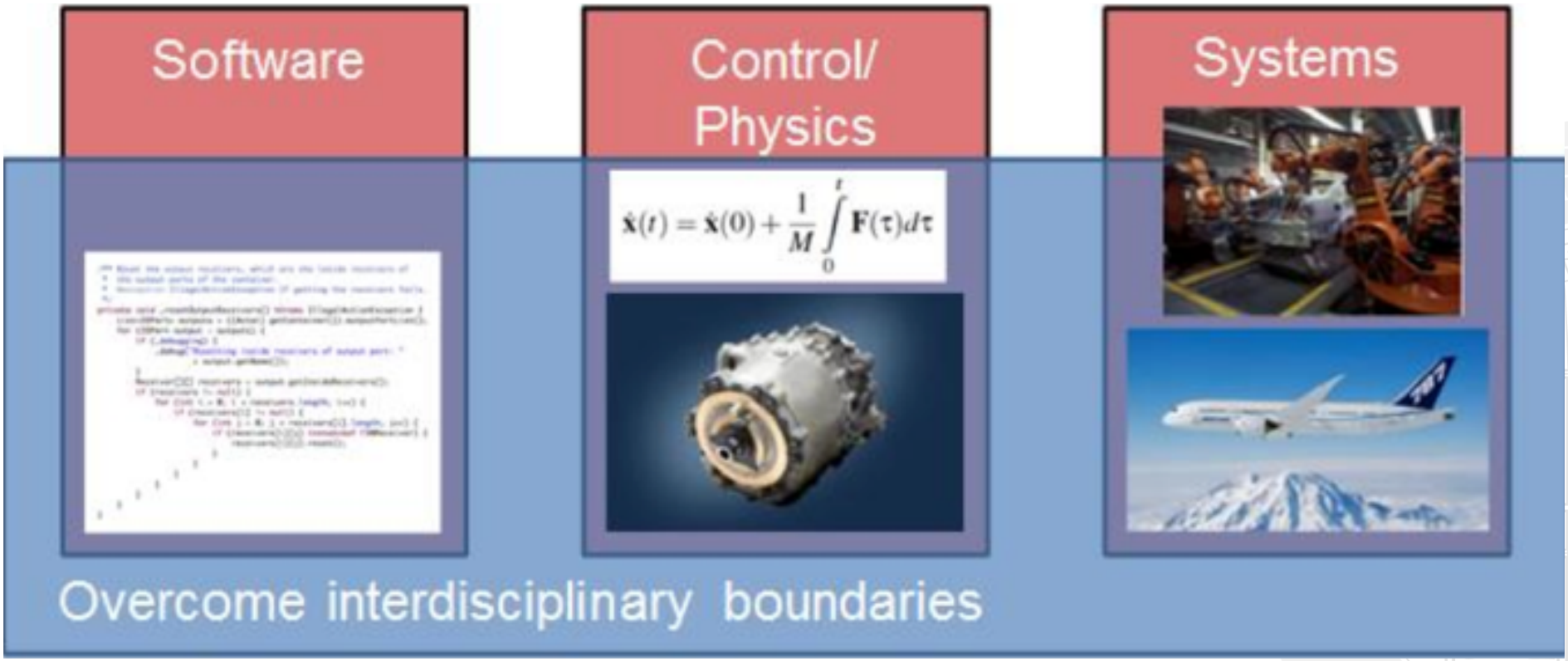


Typical Features

- tightly-constrained resources
- efficiency
  - energy efficient, code-size efficient, run-time efficient, weight efficient, cost efficient
- dependability,
  - reliability, availability, safety, maintainability
- reactiveness & real-time

Cyber-Physical Systems (CPS)

- integrating computation with(in) physical systems
- unlike information processing systems
  - handling time
  - correctness, accuracy, precision
  - asynchrony, true concurrency



Technology

- single-purpose processors—e.g., PLA, PAL, FPGA
- single board micro controller–e.g., Arduino
- systems-on-chip (SOC) & single-board CPU—e.g., Raspberry

Focus on...

1. Distributed Systems & Middleware
   - Definitions
   - Middleware
2. Blockchain & Smart Contracts
   - Basics
   - Smart Contracts
3. The Cloud
   - Premises
   - Definitions
   - Background Technologies
   - Cloud Architecture
   - Cloud Management
4. The Internet of Things
   - Premises
   - Basics

Internet of Things (IoT)

- devices and systems connecting physical world to the Internet
- e.g.
  - smart objects, smart meters
  - wearable devices
  - asynchrony, true concurrency
  - smart watches, human implanted devices, smart glasses
  - home automation systems and lighting controls
  - smartphones
  - wireless sensor networks
  - ...

IoT Systems

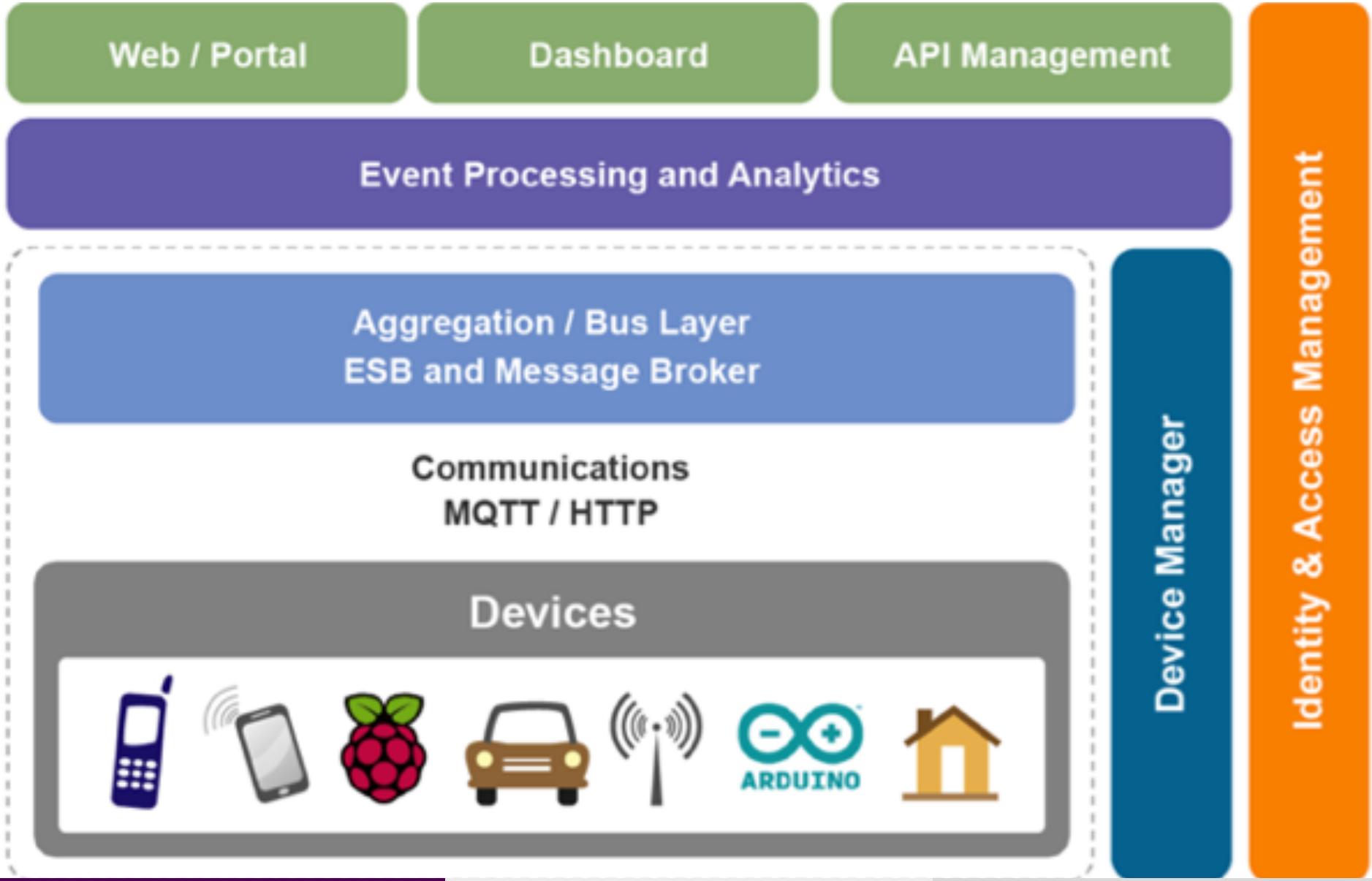
Three sorts of components of IoT systems

- devices
- server-side
- (possibly) low-power gateway

Classes of devices

- small devices: MCU, SOC—e.g., ArduinoUno
- medium devices—e.g., micro-router, embedded OS
- - complete platforms—e.g., Linux, smartphones, Raspberry Pi

IoT Middleware



IoT Platforms

- Samsung
  - SmartThings https://smartthings.developer.samsung.com/
- Google
  - AndroidThings https://developer.android.com/things
  - nest https://nest.com/works-with-nest/
- Intel
  - IoTivity https://www.iotivity.org/
- Apple
  - Homekit http://www.apple.com/it/ios/homekit/
- Qualcomm / AllSeenAlliance
  - AllJoynframework https://allseenalliance.org/framework
- Oracle IoT platform
  - http://www.oracle.com/us/solutions/internetofthings
- ARM mbed
  - https://mbed.org/
- WSO2
  - http://wso2.com/landing/internet-of-things/

Conclusion

IoT Overall

- objects that compute and connect
- their function improved / augmented
- possibly cooperating with other objects
- producing / consuming / moving possibly huge amounts of data—streams
- complex event processing (CEP)

Directions

- Web of Things (WoT)
- Internet of Intelligent Things (IoIT)

References I

Androulaki, E., Barger, A., Bortnikov, V., Cachin, C., Christidis, K., De Caro, A., Enyeart, D., Ferris, C., Laventman, G., Manevich, Y., Muralidharan, S., Murthy, C., Nguyen, B., Sethi, M., Singh, G., Smith, K., Sorniotti, A., Stathakopoulou, C., Vukolić, M., Cocco, S. W., and Yellick, J. (2018).
Hyperledger fabric: A distributed operating system for permissioned blockchains.
In *13th EuroSys Conference (EuroSys '18)*, New York, NY, USA. ACM.

Baun, C., Kunze, M., Nimis, J., and Tai, S. (2011).
Cloud Computing. Web-Based Dynamic IT Services.
Springer Berlin Heidelberg.

Coulouris, G., Dollimore, J., Kindberg, T., and Blair, G. (2012).
Distributed Systems. Concepts and Design.
Pearson, 5th edition.

Haber, S. and Stornetta, W. S. (1991).
How to time-stamp a digital document.
In Menezes, A. J. and Vanstone, S. A., editors, *Advances in Cryptology-CRYPT0' 90*, pages 437–455. Springer.

References II

Mariani, S. and Omicini, A. (2013).
TuCSoN on cloud: An event-driven architecture for embodied / disembodied coordination.
In Aversa, R., Kolodzej, J., Zhang, J., Amato, F., and Fortino, G., editors, *Algorithms and Architectures for Parallel Processing*, volume 8286 of *LNCS*, pages 285–294. Springer International Publishing Switzerland.
13th International Conference (ICA3PP-2013), Vietri sul Mare, Italy, 18-20 December 2013. Proceedings, Part II.

Nakamoto, S. (2008).
Bitcoin: A peer-to-peer electronic cash system.

Szabo, N. (1997).
Formalizing and securing relationships on public networks.
First Monday, 2(9).

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

Wood, G. (2014).
Ethereum: a secure decentralised generalised transaction ledger.

Distributed Systems
From Middleware to the IoT
via Cloud and Blockchain

Digital Culture

Andrea Omicini
andrea.omicini@unibo.it

Master in Gestione d'Impresa / Bologna Business School

Academic Year 2018/2019