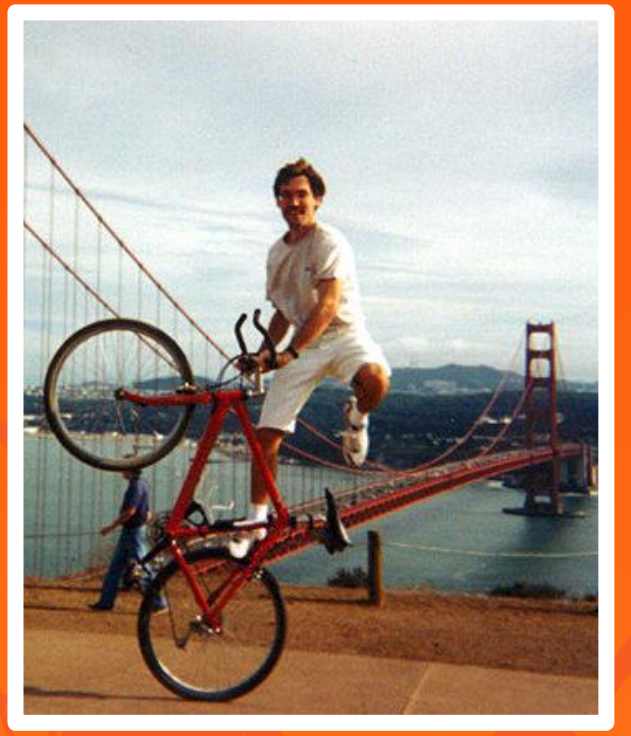
Architecture in the Age of Things

Frank Buschmann, Gregor Hohpe



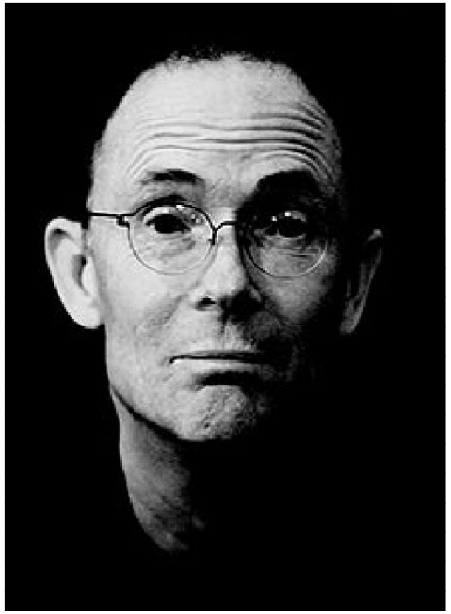
Frank Buschmann

Siemens AG, Corporate Technology

Frank.Buschmann@siemens.com

Gregor Hohpe

info@EnterpriseIntegrationPatterns.com



The future is already here,

it’s just not evenly distributed yet.

William Ford Gibson

Blurring of the physical and on-line world

Phones and watches are smart today, even dumb things become smart

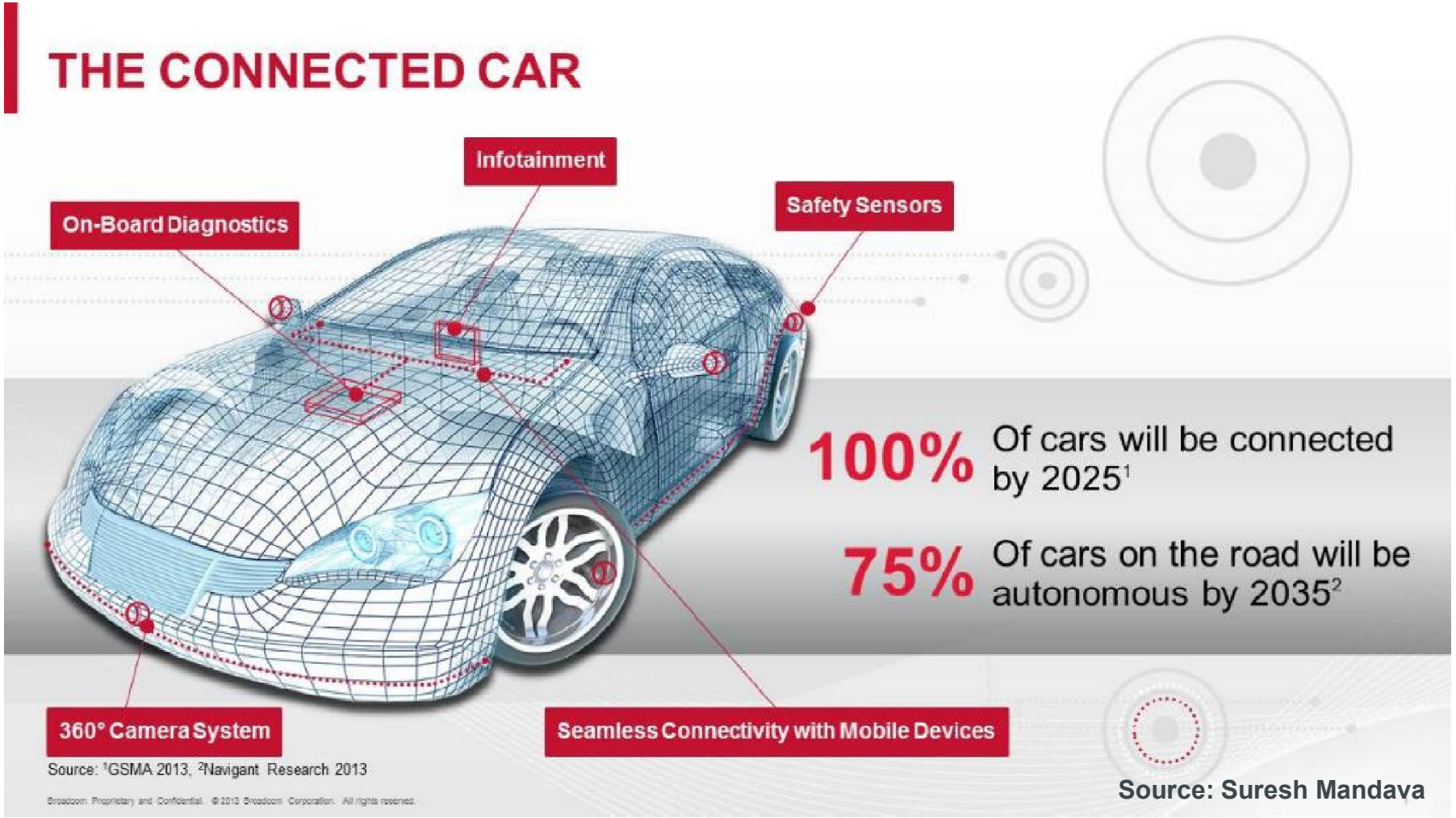
nest

UNDER ARMOUR

PHILIPS

Blurring of the physical and on-line world

More complex things become smart



Source: Suresh Mandava

Industry examples
Insurance

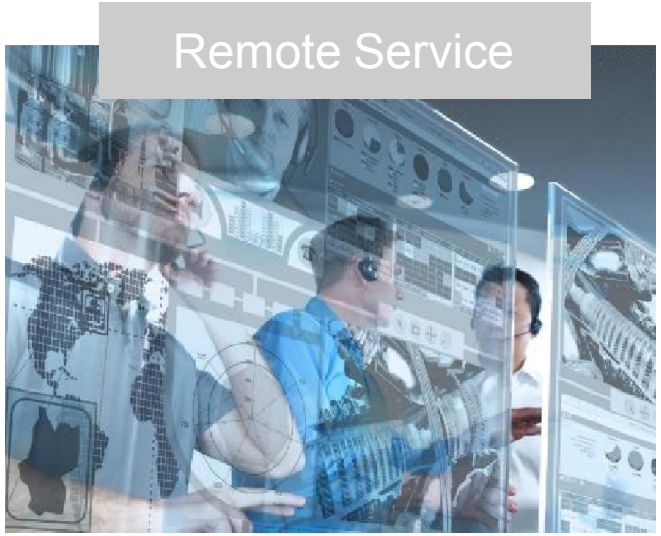
Automotive Telematics - Pay As/How you Drive

Home Automation and Security - Claim Prevention

Source: Allianz SE

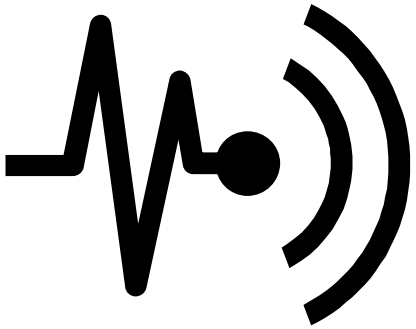
Industry examples
Energy and Mobility

Source: Siemens AG



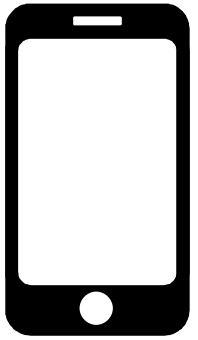
Technology advances helped distribute the future more evenly

Tipping points everywhere

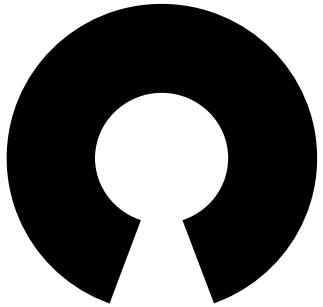


Sensors

Battery Power

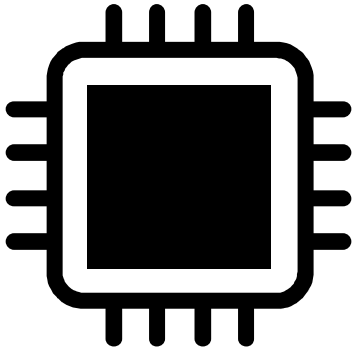


Universal User Interface

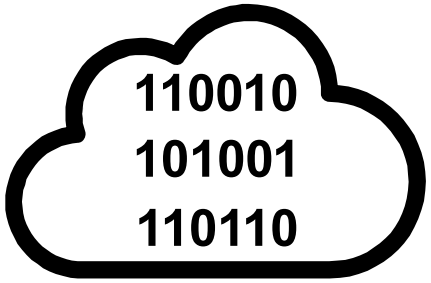


Open Source Software

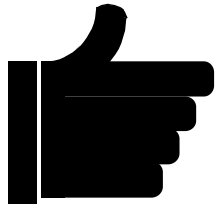
Connectivity



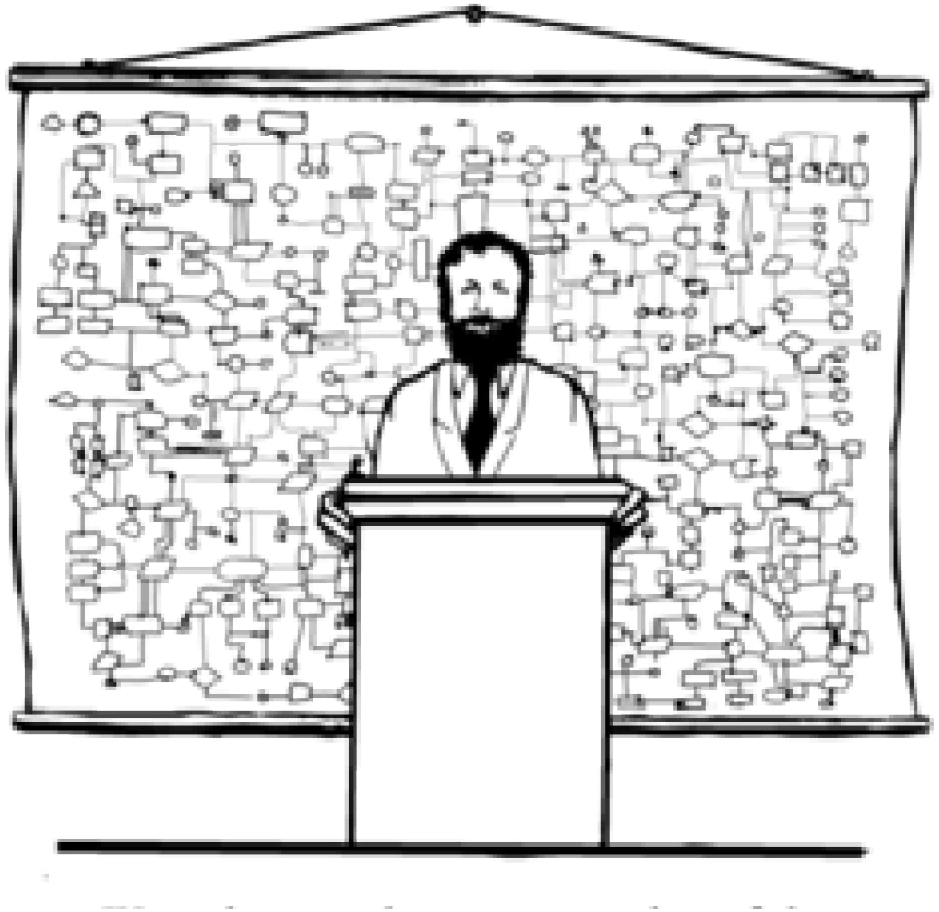
Low Power Processors



Big Data Analytics

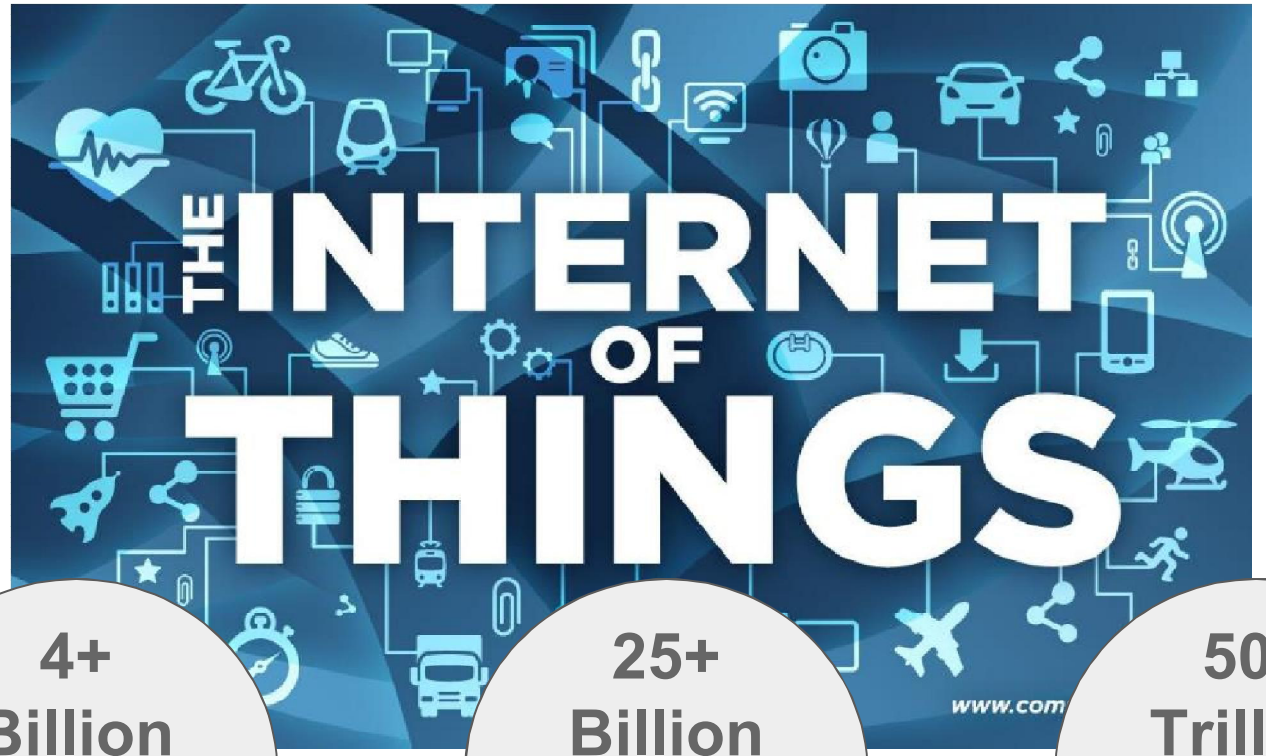


Acceptance



Now that the future is here how to design systems in the age of the IoT?

Do you think you can control



4+ Billion connected people

25+ Billion connected systems

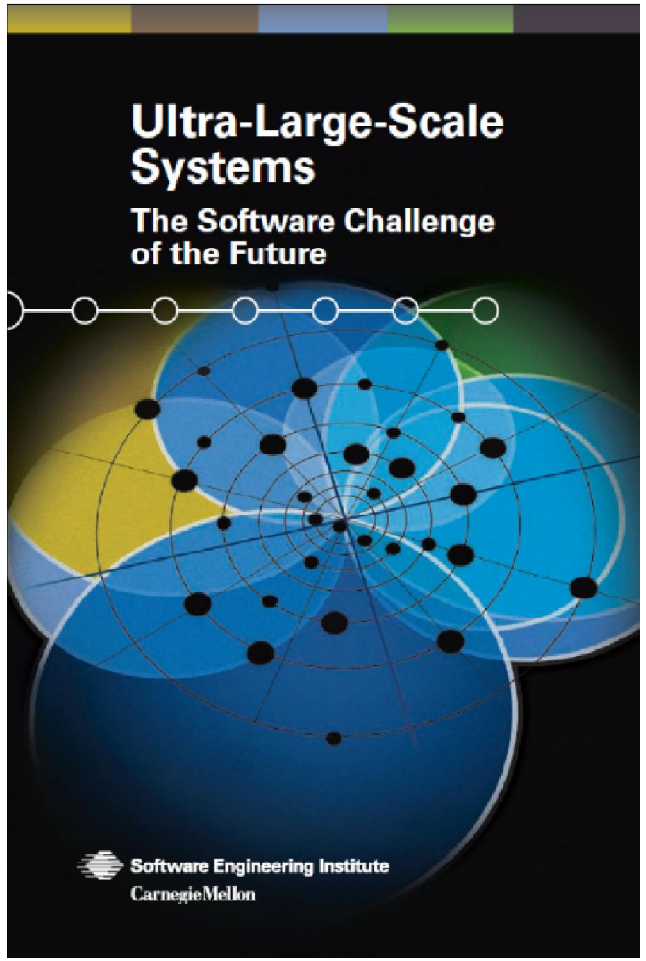
50+ Trillion Gigabyte of data

Source: International Data Corporation (IDC)

in 2020

The IoT is an ultra-large-scale system

ULS will push far beyond the size of today's systems by every measure



Scale: code; users; data managed; connections among software components; hardware elements

Failure is the norm

Decentralized Operations

Inherently **Conflicting**, Unknowable, and Diverse **Requirements**

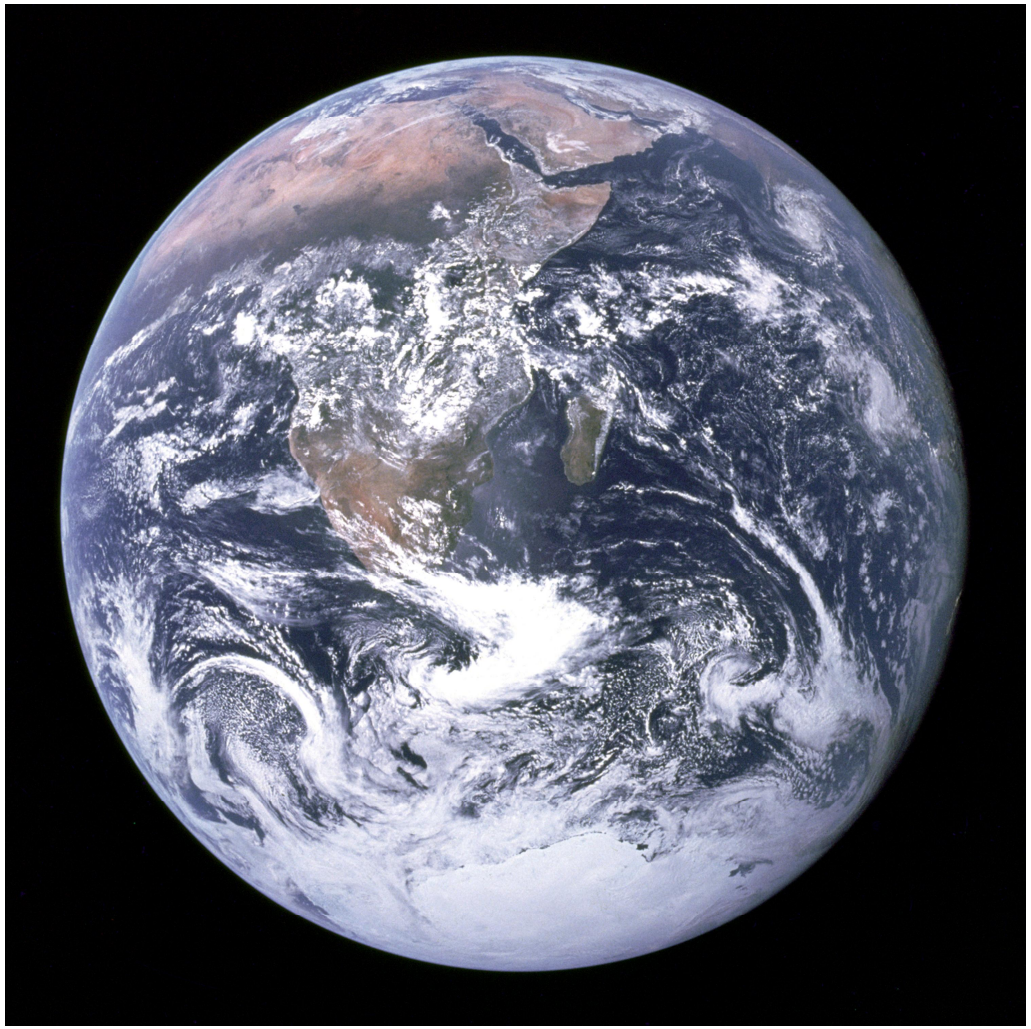
Continuous Evolution and **Deployment**

Heterogeneous, **Inconsistent**, and **Changing Elements**

Erosion of User / System Boundary

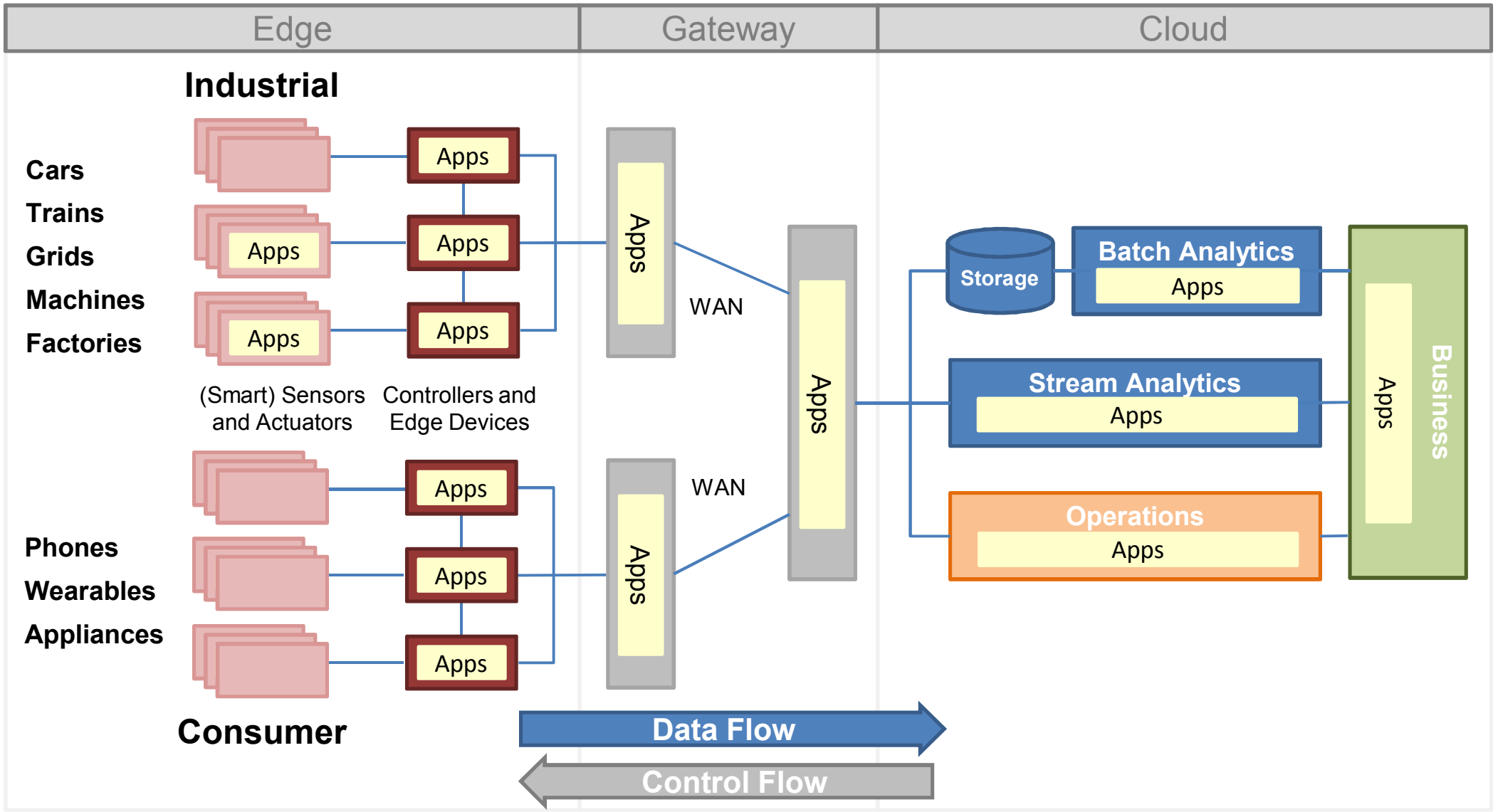
High-level view

The earth from the view of Apollo 17 on December 7. 1972

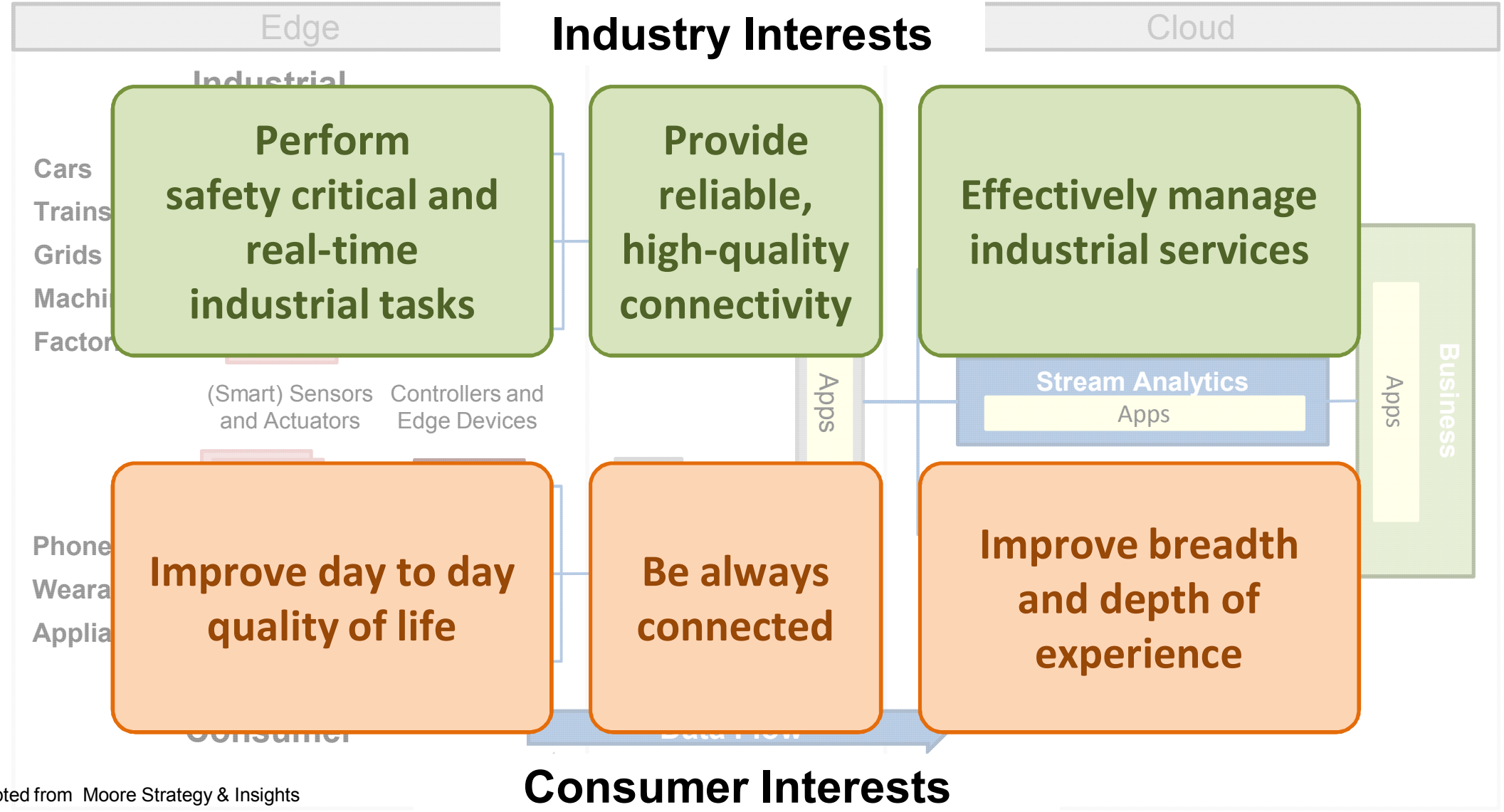


Source: NASA/Apollo 17 crew; taken by either Harrison Schmitt or Ron Evans

Base-line architecture sketch

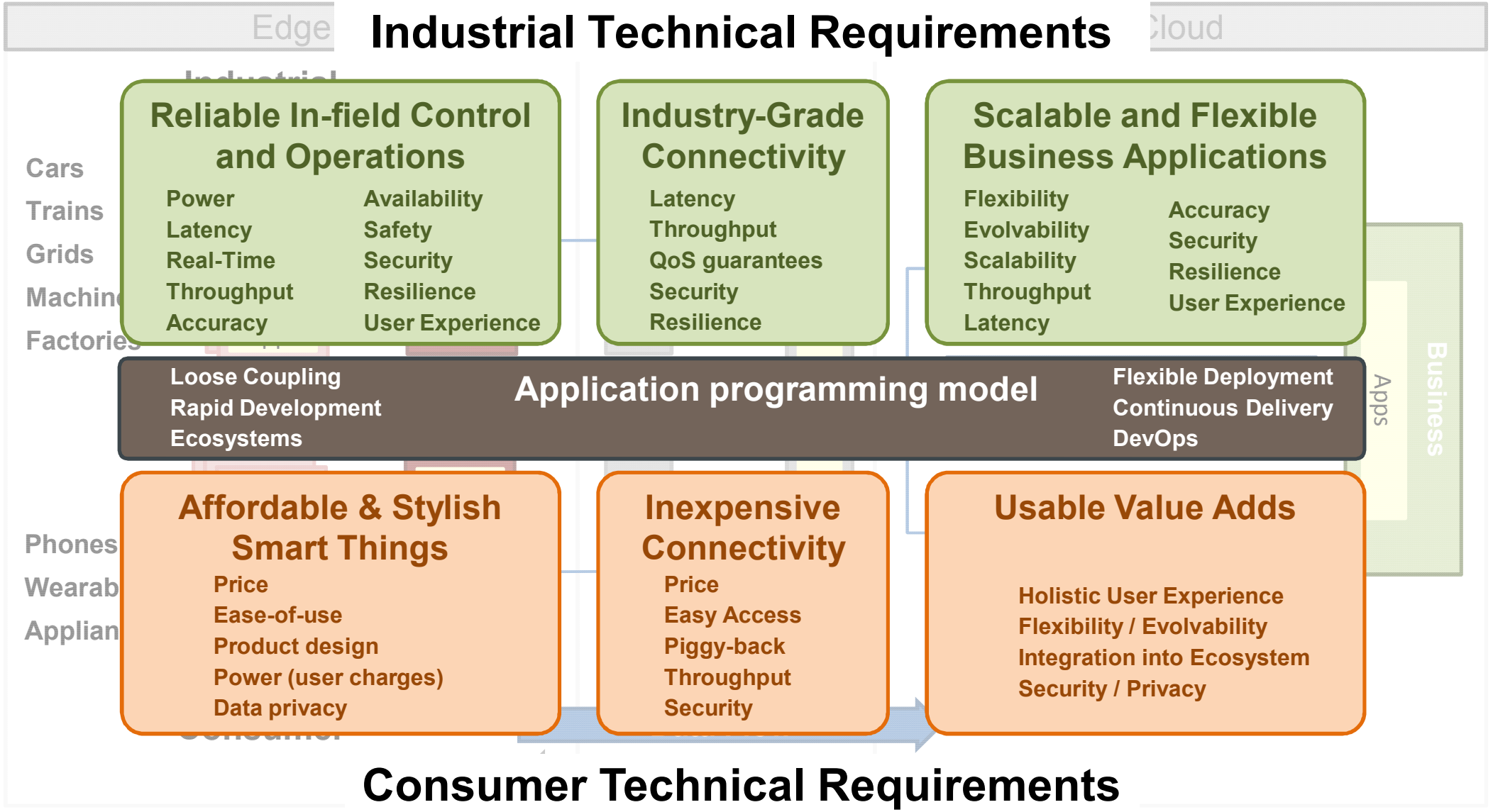


Industry vs. consumer interests



Adapted from Moore Strategy & Insights

Industry vs. consumer technical requirements



Base-line Architecture

Micro Services

Event-driven Architectures

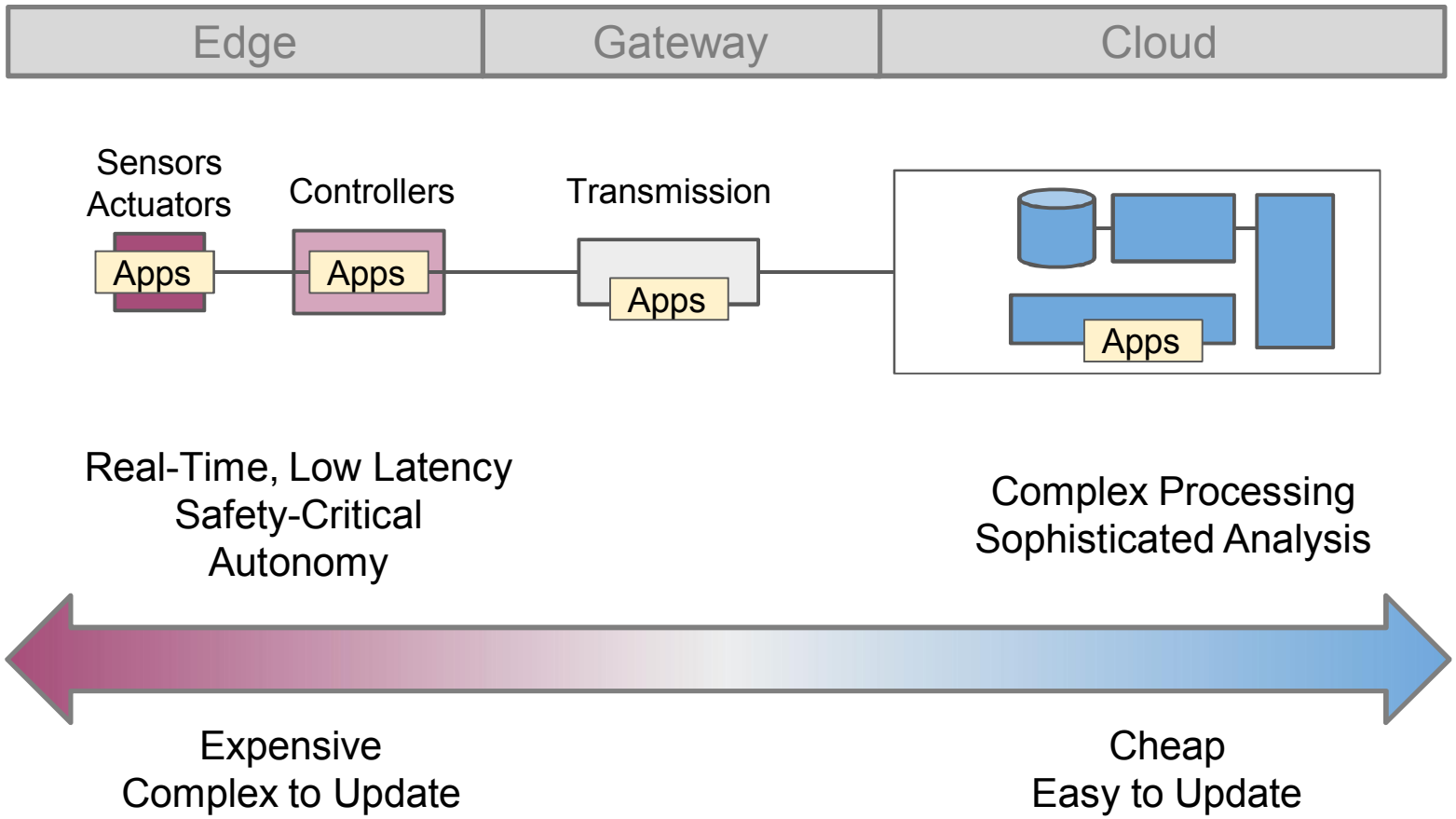
Integrated Runtime

Ecosystem Readiness

Photo from Pexels

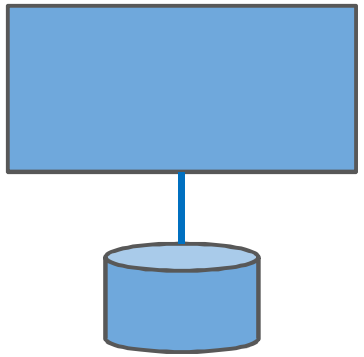
Base-Line Architecture

Trade-offs in Allocating Application Logic

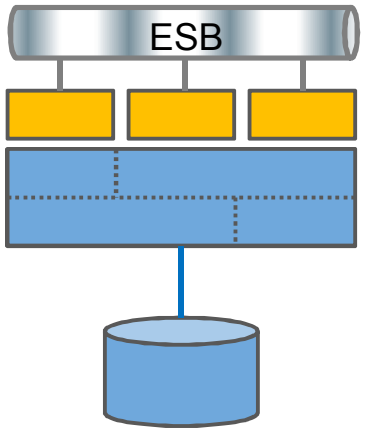


Base-Line Architecture

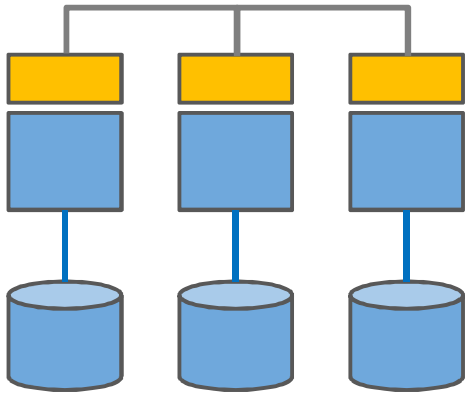
Microservice architecture style



- All functions in a single deployment
- Often scales only vertically



- Service interfaces over monolith(s)
- Integration of multiple independent systems
- Data integrity through transactions

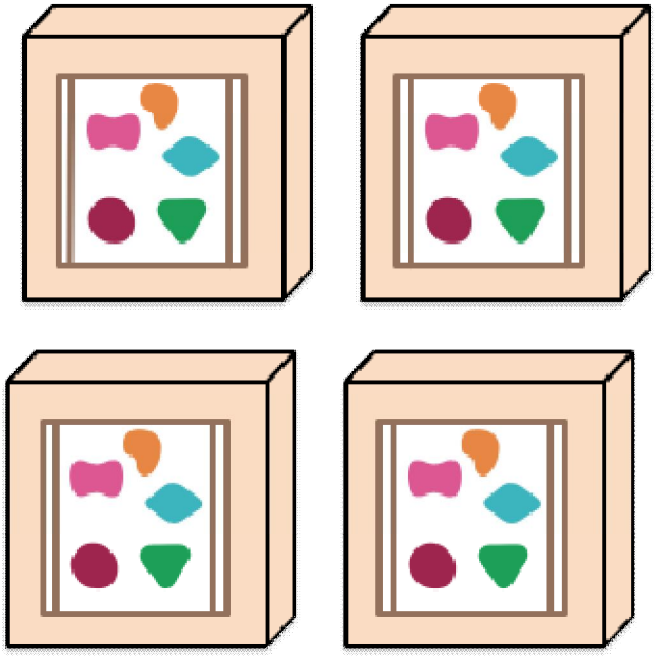


- Focused on one thing
- Independently deployable
- “Dumb Pipe”
- Small enough, but not smaller
- Decentralized data management

Adapted from Sam Newman & Martin Fowler

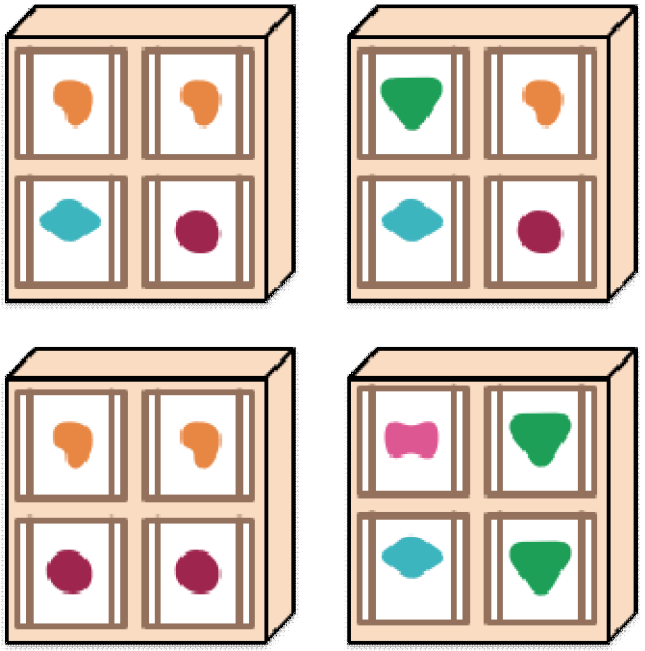
Base-Line Architecture
Scalability

Monolith



Scales by replicating
the application

Microservices

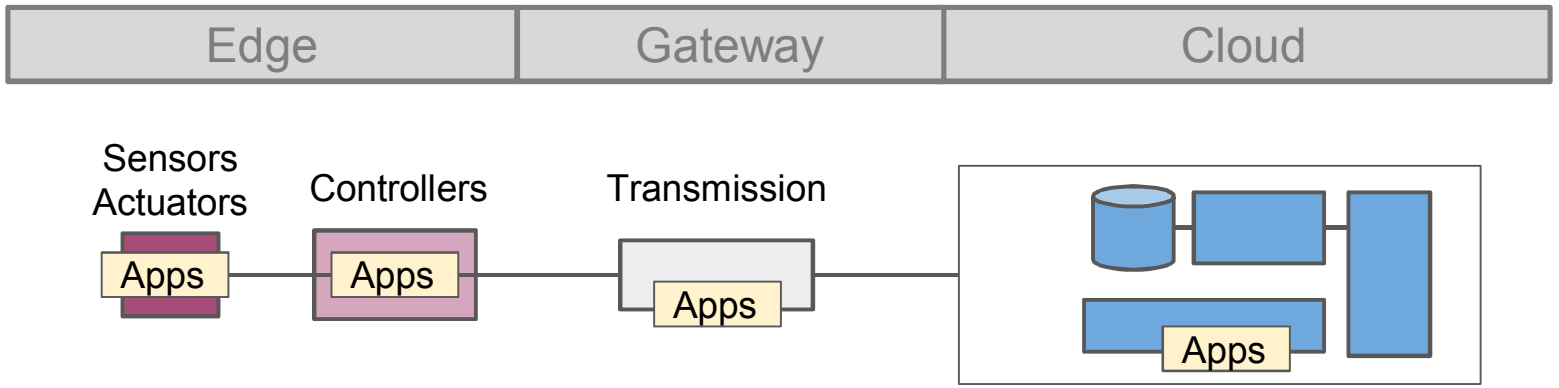


Scales independently
by distributing services

Image source: Martin Fowler

Base-Line Architecture

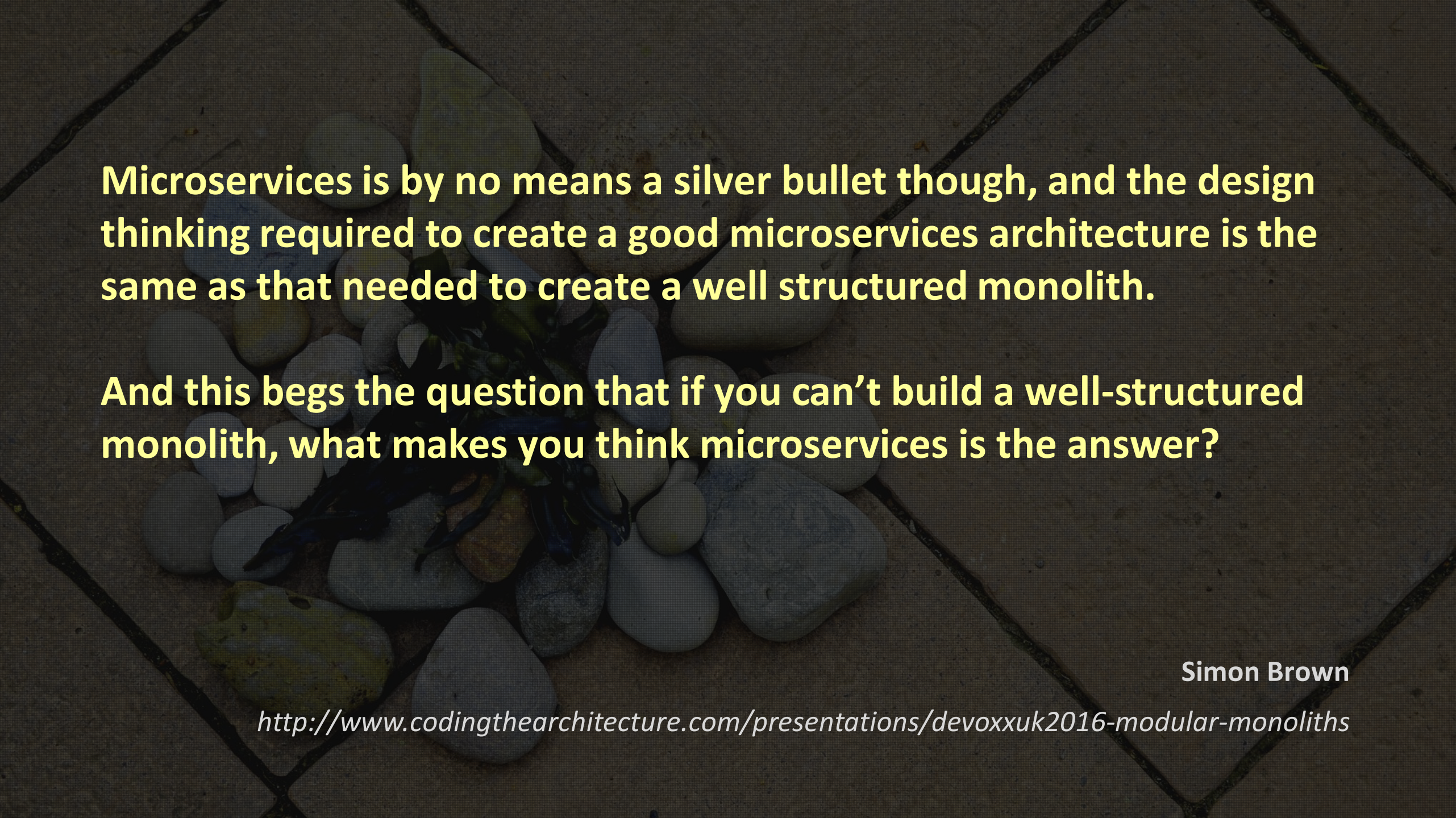
Microservices support shifting application logic along the continuum



A modular architecture of independently deployable units enables shifting logic along the continuum

Virtualization and containerization help achieving the desired deployment flexibility. Vision: build once, deploy anywhere

Yet flexible app deployability is often limited by different run-time platforms and architectures from edge to cloud



Microservices is by no means a silver bullet though, and the design thinking required to create a good microservices architecture is the same as that needed to create a well structured monolith.

And this begs the question that if you can't build a well-structured monolith, what makes you think microservices is the answer?

Simon Brown

http://www.codingthearchitecture.com/presentations/devoxxuk2016-modular-monoliths

Base-Line Architecture

Resilience through asynchronous communication and event streaming

If you are synchronous, you are not resilient

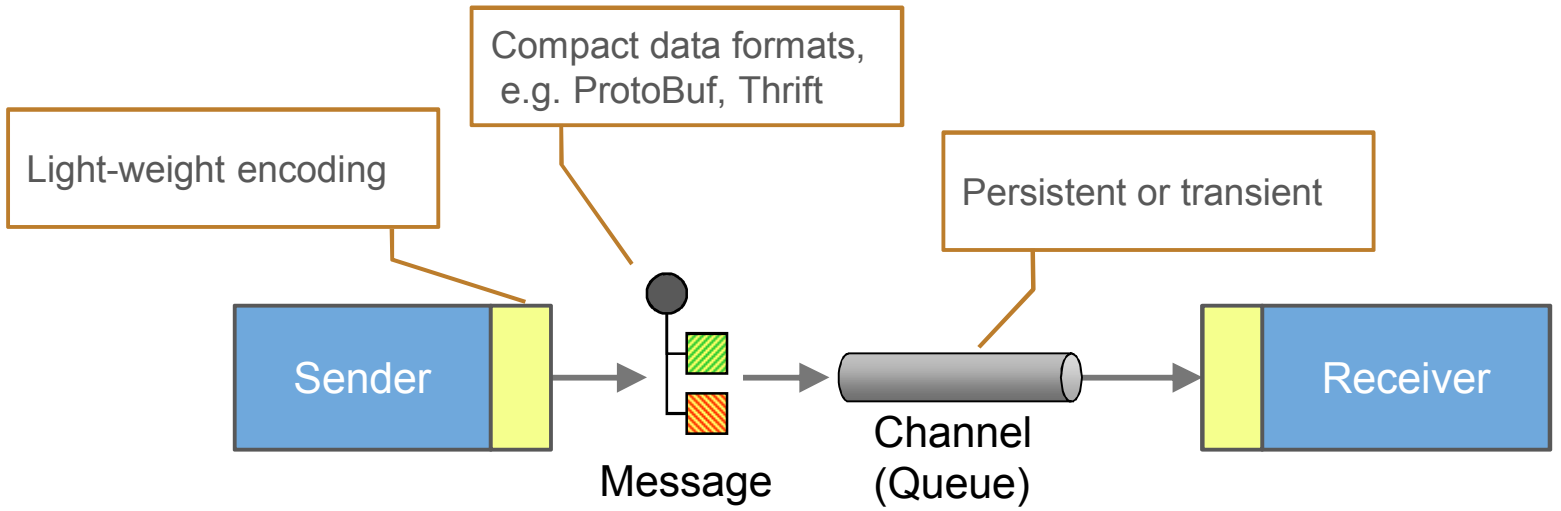
Minimize coupling and dependencies to increase resilience and autonomy.

Communication is slow and unreliable. Never block on remote communication.

Beware of “retry storms” – simple error handling

Base-Line Architecture

Asynchronous Communication architecture style

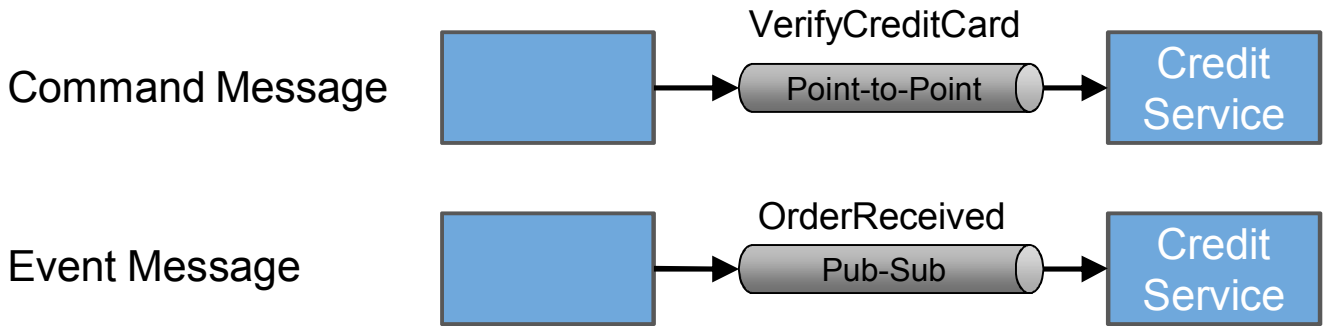


Asynchronous communication acknowledges the limitations of the underlying medium.	
Systems send messages across Channels	Simplified interaction
Channels have logical addresses	Location Decoupling
Placing a message into the Channel is quick (“fire-and-forget”). The Channel queues messages until the receiving application is ready	Temporal Decoupling

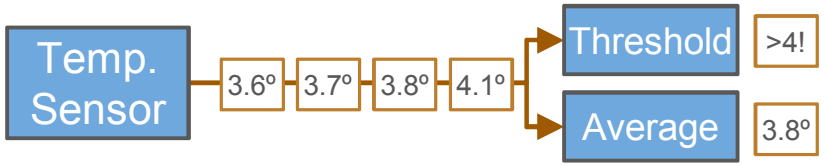
Base-Line Architecture

Eventing and event streaming architecture style

Events shift logic towards the receiver. More loosely coupled.



Event Streaming



Canonical model for sensor data.

Events are often time stamped for analysis in a Time Series Database.

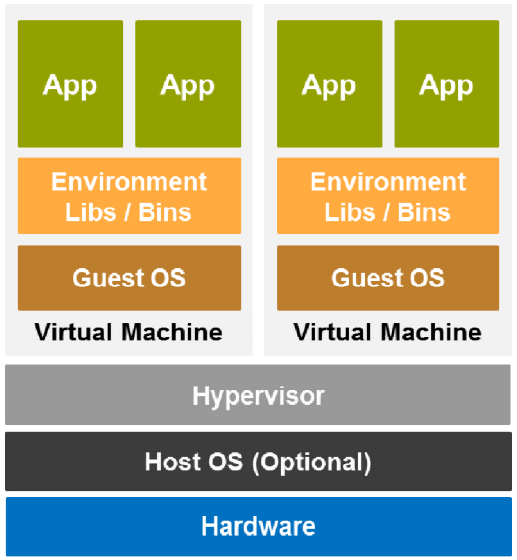
Often OK to drop events. Alleviates need for retries.

New events supersede old events.

Patterns similar to messaging, often augmented by temporal patterns.

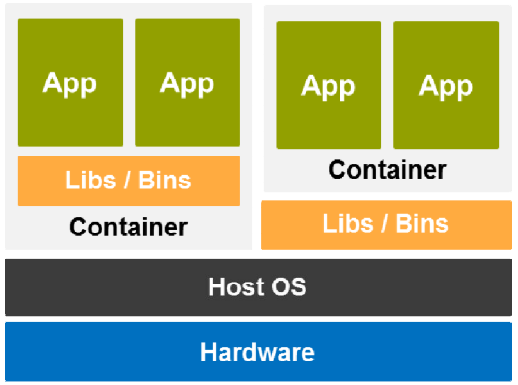
Base-Line Architecture

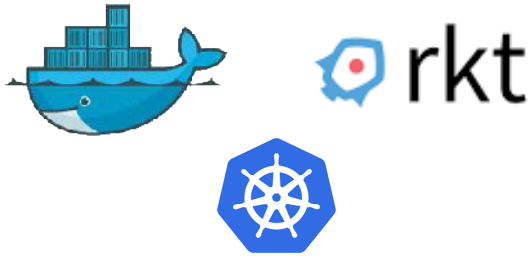
Virtualization and containerization – two approaches for app delivery



Suitable for all system architectures
Supports legacy applications
Security (hypervisor as barrier)
Off-the-shelf technologies available

Heavyweight (multiple OS)
Limited deployment flexibility (Host OS dependencies)



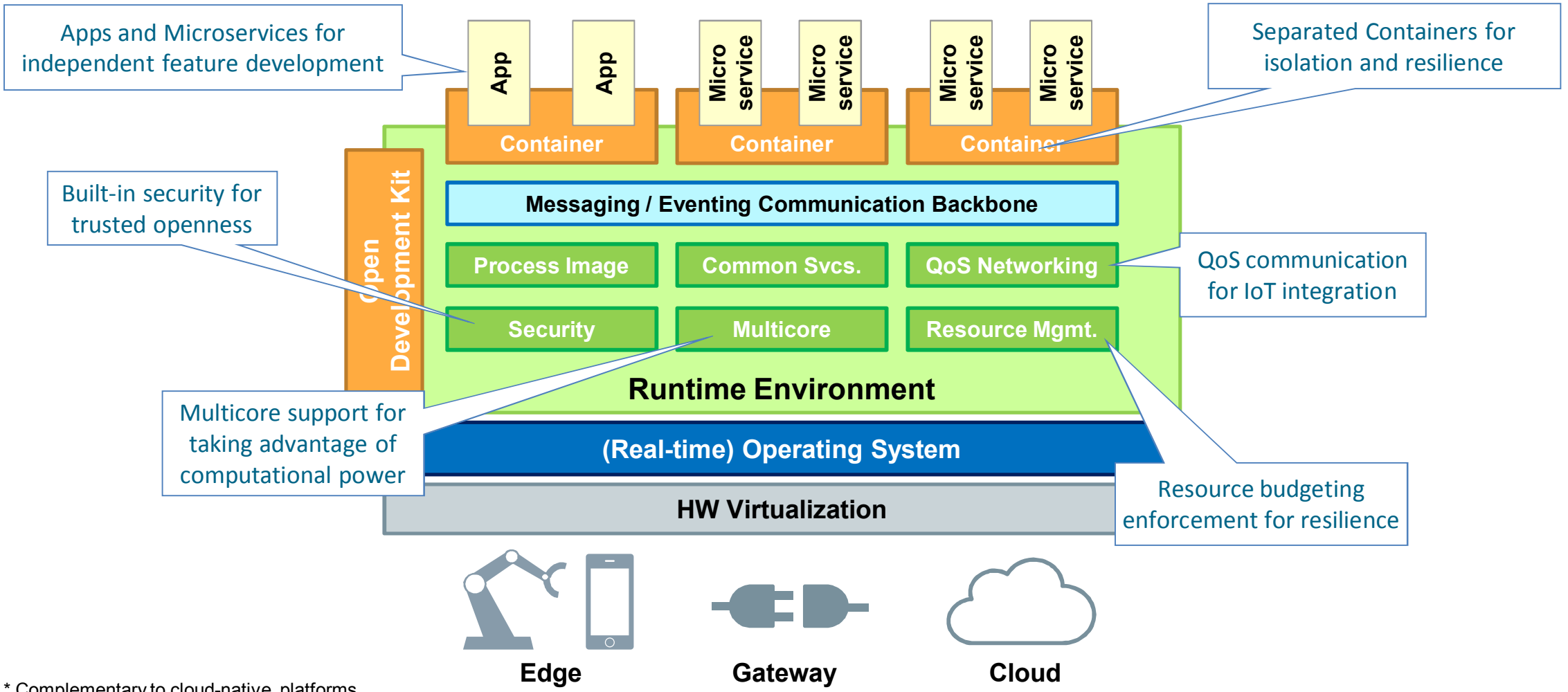


Suitable for apps and microservices
Lightweight
High deployment flexibility, DevOps friendly
Off-the-shelf technologies available

Security (OS vulnerabilities)
Limited legacy support

Base-Line Architecture

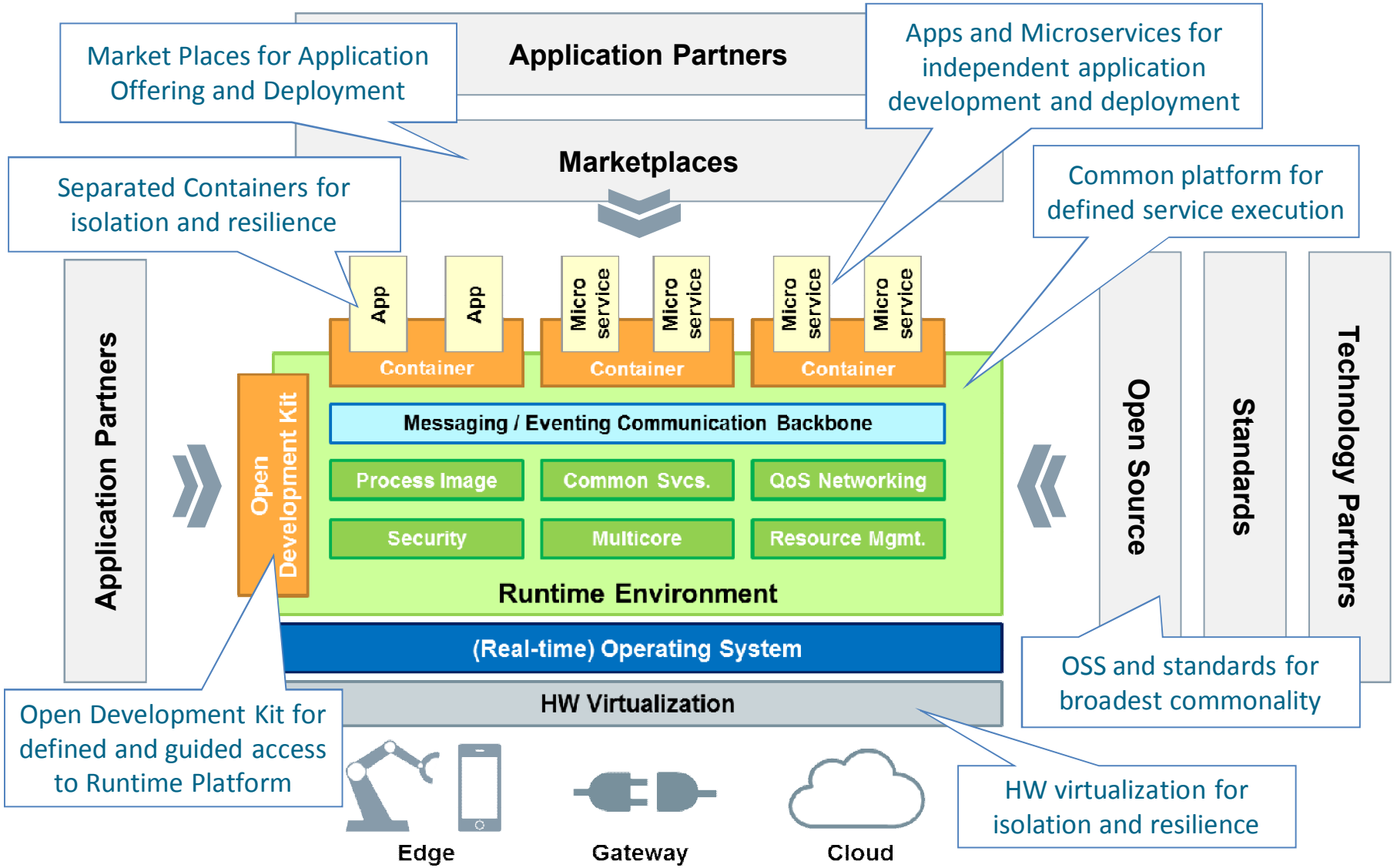
*Industrial-grade, flexibly deployable IoT runtime for the cloud to edge continuum**



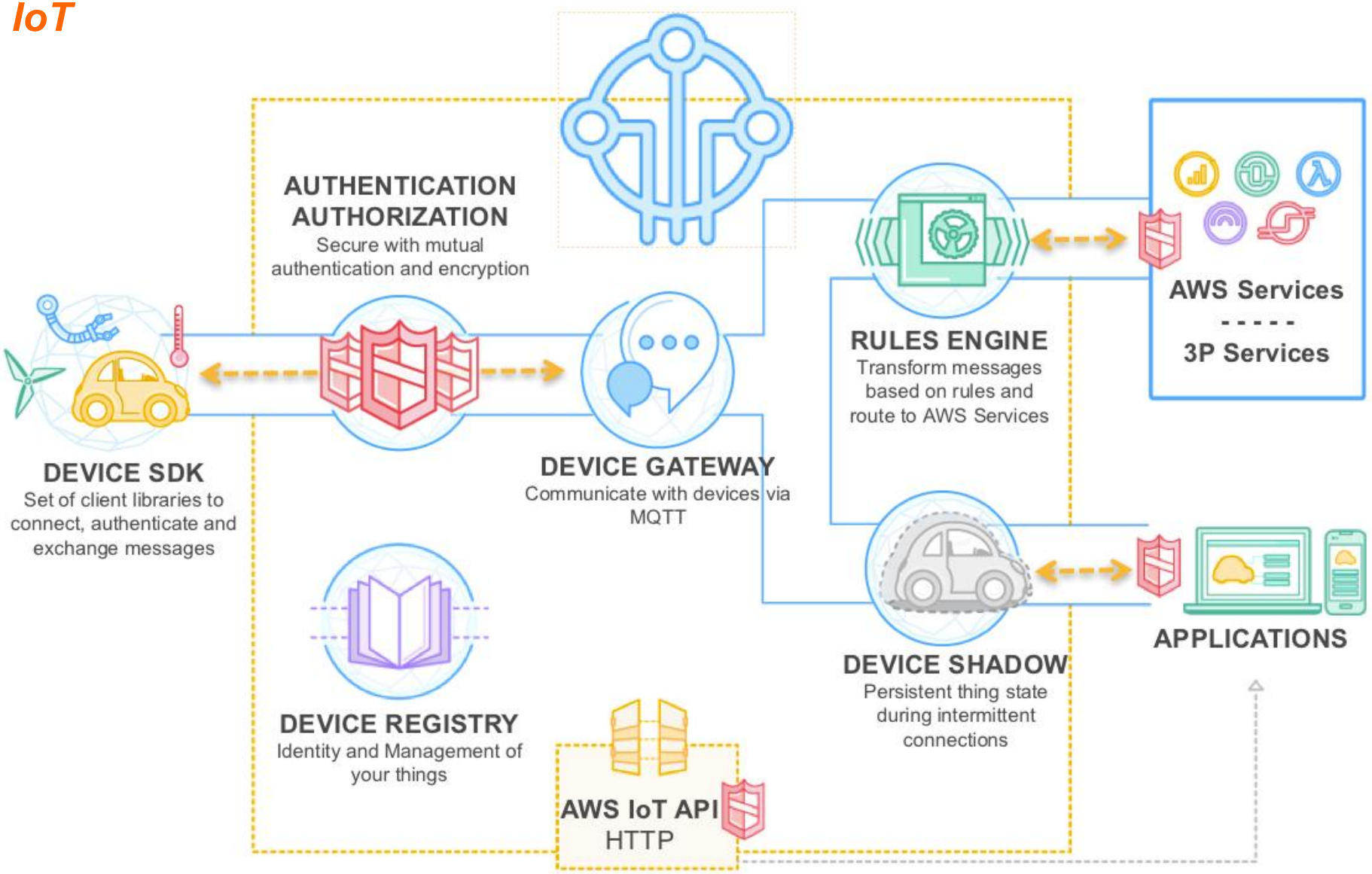
* Complementary to cloud-native platforms

Base-Line Architecture

IoT runtimes are ecosystem and DevOps enabled



IoT in a box
Amazon IoT



Cloud

Lambda Architecture

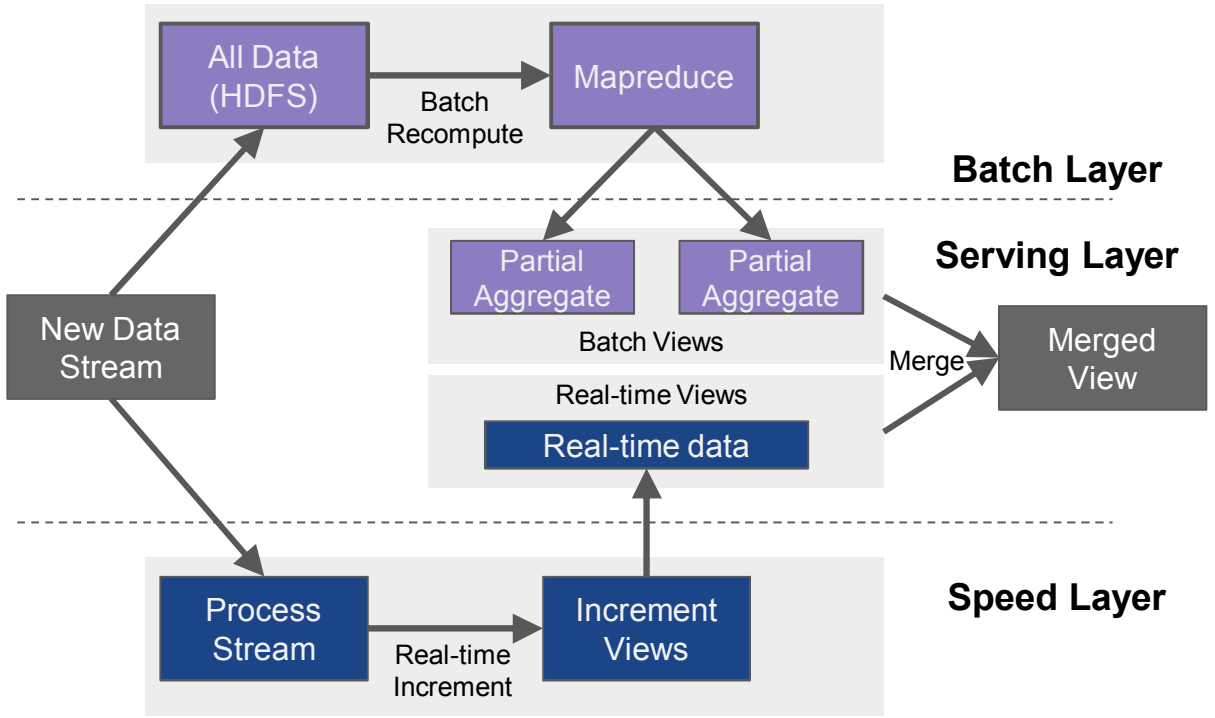
Batch Data Processing

Real-time Data Processing

Photo by imagesthai.com from Pexels

Cloud

The Lambda Architecture for big data processing and analytics applications*



Precomputes results via distributed processing of big data. Aims at accuracy.

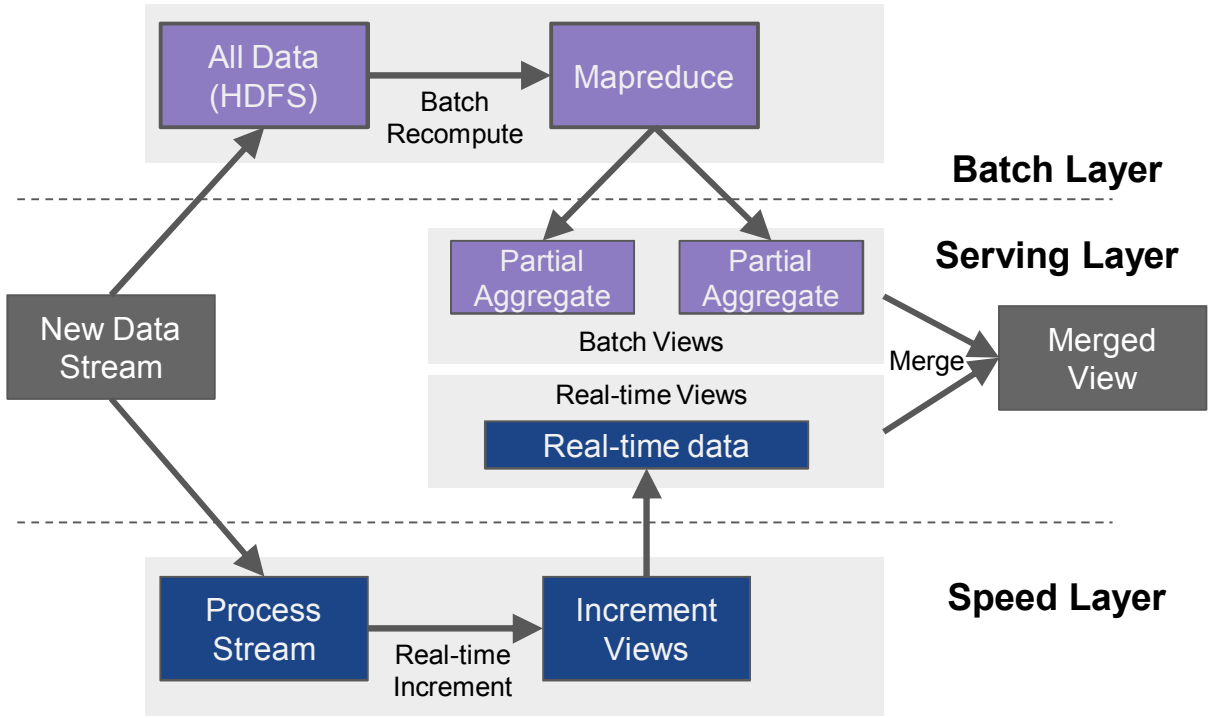
Stores output of batch and speed layer. Responds to user queries.

Processes data in real-time. Sacrifices throughput to achieve minimal latency.

*Originally designed by Nathan Marz for Twitter; now adopted by many companies,

Cloud

Multiple architecture styles define the Lambda architecture



Map / Reduce

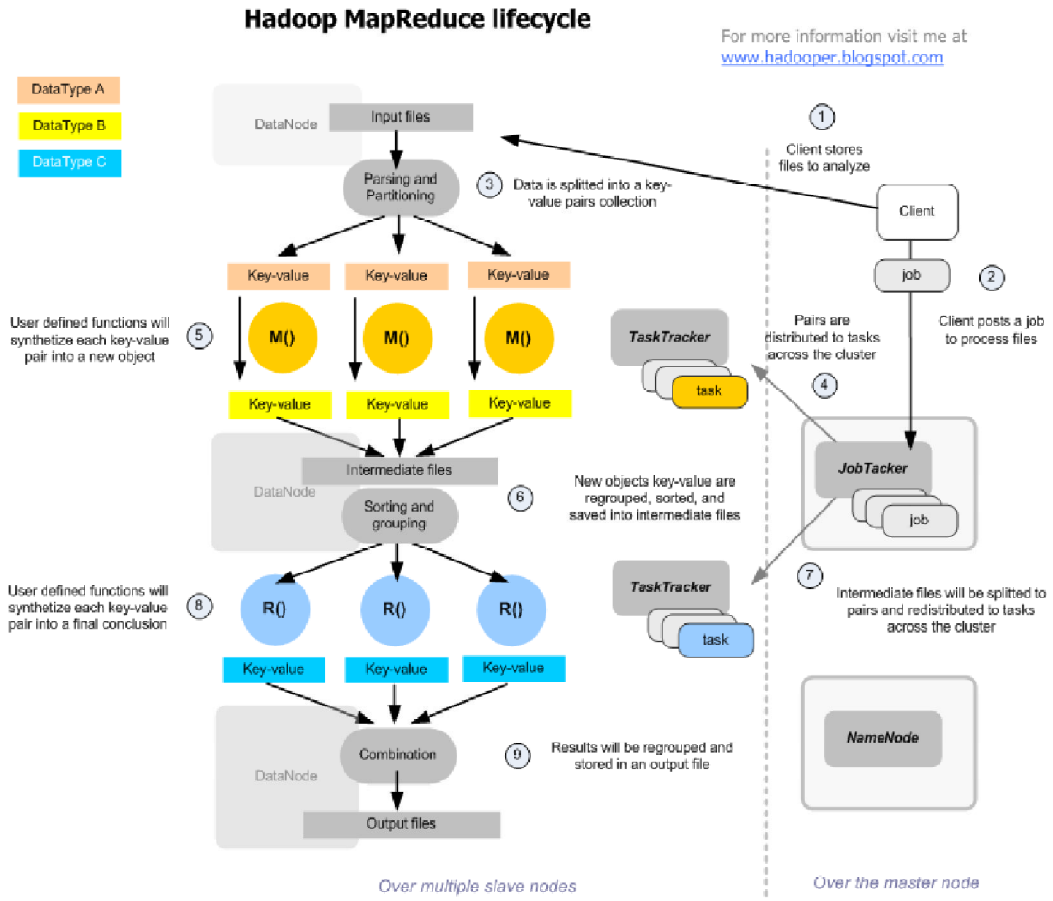
Publich/Subscribe Observer

Command Processor Command

Parallel Pipes and Filters

Cloud

Batch layer with Map / Reduce design (with Hadoop)



Structure and Behavior

Data nodes manage data items and files regarding parsing, partitioning, sorting, grouping data

User functions create intermediate results from raw data and final results from intermediate results

Multiple Map / Reduce steps can be chained

Scalability and Performance

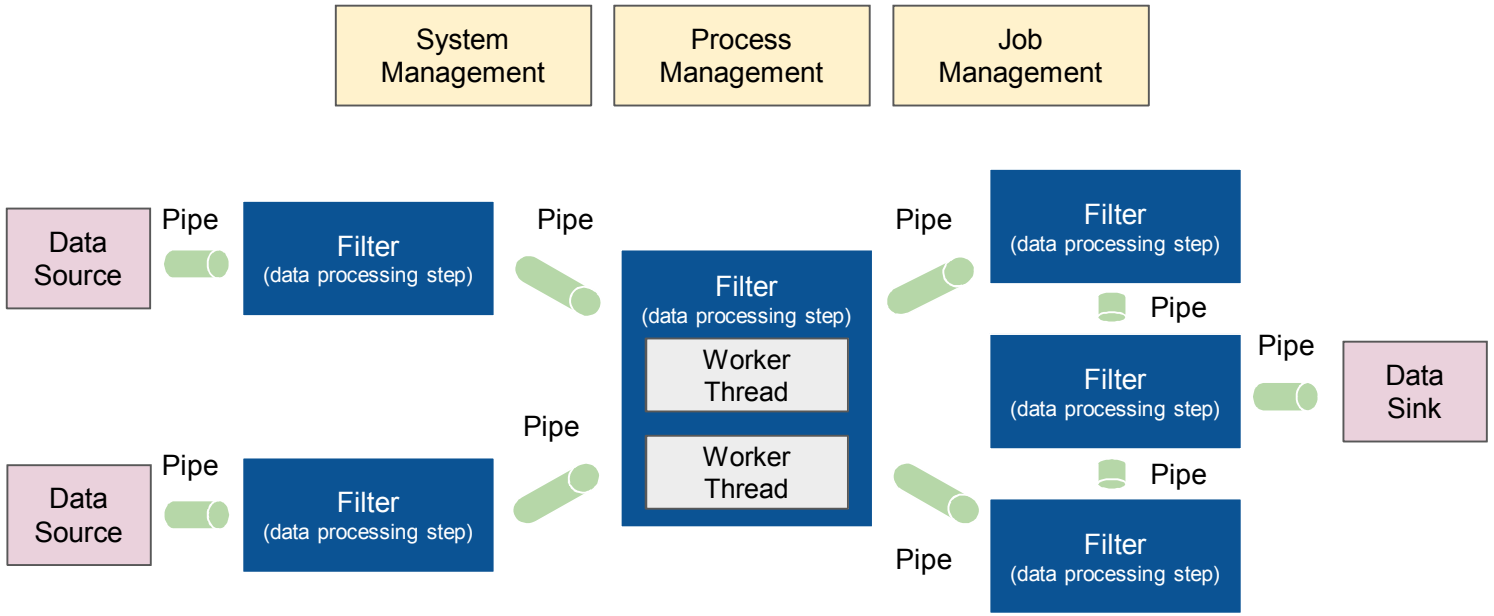
Data management steps run on separate nodes

User functions run on multiple nodes, processors, cores, threads

Data completeness ensures maximum accuracy of results

Cloud

Speed layer with Parallel Pipes and Filters design



Structure and Behavior

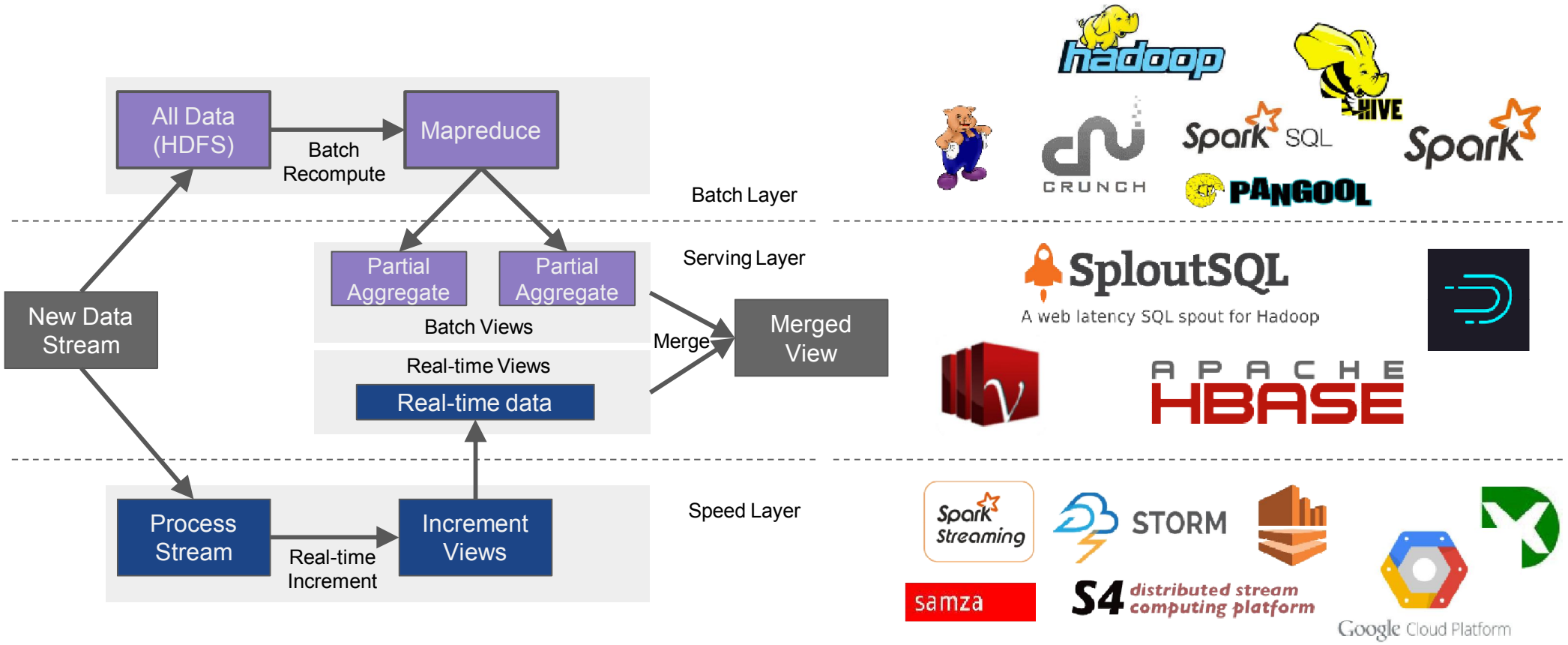
- Pipes transport data streams
- Filters perform data processing steps
- The Pipes and filters network defines the data processing job
- Forks and joins are possible
- Filters start as soon as they receive data

Scalability and Performance

- Filters can run on different cores, processors or nodes
- Filters can consist of multiple threads, each of which runs on its own core
- Multiple filters with the same processing task can run in parallel

Cloud

Many off-the-shelf and open source realization technologies available

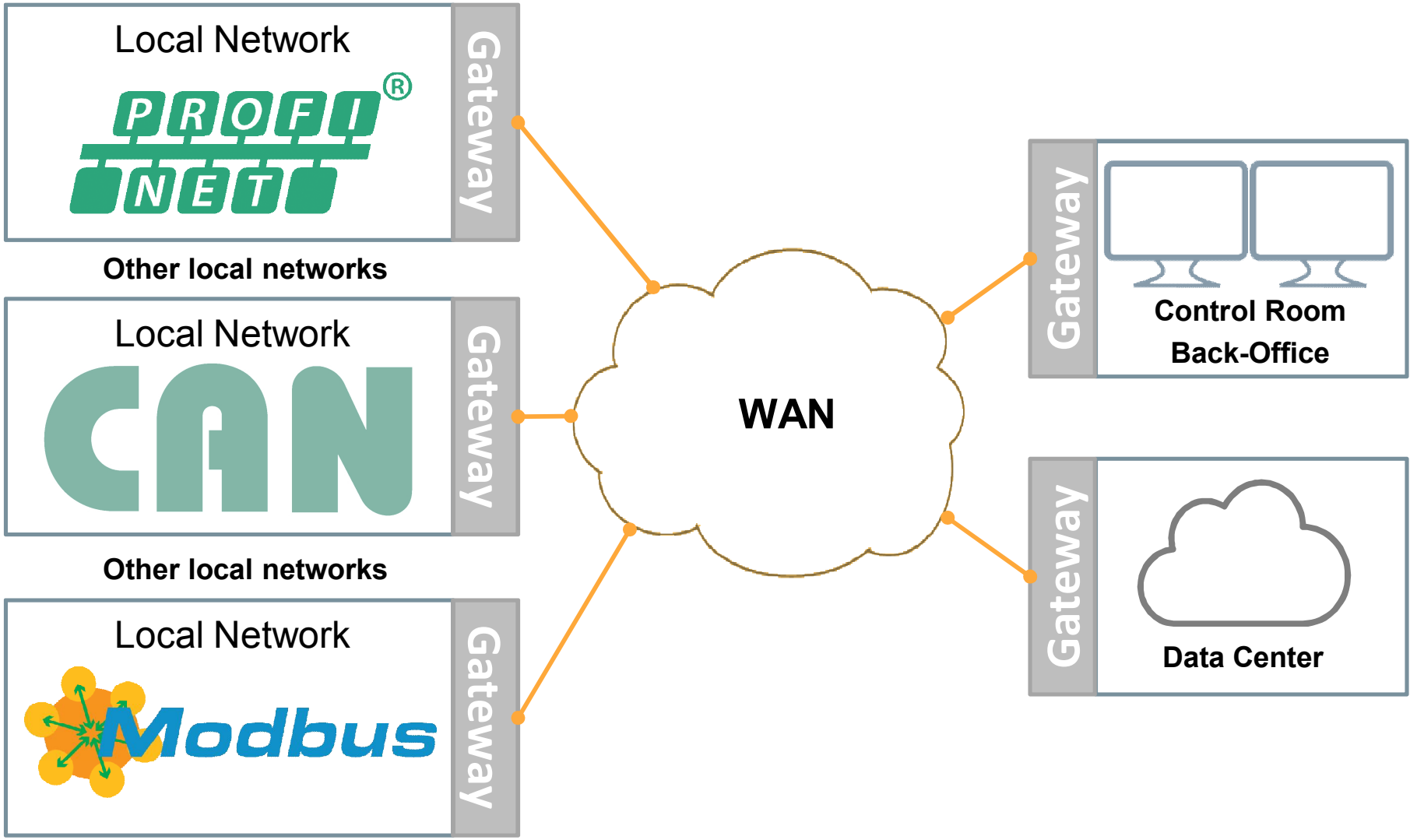


Gateway Architecture

„STL Skyline 2007 crop (Gateway Arch)“ by Buphoff - http://commons.wikimedia.org/wiki/File:STL_Skyline_2007_edit.jpg. Licensed under CC BY-SA 3.0 via Wikimedia Commons

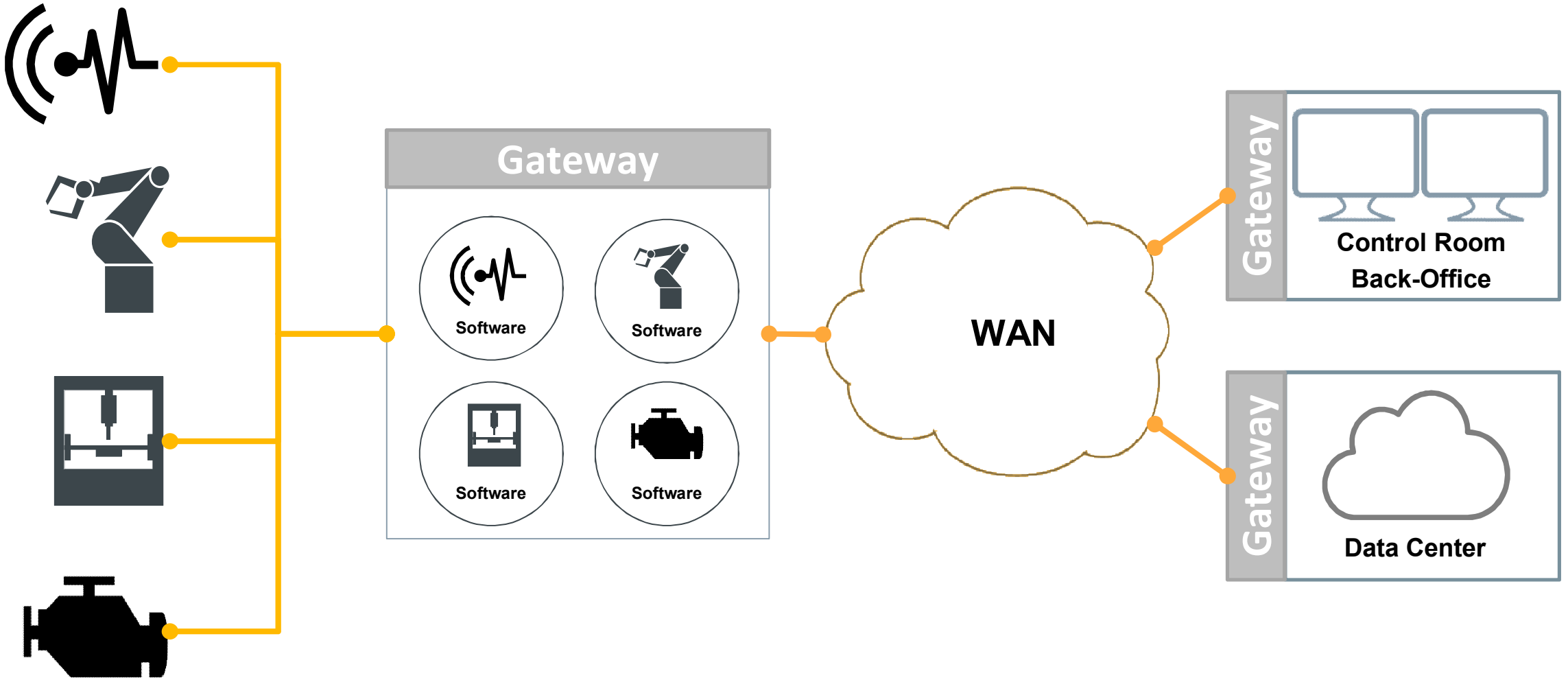
Gateway

Bridging of separated communication networks, including protocol adaptation



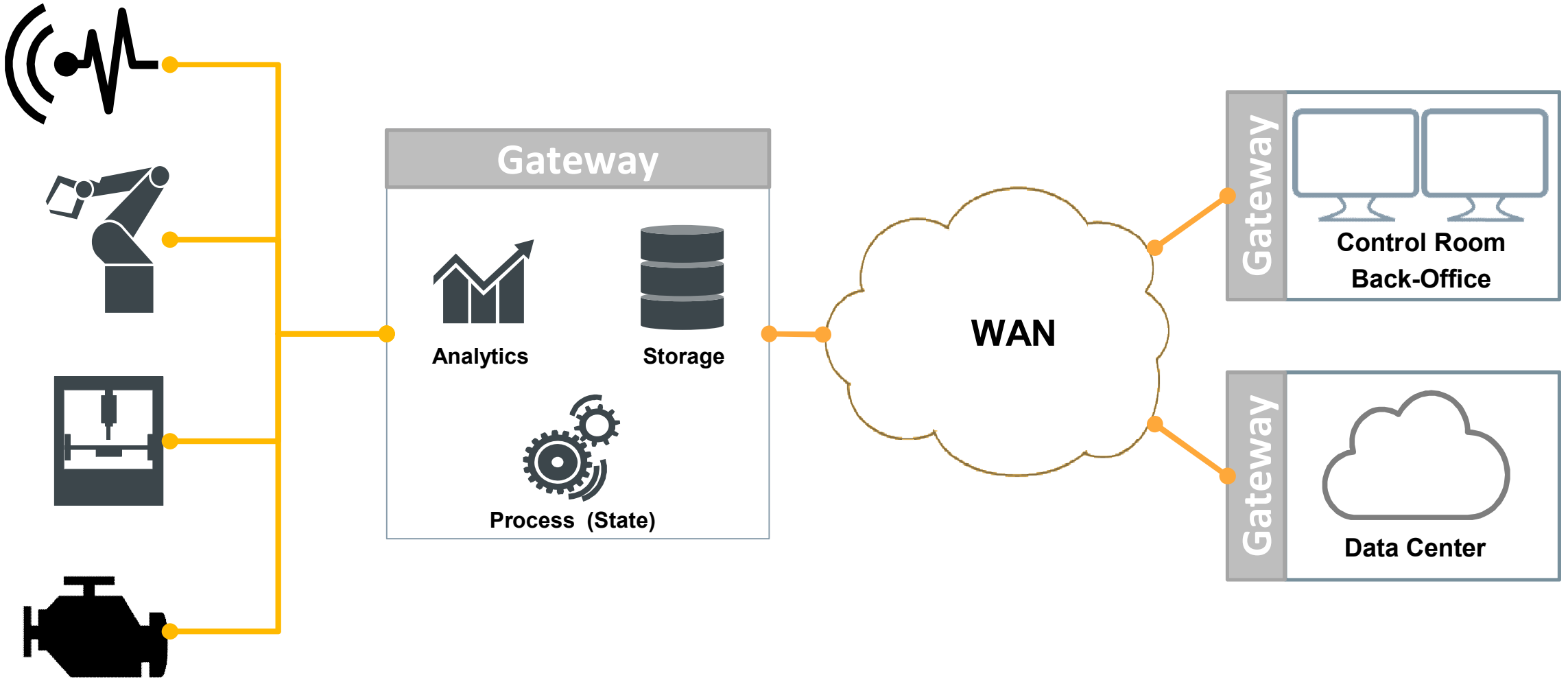
Gateway

Acting as a proxy to connect dumb things to the IoT, making them intelligent



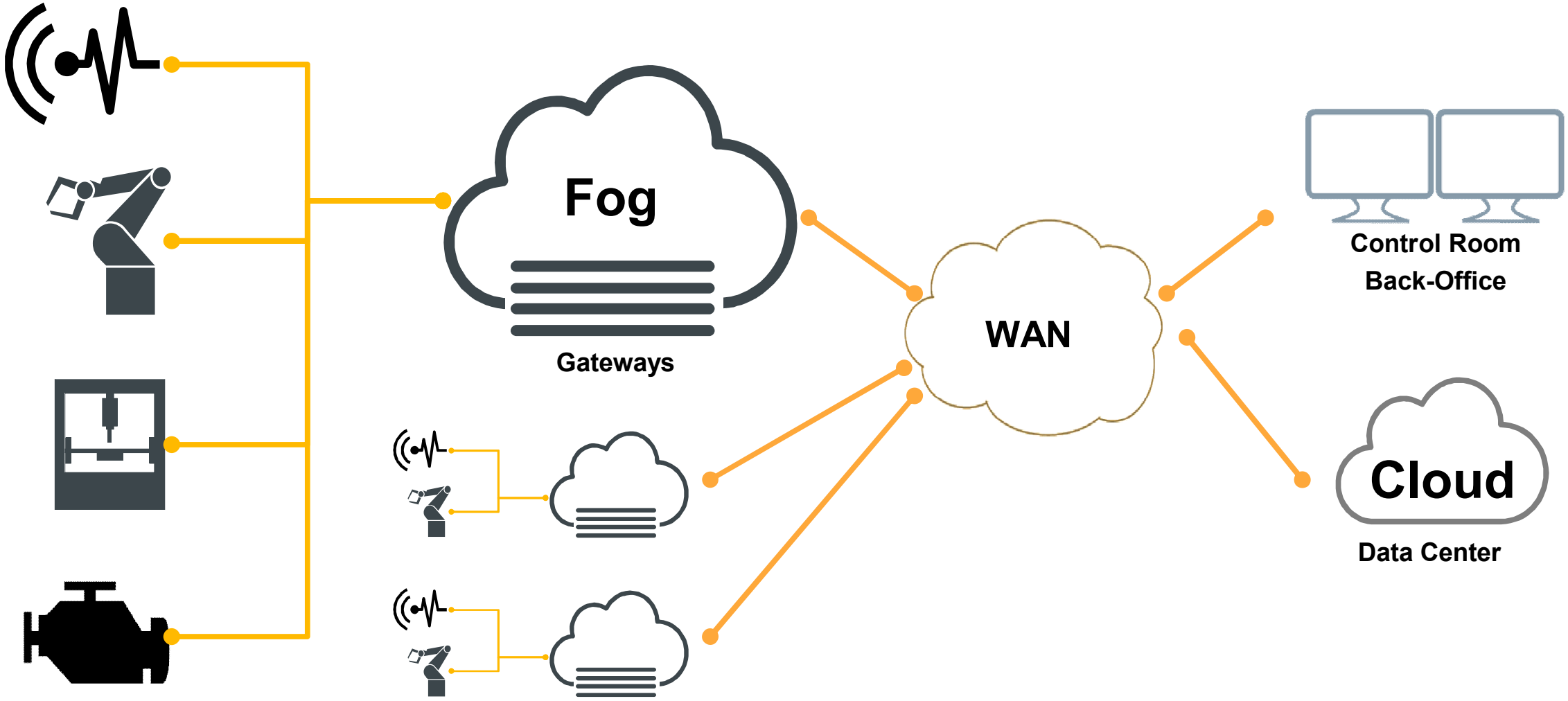
Gateway

Providing common services to things close to the field



Gateway

The natural place for fog computing



Edge

- Industrial vs. Consumer IoT
- Things of Things
- Fail Operational Things
- Security and Safety

Photo by Rhy Davies: Beachy Head, East Sussex

Edge
Industrial IoT versus consumer IoT

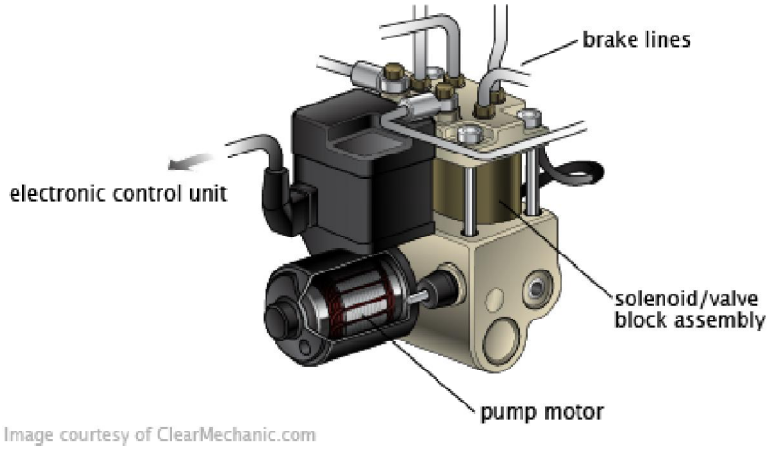


Image courtesy of ClearMechanic.com

Resilience

Safety

Speed / real-time response

Push processing into device (edge computing)

Price - minimize local processing

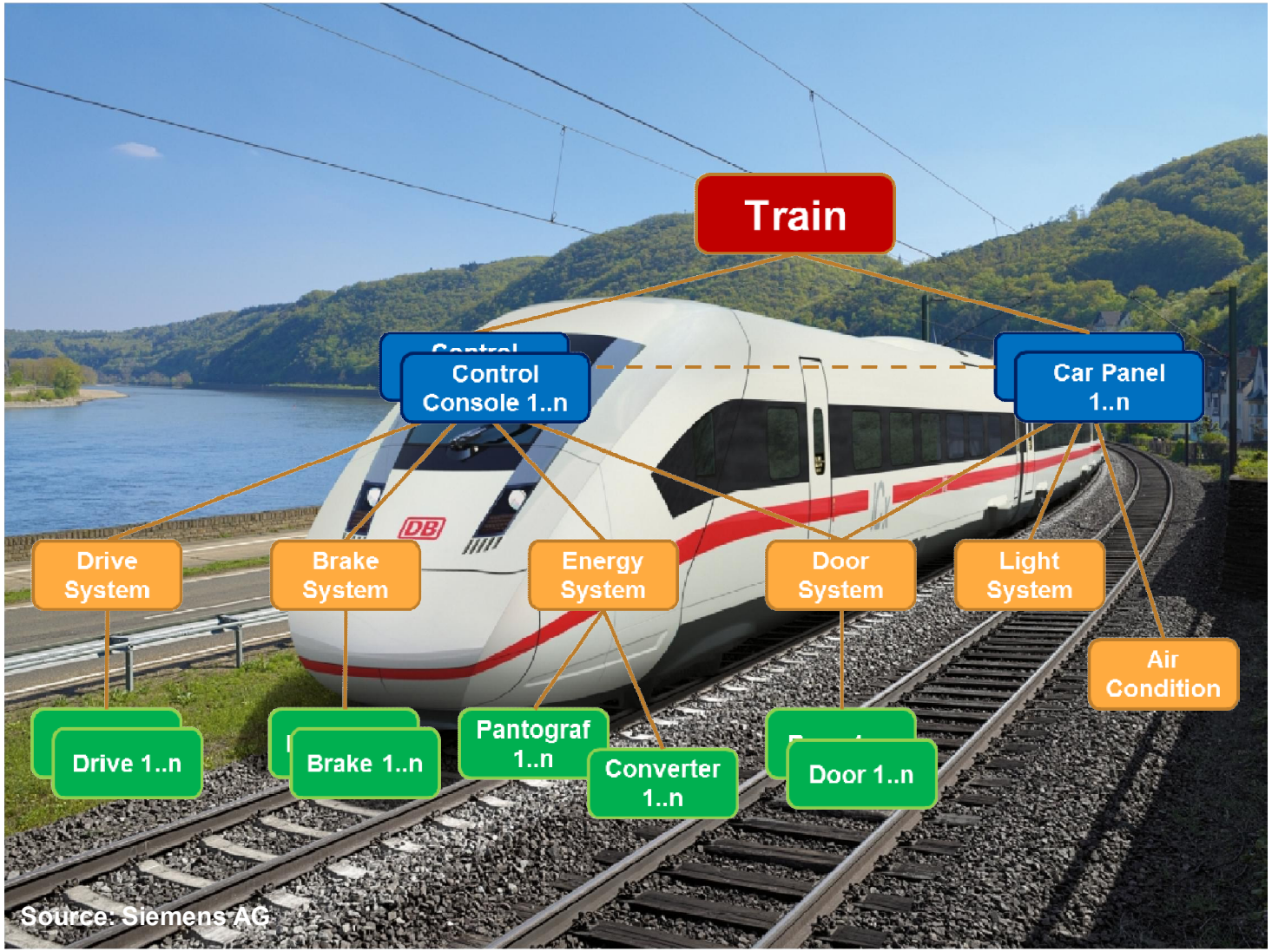
Push processing into smartphone

Design

Flexibility / Updates

Edge

Things of things enable scale and autonomy



Edge

From Failure Accepted to Fail Operational

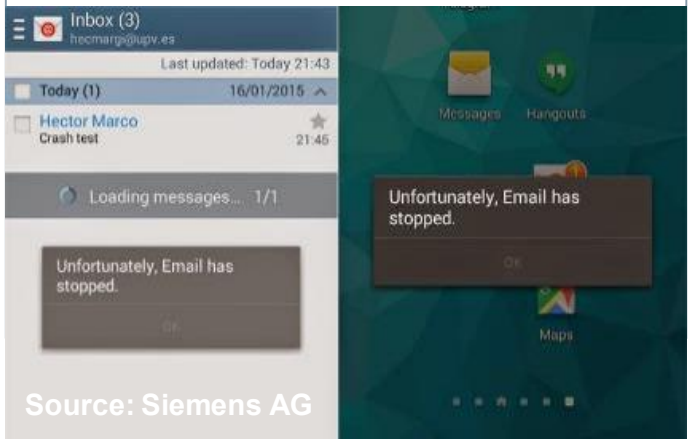
Edge

Failures must result in degraded functionality, not in dysfunctionality

Failure Accepted

On failure close apps and restart

- Accepted in consumer IT
- Doesn't work where life is endangered
- Will not work for IoT



Fail-safe

Stop on failure, enter safe state

- Common practice in industry, but burns money
- No op for process industry, avionics, power,
- Unusable on frequent incidents

Fail-operational

Continue (degraded) service delivery in case of failure:

- Failure of things is the norm
- Safe state would result in a (set wise) global shutdown

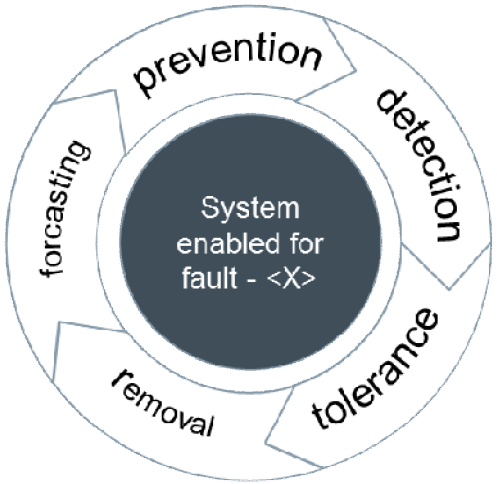
Edge

Fail Operational = Resilience + Robustness

System Resilience

To maximize availability of an IoT System and its constituent things

- When MTBF cannot be influenced, MTTR must be minimal
- When IoT parts fail other IoT parts must still work correctly



Robustness of Applications

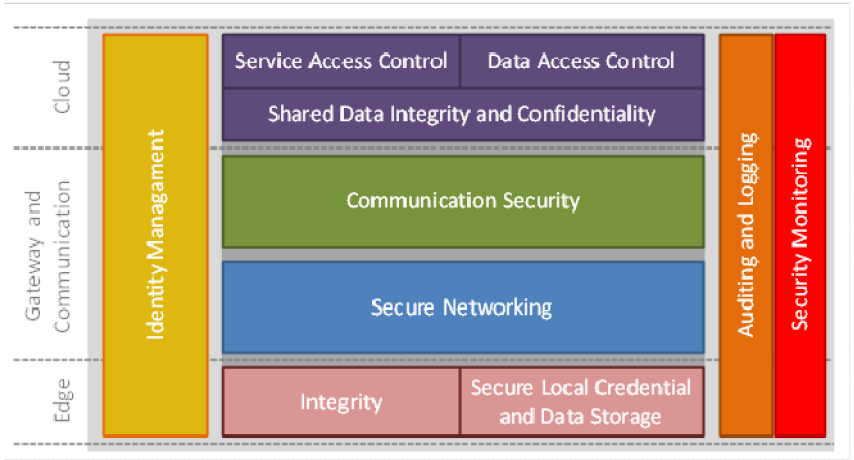
Concrete Applications must produce useful results in case of

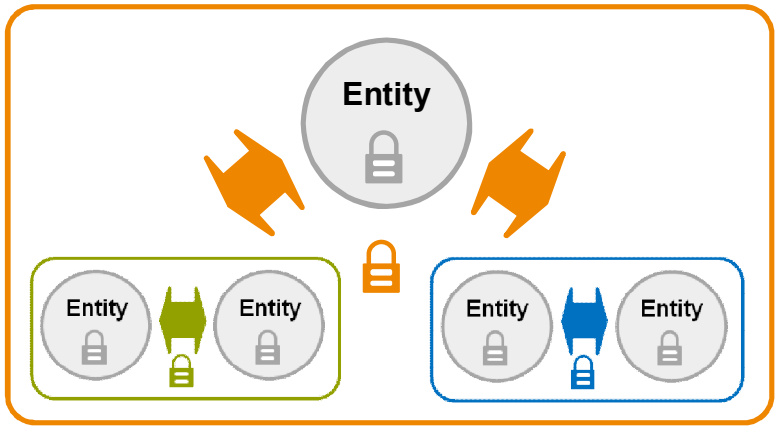
- Noise on input data, missing input data
- Loss of connectivity
- Resource degradation (time, memory. etc.)
- Failure of environment

Edge

Mix of known and novel security approaches are necessary to secure things

IoT key security challenges	IoT key security responses
Open system Highly dynamic – things join, leave, re-arrange continuously Never consistent Heterogeneity	All known security techs apply Recursive security architecture: IoT entities are realized as secure “atoms” w/ defined security interfaces & properties Composition of IoT entities results in an automatic determination of aggregate security properties

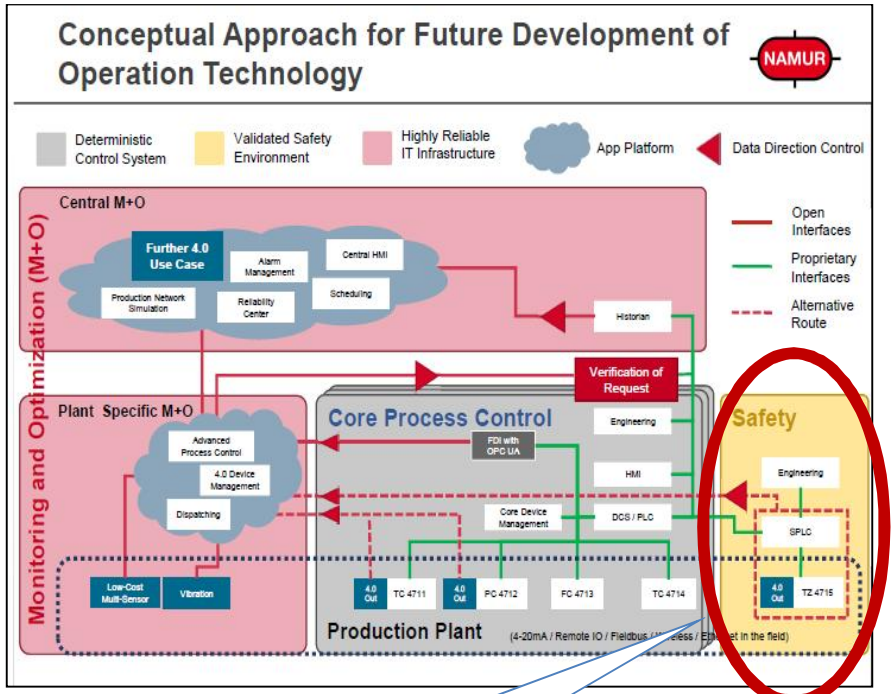




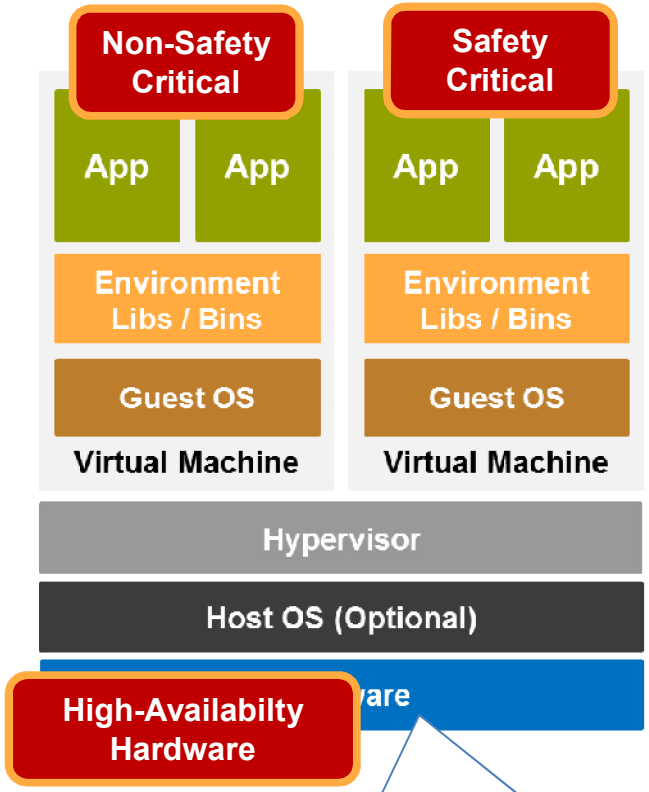
Edge

keep safety separate and make sure non-safety functions do not interfere

Distributed Control System (DCS) reference architecture proposed by NAMUR



Strict separation via separate hardware and no cloud connectivity



Mixed criticality deployment via virtualization or containerization

Network

Protocols

Topologies

Semantics

Management

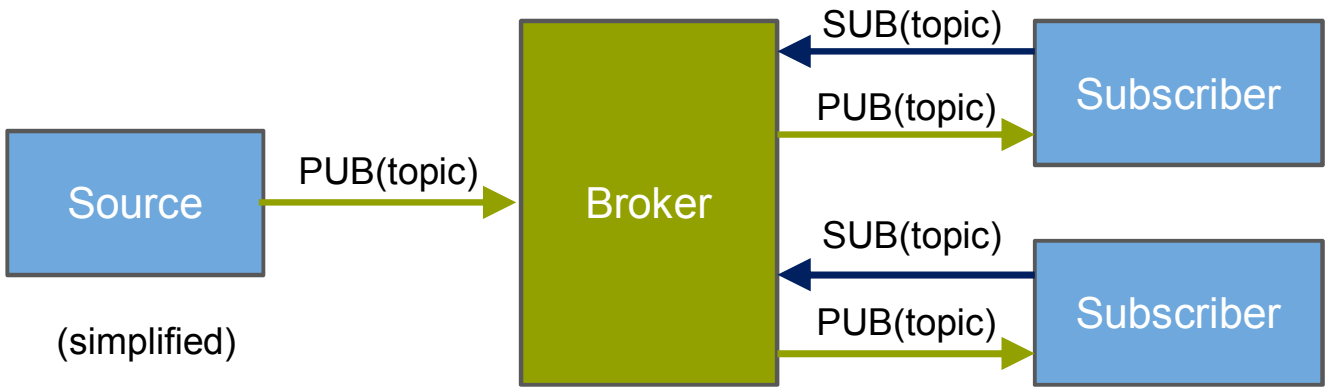
„Spinnennetz in Tannenspitze“ by Frank Liebig – Archiv, retired veterinarian. Frank Liebig. License under CC BY-SA 3.0 de via Wikimedia Commons

Network

MQTT: easy to use, wide spread

Broker-based, light-weight, session-oriented publish-subscribe protocol

- Wire Format: Fixed Header (2+ bytes), Variable Header, Payload (blob)
- Data encoding: e.g. variable length fields
- Verbs (commands): CONNECT, PUBLISH, SUBSCRIBE
- Topic hierarchy and wildcards: sport/tennis/player1/score/Wimbledon
- QoS Levels: At most once, at least once, exactly once
- Encryption and authentication



Alternative: **AMQP** – focus on large-scale reliability, flexibility, and security

Network

Time Sensitive Networking (TSN): quality of service

A set IEEE 802.1 standards to provide deterministic performance within standard Ethernet:

- Timing and Synchronization for Time-Sensitive Applications
- Frame Preemption
- Enhancements for Scheduled Traffic
- Path Control and Reservation
- Frame Replication and Elimination for Reliability (Seamless Redundancy)
- Stream Reservation Protocol Enhancements and Performance Improvements
- Cyclic Queuing and Forwarding
- Per-Stream Filtering and Policing
- Time-Sensitive Networking for Fronthaul

Network

Many other standard communication protocols

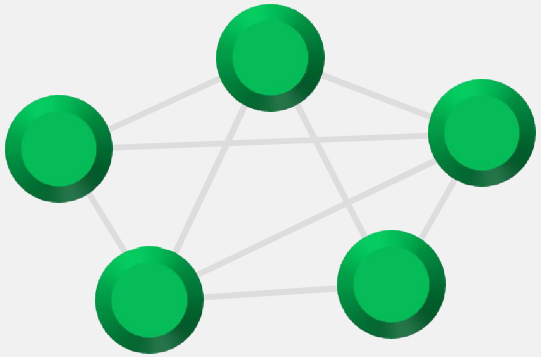
And many more

The nice thing about standards is that you have so many to choose from

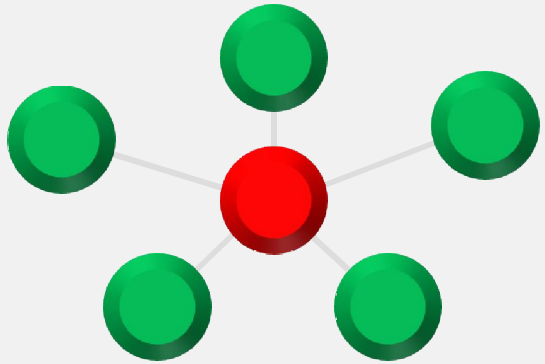
[Andrew S. Tanenbaum]

Network

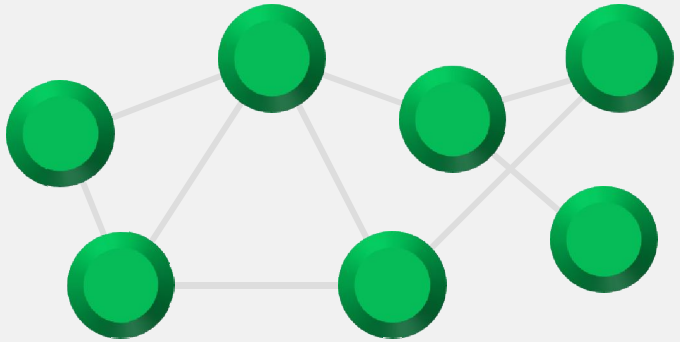
Different network topologies for different purposes



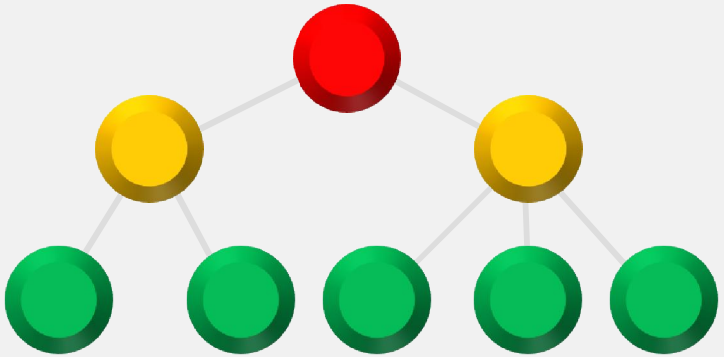
Point-to-point for
federated operations



Hub-and-Spoke for connecting
dumb things to smart hubs



Meshed for
task-oriented operation



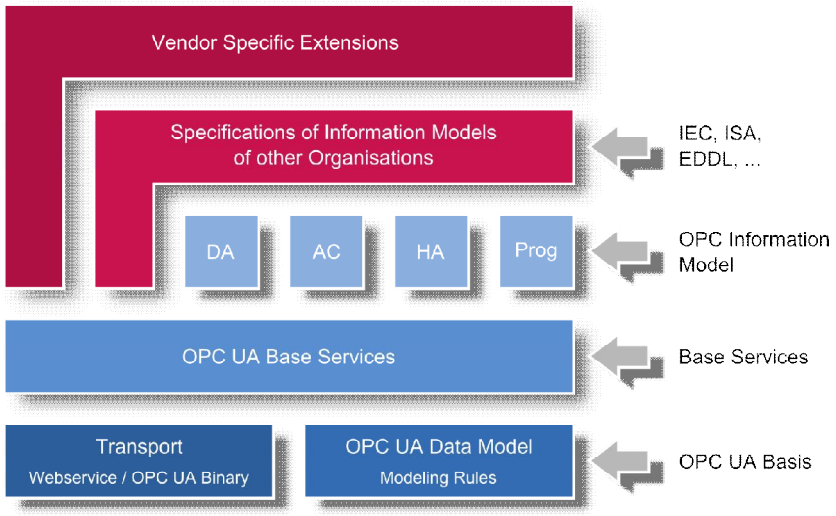
Hierarchical for
modular and scalable operation

Network

Semantics is essential for meaningful interaction between things

OPC UA is an industrial interoperability standard.

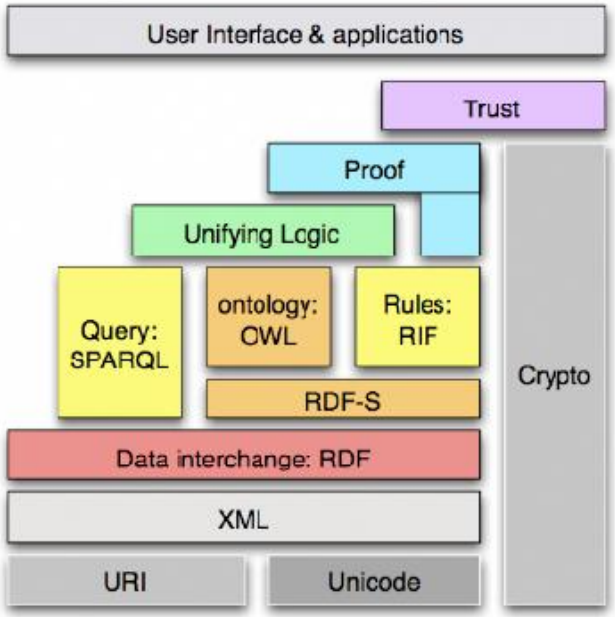
It allows to describe machine data in a computer-readable form enriched with semantics



The **Web Ontology Language** (**OWL**) is a family of knowledge representation languages

They are characterized by formal semantics.

They are built upon the Resource Description Framework standard



Network

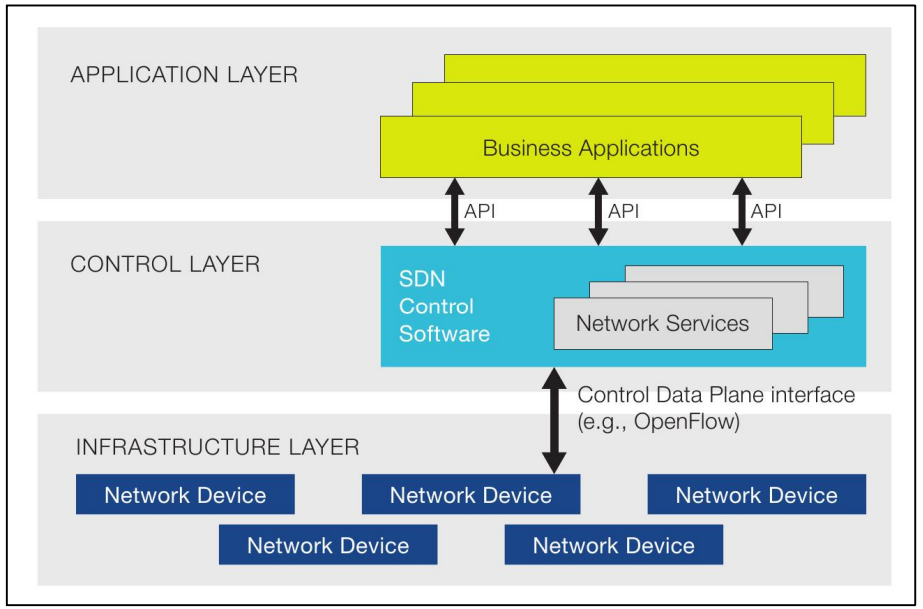
Software Defined Networking (SDN) automates network engineering and management

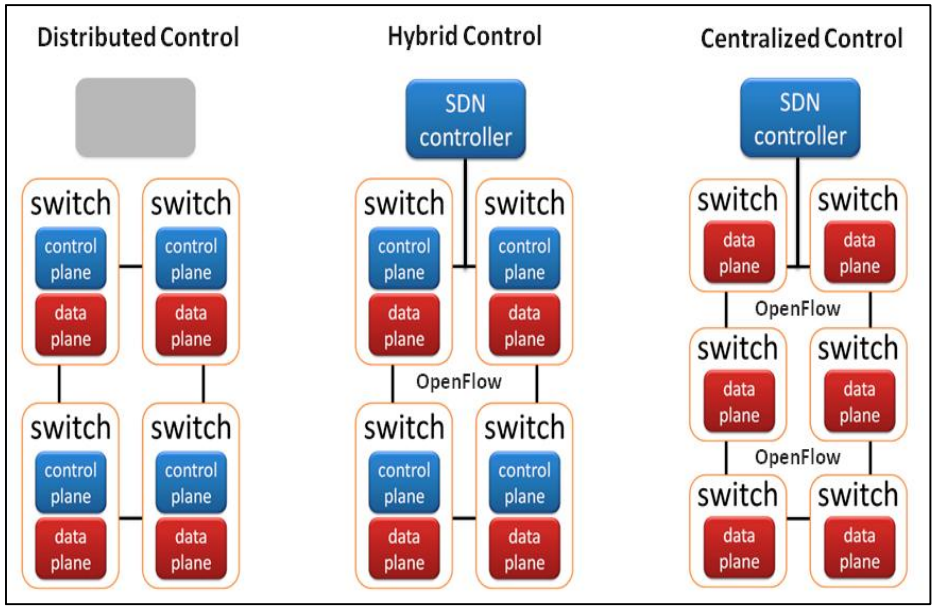
Network control and data planes are explicitly separated

A dedicated network control software (SDN controller) is directly (software) programmable with defined network management policies

The SDN Controller autonomously configures the physical network for each requested communication service by applications

Multiple deployments possible





Source: Open Networking Foundation

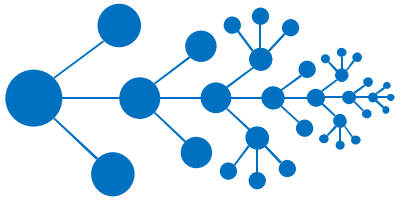
The Architect

29th scene from Matrix Reloaded

Screenshot from the film “Matrix Reloaded”, 2003, Warne Bros Ltd.

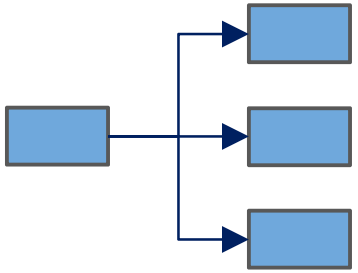
Architectural Thinking for IoT Systems

Ultra-large Scale



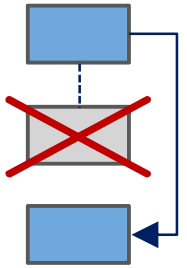
You are not in control

Distributed



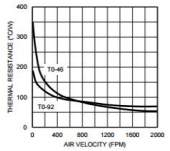
Concurrency, messaging, events

Resilience



Things will always be broken

Full Stack

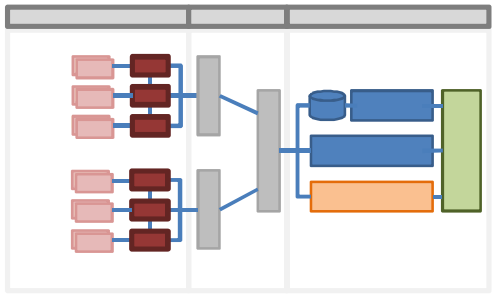


Bit	7	6	5	4	3	2	1	0
byte 1	MQTT Control Packet type				Flags specific to each MQTT Control Packet type			
byte 2...	Remaining Length							

```
ldi r16, 0x01
out TCNT1H, r16
```

Sensors, protocols, processors

End-to-end



Edge to cloud

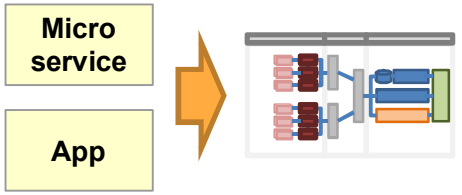
Beyond Tech

$$$

Privacy, Liability

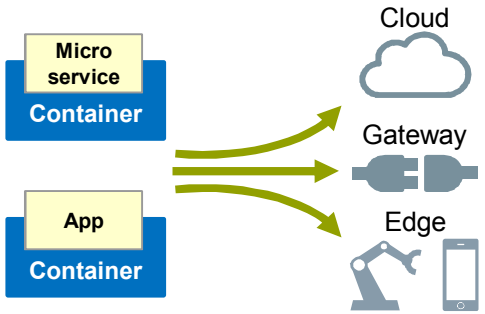
Architectural Thinking for IoT Systems

Apps / Services as Deployment Unit



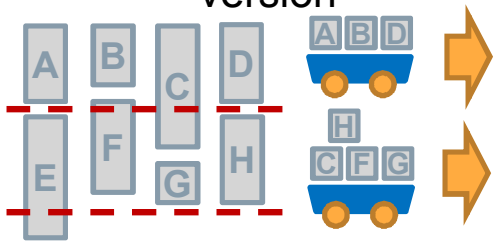
Evolution and Maintenance by App/Service replacement

Deployment Flexibility



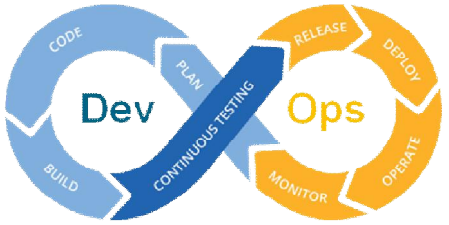
Containerization / Virtualization

No dedicated system version



Continuous Deployment / Release trains

Feedback Loop to Dev



DevOps
Knowledge on real usage

Architectural Thinking for IoT Systems

The true challenges in IoT are cultural

The courage to give up control
over the Internet of Things!

The understanding that we are not users,
but part of the Internet of Things.

Our responsibility to make the Internet
of Things a good place to live!!

Frank Buschmann
“Quo Vadis Software Architecture”, OOP 2011

Technology advances helped distribute the future more evenly

Tipping points in hardware and software