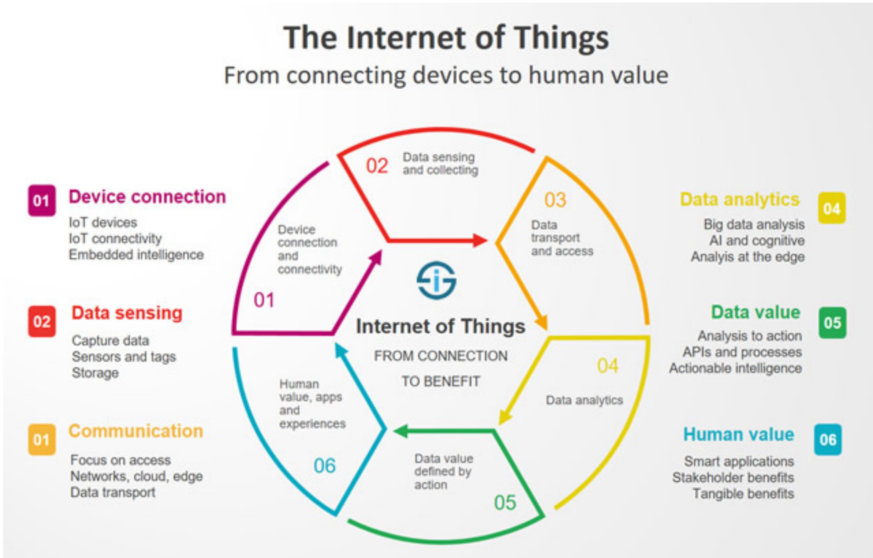
Internet of Things

Introduction

Ioannis Chatzigiannakis

Sapienza University of Rome
Department of Computer, Control, and Management Engineering (DIAG)

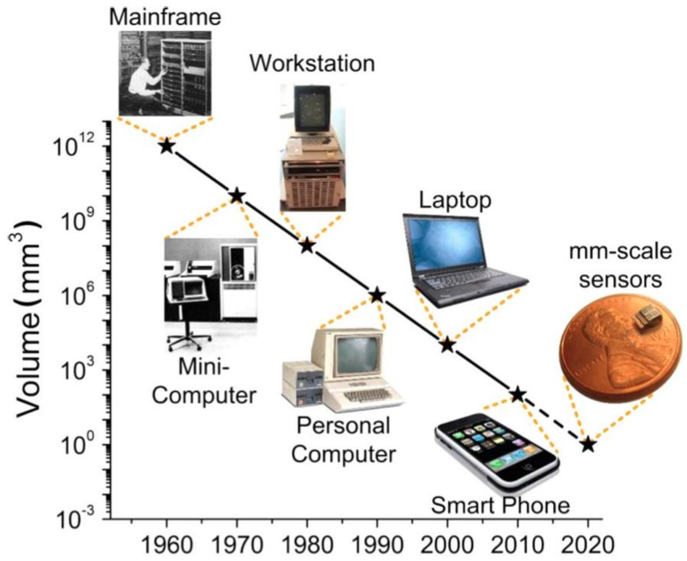
Lecture 2:
Application areas and Use cases, Networking Technologies, Data processing architectures, Opportunities and Challenges



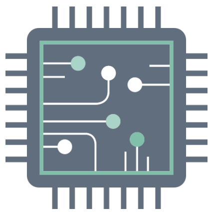
1. What kind of Sensors are available?
2. What kind of Data are we collecting?
3. How often do we need to collect Data?
4. How are Devices connected?
5. What types of Smart Services can we offer?
6. Where is Data processed?
7. What kind of collective intelligence do you expect will emerge?

1. What kind of Sensors are available?
2. What kind of Data are we collecting?
3. How often do we need to collect Data?
4. How are Devices connected?
5. What types of Smart Services can we offer?
6. Where is Data processed?
7. What kind of collective intelligence do you expect will emerge?

From Vacuum Tubes . . . millimeter-scale SoC



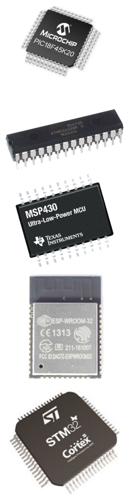
The rise of the IoT semiconductor



Semiconductor components that either individually or collaboratively contribute to the functionality of an IoT device.

- Microcontrollers (MCU),
- Microelectromechanical Systems (MEMS),
  - Sensors,
  - Actuators,
  - Energy harvesting.
- Connectivity chipsets,
- Embedded AI chipsets,
- Security chipsets.

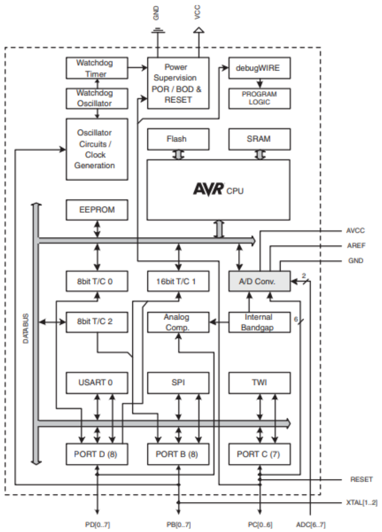
IoT Microcontrollers



A microcontroller (MCU) is a small single integrated circuit and it contains a CPU, RAM, ROM, I/O and timers.

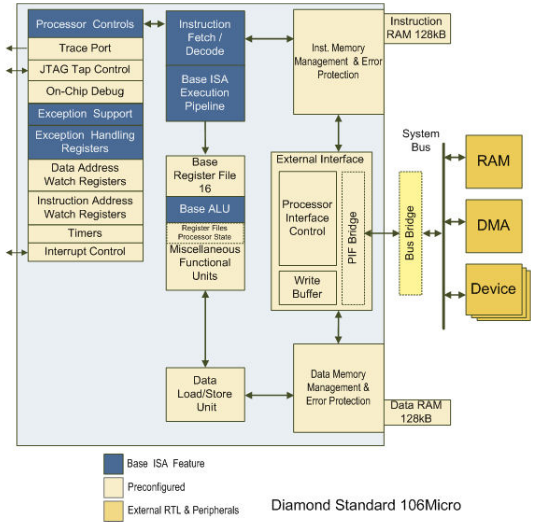
- 8-bit
  - PIC from Microchip
  - AVR architecture (e.g., ATMEGA328P)
- 16-bit
  - MSP430 by Texas Instruments
- 32-bit
  - Espressif ESP8266, ESP32
  - PIC32
  - ARM Cortex M0/M3/M4
  - . . .

AVR Architecture (e.g, ATMEGA328P)



- Originally made by Atmel which was then acquired by Microchip in early 2016
- Used in Arduino dev boards
- 8bit RISC

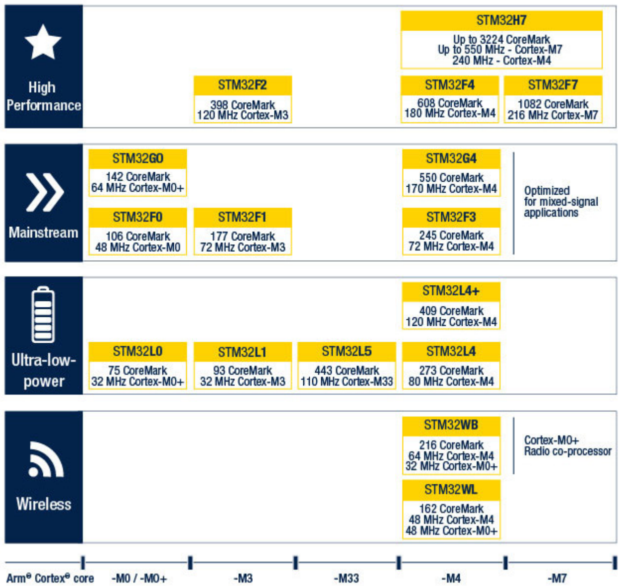
Tensilica L106 MCU (e.g., ESP8266)



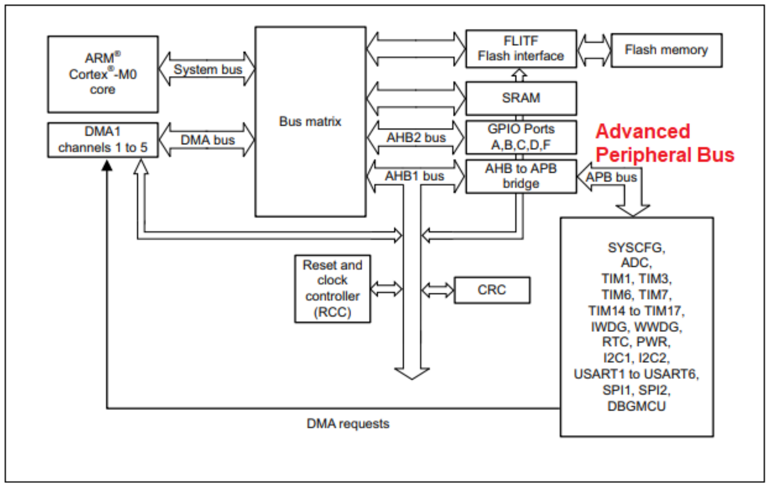
Diamond Standard 106Micro

- Cache-less controller.
- Employs a 5-stage pipeline.
- Modelessly switches between 24- and 16-bit narrow instructions.

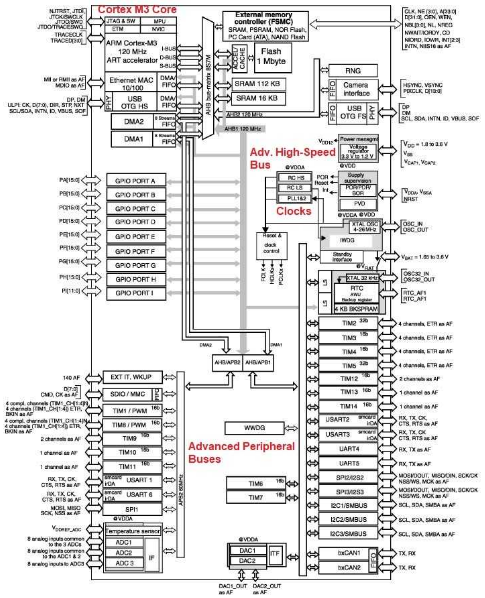
STM32 MCUs Family



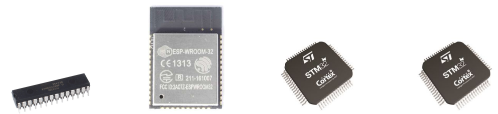
STM32 Cortex-M0 Block diagram



STM32 Cortex-M3 Block diagram



IoT Microcontrollers: Comparison



	ATMEGA328P	ESP8266	STM32 M0	STM32 M3
Freq.	16 MHz	80 MHz	48 MHz	120 MHz
CoreMark	11	191	106	398
Mem	32KB	-	32KB	128KB
Timers	3	2	4	14
GPIO	23	16	26	140
ADC	10-bit	10-bit	12-bit	12-bit
SPI/I2C/I2S/UART	2/1/0/1	2/1/2/2	1/1/0/1	3/3/2/6
Comm.	-	802.11	-	-

IoT Microcontrollers: Considerations

IoT devices are embedded and must be designed with respect to the system requirements:

1. Environmental conditions of operation.
2. Type of sensors connected.
3. Type of actuators controlled.
4. Required power and available power sources.
5. Where is Data processed?
6. Unit cost per device.
7. Expected lifetime.

Hardware Communication Protocols

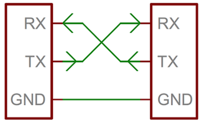
- Microcontrollers can communicate with:
  - PC,
  - Sensors & Actuators,
  - Other modules, e.g., displays, sd cards ...
- Different communication protocols available:
  - UART – bi-directional, asynchronous and serial data transmission.
  - I2C – a half-duplex communication protocol.
  - SPI – a full-duplex commination protocol.
  - One-Wire – a bus system with power supply.
- For some simple sensors we can also use the ADC.

UART Interface

Universal Asynchronous Reception and Transmission

- Simple serial communication protocol operating in 3 modes:
  1. Simplex – data transmission in one direction
  2. Half-duplex – data transmission in either direction but not simultaneously
  3. Full-duplex – data transmission in both directions simultaneously
- Two data lines: one to transmit (TX), one to receive (RX).
- UART is an asynchronous serial transmission:
  - No clock is used.
  - Uses start and stop bits to signal start/end of packets.
- Data frame size is 8 bits + 1 parity bit for error checking.
- Data transmission speed (BAUD Rate) is set to 115,200kbps.

USART



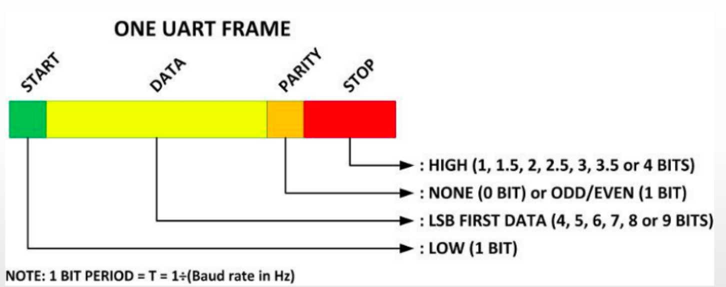
The basic bi-directional communication requires two lines:

- TX - Transmit
- RX - Receive

USART frame

Data is transferred within frames. A frame is composed by:

- START - single bit
- DATA - from 4 to 9 bits
- PARITY - from none to 1 bit
- STOP - from 1 to 4 bits



USART

Being asynchronous, we need to set extra parameters to allow RX and TX to communicate properly. These are:

- BAUD RATE (speed of data exchange)
- PARITY
- DATA SIZE
- STOP BITS

Example: 9600, 8, N, 1. Where:

- 9600 is the baud rate
- 8 is the number of bits in data field
- N is the parity, no parity used
- 1 is the number of stop bits

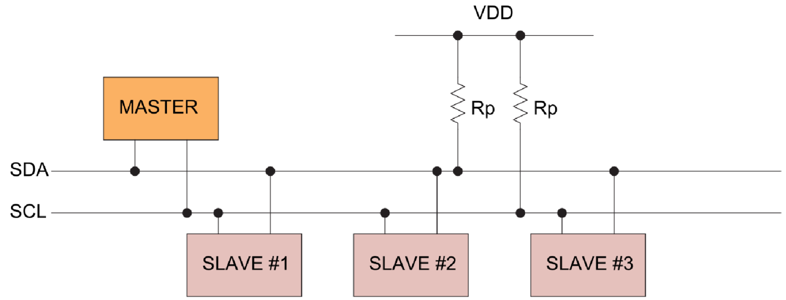
I2C: Inter-integrated-circuit Interface

- Similar to UART but not used for PC-device communication.
- I2C forms a shared bus using only two wires:
  - SCL: serial clock line – used for synchronizing
  - SDA: serial data line acceptance port – used for RX/TX
- I2C uses an address system – up to 128 devices.
  - When controller wants to send data to a peripheral, first states the address of the peripheral before sending any data.
  - When controller wants to receive data from a peripheral, first states the address of the peripheral and waits for data.
- Useful for IoT devices that require many different parts.
- Standard mode devices can communicate from 0 to 100 kbit/s.
- Fast mode devices can receive and transmit at 400kbit/s.
- High-speed devices can communicate up to 3.4 Mbit/s.
- I2C speed dependends on data speed, wire quality and external noise.

I^{2}C

I2C uses only two bi-directional open-drain lines, pulled up with resistors:

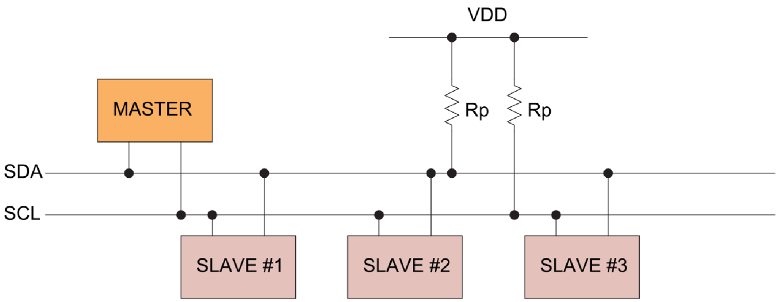
- Serial Data Line (SDA), used for sending and receiving data
- Serial Clock Line (SCL), used for synchronized the different devices



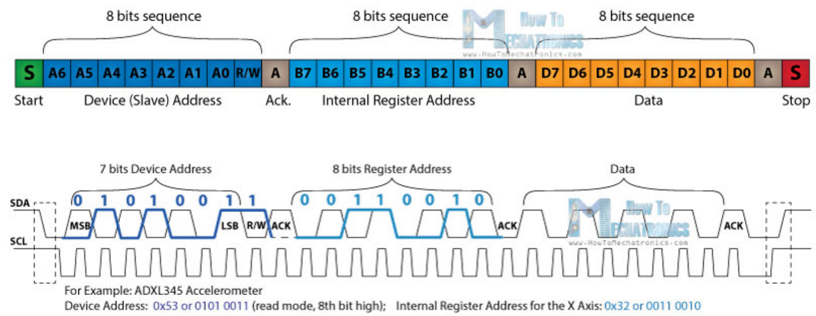
I^{2}C

I2C uses only two bi-directional open-drain lines, pulled up with resistors:

- Serial Data Line (SDA), used for sending and receiving data
- Serial Clock Line (SCL), used for synchronized the different devices



I^{2}C protocol



SPI: Serial Peripheral Interface

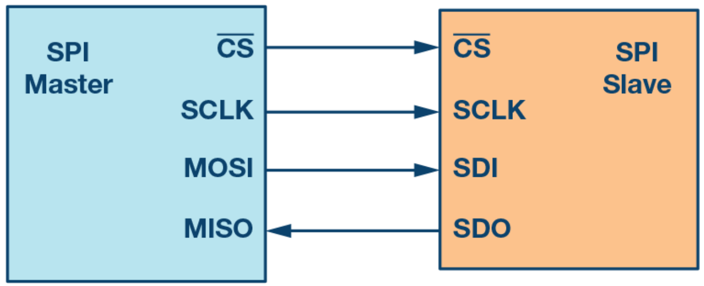
- Similar to I2C, designed for communication between MCU.
- Full-duplex – can send/receive data simultaneously.
- Can operate up to 8Mbits.
- data/clock lines are shared between devices.
- Each device will require a unique address wire.
- There is no limit to the number of SPI device that can be connected.
- The SPI communicates via 4 ports which are:
  1. MOSI – Controller Data Output, Peripheral Data Input
  2. MISO – Controller data input, Peripheral data output
  3. SCLK – clock signal, generated by controller
  4. NSS – Peripheral enable signal, used by controller
- No start/stop bits – data can be transmitted continuously without interruption.
- No form of error check unlike in UART (using parity bit).

SPI

Only four signal lines are required:

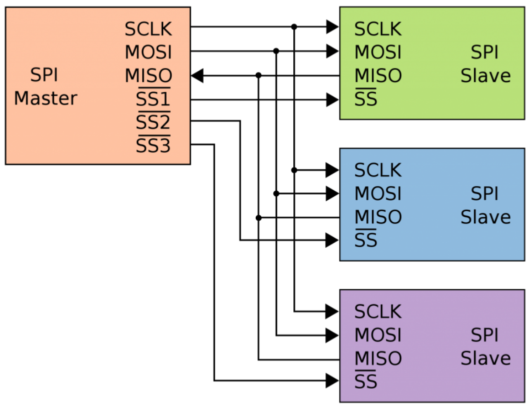
- MISO – Master Input Slave Output
- MOSI – Master Output Slave Input
- SCLK – Serial Clock
- SS – Slave Select

It is used for short distance communications, MISO and MOSI should be tri-state GPIOs.



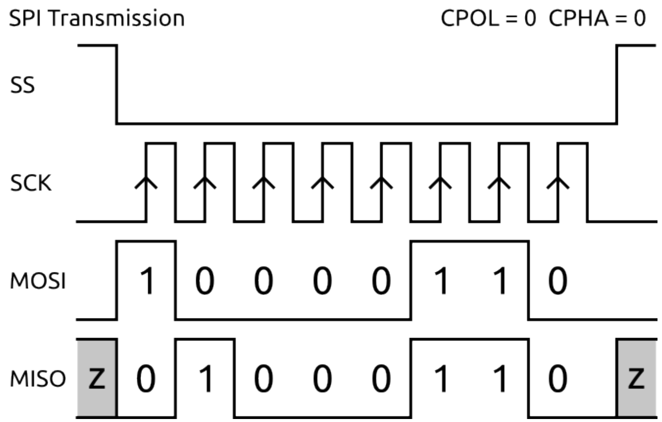
SPI

It is a shared bus with low GPIO requirements and it is sensibly faster than I2C (some peripherals exceed 10Mbit/s).



SPI

During each clock cycle, a bit is transferred from the MASTER to the SLAVE and a bit is transferred from the SLAVE to the MASTER.

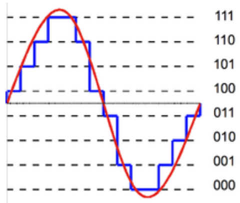


Hardware Peripheral Protocols: Comparison

	UART	I2C	SPI	One-wire
Complexity	Low	Low	High	Low
Speed	115Kbps	3.4Mbps	8Mbps	16.3kpbs
No. wires	1	2	4	1
Duplex	Full	Half	Full	Half
Controllers	1	Many	1	1
Tot devices	1	127	many	20

Analog-to-Digital Converter

Analog-to-Digital Converter - is a system that converts an analog signal into a digital signal.



Physical values are often *analog*. A digital circuit needs to convert them into *digital* values in order to handle them.

ADC Resolution

Our MCU has a single ADC with 12bit resolution and 2.4 Mbps. 12 bit means that the result of an AD conversion gives up to 4096 different values.

If the working range of our ADC is from 0V to 4.096V, then each bit represents 1 mV.

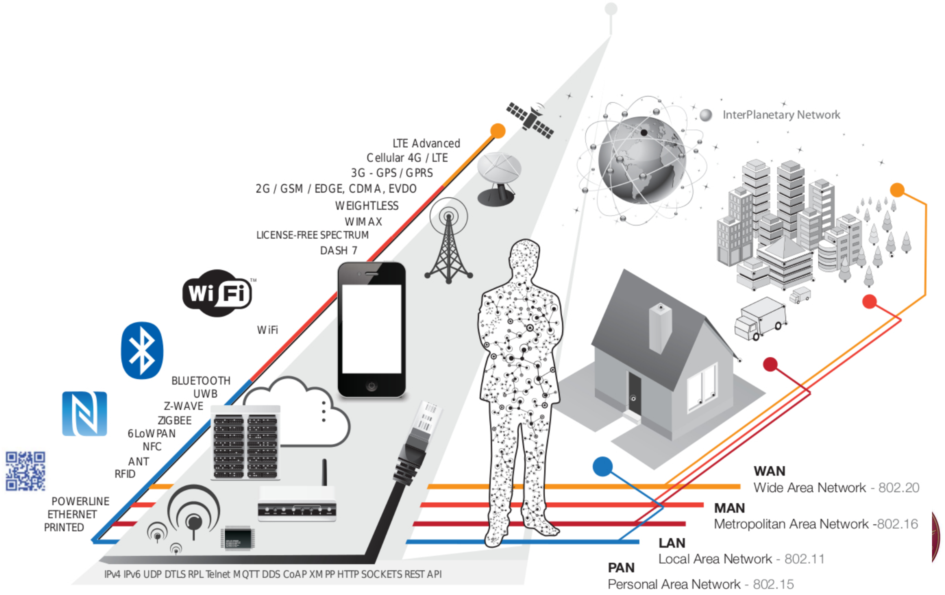
If the working range of our ADC is from -1.5V to 1.5V, then each bit represents $(1.5 + 1.5) / (2^{12}) = 3 / 4096 = 0.73$ mV.

ADC Sampling sequence

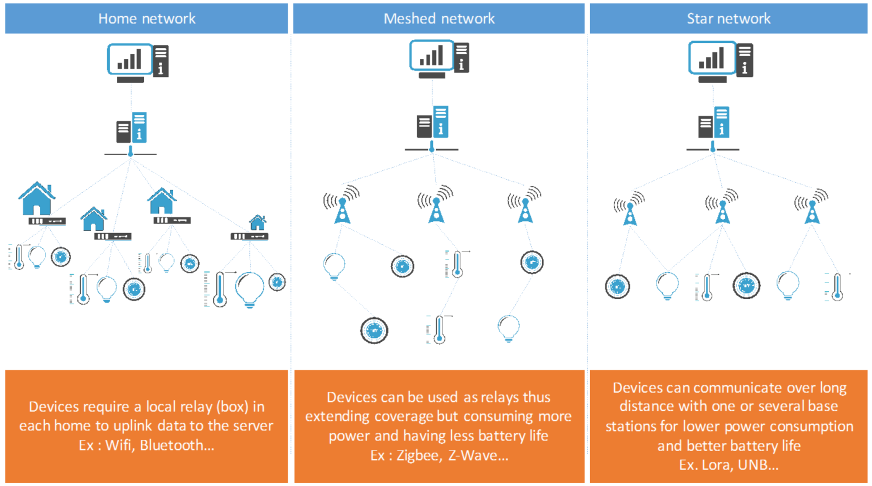
ADC is considered a slow peripheral, thus a reading sequence should follow this approach:

1. Initialize the ADC peripheral
2. Define an interrupt routine
3. Send the sampling command and do other stuff while waiting
4. The interrupt routine will be executed on ADC sampling completed

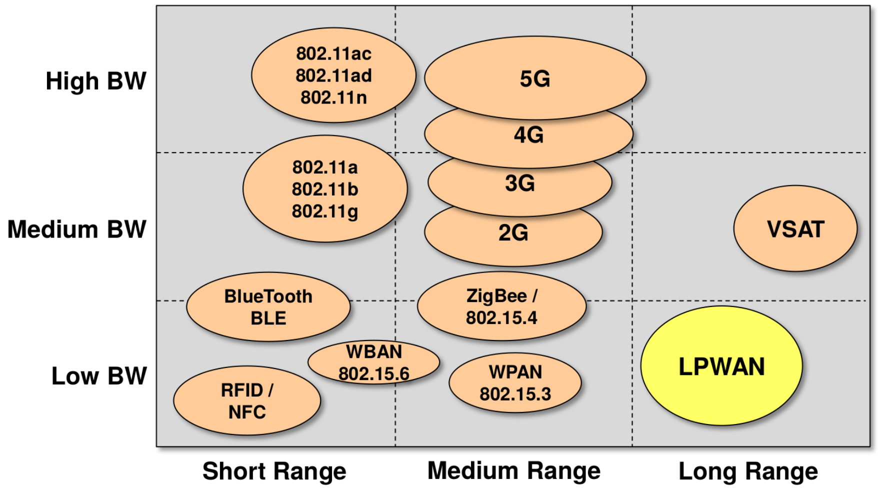
Broad range of connectivity options



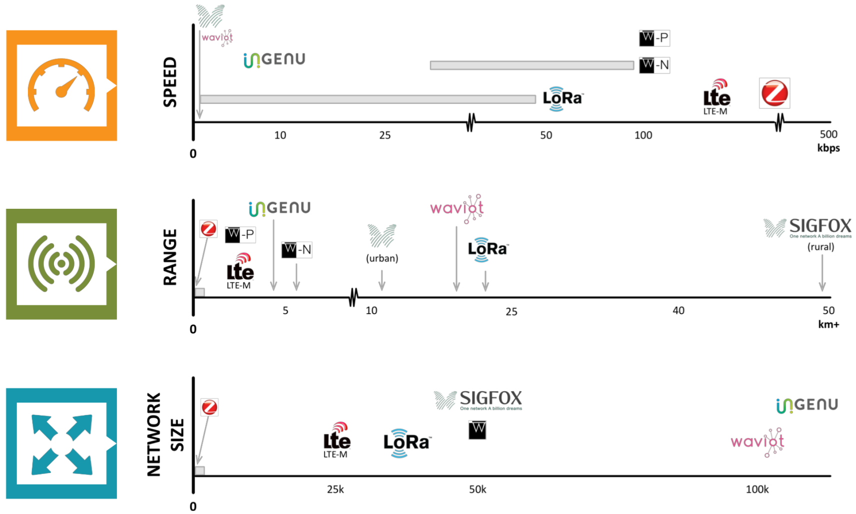
Wireless Network Technologies: Topologies



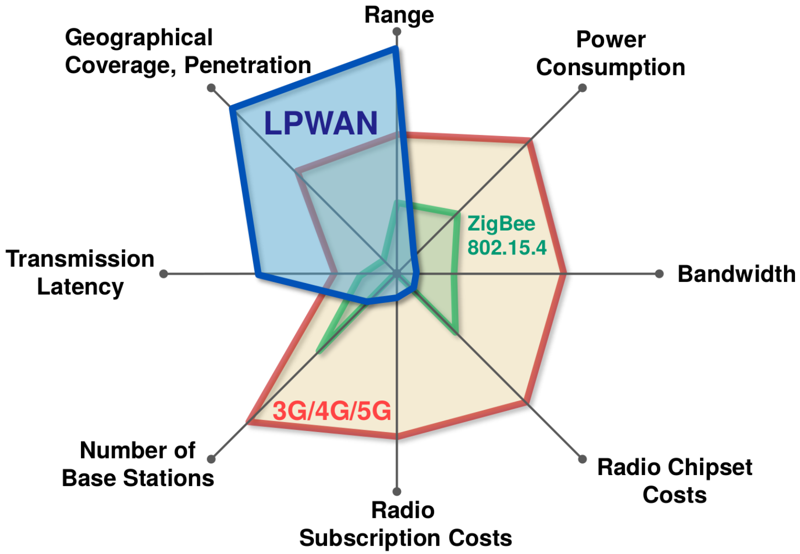
Wireless Network Technologies: Coverage & Bandwidth



Wireless Network Technologies: Capacity



IoT Networking Considerations



IoT Networking Considerations: Bandwidth

- Volume – the data each device gathers and transmits.
  - Constant transmission vs Periodic sampling.
  - Resolution of sensing.
  - Packet size limitations – message fragmentation.
- Network size – the number of devices deployed.
- Velocity – frequency of transmitted data.
  - constant stream vs intermittent bursts,
  - peak periods of increased volume?
- Power usage
- Interminent connectivity
- Interoperability
- Security
  - Authentication – Key freshness
  - Encryption

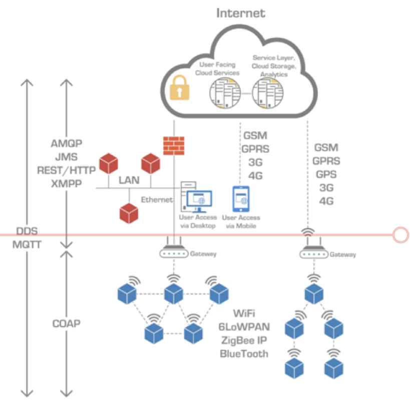
Network Components and Protocols

Stage4: Data-Centers and Cloud

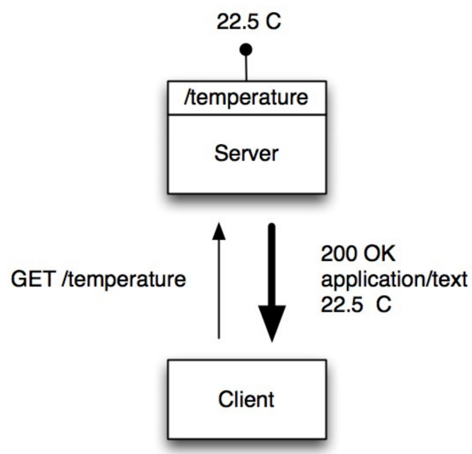
Stage3: Edge IT

Stage2: Internet GW & data acquisition/ aggregation /control systems

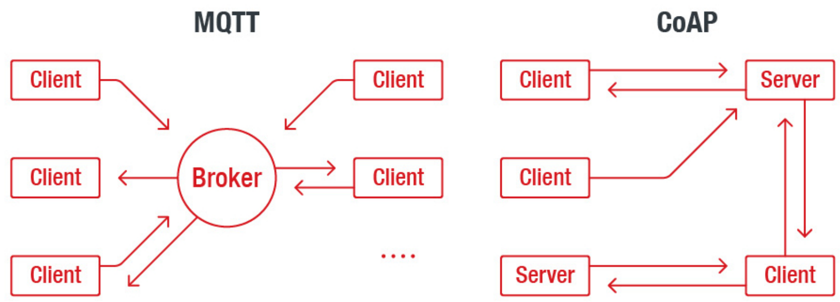
Stage1: sensors & Actuators



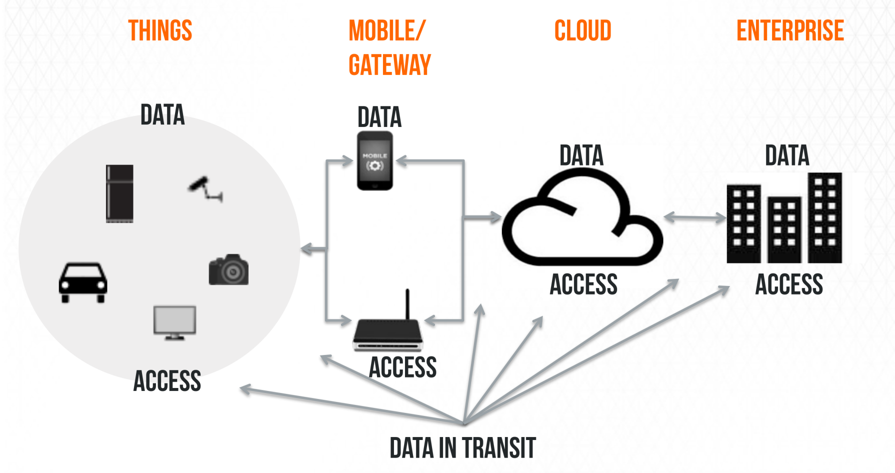
Web-inspired Data Collection



Many-to-Many vs One-to-One



IoT Security



IoT Security Challenges

- Unique device identification
- Device authenticity
- Data integrity
- Device-User association

- Low-friction human interaction
- Limited encryption capabilities
- Limited resources
- Limited clock synchronization
- Firmware upgrades

IoT Security Design Rules

- Built-in security – Security by design
  - Identity & Access management part of the design
- Use well-established cryptographic primitives
  - Use good key lengths
- Obscurity does not provide security
- Ensure data and credentials are encrypted
  - When transmitting
  - While storing
- Use a secure channel to transmit firmware
  - Ensure firmware does not contain hardcoded credentials
  - Ensure upgrade is signed and verified
  - Do not send the public key with the firmware, e.g., use a hash
  - Ensure your GIT repositories do not contain your private keys
- Ensure physical access to the device is controlled
  - Use a TPM hardware module to protect against disassembly access to internal storage (RAM/ROM)

Privacy as part of the design

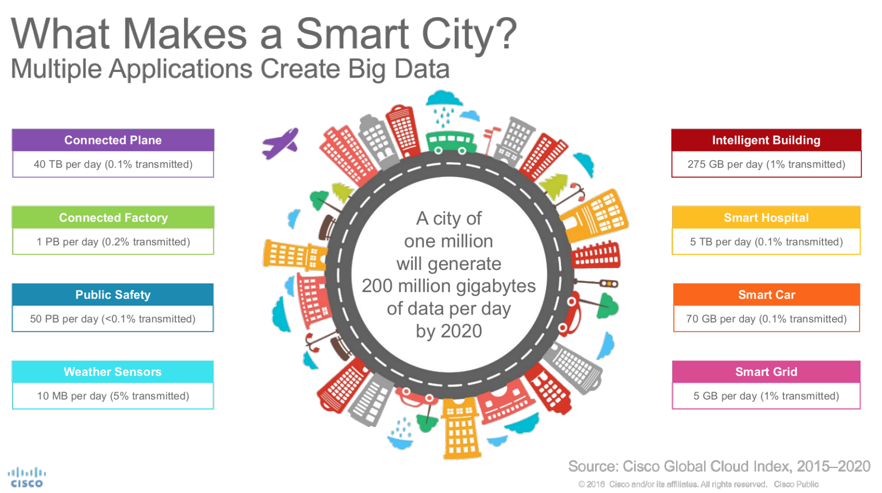
- Collect only the minimum necessary data
- Ensure sensitive data are properly encrypted and stored
- Ensure the device properly protects personal data
- Always request concent from the user when about to store or transfer sensitive data

Intelligent big data analytics

- IoT is a major data provider.
- Apply cognitive computing techniques over Iot data
  - in batch mode
  - in streaming mode
  - in real-time or near real-time
  - over historical data
- A multitude of complementary approaches
  - Statistics
  - Modeling
  - Data Mining
  - Machine learning
  - Artificial Intelligence

IoT Data Engineering & Analytics

- We wish to process the data arriving from the sensors.
- Data Cleaning – Erroneous Values
- Data Enrichment – Missing Values
- Produce statistics for predefine period of time:
  - Every Hour
  - Every Day
  - Every Week
  - ...
- Carry out various data mining tasks:
  - Identify anomalies
  - Identify seasonality of values
  - Identify corellation between values
  - ...



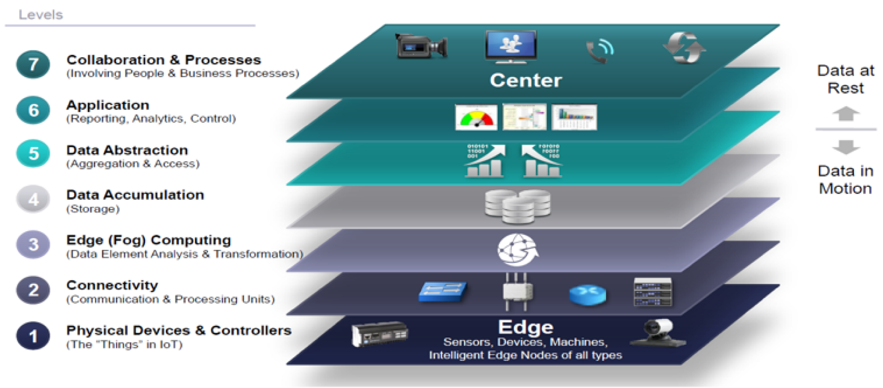
Data Volume vs Network Latency

Close → Far

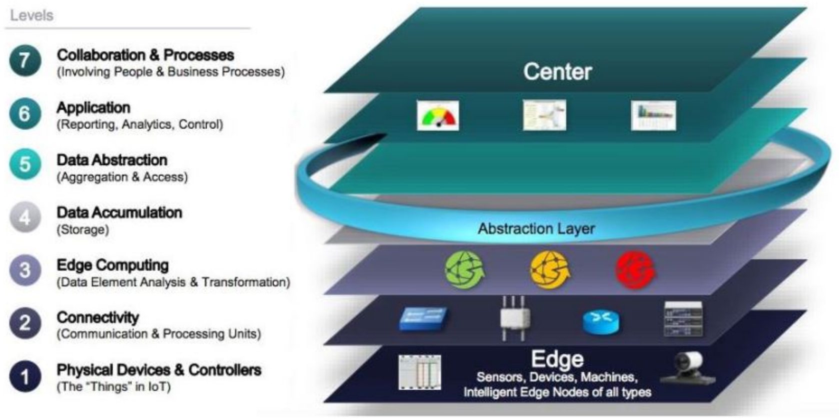
Data Size						
100 PB	124 days	3 years	34 years	340 years	3,404 years	34,048 years
10 PB	12 days	124 days	3 years	34 years	340 years	3,404 years
1 PB	30 hours	12 days	124 days	3 years	34 years	340 years
100 TB	3 hours	30 hours	12 days	124 days	3 years	34 years
10 TB	18 minutes	3 hours	30 hours	12 days	124 days	3 years
1 TB	2 minutes	18 minutes	3 hours	30 hours	12 days	124 days
100 GB	11 seconds	2 minutes	18 minutes	3 hours	30 hours	12 days
10 GB	1 second	11 seconds	2 minutes	18 minutes	3 hours	30 hours
1 GB	0.1 seconds	1 second	11 seconds	2 minutes	18 minutes	3 hours
	100 Gbps	10 Gbps	1 Gbps	100 Mbps	10 Mbps	1 Mbps

Network Bandwidth

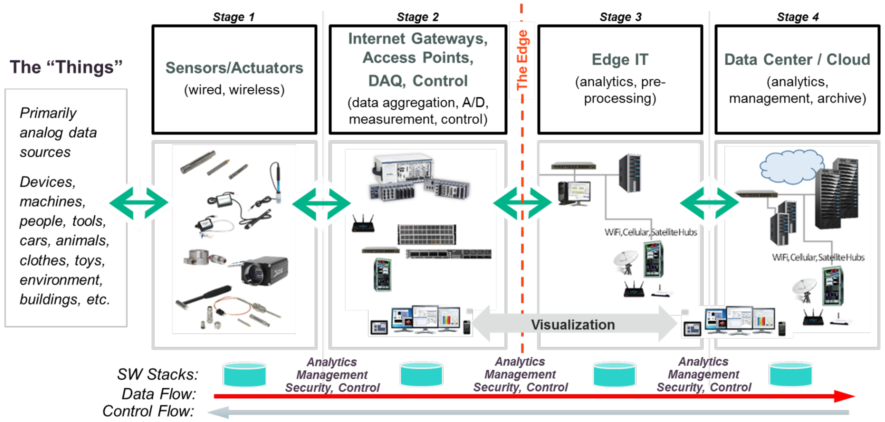
Main Architectural Levels (Cloud-facing)



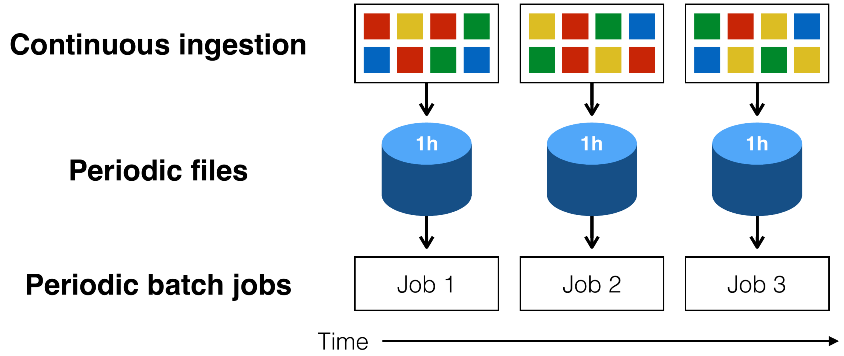
Main Architectural Levels (Edge-facing)



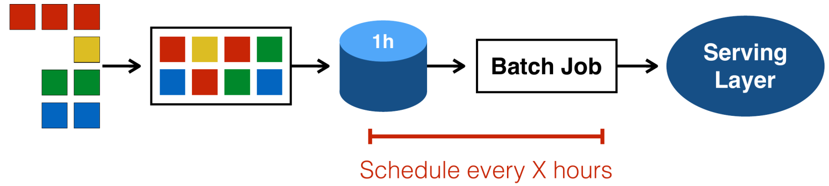
IoT Data Processing Stages



Batch processing



High Latency



- Latency from event to serving layer usually in the range of hours.

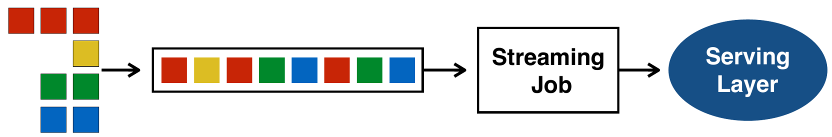
Stream-based Data Processing

- Today a variety of mature stream processors are available: Flink, Spark . . . .
- Stream-based processing is enabling the obvious: continuous processing on data that is continuously produced.
  - Monitor data and react in real time.
  - Implement robust continuous applications.
  - Adopt a decentralized architecture.
  - Consolidate analytics infrastructure.
- Enables Contunuous Analytics
  - A production data application that needs to be live 24/7 feeding other systems (perhaps customer-facing)
  - Need to be efficient, consistent, correct, and manageable

Streaming vs Real-time

- Streaming != Real-time
- E.g., streaming that is not real time: continuous applications with large windows
- E.g., real-time that is not streaming: very fast data warehousing queries
- However: streaming applications can be fast
- When and why does this matter?
  - Immediate reaction to life
    - E.g., generate alerts on anomaly/pattern/special event
  - Avoid unnecessary tradeoffs
    - Even if application is not latency-critical
    - With stream processing frameworks you do not pay a price for latency!

Streaming all the Way

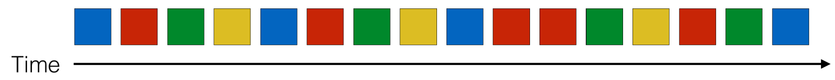


Windowing

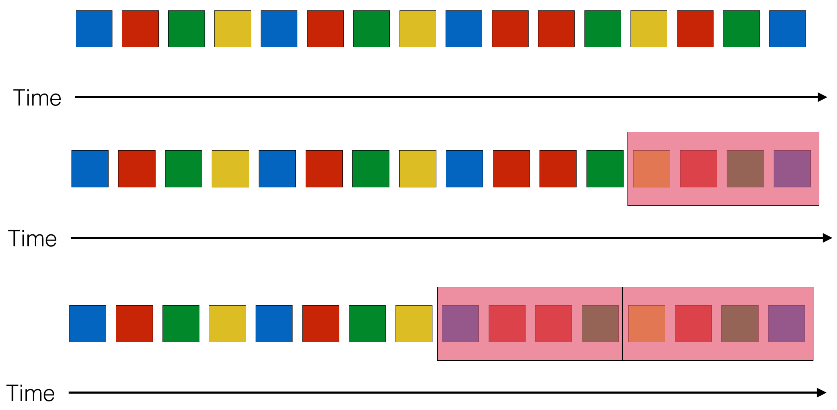
Aggregates on **streams** are scoped by **windows**

Time-driven *e.g. last X minutes*

Data-driven *e.g. last X records*

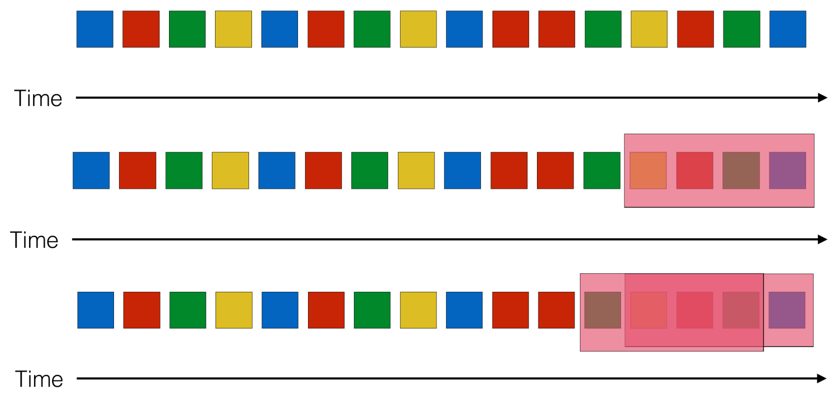


Tumbling Windows (No Overlap)



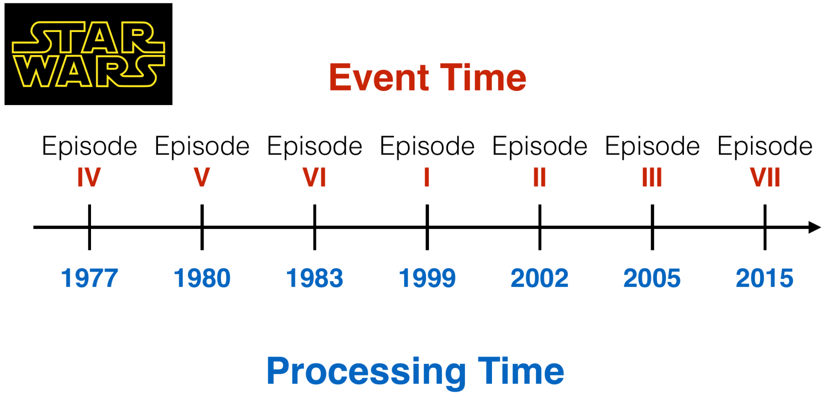
- Example: Average value over the last 5 minutes.
- Maximum value over the last 100 readings.

Sliding Windows (With Overlap)

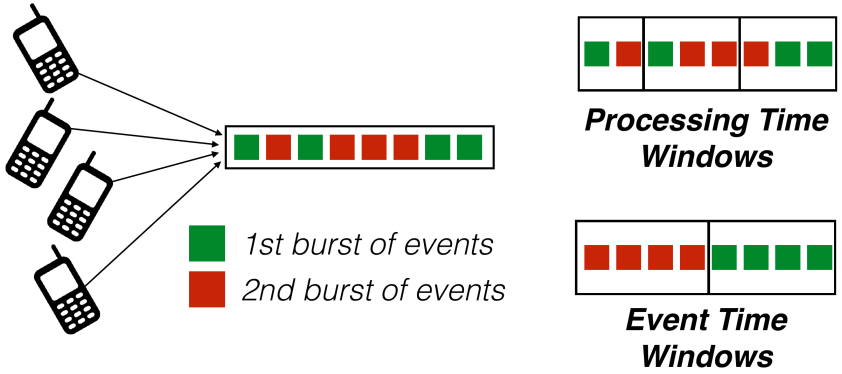


- Example: Average value over the last 5 minutes, updated each minute.
- Maximum value over the last 100 readings, updated every 10 readings.

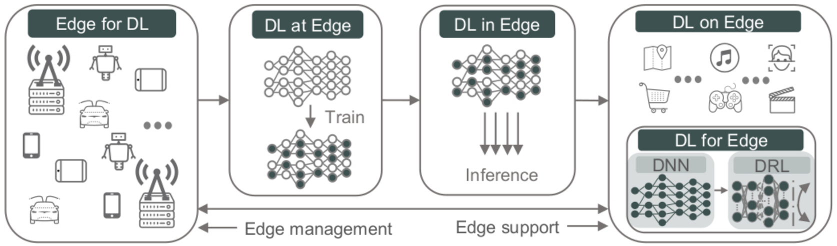
Out of Order Events: Example 1



Out of Order Events: Example 2



Cloud vs Edge Intelligence



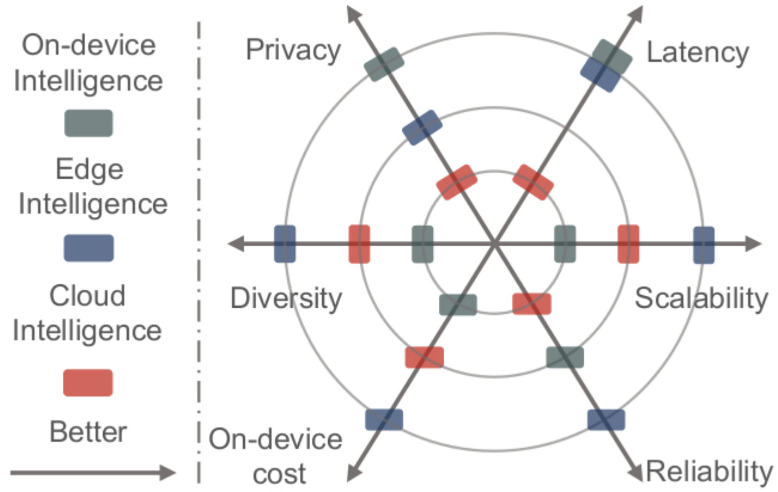
DL at Cloud

- Privacy – all this data (most of which is personal) is traveling to and from servers and the cloud.
- Latency and network costs – even if you have a fast server, transferring the data to the server for training is often the bottle-neck.
  - Example: the data collected from 1 hour of driving by a connected car, takes over 10 hours to upload.
- Scalability – the more connected devices there are the more server power you need and the more expensive the transfer of the data gets.
- Training costs / server costs – training costs are growing as data is growing and new models are being developed.

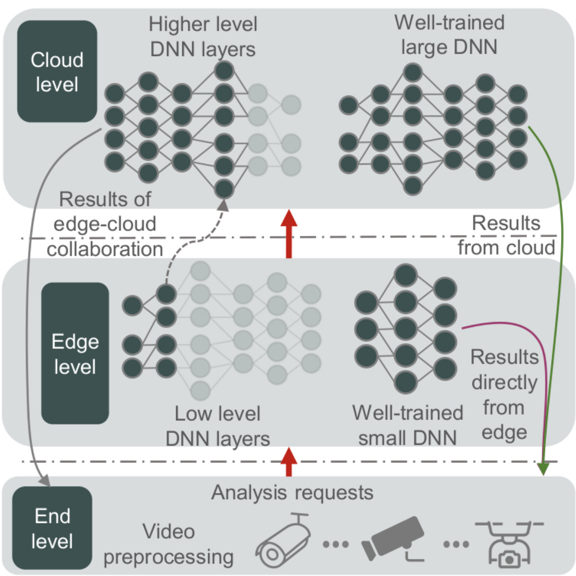
DL at Edge

- Privacy and information security – no need to hash, anonymize, encrypt, or use any other forms for rendering data private.
- Reduced time and network latency – classification is carried out at the point where data are produced.
- Scalability – embedded hardware accelerators (e.g., GPU, ASIC, FPGA) can improve memory access and parallelize the execution of tasks and potentially providing real-time analysis.
- Training costs / server costs – training at the edge, or training in collaboration with the cloud can potential reduce the costs while keeping accuracy at high levels.

Cloud vs Edge Intelligence: Comparison



Cloud – Edge Collaboration



Cloud – Edge Collaboration