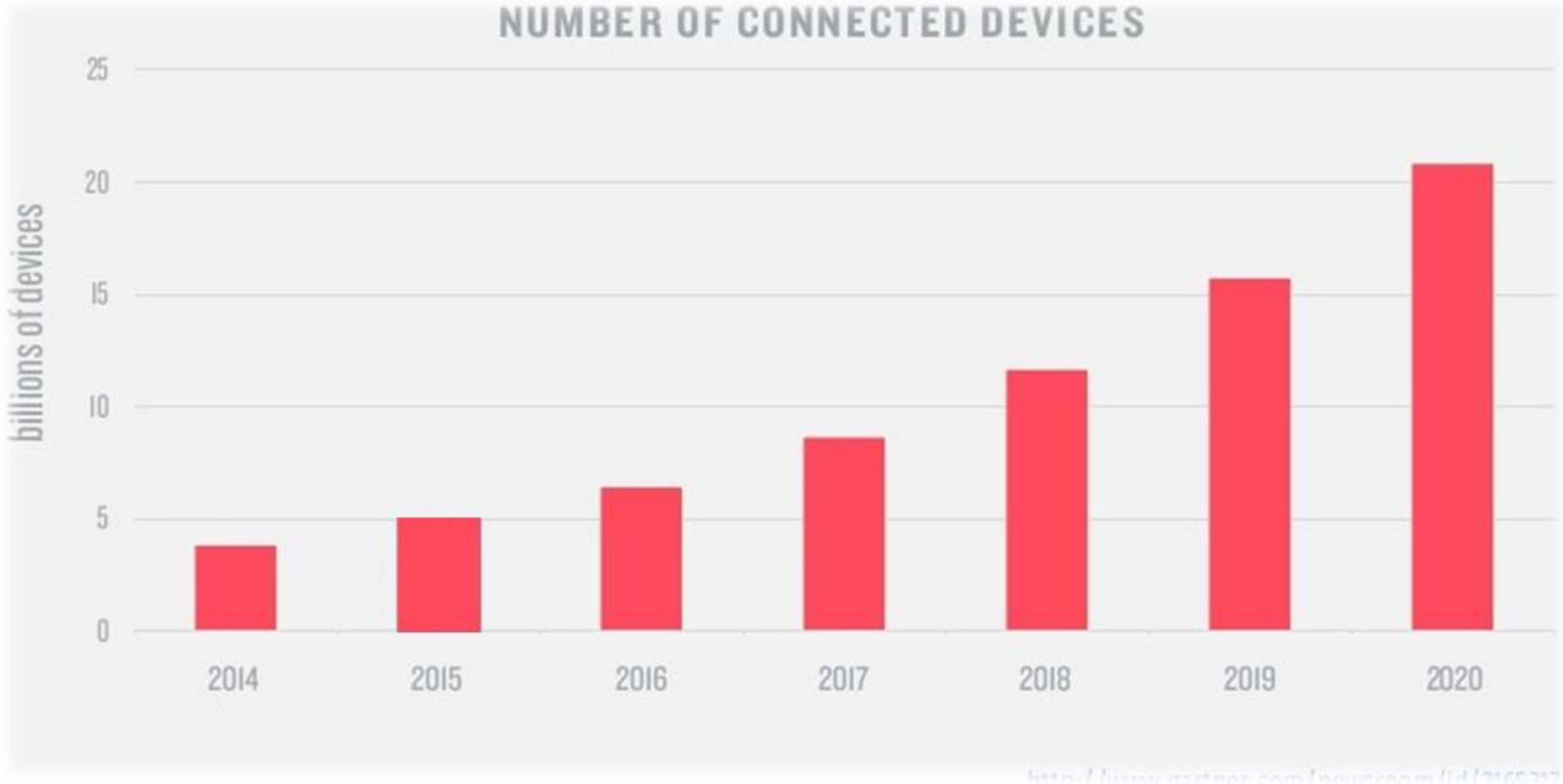
IoT System Design Cycle

IoT – The Internet of Things

- IoT is the term that represents a collection of **ideas, devices and processes.**
- **Each thing is represented by a device or a sensor**
- These things are usually working together to **create larger solutions by** sending and reacting to **data from an ecosystem**

Why DOES IoT MATTER?

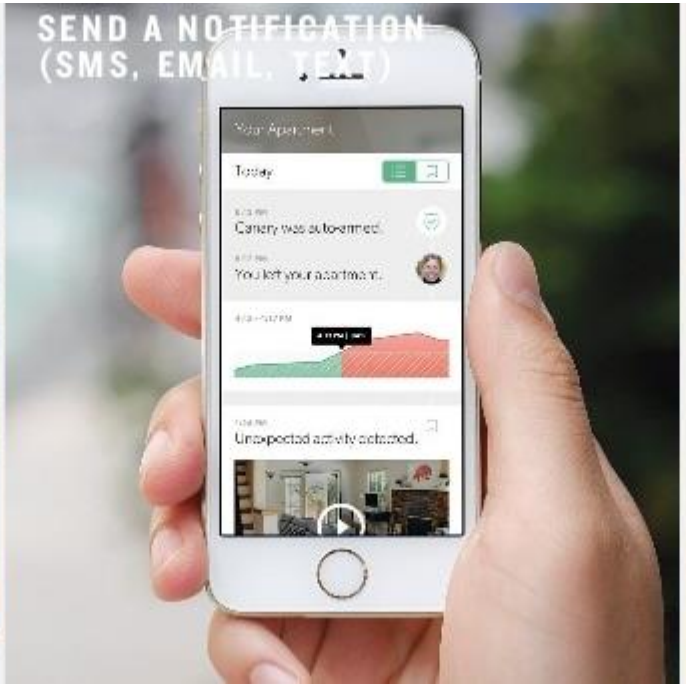


The IoT Lifecycle

Collect – Devices and Sensors are collecting data everywhere

Communicate – Sending data and events through networks to some destination

Analyze – Creating information from the data

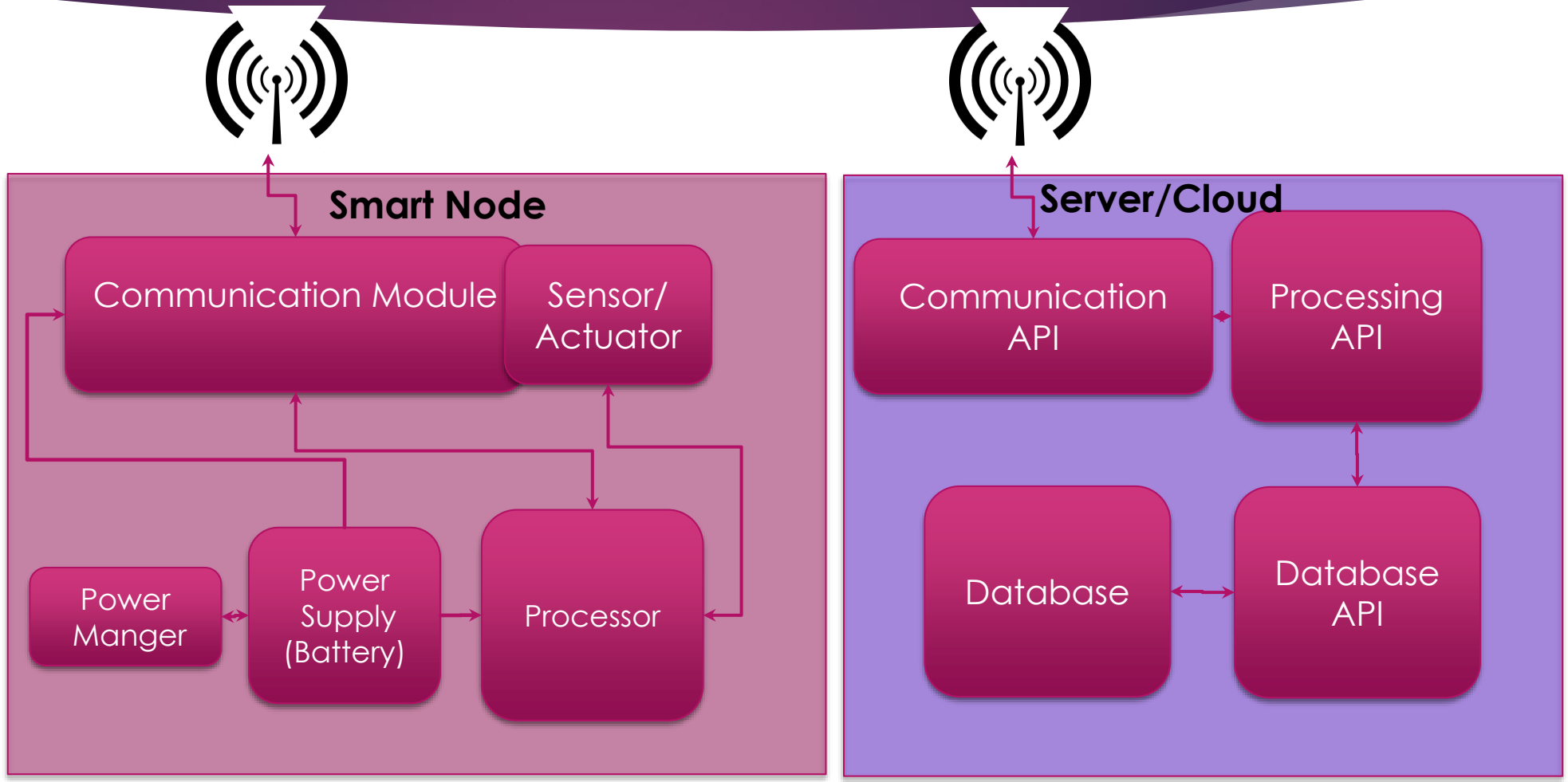


Act – Taking action based on the information and data

IoT Architecture

- Core infrastructure
  - Sensors, actuators
  - Servers
  - Communication network
- Software aspects
  - Middleware to connect and manage all the heterogeneous components
  - Standardization to connect many different devices
- There is no single consensus on architecture for IoT, which is agreed universally.
- Different architectures have been proposed by different researchers.

Generic Architecture



Design Challenge

Time-to-prototype: the time needed to build a working version of the system

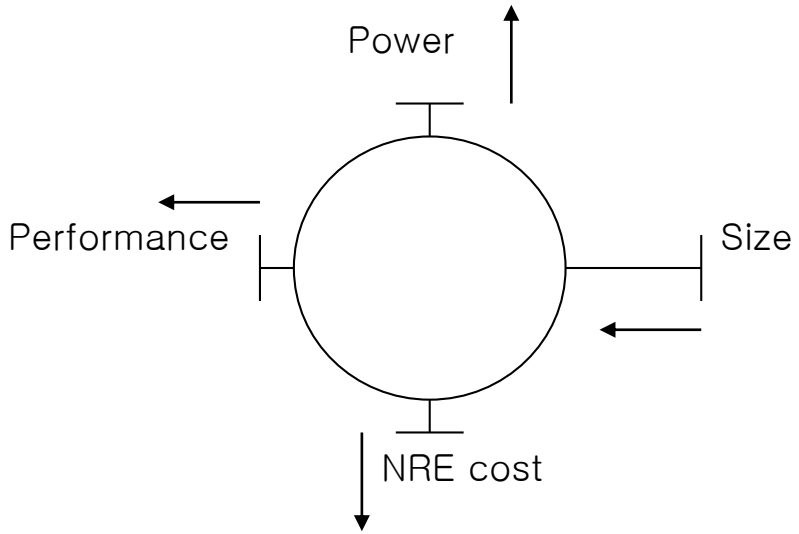
Time-to-market: the time required to develop a system to the point that it can be released and sold to customers

NRE cost (Non-Recurring Engineering cost): The one-time monetary cost of designing the system

Flexibility: the ability to change the functionality of the system without incurring heavy NRE cost

Design metric competition -- improving one may worsen others

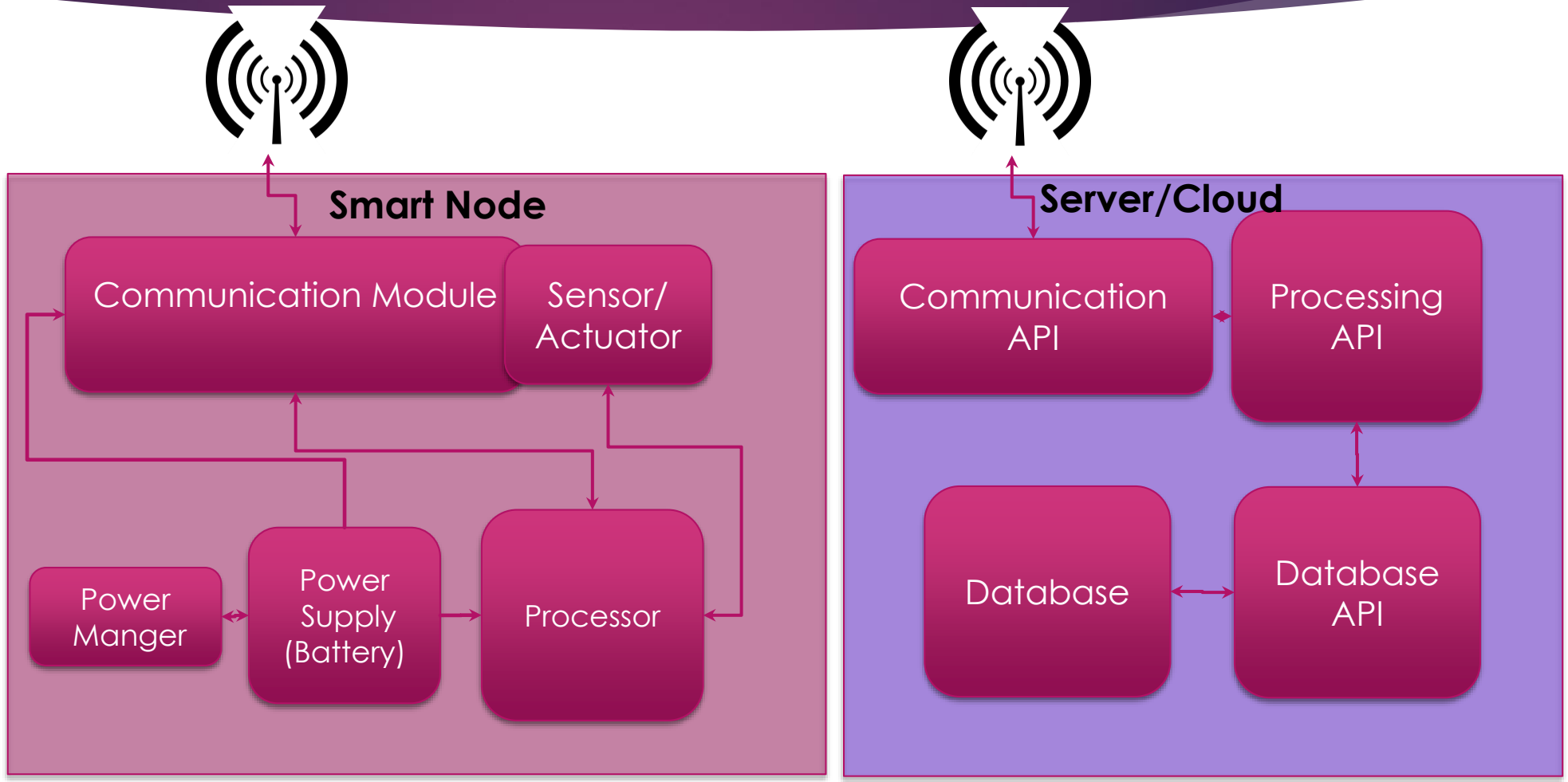
- Design metric
  - A measurable feature of a system's implementation
  - Optimizing design metrics is a key challenge
- Expertise with both **software and hardware** is needed to optimize design metrics
  - Not just a hardware or software expert, as is common
  - A designer must be comfortable with various technologies in order to choose the best for a given application and constraints



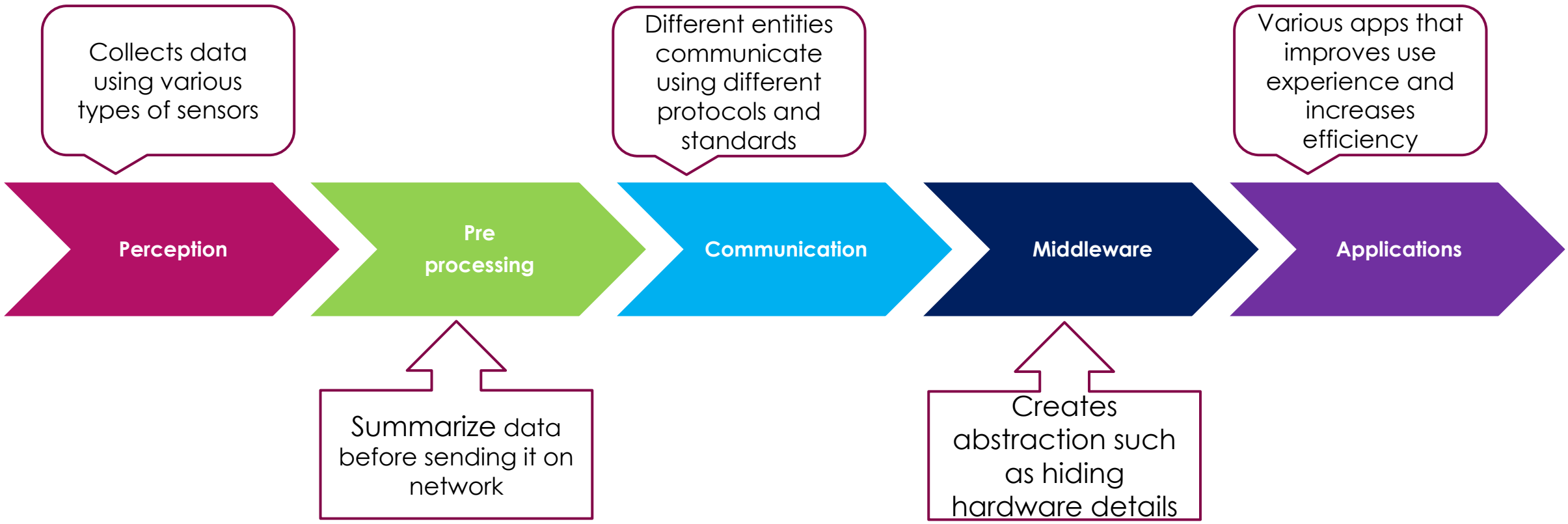
Design Questions

- How much Power is needed?
  - Is power an issue?
  - Average Current needed?
  - Max. Transient Current
- Do I need communications/Networking module?
  - How far? (Distance Travelled)
  - Authorized Frequencies
  - Antenna Size
  - Required Power
  - It is 1 device or in a network?
  - Do I need routing? .....

Generic Architecture



IoT Taxonomy

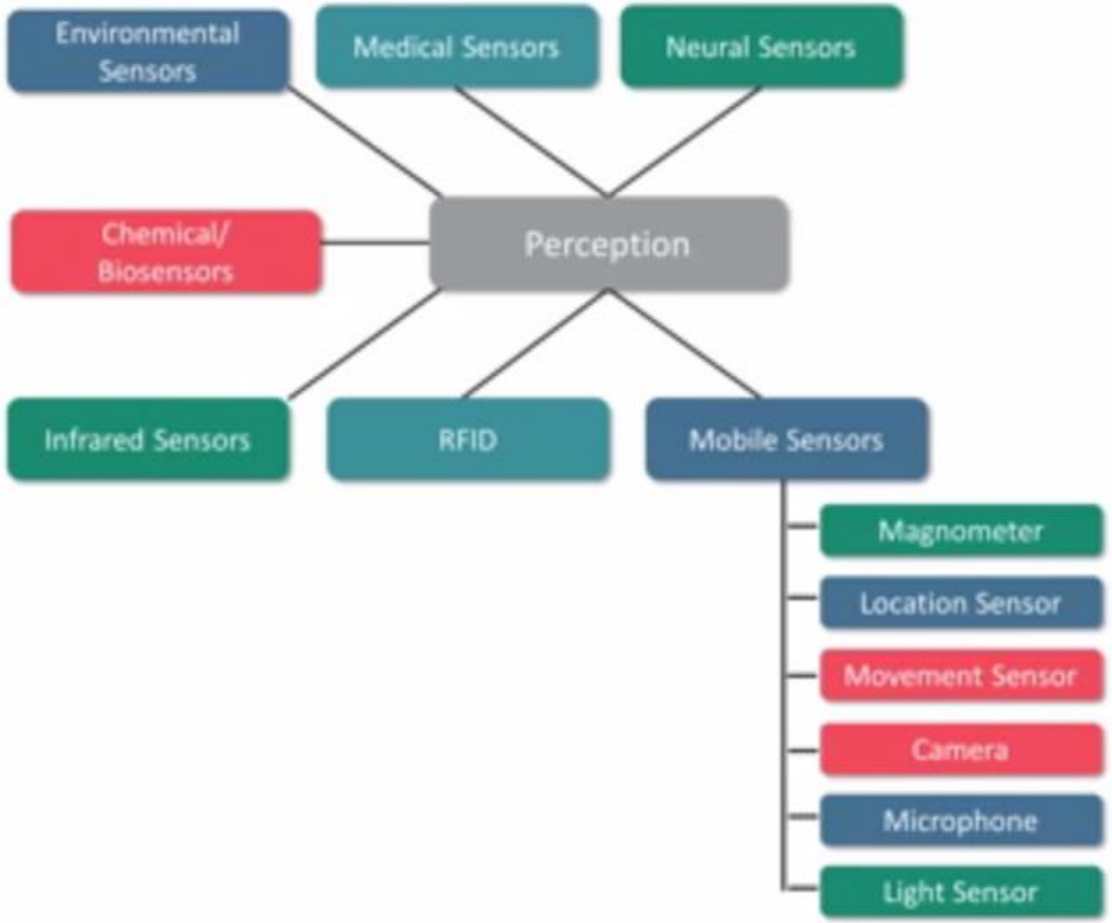


Perception

IoT applications need one or more sensors for collecting data from environment

Sensors are small in size, low cost and consumes low energy

Various types of sensors are used for building smart applications



Preprocessing

Limitations of processing everything in cloud

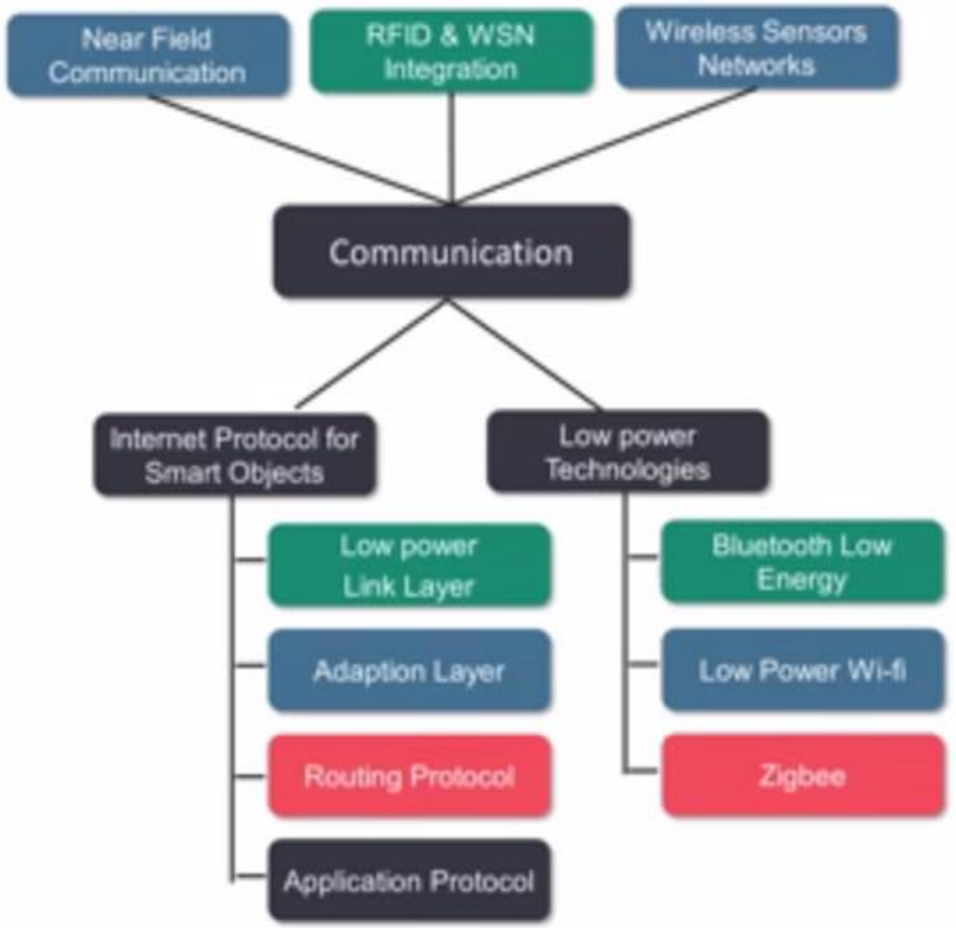
- Mobility: most of the smart devices are mobile.
- Reliable and real time actuation: Latency sensitive applications need real-time responses
- Scalability: more devices means more requests to the cloud, thereby increasing the latency.
- Power constraints: communication consume lot of power

Decentralized computing infrastructure in which data, compute, storage and applications are located somewhere between the data source and the cloud.

Communication

Communication challenges which needs to be addressed

- Addressing and identification – every smart device needs to be identified with a unique address in the network
- Low power communication – communication consuming low power
- Routing protocols with low memory requirement and efficient communication patterns.
- High speed and nonlossy communication.
- Mobility of smart things.



Middleware

Ubiquitous computing (or "ubicomp") is a concept in software engineering, hardware engineering and computer science where computing is made to appear anytime and everywhere. In contrast to desktop computing, ubiquitous computing can occur using any device, in any location, and in any format.

- Ubiquitous Computing is the core of IoT – Computing and connectivity in all the things around us.
- Interoperability of such heterogeneous devices needs well defined standards
- But standardization is difficult because of varied requirements of the different applications and devices.
- Middleware platform – which will abstract the details of the things for application. It will hide details of smart things.
- It acts as a bridge between the things and the applications.
- It needs to provide services to the application developers so that they can focus more on the requirement of applications rather than on interacting with the baseline hardware.
- It abstracts hardware and provides an application program interface (API) for communication, data management, computation, security and privacy etc…

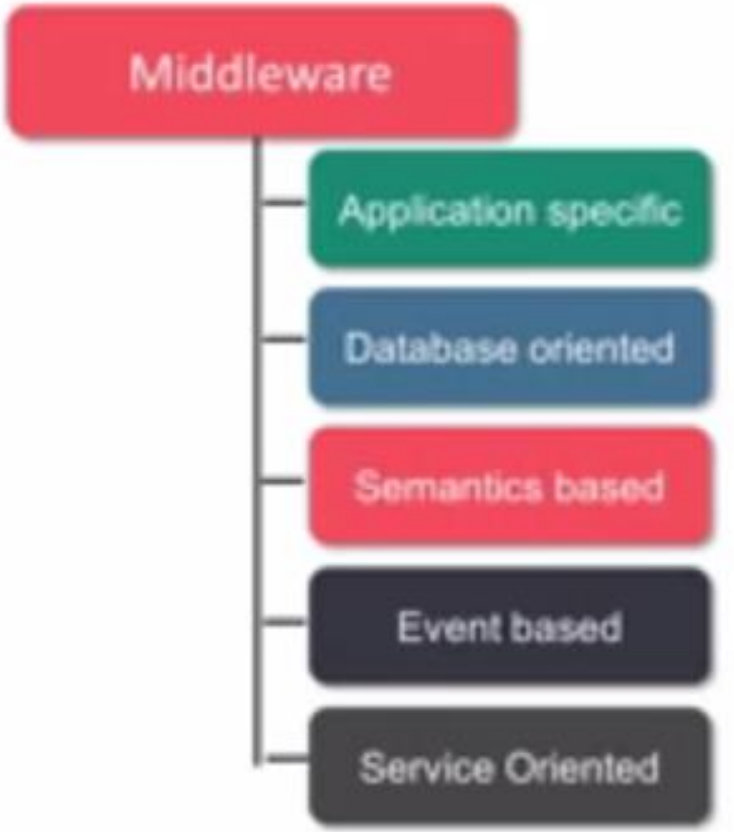
Middleware – challenges addressed

- *Interoperability and programming abstractions* - Interoperability is of three types: network, semantic, and syntactic.
- *Device discovery and management* - this feature enables the devices to be aware of all other devices in the neighborhood and the services provided by them.
- *Scalability* - by making required changes when the infrastructure scales.
- *Big data and analytics* - extrapolate data by using sophisticated machine learning algorithms.
- *Security and privacy*
- *Cloud services* - to seamlessly run on different types of clouds and to enable users to leverage the cloud to get better insights from the data collected by the sensors.
- *Context detection -* to extract the context by applying various types of algorithms. The context can subsequently be used for providing sophisticated services to users.

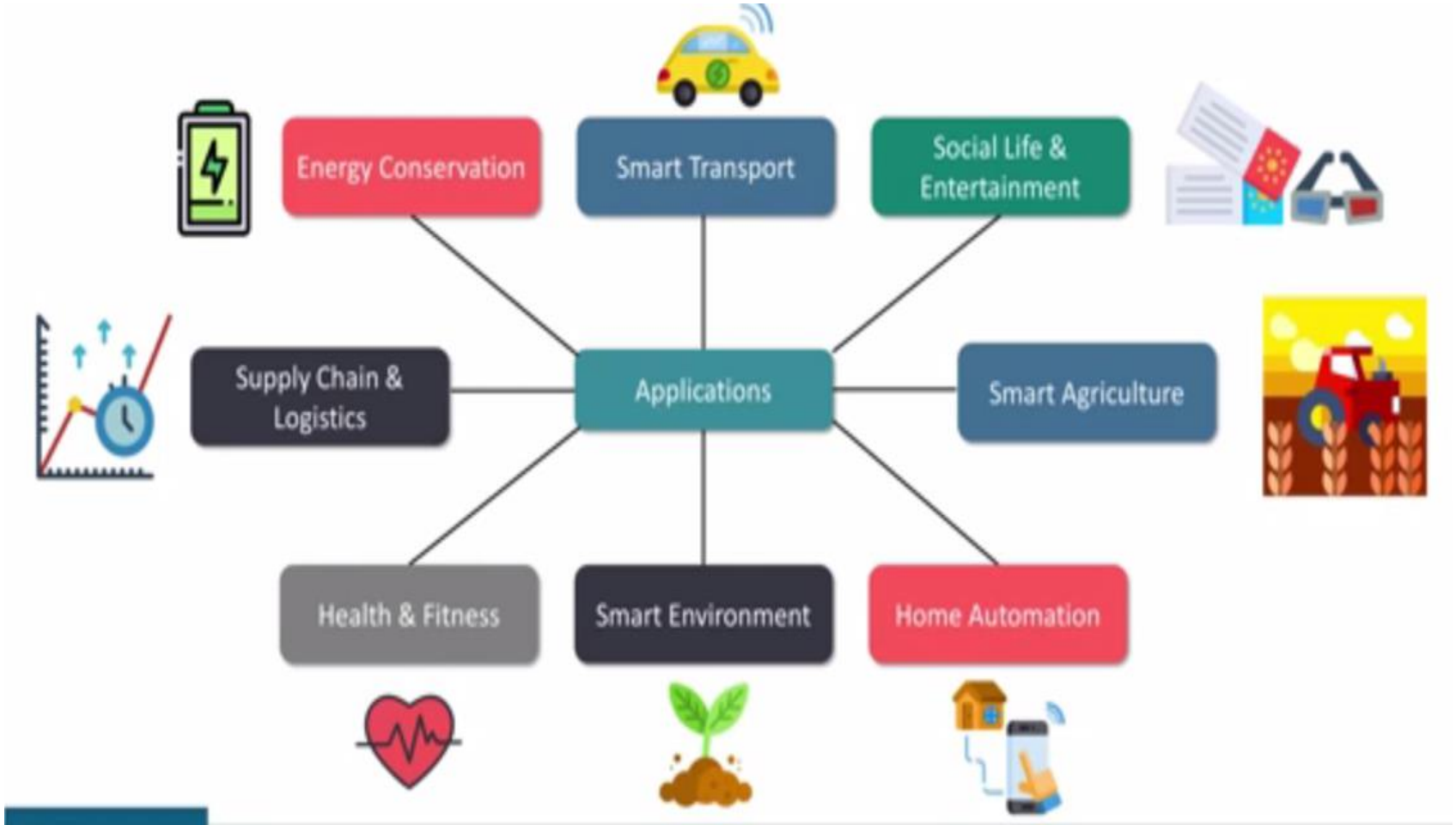
Classification of the Middleware based on their design

Popular IoT middleware

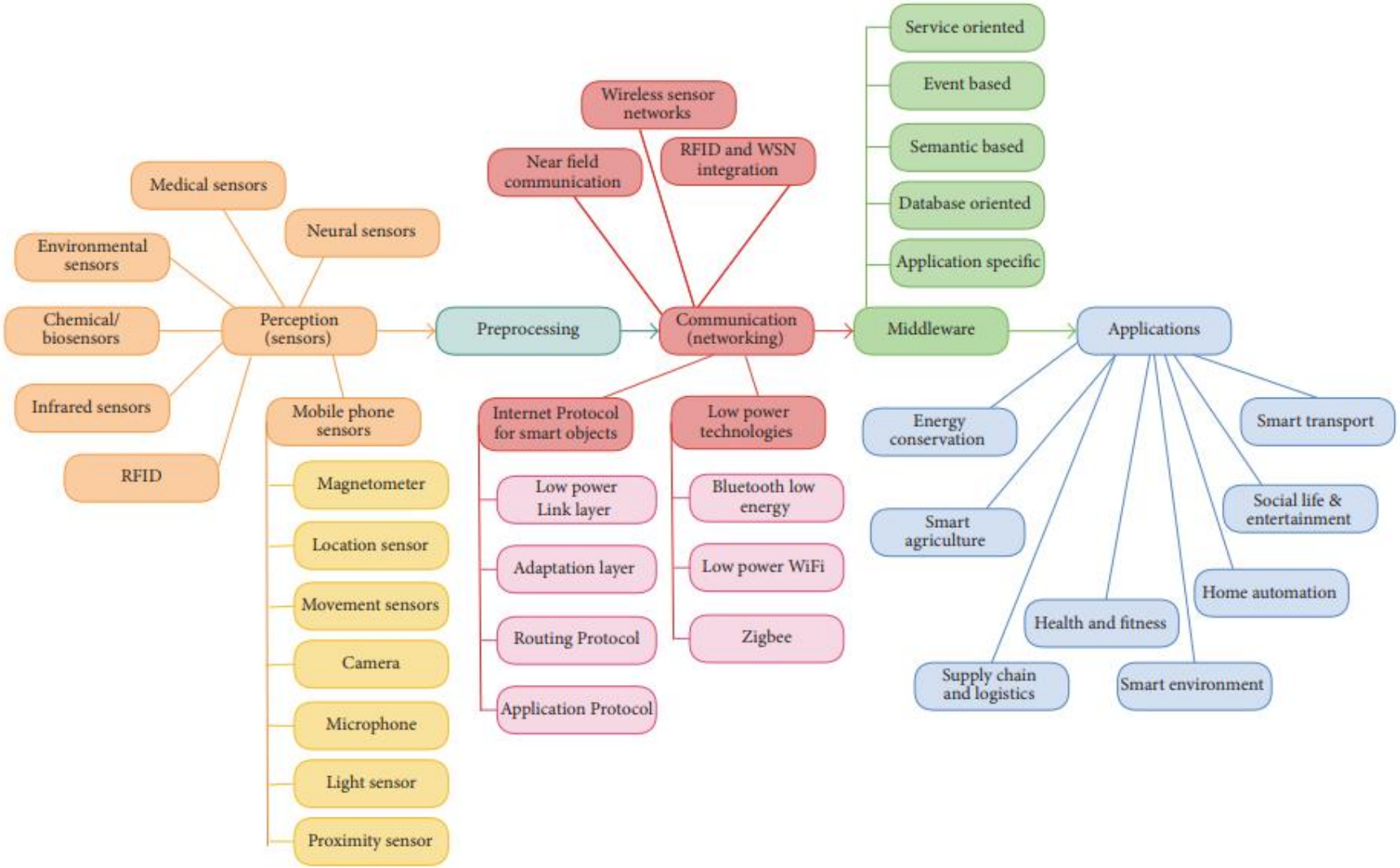
- FiWare
- OpenIoT



Applications



IoT Taxonomy



Design considerations for designing a practical IoT network.

- Design of the sensors – power and logistics
- Communication - The power required to transmit and receive messages is a major fraction of the overall power, and as a result a choice of the networking technology is vital.
- Middleware - to choose between an open source middleware such as FiWare or a proprietary solution.
- Application layer - includes data mining, data processing, and visualization APIs.

Thank you!