Appunti IoT Systems

Capitolo 7

Architettura IoT e livello di astrazione

Simone Coco

Architettura IoT e livello di astrazione

1. **IoT is not a single technology:** IoT non è una singola tecnologia ma è un insieme di più tecnologie: lo smart object non si interfaccia con un'unica realtà ma con più ambiti; quindi non si parla di una singola tecnologia ma di un vero e proprio agglomerato di tecnologie.
2. **The data collected:** IoT sviluppa un paradigma di progettazione delle applicazioni che viene detta **data driven**, di applicazioni basate sui dati: questo porta ad un problema di processamento e immagazzinamento dei dati.
3. **The storage and processing capabilities:** Lo storage ed il processamento a volte non può avvenire all'interno degli smart object poiché questi hanno delle **risorse limitate:** alcuni non predispongono di CPU per una complessa computazione, alcuni sono alimentati in potenza poiché alimentati da batterie, altri non possono conservare molte informazioni per la poca memoria, etc…
4. **Preprocessing:** Si utilizzano molte tecniche: quando i dati vengono captati dai sensori, questi possono essere "**filtrati" (processati)** per estrarre da essi i dati informativi (esempio: se do un comando ad Alexa, non viene probabilmente non viene processata l'intera frase come unico dato ma vengono estratte delle parole chiave che innescano i funzionamenti desiderati). E' giusto chiedersi però se sia meglio processare questi dati nello smart object o da un'altra parte.
5. **Main challenge:** La sfida principale è quindi **ottenere i dati che servono, al giusto livello di dettaglio e nei tempi desiderati.** (dati che siano utili, dettagliati al punto giusto e immediati). Ottenere questo risultato è una grande sfida perché cerca di ottenere tre caratteristiche del dato che a volte possono anche entrare in "contrasto" (esempio: un dato molto complesso poiché molto dettagliato potrebbe non essere ricevuto in tempi immediati se non vi è un processore molto potente)
6. **Communication challenge:** Altre sfide sono quelle legate alla **comunicazione**: gli smart objects devono essere in grado di connettersi. Gli smart objects sono perlopiù connessi con connessioni wireless che "soffrono" di eventuali distorsioni
7. **Core infrastructure of an IoT framework:** Alla base di un sistema smart vi sono: sensori, attuatori, microcontrollori, server, comunicazioni e middleware per la gestione di smart objects eterogenei

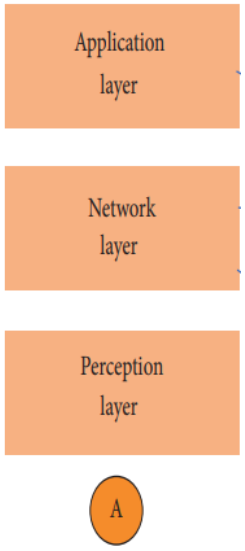
A fronte di questo elenco di caratteristiche abbiamo capito **che la tecnologia IoT non riguarda un unico campo di applicazione** e che **gli smart objects che interagiscono tra loro, normalmente, non sono dotati di risorse "potenti"**: essi sono spesso alimentati da batteria che ne limita la potenza dell'object, connessi **attraverso una rete wireless** e quindi con collegamenti che potrebbero essere soggetti a distorsione, inoltre, spesso potrebbe essere utile **un processamento dei dati che ne semplifica la gestione** per ottenere delle risposte immediate.
A fronte di ciò allora è bene stabilire una certa **architettura IoT** per gestire questi aspetti così da garantire che ogni parte possa lavorare al meglio.

Un'architettura distribuita è potenzialmente molto vantaggiosa: pensando alla pila TCP/IP (o ISO) su internet, è uno strumento potentissimo perché permette un livello di astrazione dei livelli inferiori che permette di semplificare un sistema complesso in un'organizzazione di quest'ultimo a livelli che

ne semplifica la complessità; suddividendo in livelli, infatti, ogni dispositivo opera all'interno di un certo livello, distaccandosi da altri dispositivi che si occuperanno di altri livelli: ad esempio, basti pensare al computer che è direttamente interfacciato al livello 5 della pila TCP, quello applicativo ed al router che si interfaccia al livello 3, quello di trasporto. A livelli differenti corrispondono quindi compiti differenti e questo riduce la complessità del sistema.

Anche in IoT si segue questo approccio "a livelli", possiamo osservare due principali proposte: la prima (A) è nata prima e prevede tre livelli di astrazione:

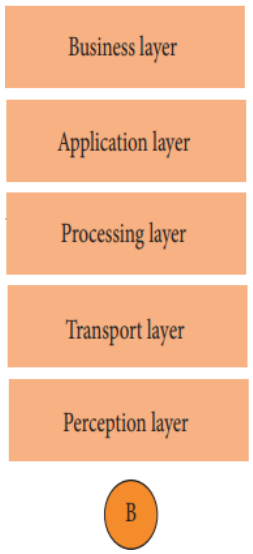
1. **Perception layer:** il livello di percezione è quello più vicino al mondo fisico in cui sono presenti i sensori, gli attuatori e gli strumenti di processing quali i microcontrollori; questi si occupano di captare i dati (dal mondo esterno verso il microprocessore -sensori- o viceversa -attuatori-) e di passarli al livello successivo.
2. **Network layer:** il livello di collegamento (che non è quello della pila OSI) è responsabile della connessione tra smart objects e tra lo smart object ed un server attraverso la rete.
3. **Application layer:** il livello applicativo, attraverso il livello di network, sfrutta le risorse a disposizione per il corretto funzionamento.



Questa architettura ha un livello di astrazione che **non è però molto dettagliato**: il livello di complessità potrebbe essere ancora diminuito integrando nuovi livelli.

Un'altra architettura IoT è la (B) che sfrutta cinque livelli di astrazione per il funzionamento:

1. **Perception layer:** è il livello analogo all'architettura precedentemente descritta.
2. **Transport layer:** il livello network della pila precedentemente viene suddiviso in due parti. La prima è il transport layer che si occupa di trasportare i dati misurati dal sensore, quindi deve poter interfacciarsi con i sensori e i dati che essi trasportano.
3. **Processing Layer:** la seconda parte in cui viene suddiviso il livello network è il processing layer. Questo livello non deve preoccuparsi di ciò che fanno i sensori: ma, piuttosto, si poggia al transport layer da cui prende semplicemente i dati che esso trasporta.
4. **Application Layer:** è il livello analogo all'architettura precedentemente descritta. E' il livello **middleware** che ovvia al problema di eterogeneità, rendendo una soluzione valida per tutte le tipologie di smart objects. E' importante un'astrazione simile vista la diversa natura possibile degli smart objects
5. **Business Layer:** livello per l'organizzazione dei processi e per gli scenari di profitto.

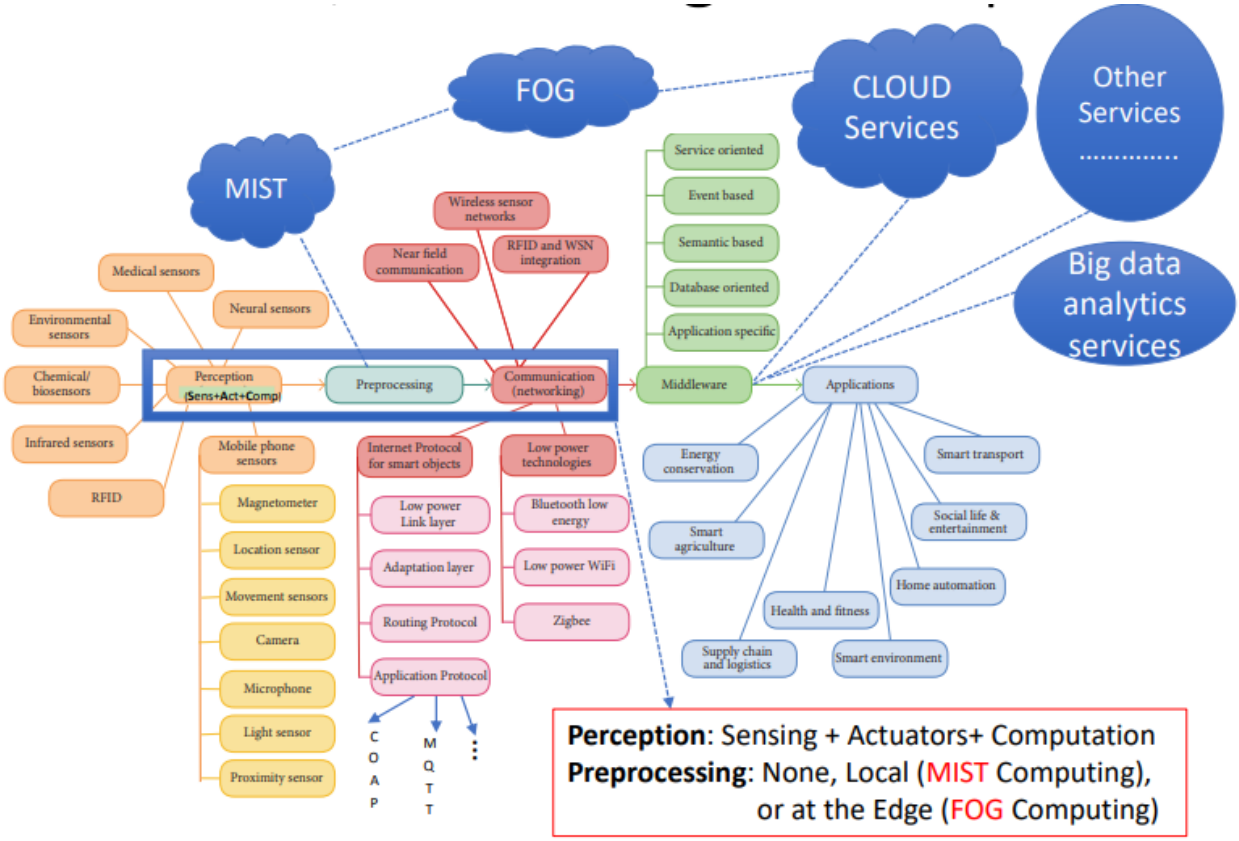


E' sempre bene scegliere il giusto numero di livelli per evitare che vi siano delle funzioni complesse in alcuni di essi (utilizzando tre livelli, ad esempio, si hanno livelli più complessi rispetto all'architettura a cinque livelli).

Un dispositivo Bluetooth nel mondo IoT sarebbe difficilmente integrabile perché l'IoT segue lo standard TCP/IP e la tecnologia Bluetooth è una soluzione non standard, per questo motivo potrebbero esserci alcuni problemi. E' dunque importante suddividere la complessità di un sistema e di rendere una soluzione standard.

Abbiamo parlato di processing e di preprocessing, il prossimo step è quello della **communication**

Architettura IoT, tecnologie e protocolli



Come si può notare nella foto, si ha una prima parte formata dal **perception**: in questo livello si ha l'insieme di sensori, di attuatori e di computatori (microcontrollori).
Dopo che i dati vengono letti, come abbiamo detto, essi devono essere salvati: vi è allora la fase di **preprocessing:** gli smart objects, come abbiamo detto, non hanno capacità di computazione elevate o grandi spazi di memoria. Queste risorse però possono essere associate ad un **cloud**, così facendo si rimedia alle risorse mancanti quali risorse per la memoria e per la computazione.
La struttura utilizzante il cloud è stata la prima utilizzata ma è ora in disuso: la connessione con il cloud richiedeva un determinato traffico ed introduceva una certa latenza (per la connessione da remoto) che dava una certa limitazione per alcuni smart objects che dovevano dare delle risposte real time. Parliamo allora di soluzioni alternative per il preprocessing: **Mist** e **Fog**.
Il **Fog** computing consiste nell'inserire delle risorse di computazione che non sono nel cloud ma sono più vicine possibili alla rete, che stanno nel bordo (edge) della rete. Ad esempio, nel gateway che dista al massimo 100m da dove è posizionato lo smart object.
Il **Mist** invece consiste nell'instaurare delle risorse per computazione situate nello smart object dotandolo della capacità di computazione locale, in modo da effettuare alcune azioni computazionali nel **perception,** senza doversi poggiare su alcun cloud o risorsa esterna.
La comunicazione, inoltre, prende la maggior parte della potenza erogata, quindi riducendo questa a funzioni che non sono le minime indispensabili al funzionamento dello smart object si riducono i consumi. Quando non è previsto né Mist né Fog allora si parla di Cloud.
In seguito, vi è la parte dedicata alla comunicazione (**communication)** ed in seguito quella di **middleware** che instaura i servizi per il livello di **processing**: si parla di servizi per ovviare alla eterogeneità dei dispositivi.
A valle di tutto vi è l'applicazione (livello di applicazione) che definisce in quale campo opera lo smart object (Energy, Health, Transport,…) .
Ogni livello aggiunge un'astrazione che permette al livello superiore di operare più facilmente.

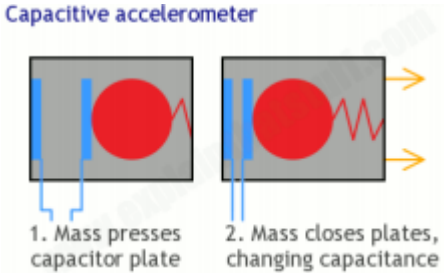
Il Perception Layer

Perception: Sensori

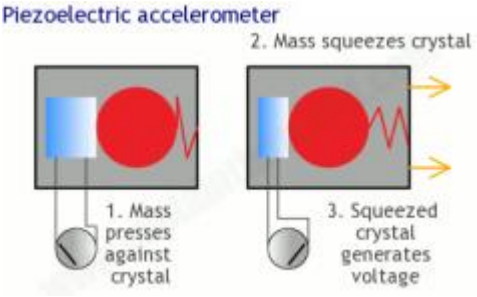
I sensori sono dei dispositivi che consentono la **context awereness**: la **consapevolezza del contesto in cui opera lo smart object**.
I sensori si possono suddividere in Embedded (integrati nel dispositivo, come quelli di uno smartphone) e quelli separati (disaccoppiati dal dispositivo e collegati esternamente ad esso).

Parliamo dei **sensori dello smartphone:**

1. **Accelerometro:** è un trasduttore che traduce il movimento in una grandezza rilevabile dal punto di vista elettrico: capta un'accelerazione e la quantifica digitalmente per passarla all'unità di computazione.
   Possiamo distinguere tre tipi di accelerometri:
   - **Accelerometro capacitivo**: è quello integrato nello smartphone; vi è una massa (in rosso) all'interno dell'accelerometro (in grigio), che poggia su un'armatura di un condensatore (in blu): muovendo lo smartphone la massa si muove e pressa sull'armatura del condensatore, facendo diminuire la distanza. Diminuendo la distanza **varia la capacità:** in questo modo misurando la variazione di capacità si misura automaticamente anche la variazione di velocità e quindi l'accelerazione. In figura è rappresentato per un solo asse ma in realtà esso lavora sui tre assi calcolando il vettore accelerazione sulle tre direzioni.

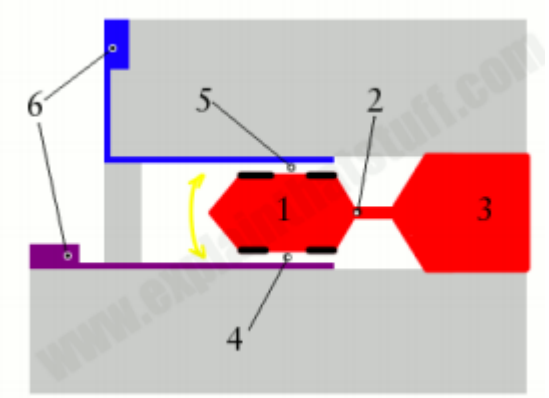


   - **Accelerometro Piezoelettrico:** il materiale piezoelettrico genera una tensione quando varia la sua forma genera una tensione che, misurandola, da un'idea sulla forza che è stata applicata ad esso.
     Una massa (sfera rossa) è collegata ad un corpo di materiale piezoelettrico (in azzurro). Quando questo corpo viene pressato dalla sfera rossa, si può misurare una determinata tensione, proporzionale alla forza che viene generata dalla massa. Quindi attraverso una grandezza elettrica si può misurare la forza impressa e dunque l'accelerazione che da essa ne scaturisce.



   - **Accelerometro a semiconduttore:** Prima di parlare di accelerometri a semiconduttore, facciamo una premessa: esistono i dispositivi MEMs che sono dei dispositivi meccanici-elettronici che vengono integrati all'interno del silicio: dove vi sono i transistor, i diodi, etc.. è possibile integrare anche dei trasduttori (sensori e attuatori) che assumono quindi dimensioni minime. Ricordiamo che negli smart objects "**size matters**": la dimensione è fondamentale, quindi integrando un sensore o un attuatore in un chip si risparmia nelle dimensioni e questo è molto vantaggioso; questo ha consentito l'integrazione di questi dispositivi all'interno di smartphone e di dispositivi in cui si ha poco spazio.

L’accelerometro a semiconduttore è così formato:



- Vi è un elettrodo (in rosso) che si può muovere su e giù (per minime distanze) che serve per misurare l’accelerazione nel suo asse.
- L’elettrodo può avvicinarsi a due lamelle e questo permette la misura dell’accelerazione, misurando la capacità tra i piatti che stanno sull’elettrodo (in nero nella figura) e la lamella superiore (blu) oppure tra i piatti inferiori (in nero)e la lamella inferiore (viola).
- L’elettrodo piccolo viene collegato ad un possibile elettrodo più grande (in rosso, più grande) così che l’accelerometro di possa collegare ad un circuito più grande.

Il vantaggio di questo accelerometro sta nel fatto che esso viene fabbricato in un singolo microchip, proprio come i transistors. Questo è vantaggioso non solo negli smartphone ma, soprattutto, anche negli smart watch poiché avendo meno spazio, con un singolo chip, si possono captare i movimenti dovuti all’attività fisica (o i semplici movimenti quotidiani) per monitorarli.

2. **Giroscopio:** capta l’orientamento del telefono (se orizzontale o verticale), ciò avviene grazie ad un cambiamento di capacità: un trasduttore modifica la capacità quando viene messa in movimento una massa in un determinato orientamento. [Simile all’accelerometro capacitivo]

3. **Camera e microfono:** due sensori più evoluti sono la camera e il microfono che servono per captare i segnali visivi e audio e sono molto sviluppati per le operazioni IoT (in una smart home si potrebbe interagire con l’ambiente attraverso la camera o il microfono del telefono).
   [10 anni fa in America è stato sfruttato il microfono per fini pubblicitari: in prossimità dei negozi venivano mandate delle frequenze che l’orecchio umano non riusciva a sentire ma che il telefono riusciva a captare, in questo modo, una volta captata l’onda sonora venivano mandati al telefono interessato una serie di pubblicità su delle determinate offerte del negozio.]

4. **Magnetometro:** serve per misurare il **campo magnetico** può essere utilizzato come un “compasso digitale” o in applicazioni in cui deve essere fatta la rilevazione di metallo circostante. È utile per **individuare il telefono in un ambiente chiuso** dove il GPS non funziona, migliore del segnale WiFi (nei posti affollati) o del Bluetooth (per grandi distanze) si fa una mappatura dei percorsi e il magnetometro riesce a registrare un’impronta magnetica (grazie ai pilastri e altre parti metalliche) e può essere utile nella navigazione indoor (al chiuso) se poi viene integrato con servizi Bluetooth, etc..

5. **GPS:** è una triangolazione con i satelliti che permettono di localizzare il telefono in una mappa.

6. **Sensore luminosità:** sensore per captare la luce esterna, viene utilizzato, ad esempio, per adattare la luce dello schermo alla luce esterna.

7. **Sensore di prossimità:** usa dei LED ad infrarossi che, con il rimbalzo dei raggi infrarossi, riesce a stimare la distanza da un oggetto (viene utilizzato, ad esempio, quando il telefono viene portato all'orecchio per spegnere temporaneamente lo schermo, fintanto che il telefono resta attaccato all'orecchio).

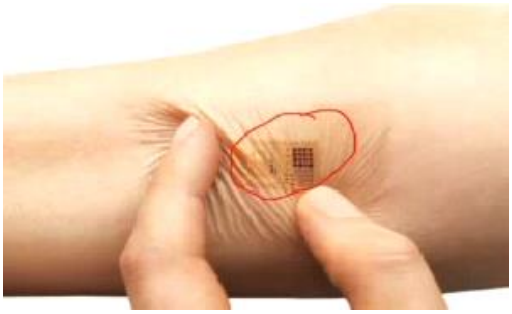
8. **Termometro, Barometro, Sensore di umidità:** sono dei sensori che non hanno tutti i telefoni (sono integrati, ad esempio, sul Samsung Galaxy S4) che servono rispettivamente per misurare la temperatura, la pressione e l'umidità circostante.

Con i sensori dello smartphone si possono creare delle smart application, quale ad esempio l'activity detection che permette, attraverso i sensori ed un algoritmo di machine learning, di capire un'attività fisica: Corsa, Bici, Salita o discesa delle scale…
Un'applicazione simile consente allo smart phone di capire il contesto esterno (context awereness) e questo permette di creare delle **context awereness application:** applicazioni basate sul contesto esterno, con la possibilità di attivare determinati servizi a seconda dell'ambiente circostanze (se si è in bici, a casa, a correre fuori o in auto…).
Concludiamo affermando che **i sensori servono per avere un approccio fisico con il mondo esterno.**
I sensori dello smartphone sono tutti **sensori embedded**: che fanno parte dello smartphone stesso.

Esistono, tuttavia, anche sensori che non lavorano in maniera integrata ma **esternamente dallo smart object.** Facciamo alcuni esempi:

1. **Sensori di tipo medico:** sono utilizzati, ovviamente, nell'ambito medico.

   Alcuni di essi sono i **werable sensors** che vengono integrati in sistemi werable (come gli smart watch) per health monitoring (battito cardiaco, pressione, elettrocardiogramma, rate di respirazione, temperatura…) o per altri servizi.

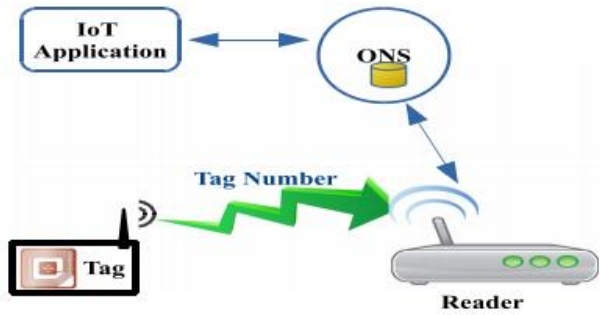
   Un altro esempio sono i **Monitoring patches** che sono una sorta di "tatuaggio" che si appiccica sulla pelle ed è in grado di monitorare una serie di valori per alcuni giorni.

   Un altro esempio sono i **sensori neurali** che permettono di fare un elettroencefalogramma: i neuroni comunicano elettricamente e quindi generano, seppur in piccolo, un determinato campo elettrico; questo campo genera delle frequenze che si possono suddividere: α, β, ϑ… Ed in base a queste frequenze si può capire lo stato cerebrale: si riesce a capire se una persona è calma, preoccupata, etc.. Con questi sensori si può misurare, ad esempio, il livello di stress per migliorare la vita dei pazienti.

2. **Sensori ambientali e chimici:** misurano parametri fisici come temperatura, pressione, umidità o inquinamento. Vi sono applicazioni, ad esempio, che riescono a rilevare sostanze biochimiche.
   Sono stati inventati sei sensori molto sofisticati che vengono chiamati **"naso elettronico" [e-nose]** e **"lingua elettronica"[e-tongue].**
   Questi sensori possono essere utilizzati per monitorare **l'inquinamento delle smart city** o nella **smart kitchen** o **nei prodotti di agricoltura** per testare il cibo**.**

3. **Sensori ad identificazione di radio frequenza (RFID):** sono praticamente delle etichette (Tag) che sfruttano il campo magnetico per essere identificati. [Vengono utilizzati spesso nei vestiti per evitarne il furto].
   Esistono due tipologie di RFID: **passivo** o **attivo**:
   - **Attivo:** è alimentato
   - **Passivo:** non è alimentato e sfrutta il campo magnetico generato dal lettore (reader) e sfrutta l'energia di tale campo per trasmettere un codice al lettore e farsi identificare.



Perception: Attuatori

Gli attuatori sono quella parte di sistema che permette di modificare l'ambiente esterno: è un trasduttore che trasduce il segnale elettrico proveniente da un sistema in un segnale utile al mondo esterno (ad esempio lo trasforma in movimento, luce, suono, etc…)
Possiamo suddividerli in base al principio fisico che sfruttano in **tre categorie:**

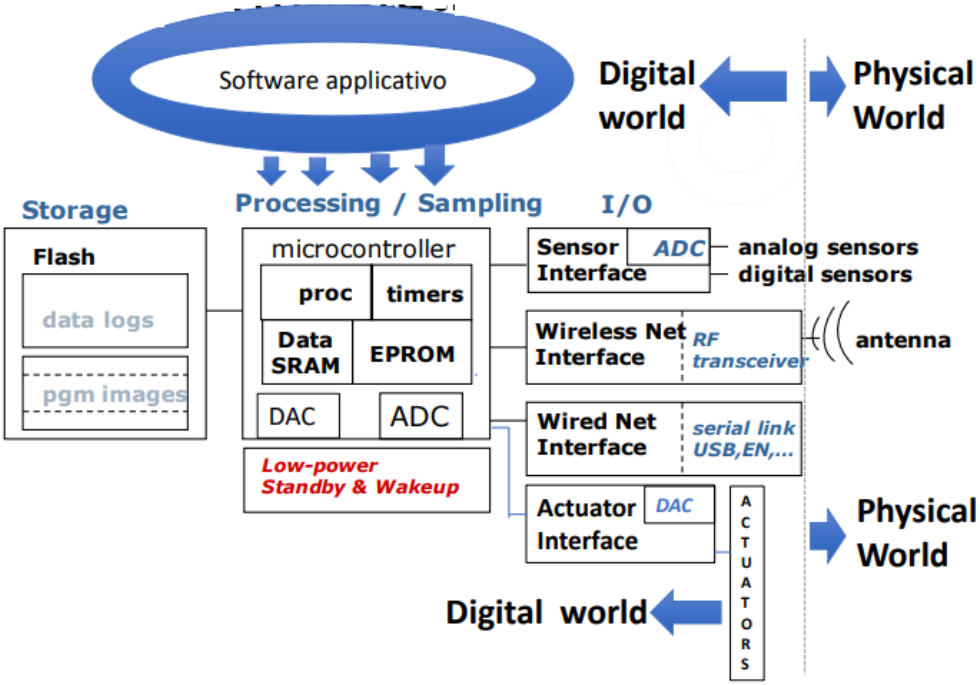
1. **Attuatori elettrici:** modificano l'ambiente esterno (compiendo un movimento, generando un suono o un segnale luminoso, etc..) sfruttando un segnale elettrico.
2. **Attuatori idraulici:** attivano un movimento meccanico (o comunque modificano in qualche modo l'ambiente) sfruttando la potenza dei fluidi
3. **Attuatori pneumatici:** modificano l'ambiente esterno grazie ad una pressione generata.

Perception: Computation

Parlando del lato di computazione del primo livello della pila (Perception) parliamo dei **microcontrollori** che, collegando ad esso dei sensori in input per avere la percezione dei segnali dal mondo esterno e a degli attuatori in output per modificare l'ambiente esterno a seconda dei risultati ottenuti, e collegando ad esso un determinato software diventa praticamente uno **smart object.**

Il microcontrollore è il "cervello" del sistema; vi sono molte piattaforme per lo sviluppo IoT che integrano vari microcontrollori, ad esempio Arduino.

Esso è formato da un processore, dei timers, può avere delle memorie SRAM e delle memorie EPROM e dei convertitori Analogici/Digitali (ADC) o Digitali/Analogici (DAC). A queste parti è anche agganciata una flash memory dove è possibile memorizzare i dati. Il microcontrollore, attraverso le interfacce, si collega a dei sensori digitali/analogici, ad un antenna (attraverso l'interfaccia Wireless Net), a degli attuatori, etc…



Il microcontrollore deve essere usabile: per essere usabile ha un sistema operativo che può essere più semplice (come quello per sistemi embedded) o più complesso, come i sistemi operativi basati su Linux (ma comunque semplificati).

Il Preprocessing

Il livello successivo al Perception Layer è quello del **preprocessing.**

L'IoT è formato da sistemi che connettono smart object di natura eterogenea, il fatto che questi siano eterogenei induce a pensare che esistono sistemi che inglobano altri sistemi più piccoli con l'obiettivo di raggiungere un certo **livello di smartness**.

Chiaramente, nel complesso, un sistema così descritto è piuttosto articolato, formato da parti addette a compiti diversi che, chiaramente, generano e manipolano dati di diversa natura. Aumentando il numero di dispositivi, inoltre, aumenta anche il numero di dati che vengono generati e/o manipolati, quindi è quasi un controsenso dire che si hanno poche risorse di memoria e grandi quantità di dati da memorizzare.

Si necessita dunque di una tecnologia che consenta di superare i limiti fisici imposti dalle risorse (native) limitate degli smart objects.

Cloud Computing

La prima tecnologia di cui parliamo è quella del **cloud computing,** nata inizialmente con lo scopo di risolvere problematiche, principalmente per le aziende, per la gestione e la manutenzione dei datacenters o per la disaster recovery (perdita di dati in seguito ad incendi, allagamenti o altre catastrofi ai datacenters): venivano affittati dei cloud per preservare i dati dei datacenter, attraverso l'utilizzo delle virtual machines: un'azienda avente un "datacenter fisico" ha a disposizione una macchina virtuale che si configuri con i software di sistema e quelli applicativi e con le risorse (CPU, memoria) che ha il "datacenter fisico" con lo scopo di **virtualizzare il datacenter**, in maniera tale che questo non può cadere vittima di catastrofi; veniva inoltre garantito che il sistema non sarebbe andato down, mantenendo il funzionamento originale (del datacenter fisico).
Il cloud computing ha dovuto affrontare parecchie difficoltà: all'aumentare del numero di istanze (di macchine virtuali) che giravano su un'unica macchina andavano a peggiorare le prestazioni; si andò dunque incontro al **cloud computing distribuito** (decentralizzando il cloud si evitava che una macchina venisse "sovraccaricata" con troppe virtual machine al suo interno). In questo modo si è trovata una soluzione scalabile così che all'aumentare delle macchine virtuali non vi era un calo di prestazioni.
Questa tencologia ha portato ad un trasferimento dei datacenter fisici in datacenter basati sul cloud computing.
Vi è un modello di accesso alle reti on-demand per la condivisione di risorse per la computazione come le reti, i server, le applicazioni, etc..
I servizi cloud consentono agli individui e alle compagnie di usare in remoto software da terze parti e componenti hardware

Parliamo di cloud computing perché il nostro obiettivo è quello di ampliare le risorse dello smart object, ed **il cloud computing potrebbe venire in contro alla nostra esigenza**, in modo da assegnare ad uno smart object una capacità di memorizzazione maggiore e di assegnare delle risorse hardware migliori (appartenenti alla macchina fisica che ospita la macchina virtuale sul quale si poggia lo smart object una volta connesso al cloud).
E' fondamentale pensare di utilizzare un cloud per un grande sistema IoT: i sistemi complessi possono memorizzare, salvare i dati ed estrarne l'intelligenza da essi utilizzando delle grandi risorse che potrebbe offrire il cloud computing

Il cloud computing potrebbe risolvere molti problemi, tuttavia l'utilizzo di quest'ultimo potrebbe **introdurre alcuni problemi.**

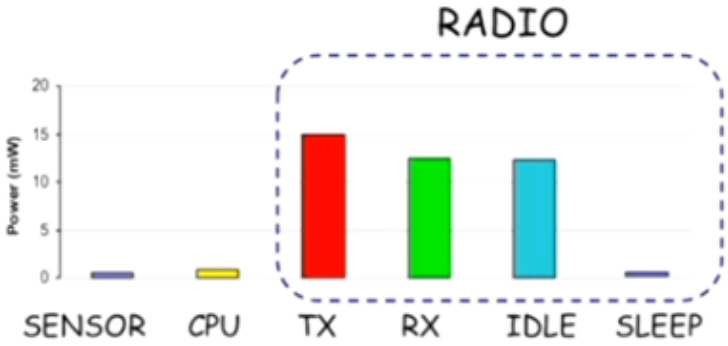
Problematiche:

1. **Mobility:** la maggior parte degli smart objects sono mobili: cambiano posizione continuamente e comunicano quando essi sono in movimento. Se la connessione venisse effettuata con un cloud computing potrebbe essere difficile mantenere una connessione praticamente permanente in differenti zone.

2. **Reliable and real time actuation:** la comunicazione con il cloud per scambiare i dati prendono tempo, introducendo quella che viene chiamata **latenza:** ritardo temporale che nasce dal trasferimento dei dati da un mittente al destinatario e viceversa. Vi possono essere delle applicazioni **Latency sensitive**, ovvero sensibili alla latenza, che non supportano un elevata latenza. La **comunicazione con il cloud** è una comunicazione che potrebbe essere persa (link wireless) o che potrebbe andare più o meno veloce introducendo più o meno latenza.
   Inoltre, anche se i cloud sono nati per memorizzare una grande quantità di dati e il mondo IoT produce una quantità enorme di dati, il problema non interessa più la memorizzazione di questi ma si focalizza sul trasporto: troppi dati che viaggiano in una rete (o in un collegamento quale potrebbe essere quello tra uno smart object come una smart city ed un cloud), questo porta ad una **saturazione della rete che farebbe aumentare la latenza**.

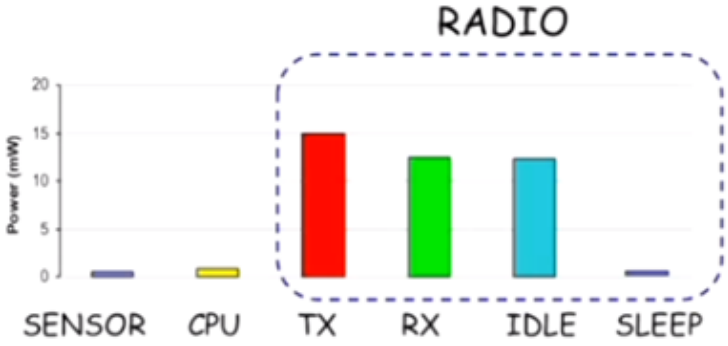
3. **Scalability:** Il sistema deve essere scalabile: aumentando i dispositivi IoT e le applicazioni, aumentano anche i dati e le risorse da associare. Quindi **si deve avere una buona scalabilità non solo dei cloud ma anche della rete:** I cloud devono scalare aumentando la capacità di memorizzazione dei dati e le risorse virtualizzate associate al sistema IoT e la rete deve garantire una certa quantità di banda (che tende ad aumentare all'aumentare dei dati provenienti da più dispositivi) per evitare, come abbiamo detto, l'eventuale latenza

4. **Power constraints:** i dispositivi IoT sono di tipo **power costraints**: molti dispositivi IoT sono alimentati da batteria, quindi la potenza (direttamente collegata alla durata della batteria visto che maggiore è la potenza erogata e minore è la durata della batteria) **è un aspetto fondamentale** (ancor più di quanto lo possano essere gli altri parametri già visti)**. La comunicazione fa dissipare potenza** (come abbiamo già detto, la maggior parte di potenza di un oggetto smart viene erogata per la comunicazione); dal seguente grafico possiamo notare l'analisi su un nodo IoT della potenza da esso sfruttata [Supponiamo che tale nodo smart sia un semplice microcontrollore con a bordo dei sensori e degli attuatori]:



La parte radio, quella addetta alla comunicazione, è quella che spreca più potenza: avremo uno spreco maggiore per la trasmissione (TX), uno spreco leggermente minore (ma comunque alta) per la trasmissione e l'Idle (stato in cui il dispositivo resta on ma non riceve e non trasmette) e lo spreco è minimo quando la parte radio è staccata.

Facendo riferimento quindi ad un modello architetturale in cui il nodo smart, attraverso dei sensori, acquisisce i dati e li trasmette direttamente al cloud significa che **la sezione di comunicazione è sempre attiva e quindi si ha una grande dissipazione di potenza.**
La soluzione a questo problema la possiamo ottenere osservando nuovamente il grafico:



La CPU, quando manda in esecuzione un programma, dissipa una potenza trascurabile rispetto alla sezione di comunicazione, quindi l'idea è quella di fare delle operazioni sui dati attraverso la CPU, eseguendo quello che viene chiamato **preprocessing**, effettuando delle operazioni di pre-processamento: filtrando i dati utili e separandoli da quelli non utili, effettuando delle decisioni immediate (che spettano alla CPU) evitando di trasmettere eventuali dati in più. Quindi **investendo in computazione si spreca meno potenza, ma investire in computazione significa risparmiare in comunicazione perché se alcune decisioni vengono prese in locale (dalla CPU del microcontrollore) sul nodo, piuttosto che sul cloud si risparmia tanta potenza di comunicazione**
[NB: questo principio è fondamentale e deve essere ben chiaro ai fini dell'esame -> probabile domanda.**]**

Quindi, come abbiamo visto, il cloud computing sarebbe un vantaggio, apparentemente, poiché esso porterebbe a maggiori risorse da un punto di vista di memorizzazione e computazionale, tuttavia, **esso introduce molti problemi legati ai tempi di risposta e ai consumi in potenza**.

Inoltre, in uno scenario futuro, lo sviluppo di massa dell'IoT porterebbe ad una crisi del cloud computing: ci sarebbero troppi dati che saturerebbe, oltre che la rete, anche la memorizzazione e ci sarebbero così tanti dispositivi che non sarebbe possibile associare le risorse computazionali in maniera efficiente.

Il cuore del problema sta nel fatto che **l'architettura cloud è di natura centralizzata** che, seppur scalabile e remoto, crea i problemi che abbiamo menzionato.

Paradigma del fluid computing

Da circa 5/6 anni a questa parte è nato un **nuovo paradigma** che prende il nome di **fluid computing** che fa riferimento ai sistemi distribuiti, ovvero al paradigma del **distributed computing,** con lo scopo di utilizzare il cloud come risorsa e non come "ostacolo"**: si è pensato di rimuovere il vincolo dei cloud datacenter che sono virtuali ma "fissi" in posti situati in qualche parte nel mondo, immaginando di avere un cloud che si potesse spostare, avvicinandosi il più possibile verso la provenienza dei dati (avere quindi questa "fluidità" del cloud): l'idea è quella di utilizzare un approccio che "miscela" il Fog Computing e il Mist Computing.**

Ricorda:
Il **Fog** computing consiste nell'inserire delle risorse di computazione che non sono nel cloud ma sono più vicine possibili alla rete, che stanno nel bordo (edge) della rete. Ad esempio, nel gateway che dista al massimo 100m da dove è posizionato lo smart object.
Il **Mist** invece consiste nell'instaurare delle risorse per computazione situate nello smart object dotandolo della capacità di computazione locale, in modo da effettuare alcune azioni computazionali nel **perception,** senza doversi poggiare su alcun cloud o risorsa esterna.

La **metafora del meteo:** la parola "cloud" allude al fatto di pensare la capacità di storage e di computing virtuale, come se fossero delle nuvole, lontane e distaccate dal terreno: non è importante sapere dove la "nuvola" è situata ma importa ricevere i servizi di cui si necessita. Possiamo allora parlare di fog: nebbia e di mist: "nebbiolina", nebbia fievole; la differenza tra queste è che le nuvole sono dense di pioggia e stanno in alto, lontano da terra. La nebbia sta sotto le nuvole e sono formate da lievi goccioline, la nebbiolina sta praticamente al suolo. Inizialmente è nato il cloud, poi il fog computing e poi il mist computing.
La metafora vuole far capire che:

- **I cloud sono molto lontani dai dispositivi però possono contenere molti dati** (l'acqua della metafora allude ai dati);
- **il fog computing è un'infrastruttura che sta in mezzo tra i dispositivi e il cloud: mantiene alcune caratteristiche del cloud ma è più prossima ai nodi che vi si connettono** (sta al "bordo" della rete), ovvero sta più vicina al punto in cui vengono generati i dati. Un nodo di fog computing non può avere la quantità di storage e di computing di un cloud.
- **Il mist computing, rispetto al fog computing, è più vicina ai dispositivi IoT ma ha ancor meno capacità di processamento e di storage**

Il paradigma **fluid computing** vuole spiegare che è un riferimento alla modalità di computing distribuito dai grossi datacenter ai singoli dispositivi. La forza di questo concetto è **l'utilizzo di risorse appropriate a seconda del livello**: mettere tutto su cloud non è una soluzione appropriata perché i tempi di risposta introducono latenza, mettendo i dati più vicini si ha una minore latenza (ma con minore capacità).

I livelli cloud fog e mist sono totalmente differenti ognuno con dei vantaggi e svantaggi:

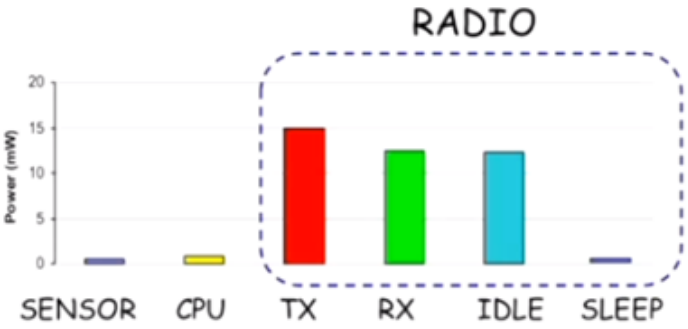
	Cloud computing	**Fog computing**	**Mist computing**
Vantaggi	Grandi capacità di storage e di computing	Livello "al bordo di rete", più vicino ai dispositivi (latenza minore). Minori costi perché non è un datacenter (posso usarne di più per aumentarne le capacità)	Risiede direttamente sui nodi quindi si trovano proprio dove provengono i dati.
Svantaggi	Problemi nella comunicazione dovuti alla latenza e ai consumi	Minore capacità di storage e computing	Capacità di storage e computing notevolmente ritotte poiché i nodi non hanno molte risorse di storage e computazionali

Nonostante le differenze e i pregi e i difetti che esistono, le tre paradigma possono cooperare per ricreare un ambiente che sia il migliore possibile.

Il **fog computing** mantiene le caratteristiche del cloud, seppur con risorse minori; il paradigma del fog computing prevede l'inserimento di una capacità di computazione nella connessione tra i sensori dei dispostivi ed i server cloud. Questa capacità sta tipicamente in dei gateway che collegano tutti i sensori e gestiscono la connettività.
Con questo funzionamento si diminuisce la latenza: un esempio di scenario per il fog computing sono i veicoli con guida automatizzata, in cui i comandi in real time stanno alla base di tutto per evitare incidenti o disastri. Il fog computing è più adatto perché sta più "vicino" al veicolo.

Esempio di fog computing: Il fatto di dire che nel fog computing spostiamo la capacità di computazione e di storage "ai bordi della rete" significa, praticamente, utilizzare degli **smart gateway** che possono essere messi in prossimità dei sistemi smart: ad esempio, se si vuole utilizzare la tecnologia IoT sui veicoli, i semafori possono essere la giusta locazione degli smart gateway, in questo modo essi stanno praticamente in prossimità delle automobili e i dati possono essere computati con poca latenza e con delle buone risorse di computazione e di storage (non al pari di quelle del cloud ma comunque buone risorse). Un altro esempio nell'ambito della smart grid potrebbe essere messo un smart gateway tra la rete e l'utente per bilanciare il carico tra domanda e offerta di energia.

Il **mist computing** incrementa la potenza di computazione sui microcontrollori dei dispositivi. Questa potenza deriva da dei microchip embedded sui dispositivi.
Con l'avanzare della tecnologia, i nodi smart hanno consumato sempre meno potenza (grazie a dei chip low power), questo significa che è stato possibile aumentare la capacità di computing dei nodi. Questo significa che si sono sviluppati dei nodi con una **buona capacità di calcolo**, quindi dato che vi sono maggiori risorse è utile sfruttarle.



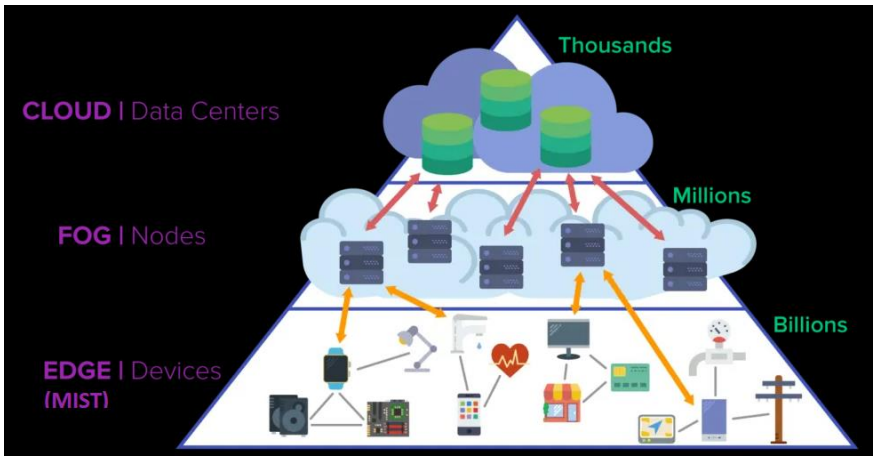
Prendendo nuovamente in considerazione il seguente grafico, possiamo notare che avendo una CPU con maggiore capacità di calcolo, quindi si possono fare più lavori sui dati captati dai sensori. Lavorando su essi (**preprocessing:** filtraggio, selezione…), vengono trasferiti verso il livello più alto (il fog) meno dati: quindi si ha meno traffico e meno spreco di energia.
Investendo sulla computazione con microcontrollori che lavorano con CPU con maggiori potenze di calcolo ma che lavorano a bassi consumi si hanno dei miglioramenti notevoli, quindi ha senso di parlare di mist computing.

Quindi se si utilizzano tutti e tre i paradigmi e si utilizzano le giuste risorse nei giusti ambiti si ottiene un sistema "omogeneo" che riesce a lavorare con continuità e nel migliore dei modi. Questa gerarchia Cloud-Fog-Mist computing potenzia lo spettro di domini applicativi che l'Iot può ricoprire poiché in questo modo si riescono a ricoprire più campi che richiedono diverse soluzioni (le soluzioni real time per i veicoli intelligenti piuttosto che le soluzioni che interessa grandi dati per altre applicazioni) e potenzia notevolmente il funzionamento dei dispositivi IoT.

Vantaggi di Fog e Mist Computing sul Cloud Computing.

Ricapitolando, grazie alla loro prossimità Mist e Fog Computing garantiscono una latenza minima che offrono ritardi minori e performance migliori, specialmente per quei dispositivi che richiedono una risposta real time.
Abbiamo detto che il Fog Computing si comporta similmente al Cloud Computing, tuttavia è bene specificare che c'è un'immensa differenza tra la potenza computazionale e di memorizzazione dei cloud e quella dei singoli gateway dei fog computing. Tuttavia, essendo la fog computing una soluzione più economica, allestendo più gateway si riesce comunque ad ottenere una buona potenza di computazione e di computazione. Replicando dei Gateway si ottiene la risoluzione proposta dal cloud computing (che di fatto l'ha reso un mezzo potente): quella del **disaster recovery**

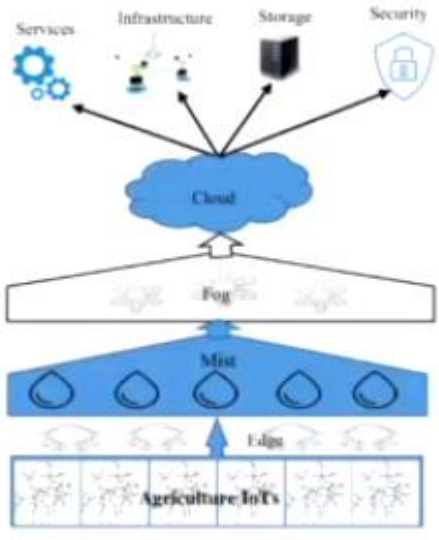
Non tutti i nodi al livello dei dispositivi IoT si possono sfruttare per il Mist Computing: alcuni nodi che sono equipaggiati meglio possono adoperare il mist computing per sé e per altri dispositivi che non supportano delle grandi unità computazionali [in figura si possono notare alcuni nodi che non si collegano direttamente ai nodi fog]
Il gateway per il Fog Computing, che si deve posizionare in una distanza intermedia tra cloud e mist, potremmo vederlo implementato, ad esempio, in una **cell tower** per tecnologia 3G/4G. Grazie al 5G si potrebbero implementare più gateway grazie al sistema di celle delle cell tower.



Vantaggi sul cloud computing:

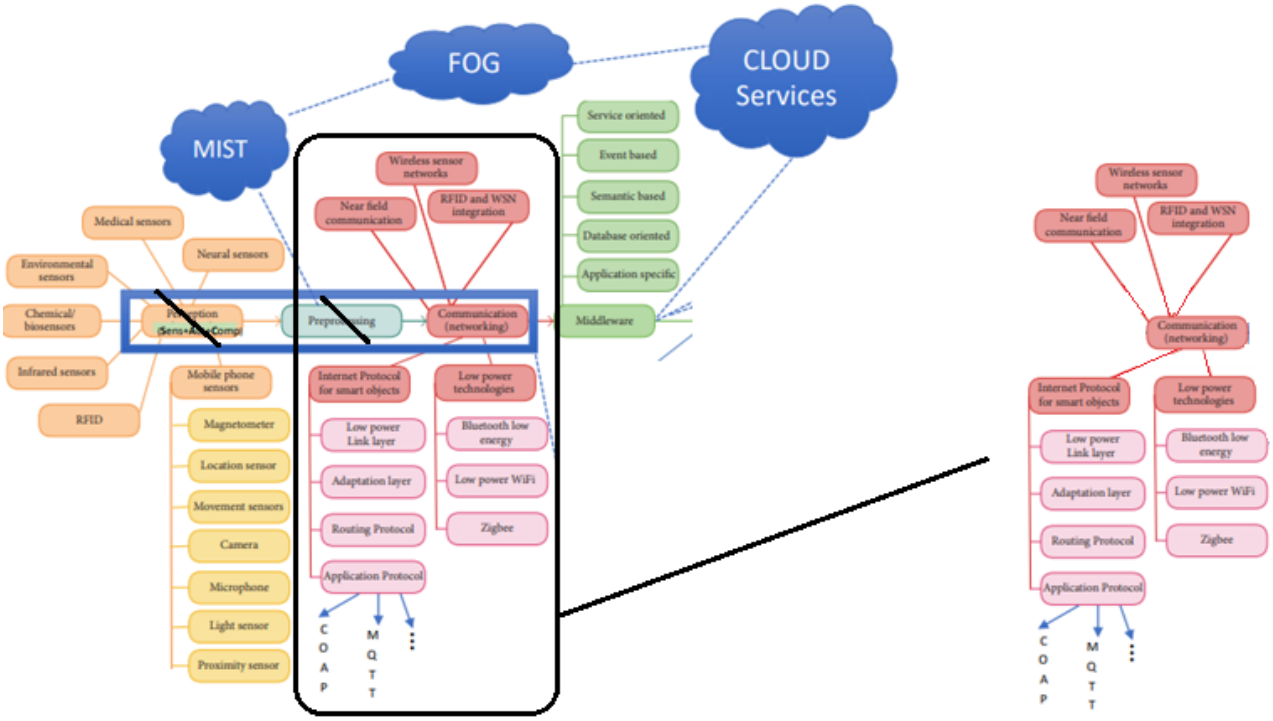
- **Locazione:** le risorse sono posizionate vicino allo smart object e diminuisce la latenza
- **Distribuzione:** gli smart gateway possono diventare dei "micro datacenter" ed è facile distribuirli per i bassi costi
- **Scalabilità:** è più semplice scalare i gateway di un fog computing piuttosto che un datacenter poiché costa meno. L'architettura proposta è quindi scalabile e si approccia bene alla dinamicità dello sviluppo dell'IoT.
- **Resilienza:** si possono replicare servizi utilizzando più gateway per evitare che nascano problematiche dovute a catastrofi (Disaster Recovery), la resilienza aumenta grazie alla densità di dispositivi
- **Supporto alla mobilità:** dato che i gateway sono sparsi dappertutto la mobilità non è un problema
- **Real-Time:** vista la poca latenza, il fog computing (insieme al mist computing) sono adatti per servizi real-time.
- **Standardizzazione:** utilizzando dei gateway standard questa soluzione può essere standardizzata
- **On the fly analysis:** ciò che veniva trasportato verso il cloud per le analisi sui dati e poi si aspettava come risposta può essere fatto, in piccolo, in locale: vengono fatte delle "analisi al volo" (dipende dal tipo di analisi), si può, in altre parole, introdurre il preprocessing per trasferire dei dati più semplici al cloud.

Il fog computing ed il mist computing hanno il potenziale per incrementare la performance complessiva delle applicazioni IoT poiché è prevista l'esecuzione di servizi di alto livello, che prima erano dedicate al cloud, in locale rendendo così il trasporto dell'informazione meno "pesante".



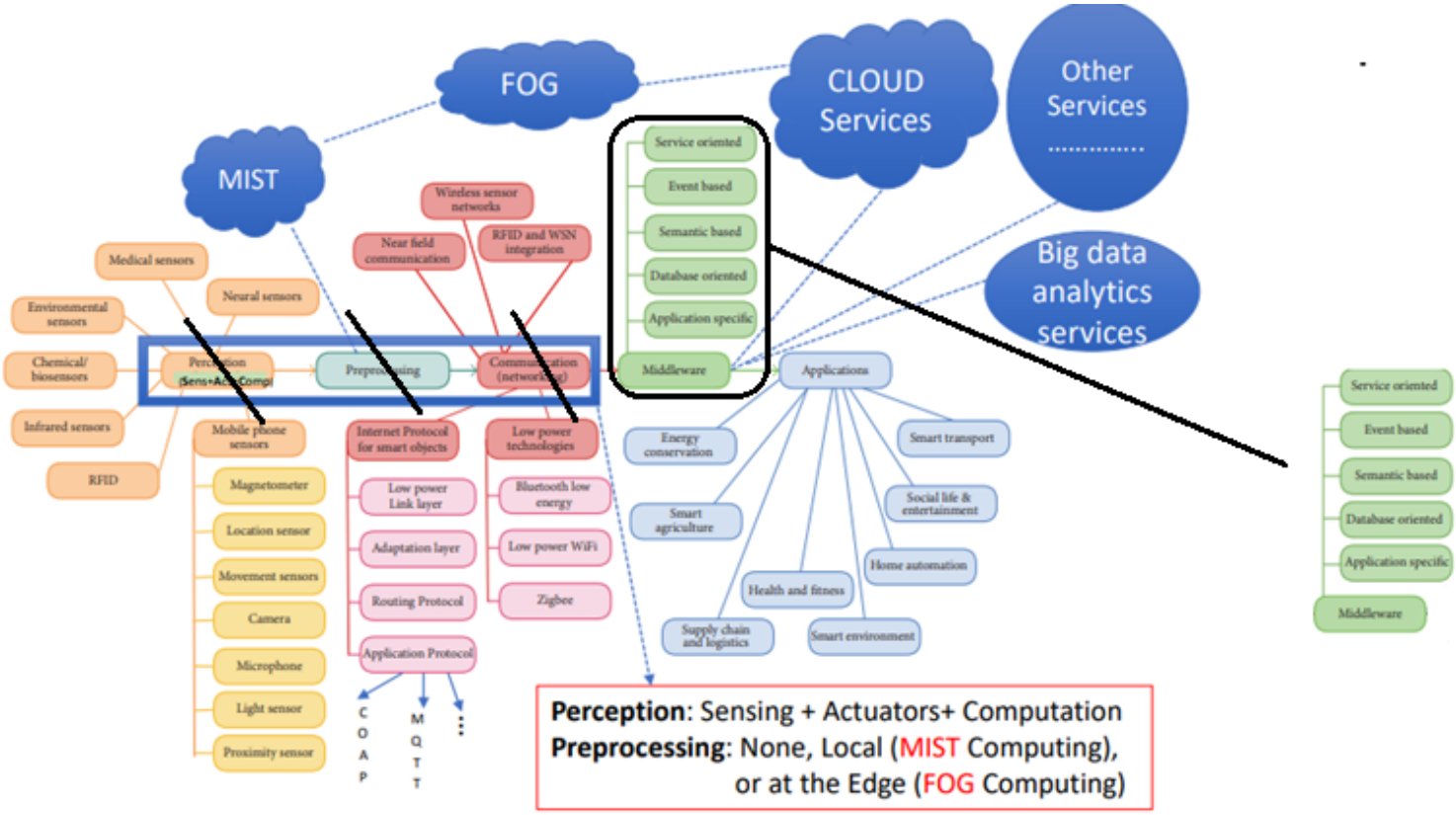
La Communication

Abbiamo parlato di processing, parlando dei vari sensori attuatori e commutatori esistenti e di preprocessing, introducendo il paradigma fluid computing,; Adesso il prossimo step da affrontare è quello della **communication.** Tuttavia, parleremo del livello di comunicazione successivamente, trattandolo in dettaglio.



Il Middleware

Dopo aver parlato di processing, preprocessing, e di communication parliamo del livello **middleware** il prossimo step è quello della **communication**



Il middleware è uno strato che si occupa di eseguire alcune funzioni per rendere una tecnologia IoT "standard", in modo tale che diversi smart objects possono comunicare e interagire a prescindere dalla loro differente natura.

Un middleware viene normalmente utilizzato quando si hanno componenti di un sistemo che sono eterogenei: questa eterogeneità introduce una notevole complessità; il middleware permette di astrarre questa difficoltà, mascherando le differenze.

L'IoT è un insieme di applicazioni, oggetti e protocolli eterogenei; si immagini un sistema IoT per smart home: utilizzando un apposito protocollo di comunicazione. Questa smart home può far parte ad un condominio smart che potrebbe utilizzare componenti diversi (sensori, attuatori, …) e diversi protocolli di comunicazioni. A sua volta il condominio potrebbe prender parte di un quartiere smart ed anch'esso potrebbe avere componenti e protocolli differenti dal condominio. Questo unico sistema IoT deve chiaramente cooperare per aggiungere smartness e per ottenere dei risultati. Per la cooperazione di questi sistemi con opportune applicazioni si deve far fronte all'eterogeneità dei sistemi, per sollevare il sistema da questo problema **si utilizza il middleware che instaura un livello di astrazione adatto per consentire agli smart objects di lavorare nascondendo i dettagli di questi ultimi e dei protocolli utilizzati, ponendosi come "ponte software" tra le applicazioni e gli oggetti**.

Il middleware nasconde la complessità dei livelli sottostanti astraendo l'hardware e fornendo servizi come le Application Programming Interface API (interfacce di programmazione) per la comunicazione, per la gestione dei dati, per la computazione, per la sicurezza e per la privacy

Quando è nata la tecnologia IoT non esisteva il middleware; questo è nato successivamente, ad oggi ne esistono molti open source ed altri brandizzati. Un buon middleware è quello che riesce a gestire una vasta gamma di eterogeneità di sensori e protocolli e che riesce a prevedere la gestione di servizi futuri, facilitandone lo sviluppo e l'utilizzo.

I caratteri principali del middleware sono **interoperabilità** e **astrazione;**
Tipicamente si parla di **interoperabilità** su tre livelli:

1. **Interoperabilità network (Network interoperability):** capacità di esporre interfacce con protocolli eterogenei attraverso API, nascondendo al programmatore il dettaglio del protocollo utilizzato.
2. **Interoperabilità sintattica (Syntatic interoperability):** capacità di rendere trasparente la differenza della natura dei dati
3. **Interoperabilità semantica (Semantic interoperabilty):** si riesce ad esporre una descrizione dell'oggetto prescindendo dalla sua natura, esponendo ciò che esso è in grado di fare (i suoi servizi): si può utilizzare un sensore di temperatura prescindendo da come esso sia formato, fornendo all'applicazione (strato superiore) il dato relativo alla temperatura, questo è possibile grazie al middleware che astrae ciò che sta sotto attraverso l'uso di API adatte.

Caratteristiche di un Middleware

Device discovery and management (scoperta e gestione dei dispositivi e dei servizi): In un ambito IoT questa è un'importante caratteristica per il middleware: esso **abilita i dispositivi ad essere coscienti (Aware) della presenza di altri smart objects nell'ambiente circostante e dei servizi che essi offrono**, questo è possibile grazie al middleware che attraverso un'API mostra i servizi e i dispositivi circostanti. In un ambito IoT in continua evoluzione (dinamica), questa caratteristica è preponderante.

Scalabilità: il sistema può crescere quindi è bene che il middleware possa essere scalabile per trattare più dispositivi possibili

Big data and analytics: approcciandoci al mondo IoT, deve avere a che fare con grandi quantità di dati e deve essere in grado di gestirli e analizzarli.

Sicurezza e privacy: devono essere garantite la privacy per evitare che qualcuno possa spiare le azioni che vengono eseguite e la sicurezza per evitare che qualcuno possa manipolare gli smart objects che si possiedono

Cloud services: deve essere chiaramente in grado di offrire servizi per i cloud poiché sono fondamentali per un sistema IoT.

Il middleware riesce dunque a virtualizzare l'hardware (sensori, attuatori) e il modo di esporsi ad esso (protocolli e funzionamenti) per garantire un funzionamento delle applicazioni più semplice, sollevando queste dalla gestione di apparati di diversa natura (con tecnologie, hardware e dati differenti)

Big Data e Big Data Analytics

Il sistema IoT è un sistema pervasivo in più ambiti; esso è formato da tanti dispositivi che generano dati e da applicazioni che vengono detti data driven. I dati sono quindi la maggiore risorsa su internet, tant'è che con l'aumento dei dispositivi connessi aumentano anche i dati che vanno a confluire nella rete.

Il problema della memorizzazione e dell'analisi dei dati è quindi di fondamentale importanza. Parleremo allora di **big data: set di una quantità enorme di dati**, fondamentali per il funzionamento delle applicazioni IoT. Ciò che rende importante e rilevante l'utilizzo dei big data è la **possibilità di analizzare e di trarre dunque conoscenza dai dati degli utenti**, il che si traduce essere un enorme vantaggio sulle strategie di mercato.

I big data esistono da tempo: social media, acquisti online, organizzazione di viaggi.. tutte queste operazioni generano dati, ovvero generano big data (insieme dei dati); questi dati quindi nascono a prescindere dall'esistenza dell'IoT, tuttavia in questa sede vogliamo focalizzare la nostra attenzione sui big data creati e manipolati all'IoT.

Nell'arco di cinque anni, si stima che i dispositivi IoT raddoppieranno e con essi si stima un aumento dei dati, fino alla produzione di 20 zettabyte ogni anno (ZB = 10^{21}). Dal 2025 si stima che si connetterà circa 4800 volte al giorno (dieci volte rispetto ad oggi). Quindi da qui a 5 anni si stima un'esplosione dell'IoT e questi numeri sono chiaramente destinati ad incrementare.
Quindi raccogliere, analizzare ed agire su questi dati diventa fondamentale.

IoT Data Essential

I dai esistono già da tempo e questi sono manipolati dai CRM che monitorano il comportamento delle utenze online, analizzando i momento di acquisto, quelli di vendita, etc… Come già detto, i big data esistono da tempo, tuttavia i **big data dell'IoT differiscono dai classici big data;** possiamo allora dire che l'IoT Big Data è:

- **Streaming:** si ha un flusso continuo di dati che va ad alimentare lo storage, chepuò provanire da un percorso di un automobile, la temperatura di un ambiente, il movimento di un utente. Questi dati necessitano di strategie specifiche per essere processati ed interpretati per estrarre da essi un significato, filtrandoli da eventuali informazioni futili.
- **Real time:** per molti device nell'IoT, e anche per alcune strategie di mercato, i dati devono essere gestiti in real time, ad esempio: tali dati arrivano in un certo momento e questi possono attivare una manovra di marketing in maniera istantanea. O ad esempio un avvertimento che segnala un temporaneo sconto del 15% su un intervento di riparazione poiché in quel determinato momento l'operatore è libero. Tutti questi esempi sono accomunati dal fatto che in essi il tempo ha un'importanza fondamentale.
- **Non strutturati:** non si parla di dati salvati nel database, sono dati che non possono essere messi in un modello relazionati. Un esempio di questi dati sono quelli dal gaming, quelli della home security o queli provenienti da veicoli smart.
- **Complessi:** sono dati complessi e si possono riconoscere solo dopo un'analisi solo dopo che venga studiato lo scenario di applicazione. Ad esempio, un condizionatore smart può funzionare solo in seguito ad un'analisi della routine della famiglia che vive nell'ambiente in cui esso è installato; solo dopo tale analisi potrà suggerire una temperatura o potrà funzionare in modo tale da risparmiare energia. Quindi i comportamenti possono venir fuori solo dopo un'accurata analisi dei dati.
- **Identificazione:** portando dietro i device questi possono essere utilizzati per il riconoscimento (come ad esempio il telefono) o può anche essere possibile mantenere l'anonimato

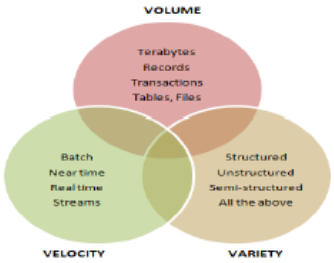
- **Mobili:** sono dati provenienti da oggetti in movimento e quindi devono essere gestiti diversamente da come vengono gestiti i big data già esistenti
- **Sensibili:** sono dati molto sensibili che possono presentare diversi livelli di vulnerabilità ed è quindi necessario una certa gestione accurata dei dati.

I dati IoT hanno queste caratteristiche che li contraddistinguono dai dati circolanti su internet.

Big Data Analytics

I big data devono essere analizzati per rendere questi ultimi utilizzabili e sensati. Parliamo allora di big data analytics: il processo di raccolta e analisi di grandi volumi di dati (big data) per estrarre informazioni nascoste un campo. Esiste una classificazione degli analytics che fa riferimento alle **3V:**

1. **Volume dei dati (Volume)**
2. **Velocità dei dati (Velocity)**
3. **Varietà dei dati (Variety)**



3V Concept

Si può ampliare questo concept aggiungendo altre due "V", riferendoci quindi ad un concept a **5V:**

4. **Veracità dei dati (Veracity) -dati più o meno veri-**
5. **Valore dei dati (Value) -valori statistici-**

5V Concept:

Ciò che però ci chiediamo è: in che modo si possono analizzare i big data? (Cosa significa fare Big Data Analytics?) Cosa sono le cose che tipicamente vengono fatte su questi dati?

Tipi di Big Data Analytics:

1. **Analisi Descrittiva:** analytics risponde ad una domanda fondamentale: "**cosa sta avvenendo?**". È una **fase preliminare** che crea un **set di dati storici.** Questa analisi raccoglie di dati per stimare i comportamenti futuri (un esempio sono i dati sul coronavirus)
2. **Analisi Diagnostica:** risponde alla domanda "**cosa è avvenuto?**", prova a trarre una causa di un determinato evento accaduto, quindi a capire **perché una cosa è avvenuta.**
3. **Analisi Predittiva:** risponde alla domanda **"cosa è probabile che avvenga?**" e serve a **stimare cosa potrebbe accadere in futuro**. Questo è un punto forte per il marketing che, con opportune strategie, potrebbe far business investendo sul futuro. Queste tecniche di predizione ed estrazione delle informazioni vengono eseguite da **intelligenza artificiale che analizza i dati correnti e scenari su cosa potrebbe accadere in futuro**
4. **Analisi prescrittiva:** è un'analisi più razionale che risponde alla domanda "**cosa dovrebbe essere fatto?**"

Queste 4 tipologie di analisi spiegano come mai le applicazioni data driven sono quelle più diffuse perché questi 4 punti possono essere declinati su diversi ambiti e con diversi smart objects.

Quindi questa evoluzione dei dati porta a delle sfida per i big data analytics che non possono basarsi più su schemi tradizionali già in uso quali i database relazionali (utilizzati in passato, ad esempio, per alcuni social o per alcune app di e-commerce o per qualsiasi altra applicazione in cui è di fondamentale importanza salvare i dati). Proprio per queste sono nate nuove tecnologie, come **Apache Hadoop,** un software opensource che molte aziende utilizzano per fare analisi sui dati: un esempio è Facebook che lo utilizza per analizzare i dati dai social network