Alma Mater Studiorum · Università di Bologna

SCUOLA DI SCIENZE
Corso di Laurea in Informatica

Approccio analitico all'infrastruttura IoT con proposta operativa open source

Relatore:
Chiar.mo Prof.
Maurizio Gabbrielli

Presentata da:
Giovanni Minelli

III Sessione
Anno Accademico 2018/2019

Introduzione

L'obbiettivo dell'IoT é quello di facilitare l'interconnessione tra il mondo fisico e quello digitale in modo da collegare applicazioni e dispositivi per scopi legati alla vita quotidiana ma questo compito, oggi giorno, é sempre piú arduo da svolgere. All'inizio del 2019 é stata riportata la presenza di 25 miliardi di dispositivi connessi ad Internet di cui la metá appartenenti alla famiglia dei dispositivi IoT. In un ambiente cosí vasto per varietá di dispositivi dovuta al continuo avanzamento tecnologico, la gestione delle comunicazioni tra le varie categorie e versioni di periferiche é diventata ormai un'attivitá spesso difficile da svolgere. Per questo motivo é importante che i fornitori di servizi adottino soluzioni sempre piú aperte che abilitino all'uso tutte le categorie passate, presenti e future. In caso contrario a subirne le conseguenze sono poi gli utenti, che si ritrovano ad avere dispositivi SMART ma "stupidi" poiché non riescono a comunicare tra di loro o ad integrarsi in infrastrutture IoT giá esistenti, a meno di interventi ad hoc. E' forse giusto doversi "affiliare" ad un'unica famiglia di dispositivi IoT per avere garanzie di funzionamento e intercomunicazione?
L'obbiettivo di questa tesi é quello di fornire una soluzione modulare, scalabile e soprattutto opensource, senza quindi doversi affidare a tecnologie o protocolli proprietari; risolvendo al contempo problemi legati all'uso di dispositivi per quanto riguarda compatibilitá, registrazione e autenticazione senza dover affrontare sempre i soliti problemi ricorrenti d'implementazione. Verranno presentate nel dettaglio le caratteristiche di un'infrastruttura IoT per comprendere piú approfonditamente la struttura della soluzione proposta. Verrá spiegato in che modo questa si integra e dove si va ad inserire all'interno dello stack di un sistema IoT ed i vantaggi che

riesce ad apportare sia internamente al sistema, che esternamente a dispositivi e consumatori finali.

Per presentare il tutto viene proposta un'infrastruttura IoT full stack che sia in grado di risolvere o per lo meno venire incontro alle necessitá e problemi comuni di questo settore. La progettazione e realizzazione di questa é avvenuta con l'intento di essere il piú possibile elastica per essere applicabile alle diverse situazioni e fornire spunti operativi per la realizzazione di infrastrutture IoT. Nonostante risenta delle limitazioni dovute all'ambiente di testing nel quale risiede, vengono mantenuti saldi gli aspetti di sicurezza, affidabilitá e scalabilitá all'interno della struttura.

Indice

Elenco delle figure

Capitolo 1

Internet of Things

In questo capitolo verrá introdotto il lettore all'IoT. Come viene definito, che ruolo ricopre nella societá attuale, i vantaggi che apporta e in queli ambiti mira ad integrarsi.

1.1 Cos'é l'Internet of Things (IoT)?

Il concetto su cui si base l'IoT risale a molto prima dell'effettiva nascita di questo termine. Esso fu usato per la prima volta dall'imprenditore tecnologico britannico, Kevin Ashton nel 1999, per descrivere un sistema nel quale gli oggetti situati nel mondo fisico possono essere collegati a internet attraverso i sensori. L'idea alla base é quella di voler creare un mondo interconnesso in cui l'ambiente si adatta in modo interattivo e informativo in base alle entitá di cui é composto o con le quali interagisce. Troviamo i primi esempi di tecnologie impiegate al monitoraggio e controllo remoto giá decenni prima: per esempio nel 1970 erano giá in commercio sistemi per monitorare da remoto le reti elettriche tramite le linee telefoniche, oppure quando all'inizio degli anni 80' alcuni programmatori della Carnegie Melon University connetterono la macchinetta delle bevande del campus a internet per poter controllare disponibilitá e temperatura dei prodotti prima di recarvisi di persona.
I primi "visionari" apportarono significativi sviluppi nel campo industriale con

connsessioni machine-to-machine (M2M) applicate a attrezzature e macchinari integrati nel processo produttivo per incrementare l'efficienza e per monitoraraggio remoto. Queste pratiche, peró, non vennero prese veramente in considerazione da imprenditori e proprietari d'attivitá, prima della seconda metá degli anni 90, grazie a nuovi apporti tecnlogici nelle infrastrutture wireless. Oggi, questi stessi intenti sono riproposti su scala piú estesa, con l'industria 4.0.
Ashton raccolse questi concetti e descrisse quest'insieme di necessitá nel suo discorso di presentazione per Procter & Gamble, ma cosa piú importante, le sue parole spostarono il focus della questione sull'elemento principe, che di fatto racchiude tutto il valore del sistema: i dati.

> "Today computers, and, therefore, the Internet, are almost wholly dependent on human beings for information. Nearly all of the roughly 50 petabytes of data available on the Internet were first captured and created by human beings by typing, pressing a record button, taking a digital picture or scanning a bar code. The problem is, people have limited time, attention, and accuracy. All of which means they are not very good at capturing data about things in the real world. If we had computers that knew everything there was to know about things, using data they gathered without any help from us, we would be able to track and count everything and greatly reduce waste, loss and cost. We would know when things needed replacing, repairing or recalling and whether they were fresh or past their best."

Gli anni a venire rappresentarono la concretizzazione dell'idea espressa da Ashton grazie agli sviluppi tecnologici introdotti nel nuovo millennio, con un vera e propria diffusione di massa a partire dal 2009.
La Cisco Internet Business Solutions Group (IBSG) attribuisce la nascita dell'IoT proprio in questo periodo (2008-2009) quando il numero di dispositivi connessi a internet superó quello di persone al mondo. Nel 2010 questo numero era giá raddoppiato a 12.5 miliardi. Nel 2011 fu annunciato pubblicamente il rilascio dell'IPv6 per far fronte all'enorme quantitá di device in diffusione nel mondo.
Nel 2012 Gartner citó l'IoT come una delle tecnologie piú emergenti nel campo informatico e nel 2014 venne riportata in prima posizione per aspettativa di sviluppi

futuri all'interno dell'omonimo grafico che rappresenta la maturitá e il trend di diffusione delle tecnologie (Gartner Hype Cycle Figura 1.1).
Oggi giorno ci sono 25 miliardi di dispositivi connessi alla rete Internet (Figura 1.2) di cui la metá sono appartenenti alla famiglia dell'Enterprise IoT(sensori e attuatori in generale) e la restante parte é composta principalmente da dispositivi SMART.
E' facile notare come questo trend sia in aumento negli anni a venire facendoci capire quanto queste tecnologie che giá oggi sono integrate nelle nostre routine quotidiane, ricoprano un ruolo fondamentale nel nostro futuro.

Figura 1.1: Gartner Hype Cycle 2014

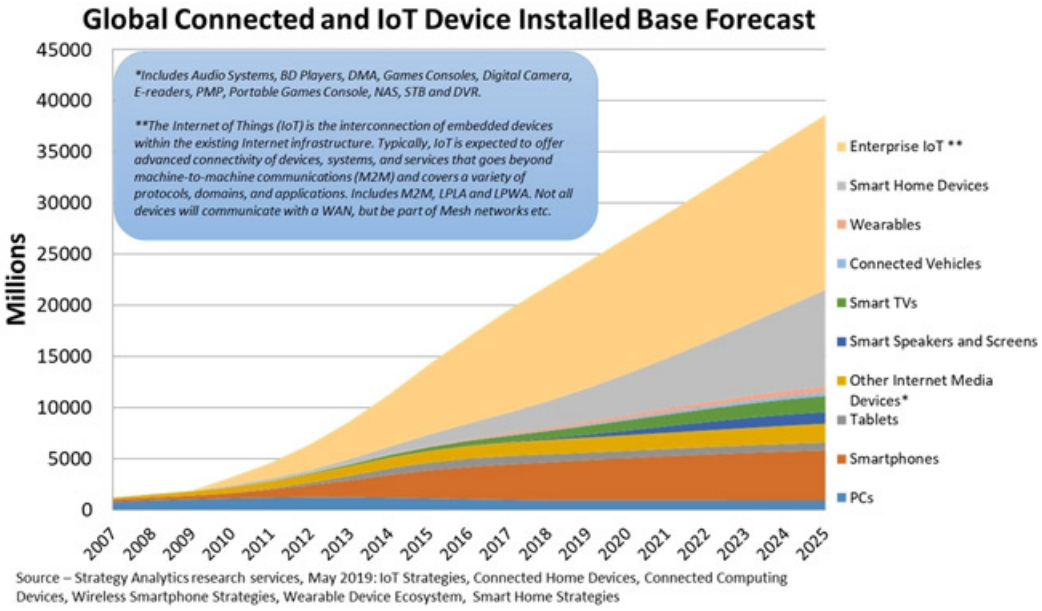

Figura 1.2: Crescita del numero di dispositivi connessi

1.1.1 Definizione

Dare una definizione specifica all'IoT é complesso poiché é un ambito talmente esteso che difficilmente se ne puó esprimere ogni aspetto in una singola asserzione. Nel tempo, varie fondazioni ed enti hanno proposto definizioni piú o meno riconosciute come valide nell'ambiente tecnico, ma ad oggi non ne esiste ancora una ufficiale. Di seguito alcune d'esempio:

- Una prima definizione importante fu data dall'**ITU-T** nel 2005: "IoT é una combinazione di identificazione degli oggetti (RFID), rilevamento (sensori) e interazione (nanoattuatori) dei cambiamenti dell'ambiente."

- **Internet Architecture Board** (IAB), RFC 7452 [**?**] "Architectural Considerations in Smart Object Networking": Il termine "Internet of Things" (IoT) denota una tendenza in cui un grande numero di dispositivi embedded impiegano i servizi offerti dai protocolli internet. Molti di questi dispositivi, spesso chiamati *smart objects*, non sono direttamente gestiti da essere umani, ma esistono come componenti in edifici o veicoli o sparsi nell'ambiente.

- **IEEE Communications Magazine** definisce l'IoT in congiunzione di internet e servizi cloud: L'Internet of Things (IoT) è un framework nel quale tutte le cose hanno una rappresentazione e una loro presenza in Internet. Più specificamente, l'Internet of Things ha come scopo quello di offrire nuove applicazioni e servizi facendo da ponte tra il mondo fisico e quello virtuale, in cui comunicazioni Machine-to-Machine rappresentano la comunicazione di base che abilita l'interazione tra le cose e il cloud"

- **Dizionario Treccani** nella sezione del lessico del XXI Secolo riporta: "Rete di oggetti dotati di tecnologie di identificazione, collegati fra di loro, in grado di comunicare sia reciprocamente sia verso punti nodali del sistema, ma soprattutto in grado di costituire un enorme network di cose dove ognuna di esse é rintracciabile per nome e in riferimento alla posizione"

Le varie definizioni dell'IoT non necessariamente sono in disaccordo, piuttosto sottolineano differenti aspetti del fenomeno, da differenti punti di vista e casi d'uso. Anche se questa sovrabbondanza di termini e significati puó essere fuorviante per consumatori e meno esperti ("Internet of Everything","Smart world","Industria 4.0") il concetto alla base resta sempre lo stesso ed é validante per la maggior parte delle diciture piú diffuse.

1.1.2 Vantaggi, applicazioni e usi

Quando si parla di IoT pensiamo volgarmente a sistemi in cui ogni dispositivo tecnologico possa interfacciarsi e comunicare con altri dispositivi. Immaginiamo una sveglia che decidiamo di impostare alle 6 del mattino, che in modo autonomo comunichi la sua configurazione ai dispositivi della rete di casa, scaturendo quindi varie azioni secondo comportamenti programmati. Ad esempio mezz'ora prima della sveglia vogliamo che l'acqua del boiler si inizi a scaldare per poter fare la doccia, vogliamo che le tapparelle si alzino qualche minuto prima in modo da avere la stanza illuminata al nostro risveglio, e che infine l'attivazione della sveglia segnali alla macchinetta del caffé di accendersi in modo da essere pronta all'uso. Questo banale esempio incentrato sul consumatore racchiude il concetto di *smart*

environment che fu definito per la prima volta dall'antenato dell'oggi definito *ubiquitous computing*, Mark Weiser:

> "A physical world that is richly and invisibly interwoven with sensors, actuators, displays, and computational elements, embedded seamlessly in the everyday objects of our lives, and connected through a continuous network."

Se estendiamo questo concetto al di fuori delle mura di casa possiamo vedere infinite possibilitá di applicazione dell'IoT nella vita di tutti i giorni, automatizzando routine quotidiane, definendo sequenze di azioni e reazioni, e assorbendo dati dall'ambiente circostante. Se ben integrata con ció che la circonda questa tecnologia puó portare innumerevoli vantaggi per il consumatore finale, il quale ruolo puó essere ricoperto da chiunque: dal singolo individuo all'intera comunitá cittadina, dall'operaio di un'azienda metallurgica al paziente ricoverato in ospedale.
Ricordiamo peró che l'IoT non é incentrato sull'utente ma sui dati. Per far si che tutto funzioni abbiamo infatti bisogno di raccogliere e archiviare una grossa mole di dati e spesso é necessario che questi vengano processati in teampo reale. Big Data e algoritmi di ML sono infatti argomenti intrinsecamente collegati all'IoT ed é anche grazie agli sviluppi in questi campi che c'é stata una cosí larga diffusione di dispositivi SMART negli ultimi anni.
Nell'ambiente domestico ormai tutti abbiamo uno smart speaker con assistente vocale integrato, collegato alla rete internet di casa, magari anche con lampadine smart, prese intelligenti, termostati e termosifoni tutti comandabili tramite esso. Quasi tutte le famiglie hanno un televisore smart in casa: da dati statistici sono presenti 6.5 milioni di smart tv in Italia su 16 milioni di nuclei familiari [6]. Eppure, questi numeri, che ne sottolineano la diffusione nella massa, sono solo un "effetto secondario" di questa tecnologia, una piccola percentuale della sua grande potenzialitá. Il vero campo di applicazione é quello dei settori lavorativi: principalmente primario e secondario dove i dati sono preponderanti e tutto é misurabile tramite sensori.
Secondo i dati raccolti [4] risalenti al 2017, sette aziende su dieci (72%) sfruttano dispositivi e sensori IoT nel luogo di lavoro e oltre tre quarti (78%) di queste dichiarano che ció ha contribuito ha migliorare l'efficacia del team IT, mentre il

75% ha registrato un aumento della redditivitá.
Nel comparto industriale nasce la nacessitá di interconnettere sistemi eterogenei spesso non progettati per lavorare in sinergia tra loro. L'introduzione di tecnologie IoT ha abilitato in modo piú naturale la possibilitá di instaurare canali di comunicazione tra processi e macchinari, e trasferimenti di informazioni da apparecchiature moderne a tecnologie legacy, da sistemi analogici e manuali a complesse infrastrutture virtuali.
Chi ha adottato l'IoT ha registrato aumenti significativi nell'efficienza del processo di business (83%), d'innovazione (83%) e di visibilitá rispetto ai competitors (80%).
Nella sanitá sei strutture sanitarie su dieci si avvalgono giá di soluzioni IoT: i monitor paziente (64%) e i dispositivi per imaging e raggi X (41%) sono fra i principali dispositivi connessi alla rete, oltre ai numerosi sensori per monitorare e mantenere i dispositivi medicali.
Altra diffusa applicazione dell'IoT riguarda la creazione di servizi per i punti vendita, ad esempio fornire ai clienti informazioni sui prodotti e offerte personalizzate (30%). Un'ulteriore 18% utilizza l'IoT per controllare da remoto i fattori ambientali come il riscaldamento e l'illuminazione all'interno dei negozi.
Il settore agricolo, Il *Precision farming* o *Agrifood* é uno dei settori con la piú elevata opportunitá di sviluppo e con la piú bassa penetrazione, ad oggi, di soluzioni digitalizzate. Negli ultimi anni anche sono nate numerose realtá legate all'utilizzo dei droni, sensoristica ambientale e territoriale, applicazioni per il meteo e automazione della parte gestionale.
Per finire, il terziario, dove citiamo applicazioni legate alla mobilitá cittadina (Smart Car e Connected Car) ma anche applicazioni legate al mondo del trasporto ferroviario, con treni supportati da sensori IoT, e alla pubblica amministrazione (energia, sostenibilitá, rifiuti, ambiente) con i contatori predisposti al telecontrollo e alla telegestione o i sistemi di localizzazione e per il monitoraggio del corretto funzionamento delle macchinette per il gioco d'azzardo.
Questi sono solo alcuni degli esempi di utilizzo quotidiano e le realtá cambiano di paese in paese a seconda delle esigenze specifiche della comunitá.

Il futuro per queste tecnologie sembra radioso nonostante siamo ancora in un periodo di definizione delle sue componenti e delle sue possibili applicazioni nel mondo reale. Studi e ricerche proseguiranno di certo negli anni a venire, soprattutto grazie alla diffusione della tecnnologia per la realtá simulata e anche alla ormai prossima introduzione del 5G, il quale rappresenta la chiusura del cerchio per l'ecosistema dell'Internet of Things. Grazie alle sue caratteristiche e al cosiddetto *network slicing*, consentirá la connessione di un numero di dispositivi molto piú elevato di quanto non sia possibile oggi, assicurando allo stesso tempo miglioramenti in prestazioni, tempi di latenza e affidabilitá.

Capitolo 2

Analisi strutturale

Affrontiamo in questo capitolo nello specifico come viene strutturata l'architettura di un sistema IoT, quali sono i componenti che ne fanno parte, i pattern strutturali, i modelli e i protocolli di comunicazione. Per finire verrá trattato nel particolare il ciclo di vita dei dati all'interno di un sistema IoT.
Forniamo al lettore un rapido glossario in modo da poter comprendere appieno i concetti descritti successivamente e per un veloce ripasso del significato di termini e sigle.

Dispositivo: Componente che include hardware e software ed interagisce direttamente con il mondo fisico. Solitamente ad esso é associato un identificatore univoco, un tipo o classe d'appartenenza, un modello ed un numero seriale.

Tenant: Divisione a livello logico che indica un gruppo o una classe d'appartenenza.

Edge: Luogo virtuale che si frappone tra OT e IT, ossia il collegamento tra dispositivi e rete, tra mondo fisico e mondo virtuale.

Gateway: Permette ai dispositivi non direttamente connessi ad internet di raggiungere i servizi cloud. La sua funzione non é per forza limitata a componente

di rete ma puó essere estesa delegando la responsabilitá di effettuare azioni sui dati.

Cloud: Vasta rete di server remoti, ubicati in tutto il mondo ma collegati tra loro e che operano come un unico ecosistema. Solitamente é il luogo dove vengono gestiti dati e servizi.

Sensore: Dispositivi specializzati nella raccolta di dati.

Attuatore: Dispositivi che agiscono in seguito ad azioni, eventi o segnali ricevuti.

SOA: Service-Oriented Architecture - modello architetturale per lo sviluppo, costruzione e pubblicazione di un servizio di rete.

IT: Information Technology - Reparto informatico

OT: Operational Technology - Reparto operativo

QoS: Quality of Service - proprietá che si riferisce ad ogni tecnologia che controlla e gestisce risorse. Disturbi, limitazioni ed elementi che interferiscono con il servizio abbassano questo valore.

QoD: Quality of Data - proprietá che si riferisce ai dati scambiati in una trasmissione. I dati vengono considerati di alta qualitá quando questi possono essere usati per effettuare operazioni, prendere decisioni o fare previsioni.

2.1 Architettura

In parte a causa della nascita decentralizzata e caotica di queste tecnologie ed in parte per necessitá dovute alla gestione di domini eterogenei e molto vari tra loro, non é mai stata formalizzata una struttura canonica da seguire per la composizione dell'intero stack di un architettura IoT. Situazioni diverse possono formalizzarsi nella necessitá dell'ausilio di piú o meno risorse e nell'integrazione di piú o meno moduli che interagiscono tra loro, e ció influisce non solo sul design stesso dell'architettura ma anche sulla funzione operativa che assumono i

componenti.
Per risolvere questo problema e creare una situazione di intesa tra i partecipanti del settore alcune fondazioni hanno introdotto una serie di standard architetturali e alcune architetture di riferimento [22].
Un tentativo iniziale per coordinare gli sforzi nell'ambito, fu realizzato nel 2011 nel contesto di vari progetti raggruppati sotto il nome di FP7. I principali progetti riconosciuti sotto questo programma furono IoT-I e IoT-A (nonostante la volontá impiegata, da un sondaggio risulta che il 25% dei partecipanti non credeva che sarebbero riusciti a realizzare un modello architetturale comune di riferimento, come prefissato).
Parallelamente troviamo un'iniziativa supportata da molte industrie: il progetto FI-WARE, che inizió a lavorare sul design di una piattaforma dell'"internet di nuova generazione" implementando anche la gestione di sistemi IoT.
Uno standard importante é stato fissato nel 2012 con la pubblicazione del protocollo NGSI (Next Generation Service Interfaces Architecture) da parte dell'OMA per la gestione del contesto applicativo di sistemi in genere e ancora nel 2013 con la definizione del livello di servizio atto a comunicazioni M2M da parte dell'ETSI con il protocollo OPC UA.
FI-WARE supporta giá i due standard citati e seppure ancora in sviluppo si prospetta come pilastro del campo IoT, almeno in contesto europeo.

2.1.1 Livelli logici e componenti fisici

Per descrivere l'infrastruttura di un sistema IoT, suddividiamo lo stack architetturale e affrontiamo livello per livello, analizzando i componenti e il flusso di dati [24].
Rappresentiamo la struttura logica verticalmente nel seguente modo (Figura 2.1):

- Le applicazioni, che stanno in cima alla nostra architettura, sono il punto d'accesso dell'utente finale a tutte le funzionalitá del sistema, e non verranno affrontate nello specifico all'interno di questa trattazione. -

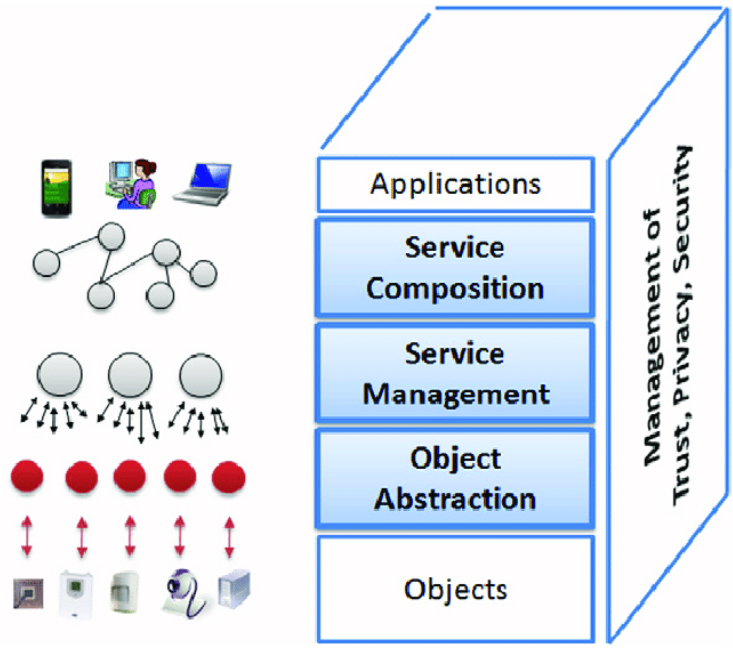

Figura 2.1: Livelli dell'infrastruttura logica

Object abstraction

L'IoT comprende una vasta gamma di oggetti di diversa natura ognuno accedibile tramite specifici linguaggi e funzioni. Poiché anche la semantica di questi cambia di produttore in produttore, diventa necessario introdurre un livello di astrazione in grado di armonizzare l'accesso ai differenti dispositivi tramite un'unica porta. A volte il singolo dispositivi puó offrire specifici servizi web accedibili tramite l'indirizzo IP ad esso assegnato, ma come puó questo contribuire ad una gestione ben organizzata in ambienti con migliaia di dispositivi?
Il livello di aggregazione sará formato da due sottolivelli: l'interfaccia e il livello di comunicazione.

Il primo fornisce un'interfaccia (comunemente web) che mette a disposizione i metodi specifici disponibili. E' responsabile della gestione di tutti i messaggi e dei comandi in entrata e in uscita tra dispositivi e mondo esterno. Il secondo sottolivello implementa la logica che sta dietro all'interfaccia, traducendo i metodi in un set di comandi specifici per il singolo oggetto collegato alla rete.
Alcune architetture propongono l'integrazione dello stack TCP/IP nei singoli device, con implementazioni piú leggere come TinyTCP, uIP e IwIP, i quali forniscono un'interfaccia strutturata simil-socket, ma apposita per funzionare in contesti a risorse limitate quali possono essere i dispositivi *embedded*. E' inoltre possibile integrare web server negli oggetti stessi, i quali svolgeranno la funzione di abstraction layer di loro stessi. Comunque, sempre piú spesso questa funzione di aggregazione viene fornita da un proxy, il quale é responsabile dell'apertura delle connessioni tra i socket dei dispositivi e inviare a questi i segnali secondo lo specifico linguaggio supportato restando in attesa della risposta. Alcune implementazioni possono prevedere anche fasi di conversione in linguaggi standard del sistema e fasi di elaborazione dei dati per ridurne la complessitá.

Service management

Questo livello fornisce le funzioni principali che ci si aspetta siano accessibili per ogni oggetto e per permetterne la gestione. Questo insieme di servizi racchiude: metodi di dynamic discovery per individuare dispositivi nella rete, servizi di registrazione e configurazione, e monitoraggio dello stato. A questo livello alcune implementazioni forniscono funzioni aggiuntive relative al QoS, alla sicurezza dei dispositivi (sia per comunicazioni in entrata che in uscita) o relative alla semantica applicativa degli stessi (gestione del contesto operativo). Questo livello puó includere metodi di pubblicazione remota di nuovi servizi a run-time per soddisfare i bisogni dell'utente o delle applicazioni stesse, ad esempio in caso di aggiornamento di dipendenze esterne. Inoltre, qui é mantenuto l'elenco dei servizi attivi associati ai singoli oggetti della rete. Il livello superiore potrá, eventualmente, aggregare questi servizi per fornire dettagli e implementazioni piú complesse con dati e servizi messi a disposizione da questo layer.

Service composition

Nel caso di sistemi SOA-based é il livello dell'architettura piú alto. Fornisce le funzionalitá per l'aggregazione dei singoli servizi offerti dai dispositivi connessi alla rete, per la costruzione di applicazioni specifiche. Ció che si trova in questo layer non ha conoscenza dei dispositivi connessi: gli unici componenti con cui puó interagire sono i servizi messi a disposizione dal livello sottostante. Nell'implementazione di questo layer é buona norma mantenere una repository di tutte le istanze di servizi connessi, che sono eseguite a runtime, per poter costruire servizi composti. La logica che esprime la creazione, la gestione e le interazioni esterne di questi servizi viene solitamente descritta tramite linguaggi di workflow (come BPEL - Business Process Execution Language, e Jolie). Il singolo workflow di un servizio puó anche comparire all'interno di altri, in modo da creare istanze eseguibili in seguito al verificarsi di una coordinazione di eventi o sequenze di azioni svolte da singole entitá non in comunicazione tra loro.

Data management

Il luogo logico ove avviene la maggior parte del lavoro computazionale. Qui i dati vengono gestiti in base al contesto di provenienza ed alle informazioni che esplicitano. Sono previste fasi piú o meno complesse e approfondite di analisi ed elaborazione, a seconda dell'implementazione. Successivamente i dati vengono registrati in una qualche struttura database su supporti di memoria fisici o in cloud. Il database utilizzato potrebbe inoltre variare a seconda del tipo di dato da memorizzare. Esistono differenti tipologie di database sia nella categoria dei database relazionali che per in quella dei non relazionali, ed ognuno puó apportare rallentamenti o vantaggi significativi per specifiche strutture dati a seconda del tipo di indicizzazione.

Associamo ora questa visione logica del nostro sistema con l'implementazione fisica, anch'essa divisa in 4 livelli. Il livello 1 é composto solitamente da sensori e attuatori. Il livello 2 include dispositivi e sistemi di aggregazione e conversione (da analogici a digitali). Al livello 3 (Edge IT) servizi e applicazioni preprocessano i

dati prima che vengano spostati in cloud o archiviati in database. Infine al livello 4 i dati sono analizzati, elaborati e/o salvati in modo persistente in sistemi tradizionali (datacenter remoti, database, ecc).
Il livello 1 che si occupa di rilevazioni e eventi fisici é gestito interamente dai reparti OT mentre i livelli 3 e 4 dall'IT, sia che la computazione avvenga in cloud, sia che avvenga in server locali. L'"edge" ossia la demercazione tra l'ambiente fisico e quello digitale e quella linea di separazione verticale che si frappone tra i compiti dell' OT e IT, e a seconda dell'implementazione puó sfumare in una direzione piuttosto che nell'altra rendendo comuni ad entrambe le visioni, alcuni aspetti.
Vediamo ora piú approfonditamente i componenti dei singoli livelli (Figura 2.2).

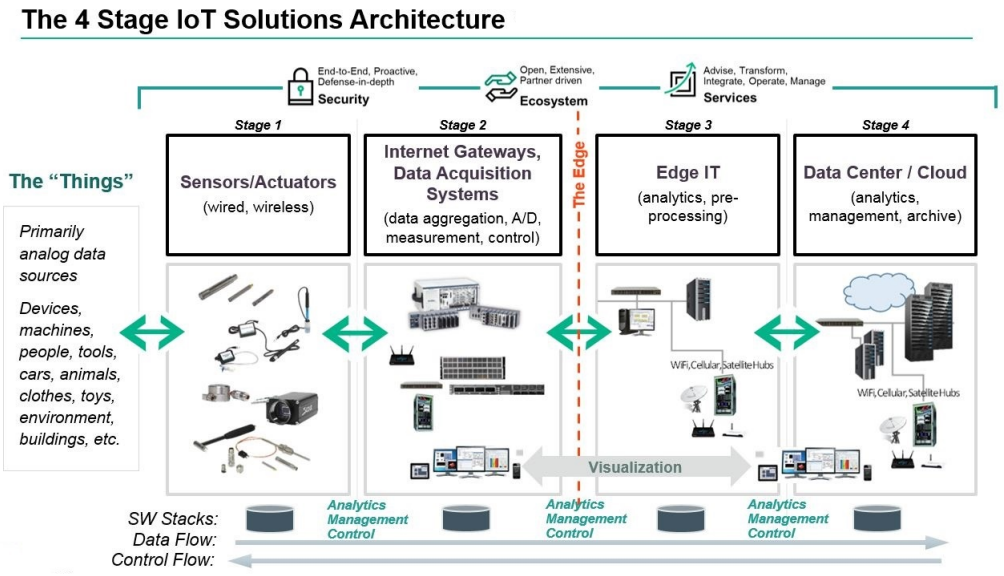

Figura 2.2: Livelli dell'infrastruttura fisica

*Tralasciamo in questa visione aspetti piú delicati come la registrazione e gestione dei servizi che peró abbiamo citato nella descrizione dei layer logici.

Livello 1. Sensori e attuatori

I sensori raccolgono dati dall'ambiente circostante o da apparati specifici (come strumenti, macchinari e attrezzature) che sono oggetto di misurazione e li trasmettono come dati utili. Un esempio molto vicino alla realtá quotidiana di tutti puó essere il giroscopio del telefono usato dal sistema operativo per identificare l'inclinazione del dispositivo nello spazio.
Gli attuatori possono intervenire per cambiare lo stato fisico di ció che genera i dati. L'interruttore di un alimentatore o una valvola per la regolazione del passaggio dell'aria possono essere attuatori.

Livello 2. I gateway internet

I dati provenienti dai sensori sono in forma analogica e necessitano di essere convertiti e raggruppati tra loro in stream di dati digitali per essere analizzati e processati. I sistemi di acquisizione dati (DAS) sono incaricati di svolgere questi compiti: connettono i sensori in rete, aggregano gli output e convertono i dati da analogici a digitali. Il gateway riceve i dati aggregati e digitalizzati e li reindirizza su Wifi o reti LAN cablate.
Il secondo livello é spesso molto vicino a sensori e attuatori. Per esempio ad una pompa possono essere collegati una decina di sensori per monitorarne diversi parametri. Questi saranno connessi ad un sistema di acquisizione, e comunicheranno direttamente al gataway le rilevazioni. Il sistema di acquisizione potrebbe essere fisicamente attaccato alla pompa, mentre il gateway, adiacente ad essa, si occuperá di mandare i dati al livello 3 o direttamente al livello 4 per la memorizzazione.

Livello 3. Edge IT

Aggregati e digitalizzati, i dati sono ora pronti per passare al mondo dell'IT. Potrebbe essere necessario un'ulteriore elaborazione o analisi prime di salvarli in modo permanente, ed ecco che entra in gioco l'Edge IT. Questi sistemi possono essere dislocati sia in prossimitá del punto di acquisizione sia in locali adibiti, piú lontani ma solitamente sempre nello stesso stabile. Viene preferita una posizione

protetta data la tipologia di hardware in discussione. Ad esempio in un ambiente aziendale, questi sistemi possono trovarsi in un ufficio o una sala server, oppure dentro armadi cablati posizionati sul campo dove avvengono le rilevazioni (ambienti edge, al bordo dell'infrastruttura).
Nel caso si abbiano particolari esigenze per quanto riguarda il salvataggio o problemi di rete dovuti a trasferimento di grosse quantitá di dati, questo é il livello dove possono essere implementati algoritmi di aggregazione e di processing in modo che solo una parte dei dati debba essere mandata interamente fino alla destinazione finale. Inoltre qui é possibile giá implementare applicazioni di visualizzazione grafica tramite l'uso di dashboard, mappe e grafici.
Infrastrutture di virtualizzazione sono spesso impiegate a questo livello data la loro relativa velocitá e semplicitá per la pubblicazione e la gestione remota.

Livello 4. I datacenter e il cloud

Dati che necessitano analisi approfondite ed elaborazioni onerose in termini di risorse ma che soprattutto non devono produrre risultati immediati (ad esempio la decisione di fermare una pala eolica in caso di raffiche di vento pericolose), sono inviati in datacenter o sistemi cloud, dove tramite elaboratori piú potenti potranno essere gestiti e salvarti in modo sicuro. E' necessario del tempo prima di ottenere risultati provenienti da questo livello ma in compenso é possibile eseguire analisi piú approfondite combinando i dati di sensori provenienti da piú luoghi o da timeframe differenti. Il livello 4 risiede solitamente nel cloud o in sistemi cloud ibridi solitamente affittati dal consumatore. In ogni caso, il tipo di piattaforma non influisce sul tipo di elaborazione dei dati.
Quando si affronta il tema del salvataggio dati non si puó partire per preconcetti ed é necessario vagliare tutte le possibilitá poiché quando bisogna gestire grandi quantitá di dati, anche semplici operazioni di ricerca possono impiegare un tempo non indifferente. Ci sono differenti opzioni tra cui scegliere come SQL o NoSQL DB, Object/Document DB e piattaforme basate sul file storage. Ovviamente la scelta va presa tenendo presenti i costi che comportano questi servizi e i tipi di dati che dovranno essere salvati.

2.1.2 Modelli di computazione

L'architettura vista nella sezione precedente descrive una generica infrastruttura di un sistema IoT, la quale varierá negli aspetti, a seconda del contesto o delle necessitá del consumatore. In particolare puó cambiare il momento, ossia il luogo, nel quale effettivamente viene svolta l'analisi e l'elaborazione dei dati.
Sostanzialmente, in un'architettura IoT, azioni di *data processing* possono verificarsi in ognuno dei 4 livelli, ma essendo processi onerosi in termini di risorse computazionali dovremo prendere in considerazione i limiti fisici dettati dall'hardware a disposizione ad ogni livello, durante la decisione del flusso operativo. Poiché i dati sono il fulcro principale del sistema sará necessario scegliere tra l'immediatezza o la profonditá d'analisi da effettuare su di essi. Piú é necessario che la risposta o la reazione ad un certo evento sia immediata, piú la computazione dei dati dovrá avvenire in prossimitá del dispositivo finale. In altri casi i dati potrebbero avere un'utilitá ma non immediata (misurazioni per analisi statistiche), oppure la mole di dati potrebbe essere talmente enorme da non essere processabile in streaming (esperimenti scientifici tipo LHC CERN), in tal caso ci si limiterá a salvare i dati per poi processarli in un secondo momento. Esistono molti modelli raggruppabili fondamentalmente in 4 categorie:

Fog computing

L'architettura del fog computing é una via di mezzo tra l'*edge* e il *cloud computing* applicata per spostare il centro operativo a metá tra datacenter e sensori. Questa architettura introdotta da Cisco nel 2012 permette l'elaborazione dei dati in locale estendendo la tradizionale architettura "agli estremi della rete" mediante l'uso di dispositivi in grado di processare e gestire i dati in maniera autonoma come fossero degli *edge server*. La potenza computazionale é limitata, cosí come anche lo spazio di salvataggio locale, ma é possibile effettuare analisi e veloci inspezioni sui dati, abilitando il sistema anche a situazioni di traffico real time [21]. Il *fog layer* é composto da molti nodi (o celle), ognuno incaricato di gestire un set di dispositivi di rete con spazio d'archiviazione e potenza di calcolo propria . Quando un nodo riceve una richiesta, questo si occupa del suo riconoscimento decidendo

dove debba essere elaborata e soddisfatta: se localmente o nel cloud. I campi di applicazione principali sono l'eHealth e l'ambito militare.

Edge computing

In questa architettura, l'elaborazione avviene in loco, lontano dal core del sistema. Questo modello permette di processare i dati in primo luogo sui dispositivi edge per poi spostarsi verso il cloud. I dispositivi possono non essere collegati costantemente alla rete e necessitano quindi di una copia dei dati o di un riferimento ad essi per effettuarne l'elaborazione anche offline. I due piú grandi vantaggi sono i bassi tempi di latenza, poiché i dati non devono "andare lontano" per essere elaborati, e l'efficiente utilizzo delle risorse e dei dispositivi della rete, che sono in grado di agire sia da consumatori che da produttori di dati. In particolare, ogni dispositivo puó avere funzioni diverse:

1. Occuparsi di questioni inerenti alla sicurezza
2. Funzioni di filtraggio e selezione/trashing sui dati
3. Salvataggio locale dei dati

Un esempio applicato di questo modello computazionale é l'uso di videocamere (actioncam indossabili o dashcam da auto) da parte delle forze dell'ordine che invece di registrare e inviare tutte il materiale filmato in cloud, passano i dati a cloudlet (applicazioni che risiedono localmente vicino all'utente) che tramite computer vision e analisi realtime estraggono le informazioni utili inviando in cloud solo i metadati relativi ai risultati ottenuti [21].

Cloud computing

Qui i dati sono mandati direttamente al datacenter per essere analizzati, elaborati e salvati per renderli accessibili alle applicazioni tramite la rete. Questo modello comporta tempi di latenza tra input e output molto alti ma in compenso permette di usufruire di una potenza di calcolo superiore concessa da potenti elaboratori o anche da reti di calcolo distribuito. Non é indicato per applicazioni con trasmissione

di dati in streaming ad alta velocitá, poiché l'infrastruttura di rete rappresenterebbe un collo di bottiglia per lo scambio di dati all'interno del sistema.
Solitamente é un servizio "in affitto" fornito da gestori di datacenter e a seconda delle necessitá sono disponibili varianti che differiscono molto, sia nel prezzo che nel pachetto d'offerta:

1. Infrastructure as a Service (IaaS): il cliente compra tutte le risorse dell'infrastruttura fisica come hard disk, periferiche o interi server e ha il controllo totale sull'architettura. L'unica cosa su cui non puó agire, come negli altri modelli é l'infrastruttura cloud sottostante.

2. Platform as a Service (PaaS): tutto ció che risiede al di sotto dell'applicazione é affittato. Il cliente é solo abilitato a pubblicare la propria applicazione sull'infrastruttura.

3. Software as a Service(SaaS): modello software distribuito. Tutto il software e i servizi sono gestiti da un service provider e messi a disposizione del cliente tramite internet; il quale puó accedere ad essi solamente attraverso interfaccia remota (web o tramite altri protocolli) con la possibilitá di cambiarne solo in parte la configurazione.

4. Mobile Backend as a Service (MBaaS): anche conosciuto come Backend as a Service (BaaS), fornisce un'applicazione web o mobile con un indirizzo per connettere l'applicazione con il backand cloud storage. MBaaS fornisce funzioni come gestione utenti, notifiche e integrazione con i servizi dei social networks. Questo servizio cloud mette a disposzione Application Programming Interface (API) e Software Development Kits (SDK).

L'uso piú frequente di questa soluzione nasce dalla necessitá di analizzare grandi moli di dati provenienti da differenti sorgenti usate per ricavare modelli comportamentali o statistici come nel campo dell'e-commerce o nell'uso di sistemi operativi, siti internet e applicazioni per mobile.

Distributed computing

Questa architettura é specifica per l'elaborazione di grandi volumi di dati. Nell'IoT, poiché i sensori pubblicano continuamente dati, si verificano spesso situazioni difficili da gestire in termini di efficienza e risposta realtime. La computazione distribuita mira a risolvere questi problemi dividendo i dati in pacchetti e assegnandone l'elaborazione a computer diversi. Rispetto al *cloud computing*, unendo le potenzialitá della computazione distribuita a quelle del *fog computing* otteniamo:

- una diminuzione del traffico internet
- un aumento nella velocitá di elaborazione dei dati
- una diminuzione dell'uso della CPU
- una riduzione del consumo di energia {Cognitive [19]
- capacitá di processare un volume piú elevato di dati

2.1.3 Modelli di comunicazione

Poiché l'IoT é l'integrazione di varie e differenti tecnologie, la connettivitá e la comunicazione diventano due aspetti molto rilevanti nell'architettura. Nel Marzo 2015, l'Internet Architecture Board (IAB) ha rilasciato il documento RFC 7452 [**?**], il quale affrontando l'argomento dell'architettura di rete degli smart object, delinea un quadro di quattro modelli di comunicazione utilizzati dai dispositivi IoT.

Device to Device (D2D)

Questo tipo di comunicazione identifica due o piú dispositivi connessi che comunicano tra di loro attraverso un server che svolge la funzione di intermediario. Questi utilizzano solitamente protocolli leggeri per comunicare come ZigBee, Z-Wave o Bluetooth, poiché i pacchetti di dati trasmessi sono spesso di piccole dimensioni. Per esempio, lampadine, interruttori di luce, e termostati si trasmettono segnali per scaturire eventi.

In normali infrastrutture l'uso out of the box di dispositivi in un ambiente comune é limitato dai problemi di compatibilitá di comunicazione. Ció costringe il semplice utente inesperto ad affidarsi a famiglie di prodotti piuttosto che scegliere il singolo dispositivo in base a preferenze ed esigenze.

Device to Cloud (D2C)

In questo tipo di comunicazione i dispositivi IoT si connettono direttamente ad un servizio cloud. Vengono usati protocolli standard come CoAP, DTLS, UDP, IP tramite Wifi o Ethernet. Solitamente le case produttrici del servizio cloud sono le stesse che forniscono i dispositivi SMART.
Soprattutto nella realizzazione di soluzioni personalizzate, possono nascere problemi di compatibilitá, risolvibili tramite l'introduzione di livelli intermediari di servizio nel caso i metodi d'accesso siano stati resi pubblici dal produttore.

Device to Gateway (D2G)

Il modello descritto precedentemente é funzionale finché vengono usate bande di trasmissione standard come quella supportata dal WiFi basata sullo standard IEEE 802.1. Nel caso invece sia necessario utilizzare bande alternative tipo 802.15.4 o speciali funzionalitá dell'application-layer é necessario fornire il dispositivo smart di un gateway che fará quindi da intermediario tra le differenti tecnologie e potrá fornire specifiche funzionalitá alla rete.
Modelli di questo genere vengono applicati per esempio nel contesto di dispositivi mobile come smartphone, e accessori *wearable*. I primi svolgono la funzione di gateway mentre i secondi di dispositivi con sensori e attuattori integrati. Nel caso specifico degli smartphone va tenuta presente la possibilitá di connettivitá intermittente tra dispositivi che non deve peró inficiare sul funzionamento generale del sistema. Questo continuerá a svolgere i suoi compiti anche se scollegato ad internet e alla riconnessione con il gateway sará di nuovo in grado di inviare e ricevere segnali e nel caso spedire tutti i dati accumulati nel frattempo. I dispositivi mobile forniscono, inoltre, un flessibilissimo sistema di aggiornamento dato dal sistema operativo (iOS/Android), e utilizzato dai produttori di dispositivi smart per

implementare metodi di comunicazione proprietari e per mantenere aggiornati i propri prodotti. Questo ha portato ad una situazione di totale assenza di interoperabilitá che costringe l'utente ad avere perloppiú multiple applicazioni installate per interagire con i rispettivi dispositivi posseduti.

2.1.4 Protocolli

Perché l'IoT riesca a diffondersi per un uso di massa é prima necessario risolvere problemi fondamentali come la mancanza di standard. L'adozione di un approccio unificato potrebbe senz'altro ridurre l'attuale frammentazione del mercato portando a una maggiore garanzia di interoperabilitá e compatibilitá funzionale all'uso. Un primo gruppo che si é posto in prima linea per la definizione di questi standard é l'Open Interconnect Consortium (OIC) a cui é seguita la creazione della AllSeen Alliance definita come "Linux Foundation collaborative project" la cui mission si sintetizza nel supportare tecnologie di ampia adozione e nell'accelerare lo sviluppo di framework di comunicazione basati sulla tecnologia opensource. Quest'ultima si é poi unificata assieme all' Open Connectivity Foundation (OCF) sotto questo nome. Tra le altre fondazioni o gruppi di rilievo nel settore troviamo l'Eclipse IOT foundation, il Thread group, l'International Society of Automation (ISA) e il W3C i quali, negli anni, hanno proposto o sviluppato numerosi protocolli che permettono ai dispositivi di comunicare tra loro in diversi modi [16]. Qui di seguito é riportata una categorizzazione di essi secondo il loro uso applicativo ed un elenco suddiviso secondo lo stack ISO/OSI con una breve descrizione.

- Infrastruttura (ex: 6LowPAN, IPv4/IPv6, RPL)
- Identificazione (ex: EPC, uCode, IPv6, URIs)
- Comunicazione / Trasporto (ex: Wifi, Bluetooth, LPWAN)
- Scoperta (ex: Physical Web, mDNS, DNS-SD)
- Protocolli dati (ex: MQTT, CoAP, AMQP, Websocket, Node)

- Gestione dispositivi (ex: TR-069, OMA-DM)
- Sematica (ex: JSON-LD, Web Thing Model)
- Framework multi-layer (ex: Alljoyn, IoTivity, Weave, Homekit)

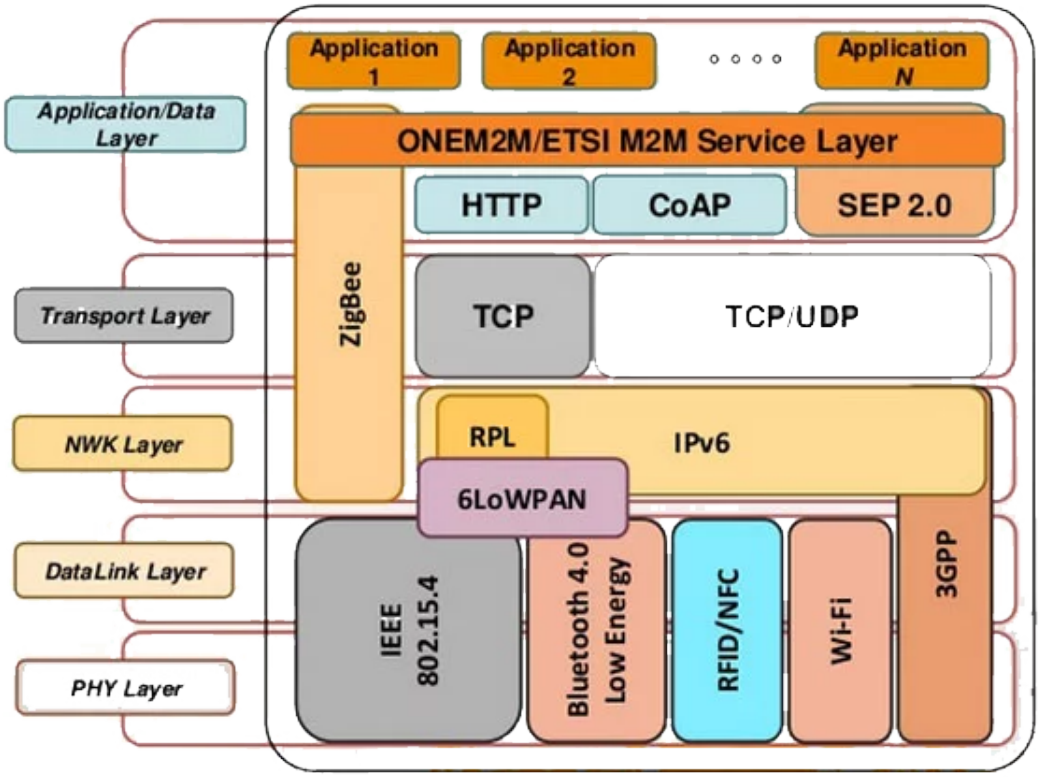

Figura 2.3: Schema di mapping con lo stack ISO/OSI

Protocolli di comunicazione a basso livello (Fisico)

Permettono il collegamento diretto ai dispositivi. Vengono scelti a lato applicativo principalmente in base a necessitá legate al range di comunicazione e al consumo energetico. Nonostante vi siano protocolli wireless giá affermati nell'ambiente delle comunicazioni questi non sono propriamente adatti all'uso in sistemi IoT, poiché nati e sviluppatisi in tempi precedenti alle soluzioni IoT e

quindi propensi ad altre applicazioni. Sono state presentate negli ultimi anni molte nuove proposte, per esempio l'utilizzo innovativo delle reti sub-Ghz. Reti operanti al di sotto di 1Ghz, le quali offrono interessanti vantaggi in ambito IoT, rispetto alle trasmissioni sulla classica banda dei 2,4Ghz, in termini di consumi e di distanza di comunicazione tra nodo e nodo (sono infatti definite reti a *long range*).

Per le comunicazioni IoT é stato introdotto anche un nuovo standard, IEEE 802.15.4, riferito all'accesso al livello fisico ed alle reti wireless a basse frequenze (LR-WPANs). E' mantenuto dall'802.15 working group e rappresenta la base di molte specifiche per la comunicazione (ZigBee, ISA100.11a, WirelessHART, e MiWi), le quali estendono lo standard descrivendo i livelli superiori.

SigFox: tecnologia a basso consumo energetico basata su comunicazioni wireless. Permette trasferimenti di piccole quantitá di dati a velocitá variabili 10-1000 bits/s entro un range di 50km. E' impiegato principalmente per le comunicazioni M2M, in ambito medico (monitor paziente), smart city (illuminazione e sensori ambientali) e dispositivi di sicurezza (telecamere).
Cellular: tecnologia che trova il suo massimo impiego con dispositivi che necessitano di trasmettere dati su grandi distanze necessariamente collegati ad una fonte di alimentazione costante (dato l'alto consumo energetico). Utilizza le reti cellulari (GSM/3G/4G) fornendo connessioni affidabili e veloci. Non é adatta a comunicazioni M2M o per comunicazioni in rete locale.
6LoWPAN: é il protocollo per le comunicazioni IoT piú usato poiché si base sul protocollo IP standard. Permette la connessione diretta tra reti IP senza necessitá di traduzione dei segnali o di entitá intermediarie come gateway e proxy. E' stato creato dall'IETF per far uso degli standard IP giá esistenti, ma tramite collegamenti a basso consumo energetico come previsto dall' IEEE802.15.4. Si propone come livello intermediario tra il livello MAC e quello di rete (IPv6). Supporta differenti topologie di rete (mesh e topologie a stella), ha un basso consumo di banda (oltre che energetico) e supporta indirizzi di differenti lunghezze. Questo protocollo opera solo su frequenze nel range di 2.4 GHz con una velocitá massima di trasmissione di 250 kbps.

ZigBee: protocollo creato dalla ZigBee Alliance. E' il principale contendente di 6LoWPAN e anch'esso si basa su reti a basso consumo secondo lo standard IEEE802.15.4. E' stato creato per diventare il protocollo standard di comunicazione in campo IoT per reti piccole ed a basso consumo energetico ma che allo stesso tempo hanno necessitá di trasmettere a modeste distanze. Cosí come 6LoWPAN trasmette solo su onde a 2.4 GHz a velocitá di 250 kbps con una massima gittata di 200m.

Thread: conosciuto come il "protocollo di Google",é un prodotto opensource rilasciato dal Thread Group (tra le quali fondatrici compare la Nest, controllata da Google). [8] Basato su reti dello standard IEEE802.15.4 come ZigBee, é una combinazione di 6LowPAN, IPv6 e UDP con aggiunte importanti apportate dal Thread Group, che lo rendono sicuro, leggero e compatibile con il routing IPv6. Il protocolloé stato rilasciato come progetto opensource alla versione 1.1.1 da Luglio 2017, scatenando forti reazioni nel mondo dello sviluppo software IoT. Data la sua versatilitá e fruibilitá gratuita é utilizzato spesso per progetti privati, principalmente nell'ambito della domotica, in rivalitá con ZigBee.

BLE: anche noto come Bluetooth,é specifico, nell'ambito IoT, per comunicazioni a corta distanza con poca banda e necessitá di bassa latenza. I vantaggi che offre sono il basso consumo energetico, la velocitá di setup ridotta e la possibilitá di interconnettere un numero illimitato di nodi (supporto di reti con topologia a stella). Usa frequenze a 2.4 GHz con velocitá fino a 3 Mbps entro una massima distanza di 100m.

RFID: nonostante sia una tipologia di tecnologia piuttosto che un protocollo viene citato in questa sezione per il ruolo di trasmissione che ricopre nelle comunicazioni con i dispositivi. I sistemi RFID sono composti da un lettore e un trasponder a radio frequenze (RF TAG). I secondi sono programmati in modo da contenere informazioni e possono essere "letti" tramite il lettore a distanza. Possono essere di due tipi: attivi - sono alimentati a batteria, usano alte frequenze e sono piú costosi; passivi - usano frequenze piú basse e non necessitano di alimentazione. Poiché l'informazione viene programmata permanentemente nel tag, gli usi pratici sono limitati. Per esempio viene utilizzato nel campo dello smart shopping per mantenere

traccia del magazzino. Permette comunicazioni a velocitá di trasmissione fino a 4Mbs.
NFC: altro esempio di tecnologia di comunicazione con dispositivi fisici, che prevede lo scambio di informazioni mettendo a contatto i dispositivi. Il funzionamento é similare a quello degli RF TAG ma le distanze sono piú ravvicinate e le possibili applicazioni molto piú articolate. I tag NFC possono agire in tre modalitá diverse: passivi - possono solo essere letti e vengono impiegati ad uso identificativo; attivi - permettono lettura e scrittura; modalitá peer-to-peer - permettono il transito di informazioni. Molto usato nel campo mobile e dei pagamenti elettronici. Agiscono sulla banda di frequenza dei 13.56 MHz con comunicazioni fino a 424kbs.
Z-Wave: é un protocollo a basso consumo energetico sviluppato da Zensys principalmente per la domotica. Il suo funzionamento dipende da due componenti: il controller il quale puó inviare messaggi, e lo slave che puó solo eseguirli, rispondere o reinviarli. Il sistema forma una rete mesh, dove ogni componente funziona come una sorta di repeater. Permette collegamenti fino a 50 nodi ed é ottimo per scambi di piccoli pacchetti di dati a basse velocitá (100kbs) entro un range massimo di 30mt da nodo a nodo. Il vantaggio che offre é che gli slave possono funzionare anche con fonti energetiche limitate, come piccole batterie.

Protocolli di rete (Network)

IP - v4 e v6: la trasmissione dei dati avviene tramite pacchetti e i partecipanti alla comunicazione sono identificabili tramite un'indirizzo univoco all'interno della rete. La versione dominante del protocollo IP é IPv4, (Internet Protocol Version 4) ma ci si sta dirigendo sempre piú verso l'adozione totale dell'IPv6 per necessitá legata allo spazio d'indirizzamento limitato della versione precedente. A dicembre 2014 veniva ancora utilizzato l'IPv4 dal 95% delle connessioni attive globalmente, mentre oggi, 5 anni dopo, la percentuale é scesa sotto il 70

Protocolli di trasporto

UDP: protocollo *connectionless* che funziona cioé senza aspettare che sia stata stabilita una connessione. Le informazioni vengono inviate in ogni caso senza avere la garanzia che queste arrivino realmente al destinatario. E' il piú economico in quanto ad overhead sui dati scambiati ed é preferibile quando la velocitá e l'efficienza sono piú importanti dell'affidabilitá, ad esempio per trasmissioni realtime di dati.

TCP: protocollo *connection-oriented*, ossia che richiede di creare un circuito virtuale predeterminato tra sorgente e destinatario, prima di iniziare lo scambio di dati vero e proprio. Piú vecchio rispetto all'UDP,é stato introdotto nel 1980, e trova il suo maggiore impiego nella navigazione internet.

Protocolli a livello applicativo (Data)

Sono protocolli di messaggistica usati per lo scambio di dati ad alto livello tra persone, applicazioni e dispositivi. Il protocollo viene scelto anche in base al paradigma di comunicazione supportato:

- request/response (polling), le informazioni vengono richieste e si rimane in attesa di una risposta, generalmente questo paradigma introduce un timer in modo da fare richieste continue per avere sempre il dato aggiornato.
- publish/subscribe (push-based), i dispositivi che comunicano per mezzo di questo paradigma possono iscriversi a certi topic per cui vogliono ricevere aggiornamenti oppure pubblicare messaggi su tali canali. Un'altro componente chiamato broker si occuperá di gestire iscrizioni, pubblicazioni e lo scambio dei messaggi. In questo modo i client vengono notificati solo al momento del bisogno evitando quindi di saturare la rete con continue richieste di aggiornamenti, non necessarie.

HTTP: Uno dei protocolli maggiormente utilizzati per la connessione dei dispositivi al cloud, mediante l'uso dell'architettura REST (REpresentation State Transfer). Nonostante ció, non essendo stato progettato per l'utilizzo specifico nell'ambito

IoT, presenta numerosi punti a sfavore in confronto agli altri protocolli di messaggistica piú recenti. Funziona secondo il pattern request/response il ché prevede un alto livello di traffico causato da polling sui dispositivi e un overhead sui dati scambiati molto alto a causa dell'header ricco di informazioni. Con l'introduzione dell'HTTP/2 é stata resa piú efficiente la connessione e l'uso delle risorse oltre ad aver introdotto la compressione dell'header e la possibilitá di aprire piú connessioni nella stessa istanza TCP.

CoAP: (Constrained Application Protocol) é stato progettato dall'Internet Engineering Task Force (IETF). E' un protocollo pensato principalmente per essere usato con dispositivi con vincoli prestazionali o di consumo energetico. E' nato con l'intento di fornire un modello di comunicazione uno a uno mediante un'infrastruttura client server con paradigma request/response ed interoperare nativamente con HTTP e RESTful. Nel tempo ne sono state estese le potenzialitá con l'intervento del CoRE Group, implementando: il supporto per multi-cast, per la scoperta di risorse fornite da altri servizi CoAP e un servizio di sottoscrizione simile al broker MQTT ma con il paradigma observer. Lavora sul protocollo di trasporto UDP, il quale non garantisce la consegna dei pacchetti ma in compenso é molto veloce e efficiente per applicazioni che trattano dati *time-sensitive*.

MQTT: protocollo nato nel 1999, con l'obiettivo di gestire le connessioni M2M in maniera piú efficiente rispetto alle offerte del panorama tecnologico del tempo. Le caratteristiche principali di MQTT sono quelle di essere leggero, open, e semplice da implementare. Permette la distribuzione efficiente di messaggi da uno a molti destinatari utilizzando il paradigma publish/subscribe (client-server) il quale inoltre permette di mantenere basso il livello di traffico della rete. Questo insieme di caratteristiche rendeno il suo funzionamento ottimale anche quando si ha a che fare con reti non perfette in termini di stabilitá della connessione e di dispositivi con limitazioni di memoria o prestazioni. Lavora in modo asincrono in cima allo stack TCP/IP ed é progettato per avere un basso overhead comparato con altri protocolli dello stesso livello. Un sistema MQTT é formato da due parti: il client - qualsiasi dispositivo (controller o server ad esempio) che ha in esecuzione la libreria MQTT ed é connesso ad un broker; il broker - il quale gestisce le connessioni client,

filtrando i messaggi e decidendo in base alle iscrizioni a chi destinarli.
XMPP: (Extensible Messaging and Presence Protocol) nato anch'esso nel 1999 inizialmente per comunicazioni in tempo reale tipo chat, é stato poi deprecato poiché non riusciva a soddisfare le applicazioni piú moderne. Riscoperto negli ultimi anni, ne sono state rivalutate le potenzialitá per l'ambito IoT in quanto leggero e abile a trasmissioni di piccoli messaggi con necessitá di bassa latenza. Lavora sul protocollo TCP e fornisce un modello sia publish/subscribe (asincrono) che request/response (sincrono).
AMQP: (Advanced Message Queuing Protocol) é un protocollo descritto da uno standard aperto che fornisce funzionalitá di instradamento e gestione della coda dei messaggi. Definisce il comportamento del broker e del client di messaggistica al fine di rendere interoperabili le implementazioni di diverse case produttrici. Il grande vantaggio che offre é quello di salvare il messaggio e spedirlo in modo affidabile anche in presenza di problemi di connettivitá. Come nel caso dell'MQTT il livello di affidabilitá é specificato nel messaggio stesso che presenta un campo definibile dal broker, in tre modalitá:

1. "Al massimo una volta", il messaggio viene spedito una volta sola, che abbia successo o meno

2. "Almeno una volta", il messaggio verrá sicuramente consegnato almeno una volta

3. "Esattamente una volta", il messaggio verrá consegnato una volta sola

Websocket: protocollo sviluppato come parte di HTML 5 per facilitare la comunicazione con il TCP. Non utilizza né request/response né publish/subscribe. La sessione viene inizializzata in maniera simile all'handshake della connessione HTTP tramite un'interfaccia Javascript dedicata. Una volta instanziato il canale di comunicazione, client e server possono scambiare messaggi in maniera asincrona e bilaterale senza vincoli.

2.2 Lifecyle dei dati

Dopo aver analizzato un sistema IoT dal punto di vista strutturale, concentriamoci ora sulla parte funzionale, ovvero i dati. I dispositivi IoT generano e trasferiscono grandi quantitá di dati e mantenere traccia di tutte le connessioni, gestire il traffico, l'analisi e la consumazione delle informazioni in modo sicuro ed efficiente non é mai scontato.
Inoltre quando si parla di gestione dei dati non é piú possibile generalizzare, ma diventa necessario entrare nel dettaglio del contesto applicativo poiché ogni sistema presenta difficoltá differenti che vanno affrontate nello specifico. Per esempio, nel caso di un sistema dedicato alle smartcity si presenta la necessitá di coordinare migliaia di dispositivi connessi contemporaneamente e gestirne la relativa mole di dati generati, mentre in un sistema aziendale la problematica principale potrebbe essere, piuttosto, la gestione realtime delle rilevazioni provenienti da sensori collegati a macchine utensili.
Ad un alto livello di gestione possiamo adottare due strategie differenti a seconda della conoscenza che abbiamo sui dati (struttura, contenuto, etc):

- Gestione di dati conosciuti: il sistema conosce i dati che raccoglie ed é in grado di processarli e analizzarli nel dettaglio

- Gestione di dati sconosciuti: il ”cosa” fare con i dati raccolti non é definito perció questi vengono riportati cosí come sono stati raccolti per essere analizzati ed elaborati con metodi piú avanzati. Poiché non filtrati, e scremati da informazioni di contesto, la mole di dati scambiati in rete sará sicuramente maggiore ma solitamente in soluzioni simili non é previsto *reporting* sostanzioso (anche per il minor livello d'affidabilitá sui valori fornito) e sará possibile computare i risultati senza stringenti obblighi temporali.

Nel primo caso, ogni layer architetturale deve gestire i dati a seconda delle necessitá e guidarli verso lo stato successivo del loro ciclo di vita, il quale inizia al livello fisico (sensori e dispositivi) con la raccolta dei dati, procede mediante meccanismi di trasporto e comunicazione verso il livello successivo dove avviene

l'elaborazione e l'analisi, e finisce nella fase di visualizzazione e monitoraggio nel livello applicativo passando per il processo di salvataggio in cloud o in locale.
Scomponiamo ora le fasi relative alla gestione dei dati per descriverne meglio gli aspetti specifici (Figura 2.4):

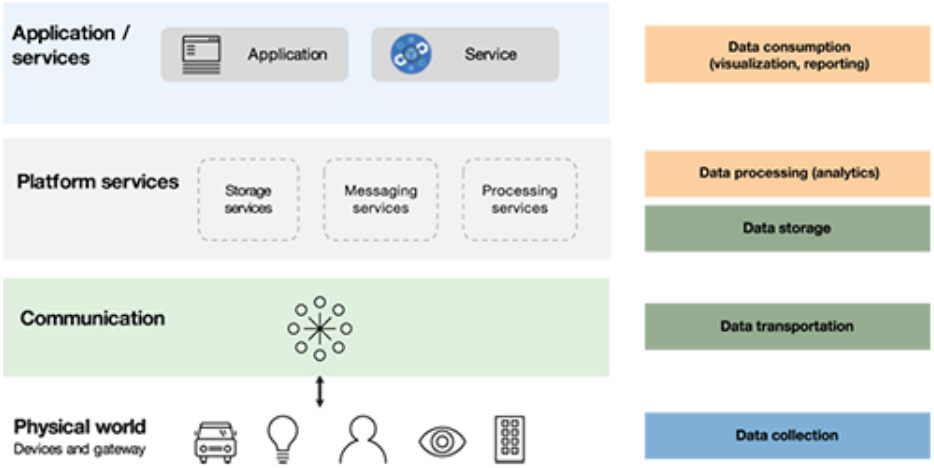

Figura 2.4: Fasi di gestione dei dati

2.2.1 Raccolta dei dati

La raccolta dei dati é il primo passo del loro ciclo di vita. Essi vengono generati dai sensori, i quali possono essere pochi dispositivi sparsi per casa fino ad arrivare a migliaia distribuiti in intere cittá a seconda del sistema in considerazione. Possono essere integrati in dispositivi statici come rilevatori di fumo, o in movimento come la dashcam di un'automobile. Inoltre, come descritto nella sezione dei protocolli di comunicazione, essi sono di molti tipi e possono variare di produttore in produttore. La varietá d'interfaccia di questi dispositivi é un aspetto da tenere in considerazione durante questa fase.
A seconda dello standard utilizzato e dal tipo di dato sará poi necessario aggregare o filtrare i dati, prima di mandarli al server. Solitamente questa fase prevede anche un controllo sulla qualitá del dato, poiché spendere risorse (batteria, banda e computazione) per inviare dati non utilizzabili non é un approccio molto valido. Le

modalitá di implementazioni possono peró cambiare a seconda dell'infrastruttura e dei dispositivi che la compongono.

Qualitá dei dati (QoD)

Coerenza: la coerenza dei dati si riferisce alla correlazione tra una serie di punti segnalati dallo stesso sensore in un breve periodo di tempo. Ad esempio, non si prevede che un sensore di temperatura segnali letture molto diverse o subisca sbalzi di temperatura improvvisi entro pochi secondi, o ancora, non é previsto che un sensore di geolocalizzazione segnali posizioni distanti molti chilometri in pochi secondi. Se si verifica un caso del genere, é probabile che il sensore sia difettoso oppure scalibrato. In alcuni casi controllati, é possibile includere valori provenienti da multipli sensori e quindi correlare statisticamente le letture per determinare quali dati sono incoerenti. Un sensore che segnala regolarmente dati incompatibili con le letture di altri sensori potrebbe essere automaticamente segnalato per un controllo dello stato.
Completezza: la completezza dei dati si riferisce alla disponibilitá di tutti i punti all'interno di un determinato spazio; ad esempio, dati mancanti potrebbero essere dati relativi a serie temporali non rilevate. Sono considerati dati "di qualitá" anche dati costruiti includendo dati rilevati e validati da sensori differenti da quello in analisi, in modo peró da ottenere una rilevazione temporale completa. Laddove pertinente, quindi, completezza puó fare riferimento alla capacitá di combinare e correlare dati provenienti da piú sensori o persino da altri sistemi di informazione.
Tempestivitá: poiché gran parte dei dati raccolti da sistemi IoT sono dati in tempo reale o quasi, determinare se i dati del sensore sono arrivati in tempo nel punto richiesto della rete diventa fondamentale. Avere dati con questa proprietá permette di agire tempestivamente per prevenire o contenere il verificarsi di eventi e questo aspetto diventa ancora piú critico nel caso sia necessario sincronizzare letture di piú sensori.
Affidabilitá: l'affidabilitá dei dati é di grande rilevanza in generale, infatti un sistema IoT basato su dati non affidabili non avrebbe ragione di esistere poiché non potrebbe raggiungere alcuno scopo utile. Si pensi ad esempio a che conseguenze

potrebbero portare letture sbagliate di sensori collegati a bracci robotici per la chirugia medica o installati su automobili con guida autonoma. Un dato é affidabile, quando le misurazioni di un sensore sono accurate e ripetibili nel tempo. La precisione della lettura si riferisce alla capacitá di fornire ad ogni lettura, dei valori con una certa sensibilitá d'errore. La ripetibilitá si riferisce al fatto che il sensore sará in grado di produrre la stessa misurazione con la stessa precisione se inserito nello stesso scenario. Ad esempio, se un sensore di geolocalizzazione viene portato nello stesso angolo di strada in due momenti diversi, questo deve produrre le stesse medesime letture, con la precisione richiesta.
Il legame tra la qualitá del dato (QoD) e la qualitá del servizio (QoS) é molto stretto, infatti agendo sul primo si influenza in modo diretto anche il valore del secondo e quindi l'esperienza dell'utente finale all'uso applicativo [20]. Inoltre, i dispostivi stessi possono essere mal calibrati o possono necessitare regolazioni periodiche: gli ambienti industriali solitamente prevedono l'uso di strumenti molto sensibili e precisi. E' perció necessario far affidamento su ottimi metodi di controllo con frequenti raccolte di dati, oppure effettuare periodiche verifiche manuali.

2.2.2 Normalizzazione e modellazione

Poiché i dati provengono da piú e varie sorgenti questi spesso si presentano disorganizzati (*unstructured data*) ed é necessario un processo di elaborazione nel quale viene assegnata una struttura ad essi, in modo che assumano significato e siano usabili dall'applicazione finale. Per comunicare dati in maniera strutturata seguendo regole standard per il contesto, oppure per applicare strutture dati specifiche per l'infrastruttura in questione, vengono utilizzati i metadati, ossia dati che contengono informazioni sui dati. L'uso di questi, permette di gestire finemente i flussi di informazioni provenienti dai dispositivi, applicando eventualmente anche modelli piú generali per ricoprire tutte le casistiche di scambio dati tra un gruppo di specifici dispositivi e l'infrastruttura centrale. A causa della varietá di dispositivi e delle loro applicazioni nel mondo reale é piuttosto difficile definire una struttura standard che fornisca il giusto livello di dettaglio. Si rischia di generalizzare troppo, avendo quindi uno scambio di dati eccessivo rispetto al necessario, o di

generalizzare troppo poco. E' giusto quindi non adottare una metodologia "one size fits all" in questo caso, ma cercare di gestire i dati al meglio mantenendoli separati in gruppi distinti a seconda del contesto.

2.2.3 Anlisi dei dati: processing

Device e sensori generano una grandissima quanitá di dati grezzi i quali prima di essere mostrati all'utente, per mezzo di grafici o tabelle, il piú delle volte necessitano di una consistente fase di elaborazione per poterne estrarre informazioni utili e significative. Questa fase viene divisa in *preprocessing* e *processing*.
I dati standardizzati e strutturati in modo uniforme devono infatti essere filtrati in modo da rimuovere infromazioni inutili, dati ridondanti e/o errati in modo da migliorarne l'accuratezza. Possono anche essere combinati con dati provenienti da fonti diverse per arricchirne il significato intrinseco ed estrarre correlazioni tra i vari campi nella fase di *processing*. L'analisi dei dati é peró un'operazione costosa in termini di computazione ed é necessario definire in fase di design dell'infrastruttura a che livello é meglio effettuare una certa operazione, valutando anche i tempi e i costi che comporta lo spostare i dati da un componente ad un altro. A questo scopo vengono definite tre politiche di gestione dei dati.
Politiche di **raccolta dei dati**, che determinano il modo in cui i dati vengono raccolti e inviati per ulteriori elaborazioni. I dati possono essere inviati come dati non elaborati oppure immediatamente dopo l'elaborazione iniziale, a seconda delle esigenze strutturali e in base alle capacitá del dispositivo. Tali politiche stabiliscono anche il protocollo di sicurezza che garantisce che i dati ricevuti siano dati giá considerabili validi.
Politiche di **filtraggio dei dati**, che definiscono la modalitá di gestione dei dati dopo la raccolta. Per esempio puó essere piú conveniente ai fini del sistema inviarne solo un aggregato che ne racchiuda peró il valore per risparmiare banda e batteria dei dispositivi oppure effettuare un controllo al momento della raccolta per inviare solo i dati considerabili validi.
Politiche di **caricamento e salvataggio dei dati**, che definiscono quali dati vadano memorizzati per future elaborazioni senza perdita di significato e di valore

intrinseco. Tra l'enorme quantitá di dati generati dai dispositivi IoT, solo gli eventi di interesse, devono essere caricati sul cloud per l'elaborazione e l'archiviazione. Quindi, in genere, vengono inviati solo i dati filtrati e o elaborati mentre i dati grezzi vengono raramente mantenuti per l'archiviazione a lungo termine. Tuttavia, ciascun livello della rete potrebbe archiviare alcuni dati rilevanti ai fini di revisioni e verifiche dettagliate, identificazione dei difetti o controllo dello stato.

Queste problematiche sono di grande rilevanza soprattutto nei sistemi realtime i quali devono essere in grado di:

- Gestire una grande quantitá di dati con traffico in tempo reale
- Fornire un rapido accesso ai dati, poiché un ritardo li renderebbe obsoleti e quindi non piú rilevanti o usabili per prendere decisioni. Ad esempio un sensore di frenata per una macchina con guida assistita o autonoma.
- Gestire errori e mancate letture
- Gestire l'interoperabilitá dei suoi componenti e l'estensibilitá del servizio senza doverne interrompere l'esecuzione

In un'architettura IoT l'analisi puó essere effettuata in tutti e 4 i livelli visti ma quando ci si trova a dover rispondere a domande come: "Sto per fare un incidente? Mi devo fermare?" oppure "Dopo quanti battiti cardiaci non rilevati devo allarmarmi?" diventa fondamentale mantenere i dati il piú vicino possibile alla fonte per analizzarli e ottenere risposte rapide. Di contro, peró, piú spostiamo i dati verso il core del sistema piú potremmo fare uso di maggiori risorse di calcolo per ottenere dettagli dai semplici valori, applicando sistemi avanzati di previsione, rilevazione e decisione tramite retrospettive, tecniche di aggregazione di dati e modelli di *machine learning*. [**?**]

2.2.4 Salvataggio e consumo

I dati provenienti dai dispositivi dovranno poi essere salvati, ma con quale criterio viene fatto ció?

Anche questa volta dobbiamo scendere nello specifico dell'infrastruttura per analizzare il tipo di servizio, ossia lo scopo per cui i dati vengono raccolti e per il quale saranno utilizzati. Ad esempio, per applicazioni con flussi di dati continui in streaming, salvare in cloud i dati grezzi direttamente alla rilevazione diverrebbe molto oneroso, anche utilizzando una linea internet ad alta velocitá; perció in questi casi sará necessario optare per un salvataggio in locale per poi spostare il contenuto periodicamente in database remoti. In alternativa si potrebbe optare per un'opzione piú parsimoniosa in termini di spazio e decidere di salvare solamente i risultati derivanti dall'elaborazione dei dati, perdendo peró la possibilitá di effettuare ulteriori analisi se in futuro avessimo bisogno di dati oggi classificati come non rilevanti.
Salvare tutto é possibile ma sconsigliabile. Anche se il costo di salvataggio dei dati sta diminuendo con il passare del tempo, il volume per i dati di un sistema IoT, é solitamente un fattore caratterizzante, e resta comunque un fattore importante da non sottovalutare.
Anche l'allocazione logica all'interno dei database non é banale poiché a seconda del tipo di dato, della quantitá e del suo utilizzo futuro potrebbe essere piú conveniente l'uso di basi di dati relazionali piuttosto che non.
Alla fine del ciclo di vita avviene l'effettivo consumo dei dati ossia la visualizzazione, il reporting e la reazione del sistema a certi valori o eventi. In questo modo sistemi piú interconnessi e automatizzati potranno agire autonomamente inviando segnali ad attuatori per modificare o ripristinare lo stato del sistema stesso, alternativamente l'azione spetterá all'utente, il quale verrá allertato per mezzo di segnalazioni automatiche.

2.2.5 Privacy e sicurezza

Durante tutti questi passaggi é necessario, inoltre, tenere in considerazione il fattore sicurezza dei dati. Non esiste ad oggi uno strumento del diritto internazionale vincolante, che disciplini in maniera uniforme e complessiva internet, sia per quanto riguarda la tutela della riservatezza dei dati personali su scala globale sia per quanto riguarda la *cyber security*. Vari trattati possono essere applicati in maniera laterale all'argomento, quali: la dichiarazione universale dei diritti umani

del ’48 e il patto internazionale sui diritti civili e politici del ’66, ma sono trattati che sono stati negoziati in un momento in cui il digitale non esisteva quindi sono applicabili al tema solo in via interpretativa.
Negli ultimi anni le regole e leggi di sicurezza per il trattamento e il salvataggio di questi dati sono state aggiornate per tutelare maggiormente gli utilizzatori dei servizi (COPPA, HYPPA, GDPR). Molto discusse sono state le azioni normative dettate dall’UE nel 2016 per difendere il diritto alla privacy delle persone e assicurare che i soggetti che si occupano del trattamento dei dati lo facciano in modo lecito.
Le principali linee guida da tenere in considerazione per la privacy dei dati sono:

- **Classificazione dei dati basata sulla sensibilitá:** I dati vanno definiti in modo chiaro in termini di informazioni di identificazione personale (PII), informazioni personali sensibili (SPI), informazioni sicure e informazioni pubbliche.

- **Controlli e validazione sui dati caricati:** Dopo il caricamento dei dati, i dati devono essere protetti da accessi e manipolazioni non autorizzate. Controlli necessari devono essere stabiliti per ciascuna categoria di dati.

- **Mantenimento della privacy dei dati:** La soluzione deve applicare tecniche come il partizionamento dei dati, l’anonimizzazione o la crittografia dei dati per ridurre i rischi dovuti a falle del sistema o accessi non autorizzati.

- **Controllo d’accesso:** Va stabilito un controllo granulare degli accessi in base al ruolo, per ciascuna sezione dei dati. Queste misure di sicurezza devono impedire l’accesso tramite accessi generici e imporre che ogni utente utilizzi le proprie credenziali individuali.

- **Protezione nel contesto della multi-tenancy:** Dato che quasi tutti i dati IoT sono archiviati e gestiti nel cloud, la sicurezza multi-tenancy diventa fondamentale. Devono essere implementate misure di sicurezza durante le fasi critiche per evitare mescolanza di dati appartenenti a sorgenti distinte.

Capitolo 3

State of the art

L'IoT rappresenta una promessa per il futuro: per questo in molti, sia piccoli privati che grandi aziende, stanno investendo per assicurarsi una posizione di rilievo in questo campo oppure approfittare delle potenzialitá che offre prima degli altri per avere un vantaggio competitivo nel mercato.
Ma perché oggi? Le tecnologie su cui fa affidamento l'IoT esistono da piú di vent'anni eppure solo negli ultimi anni c'é stata una grande diffusione nelle masse. Le innovazioni tecnologiche hanno bisogno di tempo per guadagnare consenso e fiducia da parte della gente e soprattutto devono trovare un posto all'interno della societá per integrarsi nella vita quotidiana senza stravolgerla. Basti pensare allo sviluppo e alla diffusione del web fino ai giorni nostri. Ci sono voluti quasi 30 anni prima che Internet, da una rete interna al CERN per la comunicazione e pubblicazione di articoli scientifici diventasse la rete di interconnessione globale che conosciamo oggi.
Negli ultimi anni, peró, sono stati fatti passi enormi nella diffusione di sistemi IoT e sono aumentate a dismisura le applicazioni pratiche. I dispositivi embedded e la tecnologia hardware che li compone offre oggi maggiori possibilitá. Dispositivi indossabili, smartphone e sensori sono diventati d'uso comune tra la gente introducendo anche nuove metodologie d'interfacciarsi con la tecnologia e di relazionarsi con i dati. L'internet é molto piú accessibile grazie ai notevoli miglioramenti nelle infrastrutture fisiche di connessione al punto che l'uso dei suoi servizi é diventato

quasi indispensabile nella vita di tutti i giorni, e la societá stessa in cui viviamo é oggi molto piú aperta alla tecnologia e all'innovazione.

3.1 Panorama tecnologico

Le possibilitá d'applicazione nel mondo reale sono veramente vastissime ma essendo ancora ad uno stadio di definizione questa tecnologia non é pienamente sfruttata in tutti i campi [17, 24]. Riportiamo una tabella riassuntiva dei maggiori traguardi e applicazioni nei vari settori.

SETTORE	APPLICAZIONE	ESEMPI
UMANO E SOCIALE	Instaurare e mantenere rapporti sociali; conoscere l'ubicazione di oggetti e persone; relazionarsi in modo piú immediato.	- Invio automatizzato di messaggi d'auguri. - Invio di messaggi relativi ai propri spostamenti. - Insegne che segnalano lo stato di uno sportello al pubblico. - Aggiornamento automatico delle informazioni relative ad attivitá sociali su portali e social networks tramite tag RFID di posizione.
	Sfera personale	- Fitness tracker, dispositivi medicali per il monitoraggio dello stato di salute e di benessere, e diete su misura con dispositivi ingeribili o indossabili. - Dispositivi sincronizzati con agenda e calendario per ricordare all'utente eventi e impegni. - Identificazione e rintracciamento di piccoli dispositivi per rintracciare portafoglio o mazzi di chiavi. - Suggerimenti per negozi e ristoranti su base individuale grazie a rilevazioni d'utilizzo dei dispositivi, ricerche frequenti, e rilevamenti GPS.
	Gestione delle attivitá sociali	- Pubblicizzazione automatica di feste ad eventi in base alla partecipazione e al posizionamento. - Sincronizzazione tra calendario, agenda e dispositivi vari (orologio, telefono, TV, bilancia, etc) in base ad eventi, attivitá di ricerca frequenti e posizione.

Continua

Tabella 3.1 – *Continua dalla pagina precedente*

SETTORE	APPLICAZIONE	ESEMPI
CASA	Automazione nelle routine casalinghe e interazione tra i dispositivi	- Assistente vocale casalingo che puó comandare i dispositivi di casa secondo routine prestabilite o comandi vocali impartiti dall'utente. - Frigoriferi che effettuano in automatico le ordinazioni al supermercato e che consigliano menu e ricette.
	Sistemi di sorveglianza privata	- Citofoni e videocamere comandabili da smartphone. - Baby-monitor musicali che reagiscono ad eventi sonori e inviano notifiche egli endpoint configurati.
	Sistemi atti al risparmio e anti-spreco	- Prese di corrente intelligenti che si scollegano dalla rete elettrica in determinate fasce orarie o a seconda dell'utilizzo di corrente in atto. - Contatori di corrente, luce e gas, monitorabili a distanza. - Rubinetti e lavandini con applicazioni e sensori per evitare sprechi.
LAVORATIVO	Industrial IoT (Industria 4.0)	- Braccialetti per monitorare l'operato dei lavoratori. - Collegamento di dispositivi di monitoraggio a catene di montaggio e macchinari per lavorazioni industriali. - Cuffie antirumore intelligenti che vengono attivati e disattivati da sensori ambientali o da input impartiti in base ai processi produttivi in atto.
	Allevamenti e Precision Farming	- Monitoraggio di colture e allevamenti tramite sistemi di riconoscimento per prevenire diffusione di malattie e curare o eliminare i difetti. - Droni di fertilizzazione, spargimento di pesticidi o antibatterici. - Sistemi di irrigazione automatizzati in base a sensori di monitoraggio del terreno. - Identificazione tramite tag degli animali o degli appezzamenti di terra con mantenimento di relative statistiche per azioni mirate.

Continua

Tabella 3.1 – *Continua dalla pagina precedente*

SETTORE	APPLICAZIONE	ESEMPI
		- Sensori per la gestione dello stato e dell'ambiente all'interno delle serre.
CLIENTI E SERVIZI	Negozi IoT	- Negozi dove sensori e videocamere permettono acquisti senza necessitá di casse e personale addetto (Amazon shops). - Analisi comportamentali e campagne sconti mirate per gli utenti. - Ottimizzazione dei processi d'inventario e di gestione degli articoli per i negozi.
	Enhanced gaming rooms	- Postazioni per la realtá aumentata con sensori in comunicazione con l'ambiente di gioco. - Stanze sensoriali.
	Mobile payment and tagging (NFC)	- Pagamenti tramite smartphone con NFC. - Pagamenti automatici preautorizzati, all'uso effettivo del servizio da parte dell'utente (Amazon dash)
	Cucina e ristorazione	- Applicazioni e dispositivi per la comunicazione tra sala e cucina. - Ristoranti e bar con pietanze preparate da robot. - Prodotti e merci primarie identificabili per mezzo di codici per risalire ad informazioni riguardanti produzione e filiera.
CITTA' E COMUNITA'	Sanitá e settore ospedaliero	- Tracciamento di stato e posizione dei macchinari all'interno delle strutture ospedaliere. - Dispositivi di collegamento diretto tra pazienti, infermieri e dottori, per notifiche e chiamate d'urgenza. - Dispositivi e sensoristica per la raccolta dati e il monitoraggio dei pazienti, con azioni programmate di risposta preventiva a stati ed eventi particolari. - Processi di autenticazione e autorizzazione all'accesso a dati sensibili dei pazienti.

Continua

Tabella 3.1 – *Continua dalla pagina precedente*

SETTORE	APPLICAZIONE	ESEMPI
	Inquinamento	- Telecamere di sorveglianza all'accesso di zone restrittive, con riconoscimento per tipologia di veicolo. - Sensori e applicazioni su automezzi per controllare e limitare la produzione di agenti inquinanti (perdite d'olio, fumi di scarico) che si adattano all'uso e alla guida dell'autista.
	Servizi per la communitá	- Sensori e dispositivi fisici per deviare e smaltire traffico in caso di ingorghi o incidenti in un percorso. - Servizi di sorveglianza intelligenti. - Sistemi di monitoraggio degli agenti atmosferici per prevenire e affrontare inondazioni, tempeste ecc. - Mappe e applicazioni con supporto della realtá aumentata per turismo e musei.
TRASPORTI E LOGISTICA	Spedizione merci	- Tracciamento dei pacchi per il monitoraggio della posizione e dello stato delle spedizioni mediante identificativi univoci e sensori GPS sui mezzi di trasporto. - Gestione dei magazzini organizzando giacenze e collocamento delle merci.
	Mezzi di trasporto	- Veicoli con guida assistita per mezzo di sensoristica anti-tamponamento, anti-sbandamento, sensori e telecamere di parcheggio. - Applicazioni per l'autenticazione personale per lo sblocco dell'auto o per la verifica del biglietto sui mezzi pubblici. - Monorotaie e metropolitane comandate da remoto con sistemi di posizionamento e di controllo in tempo reale.
	Infrastrutture di collegamento	- Percorsi stradali adibiti con sensoristica specifica o dedicati all'uso di mezzi speciali (es. veicoli elettrici con ricarica wireless). - Segnaletica che sfrutta la realtá aumentata per comunicare.

Per quanto riguarda la realizzazione dei sistemi IoT, l'approccio piú usato inizialmente era quello in forma verticale ovvero soluzioni device-to-cloud definite silos applicativi. Queste venivano pensate e realizzate per funzionare interagendo con l'ambiente ma di fatto senza l'intenzione di integrarsi con esso. I dati generati dal dispositivo venivano spediti allo specifico applicativo in cloud senza essere influenzati da altri possibili dispositivi. Gli utenti oggi richiedono sempre piú spesso la possibilitá di esportare o analizzare i dati e integrarli con altre sorgenti perció ci si sta orientando sempre di piú verso forme di sviluppo e di progettazione orizzontali, che permettono una visione complessiva dell'ambiente e dei processi in atto, utilizzando protocolli standard di comunicazione o fornendo API d'accesso ai dati. Il nuovo IoT sta cioé passando dalla presenza di apparati con uno scopo ben preciso e con una funzionalitá chiaramente identificata ad ambienti che permettano di gestire situazioni e orchestrare realtá diverse tramite multiple sorgenti di dati: sicurezza, energia, comfort etc.
Assume sempre piú importanza il componente che si occupa di questa integrazione di dati e servizi(*system integrator* o *IoT connector*) in modo che sia possibile sviluppare un livello di conoscenza molto piú approfondito degli ambienti reali. Vedremo di seguito alcune delle maggiori tecnologie e tecniche che permettono di fornire tale introspezione e riescono oltremodo ad apportare significativi vantaggi alla tecnologia che sta alla base di un dispositivo IoT.

3.1.1 Machine Learning

L'Intelligenza artificiale, lo ricordiamo, puó essere definita come la capacitá delle macchine di svolgere compiti e azioni tipiche dell'intelligenza umana (pianificazione, comprensione del linguaggio, riconoscimento di immagini e suoni, risoluzione di problemi, riconoscimento di pattern, ecc) basandosi sull'utilizzo di speciali algoritmi. Il Machine learning fa invece riferimento ai metodi per consentire ai software di AI di adattarsi, permettendo cosí alle macchine di apprendere come svolgere un compito o un'attivitá senza essere state preventivamente programmate per tale scopo.
Gestire dati e interazioni tra i dispositivi responsabili della loro generazione sta

diventando sempre piú complesso: basti pensare che, secondo IDC, la quantitá globale di dati generati dai dispositivi IoT (Internet of Things) raggiungerá lo stratosferico ammontare di 79,4 zettabyte (ZB) annui entro il 2025.
L'AI é in grado di far fronte a questo problema migliorando prestazioni, efficienza e aggiungendo scalabilitá e nuove funzionalitá alle soluzioni IoT. Consente di automatizzare compiti manuali guidati da pattern prestabiliti, riuscendo a integrarsi ad esempio in processi di gestione risorse e ottimizzazione delle prestazioni, dalla manutenzione predittiva, sino alla videosorveglianza, passando dalle applicazioni per la casa intelligente sino al marketing contestuale per grossi rivenditori.
L'intelligenza artificiale puó essere applicata per esempio per incrementare l'accuratezza e automatizzare la gestione delle query in base al probabile consumo di risorse, fornendo cosí un sistema piú stabile e affidabile e, soprattutto, in grado di assegnare le prioritá, riducendo la necessitá di intervento manuale nella gestione e nel monitoraggio del database. In questo modo, le imprese possono ridurre il tempo impiegato per generare informazioni e migliorare le decisioni aziendali.
Numerose sono le possibili implementazioni per analisi dei dati e data mining: clustering dei segnali e autoestrazione delle feature significative per il sistema [12][13], algoritmi di analisi non supervisionata su dati strutturati e non, uso di algoritmi di anomaly detection su dati in tempo reale oppure di previsione di risultati futuri tramite analisi a posteriori (effettuate sui dati tramite algoritmi di classificazione e regressione, mentre sul significato che hanno i dati tramite algoritmi detti *Concept drift*). Altre implementazioni [14, 15] permettono inoltre la sicurezza della rete, controllando finemente i dispositivi che accedono e che fanno uso del sistema imparandone a riconoscere i comportamenti e gli usi, classificando agenti malevoli in maniera semi-supervisionata o non supervisionata.

3.1.2 CIoT

E' un applicazione pratica dell'intelligenza artificiale ma dá forma ad una branca dell'IoT totalmente a se stante date le possibiliá che apre. Per estrarne il massimo potenziale e quindi operare in contesti sempre piú complessi, dobbiamo pensare al nostro sistema IoT come un entitá cosciente in grado di fare ragio-

namenti logici. La Cognitive Internet of Things (CIoT) é un paradigma di rete piuttosto nuovo nel quale oggetti fisici e virtuali, sono tutti interconnessi tra loro e si comportano come entitá sociali all'interno del sistema. L'intervento umano é quasi inesistente in questo contesto e i dispositivi comunicano e agiscono seguendo cicli di percezione-azione in modo consono all'ambiente da cui ricevono feedback. Usando metodologie di comprensione approfondita dell'ambiente e di miglioramento basato sulle informazioni ricevute dai vicini é possibile implementare meccanismi di adattamento automatici ed efficienti nell'utilizzo delle risorse [18]. Due sono gli obbiettivi principali:

- Unire il mondo fisico (composto da oggetti, risorse, etc) e quello sociale (composto da necessitá, abitudini, ...), per formare sistemi piú articolati e complessi.

- Permettere allocazione di risorse in modo automatico e intelligente con metodi di previsione.

CIoT é visto come l'attuale IoT con l'integrazione di meccanismi di automazione intelligenti, per promuoverne le potenzialitá e fornire servizi piú articolati. Il tentativo di introdurre un sistema di ragionamento cognitivo all'interno del design del sistema é molto primordiale al momento, poiché molti dispositivi necessitano ancora di una forte componente umana per poter funzionare, ma con il costante sviluppo di connessioni tra oggetti e dispositivi, aumento di tempo e sforzi impiegati ed efficienza delle tecnologie, queste applicazioni emergeranno nell'uso quotidiano. I vantaggi che apporta sono molteplici come il permettere risparmio di tempo e fatica in routine quotidiane, minimizzare gli sprechi di risorse e migliorare i servizi proposti.

3.1.3 Digital Twin

Con Digital Twin itendiamo la copia digitale di un elemento fisico, la rappresentazione virtuale e dinamica di un componente materiale di un sistema. Per essere definito tale peró questa entitá deve racchiude 5 caratteristiche fondamentali

della sua controparte: l'identitá, la quale puó anche essere non univoca (vengono inclusi sia rapporti con cardinalitá 1 a 1 che 1 a N); la rappresentazione, che definisce le caratteristiche fisiche e strutturali della controparte fisica definendola tramite metadati; lo stato, permettendo considerazioni e misurazioni in tempo reale sulle caratteristiche del componente; il comportamento, il quale riflette i risultati che possono comportare azioni causate da stimoli basilari come forze applicate, processi chimici o cambiamenti di temperatura all'interno del contesto; il contesto, ovvero l'ambiente nel quale é immerso, opera ed interagisce. Questa tecnica implementativa, che rende possibile la raccolta e l'aggregazione di dati ad alta fedeltá in un sistema, trova il suo maggiore impiego in applicazioni rivolte alla gestione di componenti e apparecchiature di grande valore. Gli scopi sono molteplici, tra cui:

- Controllo degli stress strutturali
- Pianificazioni di ispezioni o manutenzioni in base a dati storici reali
- Mantenimento di uno storico delle correlazioni tra carico, potenza e risultati per gestire le politiche del sistema
- Prevenzione e previsione dei danni
- Visioni complessive approfondite, con grafici e dati temporali per feedback
- Analisi dell'andamento passato, presente e futuro
- Coordinamento tra piú entitá
- Controlli per ridurre i costi e minimizzare gli sprechi
- Visualizzazione e ispezione di dettagli non visibili o in posizioni non accessibili

Il mantenimento di questa entitá virtuale é vista come una sfida in quanto a problematiche di sicurezza, qualitá e disponibilitá dei dati oltre che di gestione di streaming dati realtime. La sua implementazione implica spesso la necessitá di adottare infrastrutture con una componente di *edge computing* preponderante per

poter mantenere i dati elaborati il piú possibile vicino alla fonte e minimizzare la latenza. Comparato al *machine learning*, notiamo come entrambe queste tecniche possano sfruttare i dati per ricavarne ulteriore valore seppure in modi distinti. Questa diversitá peró non ne preclude l'uso reciproco ed infatti spesso vengono implementate entrambe in maniera congiunta [9]. Per esempio nella gestione di un parco di pale eoliche possono essere implementati sia modelli di DT per mantenere precise rappresentazioni virtuali delle attrezzature in funzione, sia algoritmi predittivi di analisi per scoprire complesse relazioni e pattern nelle forze in gioco a cui sono sottoposti i componenti. Gli algoritmi di *machine learning* vengono scritti sulla base di regole fissate al contesto, poi i modelli vengono allenati su set di dati preraccolti, puliti e controllati, rappresentati gli stati del DT. A seconda dell'obbiettivo dell'algoritmo potremmo ottenere previsioni o informazioni dettagliate sui componenti del nostro sistema. Particolare importanza, specialmente in un contesto simile con flussi di dati realtime, va attribuita alla parte analitica e al salvataggio dei dati che dovrá essere gestito in maniera accorta per evitare trashing del sistema.
Categorizziamo i nostri cloni virtuali in:

Part twin: come lascia intendere il nome é solo parte di un sistema piú grande. Per esempio, consideriamo l'infrastruttura di un impianto energetico industriale. Questa impianto avrá un DT che ne rappresenta lo stato operativo e la sua condizione. I dati possono derivare da sensori di vibrazione e rilevatori strutturali ed essere messi in confronto con i dati standard per analisi predittive.

Product twin: é semplicemente un set di *Part twin* che interagiscono e riflettono lo stato reciproco. Riferendoci sempre all'esempio precedente possiamo intenderlo come un intero generatore con tutti i sensori e rilevatori ad esso annessi.

System twin: anche questo é da intendere come un'entitá a livello superiore della precedente categoria. Come aggregazione di *product twin* non é peró fisicamente un insieme di dispositivi implementati sullo stesso prodotto ma piuttosto una

visione d'insieme dell'intero sistema. Per l'appunto nel nostro esempio indicando questo tipo di DT si intenderebbe la rappresentazione dell'intero impianto con relative proiezioni future basate su dati del presente e del passato.

3.1.4 Clustering

Ogni applicazione IoT si basa sull'acquisizione di dati dall'ambiente circostante attraverso sensori e dispositivi. Successivamente, questi dati dovranno essere analizzati (esternamente o internamente all'applicazione) per ricavarne il valore intrinseco e farne uso. Quando si ragiona su grandi strutture con multiple sorgenti di dati ed eterogeneitá di modelli nasce spontaneamente la necessitá di semplificare la visione del sistema e gestire questi flussi in maniera ordinata. Il raggruppamento di dispositivi e sensori é chiamato *clustering*. Questa metodologia permette di gestire la complessitá del sistema e ridurre i processi e le fonti di acquisizione di informazioni al minimo numero di connessioni necessarie. Il clustering consente inoltre il prolungamento della vita della rete e dei dispositivi stessi i quali verranno sottoposti a minori sforzi computazionali e d'interazione con l'applicazione centrale; fornisce un servizio di scheduling alle comunicazioni tra dispositivi e nodi centrali e consente un notevole risparmio energetico del sistema in generale. Per un raggruppamento efficiente in un ambiente IoT é necessario avere:

- Una classificazione nella struttura sottostante composta da dispositivi e sensori che puó essere realizzata per esempio tramite indirizzo IP o con una nomenclatura ben definita.

- Un tempo di accesso e di comunicazione molto piccolo per garantire la qualitá del dato.

Multipli nodi agiscono da stazioni centrali di riferimento per una rete sottostante, permettendo una comunicazione verso l'applicativo con trasmissioni ad "1 Hop". I dispositivi che vogliono infatti comunicare un'informazione si connetteranno al *Cluster Head* (CH) che si incaricherá di spedire il pacchetto al nodo applicativo.

Tutti i CH si troverranno necessariamente ad 1 nodo di distanza dall'unitá centrale per mantenere basso l'overhead di comunicazione.
Questa soluzione é particolarmente apprezzata negli ambienti dinamici [10] dove i dispositivi si connettono per brevi sessioni o cambiano spesso la loro posizione all'interno della struttura di rete. Notoriamente é piuttosto facile gestire una rete di dispositivi relizzata a cluster (la struttura é quella gerarchica di una semplice rete LAN), ma la questione si complica un po' quando la formazione di queste avviene dinamicamente: nella realtá IoT, quando un dispositivo si connette, ha infatti la possibilitá di diventare egli stesso un CH o assumere il ruolo di nodo sottoposto. Il clustering dinamico implica frequenti cambi di questo genere con la selezione di CH differenti a intervalli di tempo stabiliti.
Ci sono molti algoritmi e meccanismi che usano questa struttura a cluster [11], per citarne alcuni: Distributed Clustering Algorithm (DCA) , Spanning Tree (o BFS Tree), on-Demand Clustering Routing e Energy Efficient Clustering EELBC che riserva una particolare attenzione al consumo energetico. Un differente approccio di clustering prevede l'applicazione di algoritmi di *data reduction* per compattare i dati cambiandone la dimensionalitá oppure algoritmi di compressione per ridurre la quantitá di byte trasmessi senza (o con minima) perdita d'informazioni
citeWSN, 6V.

3.1.5 Altre tecnologie

Nel campo delle tecnologie troviamo il 5G, e le esperienze di realtá simulata o aumentata (VR, AR e MR). L'avvento dell'era delle cumunicazioni su 5G rivoluzionerá il mondo IoT. Secondo Aleksander Poniewierski, EY Global IoT Leader,

> "IoT cannot thrive without effective and affordable wireless connectivity, interoperability and common standards. We believe 5G has the potential to make a ground-breaking impact on the way in which future IoT ecosystems are designed, especially in the areas of scalability, latency, reliability, security and the level of individual control on connectivity parameters."

Al di lá delle questioni prestazionali, la connessione degli oggetti alla rete 5G consentirebbe di tracciare costantemente e identificare in tempo reale tutti gli oggetti collegati alla rete, risolvendo molte delle problematiche legate alla comunicazione tra smart object, sensori e attuatori. Per consentire il collegamento degli oggetti alla rete 5G ci sono una serie di progetti in corso d'opera, come quello del produttore britannico di chip, Arm, che ha recentemente realizzato una piattaforma software che consente di equipaggiare con SIM virtuali i dispositivi IoT del futuro. L'esperienza all'interno di un mondo virtuale, apre invece nuove frontiere all'utilizzo delle tecnologie IoT che potranno essere integrate in questi ambienti in modo naturale per migliorare la sensazione di immersivitá e colmare fisicamente il divario tra il mondo fisico e quello virtuale. Le tecnologie VR, AR e MR hanno una proiezione di crescita annuale del 71.6%, secondo un analisi dell'EY [25], compagnia di distribuzione di servizi su scala mondiale, e raggiungeranno un valore di mercato globale del valore di 147 miliardi di dollari tra hardware software e servizi entro il 2022.

3.2 Problematiche attuali

Al momento esistono numerosi scogli da superare prima che l'IoT possa raggiungere il massimo del suo potenziale:

- **Larga scala:** Il numero di dispositivi IoT in commercio aumenta ogni anno e ció porta alla necessitá di particolari infrastrutture di rete per metterli in comunicazione. Inoltre la gestione dei dati generati e la loro interpretazione puó diventare critica e va gestita in maniera appropriata.

- **Intelligenza:** L'unione di software e hardware, rende le infrastrutture IoT intelligenti. Quest'intelligenza peró deve essere indirizzata in maniera adeguata per poter essere sfruttata dalle applicazioni finali che faranno uso dei dati. La popolaritá delle tecnologie smart risiede proprio nell'interazione e nella condivisione delle informazioni raccolte singolarmente, ma senza adeguati metodi d'accesso e presentazione dei dati queste risultano di scarsa utilitá o di difficile utilizzo agli occhi dell'utente.

- **Sensori:** L'IoT non puó esistere senza sensori. I sensori sono usati per rilevare cambiamenti nell'ambiente e rifletterne gli eventi virtualmente. Essi forniscono puramente dei dati che vanno elaborati e analizzati in modo accurato per poter ricavare informazioni sull'ambiente e conoscenza degli avvenimenti fisici in atto.

- **Sistemi complessi:** L'IoT puó prevedere miliardi di componenti in un singolo sistema. Effettuare operazioni che involgono l'uso di molteplici e differenti elementi spesso limitati in termini di memoria, energia e tempo rende il processo di coordinamento molto complesso.

- **Ambiente dinamico:** Le tecnologie IoT risiedono per la maggior parte dei casi in ambienti e reti dinamiche dove i dispositivi possono collegarsi o scollegarsi continuamente, senza influire sul funzionamento del sistema in sé. I dispositivi devono inoltre adattarsi dinamicamente a possibili cambiamenti, per esempio in un sistema di sorveglianza avanzato le telecamere devono cambiare modalitá (diurna/notturna) in seguito all'attivazione di un crepuscolare e devono regolare la risoluzione in maniera dinamica al rilevamento di movimenti per evitare di salvare continuamente registrazioni di grandi dimensioni, potenzialmente di scarsa utilitá.

- **Grandi quantitá di dati:** I dispositivi generano enormi quantitá di dati i quali necessitano di essere salvati e mantenuti sicuri.

- **Eterogeneitá:** I sistemi IoT comprendono una grande varietá di dispositivi. Piattaforme, sistemi operativi e servizi necessitano di essere connessi a questi, spesso per mezzo di diversi protocolli e piattaforme di traduzione delle informazioni.

- **Risorse limitate:** i dispositivi IoT sono spesso piccoli sia in dimensioni che in potenza ed hanno quindi a disposizione immediata una quantitá molto limitata di risorse. Accorgimenti posti al risparmio (per esempio per non consumare la batteria) sono sempre da prendere in considerazione durante la definizione dell'infrastruttura.

- **Connettivitá:** Permette l'acceso alla rete abilitando a comportamenti intelligenti, oggetti di tutti i giorni. Non é sempre implementabile in tutti i contesti per motivi tecnologici e ambientali ed una volta ottenuta é difficile da mantenere.

- **Auto-configurazione**: il mercato di distribuzione di prodotti IoT necessita di dispositivi pronti all'utilizzo (PnP) per venire incontro all'utente finale. I singoli dispositivi devono avere la capacitá di auto-configurarsi in base all'ambiente in cui vengono inseriti per fornire un servizio immediato e di qualitá. Accedendo alla rete, inoltre, devono potersi aggiornare, e poter comunicare con gli altri dispositivi con un minimo se non nullo, intervento umano.

- **Identitá univoca:** Ogni dispositivo IoT deve avere un'identitá univoca rappresentata per mezzo di un identificatore che puó essere un codice, un indirizzo IP o una posizione geografica ad esso associabile. Questa identitá ne agevola la gestione nel sistema per interrogazioni, monitoraggio e controllo specifico. Per esempio, potrebbe essere necessario effettuare aggiornamenti o reset di singoli dispositivi da remoto.

- **Conoscenza del contesto:** I sensori spesso si limitano a raccogliere dati dall'ambiente senza peró avere informazioni ulteriori riguardanti il contesto in cui fisicamente sono inseriti.

3.2.1 Quantitá e qualitá: le 6v

La complessitá di un sistema IoT emerge in due forme: complessitá dell'infrastruttura, e legata ai dati. Quest'ultima in particolare é dovuta alla spesso grande quantitá di dati che sono coinvolti: i big data. Ne racchiudiamo caratteristiche e problematiche sotto la nomenclatura delle 6V:

- **Volume:** La quantitá di dati viene detta volume ed é un fattore spesso critico in un sistema IoT. Non esiste una soglia dalla quale i dati scambiati iniziano

ad essere denominati big ma in senso comune si intende una quantitá abbastanza grande da non poter essere processata a basso livello dell'infrastruttura con la conseguente necessitá di spostarli in server centrali o nel cloud. Per questo motivo la soglia puó variare da qualche centinaia di Gigabyte fino a decine di Terabyte.

- **Velocitá:** La velocitá dei dati scambiati in un sistema é determinata dalla frequenza delle comunicazioni con i componenti del sistema e dai vari stream di dati. E' necessario prestare massima attenzione in caso di elaborazioni in realtime per assicurarsi che il verificarsi di rallentamenti non causi trashing nel sistema.

- **Varietá:** Sistemi IoT raccolgono flussi di dati provenienti da piú sorgenti che producono dati in vari formati. Questa etereogeneitá va tenuta in considerazione sia durante la raccolta che durante l'analisi perché impatta sulle performance del sistema, sulla sua capacitá di trarre significato dai dati e identificare pattern nascosti.

- **Veridicitá:** L'utilitá dei dati aumenta soprattutto quando siamo in grado di garantirne la provenienza e l'affidabilitá e fare ció puó diventare complesso senza compromettere le performance del sistema. Questa proprietá si riferisce anche all'affidabilitá del sistema stesso.

- **Variabilitá:** Poiché gli stream di dati in arrivo non hanno tutti le stesse caratteristiche, é necessario adottare una serie di misure atte alla gestione peculiare di casistiche che possono portare al verificarsi di problemi. Ad esempio, flussi di dati asincroni o strutture di dati a campi variabili.

- **Valore:** Il valore dei dati ne definisce l'utilitá e la fruibilitá nel contesto applicativo finale. Questa proprietá é riferita piú che altro ai risultati dell'analisi e non alla gestione dei dati all'interno del sistema ed é direttamente proporzionale al valore delle altre 5v.

La complessitá dei dati viene ulteriormente influenzata dalla totale assenza di strutture di datiall'interno dello stream di informazioni, oppure dalla presenza di

vincoli e legami interdimensionali ed intradimensionali e dalla varietá di scopi per il quale dovranno essere impiegati tali dati. Per esempio, attributi quali eta, genere e carattere possono essere attributi caratterizzanti, per l'analisi dimensionale di una persona ma anche interdimensionale quando tali attributi sono riferiti a membri di un nucleo familiare e ne si sta valutando il consumo energetico casalingo.

3.2.2 Sensori e attuatori

Le limitazioni che vanno prese in considerazione quando si ha a che fare con dispositivi IoT sono spesso la scarsa disponibilitá di memoria interna e quindi la necessitá forzata di dover inviare immediatamente i dati raccolti senza possibilitá di mantenimento in cache o raccolta di dati offline; la connettivitá, che comprende tutti quegli aspetti che influenzano la scelta dei protocollo, come le radiofrequenze supportate dell'hardware, il range di trasmissione, la necessitá di affidabilitá e feedback sui pacchetti inviati, etc; per finire i limiti tecnologici di miniaturizzazione dell'hardware per poterlo installare fisicamente nei luoghi d'interesse senza alterarne lo stato o influenzare le rilevazioni.

Il problema energetico

Esistono tantissimi tipi di sensori e attuatori ma tutti hanno in comune la necessitá di dover essere alimentati per poter funzionare. Alcuni necessitano di fonti energetiche trascurabili come i tag RFID che vengono eccitati dai lettori al rilascio di un segnale, altri necessitano di un accesso alla rete elettrica costante per poter funzionare.
Ovviamente con lo scalare della potenza dei dispositivi aumentano anche i requisiti di energia elettrica ed é spesso vincolante dover sottostare a queste limitazioni. Poiché molte soluzioni di sistemi IoT vengono pensate e realizzate per mezzo di componenti alimentati a batteria piuttosto che direttamente alla corrente, va posta un'attenzione speciale a tutte quelle misure che possono favorire in qualche modo l'ottimizzazione energetica all'interno del sistema e questi accorgimenti vanno

presi in primo luogo durante la realizzazione dell'infrastruttura [19].
Le operazioni piú dispendiose in termini di risorse dovranno essere gestite da elaboratori piú potenti quindi si tenderá a centralizzare la computazione e l'analisi cercando peró di mantenere bassa la latenza in accordo con gli obbiettivi e le necessitá del sistema; vanno definite architetture e suite di protocolli per efficienza e modalitá d'uso: spesso la scelta viene fatta ad hoc per la soluzione; vanno ottimizzati gli algoritmi in modo che siano computazionalmente efficienti; le trasmissioni di dati devono essere ridotte allo stretto necessario; infine va valutata l'introduzione di piú o meno complessi algoritmi di compressione: riducono la quantitá di byte scambiati ma comportano overhead temporali nelle comunicazioni.
Il consumo energetico dovuto ai singoli dispositivi é solo parte del problema, infatti anche la stessa infrastruttura wireless di comunicazione ha un costo di mantenimento, per non parlare delle centinaia di datacenter che sono costantemente connessi e svolgono la funzione di servizio cloud per la nostra infrastruttura. Con una visione globale del panorama tecnologico riscontriamo nell'immediato la necessitá di un potenziamento nelle infrastrutture di comunicazione senza fili per poter gestire il traffico generato da un numero sempre crescente di dispositivi: solamente negli ultimi anni il consumo di energia dovuto alle infrastrutture di rete é passato dal 20% al 30% secondo una statistica dei consumi globali[7]. Se l'attivitá dovuta all'introduzione di 3-4 miliardi di smartphone puó causare un simile impatto si prospetta che quando al mondo ci saranno piú di 50 miliardi di dispositivi connessi, il consumo energetico globale causato dalla rete ammonterá a piú del 50%.

3.2.3 Salvataggio dei dati

Rendere i dati accessibili nel tempo presuppone il loro salvataggio. Ma quali dati vanno salvati? Dove? In che modo?
A seconda dell'applicazione e dell'utilitá dei dati potremo rispondere in modo diverso a queste domande: solo i dati grezzi per poter mantenere uno storico completo delle rilevazioni; salvare dopo una prima fase di preprocessazione eliminando feature inutilizzate e dati non validi; salvare i risultati provenienti dall'attivitá di

analisi ed elaborazione perdendo peró tutte quelle informazioni che non hanno portato a risultati significativi o non sono state prese in considerazione.
Solitamente, la quantitá di dati da gestire é molto grande ed i dati non vengono salvati ad ogni stato del sistema, perció vanno applicati accorgimenti per la gestione dei dati ed eventualmente é possibile applicare complessi algoritmi di compressione o aggregazione modellati ad hoc sulla la struttura dei dati raccolti per ridurre il piú possibile lo spazio necessario a salvare le informazioni. Tuttavia soluzioni simili sono ancora molto acerbe dal lato dello sviluppo, soprattutto in un contesto di streaming dati in tempo reale come puó essere quello di un sistema IoT.
Molte limitazioni sono imposte anche dai sistemi stessi di salvataggio dati. Troviamo in contrapposizione i database relazionali e quelli NoSQL. I database relazionali sono notoriamente meno performanti nel gestire grandi quantitá di dati e nello scalare orizzantalmente [**?**], per questo vengono preferiti modelli NoSQL per soluzioni IoT. Nonostante ció, tutte le opzioni vanno vagliate e ponderate poiché la modalitá di salvataggio dei dati influisce molto sulle performance finali del sistema. Citiamo tra i piú diffusi i database chiave-valore, a grafo, orientati all'uso di righe e colonne per l'indicizzazione oppure per collezioni di documenti.

3.2.4 Sicurezza

Purtroppo, come per la complessitá del software, anche per le soluzioni IoT la scalabilitá rappresenta un problema: all'aumentare di dispositivi e all'espandersi dell'infrastruttura, la gestione della sicurezza diventa sempre piú complessa sia per la protezione dei dispositivi che per quella dei dati personali.
Ad oggi, i casi di cronaca riguardanti falle di sicurezza o problemi riguardanti i dispositvi IoT sono piuttosto frequenti.

> "Hotel in Giappone con Robot-maggiordomo hackerati: una falla di sicurezza nel TAG NFC permetteva di accedere allo streaming video del robot, in ogni momento" - "Il termostato NEST permette un facile accesso a casa vostra a malintenzionati tramite un sistema di update del firmware non sicuro" - "Telecamere di sicurezza Ring accedute dalla rete tramite credenziali rubate

dai server, ed usate per spaventare gli abitanti con l'altoparlante integrato"

Questa situazione é dovuta in parte alla mancanza di supporto e aggiornamento per contrastare l'invecchiamento del firmware o delle architetture, nonché a una generale mancanza di controllo delle proprie infrastrutture. Tra le organizzazioni odierne, molti dipartimenti IT non sono nemmeno a conoscenza della totalitá di dispositivi IoT connessi nelle loro reti, rendendo quasi impossibile il compito di correggerne i problemi di sicurezza. Secondo un recente rapporto del Ponemon Institute, nel 2017 solo il 15% dei partecipanti al sondaggio aveva subito una violazione dei dati relativi all'IoT. Questo numero é salito fino al 26% nel rapporto del 2019, nel quale sono stati intervistati 625 esperti di gestione aziendale. Inoltre, alla domanda se fosse probabile che le loro organizzazioni avessero potuto subire un attacco informatico di tipo *Denial of Service* (DoS) causato da dispositivi o applicazioni IoT non garantiti, nei 24 mesi successivi, l'87% degli intervistati ha risposto affermativamente, secondo il rapporto.
Piú specificamente i maggiori problemi di sicurezza sono dati dall'autenticazione e dall'integritá dei dati. Data la volontá di ridurre i tempi di comunicazione e la tendenza verso soluzioni parsimoniose in risorse e messaggi scambiati, le comuni modalitá di instaurare canali sicuri non sono ben viste e sono quindi necessarie soluzioni alternative. Inoltre i comuni sistemi di autenticazione comportano alti costi di gestione necessitando spesso di infrastrutture apposite e server dedicati per lo scambio di token o chiavi di cifratura per poter certificare la propria identitá.
L'integritá dei dati, ossia la proprietá che garantisce che i dati transitati da mittente a destinatario non siano stati alterati senza che il sistema ne sia consapevole, va estesa anche alle situazioni di stallo comunicativo, quando i dati sono fisicamente salvati sui nodi della rete. Questi sono spesso lasciati incustoditi e devono perció essere resi inaccessibili da intromissioni malevole anche a livello fisico. Mentre a livello digitale siamo purtroppo limitati dalla potenza dei dispositivi che in alcuni casi, ad esempio i tag RFID, supportano l'uso di password per la cifratura troppo corte per essere considerate sicure; e anche se queste fossero sufficientemente complesse, rimane sempre la questione dell'utilizzo in sé della cifratura, che comporta consumi

e rallentamenti significativi alle comunicazioni in dispositivi a risorse limitate quali sensori e attuatori. E' quindi necessario lo sviluppo di metodi alternativi per la gestione di comunicazioni sicure nei sistemi IoT come per esempio soluzioni di *edge data collection* che agiscono come endpoint sicuri.
La risposta a questa domanda di sicurezza arriva primariamente dai dati stessi, ovvero dalla conoscenza approfondita del sistema e dell'ambiente che lo circonda. Attraverso una visione globale ed avendo ben chiaro lo scopo e le esigenze che stanno alla base dei sistemi é poi possibile disegnare soluzioni appropriate che meglio permettano di raggiungere quegli obiettivi minimizzando l'esposizione a tutti i fattori di rischio conosciuti.
Ricordiamo peró che le violazioni di privacy legate ai dispositivi IoT possono essere causate sia da attaccanti malevoli che fanno uso di breccie nella sicurezza dei dispositivi che dalle compagnie produttrici stesse le quali distruggono la fiducia che i consumatori ripongono all'acquisto di un dispositivo.

> "Google Nest Casa sicura contiene un microfono nascosto al suo interno non presente tra le specifiche pubbliche: la casa produttrice si difende dicendo che é stato un errore di pubblicizzazione e che esso sarebbe servito per applicazioni future non ancora distribuite"

Le aziende produttrici di dispositivi sono in grado di profilare i singoli individui raccogliendo informazioni dall'ambiente e dai dispositivi con il quale interagiscono, e purtroppo ad oggi é impossibile avere la certezza che le informazioni ricavate siano usate in modo ponderato e innocuo oppure vengano usate per scopi meno legittimi o ancora peggio rivendute a terzi. Conoscendo preferenze e abitudini di una persona diventa facile fare campagne di merketing o di propaganda, mirate sul singolo individuo.
Solo la possibilitá che delle aziende possano avere questo potere d'influenza su una persona, porta gli utenti piú diffidenti ad allontanarsi dalle soluzioni proposte dalle "big tech industries" per mantenere il proprio anonimato e privacy, e a volte ad ostacolare attivamente la diffusione di tecnologie che invece potrebbero migliorare e semplificare la vita. Fortunatamente gli enti governativi dedicano sempre piú interesse a questi argomenti per regolamentare e punire i trasgressori a tutela dei

consumatori.
Contromisure appropriate per la protezione della privacy garantiscono che:

- I sistemi di monitoraggio non raccolgano informazioni sulla posizione e sui movimenti del singolo individuo ma solo considerando aggregati di persone (posizioni e movimenti salvati non devono poter essere associabili ad una singola identitá)
- Le persone devono essere informate dello scopo e della posizione in cui i dati sono mantenuti
- I dati devono essere mantenuti solo al fine d'utilizzo per cui sono stati raccolti ed essere poi eliminati

Inoltre vengono applicate soluzioni di anonimizzazione, identificazione temporanea, e cifratura nel salvataggio centralizzato dei dati.

Capitolo 4

Progetto

4.1 Intento

Dopo aver analizzato approfonditamente la composizione e il funzionamento logico di un'infrastruttura IoT, averne elencato i vantaggi e descritto le problematiche piú comunemente riscontrate, possiamo affermare che la fase di design e la scelta dei componenti sono forse le piú critiche poiché scelte sbagliate qui possono influire negativamente tutti i passaggi successivi, minando la stabilitá e funzionalitá del sistema. In questo percorso di realizzazione di un sistema IoT, questa tesi si propone come una strada giá spianata per chi si trovi davanti ad un bivio e debba valutare piú opzioni in fase di progettazione. Nel modello realizzato, le soluzioni adottate attribuiscono una struttura modulare e customizzabile per semplificare i processi e contenere le difficoltá implementative che possono presentarsi andando a comporre sistemi ad hoc, e che al contempo riesca ad integrarsi con strutture e componenti giá presenti nell'ambiente. Ad un primo acchito si potrebbe dire che i problemi maggiori dell'IoT nascano dalla connessione tra il mondo fisico delle "cose" e quello virtuale, ma come abbiamo visto precedentemente esistono varie soluzioni a tale scopo, molte delle quali sviluppate propriamente per l'ambito IoT; anche i servizi di messaging, ossia quelli che permettono la comunicazione degli applicativi con i dispositivi, inizialmente limitati e instabili nel gestire grosse moli di dati, hanno raggiunto un notevole livello di sviluppo allo stato attuale. Di contro,

peró, riuscire a connettere il tutto fornendo servizi articolati ma stabili e sicuri non é cosí scontato, e oltremodo non lo é pretendere un sistema interoperabile, adattabile, scalabile, accessibile in modo sicuro e soprattutto che possa gestire in maniera uniforme qualsiasi tipo di dispositivo e farlo dialogare con il lato virtuale del sistema.
Questo é ancora un campo giovane e fertile e non esistendo soluzioni ”standard”, spesso chi si trova nella posizione di dover realizzare l'infrastruttura, si affiderá a soluzioni mirate alle proprie problematiche attuali implementando ove necessario moduli aggiuntivi sviluppati internamente per far interagire il tutto.
A questo proposito viene presentato Hono ovvero la soluzione realizzata dall'Eclipse Foundation in ambito IoT. In contrapposizione a piú comuni soluzioni pensate ad hoc per il singolo sistema, le quali portano alla formazione di silos applicativi e che non permettono una scalabilitá orizzontale al sistema, l'Eclipse IoT Working Group ha proposto un approccio piú generico per la realizzazione di piattaforme IoT *cloud-based* che permettesse l'implementazione di soluzioni senza richiedere agli sviluppatori di risolvere piú e piú volte i soliti problemi ricorrenti. Il progetto aveva lo scopo iniziale di elaborare un modo per astrarre i protocolli di comunicazione e fornire trasparenza d'accesso ai dispositivi, agli altri componenti di back-end. Tutto ció gestendo elevati flussi di informazioni, che possono andare da qualche centinaio di migliaia fino a milioni di messaggi al secondo.

Questo componente svolgerebbe la funzione di un *IoT Connector*, un collegamento tra l'interfaccia dei protocolli dei dispositivi, ovvero componenti di back-end, e applicazioni o client di amministrazione per la gestione di firmware e dispositivi, a stretto contatto con l'utilizzatore finale. Gli sviluppatori possono utilizzare il connettore IoT per interagire in modo uniforme e trasparente con tutti i tipi di dispositivi senza la necessitá di preoccuparsi dei particolari implementativi per i protocolli di comunicazione che utilizzano i dispositivi. Piú soluzioni possono utilizzare la stessa istanza del connettore IoT in esecuzione in un ambiente cloud, per condividere i dati e le funzionalitá di tutti i dispositivi collegati. Inoltre, viene garantita la separazione dei contesti e controllo sulle

autorizzazioni d'accesso a dati specifici, sia dall'interno che dall'esterno: i componenti che accedono all'infrastruttura devono essere autenticati e autorizzati. Il connettore IoT deve soddisfare una serie di requisiti non funzionali specifici per l'ambiente cloud di distribuzione e per la piattaforma architetturale prevista, ad esempio deve garantire la scalabilitá orizzontale all'infrastruttura. Questa funzione di connettore per altro lavora ad alto livello senza entrare nel dettaglio dei dati scambiati, quindi non conoscendo il dominio applicativo. Infatti da un punto di vista cosí generico, tecnicamente non fa differenza se una lettura del sensore ricevuta tramite un adattatore di protocollo MQTT rappresenta una temperatura o un valore d'umiditá. In entrambi i casi la responsabilitá del connettore IoT é di inoltrare i messaggi contenenti i valori a uno o piú consumatori autorizzati senza introdurre troppa latenza.

4.2 Hono

4.2.1 Descrizione

Hono come progetto dell'Eclipse IoT Working Group é uscito definitivamente dalla fase embrionale per approdare sul mercato in forma stabile, e certificato all'uso operativo solo nel Luglio 2019. Tutt'ora in sviluppo é attualmente alla versione 1.1.0 rilasciata a Gennaio 2020 con un piano di rilascio che prevede un nuova versione ogni 3 mesi.
Hono é essenzialmente un livello d'astrazione all'interno dell'infrastruttura IoT. Come previsto dagli obbiettivi del progetto di sviluppo aiuta e semplifica il processo di connessione tra differenti tipi di dispositivi e applicazioni/servizi in esecuzione. Definisce API per far interagire lo stack applicativo nel cloud, fornendo modalitá d'autenticazione, di gestione delle credenziali e di comunicazione dall'esterno. Alcuni servizi collegati a queste API sono giá inclusi nel pacchetto di Hono come implementazione standard, come gli adapter dei protocolli HTTP e MQTT, mentre per altri é definita solo l'API e non il servizio in se che normalmente é giá previsto dal sistema con il quale va ad integrarsi e che quindi il piú delle volte dovrá solo

essere collegato al servizio interno di messaging di Hono. Nasconde agli sviluppatori gli aspetti riguardanti i problemi di connessione e trasmissione, fornendo un'infrastruttura di comunicazione uniforme per tutte le soluzioni IoT tramite l'uso di AMQP 1.0. Questo protocollo, in particolare, é di comune utilizzo a lato server, ed é in grado di gestire i requisiti di velocitá e volume dei comuni sistemi IoT. Molti dispositivi utilizzano per comunicare MQTT o semplici chiamate REST tramite il protocollo HTTP, per semplicitá. LoRaWAN, CoAP, Sigfox, etc. tutti hanno i loro pro e contro e pretendere che gli utilizzatori si adeguino alla scelta dei progettisti e sviluppatori di Hono é impensabile. Viene quindi fornita la possibilitá di implementare adapter personalizzati per i propri bisogni, ed essendo un progetto opensource é possibile pubblicare il proprio modulo per renderlo disponibile a tutta la comunitá, ed installabile come estensione del programma.
Fornisce un articolato servizio di messaging interno in grado di gestire contesti multi tenant in modo che una singola istanza di Hono possa essere usata da piú stream di dati (ossia da piú applicazioni) contemporaneamente. Ció viene sicuramente comodo a quelle piccole realtá che prevedono pochi dispositivi ma che hanno la necessitá di dialogare tra loro o scambiare informazioni per ottenere una conoscenza approfondita del sistema. Inoltre sono supportati sia messaggi di tipo evento che di telemetria per la comunicazione tra dispositivi e applicazioni, e una speciale API é dedicata alla gestione dei dispositivi.
L'architettura interna (Figura 4.1) é stata realizzata seguendo un modello di programmazione reactive e implementata tramite un insieme di microservizi dentro a container docker. Essendo pensata per essere scalabile e distribuita tutti i componenti possono essere pubblicati in cloud e comunicano tra loro mediante il protocollo AMQP. La gestione dei microservizi puó quindi avvenire a livello individuale ed essi potranno scalare linearmente e indipendentemente dagli altri. Le funzionalitá principali che danno forma al core di Hono sono:

- Autenticazione dei dispositivi: Hono supporta una grande varietá di protocolli d'autenticazione, tra cui username/password e mediante certificato fornito dal client. Inoltre il protocollo CoAP supporta l'autenticazione basata sullo scambio di chiavi precondivise durante l'istanza della comunicazione DTLS.

- Gestione dei dispositivi: modulo per la regsitrazione e gestione della configurazione dei dispositvi del sistema. Permette il salvataggio delle configurazioni e il caricamento da file a seconda delle necessitá.
- Monitoraggio e tracciamento: supporta operazioni di gestione amministrativa fornendo totale accesso ai singoli componenti e metriche di tracciamento per i messaggi scambiati internamente. Le misurazioni acquisite possono poi essere salvate in database temporali e visualizzate in dashboard grafiche. Hono fornisce strumenti di monitoraggio anche tramite OpenTracing, e supporta una varietá di programmi di tracciamento tipo Jeager.
- Comunicazione: il componente di messaging é importante per il modo in cui Hono comunica con esso ma di fatto non é parte integrante di Hono.

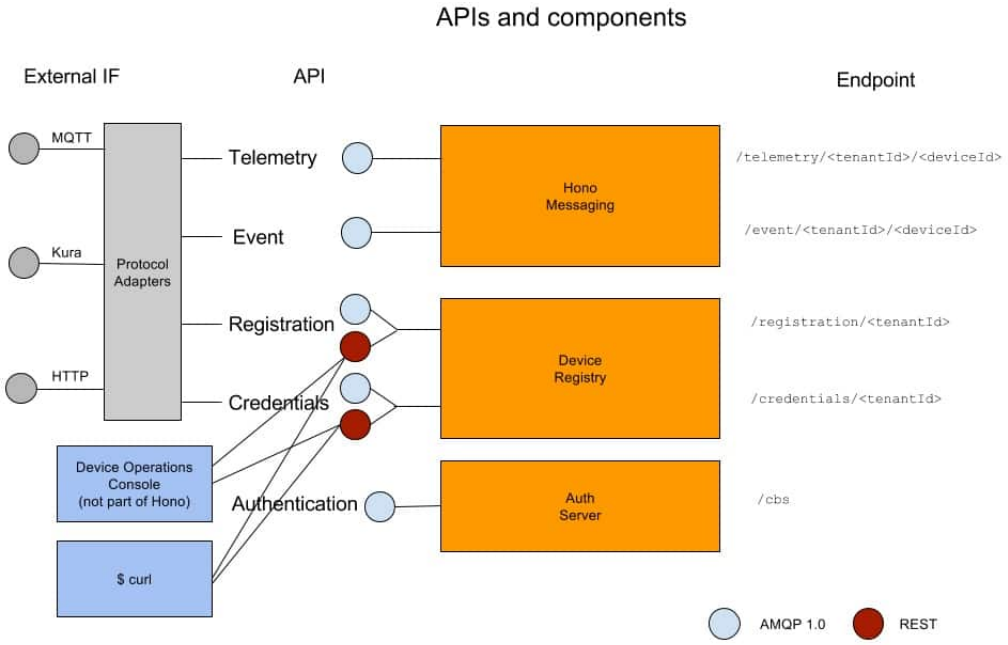

Figura 4.1: Eclipse Hono: componenti ed endpoint

4.2.2 Dispositivi

Ogni device é riconosciuto univocamente da Hono per mezzo della tupla *tenant-id*/*device-id*. Questa viene composta e associata all'identitá del dispositivo in seguito ad una registrazione su uno specifico tenant. Per registrare un dispositivo il client dovrá fornire il *device-id*, l'*auth-id* e un segreto ossia una modalitá di autenticazione prevista tra i tipi di credenziali supportate. L'*auth-id* viene solitamente inteso come un username e viene utilizzato per le comunicazioni interne del sistema solo durante il processo di autenticazione. Terminata questa operazione i riferimenti successivi al dispositivo saranno fatti per mezzo del *device-id* il quale viene fornito sottoforma di stringa da parte del client, e scelto in maniera totalmente arbitraria senza alcun vincolo relativo al tipo di dispositivo o al tipo di adapter usato per la comunicazione.
I dispositivi che aprono una connessione con un adapter sono tenuti a specificare l'indirizzo del gateway per mezzo del quale comunicano, e che fungerá da porta d'accesso nel caso il dispositivo dovesse essere contattato. Questa informazione non viene salvata permanentemente nel sistema ma ne viene tenuta traccia solamente in memoria per non rallentare inutilmente il sistema in caso di frequenti aggiornamenti del campo. La flessibilitá che offre quest'implementazione permette l'uso di Hono anche in contesti dinamici dove i dispositivi autenticati possono cambiare posizione mantenendo aggiornato il loro punto d'accesso per messaggi inviati da applicazioni o da client.
A scopo di gestione dei dispositivi vengono utilizzate le API di "Device Registration" e "Device Connection".

Il mantenimento della connessione avviene in automatico per connessioni tramite protollo AMQP o MQTT che supportano il paradigma publish/subscribe mentre per connessioni HTTP viene utilizzato il campo TTD che indica per quanto tempo il dispositivo sará disponibile e quindi quando si disconnetterá. Questo valore verrá aggiornato periodicamente tramite pacchetti di notifica vuoti per non far scadere la connessione.

4.2.3 Multitenancy

Hono supporta il partizionamento logico in gruppi chiamati *tenant*. Solitamente i tenant vengono usati per raggruppare e separare porzioni di sistemi e sono identificabili univocamente attraverso una stringa chiamata *tenant-id*. Sono utili sia a lato implementativo per l'applicazione che dovrá comunicare con i dispositivi, sia per il mantenimento di un ordine logico nello sviluppo dell'infrastruttura, ad esempio possono essere usati per identificare i dispositivi allocati fisicamente ad ogni piano di un palazzo oppure per raggruppare sensori acustici, d'inquinamento e di movimento in un paese.
Ogni endpoint delle API AMQP, tranne quello dedicato ai tenant, é accessibile solo per mezzo dello specifico *tenant-id*, ad esempio:
- telemetry/TENANT1
- registration/TENANT1
Questo conferisce una separazione logica a livello di tenant estesa a tutto il sistema:

- La registrazione di un dispositivo avviene strettamente su un tenant
- Le credenziali associate ad un dispositivo sono strettamente legate al tenant su cui é stata effettuata la registrazione
- I protocol adapters possono essere abilitati o disabilitati per il singolo tenant
- I dati inviati e ricevuti dai dispositivi e dalle applicazioni sono isolati per tenant (Event API, Telemetry API, Command & Control API) cosí come anche la gestione del traffico e la limitazione delle risorse legate alla comunicazione

Questo isolamento é essenziale per permettere ad un'architettura scalabile e distribuita di gestire insimemi di risorse indipendenti. La flessibilitá di Hono permette comunque di creare dispositivi e credenziali su tenant non (ancora) esistenti, che dovranno peró essere creati in un secondo momento per fare uso delle relative API.

4.2.4 Opensource

L'opensource software é un modello di sviluppo e distribuzione del software, in termini di codice sorgente, basato sulla presenza di una forte comunitá: persone

che contribuiscono allo sviluppo del codice, alla correzione di errori e difetti, alla definizione di test e documentazione. Il codice é di proprietá condivisa e viene normato in modo che chiunque possa farne uso. Le ragioni dell'opensource sono:

- **Costi:** un software aperto non ha costi di licenza
- **Affidabilitá e Qualitá:** lo sviluppo é incentrato sull'esperienza e sui feedback degli utilizzatori per conquistarne la fedeltá. Alti tassi di adozione da parte della comunitá garantiscono sostenibilitá al progetto
- **Trasparenza e Flessibilitá:** l'accesso al codice sorgente del programma é libero e alla portata di tutti. Ció consente un veloce adattamento o personalizzazione del programma ai reali bisogni del cliente e crea indipendenza dallo specifico fornitore di software

Questi attributi sono oggi alla base della scelta di soluzioni software da parte di molte amministrazioni e aziende private. Alcune di queste con viste piú lungimiranti, impegnano parte della propria forza lavoro a progetti opensource per la diffusione di prodotti e sistemi, sostenere una mentalitá di condivisione e formulazione di codice e software open by design.

Il progetto Hono a nome dell'Eclipse Foundation (vedi appendice per approfondimento) é stato sviluppato all'interno dell'Eclipse IoT Working Group: sezione operativa della fondazione con all'attivo piú di 35 progetti mirati alla diffusione e regolamentazione dell'uso di tecnologie IoT. Hono in particolare é il risultato del lavoro condiviso di 27 persone inclusi molti sviluppatori appartenenti ad aziende private come Microsoft, Bosch e Red Hat e l'utilizzo di questa piattaforma viene ampiamente supportato dall'IBM.

4.2.5 Sicurezza

Nonostante tutti i piú popolari protocolli si basino su TCP/IP, il quale supporta la negoziazione di connessioni sicure, quando si tratta l'argomento IoT, si sente spesso parlare di dati trasmessi in chiaro su reti pubbliche e dati personali trafugati

dai sistemi. Sin dalle prime versioni del progetto Hono, il team di sviluppo ha implementato meccanismi di sicurezza e cifratura dei dati. L'infrastruttura supporta trasmissioni tramite protocollo TLS e tutti i componenti sono configurati di default per l'utilizzo di connessioni sicure. E' comunque possibile impostare da configurazione, l'utilizzo di connessioni non sicure per situazioni di testing oppure una doppia modalitá di connessione che prevede l'apertura di porte sia sicure che non. L'handshake previsto durante l'instaurazione della connessione con protocollo TLS supporta metodi di cifratura tramite l'utilizzo di credenziali quali username e password e in aggiunta, dalle ultime versioni (0.9-M2) é stata introdotta anche la possibilitá di autenticare i dispositivi tramite l'utilizzo di certificati X.509 come parte della comunicazione TLS sia per l'HTTP adapter che per l'MQTT.
Questa scelta implementativa é stata pensata proprio per ridurre i tempi di verifica all'interno del sistema mantenendo peró alti gli standard di sicurezza. Usualmente il procedimento di autorizzazione di un dispositivo che vuole autenticarsi con username e password avviene nei seguenti passaggi:
- Il dispositivo fornisce le credenziali all'adapter.
- L'adapter richiede le credenziali cifrate al server di autenticazione per lo specifico *device-id* associato al tenant.
- Ottenute le credenziali dal server, l'adapter confronta i dati forniti dal dispositivo e se specificato, applicherá le dovute misure di cifratura (ad esempio al momento della registrazione delle credenziali potrebbe essere specificato di cifrare la password con un particolare algoritmo oppure salvarla in chiaro)

Questi passaggi devono peró essere svolti per ogni sensore/attuatore. Tramite l'uso dei certificati é invece possibile associare un ente certificatore unico per tenant e una volta richiesta la chiave pubblica dell'ente questa viene mantenuta in memoria dell'adapter riducendo sensibilmente il carico di comunicazioni necessarie per connettere molti dispositivi appartenenti allo stesso tenant.
Una volta autenticato il dispositivo e instaurata la connessione, Hono si assicura che i dati prodotti da tale componente appartenente ad un particolare tenant possano essere consumati solo da client autorizzati per tale tenant per mantenere stagni i flussi di dati tra i diversi gruppi d'appertenenza.

Poiché l'infrastruttura é stata pensata per essere installata principalmente su cluster o elaboratori che comunicano a distanza, e quindi principalmente tramite l'uso di reti pubbliche, anche tutte le connessioni tra componenti interni sono rese sicure tramite l'uso di TLS: *Protocol Adapter* $\longrightarrow$ $Device\ Registry$ *e* $Device\ Registry \longrightarrow$ $Auth\ Server$ ad esempio.

4.2.6 Limitazioni, Monitoring e Tracing

Per la gestione operativa, il progetto Hono, prevede modalitá personalizzate per il controllo del funzionamento del sistema e l'uso di componenti aggiuntivi esterni per limitare l'uso delle risorse. Questi ultimi vengono eseguiti parallelamente al sistema ed effettuano controlli a runtime per assicurare che tutti i container abbiano sempre una quantitá sufficiente (e non eccessiva) di risorse per poter funzionare correttamente. In particolare viene gestita finemente l'allocazione di memoria e di core destinati alla JVM in esecuzione nei container dei componenti Hono.

** Poiché tutti i componenti Hono sono scritti in Java, per poter funzionare necessitano dell'esecuzione interna al container stesso della JVM. In maniera automatica, questa viene avviata con un allocazione pari a circa 1/4 delle risorse dell'ambiente (il container), ma di fatto seguendo una buona implementazione dell'infrastruttura cloud un container conterrá un solo processo ossia quello della JVM, quindi si agisce con parametri a runtime per aumentare la disponibilitá di risorse dedicate che in caso contrario rimarrebbero inutilizzate (il limite di memoria viene impostato a 70-80% nella configurazione di default). **

Hono supporta inoltre limitazioni a livello di tenant sul numero di dispositivi connessi e per la quantitá di dati che i dispositivi possono pubblicare in un determinato intervallo di tempo.

Sistemi di tracing interni e di reporting delle performance del sistema sono molto utili principalmente in ambienti di produzione. Hono supporta l'implementazione di entrambe queste estensioni come componenti aggiuntivi.
Le metriche sono raccolte come eventi temporali e forniscono dettagli sul funzionamento effettivo del sistema tenendo traccia dei Key Performance Indicators (ad

esempio delay, tempi di connessione e di risposta dei singoli componenti) i quali possono indicare lo stato di salute del sistema, anticipare una carenza di risorse o un mal funzionamento. Questi dati sono raccolti in due modalitá differenti a seconda del componente scelto per lo scopo:
- tramite **Micrometer**: i componenti pubblicano i dati delle metriche sull'endpoint dedicato
- tramite **Prometheus**: i componenti aprono un'endpoint dedicato e il server Prometheus viene configurato per fare polling su ognuno di essi

In seguito dati e metriche possono essere salvati in database (ad esempio InfluxDB) e mostrati in dashboard grafiche riassuntive (ad esempio Grafana).
Quanto riguarda il tracing invece, Hono supporta componenti del progetto OpenTracig. Per abilitare il reindirizzamento dei log, ogni componente deve peró essere configurato al momento del deploy indicando l'istanza del componente di tracing.

4.2.7 Deployment

L'installazione dell'infrastruttura di Hono puó essere fatta su una piattaforma cloud a scelta come AWS, MS Azure o GCP. I componenti sono forniti come immagini di container docker, che possono essere eseguiti su un qualsiasi orchestratore. Openshift e Kubernetes in particolare sono quelli consigliati.
Questa modalitá di pubblicazione permette un tipo di sviluppo agile all'interno dell'infrastruttura con un'architettura a microservizi. Ogni team puó in questo modo focalizzarsi su uno specifico servizio per poi integrarlo nel sistema a sviluppo terminato.
Il deploy default per Eclipse Hono é composto da 6 microservizi ovvero i componenti principali, e dalle mappe di configurazione default per Prometheus e InfluxDB. Tale gestione permette una facile integrazione ed uso con altri progetti opensource come EnMasse, Grafana e Keycloak.
Attualmente per il deploy dell'immagine vengono messi a disposizione template Helm a differenza delle vecchie versioni le quali mettevano a disposizione dell'utilizzatore template specifici di Openshift secondo il modello Source-2-Image. Questo cambio di rotta é stato adottato per una maggiore versatilitá dato il supporto

nativo di tale formato da parte di differenti orchestratori.

Poiché flussi di dati provenienti da diversi tenant vengono nativamente gestiti in modo separato per ragioni di sicurezza e segretezza, é consigliabile all'atto pratico pubblicare una singola installazione di Hono, e sfruttare questa caratteristica unita alle potenzialitá di scalabilitá per gestire soluzioni IoT comuni, piuttosto che fare deploy multipli in cloud per i singoli sistemi applicativi. Ció permetterá un uso piú efficiente della rete e risparmi in computazione data la condivisione delle risorse tra i tenant.

4.2.8 Come si integra nel mondo IoT

L'obbiettivo dell'IoT Working Group, ossia il gruppo di sviluppo che si occupa del progetto Hono, é quello di fornire soluzioni flessibili e modulari nell'ambiente IoT. E' importante che le nuove tecnologie d'infrastruttura e software si possano integrare con sistemi giá esistenti per aggiungerne funzionalitá senza necessariamente dover rivoluzionare tutta la struttura. A maggior ragione, Hono, come portatore di questo principio permette connessioni ed integrazioni con altri servizi e progetti, sia opensource che non. Per citarne alcuni, il router Apache Camel, servizi di messaging tipo Enmasse che sfruttano AMQP 1.0 e qualsiasi servizio di visualizzazione o salvataggio basato su eventi temporali. Inoltre é consigliato da parte del team di sviluppo stesso di Hono l'uso congiunto di altri servizi, come Ditto per il mantenimento del *Digital Twin* dei dispositivi, Kura per funzioni di gateway (per il quale viene fornito addirittura un adapter dedicato), HawkBit per la gestione degli aggiornamenti e Kafka per la gestione di cluster.
In generale, Hono riesce a fornire sia un ambiente integrato per nuovi sviluppatori, utile per realizzare infrastrutture casalinghe per esempio con Raspberry Pi, Node.js e Mosquitto sia per connettere milioni di dispositivi all'interno di un infrastruttura molto piú complessa ed estesa. Tutto ció corredato da presupposti di software opensource e di continuo sviluppo per adattarsi alle necessitá della comunitá.

Scalabilitá ed EnMasse

Per soddisfare la proprietá funzionale della scalabilitá orizzontale, Hono, é stato progettato con un'architettura a microservizi i quali vivendo dentro a singoli container possono essere ridimensionati facilmente ed in modo indipendente. Gli adattatori, i servizi di gestione e di autenticazione dei dispositivi, e il componente di messaggistica sono disaccoppiati tra di loro e possono quindi scalare svincolati da limitazioni interne.
Il deploy standard di Hono prevede un servizio di messaggistica posticcio che sará necessario sostituire per soluzioni che prevedono un intenso uso di questo componente. Quando si tratta di supportare un gran numero di dispositivi collegati, ed un elevato throughput in termini di messaggi scambiati, la soluzione "single router, single broker" proposta mostra i suoi limiti.
L'infrastruttura di messaggistica deve essere elastica in modo da consentire a Hono di gestire dispositivi e consumatori e per supportare picchi nel numero di dispositivi connessi e traffico generato. Sarebbe uno spreco allocare staticamente un alto numero di istanze broker AMQP 1.0 per un piccolo numero di dispositivi e per un traffico ridotto. Allo stesso modo, il ragionamento inverso sarebbe limitante per il servizio offerto. Poiché questi cambiamenti possono avvenire di frequente, é necessario adottare un servizio adatto al ruolo da ricoprire: EnMasse é il suggerimento della community e del team di sviluppo.
Come piattaforma di messaggistica opensource oltre a fornire la scalabilitá e l'elasticitá necessarie per sostenere e non limitare il servizio di Hono, supporta tutti i modelli di messaggistica comuni (request/response, publish/subscribe) e comunica tramite due protocolli: sia AMQP 1.0 che MQTT. EnMasse offre multi-tenancy, il che significa che diversi tenant possono condividere la stessa infrastruttura, pur rimanendo isolati l'uno dall'altro. Infine, fornisce sicurezza per quanto riguarda la protezione delle connessioni, tramite il protocollo TLS, e per l'autenticazione dei client che utilizzano Keycloak come sistema di gestione delle identitá. Inoltre si integra perfettamente nella visione di deployment di Hono essendo completamente containerizzato e funzionando sulle piú diffuse piattaforme di orchestrazione come Kubernetes, Docker Swarm e OpenShift (Figura 4.2).

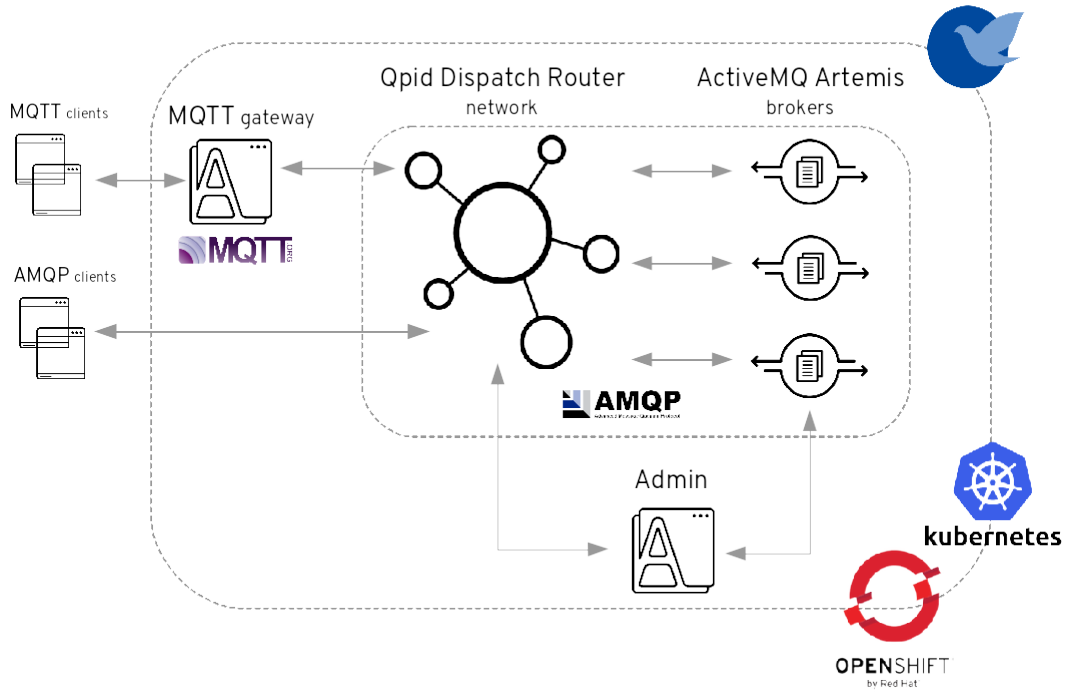

Figura 4.2: Struttura di Enmasse

Per rispondere in modo piú analitico alla domanda iniziale di scalabilitá, l'integrazione tra Hono ed EnMasse é stata testata a scopi dimostrativi con ottimi risultati: all'aumentare delle risorse e scalando gli adapter in modo lineare é possibile mantenere una media di 5000 messaggi al secondo per singola istanza di adapter fino all'esaurimento delle risorse hardware sottostanti.

4.3 Infrastruttura

- I dispositivi si collegano al relativo adapter secondo il protocollo di comunicazione supportato. Questi usano il Device registry (che a sua volta fa uso dellâAuth server) per autenticare i dispositivi e garantirne lo stato di registrazione. Successivamente inviano i dati ricevuti dai dispositivi al servizio di messaging che fa da mediatore per lâapplicazione consumer. Questa, collegata al Dispatcher router tramite protocollo AMQP 1.0 é in grado di ricevere messaggi e di inviare comandi tramite lâapposita API -

Tutte le interazioni tra i componenti sono basate sul protocllo AMQP 1.0 e vengono raggruppate con le seguenti denominazioni (Figura 4.3):

- Credentials API
- Tenant API
- Device Registration API
- Device Connection API
- Command & Control API
- Telemetry API
- Event API

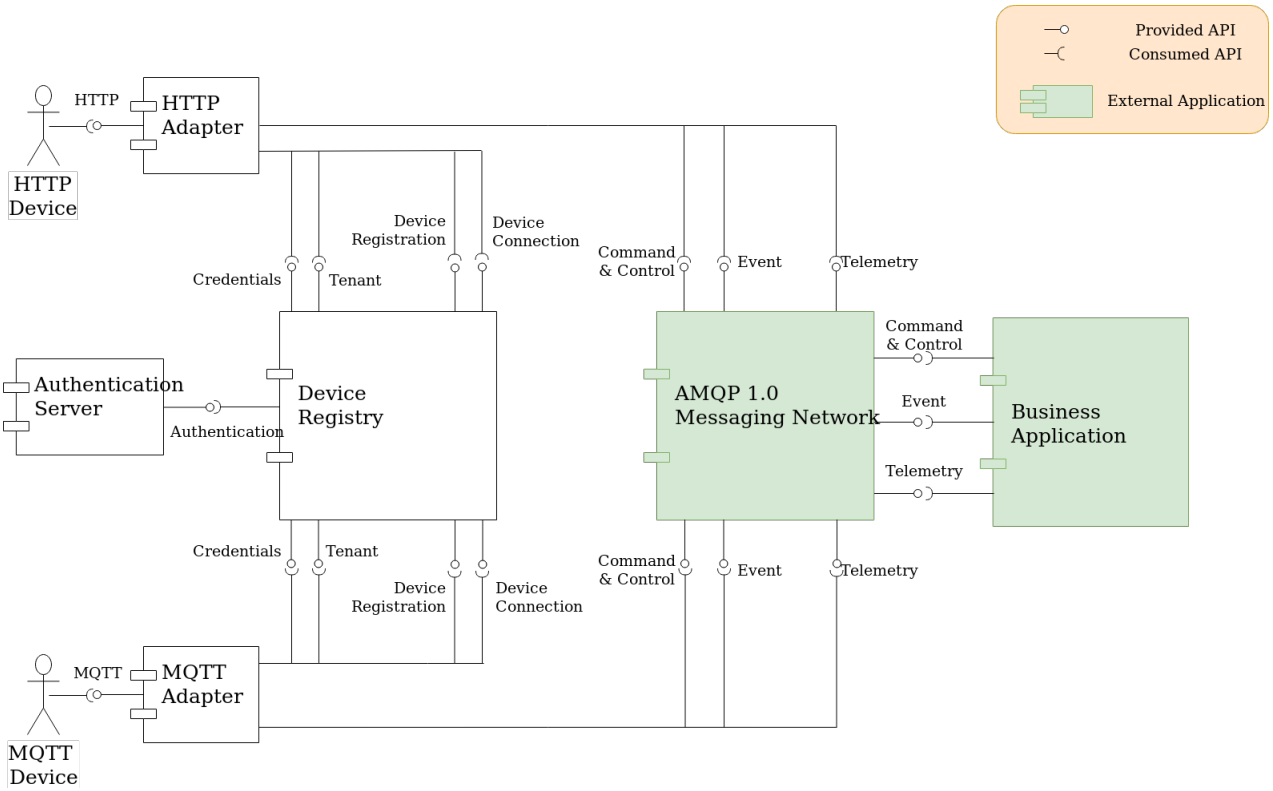

Figura 4.3: Visione generale sull'infrastruttura di Hono

4.3.1 Device Registry

Implementa ”Credentials API”, ”Tenant API”, ”Device Registration API” e ”Device Connection API”. I client aprono le connessioni e vengono autenticati tramite l'utilizzo di una porta sicura. I dati relativi alle varie API vengono salvati localmente in file cifrati, rispettivamente uno per il servizio delle credenziali, uno per salvare i tenant disponibili e uno per le registrazioni attive nel sistema. Un servizio interno all'infrastruttura, si occuperá quindi di associare i device alle relative credenziali. Seppur fornisca le funzioni basilari di autenticazione viene sconsigliato dalla documentazione stessa l'uso di questo modulo operativo in sistemi complessi o che prevedono numerose connessioni contemporanee poiché non é stato designato per essere scalabile e funzionare in modo condiviso tra piú istanze di Hono.

4.3.2 Athentication server

Tutti i componenti prima di poter dialogare tra loro devono essere autenticati da questo servizio, il quale implementa l'endpoint della ”Credentials API”. Gli agenti esterni come dispositivi e consumer non dialogano direttamente con il server d'autenticazione ma vengono autenticati indirettamente da componenti intermediari nella comunicazione: i primi tramite gli *adapter* mentre i secondi da un istanza del *dispatch router*.
L'autenticazione dei device avviene nel modo seguente:

1. Il dispositivo fornisce le credenziali o il certificato al momento della connessione con l'adapter.

2. L'adapter, il quale a sua volta si sará autenticato con il servizio, passa le informazioni e aspetta la comunicazione di risposta, e in caso positivo presenterá un token d'accesso in formato JSON.

3. Quindi per le successive operazioni il dipositivo fornirá l'identificativo d'accesso associato al suo token il quale conterrá le indicazioni relative ai suoi permessi.

Per l'autenticazione dei componenti del sistema viene utilizzato un meccanismo di autenticazione SASL PLAIN(RFC 4422). I componenti interni client aprono una connessione AMQP verso i componenti server ed inviano le informazioni necessarie all'autenticazione. Il server usando le credenziali fornite dal client nello scambio SASL PLAIN si collega all'Auth server e ritrasmette le informazioni ricevute. In caso di autenticazione riuscita, l'Auth Server emette un JSON Web Token (JWT) che asserisce l'identitá del componente client e le sue autorizzazioni. Il server, quindi, collegherá questo token alla sua connessione AMQP con il client e ne fará uso per prendere decisioni di autorizzazione relative alle richieste in arrivo su tale connessione.

Il servizio di messaging che si occupa di gestire le applicazioni e il relativo consumo o spedizione di messaggi, é anche incaricato dell'autenticazione di queste. Poiché questo servizio non é interno ad Hono il meccanismo viene implementato ma il suo utilizzo é a discrezione del componente stesso che eventualmente potrá occuparsi internamente di certificare le applicazioni o usare l'API da facade per l'autenticazione tramite l'Auth server interno. In questo caso il procedimento sará similare a quello descritto per i componenti interni del sistema. Nella configurazione standard di Hono l'Apache Qpid Dispatch Router autentica i consumer usando un meccanismo SASL arbitrario. Le credenziali dei consumer sono salvate in un file e caricate all'inizializzazione dei servizi.

Le identitá e le autoritá di certificazione utilizzate dall'Auth Server per la verifica delle credenziali e l'emissione di token, sono definite in un file di configurazione letto all'avvio sul server. Queste vengono utilizzate sia per i componenti di sistema che per le applicazioni esterne. Tutti i dati vengono quindi conservati in memoria fisica del componente o centralizzati in un unico spazio sicuro accedibile solamente in modo diretto, senza alcuna API dedicata.

4.3.3 Adapters

Hono di base fornisce 4 diversi adpater per il collegamento di dispositivi: HTTP, AMQP, MQTT, CoAP e un adapter specifico per retrocompatibilitá con vecchie versioni di Kura. Eventuali altre implementazioni per protocolli specifici o modifiche a quelli esistenti sono possibili mediante l'apposita configurazione al momento del deploy del servizio di messaging.
Gli adapter principalmente si occupano di esporre gli endpoint di collegamento per l'API di telemetria e degli eventi: ricevono i dati e li inviano al servizio AMQP di messaging previa autenticazione e instaurazione di una connessione sicura con questo. Svolgono peró altre funzioni in maniera trasparente all'utente e ai dispositivi:

- Si occupano della registrazione e autenticazione dei dispositivi facendo uso rispettivamente della Device Registration API e Credentials API a cui devono essere collegati.
- Quando un dispositivo tenta di instaurare una nuova connessione, fanno uso della Tenant API per richiedere informazioni al tenant specificato dal dispositivo per controllarne l'appartenenza.
- All'arrivo di una richiesta o di un messaggio da inviare ad un determinato dispositivo si collegano ad un'istanza dell'API Device Connection per determinarne il gateway di comunicazione ed inviare tali informazioni all'indirizzo corretto.
- Possono utilizzare metriche raccolte da un servizio esterno ad Hono (ad esempio Prometheus) allo scopo di applicare limitazioni d'utilizzo configurate a livello di tenant. Ad esempio il numero di dispositivi che si possono connettere contemporaneamente.

4.3.4 Messaging Network

Hono necessita di un servizio AMQP di messaging come backend all'infrastruttura, di un broker e di un componente che svolga la funzione di router e si occupi

di reindirizzare i messaggi al corretto destinatario.
Nel deploy standard fornito, l'AMQP 1.0 *Messaging Network* é presente ma non é un componente proprio del progetto, poiché Hono non intende proporsi come servizio di messaging, ma come servizio intermediario tra questo e gateway/dispositivi, fornendone quindi solo gli endpoint di comunicazione interna. Questo servizio viene implementato tramite gli artefatti di altri progetti opensource della famiglia Apache, in modo da fornire un ambiente il piú possibile completo giá al primo deploy dell'infrastruttura e praticamente utilizzabile "out of the box" con i dovuti accorgimenti di configurazione.
Alla versione attuale (1.1.0) il servizio integrato é composto da una singola istanza di Apache Qpid Dispatch Router connesso ad una singola istanza dell'Apache Artemis broker (Figura 4.4). Nonostante ci si aspetti un servizio in grado scalare assieme a tutta l'infrastruttura di Hono, vien da sé che tale configurazione non rispetta questi requisiti funzionali e questo poiché di fatto l'implementazione reale da utilizzare in un ambiente "di produzione" spetterá agli addetti dell'infrastruttura tramite componenti esterni probabilmente giá esistenti. Il team di Hono fa presente che questa modifica all'infrastruttura é necessaria solo per ottenere un sistema piú efficiente e soprattutto allineato ai presupposti di scalabilitá presentati inizialmente, e che comunque puó essere facilmente applicata implementando una versione integrale del progetto Enmasse da cui provengono i due componenti di messaging citati.

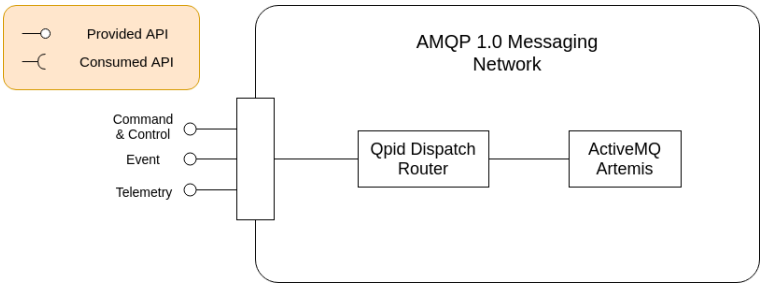

Figura 4.4: Servizio di messaging di Hono

Il *Dispatcher Router* espone gli endpoint delle API di telemetria, eventi e di

controllo dei dispositivi, alle applicazioni e client autenticati con Hono.
- **Telemetria:** Hono ottimizza il throughput dei dati di telemetria. Le applicazioni e i servizi utilizzano la Telemetry API per ricevere i dati pubblicati dai dispositivi. Questi dati possono essere consegnati con due modalitá di servizio: al massimo una volta (default) o almeno una volta.
- **Eventi:** Hono supporta la possibilitá di invio di messaggi da parte dei dispositivi verso le applicazioni. La stessa modalitá viene utilizzata per comunicare all'applicazione informazioni relative al dispositivo, per esempio la sua disconnessione o il raggiungimento di una soglia di invio messaggi. In caso di scollegamento dell'applicazione o di temporanea non diponibilitá di connessione i messaggi persisteranno in una coda per essere poi consegnati alla restaurzione del canale comunicativo. Gli eventi vengono spediti sempre in modalitá "cosegnato almeno una volta".
- **Command & control:** Hono permette alle applicazioni di inviare comandi a specifici dispositivi per scaturire azioni, cambiare lo stato di attuatori, modificarne la configurazione o anche aggiornarne il firmware. Sono supportati due pattern diversi: il segnale viene inviato una volta senza che l'applicazione si aspetti una risposta dal dispositivo, oppure un comando di tipo "Request/Response" al quale l'applicazione si aspetta una risposta. Questa potrá essere ricevuta sia in modo asincrono che bloccante per la continuazione dell'esecuzione stessa, a seconda dell'implementazione a codice.

4.4 Progetto

Al fine di analizzare accuratamente difficoltá e limitazioni dovuti alla definizione e realizzazione di un sistema IoT, viene proposta a scopi puramente didattici, come progetto supplementare, un'infrastruttura full stack che includa tutti i passaggi del ciclo di vita dei dati: dalla raccolta alla presentazione. La strutturazione di questa ha agevolato la comprensione delle tecnologie coinvolte e ha permesso di fornire dettagli e implementazioni pratiche d'esempio per affrontare le difficoltá che abbiamo presentato nei capitoli precedenti. La dimostrazione pratica fa uso

dell'infrastruttura di Hono come componente principale il quale risolve giá out of the box problematiche di rilievo quali l'autorizzazione e l'autenticazione tra *producers* e *consumers*. Affiancati al core strutturale sono stati implementati un gateway di connessione per sensori fisici, componenti per la simulazione di flussi di dati sia in produzione che in consumo, ed una dashboard di visualizzazione delle rilevazioni trasmesse.

Figura 4.5: Struttura dell'infrastruttura di sistema

4.4.1 Struttura

La soluzione proposta é stata strutturata in modo da essere in grado di adattarsi alle situazioni piú classiche che si presentano in questo settore secondo i modelli analizzati. Per chiarezza espositiva viene esplicitato un esempio di caso d'uso in modo da poter avere una visione piú figurata delle possibilitá d'utilizzo e delle reali complicazioni sul campo.

- In una strada ad alta velocitá futuristica é necessario gestire mezzi di diverso genere che si muovono in modo semi autonomo, grazie all'ausilio di sensori e attuatori integrati nel veicolo i quali comunicano ed interagiscono con gli automezzi circostanti e con la strada stessa. I sensori saranno incaricati di raccogliere e comunicare valori di distanza di prossimitá e di posizione geografica ad un edge gateway installato sul veicolo, il quale sará in grado di effettuare elaborazioni ad alto livello dei dati e comunicare nell'immediato con i mezzi vicini. Questi edge gateway saranno inoltre connessi all'infrastruttura cloud, e si occuperanno di inviare prontamente i dati per permettere valutazioni e ispezioni piú approfondite sui valori rilevati e soprattutto abilitare una gestione globale delle strade tramite controlli remoti attivi e/o passivi in maniera automatizzata. Infine i dati elaborati

saranno fruibili pubblicamente per l'ottenimento di statistiche e dettagli sempre aggiornati sullo stato dei percorsi stradali -

Come detto, l'esempio funge da campione di raffronto in quanto situazione realistica e non é da intendersi perció come "problema alla nostra situazione". La struttura generica e modulare permette di affrontare le complicazioni peculiari del caso integrando e sostituendo blocchi di software per comporre un sistema in grado di prioritarizzare certi aspetti piuttosto che altri a seconda dell'implementazione ma con il minimo sforzo di sviluppo.
Si pensi ad esempio al componente di messagistica che nel nostro caso d'uso dovrá essere molto performante per gestire elevate velocitá di risposta e una grande quantitá di peer connessi per volta: sará necessario sostituire quello fornito dal deploy standard di Hono, ma dato l'accoppiamento lasco tra i moduli, modificando un componente non viene intaccato il funzionamento degli altri.
Tutto il sistema cloud formato dai componenti di Hono, il servizio di messaggistica, i componenti di simulazione virtuali, consumer e server web per la gestione della dashboard sono eseguiti e mantenuti in ambiente containerizzato grazie all'utilizzo di immagini docker per il rilascio delle versioni.
I container sono unitá standard di software che impacchettano il codice e tutte le sue dipendenze in modo che l'applicazione venga eseguita in modo rapido e affidabile da un ambiente di elaborazione a un altro. Per di piú é possibile supervisionare e gestire in modo immediato e automatizzato lo stato dei container e dei processi in esecuzione al loro interno tramite l'ausilio di sistemi di orchestrazione.

Un aspetto di particolare importanza che viene sottolineato nell'esempio ma che come detto precedentemente é anche parte dei problemi di attualitá del mondo IoT, é quello della compatibilitá: ovvero abilitare il sistema a dialogare con tutti i tipi di dispositivi. Per quanto possa essere futuristico il nostro esempio sarebbe insensato e oltremodo improbabile, dato l'atteggiamento assunto fino ad oggi nel mondo IoT, presupporre che tutte le case produttrici di veicoli adottino un modello unificato di comunicazione con agenti esterni. Questo presupposto potrebbe comportare lo sviluppo di moduli dedicati per la comunicazione ma non deve peró influenzare

l'intero stack. L'infrastruttura propone e al contempo dimostra, tramite l'utilizzo dei diversi adapter di Hono, come si possa affrontare semplicemente questo problema. Inoltre con Kura viene affrontata la necessitá di introspezione sui dati in entrata e in uscita, e di gestione di sensori e di attuatori appartenenti ad uno stesso ecosistema, quale puó essere il singolo veicolo d'esempio. Si tratta di un componente sempre appartenente alla famiglia di software dedicati all'IoT dell'Eclipse Foundation, molto conosciuto e utilizzato nell'ambiente data la possibilitá di interfacciarsi a basso livello con l'hardware pur permettendo un'ottima ed intuitiva gestione lato software. Semplifica la gestione dei dispositivi (es. network, firewall), permette lo sviluppo di applicazioni integrate di interazione con sensori e attuatori e fornisce la possibilitá di gestire le connessioni verso il cloud. E' stato rilasciato in versione stabile nel 2014 e tutt'ora é supportato dalla community di sviluppatori opensource.
Lo sviluppo di Hono ha tenuto in considerazione anche la compatibilitá con software "datato" come Kura, in questo caso, fornendo un adapter dedicato per la connessione. Nelle versioni successive di questo software la comunicazione é stata standardizzata all'uso del protocollo MQTT.

4.4.2 Sensori e simulatori

Per la pubblicazione dei dati é stato scelto di implementare diverse applicazioni per testare vari aspetti dell'infrastruttura.
Un sensore fisico é stato collegato al cloud per mezzo di un gateway sviluppato come componente aggiuntivo di Kura. Questo pezzo di software si occupa di raccogliere i dati e trasmetterli tramite le librerie MQTT all'endpoint specificato, ossia all'adapter dedicato di Hono. Le librerie di comunicazione sono basate su Mosquitto il server MQTT appartenente all'Eclipse Foundation il quale essendo molto leggero si presta perfettamente allo scopo, riuscendo a gestire multiple connessioni contemporaneamente anche su dispositivi limitati: da specifiche un'istanza necessita una quantitá di memoria RAM intorno ai 3MB per 1000 client connessi.

Un'ulteriore implementazione, piú semplice ma a suo modo topica, per dimostrazioni di adattabilitá e scalabilitá, é stata fornita attraverso la realizzazione di due simulatori virtuali collegati direttamente agli endpoint di Hono: uno mediante protocollo HTTP ed uno mediante MQTT.

4.4.3 Edge gateway

Nella pratica l'implementazione di un edge gateway é spesso fondamentale in quanto i dispositivi direttamente collegati ai sensori non sono in grado di comunicare con l'infrastruttura Cloud. Le ragioni possono essere multiple:

- Il dispositivo non supporta il protocollo IP
- Il dispositivo non ha le capacitá di memoria/computazionali per utilizzare il protocollo di comunicazione necessario
- Per questioni di sicurezza

In questo caso il dispositivo comunica con il sensore con un metodo stabilito per poi inviare i segnali al cloud. Nell'esempio dimostrativo il gateway é stato usato principalmente per motivi didattici e di testing, poiché il dispositivo a cui é stato collegato il sensore non presenta limitazioni particolari dovute al caso.
La comunicazione $sensore \rightarrow dispositivo$ avviene a livello fisico per mezzo di una libreria che si interfaccia direttamente con la scheda a basso livello. Viene comunque fatto notare che la comunicazione sarebbe potuta essere instaurata anche con altri protocolli, Bluetooth o MQTT ad esempio. Questa funzione di intermediario che svolge il gateway semplifica la comunicazione con il cloud e sottolinea gli aspetti di sicurezza limitando le comunicazioni con l'esterno, ad una singola entitá.

4.4.4 Cloud e Hono

L'infrastruttura cloud basata su Hono si occupa degli aspetti di autenticazione, autorizzazione, messaggistica, statistica e monitoraggio del sistema. Tutto ció in

maniera distribuita, scalabile, modulare e aperta ad eventuali integrazioni.
A questo livello il sistema non é a conoscenza del tipo di dati trasmessi. I dati vegono semplicemente passati ai componenti upstream autorizzati che ne faranno richiesta indistintamente da tipo o provenienza di adapter sul quale sono stati pubblicati. Aspetti e caratteristiche dei componenti di Hono, cosí come il loro funzionamento interno ed esterno, sono stati approfonditi nei paragrafi precedenti del capitolo.

4.4.5 Consumer e Web UI

Tramite credenziali autorizzate i consumer si possono collegare all'endpoint upstream di Hono e ricevere i messaggi pubblicati. I dati ricevuti sono soggetti ad una fase di parsing, analizzati, elaborati e messi a disposizione su un API pubblica. Per semplificare la dimostrazione, non vengono salvati su alcun supporto database, seppur questa opzione sia facilmente implementabile.
Infine, i dati elaborati e distribuiti sull'API sono letti e presentati all'utente finale per mezzo di grafici e tabelle su una SPA sviluppata per lo scopo. Seppur non presentata nell'infrastruttura d'esempio questa struttura permette abilmente anche lo stream di dati in senso opposto: grazie alle funzioni di comunicazione di Hono i componenti upstream potranno comunicare con i dispositivi senza preoccuparsi di instaurare nuove connessioni o mantenere informazioni aggiuntive relative ai destinatari dell'evento da spedire. Ricordiamo infatti che i dispositivi collegati ad Hono possono aggiornare ad ogni messaggio inviato, l'indirizzo del gateway di riferimento, in modo da poter essere costantemente contattabili anche se in movimento; in questo modo sará l'adapter il componente incaricato di recapitare il comando all'indirizzo specificato. Tornando all'esempio presentato precedentemente, nel caso ci sia un restringimento di carreggiata dovuto a manutenzione straordinaria, un amministratore potrá comunicare con l'infrastruttura cloud per inviare l'evento ai singoli edge gateway installati sui veicoli, i quali si occuperanno di attuare misure adeguate alla situazione.

4.4.6 Sezione tecnica

Per la dimostrazione pratica dell'infrastruttura sono stati usati:

Sensore fisico e dispositivo

Seppur non fondamentale per dimostrare il funzionamento dell'infrastruttura é stato usato un sensore fisico per la pubblicazione dei dati. Nel dettaglio, é stato realizzato un circuito su scheda forata che permette il collegamento del sensore a un Raspberry Pi 3. I componenti usati a tale scopo sono i seguenti:
- 1 Sensore di distanza HC-SR04
- 1 Scheda millefori
- Cavi jumper di connessione
- 2 resistenze da 330 Ohm e 470 Ohm

Lo schema del circuito viene fornito nella repository pubblica dedicata [1].

Edge gateway

Data la volontá d'implementazione di un componente gateway é stato scelto ed installato Kura sul RPi3.
Un componente aggiutivo [1] é stato sviluppato appositamente per permettere al sensore di connettersi al cloud e trasmettere i dati. Questi possono avere tre origini diverse a seconda della configurazione:
- Dati generati
- Dati raccolti per mezzo dell'esecuzione di un broker MQTT aggiuntivo (necessita quindi un client esterno)
- Dati raccolti direttamente dal sensore specificando i pin d'accesso per l'invio e l'ascolto dei segnali, utilizzando la libreria GPIO di basso livello

I dati saranno poi inviati all'endpoint specificato mediante l'interfaccia grafica, tramite le librerie standard MQTT (Figura 4.6).

Figura 4.6: Struttura dell'infrastruttura edge con Kura

Cloud e Hono

I componenti di Eclipse Hono sono forniti come immagini di container che possono essere eseguiti su un arbitrario agente di orchestrazione. Al fine di instaurare un ambiente funzionante senza peró doversi affidare a servizi di terze parti a pagamento come GKE o AKS, ho optato per l'installazione di un cluster locale. La prima scelta é ricaduta su Minishift (versione di Openshift), data l'ampia disponibilitá di risorse e template forniti dagli sviluppatori di Hono per tale piattaforma, ma é stato poi accantonato per problemi di instabilitá del sistema e rimpiazzato con Minikube (versione di Kubernetes).
Per controllare lo stato dei componenti cloud é implementato nel sistema anche un servizio di monitoraggio che effettua polling su endpoint dedicati dei singoli container. Questa servizio é svolto da un'istanza del server Prometheus.

Simulatori e Producer

[2] Approfittando dell'infrastruttura Kubernetes giá instaurata, anche simulatori, consumer e server web sono stati eseguiti in container locali. Le istanze di simulazione dei producer sono state sviluppate in Java e realizzate in maniera altamente parametrizzata tramite variabili d'ambiente. In particolare é possibile

specificare indirizzi di connessione, numero di thread e di device istanziati per singolo POD e velocitá di invio dei dati. All'avvio, i due servizi, tenteranno di connettersi agli adapter tramite credenziali e nel caso non trovino riscontro positivo effettueranno in automatico un'autoregistrazione sul tenant d'interesse. Soddisfatto il prerequisito d'autenticazione la connessione potrá essere instaurata per iniziare la pubblicazione di dati. La presenza di due servizi distinti é giustificata dalla volontá di voler testare singolarmente i protocolli di comunicazione: uno sfrutta l'endpoint MQTT e l'altro HTTP.
Quest'implementazione permette anche testing di performance del sistema poiché i container contentenenti i servizi sono facilmente scalabili per aumentare il carico di messaggi inviati verso l'infrastruttura cloud.

Consumer

Servizio che si connette al dispatcher della struttura per ricevere i dati pubblicati dai producer. I messaggi ricevuti vengono filtrati per canale di comunicazione e non viene fatta nessuna assunzione sul tipo di adapter utilizzato del dispositivo in trasmissione. Il collegamanto avviene tramite il protocollo AMQP, mediante librerie di connessione fornite nella repository ufficiale di Hono. Anche questo livello é reso sicuro tramite cifratura del canale di comunicazione e uso di credenziali d'autorizzazione per instaurare la connessione. Il consumer ha la responsabilitá di effettuare il parsing dei dati ricevuti ed esporne i valori nell'API. Manterrá inoltre lo stato dei dispositivi (online/warning/offline) in base al loro *rate* di comunicazione dei valori. Quando un dispositivo non pubblica dati per un periodo abbastanza lungo di tempo, questo viene considerato offline e quindi la sua identitá viene eliminata dall'API. Anche in questo caso tutti i parametri di configurazione sono stati resi dinamici attraverso l'uso di variabili d'ambiente.

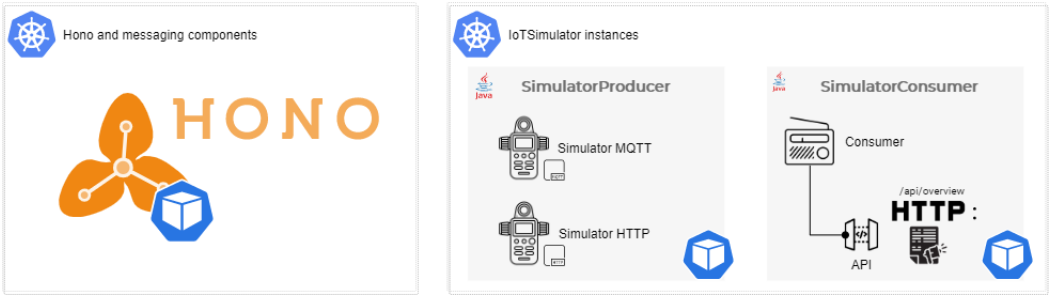

Figura 4.7: Componenti cloud

Web UI

[3] La dashboard di visualizzazione dei dati raccolti é stata realizzata in ambiente web con React. Questa permette di monitorare l'API di un singolo consumer il cui IP é definibile dinamicamente da interfaccia. Viene quindi mostrata una semplice rappresentazione grafica dei dati rilevati su tale endpoint mediante tabelle, grafici animati e rappresentazioni di veloce fruizione. La pagina contiene informazioni di stato sui singoli dispositivi connessi, e le rilevazioni da essi pubblicate in ordine cronologico. In base al timestamp associato ai valori presenti nello stream viene anche calcolata la velocitá media di trasmissione per il singolo producer e quella totale in ricezione per il consumer.
Metriche raccolte e dettagli sullo stato dei componenti cloud, ovvero dati destinati ad amministratori del sistema, sono visualizzabili tramite Grafana. Il servizio previa configurazione con il server di monitoraggio offre una dashboard web ricca di componenti predisposti per la visualizzazione dei dati (Figura 4.8).

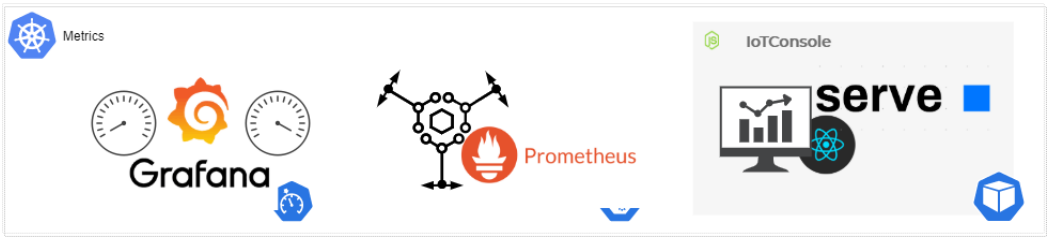

Figura 4.8: Componenti per la visualizzazione delle metriche

4.5 Collaborazione

Durante la costruzione dellâinfrastruttura e soprattutto nello studio dei singoli componenti di Hono ho fatto affidamento principalmente alla documentazione messa a disposizione sul sito web ufficiale. Per alcuni dubbi peró, non riuscendo a trovare riscontri nella guida, mi sono messo in contatto con la cummunity di sviluppatori che é stata molto disponibile nei chiarimenti e dove necessario nella integrazione della documentazione. Ció ha contribuito ad allineare il mio pensiero ad una metodologia di sviluppo open a codice condiviso. Seguendo le linee guida del progetto ho quindi partecipato attivamente anche io nello sviluppo.

Per testare il funzionamento di Hono viene messo a disposizione un client a riga di comando. Avviando questo, con specifici parametri, é possibile connettersi direttamente all'endpoint AMQP o all'indirizzo pubblico del componente di messaging, testando l'infrastruttura sia in funzione di dispositivo sia di applicativo consumer.
La parte di cui mi sono occupato personalmente é stata il rinnovamento dellâinterfaccia a riga di comando. L'obiettivo primario era di semplificare l'interazione dell'utente e agevolare il testing.
L'esecuzione dei comandi era precedentemente legata alla conoscenza dei singoli parametri a riga di comando e non prevedeva ulteriori modalitá d'interazione se non inserendo tutte le informazioni all'avvio.

```
root@debian:~/executer# java -jar hono-cli-*-exec.jar --hono.client.host=$AMQP_NETWORK_IP
                        --hono.client.port=15672 --hono.client.username=consumer@HONO
                        --hono.client.password=verysecret --spring.profiles.active=receiver
                        --tenant.id=$MY TENANT
```

Figura 4.9: Comando di avvio client

Il mio contributo prevede l'utilizzo della libreria di shell spring. L'interfaccia mette a disposizione un comodo prompt per la selezione del comando da eseguire,

controllo dei parametri e una dettagliata spiegazione della loro utilitá. Per questioni di retrocompatibilitá ho mantenuto la possibilitá di inserire i parametri di configurazione all'avvio, i quali potranno poi essere usati come ambiente default per l'esecuzione dei comandi interni. I parametri opzionali dei comandi andranno cosí a sovrascrivere temporaneamente le opzioni default durante la singola esecuzione. Vengono integrati, per fornire maggior possibilitá d'interazione ed eventualmente anche per sviluppi futuri, un helper in grado di stampare messaggi colorati a seconda del livello d'importanza di output ed un input reader, con anche funzioni di autocompletamento, per interrogare l'utente. L'esecuzione dei comandi e l'apertura di nuove connessioni vengono gestite anche con l'utilizzo di multithreading a seconda delle necessitá, per evitare il verificarsi di situazioni bloccanti.

```
root@debian:/home/gio/Scrivania# java -jar hono-cli-1.2.0-SNAPSHOT-exec.jar

  _____    _ _                  _   _
 |  ___|   | (_)                | | | |
 | |__   ___| |_ _ __  ___  ___  | |_| | ___  _ __   ___
 |  __| / __| | | '_ \/ __|/ _ \ |  __  |/ _ \| '_ \ / _ \
 | |___| (__| | | |_) \__ \  __/ | |  | | (_) | | | | (_) |
 |_____\___|_|_| .__/|___/\___| |_|  |_|\___/|_| |_|\___/
               | |
               |_|

Eclipse Hono Example Client  (v1.2.0-SNAPSHOT)
using Spring Boot  (v2.2.4.RELEASE)

Go to https://www.eclipse.org/hono for more information.

HONO:>help
AVAILABLE COMMANDS

Built-In Commands
        clear: Clear the shell screen.
        exit, quit: Exit the shell.
        help: Display help about available commands.
        history: Display or save the history of previously run commands
        script: Read and execute commands from a file.
        stacktrace: Display the full stacktrace of the last error.

Northbounds Commands (Application)
        receive: Start the receiver
                                [-H --host] [-P --port] [-u --username] [-p --password] [-T --tenant] [-M --message]
        send-command: Send a command
                                [-H --host] [-P --port] [-u --username] [-p --password] [-T --tenant]

Southbounds Commands (Device)
        receive-command: Simulate a device receiving and responding to command request messages through AMQP adapter.
                                [-H --host] [-P --port] [-u --username] [-p --password] [-c --certPath] [-k --keyPath] [-t --trustStorePath]
        send: Send telemetry data or events to Hono through AMQP adapter.
                        The default configuration will search for credentials to proceed as an authenticated device.
                        If no credentials were provided the connection will proceed as unauthenticated so be sure to set rightly the address.
                                [-H --host] [-P --port] [-u --username] [-p --password] [-c --certPath] [-k --keyPath] [-t --trustStorePath] [-A --address] [-M --message]


HONO:>
```

Figura 4.10: Shell testuale del client

Conclusioni

Uno studio approfondito delle modalitÃ¡, degli standard e delle tecniche di costruzione di infrastrutture IoT é stato eseguito. Sono stati analizzati i componenti e i protocolli generali comunemente adottati; é stata analizzato il flusso operativo dei dati in un sistema IoT, e passo per passo descritto come ne avviene la gestione. é stata presentata una piccola porzione del panorama attuale attraverso esempi esplicativi per avere una visione d'insieme di quali sono le applicazioni reali e quali invece sono i problemi che ne rallentano o impediscono la crescita. Infine per fornire una possibile soluzione a molte delle problematiche presentate viene proposta un'infrastruttura IoT con un core cloud implementato tramite l'utilizzo di Hono. Componenti e funzionalitá vengono affrontati nel dettaglio per dimostrarne le potenzialitá applicative e diversi moduli di testing accessori all'infrastruttura sono stati sviluppati ed implementati.

Protocolli di comunicazione e sicurezza dei sistemi sono temi molto ampi e oltremodo aspetti delicati e di rilievo nell'operativitá di un sistema IoT. Una visione completa necessiterebbe trattazioni a se stanti per entrambi i campi ma per non deviare dal tema principale si é preferito inserire un'analisi solo in senso generale senza entrare nei dettagli tecnici.

Durante questa valutazione cio che mi ha stupito maggiormente é stato rendermi conto di quanta poca consapevolezza abbiamo in generale della tecnologia che usiamo tutti i giorni. Senza rendercene conto, oggi, molti dei nostri comportamenti sono condizionati dalla tecnologia e da strumenti digitali. Questi sono entrati

lentamente nelle nostre routine abituandoci alla loro presenza, e ormai non ci generano piú senso di stupore o di anormalitá. Mettere in carica l'orologio prima di andare a letto o vedere qualcuno parlare da solo nel corridoio di casa aspettando una risposta da una voce proveniente da un mobile sono comportamenti che solo 50 anni fa ci avrebbero fatto dubitare delle sanitá mentale del nostro interlocutore, mentre oggi sono ordinari metodi di interazione.

Sarebbe necessaria piú coordinazione tra gli esponenti del settore e soprattutto converrebbe fare un po' di chiarezza nell'ambiente, per far prendere maggior coscienza agli utilizzatori comuni di ció che il mercato mette a loro disposizione. In questo modo consumatori e produttori potranno collaborare piú a stretto contatto per l'innovazione e la diffusione di nuove tecnologie IoT.

Appendice A

Eclipse Foundation e l'opensource

La Fondazione Eclipse offre alla comunitá globale di individui e organizzazioni un ambiente maturo, scalabile e favorevole alle imprese per la collaborazione e l'innovazione del software open source.

Il progetto Eclipse é stato originariamente creato da IBM nel novembre 2001 e supportato da un consorzio di fornitori di software. Ancora oggi questo software continua ad essere utilizzato da milioni di sviluppatori.
Nel gennaio 2004 é stata creata come societá indipendente senza fini di lucro la Fondazione Eclipse, per fungere da amministratore della comunitá formatasi. Focalizzata sulla creazione di un ambiente per progetti open source di successo e per promuovere l'adozione della tecnologia Eclipse in soluzioni commerciali e open source, quest'organizzazione, é oggi supportata da oltre 275 membri che ne valorizzano l'esclusivo modello di gestione a gruppo di lavoro. La strategia aziendale é basata su un approccio aperto all'innovazione e mirato alla formazione di una comunitá forte. I committer di Eclipse sono in genere impiegati in organizzazioni o sviluppatori indipendenti che impiegano volontariamente il loro tempo per lavorare ai progetti Eclipse.
Grazie al fondamentale supporto della comunitá di sviluppo e al continuo mantenimento e aggiornamento dell'ecosistema, il modello open source Eclipse, spicca nel mondo dello sviluppo software.

Bibliografia

[1] Gateway https://github.com/johnMinelli/iot-gateway

[2] Simulator https://github.com/johnMinelli/hono-simulator

[3] Console https://github.com/johnMinelli/iot-simulator-console

[4] HPE Aruba IoT Research Report
https://www.scribd.com/document/384136780/HPE-Aruba-IoT-Research-Report-1.

[5] Tschofenig, H., et. al., Architectural Considerations in Smart Object Networking. Tech. no. RFC 7452. Internet Architecture Board, Mar. 2015. Web.
https://www.rfc-editor.org/rfc/rfc7452.txt.

[6] Secondo Rapporto Auditel-Censis: Tra anziani digitali e stranieri iperconnessi, lâItalia in marcia verso la Smart TV
http://www.censis.it/sites/default/files/downloads/Secondo%20Rapporto%20Auditel%20Censis.pdf.

[7] P. Corcoran, "The Internet of Things: Why now, and what's next?," in IEEE Consumer Electronics Magazine, vol. 5, no. 1, pp. 63-68, Jan. 2016.

[8] I. Unwala, Z. Taqvi and J. Lu, "Thread: An IoT Protocol," 2018 IEEE Green Technologies Conference (GreenTech), Austin, TX, 2018, pp. 161-167.

[9] Erikstad, S. O. (2017). Merging Physics, Big Data Analytics and Simulation for the Next-Generation Digital Twins. In V. Bertram (Ed.), High-Performance Marine Vehicles (pp. 141â151). Durbanville, South Africa.

[10] C. Tsai, C. Lai, M. Chiang and L. T. Yang, "Data Mining for Internet of Things: A Survey," in IEEE Communications Surveys Tutorials, vol. 16, no. 1, pp. 77-97, First Quarter 2014.

[11] Sholla, Sahil Kaur, Sukhkirandeep Begh, Gh Rasool Mir, Roohie Chishti, Ahsan. (2017). Clustering Internet of Things: A Review. Journal of Science and Technology: Issue on Information and Communications Technology. 3. 26. https://doi.org/10.31130/jst.2017.61.

[12] M. Saeid Mahdavinejad, M. Rezvan, M. Barekatain, P. Adibi, P. Barnaghi, Amit P. Sheth, Machine learning for internet of things data analysis: a survey, Digital Communications and Networks, Volume 4, Issue 3, 2018, Pages 161-175, ISSN 2352-8648, https://doi.org/10.1016/j.dcan.2017.10.002.

[13] A. C. Onal, O. Berat Sezer, M. Ozbayoglu and E. Dogdu, "Weather data analysis and sensor fault detection using an extended IoT framework with semantics, big data, and machine learning," 2017 IEEE International Conference on Big Data (Big Data), Boston, MA, 2017, pp. 2037-2046.

[14] L. Xiao, X. Wan, X. Lu, Y. Zhang and D. Wu, "IoT Security Techniques Based on Machine Learning: How Do IoT Devices Use AI to Enhance Security?," in IEEE Signal Processing Magazine, vol. 35, no. 5, pp. 41-49, Sept. 2018.

[15] Y. Meidan, M. Bohadana, et al. 2017. Detection of unauthorized iot devices using machine learning techniques. arXiv preprint arXiv:1709.04647 (2017).

[16] S. Al-Sarawi, M. Anbar, K. Alieyan and M. Alzubaidi, "Internet of Things (IoT) communication protocols: Review," 2017 8th International Conference on Information Technology (ICIT), Amman, 2017, pp. 685-690.

[17] M. Mohammadi and A. Al-Fuqaha, "Enabling Cognitive Smart Cities Using Big Data and Machine Learning: Approaches and Challenges," in IEEE Communications Magazine, vol. 56, no. 2, pp. 94-101, Feb. 2018.

[18] Q. Wu et al., ”Cognitive Internet of Things: A New Paradigm Beyond Connection,” in IEEE Internet of Things Journal, vol. 1, no. 2, pp. 129-143, April 2014.

[19] N. A. M. Alduais, J. Abdullah, A. Jamil and L. Audah, ”An efficient data collection and dissemination for IOT based WSN,” 2016 IEEE 7th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON), Vancouver, BC, 2016, pp. 1-6.

[20] Pawluk, Przemyslaw, Marin Litoiu and Nick Cercone. âFrom QoD to QoS - Data Quality Issues in Cloud Computing.â CLOSER (2011).

[21] A. Al-Qamash, I. Soliman, R. Abulibdeh and M. Saleh, ”Cloud, Fog, and Edge Computing: A Software Engineering Perspective,” 2018 International Conference on Computer and Applications (ICCA), Beirut, 2018, pp. 276-284.

[22] S. Krćo, B. Pokrić and F. Carrez, ”Designing IoT architecture(s): A European perspective,” 2014 IEEE World Forum on Internet of Things (WF-IoT), Seoul, 2014, pp. 79-84.

[23] S. Amghar, S. Cherdal and S. Mouline, ”Which NoSQL database for IoT Applications?,” 2018 International Conference on Selected Topics in Mobile and Wireless Networking (MoWNeT), Tangier, 2018, pp. 131-137.

[24] Atzori, L., Iera, A., Morabito, G. (2010). The internet of things: A survey. Computer Networks , 54 (15), 2787â2805. https://doi.org/10.1016/j.comnet.2010.05.010.

[25] EY: Future of LoT https://www.scribd.com/document/418400142/EY-Future-of-Lot.

[26] Google IPv6 Statistics https://www.google.com/intl/en/ipv6/statistics.html#tab=ipv6-adoption.

[27] Ur Rehman MH, Liew CS, Abbas A, Jayaraman PP, Wah TY, Khan SU. Big data reduction methods: a survey. Data Sci Eng. 2016;1(4):265â84. https://doi.org/10.1007/s41019-016-0022-0.