NICOLA MAZZOCCA

Cloud Edge Fog

Di Franca Rocco di Torrepadula
Alessandra Somma
Francesco Vitale

Contents

Chapter 1

Cloud Computing

1.1 Cenni storici

Il Cloud Computing nasce come evoluzione di due tecnologie già esistenti, ovvero il ***Grid Computing*** e l'***Utility Computing.*** Alla base del Grid Computing vi è la condivisione coordinata di risorse all'interno di una dinamica e multi-istituzionale *organizzazione virtuale.* Per risorse non si intendono solo file, ma anche risorse software ed hardware. Un sistema Grid è *"un sistema che coordina le risorse che non sono soggette a controllo centralizzato, utilizzando standard, protocolli aperti e generici, e interfacce per fornire qualità elevate di servizio*" [1].

Per rendere tale infrastruttura maggiormente scalabile si passa al modello Utility Computing, in cui le risorse sono messe a disposizione *on-demand*: in tal modo l'utente paga il servizio offerto in base all'utilizzo che ne fa.

Il Cloud Computing nasce dall'unione dei principali vantaggi dei due modelli presentati, ovvero sfruttare la cooperazione di diverse risorse per aumentare la capacità computazionale e fornire tali servizi in modalità on-demand, ad un prezzo che dipende dall'effettivo utilizzo degli stessi. Nel modello Cloud Comuting, il focus principale è fornire servizi più veloci e di maggiore qualità ad un prezzo più basso [2].

1.2 Definzione di Cloud Computing

Il NIST (National Institute of Standards and Technology) definisce il Cloud Computing nel seguente modo: "*il Cloud Computing è un modello per consentire un accesso alla rete contemporaneo, conveniente e on-demand, ad un pool condiviso di risorse informatiche configurabili (ad esempio, reti, server, storage, applicazioni e servizi) che può essere rapidamente fornito e rilasciato con il minimo sforzo di gestione o di interazione da parte del fornitore di servizi* [3]. Come descrive il NIST, le cinque caratteristiche essenziali di un sistema Cloud sono le seguenti:

- ***On-demand self-service.*** Un cliente può unilateralmente rifornirsi capacità di calolo, come tempo di calcolo o capacità di immagazzinamento dei dati, senza necessitare di interazione umana.
- ***Broad network access.*** Le risorse sono disponibili in rete e sono accessibili tramite meccanismi standard che promuovono l'uso da piattaforme client eterogene, thin o thick (smartphone, tablet, laptop e workstation).
- ***Resource pooling.*** Le risorse fornite dal provider sono realizzate e distribuite in modo da poter servire più utenti, realizzando un modello *multi-tenant.* Grazie a tale

modello le risorse sono assegnate dinamicamente, coerentemente con le richieste di ogni utente. In tal modo, si ottiene la **trasparenza della locazione** in quanto l'utente è ignaro dell'ubicazione fisica delle risorse che utilizza.

- ***Rapid elasticity.*** Le risorse possono essere fornite e rilasciate in modo elastico, e in alcuni casi automatico, così da scalare rapidamente in accordo con la richiesta. In tal modo all'utente le risorse appaiono spesso illimitate, accessibili in ogni quantità in ogni istante.

- ***Measured service.*** Il sistema cloud controlla ed ottimizza l'uso delle risorse, monitorandolo. Infatti, l'utilizzo delle risorse può essere monitorato, controllato e descritto utilizzando un determinato livello di astrazione.

1.3 Modelli di servizio (SaaS, PaaS, IaaS)

I modelli di servizio descrivono come un provider fornisce servizi, utilizzabili dall'utente. Il NIST in [3] definisce tre modelli fondamentali di erogazione dei servizi, che differiscono per l'accessibilità delle risorse.

- ***Software as a Service (SaaS)***. Il consumer può utilizzare i servizi offerti dal provider, che girano sull'infrastruttura cloud. Le applicazioni sono accessbili da client eterogenei, tramite un'interfaccia anche *thin*, come un web browser. Il consumer non può in alcun modo controllare o gestire l'infrastruttura cloud, può unicamente configurare impostazioni utente per personalizzare l'utilizzo della piattaforma o dell'applicazione. In particolare, tale modello può essere ***business-oriented*** o ***end-user-oriented***, a seconda che i servizi siano realizzati per aziende di grosse dimensioni o per il singolo utente.

- ***Platform as a Service (PaaS).*** Il consumer può distribuire sull'infrastruttura cloud applicazioni, da esso create o acquistate, che usano linguaggi di programmazione, librerie, servizi e tool supportati dal provider (senza precludere l'uso di linguaggi, librerie, servizi e tool compatibili, di terze parti). Il consumer non gestisce l'infrastruttura cloud sottostante (rete, server, sistema operativo, storage etc.) ma ha il controllo dell'applicazione ed eventualmente su impostazioni dell'ambiente ospitante. In tal modo l'utente ha flessibilità di sviluppo e distribuzione, senza dover tener conto delle problematiche dell'infrastruttura sottostante, in particolare hardware. Le caratteristiche di un sistema PaaS sono essenzialmente due ([4]):

  - *Delegation.* La piattaforma PaaS fornisce i contenitori ed un ambiente per le applicazioni che ospita. In tal modo gli sviluppatori di tali applicazioni non devono conoscere l'infrastruttura.
  - *Transaction-intensive.* La maggior parte delle applicazioni PaaS sono Web, dunque sono sopposte ad un elevato carico di transazioni per unità di tempo (transaction-intensive). Per tale motivo, ogni transazione dev'essere quanto più leggera possibile, per non portare ad un decadimento delle prestazioni.

- ***Infrastructure as a Service (IaaS)***. Il consumer usufruisce di risorse computazionali, di storage, di connessione etc. per eseguire software arbitrari, tra cui anche sistemi operativi e applicazioni. L'utente non può gestire l'infrastruttura sottostante ma ha il controllo su sistema operativo, storage e applicazioni, inoltre ha un controllo limitato su alcuni componenti di rete, come i firewall. Quelle che sono messe a disposizione, dunque, sono features hardware, per tale motivo all'IaaS ci si riferisce spesso come HaaS (Hardware as a Service).

Nei sistemi IaaS bisogna considerare alcune problematiche, come la sicurezza dei dati sensibili e la disponibilità del servizio.

1.4 Architettura Cloud

L'architettura di un sistema cloud viene descritta dal NIST in [2] e definisce *cosa* devono fornire i servizi e non *come* realizzare l'implementazione. In figura 1.1 vediamo l'architettura di riferimento del NIST, che identifica cinque attori principali e le loro funzionalità all'interno del sistema cloud.

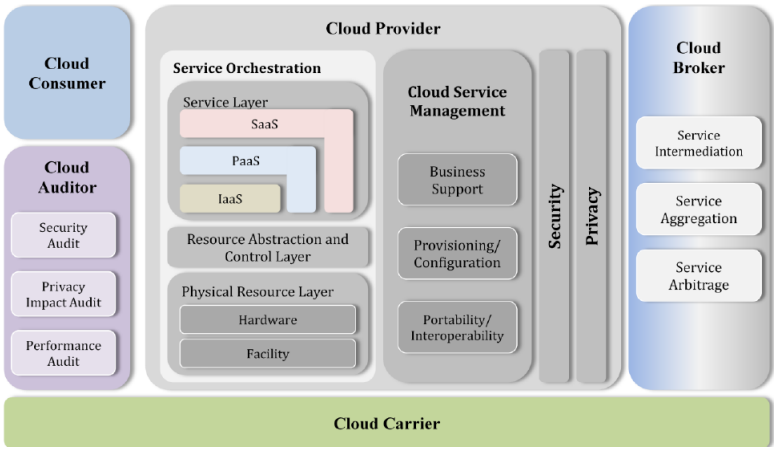


Figure 1.1: Modello NIST.

Ogni attore è un'entità (sistema o persona) che partecipa ad una transazione o processo del sistema Cloud.

- ***Cloud consumer.*** Entità che utilizza servizi offerti da Cloud providers.
- ***Cloud provider.*** Entità responsabili di rendere servizi disponibili a parti interessate.
- ***Cloud auditor.*** Parte che conduce una valutazione indipendente dei servizi cloud, delle operazioni del sistema informativo, delle prestazioni e della sicurezza del sistema Cloud.
- ***Cloud broker.*** Entità che gestisce l'uso, le performance e la distribuzione dei servizi cloud; negozia le relazioni tra provider e consumer.
- ***Cloud carrier.*** Intermediario che fornisce connettività e trasporto dei servizi cloud, da provider e consumer.

In figura 1.2 vediamo le interazioni tra gli attori. Il cloud consumer può richiedere il servizio direttamente ad un cloud provider oppure tramite il broker, che ad esempio potrebbe fornire un nuovo servizio combinando servizi di più provider. Il cloud auditor conduce una valutazione indipendente dei servizi e potrebbe contattare gli altri attori per collezionare le informazioni necessarie.

1.4.1 Cloud Consumer

I cloud consumer sono i principali stakeholder di un sistema cloud. Sono entità che utilizzano servizi offerti da cloud providers: scaricano il catalogo di servizi offerti, richiedono il servizio

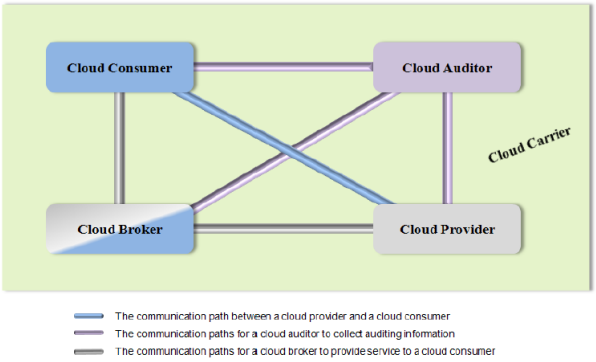


Figure 1.2: Comunicazione tra gli attori.

desiderato accettando un "contratto" e usufruiscono di tale servizio. I cloud consumer e il cloud provider stipulano un contratto chiamato ***SLA (Service Level Agreements)*** che definisce i requisiti tecnici soddisfatti dal cloud provider. Gli SLA possono riguardare caratteristiche come la qualità del servizio offerto o la sicurezza, ma indicano anche restrizioni a cui è sottoposto il consumer. In figura 1.3 vediamo esempi di servizi offerti a un cloud consumer.

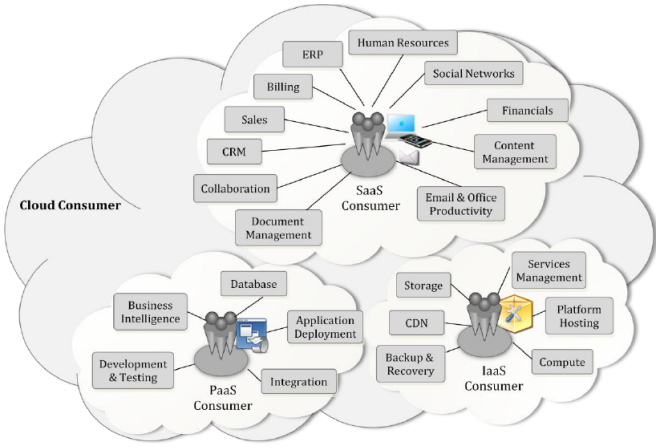


Figure 1.3: Esempi di servizi offerti ai cloud consumer.

Per applicazioni SaaS i consumer possono essere: aziende, che forniscono ai loro dipendenti l'accesso ad applicazioni software; end-user, che usano direttamente tali applicazioni; amministratori di applicazioni software, che configurano le applicazioni per gli utenti finali.

I consumer di applicazioni PaaS invece possono essere sviluppatori, tester o deployer di applicazioni, che sfruttano i tool e le risorse di esecuzione fornite dal cloud provider.

Infine, i consumer di applicazioni IaaS hanno accesso a risorse virtuali, infrastrutture o componenti di rete ed in genere a risorse di caloclo tramite cui possono eseguire software arbitrati. Per tali motivi, i consumer sono tipicamente sviluppatori di sistemi, amministratori di sistemi o IT mangers. In generali i consumer, di un qualunque tipo di applicazione cloud, pagano tale servizio in relazione al tempo in cui l'hanno utilizzo o alla quantità di risorse impiegate.

1.4.2 Cloud Provider

Il cloud provider è l'entità che rende disponibili i propri servizi a terze parti. Esso acquisisce e gestisce le infrastrutture di calcolo necessarie ad erogare il servizio, esegue il software cloud che fornisce i servizi e fa in modo che il servizio sia disponibile ai cloud consumer mediante la rete. In figura 1.4, vediamo le principali attività svolte da un cloud provider.

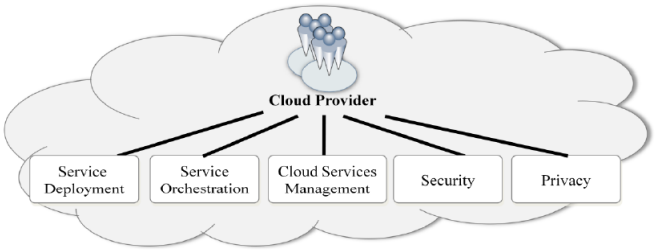


Figure 1.4: Principali attività di un cloud provider.

Per applicazioni SaaS i cloud provider forniscono, configurano, effettuano manutenzioni e aggiornano i software dell'infrastruttura, in modo tale che i servizi siano offerti con il livello di qualità atteso dai consumer.

Per applicazioni PaaS i cloud provider gestisco l'infrastruttura di calcolo della piattaforma, fornendo servizi come database, middleware , IDE o SDK.

Infine, nelle applicazioni IaaS il cloud provider acquista le risorse di calcolo fisiche ed esegue il software cloud necessario a rendere le risorse disponibili ai consumer, tramite servizi di interfacciamento o di astrazione. Come abbiamo visto nella sezione 1.3, i consumer di sistemi IaaS hanno più libertà nella configurazione del sistema ma solo il cloud provider ha il controllo dell'infrastruttura hardware, come i server, i meccanismi i rete, i device di storage, l'host OS e l'hypervisor.

In figura 1.5, vediamo il controllo di cloud consumer e provider sul sistema, a seconda della differente modello di servizio (SaaS, PaaS, IaaS).

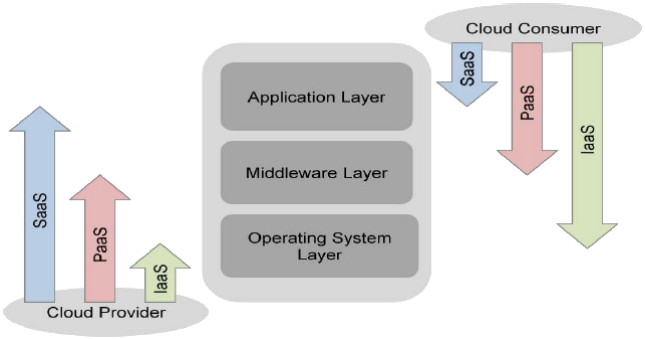


Figure 1.5: Livello di controllo di provider e consumer.

1.4.3 Cloud Auditor

Un cloud auditor è un'entità che svolge una valutazione indipendente del servizio cloud. In particolare, un cloud auditor deve verificare la conformità rispetto a determinati standard e valutare il servizio offerto in termini di security control, privacy, performance, etc. I security control sono l'insieme di contromisure e tecniche adottate in un sistema per ottenere

i requisiti ***CIA (Confidentialiy, Integrity, Availability)*** [5]. A tal proposito, il cloud auditor valuta i security control in termini di quanto correttamente sono implementati, operano e producono i risultati attesi.

1.4.4 Cloud Broker

A causa dell'evoluzione del cloud computing, l'integrazione dei servizi messi a disposizione dai provider potrebbe essere troppo complessa per il cloud consumer. In tal caso, il cloud consumer non richiede il servizio direttamente al provider ma si affida a un cloud broker, che si occupa della distribuzione del servizio e della negoziazione tra providers e consumers. I servizi forniti da un cloud broker sono di tre tipologie:

- ***Service Intermediation.*** Il cloud broker espande i servizi forniti dal provider, ad esempio aggiungendo nuove funzionalità.
- ***Service Aggregation.*** Il cloud broker integra servizi differenti in uno o più nuovi servizi.
- ***Service Arbitrage.*** Tale servizio è simile al precedente con la differenza che i servizi che vengono aggregati non sono fissati, il broker ha la possibilità di scegliere servizi erogati da aziende differenti.

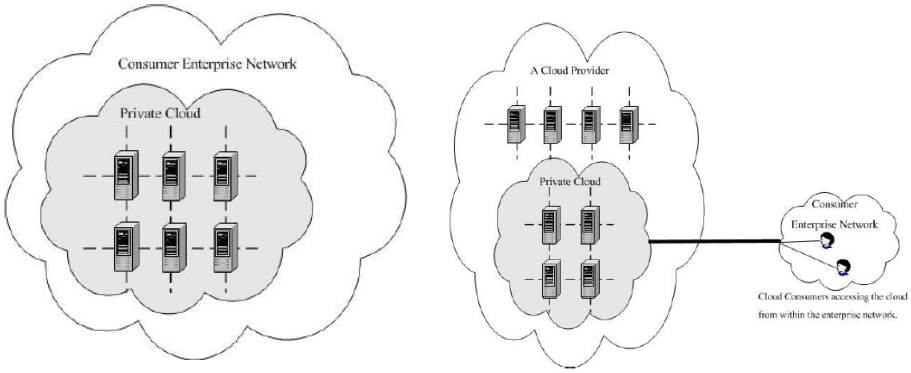
1.4.5 Cloud Carrier

Il cloud carrier è un intermediario che fornisce servizi di connettività e trasporto tra i cloud consumers e i cloud providers. Quando un cloud provider stipula uno SLA con i cloud consumer, potrebbe richiedere al cloud carrier di fornire connessioni dedicate e sicure per fornire i propri servizi ai cloud consumer, in accordo con le condizioni dello SLA.

1.5 Modelli di distribuzione

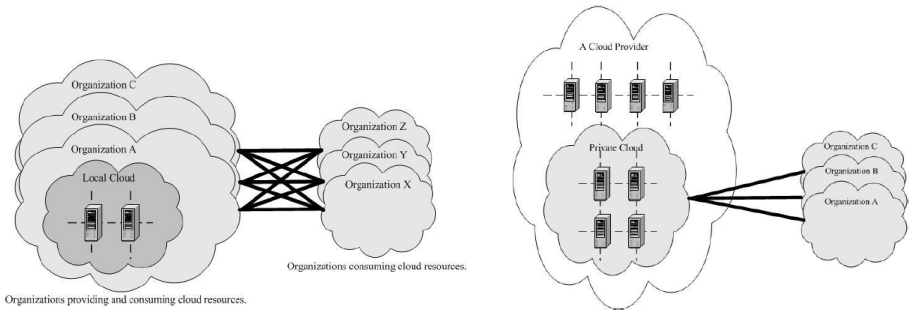
Il NIST, nel definire il concetto e le caratteristiche del Cloud Computing, definisce anche 4 modelli di distribuzione [3], ovvero:

- ***Private cloud.***L'uso dell'infrastruttura cloud è *concesso esclusivamente ad un'organizzazione, composta da più consumers.* Il proprietario e gestore dell'infrastruttura può essere l'organizzazione stessa, una terza parte o una combinazione di esse. Fisicamente, l'infrastruttura può essere ospitata negli stessi locali dell'organizzazione (*on-site private cloud*, figura 1.6a) consumatrice o di terze parti (*outsourced private cloud*, figura 1.6b).
- ***Community cloud.*** L'uso dell'infrastruttura cloud è *concesso esclusivamente ad una specifica community di consumers*, provenienti da organizzazioni che hanno interessi comuni. L'infrastruttura può appartenere ed essere gestita da una o più aziende della community, una terza parte o una combinazione di esse. Fisicamente, l'infrastruttura può essere ospitata negli stessi locali di una delle organizzazioni consumers (*on-site community cloud*, figura 1.7a)) o di terze parti (*outsourced community cloud*, figura 1.7b).
- ***Public cloud***. *L'infrastruttura cloud e le risorse computazionali sono accessibili al pubblico generale, tramite una rete pubblica.* L'infrastruttura tipicamente è ospitata nei locali del cloud provider, che può essere un'azienda, un'organizzazione accademica o del governo (figura 1.8).



(a) *on-site private cloud* (b) *outsourced private cloud*

Figure 1.6: Private cloud.



(a) *on-site community cloud* (b) *outsourced community cloud*

Figure 1.7: Community cloud.

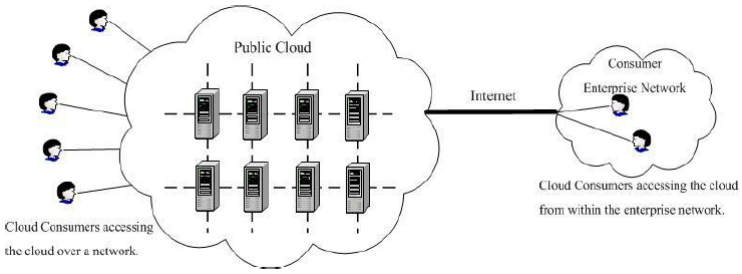


Figure 1.8: Public cloud.

- ***Hybrid cloud.*** L'infrastruttura cloud è *composta da due o più infrastrutture cloud distinte*, tra quelle precedentemente esposte, che rimangono entità uniche ma sono vincolate insieme da una tecnologia standardizzata o proprietaria che consente la portabilità dei dati e delle applicazioni (figura 1.9).

I quattro modelli di distribuzione visti cono definiti dal NIST. Oggi si parla anche di ***Multicloud Computing*** che consiste nell'aggregare molteplici cloud dello stesso tipo (pubblico/privato/community), offersi da differenti fornitori. Questa tipologia di cloud si differenzia dal cloud ibrido in quanto adesso i cloud che aggreghiamo sono della stessa tipologia, cosa che non avveniva nel cloud ibrido.

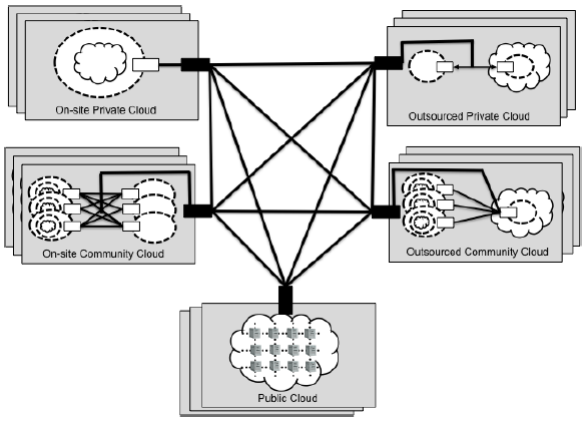


Figure 1.9: Hybrid cloud.

1.6 Service orchestration

Il termine *service orchestration* si riferisce alla composizione dei componenti di sistema per supportare le attività dei Cloud Provider nell'organizzazione, coordinamento e gestione delle risorse informatiche al fine di fornire servizi cloud ai Cloud Consumer [2]. Il NIST descrive ciò con un modello a 3 livelli (figura 1.10). Il livello più alto è il *service layer*, dove sono definite le interfacce tramite cui i cloud consumers accedono ai servizi offerti dai provider. Abbiamo presentato SaaS, PaaS e IaaS come 3 modelli differnti, tuttavia è possibile che un'applicazione SaaS sia costruita sopra una PaaS così come una PaaS può essere costruita sopra una IaaS.

Il livello intermedio è il *resource abstraction and control layer* e contiene i componenti utilizzati dal provider per gestire l'accesso alle risorse fisiche. Componenti di resource abstraction sono, ad esempio, macchine virtuali ed hypervisor, usate per garantire un uso efficiente, sicuro ed affidabile delle risorse fisiche sottostanti. Esempi di ricorse di control sono i componenti software che si occupano dell'allocazione delle risorse o dell'access control.

Infine, al livello più basso abbiamo il *physical resource layer*, dove troviamo tutte le risorse fisiche, come CPU, memorie, router, firewall, switch, componenti di storage o componenti di ventilazione.

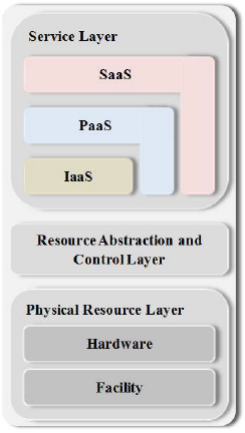


Figure 1.10: Service Orchestration.

1.7 Sicurezza nel cloud

Grazie al cloud computing, gli utenti possono utilizzare servizi accedendo al sistema dovunque e in ogni momento, tramite Internet. Gli utenti possono inoltre utilizzare il servizio offerto on-demand, senza doversi preoccupare dei problemi di gestione e manutenzione retrostanti. Queste caratteristiche del cloud computing, che ne rappresentano la forza, sono anche la causa di grosse problematiche a livello di sicurezza. Se pensiamo, ad esempio, alla sicurezza dei dati, nel caso di classici data center le applicazioni sono distribuite all'interno dell'istituzione, dunque in confini fidati, monitorati e soprattutto *statici*. Per essere certi della sicurezza raggiunta dal data center possiamo utilizzare meccanismi come VPN, firewall, IDS (Instrusion Detection Systems), IPS e autenticazione a più fattori. Con l'utilizzo di servizi cloud, in particolare cloud pubblici, tali confini diventano *dinamici*. I dati possono trovarsi in posti differenti, dove le stesse politiche di gestione della data security possono essere differenti [6].

La sicurezza del cloud si riferisce alla protezione dei dati, delle applicazioni e dell'intera infrastruttura del cloud computing. In particolare, **il modello di distribuzione più vulnerabile, dal punto di vista della sicurezza, risulta essere il public cloud**. Infatti, questa tipologia di cloud è *multi-tenant*, vi accedono più utenti, tramite diversi punti di accesso. Questo rende il sistema più facilmente attaccabile, in particolare manca l'isolamento di cui godono i cloud privati. In quest'ottica i cloud ibridi sono vantaggiosi, essendo una composizione di cloud differenti ci permettono di scegliere dove memorizzare, ad esempio, i dati: abbiamo la possibilità di collocare i dati più sensibili sul cloud privato e quelli meno sensibili sul cloud pubblico.

Anche la scelta del modello di servizio pone problematiche di sicurezza differenti. Ad esempio, nel caso di sistemi SaaS avremo per lo più problemi relativi alla privacy dei dati, integrità accesso, autenticazione, disponibilità e problemi di backup. Nel caso di sistemi IaaS ci possono essere problematiche di sicurezza legati ai sistemi operativi o agli hypervisor [7]. La sicurezza del cloud si suddivide in quattro livelli, ovvero infrastructure security, system service security, application security e data security. A seconda del modello di servizio, i quattro livelli di sicurezza sono divisi diversamente tra cloud service provider e cloud service costumer, come mostra la figura 1.11 [8]

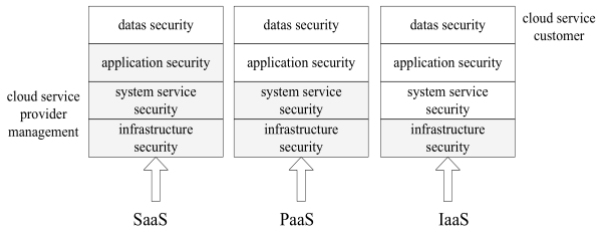


Figure 1.11: Divisione dei quattro livelli di sicurezza tra consumer e provider.

L'ENISA (European Network and Information Security Agency) ha descritto i principali *security risks* del cloud computing) [9], ovvero:

- ***Loss of governance.*** Utilizzando le infrastrutture cloud, i client cedono parte dei controlli al cloud provider, in un modo che potrebbe impattare la sicurezza. Nello SLA inoltre il provider potrebbe non impegnarsi a fornire i servizi di sicurezza, lasciando così un gap.

- ***Lock-in.*** Ad oggi non sempre viene fatto lo sforzo di utilizzare tool, formati e procedure che possano garantire la portabilità dei dati, delle applicazioni e del servizio.

Questo può rendere difficile, per il consumer, migrare da un provider ad un altro, o tornare ad un servizio interno all'ambiente di IT.

- ***Isolation failure.*** Il multi-tenancy e la condivisione delle risorse sono le caratteristiche principali del cloud computing. Tuttavia, ci possono essere dei fallimenti nel meccanismo che manteiene separato lo storage, la memoria o ad esempio il routing di client differenti (come nel caso di *guest-hopping attack*).

- ***Management interface compromise.*** Le interfacce di gestione del cliente in un cloud pubblico sono accessibili tramite internet e permettono, tramite essere, di accedere a un grosso insieme di risorse. Per tale motivo, danno luogo a nuovi rischi, soprattutto se combinate con l'accesso remoto e le vulnerabilità già presenti nei browser web.

- ***Data protection.*** Talvolta, il cloud consumer può avere difficoltà nel controllare le pratiche di gestione dei dati, adottate dal cloud provider, e quindi essere sicuro che i dati siano trattati nel modo corretto. In altri casi, invece, i cloud provider danno evidenza del modo in cui trattano i dati, esibendo anche certificazioni.

- ***Insecure or incomplete data deletion.*** Quando viene richiesta l'eliminazione di una risorsa cloud ciò potrebbe non comportare sempre la cancellazione effettiva dei dati. In alcuni casi questa può anche non essere possibile, o almeno non tempestivamente, o perchè le cope dei dati memorizzati non sono attualmente disponibili o perchè il disco da formattare contiene anche dati di altri client.

- ***Malicious insider.*** I danni che possono essere causati da insider malevoli, anche se rari, sono i più gravi. Le architetture cloud necessitano infatti di alcuni ruoli che sono estramente rischiosi, come l'amministratore del sistema e l'amministratore di sicurezza, lato provider.

- ***Customers' security expectations.*** La percezione che ha il costumer del livello di sicurezza potrebbe non rispecchiare correttamente l'effettiva sicurezza offerta dal cloud provider. In alcuni casi i cloud provider riducono i costi (e dunque i prezzi dei servizi offerti), andando a sacrificare alcuni aspetti di sicurezza.

- ***Availability Chain.*** L'affidamento alla connettività di Internet da parte del customer genera, in molti casi, un single point of failure.

1.7.1 Sicurezza nel cloud pubblico

Queste problematiche riguardano in generale il mondo cloud ma in particolare il cloud pubblico. In quest'ultimo caso le preoccupazioni principali sono la perdita dei dati (***data leakage***) ed attacchi di ***spoofing***. Durante tali attacchi entità malevole fingono di leggere, modificare o eliminare i dati dal servizio come se fosse un costumer, un operatore o sviluppatore del sistema. Oppure, riesce a gestire la piattaforma di controllo o ad utliizzare le vulnerabilità del sistema per infiltrarvisi, controllando o interrompendo il funzionamento del servizio. In tal caso si parla di attacco di tipo ***denial of service***, si sovraccarica il sistema con richieste lecite ma malevole, così da impedirne l'accesso agli utenti leggitimi [8].

Inoltre, a causa della condivisione di risorse fisiche tra macchine virtuali o delle dipendenze degli utenti nello stesso sistema operativo possono essere sferrati attacchi di virtualizzazione, come ***virtual machine hopping, virtual machine escaping*** o ***side channel attacks***. In un ambiente virtualizzato, i cloud costumer gestiscono i servizi tramite le API. La sicurezza e la disponibilità dei servizi cloud dipende allora dalla sicurezza delle API. Una volta minata la sicurezza delle API possono verificarsi attacchi di ***data tampering***, ***data***

repudation e così via [8].

Per quanto riguarda i dati, possiamo suddividerli in dati relativi agli utenti e alle imprese e dati riguardanti le operazioni giornaliere. Nel primo caso, i dati tipicamente comprendono informazioni private e personali, che devono essere trattate in accordo con la ***normativa europea GDPR (General Data Protection Regulation)***. Secondo tale normativa, le aziende devono ottenere il consenso degli utenti per raccogliere, memorizzare e utilizzare i dati personali. Dunque, anche nel caso del cloud è necessario che i dati vengano gestiti e acceduti in accordo con la normativa GDPR, utilizzando meccanismi di autenticazione, e non solo, per accedere ad essi. I dati relativi alle operazioni giornaliere, invece, sono classificati dal provider a seconda della loro sensibilità e criticità. Al fine di garantire la sicurezza dei dati dei client, i cloud provider forniscono ai clienti misure di protezione dei dati per garantire l'integrità, la disponibilità e la riservatezza della trasmissione e memorizzazione dei dati. In termini di trasmissione e archiviazione dei dati, i cloud provider forniscono algoritmi di crittografia per garantire l'integrità e la riservatezza della trasmissione dei dati. In termini di disponibilità dei dati, i provider di servizi cloud implementano un meccanismo di backup multi-point per garantire la coerenza del backup dei dati. Per quanto riguarda la sensibilità dei dati, i cloud provider devono garantire un isolamento dei dati negli ambienti client e impediscono operazioni non autorizzate. Devono inoltre assicurare che i dati sensibili cancellati dai client siano completamente distrutti, così da impedire un utilizzo multiplo dei dati stessi [8].

1.8 OpenStack: un esempio di IaaS [1]

OpenStack è un progetto IaaS nato nel 2010, dalla collaborazione di NASA e Rackspace cloud, in costante evoluzione.

> *OpenStack è un sistema operativo cloud che controlla grandi pool di risorse di elaborazione, storage e rete in un datacenter, tutti gestiti e forniti tramite API con meccanismi di autenticazione comuni*

OpenStack non è semplicemente un esempio di IaaS ma fornisce componenti aggiuntivi relativi ad orchestrazione, fault e service management, al fine di garantire un'elevata disponibilità di applicazioni utenti. Tale piattaforma è realizzata in Python, cercando di esplicare al meglio il principio di modularità. Infatti, i componenti di OpenStack, che comunicano tramite apposite API, sono realizzati in modo tale da ottenere flessibilità, scalabilità, indipendenza ed elasticità. In particolare, la **flessibilità** è uno dei punti di forza di tale piattaforma, oltre alla sua natura **open-source**. Utilizzando OpenStack possiamo infatti decidere quali componenti utilizzare ed eventualmente aggiungerne ulteriori successivamente.

1.8.1 Componenti

OpenStack è formato da una serie di componenti, realizzati come progetti indipendenti, realizzati separatamente, che comunicano tramite apposite API per realizzare i servizi offerti all'utente.

Compute (Nova)

Nova è il progetto Openstack che si occupa di fornire istanze di calcolo (***server virtuali***). Nova supporta la creazione di macchine virtuali, server baremetal ed ha un supporto limitato per i container di sistema. Tale progetto viene eseguito come un insieme di demoni su

[1] Il seguente paragrafo fa riferimento alla documentazione officiale Openstack, disponibile su https://docs.openstack.org/

server Linux. Gli utenti del sistema usano Nova per creare e gestire server o tramite tool o direttamente tramite API.

Identity (Keystone)

Keystone è un servizio Openstack che fornisce le API per l'autenticazione client, service discovery e autorizzazione multi-tenant distribuita, implementando le Openstack's Identity API. Essenzialmente, Keystone è il servizio che si occupa di **gestire identificazione, autenticazione ed autorizzazione** in OpenStack. Keystone è organizzato come un gruppo di servizi interni esposti su uno o più endpoint. Quando un utente si autentica al sistema, Keystone convalida le credenziali utente/progetto con il servizio Identity e, con successo, crea e restituisce un token con il servizio Token. Tale token permetterà di identificare l'utente nel sistema, fino alla sua scadenza.

Il servizio Identity fornisce autenticazione delle credenziali e dati per utenti e gruppi. Nel caso di base, questi dati sono gestiti dal servizio Identity, che consente di gestire anche tutte le operazioni CRUD associate ad essi. In casi più complessi, i dati sono invece gestiti da un autorevole servizio di backend. Un esempio di questo sarebbe quando il servizio Identity funge da interfaccia per ***LDAP***. In tal caso il server LDAP è la *source of truth* e il ruolo del servizio Identity è quello di trasmettere tali informazioni in modo accurato.

Object Storage (Swift)

Object Storage (Swift) è il servizio OpenStack utilizzato per l'archiviazione dei dati ridondante e scalabile, utilizzando cluster di server per memorizzare petabyte di dati. Si tratta di un **sistema di archiviazione a lungo termine per grandi quantità di dati statici** che possono essere recuperati e aggiornati. Object Storage utilizza un'architettura distribuita senza un punto centrale di controllo, fornendo maggiore scalabilità, ridondanza e permanenza. Gli oggetti sono scritti su più dispositivi hardware; il software Openstack che garantisce la replica dei dati e la loro integrità, in tutto il cluster. **Gli storage cluster scalano orizzontalmente, aggiungendo nuovi nodi**. Se un nodo fallisce, Openstack lavora per replicare il suo contenuto da altri nodi attivi. Poiché Openstack utilizza la logica del software per garantire la replicazione e la distribuzione dei dati su diversi dispositivi, è possibile utilizzare dischi rigidi e server economici al posto di apparecchiature più costose.

Image Service (Glance)

Il progetto Image (Glance) fornisce il servizio tramite cui gli utenti possono caricare e scoprire risorse di dati da utilizzare con altri servizi. Questo include attualmente le definizioni di metadati e immagini. Glance è il nome attribuito al progetto OpenStack che conferisce agli utenti la possibilità di caricare, modificare e utilizzare gruppi di dati sintetici condivisibili con le altre sezioni della piattaforma; tali tipi di dati sintetici sono le immagini e i metadati

Block Storage (Cinder)

Cinder è un servizio di Block Storage per Openstack. È progettato per presentare le **risorse di archiviazione agli utenti finali** che possono essere consumate dall'Openstack Compute Project (Nova). Questo viene fatto attraverso l'uso di un'implementazione di riferimento (LVM) o di driver plugin per altri storage. Cinder **virtualizza la gestione dei dispositivi di archiviazione a blocchi e fornisce agli utenti finali un'API self service** per richiedere e consumare tali risorse senza dover effettivamente sapere dove e come lo storage è realmente distribuito.

Networking (Neutron)

Il servizio Openstack Networking (neutron) consente agli utenti di impostare e definire la connettività di rete e l'indirizzamento nel cloud. Openstack Networking gestisce la creazione e la gestione di un'infrastruttura di rete virtuale, comprese reti, switch, sottoreti e router per i dispositivi gestiti dal servizio di elaborazione Openstack (nova). Possono essere utilizzati anche servizi avanzati come firewall o reti private virtuali (VPN). Tale servizio assicura che la rete non sarà il collo di bottiglia o fattore limitante in un cloud e offre agli utenti una reale gestione self-service anche delle loro configurazioni di rete

Openstack Networking è costituito dal ***neutron-server***, un database per l'archiviazione persistente, e da agenti plug-in, che forniscono altri servizi come l'interfaccia con i meccanismi di rete nativi di Linux, dispositivi esterni, o controller SDN.

Dashboard (Horizon)

Horizon è l'implementazione canonica del servizio Dashboard di Openstack, che fornisce un'interfaccia utente web-based per i servizi Openstack tra cui Nova, Swift e Keystone.

Chapter 2

Edge Computing

2.1 Cenni storici

Le radici dell'edge computing risalgono alla fine degli anni '90, quando Akamai ha introdotto le reti di distribuzione dei contenuti (CDN) al fine di accelarare le prestazioni web. Una rete CDN usa nodi edge vicino agli utenti finali per precaricare e memorizzare in cache contenuti web. Questi nodi possono eseguire anche personalizzazioni dei contenuti (come aggiunta di pubblicità pertinente). Le Content Delivery Networks sono particolarmente utili per i contenuti video, poiché il risparmio di larghezza di banda è notevole. **L'edge computing generalizza ed estende il concetto della CDN sfruttando l'infrastruttura di cloud computing**. Nel 1997, Brian Noble e i suoi colleghi hanno dimostrato per la prima volta il potenziale valore dell'edge computing per il mobile computing. L'emergere del cloud computing a metà degli anni 2000 ha portato il cloud a diventare l'infrastruttura più ovvia da sfruttare da un dispositivo mobile. Oggi, *Siri di Apple e i servizi di riconoscimento vocale di Google scaricano entrambi i calcoli sul cloud* [10]. Sfortunatamente, il consolidamento implica un'ampia separazione tra un dispositivo mobile e il suo data center cloud. "*Ang Li ed i suoi colleghi hanno riferito che il tempo medio di andata e ritorno da 260 punti di osservazioni globali alle istanze ottimali di Amazon Elastic Compute Cloud (EC2) è di 74 ms. A questo deve essere aggiunta la latenza del primo hop wireless. In termini di* ***jitter****, deve essere inclusa la varianza inerente a una rete multihop*" [10].

2.2 What is Edge Computing

La proliferazione di Internet of Things (IoT) e il successo di ricchi servizi cloud hanno contribuito alla nascita di nuovo paradigma di elaborazione, l'edge computing, permettendo di rispondere a requisiti, quali tempo di risposta, limiti di durata della batteria, risparmio sui costi di larghezza di banda, nonché alla sicurezza dei dati e alla privacy [11].
Edge computing significa letteralmente **elaborazione ai margini**, ovvero *elaborazione delle informazioni vicino all'ubicazione fisica dell'utente o alla sorgente dati.* Il termine "edge" si riferisce a ***qualsiasi risorsa di rete e di elaborazione che si trova tra le sorgenti dati e i cloud datacenters***. Ad esempio, uno smartphone è il confine tra l'individuo ed il cloud; un gateway è il confine tra la casa ed il cloud.
Nell'edge computing, il cloud raccoglie i dati dai database esistenti e da dispositivi finali come sensori e telefoni cellulari. I dispositivi agiscono sia come **consumatori di dati** che come **produttori di dati**.

Le richieste tra i dispositivi finali e il cloud sono bidirezionali, come è possibile vedere dalla figura 2.1 [12], invece che solo dai dispositivi finali al cloud, come in passato. I nodi alla periferia della rete eseguono molte attività di elaborazione, tra cui elaborazione dei dati, memorizzazione nella cache, gestione dei dispositivi e protezione della privacy, per ridurre il traffico dai dispositivi al cloud. L'edge computing mostrato in figura 2.1 si riferisce alle ***tecnologie abilitanti che consentono di eseguire il calcolo ai margini della rete***.

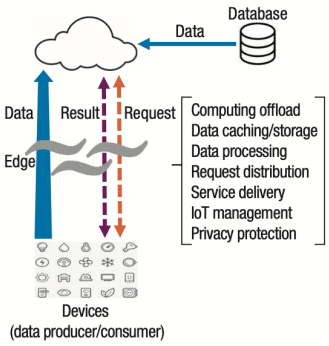


Figure 2.1: Data consumer/producer.

2.3 Architettura edge

> "*L'edge computing avvicina i workloads al luogo in cui vengono creati i dati e vengono intraprese le azioni, riducendo la latenza, diminuendo le richieste di rete, aumentando la privacy delle informazioni personali e sensibili e migliorando la resilienza operativa. L'IDC (International Data Corporation) stima che vedremo crescere il numero di* ***edge devices*** *intelligenti sul mercato fino a 150 miliardi entro il 2025.*" - Rob High, Jr., CTO IBM Edge Computing, IBM Fellow, and Vice President [13].

Un **edge device** è un componente hardware che **controlla il flusso di dati al confine tra due reti**. Il cloud computing e l'Internet of Things (IoT) hanno elevato il ruolo dei dispositivi edge, richiedendo più servizi di intelligenza, elaborazione e intelligenza artificiale nella rete edge. Indipendentemente dalle dimensioni, i dispositivi sono dotati di diversi tipi di sensori: sensori che guidano attuatori, sensori che acquisiscono e inviano informazioni tramite feed audio e video e sensori che trasmettono dati grezzi che richiedono analisi e azione immediata.
Gli elementi fondamentali di un'architettura edge, come riportato in figura 2.2 [13], sono i seguenti:

- **devices**: i dispositivi edge e IoT sono attrezzati per eseguire analisi, applicare regole di intelligenza artificiale e persino archiviare alcuni dati localmente per supportare le operazioni all'edge. I dispositivi possono gestire analisi e inferenze in tempo reale senza il coinvolgimento del server perimetrale o della regione aziendale.

- **edge server o gateway**: i server perimetrali vengono utilizzati per distribuire le applicazioni sui dispositivi. Sono in costante comunicazione con i dispositivi utilizzando *agents* installati su ciascuno dei dispositivi.

- **edge network o micro data center**: le nuove tecnologie di rete hanno portato alla rete edge o al micro data center, che può essere visto come un cloud locale con cui i dispositivi possono comunicare. La rete edge *riduce la distanza che i dati dai dispositivi devono percorrere e quindi diminuisce i problemi di latenza e larghezza di banda soprattutto* ***con l'avvento del 5G***.

- **multicloud ibrido aziendale**: questa regione offre l'archiviazione e la gestione del modello classico a livello aziendale, la gestione dei dispositivi e, in particolare, analisi e dashboard a livello aziendale. Questa regione può essere ospitata nel cloud o in un data center locale.

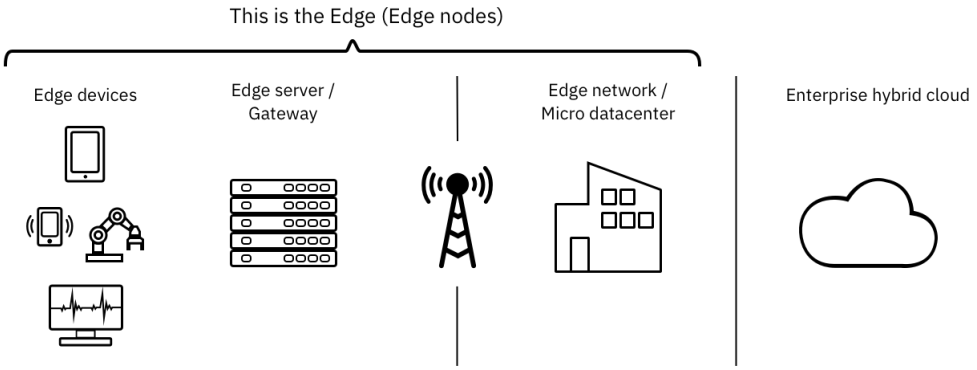


Figure 2.2: Architettura Edge.

2.4 Why Edge Computing

Le applicationi IoT generano enormi quantità di dati dai sensori; i big data vengono poi analizzati per determinare le reazioni ad eventi o per effettuare analisi o statistiche. L'invio di queste moli di dati al cloud richiederebbe una banda di rete eccessivamente larga. Nell'edge computing, *queste grandi quantità di dati possono essere elaborate ai margini della rete*, piuttosto che trasmettere i dati all'infrastruttura cloud centralizzata, comportando anche un maggior consumo di energia. L'edge computing permette di fornire **servizi con una risposta più rapida ed una qualità migliore rispetto al cloud computing** [14].

- **Push from Cloud Services**: posizionare le attività di calcolo in cloud è un modo efficiente per elaborare dati, poiché la potenza di calcolo del cloud surclassa la capacità degli edge devices: tuttavia, mentre la velocità d'elaborazione dei dati è in rapida evoluzione, la larghezza di banda della rete è arrivata ad un punto di arresto. Con la crescente quantità di dati generati all'edge, **la velocità del trasporto dei dati sta diventando un collo di bottiglia nel paradigma del cloud computing**. Ad esempio, in un veicolo autonomo, 1 gigabyte di dati viene generato dall'auto ogni secondo e richiede un'elaborazione in tempo reale affinché il veicolo prenda le decisioni corretta. Se tutti i dati dovessero essere inviati al cloud per l'elaborazione, il tempo di risposta sarebbe troppo lungo; inoltre, sarebbe difficile per la larghezza di banda e l'affidabilità attuali della rete supportare molti veicoli in un'area. In questo caso, i dati devono essere elaborati all'edge per tempi di risposta più brevi.
- **Pull from IoT**: molti tipi di dispositivi elettrici entreranno a far parte dell'IoT e svolgeranno il ruolo **sia di produttori di dati che di consumatori di dati**, come sensori di qualità dell'aria, forni a microonde connessi a Internet. Si deduce che il numero di edge devices aumenterà fino al miliardo in pochi anni. I dati grezzi prodotti da questi dispositivi saranno numerosi ed il cloud computing convenzionale sarebbe troppo inefficiente per gestirli. Pertanto, la maggior parte dei dati prodotti dall'IoT non verrà mai trasmessa al cloud, piuttosto i dati verranno consumati ai margini della rete.

Nota 2.1: Osservazione

Nel cloud computing convenzionale, i produttori di dati generano dati grezzi e li trasferiscono nel cloud, mentre i consumatori di dati inviano una richiesta di consumo dei dati al cloud. Questa struttura non è sufficiente per l'IoT. Innanzitutto, la quantità di dati all'edge è troppo grande, il che comporterà una

larghezza di banda e un utilizzo delle risorse di elaborazione non necessari. In secondo luogo, ***i requisiti di protezione della privacy rappresenteranno un ostacolo per il cloud computing nell'IoT***. Infine, scaricare alcune attività di elaborazione sull'edge potrebbe essere più **efficiente dal punto di vista energetico**.

- **Change From Data Consumer to Producer**: nel paradigma del cloud computing, gli end devices in genere sono consumatori di dati, come quando guardiamo un video di YouTube sul nostro smartphone. Tuttavia, *le persone ora producono anche dati dai propri dispositivi mobili.* Questo passaggio da consumatore di dati a produttore/consumatore di dati richiede più posizionamento delle funzioni ai margini, come quando gli utenti scattano foto o registrano video.

2.5 Tecniche per implementare l'edge computing

2.5.1 Cloudlet

Un cloudlet è un data center cloud su piccola scala che consente di raggiungere una ***end-to-end responsiveness*** tra un dispositivo mobile e il cloud associato. E' un sistema affidabile e ricco di risorse o di un gruppo di sistemi ben collegati tra loro e Internet [14]. Come

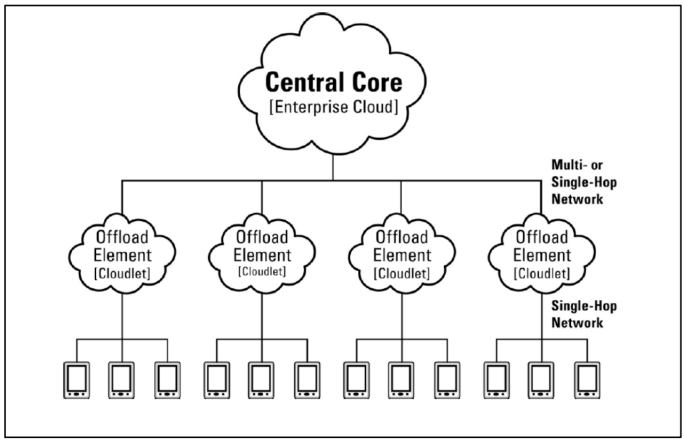


Figure 2.3: Architettura Cloudlet.

si può vedere dalla figura 2.3, **il cloudlet è il livello intermedio in un'architettura gerarchica a tre livelli** (three-tiers) per ottenere buoni tempi di risposta. Esso si pone tra i dispositivi mobili e i servizi in Cloud centralizzati, all'edge della rete Internet, alla quale è connesso tramite un link veloce e affidabile. Le entità decentralizzate risiedono ad un solo hop wireless di distanza dai dispositivi finali associati. Il Cloudlet, come la definizione suggerisce, permette di costruire un **data-center in a box**, ovvero di ricreare l'ambiente di invocazione dei servizi Cloud tramite risorse presenti sulla rete locale, che rappresenta l'edge della core network che collega al Cloud remoto.
Le caratteristiche principali dell'architettura seguenti:

- **soft state**: i servizi cloudlet sono intesi come completamente auto gestiti, per questo si limiteranno a contenere uno stato temporaneo, proveniente dai dispositivi o dal cloud

in caso di cache dei contenuti; lo stato significativo viene dirottato direttamente sui server Cloud di supporto.

- **powerful, well-connected and safe**: dovendo ricevere un notevole carico computazionale dai dispositivi mobili pertinenti alla propria area, le centraline cloudlet devono essere dotate di una tecnologia adatta all'esecuzione di tali computazioni resource-intensive, inclusa un'adeguata alimentazione. Tipicamente sono connesse ad un canale Internet, tramite il quale accedono al cloud, via cavo con tecnologie stabili e veloci.
- **close at hand**: la prossimità logica dei clienti alle macchine che contattano si traduce in una connessione a bassa latenza, con alta banda disponibile e con la possibilità di fare delle stime e previsioni precise delle sue performance.
- **builds on standard cloud technology**: si adottano i classici meccanismi che vengono usati nelle piattaforme Cloud, ovvero la gestione dei task provenienti dai dispositivi mobili viene demandata alle macchine virtuali, che sono a loro volta amministrate da classiche infrastrutture Cloud come *Amazon EC2* oppure *Openstack*.

Nota 2.2: Differenza cloud e cloudlet

Un cloudlet è molto più agile nel suo provisioning perché i dispositivi edge associati sono altamente dinamici e mobili. Inoltre, è necessario un buon **handoff** per supportare la mobilità senza interruzioni dei dispositivi periferici. Infine, per connettersi o comunicare con il cloud, *un nodo mobile si connette prima ad un cloudlet vicino e poi al cloud per completare la sua comunicazione.* **I cloudlet riducono i dati sulla rete e la latenza di risposta ai nodi mobili**.

Handoff

L'handoff è il meccanismo, trasparente alle applicazioni, che avvicina il servizio generico in corso, ad una centralina più conveniente di quella attuale, dalla quale l'utente si sta allontanando, per evitare un eccessivo degrado delle prestazioni dell'applicazione in esecuzione.

2.5.2 Mobile Edge Computing

Il concetto di MEC è stato definito dall'**European Telecommunications Standards Institute** (ETSI) come una nuova tecnologia che "*fornisce un ambiente di servizi IT e capacità di cloud computing ai margini della rete mobile, all'interno della rete di accesso radio (RAN) e in stretta vicinanza agli abbonati di telefonia mobile*" [15].
Si comprende che l'idea alla base del modello è migrare le possibilità offerte dal cloud sull'edge delle reti mobili, rappresentato dalla **Radio Access Network (RAN)**, quella parte dell'infrastruttura di rete che connette i dispositivi finali alla rete Internet esterna. In altre parole, il MEC può essere visto come un server Cloud, installato in prossimità dei dispositivi mobili, in una rete ad accesso radio, che effettua delle operazioni specifiche che non è possibile portare a termine con una tradizionale infrastruttura Cloud.

In figura 2.4 è riportata l'architettura MEC. Diversi tipi di dispositivi mobili e sensori (i big data e le piattaforme social) sono collegati alla rete principale (ovvero Internet mobile) attraverso la rete edge, RAN e MEC, e la rete principale è connessa alla rete cloud privata. Con l'evoluzione della RAN basata su LTE, è diventato più fattibile distribuire MEC che

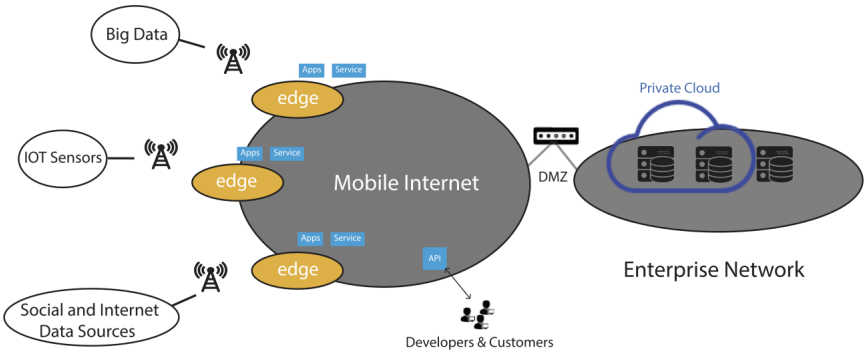


Figure 2.4: Architettura MEC.

avvicina i servizi cloud agli abbonati mobili. Pertanto, *ciascuna piattaforma edge rappresenta un edge cloud con applicazioni e servizi specifici per l'ambiente mobile di destinazione.* **MEC costituisce server geo-distribuiti o server virtuali con servizi IT integrati**. Questi server sono implementati localmente presso le sedi degli utenti mobili, ad esempio, parchi, terminali di autobus e centri commerciali [16].
MEC può utilizzare elementi di rete cellulare, come la stazione base, il punto di accesso WiFi o il punto di accesso femto (la stazione base cellulare a bassa potenza). Il mobile edge computing può essere dispiegato in una posizione fissa, ad esempio, in un centro commerciale o su un dispositivo mobile situato in qualsiasi oggetto in movimento, ad esempio, auto o autobus. MEC può essere distribuito in una stazione base LTE o in un sito di aggregazione cellulare multitecnologia (3G/LTE) [17]. Per riassumere, la proposta di valore chiave di MEC è che offre il cloud computing spingendo le risorse cloud, come elaborazione, rete e archiviazione ai margini della rete mobile, al fine di soddisfare i requisiti delle applicazioni che sono affamati di calcolo (ad esempio applicazioni di giochi), sensibile alla latenza (ad esempio, applicazioni AR) e che richiede una larghezza di banda elevata (ad esempio analisi dei big data mobili). L'arricchimento delle funzioni di rete risulta trasparente all'utenza finale che ne beneficia assistendo ad un aumento della percezione positiva (**QoE, Quality of Experience**) sui servizi utilizzati. L'accesso ai contenuti e alle applicazioni è accelerato; in quanto esse risiedono in una parte della rete prossima all'utenza che massimizza la velocità dell'interazione.

2.6 Security

Una grande sfida per creare un ecosistema in cui tutti gli attori (utenti finali, fornitori di servizi, fornitori di infrastrutture) beneficiano dei servizi forniti dai paradigmi edge è la sicurezza. Per prima cosa, teconolgie abilitanti della maggior parte dei paradigmi periferici sono reti wirelss, sistemi distribuiti, peer-to-peer e piattaforme di virtualizzazione. **E' necessario proteggere non solo questi blocchi, ma anche orchestrare i meccanismi di sicurezza**, consentendone integrazione ed interoperabilità. ***Assicurando la sicurezza di tutte le tecnologie abilitanti, non garantiamo la sicurezza dell'intero sistema.*** Una volta che le capacità simili al cloud computing vengono portate ai margini della rete, sorgono nuove situazioni (ad es. collaborazione tra data center periferici eterogenei, migrazione di servizi su scala locale e globale) la cui sicurezza non è stata ampiamente studiata [18]. Ad esempio, i meccanismi di sicurezza dovrebbero essere il più autonomi possibile e non dipendere dalla continua esistenza di un'infrastruttura centralizzata. Ci sono due ragioni principali

per questo: non solo ci saranno situazioni (ad es. attacchi dannosi, connettività intermittente, applicazioni distribuite) in cui non è disponibile un sistema di controllo centralizzato, ma è anche necessario tenere in considerazione la latenza del meccanismi di sicurezza. Un altro esempio sono i limiti tecnologici degli elementi dell'infrastruttura. Ad esempio, alcuni data center edge potrebbero essere composti da microserver (ad es. Raspberry Pi) privi dei meccanismi di protezione hardware dei server commodity] o includere dispositivi edge legacy con connettività limitata, il che limita i protocolli di autenticazione che possono essere implementati [19]. Inoltre, i meccanismi di sicurezza devono considerare l'esistenza di dispositivi mobili, che possono utilizzare i data center periferici sempre e ovunque [18].
Bisogna anche tener presente che **l'intero sistema eredita anche le minacce alla sicurezza che sono presenti nei loro elementi costitutivi e nello scenario applicativo** Ad esempio, nell'Internet of Things, non solo abbiamo bisogno di combinare e proteggere più livelli di tecnologie (dalla rete al mobile al cloud), ma dobbiamo proteggere una superfice di attacco generata dalla connettività e dall'accessibilità globale in un ecosistema eterogeneo. Non per ultimo, va considerato l'impatto di un eventuale attacco di sicurezza.
L'intero ecosistema di edge computing non è mai controllato da un singolo proprietario; i data center edge sono in grado di fornire servizi senza dipendere continuamente da un'infrastruttura centrale. Tutte le risorse rilevanti, compresa l'infrastruttura di rete, l'infrastruttura di servizio, l'infrastruttura di virtualizzazione e i dispositivi degli utenti, **sono controllati non da una singola entità ma da vari attori (anche utenti finali) che hanno bisogno di collaborare tra loro**. Una conseguenza di questa situazione è che ***ogni elemento dell'infrastruttura può essere preso di mira o sovvertito in qualsiasi momento***.
In realtà, uno dei principi di base di questi paradigmi è che le funzionalità sono fornite in stretta prossimità degli utenti finali. Di conseguenza, un data center periferico fornirà servizi principalmente a entità locali (ad esempio utenti mobili situati nelle vicinanze, entità all'interno di un edificio). Questa particolarità dei paradigmi edge è un'arma a doppio taglio: da un lato, *limita l'impatto di un attacco all'ambiente locale*; d'altra parte, s*e un avversario può controllare un data center periferico, potrebbe essere in grado di controllare quasi tutti i servizi forniti in quella posizione geografica.* In altre parole, sottolineaiamo la mancanza di un perimetro globale nell'edge computing influenza la sicurezza di questo paradigma.
Un'altra conseguenza di ciò è **la natura dei diversi profili di attaccante che prenderanno di mira i paradigmi edge**. Anche se esisteranno i tradizionali aggressori 'esterni' (ovvero avversari che non controllano alcun elemento dell'intera infrastruttura), esisteranno molti avversari che controlleranno uno o più elementi dell'infrastruttura: dispositivi utente, macchine virtuali, server, sezioni di rete, anche interi data center edge. Questa situazione è simile all'attuale Internet, in cui gli avversari malintenzionati possono assumere il controllo degli elementi esistenti o implementare i propri. *Questi avversari sono sia "interni" che "esterni", poiché controllano una parte dell'infrastruttura ma non le altre.*

2.6.1 Meccanismi di sicurezza

Per creare un livello di difesa efficace contro le diverse minacce, *occorre implementare vari tipi di servizi e meccanismi di sicurezza.* Tutti i meccanismi di sicurezza devono tenere conto di vari requisiti e vincoli comuni, come ridurre il più possibile la latenza delle loro operazioni, supportare dispositivi mobili e altre entità mobili (macchine virtuali), ottenere l'interoperabilità tecnica, funzionale e semantica, gestire i limiti delle tecnologie esistenti e fornire supporto per operazioni disconnesse [18].

1. **Identità e autenticazione**: in tutti i paradigmi edge, ci sono più attori (utenti finali, fornitori di servizi, fornitori di infrastrutture), servizi (macchine virtuali, container) e infrastrutture (dispositivi utente, data center edge, infrastrutture core) che interagiscono in un ecosistema in cui coesistono domini con livelli di fiducia differenti. Questa situazione comporta numerose sfide, poiché **non solo dobbiamo assegnare**

un'identità a ogni entità, ma dobbiamo anche consentire a tutte le entità di autenticarsi reciprocamente. Si rendono allora necessari **meccanismi di federazione delle identità, sistemi di autenticazione inter-realm**. A causa di vari requisiti (latenza, disponibilità di un server centrale), è anche auspicabile che un'entità possa fornire una prova della propria identità senza contattare un server centrale. In alcuni casi, come cloudlet personali, andranno studiati meccanismi di autenticazione distribuita.

2. **Sistemi di controllo accessi**: un'infrastruttura di autorizzazione è fondamentale per i paradigmi edge ai fini del controllo delle credenziali delle varie entità per autorizzare le loro richieste di eseguire determinate azioni. Se non sono presenti meccanismi di autorizzazione, chiunque non disponga di credenziali adeguate può utilizzare in modo improprio le risorse dell'infrastruttura. Ad esempio, una terza entità malevola potrebbe impersonare l'amministratore di sistema e controllare i servizi dell'infrastruttura. E' fondamentale implementare **un'infrastruttura di autorizzazione in ogni dominio di fiducia, in modo da consentire ai proprietari di tali domini di diffondere, archiviare e applicare le proprie politiche di sicurezza**. Queste infrastrutture dovrebbero essere in grado di elaborare le credenziali di una qualsiasi entità, ma anche tenere conto di fattori, come l'ubicazione geografica, la proprietà delle risorse e tanto altro nella definizione delle policy di autenticazione.

3. **Protocollo e sicurezza di rete**: se l'infrastruttura di rete non è protetta, l'intero ecosistema di servizi sarà minacciato da avversari dannosi interni ed esterni. È quindi necessario **proteggere le tecnologie ed i protocolli di comunicazione utilizzati dai paradigmi edge**. Ad esempio, esistono varie tecnologie di comunicazione wireless (ad esempio Wi-Fi, 802.15.4, 5G, Sigfox, LoRa) che potrebbero essere utilizzate per servire i clienti locali. Pertanto, i data center periferici e i loro amministratori devono comprendere e utilizzare i protocolli di sicurezza e le estensioni implementate da tali tecnologie [18]. Per cominciare, è necessario configurare adeguatamente i diversi elementi dell'infrastruttura di rete. Tutti questi elementi verranno distribuiti in diverse posizioni geografiche, che saranno gestite da diversi amministratori. Inoltre, ci saranno situazioni in cui entità che appartengono a diversi domini di fiducia (ad es. data center edge di diversi proprietari di infrastrutture) interagiranno tra loro. In questo scenario molto eterogeneo, dobbiamo stabilire **una connessione sicura tra entità che potrebbero anche utilizzare diverse tecnologie di comunicazione**.

4. **Gestione della fiducia**: il concetto di fiducia va oltre l'idea di "non sapere con chi sto interagendo", che è per lo più risolto implementando meccanismi di autenticazione e stabilendo relazioni di fiducia tra i diversi domini. Dobbiamo infatti tener conto del termine di incertezza dato dal non sapere come si comporterà il partner con cui interagiamo. Gli utenti possono avere diversi fornitori di servizi disponibili nelle loro vicinanze, i fornitori di servizi possono scegliere tra molti fornitori di infrastrutture e così via. Tuttavia, tali peer potrebbero non soddisfare le nostre aspettative: la latenza del servizio potrebbe essere elevata, il tasso di rilevamento delle anomalie potrebbe essere basso o i dati potrebbero essere imprecisi o addirittura essere maliziosi. Sono quindi necessarie **infrastrutture di gestione della fiducia**, che comportano numerosi vantaggi, dal miglioramento dei processi decisionali di tutte le entità al miglioramento della gestione dei dati personali (es. riduzione della granularità delle informazioni trasmissione che viene trasmessa a entità di bassa reputazione). **Tutte le infrastrutture di gestione del trust dovrebbero essere in grado di scambiare tra loro informazioni di trust compatibili, anche se situate in domini di fiducia differenti**.

5. **Intrusion detection systems**: senza adeguati meccanismi di rilevamento e preven-

zione delle intrusioni, qualsiasi attacco riuscito non verrà rilevato, minando lentamente la funzionalità dell'intera infrastruttura. È fondamentale **garantire che l'intera infrastruttura sia coperta da tali meccanismi di difesa**. Abbiamo già detto che l'impatto della maggior parte degli attacchi è solitamente limitato a un ambiente locale. Pertanto, le infrastrutture locali, come i data center edge, possono essere incaricate di monitorare tutti i loro elementi - connessioni di rete, macchine virtuali, ecc. - e l'ambiente circostante. Tuttavia, le challenges legate alla gestione di una rete interconnessa di meccanismi di rilevamento e prevenzione in un'infrastruttura eterogenea, decentralizzata e distribuita sono numerose. Gli attacchi specifici che possono essere lanciati contro i paradigmi edge devono essere compresi. Se viene utilizzato un database di attacchi (ad es. signature based IDS), deve essere aggiornato e protetto in ogni momento.
Tutti i meccanismi di difesa, indipendentemente dalla loro ubicazione, devono essere in grado di scambiarsi informazioni tra loro in un formato interoperabile. Tali informazioni dovrebbero essere costantemente disponibili per *rilevare minacce più persistenti*. Infine, i meccanismi di difesa devono comportarsi nel modo più autonomo possibile, al fine di *ridurre l'overhead di manutenzione e migliorare l'usabilità dell'infrastruttura di sicurezza.*

6. **Privacy**: Oltre ad avversari maligni, è anche possibile trovare avversari "onesti ma curiosi". Questi avversari sono solitamente entità autorizzate il cui obiettivo secondario è conoscere meglio le entità che utilizzano i loro servizi. Questa conoscenza può quindi essere utilizzata in vari modi: **profilazione dell'utilizzo, rilevamento della posizione, divulgazione di informazioni sensibili**. Ciò rappresentano una **minaccia per la privacy degli utenti**. Tutti i paradigmi edge sono ecosistemi aperti, in cui più domini di fiducia sono controllati da diversi proprietari di infrastrutture. Questo vuol dire che **non è possibile sapere in anticipo se un determinato fornitore di servizi è abbastanza affidabile da rispettare la privacy degli utenti**. E' essenziale fornire agli utenti vari meccanismi efficienti che non solo proteggano le loro informazioni, ma consentano anche agli utenti di interrogarle ed elaborarle. Inoltre, gli utenti hanno il diritto di proteggere la propria identità e i propri dati personali, ma hanno anche la responsabilità di comportarsi in modo onesto. Se un utente si comporta male, dovrebbe essere possibile utilizzare alcuni meccanismi per identificare il malintenzionato.

7. **Virtualizzazione**: l'infrastruttura di virtualizzazione è uno degli elementi centrali dei paradigmi edge, quindi è fondamentale proteggerla implementando meccanismi di sicurezza in tutti i data center edge. Senza questi meccanismi, non solo gli insider malintenzionati possono assumere il controllo delle macchine virtuali distribuite dagli utenti, ma anche le macchine virtuali dannose possono manipolare i servizi dei data center periferici. **Contromisure possibili sono policy di isolamento, protezione avanzata dell'hypervisor, separazione di ruoli e VM, astrazioni di rete** e molte altre.

8. **Tolleranza ai guasti e resilienza**: nessun paradgima sarà mai immune alle minacce al 100 %. Errori di configurazione, vulnerabilità, software obsoleto e altre debolezze consentono ai possibili attaccanti di disabilitare o assumere il controllo di elementi dell'intera infrastruttura. È necessario integrare vari meccanismi e strategie (operazioni ridondanti, capacità di failover, meccanismi di ripristino di emergenza [18]) che consentano all'infrastruttura del servizio di continuare il funzionamento previsto. Tuttavia, l'implementazione dei data center periferici ai margini della rete è un'arma a doppio taglio. *Da un lato, i meccanismi di protezione possono trarre vantaggio dal fatto che diversi fornitori di infrastrutture potrebbero essere disponibili nello stesso luogo.*

D'altra parte, poiché i servizi sono forniti a livello locale, potrebbero esserci situazioni in cui non è disponibile alcuna sostituzione.

9. **Forensics**: indipendentemente dai meccanismi di protezione messi in atto, i paradigmi edge possono essere attaccati. Questi attacchi lasceranno dietro di sé alcune prove che possono essere utilizzate per rivelare informazioni sull'aggressore e sui suoi metodi. L'obiettivo della scientifica è **identificare, recuperare e preservare queste prove**, in modo che possano essere presentate in tribunale. La gestione delle prove in paradigmi edge è una questione molto complessa, ancora una volta causa dell'esistenza di più attori, infrastrutture, tecnologie e scenari.

Chapter 3

Fog Computing

Per Fog Computing si intende un paradigma per l'elaborazione dati che calca alcuni dei principi del paradigma di Cloud Computing rispondendo tuttavia ad alcune limitazioni che quest'ultimo presenta.
Determinare una definizione univoca per un concetto così sfumato non è un compito semplice. Di seguito viene riportata la definizione fornita dalla *Open Fog Consortium*, ossia un consorzio di società che sostiene e propone il Fog Computing come paradigma principe per il superamento dei limiti posti dal Cloud Computing [20]:

Fog computing is a horizontal, system-level architecture that distributes computing, storage, control and networking functions closer to the users along a cloud-to-thing continuum

Similmente a quanto appena esposto, altri riferimenti scientifici si riferiscono al paradigma in modo simile. Secondo [21]:

Fog is an architecture that distributes computation, communication, control and storage closer to the end users along the cloud-to-things continuum

Gli autori di [22] si riferiscono al Fog Computing come a una speciale implementazione dell'Edge Computing:

The Fog Computing implementation is a decentralized Computing infrastructure based on Fog Computing Nodes placed at any point of the architecture between the end devices and the cloud.

Una definizione più operativa la si ritrova in [23]:

Fog computing can enhance the CoT paradigm by providing small platforms located at the network edges in smart cities. These platforms can operate cloud-like services to support different IoT applications.

Analizzando le varie definizioni, è possibile trovare dei punti in comune. Tali punti sono:

- *Decentralizzazione dei servizi*
  Per questo punto, il paradigma si caratterizza per offrire dei servizi dislocati su nodi fisicamente disaccoppiati e geograficamente distribuiti;

- *Architettura gerarchica*
  Per *cloud-to-things continuum* si intende lo spazio che intercede il dispositivo da cui è originata la richiesta di un servizio e l'ambiente cloud capace di servire tale richiesta facendo uso di ingenti capacità di calcolo e immagazzinamento dati. Il paradigma Fog Computing serve tali servizi organizzando le sorgenti di calcolo e immagazzinamento dati in maniera gerarchica, definendo punti di servizio *vicini* alla sorgente della richiesta, fino ad arrivare, in maniera graduale, a punti di servizio sempre più *lontani*, culminando, infine, con un ambiente Cloud.

Nelle seguenti sezioni si approfondiranno le *motivazioni* che hanno portato allo sviluppo di questo paradigma, l'*architettura* del paradigma e le *sfide* che devono essere indirizzate per portare a un'immersione ancora più pervasiva del paradigma nelle più disparate applicazioni IoT, come applicazioni Smart City, Smart Home, E-Health, ecc.

3.1 Motivazione

Le motivazioni che hanno spinto all'adozione di questo paradigma derivano dalle limitazioni che un ambiente Cloud possiede nei confronti di certe tipologie di applicazioni.
Puliafito et al. [20] elencano i seguenti limiti nell'ambito del Cloud of Things, ossia del paradigma di Cloud Computing applicato alle applicazioni IoT:

- *Latenza*
  La più banale limitazione nell'uso del Cloud per effettuare elaborazione dati riguarda la latenza con cui le richieste dei servizi vengono servite. Ciò è dovuto al fatto che il Cloud Computing è un paradigma di computazione centralizzato: le risorse di calcolo e immagazzinamento dati sono localizzate in punti geografici precisi e, spesso, lontani abbastanza da portare ad avere latenze importanti, nell'ordine delle centinaia di millisecondi;

- *Consumo di banda*
  Nel Cloud of Things, i dispositivi IoT sono direttamente connessi a Internet, usufruendo quindi dei suoi servizi, tra cui chiaramente la possibilità di interagire con server Cloud. Il più delle volte, i servizi richiesti necessitano di trasmettere dati, spesso grezzi, da far elaborare in Cloud. I dati scambiati dall'ingente quantità di dispositivi connessi a Internet porta ad avere un consumo di banda non trascurabile, che può portare a un degrado della qualità del servizio;

- *Sicurezza*
  I dispositivi comunicano i propri dati a dei server Cloud tramite Internet. Internet, dal suo canto, non offre, così com'è, controlli di sicurezza che possano assicurare *confidenzialità*, *integrità* e *disponibilità*. Essendo i dispositivi IoT spesso privi delle risorse necessarie ad attuare dei controlli di sicurezza, quali ad esempio l'applicazione di tecniche crittografiche per la trasmissione e ricezione crittata di dati, è chiaro che la sicurezza è minata;

- *Context Awareness*
  Per *Context Awareness* ci si riferisce a tutte le informazioni che possano dare, a chi elargisce servizi, indicazioni su come migliorare la qualità del servizio. Dato l'alto disaccoppiamento tra i Data Center Cloud e i dispositivi che fanno uso dei loro servizi, questa proprietà è del tutto ignorata nel Cloud of Things;

- *Ambienti ostili*
  Talvolta la connettività dei dispositivi IoT potrebbe essere compromessa, impedendo a questi ultimi di comunicare con il Cloud.

In aggiunta a queste osservazioni, che sono più o meno quelle generalmente citate dai vari articoli scientifici presi in considerazione, ci sono anche le seguenti:

- *Eterogeneità*
  I dispositivi che comunicano con il Cloud sono tra i più disparati, sia in termini hardware che software. Ciò significa che per comunicare con il Cloud, i dispositivi devono essere in grado di supportare i protocolli richiesti da quest'ultimo, oppure dev'essere il Cloud stesso ad ospitare un'ampia gamma di opzioni per la comunicazione.

L'eterogeneità dei dispositivi IoT rende difficoltosa la comunicazione: adattare i protocolli supportati dai dispositivi a quelli richiesti dal Cloud non è un requisito semplice da rispettare;

- *Monitoraggio*
  I dispositivi IoT potrebbero essere poco affidabili, o essere soggetti a perturbazioni fisiche che comprometterebbero il loro funzionamento. Monitorare i dispositivi IoT può tornare utile in contesti applicativi in cui bisogna assicurarsi che tali dispositivi funzionino così come previsto. Dato l'alto disaccoppiamento tra Cloud e dispositivi IoT, questo requisito è anch'esso difficile da rispettare.

Queste osservazioni chiedono a gran voce delle soluzioni adatte per i nuovi contesti applicativi d'interesse anche aziendale. Tra queste soluzioni, spicca chiaramente il Fog Computing, del quale, nella prossima sezione, ne verrà esplorata l'architettura così come definita nella letteratura scientifica.

3.2 Architettura

Quando si parla di architettura Fog, spesso non è chiaro, dalla letteratura scientifica, a cosa ci si riferisca. Il problema di fondo sta nel fatto che ci sono tanti ambiti nel mondo ICT che sono interessati allo sviluppo di questo paradigma. [24] elenca una lista dei campi interessati allo sviluppo del paradigma del Fog Computing:

- Networking;
- Ingegneria del software;
- Sicurezza;
- Ingegneria dei sistemi.

Il risultato è una definizione convoluta di architettura, che spesse volte cela un'esplicita specifica di quale aspetto del Fog Computing si sta analizzando. In effetti, tutti gli articoli scientifici analizzano, in qualche misura, l'architettura del Fog Computing. Tuttavia, non sempre viene specificata quale *vista architetturale* viene presa in considerazione. Prendendo come esempio l'ingegneria del software, è chiaro che nello sviluppo di un sistema software ci sono diverse viste che bisogna considerare quando si cerca di 'catturarne' l'architettura. Esistono tante tassonomie che ordinano le varie viste che si possono fornire per un'architettura; [24] considera la seguente:

- *Logical view*;
- *Process view*;
- *Physical view*;
- *Development view.*

La miglior cosa sarebbe quella di fornire un framework che consenta di inquadrare che *vista* del Fog Computing i numerosi riferimenti scientifici analizzano nel discutere di aspetti inerenti a questo paradigma. [24] cerca di definire questo framework, definendo, per il Fog Computing, tre dimensioni/viste:

- *Device dimension*
  In questa dimensione ci si focalizza sui differenti tipi di nodi schierabili nell'ambito del Fog Computing;

- *System dimension*
  In questa dimensione ci si focalizza sullo stack tecnologico che descrive il Fog Computing, costituito di livelli hardware e software, inclusivi dei meccanismi di *virtualizzazione*, dell'uso di *middleware* e della *logica applicativa* del sistema schierato;

- *Functionality dimension*
  In questa dimensione ci si focalizza sulle funzionalità offerte dall'infrastruttura Fog, spesso astratte sotto forma di servizi che verranno logicamente organizzati nei livelli individuati dalla System dimension e fisicamente schierati sui dispositivi individuati dalla Device dimension.

Nell'articolo [24] queste tre dimensioni vengono anche graficamente rappresentate in Figura 3.1. Il modo in cui le dimensioni sono rappresentate è significativo: tali dimensioni sono tra

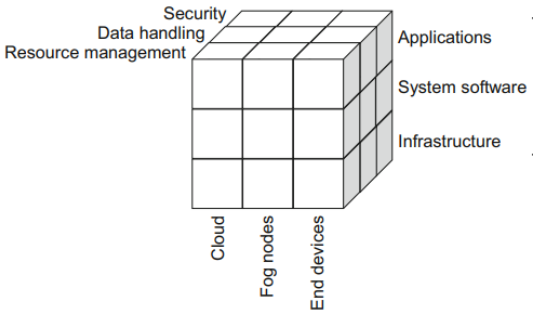


Figure 3.1: Dimensioni dell'architettura Fog

loro ortogonali. Con ciò si intende che è possibile lavorare isolatamente su ciascuna dimensione senza preoccuparsi delle altre. Tuttavia, è anche possibile catturare insieme alcune dimensioni. Si possono, ad esempio, combinare le viste *Device dimension* e *Functionality dimension*. Ancora, più specifico, si potrebbe considerare un solo valore della *Device dimension*, come ad esempio i nodi fog, combinato all'intera *Functionality dimension*. Nell'ultimo caso, graficamente si otterrebbe la situazione in Figura 3.2.

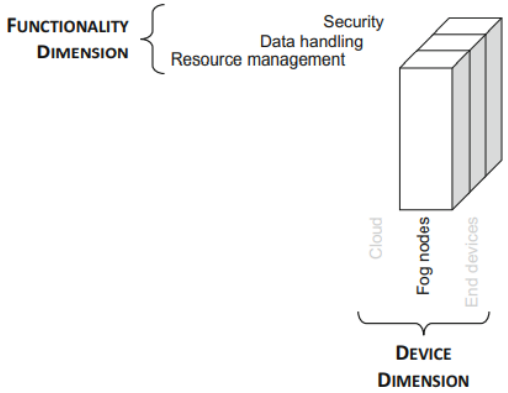


Figure 3.2: Combinazione di dimensioni dell'architettura Fog

A questo punto, è possibile classificare le viste catturate dalla moltitudine di riferimenti scientifici.
Nel contesto Smart City, [23] mostra una vista architetturale di un'applicazione basata sul paradigma del Fog Computing che fa uso di un middleware. La vista fornita da questo riferimento si incastra nella *System dimension*, fornendo un'organizzazione ad alto livello dei servizi nei livelli rappresentati in Figura 3.3.

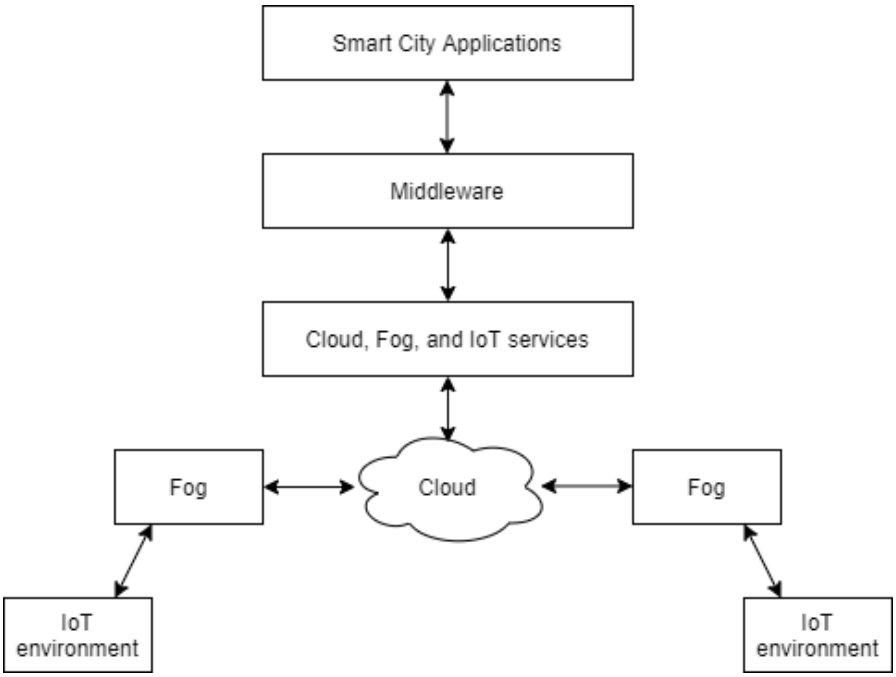


Figure 3.3: Architettura a livelli recuperata da [23]

Leggere quest'architettura non è difficile: Le applicazioni IoT forniscono servizi di alto livello agli *end-users*. In questo articolo, applicazioni Smart City sono considerate. Esse offrono servizi come:

- Monitoraggio di parametri ambientali nei quartieri della città;
- Monitoraggio del traffico;
- Segnalazione di situazioni di pericolo per la sicurezza delle persone.

Questi servizi vanno intesi come servizi di alto livello incapsulati nel livello *Smart City Applications*.
Il livello *Middleware* incapsula tutti i servizi che mediano tra l'infrastruttura software e l'infrastruttura hardware capace di realizzare servizi di alto livello usabili dagli utenti.
Il livello *Cloud, Fog and IoT services* è composto di tutti i servizi schierati sui vari nodi Cloud, Fog e IoT che implementano le richieste di alto livello sottomesse dall'applicazione e mediate dal *Middleware*.

[20] Presenta una vista dell'architettura del Fog Computing che si incastra nelle *Device* e *System* dimensions, raffigurata in Figura 3.4
Per avere una comprensione più approfondita dell'architettura del Fog Computing è necessario guardare ai vari componenti che caratterizzano questo paradigma, più che all'architettura a livelli che, d'altronde, caratterizza la stragrande maggioranza dei sistemi informatici, e quindi non aggiunge più di quanto non si possa già intuire. Abbiamo, nello specifico:

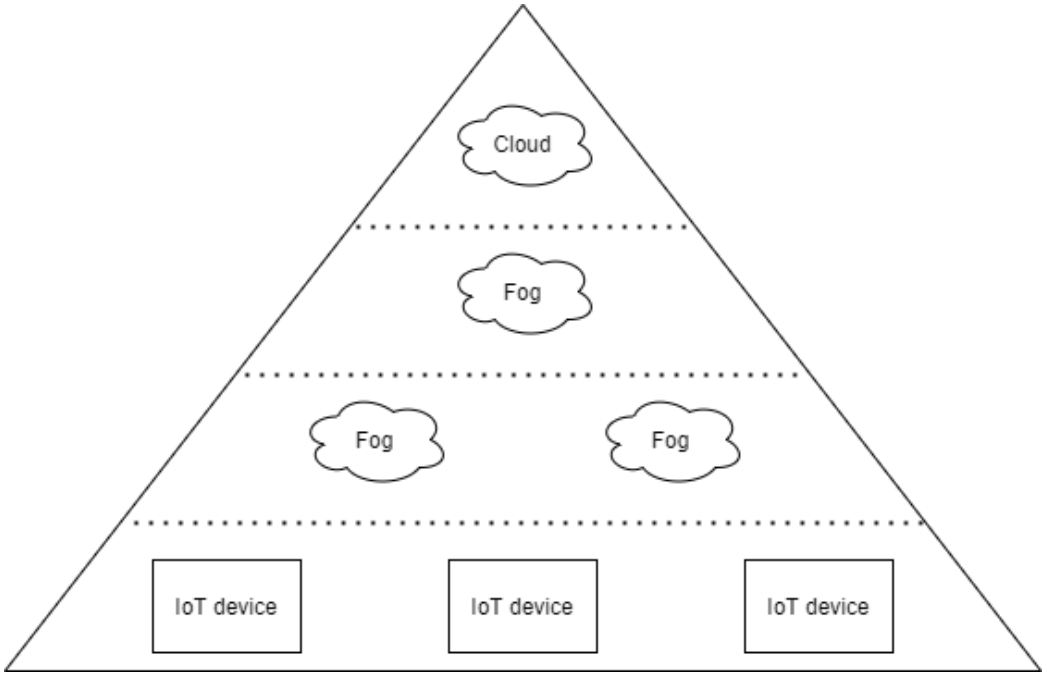


Figure 3.4: Vista gerarchica del Fog Computing presentata da [20]

- *Edge Node*
  Gli Edge Node sono i nodi schierati all'estremità della rete e sono spesso fortemente accoppiati con sensori e attuatori. Possiedono limitate capacità di calcolo, immagazzinamento dati e networking. Il loro scopo è quello di rispondere alle richieste più critiche dal punto di vista della latenza e della connettività, agendo talvolta da *gateway* verso il mondo esterno;

- *Fog Node*
  I Fog Node sono i nodi che sono spesso schierati nelle posizioni intermedie nella gerarchia. A differenza degli Edge Node, le loro capacità di calcolo, immagazzinamento dati e networking sono nettamente superiori, consentendo l'hosting di servizi raffinati che consentono differenti possibilità, tra cui:

  - Virtualizzazione (mediante macchine virtuali o container) e istanziazione di servizi a disposizione per i client;
  - Supporto alla *multi-tenancy*;
  - Coordinazione e load balancing del lavoro con altri Fog Node;
  - Mediazione tra Edge e Cloud;
  - Integrazione di vari dispositivi Edge tramite la conversione di protocolli di networking.

- *Cloud Node*
  I Cloud Node sono i nodi Cloud già precedentemente esplorati nella presente dispensa. Essi comunicano con i Fog Node di alto livello schierati in prossimità del Cloud. L'intervento del Cloud si rende necessario qualora i layer Fog non abbiano avuto sufficienti risorse o non siano stati in grado di coordinarsi per offrire i servizi richiesti dagli utenti dell'applicazione.

La lettura di queste architetture porta alla seguente conclusione: **il paradigma di Fog Computing è schierato per fornire servizi agli utenti di applicazioni IoT nei rispetti dei requisiti che altrimenti non sarebbe possibile rispettare usufruendo unicamente del Cloud Computing.**

3.2.1 Piattaforme

Così come per il Cloud Computing molte piattaforme sono emerse per la gestione dei vari aspetti del modello (aspetti coperti nella sezione inerente al Cloud Computing), così anche per il Fog Computing la ricerca ha portato allo sviluppo di molte piattaforme dedite alla gestione dei Fog Node schierati in un'applicazione IoT che faccia uso del paradigma di Fog Computing.
Il termine 'piattaforma' è molto generico: Puliafito et al. [20] hanno scandito efficacemente il significato di questo termine nell'ambito del Fog Computing, differenziando tre tipi di piattaforme:

- *Piattaforme software*
  Una piattaforma software è definita come:

  A software environment providing at least the basic functionalities and mechanisms that are necessary for the deployment and execution of IoT applications over a Fog infrastructure

  Queste piattaforme forniscono l'infrastruttura software necessaria a gestire i vari Fog Node schierati nel sistema. Esse sono spesso costituite da due componenti software principali:

  - Un componente software *manager* centralizzato e localizzato nel Cloud capace di gestire i Fog Node schierati nel sistema, facendo in modo di:
    * Registrare i Fog Node;
    * Virtualizzare le risorse dei Fog Node (tramite containerizzazione o macchine virtuali);
    * Allocare i servizi nei Fog Node.
  - Un componente software *cablato nei Fog Node*, capace di interloquire con altri Fog Node e con il componente software manager. Questo componente può essere un sistema operativo, oppure uno strato software *on-top* rispetto all'infrastruttura software del Fog Node.

  Esempi di piattaforme software sono *Nebbiolo* e *Stack4Things*.
  L'architettura di Nebbiolo [25] è quella raffigurata in Figura 3.5. Come si può notare, il componente *fogSM* è il componente manager hostato nel Cloud e il componente *fogOS* è il componente software cablato nel Fog Node (in questo caso, un sistema operativo).
  Per quanto riguarda Stack4Things, l'architettura è quella raffigurata in Figura 3.6 [26]. Il componente *Stack4Things* è il componente manager hostato nel Cloud (integrando l'infrastruttura messa a disposizione da OpenStack, piattaforma di management Cloud già esplorata nella sezione sul Cloud Computing), e il componente *s4t lighting-rod* è il componente software cablato nel Fog Node;

- *Framework di sviluppo*
  Un framework di sviluppo per l'IoT è definito come:

  A set of tools (e.g., libraries, microservices, abstraction layers) easing the development of Fog applications for the IoT and assisting the developer in focusing on the application logic rather than on the distributed nature of the Fog infrastructure on top of which the application wil be deployed

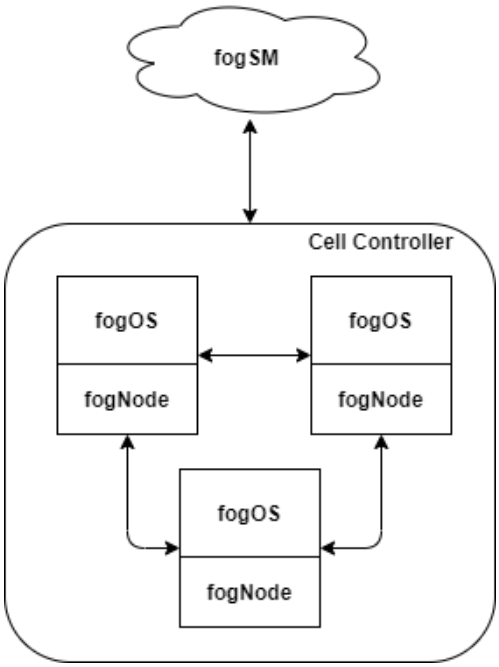


Figure 3.5: Architettura della piattaforma della Nebbiolo Technologies

Come tutti i framework, i framework per lo sviluppo di applicazioni IoT si configurano come una serie di librerie che caratterizzano, a tempo di esecuzione, un *ambiente di esecuzione* che mette a disposizione un'ampia gamma di servizi per, tra le altre cose:

- Interfacciarsi a dispositivi e sensori;
- Utilizzare protocolli come MQTT e CoAP per lo scambio di messaggi e consumo di servizi;
- Interfacciarsi ai servizi Cloud.

Talvolta questi framework di sviluppo sono rilasciati con le piattaforme software al fine di interagire con queste ultime in maniera più agevole. Ad esempio, la piattaforma software della Nebbiolo Technologies, la cui architettura è stata studiata nel precedente punto, possiede, nell'ambito del componente fogOS, un framework di sviluppo per l'usufrutto dei servizi della piattaforma;

Una cosa notevole è che la descrizione svolta finora del paradigma di Fog Computing ha prescisso in buona parte degli aspetti tecnologici che abilitano, di fatto, la fruizione del paradigma stesso. In effetti, le idee scaturite dal Fog Computing possono essere implementate solo nel caso ci sia effettivamente un *pull tecnologico* dal basso, ossia ci siano a disposizione soluzioni hardware economiche per l'implementazione del sistema così come presentato ad alto livello nella sua architettura distribuita e gerarchica. Per questo, nella prossima sezione, si approfondiranno quelli che sono gli aspetti tecnologici del Fog Computing.

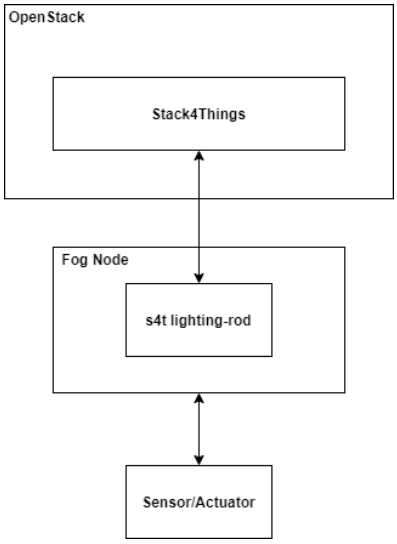


Figure 3.6: Architettura della piattaforma Stack4Things

3.2.2 Tecnologie abilitanti

La letteratura scientifica spesso si sofferma sugli aspetti di alto livello del Fog Computing, trascurando i dettagli sulle piattaforme hardware che abilitano, di fatto, lo schieramento di questo paradigma.

[27] presenta un'interessante modello in cui vengono forniti, insieme alla suddivisione a livelli dei vari strati di un'applicazione IoT sviluppata secondo il paradigma del Fog Computing, dettagli sulle proprietà hardware dei nodi da dispiegare per ciascun livello dell'architettura.
Il lavoro parte prima con la presentazione di varie architetture IoT (Figura 3.7)

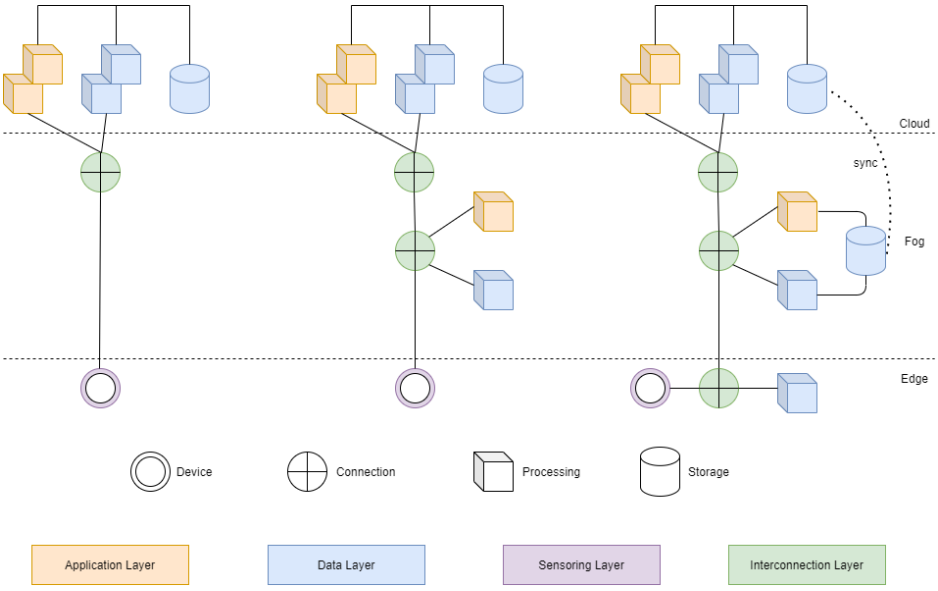


Figure 3.7: Comparazione di varie architetture per applicazioni IoT

Elemento	Classe	Hardware	Servizio
	C5	High	Services, Microservice and Serverless
	F4	Storage $\geq$ 16GB Memory $\geq$ 4GB	Services, Microservice and Serverless
	F3	Storage $\leq$ 8GB Memory $\leq$ 2GB	Microservice and Serverless
	E2	Storage $\leq$ 4GB Memory $\leq$ 512MB	Serverless
	E1	No Storage, Low Memory	Serverless

Gli autori danno, in questo caso, due differenti viste, che accoppiano la *System dimension* con la *Device dimension*: il partizionamento logico dei differenti livelli, ossia i livelli *Application*, *Data*, *Sensoring* e *Interconnection* sono disaccoppiati dalla vista fisica che partiziona i nodi in differenti tipi, ossia i nodi *Device*, *Connection*, *Processing* e *Storage*. Quindi, anche se, per esempio, i servizi del livello Interconnection sono omogeneamente collassati in un singolo livello, essi sono, a tempo di esecuzione, schierati su differenti e indipendenti componenti.
L'architettura IoT di interesse, ossia quella organizzata secondo il paradigma del Fog Computing, è ulteriormente specificata in accordo con il modello in Figura 3.8.

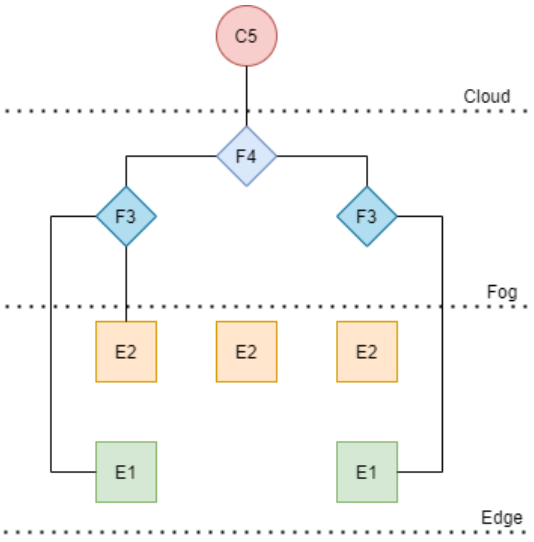


Figure 3.8: Architettura presentata da [27]

Dove ogni nodo ha specifiche caratteristiche definite dalla tabella presente in cima alla pagina. Da questa vista l'interessante osservazione degli autori è che perché i dispositivi possano essere dispiegati sui vari strati dell'architettura di Fog Computing, delle specifiche hardware devono essere rispettate. Oltretutto, in base alle specifiche hardware, il *tipo* di servizio erogato è differente. Gli autori discriminano tre differenti tipologie di servizi:

- *Service*
  Un servizio di tipo Service è tipico delle architetture SOA, dove un'applicazione viene sviluppata prevedendo a priori che per rispettare le proprie specifiche funzionali e

non-funzionali, tale applicazione dovrà far uso di servizi esterni forniti da entità indipendenti;

- *Microservice*
  Un servizio di tipo Microservice è tipico delle architetture a microservizi, dove un'applicazione viene sviluppata prevedendo che l'insieme di funzionalità sia sviluppato isolando ciascun gruppo omogeneo di funzionalità in un servizio: per implementare la logica applicativa, l'applicazione orchestrerà oppure coordinerà questi microservizi;

- *Serverless*
  Un servizio di tipo Serverless è tipico delle architetture monolitiche, dove un'applicazione viene sviluppata prevedendo che le funzionalità siano sviluppate prevedendo a basso livello meccanismi di chiamate a funzioni;

Ora che è stato fornito un dettaglio di più basso livello, è chiaro che il Fog Computing necessita di hardware e tecniche di sviluppo software specifici.
Puliafito et al. [20] hanno anch'essi fornito una schematizzazione delle piattaforme hardware più indicate a svolgere il ruolo di Fog Nodes. Tipicamente, le piattaforme hardware di questo tipo possiedono:

- Risorse di calcolo sufficientemente elevate, sotto forma di CPU, FPGA o GPU;
- Supporto per una classe sufficientemente ampia di interfacce per periferiche esterne, come USB, HDMI, UART, GPIO, ecc.
- Interfacce di rete Wi-Fi, Ethernet, Bluetooth, ecc.
- Connettori per dispositivi di storage esterni sufficientemente capienti, dato che questi dispositivi devono disporre di sistemi operativi per supportare i meccanismi di virtualizzazione, come macchine virtuali oppure containerizzazione.

Viene anche presentata, nella seguente pagina, una comprensiva tabella di piattaforme hardware classificate come idonee per essere dispiegate come Fog Nodes.

Costruttore	Modello	Risorse Hardware	Connettività	Interfacce per sensori e attuatori esterni
Nebbiolo Technlogies	fogNode	4-8 cores x86 i5/i7 CPUs, 8-16GB RAM	Ethernet	No
TTTech	MFN 100	Intel Atom 4 cores 1.8GHz CPUs, 4-8GB RAM	Ethernet	2 USB ports
Cisco	800 Series Industrial Integrated Services Routers	Intel Atom 2 cores 1250MHz CPU, 2GB RAM	Ethernet, LTE	2 interfacce seriali asincrone
Cisco	Compute Modules for the 1000 Series	AMD GX-410VC 4 cores 800MHz CPU, 4GB RAM	Ethernet	1 porta USB
Dell	Edge Gateway 5000	Intel Atom E3825 CPU, 2GB RAM	Ethernet, Wi-Fi, BLE, LTE	6 differenti interfacce seriali
Dell	Embedded Box PCs	4 cores x86 i5/i7 CPU, 4-32GB RAM	Ethernet, Wi-Fi	5 porte USB, 2 differenti interfacce seriali
HPE	GL20 IoT Gateway	Intel 4300U 2 cores i5 CPU, 8GB RAM	Ethernet, Wi-Fi	5 porte USB, 2 differenti interfacce seriali
HPE	Edgeline EL1000/4000	1-4 Intel Xeon D 8-16 cores ciascuno, fino a 128GB RAM	Ethernet	Slot di espansione PCIe
Raspberry Pi Foundation	Raspberry Pi 3 Model B+	1.4GHz 4 cores ARM Cortex-A53 CPU, 1GB RAM	Ethernet, Wi-Fi, BLE	4 porte USB, 40 pin GPIO, Camera Serial Interface
Qualcomm	DragonBoard 820c	2.35GHz 4 cores CPU, Adreno 350 GPU, 3GB RAM	Wi-Fi, Bluetooth	3 porte USB, pin, Camera Serial Interface
Qualcomm	DragonBoard 410c	1.2GHz 4 cores ARM Cortex-A53 CPU, Adreno 306 GPU, 1GB RAM	Wi-Fi, Bluetooth	3 porte USB, pin, Camera Serial Interface
Intel	Edison	500MHz 2 cores CPU, 100MHz MCU, 1GB RAM	Wi-Fi, Bluetooth	40 pin GPIO

3.3 Sfide

Chiaramente l'impiego del Fog Computing trasporta con sé una serie di sfide che sono spesso il fulcro di molti articoli scientifici. Puliafito et al. [20] svolgono una review delle varie sfide poste al paradigma di Fog Computing.
Le sfide dipendono chiaramente dalle caratteristiche del Fog Computing, tra cui ritroviamo:

- Distribuzione geografica;
- Alta eterogeneità;
- Potenza computazionale;
- Performance di connettività;
- Ambienti vulnerabili di schieramento.

Date queste caratteristiche, le sfide da affrontare sono:

- *Supporto alla mobilità*
  Per supporto alla mobilità si intende il supporto fornito ai dispositivi IoT mobili in grado di cambiare la propria posizione mentre fanno uso dei servizi offerti dai Fog Node. L'IoMT (Internet of Mobile Things) rappresenta quindi una sfida proprio a causa della *distribuzione geografica* e della *potenza computazionale* dei Fog Node: un dispositivo IoT mobile è interessato a usufruire dei servizi Fog facendo uso del Fog Node più opportuno, in termini di vicinanza geografica e di potenza computazionale;

- *Orchestrazione*
  Per orchestrazione si intende:

  To coordinate, arrange, and jointly manage computing resources and services to satisfy specific functional and non-functional requirements

  Chiaramente, essendo le risorse e i servizi a disposizione tra i più disparati, essi vanno orchestrati per fornire le funzionalità, nei rispetti anche dei requisiti non-funzionali, agli utenti dell'applicazione, che chiaramente sono i dispositivi IoT stessi. Ciò che in qualche modo rende una sfida l'orchestrazione è la *distribuzione geografica*, l'*alta eterogeneità*, la *potenza computaizonale* e le *performance di connettività* dei Fog Node;

- *Modelli di schieramento*
  Per modello di schieramento si intende il modello per la predisposizione dei Fog Node nell'ambiente IoT. Di modelli di schieramento proposti ce ne sono diversi, tra cui:

  - Scoperta dinamica di Fog Node supportata da un modello di guadagno da parte dei provider;
  - Contratti sottoscritti in anticipo con provider Fog;
  - Federazione di Fog Nodes;
  - Scambio Fog-Cloud.

  Ciascuno di questi modelli di schieramento presenta delle criticità legate alla *distribuzione geografica*, *alta eterogeneità* e all'*ambiente potenzialmente vulnerabile di schieramento*;

- *Security*
  Uno degli aspetti che più ha lasciato interdetto il mondo industriale per l'adozione del paradigma di Fog Computing è sicuramente la security. Se, come abbiamo visto, da un lato l'adozione del Fog Computing consente di scaricare i dispositivi IoT, spesso poveri

in termini di risorse computazionali, dall'implementazione di controlli di sicurezza, l'ambiente Fog è comunque soggetto a diverse criticità in termini di sicurezza.
Per citare alcuni esempi di criticità, ritroviamo:

- Attacchi *man-in-the-middle* ai Fog Node;
- Privacy e confidenzialità dei dati nell'affidarsi ai meccanismi automatici di migrazione dei servizi fog;
- Regolamentazioni territoriali sul trattamento dei dati.

Dunque, a intaccare la sicurezza sono le caratteristiche di *distribuzione geografica*, *alta eterogeneità*, *potenza computazionale* e *ambienti di schieramento vulnerabili*.

Chapter 4

Internet of Things

Le origini del termine "Internet of Things" risalgono al 1999 e sono da attribuire a Kevin Ashton, l'allora direttore esecutivo dell'Auto-ID center del **MIT** (Massachusetts Institute of Technology) durante il lavoro sulle infrastrutture di identificazione a radiofrequenza (RFID) in rete. L'Unione Internazionale delle Telecomunicazioni (**ITU**) definisce l'Internet degli oggetti come "*un'infrastruttura globale per la società dell'informazione, che abilita servizi avanzati interconnettendo cose (fisiche e virtuali) basate su informazioni e comunicazioni interoperabili, esistenti e in evoluzione tecnologie*" [28].

I campi di applicazione delle tecnologie IoT sono tanto numerosi quanto diversi, poiché le soluzioni IoT si stanno estendendo sempre più a tutte le aree della vita quotidiana. Le aree di applicazione più importanti includono, ad esempio, l'industria intelligente, dove lo sviluppo di sistemi di produzione intelligenti e siti di produzione connessi va sotto il nome di **Industry 4.0**. Nella **smart home**, l'attenzione è posta su termostati intelligenti e sistemi di sicurezza, mentre le applicazioni **smart energy** si concentrano su contatori intelligenti di elettricità, gas e acqua. Le soluzioni di **trasporto intelligente** includono, ad esempio, il monitoraggio della flotta di veicoli e l'emissione di biglietti mobili, mentre in ambito medico intelligente vengono affrontati argomenti come la sorveglianza dei pazienti e la gestione delle malattie croniche. Infine, per quanto riguarda le **smart cities**, soluzioni IoT sono il monitoraggio in tempo reale della disponibilità di parcheggi e l'illuminazione intelligente delle strade.

4.1 Storia dell'IoT

Presentiamo un excursus storico per approfondire la nascita dell'Internet of Things, un termine, un concetto, un'infrastruttura di cui si parla quotidianamente.[1]

1990 John Romkey creò un tostapane, considerato il primo dispositivo IoT, che poteva essere acceso e spento su Internet per la conferenza INTEROP dell'ottobre '89. Dan Lynch, presidente di Interop, ha promesso a Romkey che, se Romkey fosse stato in grado di "far apparire il suo tostapane in rete", l'elettrodomestico avrebbe ottenuto un posto da protagonista tra gli espositori della conferenza. Il tostapane era collegato a un computer con rete TCP/IP.

1991 Tim Berners-Lee crea la prima pagina web.

1991 Mark Weiser scrivere l'articolo "Il computer per il 21° secolo":

[1] La fonte di informazioni per la realizzazione della sezione 4.1 è: https://www.postscapes.com/iot-history/

"Le tecnologie più profonde sono quelle che scompaiono. Si intrecciano nel tessuto della vita di tutti i giorni fino a renderle indistinguibili."

1995 Internet diviene commerciale con Amazon ed Ebay.

1998 Google viene incorporata.

1998 Scott Brave, Andrew Dahley e il professor Hiroshi Ishii sviluppano inTouch:

"inTouch applica oggetti fisici distribuiti sincronizzati per creare un telefono tangibile, una comunicazione tattile a lunga distanza."

1998 Mark Weiser costruisce una fontana d'acqua fuori dal suo ufficio il cui flusso e altezza imitano l'andamento del volume e dei prezzi del mercato azionario.

*"L'**ubiquitous computing** è all'incirca l'opposto della realtà virtuale, dove la realtà virtuale mette le persone all'interno di un mondo generato dal computer, l'ubiquitous computing costringe il computer a vivere qui nel mondo con le persone."*

1999 Il termine Internet of Things viene coniato da Kevin Ashton, direttore esecutivo dell'Auto-ID Center.

1999 Viene aperto Auto-ID Labs, fondato da Kevin Ashton, David Brock e Sanjay Sarma che hanno contribuito l'**EPC**, un sistema globale di identificazione degli articoli basato su RFID destinato a sostituire il codice a barre UPC.

2000 LG lancia *il primo frigorifero connesso al mondo*. A causa del prezzo del prodotto (oltre 20.000 dollari), questo dispositivo IoT non è stato certo un successo di vendite, ma ha prefigurato una tendenza.

2005 L'Unione Internazionale delle Telecomunicazioni (ITU) dell'ONU pubblica il suo primo rapporto sul tema dell'IoT.

"Una nuova dimensione è stata aggiunta al mondo delle tecnologie dell'informazione e della comunicazione (TIC): connettività da qualsiasi momento e luogo per chiunque, ora avremo connettività per qualsiasi cosa. Le connessioni si moltiplicheranno e creeranno una rete dinamica completamente nuova di reti - un Internet delle cose."

2006-2008 Sviluppo nella consapevolezza del concetto di IoT grazie alla prima conferenza europea sull'IoT in Svizzera, che ha riunito i principali ricercatori e professionisti del mondo accademico e industriale.

2008-2009 Il Business Solutions Group di Cisco segna la nascita dell'IoT in questi anni in cui il numero di "things" connesse ad Internet è maggiore del numero di persone connesse.

2008 Il National Intelligence Council degli Stati Uniti ha elencato l'Internet of Things come una delle 6 "Tecnologie civili dirompenti" con potenziali impatti sugli interessi degli Stati Uniti fino al 2025.

2010 Il premier cinese Wen Jiabao definisce l'IoT un settore chiave per la Cina e ha in programma di effettuare importanti investimenti in esso.

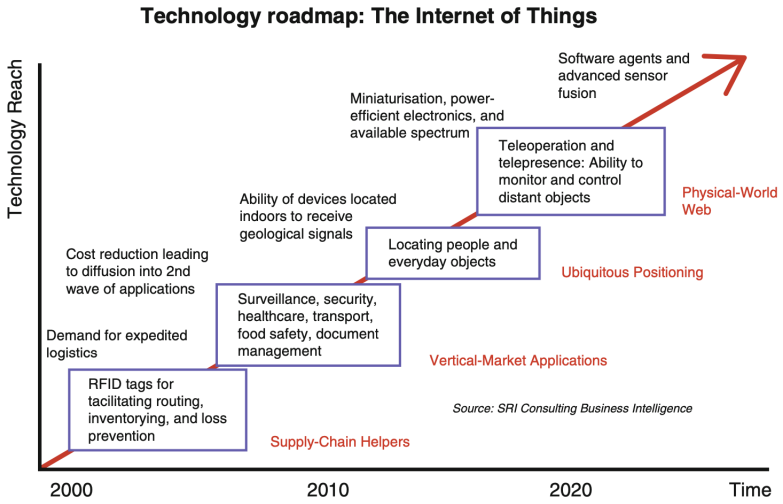


Figure 4.1: Roadmap dell'Internet of Things.

In figura 4.1 è possibile vedere la roadmap tecnologica dell'IoT: dagli anni 2000, i tag RFID erano utilizzati per tracciare l'instradamento dell'inventario per evitare la perdita di logistica; questo ha spianato la strada all'aumento della domanda di logistica fornendo supporto agli aiutanti della catena di approvvigionamento. Tra la metà del 2000 e il 2010 le applicazioni del mercato verticale come sorveglianza, sicurezza, sanità, trasporti, sicurezza alimentare e mappatura dei documenti hanno registrato un grande sviluppo negli aspetti tecnologici. Questo sviluppo nelle applicazioni del mercato verticale ha portato alla progressione dell'**ubiquitous computing**, in cui la tecnologia aiuta a localizzare le persone e gli oggetti di uso quotidiano. La localizzazione degli oggetti che risiedono ovunque, anche all'interno o nei sotterranei, dove le tecnologie satellitari potrebbero non essere disponibili o inadeguate, è chiamata **posizionamento onnipresente**.
Il progresso si muove verso il web del mondo fisico per la miniaturizzazione, l'elettronica efficiente dal punto di vista energetico e lo spettro disponibile per controllare oggetti distanti. Possiamo definire l'**Internet of Things come la rete di sensori in cui i dati vengono scambiati sotto la connettività utilizzando protocolli e sistemi di rete**. Lo scambio di dati può essere bidirezionale o unidirezionale in cui qualsiasi sistema potrebbe inviare o ricevere dati o entrambi. Le organizzazioni o il campo delle applicazioni che fanno uso dei dati utilizzeranno **cloud, fog o edge computing.** Queste paradigmi informatici aiutano ad utilizzare notevolmente la varietà di tecniche di elaborazione, risorse che forniscono l'archiviazione dei dati e queste includono anche l'Internet delle cose industriale (**IIoT**) [29]. Sebbene questi meccanismi di calcolo suonino simili, formano strati diversi nell'IIoT.

4.2 Stack tecnologico

L'implementazione di un prodotto connesso richiede in genere la combinazione di più componenti software e hardware in uno stack multistrato di tecnologie IoT. Come illustrato in 4.2 [28], uno stack tecnologico IoT è composto da tre strati principali, ovvero il **livello oggetto/dispositivo**, il **livello connettività** e il **livello cloud IoT**. A livello del dispositivo, l'hardware specifico dell'IoT, come sensori, attuatori o processori aggiuntivi, può essere aggiunto ai componenti hardware di base esistenti e il software incorporato può essere modificato o integrato di nuovo per gestire e utilizzare la funzionalità dell'oggetto fisico.

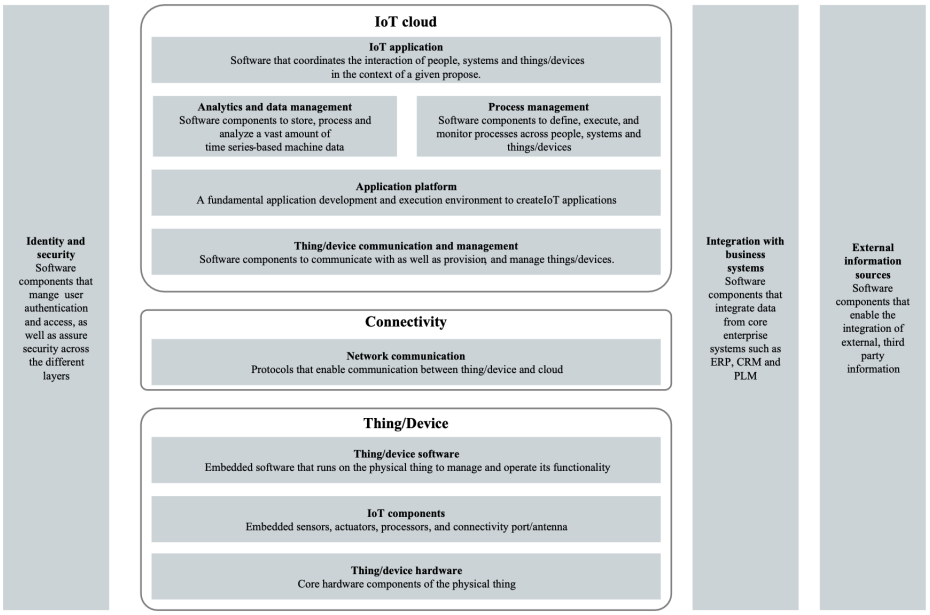


Figure 4.2: Stack tecnologico IoT.

A livello di connettività, protocolli di comunicazione come MQTT consentono la comunicazione tra la singola thing e il cloud. E a livello di cloud IoT, il software di comunicazione e gestione dei dispositivi viene utilizzato per comunicare, fornire e gestire le cose connesse, mentre una piattaforma applicativa consente lo sviluppo e l'esecuzione di applicazioni IoT [28]. Il **software di analisi e gestione dei dati** serve per archiviare, elaborare e analizzare i dati generati dalle cose connesse; il s**oftware di gestione dei processi** aiuta a definire, eseguire e monitorare i processi tra persone, sistemi e cose; il **software applicativo IoT** coordina l'interazione di persone, sistemi e cose per un determinato scopo. Attraversando tutti i livelli, i **componenti software gestiscono gli aspetti di identità e sicurezza, nonché l'integrazione con i sistemi aziendali**, ad esempio per ERP o CRM, e con fonti di informazioni esterne.
Un concetto molto ricorrente è la **piattaforma IoT**: il termine "piattaforma" indica "*un gruppo di tecnologie utilizzate come base su cui vengono sviluppate altre applicazioni, processi o tecnologie*". Nell'Internet degli oggetti, le piattaforme IoT sono prodotti software, che offrono *insiemi di funzionalità indipendenti dall'applicazione* che possono essere utilizzate per creare applicazioni IoT. Non esiste una configurazione standard di una piattaforma IoT, poiché ciascun fornitore si concentra su aspetti diversi, ma esiste una moltitudine di piattaforme IoT, che soddisfano esigenze e aree di applicazione specifiche. Mentre alcune piattaforme IoT, ad esempio ***Eclipse***, sono piuttosto focalizzate sulle things e offrono funzionalità per sviluppare e gestire applicazioni incorporate su tali things, altre piattaforme IoT, come ***Xively***, si concentrano su funzionalità specifiche dell'IoT per integrare piattaforme non-IoT. Infine, un terzo tipo di piattaforme IoT offre un set di funzionalità il più completo possibile, andando oltre le tecnologie IoT di base per ottenere una **piattaforma IoT all-in-one**.

4.3 IoT Applications

Di seguito, presentiamo una tabella riassuntiva delle applicazioni dell'IoT, alcune delle quali saranno affrontate più nel dettaglio nelle sottosezioni a seguire.

No.	*Applicazione*	Descrizione
1	*Infrastructure management*	Controllo del funzionamento delle infrastrutture urbane e di massa.
2	*Industrial IoT*	Fornisce all'ingegneria industriale sensori e analisi dei big data per creare macchine brillanti.
3	*Smart retail*	Effettua l'ordine automatico delle provviste necessarie per i vivi.
4	*Smart supply chain*	Notifica al destinatario i dettagli dell'ordine effettuato.
5	*Trasporto e logistica*	Garantire meno incidenti fornendo più regole per la gestione dei veicoli.
6	*Connected cars*	Introduzione del concetto di collegamento delle auto con i luoghi vicini.
7	*Self-parking vehicles*	Precezione dei segnali intorno al veicolo e raggiungimento del parcheggio con un comando.
8	*Smoke detectors*	Un ambiente contaminato da emissioni industriali è controllato con questi dispositivi.
9	*Smart cities*	Ambiente ecologico con meno inquinamento ed elementi inquinanti.
10	*Constructions*	Macchine automatizzate per costruire edifici.
11	*Smart homes*	Assistenza nei lavori domestici grazie all'IoT.
12	*Smart panels*	La contaminazione dell'acqua e altre cose vengono saturate utilizzando tali dispositivi.
13	*Smart wearables*	Ricezione degli appuntamenti importanti tramite promemoria.
14	*Environment monitoring*	Sensi per assistere la vita umana e animali con avvisi in caso di catastrofi naturali.
15	*Biological sensors*	Utilizzati per studiare il DNA o altre molecole.
16	*Smart surveillance*	Garantisce sicurezza dal furto di informazioni personali.

4.3.1 Infrastructure management

L'infrastruttura IoT è composta da tre parti principali quali *l'infrastruttura di rilevamento, l'infrastruttura informatica e l'infrastruttura di archiviazione*. L'infrastruttura di rilevamento include dispositivi di comunicazione e rilevamento. L'infrastruttura informatica è necessaria per calcolare il consumo di energia e per gestire efficacemente i carichi dinamici da vari sensori. L'infrastruttura di archiviazione tradizionale deve essere sostituita con un'infrastruttura di archiviazione aggiornata per soddisfare gli utenti, poiché c'è un aumento della necessità di sensori. La gestione dell'infrastruttura è necessaria per monitorare e controllare i dispositivi nelle infrastrutture delle aree rurali e urbane. Questa infrastruttura include varie applicazioni come ponti, ferrovie, fattorie, ecc. L'infrastruttura è controllata monitorando il cambiamento di stato dei dispositivi dell'applicazione.
La gestione dell'infrastruttura aiuta a prendere decisioni rapide e anche a risparmiare denaro con analisi in tempo reale. Può anche essere utilizzato per programmare un evento come

attività di riparazione e manutenzione nell'industria, che sarà svolto coordinando l'attività tra i fornitori di servizi e gli utenti. Con queste strutture, **l'IoT aiuta a migliorare la qualità del servizio, il processo di gestione tempestivo degli eventi, il coordinamento delle risposte e anche la riduzione dei costi**. Molti settori come la produzione, l'agricoltura, la gestione dell'energia, il monitoraggio ambientale, l'edilizia e l'automazione domestica e così via traggono vantaggio da questa gestione dell'infrastruttura.

4.3.2 Smart retail

L'IoT offre una serie di applicazioni per i rivenditori per migliorare le operazioni dei negozi, riducendo al minimo i furti; aumentando gli acquisti dei clienti vendendo loro un numero maggiore di prodotti: consente di comunicare al cliente tutto ciò che c'è nel negozio e lo aiuta ad avvicinarsi alla decisione di acquisto. *I tag RFID sono attaccati a ogni prodotto nel negozio e alla sua disponibilità viene controllato periodicamente.* I clienti ottengono le informazioni sulla merce tramite Bluetooth o altri ricevitori digitali. L**a firma digitale porterà il cliente al prodotto**.
Per motivi di sicurezza nei negozi al dettaglio sono posizionate più videocamere. Utilizzando queste telecamere, le immagini vengono acquisite e trasmesse in streaming al cloud per l'archiviazione e anche per ulteriori analisi, se necessario. Queste sono utilizzate per motivi di sicurezza ma possono essere utilizzate anche per monitorare il comportamento del cliente all'interno del negozio. Per implementare un archivio protetto, il negozio necessita di **almeno un gateway per collegare varie apparecchiature con i sensori e anche per raccogliere i dati dell'immagine e archiviarli nella piattaforma cloud**.

4.3.3 Smart supply chain

L'IoT supporta la gestione della catena di approvvigionamento. La sicurezza è l'aspetto più importante che deve essere considerato. È necessario *evitare che tutte le informazioni cadano nelle mani sbagliate o vengano violate.* I **sensori devono inviare solo informazioni specifiche, che devono essere conservate in un ambiente cloud privato protetto**.

Monitoraggio del prodotto I sensori RFID e GPS tracciano il prodotto dal luogo di produzione al luogo di stoccaggio. Monitoreranno il prodotto e quindi *il produttore può ottenere dati granulari come la temperatura alla quale un articolo è conservato e così via.* Questi dati aiuteranno a improvvisare **il controllo di qualità, le consegne puntuali e le previsioni di consegna dei prodotti**.

Rapporti con i fornitori I dati ottenuti dal monitoraggio del prodotto sono anche utili alle aziende per l'aggiornamento dei propri programmi di produzione: testeranno il modo in cui i fornitori gestiscono le forniture e inviano le merci al produttore. Le merci di alta qualità definiscono un rapporto migliore con i clienti e il cliente migliore viene mantenuto.

Previsione e inventario I dispositivi intelligenti sono attentamente monitorati e gestiscono più variabili rispetto alle applicazioni precedenti. Il monitoraggio delle apparecchiature fornisce informazioni sulla manutenzione predittiva e avvisa i tecnici in caso di potenziali guasti e interruzioni costose.

4.3.4 Connected cars

Internet of Things apre alle case automobilistiche un modo fantastico e innovativo per aggiornare le loro auto in un mercato altamente competitivo: questo migliora anche l'idea, secondo cui **una persona può guidare in tutto il mondo e può rimanere connessa**

tutto il tempo. Prima dell'invenzione dell'IoT nelle automobili, bisognava cercare il percorso manualmente, mentre l'IoT lo ha reso più conveniente. Ad esempio, una persona potrebbe impostare il luogo di destinazione utilizzando i sistemi di posizionamento a terra (GPS) e l'autista verrà indirizzato al luogo di destinazione.
La **modalità pilota automatico** è la tecnologia più moderna introdotta negli ultimi tempi. Le telecamere ad alta risoluzione possono essere utilizzate per rilevare oggetti morbidi e duri nella corsia e anche nei dintorni del veicolo. Il radar rivolto in avanti con elaborazione avanzata raccoglie dati aggiuntivi sul mondo su una lunghezza d'onda ridondante che aiuta l'utente a vedere attraverso forti piogge, nebbia, polvere e persino l'auto dietro e davanti. Questo potrebbe anche aiutare a rilevare i segnali stradali e i segnali che sono già integrati nel sistema per poter guidare l'auto nel modo più sicuro possibile. Può anche adattarsi ai cambiamenti nelle strade ambientali in tutto il mondo e potrebbe anche offrire automaticamente il cambio di corsia. *IHS Markit prevede che entro il 2023 circoleranno oltre 70 milioni di auto connesse.* Progettare una piattaforma per veicoli connessi su cloud IoT core sarà una soluzione migliore per la gestione delle auto connesse utilizzando il **cloud IoT core su Google Cloud Platform** (GCP). I casi d'uso che possono essere definiti combinando l'archiviazione dei dati e le piattaforme sono l'assicurazione basata sull'utilizzo, la manutenzione predittiva, il monitoraggio delle merci e l'esperienza a bordo del veicolo personalizzata. Le principali sfide che un team di sviluppo dovrà affrontare in questo dominio sono la gestione dei dispositivi, l'acquisizione dei dati, l'analisi dei dati, le applicazioni e i modelli predittivi. Oltre a queste sfide, lo sviluppatore deve anche concentrarsi sulla sicurezza e impostare il livello limite per le applicazioni che progettano che vengono implementate nel sistema basato su cloud [28].

4.3.5 Self-parking vehicles

Le auto a guida autonoma vengono apprese dal cloud di dati, **le raccolte di dati sono freni, prestazioni di pneumatici e trasmissione, consumo di carburante ed efficienza del veicolo**. Il motivo principale per la guida autonoma è l'elevata sicurezza in quanto rispetterà tutte le regole e le norme del traffico. In media, la copertura dei parcheggi occupa il 31% dell'uso del suolo nelle grandi città, come San Francisco, e anche di più, l'81% a Los Angeles e il 76% a Melbourne, mentre all'estremità inferiore troviamo New York (18%), Londra (16%) e Tokyo (7%) [28]. Pertanto, la rimozione delle aree di parcheggio non necessarie è il primo compito dei comuni per creare una migliore pianificazione urbana. I dati sono forniti con elevata sicurezza e gli hacker non possono accedere facilmente ai dati senza l'autorizzazione dell'utente. Si crea confusione quando l'utente cerca di parcheggiare il proprio veicolo, ciò comporta una grande pressione e stress per l'utente che si traduce anche in una perdita di tempo. *Poiché il veicolo a guida autonoma viene parcheggiato automaticamente nell'area di parcheggio senza alcuna confusione, riduce la pressione e il tempo per l'utente.* L'IoT prevede lo scenario corrente ed esegue l'azione in base al problema.

4.3.6 Smart cities

Le città intelligenti *aumentano la qualità dello stile di vita migliorando infrastrutture come la fornitura di acqua pulita, energia elettrica affidabile e un'illuminazione pubblica efficiente.* Il motivo principale di una città intelligente è **fornire un approvvigionamento idrico più efficiente**, una soluzione innovativa alla congestione del traffico e un trasporto pubblico più affidabile. L'acqua può essere notevolmente risparmiata fornendo le seguenti strutture: rilevamento di perdite nei tubi; riciclaggio delle acque reflue; gestione dell'acqua piovana [28]. I vantaggi della città intelligente includono maggiore sicurezza, traffico ridotto, livelli più bassi di inquinamento, uso più efficiente dell'energia e miglioramento della qualità della vita complessiva per i futuri abitanti delle città. L'obiettivo principale delle città intelligenti è stato esteso per mantenere i cittadini al sicuro, per fornire opportunità di lavoro educativo

e tecnologico ai cittadini. *Gujarat International Finance Tec-City* è un ottimo esempio di città intelligenti. Ha una migliore pianificazione e sviluppo della città e la struttura Internet è fornita in tutta la città con una fornitura più rapida di prodotti a una spesa operativa inferiore. Il problema principale sta nella contaminazione dell'aria e per risolvere questo problema vengono realizzati molti edifici verdi.

4.3.7 Smart homes

Google sta portando l'automazione in tutto. Poiché tutto in casa è ben organizzato e completamente informatizzato, si può controllare tutto semplicemente con un'applicazione mobile per monitorare i dispositivi installati in casa. *I dispositivi che utilizzano l'elettricità possono essere aggiunti alla rete.* Le case intelligenti forniscono anche un'elevata sicurezza che impedisce agli hacker di penetrare. Diverse aziende hanno sviluppato elettrodomestici intelligenti che includono frigoriferi intelligenti, televisori e condizionatori d'aria. Ad esempio, Nest è un termostato automatizzato che rileva la temperatura e regola il condizionatore d'aria ai mezzi favorevoli, cioè, se la temperatura esterna è alta, abbassa la temperatura interna e viceversa, il che mette a proprio agio le persone.
E' stata addirittura progettato una casa intelligente in modo tale che il dispositivo registrerà le attività quotidiane all'interno della casa e memorizzerà le informazioni nel cloud storage. Il dispositivo acquisirà familiarità con queste informazioni e, quindi, in una fase successiva, il dispositivo eseguirà operazioni simili senza la necessità dell'intervento umano.

Chapter 5

Virtualizzazione e hypervisor

5.1 Introduzione

La virtualizzazione è un approccio nato negli anni '70, quando furono stabiliti i principi di base e progettati interi stack di elaborazione (hardware, VMM e OS) per supportare in maniera efficiente le macchine virtuali.
All'inizio degli anni '80 con la maturazione dei sistemi operativi, le macchine virtuali furono scartate in favore della costruzione di sistemi operating system-centric. Negli anni '80 e '90, con l'avvento del personal computer e del client/server, le macchine virtuali sono state relegate al mainframe. Ad esempio, i processori sviluppati in quegli anni (*MIPS, Sparc, x86*) non erano progettati esplicitamente per fornire supporto architetturale alla virtualizzazione data la mancanza di requisiti aziendali [30].

Nel 1997 il documento Disco ha rivisitato le macchine virtuali con una nuova prospettiva, in particolare come base per eseguire sistemi operativi comuni su multi-processori scalabili. Nel 1999 VMWare ha rilasciato VMware Workstation 1.0, *la prima soluzione di virtualizzazione commerciale per processori x86*. Alcun ianni dopo l'approccio è stato introdotto sulle piattaforme mobili.

Disco, VMware Workstation, VMware ESX Server, VirtualPC , Xen, Denali e Cells erano tutti originariamente progettati per *architetture che non fornivano supporto per la virtualizzazione*. Ognuno di questi sistemi ha poi adottato approcci diversi per aggirare i limiti dell'hardware del tempo: molte innovazioni di quell'epoca come **architetture hosted, paravirtualizzazione, migrazione in tempo reale, memory ballooning** rimangono rilevanti ancora oggi.
Con l'innovazine combinata di hardware e software (Intel ha introdotto il supporto hardware per macchine virtuali nel 2004) le macchine virtuali sono diventate di fondamentale importanza per organizzazioni IT, live migration, security.
Oggi, *le macchine virtuali sono onnipresenti negli ambienti aziendali*, utilizzate per virtualizzare server e desktop. Costituiscono **la base di tutti i cloud IAAS** (Infrastructure-as-a-Service), inclusi Amazon EC2, Google CGE, Microsoft Azure e OpenStack.

5.2 Definizioni

- La **virtualizzazione** è l'applicazione del principio di stratificazione attraverso la **modularità forzata**, per cui la risorsa virtuale esposta è identica alla risorsa fisica sottostante virtualizzata.

- Una **macchina virtuale** è un'astrazione di un'ambiente di elaborazione completo tramite la virtualizzazione combinata del processore, della memoria e dei componenti I/O di un computer.
- L'**hypervisor** è un software di sistema specializzato che gestisce ed esegue macchine virtuali.
- Il **monitor della macchina virtuale** (VMM) si riferisce alla parte dell'hypervisor che si concentra sulla virtualizzazione della CPU e della memoria.

5.2.1 Virtualizzazione

La virtualizzazione è l'**applicazione del principio di stratificazione attraverso la modularità forzata**, per cui *la risorsa virtuale esposta è identica alla risorsa fisica sottostante che viene virtualizzata* [30]. **La stratificazione è la presentazione di una singola astrazione**, realizzata aggiungendo un livello di riferimento indiretto, quando (i) l'indirizzamento si basa su un singolo livello inferiore e (ii) utilizza uno spazio dei nomi ben definito per esporre l'astrazione. Inoltre, **la modularità applicata garantisce che i client del livello non possano aggirare il livello di astrazione**, ad esempio per accedere direttamente alla risorsa fisica o avere visibilità sull'utilizzo dello spazio dei nomi fisico sottostante. La virtualizzazione non è quindi altro che un'istanza di stratificazione per la quale l'astrazione esposta è equivalente alla risorsa fisica sottostante.
Questa combinazione di indirezione, modularità forzata e compatibilità è un modo particolarmente potente sia per ridurre la complessità dei sistemi informatici sia per semplificare le operazioni.

Esempio: consideriamo i RAID, array ridondanti di dischi economici aggregati per formare un unico disco virtuale. Poiché l'interfaccia è compatibile (è un dispositivo a blocchi sia per i dischi virtuali che per quelli fisici), un filesystem può essere deployato in modo identico, indipendentemente dal fatto che il livello RAID sia presente o meno. Questo perchè il livello RAID gestisce le proprie risorse internamente, nascondendo gli indirizzi fisici dall'astrazione, i *dischi fisici possono essere scambiati nel disco virtuale in modo trasparente dal filesystem che lo utilizza*; questo semplifica le operazioni, in particolare quando i dischi si guastano e devono essere sostituiti.

La virtualizzazione **non** è quindi sinonimo di macchine virtuali. Inoltre, non è limitato a nessun particolare campo dell'informatica o a nessuno livello dello stack di calcolo.

- **Virtualizzazione nell'architettura del computer**: un esempio ben noto è la MMU, l'unità di gestione della memoria virtuale. La MMU infatti aggiunge un livello di riferimento indiretto che nasconde gli indirizzi fisici dalle applicazioni, in generale attraverso una combinazione di meccanismi di segmentazione e paging. Poiché sia la memoria fisica sia la memoria virtuale espongono la stessa astrazione di memoria indirizzabile a byte, la stessa architettura del set di istruzioni può funzionare in modo identico con la memoria virtuale quando la MMU è abilitata e con la memoria fisica quando è disabilitata.
- **Virtualizzazione all'interno dei sistemi operativi**: al suo interno, un sistema operativo espone in modo sicuro le risorse di un computer - CPU, memoria e I/O - a più applicazioni simultanee. Un sistema operativo controlla la MMU per esporre l'astrazione di spazi di indirizzi isolati ai processi; programma i thread sui core fisici in modo trasparente, quindi multiplexing nel software una risorsa fisica limitata (la CPU); monta più filesystem distinti in un unico spazio dei nomi virtualizzato.

- **Virtualizzazione nei sottosistemi I/O**: la virtualizzazione è onnipresente nei dischi e nei controller dei dischi, dove la risorsa da virtualizzare è un array di settori indirizzato a blocchi. Questo approccio è utilizzato dai controller RAID e dagli array di archiviazione.

Che sia eseguita in hardware, software o incorporata in sottosistemi, la virtualizzazione viene sempre ottenuta utilizzando e combinando tre semplici tecniche, illustrate nella figura 5.1. Innanzitutto, **il multiplexing espone una risorsa tra più entità virtuali.** Ci sono due tipi di multiplexing, nello spazio e nel tempo. Con il **multiplexing dello spazio**, la risorsa fisica viene partizionata (nello spazio) in entità virtuali. Ad esempio, il sistema operativo multiplexa diverse pagine di memoria fisica su diversi spazi di indirizzi. Per raggiungere questo obiettivo, il sistema operativo gestisce le mappature da virtuale a fisico e si affida al supporto architettonico fornito dalla MMU. Con il **multiplexing temporale**, la stessa

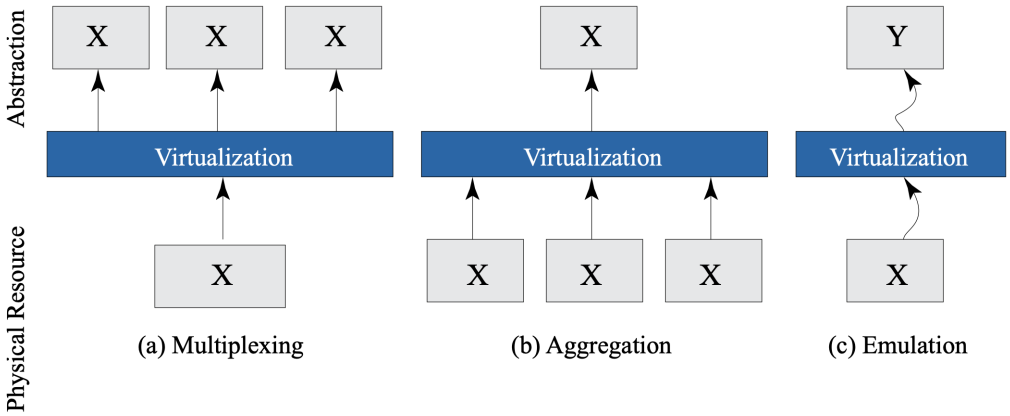


Figure 5.1: Tre tecniche base di virtualizzazione.

risorsa fisica viene programmata temporalmente tra entità virtuali. Ad esempio, lo scheduler del sistema operativo esegue il multiplex del core della CPU e dei thread hardware tra l'insieme di processi eseguibili. L'operazione di cambio di contesto salva il register file del processore nella memoria associata al processo in uscita, quindi ripristina lo stato del register file dalla posizione di memoria associata al processo in entrata.

L'**aggregazione** fa il contrario, **richiede più risorse fisiche e le fa apparire come un'unica astrazione**. Ad esempio, un controller RAID aggrega più dischi in un unico volume. Una volta configurato, il controller garantisce che tutte le operazioni di lettura e scrittura sul volume si riflettano in modo appropriato sui vari dischi del gruppo RAID. Il sistema operativo formatta quindi il filesystem sul volume senza doversi preoccupare dei dettagli del layout e della codifica.

Infine, l'**emulazione** si basa su un **livello di riferimento indiretto nel software per esporre una risorsa o un dispositivo virtuale che corrisponde a un dispositivo fisico**, anche se non è presente nel sistema informatico corrente. Gli emulatori tra architetture eseguono un'architettura del processore su un'altra: ad esempio, Apple Rosetta emula un processore PowerPC su un computer x86 per compatibilità con le versioni precedenti.
L'astrazione virtuale corrisponde a un particolare processore con un ISA ben definito, anche se il processore fisico è diverso. Memoria e dischi possono emularsi a vicenda: un disco RAM emula la funzione di un disco che utilizza DRAM come archivio di supporto. Il processo di paging della memoria virtuale fa l'opposto: il sistema operativo utilizza i settori del disco

per emulare la memoria virtuale.
Il multiplexing, l'aggregazione e l'emulazione possono naturalmente essere combinati insieme per formare uno stack di esecuzione completo.

5.2.2 Macchine virtuali

Una macchina virtuale è un ambiente di elaborazione completo con capacità di elaborazione, memoria e canali di comunicazione isolati. E' possibile fare la seguente classificazione:

- **Macchine virtuali basate sul linguaggio**, come Java Virtual Machine, Microsoft Common Language Runtime, motori Javascript incorporati nei browser e in generale l'ambiente di runtime di qualsiasi linguaggio gestito.
- **Macchine virtuali leggere**, che si basano su *una combinazione di meccanismi di isolamento hardware e software per garantire che le applicazioni in esecuzione direttamente sul processore* (ad esempio, come codice x86 nativo) siano isolate in modo sicuro da altre e dal sistema operativo sottostante. Questo include sistemi server-centric come Denali così come sistemi desktop-centric come Google Native Client. Le soluzioni basate su container Linux come Docker rientrano nella stessa categoria.
- **Macchine virtuali a livello di sistema**, in cui l'ambiente di elaborazione isolato assomiglia all'hardware di un computer in modo che la macchina virtuale possa eseguire un sistema operativo standard e le sue applicazioni, in completo isolamento dalle altre macchine virtuali e dal resto di l'ambiente. Tali macchine virtuali applicano il principio della virtualizzazione a un intero sistema informatico. *Ogni macchina virtuale ha la propria copia dell'hardware sottostante.* Ogni macchina virtuale esegue la propria istanza del sistema operativo indipendente, chiamata **sistema operativo guest**.

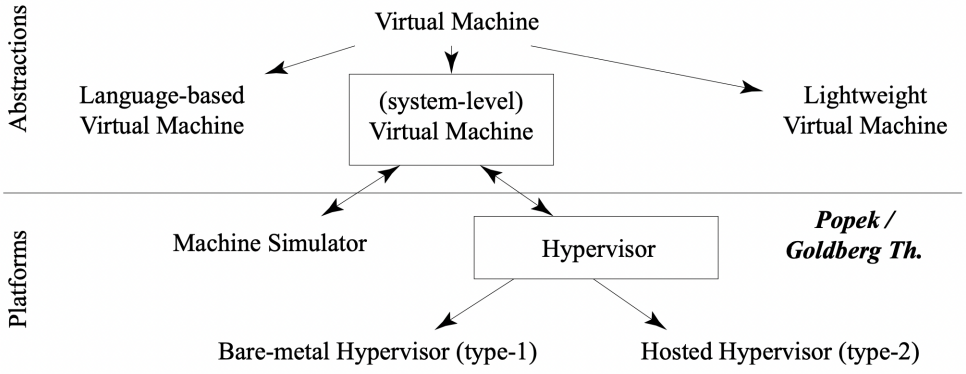


Figure 5.2: Classificazione macchine virtuali [30].

Nella figura 5.2 sono presenti anche le varie piattaforme che eseguono macchine virtuali a livello di sistema. Chiamiamo queste piattaforme un **hypervisor o un simulatore di macchina**, a seconda delle tecniche utilizzate per eseguire la macchina virtuale:

- **un hypervisor** si basa sull'esecuzione diretta sulla CPU per la massima efficienza, idealmente per eliminare del tutto i sovraccarichi delle prestazioni. Nell'esecuzione diretta, **l'hypervisor configura l'ambiente hardware, ma poi lascia che le istruzioni della macchina virtuale vengano eseguite direttamente sul processore**. Poiché queste sequenze di istruzioni devono operare all'interno dell'astrazione

della macchina virtuale, la loro esecuzione provoca traps che devono essere emulate dall'hypervisor. Questo paradigma **trap-and-emulate** è fondamentale per la progettazione di hypervisor;

- **un simulatore di macchina** è tipicamente implementato come una normale applicazione a livello utente, con l'obiettivo di fornire una simulazione accurata dell'architettura virtualizzata.

5.2.3 Hypervisor

Un hypervisor è un software di sistema che esegue macchine virtuali con l'obiettivo di ridurre al minimo i costi di esecuzione. Quando più macchine virtuali coesistono simultaneamente sullo stesso sistema di computer, *l'hypervisor multiplexa (cioè alloca e schedula) le risorse fisiche in modo appropriato tra le macchine virtuali.*

Nota 5.1: Osservazione

Una macchina virtuale è considerata un duplicato efficiente e isolato della macchina reale. Il **VMM - Virtual Machine Monitor** ha tre caratteristiche essenziali. In primo luogo, il VMM fornisce un ambiente per i programmi che è essenzialmente identico alla macchina originale; secondo, i programmi in esecuzione in questo ambiente mostrano nel peggiore dei casi solo lievi diminuzioni di velocità; e infine, il VMM ha il controllo completo delle risorse di sistema.

L'hypervisor applica **il principio di stratificazione** al calcolatore secondo tre criteri specifici di e**quivalenza, sicurezza e prestazioni**.

Equivalenza: la duplicazione garantisce che la risorsa esposta (ovvero la macchina virtuale) sia equivalente al computer sottostante. Questo è un requisito forte, che è rilassato in una certa misura quando l'architettura lo richiede.

Sicurezza: l'isolamento richiede che le macchine virtuali siano isolate l'una dall'altra e dall'hypervisor, il che rafforza la modularità del sistema.

Prestazioni: il sistema virtuale deve mostrare nel peggiore dei casi una lieve diminuzione della velocità. Questo requisito finale separa gli hypervisor dai simulatori di macchine. Sebbene i simulatori di macchina soddisfino anche i criteri di duplicazione e isolamento, non soddisfano i criteri di efficienza poiché anche i simulatori di macchina veloci che utilizzano la traduzione binaria dinamica rallentano l'esecuzione del sistema target.

5.3 Hypervisor di tipo 1 e tipo 2

Dalla figura 5.2, si osserva che le architetture dell'hypervisor possono essere classificate nei cosiddetti tipo 1 e tipo 2: un hypervisor di tipo 1 controlla direttamente tutte le risorse del computer fisico; al contrario, un hypervisor di tipo 2 opera "come parte di" o "sopra" un sistema operativo host esistente.
Il requisito di implementazione dell'hypervisor specifica che le istruzioni vengano eseguite direttamente sull'host. Non indica come l'hypervisor ottiene il controllo per quel sottoinsieme di istruzioni che devono essere interpretate. Ciò può essere eseguito da un programma in esecuzione sulla macchina host o da un programma in esecuzione con un sistema operativo sulla macchina host. In caso di esecuzione in un sistema operativo, le primitive

del sistema operativo host possono essere utilizzate per semplificare la scrittura sul monitor della macchina virtuale. Di conseguenza, sorgono due ulteriori categorie di **VMM (Virtual Machine Monitor)**:

- **tipo 1 o bare-metal**: il VMM viene eseguito su una macchina bare (senza SO);
- **tipo 2 o hosted**: il VMM viene eseguito su un host esteso, sotto il sistema operativo host.

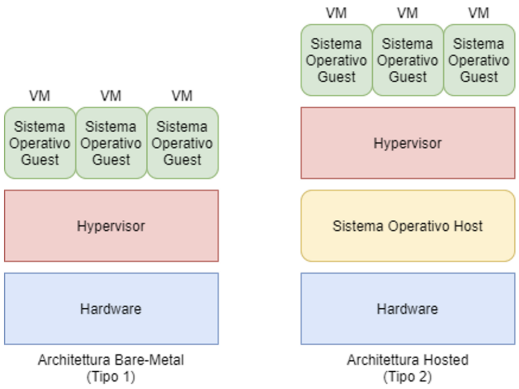


Figure 5.3: Tipi di hypervisor

In entrambi i tipi, il VMM crea la macchina virtuale. Tuttavia, in un ambiente di tipo 1, il VMM su una macchina bare deve eseguire lo scheduling del sistema e l'allocazione (reale) delle risorse. Di conseguenza, il VMM di tipo 1 può includere tale codice non specificamente necessario per la virtualizzazione. In un sistema di tipo 2, le funzioni di allocazione delle risorse e creazione dell'ambiente per la macchina virtuale sono suddivise in modo più chiaro. Il sistema operativo esegue la normale allocazione delle risorse di sistema e fornisce una macchina estesa standard.

Osservazione: l'enfasi è sull'allocazione delle risorse e non sul fatto che l'hypervisor venga eseguito in modalità privilegiata o non privilegiata. In particolare, un hypervisor può essere di tipo 2 anche quando viene eseguito in modalità kernel, ad esempio Linux/KVM e VMware Workstation funzionano in questo modo.

Entrambi i tipi si trovano comunemente nei sistemi attuali. Innanzitutto, **VMware, Xen e Microsoft Hyper-V sono tutti hypervisor di tipo 1**. Anche se Xen e Hyper-V dipendono da un ambiente host chiamato dom0, l'hypervisor stesso prende le decisioni sull'allocazione delle risorse e sulla pianificazione. Al contrario, **VMware Workstation**, VMware Fusion, KVM, Microsoft VirtualPC, Parallels e **Oracle VirtualBox sono tutti hypervisor di tipo 2**: collaborano con il sistema operativo host in modo che il sistema operativo host pianifichi tutte le risorse di sistema; il sistema operativo host pianifica l'hypervisor come se fosse un processo.
Alcuni hypervisor come VMware Workstation e Oracle VirtualBox sono portabili su diversi sistemi operativi host, mentre Fusion e Parallels funzionano con il sistema operativo host Mac OS X, Microsoft Virtual PC funziona con il sistema operativo host Windows e KVM funziona come parte dell'host Linux sistema operativo. Tra questi sistemi di tipo 2, KVM fornisce la migliore integrazione con il sistema operativo host, poiché il componente in modalità kernel degli hypervisor è integrato direttamente all'interno dell'host Linux come modulo del kernel.

La Figura 5.4 illustra *i componenti architettonici chiave di un sistema informatico virtualizzato*: mostra tre macchine virtuali, ciascuna con il proprio hardware virtuale, il proprio sistema operativo guest e le proprie applicazioni. **L'hypervisor controlla le risorse fisiche effettive e viene eseguito direttamente sull'hardware**: in questa architettura semplificata, l'hardware (virtuale o fisico) è costituito da elementi di elaborazione, che comprendono una o più CPU, la loro MMU e la memoria coerente con la cache.
Gli elementi di elaborazione sono collegati a un bus I/O, con due dispositivi I/O collegati: un disco e una scheda di interfaccia di rete; questo è rappresentativo di *una distribuzione del server*. Una piattaforma desktop includerebbe dispositivi aggiuntivi come una tastiera, video, mouse, porte seriali, porte USB, ecc. Una piattaforma mobile potrebbe inoltre richiedere un GPS, un accelerometro e radio.
Nella sua forma più elementare, un hypervisor utilizza due delle tre tecniche di virtualiz-

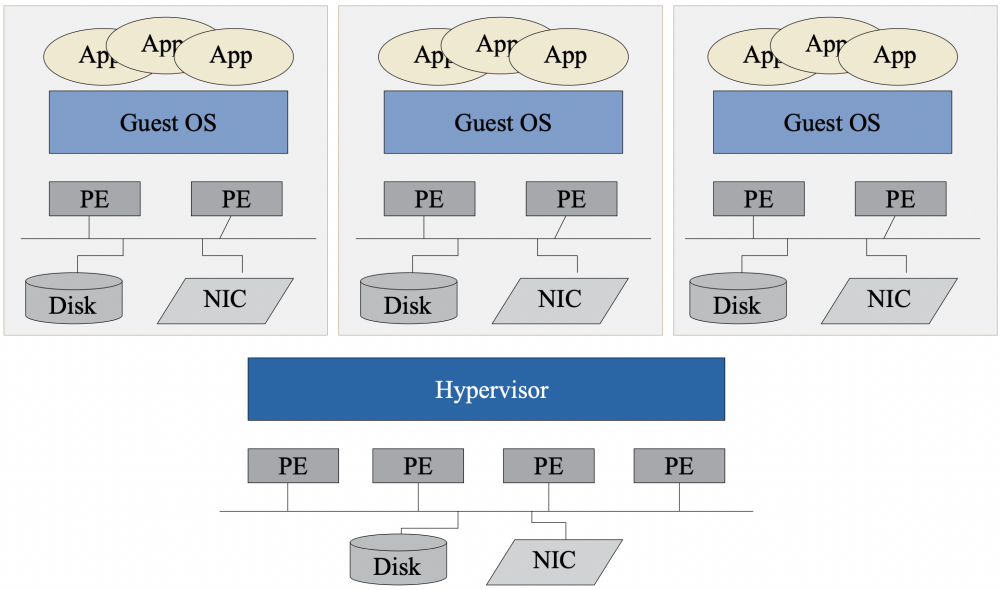


Figure 5.4: Componenti architetturali di un sistema virtualizzato.

zazione chiave, multiplexa (nello spazio e possibilmente nel tempo) il PE fisico attraverso le macchine virtuali ed emula tutto il resto, in particolare l'I/O bus e i dispositivi I/O. Questa combinazione di tecniche è necessaria e sufficiente nella pratica per raggiungere i criteri di efficienza. È necessario perché senza un meccanismo efficace per multiplexare la CPU e la MMU, l'hypervisor dovrebbe emulare l'esecuzione della macchina virtuale. In effetti, **la differenza principale tra un simulatore di macchina e un hypervisor è che il primo emula l'architettura del set di istruzioni della macchina virtuale, mentre il successivo la multiplexa.** Il multiplexing della CPU è un'attività di pianificazione, molto simile a quella eseguita dal sistema operativo per programmare i processi. L'entità di pianificazione (qui, l'hypervisor) imposta gli ambienti hardware (file di registro, ecc.). E quindi consente l'esecuzione dell'entità pianificata (la macchina virtuale) direttamente sull'hardware con privilegi ridotti.
Questa tecnica di pianificazione è nota come **esecuzione diretta** poiché l'hypervisor consente alla CPU virtuale di eseguire direttamente le istruzioni sul processore reale. Ovviamente, **l'hypervisor è anche responsabile di garantire la proprietà di sicurezza della macchina virtuale**. Assicura quindi che la CPU virtuale venga sempre eseguita con privilegi ridotti, ad es. fa in modo che non possa eseguire istruzioni privilegiate. Di conseguenza, l'esecuzione diretta della macchina virtuale porta a frequenti trap ogni volta che il

sistema operativo guest tenta di eseguire un'istruzione privilegiata, che deve essere emulata dall'hypervisor. Gli hypervisor progettati attorno all'esecuzione diretta seguono quindi un paradigma di programmazione trap-and-emulate, in cui la maggior parte del sovraccarico di esecuzione è dovuto all'hypervisor che emula trappole per conto della macchina virtuale.
La **memoria fisica è anche multiplexata tra le macchine virtuali**, in modo che ciascuna abbia l'illusione di una quantità di memoria fisica contigua e di dimensioni fisse. È simile all'allocazione tra i processi eseguita da un sistema operativo. Le sfide uniche nella creazione di un hypervisor risiedono nella virtualizzazione della MMU e nella capacità di esporre l'ambiente di esecuzione a livello di utente e di kernel alla macchina virtuale.

Chapter 6

Classificazione dei sistemi di elaborazione

6.1 Sistemi di elaborazione

Un sistema può essere definito per scomposizione gerarchica come un'*interconnessione* di elementi strutturali, configurati in maniera tale da collaborativamente esporre un comportamento che implementa quanto specificato. Più specificamente, parlando di sistemi di elaborazione, gli elementi strutturali di cui è composto il sistema sono nominati *processing elements*, ossia componenti in grado di elaborare digitalmente l'informazione in ingresso.

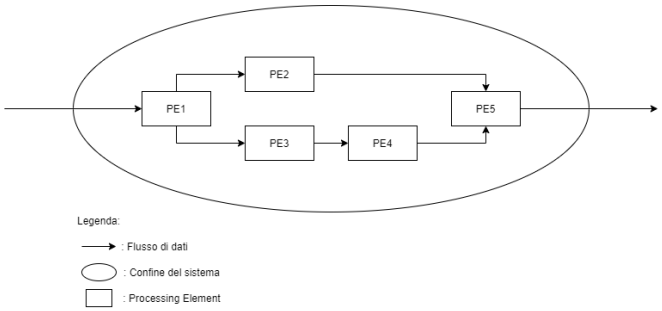
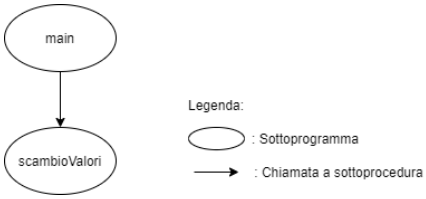


Figure 6.1: Descrizione di un sistema di elaborazione per scomposizione gerarchica

6.2 Descrizione di architetture

Ogni sistema è sviluppato secondo un'*architettura*, che può o meno essere ben specificata. Per architettura si intende l'insieme di decisioni, sotto diversi *punti di vista*, che il progettista ha preso nello sviluppare il sistema.
In base allo *stile architetturale*, il progettista è *vincolato* a sviluppare il sistema ricorrendo a classi di componenti e connettori notevoli, linguaggi descrittivi adeguati e certe classi di modelli architetturali.
Il modo in cui l'architettura è documentata è fortemente impattato dal *livello di astrazione* a cui il progettista si colloca. Il più alto livello di astrazione prevede di partizionare strutturalmente il sistema come una configurazione di *entità* la cui natura (hardware o software) non è ben specificata. La maniera con cui queste entità sono interconnesse e comunicano tra loro è astratta con *canali di comunicazione* complessi. Il comportamento è descritto con

linguaggi di modellazione e programmazione di alto livello, come diagrammi UML dinamici oppure C++.



(a) Descrizione strutturale di alto livello

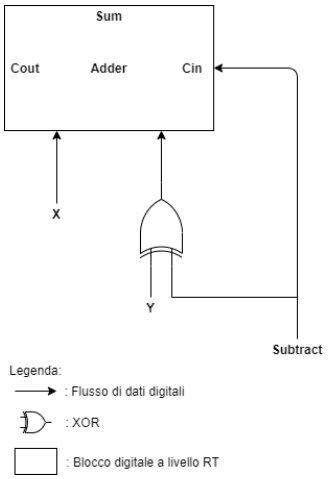
```
void main(){
    int a = 0;
    int b = 1;
    scambiaValori(&a,&b);
}

void scambiaValori(int* v1, int* v2){
    int temp;
    temp = *v2
    *v2 = *v1;
    *v1 = temp;
}
```

(b) Descrizione comportamentale di alto livello

Figure 6.2: Descrizione di alto livello

Il livello di astrazione a cui ci si colloca per la descrizione di architetture hardware è il livello RT (Register Transfer). Questo livello di astrazione modella entità puramente hardware che comunicano tra loro mediante scambio di segnali digitali. Per un approfondimento su come organizzare architetturalmente sistemi di elaborazione digitali a livello RT si rimanda alla dispensa di Mercogliano.



(a) Descrizione strutturale a livello RT

```
entity Adder is
   port(
   X : in std_logic;
   Y : in std_logic;
   Subtract: in std_logic;
   Cout : out std_logic;
   S : out std_logic
);
end Adder;


architecture Behavioral of Adder is

begin

S <= (X xor (Y xor Subtract)) xor Subtract;
Cout <= (X AND (Y xor Subtract)) OR (X AND (Y xor Subtract)) OR ((Y xor
Subtract) AND Cin);

end Behavioral;
```

(b) Descrizione comportamentale a livello RT

Figure 6.3: Descrizione a livello RT

6.3 IP Cores

Posto che un sistema digitale descritto a livello di astrazione RT viene specificato mediante HDL comportamentali e strutturali, gli IP (Intellectual Properties) sono delle specifiche fornite da terzi in un linguaggio HDL e integrabili all'interno di altri progetti per sistemi digitali.

Il mercato degli IP Core è particolarmente sviluppato in settori laddove venditori di dispositivi elettronici possono recuperare moduli già specificati e progettare, sulla base di tali moduli, altri moduli compatibili con le loro interfacce, al fine di descrive un sistema complessivo ottenuto mediante composizione di IP Core proprietari e self-made.
Alcune aziende, come la Intel, non hanno scommesso sin dall'inizio su questo mercato: questi venditori hanno, da sempre, fornito i propri prodotti come *scatole chiuse*, già implementate e impresse su piastre di silicio. Altre aziende, come ARM (Advanced RISC Machine), hanno preso la decisione diametralmente opposta: hanno puntato principalmente sulla produzione di IP Core rilasciabili tramite un accordo con venditori di terze parti interessati all'acquisto di questi moduli hardware. ARM *non produce* alcun processore, bensì li *specifica* e ne descrive le potenziali *prestazioni* basandosi su parametri di prestazioni relativi, come ad esempio $\frac{CoreMark}{MHz}$, che misura quanti benchmark il processore può eseguire per MHz.

6.4 Classificazione di sistemi di elaborazione

Esistono tante tassonomie che classificano i sistemi di elaborazione. Conte et al. [**?**] ne hanno elencate alcune, tra cui quella basata sulla *implementazione* dei sistemi.

6.4.1 System-on-a-Chip

I System-on-a-Chip (SoC) sono dei sistemi integrati su un singolo chip. Essi integrano, nel singolo chip, molti componenti, tra cui i più importanti come sistemi monoprocessore o multicore o manycore, memoria volatile (tipicamente SRAM) e memoria di massa (tipicamente memorie Flash). Oltre ciò, i SoC ospitano un certo numero di periferiche, il cui accesso avviene tramite registri *memory-mapped*.
Solitamente i SoC sono il cuore di ciò a cui generalmente ci si riferisce come *sistemi embedded*, oppure *microcontrollori*. Un SoC integra un gran numero di componenti: i venditori di microcontrollori, come la ST Microelectronics, personalizzano il proprio SoC recuperando IP core già progettati, da integrare all'interno del proprio progetto, e sviluppando IP Core personalizzati che si accoppiano a quelli già progettati e licenziati. Naturalmente, data la natura di questo mercato, aziende come ARM sono i leader del settore, a differenza di altri player come Intel.
La personalizzazione del SoC è importante quando si lavora con particolari vincoli tecnologici come *potenza dissipata*, *tensione di alimentazione*, *spazio* e *velocità*. Volendo, come esempio, prendere ARM, i progettisti hanno specificato tanti e diversi processori. Seppure classi di processori diversi possono aderire alla stessa architettura, la loro *microarchitettura* differisce.
Prendendo come esempio il Raspberry Pi 2B, questo sistema integra un SoC che a sua volta integra un processore ARM-Cortex A7, che aderisce all'instruction set ARMv7, ma nella variante A, che è adatta per l'esecuzione di applicazioni di alto livello, come sistemi operativi Linux. Similmente, il STM32F303VC integra un SoC che a sua volta integra un processore ARM. Tuttavia il processore integrato è l'ARM-Cortex M3, che aderisce anch'esso all'instruction set ARMv7, ma nella variante M, che modella processori a basso consumo e adatti ad applicazioni dedicate.

Una domanda potrebbe a questo punto essere: come è possibile integrare IP Core proprietari all'interno del proprio progetto? La risposta sta nel modo in cui l'IP Core da integrare è organizzato: per poter utilizzarlo è necessario utilizzare interfacce compatibili con quelle messe a disposizione dall'IP Core. Tipicamente, per sistemi embedded, IP Core ARM che offrono specifiche di processori si interfacciano all'esterno con bus aderenti allo standard AMBA, ossia uno standard per lo sviluppo di interfacce su bus molto impiegato per lo sviluppo di sistemi embedded.
Yiu [31] propone lo schema a blocchi in Figura 6.4 che modella a livello RT l'organizzazione

dei processori Cortex-M3 e Cortex-M4.

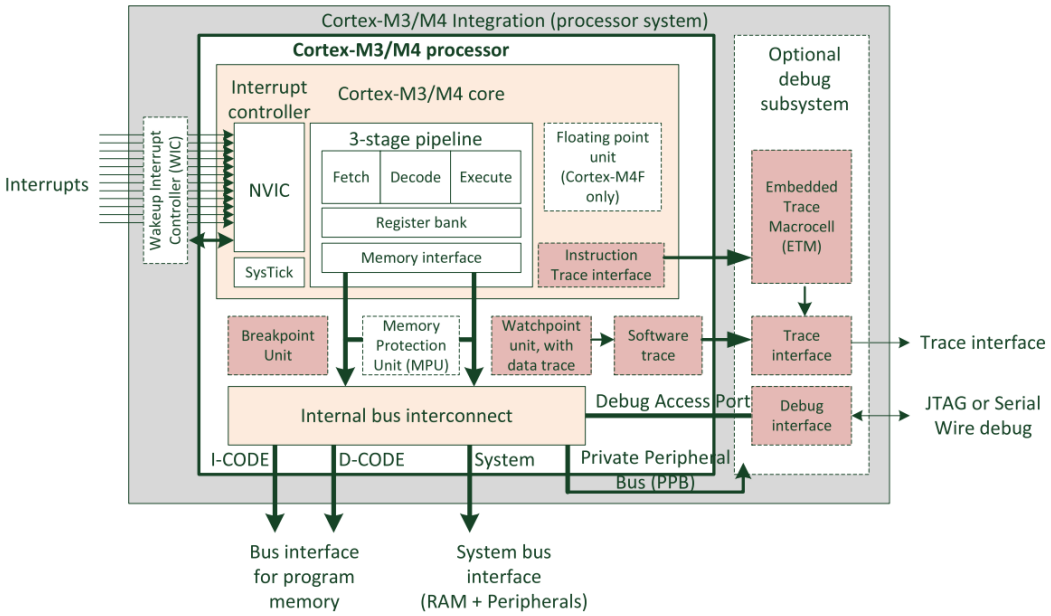


Figure 6.4: Schema a blocchi dei Cortex-M3 e M4

I blocchi tratteggiati sono i blocchi la cui integrazione, all'interno del processore, è a scelta del progettista. Il Cortex-M3 implementa un sottoinsieme di questi blocchi (in particolare, non c'è per il Cortex-M3 una Floating Point Unit). Oltretutto, l'infrastruttura di debug è opzionale, così come la Memory Protection Unit. Interessante, tuttavia, è il fatto che vengono offerte delle *interfacce* all'esterno, che in questo caso si suddividono in *interfacce per la program memory*,*interfacce per il system bus* e *interfacce per il debug*. A questo punto, un azienda produttrice di SoC, può personalizzare il proprio SoC integrando questo IP Core con i propri componenti sfruttando le interfacce per l'interconnessione. Volendo esemplificare il concetto, si può dare uno sguardo allo schema a blocchi fornito da ST per le schede STM32F303xx (disponibile nel reference manual dell'opportuna famiglia di schede ST) (Figura 6.5).
Su questo schema a blocchi si puà innanzitutto notare che il Cortex-M4 occupa una parte piccolissima. La maggior parte dello schema è ingombrato dalal presenza di tanti altri componenti, tra cui periferiche, sensori, memorie, sistemi di interconnessione. La personalizzazione è qui, quindi, che si sviluppa: un venditore di microcontrollori è libero di integrare tutti questi componenti per sviluppare la propria soluzione da offrire agli acquirenti.
Volendo terminare la discussione a riguardo dei SoC, accenniamo a come è possibile *costruire* questi sistemi. La costruzione dei sistemi SoC può procedere in due modi:

- Integrazione dei diversi componenti su una piastra di silicio a tempo di costruzione (ASIC);
- Integrazione dei diversi componenti e produzione di un IP Core sintetizzabile su una circuiteria logica programmabile (FPGA).

Il primo approccio porta chiaramente alla produzione di un sistema ottimizzato in quanto sviluppato ad-hoc per il progetto originario. Tuttavia la sintesi di questi circuiti porta a un grado di programmabilità nullo in termini di configurazione del circuito digitale. Invece, con il secondo approccio, è possibile sintetizzare in qualsiasi momento il proprio IP Core, con il costo però di un'implementazione meno ottimizzata.

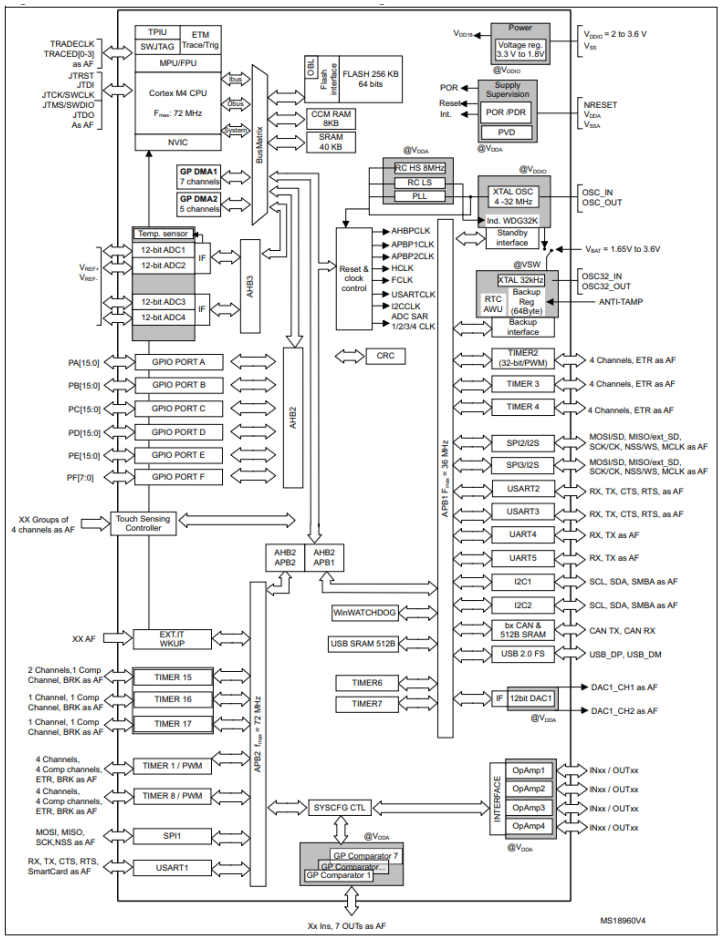


Figure 6.5: Schema a blocchi delle schede STM32F303xx

6.4.2 System-on-a-Board

I System-on-a-Board, noti anche come Single Board Computers (SBC), sono dei sistemi ottenuti integrando, insieme al SoC altri componenti, come:

- Periferiche esterne;
- Moduli fisici per l'interfacciamento a connettori altrimenti incompatibili (Phy);
- Ambienti di debug.

Consideriamo l'immagine che fotografa il System-on-a-Board STM32F3DISCOVERY (Figura 6.6).
Questo sistema integra un SoC STM32F303xx, ossia il chip centrale, e un certo numero di altre periferiche, come ad esempio un *accelerometro*. Sono chiaramente presenti dei connettori, utili per l'interfacciamento fisico con le periferiche esterne. Oltretutto, la scheda integra al suo interno tutta la circuiteria necessaria a gestire il debug del SoC: questa cosa non è scontata, in quanto, spesso, sono necessari dei dispositivi appositi per eseguire il debug del sistema, che convertono il protocollo USB, utilizzato dal sistema host interessato a effettuare il debug, con il protocollo JTAG, richiesto dal SoC (così come si vede nello schema a blocchi). Sulla board sono anche presenti dei dispositivi Phy per accoppiare connettori che altrimenti non potrebbero essere gestiti dal SoC.

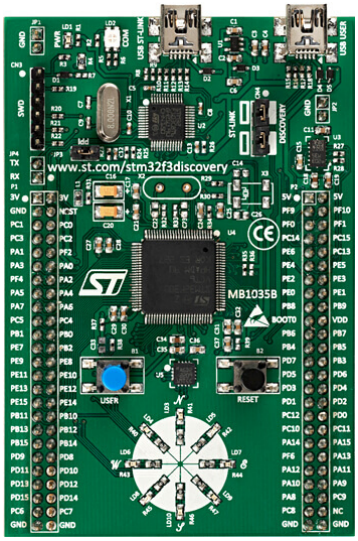


Figure 6.6: Foto della STM32F3DISCOVERY

Bibliography

[1] I. Foster, "What is the grid? a three point checklist," GRID today, vol. 1, pp. 32–36, 01 2002.

[2] F. Liu, J. Tong, J. Mao, R. Bohn, J. Messina, L. Badger, and D. Leaf, NIST Cloud Computing Reference Architecture: Recommendations of the National Institute of Standards and Technology (Special Publication 500-292). North Charleston, SC, USA: CreateSpace Independent Publishing Platform, 2012.

[3] P. Mell and T. Grance, "The nist definition of cloud computing," Tech. Rep. 800-145, National Institute of Standards and Technology (NIST), Gaithersburg, MD, September 2011.

[4] W. Zhang, X. Huang, N. Chen, W. Wang, and H. Zhong, "Paas-oriented performance modeling for cloud computing," in 2012 IEEE 36th Annual Computer Software and Applications Conference, pp. 395–404, 2012.

[5] "Sp 800-53 rev. 3. recommended security controls for federal information systems and organizations," tech. rep., Joint Task Force Transformation Initiative, Gaithersburg, MD, USA, 2009.

[6] X. Weiquan and W. Houkui, "The design research of data security model based on public cloud," in 2013 Ninth International Conference on Computational Intelligence and Security, pp. 607–609, 2013.

[7] N. Kajal, N. Ikram, and Prachi, "Security threats in cloud computing," in International Conference on Computing, Communication Automation, pp. 691–694, 2015.

[8] W. Wu, Q. Zhang, and Y. Wang, "Public cloud security protection research," 2019 IEEE International Conference on Signal Processing, Communications and Computing (ICSPCC), pp. 1–4, 2019.

[9] "Cloud computing: Benefits, risks and recommendations for information security," tech. rep., European Network and Information Security Agency (ENISA), December 2012.

[10] M. Satyanarayanan, "The emergence of edge computing," Computer, vol. 50, no. 1, pp. 30–39, 2017.

[11] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," IEEE Internet of Things Journal, vol. 3, no. 5, pp. 637–646, 2016.

[12] W. Shi and S. Dustdar, "The promise of edge computing," Computer, vol. 49, no. 5, pp. 78–81, 2016.

[13] "Edge computing architecture – ibm,"

[14] Edge Computing From Hype to Reality.

[15] Huawei, IBM, Nokia, Intel, NTT, and Vodafone, "Mobile-edge computing," 2014.

[16] T. H. Luan, L. Gao, Z. Li, Y. Xiang, G. Wei, and L. Sun, "Fog computing: Focusing on mobile users at the edge," 2016.

[17] N. Abbas, Y. Zhang, A. Taherkordi, and T. Skeie, "Mobile edge computing: A survey," IEEE Internet of Things Journal, vol. 5, no. 1, pp. 450–465, 2018.

[18] R. Roman, J. Lopez, and M. Mambo, "Mobile edge computing, fog et al.: A survey and analysis of security threats and challenges," Future Generation Computer Systems, vol. 78, pp. 680–698, 2018.

[19] I. Stojmenovic, S. Wen, X. Huang, and H. Luan, "An overview of fog computing and its security issues," vol. 28, no. 10, 2016.

[20] C. Puliafito, E. Mingozzi, and F. L. et al., "Fog computing for the internet of things: A survey," CM Transactions on Internet Technology.

[21] M. Chiang and T. Zhang, "Fog and iot: An overview of research opportunities," IEEE Internet of Things Journal, 2016.

[22] K. Dolui and S. K. Datta, "Comparison of edge computing implementations: Fog com- puting, cloudlet and mobile edge computing," Global Internet of Things Summit (GIoTS), 2017.

[23] N. Mohamed, J. Al-Jaroodi, S. Lazarova-Molnar, I. Jawhar, and S. Mahmoud, "Ieee smartworld, ubiquitous intelligence computing, advanced trusted computed, scalable computing communications, cloud big data computing, internet of people and smart city innovation," 2017.

[24] Z. Mann, "Notions of architecture in fog computing," 2021.

[25] "Nebbiolo architecture,"

[26] D. Bruneo, S. Distefano, F. Longo, G. Merlino, A. Puliafito, V. D'Amico, M. Sapienza, and G. Torrisi, "Stack4things as a fog computing platform for smart city applications," IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), 2016.

[27] A. Souza, L. Izidio, A. Rocha, and N. C. T. Batista, "Sapparchi: An architecture for smart city applications from edge, fog and cloud computing," 2019 IEEE International Smart Cities Conference (ISC2), 2019.

[28] Wortmann, Felix, Flüchter, and Kristina, "Internet of things - technology and value added," vol. 57, 2015.

[29] L. D. Xu, W. He, and S. Li, "Internet of things in industries: A survey," IEEE Transactions on Industrial Informatics, vol. 10, no. 4, pp. 2233–2243, 2014.

[30] E. Bugnion, J. Nieh, and D. Tsafrir. Morgan and Claypool Publishers, 2017.

[31] J. Yiu, The Definitive Guide to the ARM Cortex-M3. 2009.