UNIVERSITÀ DI PARMA

Dipartimento di Ingegneria e Architettura

Corso di Laurea in Ingegneria Informatica, Elettronica e delle Telecomunicazioni

Titolo Italiano

English Title

Relatore:

Chiar.mo Prof. Michele Amoretti

Correlatore:

Dott. Ing. Gabriele Penzotti

Tesi di Laurea di:

Filippo Scaramuzza

ANNO ACCADEMICO 2020-2021

Ai miei genitori.

Ringraziamenti

Indice

Introduzione

Capitolo 1

Stato dell'arte

1.1 Cloud Computing nell'era dei Big Data

Il NIST (*National Institute of Standards and Technology*) definisce il *Cloud Computing* come un modello che promuove l'accesso globale alle risorse informatiche condivise, tipicamente *on-demand* [1]. L'infrastruttura di questo paradigma, nella sua versione più semplice, è relegata principalmente in *data center*, ovvero dei raggruppamenti di risorse virtualizzate altamente accessibili che possono essere riconfigurate dinamicamente per garantire la scalabilità dei servizi. Questi fungono da nodi centrali e garantiscono agli utenti un'infrastruttura, una piattaforma oppure un servizio software utile per le loro applicazioni e i propri scopi (IaaS, Paas, SaaS).

Nonostante il *Cloud Computing* abbia indubbiamente un ruolo chiave nel rendere accessibile una potenza di calcolo, altrimenti troppo difficile da poter essere realizzata in proprio, nella moderna era dei Big Data il tempo richiesto per accedere ad alcune applicazioni *cloud-based*, che concentrano l'intera elaborazione dei dati nei suddetti *data-center*, potrebbe essere troppo elevato e rendere questo paradigma impraticabile per applicazioni *real-time* o, in generale, ovunque la latenza debba essere ridotta al minimo. Inoltre l'ormai noto incremento dei dispositivi connessi in ambito IoT (*Internet of Things*)

ed il relativo rapido aumento dei dati generati nell'*edge* [1] della rete richiedono che le risorse di calcolo siano geograficamente situate il più vicino possibile ai dispositivi stessi, così da diminuire al massimo la latenza, aumentando di conseguenza il *throughput* della rete.

Per affrontare queste problematiche è quindi necessario garantire il cosiddetto *Cloud-to-Thing Continuum*, ovvero la possibilità di rendere disponibili potenza computazionale, di storage e di networking ovunque nella rete, dal cloud agli end-nodes. In questo ambito sono state avanzate numerose proposte, ad esempio il *Fog Computing*, sia in ambito industriale che accademico [2, 3].

1.2 Fog Computing ed altri paradigmi nel Cloud-to-Thing Continuum

1.2.1 Fog Computing e architettura OpenFog

Con *Fog Computing* si intende un'architettura a livello di sistema che distribuisce le funzioni di elaborazione, archiviazione, controllo e rete più vicine agli utenti lungo un *Cloud-to-Thing Continuum* [2].

Il *Fog Computing* dunque è innanzitutto caratterizzato da un approccio distribuito. Ciò deriva dal bisogno di superare i limiti dell'approccio centralizzato del Cloud Computing, come latenza, privacy e sovraccarico dei dati. In secondo luogo i nodi Fog, possono essere posizionati ovunque nella rete tra gli end-node e il cloud. Questa flessibilità che contraddistingue il Fog Computing, sottolinea la visione di questo paradigma non come una sostituzione, bensì come un'estensione del Cloud Computing, con lo scopo di colmare il divario tra quest'ultimo e i dispositivi IoT, garantendo quindi il continuum *Cloud-to-Thing*.

[1] Con *edge* si intende la zona perimetrale della rete, in cui risiedono gli end-node. L'edge è caratterizzato da un punto di demarcazione (ad esempio un *gateway*) che lo separa dal *network core*, ovvero la parte centrale gestita dagli Internet Service Provider.

Ad esempio, in un'applicazione che si occupa di analisi di Big Data prodotti da migliaia di dispositivi IoT, lo strato di Fog (dall'inglese Fog, *Nebbia*) che si pone ad un livello "più basso", ovvero più vicino ai dispositivi, rispetto al Cloud (dall'inglese Cloud, *Nuvola*) potrebbe svolgere una funzione di filtraggio, pre-elaborazione e aggregazione del flusso dati, rendendo eventualmente disponibili alcuni risultati ai livelli inferiori e alleggerendo il carico nel Cloud a cui rimarrebbero i task più complessi ma su una mole di dati molto ridotta e pre-elaborata.

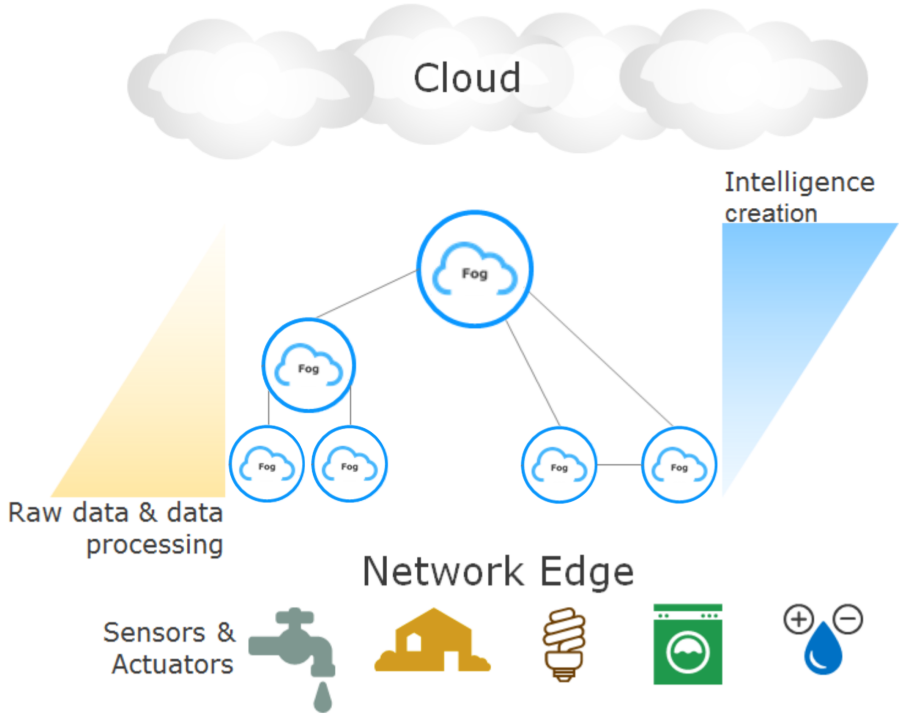

Figura 1.1: Architettura OpenFog N-Tier [2]

Negli anni sono state proposte alcune architetture di particolare rilievo per scenari di Fog Computing. Molte ricerche fanno riferimento ad un'architettura a 3 livelli, composta da Cloud, Fog e IoT [4]. Per questo lavoro di Tesi l'architettura di riferimento è quella fornita dall'*OpenFog Consortium*. Quest'ultimo è stato fondato da ARM, Cisco, Dell, Intel, Microsoft e dall'Università di Princeton, nel 2015. Ad oggi OpenFog e i suoi membri fanno

parte dell'*IIC* (*Industrial Internet Consortium*).

L'architettura di riferimento è detta *N-Tier Architecture*, il cui schema è mostrato in Figura 1.1. Questa mantiene comunque una struttura composta da tre macro-entità: il cloud, il livello Fog e gli end-node/dispositivi IoT. Il livello Fog, però, è ulteriormente scomposto in sotto-livelli (*tier*) che più si allontanano dai dispositivi, più aumentano le loro capacità computazionali. Il numero di livelli Fog da adottare dipende dai requisiti dello scenario secondo diversi parametri, ad esempio il numero di dispositivi IoT, il carico di lavoro, le capacità dei nodi ad ogni livello, i requisiti di latenza minima e così via. Inoltre i nodi Fog in ogni tier possono essere collegati tra loro formando una maglia capace di fornire caratteristiche aggiuntive, come resilienza, tolleranza ai guasti, bilanciamento del carico e così via. Questo significa che i nodi sono in grado di comunicare sia orizzontalmente che verticalmente all'interno dell'architettura Fog.

I vantaggi fondamentali dell'architettura di riferimento definita dall'OpenFog Consortium sono riassunti con il termine *SCALE* [2], ovvero:

- **Security**. Sicurezza aggiuntiva per garantire transazioni sicure e affidabili. Dal *Security Pillar* dell'architettura di riferimento, si evince la necessità di uno o più "nodi fidati" (*Root of Trust*) quando si trattano dati sensibili.

- **Cognition**. L'infrastruttura Fog è consapevole dei requisiti e degli obiettivi delle applicazioni, quindi distribuisce le capacità di elaborazione, comunicazione, controllo e archiviazione lungo il continuum Cloud-to-Things, creando applicazioni che soddisfano meglio le esigenze specifiche dello scenario.

- **Agility**. Lo sviluppo di un nuovo servizio è solitamente lento e costoso, a causa dei costi e dei tempi necessari ai grandi fornitori per avviare o adottare l'innovazione. Il Fog Computing, invece, offre innovazione rapida e scalabilità conveniente, in cui individui e piccoli team possono utilizzare strumenti di sviluppo aperti (ad esempio API e SDK, secon-

do il pilastro *Openness* definito da OpenFog [2]) e la proliferazione di dispositivi IoT per offrire nuovi servizi.

- **Latency**. L'architettura Fog supporta l'elaborazione e l'archiviazione dei dati vicino all'utente, con conseguente bassa latenza. Pertanto, il Fog computing soddisfa perfettamente la richiesta di elaborazione in tempo reale, in particolare quindi per le applicazioni *real-time*.

- **Efficiency**. Riconoscimento e condivisione delle risorse inutilizzate dei dispositivi finali che partecipano al networking.

1.2.2 Possibili applicazioni del Fog Computing

Smart Cars e Traffic Control

I veicoli smart (*Smart Cars*) sono in grado di produrre giornalmente svariati terabyte di dati grazie alla combinazione di sensori *LIDAR* [2], dei sensori GPS, delle videocamere intelligenti a bordo dei veicoli per il riconoscimento delle immagini e così via. Oltre ai dati prodotti dalle *Smart Cars* ci sono quelli generati dai sistemi di controllo intelligenti, ovvero l'infrastruttura del *Traffic Control* (semafori intelligenti, sensori di rilevamento del traffico e delle congestioni, videocamere, ecc.). È evidente come un modello *cloud-only* non sia adatto, vista l'enorme mole di dati, a garantire un throughput sufficientemente elevato. Un'architettura Fog potrebbe invece soddisfare le particolari esigenze di questo scenario. In quest'ultimo, i veicoli sono nodi Fog che comunicano tra loro, o con i sistemi di controllo. In particolare i nodi Fog a bordo delle *Smart Cars* possono offrire i normali servizi di *infotainment*, ADAS [3], guida autonoma, sistemi anticollisione, navigazione, e così via, tramite la connessione all'infrastruttura Fog con l'ausilio di diverse

[2]LIDAR (*Light Detection and Ranging*) è un metodo per determinare la presenza di oggetti, la loro forma e la loro distanza dall'osservatore utilizzando un laser e misurando il tempo necessario alla luce riflessa per tornare al trasmettitore laser.

[3]ADAS (*Advanced Driver Assistance Systems*) è un insieme di tecnologie che assistono il guidatore nelle operazioni di guida e parcheggio in modo sicuro.

tecnologie, come DSRC (*Dedicated Short Range Communications*) o le reti cellulari (3G, LTE, 5G, etc.).

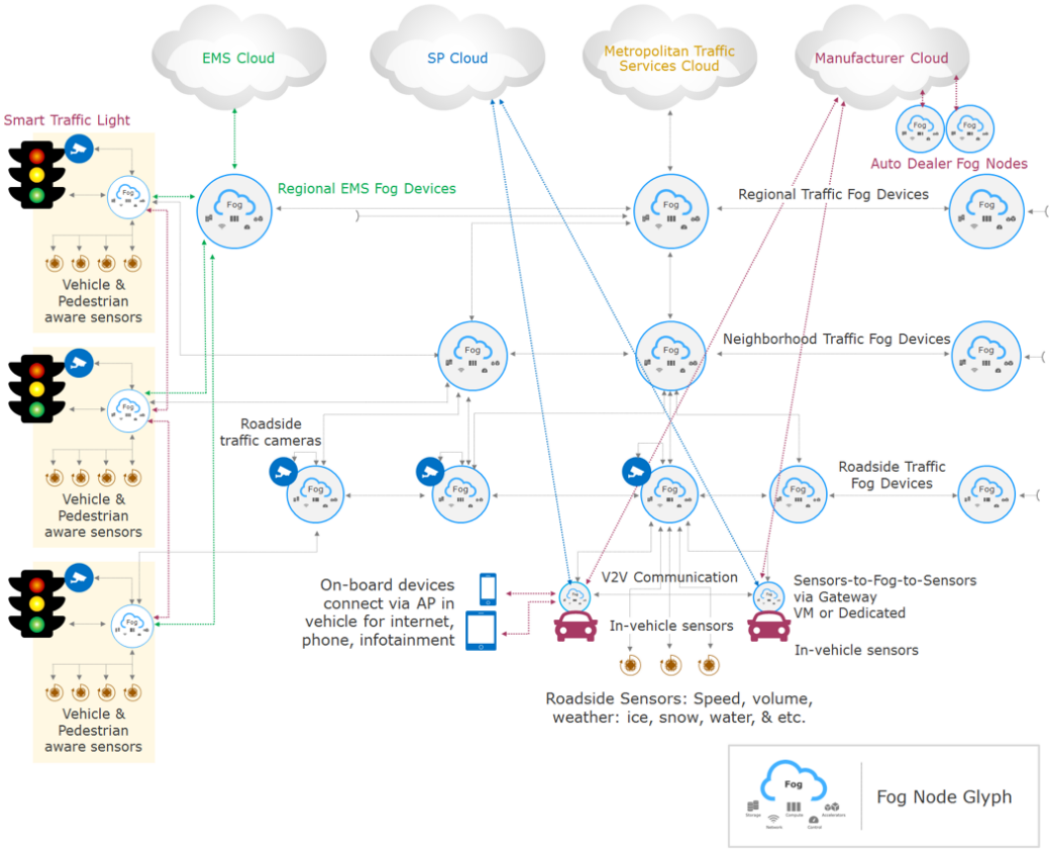

Figura 1.2: Architettura di uno scenario Fog in ambito Smart Cars e Traffic Control [2]

L'architettura di riferimento, mostrata in Figura 1.3, è strutturalmente gerarchica, con 3 livelli di nodi Fog. Il primo livello (*Roadside Fog Nodes*) si occupa di raccogliere i dati dai vari sensori e telecamere. I nodi Fog in questo livello eseguono alcune veloci analisi, utili ad esempio a comunicare ai veicoli in transito particolari condizioni del traffico o della strada. I dati aggregati dal primo livello sono inviati al secondo e/o al terzo livello (non c'è necessariamente una gerarchia nelle comunicazioni, questo per definizione dell'architettura Fog e della sua flessibilità), ovvero i *Neighborhood Fog Nodes* e i *Regional Fog Nodes*. In genere, ogni livello Fog nella gerarchia

fornirà ulteriori capacità di elaborazione, storage e rete. Ad esempio, livelli di gerarchicamente più alti garantiranno un trattamento aggiuntivo per fornire analisi dei dati o capacità di archiviazione di grandi dimensioni, utili ad esempio per analisi sul lungo periodo o per inviare dati ad altre zone della rete stradale per particolari task.

Smart Cities e Smart Buildings

Un altro interessante impiego del Fog Computing è nell'ambito delle *Smart Cities* e degli *Smart Buildings*. Infatti sebbene la maggior parte delle città moderne disponga di una o più reti cellulari che forniscono una copertura dell'intera città, queste reti hanno spesso limiti di capacità e larghezza di banda che soddisfano appena le esigenze degli attuali abbonati. Ciò lascia poca larghezza di banda per i servizi più avanzati previsti in una città intelligente nell'era dei Big Data, come *Smart Parking*, *Traffic Control*, ospedali intelligenti, controllo sul consumo elettrico e così via.

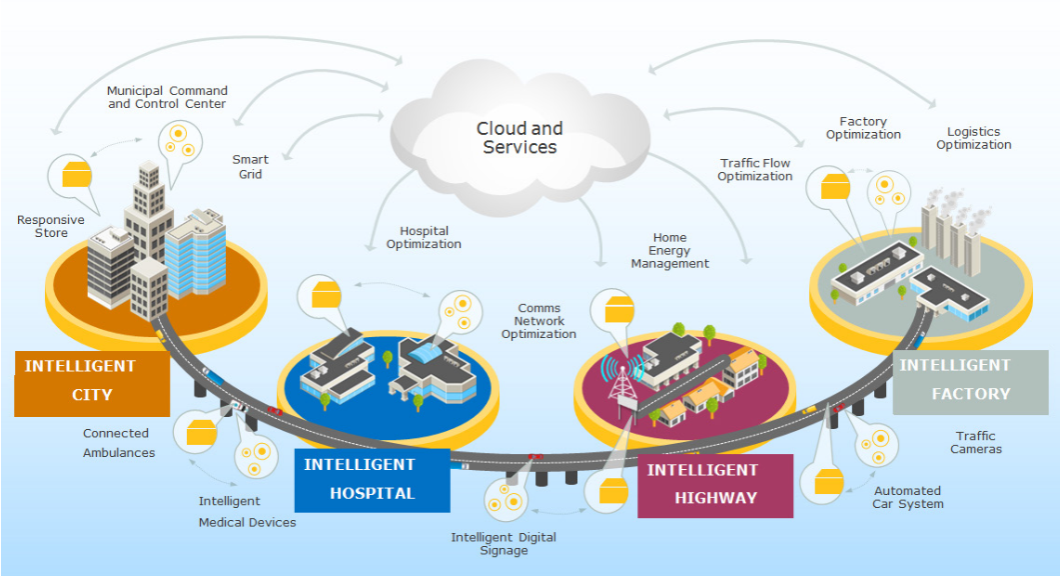

Figura 1.3: Opportunità per una *Smart City* [2]

Questo caso d'uso è già realtà, ad esempio, nella città di Barcellona, dove i risultati hanno dimostrato che il Fog Computing è la possibile chiave di

volta per l'implementazione di servizi molto avanzati e complessi, in città che producono decine di milioni di Gigabyte di dati giornalmente [5].

1.2.3 Altri Paradigmi

Per analizzare i vari altri paradigmi che possono essere considerati un'estensione, più che un'alternativa al Fog Computing, è utile introdurre il concetto di *Edge Computing*. Quest'ultimo è un un paradigma nato dall'esigenza di spostare la capacità di calcolo verso l'edge [4] della rete. In [6] viene definito l'Edge Computing come *"tecnologie abilitanti che consentono di effettuare calcoli ai margini della rete sui dati, a valle per conto di servizi cloud e a monte per conto di servizi IoT"*. L'idea è quella di estendere le capacità dal cloud all'edge della rete, con l'obiettivo di portare la potenza di calcolo il più vicino possibile ai generatori dei dati, ovvero ai dispositivi IoT. Nonostante sia il Fog Computing che l'Edge Computing muovano le capacità computazionali vicino agli end-node, OpenFog afferma che l'Edge Computing venga erroneamente chiamato Fog Computing (e viceversa): la fondamentale distinzione sta nel fatto che il Fog Computing è strutturalmente gerarchico e fornisce potenza computazionale, networking e storage ovunque nella rete, dal Cloud agli end-node (*Cloud-to-Thing Continuum*), mentre l'Edge Computing tende ad essere limitato all'edge [2].

I principi fondanti dell'Edge Computing possono essere messi in pratica in diversi modi, in termini di tipo di dispositivi utilizzati, protocolli di comunicazione adottati e così via. I principali paradigmi sono esposti nel seguito.

Mobile Cloud Computing e Cloudlet Computing

Il *Mobile Cloud Computing* (MCC) è basato sul concetto del *mobile offloading*: l'idea alla base, per un dispositivo mobile, è quella di delegare, quando

[4] Con *edge* si intende la zona perimetrale della rete, in cui risiedono gli end-node. L'edge è caratterizzato da un punto di demarcazione (ad esempio un *gateway*) che lo separa dal *network core*, ovvero la parte centrale gestita dagli Internet Service Provider.

possibile, storage e calcoli ad entità remote (ad esempio il cloud) in modo da ridurre il carico di lavoro e ottimizzare il consumo di energia. In realtà oggi il concetto di MCC è stato esteso tenendo in considerazione i principi dell'Edge Computing. La nuova interpretazione del MCC è quella di delegare l'elaborazione e lo storage dei dati a dispositivi situati all'edge della rete, piuttosto che al cloud. L'implementazione più comune di questa visione è il *Cloudlet Computing* (CC), che consiste nell'utilizzare dei *cloudlet* [5] per eseguire elaborazione ed archiviazione dei dati vicino ai dispositivi finali.

Multi-access Edge Computing

Esattamente come il *Mobile Cloud Computing* è un'estensione del *Mobile Computing* attraverso il *Cloud Computing*, analogamente, il *Multi-access Edge Computing* (MEC) è un'estensione del *Mobile Computing* attraverso l'*Edge Computing*. In [7] il MEC viene definito come una piattaforma che fornisce funzionalità IT e di cloud computing all'interno della *Radio Access Network* (RAN) in 4G e 5G, in prossimità dei dispositivi mobili. Il Multi-access Edge Computing è stato precedentemente definito come *"Mobile Edge Computing"*, ma il paradigma è stato ampliato per includere una più ampia gamma di applicazioni oltre alle attività specifiche per dispositivi mobili. Esempi di applicazioni di MEC includono analisi video, *Smart Cars*, monitoraggio della salute e realtà aumentata.

Mist Computing

Spesso chiamato anche *IoT Computing*, il *Mist Computing* viene impiegato per raggiungere l'edge più estremo dei dispositivi connessi (micro-controllori e sensori) [8]. Il Mist Computing può essere visto come la prima posizione di calcolo nel continuum IoT-Fog-Cloud. Infatti può aiutare a conservare la larghezza di banda e la carica della batteria poiché sono solo i dati es-

[5] Un *cloudlet* è una piccola infrastruttura cloud *affidabile*, situata nell'edge della rete disponibile per i dispostivi mobili vicini, che collabora con il cloud per servire i dati in modo più efficiente.

senziali ad essere trasmessi al gateway, al server o al router. Inoltre il Mist Computing offre l'utilizzo di meccanismi di controllo dell'accesso ai dati che possono garantirne la riservatezza a livello locale. Di contro, questo paradigma ha che i micro-controllori e i sensori utilizzati nell'infrastruttura possono essere utilizzati solo per un'elaborazione leggera, pertanto il loro utilizzo è ristretto ad un numero piuttosto esiguo di applicazioni, a meno dell'implementazione dello stesso nel continuum Clout-to-Thing venendo quindi incluso in un'architettura più ampia, basata ad esempio sul Fog Computing [9].

1.3 Simulatori di Fog Computing, YAFS

La progettazione e l'ottimizzazione di sistemi distribuiti su larga scala, richiedono una descrizione realistica del flusso dei dati, della mole enorme di richieste emesse dai nodi, dei servizi disponibili e ogni aspetto necessario alla comprensione del comportamento di sistemi che scambiano dosi massicce di dati. La via più semplice per comprendere le potenzialità e allo stesso tempo i limiti di scenari così complessi è quella della *simulazione*.

La ricerca nell'ambito del Fog Computing vanta un numero consistente di strumenti di simulazione più o meno avanzati. Tra i più noti vi sono *iFogSim* [6] [10] ed *EdgeCloudSim* [7] [11], entrambi basati su *CloudSim*, un simulatore di architetture cloud. Oltre alla simulazione un altro importate strumento che garantisce esperimenti ripetibili e controllabili è l'*emulazione*. Tra i software più rilevanti esistono *EmuFog* [8] [12] e *FogBed* [9] [13]. Entrambi consentono all'utente di progettare scenari Fog con applicazioni basati su Docker o Mininet.

[6] Disponibile su: `https://github.com/cloudslab/ifogsim`
[7] Disponibile su: `https://github.com/CagataySonmez/EdgeCloudSim`
[8] Disponibile su: `https://github.com/emufog/emufog`
[9] Disponibile su: `https://github.com/fogbed/fogbed`

1.3.1 YAFS, *Yet Another Fog Simulator*

Per questo lavoro di tesi, è stato utilizzato, tra le altre cose, il simulatore *YAFS* [10] (*Yet Another Fog Simulator*), un simulatore ad eventi discreti sviluppato in Python e basato su *SimPy*, ovvero un framework DES (*Discrete Event Simulator*) basato sui processi anch'esso sviluppato in Python [14]. YAFS è progettato per analizzare progettare applicazioni in scenari Fog e incorpora le strategie per il placement, lo scheduling e l'instradamento dei dati. I punti chiave che hanno portato alla scelta di YAFS sono esposti nel seguito:

- **Placement, Scheduling, Routing e processi personalizzati**. L'algoritmo di placement viene invocato all'avvio e viene eseguito durante la simulazione secondo una distribuzione personalizzata. L'algoritmo di routing sceglie il percorso che collega il trasmettitore e il recettore e l'algoritmo di scheduling sceglie l'applicazione che deve eseguire il task associato alla richiesta. Oltre agli algoritmi appena esposti che possono essere definiti dall'utente, quest'ultimo può implementare funzioni personalizzate che possono essere invocate in fase di esecuzione per fornire implementazioni flessibili di eventi reali come il movimento delle fonti del carico di lavoro, la generazione di guasti della rete e la raccolta di dati specifici utilizzando anche applicazioni di terze parti.

- **Creazione dinamica delle sorgenti dei messaggi**. Ogni fonte di carico di lavoro rappresenta la connessione alla rete di un utente, di un sensore IoT o di un attuatore che richiede un servizio. Ogni sorgente è associata ad un'entità DES di rete che genera richieste secondo una distribuzione personalizzata. Le sorgenti del carico di lavoro possono essere create, modificate o rimosse dinamicamente, consentendo la modellazione dei movimenti degli utenti in un generico ecosistema.

- **Topologia della rete**. YAFS basa la struttura della topologia sulla *Complex Network Theory*, grazie all'implementazione della libreria

[10] Disponibile su: `https://github.com/acsicuib/YAFS`

NetworkX, da cui deriva la possibilità di applicare tutti gli algoritmi che ne derivano, ottenendo quindi indicatori di maggior interesse sulle topologie adottate.

- **Risultati**. YAFS esegue la registrazione automatica basata su CSV di due tipi di eventi:
  - generazione del carico di lavoro e del calcolo eseguito su di esso
  - trasmissioni dei messaggi sui collegamenti.

  I risultati sono salvati in formato *raw* con un impronta noSql con l'idea che da dati in formati più semplici derivino analisi più veloci.

Capitolo 2

Architettura Funzionale del Sistema Realizzato

2.1 Simulazione di sistemi distribuiti

Simulazioni per il design e l'ottimizzazione di sistemi distribuiti su larga scala, come e perché.

2.2 Descrizione dello scenario simulato

2.2.1 Fog Computing in ambito Precision Agriculture

Introduzione generica alla precision agriculture e motivi dietro alle scelte progettuali nonché al paradigma Fog.

2.2.2 Topologia ed architettura di riferimento

Spiegazione della struttura a livelli e descrizione dei singoli livelli, anche facendo riferimento alla questione relativa alla privacy.

Capitolo 3

Implementazione

3.1 Descrizione generale delle simulazioni

Schema generale con le varie fasi dell'esecuzione delle simulazioni: dalla generazione della rete, al placement, alle simulazioni e alle analisi.

3.1.1 Il simulatore YAFS

Descrizione del simulatore e delle componenti principali (topologia, applicazioni e servizi, users e così via).

3.2 Realizzazione del software di simulazione

Riferimenti a python e conda, sull'ambiente di sviluppo e metodologia utilizzata.

3.2.1 Implementazione della generazione del grafo

Descrizione della generazione del grafo (strutture interne dei livelli, fattori di riduzione, ecc..)

3.2.2 Implementazione dell'algoritmo di placement

Funzionamento dell'algoritmo con spiegazioni step-by-step

3.2.3 Implementazione della simulazione

Come nelle repo GitHub, spiegazione delle varie parti che compongono il core della simulazione con YAFS.

Capitolo 4

Risultati

4.1 Simulazioni

Qui elenco delle varie simulazioni con spiegazione dei risultati ed eventuali conclusioni che da essi si ottengono.

Conclusioni

Appendice A

Appendice

Bibliografia

[1] P. Mell and T. Grance. The nist definition of cloud computing. *NIST Special Pubblication 800-145*, 2011.

[2] OpenFog Consortium. *OpenFog Reference Architecture for Fog Computing*, 2017.

[3] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli. Fog computing and its role in the internet of things. In *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*, 2012.

[4] Jie Lin, Wei Yu, Nan Zhang, Xinyu Yang, Hanlin Zhang, and Wei Zhao. A survey on internet of things: Architecture, enabling technologies, security and privacy, and applications. *IEEE Internet of Things Journal*, 2017.

[5] J. Garcia, E. Simó, X. Masip-Bruin, E. Marín-Tordera, and S. Sànchez-López. Do we really need cloud? estimating the fog computing capacities in the city of barcelona. In *2018 IEEE/ACM International Conference on Utility and Cloud Computing Companion (UCC Companion)*, 2018.

[6] Weisong Shi, Jie Cao, Quan Zhang, Youhuizi Li, and Lanyu Xu. Edge computing: Vision and challenges. *IEEE Internet of Things Journal*, 2016.

[7] F. Giust. Mec deployments in 4g and evolution towards 5g. *MEC Deployments in 4G and Evolution Towards 5G*, 2018.

[8] J.S. Preden, K. Tammemae, A. Jantsch, M. Leier, A. Riid, and E. Calis. The benefits of self-awareness and attention in fog and mist computing. *Computer*, 2015.

[9] Manas Kumar Yogi, K. Chandra sekhar, and G. Vijay Kumar. Mist computing: Principles, trends and future direction. *International Journal of Computer Science and Engineering*, 4, 2017.

[10] Harshit Gupta, Amir Vahid Dastjerdi, Soumya K. Ghosh, and Rajkumar Buyya. ifogsim: A toolkit for modeling and simulation of resource management techniques in the internet of things, edge and fog computing environments. *Software: Practice and Experience*, 47, 2017.

[11] Cagatay Sonmez, Atay Ozgovde, and Cem Ersoy. Edgecloudsim: An environment for performance evaluation of edge computing systems. In *2017 Second International Conference on Fog and Mobile Edge Computing (FMEC)*, 2017.

[12] Robson Mayer, Menaouar Berrehil El Kattel, Maicon Douglas Possamai, Claudio Bruning, and Sérgio Vidal Garcia Oliveira. Analysis of a multiphase interleaved bidirectional dc/dc power converter with coupled inductor. In *2017 Brazilian Power Electronics Conference (COBEP)*, 2017.

[13] Antonio Coutinho, Fabiola Greve, Cassio Prazeres, and Joao Cardoso. Fogbed: A rapid-prototyping emulation environment for fog computing. In *2018 IEEE International Conference on Communications (ICC)*, 2018.

[14] I. Lera, C. Guerrero, and C. Juiz. Yafs: A simulator for iot scenarios in fog computing. *IEEE Access*, 2019.

[15] Q. Wu, J. Shen, B. Yong, J Wu, F. Li, J. Wang, and Q. Zhou. Smart fog based workflow for traffic control networks. *Future Generation Computer Systems*, 2019.

[16] NIST. Mobile cloud computing. [online]. *Technical Report, National Institute of Standards and Technology.* `https://www.nist.gov/programs-projects/mobile-cloud-computing`.

[17] R. Román, Javier López, and M. Mambo. Mobile edge computing, fog et al.: A survey and analysis of security threats and challenges. *Future Gener. Comput. Syst.*, 2018.