



UNIVERSITÀ DEGLI STUDI DI FIRENZE
SCUOLA DI INGEGNERIA - DIPARTIMENTO DI INGEGNERIA
DELL'INFORMAZIONE

Tesi di Laurea Triennale in Ingegneria Informatica

**APPLICAZIONE DEL PARADIGMA SDN-IoT A
SISTEMI DI INDUSTRIA 4.0**

Candidato
Lorenzo Pesci

Relatore
Prof. Francesco Chiti

Correlatore
Dott. Michele Bonanni

Anno Accademico 2020-2021

Indice

Introduzione	i
1 Stato dell'arte	1
1.1 Wireless Sensor Network	1
1.1.1 Topologia di una WSN	2
1.2 Software Defined Networking	4
1.3 SDN-WISE	4
1.3.1 Approccio SDN-WISE	4
2 Internet of Things	7
2.1 Things of the Internet of Things	7
2.2 Evoluzione	8
2.3 Strati dell'Internet of Things	8
3 Requisiti e Tecnologie	10
3.1 Routing	10
3.1.1 Caratteristiche	10
3.1.2 Packet Forwarding - Inoltro Pacchetti	11
3.1.3 Routing Protocols	12
3.1.4 Elementi del Router	14
3.2 SDN e NFV	15

3.2.1	Software Defined Networking	15
3.2.2	Network Function Virtualization	18
4	SDN: Background and Motivation	20
4.1	Requisiti dell’Evoluzione Network	20
4.1.1	Incremento Della Domanda	20
4.1.2	Fornitura in Aumento	21
4.1.3	Traffic Pattern molto complessi	21
4.1.4	Le Architetture di Rete Tradizionali sono Inadeguate .	22
4.2	L’Approccio SDN	24
4.2.1	Requisiti	24
4.2.2	Architettura SDN	25
4.2.3	Caratteristiche dell’SDN	28
4.3	SDN- e NFV-Related Standard	28
4.3.1	Standard Developing Organizations	29
4.3.2	Industry Consortia	30
4.3.3	Open Development Initiatives	31
5	SDN Data Plane and OpenFlow	32
5.1	SDN Data Plane	32
5.1.1	Funzioni del Data Plane	32
5.1.2	Protocollo del Data Plane	33
5.2	OpenFlow	33
5.2.1	Flow Table Structure	35
5.2.2	Flow Table Pipeline	35
5.2.3	Group Table	36
5.3	Protocollo OpenFlow	37

6 SDN Control Plane	38
6.1 SDN Control Plane Architecture	38
6.1.1 Funzioni del Control Plane	38
6.1.2 Southbound Interface	39
6.1.3 NorthBound Interfaces	39
6.1.4 Routing	40
6.2 ITU-T Model	42
6.3 Rest	43
6.3.1 Esempio di REST API	44
6.4 Cooperazione e coordinazione fra Controllers	45
6.4.1 Controller Centralizzati vs Controller Distribuiti	45
6.4.2 Border Gateway Protocol	46
7 The Internet of Things: Components	48
7.1 L'inizio dell'era IOT	48
7.2 Gli Scopi dell'Internet of Things	49
7.3 Componenti di oggetti abilitati per l'IoT	50
7.3.1 Sensori	50
7.3.2 Attuatori	52
7.3.3 Microcontrollers	52
7.3.4 Transceivers	53
7.3.5 RFID	54
8 The Internet of Things: Architecture and Implementation	55
8.1 IoT Architecture	55
8.1.1 ITU-T IoT Reference Model	55
8.1.2 IoT World Forum Reference Model	58
8.2 IoT Implementation	59

8.2.1	Cisco IoT System	59
9	Architettura del sistema	62
9.1	Attori partecipanti	63
9.1.1	Controller	63
9.1.2	Sink	66
9.1.3	Mote	66
10	Framework utilizzati	68
10.1	Contiki	68
10.2	Cooja	69
10.3	IntelliJ IDEA e Linux Ubuntu 14.04	73
11	Simulazioni	75
11.1	Scenari Analizzati	75
11.1.1	Scenario Caso Ottimo	76
11.1.2	Scenario Caso Pessimo	78
11.1.3	Scenario Caso Intermedio - Restart	80
11.1.4	Scenario Caso Intermedio - Reload	82
11.2	Risultati delle simulazioni	85
11.2.1	Misurazione del Ritardo	85
11.2.2	Quantità prodotte	90
12	Conclusioni	91
	Bibliografia	93

Introduzione

È un'opinione condivisa che il successo del paradigma Industry 4.0 (I4.0) sarà abilitato dall'uso sinergico dell'internet of Things (IoT), le comunicazioni Machine to Machine (M2M) e i Cyber Physical System (CPS). A riguardo, è doveroso premettere, tuttavia, che esistono numerose limitazioni legate alle architetture protocollari che le reti di telecomunicazioni tradizionali non sono in grado di risolvere compiutamente e dinamicamente. Il Software Defined Networking (SDN) è, invece, un approccio recentemente introdotto capace di affrontare queste problematiche in una prospettiva integrata. [1]

Specificatamente, SDN, offrirà un efficace supporto per ambienti eterogenei in cui numerosi ed eterogenei smart devices cooperano tra loro per eseguire compiti complessi senza supervisione e assistenza manuale. Addirittura, i prodotti stessi saranno dei soggetti attivi partecipanti al processo decisionale e coinvolti nell'ottimizzazione della loro stessa produzione. Ciò permetterà di personalizzare i processi lavorativi in modo tale da accrescere la produttività, soddisfare i bisogni dei clienti, migliorare le tecnologie industriali ed aumentare l'efficienza energetica.

La crescita di interesse per l'Industria 4.0 (I4.0) sta motivando la ricerca di nuove tecnologie che possono aiutare la sua compiuta realizzazione. L'In-

ternet of Things (IoT), la comunicazione Machine to Machine (M2M) ed i Cypher Physical System (CPS) sono elementi essenziali per la realizzazione di questo paradigma innovativo, anche se sussistono problemi e limitazioni legati alla comunicazione che le attuali reti di comunicazioni non possono risolvere. Il Software Defined Networking (SDN) è un nuovo paradigma di networking che può gestire molte di queste problematiche.

Generalmente I4.0 rappresentano ambienti eterogenei in cui centinaia di smart devices, con caratteristiche diverse, cooperano tra loro per eseguire compiti complessi senza supervisione e assistenza o manuale. In particolare, i prodotti all'interno di queste Intelligent Industries non saranno più oggetti passivi che subiranno le scelte gestionali ma parteciperanno e diventeranno essenziali per il processo decisionale e per l'ottimizzazione della loro produzione. [2]

Il principale fine di tali manifatture è quello di integrare l'Information Technology negli impianti industriali e personalizzare i processi lavorativi in modo tale da accrescere la produttività, soddisfare i bisogni dei clienti, migliorare le tecnologie industriali ed aumentare l'efficienza energetica. Infatti l'obiettivo di questo progetto di tesi sarà proprio questo.

Alcune delle caratteristiche che le *Smart Industries* dovranno avere sono:

- **Personalizzazione di massa:** i processi di produzione dovranno soddisfare i vari requisiti degli ordini produttivi permettendo di includere individui nella progettazione ed abilitare modifiche anche all'ultimo minuto.
- **Flessibilità:** processi di produzione intelligenti ed auto-configuranti dovranno considerare diversi aspetti come il tempo, la qualità, i prezzi e gli aspetti ecologici.
- **Visibilità di fabbrica e processi di decisione ottimizzati:** IoT fornirà

una trasparenza end-to-end real time consentendo un'ottimizzazione all'interno dell'area di produzione e nella fabbrica stessa.

- **Catena di produzione connessa:** IoT aiuterà i produttori a comprendere meglio le informazioni sulla catena di produzione che potranno essere fornite in real time. Collegando le macchine e le attrezzature ai fornitori, tutte le parti potranno comprendere le interdipendenze, il flusso dei materiali ed i tempi del ciclo di produzione.

- **Gestione Energetica:** il miglioramento dell'efficienza energetica richiede la conoscenza dei livelli di consumo della linea di produzione e delle macchine. I contatori intelligenti potranno fornire dati in tempo reale e prendere decisioni in base alle loro capacità ed in collaborazione con servizi esterni.

- **Creare valore dai big data raccolti:** nuovi miglioramenti potranno essere ottenuti dall'analisi di grandi quantità di dati prodotti dai dispositivi IoT.

- **Monitoraggio remoto:** la tecnologia IoT consentirà il coinvolgimento di terzi (ad esempio fornitori) nel monitoraggio, nel funzionamento e nella manutenzione della fabbrica.

- **Manutenzione proattiva:** il monitoraggio e la valutazione delle prestazioni in tempo reale avranno un impatto positivo sul miglioramento della manutenzione proattiva.

L'adozione delle I4.0 con le caratteristiche sopra citate porta con sé ulteriori problemi soprattutto nel caso di fabbriche medio-grandi. Le motivazioni specifiche sono le seguenti:

- Migliaia di dispositivi IoT da gestire, di diversi fornitori e con diverse tecnologie potrebbero significare dozzine di strumenti e di interfacce utente da utilizzare per poterli gestire.

- L'eterogeneità dei dispositivi IoT potrebbe comportare protocolli di comunicazione e formati di dati diversi. Questo tradotto può essere interpretato come una mancanza di interoperabilità.
- Le reti convenzionali e le macchine industriali coesisteranno con le nuove infrastrutture IoT e tutte saranno "connesse" per raggiungere la piena adozione dell'I4.0.
- Il traffico dati nell'infrastruttura di rete industriale utilizzata potrebbe aumentare in modo significativo, causando ritardi e congestioni. Il traffico dati deve essere instradato in modo efficiente per evitarlo.

i primi due potrebbero essere risolti utilizzando una piattaforma Cloud IoT locale o remota, mentre le ultime due rimarrebbero parzialmente, o totalmente, irrisolti. Alcune proposte nella recente letteratura scientifica hanno delineato alcune soluzioni software per risolvere questi problemi ma non ha considerato l'eterogeneità dello scenario applicativo.

Invece, l'approccio SDN, attraverso la suddivisione netta tra Data Plane e Control Plane, può risolvere gran parte di queste problematiche e diventare la chiave abilitante per l'I4.0. Sebbene SDN sia stato inizialmente proposto per orchestrare reti IT, attualmente alcuni Controller SDN includono *plugin* per connettere le *Southbound interface* a dispositivi IoT e reti. Grazie alla sua modularità, SDN permette a chiunque di sviluppare sia nuovi plugin per quelle tecnologie coinvolte negli scenari industriali, sia innovative applicazioni per la gestione della rete. Inoltre, grazie alla proprietà di *clusterizzazione* che alcuni controller SDN possiedono (ONOS, OpenDayLight), è possibile ottenere un'altra scalabilità, affidabilità e tolleranza ai guasti.

I principali benefici di SDN che possono essere sfruttati dall'I4.0 sono riassunti di seguito:

- La gestione dei task è automatizzata e isolata dalla complessità dell'infrastruttura fisica attraverso interfacce *easy-to-use*.
- Nuovi servizi e applicazioni possono essere forniti in breve tempo; inoltre l'amministrazione IT ha la possibilità di programmare funzionalità e esercizi di rete eliminando la dipendenza dai costruttori hardware.
- Le politiche di routing su base flusso possono essere configurate e gestite per l'intera rete usando la stessa soluzione software.
- Le applicazioni possono sfruttare l'informazione centralizzata sullo stato della rete, reagendo in tempo reale ed eseguire cambiamenti aumentando le prestazioni della rete stessa.
- I costi operativi per la gestione della rete industriale sono significativamente ridotti.

Riassumendo, nel contesto delle I4.0 o *Industrial Internet of Things (IIOT)*, si rende necessario la gestione di una rete di sensori ed attuatori, eventualmente mobili, su una scala spaziale estesa ed in linea di principio, eterogenei. Ciò comporta alcune problematiche che le attuali reti di telecomunicazioni non riescono a gestire. L'approccio SDN, essendo stato studiato ed implementato per sopperire alle limitazioni delle reti tradizionali, rappresenta la soluzione ottimale per le esigenze di una fabbrica diffusa. Dato il numero elevato di dispositivi IoT che saranno impiegati nelle I4.0, risulta necessario l'adozione di un Control Plane distribuito (*cluster di controller SDN*) con specifici controller SDN, per ogni dominio omogeneo, interconnessi attraverso l'interfaccia East-WestBound.

Dato che tali controller rappresentano un punto di aggregazione dinamica

delle informazioni di rete, possono essere sfruttati da soluzioni di ML e da applicazioni per ottimizzare il processo produttivo e per reagire tempestivamente a situazioni di guasto. Tale architettura sarebbe in grado di supportare procedure di consenso distribuito naturalmente P2P sia tra controller sia tra dispositivi inter e intra domain. Inoltre, la proprietà di re-indirizzamento dinamico e programmabile dei flussi dati potrebbe essere sfruttata da un Network Function Orchestrator per il rapido dispiegamento, importazione, adattamento, migrazione e chaining di funzioni virtuali.

Un aspetto particolarmente interessante è rappresentato, infine, dalla possibilità di trasferire una parziale visione della rete agli Switch coinvolti, laddove questo potrebbe aiutare ad alleggerire le responsabilità del Controller limitandole alle condizioni operative più eccezionali e sfidanti, mentre gli Switch potrebbero autonomamente gestire contesti con un grado di adattività inferiore. In particolar modo, differentemente dai normali Switch stateless usati nelle tradizionali SDN, gli Switch utilizzati nelle IIoT potrebbero possedere uno stato che verrebbe modificato a seconda delle informazioni ricevute dai dispositivi IoT ad esso connessi. A tale stato corrisponderebbe un comportamento diverso da parte dello Switch che quindi, in modo dinamico ed automatico e senza la collaborazione del Controller, cambierebbe la sua politica di inoltro. Questo consentirebbe di aumentare la scalabilità della rete e diminuire maggiormente la probabilità di congestioni.

In questo progetto di tesi, il sistema é stato realizzato secondo il paradigma SDN, nello specifico é stato utilizzato SDN-WISE.

Capitolo 1

Stato dell'arte

Negli ultimi decenni, i progressi tecnologici nella miniaturizzazione dei dispositivi elettronici per l'elaborazione delle informazioni e i graduali processi di diffusione e di implementazione dei dispositivi di comunicazione wireless hanno contribuito notevolmente all'affermazione di una nuova realtà che facilmente si presta a numerosi impieghi: le reti di sensori wireless o WSN.

In questo capitolo vedremo come è caratterizzata una WSN e i suoi pregi e difetti. Vedremo il perché Software Defined Networking, il quale verrà approfondito maggiormente nei capitoli successivi in merito alla sua architettura e le sue funzionalità, è così importante per WSN e vedremo l'architettura che nasce dalla fusione di WSN con il protocollo. Infine verrà trattato un software simile al precedente: SDN-WISE, sul quale si basa il sistema realizzato.

1.1 Wireless Sensor Network

Con Wireless Sensor Network(WSN) si indica una topologia di rete informatica che, caratterizzata da una architettura distribuita, è costituita da un insieme di dispositivi elettronici autonomi in grado di prelevare dati dal-

l'ambiente circostante e di comunicare tra loro. In una rete WSN, ciascun sensore ha una regione di monitoraggio nel quale può monitorare eventi o oggetti nella regione. In aggiunta i sensori possono comunicare tra loro tramite interfacce che rientrano nella loro regione.

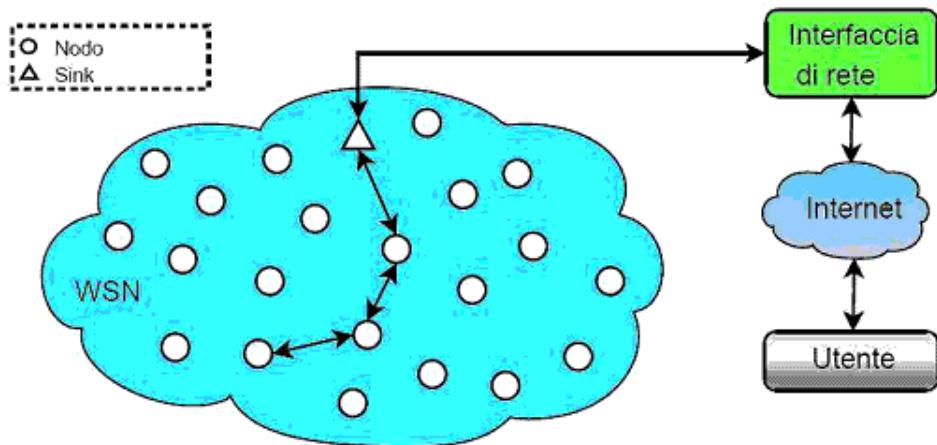


Figura 1.1: Rete WSN

La comunicazione avviene in maniera asimmetrica, in quanto i nodi inviano i dati monitorati verso dei nodi speciali chiamati Sink, i quali sono gli unici nodi detta rete che si interfacciano con un calcolatore o un cloud. Le reti WSN devono avere diversi requisiti, fra i quali bassi consumi, scalabilità, flessibilità ed i nodi devono essere di piccole dimensioni ed a basso costo.

1.1.1 Topologia di una WSN

Le topologie di reti utilizzate in una WSN sono sostanzialmente tre:

- **Stella:** prevede un nodo centrale che funge da coordinatore della rete. Un qualsiasi altro nodo per comunicare con gli altri invia il messaggio al coordinatore che lo inoltra al destinatario; tutti i messaggi eseguono al massimo

due hop. È la topologia più semplice che permette l'uso di protocolli e algoritmi altrettanto semplici.

- **Mesh:** nelle reti mesh o peer-to-peer ogni nodo della rete può comunicare con gli altri nodi della rete che rientrano nella sua copertura radio. I nodi sono considerati tutti uguali, perciò hanno tutte le stesse prestazioni. Viene definita *multihop*, ovvero un messaggio può attraversare più nodi intermedi. I vantaggi sono che si possono creare reti più grandi e si possono avere percorsi ridondanti che aumentano affidabilità e robustezza, introducendo però algoritmi complessi.

- **Albero:** i nodi formano un albero. I messaggi partono da un nodo della rete e risalgono verso la radice, che raccoglie dati e funge da coordinatore, ovvero il Sink. Il rischio più grande di questa topologia è proprio il sovraccarico del Sink.

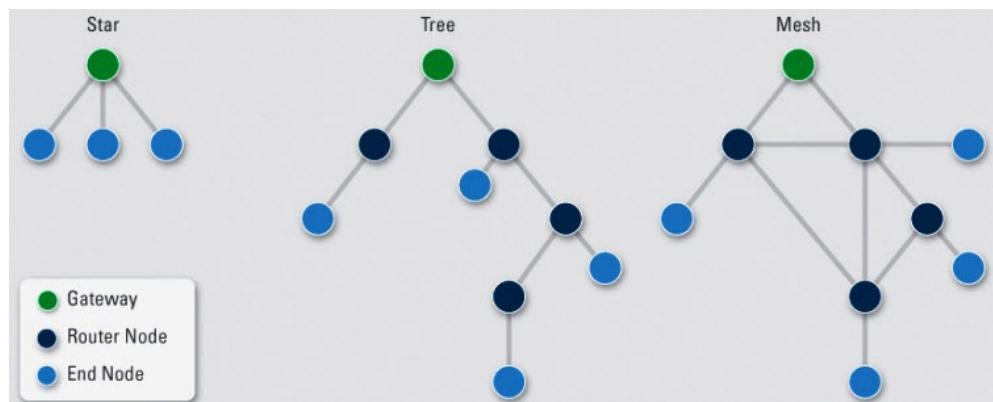


Figura 1.2: Topologie Rete WSN

WSN ha un grandissimo potenziale ma ancora non ha raggiunto la sua efficacia ottimale. Ciò è dovuto alle sfide intrinseche presenti e la domanda crescente dalle applicazioni, il che ha portato ad aumentare l'interesse nella

ricerca negli ultimi anni. Le maggiori sfide che dovrà affrontare WSN saranno sulla efficienza energetica, migliorare la comunicazione ed il routing.

1.2 Software Defined Networking

SDN è un paradigma il cui compito principale è quello di disaccoppiare le funzioni di controllo della rete da quelle di inoltro del traffico. Questo paradigma nasce proprio per risolvere i problemi delle reti tradizionali, ovvero la scarsa scalabilità e complessità, le quali non riuscivano a soddisfare le richieste di mercato dovute alla richiesta maggiore di applicazioni sempre più complesse ed un maggior numero di utilizzatori.

Questo paradigma verrà abbondantemente trattato e definito in ogni sua sfaccettatura nei capitoli successivi. [3]

1.3 SDN-WISE

SDN-WISE, è una soluzione Software Defined Network per Wireless Sensor Network. A differenza di SDN-WSN è stateful e propone due nuovi obbiettivi:

- riduzione di scambi di messaggi tra nodi e controller
- programmare i nodi come macchine a stati finiti così da poter abilitare operazioni che non potevano essere supportate da soluzioni stateless.

1.3.1 Approccio SDN-WISE

I Nodi di SDN-WISE sono racchiusi in tre strutture dati: WISE States Array, Accepted IDs Array e WISE Flow Table. Il Controller, come per l'approccio SDN, definisce le politiche di networking che verranno implementate

dai vari sensori. In ogni momento i nodi SDN-WISE sono caratterizzati da uno stato corrente per ogni controller attivo. Il Wise States Array è una struttura dati che contiene il valore dello stato corrente.

In SDN-WISE, come nell'approccio SDN-WSN, è implementato il protocollo Sensor OverFlow (SOF) che permette ai nodi della rete di avere tabella di flusso (WISE Flow Table-WFT). Nel caso in cui un pacchetto venga processato, il sensore esplorerà le entries della WFT. Ciascuna entry of WFT contiene una sezione Matching Rules, nel quale vengono definite le condizioni per processare un pacchetto. In caso contrario il nodo richiede aiuto al controller.

In SDN-WISE, la rete di sensori e i sink possono essere distinti. I Sink sono gateway tra il nodo sensore e il controller.

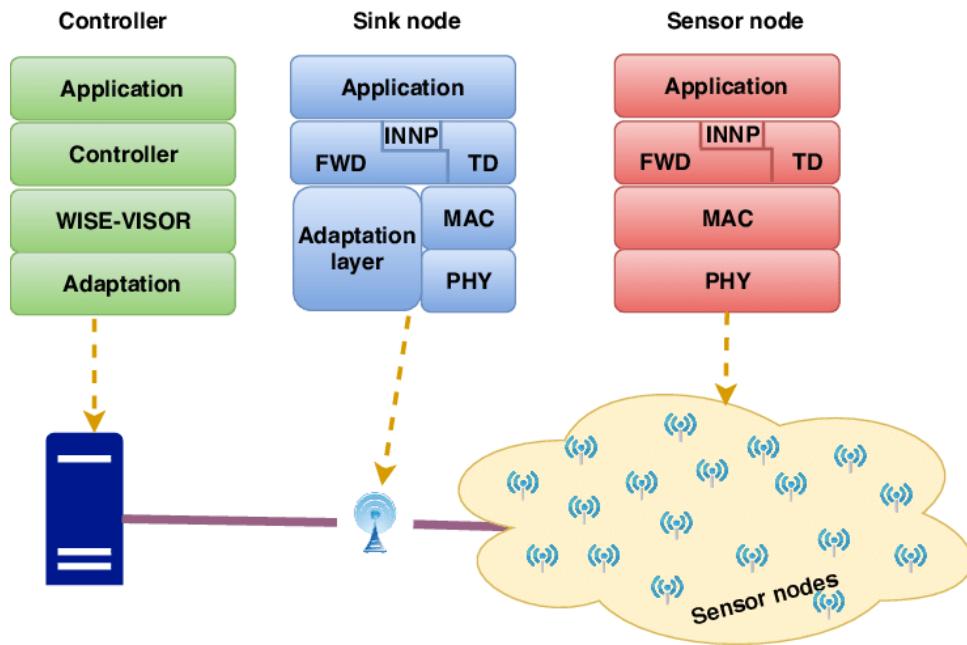


Figura 1.3: Architettura SDN-WISE

I Pacchetti SDN-WISE hanno un header fisso, che consiste di 10 byte, divisi in vari campi come segue, figura:

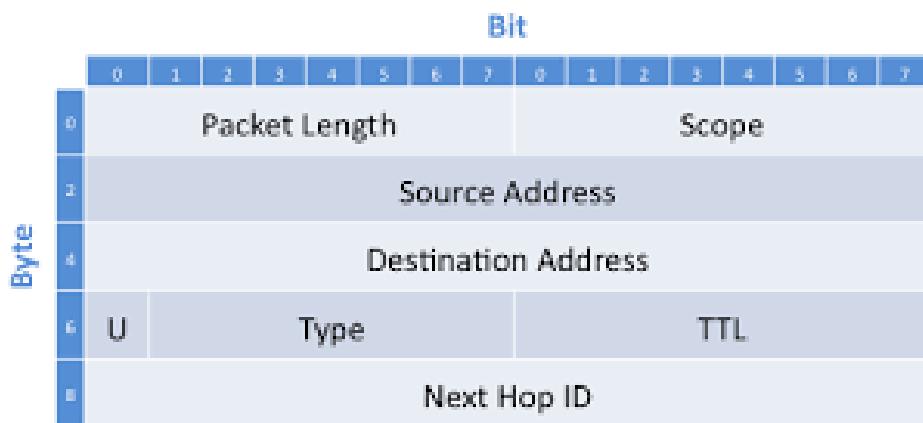


Figura 1.4: Header SDN-WISE

- Il campo lunghezza del pacchetto contiene la lunghezza totale, incluso il payload, in bytes.
- Scope identifica un gruppo di Controller che hanno espresso interesse nel contenuto del pacchetto. Inizialmente è inizializzato a zero, ma può essere modificato attraverso appropriate entry della WFT.
- Indirizzo del Mittente e del Destinatario, come specificato nel nome, indicano gli indirizzi rispettivamente del nodo che ha generato il pacchetto e il nodo che lo riceve.

Capitolo 2

Internet of Things

L'Internet of Things (IOT) è l'ultima tecnologia sviluppata nella lunga e continua rivoluzione nel mondo della comunicazione ed elaborazione dati. La sua dimensione ed influenza nella vita di tutti i giorni, nel mondo del business e nel mondo politico ha schiacciato ogni tipo di tecnologia avanzata sviluppata in precedenza.

2.1 Things of the Internet of Things

Internet of Things (IOT) è un termine che si riferisce ai collegamenti in sviluppo che spaziano dai piccoli dispositivi smart fino agli applicativi con piccoli sensori. Il tema dominante è la possibilità di mettere in comunicazione persone con oggetti ed oggetti con oggetti stessi. Attualmente l'internet che conosciamo oggi permette di interconnettere bilioni di industrie e di oggetti personali che usualmente si interfacciamo con sistemi Cloud. Invece l'IOT è principalmente guidato da dispositivi embedded. Questi dispositivi sono usualmente a bassa larghezza di banda, bassa ripetizione per l'acquisizione

dati ed a bassa larghezza di banda per utilizzo dati che comunicano fra loro attraverso la user interface.

2.2 Evoluzione

Le evoluzioni di Internet che hanno portato allo sviluppo dell'IOT sono quattro:

- INFORMATION TECHNOLOGY: PCs, servers, routers, firewalls e più in generale dispositivi IT con connessione internet.
- OPERATIONAL TECHNOLOGY (OT): macchine a applicativi non costruiti da compagnie IT, come ad esempio macchinari medici, SCADA(supervisory control and data acquisition) e processi di controllo.
- PERSONAL TECHNOLOGY: smartphones, tablets ed ebook reader, acquistati da vari clienti, che utilizzano connessione ad internet. Inoltre i dispositivi presentano varie forme di connettività wireless.
- SENSOR/ACTUATOR TECHNOLOGY: singoli dispositivi che utilizzano connettività wireless che fanno parte di grandi sistemi.

2.3 Strati dell'Internet of Things

La letteratura tecnica e quella business si focalizzano su due aspetti principali dell'IOT ovvero gli oggetti che sono collegati ad Internet che interconnette loro stessi.

La migliore visione dell'IOT è vederlo come un enorme sistema che consiste in cinque strati:

- Sensori e attuatori: questi sono oggetti. I sensori osservano l'ambiente e riportano delle misurazioni. Gli attuatori operano direttamente su quell'ambiente.

biente.

- Connectivity: un dispositivo può collegarsi via wireless o via cavo ad un network e mandare collezioni di dati al data center (sensore) oppure ricevere comandi da un controllore (attuatori).
- Capacity: il network che supporta i dati deve essere capace di supportare un enorme flusso di dati.
- Storage: necessita di un largo storage per salvare e per mantenere i dati (backup). Generalmente avviene lato Cloud.
- Data Analytics: per un gran numero di dispositivi viene generato un flusso dati enorme che richiede di essere analizzato e processato.

Capitolo 3

Requisiti e Tecnologie

Questo capitolo introduce due meccanismi che sono fondamentali nel flusso operativo del network e nella sua capacità di trasmettere ed inviare pacchetti: il routing ed il congestion control.

3.1 Routing

3.1.1 Caratteristiche

La caratteristica principale di Internet è quella di ricevere pacchetti da una fonte ed inviarli ad una nuova destinazione. Per permettere questo è necessario che venga definita una strada attraverso la rete e generalmente vengono utilizzate le performance della rete come criterio di decisione.

Il criterio più semplice che viene utilizzato il cammino minimo attraverso la rete, ovvero quello che passa attraverso il minor numero di nodi. È un criterio di misurazione molto semplice che permette di minimizzare le risorse utilizzate dalla rete.

Le politiche di routing vengono decise anche in base ad alcuni criteri, come ad

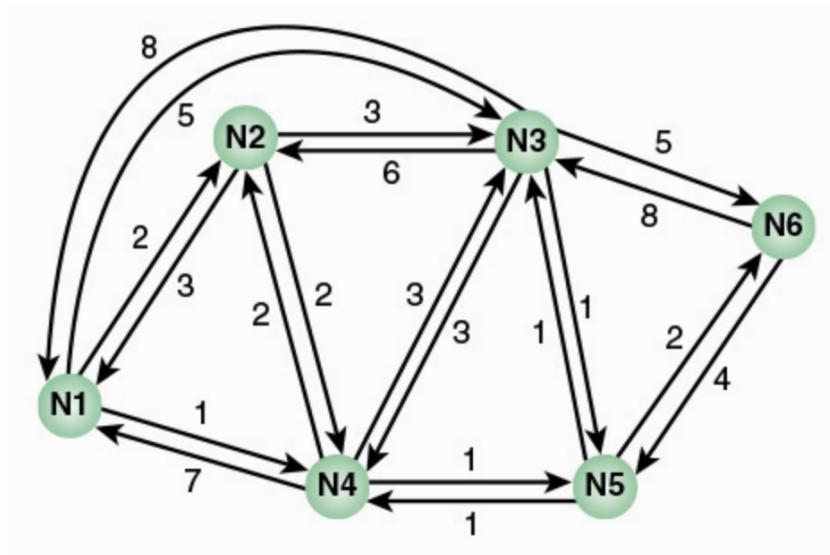


Figura 3.1: Esempio di architettura di rete

esempio scegliere la strada più veloce, minimizzare il tempo di delay oppure si può decidere di adottare solo alcune strade sicure per politiche di sicurezza.

3.1.2 Packet Forwarding - Inoltro Pacchetti

La funzione chiave di ogni router è quella di accettare pacchetti in arrivo e reindirizzare quest'ultimi. Per favorire questo processo ogni router è dotato di forwarding tables, ovvero delle tavole di inoltro pacchetti che mostrano per ogni nodo le caratteristiche e le strade da percorrere per raggiungere il nodo successivo.

Ogni router può essere responsabile di scoprire ed individuare le strade più appropriate oppure alternativamente può essere presente un network control center (centro di controllo della rete) che disegna le strade e mantiene una tabella di inoltro centrale.

Le tabelle più semplici, come quella mostrata in figura, contengono solo il

CENTRAL FORWARDING TABLE						
		From Node				
		1	2	3	4	5
To Node	1	-	1	5	2	4
	2	2	-	5	2	4
	3	4	3	-	5	3
	4	4	4	5	-	4
	5	4	4	5	5	-
	6	4	4	5	5	6

Node 1 Table		Node 2 Table		Node 3 Table	
Destination	Next Node	Destination	Next Node	Destination	Next Node
2	2	1	1	1	5
3	4	3	3	2	5
4	4	4	4	4	5
5	4	5	4	5	5
6	4	6	4	6	5

Node 4 Table		Node 5 Table		Node 6 Table	
Destination	Next Node	Destination	Next Node	Destination	Next Node
1	2	1	4	1	5
2	2	2	4	2	5
3	5	3	3	3	5
5	5	4	4	4	5
6	5	6	6	5	5

Figura 3.2: Packet Forwarding Tables

nodo di destinazione, ma possono contenere anche informazioni addizionali come il source address, packet flow identifier ed il livello di sicurezza del pacchetto:

- FAILURE: quando un nodo o un collegamento fallisce non può più essere usato come parte della strada da seguire.
- CONGESTION: quando un'area della rete è fortemente congestionata, si desidera reindirizzare i pacchetti al di fuori dell'area della congestione, facendogli intraprendere strade alternative.
- TOPOLOGY CHANGE: l'inserimento di nuovi collegamenti o nodi modifica il routing (instradamento).

Per un sistema di routing adattivo, l'informazione sullo stato della rete può essere scambiata tra i nodi o tra i nodi e il central controller.

3.1.3 Routing Protocols

Ogni router è capace di prendere decisioni di instradamento dei pacchetti in base alla conoscenza e alla topologia della rete ed alle condizioni di traffico

e delay di internet.

Infatti tra i router è richiesto un accordo di cooperazione dinamica, in particolare i router devono evitare porzioni della rete che falliscono e devono evitare porzioni della rete dove ci può essere congestione. Per permettere questo tipo di decisioni i routers si basano su protocolli di routing.

Esistono due categorie di protocolli di routing che sono basate sul concetto di autonomous system (AS).

Un sistema AS richiede determinate caratteristiche:

1. Un AS è un set di router e di reti controllato da una singola organizzazione.
2. Un AS consiste in un gruppo di routers che si scambiano informazioni attraverso protocolli di routing.
3. Ad eccezione del tempo di fallimento, un AS è connesso (graph-theoretic sense); c'è una strada per ogni coppia di nodi.

Un protocollo routing, chiamato Interior Router Protocol (IRP) passa le informazioni fra router all'interno di un AS e non necessita di essere implementato all'estero del sistema. Questa flessibilità permette agli IRPs di essere personalizzati su misura all'interno di specifiche applicazioni o per specifici requisiti.

Può succedere che una rete sia costituita da più di un AS, come ad esempio una rete lan che collega più zone di un'università, che possono essere collegate per formare un unico AS. In questo caso gli algoritmi di routing e le tabelle di routing usate dai router possono cambiare. Infatti in questo caso al router devono essere passate anche informazioni esterne al singolo sistema AS e servirà un protocollo per passare informazioni attraverso routers di differenti AS.

Questo protocollo prende il nome di exterior router protocol (ERP). Il proto-

collo ERP necessita di passare meno informazioni rispetto ad un IRP, poichè se ad esempio devo passare un'informazione fra due host di due reti AS, un router nel primo AS determina il target di AS ed indica una strada per raggiungere quel determinato sistema; una volta che il pacchetto è entrato in quel sistema, il router all'interno del sistema può cooperare per indirizzare quel pacchetto.

L'ERP non conosce i dettagli e la strada che il pacchetto deve seguire una volta entrato nell'AS.

3.1.4 Elementi del Router

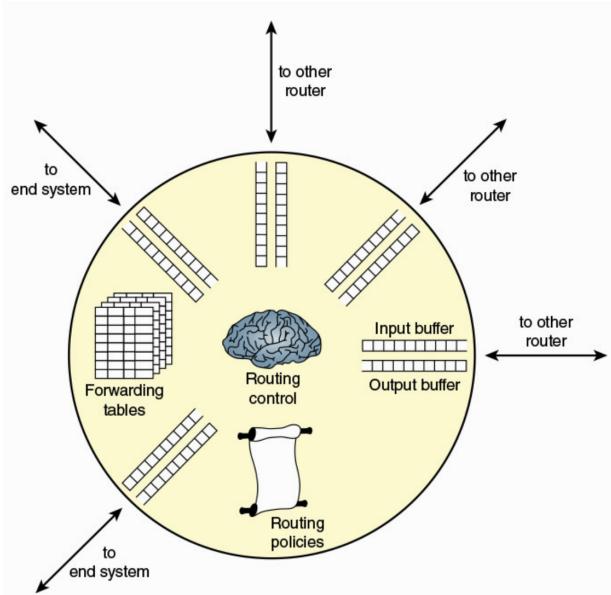


Figura 3.3: Elementi del router

La figura mostra una funzione di controllo del routing che include in particolare esecuzione di protocolli di routing, mantenimento di tabelle di routing e la supervisione alle politiche di controllo della congestione.

Ogni router è dotato di porte I/O connesse a quest'ultimo. Su ogni porta

arrivano e partono pacchetti; per semplificare il modello possiamo dire che da una parte si accettano i pacchetti in arrivo e dall'altra si mantengono i pacchetti in attesa che partano.

I pacchetti che arrivano sono memorizzati nell'input buffer della corrispondente porta, dopodichè il router lo esamina e prende una decisione basata sulle tabelle di inoltro dei pacchetti e sposta il pacchetto sul corrispondente output buffer.

I pacchetti in coda per l'output sono trasmessi il più velocemente possibile e vengono messi in coda secondo la politica FIFO (First In First Out). Più comunemente ci sono anche altre regole più complesse che gestiscono questo flusso, come ad esempio la dimensione del pacchetto.

3.2 SDN e NFV

Con lo sviluppo e l'incremento del volume e della varietà del traffico internet, dovuto alla grande ricerca come i big-data, cloud computing e traffico mobile, è diventato sempre più difficile trovare un punto d'incontro tra QoS e requisiti QoE. Le reti hanno bisogno di essere più flessibili e adattive ed è per questo che si sono sviluppate due tecnologie adottate da diversi servizi network, ovvero il software-defined networking (SDN) e il network functions virtualization (NFV). [4]

3.2.1 Software Defined Networking

L'SDN ha raggiunto un così ampio livello che sta rimpiazzando i tradizionali modelli network, infatti permette un enorme livello di flessibilità e di personalizzazione che va incontro alle nuove esigenze di networking e alle esigenze di tutto il compartimento IT come ad esempio il cloud, la mobilità,

social network e video.

FUNZIONALITÀ DELL’SDN

Ci sono due elementi che si prendono parte all’inoltro dei pacchetti, ovvero una funzione di controllo (control function) che decide la strada da seguire e la relaziona in base al traffico della rete, e una funzione dati (data function) che inoltra i dati basandosi sulle politiche della function control. Prima dell’avvento dell’SDN queste funzioni erano integrate nei dispositivi connessi alla rete (router, bridge, packet switch). Con l’SDN, un controller centrale racchiude tutte le funzioni più complesse, incluso il routing, la nomenclatura, la policy e i controlli di sicurezza.

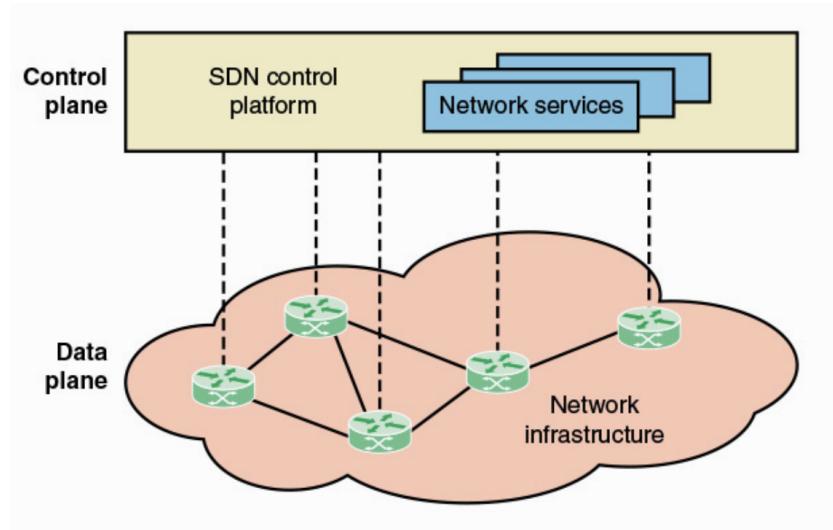


Figura 3.4: Software Defined Networking

Questo costituisce l’SDN control plane che è formato da uno o più SDN controller, i quali definiscono il flusso di dati che occorre nell’SDN data plane. Tutto il flusso attraverso la rete è configurato dal controller che verifica la fattibilità della comunicazione attraverso le politiche della rete. Se il con-

trollore accorda il flusso, calcola una strada che quest'ultimo deve seguire ed apre il flusso a tutti gli switch presenti sulla rete, i quali si occupano solo di controllare il flusso. Questi switches costituiscono il data plane. La comunicazione fra controller e switches usa protocolli standard.

KEY DRIVERS

Un fattore chiave dell'SDN è la incredibile diffusione di server virtuali. Questi ultimi permettono di mascherare le risorse del server, il numero e le entità che compongono il server, i processori, il sistema operativo e gli utenti. Questo è possibile partizionando una singola macchina in multipli ed individuali servers conservando le risorse hardware. Inoltre rende possibile la migrazione del server in caso di possibili fallimenti della rete. La virtualizzazione dei server è diventato un elemento centrale in combinazione con i big data ed il cloud computing, ma ha creato problemi con l'architettura rete tradizionale. Un problema è la configurazione delle virtual LAN, VLAN, che necessitano di essere riconfigurate ogni volta che viene spostato un server virtuale. Un altro effetto della virtualizzazione dei server è che il modello è molto diverso dal classico client/server a cui siamo abituati, poiché i server sono in continua comunicazione fra loro ed i flussi server-to-server cambiano di locazione ed intensità tantissime volte, richiedendo un approccio flessibile al network. Un altro fattore che richiede una risposta rapida nell'allocare risorse network è la continua evoluzione ed il continuo aumento nell'utilizzo di dispositivi mobili, come smartphone, tablet e computer portatili che rapidamente cambiano il loro punto di connessione alla rete, la quale deve essere in grado di cambiare rapidamente le risorse, il QoS e mantenere la sicurezza sui dispositivi mobili. L'infrastruttura network esistente è in grado di rispondere ai cambiamenti per il controllo del traffico, ma questo richiede una grande spesa di tempo e

di infrastrutture se la rete è molto grande e ci sono molti dispositivi. La rete deve configurare ogni tipologia di dispositivo separatamente ed aggiustare le performance ed i parametri di sicurezza sessione per sessione e per tipo di applicazione. Su una larga scala, quando una macchina virtuale, VM, viene creata può richiedere ore o perfino giorni per essere configurata.

3.2.2 Network Function Virtualization

Attualmente la tecnologia delle VM attraverso internet o nell'industria del network è stata usata con la funzione di server a livello applicativo come ad esempio database servers, cloud server, web servers and email servers. La stessa tecnologia può essere però utilizzata per i dispositivi della rete come ad esempio routers, LAN switches, firewall e IDS/IPS servers. Il Network Function Virtualization (NFV) scollega le funzioni della rete, come per esempio il routing, il firewallsing, intrusion detection ed il Network Address Translation dalle piattaforme proprietarie hardware e implementa queste funzioni via software. Utilizza tecnologie virtuali standard che girano su dispositivi hardware molto performanti per virtualizzare queste funzioni del network ed è applicabile sia alle infrastrutture di rete via cavo che wireless. NFV ha un delle proprietà in comune con SDN: - muove funzionalità sul lato software - usa piattaforme hardware dedicate invece di piattaforme hardware proprietarie - usa applicazioni standard o applicazioni aperte (APIs) - supporta con più efficienza modifiche e cambiamenti della rete NFV ed SDN sono indipendenti ma complementari poichè quest'ultimo scollega i dati ed il piano di controllo, rendendo il controllo e l'instradamento della rete più flessibile ed efficiente. NFV invece scollega le funzioni della rete da specifiche piattaforme hardware attraverso la virtualizzazione per rendere queste funzioni più flessibili ed efficienti. Anche SDN può essere virtualizzato, ma invece che

usati singolarmente, SDN ed NFV possono essere combinati per raggiungere grandi benefici e grandi performance.

Capitolo 4

SDN: Background and Motivation

4.1 Requisiti dell'Evoluzione Network

Un incredibile numero di articoli sta invitando le aziende che sviluppano dispositivi network e gli utenti a rivalutare il concetto dell'approccio tradizionale all'architettura della rete. Questi articoli sono raggruppabili in incremento della domanda, fornitura ed i pattern del traffico.

4.1.1 Incremento Della Domanda

- Cloud Computing: c'è stato un enorme passaggio sia per i servizi pubblici che privati ai servizi cloud.
- Big Data: processare enormi quantità di data-set richiede l'utilizzo di migliaia di server in parallelo, che devono essere interconnessi gli uni con gli altri. C'è un'enorme e costante incremento di domanda per la capacità delle reti network ed i data center.
- Mobile Traffic: gli impiegati accedono alla rete della loro azienda costantemente anche dall'esterno, con i loro smartphone, tablet e pc. Questi dispositivi hanno sofisticate app che generano traffico audio-video, aumentando

continuamente il carico della rete.

- The Internet of Things (IOT): la maggior parte degli oggetti genera un traffico modesto, anche se ci sono delle eccezioni come le video camere di sorveglianza. Un grande numero di questi dispositivi può rappresentare per la rete dell'azienda un grosso carico.

4.1.2 Fornitura in Aumento

Con l'incremento della domanda per la rete, anche le tecnologie della rete per assorbire questo grande traffico sono in continuo aumento ed evoluzione, in particolare dopo che le reti 4G e 5G hanno dato la possibilità ai dispositivi mobili di connettersi in massa alla rete. L'incremento delle tecnologie di trasmissione della rete è dovuto anche all'incremento dei vari dispositivi, come i LAN switches, routers, firewall, intrusion detection/prevention systems. Anno dopo anno stanno diventando sempre più grandi, con memorie più veloci, con buffer più grandi e ad accesso più rapido, tanto veloci quanto un processore.

4.1.3 Traffic Pattern molto complessi

Con lo sviluppo di tutte queste tecnologie, si è sviluppato anche lo schema di traffico della rete e le architetture di rete tradizionali faticano a stare al passo con la domanda. Infatti fino ad oggi l'architettura dominante nelle imprese è il modello client-server che permette interazioni fra un client ed un server; in questo ambiente la rete può essere configurata staticamente ed è facilmente prevedibile il traffico della rete.

Un numero di sviluppatori ha mostrato quale potrebbe essere un modello più dinamico e con complessi schemi di traffico all'interno dell'industria:

- Gli applicativi client/server devono accedere a server e database multipli

che devono comunicare fra loro, generando traffico orizzontale fra server e traffico verticale come il modello client/server.

- La convergenza sulla rete di voce, dati, video genera un traffico impredicibile.
- Le UC (Unified Communications) hanno un pesante di applicazioni che necessitano un accesso multiplo ai servers.
- Il pesante uso di dispositivi mobili permette all'utente di accedere in ogni momento a dati e applicativi aziendale da ogni dispositivo e da ogni luogo. Infatti il traffico mobile è diventato una significante parte dell'infrastruttura di rete industriale.
- Il diffuso uso di cloud pubblici ha spostato quello che prima era il traffico locale per molte industrie, risultando in continuo aumento e con flussi di caricamento impredicibili.
- La nuova pratica comune di applicazioni e database server virtuali ha incrementato significativamente gli hosts che richiedono un alto volume di traffico per accedere alla rete e di conseguenza in ogni cambiamento fisico di dislocazione delle risorse del server.

4.1.4 Le Architetture di Rete Tradizionali sono Inadeguate

Con il grande sviluppo degli schemi di trasmissione e le crescenti performance dei dispositivi network, l'architettura di rete tradizionale risulta sempre più inadeguata.

Il modello tradizionale di Internet si basa su un'architettura che utilizza il protocollo TCP/UDP, il quale ha tre caratteristiche portanti:

- Sistema di indirizzamento finale a due livelli
- Routing basato sulla destinazione

- Sistema distribuito, controllo autonomo

Tradizionalmente il routing era basato su ogni indirizzo di destinazione di ogni singolo pacchetto. In questo approccio di tipo DATAGRAM, dove ogni pacchetto è indipendente dal successivo, i pacchetti successivi possono seguire differenti strade attraverso internet ed i router si impegnano per trovare il cammino per ogni singolo pacchetto che comporti il minor ritardo. Più recentemente, per soddisfare i requisiti del QoS, i pacchetti vengono processati in termini di flussi di pacchetti.

L'ONF (Open Networking Foundation) ha delineato quattro limitazioni generali della tradizionale architettura di rete:

- Architettura statica e complessa: per rispondere alla domanda di così tanti livelli di QoS, alto flusso di traffico e requisiti di sicurezza l'architettura è diventata sempre più complessa e difficile da gestire con un numero enorme di protocolli che aggiungono sempre più informazioni alla rete. La difficoltà più grande si ha quando si aggiungo o si muovono dispositivi all'interno della rete.
- Politiche inconsistenti: per implementare la sicurezza della rete su larga scala è necessario effettuare migliaia di modifiche. Infatti quando in una rete molto vasta si aggiunge una macchina virtuale (VM), quest'ultima può richiedere molte ore o addirittura giorni per essere configurata.
- Impossibile da ridimensionare: con l'aumento della domanda per la rete, sia in volume che in varietà, una soluzione adottabile potrebbe essere quella di aggiungere switch e dispositivi di trasmissione, ma questo risulta complesso a causa della staticità della rete. Allora si può pensare a sovrascrivere porzioni di sottorete basandosi sulla predicitività dei modelli di traffico, ma a causa dell'incremento dell'utilizzo della virtualizzazione e la varietà di applicazioni multimediali, tutto questo diventa impossibile.

- Dipendenza dal fornitore: per la natura attuale del traffico e della richiesta di rete, le imprese devono proporre e sviluppare nuove capacità e servizi per rispondere al cambiamento del business e alla domanda degli utenti. Un conspicuo numero di open interface ha abbandonato il mondo delle imprese per la scarsa produttività dei fornitori.

4.2 L'Approccio SDN

In questo capitolo parleremo dell'SDN mostrando come è stato disegnato e progettato per rispondere ai requisiti che richiede l'evoluzione della rete.

4.2.1 Requisiti

La ODCA (Open Data Center Alliance) ha diramato una lista di requisiti per definire un approccio moderno alla rete:

- Adaptività: la rete deve aggiustare e rispondere dinamicamente in base alla necessità della singola applicazione, delle condizioni della rete o dell'esigenza di una azienda.
- Automazione: i cambiamenti delle policy devono essere diramati automaticamente, in modo tale da ridurre l'errore umano e il lavoro manuale.
- Manutenzione: l'introduzione di nuove caratteristiche e capacità (update software, patches) deve avvenire senza interrompere la continuità e senza causare interruzioni alle operazioni.
- Modello di gestione: i software di gestione della rete devono permettere la gestione della rete a livello di modello, invece di implementare cambiamenti concettuali che necessitano di modificare singolarmente ogni elemento.
- Mobilità: le funzioni di controllo devono soddisfare anche la mobilità, inclusi dispositivi mobili degli utenti e server virtuali.

- Sicurezza integrata: le applicazioni di rete devono integrare senza interrompere la sicurezza, che deve essere ritenuta come uno dei servizi chiave e non come una aggiunta finale.
- Ridimensionamento su scala: l'implementazione deve avere la capacità di adattare la rete e i suoi servizi in base alla richiesta.

4.2.2 Architettura SDN

Una analogia che possiamo disegnare sull'evoluzione dell'architettura della rete è il paragone che nasce spontaneo con lo sviluppo verticale delle architetture dei computer e dei loro sistemi operativi. Infatti ad oggi tutto l'ambiente di sviluppo hardware e software è dotato di una grandissima flessibilità; questo infatti permette ha permesso lo sviluppo di macchine virtuali che possono essere spostate da un server all'altro e attraverso piattaforme di diversa natura hardware e software.

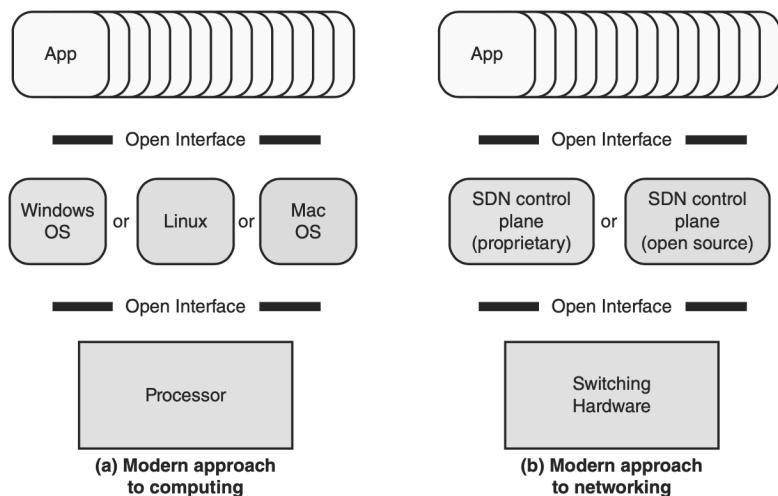


Figura 4.1: Approccio Moderno

L’ambiente delle reti ad oggi si trova a dover fare i conti con alcune limitazioni e la difficoltà maggiore è quella di fare interfacciare le diverse applicazioni con l’infrastruttura della rete.

Il concetto centrale dell’approccio SDN è quello di permettere agli sviluppatori e ai gestori delle reti di avere lo stesso tipo di controllo che ad oggi si ha sui server x86. Il concetto di open interface applicato all’architettura SDN è quello di avere differenti applicativi che comunicano nello stesso modo con i controller SDN.

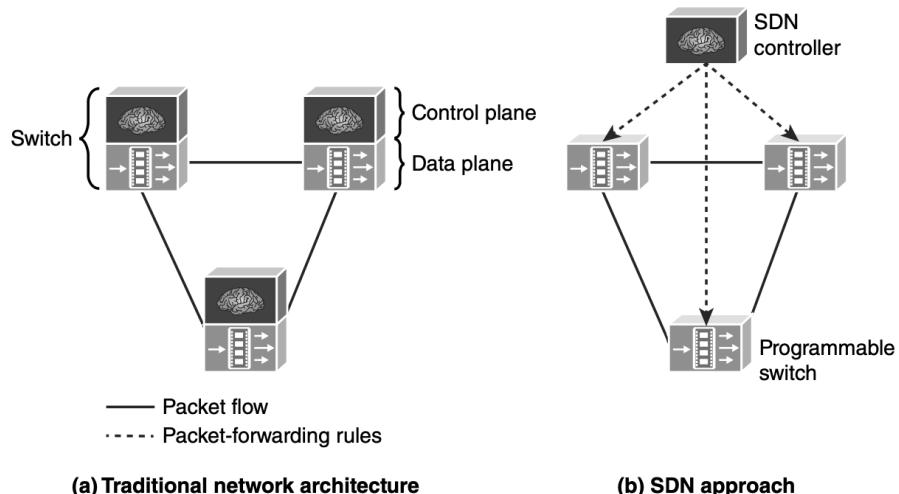


Figura 4.2: Control and Data Plane

Il data plane consiste in vari switches fisici e switches virtuali. In entrambi i casi gli switches si occupano di inoltro dei pacchetti. Le altre caratteristiche interne come ad esempio buffer o priority parameters sono modellate in base alle richieste del cliente. Ogni switch deve implementare un modello, fisico o astratto, di inoltro pacchetti. Questo modello è definito in termini di un open application programming interface (API) tra il control plane e il data

plane (southbound API).

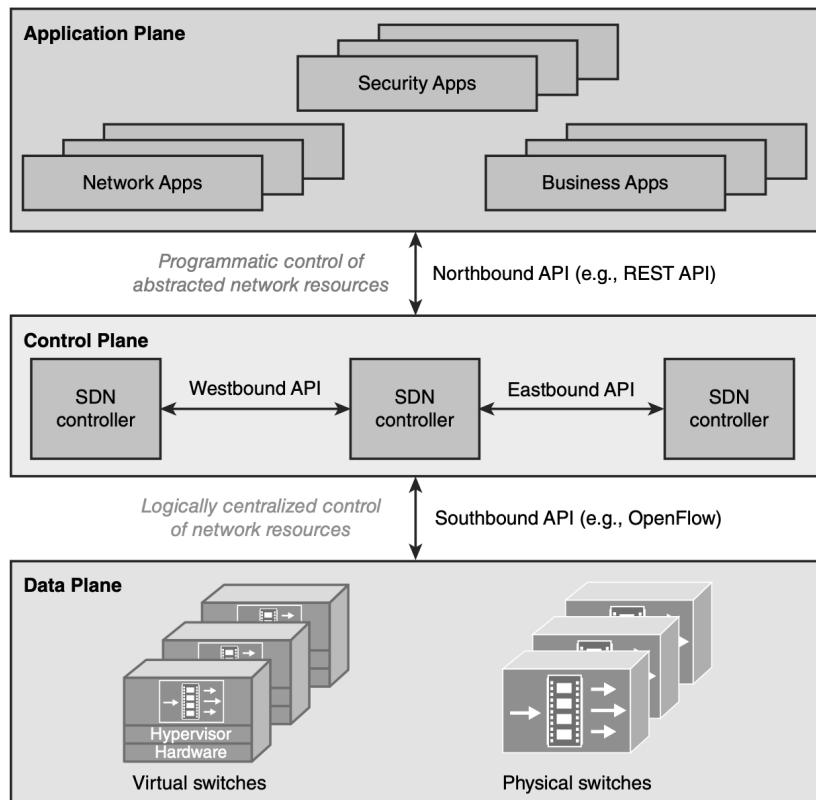


Figura 4.3: Architettura SDN

I controller SDN possono essere implementati direttamente su un server o su un server virtuale. OpenFlow o altre open API sono utilizzate per controllare gli switch nel data plane. In addizione, i controller utilizzano le informazioni sulle capacità e sulla domanda ottenuti dall'equipaggiamento della rete attraverso cui passa il flusso dei dati. I controller SDN mostrano inoltre le Northbound API che permettono agli sviluppatori e ai gestori delle reti di sviluppare un alto numero di applicativi personalizzati, molti dei quali non era possibile realizzare prima dell'avvento dell'SDN.

Sono inoltre pensate delle horizontal APIs (Eastbound/Westbound) che sono capaci di comunicare e cooperare fra gruppi di controllers per sincronizzare

gli stati.

A livello applicativo ci sono una varietà di applicazioni che possono interagire con i controller SDN. Un esempio sono reti ad alta efficienza energetica, monitoraggio di sicurezza, controllo degli accessi e gestione della rete.

4.2.3 Caratteristiche dell’SDN

Mettendo tutto insieme, le caratteristiche chiave dell’SDN sono le seguenti:

- Il control plane è separato dal data plane. I devices del data plane sono semplici dispositivi di inoltro pacchetti (packet-forwarding).
- Il control plane è implementato in un controller centralizzato. Il controller SDN ha quindi una visione centralizzata della rete o delle reti sotto il suo controllo. Il controller è un software portatile che può girare su server ed è capace di programmare l’inoltro dei dispositivi forte della sua visione centralizzata.
- Le open interfaces sono definite tra i dispositivi nel control plane e quelli nel data plane.
- La rete è programmabile in base alle applicazioni che girano in cima ai controller SDN. I controller SDN presentano una visione astratta delle risorse di rete per i vari applicativi.

4.3 SDN- e NFV-Related Standard

A differenza di altre aree tecnologiche, non c’è nessun corpo responsabile di sviluppare gli open standard per l’SDN o NFV. Piuttosto ci sono un alto numero di organizzazioni, consorzi industriali e svariate iniziative impegnate nel creare degli standard e delle linee guida per l’SDN e l’NFV.

4.3.1 Standard Developing Organizations

Internet Society, ITU-T ed ETSI stanno avendo un contributo chiave nella standardizzazione dell'SDN e dell'NFV.

Internet Society

I gruppi più attivi all'interno dell'Internet Society (ISOC) sono due: IETF e IRTF. L'ISOC è il comitato di coordinamento per il design, l'ingegnerizzazione e la gestione di Internet. Le aree coperte includono operazioni all'interno dello stesso Internet, standardizzazione di protocolli usati da sistemi o dalla rete Internet.

L'**Internet Engineering Task Force (IETF)** ha gruppi di sviluppo che lavorano in aree specifiche dell'SDN:

- *Interface to routing system (I2RS)*: sviluppa la capacità di interagire con i router e i protocolli di routing da applicare alle politiche di routing.
- *Service function chaining*: sviluppa l'architettura e le capacità dei controller di dirigere flussi di traffico attraverso la rete in modo tale che ogni piattaforma di servizio virtuale veda solo il traffico che deve dirigere.

L'**Internet Research Task Force (IRTF)** ha effettuato delle pubblicazioni che riguardano l'approccio nel costruire l'architettura dei layer SDN, discussioni in merito alle southbound API o alle varie APIs per i protocolli di routing.

ITU-T

La *International Telecommunication Union-Telecommunication Standardization Sector (ITU-T)* è un'agenzia che ha definito standard nell'area delle telecomunicazioni.

All'interno dell'ITU-T ci sono quattro team di studio (SGs):

- SG 13: network futuri, includendo il cloud computing, mobile e i network della next-generation.
- SG 11: requisiti di segnale, protocolli e test sulle specifiche.
- SG 15: trasporto, accessi e reti di casa.
- SG 16: multimedia.

ETSI: European Telecommunications Standards Institute

ETSI è riconosciuto dall'Unione Europea come una organizzazione europea standard anche se risulta un'organizzazione no-profit nonostante abbia un'impatto internazionale. Infatti ETSI ha avuto un ruolo chiave nel definire gli standard dell'NFV a livello di architettura, infrastruttura, metriche di qualità del servizio e politiche di sicurezza.

4.3.2 Industry Consortia

I consorzi di definizione degli standard hanno iniziato a svilupparsi negli anni 80 con lo scopo di fornire gli standard all'interno dell'avanzata frenetica del mondo tecnologico.

Il consorzio più importante nel mondo dell'SDN è l'*Open Networking Foundation (ONF)* che è interamente dedicato alla promozione e all'adozione dell'SDN attraverso standard di sviluppo aperti. Il più importante contributo è la realizzazione di un protocollo OpenFlow e la realizzazione di API.

Il protocollo OpenFlow è la prima interfaccia standard disegnata per l'SDN ed è già stata impiegata nello sviluppo di diverse reti, sia a livello hardware che software.

4.3.3 Open Development Initiatives

Ci sono una serie di iniziative aperte che lavorano generalmente alla realizzazione di standard pubblici o di open software. Il numero di questi gruppi è diventato sempre più attivo nello sviluppo dell'SDN e l'NFV.

OpenDaylight

OpenDaylight è un software open source realizzato per girare su Linux. I suoi membri hanno realizzato numerosi controller per un'ampia gamma di applicativi.

Open Platform for NFV

Open Platform for NFV è un progetto open source dedicato ad accelerare l'adozione degli standard dell'NFV, aumentarne la consistenza, le performance e le componenti open source.

OpenSlack

OpenSlack è un software open source che ha l'obiettivo di realizzare un cloud operating system completamente open source.

Capitolo 5

SDN Data Plane and OpenFlow

5.1 SDN Data Plane

Con il termine SDN Data Plane spesso ci si riferisce al livello infrastrutturale ovvero il livello dopo i dispositivi connessi alla rete performano il trasporto e il processo dei dati in accordo con le decisioni prese dall'SDN Control Plane.

La caratteristica chiave dei dispositivi connessi alla rete nell'SDN è che questi ultimi performano una semplice funzione di inoltro dei dati senza bisogno di software che prendano decisioni al loro posto.

5.1.1 Funzioni del Data Plane

Le principali funzioni dei dispositivi network sono le seguenti:

- **Control Support Function:** interagisce con il livello di controllo dell'SDN per supportare la programmabilità attraverso l'interfaccia di risorse e controllo. Gli switch comunicano con il controller comunica con gli switch attraverso il protocollo OpenFlow degli switch.
- **Data Forwarding Function:** accetta i flussi di dati provenienti da altri

dispositivi network per inoltrarli sulle strade che sono state predefinite in accordo con il livello applicativo dell’SDN.

Queste regole di instradamento utilizzate dai dispositivi network sono rappresentate dalle tabelle di instradamento che indicano per determinate categorie di pacchetti qual’è il loro prossimo passo da fare all’interno della rete.

5.1.2 Protocollo del Data Plane

Il protocollo è supportato da tutti i dispositivi network. Il flusso dei pacchetti dati consiste nel flusso di pacchetti IP; può quindi essere necessario per le tabelle di instradamento definire le entrate nei campi ad alto livello dei pacchetti, come avviene nel TCP o nell’UDP. La rete esamina l’IP header ed altri possibili header in ogni pacchetto per prendere le decisioni di instradamento.

5.2 OpenFlow

Per tradurre il concetto di SDN nella implementazione pratica devono verificarsi due condizioni:

- deve essere presente una architettura logica comune fra gli switch, i router e gli altri elementi della rete; inoltre il tutto deve essere controllato da un controller SDN.
- un protocollo standard e sicuro è necessario tra il controller SDN ed i dispositivi network.

Questi requisiti sono indirizzati dall’OpenFlow, il quale è sia un protocollo tra l’SDN ed i dispositivi della rete che una specifica logica degli switch facenti parte della rete. Ogni switch è infatti in grado di connettersi agli **OpenFlow switches** e possibilmente il tutto si connette direttamente ai

dispositivi finali che sono fonti o destinazione del flusso di pacchetti. Dalla parte degli switch l’interfaccia è nota come **OpenFlow Channel** e queste connessioni avvengono tramite le Open Flow ports che servono anche per connettere gli switch al controller SDN.

OpenFlow ridefinisce tre tipi di porte:

- **Physical Port**: corrispondono all’interfaccia hardware degli switch
- **Logical Port**: queste porte non corrispondono direttamente a qualche componente hardware degli switch. Le porte logiche sono infatti delle astrazioni che possono essere definite negli switch utilizzando dei metodi non-OpenFlow (link, tunnel, loopback interface). Possono includere l’incapsulamento dei pacchetti e possono mappare varie porte fisiche. È necessario che interagiscano con il processo OpenFlow così come fanno le porte fisiche.
- **Reserved Port**: definite da specifiche OpenFlow. Specificano regole generali di instradamento come l’invio e la ricezione dal controller o l’inoltro utilizzando metodi non-OpenFlow, come fanno gli switch normali.

Le specifiche OpenFlow definiscono tre tipi di tabelle nell’architettura logica degli switch:

- **flow table**: abbinano pacchetti in ricezione ad un particolare flusso e specificano quale tipo di funzione deve processare i pacchetti. Ci possono essere più tabelle di flusso che operano nella pipeline.
- **group table**: le flow table possono dirigere il flusso alle group table che possono intraprendere una varietà di azioni che va ad inficiare uno o più flussi di pacchetti.
- **meter table**: è in grado di innescare una moltitudine di azioni in un flusso di pacchetti.

5.2.1 Flow Table Structure

Il blocco principale dell’architettura logica degli switch sono le flow table. Ogni pacchetto che entra in uno switch passa attraverso una o più flow tables. Ogni flow table è formata da un numero di righe, chiamate **entries**, formata da sette componenti come si evince dalla figura.

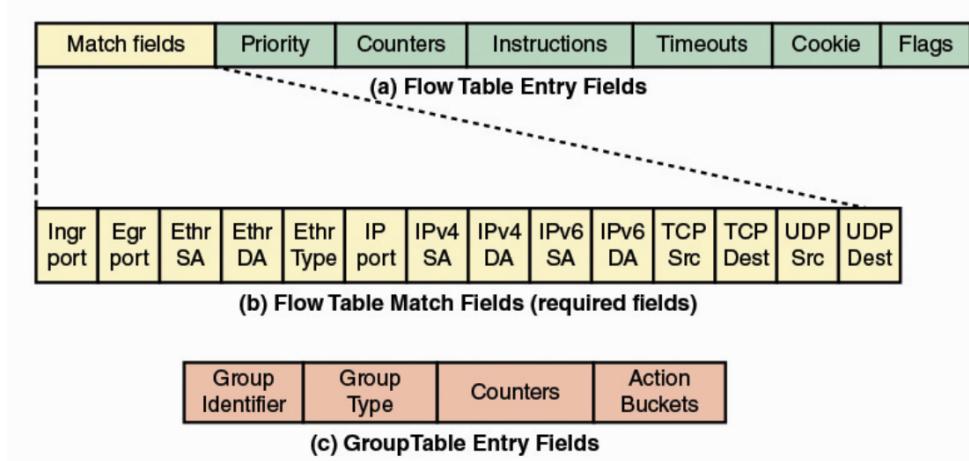


Figura 5.1: OpenFlow Table Entry Format

5.2.2 Flow Table Pipeline

Uno switch include uno o più flow tables. Se è presente più di una flow table, quest’ultime sono organizzate come una pipeline, con le tabelle organizzate con numeri crescenti a partire da zero.

L’uso di tabelle multiple in una pipeline invece di una singola flow table aggiunge al controller SDN una notevole flessibilità.

L’OpenFlow specifica due stati di processing:

- **Ingress processing:** questo stato accade ogni volta, a cominciare dalla Tabella 0 ed utilizza l’identità della porta di input. La Tabella 0 può essere anche l’unica ed in quel caso il processo è semplificato al processo di quella

singola tabella e non c'è egress processing.

- **Egress processing:** è il processo che inizia dopo aver determinato la porta di output. Questo stato è opzionale ma se occorre coinvolge una o più tabelle. La separazione dei due stati è indicata dall'identificatore numero della prima egress table. Tutte le tabelle che hanno un numero inferiore della prima egress devono essere usate come ingress table e viceversa nessuna tabella con un numero maggiore o uguale della prima egress table può essere usata come ingress table.

5.2.3 Group Table

Nel corso del processo della pipeline una flow table può dirigere il flusso di pacchetti ad una group table o ad un'altra flow table. Le group table o le group actions abilitano l'OpenFlow a rappresentare un set di porte come una singola entità per l'inoltro dei pacchetti.

Differenti tipi di gruppi sono a disposizione per rappresentare differenti astrazioni di inoltro pacchetti, come il broadcasting ed il multicasting.

Ogni group table è costituita da un numero di righe, chiamate group entries, formata da quattro componenti:

- **Group Identifier:** un intero a 32 bit senza segno che unicamente identifica il gruppo. Un gruppo è definito come una entry della group table.
- **Group Type:** per determinare la semantica del gruppo.
- **Counters:** aggiornato quando i pacchetti sono processati da un gruppo.
- **Action bucket:** una lista ordinata di action bucket dove ognuna di queste contiene un set di azioni da eseguire e parametri associati.

5.3 Protocollo OpenFlow

Il protocollo OpenFlow descrive lo scambio di messaggi che prende luogo tra l'OpenFlow Controller e l'OpenFlow Switch. Tipicamente il protocollo è implementato all'inizio del TLS per favorire un canale OpenFlow sicuro.

Il protocollo OpenFlow abilita il controller ad eseguire le azioni di add, update e delete al flusso il ingresso alla flow table. Quest'ultimo supporta tre tipi di messaggi:

- **Controller to Switch:** questi messaggi iniziano dal controller ed in qualche caso richiedono una risposta dagli switch. Questa classe di messaggi abilita il controller a controllare lo stato logico dello switch, inclusa la sua configurazione ed i dettagli del flusso e degli ingressi nelle group table. Questo messaggio è inviato dal controller ad uno switch quando quello switch invia un pacchetto al controller ed il controller decide di non scartare il pacchetto ma di reindirizzarlo alla porta di output dello switch.
- **Asincrono:** questi tipi di messaggi sono inviati senza la sollecitazione del controller. Questa classe include vari messaggi di stato al controller. Include il messaggio di Packet-in, che può essere utilizzato dallo switch per inviare un pacchetto al controller quando non c'è nessun match sulla flow table.
- **Simmetrico:** questi messaggi sono inviati senza sollecitazioni ne da parte del controller ne dello switch. Sono semplici ed utili. Messaggi di "Hello" sono tipicamente inviati avanti e indietro fra il controller e lo switch quando la connessione viene stabilita. Questi messaggi di invio-risposta possono essere utilizzati sia dal controller che dallo switch per misurare la latenza, la banda, la interconnessione o per verificare che il dispositivo è online e funzionante.

Capitolo 6

SDN Control Plane

6.1 SDN Control Plane Architecture

L’SDN control layer mappa le richieste dell’application layer in specifici comandi e direttive per il Data Plane. Il control layer viene implementato come un server o un set di server noti come SDN controllers. [5]

6.1.1 Funzioni del Control Plane

Le funzionalità in dotazione all’SDN Controller posso essere viste come quelle di un **network operating sistem (NOS)**. Come su un convenzionale sistema operativo, un NOS ha in dotazione servizi essenziali, comuni interfacce applicative (APIs) e astrazioni di elementi di basso livello per gli sviluppatori.

La funzione di un SDN NOS permette agli sviluppatori di definire politiche per la rete e controllare la rete senza preoccuparsi delle caratteristiche dei dispositivi che possono essere sia eterogenee che dinamiche.

Un cospicuo numero di iniziative, sia commerciali che open source, ha por-

tato all’implementazione di diversi SDN Controllers: OpenDaylight, Open Network operating System (ONOS), POX, Beacon, Floodlight, Ryu ed Onix.

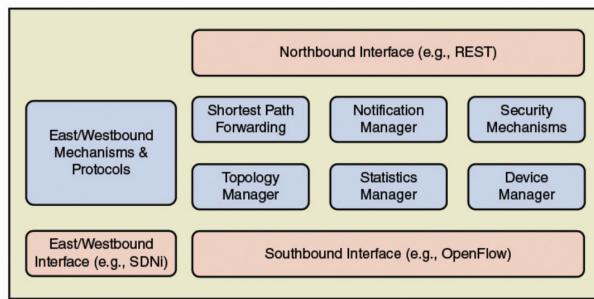


Figura 6.1: SDN Control Plane Functions and Interfaces

6.1.2 Southbound Interface

La southbound interface si occupa della connessione logica tra il Controller SDN e gli switch del data plane. Alcuni controller supportano e consentono di configurare solo un singolo southbound protocol. Un approccio più flessibile è l’uso di un southbound abstraction layer che permette una comune interfaccia per le funzioni del control plane e il supporto a molteplici APIs southbound.

L’implementazione più comune del southbound API è OpenFlow come abbiamo visto in precedenza.

6.1.3 NorthBound Interfaces

L’interfaccia northbound permette alle applicazioni di accedere alle funzioni del control plane e i vari servizi senza la necessità di conoscere i dettagli dei sottostanti switch network. L’interfaccia northbound è tipicamente vista come una API software rispetto ad un vero e proprio protocollo.

A differenza con la southbound interface non c'è nessuno standard da seguire o da rispettare. Il risultato di ciò è che sono state sviluppate innumerevoli controller complicando lo sviluppo delle applicazioni SDN. Per ovviare questo problema la Open Networking Foundation a costituito un Interface Working Group (NBI-WG) nel 2013 con l'obiettivo di definire e standardizzare un numero cospicuo di Northbound APIs. Attualmente non è stato ancora rilasciato nessuno standard.

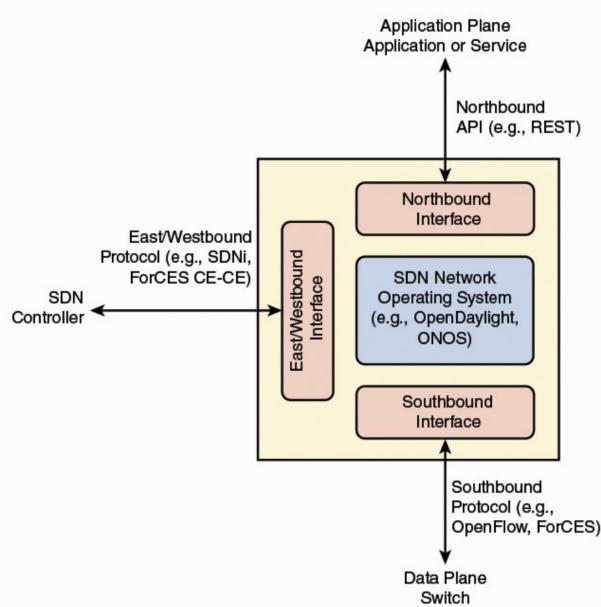


Figura 6.2: SDN Controller Interfaces

6.1.4 Routing

Il network SDN richiede delle funzioni di routing. In termini generali le funzioni di routing includono una collezione di informazioni sulla topologia e sul traffico della rete ed un algoritmo per delineare al meglio i percorsi all'interno della rete.

Esistono due categorie di protocolli di routing:

- **Interior Router Protocols (IRPs)**: opera all'interno di sistemi autonomi (AS). I protocolli di routing più diffusi sono Open Shortest Path First (OSPF) Protocol e Enhanced Interior Gateway Routing Protocol (EIGRP).
- **Exterior Router Protocols (ERPs)**: opera attraverso sistemi autonomi. L'ERP è tipicamente eseguito solamente nei nodi a margine ed il protocollo di routing più utilizzato è il Border Gateway Protocol (BGP).

Tradizionalmente le varie funzioni di routing sono distribuite all'interno dei router che si trovano nella rete. Tuttavia, in una rete SDN-controlled, ha più senso centralizzare le funzioni di routing nel controller SDN poiché il controller può sviluppare una vista della rete più consistente ed aiutare a calcolare cammini più brevi all'interno della rete oltre ad aiutare l'implementazione delle applicazioni tenendo conto delle politiche di routing.

L'applicazione centralizzata del routing rappresenta due distinte funzioni:

- **Link Discovery**: le funzioni di routing devo essere a conoscenza dei link attraverso gli switches nel data plane. Nelle reti tradizionali il link attraverso i network avviene in maniera diretta come nel caso degli Ethernet switches. In addizione il link discovery deve essere realizzato attraverso un router ed un sistema di host e attraverso il router nel dominio del suo controller e il router nel dominio adiacente.
- **Topology Manager**: mantiene la topologia di informazione per la rete e calcolerà le strade attraverso la rete. Il calcolo della rete determina il cammino più corto tra due nodi nel data plane o tra un nodo del data plane e un host.

6.2 ITU-T Model

In questo paragrafo descriveremo l'architettura SDN ad alto livello definita in ITU-T Y.3300.

L'**application layer** è dove le applicazioni SDN specificano servizi network o applicazioni business definendo le risorse della rete o i servizi fondamentali per loro. Le applicazioni interagiscono con l'SDN control layer attraverso le APIs che formano l'interfaccia application-control.

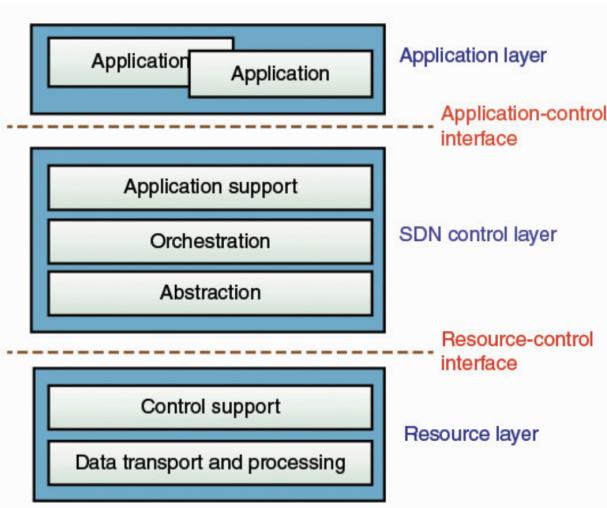


Figura 6.3: High Level Architecture of SDN(ITU-T Y.3300)

Il **control layer** racchiude un significato di controllo del comportamento delle risorse della rete. Il control layer può essere visto come i seguenti sublayers:

- **Application support:** fornisce un'API per le applicazioni SDN per accedere alle informazioni della rete e programmare specifici comportamenti della rete.
- **Orchestration:** fornisce il controllo automatico e la gestione delle risorse della rete; inoltre coordina le richieste dall'application layer verso le risorse

della rete. Orchestration racchiude topologie di rete sia fisiche che virtuali, elementi della rete, controllo del traffico e altri aspetti correlati alla rete.

- **Abstraction:** interagisce con le risorse della rete e consente un'astrazione delle risorse della rete in tutte le sue caratteristiche a supporto del management e dell'orchestration della rete.

Il **resource layer** consiste nell'interconnessione degli elementi di instradamento del data plane, gli switches. Collettivamente questi ultimi consentono di trasportare e di processare i pacchetti di dati in accordo con l'SDN control layer. La maggior parte di questi controlli avviene per conto delle applicazioni. Il resource layer può essere visto come i seguenti sublayers:

- **Control support:** supporta la programmabilità delle funzioni del resource-layer attraverso l'interfaccia del resource control.

- **Data transport and processing:** permette l'inoltro dei dati e le funzioni di routing.

6.3 Rest

REpresentational State Transfer (REST) è uno stile di architettura utilizzato per definire le APIs. È diventato lo stile di riferimento per le NorthBound APIs per i controller SDN. Una "REST API", o una API che si dice "RESTful" (ovvero che aderisce ai vincoli di REST) non è un protocollo, linguaggio o uno standard prestabilito ma rappresenta essenzialmente sei regole che le API devono obbligatoriamente seguire per definirsi RESTful.

REST CONSTRAINTS:

- **Client-server:** separazione fra interfaccia utente e dati salvati.
- **Stateless:** il server non memorizza nessun record dell'utente.

- **Cache:** ogni dato deve avere un'etichetta in cui è scritto se è "cacheable" oppure no. Nel caso in cui lo sia questa potrà essere riutilizzata in seguito.
- **Uniform interface:** REST enfatizza un'interfaccia uniforme tra i vari componenti. Per far questo REST definisce quattro regole:
 1. Le risorse sono identificate mediante un identifier, ad esempio un URI.
 2. Le risorse sono rappresentate in formati come JSON, XML, HTML.
 3. Ogni messaggio ha abbastanza informazioni per descrivere come viene processato.
 4. Un client non deve essere a conoscenza di come interagisce con il server.
- **Layered system:** una funzione è organizzata in livelli ed ogni livello comunica direttamente con il livello sotto o il livello sopra.
- **Code on demand:** consente al cliente di essere scaricato per aumentare l'estensibilità del codice.

6.3.1 Esempio di REST API

Consideriamo la funzione che descrive tutte le entità di una group table di un particolare switch. L'URI di questa funzione è il seguente:

`/stats/group/<dpid>`

dove stats (statistiche) si riferisce al set di APIs che ricevono e aggiornano le statistiche e i parametri degli switch, mentre `<dpid>` (Data Path ID) è l'identificatore univoco per lo switch. Per invocare la funzione per lo switch 1 è necessario il seguente comando allo switch manager attraverso la REST API:

`GET http://localhost:8080/stats/groupdesc/1`

IL **localhost** indica che l'applicazione sta girando sullo stesso server. Nel caso in cui fosse stata remota l'URI sarebbe stato un URL che riesce ad accedere via HTTP attraverso il web.

6.4 Cooperazione e coordinazione fra Controllers

In addizione alle northbound e southbound interfaces, un tipico controller SDN ha una east/westbound interface che gli consente di comunicare con altri controller ed altre reti. Attualmente non ci sono stati significanti progressi su protocolli open source o protocolli standardizzati per le est/westbound interfaces.

6.4.1 Controller Centralizzati vs Controller Distribuiti

Un controller centralizzato è un singolo server che controlla tutto il data plane degli switches all'interno della rete.

Nelle grandi infrastrutture di rete, l'utilizzo di un solo controller che gestisce tutti i dispositivi network della rete risulta scomodo e indesiderato. Uno scenario più gradito sarebbe che venisse divisa la rete in un numero domini SDN non sovrapposti, chiamati anche *SDN Islands* (figura 5.4) gestite da controller distribuiti.

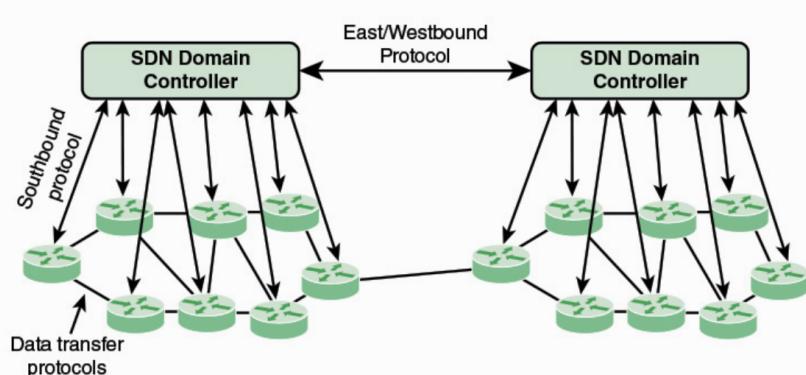


Figura 6.4: Struttura Dominio SDN

Le ragioni per utilizzare i domini SDN sono le seguenti:

- **Scalability:** il numero di devices che un controller SDN può fisicamente gestire è limitato e proprio per questo motivo una rete molto grande ha bisogno di utilizzare più controller SDN.
- **Reliability:** l'uso di controllers multipli evita il rischio di avere un singolo punto di errore.
- **Privacy:** un operatore può scegliere di implementare differenti politiche di privacy in differenti domini SDN.
- **Incremental Deployment:** un operatore di rete può essere costituito da porzioni di infrastrutture datate e non. Dividendo la rete in dominio SDN multipli controllabili permette di ottenere flessibilità nell'utilizzo e per gli sviluppi futuri.

I controller distribuiti possono essere collocati in piccole aree, in aree molto vasta oppure in una combinazione di esse. Controllers installati ravvicinati offrono un'alta produttività e sono appropriati per i data centers; invece controllers installati lontano possono contenere reti multilocation.

Tipicamente i controller distribuiti sono installati orizzontalmente ed in un'architettura distribuita è necessario un protocollo per permettere ai controller di comunicare.

6.4.2 Border Gateway Protocol

Il Border Gateway Protocol (BGP) è stato sviluppato per usare in congiunzione con i protocolli TCP/IP di Internet ed è diventato l'**exterior router protocol (ERP)** predefinito per Internet.

BGP abilita i router, comunica con i gateways ed in differenti sistemi autonomi permettendo loro di comunicare e di scambiare informazioni di routing.

Il protocollo opera attraverso dei messaggi che sono inviati attraverso connessioni TCP.

Ci sono tre funzionalità coinvolte nel BGP:

- **Neighbor acquisition:** il termine **neighbor** indica che due router condividono la stessa rete. La neighbor acquisition avviene quando due router in due sistemi autonomi differenti concordano di scambiarsi informazioni regolarmente.
- **Neighbor reachability:** viene utilizzata per mantenere la relazione. Ogni partner deve essere sicuro che l'altra parte continua ad esistere e fa ancora parte della relazione stabilita in precedenza dai router. Per far ciò i router periodicamente si scambiano dei messaggi di *Keepalive*.
- **Network reachability:** ogni router mantiene un database delle reti che può raggiungere e la strada preferita per raggiungere ogni rete. Quando viene modificato qualcosa nel database il router invia un messaggio di *Update* agli altri routers.

Capitolo 7

The Internet of Things: Components

7.1 L'inizio dell'era IOT

L'Internet del futuro coinvolgerà un vastissimo numero di oggetti che useranno uno standard di architettura comune per portare determinati servizi agli utenti. Si prevede infatti che ci saranno decine di miliardi di questi dispositivi interconnessi fra loro nei prossimi anni e questo produrrà nuove interazioni fra il mondo fisico e quello digitale. La risultante di questo paradigma della rete prende il nome di *Internet of Things* (*IOT*) e produrrà un grandissimo numero di opportunità per gli utenti, per le imprese e per chi fornisce servizi in svariati settori. In particolare le aree che beneficeranno maggiormente dello sviluppo dell'*IOT* saranno la collezione e l'analisi dei dati, automazioni, incluse nel settore del benessere e del fitness, automazioni per il controllo della casa, per risparmiare energia, per la produzione, per il trasporto, per il controllo ambientale, per lo stoccaggio e la produzione di prodotti, sicurezza e sorveglianza e molto, molto altro ancora.

Lo sviluppo tecnologico è ancora necessario in molte aree. Infatti negli ultimi anni sono stati condotti molti investimenti in ricerca e sviluppo delle reti wireless. Attualmente la ricerca e lo sviluppo di ricercatori e produttori è indirizzato su protocolli a basso consumo energetico, sicurezza e privacy, protocolli di rete ad alta efficienza energetica per una lunga durata della batteria, affidabilità delle reti e nodi. Questi sviluppi wireless sono cruciali per la crescita dell'IoT.

Ci sono altre aree in cui è stato coinvolto lo sviluppo dei devices IoT come ad esempio la capacità di social networking da parte dei dispositivi, sfruttando comunicazioni *machine-to-machine*, processando e salvando un vastissimo numero di dati in real-time e sviluppando applicazioni che fornissero agli utenti finali interfacce utili e intelligenti per questi dispositivi e dati.

7.2 Gli Scopi dell'Internet of Things

Possiamo fornire le seguenti definizioni per definire gli scopi dell'IoT:

- **Internet of Things:** un'infrastruttura globale per le informazioni della società, che abilita servizi avanzati grazie all'interconnessione (fisica e virtuale) degli oggetti basandosi su tecnologie e informazioni attualmente esistenti ed in via di sviluppo.
- **Thing:** questo è un oggetto del mondo fisico o virtuale capace di essere identificato ed integrato all'interno di reti di comunicazione.
- **Device:** è una parte dell'IoT con la capacità obbligatoria di comunicare e la capacità opzionale di fare sensing, acuazione, salvataggio dei dati e lavorazione di dati.

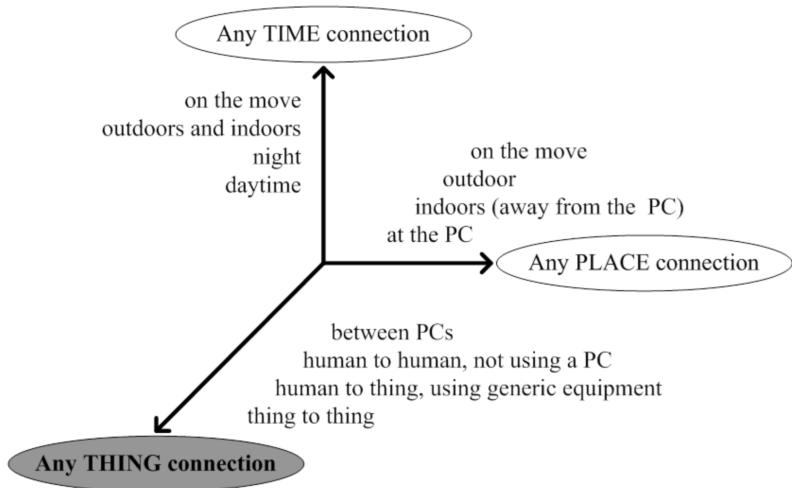


Figura 7.1: La nuova dimensione introdotta nell’Internet of Things

L’equazione che può riassumere tutto il complesso sistema dell’IoT potrebbe essere la seguente:

$$\text{Oggetti Fisici} + \text{Controller, Sensori, Attuatori} + \text{Internet} = \textbf{IoT}.$$

7.3 Componenti di oggetti abilitati per l’IoT

Gli ingredienti chiavi per un sistema di oggetti abilitati all’IoR sono sensori, attuatori, microcontrollers, un mezzo di comunicazione (transceiver) ed un mezzo di identificazione (RFID: radio-frequency identification). Il mezzo di comunicazione è un ingrediente fondamentale senza il quale i dispositivi non potrebbero partecipare in una rete. [6]

7.3.1 Sensori

Un sensore misura determinati parametri fisici, chimici o biologici e invia un segnale elettronico proporzionato con le caratteristiche rilevate, o in formato digitale oppure sotto forma di un livello di tensione analogico. In

entrambi i casi, l'output del sensore è tipicamente l'input di un microcontrollore o di un altro elemento di gestione.

Un sensore può operare in *modalità attiva* quando prende l'iniziativa di inviare i dati rilevati al controller o periodicamente o quando una determinata soglia è superata. Alternativamente il sensore può operare in *modalità passiva* inviando dati solo quando è richiesto dal controller.

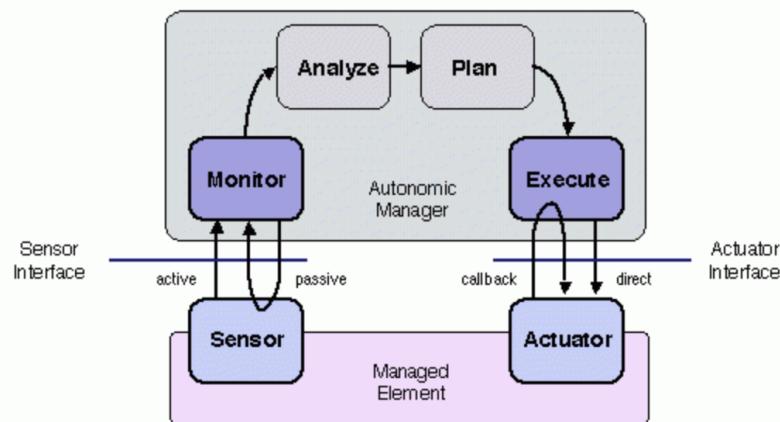


Figura 7.2: Interfacce per sensori e attuatori

La tipologie di sensori utilizzati nell'IoT sono moltissime. Ci sono sensori che sono estremamente piccoli, usando nanotecnologie, oppure sensori estremamente grandi come camere di videosorveglianza.

Ci sono due concetti chiavi nel distinguere le tipologie di sensori:

- **Accuracy:** si riferisce a quanto una misurazione si avvicina alla verità.
- **Precision:** si riferisce a quanto sono vicine tra loro più misurazioni della stessa quantità fisica. Collegata con questo concetto troviamo la **Resolution**, ovvero la qualità dell'output del sensore.

Se un sensore ha bassa accuracy, questo produce un errore sistematico. Se un sensore ha bassa precisione, produce un errore di riproducibilità.

7.3.2 Attuatori

Un attuatore riceve un segnale elettronico dal controller e risponde interagendo con l'ambiente per produrre un effetto su qualche parametro fisico, biologico o chimico di una determinata entità.

Nella modalità di operazione *direct mode*, il controller invia un segnale che attiva gli attuatori. Invece nella *callback mode* gli attuatori rispondono al controller per riportare un completamento o un problema e richiedono ulteriori istruzioni.

Gli attuatori sono generalmente classificati come segue:

- **Idraulici**: sono costituiti da un cilindro o un motore fluido che utilizza la forza idraulica per facilitare processi meccanici.
- **Pneumatici**: lavorano come quelli idraulici solo che utilizzano gas al posto di liquido.
- **Elettrici**: sono dispositivi alimentati da motori che convertono l'energia elettrica in coppia meccanica.
- **Meccanici**: funzionano mediante movimento rotazionale o lineare e sono utilizzati per convertire il movimento.

7.3.3 Microcontrollers

La cosa smart nei dispositivi che si definisco tali è fornita da un microprocessore perfettamente integrato. Adesso vedremo alcuni termini chiave per esplicare il concetto di microcontroller.

Embedded System

Il termine *embedded system* si riferisce all'uso di elettronica e di software all'interno di un dispositivo che ha una o più funzioni.

Application Processors vs Dedicated Processor

Application processor sono definiti dall'abilità del processore di eseguire complessi sistemi operativi. Un classico esempio dell'uso di embedded application processor è lo smartphone, poiché è designato ad usare svariate applicazioni e svolgere molte funzioni diverse.

La maggior parte degli embedded systems utilizza un *dedicated processor*, il quale è dedicato ad una o ad un piccolo numero di specifiche funzionalità richieste dall'host.

Micropocessors

Un processore i cui elementi sono stati miniaturizzati in un unico o in più circuiti integrati.

Microcontrollers

Un singolo chip che contiene il processore, la memoria non volatile (ROM), la memoria volatile per input/output (RAM), un clock ed una unità di controllo I/O. Il loro impiego è fondamentale per utilizzare in maniera sostanzialmente differente lo spazio logico sul disponibile.

Deeply Embedded System

È un sottoinsieme degli embedded system e possiamo dire che ha un processore il cui comportamento è difficile da osservare sia da parte del programmatore che dall'utente. Un deeply embedded system utilizza un microcontrollor, non è programmabile una volta che la logica del programma per il dispositivo è stata scritta nella ROM e non ha nessuna interazione con l'utente.

7.3.4 Transceivers

Un transceiver contiene i componenti elettronici necessari per ricevere e trasmettere dati. La maggior parte dei dispositivi IoT contiene un transceiver wireless, capace di comunicare utilizzando la Wi-Fi, ZigBee ed altri schemi

wireless.

7.3.5 RFID

La tecnologia Radio-frequency identification (RFID) utilizza onde radio per identificare oggetti e sta diventando una tecnologia che facilita sempre di più l'IoT.

Gli elementi principali per un sistema RFID sono i *tags* ed i *readers*.

I *tags* sono piccoli oggetti programmabili usati per il monitoraggio di oggetti, animali ed esseri umani. Sono disponibili in varie forme, dimensioni, funzionalità e costi.

I *readers* acquisiscono e qualche volta riscrivono le informazioni salvate sui tags che rientrano nel loro raggio di azione.

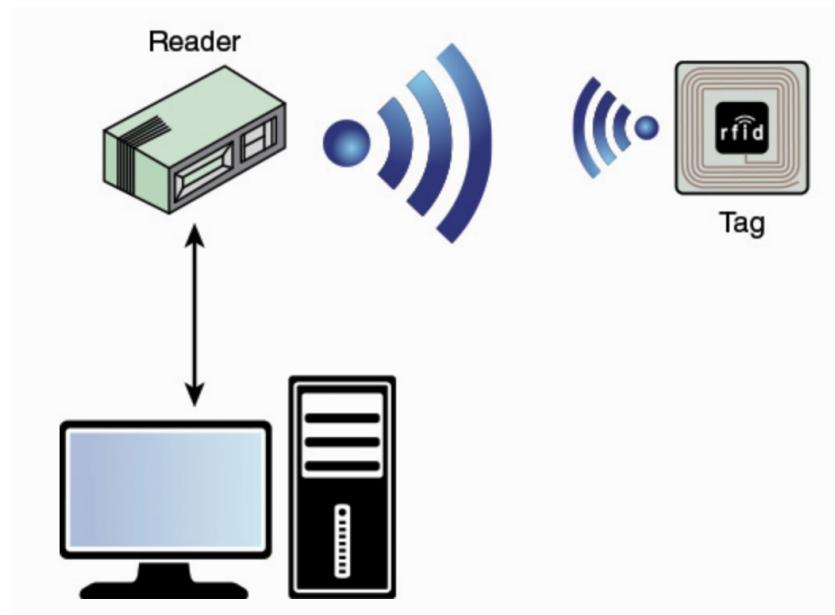


Figura 7.3: Elementi di un sistema RFID

Capitolo 8

The Internet of Things: Architecture and Implementation

8.1 IoT Architecture

Data la complessità dell'IoT, è utile avere un'architettura che specifica gli elementi chiave e la loro interconnessione. Un'architettura IoT può avere i seguenti benefici:

- avere una lista con la quale essere in grado di valutare funzionalità e completezza delle varie offerte proposte dai vendori.
- fornisce una guida agli sviluppatori su quali funzioni sono necessarie nell'IoT e come quest'ultime funzionano insieme.
- può servire come un framework per la standardizzazione, promuovere l'interoperabilità e ridurre i costi. [7], [8]

8.1.1 ITU-T IoT Reference Model

Il modello ITU-T si concentra in maggior dettaglio sui componenti fisici attuali dell'ecosistema IoT. Questa è un'analisi fondamentale poiché rende

visibili gli elementi dell'IoT che devono essere interconnessi, integrati, gestiti e resi disponibili per le applicazioni.

Devices

L'unico aspetto dell'IoT, comparato con altri sistemi network, è la presenza di un numero di oggetti fisici e dispositivi diverso dai dispositivi informatici o di elaborazione dati. Il modello ITU-T vede l'IoT funzionante come una rete di dispositivi che sono strettamente collegati agli oggetti. Sensori ed attuatori interagiscono con gli oggetti fisici nell'ambiente.

I dispositivi di acquisizione dati leggono / scrivono dati su oggetti fisici tramite l'interazione con un dispositivo di trasporto/supporto dati associato a un oggetto fisico.

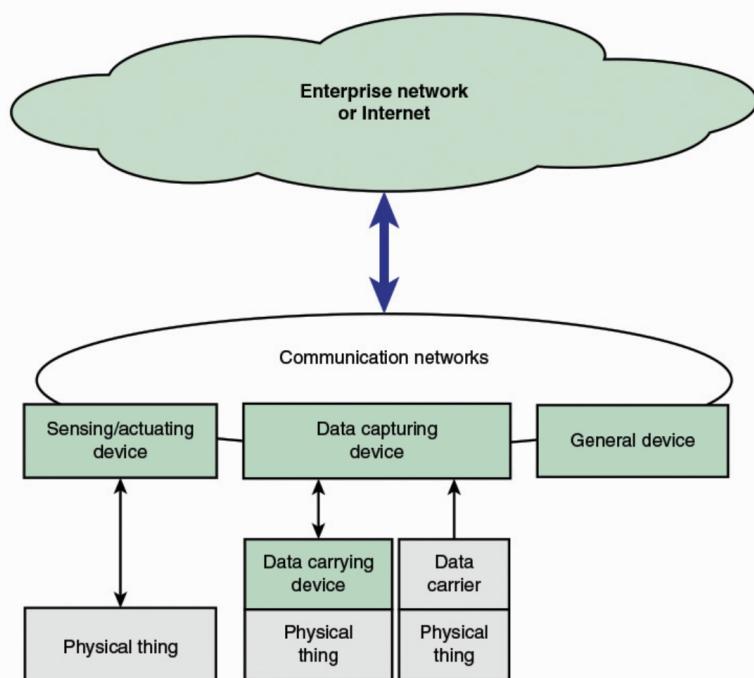


Figura 8.1: Tipi di dispositivi e la loro relazione con gli oggetti fisici

The Reference Model

Il modello di riferimento del modello IOT ITU-T consiste in quattro strati con capacità di gestione e di sicurezza che attraversano tutti gli strati.

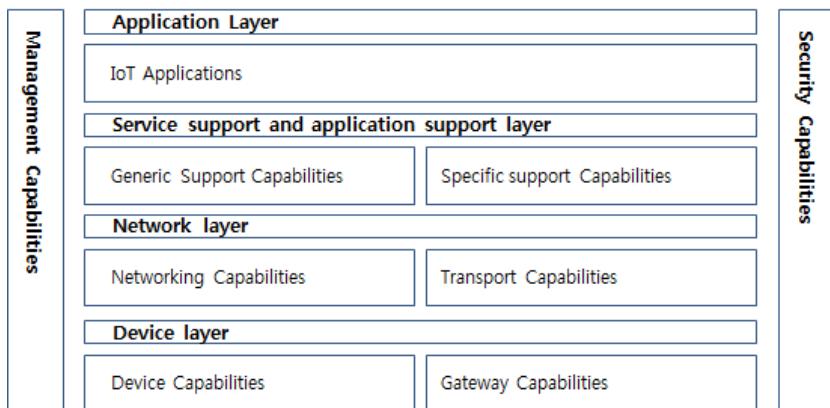


Figura 8.2: ITU-T Y.2060 IoT Reference Model

Il **network layer** esegue due funzioni basiche. Le "networking capabilities" si riferiscono all'interconnessione fra i dispositivi e le porte. Le "transport capabilities" si riferiscono al trasporto dei vari servizi dell'IoT oltre alle informazioni collegate di controllo e gestione.

Il **service support and application support layer** fornisce delle funzionalità che vengono usate dalle applicazioni. Un esempio comune riguarda l'elaborazione dei dati e la gestione del database.

L'**application layer** consiste di tutte le applicazioni che interagiscono con i dispositivi IoT.

Il **management capabilities layer** ricopre le tradizionali funzioni di rete come la gestione di guasti, configurazione e gestione delle prestazioni.

Il **security capabilities layer** include funzionalità di sicurezza generiche indipendenti dalle applicazioni.

8.1.2 IoT World Forum Reference Model

L'IoT World Forum è un evento annuale sponsorizzato dalle aziende che raggruppa insieme rappresentanti del business, governi ed accademie per promuovere lo sviluppo e l'adozione nel mercato dell'IoT.

Il comitato per l'architettura nell'IoT World Forum, che include fra le molte aziende colossi internazionali del calibro di IBM, Intel e Cisco i quali hanno rilasciato un modello di riferimento per l'IoT nel 2014. Questo modello è servito come framework di riferimento per aiutare l'industria ad accelerare lo sviluppo dell'IoT.

Questo modello di riferimento è un elemento complementare al modello di riferimento ITU-T, con la differenza che quest'ultimo si concentra maggiormente a livello di dispositivi e porte. Invece il modello IoT World Forum si concentra in maniera più ampia sullo sviluppo di applicazioni, middleware e funzioni di supporto per l'IoT aziendale.

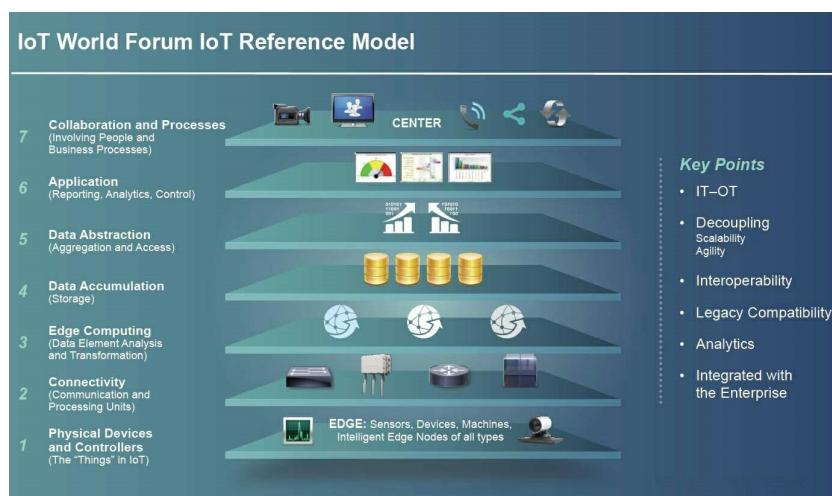


Figura 8.3: IoT World Forum Reference Model

- **Semplifica:** aiuta ad abbattere un sistema complesso rendendolo più comprensibile.

- **Chiarifica:** fornisce informazioni aggiuntive per identificare precisamente i livelli dell'IoT e stabilisce una terminologia comune.
- **Identifica:** identifica dove specifici tipi di processi sono ottimizzati nelle diverse parti del sistema.
- **Standardizza:** fornisce un aiuto alle industrie per permetter loro di creare prodotti IoT che lavorino fra di loro.
- **Organizza:** rende reale ed accessibile l'IoT, invece che semplicemente concettuale.

8.2 IoT Implementation

Abbiamo visto due modelli di riferimento, i quali forniscono un'ottima panoramica delle funzionalità ricercate durante la progettazione di un sistema IoT. Vediamo adesso un esempio della distribuzione di dispositivi e software IoT. [9], [10]

8.2.1 Cisco IoT System

Nel 2015 Cisco ha introdotto una suite di prodotti integrati e coordinati, noti come *Cisco IoT System*. La filosofia guida che ha guidato l'azienda era la previsione che entro la fine del 2020 ci fossero almeno 50 miliardi di dispositivi connessi ad Internet. Attualmente circa il 99% degli oggetti nel mondo non è connesso ad internet, ma nel lungo processo di digitalizzazione che stanno seguendo le industrie e le città sono sempre più diffuse le soluzioni IoT.

Cisco IoT System affronta la complessità della digitalizzazione offrendo una infrastruttura designata per gestire sistemi su larga scala di diverse piattaforme e il flusso di dati che queste creano. Il sistema consiste in una architettura

basata su sei pilastri con l'obiettivo di ridurre la complessità della digitalizzazione; inoltre Cisco ha proposto un buon numero di prodotti IoT ed il continuo rilascio e sviluppo di nuovi dispositivi.

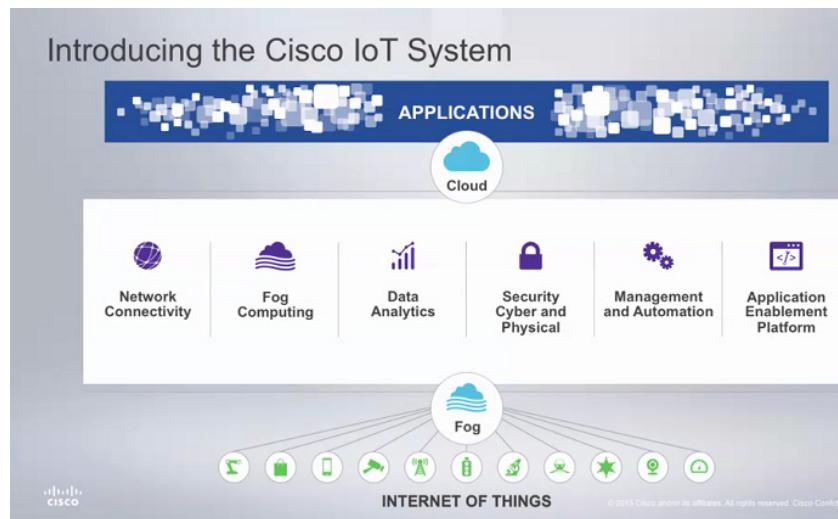


Figura 8.4: Cisco IoT System

- **Networking connectivity:** include prodotti di routing, switching e wireless appositamente progettati.
- **Fog computing:** fornisce il fog computing di Cisco o la piattaforma di elaborazione dati IOx.
- **Data analytics:** un'infrastruttura ottimizzata per implementare l'analisi dei dati, sfruttando sia il *Cisco Connected Analytics Portfolio* che software di terze parti.
- **Security:** unifica la cyber-security e la physical-security per fornire vantaggi operativi e incrementare la protezione sia per le risorse fisiche che digitali. Un esempio è il TrustSec di Cisco.
- **Management and automation:** strumenti per la gestione degli endpoints e delle applicazioni.

- **Application enabled platform:** un set di APIs per permettere di sviluppare e produrre applicazioni compatibili con le capacità del sistema IoT.

Capitolo 9

Architettura del sistema

L'idea alla base del sistema IoT è quella di progettare una catena di produzione tipica di un'industria 4.0 attraverso l'utilizzo di SDN-Wise. Per lo sviluppo del sistema sono state affiancate le tecnologie di SDN-Wise, approfondite nei capitoli precedenti, con metodologie per effettuare le lavorazioni e monitorare l'andamento della catena industriale con prevenzione per guasti e malfunzionamenti.

Il sistema progettato prevedeva già all'interno tutti i meccanismi previsti dal SDN-Wise e sono stati implementati i seguenti meccanismi per la realizzazione finale:

- **sondaggio:** per monitorare l'ambiente è stato implementato un sistema di consenso distribuito, nella quale ogni nodo che partecipa al sondaggio comunica con gli altri nodi partecipanti, invece di avere una comunicazione unicast con il Controller.
- **attuatori e meccanismo di prevenzione guasti:** per la prevenzione guasti o errore di lavorazione durante la catena industriale sono stati implementati dei meccanismi di allarme e dei nuovi mote che compiessero delle azioni per salvaguardarlo.

9.1 Attori partecipanti

Nel sistema progettato gli attori principali sono tre e partecipano tutti attivamente per la realizzazione della catena industriale con lavorazione e risposta ai guasti e agli errori di lavorazione. In particolare questi tre elementi sono **Controller**, **Sink** e **Mote**. Di seguito verranno analizzati singolarmente in maniera più approfondita.

9.1.1 Controller

Il Controller è un entità logica centralizzata, che ha la visione completa e aggiornata della topologia della rete e delle relazioni tra i nodi. Per interfacciarsi con la rete, esso apre una comunicazione TCP con il Sink, attraverso la quale il Controller può inviare pacchetti per la modifica della topologia della rete o per ricevere pacchetti sullo stato dei nodi o delle richieste da parte di essi.

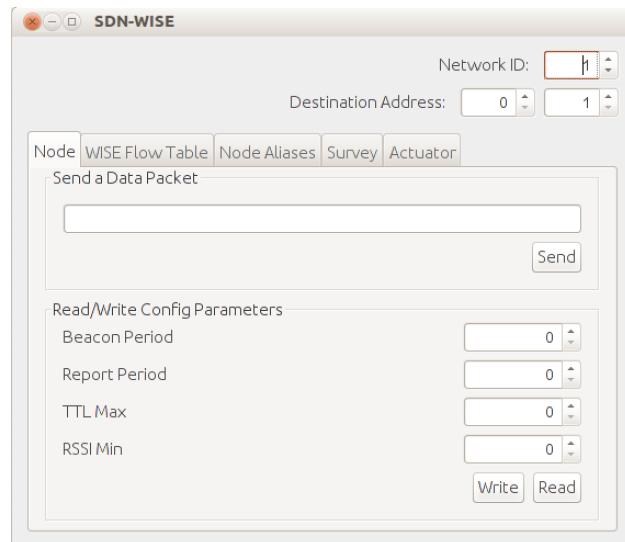


Figura 9.1: Pannello del Controller

Nel sistema progettato il Controller si presenta come nella figura soprastante. Questo pannello da la possibilità all'Operatore, che ha il compito di monitorare la catena industriale e di avviare le lavorazioni, di inviare pacchetti di vario genere per monitorare il funzionamento dell'intero sistema o per modificare la struttura topologica.

Per il monitoraggio della rete, nel sistema è previsto un pannello che riproduce la topologia della rete basandosi sui pacchetti dei report, pacchetti nel quale ogni nodo specifica i suoi vicini. Un esempio è mostrato in figura:

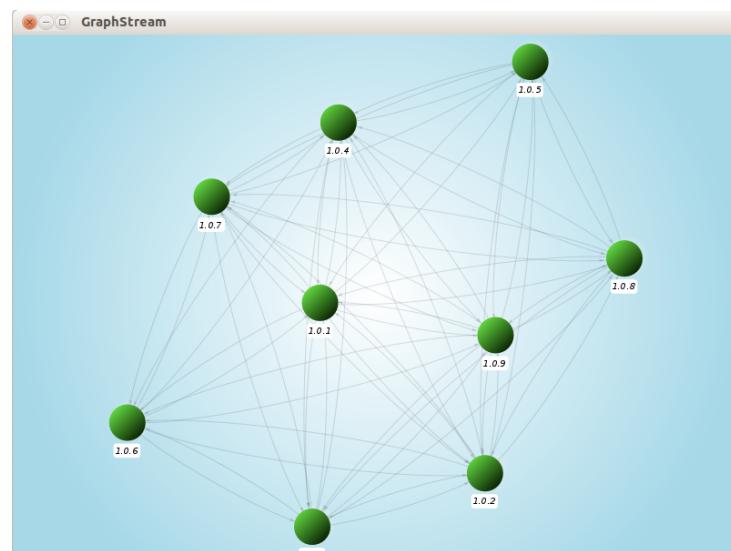


Figura 9.2: Topologia Rete

Survey e Actuator

All'interno del pannello del controller sono presenti due campi, Survey e Actuator.

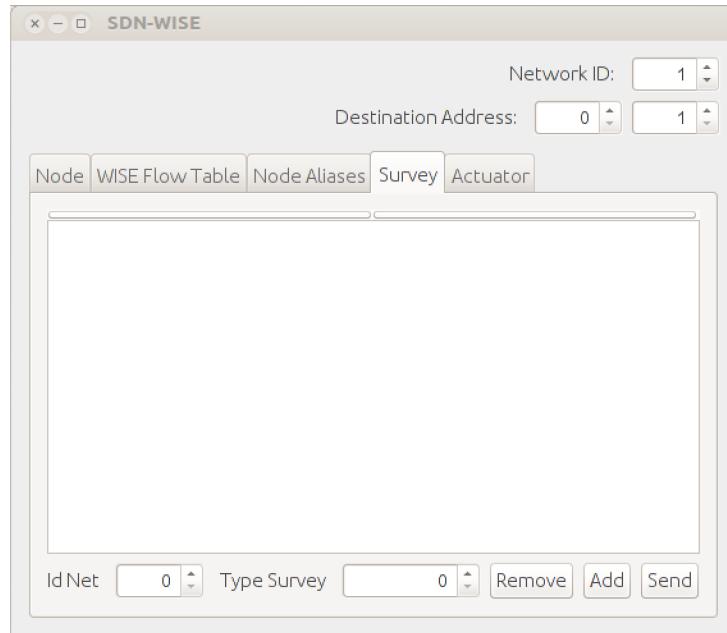


Figura 9.3: Pannello Survey

Quest'ultimi presentano una tabella a due colonne che rappresentano rispettivamente l'High Adress e Low Adress degli indirizzi dei nodi che partecipano. I due Spinner in basso al pannello rappresentano ID della rete, che andrà a inserirsi nel primo byte dell'Header, e il TYP(azione o sondaggio), che sarà il primo byte del payload.

Di fianco ai due Spinner, sono presenti tre buttoni, remove, Send e Add; questi tre attivano tre metodi differenti:

- **Remove**: rimuove un nodo dalla tabella.
- **Add**: inserisce un nuovo nodo nella tabella; per inserirlo, una volta cliccato il pulsante add apparirà il seguente pannello aggiuntivo:

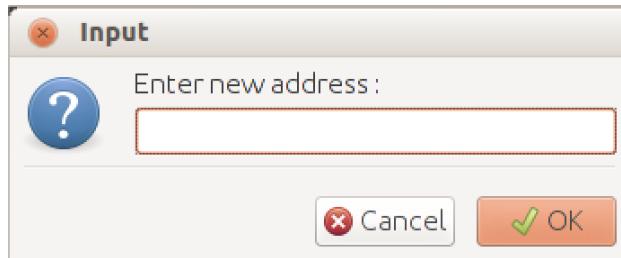


Figura 9.4: Pannello newAddress

- **Send:** invia il pacchetto al Sink.

I nodi della tabella, ogni volta che viene compiuto l'evento add, vengono aggiunti a una lista di NodeAddress; premuto il tasto send, invece, viene creata la lista di valori, inizialmente inizializzata a zero, e viene creato il pacchetto per inviarlo al Sink.

9.1.2 Sink

Il Sink è un nodo della rete ed è l'unico nodo che può dialogare con il Controller; questo fa sì che gli altri nodi della rete conoscano il suo indirizzo in modo tale che tutti possano inviare richieste o report al Controller. Dal Sink passano tutti i pacchetti, questo lo rende il perno fondamentale della rete ed è grazie a lui che il Controller può conoscere ogni dettaglio della rete. All'interno del framework *Cooja* il Sink è raffigurato con il colore verde.

9.1.3 Mote

I mote sono i restanti nodi della rete, i compiti principali sono di dialogare tra loro e con il Sink attraverso l'invio di pacchetti *beacon, report, request*. Nel sistema progettato i mote si dividono in due categorie:

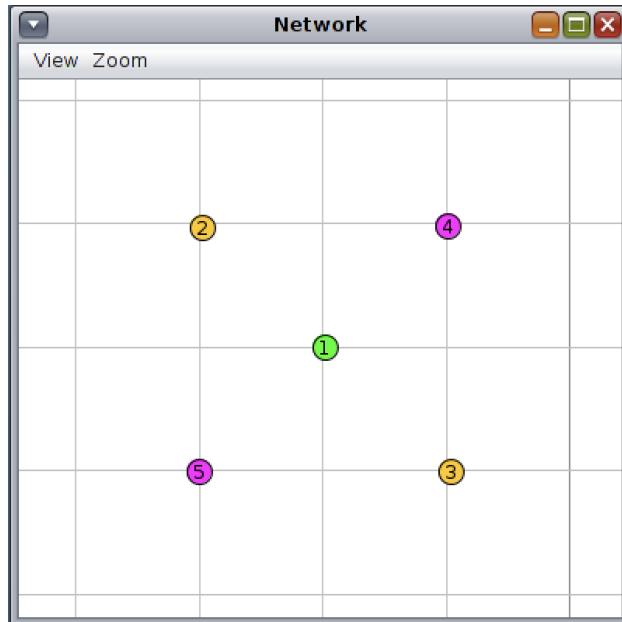


Figura 9.5: Sink, Mote e Actuator Mote

- **Mote:** hanno il compito di monitorare l'ambiente, monitorando i dati richiesti dal sondaggio.
- **Actuator Mote:** gli Actuator Mote hanno il compito di attuare un'azione specifica per prevenire un danno ambientale.

All'interno del framework *Cooja* il Mote è raffigurato con il colore giallo, mentre l'Actuator Mote con il colore viola.

Capitolo 10

Framework utilizzati

La maggior parte delle sfide legate alle WSN non sono ancora state gestite adeguatamente. Anche se il paradigma SDN promette un enorme riduzione del consumo di energia da parte dei nodi, le entità di queste affermazioni devono essere valutate e quantificate. A tal proposito SDN-WISE offre un simulatore di rete, chiamato Cooja, che gira sul sistema operativo Contiki e consente di semplificare la produzione di applicazioni per le WSN, nonché la simulazione per testing e debugging. Inoltre può essere utilizzato a scopo didattico per gli studenti. Le simulazioni vengono svolte con topologie di rete differenti, così come il numero di messaggi circolanti in rete verrà variato. [11]

10.1 Contiki

Contiki è il sistema operativo open source per creare reti IoT wireless a bassa potenza. È stato sviluppato presso lo Swedish Institute of Computer Sciences da Adam Dunkels e il suo team. È scritto nel linguaggio di programmazione C, così come ogni sua estensione. Contiki è un sistema operativo altamente portatile ed è già stato distribuito su diverse piattaforme utilizzanti

diversi tipi di processori. La maggior parte di esse spesso usano Texas Instruments MSP-430 o Atmel ATmega come microcontrollori. Contiki utilizza un modello di programmazione basato su eventi per gestire la programmazione concorrente. Il suo principale vantaggio è che tutti i processi condividono uno stack consentendo un oneroso risparmio di memoria. Questo modello viene realizzato mediante dei Protothread, i quali forniscono blocking wait, condizionato o meno, per salvare lo stato in modo continuativo. Quando il Protothread viene riattivato, riparte dall'istruzione successiva. Contiki usa anche il cosiddetto Rime stack, una pila di comunicazione per reti di sensori molto leggera, in quanto gli strati sono semplici e hanno piccole intestazioni di pochi byte. Rime supporta anche il riutilizzo del codice col principale scopo di semplificare l'implementazione delle WSN. Contiki supporta sia IPv6 che IPv4, nonché i più recenti standard per le reti wireless a bassa potenza, come 6lowpan, RPL e CoAP. La sua popolarità lo ha portato in numerosi sistemi, quali contatori di energia elettrica, monitoraggio industriale, sistemi di allarme, monitoraggio remoto della casa, del suono nella città, delle radiazioni e via dicendo. L'ultima versione Contiki 3.0 è stata rilasciata il 26 agosto 2015

10.2 Cooja

Cooja è il simulatore di rete per Contiki. È un'applicazione Java-based con interfaccia grafica basata sullo standard Java Swing toolkit. Cooja supporta la simulazione per mezzo di onde radio e l'integrazione con tool esterni per fornire funzionalità aggiuntive. L'emulazione può avvenire sia a un livello meno dettagliato, in modo che sia più veloce e consenta la simulazione di reti più grandi, sia a livello hardware, più lento ma che consente un'ispezione

precisa del comportamento del sistema.

Questo strumento ha due pacchetti software per la simulazione: Avrora e MSPSim. Il primo è usato da Cooja per l'emulazione di dispositivi basati su Atmel AVR, mentre il secondo per quelli basati su TI MSP430. La maggior parte delle piattaforme usano Microcontrollori di questo tipo, pertanto MSPSim è il pacchetto software più utilizzato. Cooja può simulare più piattaforme, come TelosB, SkyMote, Zolertia Z1 m- te, Wismote, ESB, MicaZ mote, ed è molto utile per lo sviluppo e il debugging di applicazioni per Contiki in quanto permette agli sviluppatori di:

- testare il loro codice e i sistemi prima di eseguirlo sull'hardware reale.
- stimare i consumi energetici dei nodi
- mostrare trasmissioni e ricezioni radio.

Per avviare Cooja è necessario disporsi nella cartella che lo contiene (nello specifico *sdn-wise-contiki z1/contiki/tools/cooja*) e invocare il comando *ant run* dal terminale. Cooja partirà con una finestra blu vuota non appena la compilazione sarà conclusa.



Figura 10.1: Finestra iniziale simulatore Cooja

La prima cosa da eseguire è aggiungere una estensione che mi permette di

aggiungere i mote e i sink che ho modificato nel programma sdn-wise-java.

Per aggiungere questa estensione occorre cliccare su *Settings -> Cooja Extension*.

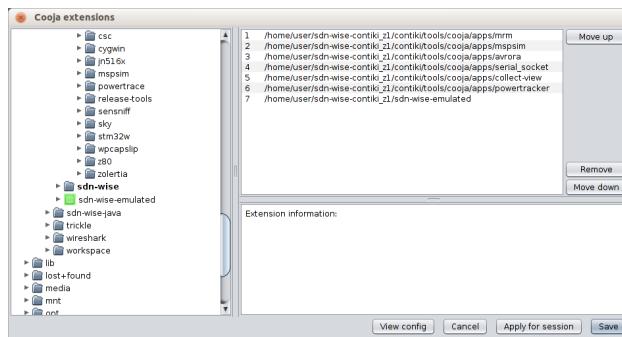


Figura 10.2: Aggiungere estensione per Cooja

Una volta aggiunta questa estensione possiamo creare una nuova simulazione (*File -> New Simulation*). Viene richiesto ora di inserire un nome per la simulazione, dopodiché ci troveremo di fronte al seguente pannello di Cooja:

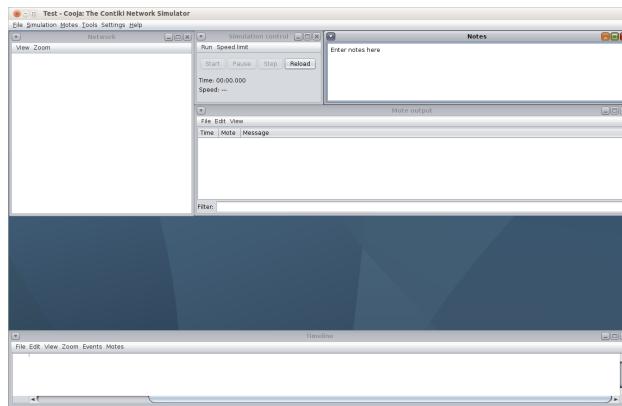


Figura 10.3: Pannello simulazione Cooja

Cliccando su *Mote -> Add motes -> Create new mote type* e scegliere tra i vari *SDN-WISE Emulated Mote/Sink/Actuator Mote/Actuator Sink* in base

alla simulazione che si vuole realizzare.

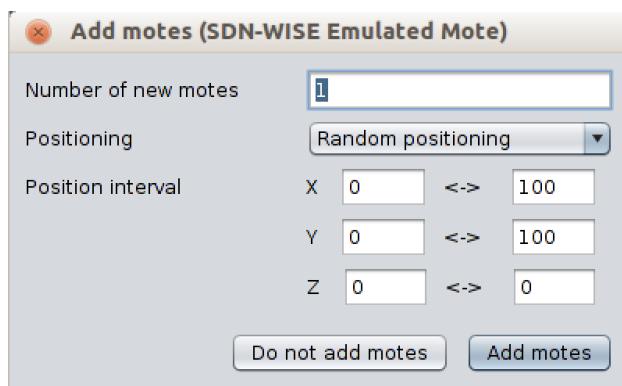


Figura 10.4: Aggiunta Mote

Il simulatore chiederà quanti mote aggiungere e in quale posizione e li inserirà come richiesto.

Nella figura 10.3 sono presenti dei tools di Cooja che verranno descritti brevemente:

- **Network:** mostra la disposizione dei nodi della rete. Si può visualizzare anche la loro posizione il loro Id, il loro tipo, e la copertura radio.
- **Simulation Control:** viene usato per far partire, stoppare, ricaricare la simulazione e indica anche il tempo trascorso.
- **Mote Output:** Mostra tutti gli output dei vari nodi della rete.
- **Notes:** un semplice Notepad per prendere appunti sulla simulazione in corso.
- **Timeline:** mostra la timeline degli eventi come log output, cambio di canale, etc.

10.3 IntelliJ IDEA e Linux Ubuntu 14.04

Tra gli strumenti utilizzati vi sono anche l'ambiente di sviluppo *IntelliJ IDEA* sviluppato da *JetBrains*, e il sistema operativo *Linux Ubuntu 14.04* alla base della macchina virtuale ospitante il framework.

Una volta lanciata la macchina virtuale è necessario eseguire queste semplici operazioni per eseguire il programma *IntelliJ Idea*:

- 1 - aprire il terminale sulla macchina virtuale
- 2 - digitare `cd /opt/idea/bin` e premere invio
- 3 - digitare `./idea.sh` e premere invio

A questo punto si aprirà il programma *IntelliJ Idea* e qualora non fosse stato ancora fatto è necessario caricare la cartella *sdn-wise-java* e tutti i suoi files.

Nello specifico la cartella *sdn-wise-java* contiene tre macro package:

- **core**: contiene tutte le informazioni riguardante i pacchetti scambiati dai nodi SDN-WISE e le informazioni che riguardano le Flow Tables.
- **control**: contiene tutte le informazioni riguardante il Controller, dalla sua creazione alle sue funzionalità; contiene il package network graph, dove al suo interno viene creato il visual network graph. Inoltre è presente anche il package loader dove è presente il file che carica il progetto SDN-WISE.
- **data**: contiene tutte le informazioni riguardante i nodi della rete, dalla creazione alle loro funzionalità e tutte le informazioni sulla batteria.

Una volta effettuati i cambiamenti al codice ed ai vari file, sempre all'interno di *IntelliJ Idea* è necessario eseguire in sequenza queste operazioni:

- cliccare su *View -> Tool Window -> maven*

Una volta che si sarà aperto il pannello eseguire in successione i seguenti

comandi:

- *clean*
- *install*.

Infine sempre da terminale eseguire: *sudo cp -avr /home/user/sdn-wise-
java/data/target/sdn-wise-data-4.0.1-SNAPSHOT-jar-with-dependencies.jar
/home/user/sdn-wise-contiki z1/sdn-wise-emulated/lib/sdn-wise-data.jar*

A questo punto è possibile lanciare la simulazione come abbiamo visto in precedenza. Per semplificare tutte queste operazioni ripetitive e meccaniche, è stato creato uno script. Il file in questione è stato chiamato *compile.sh* ed una volta eseguite le varie modifiche è sufficiente scrivere direttamente nel terminale di *IntelliJ Idea*:

- *bash compile.sh*

A questo punto, mentre contestualmente viene lanciata la simulazione, basterà cliccare il pulsante *run* su *IntelliJ Idea* ed appariranno a schermo il controller SDN-WISE ed il GraphStream.

Capitolo 11

Simulazioni

11.1 Scenari Analizzati

In questo capitolo verranno discussi quattro tipi di simulazioni effettuate sul sistema per simulare vari scenari reali che si potrebbero presentare in una catena industriale. Per spiegare al meglio la simulazione e come i vari componenti del sistema interagiscono fra di essi, sono presenti un flow chart ed un sequence diagram per ogni tipo di scenario possibile.

Per effettuare le simulazioni sono state implementate quattro possibili combinazioni mediante l'utilizzo della probabilità. In particolare:

- **Caso Ottimo:** fascia di probabilità compresa fra 0.8 ed 1.0.
- **Caso Pessimo:** fascia di probabilità compresa fra 0.0 e 0.2.
- **Caso Intermedio - RESTART:** fascia di probabilità compresa fra 0.3 e 0.5.
- **Caso Intermedio - RELOAD:** fascia di probabilità compresa fra 0.6 e 0.7.

L'obiettivo primario di questo progetto è stato quello di creare una riproduzione di una catena industriale IIoT 4.0 che fosse *loop-less* ed altamente

affidabile. Di seguito verrà descritto ogni possibile caso in maniera più approfondita.

Nella seconda parte del capitolo invece ci concentreremo sui risultati delle simulazioni, andando a misurare i ritardi all'interno dei vari casi e facendo una stima su come questo possa influenzare le quantità prodotte in una catena industriale.

11.1.1 Scenario Caso Ottimo

Il primo caso che andremo ad analizzare è il caso ottimo, ovvero quando la catena industriale IoT riesce ad eseguire tutte le lavorazioni senza rilevare anomalie o guasti.

Il verificarsi del caso ottimo avviene con una probabilità che oscilla fra il 70% ed il 100%, a seconda del caso che si vuole studiare ed analizzare; la probabilità è piuttosto bassa, ma l'obiettivo è stato quello di analizzare in maniera critica la catena di montaggio, per permettere così di creare una catena *high-quality* e con bassa tolleranza di errori o anomalie; basta pensare, ad esempio, ad una catena di montaggio dove vengono prodotti microprocessori e dove ogni minima imperfezione, anche la più microscopica, può produrre un pessimo risultato ed un prodotto di qualità scadente.

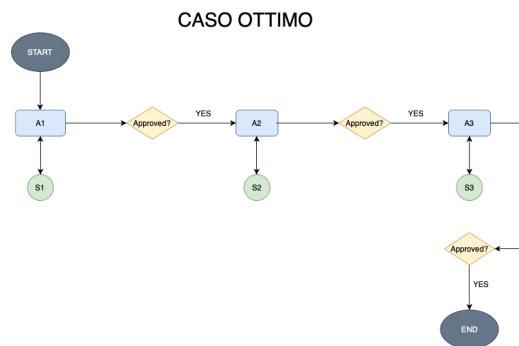


Figura 11.1: Flow Chart: Caso Ottimo

Nella figura [11.1] è rappresentato il flow-chart del caso ottimo, dove si può notare che il prodotto entra in fase di lavorazione ed arriva alla fine senza interruzioni e senza alcun rilevamento di anomalie.

Nella figura sottostante [11.2] è invece rappresentato il sequence diagram di tutte le operazioni che vengono effettuate dai vari attori della rete.

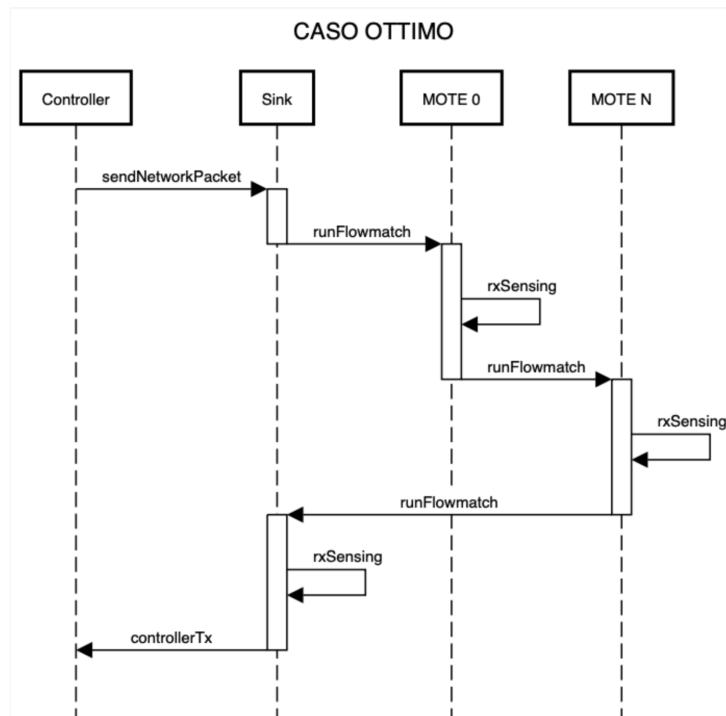


Figura 11.2: Sequence Diagram: Caso Ottimo

Vediamo adesso invece che cosa avviene sulla macchina virtuale, sia all'interno del simulatore Cooja che dentro la console di IntelliJ IDEA.

Di seguito sono riportate le immagini di quello che possiamo leggere all'interno del pannello Mote Output e nella Console di IntelliJ IDEA. Si noti bene che le simulazioni prese in esempio per le immagini sono con una rete al cui interno sono presenti un Sink, due sensori e due attuatori.

Time	Mote	Message
01:30.425	ID:1	REINVIO PACCHETTO DI SENSING
01:30.425	ID:1	SINK: RICEVUTO PACCHETTO DI SENSING
01:38.585	ID:1	pacchetto:[0, 16, 0, 2, 0, 1, 9, 100, 0, 1, 0, 2, 0, 2, 0, 3]
01:38.587	ID:2	MOTE: ARRIVATO PACCHETTO DI SENSING
01:38.587	ID:2	pacchetto:[0, 16, 0, 2, 0, 1, 9, 99, 0, 2, 0, 2, 0, 2, 0, 3]
01:38.587	ID:2	pacchetto:[0, 16, 0, 2, 0, 1, 9, 99, 0, 2, 0, 2, 0, 2, 0, 3]
01:42.686	ID:2	MOTE: CASO OTTIMO INVIO PACCHETTO CON SIZE=1 AL PROSSIMO MOTE
01:43.686	ID:2	Beacon received
01:43.686	ID:3	MOTE: ARRIVATO PACCHETTO DI SENSING
01:43.686	ID:3	pacchetto:[0, 14, 0, 3, 0, 2, 9, 99, 0, 3, 0, 1, 0, 3]
01:43.686	ID:3	SIZE = 14
01:43.687	ID:3	CASO OTTIMO INVIO PACCHETTO CON SIZE=1 AL SINK
01:48.789	ID:1	SINK: RICEVUTO PACCHETTO DI SENSING
01:48.789	ID:1	pacchetto:[0, 12, 0, 1, 0, 3, 9, 99, 0, 1, 0, 0]
01:48.789	ID:1	SINK: REINVIO PACCHETTO CON SIZE=0 AL CONTROLLER

Figura 11.3: Cooja - Mote Output: Caso Ottimo

```
CONTROLLER: RICEVUTO PACCHETTO DI SENSING
CONTROLLER: TEMPO DI ESECUZIONE: 10205ms
```

Figura 11.4: IntelliJ IDEA - Console: Caso Ottimo

11.1.2 Scenario Caso Pessimo

Il secondo caso che andremo ad analizzare è il caso pessimo, ovvero quando la catena industriale IoT rileva un'anomalia ed interrompe quindi la lavorazione, scartando il pezzo in lavorazione. È sicuramente il caso peggiore che si può presentare in fase di lavorazione, poiché oltre ad una perdita di tempo che va ad impattare la produttività della catena, abbiamo anche una perdita materiale di risorse poichè il pezzo viene scartato. Il caso pessimo si verifica con una probabilità del 10%.

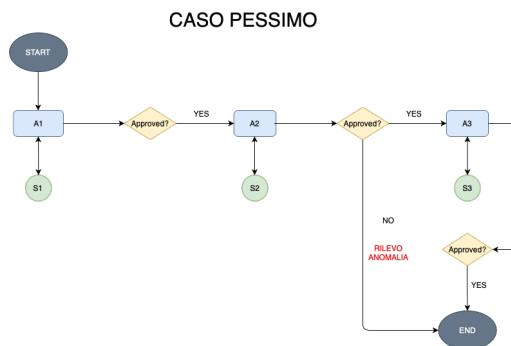


Figura 11.5: Flow Chart: Caso Pessimo

Nella figura [11.5] è rappresentato il flow-chart del caso pessimo, dove si può notare che il prodotto entra in fase di lavorazione, rileva un'anomalia e termina la lavorazione.

Nella figura sottostante [11.6] è invece rappresentato il sequence diagram di tutte le operazioni che vengono effettuate dai vari attori della rete.

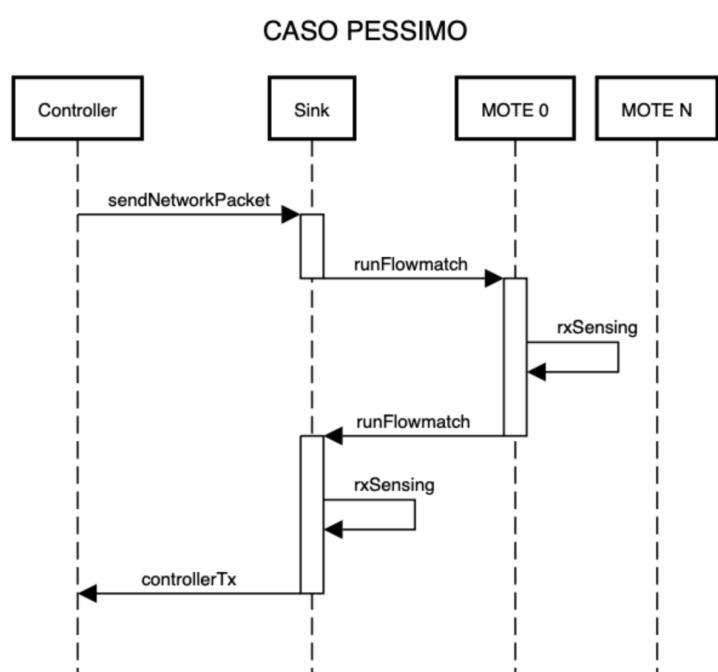


Figura 11.6: Sequence Diagram: Caso Pessimo

Vediamo adesso invece che cosa avviene sulla macchina virtuale, sia all'interno del simulatore Cooja che dentro la console di IntelliJ IDEA.

Di seguito sono riportate le immagini di quello che possiamo leggere all'interno del pannello Mote Output e nella Console di IntelliJ IDEA. Si noti bene che le simulazioni prese in esempio per le immagini sono con una rete al cui interno sono presenti un Sink, due sensori e due attuatori.

The screenshot shows a window titled "Mote output" with a table of log entries. The columns are "Time", "Mote", and "Message". The log entries describe a sensing packet being received and forwarded, and a controller receiving it.

Time	Mote	Message
03:38.129	ID-1	SINK: RICEVUTO PACCHETTO DI SENSING pacchetto:[0, 16, 0, 2, 0, 1, 9, 100, 0, 1, 0, 2, 0, 2, 0, 3]
03:38.129	ID-1	Open Path
03:38.130	ID-2	insert Rule
03:38.130	ID-1	Replacing rule IF (P.DST == 1) { FORWARD_U 1; } (TTL: 254, U: 0) at position 1
03:38.131	ID-1	Open Path
03:38.131	ID-1	insert Rule
03:38.131	ID-2	Replacing rule IF (P.DST == 2) { FORWARD_U 2; } (TTL: 254, U: 0) at position 1
03:38.132	ID-1	pacchetto[0, 16, 0, 2, 0, 1, 9, 99, 0, 2, 0, 2, 0, 2, 0, 3]
03:38.134	ID-2	SIZE: N_ 2
03:43.276	ID-3	MOTE: ARRIVATO PACCHETTO DI SENSING pacchetto:[0, 14, 0, 3, 0, 1, 9, 99, 0, 3, 0, 1, 0, 3]
03:43.276	ID-3	SIZE: N_ 1
03:43.277	ID-1	Open Path
03:43.277	ID-1	insert Rule
03:43.277	ID-3	Replacing rule IF (P.DST == 3) { FORWARD_U 3; } (TTL: 254, U: 0) at position 2
03:48.375	ID-3	MOTE: CASO PESSIMO SCARTO PACCHETTO (SIZE=-1)
03:48.375	ID-3	Open Path
03:48.375	ID-3	insert Rule
03:48.375	ID-3	Replacing rule IF (P.DST == 1) { FORWARD_U 1; } (TTL: 254, U: 0) at position 2
03:48.377	ID-1	pacchetto:[0, 12, 0, 1, 0, 3, 9, 99, 0, 1, 0, 255]
03:48.377	ID-1	SINK: INVIO PACCHETTO SCARTATO AL CONTROLLER

Figura 11.7: Cooja - Mote Output: Caso Pessimo

```
CONTROLLER: RICEVUTO PACCHETTO DI SENSING
CONTROLLER: RITARDO: 10291ms
```

Figura 11.8: IntelliJ IDEA - Console: Caso Pessimo

11.1.3 Scenario Caso Intermedio - Restart

Il terzo caso che andremo ad analizzare è uno dei due casi casi intermedi; in particolare quest'ultimo è stato denominato Restart, poiché quando viene rilevata un'anomalia il pezzo viene rinvia al l'inizio della catena di lavorazione e gli viene fatta ripetere tutta la lavorazione fin dal passo iniziale. Questo avviene per permettere una maggiore qualità ed affidabilità sulla produzione della catena di montaggio.

Il caso intermedio Restart si verifica con una probabilità del 20%.

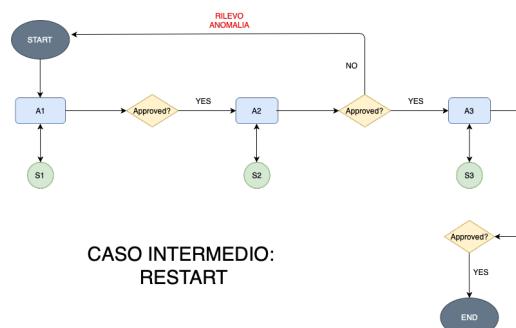


Figura 11.9: Flow Chart: Caso Intermedio - RESTART

Nella figura [11.9] è rappresentato il flow-chart del caso intermedio Restart, dove si può notare che il prodotto entra in fase di lavorazione, viene rilevata un'anomalia ed il pezzo viene rinviato all'inizio della catena di montaggio per iniziare nuovamente la lavorazione.

Nella figura sottostante [11.10] è invece rappresentato il sequence diagram di tutte le operazioni che vengono effettuate dai vari attori della rete.

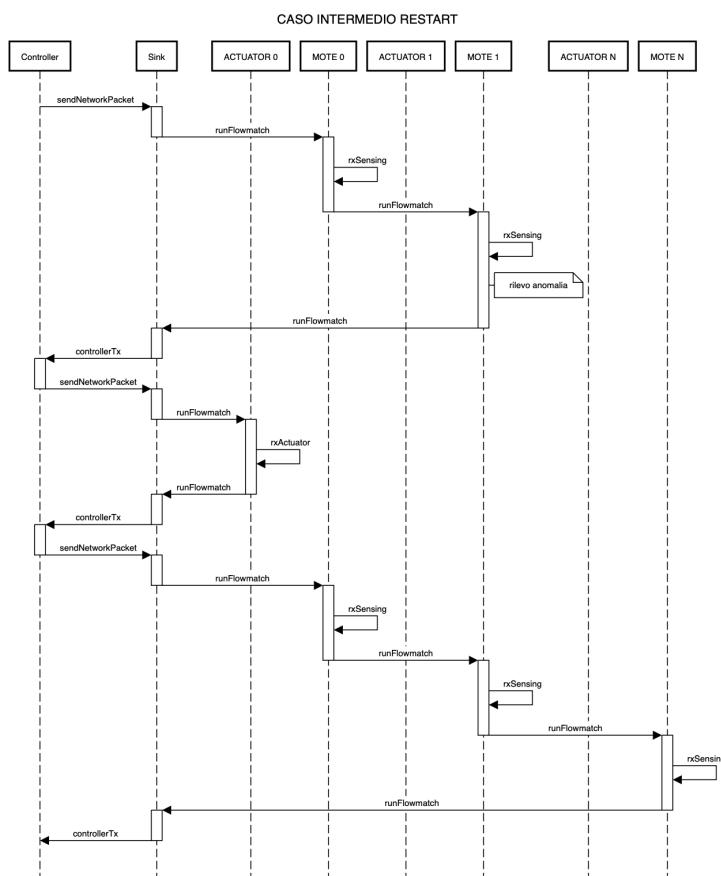


Figura 11.10: Sequence Diagram: Caso Intermedio RESTART

Vediamo adesso invece che cosa avviene sulla macchina virtuale, sia all'interno del simulatore Cooja che dentro la console di IntelliJ IDEA.

Di seguito sono riportate le immagini di quello che possiamo leggere all'interno del pannello Mote Output e nella Console di IntelliJ IDEA. Si noti bene

che le simulazioni prese in esempio per le immagini sono con una rete al cui interno sono presenti un Sink, due sensori e due attuatori.

```

Mote output
File Edit View
Time | Mote | Message
00:33.151 ID:1 SINK: RICEVUTO PACCHETTO DI SENSING
00:33.151 ID:1 pacchetto[0, 16, 0, 2, 0, 1, 9, 100, 0, 1, 0, 2, 0, 2, 0, 3]
00:33.172 ID:1 Open Path
00:33.172 ID:1 Insert Rule
00:33.176 ID:1 Inserting rule IF (P.DST == 2) { FORWARD_U 2; } (TTL: 254, U: 0) at position 1
00:33.177 ID:2 Open Path
00:33.177 ID:2 insert Rule
00:33.226 ID:2 MOTE: ARRIVATO PACCHETTO DI SENSING
00:33.226 ID:2 pacchetto[0, 16, 0, 2, 0, 1, 9, 99, 0, 2, 0, 2, 0, 3]
00:38.226 ID:2 SINK: RICEVUTO PACCHETTO DI SENSING
00:38.226 ID:2 NOTE: CASO INTERMEDIO RESTART (0 4). RITORNO ALL'INIZIO (SIZE==2)
00:38.320 ID:1 SINK: RICEVUTO PACCHETTO DI SENSING
00:38.320 ID:1 pacchetto[0, 14, 0, 1, 0, 2, 9, 99, 0, 1, 254, 0, 3]
00:38.320 ID:1 SINK: REINVIO PACCHETTO CON SIZE==2 AL CONTROLLER
00:38.320 ID:1 SINK: RICEVUTO PACCHETTO ACTUATOR CON DEST 0 4
00:38.363 ID:4 insert Rule
00:38.363 ID:4 Replacing rule IF (P.DST == 1) { FORWARD_U 1; } (TTL: 254, U: 0) at position 1
00:38.365 ID:1 Open Path
00:38.365 ID:1 insert Rule
00:38.365 ID:1 Inserting rule IF (P.DST == 4) { FORWARD_U 4; } (TTL: 254, U: 0) at position 2
00:38.366 ID:4 ATTUATORE: ARRIVATO PACCHETTO ACTUATOR SIZE 0
00:38.366 ID:4 ATTUATORE: REINIZIO LAVORAZIONE
00:38.369 ID:4 ATTUATORE: FINITA LAVORAZIONE, REINVIO AL SINK
00:38.369 ID:4 SINK: RICEVUTO PACCHETTO ACTUATOR CON DEST 0 1
00:38.375 ID:1 SINK: RICEVUTO PACCHETTO DI SENSING
00:38.375 ID:1 pacchetto[1, 16, 0, 2, 0, 1, 9, 100, 0, 1, 0, 2, 0, 2, 0, 3]
00:38.377 ID:2 MOTE: ARRIVATO PACCHETTO DI SENSING
00:38.377 ID:2 pacchetto[1, 16, 0, 2, 0, 1, 9, 99, 0, 2, 0, 2, 0, 3]
00:43.478 ID:2 MOTE: CASO OTTIMO INVIO PACCHETTO CON SIZE==1 AL PROSSIMO MOTE
00:48.627 ID:3 NOTE: CASO OTTIMO INVIO PACCHETTO CON SIZE==1 AL PROSSIMO MOTE
00:48.627 ID:3 Open Path
00:48.627 ID:3 insert Rule
00:48.627 ID:3 Replacing rule IF (P.DST == 1) { FORWARD_U 1; } (TTL: 254, U: 0) at position 2
00:48.627 ID:3 SINK: RICEVUTO PACCHETTO DI SENSING
00:48.630 ID:1 pacchetto[1, 12, 0, 1, 0, 2, 9, 99, 0, 1, 0, 0]
00:48.630 ID:1 SINK: REINVIO PACCHETTO CON SIZE==0 AL CONTROLLER

```

Figura 11.11: Cooja - Mote Output: Caso Intermedio - Restart

```

CONTROLLER: RICEVUTO PACCHETTO DI SENSING
CONTROLLER: TEMPO DI ESECUZIONE: 15488ms

```

Figura 11.12: IntelliJ IDEA - Console: Caso Intermedio - Restart

11.1.4 Scenario Caso Intermedio - Reload

Il quarto ed ultimo caso che andremo ad analizzare è l'altro caso intermedio; in particolare è stato chiamato Reload, poiché quando viene rilevata un'anomalia, viene fatta ripetere al pezzo quello step di lavorazione dove si è rilevato un qualcosa di anomalo. Questo avviene per permettere una maggiore qualità ed affidabilità sulla produzione della catena di montaggio e consente così di risparmiare risorse e tempo utile per la produzione.

Il caso intermedio Reload si verifica con una probabilità del 30%.

Nella figura [11.3] è rappresentato il flow-chart del caso intermedio Restart,

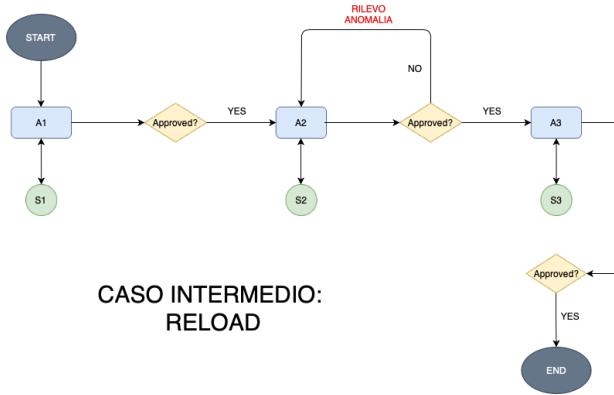


Figura 11.13: Flow Chart: Caso Intermedio - RELOAD

dove si può notare che il prodotto entra in fase di lavorazione, viene rilevata un'anomalia ed il pezzo viene rinviato all'inizio della catena di montaggio per iniziare nuovamente la lavorazione. Il caso intermedio Reload si verifica con una probabilità del 30%. Nella figura sottostante [11.14] è invece rap-

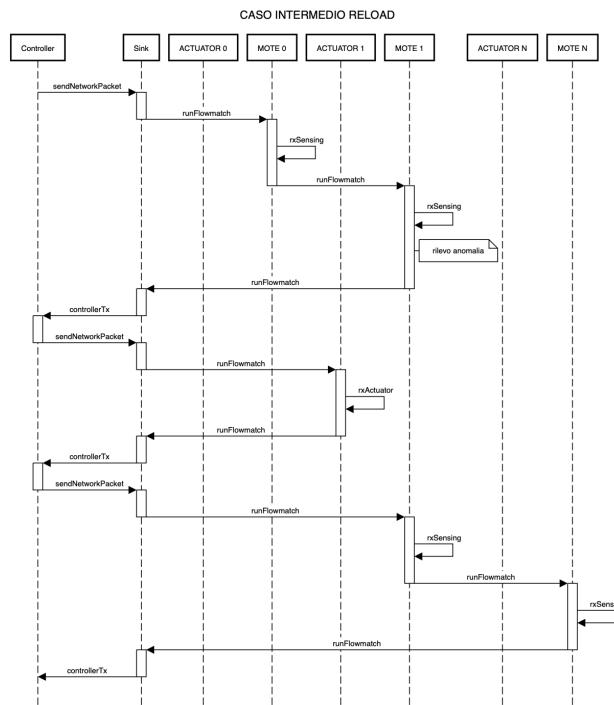


Figura 11.14: Sequence Diagram: Caso Intermedio RELOAD

presentato il sequence diagram di tutte le operazioni che vengono effettuate dai vari attori della rete.

Vediamo adesso invece che cosa avviene sulla macchina virtuale, sia all'interno del simulatore Cooja che dentro la console di IntelliJ IDEA.

Di seguito sono riportate le immagini di quello che possiamo leggere all'interno del pannello Mote Output e nella Console di IntelliJ IDEA. Si noti bene che le simulazioni prese in esempio per le immagini sono con una rete al cui interno sono presenti un Sink, due sensori e due attuatori.

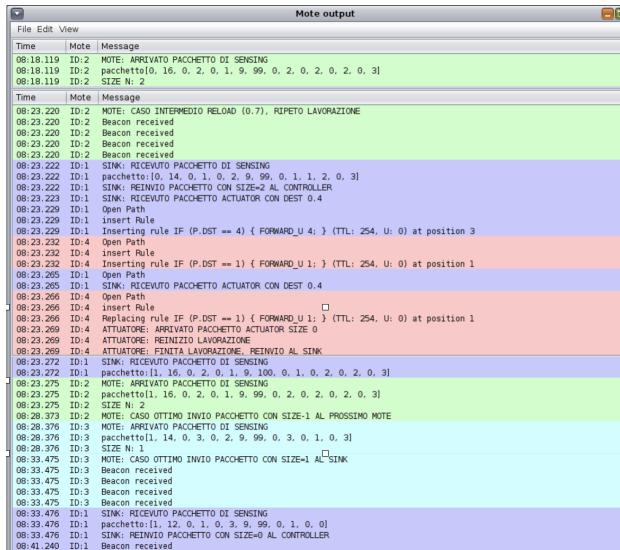


Figura 11.15: Cooja - Mote Output: Caso Intermedio - Reload

```
CONTROLLER: RICEVUTO PACCHETTO DI SENSING
CONTROLLER: TEMPO DI ESECUZIONE: 15362ms
```

Figura 11.16: IntelliJ IDEA - Console: Caso Intermedio - Reload

11.2 Risultati delle simulazioni

In questa seconda parte del capitolo vedremo i risultati delle simulazioni descritte in precedenza, soffermandoci su un'analisi accurata dei ritardi che i vari casi possono produrre sulla catena industriale IIoT 4.0 e l'impatto che possono avere sulla produzione.

11.2.1 Misurazione del Ritardo

In questo progetto di catena di montaggio IIoT applicata al paradigma dell'Industria 4.0 è interessante studiare i ritardi che i vari casi implementanti possono generare.

Prima di fare ciò occorre definire in maniera matematica come è composta la funzione ritardo complessiva, dato che influiscono su di essa molti fattori. In particolare la funzione ritardo è così definita:

$$\text{RITARDO COMPLESSIVO} = \text{Ritardo Lavorazione} + \text{Ritardo Lettura Sensore} + \text{Ritardo Trasmissione}$$

Vediamo adesso nel dettaglio cosa significano le varie voci sui ritardi:

- **Ritardo Lavorazione:** questo ritardo è dovuto al tempo che impiega ogni attuatore a svolgere la lavorazione a lui assegnata. Il tempo stimato è di circa 5.0s per ogni step di lavorazione.
- **Ritardo Lettura Sensore:** questo ritardo è dovuto al tempo che ogni sensore impiega a leggere i dati sulla lavorazione appena completata. Il ritardo medio è di circa 100ms.
- **Ritardo Trasmissione:** questo ritardo è dovuto al tempo che impiegano i vari attori della rete a comunicare fra di loro. Dai dati raccolti, possiamo

affermare che in media su una rete composta da un sink, quattro sensori e quattro attuatori il ritardo di trasmissione si attesta fra i 300ms ed i 350ms. Ci aspettiamo quindi che questo aumenti con l'aumentare delle dimensioni della rete.

Vediamo adesso nello specifico caso per caso.

Caso Ottimo

Il primo caso che valutiamo è proprio il caso ottimo. In questo tipo di simulazione non sono presenti ritardi dato che il pezzo viene lavorato dall'inizio alla fine senza incorrere in anomalie o ricevere interruzioni. Vediamo allora il grafico che raffigura sulle ascisse una catena inizialmente composta da un sink e da un minimo di un attuatore e un sensore fino ad massimo di dieci attuatori e dieci sensori, mentre sulle ordinate il tempo medio dopo dieci prove.

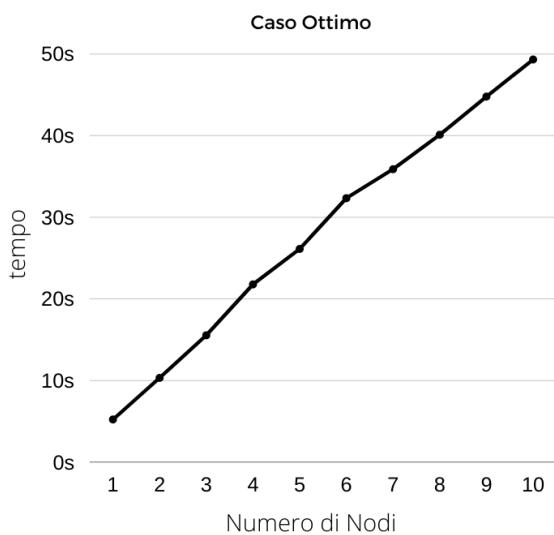


Figura 11.17: Tempo di esecuzione medio dopo dieci prove

Come ci potevamo aspettare dalla teoria, questa funzione cresce linearmente al variare del tempo e con l'aumentare delle dimensioni della rete.

Caso Pessimo

Il secondo caso che valutiamo è il caso pessimo. Questo rappresenta sicuramente il caso peggiore che si può verificare durante l'esecuzione della catena di montaggio. Infatti ricordiamo che in caso si verifichi il caso pessimo il pezzo viene integralmente scartato; da ciò possiamo quindi dedurre che il tempo di esecuzione sia in realtà tutto ritardo.

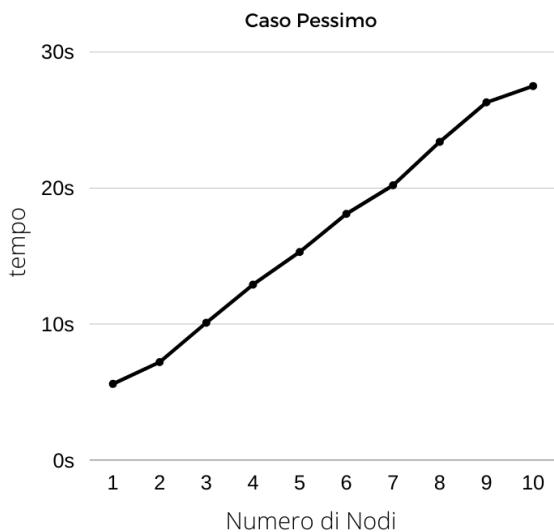


Figura 11.18: Ritardo Complessivo medio dopo dieci prove

Anche in questo caso la funzione cresce linearmente, essendo il ritardo nel caso pessimo direttamente proporzionale all'aumentare della rete e al tempo.

Caso Intermedio - Restart

Il primo caso intermedio è il caso che abbiamo chiamato *Restart*, ovvero quando durante una lavorazione si rileva una anomalia e si ritorna all'inizio della catena di montaggio, ripetendo la lavorazione.

Di seguito è riportato il grafico dove vengono messi a confronto i tempi di esecuzione del caso ottimo e del caso restart. In particolare il grafico in rosso

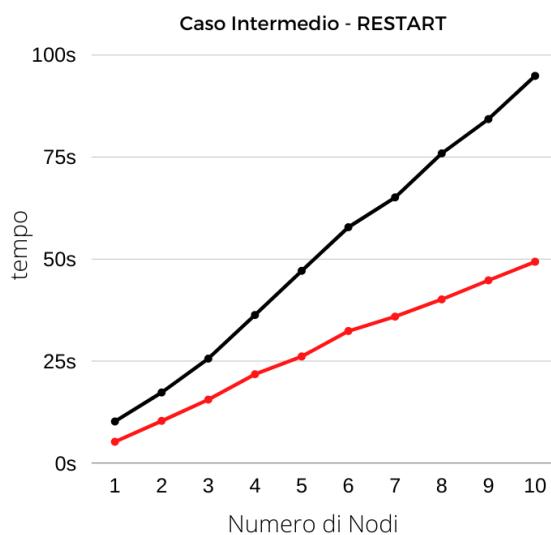


Figura 11.19: Tempi di Esecuzione a confronto: Ottimo vs Restart

rappresenta il tempo di esecuzione del caso ottimo, mentre quello in nero il tempo di esecuzione del caso intermedio Restart. È sicuramente il caso più dispendioso e come potevamo aspettarci, all'aumentare delle dimensioni della rete, aumenta il tempo di esecuzione. In particolare la discrepanza che si verifica fra i due casi è quantificabile come ritardo nell'esecuzione della catena di montaggio.

Volendo fare una stima e quantificare questo ritardo in secondi, possiamo dire che in media, considerando una rete che varia da uno fino a dieci nodi, e

considerando le dieci iterazioni svolte come prova, il ritardo medio è di circa 23,3 secondi.

Caso Intermedio - Reload

Nell'ultimo caso analizzato, ovvero il secondo caso intermedio denominato *Reload*, si ha il rilevamento di un'anomalia che comporta la ripetizione dello step di lavorazione dove quest'ultima si è verificata. Infatti il pezzo ripete la lavorazione nel punto in cui era stato fallace, per poi arrivare al termine di tutta la catena di montaggio.

Di seguito è riportato il grafico dove vengono messi a confronto i tempi di esecuzione del caso ottimo e del caso reload. In particolare il grafico in rosso

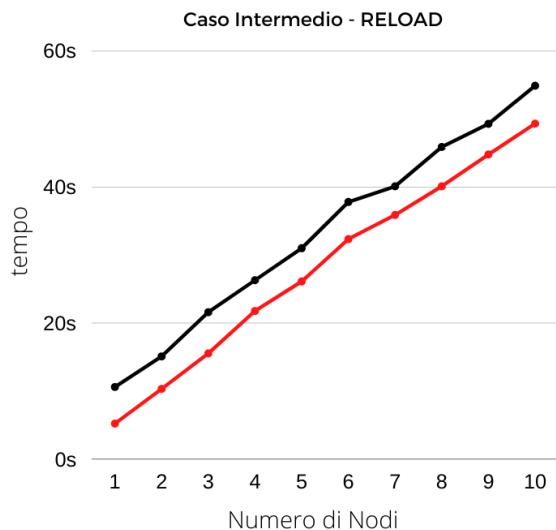


Figura 11.20: Tempi di Esecuzione a confronto: Ottimo vs Reload

rappresenta il tempo di esecuzione del caso ottimo, mentre quello in nero il tempo di esecuzione del caso intermedio Reload. È sicuramente un caso in cui ci aspettiamo del ritardo, ma come si riesce dal leggere dal grafico

risulta molto contenuto. Possiamo affermare e quantificare questo ritardo medio, sempre considerando una rete che varia da uno fino a dieci nodi, e considerando le dieci iterazioni svolte come prova, come un ritardo di circa 5.4 secondi.

11.2.2 Quantità prodotte

In questa ultima parte di capitolo andremo ad analizzare come i vari casi studiati in precedenza vanno ad influenzare le quantità prodotte da una catena industriale IIoT per l'Industria 4.0, analizzando in forma teorica il rate e la frequenza relativa. Vediamoli adesso più nel dettaglio:

$$\text{- Rate} = \frac{\text{Numero di Pezzi}}{\text{Intervallo temporale}}$$

Il rate della catena industriale viene calcolato come il numero di pezzi prodotti in un determinato intervallo temporale. Dagli studi fatti possiamo aspettarci che il rate sia massimo nel caso ottimo e che tenda a diminuire negli altri casi, toccando il minimo nel Caso Intermedio Restart, dove ho il ritardo maggiore. Il rate ha come unità di misura pezzi/secondo.

$$\text{- Frequenza Relativa} = \frac{\text{Numero di volte che si verifica evento}}{\text{Numero totale eventi}}.$$

La formula matematica della frequenza relativa può essere utilizzata nella nostra catena industriale, ad esempio, per verificare la frequenza con cui vengono scartati i pezzi rispetto alla frequenza di successo. In questo caso mi aspetto di avere il massimo nel caso ottimo, mentre il minimo sarà chiaramente nel caso pessimo, dove vengono scartati la maggior parte dei pezzi.

Capitolo 12

Conclusioni

Dalla visione generale di WSN e SDN, alla loro unione nelle SD-WSN, e in particolare mediante l'architettura SDN-WISE, la comunità scientifica si aspetta un grande giovamento per il mondo IoT e le aziende stanno investendo molto per uniformarsi agli standard previsti per l'Industria 4.0. Per quanto sperimentato e ottenuto in questa tesi si può concludere che la strada è promettente, con l'unico vincolo che il framework SDN-WISE consente di simulare una WSN, ma non verificare nella realtà i risultati ottenuti. Dunque è doveroso, alla luce dei promettenti risultati raccolti, passare a dei veri sensori.

Da questo progetto di tesi possiamo proporci la domanda, ma SDN ha davvero senso nella sua applicazione industriale? La risposta è ovviamente sì, dato che i vantaggi sono molteplici, ovvero:

- *Affidabilità*
- *Produttività*
- *Sicurezza*
- *Riduzione dei costi*

Ovviamente ci sono anche degli aspetti peggiorativi, come ad esempio la non

predicibilità dei vari risultati ottenuti.

Una possibile soluzione futura è l'introduzione della tecnologia 5G, che in realtà tanto futura non è visto che si parla già da molto tempo di questa nuova e rivoluzionaria tecnologia.

A livello di tempi di risposta si guadagnerebbe in latenza, dato che la latenza per la tecnologia 5G sappiamo essere di 1ms, dato che ha un data-rate molto più alto rispetto alle tecnologie attuali. Da ciò possiamo dedurre che quindi il ritardo lettura sensore ed il ritardo di trasmissione riceverebbero una notevole riduzione, aumentando ancora di più la produttività e diminuendo così i ritardi, anche nei casi più sfavorevoli.

Un altro aspetto da considerare della tecnologia 5G è che il Controller SDN ha a disposizione uno scheduler che schedula i vari flussi, alleggerendo di non poco il lavoro al controller.

Un possibile sviluppo futuro di questa tesi potrebbe essere, ad esempio, l'introduzione della tecnologia 5G e l'implementazione di più controller, per andare poi ad analizzare i dati ottenuti e compararli con la soluzione attuale.

Ringraziamenti

Si ringraziano il Professor Francesco Chiti ed il Dottor Michele Bonanni per l'aiuto e la collaborazione forniti durante questi mesi per lo sviluppo di questo progetto di tesi. Nonostante le limitazioni che hanno contraddistinto questo periodo, non hanno fatto mai mancare il loro supporto e la loro volontà di portare a compimento questo progetto innovativo che ha presentato molte sfide durante tutto lo svolgimento.

Bibliografia

- [1] F. Griffiths, M. Ooi, “The fourth industrial revolution - Industry 4.0 and IoT.” IEEE Instrumentation and Measurement Magazine, 2018.
- [2] M. Wollschlaeger; T. Sauter; J. Jasperneite, “The Future of Industrial Communication: Automation Networks in the Era of the Internet of Things and Industry 4.0.” IEEE Industrial Electronics Magazine, 2017.
- [3] W. Stallings, “Foundations of Modern Networking — SDN, NFV, QoE, IoT, and Cloud.” PEARSON, 2016.
- [4] K. Kaur, Garg, Aujla, N. Kumar, J. J. P. C. Rodrigues, M. Guizani, “Edge Computing in the Industrial Internet of Things Environment: Software-Defined-Networks-Based Edge-Cloud Interplay.” IEEE Communications Magazine, 2018.
- [5] M. Antonini; M. Vecchio; F. Antonelli, “Fog Computing Architectures: A Reference for Practitioners.” IEEE Internet of Things Magazine, 2020.
- [6] K. Moskvitch, “When machinery chats [Connections Industrial IOT].” Engineering and Technology, 2018.
- [7] K. Xu; Y. Qu; K. Yang, “A tutorial on the internet of things: from a heterogeneous network integration perspective.” IEEE Network, 2016.

- [8] M. Weyrich; C. Ebert, “Reference Architectures for the Internet of Things.” IEEE Software, 2015.
- [9] J. Ploennigs, J. Cohn, A. Stanford-Clark, “The Future of IoT.” IEEE Internet of Things Magazine, 2018.
- [10] D. Georgakopoulos; P. P. Jayaraman; M. Fazia; M. Villari; R. Ranjan, “Internet of Things and Edge Cloud Computing Roadmap for Manufacturing.” IEEE Cloud Computing, 2016.
- [11] “<https://sdnwiselab.github.io/docs/guides/Core.html>.”