ALMA MATER STUDIORUM – UNIVERSITÀ DI BOLOGNA

FACOLTÀ DI INGEGNERIA CON SEDE A CESENA

CORSO DI LAUREA MAGISTRALE IN INGEGNERIA E SCIENZE INFORMATICHE

AUTOMAZIONE DI RETE NELL'INDUSTRIA 4.0

Tesi in

Laboratory of Network Programmability and Automation

Relatore
Prof. FRANCO CALLEGATI

Presentata da
RICCARDO BACCA

SESSIONE III

ANNO ACCADEMICO 2022/2023

Indice

Capitolo 1

Introduzione

L'industria 4.0 (I4.0) sta promuovendo la digitalizzazione dei sistemi di produzione tradizionali puntando all'ottenimento di fabbriche flessibili e intelligenti basate su sistemi fisici e informativi tra loro integrati. In questo contesto, il Reference Architecture Model Industrie 4.0 (RAMI4.0) fornisce numerose linee guida per lo sviluppo di soluzioni conformi all'Industria 4.0, nonostante lo sviluppo di questi sistemi richieda l'adozione di nuove tecnologie e architetture che favoriscano la riduzione della complessità e il raggiungimento dell'interoperabilità cross-company.

Un primo concetto da affrontare è quello del Digital Twin: una rappresentazione digitale di un elemento industriale, un singolo macchinario, uno Switch di rete, un Controller o un prodotto finito. L'utilizzo dei Digital Twin come principale elemento di interazione tra sistemi informatici e sistemi fisici, consente lo sviluppo di soluzioni innovative, flessibili e intelligenti per lo sviluppo del settore industriale.

Come principale specifica RAMI4.0, l'Asset Administration Shell (AAS), partendo dalla rappresentazione digitale di un elemento industriale (logico o fisico), svolge un ruolo fondamentale nel trasformare tale risorsa in un componente 4.0, inserendolo in una rete aziendale che favorisca l'interoperabilità e lo scambio di informazioni tra questi componenti. Si potrà perciò accedere alle informazioni sulle risorse in tempo reale, stabilire un'interfaccia di comunicazione standardizzata e sicura e avere la possibilità di gestire l'intero ciclo di vita del singolo componente industriale.

L'obiettivo voluto con l'introduzione dei concetti di Asset Administration Shell e Digital Twin è perciò quello di ottenere interoperabilità e comunicazione

cross-company anche partendo da Digital Twin proprietari, a oggi presenti e non favorevoli nel raggiungimento dello standard imposto e voluto.

L'elaborato in questione, si pone l'obiettivo di determinare una possibile modellazione e implementazione (semplificata) dell'architettura e delle tecnologie di rete necessarie a favorire e gestire ottimamente la comunicazione tra Asset Administration Shell e componenti aderenti all'Industria 4.0, oltre a fornire un approfondimento sui concetti precedentemente introdotti.

In particolare si vogliono poter gestire differenti Switch di rete ciascuno con un proprio Controller, che ne permetta la configurazione e la scelta del comportamento finale. Ad esempio permettendo o meno un determinato traffico tra Switch o Host con la configurazione semplificata di un Firewall o fornendo un Simple Switch, fornendo libertà alle comunicazioni di rete.

L'utilizzo degli Asset Administration Shell come Digital Twin di Switch e Controller di rete, fornisce ampi sbocchi nelle realtà industriali, anche favorendo utenti meno esperti nella gestione delle reti industriali. Diviene possibile la creazione e attivazione di determinate configurazioni di rete in modo semi automatizzato e semplificato, favorendo la produzione, la raccolta di informazioni o il monitoraggio dei macchinari industriali a seconda delle necessità aziendali ed esterne.

Capitolo 2

Digital Twin e AAS

Un Digital Twin rappresenta un modello virtuale di un oggetto fisico o di un oggetto virtuale, necessario al funzionamento di un determinato apparato.

2.1 Introduzione

Il concetto di Digital Twin viene introdotto dalla quarta rivoluzione industriale, come collegamento logico tra il mondo reale e quello digitale, fornendo alle aziende una nuova modalità di controllo del processo produttivo, dalla progettazione e modellazione fino al controllo post vendita.

Sono disponibili e pubblicate numerose definizioni sul concetto di Digital Twin, dalle quali è possibile estrapolare alcuni concetti fondamentali:

- Un Digital Twin rappresenta un modello virtuale di un oggetto fisico;
- Consente l'emulazione del ciclo di vita di un determinato prodotto, favorendone la durabilità e il mantenimento nel tempo;
- Favorisce l'accessibilità delle azioni e dei differenti funzionamenti relativi all'oggetto fisico e le relative possibili configurazioni.

Conseguentemente, un Digital Twin viene composto da:

- L'oggetto fisico nel mondo reale;
- La copia digitale di quest'ultimo;
- L'insieme di collegamenti e connessioni necessari all'accoppiamento del modello fisico e del modello logico.

2.1.1 Utilizzo dei Digital Twin nell'industria 4.0

Il concetto di Digital Twin può essere considerato uno degli strumenti più utili e perspicaci per promuovere non solo il valore aziendale, ma anche l'innovazione industriale. Mediante questi strumenti, si ha l'opportunità di concentrarsi maggiormente sul Business Value aziendale e sul successo competitivo nel settore, utilizzando le nuove tecnologie come metodologia di valutazione del valore aziendale: maggiore qualità dei prodotti commercializzati in termini di durata e supporto manutentivo ed evolutivo, periodi di progettazione più brevi, possibilità di nuove fonti di reddito e un migliore controllo dei costi e delle garanzie.

Dal Digital Twin di un determinato elemento industriale, estrapoliamo dati e misurazioni, combinandoli con il ciclo di vita del rispettivo elemento fisico. Traendo conclusioni sullo stato attuale della produzione, delle performance aziendali e del processo di produzione.

In questo modo, con l'uso dei DT, è possibile testare e ottimizzare digitalmente ogni fase della linea di produzione, portando eventuali cambiamenti nella realtà in un secondo momento, ulteriormente ottimizzando i processi aziendali.

2.2 Asset Administration Shells

L'Asset Administration Shell rappresenta un concetto cardine dell'Industria 4.0. Un AAS rappresenta un componente 4.0 all'interno di una rete, di un sistema produttivo, ponendo le proprie basi sul concetto di Digital Twin ma espandendolo in termini di funzionalità, interoperabilità e supporto nel mondo dell'Industria e non solo. In particolare ogni componente nei nuovi e rivoluzionati ambienti industriali dovrà essere fornito di un AAS.

L'AAS svolge infatti alcuni compiti fondamentali per la rivoluzione 4.0:

- Permette la modellazione dell'intero processo produttivo: dalla progettazione al monitoraggio del prodotto finito;

- Favorisce e punta a un livello di integrazione degli apparati industriali ancora maggiore: l'obiettivo è ottenere fabbriche auto organizzanti e ottimizzanti;

Partendo dalla modellazione del prodotto risulta possibile verificare a priori di un investimento, l'utilità e la produttività che possa scaturire da quest'ultimo, mentre il monitoraggio del prodotto finito fornisce maggiori garanzie per ambe le parti. Il produttore raccogliendo dati ha la possibilità di verificare la storicità del prodotto e intervenire tempestivamente in caso di guasti. Il cliente ha dalla sua parte garanzie tangibili relative alla manutenzione evolutiva e correttiva del software e del prodotto stesso, da parte del produttore.

In riferimento a questo, distinguiamo due tipologie di Asset Administration Shell: Template e Instance. La prima rappresenta una tipologia di Prodotto o Macchinario, disponibile quindi in molteplici pezzi. Mentre un Instance, rappresenta la singola istanza che va a usufruire di un Template. Questa distinzione nasce a vantaggio dell'espandibilità nell'utilizzo degli AAS, non dovendo creare un sistema da zero per ogni elemento creato o acquistato.

Un AAS potrebbe essere definito Composite AAS. Quest'ultimo consiste in molteplici AAS strutturalmente collegati tra loro formando una gerarchia di elementi tra loro interconnessi, finalizzato all'ottenimento di un sistema completo e interconnesso, puntando a concetti di interoperabilità e interconnessione a livelli sempre crescenti [1].

L'interoperabilità si pone alla base della rivoluzione industriale 4.0: con il solo concetto limitante di Digital Twin, l'interoperabilità non risulta possibile partendo spesso da sistemi ed ecosistemi proprietari. Con l'avvento dell'Asset Administration Shell, i dati contenuti nelle copie digitali vengono tradotti nel formato unico e standardizzato degli AAS, favorendo concetti non solo di interoperabilità ma anche di espandibilità e crescita del Business Value aziendale.

2.2.1 Requisiti e composizione di un AAS

Procediamo con un approfondimento maggiormente tecnico sul concetto di Asset Administration Shell: figura 2.1 mostra la struttura dettagliata di un AS, in particolare illustrando la connessione tra l'elemento fisico (denominato Asset) e il corrispondete Administration Shell. Distinguiamo inoltre l'Header dal Body. L'Header contiene informazioni di base e identificative relative all'AAS e all'Asset vero e proprio, mentre il Body si compone di alcuni Submodel. Quest'ultimi utilizzati per trasportare sull'AS le caratteristiche, peculiarità e operazioni contenute nell'Asset rappresentato [16].

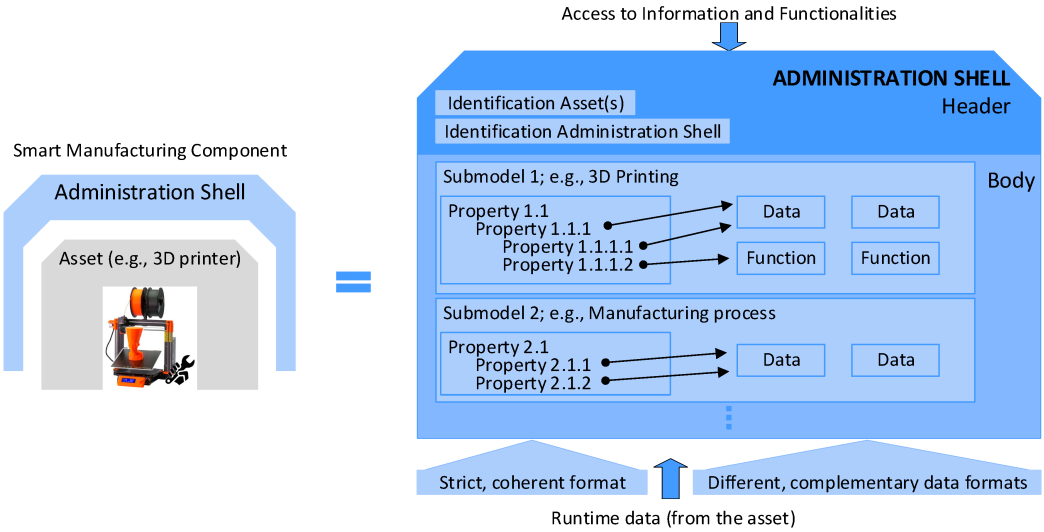


Figura 2.1: Struttura dettagliata di un AAS

Un Asset è un componente industriale, aderente a standard 4.0. Perciò fornito di un Digital Twin (eventualmente proprietario) e di un Administration Shell che lo rappresenta all'interno di un ambiente produttivo 4.0. Ciascun Asset viene caratterizzato da:

- Alcune informazioni di base: manuali, identificatori univoci, caratteristiche peculiari;
- Operazioni effettuabili e richiamabili mediante il relativo Administration Shell;
- Molteplici sensori e attuatori, dai quali risulta possibile raccogliere dati su cui effettuare analisi e implementare processi di Business Intelligence.

Ciascun Submodel contiene una o più Proprietà. Ciascuna di esse consente l'accesso a Funzioni, Dati, Sensoristica, ma anche alla possibilità di comunicazione con altri AAS, dando luogo a reti collaborative all'interno di un ambiente produttivo 4.0. Questi elementi possono essere standardizzati e rappresentanti non solo concetti ed elementi fisici ma anche (e soprattutto) virtuali. Attualmente spesso sono le informazioni e la relativa raccolta e analisi, a creare il maggiore valore per l'azienda.

2.2.2 Architettura per AAS: microservizi

Architetturalmente parlando gli AAS sfruttano a pieno il concetto dell'architettura basata su Microservizi: ciascun componente (Submodel, Property) viene implementato e fornito sotto forma di microservizio.

Questa metodologia fornisce numerosi vantaggi e si adatta al concetto di AAS, fornendo diversi strumenti relativi al Discovery dei servizi disponibili, alla sincronizzazione, alla gestione dei dati raccolti [12].

Uno dei concetti da tenere in maggiore considerazione, riguarda proprio il Service Discovery dei servizi a disposizione dell'utente finale, mediante i rispettivi AAS.

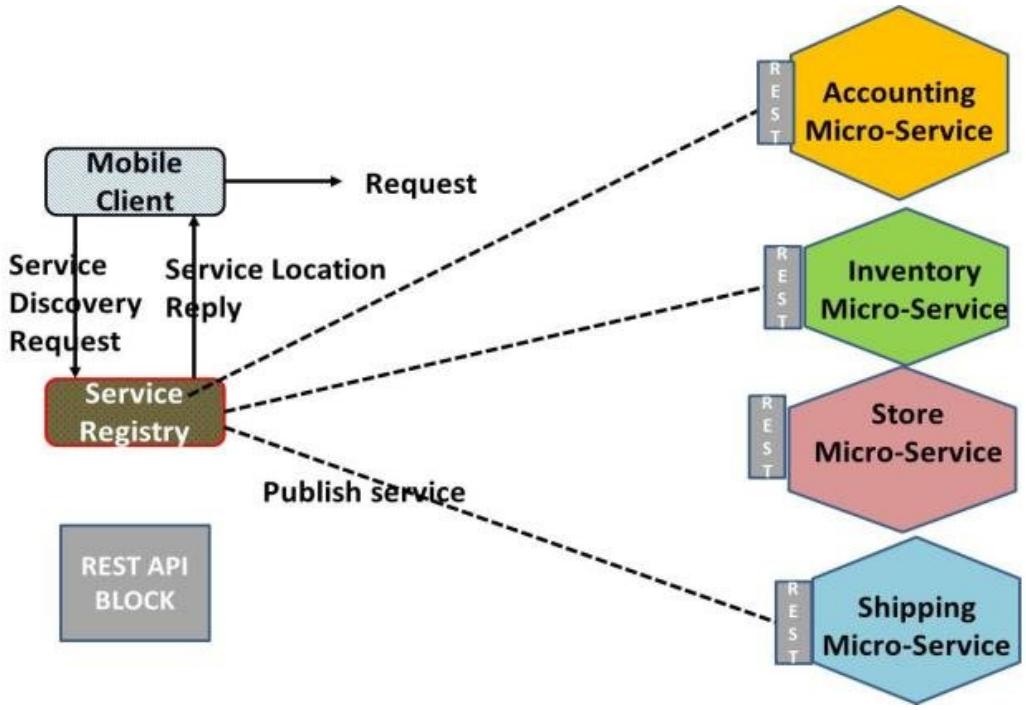


Figura 2.2: Funzionamento e utilità del Service Discovery

In figure 2.2 viene mostrata l'esemplificazione di un Service Discovery. Lo scopo è duplice:

- Autenticare l'utente che richiede l'accesso e la connessione a un microservizio;

- In caso positivo, fornirvi una modalità di connessione.

Mappando l'esempio in figura 2.2, nel mondo degli Asset Administration Shell:

- Ciascun microservizio, sarà mappato a un Administration Shell;
- Ciascun Administration Shell mette a disposizione determinati Submodels che forniscano accesso a informazioni e operazioni in esso contenute.
- Ciascun elemento, dall'AS al singolo Submodel dovrà essere univocamente identificato, in modo da garantirne univocamente accesso.

L'identificazione di ciascun elemento (Asset, Submodel, Properties, Funzioni) svolge un ruolo fondamentale, facilitando l'adozione di uno specifico AAS, Submodel o Proprietà all'interno del dominio in esame e per un determinato Asset. Differenti Identificatori sono a oggi presenti: URI (Uniform Resource Identifier), IRI (Internationalized Resource Identifier) e GUID (Globally Unique Identifiers) i più famosi e utilizzati.

Inoltre, numerosi aspetti di sicurezza devono essere mantenuti in considerazione: i singoli AAS vengono posti come intermediari tra il mondo IT (Information Technology) e OT (Operational Technology) [6].

Il mondo IT spesso viene reso disponibile a utenti esterni, mentre il mondo OT (prevalentemente relativo all'ambito produttivo) veniva spesso isolato, verso concetti di massima sicurezza e solidità. La connessione tra questi ambienti ha quindi portato diversi svantaggi rispetto alla sicurezza, ma fornendo alle aziende accesso in tempo reale a informazioni di produzione [5]. Queste consentono di aumentare quello che è il Business Value aziendale, ottimizzando la produzione aziendale, minimizzando i tempi morti ed effettuando analisi real time sui dati raccolti.

Il client richiede quindi accesso a un AS, posto da intermediario tra la rete IT aziendale e la rete OT, a sua volta composta dai singoli Asset presenti sul territorio aziendale. Possiamo esemplificare l'utilizzo del Service Discovery, nel mondo degli AAS, come in figura 2.3.

In termini di comunicazione tra microservizi, nella maggioranza dei casi a oggi si utilizza il protocollo OPC UA per fornire un iter di comunicazione Machine-To-Machine, cioè tra elementi e prodotti industriali e Machine-to-IT. Tuttavia per ricoprire altri ambiti maggiormente eterogenei, tutt'ora vengono utilizzati protocolli di comunicazione standardizzati, ad esempio REST (Representational State Transfer), come visibile in figure 2.2.

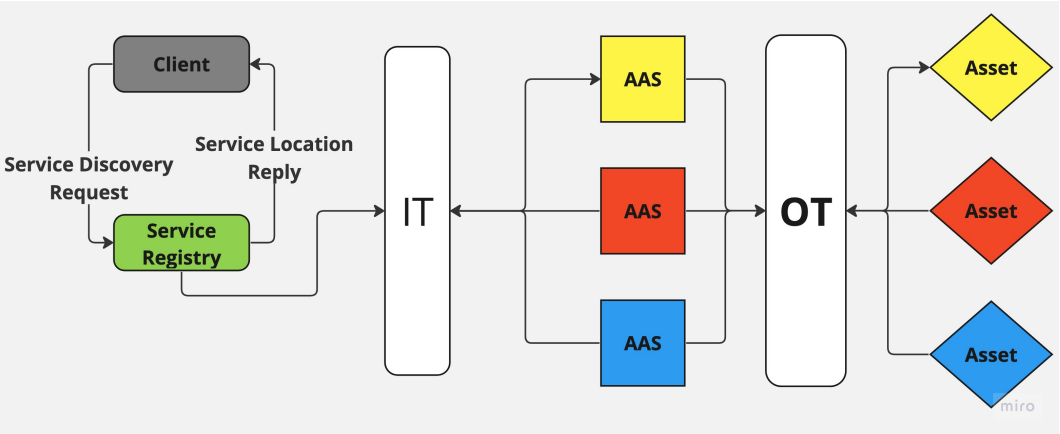


Figura 2.3: Funzionamento e utilità del Service Discovery nell'industria 4.0

2.2.3 Verso gli Active AAS

Inizialmente gli AAS vennero concepiti come "semplici" gestori e raccoglitori dei dati e delle informazioni raccolte dai prodotti e dagli elementi industriali [17]. Il concetto si è evoluto nelle metodologie e nei fatti precedentemente esaminati: fornitura e raccolta di dati, accesso a operazioni, concetti di Interoperabilità, modellazione dell'intero ciclo di vita del prodotto.

Il concetto di Active AAS diviene perciò sempre più importante: espandere ulteriormente i concetti alla base delle nuove tecnologie, arrivando alla costruzione di reti dinamiche, auto organizzanti, auto ottimizzanti e self reliant senza la necessità (presente nei Passive AAS) di una entità di controllo superiore che coordini comportamenti complessi e autonomi. Questo richiede un determinato grado di autonomia e di funzionalità decisionali all'interno dei singoli AAS. Le funzionalità attive vengono espletate dai singoli Administration Shell che collaborano, nel raggiungimento di un obiettivo finale: ottimizzazione dei processi produttivi, diminuzione dei tempi morti nella produzione, gestione automatizzata in caso di guasti sui macchinari.

Regole, comportamenti e interpretazioni dei dati di ciascun Administration Shell possono essere configurate a priori, puntando all'ottenimento di comportamenti auto coordinanti finalizzati al raggiungimento di uno o più specifici obiettivi.

Le funzionalità descritte, come illustrato in figura 2.4 non vanno però a escludere quanto prima introdotto. Ottenendo duplici funzionalità da ciascun

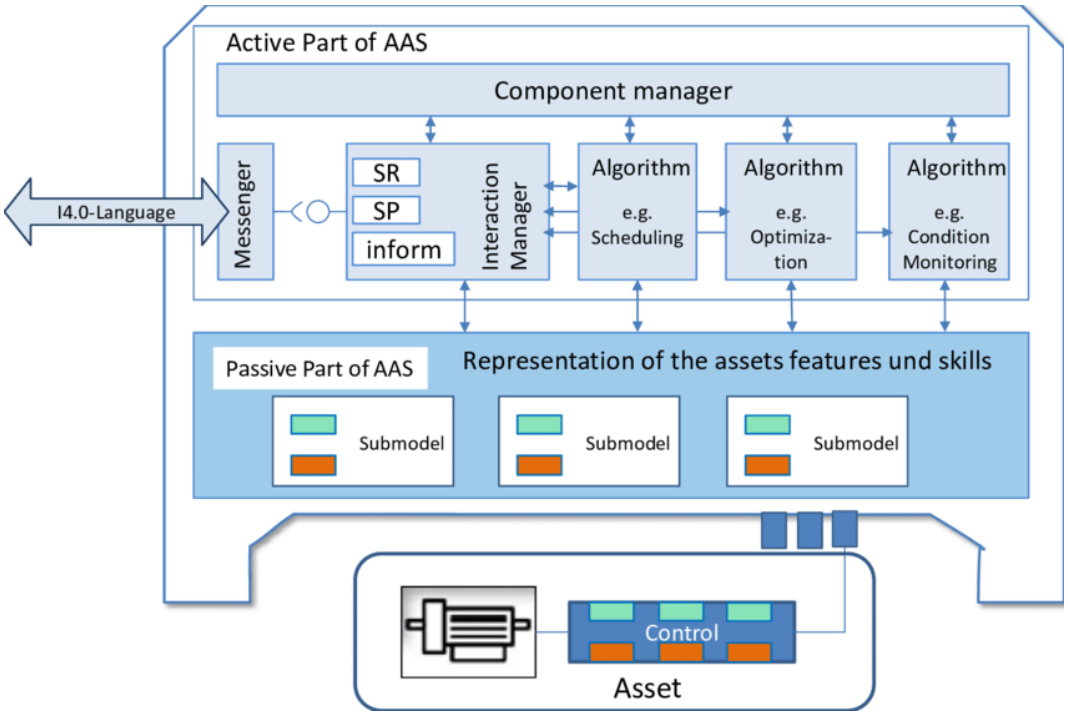


Figura 2.4: Struttura di un Active AAS

AAS: capacità e autonomia decisionale su determinati aspetti produttivi, raccolta e analisi di dati fornite mediante Submodels e Properties insieme alle operazioni e peculiarità del singolo Asset.

2.3 Requisiti di sicurezza e protezione dei dati

L'Industria 4.0 propone l'Asset Administration Shell (AAS) come implementazione ed estensione del concetto di Digital Twin, con l'obiettivo finale di scambiare dati e servizi relativi ad Asset dalla produzione fino alla dismissione. All'interno di ambienti industriali, l'integrazione degli AAS può però comportare nuovi problemi di sicurezza e protezione di cui tenere conto [7].

Si cita a titolo esemplificativo TRITON. Un nuovo malware del sistema che consente la riprogrammazione dei Controller di impianti industriali, costringendo il processo ad avviare una procedura di sicurezza che porta all'arresto automatico dei processi industriali [3].

Come mostrato in figura 2.6 l'Administration Shell di un Asset, vi interagisce ottenendo dati e misurazioni provenienti da determinati sensori, e inviano informazioni e comandi. I canali di comunicazione indicati sono il primo elemento critico e il primo vettore con il quale un attacco può fare breccia nella rete aziendale.

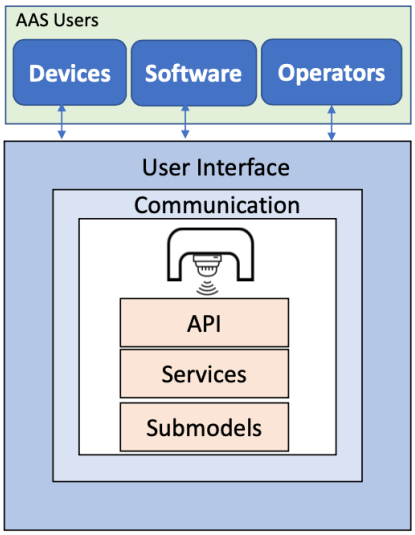


Figura 2.5: Struttura di un AAS, relativamente ai requisiti di sicurezza.

Attualmente non esiste un'analisi delle minacce relative agli AAS che copra tutte le fasi del ciclo di vita dell'Asset su scenari differenti. Per l'implementazione vengono inoltre utilizzate tecnologie rilevanti al mondo IT (Information Technology): tecnologie Web, database e protocolli di comunicazione. Tuttavia, ciascun AAS per sua natura, tende a operare all'interno del dominio OT (Operational Technology). Questo fatto crea incertezza nella scelta di un approccio standardizzato per la sicurezza dell'applicazione, a oggi mancante per il mondo dei Digital Twin e in particolare per gli AAS [2].

In particolare, gli ambiti che introducono vulnerabilità al concetto di AAS e alla sottostante rete OT, possono essere categorizzati come di seguito:

- Utenti e relative interfacce: furto d'identità o dei dati di accesso;

- Canali di comunicazione, e la stessa comunicazione: sniffing, attacchi Man-In-The-Middle;
- Modifiche e manipolazioni all'API offerta da uno specifico AAS: fallimento delle procedure di autenticazione, token di sicurezza, eccessiva esposizione dei dati verso l'esterno;
- Servizi e Submodels, correlati all'API precedenti;

Le principali mancanze alla sicurezza e alla protezione degli AAS e delle relative risorse, nasce principalmente dalla natura precedente delle reti OT. Le reti IT spesso erano esposte a internet e conseguentemente progettate per gestirne la sicurezza e la protezione. Con l'avvento del concetto di Digital Twin e Asset Administration Shell, la rete OT è stata integrata nella rete aziendale e connessa alla rete IT, esponendo numerose falle di sicurezza prima ignorate e/o non conosciute.

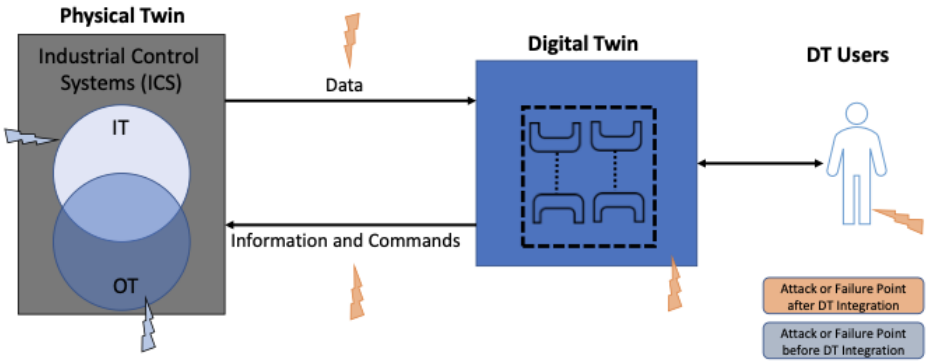


Figura 2.6: Interazione tra Digital Twin e vettori di attacchi alla sicurezza.

In ogni caso risulta quindi necessario coordinare lo sviluppo dell'AAS insieme a quello del rispettivo Asset, ma questo non è sempre possibile, dati i requisiti differenti per ogni caso d'uso che si va ad affrontare.

Possiamo generalmente distinguere tre differenti possibilità di sviluppo dell'AAS e del rispetto Asset [3]:

- Nel primo (e migliore) caso lo sviluppo dell'AAS viene svolto da zero e in contemporanea allo sviluppo del relativo Asset. Avendo carta bianca,

formulando correttamente i requisiti di sicurezza e protezione diviene possibile ottenere un sistema completo sotto questo punto di vista;

- La seconda possibilità, concerne la preesistenza dell'Asset, per cui un Digital Twin viene sviluppato: in questo caso è fondamentale una corretta stesura dei requisiti di sicurezza, tali da non creare collisioni con i requisiti operativi preesistenti;

- Nella terza, e ultima possibilità esplorata, si ha il contrario della prima: sviluppiamo un Administration Shell a priori, senza il rispettivo Asset.

In ciascuno dei casi esplorati, una corretta raccolta dei requisiti risulta fondamentale. Conoscere l'impatto dei requisiti di sicurezza sui requisiti di protezione dei dati e dell'infrastruttura è fondamentale [8].

2.4 AAS e 5G

Il contesto funzionale di un AAS per il 5G nasce dalla richiesta di una tecnologia di comunicazione wireless tra due nodi. Tuttavia, descrivere un sistema 5G come un singolo AAS potrebbe non essere pratico, poiché si tratta di un sistema ampio e complesso che comprende molte entità funzionali. Si dovrebbe quindi considerare la possibilità di introdurre ulteriori divisioni. Complessivamente il 5G fornisce un mezzo di comunicazione tra due endpoint attraverso una rete. Gli Asset 5G da descrivere potrebbero includere il collegamento di comunicazione tra i due endpoint, le apparecchiature di comunicazione presso gli endpoint e la rete complessiva [1].

Lo scopo è perciò quello di ottenere un AAS, che consenta e gestisca la comunicazione tra elementi industriali 5G-Enabled: forniti perciò degli elementi necessari a gestire una comunicazione wireless con un altro endpoint.

Introduciamo innanzitutto alcuni concetti di base relativi all'architettura di rete 5G.

Come visibile in figura 2.7, distinguiamo il 5G UE (User Equipment), nella maggior parte dei casi un dispositivo wireless, a cui fornire una determinata comunicazione verso un altro endpoint, e i seguenti componenti:

- 5G RAN (Radio Access Network): fornisce accesso alla tecnologia 5G mediante collegamento Radio (Wireless);

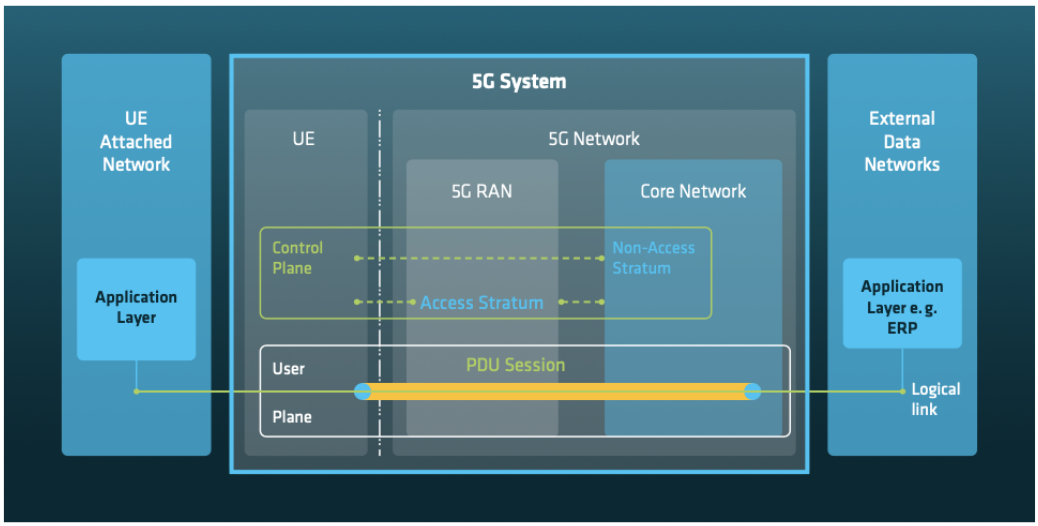


Figura 2.7: Architettura generica di un sistema 5G, tra un endpoint e una rete esterna.

- 5G CN (Core Network): fornisce la base su cui istanziare una o più 5G RAN e una o più funzioni di rete virtualizzate (e/o standardizzate).

Il sistema 5G è responsabile delle singole sessioni PDU e delle funzioni corrispondenti, dalla gestione degli indirizzi IP al QoS. Ciascuna PDU viene istanziata sopra il Transport Layer di rete, ad esempio per le connessioni radio tra UE e RAN 5G. Questa separazione della PDU relativa a ciascun utente rispetto al Transport Layer sottostante, fornisce ampia modularità ed estensibilità della tecnologia 5G. La progettazione della tecnologia 5G è altamente modulare, come si può vedere nei domini funzionali, nei livelli di rete e nei piani, che sono indipendenti l'uno dall'altro in modo da poter essere sviluppati e scalati separatamente. La modularità è evidente anche nel modo in cui i domini funzionali sono suddivisi in entità funzionali chiamate funzioni di rete (NF) appartenenti al piano di controllo del sistema 5G, descritte nelle sezioni seguenti [1].

2.4.1 Vantaggi e prospettive nell'adozione del 5G

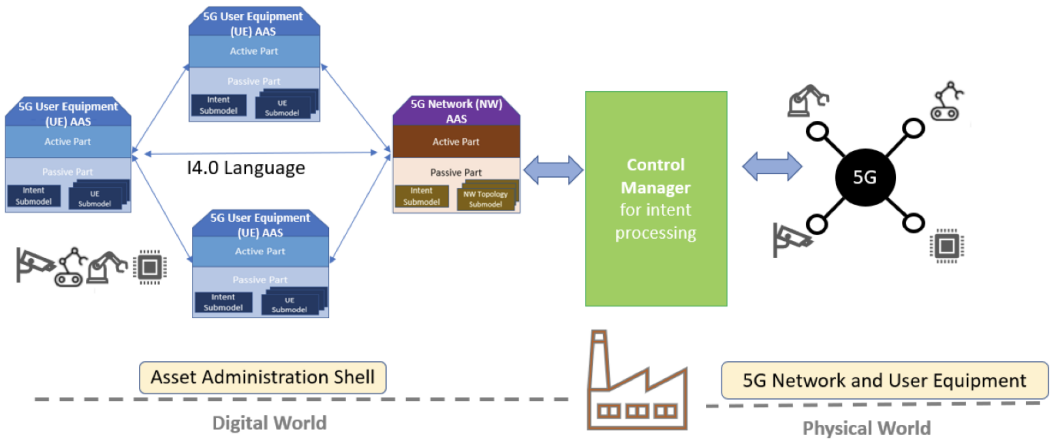
Diverse sono le proposte a oggi presenti per lo sviluppo di un 5G AAS su apparati industriali 5G Enabled. La modularità fornita dal 5G stesso, favo-

risce altrettanta libertà (e modularità) nello sviluppo degli AAS necessari a interconnettere due o più endpoint mediante tecnologia wireless. Di seguito vengono prese in considerazione due differenti implementazioni.

Nella prima, data la complessità e la modularità intrinseca nella tecnologia del 5G, si decide di utilizzare due Administration Shell [1]:

- 5G UE AAS: descrive il singolo endpoint della comunicazione wireless mediante tecnologia 5G;
- 5G Network AAS: vengono incluse tutte le informazioni in modo strutturato e standardizzato relative al 5G RAN (Radio Access Network), 5G CN (Core Network).

Altre opzioni riguardano il possibile utilizzo di un unico AAS, basato su tecnologie Open Source (ad esempio EURANISM [4] per l'endpoint 5G, Open5GS [11] per il Core Network) per l'introduzione della connettività 5G in un determinato apparato IoT e 5G Enabled [13].



Numerose sono le motivazioni nell'adozione di Asset industriali 5G-Enabled e dei relativi Administration Shell. Innanzitutto si ha la possibilità di accedere a informazioni raccolte dai macchinari dall'esterno, senza l'utilizzo necessario del Service Discovery (affrontato nella sezione 2.2.2). Diviene possibile introdurre numerosi concetti di resilienza e flessibilità nel sistema industriale 4.0, sia dal punto di vista degli Asset ma anche e soprattutto dal lato della connettività 5G e relativa raccolta, analisi, trasmissione ed elaborazione dei dati raccolti.

Capitolo 3

Tecnologie Utilizzate

Il capitolo seguente tratterà una breve introduzione alle tecnologie utilizzate nel progetto in analisi.

Partendo dalle tecnologie base per il mondo del Network attuale, il Software Defined Network (SDN) e quanto ne concerne: Mininet per la simulazione di reti locali, Open vSwitch per la programmazione dei singoli switch di rete e infine il framework Ryu, utile e necessario nella definizione del comportamento di ogni controller utilizzato e istanziato in ciascuno switch.

Verrà trattato poi in modo approfondito l'utilizzo di Eclipse Basyx: software Open Source che permette e facilita la programmazione di strutture basate sul concetto degli Asset Administration Shell. Fornisce alcuni elementi predefiniti tra cui il Server, il Registry e una Web UI che verranno approfonditi nella sezione 3.2.

Infine, verrà brevemente introdotto il concetto di virtualizzazione, mediante l'utilizzo di Docker Container nel caso di studio, alla sezione 3.3.

3.1 SDN: Software Defined Network

SDN è un paradigma di rete in cui il Control Plane della rete viene logicamente separato dal Data Plane. Questa separazione fornisce numerosi vantaggi rispetto alle reti "tradizionali" e non Software Defined. Il Control Plane viene implementato e fornito da un'entità esterna: il Controller. Mentre il Data Plane risiede negli Switch di rete, che si occupano del Packet Forwarding: l'inoltro dei pacchetti (con un comportamento definito dal rispettivo Controller) verso una determinata destinazione (ad es. contenuta nell'Header del pacchetto).

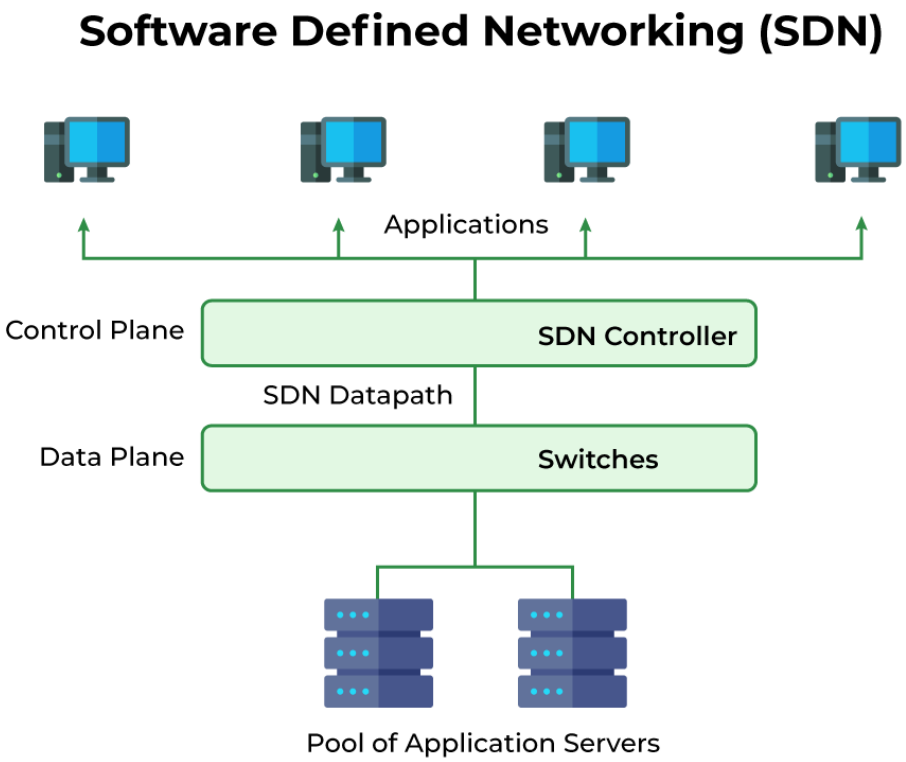


Figura 3.1: Esemplificazione di un Software Defined Network.

La comunicazione tra gli Switch e i Controller (le entità di rete che ne definiscono il comportamento) avviene mediante il protocollo OpenFlow [9]. L'adozione di reti Software Defined ha inoltre concesso numerosi vantaggi sotto diversi punti di vista:

- I costi vengono ridotti, grazie allo spostamento e all'implementazione del Control Plane nei Controller di rete;
- I Controller mantengono tutte le informazioni relative al comportamento e alla composizione di una determinata rete, perciò la manutenibilità del Control Plane viene notevolmente aumentata;

Di seguito si introducono alcune tecnologie maggiormente specifiche utilizzate nel caso d'uso in esame.

3.1.1 Mininet

Mininet consente l'emulazione di reti complesse, mettendo a disposizione numerose API (es. in Python) per la loro creazione. Vengono creati host di rete, switch, router e collegamenti su un singolo kernel Linux. Un host Mininet si comporta come un host su un Network non emulato: vi si può accedere con ssh (se si avvia sshd e si crea un ponte di rete con l'host) ed eseguire programmi arbitrari (compreso tutto ciò che è installato sul sistema Linux sottostante). I programmi eseguiti possono inviare pacchetti attraverso un'interfaccia Ethernet (anch'essa ovviamente simulata), con una determinata velocità e ritardo del collegamento. I pacchetti vengono elaborati da uno switch, router o middlebox Ethernet, il cui comportamento viene eventualmente definito da controller esterni [10].

In breve, gli host virtuali, gli switch, i link e i controller di Mininet sono corrispondenti alle reti non emulate, anche per il comportamento finale che vi si ottiene. Spesso è infatti possibile creare una rete Mininet partendo da una rete hardware o viceversa, ed eseguirvi lo stesso codice binario e le stesse applicazioni su entrambe le piattaforme.

I vantaggi offerti da una piattaforma Open Source, come mininet sono perciò numerosi: le possibilità di creazione sono numerose (ed eventualmente complesse) e non richiedono hardware e investimenti per il loro test iniziale, favorendo maggiore lavoro con minori costi e in minor tempo.

Vi sono però alcune limitazioni al suo utilizzo:

- Utilizzando un kernel linux per la virtualizzazione dei componenti di rete, non è possibile utilizzare Mininet su ogni piattaforma;

- Di default la rete emulata, è isolata dalla rete locale o da Internet, non permettendo comportamenti complessi o accessi dall'esterno. Questo comportamento è vantaggioso sotto determinati punti di vista (sicurezza di rete in primis), ma può essere arginato con l'utilizzo del Nat, a disposizione nelle Api fornite;

- Infine Mininet non fornisce le implementazioni del Controller OpenFlow (da caricare sugli Switch di rete). Perciò per introdurre comportamenti complessi all'interno della rete (ad es. Firewall), sarà necessario implementarli mediante componenti esterni.

Nel caso in esame, per introdurre comportamenti complessi nella rete emulata, si è utilizzato il framework Ryu per la programmazione dei Controller e Open vSwitch per simulare gli Switch virtuali su cui caricare i Controller.

Dal punto di vista pratico, è possibile creare una rete Mininet, con l'unico comando di seguito:

```
sudo mn --switch ovs --controller ref --topo tree,depth=2,fanout=8
```

In questo specifico caso verrà create una rete di profondità due e larghezza 8. Perciò composta da 9 Switch e 64 Host totali. Verrà inoltre utilizzato Open vSwitch come tecnologia di base per la creazione degli Switch virtuali necessari.

3.1.2 Open vSwitch

Open vSwitch è uno switch virtualizzato di classe enterprise, multilayer, distribuito mediante la licenza Apache 2.0. Il progetto nasce per offrire una forte capacità di automazione all'interno dei data centers di nuova concezione grazie al supporto del Software Defined Networking.

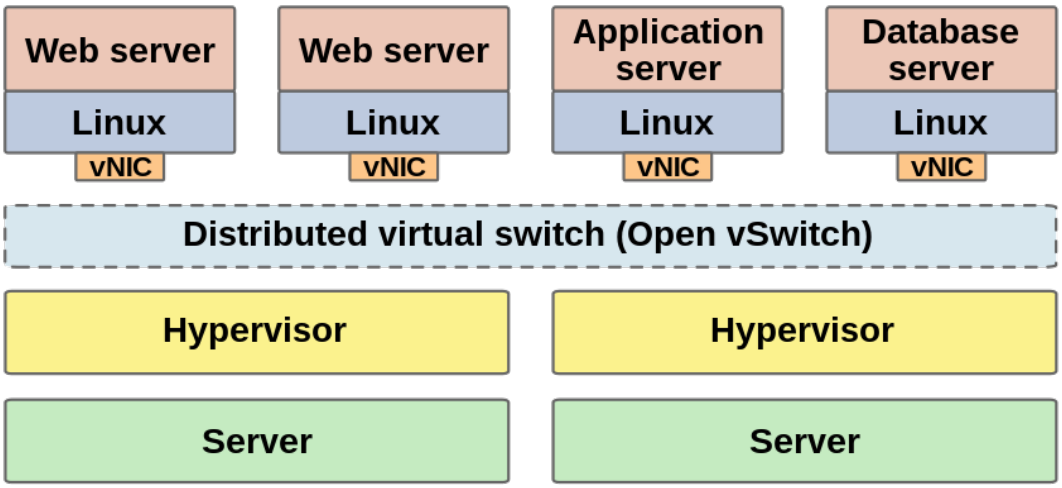


Figura 3.2: Open vSwitch nelle nuove reti Software Defined

Il principale obiettivo voluto con l'introduzione di Open vSwitch, è perciò quello di fornire uno stack tecnologico che supporti l'implementazione di Switch virtuali nei vuoi ambienti virtualizzati e nelle nuove reti Software Defined.

Mininet (introdotto nella sezione 3.1.1) utilizza come precedentemente descritto degli Switch virtuali tra i componenti principali per la creazione di una rete virtuale. Tipicamente utilizza il Linux Bridge (nativo) od Open vSwitch, per switchare i pacchetti tra le interfacce ethernet a disposizione.

3.1.3 Ryu Framework

Ryu è un framework per reti Software Defined. Fornisce componenti software con API ben definite che facilitano gli sviluppatori nella creazione di nuove applicazioni di gestione e controllo della rete, quali ad esempio i Controller per Switch Virtuali (forniti da Mininet e Open vSwitch nel caso in esame).

Ryu supporta diversi protocolli per la gestione dei dispositivi di rete, come OpenFlow, Netconf, OF-config, ecc. Questi protocolli vengono utilizzati come protocollo di comunicazione tra gli Switch (definiti ad esempio mediante Open vSwitch) e i Controller a loro connessi (creati mediante l'utilizzo del Framework e delle Api a disposizione da Ryu).

Vengono inoltre fornite numerose implementazioni di base di alcuni controller, quali ad esempio Ryu-Firewall o Ryu simple switch. Ciascuna di esse fornisce accesso alle Api, da cui risulta possibile visualizzare statistiche sull'utilizzo del Controller o apportare modifiche alle azioni di inoltro dei pacchetti configurate.

Mediante il comando riportato è possibile creare una rete virtuale mediante Mininet. Quest'ultima utilizzerà Open vSwitch per la creazione degli switch virtuali e verrà configurata per l'utilizzo con un Controller remoto, sulla porta 6633.

```
sudo mn --topo single,3 --mac --switch ovsk --controller remote -x
```

Con il comando successivo, avviamo il Controller (ryu-firewall) sulla porta 6633. Otteniamo una rete virtuale con un singolo Switch e 3 Host. Sul singolo switch verrà istanziato il Controller esterno, che fungerà da firewall.

```
ryu-manager ryu.app.rest_firewall --wsapi-port 6633
```

3.2 Eclipse Basyx

La transizione verso l'Industria 4.0 richiede software aziendali che vadano a supportare il continuo cambiamento dei processi produttivi, integrando allo

stesso tempo macchinari tra loro eterogenei in un unico sistema 4.0 integrato cross-company.

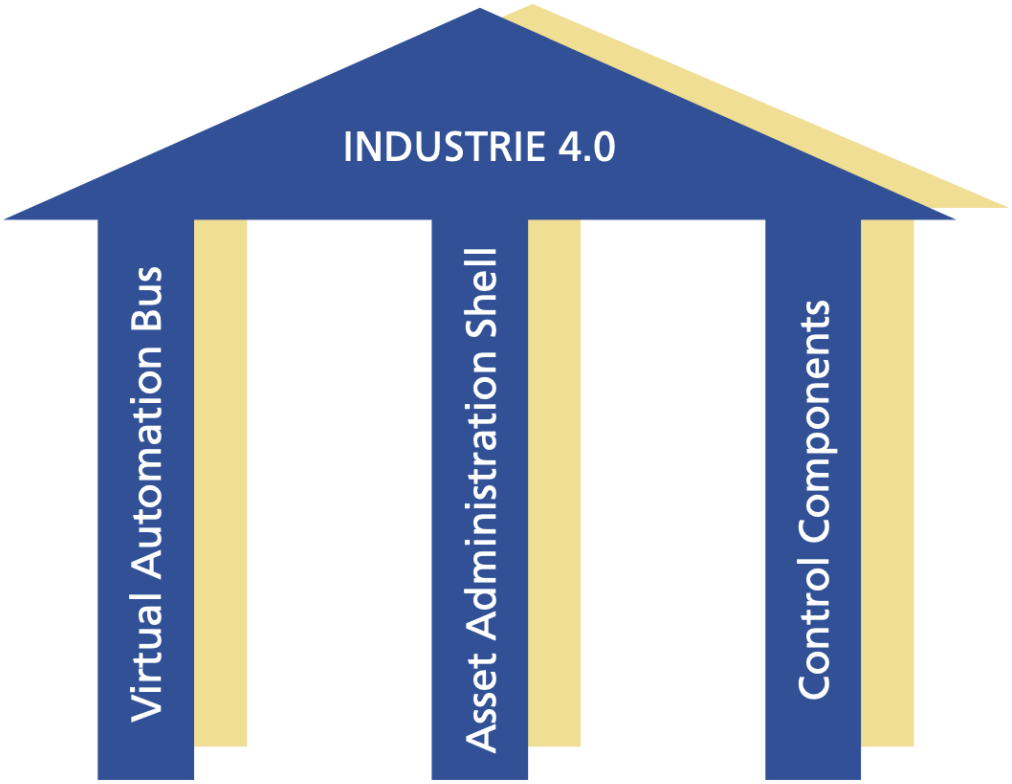


Figura 3.3: Pilastri del progetto BaSys 4.0

In questo ambito, il progetto BaSys 4.0, si è posto l'obiettivo di ottenere una soluzione che favorisse la realizzazione dei Digital Twin, necessari nelle fabbriche 4.0. L'obiettivo finale era quello di integrare e interconnettere le tecnologie esistenti, con lo scopo di realizzare e supportare la transizione 4.0.

Eclipse Basyx è il risultato del progetto BaSys 4.0. Rappresenta una piattaforma Open Source che fornisce un Middleware completo, verso la creazione e il supporto del concetto di Asset Administration Shell, con l'obiettivo di supportare la transizione all'Industria 4.0.

I componenti necessari alla realizzazione del sistema di base BaSys, sono i seguenti: Asset Administration Shell (2.2), SubModel (2.2.1), Control Components, Registry e comunicazione End-to-End.

3.2.1 Basyx Architecture

Architetturalmente, il middleware fornito e implementato da BaSyx consente di inserire su un unico mezzo di comunicazione (ad es. una rete aziendale) tutti i componenti 4.0. In questo ambito il VAB (Virtual Automation Bus) risulta fondamentale, fornendo le funzionalità necessarie a consentire la comunicazione tra reti eterogenee.

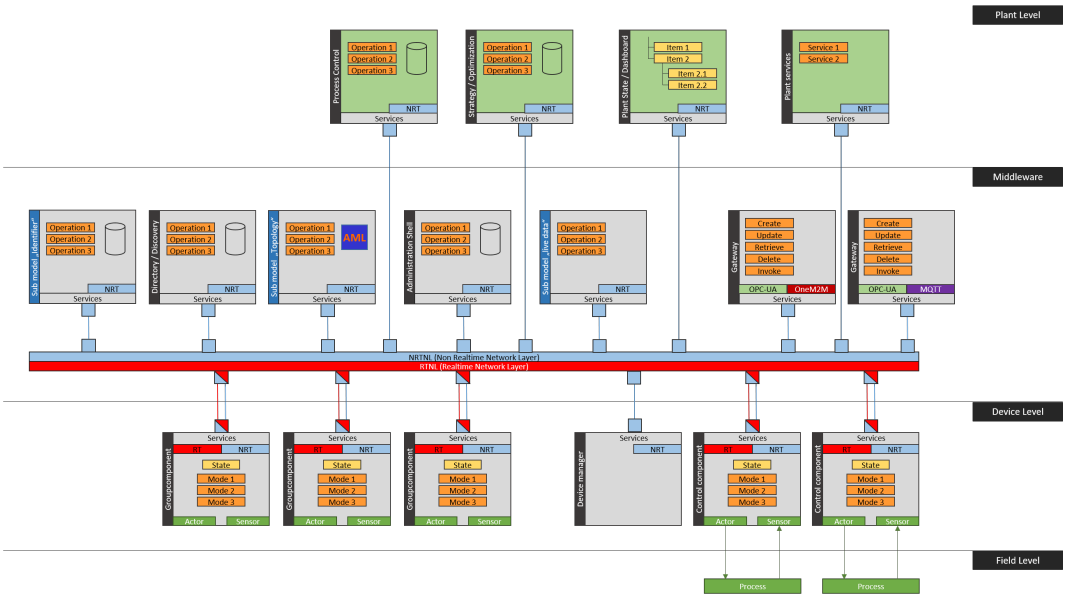


Figura 3.4: Architettura di Eclipse Basyx.

Un sistema BaSyx consiste perciò di numerosi componenti su due livelli principali, tutti connessi allo stesso mezzo di comunicazione:

- A livello aziendale, vi sono i componenti principali (Registry, Server), spesso forniti precompilati e pronti all'uso sotto forma di Docker Container;

- A livello produttivo, i componenti vengono implementati mediante Edge-Devices, alcuni dei quali supportano nativamente l'utilizzo di BaSyx.

3.2.2 BaSyx Registry e Server

Il Registry consente la registrazione di nuovi AAS e la ricerca degli AAS registrati in base al loro identificatore o identificatori univoci, svolgendo quindi la funzione di Service Discovery. Fornisce l'accesso a tutte le Asset Administration Shell registrate ed è quindi il primo punto di contatto per la maggior parte delle applicazioni e dei dispositivi Industrie 4.0. Inoltre, questo componente definisce le funzioni API di base per la registrazione e la cancellazione degli AAS, nonché per la ricerca di un AAS.

Solitamente, come anche nel caso in esame, il Registry viene fornito mediante Docker Container.

Il server, infine, funge da repository per l'hosting dei Asset Administration Shell e dei Submodels creati, fornendo tutte le API necessarie alla loro manipolazione. Richiede in ogni caso un Registry per il funzionamento e il suo utilizzo.

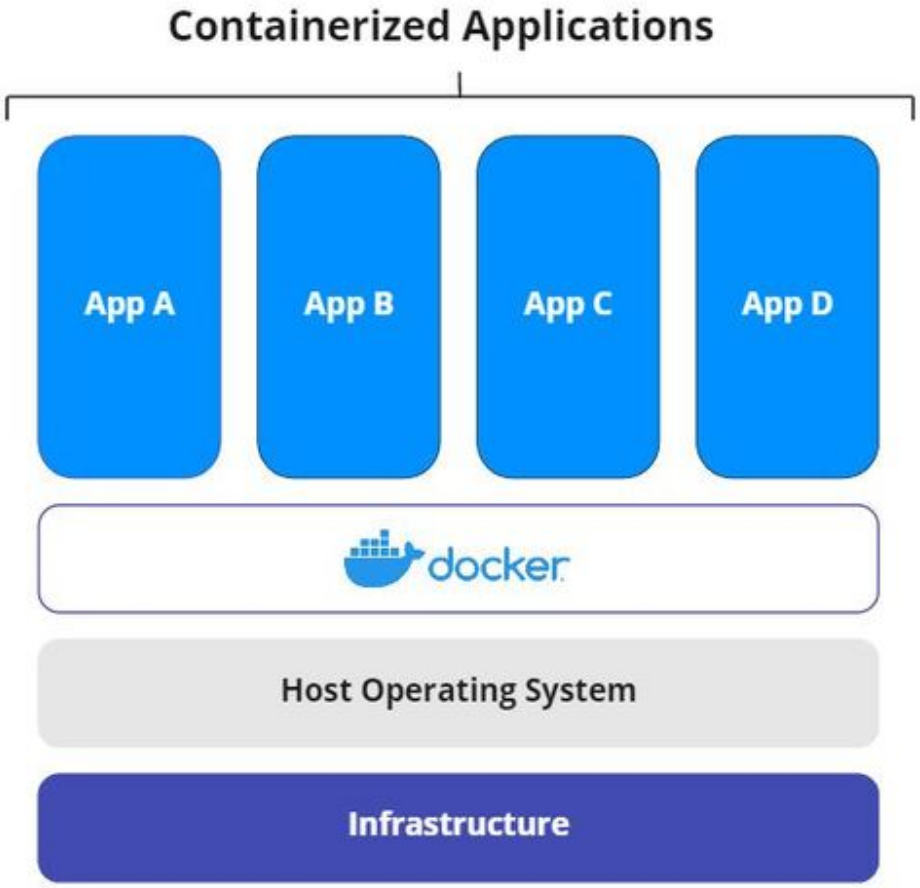
3.2.3 Control Components

I componenti di controllo rappresentano sia un'interfaccia unificata per il dispositivo, sia l'implementazione di funzioni di controllo orientate ai servizi sui controllori PLC. Svolgono perciò il compito principale di fornire un'interfaccia dei dati e delle operazioni del singolo Asset verso l'esterno, indipendentemente dal produttore o dai protocolli utilizzati dal macchinario stesso.

3.3 Virtualizzazione: Docker Containers

Docker è un software Open Source per eseguire applicativi in ambienti virtualizzati, perciò isolati, minimali e facilmente distribuibili, chiamati Container, con l'obiettivo di semplificare e snellire i processi di deployment del software.

Per favorire una corretta esecuzione dei componenti forniti da Basyx indipendentemente dal sistema operativo e dalle risorse del sistema, si è deciso di utilizzare i Docker Container forniti, per il componente Registry, Server e per la Web UI.



Nella directory generale, è presente un file Docker Compose, utilizzato per l'effettivo avvio dei componenti necessari al middleware Eclipse BaSyx.

Capitolo 4

Sviluppo del progetto

Il progetto nasce con l'obiettivo di ottimizzare e automatizzare l'architettura di rete in una fabbrica aderente allo standard imposto dall'Industria 4.0. In particolare si voleva avere la possibilità di definire il comportamento di rete (attraverso apposite configurazioni dei Controller e relativi Switch) mediante l'utilizzo di un Asset Administration Shell per il gateway tra la rete IT e la rete OT aziendale.

La rete IT nel caso in esame, è aderente alla realtà: consiste degli AAS relativi al Gateway, agli Switch e ad alcuni ipotetici macchinari industriali.

La rete OT, nel caso in esame solo esemplificata, viene invece composta dai soli Macchinari industriali 4.0-Enabled, gestiti mediante gli AAS nella rete IT.

Data la natura sperimentale del progetto, per creare scenari esemplificati ma comunque realistici, si è deciso di utilizzare Mininet (Sezione 3.1.1) insieme a Open vSwitch (sezione 3.1.2) e Ryu (sezione 3.1.3) per le configurazioni di rete successivamente presentate.

Inoltre, come stack tecnologico a supporto dell'Industria 4.0 e in particolare del concetto di Asset Administration Shell, si è utilizzato Eclipse BaSyx (sezione 3.2) come middleware Open Source. BaSyx fornisce alcuni elementi precompilati e pronti all'utilizzo mediante Docker Container (sezione 3.3), utilizzati nel progetto in questione:

- BaSyx Registry: componente che svolge le funzioni di Service Discovery a cui la Web UI è collegata per visualizzare AAS, Asset e Submodels disponibili in rete;

- BaSyx Server: svolge il ruolo di repository degli AAS e Submodels creati. Necessita in ogni caso del Registry;
- BaSyx Web UI: fornisce una Web UI precompilata e pronta all'utilizzo. Questa viene collegata al Registry e al Server per visualizzare e agire sugli AAS e Submodels disponibili.

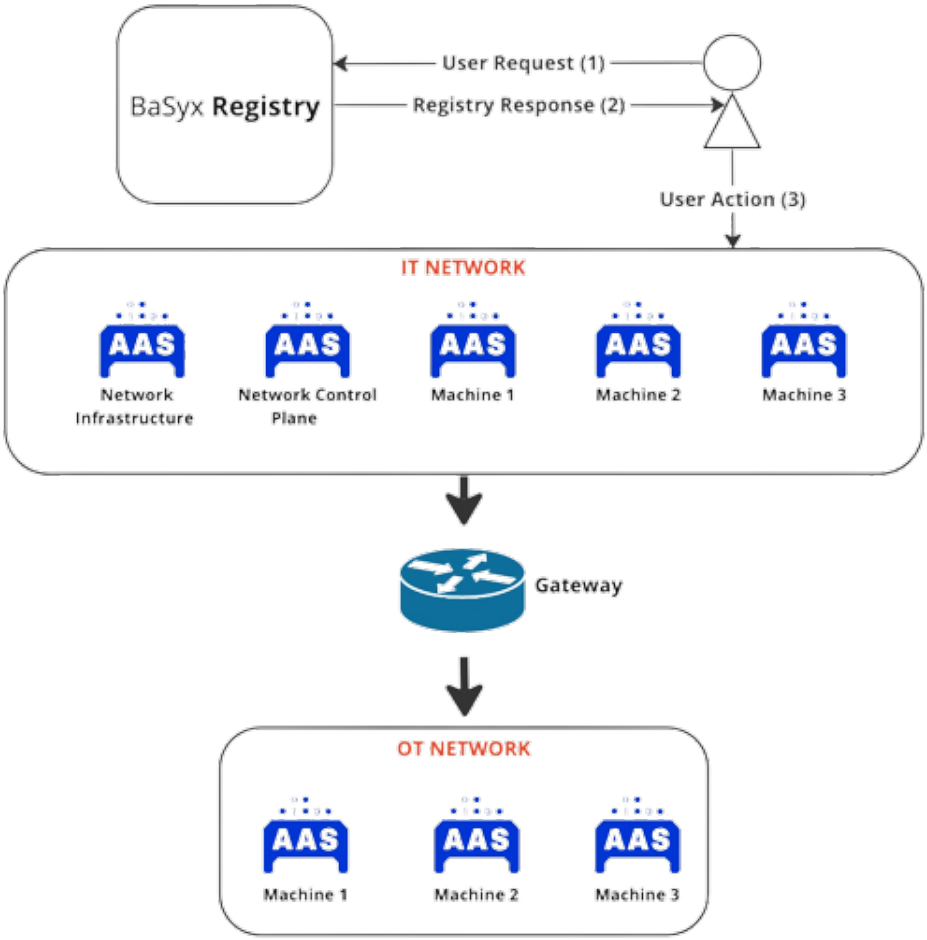


Figura 4.1: Architettura progettuale nel dettaglio.

In figura 4.1 viene mostrata l'architettura finale del progetto, opportunamente schematizzata e semplificata, i cui componenti saranno descritti in modo approfondito nelle successive sezioni.

4.1 Docker compose e virtualizzazione

BaSyx, utilizzato come stack tecnologico a supporto dell'Industria 4.0, fornisce alcuni elementi necessari al funzionamento mediante Docker Container. Un primo elemento fondamentale è quindi il file Docker compose utilizzato: quest'ultimo permette di integrare l'avvio dei tre container utilizzati: Registry, Server e Web UI.

Come visibile nel codice sotto riportato, per ciascun container vengono definiti alcuni argomenti di base: la porta di avvio e la configurazione di CORS (Cross-Origin Resource Sharing). CORS è un meccanismo basato sull'Header dei pacchetti HTTP, con il quale è possibile definire (per ciascun Server) quali pacchetti ricevere in base alla Sorgente contenuta nell'Header del pacchetto stesso.

Otterremo infine:

- BaSyx Registry sull'indirizzo http://localhost:8082
- BaSyx Server sull'indirizzo http://localhost:8081
- BaSyx Web UI sull'indirizzo http://localhost:3000

```
services:
  registry:
    image: eclipsebasyx/aas-registry:1.4.0
    ports:
      - "8082:4000"
    environment:
      basyxcontext_accesscontrolalloworigin: "*"
  server:
    image: eclipsebasyx/aas-server:1.4.0
    ports:
      - "8081:4001"
    environment:
      basyxcontext_accesscontrolalloworigin: "*"
  aas_gui:
    image: eclipsebasyx/aas-gui:v230703
    ports:
      - "3000:3000"
```

4.2 BaSyx Registry

Come descritto precedentemente, il Registry svolge il ruolo di Service Discovery: fornisce informazioni in merito ad AAS, Asset e Submodels disponibili all'utente o alla Web UI, al suo primo avvio. Perché il Registry renda disponibili le informazioni sugli AAS disponibili, nelle fasi preliminari quest'ultimi devono essere collegati al Registry, che vi accederà mediante il concetto di AASDescriptor o SubmodelDescriptor.

Ciascun Descriptor utilizzerà infine i seguenti parametri:

- L'effettiva Asset Administration Shell o Submodel precedentemente creata;
- L'indirizzo di rete a cui raggiungere l'oggetto in esame;

Inoltre ciascun elemento (AAS e Submodels), utilizza il concetto di Model Provider per mappare gli indirizzi richiesti dal Registry (o da un utente) sulla rispettiva struttura interna.

In figura 4.2, viene mostrato un diagramma di sequenza che va a mostrare il funzionamento del Model Provider nel progetto in esame:

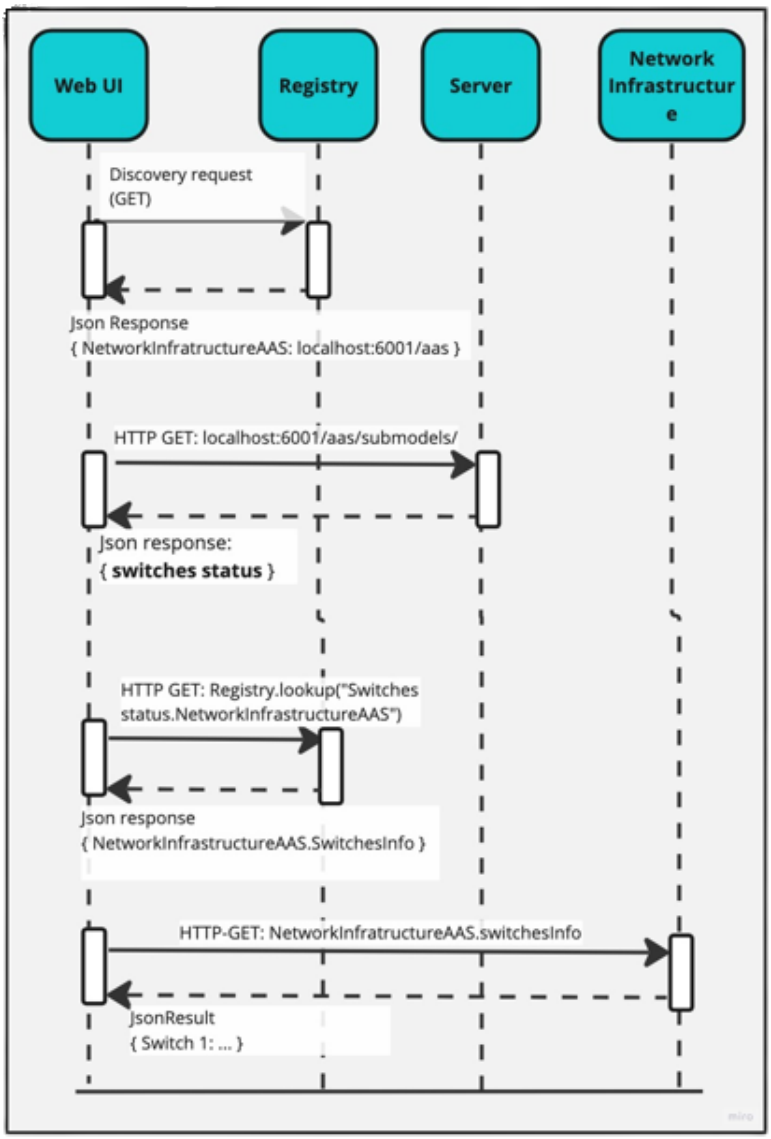


Figura 4.2: Diagramma di sequenza per mostrare l'utilizzo del Model Provider.

- La Web UI effettua una richiesta di Service Discovery al Registry: ottiene la lista e i rispettivi indirizzi IP degli AAS disponibili;

- Viene effettuata una seconda chiamata al Server, per ottenere i Submodels disponibili per uno specifico AAS: Network Infrastructure. Otteniamo la lista dei Submodels, tra cui: Switch Status;
- La Web UI invoca nuovamente il Registry per ottenere l'indirizzo IP del Submodel Provider relativo al Submodel Switch Status (che contiene la proprietà SwitchesInfo);
- Infine, avendo tutti gli elementi necessari, la Web UI invoca direttamente alla Network Infrastructure la proprietà Switch Info che restituisce un determinato risultato.

4.3 IT: Network Infrastructure e Control Plane

La rete IT viene composta dagli Asset Administration Shell, rispettivamente: Network Infrastructure, Network Control Plane e tre Macchinari denominati Machine1, Machine2 e Machine3. Ciascuno di essi viene composto da differenti Submodels e altrettanti Proprietà od Operazioni eseguibili e richiamabili.

In questa sezione verranno analizzati nel dettaglio gli Asset Administration Shell relativi al controllo e alla configurazione di rete: Network Infrastructure e Network Control Plane.

4.3.1 Network Infrastructure

L'Asset Administration Shell rappresenta il Digital Twin degli Switch interposti tra le rete IT (composta dagli AAS) e la rete OT (composta dai componenti industriali). Dato l'utilizzo di Mininet per esemplificare una determinata configurazione di rete, anche gli switch sono virtuali, utilizzando come tecnologia Open vSwitch.

Ciascuno di essi non dispone di un Controller in quanto esterno, rispettivamente sulla porta 6633 e sulla porta 6653 per Switch 1 e Switch 2 (come descritto nella sezione 4.4). Allo scopo di esemplificare differenti configurazioni di rete, si è deciso di consentire l'avvio di tre differenti Controller su ciascuno Switch, approfonditi nella sezione 4.5:

- Open Controller: ciascuno Switch agirà senza logica decisionale, inoltrando i pacchetti sempre e solo verso la sorgente voluta inizialmente;

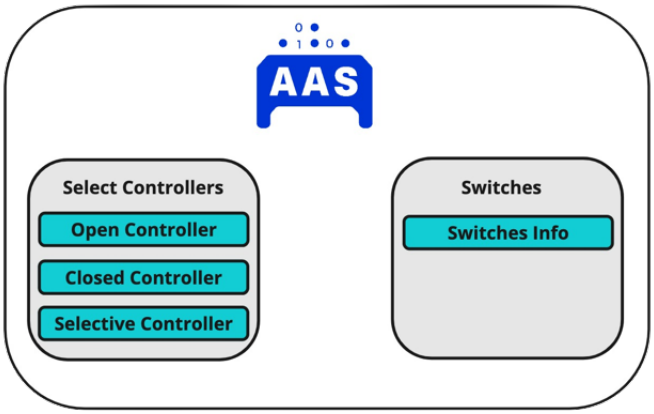


Figura 4.3: Network Infrastructure Asset Administration Shell

- Closed Controller: ciascuno Switch bloccherà i pacchetti da/verso l'altro Switch;
- Selective Controller: ciascuno Switch agirà come Firewall Default-Deny;

Come visibile in figura 4.3, l'AAS in esame è quindi composto da due Submodels: Select Controller che permette l'avvio dei controller specificati e Switch che consente la visualizzazione di alcune informazioni per verificare il corretto funzionamento di ciascun Switch.

4.3.2 Network Control Plane

L'Asset Administration Shell rappresenta il Digital Twin di un Asset non fisico ma virtuale: i Controller di rete. Network Control Plane mette infatti a disposizione due Submodel, uno per ciascun Controller di rete. Ciascun Submodel, come visibile in figura 4.4, mette a disposizione numerose operazioni, per consentire le seguenti operazioni:

- Aggregate Flows e All Flow Stats permettono di visionare le statistiche sul traffico di rete sul rispettivo Switch a cui il Controller è connesso;
- Get Role e Set Role permettono la visione e la modifica del ruolo di rete del Controller: Master o Slave;

- Set, Get e Delete Firewall Rules permettono le rispettive operazioni sulle regole da impostare sul Firewall (se attivato mediante Network Infrastructure AAS).

Ciascuna operazione indicata, avviene utilizzando le API messe a disposizione dal framework Ryu, utilizzato per lo sviluppo dei Controller a disposizione dell'utente [15], [14].

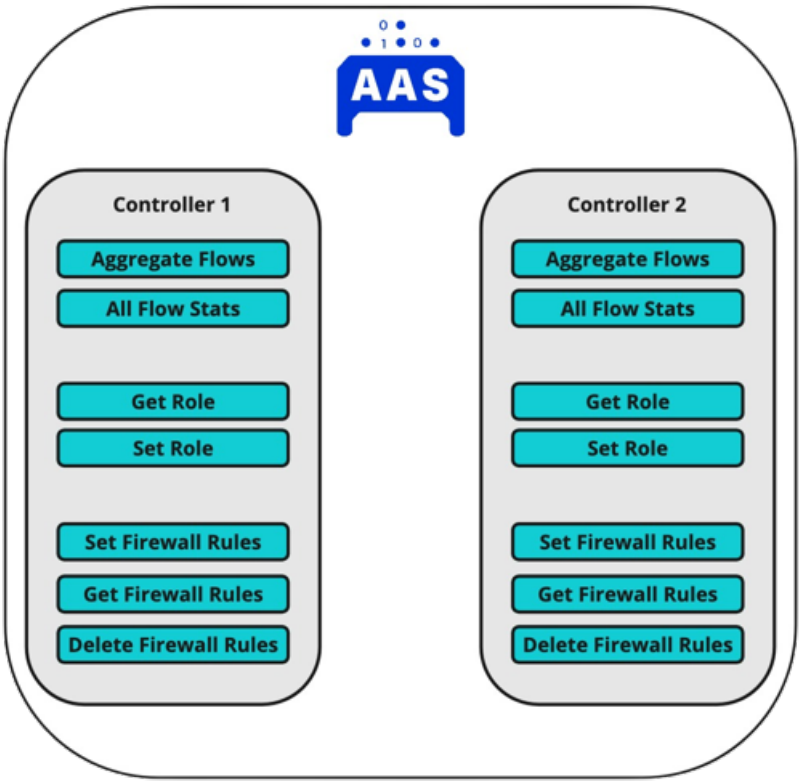


Figura 4.4: Network Control Plane Asset Administration Shell

4.4 Architettura di rete nel dettaglio

Come visibile in figura 4.1, tra la rete IT composta dagli AAS e la rete OT si interpone un Gateway composto da due Switch: questo consente la confi-

gurazione e la personalizzazione del comportamento di rete mediante gli AAS forniti, finalizzati al raggiungimento dell'architettura di rete finale.

Ciascuno Switch utilizza un Controller che può (se voluto dall'utente) svolgere la funzione di Firewall Default Deny e successivamente essere configurato con specifiche regole, tali da permettere solo uno specifico traffico tra host predefiniti.

In figura 4.6 viene mostrata l'architettura di rete allo stato iniziale: tutto il traffico tra Host viene permesso da ciascuno Switch di rete.

In particolare, la rete utilizzata viene composta da due Switch e sei Host di rete. Oltre a questi elementi, viene aggiunto un ulteriore Host (Sw1-Eth5 in figura 4.5) per collegare la rete Mininet a Internet ed essere perciò accessibile dall'esterno.

Quest'ultimo componente risulta fondamentale nel progetto in questione: per simulare configurazioni di rete realistiche e verificarne il funzionamento, si fa uso del comando Ping. Perciò l'host esterno (Sw1-Eth5) invia alcuni pacchetti all'Host selezionato dall'utente mediante la Web UI, per verificare il corretto funzionamento della configurazione di rete impostata.

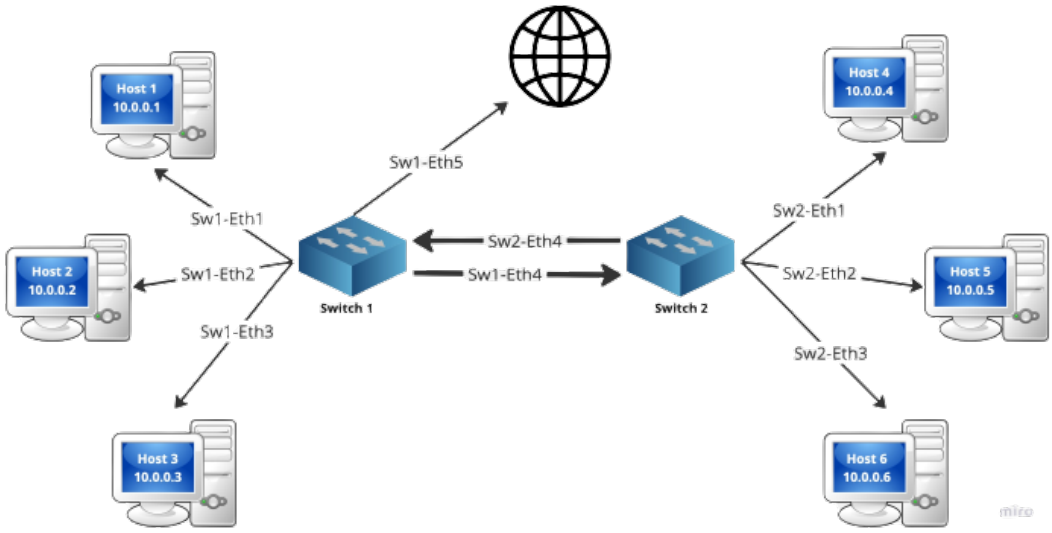


Figura 4.5: Stato iniziale dell'Architettura di rete.

4.5 Network Controller nel dettaglio

Come descritto alla sezione 4.3.1, mediante l'AAS Network Infrastructure possiamo eseguire tre differenti configurazioni dei Controller di rete: Simple Controller, Closed Controller e Selective Controller.

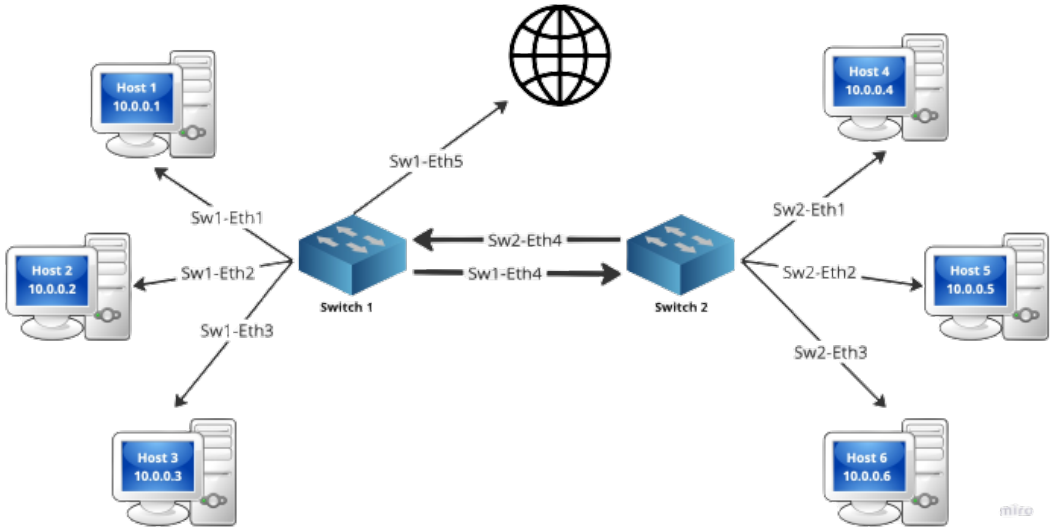


Figura 4.6: Configurazione di rete con Simple Controller.

Figura 4.6 mostra la configurazione di base della rete, per cui: non è stato ancora configurato il Controller su uno o più Switch o nella seconda ipotesi è stato avviato il Simple Controller, come descritto alla sezione 4.3.1.

Figura 4.7 mostra il caso intermedio: è stato avviato il Closed Controller. Switch 1 svolge quindi la funzione di firewall bloccando tutto il traffico di rete da e verso lo Switch 2. Viceversa lo Switch 2 non ha limiti e si ugualmente al primo caso descritto.

Infine, in figura 4.8 viene mostrato il caso più interessante: ciascuno Switch viene configurato per svolgere la funzione di Firewall Default Deny: di norma tutto il traffico viene bloccato.

Risulta perciò fondamentale con questa configurazione, inserire e/o modificare le regole contenute nel Controller, mediante i Submodel, le Proprietà e le Operazioni messe a disposizione dall'AAS Network Control Plane (descritto alla sezione 4.3.2).

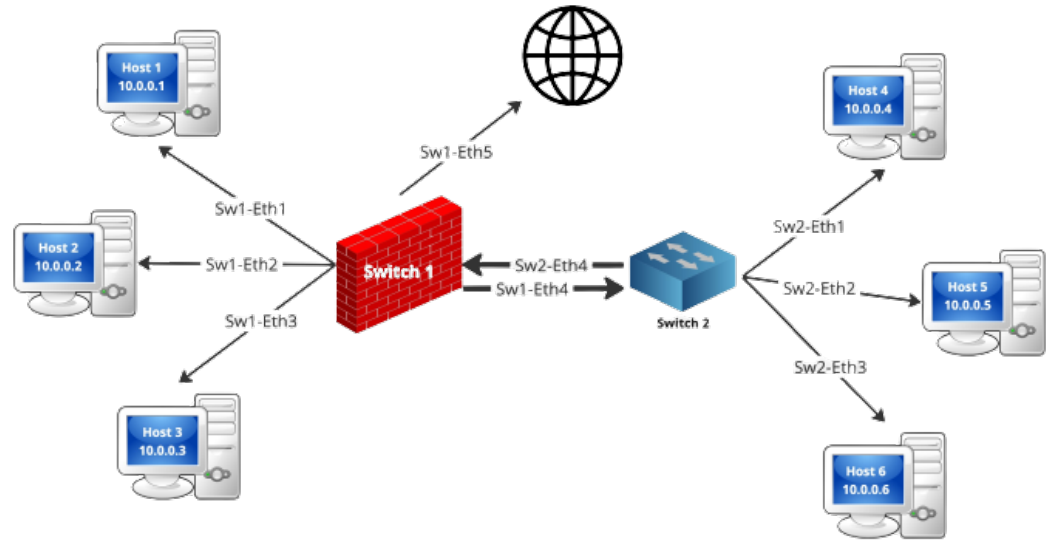


Figura 4.7: Stato iniziale dell'Architettura di rete.

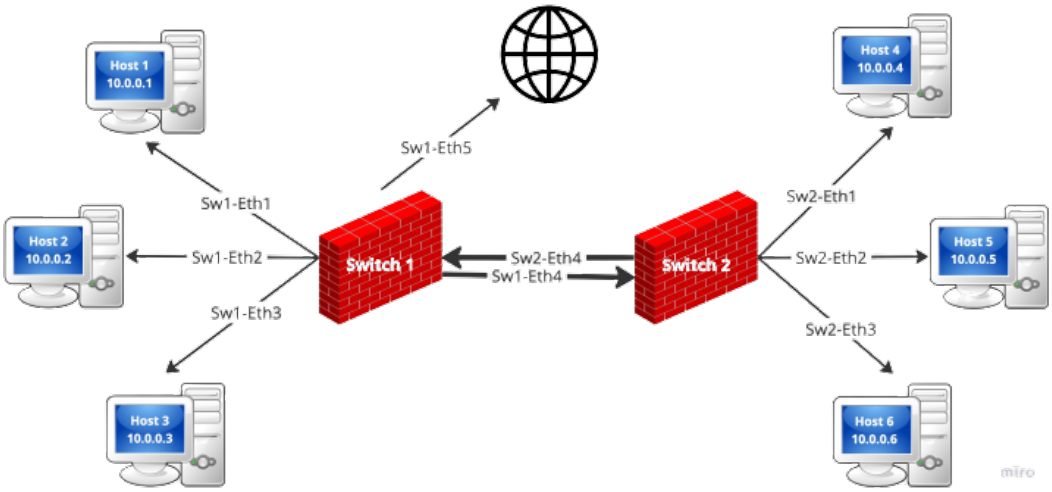


Figura 4.8: Stato iniziale dell'Architettura di rete.

4.6 OT: Machine AAS nel dettaglio

L'Asset Administration Shell in esame rappresenta il Digital Twin di un ipotetico Macchinario Industriale.

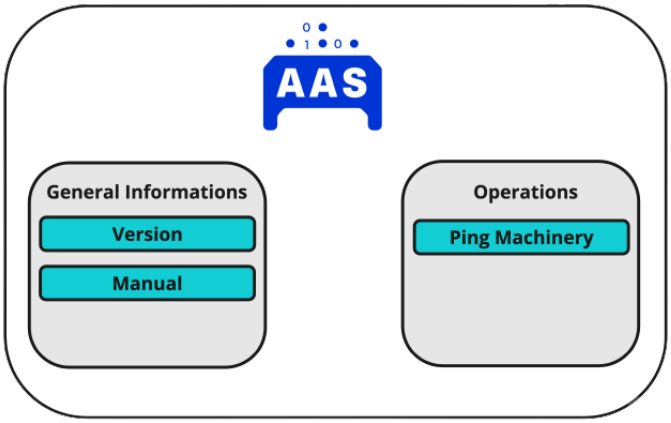


Figura 4.9: Asset Administration Shell di un ipotetico macchinario industriale.

Come visibile in figura 4.9, l'Administration Shell mette a disposizione due Submodels:

- General Informations: agisce come una repository di informazioni generiche ed esemplificate sul macchinario;
- Operations: nel caso in esame, si dispone della sola operazione "Ping Machinery". Questa permette di verificare in modo efficace la connettività verso altri nodi della rete in base alla configurazione di rete selezionata dal Network Infrastructure AAS (sezione 4.3.1) e configurata mediante il Network Control Plane AAS (sezione 4.3.2);

L'utilizzo di questo Asset Administration Shell si è reso necessario per esemplificare un caso d'uso estremamente basico del sistema in esame, volontariamente astratto. L'operazione "ping machinery" viene utilizzata per eseguire un'operazione di Ping sugli Host di rete, di cui ciascun macchinario si compone.

Come descritto nella sezione 4.4, la rete virtuale utilizzata si compone di due Switch di rete (configurabili mediante controller esterno) e sei Host di rete. Ciascun macchinario si compone di due Host, tra cui la comunicazione viene testata con la metodologia e le motivazioni prima descritte.

Capitolo 5

Risultato e Architettura finale

Nel capitolo seguente, verrà analizzato nel dettaglio il risultato finale ottenuto con lo sviluppo del progetto in questione. Diversi saranno i punti di vista affrontati:

- Partendo dalla dimostrazione dell'architettura di rete utilizzata;
- Mostrando il funzionamento finale dei componenti facenti parte di BaSyx utilizzati (Registry, Server e Web UI).

5.1 Architettura di rete

Come descritto nel precedente capitolo alla sezione 4.4, si è utilizzata un'architettura di rete composta da due Switch di rete (configurabili mediante Controller esterno) e sei Host di rete. In questa sezione, si andrà a utilizzare come strumento principale Mininet e Open vSwitch per la dimostrazione di quanto viene utilizzato.

L'avvio della rete, avviene mediante l'utilizzo del seguente comando su Shell:

```
sudo python 2switch_6host.py
```

Verrà avviato il seguente programma in linguaggio Python (qui semplificato in alcune sue parti):

```
def myNetwork():
    net = Mininet(topo=None, build=False, link=TCLink)
    sw1 = net.addSwitch('sw1')
    sw2 = net.addSwitch('sw2')

    # Adding hosts
    h1 = net.addHost('host1', ip='10.0.0.1')
    h2 = net.addHost('host2', ip='10.0.0.2')
    h3 = net.addHost('host3', ip='10.0.0.3')
    h4 = net.addHost('host4', ip='10.0.0.4')
    h5 = net.addHost('host5', ip='10.0.0.5')
    h6 = net.addHost('host6', ip='10.0.0.6')

    # Connecting hosts to switches and switch to switch
    net.addLink(h1, sw1)
    net.addLink(h2, sw1)
    net.addLink(h3, sw1)

    net.addLink(h4, sw2)
    net.addLink(h5, sw2)
    net.addLink(h6, sw2)

    net.addLink(sw1, sw2)

    h1.setMAC("00:00:00:00:00:01", h1.name + "-eth0")
    h2.setMAC("00:00:00:00:00:02", h2.name + "-eth0")
    h3.setMAC("00:00:00:00:00:03", h3.name + "-eth0")
    h4.setMAC("00:00:00:00:00:04", h4.name + "-eth0")
    h5.setMAC("00:00:00:00:00:05", h5.name + "-eth0")
    h6.setMAC("00:00:00:00:00:06", h6.name + "-eth0")

    # Connecting switches to external controller
    net.addNAT().configDefault()
    net.start()
    sw1.cmd('ovs-vsctl set-controller ' +  sw1.name + '
            tcp:127.0.0.1:6633')
    sw2.cmd('ovs-vsctl set-controller ' +  sw2.name + '
```

```
        tcp:127.0.0.1:6653')
CLI(net)
```

Il programma riportato, eseguirà le seguenti operazioni necessarie al corretto avvio di un rete virtuale Mininet:

- Creazione di una rete Mininet vuota, a cui aggiungere sei Host di rete (ciascuno con il proprio IP e indirizzo MAC);
- Aggiunta dei collegamenti tra ciascun Host dello stesso Switch e tra i due Switch;
- Start della rete e aggiunta dell'Host NAT (necessario al collegamento della rete verso l'esterno).

La configurazione finale ottenuta, viene mostrata in figura 5.1 e nel risultato dell'operazione *mininet dump*, seguente.

```
<Host host1: host1-eth0:10.0.0.1 pid=5074>
<Host host2: host2-eth0:10.0.0.2 pid=5079>
<Host host3: host3-eth0:10.0.0.3 pid=5081>
<Host host4: host4-eth0:10.0.0.4 pid=5083>
<Host host5: host5-eth0:10.0.0.5 pid=5085>
<Host host6: host6-eth0:10.0.0.6 pid=5087>
<NAT nat0: nat0-eth0:10.0.0.7 pid=5204>
<OVSSwitch sw1: lo:127.0.0.1:6633>
<OVSSwitch sw2: lo:127.0.0.1:6653>
```

Procediamo con la verifica di quanto creato, mediante il comando:

```
sudo ovs-vsctl show

Bridge "sw1"
    Controller "tcp:127.0.0.1:6633"
    fail_mode: secure
    Port "sw1"
        Interface "sw1"
            type: internal
    Port "sw1-eth1"
```

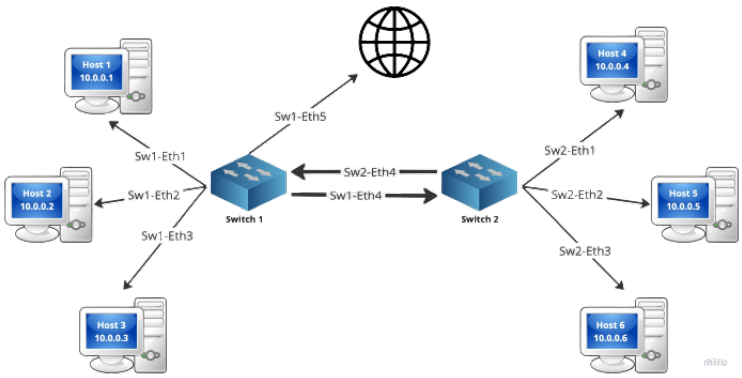


Figura 5.1: Schematizzazione della rete virtuale utilizzata nel progetto.

```
        Interface "sw1-eth1"
    Port "sw1-eth2"
        Interface "sw1-eth2"
    Port "sw1-eth5"
        Interface "sw1-eth5"
    Port "sw1-eth3"
        Interface "sw1-eth3"
    Port "sw1-eth4"
        Interface "sw1-eth4"
Bridge "sw2"
    Controller "tcp:127.0.0.1:6653"
    fail_mode: secure
    Port "sw2-eth2"
        Interface "sw2-eth2"
    Port "sw2-eth1"
        Interface "sw2-eth1"
    Port "sw2"
        Interface "sw2"
            type: internal
    Port "sw2-eth4"
        Interface "sw2-eth4"
    Port "sw2-eth3"
        Interface "sw2-eth3"
```

Come previsto, notiamo la presenza di due Switch: Sw1 e Sw2. Ciascuno Switch dispone di quattro connessioni: tre Host e la connessione verso l'altro Switch. Inoltre Sw1 dispone di una quinta connessione: il NAT (precedentemente aggiunto mediante il file di configurazione) visibile come interfaccia "nat0-eth0". Questo risulterà **l'unico punto di accesso dall'esterno** verso la rete Mininet.

Se andremo perciò a eseguire il seguente comando, per effettuare ping su Host4, dall'esterno di mininet:

```
ping 10.0.0.4
```

Noteremo monitorando l'interfaccia "nat0-eth0", mediante wireshark, il seguente transito di pacchetti:

```
Source      Destination
10.0.0.7    10.0.0.4   ICMP  98  Echo (ping) request
10.0.0.4    10.0.0.7   ICMP  98  Echo (ping) reply
10.0.0.7    10.0.0.4   ICMP  98  Echo (ping) request
10.0.0.4    10.0.0.7   ICMP  98  Echo (ping) reply
```

Possiamo notare la Sorgente del pacchetto essere sempre il NAT (10.0.0.7) e la destinazione il corretto Host4 (10.0.0.4), nonostante il Ping provenga da un nodo esterno. Effettuando la medesima operazione dall'interno della Shell mininet, ovviamente il Nat non verrà utilizzato, ma il traffico sarà solamente "interno".

5.2 BaSyx Middleware

Come descritto nella sezione 4.4, sono stati utilizzati alcuni componenti forniti d BaSyx necessari al corretto funzionamento dell'infrastruttura: il Registry, il Server e la Web UI. Ciascuno di essi viene fornito mediante Docker container e avviato mediante il file Docker compose come descritto nella sezione 4.1.

5.2.1 BaSyx Registry

Il Registry risulta fondamentale, in quanto come descritto nella sezione 4.2 svolge il fondamentale ruolo di Service Discovery. In questa sezione verrà dimostrato il diagramma di sequenza mostrato in figura 4.2.

Come precedentemente mostrato, all'avvio della Web UI, questa effettua una richiesta di Service Discovery al Registry, come visibile nei seguenti pacchetti (ottenuti mediante Wireshark):

```
Source      Destination
172.25.0.1  172.25.0.2  HTTP  GET /registry/api/v1/registry
172.25.0.2  172.25.0.1  HTTP  HTTP/1.1 200 (application/json)
```

La richiesta verso l'API fornita dal Registry fornisce in Json gli AAS disponibili e registrati in rete, mostrati mediante la Web UI. Il secondo step, risulta nella richiesta al Server delle informazioni sulla Shell voluta, visibile dai seguenti pacchetti:

```
Source      Destination
172.25.0.1  172.25.0.3  HTTP  GET /aasServer/shells (HTTP/1.1)
172.25.0.3  172.25.0.1  HTTP  HTTP/1.1 200   (application/json)
```

Successivamente la web UI (porta 3000 su localhost), richiede direttamente all'AAS voluto (nel diagramma in figura 4.2 veniva mostrato il Network Infrastructure AAS), in questo caso sulla porta 6001 di localhost.

5.2.2 BaSyx Web UI

Come descritto nelle sezioni precedenti, viene utilizzata la BaSyx Web UI come mezzo principale per esporre i risultati ottenuti dal progetto in questione. Viene anch'essa fornita mediante Docker Container, conseguentemente avviata mediante l'utilizzo del Docker Compose (descritto alla sezione 4.1.

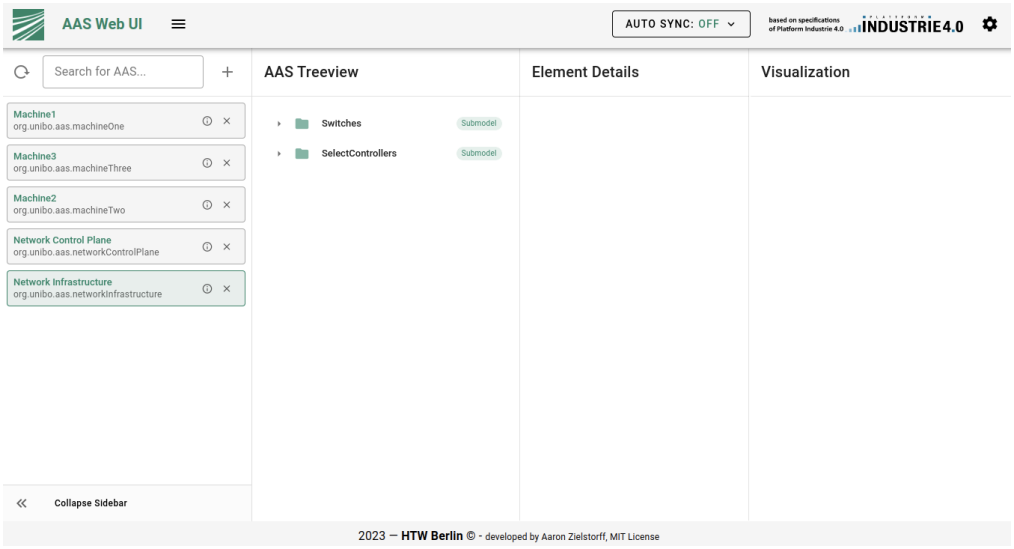


Figura 5.2: WebUI allo stato iniziale, raffigurante gli AAS disponibili dopo la richiesta di Discovery al Registry.

Allo stato iniziale, dopo l'inserimento dell'indirizzo del Registry e del Server, otteniamo la seguente prospettiva. Nelle colonne successive alla prima, verranno invece visualizzati i Submodels, facenti parte dell'AAS selezionato. Per le descrizione dettagliate di ciascun AAS si rimanda al capitolo 4.

5.3 Implementazione nel dettaglio

In questa sezione si analizzerà nel dettaglio aspetti peculiari del progetto sviluppato. Come precedentemente descritto alcuni componenti facenti parte del middleware BaSyx vengono forniti e avviati mediante Docker Container. Lo sviluppo dei singoli Asset Administration Shell e relativi Submodels è stato invece realizzato ad-hoc per il caso in esame, sfruttando anche in questo caso la documentazione e le API messe a disposizione da BaSyx.

Innanzitutto per la realizzazione di ciascun AAS, sfruttando principi di buona programmazione (ad esempio DRY - Don't Repeat Yourself), si è fatto uso di alcune classi astratte eliminando ripetizione di codice e difficoltà nel documentarlo correttamente. Il risultato ottenuto, per la creazione del singolo AAS è il seguente:

```
ShellInstance networkInfrastructure = new NetworkInfrastructure(
```

```
    6001,
    "Network Infrastructure",
    "org.unibo.aas.networkInfrastructure",
    AssetKind.INSTANCE);
```

In particolare si riporta la creazione dell'Asset Administration Shell Network Infrastructure. Ciascun AAS richiede la porta s u cui istanziare il proprio Web Server, l'Identificativo univoco e la tipologia (Instance o Kind). Successivamente si procede alla registrazione dei AAS e rispettivi Submodels sul Registry, utilizzando i rispettivi Descriptor (si rimanda alla sezione 4.2):

```
AASRegistryProxy registryProxy = new AASRegistryProxy(REGISTRYPATH);

registerDescriptors(Port, Submodels, AssetAdministrationShell){
    String IP = generateIP(Port)
    AASDescriptor aasDescriptor = new AASDescriptor(shell, IP);

    for (Submodel sm : submodels) {
        aasDescriptor.addSubmodelDescriptor(
            new SubmodelDescriptor(sm, IP + "/submodels/" + sm.idShort));
    }

    registryProxy.register(aasDescriptor);
}
```

Il codice riportato esegue le seguenti operazioni: dopo essersi collegato al Registry (indirizzo Registry Path sulla rete locale), procede con la creazione dei Descriptors necessari per l'AAS fornito e i relativi Submodels, assegnando a ciascuno di essi un indirizzo IP specifico e un identificativo univoco. I descriptors vengono infine registrati sul Registry.

Questa operazione risulta fondamentale per il corretto funzionamento del Registry che come espresso nelle sezioni precedenti, svolge la funzione principale di Service Discovery.

Un ulteriore step per la creazione degli AAS nella loro completezza è l'avvio del rispetto Web Server. Eseguito nella seguente modalità:

```
public ModelProvider(servletPort,
                     ACCESS_CONTROL_ALLOW_ORIGIN,
```

```
                    CONTEXT_PATH,
                    AssetAdministrationShell shell,
                    List<Submodel> submodels) {
    modelProvider = new MultiSubmodelProvider();
    modelProvider
        .setAssetAdministrationShell(new AASModelProvider(shell));

    for(Submodel s : submodels) {
         this.modelProvider.addSubmodel(new SubmodelProvider(s));
    }

    this.modelServlet = new VABHTTPInterface<IModelProvider>(modelProvider);
    this.context = configureContext(
        servletPort,
        ACCESS_CONTROL_ALLOW_ORIGIN,
        CONTEXT_PATH);
    this.context.addServletMapping("/*", modelServlet);
}

public void startLocalHTTPServer(String idShort) {
    BaSyxHTTPServer server = new BaSyxHTTPServer(context);
    server.start();
}
```

Nel codice riportato, si procede inizialmente con la creazione e configurazione del ModelProvider (sezione 4.2), necessario a interconnettere ciascun AAS al resto dell'infrastruttura BaSyx mediante il VAB (Virtual Automation Bus, sezione 3.2). Viene configurato il web server prima dell'avvio con le seguenti modalità:

- CORS (Cross Origin Resource Sharing) viene avviato mediante le direttive precedentemente configurate;
- Viene fornita la porta indicata in fase di creazione dell'AAS;
- Viene fornito l'indirizzo base su cui mappare i Submodels e le Proprietà/Operazioni che li compongono, in questo caso "/".

Infine come ultima operazione viene avviato il server HTTP, solo dopo l'avvio del Registry e del Server e dopo la creazione dell'AAS stesso.

La creazione dei Submodels segue le stesse modalità implementative degli Asset Administration Shell. Ciascun submodel dispone inoltre di alcune classi maggiormente specifiche ad esempio per interfacciarsi con i comandi su Shell (utilizzati nell'avvio dei Controller o nell'utilizzo del comando Ping) o per effettuare richieste alle API messe a disposizione dall'infrastruttura BaSyx.

5.4 Avvio del sistema

Il sistema ultimato viene quindi composto da:

- Componenti forniti mediante Container (e perciò avviati mediante Docker compose): BaSyx Registry, BaSyx Server, BaSyx Web UI;
- Rete virtuale, creata con l'ausilio di Mininet mediante script Python (sezione 5.1);
- Programma per la creazione degli AAS e di tutto ciò che ne concerne (sezione 5.3).

L'avvio si comporrà perciò:

- esecuzione dello script Mininet mediante Shell e mediante il comando riportato nella relativa sezione;
- avvio del docker compose riportato alla sezione 4.1, anch'esso mediante comando Shell;
- avvio dell'applicativo sviluppato per il progetto e il caso d'uso in esame.

Successivamente sarà possibile accedere ai vari componenti, ai seguenti indirizzi:

- BaSyx Registry: `http://localhost:8082/registry`
- BaSyx Server: `http://localhost:8081/aasServer`
- BaSyx Web UI: `http://localhost:3000/`
- Ciascun Web Server relativo agli AAS creati e utilizzati, agli indirizzi `http://localhost:6000/aas` e successive numerazioni.

5.5 Caso d'uso

Il progetto in esame va a supportare uno specifico caso d'uso (figura 5.3): si ipotizza la presenza di tre macchinari industriali, interconnessi mediante due Switch di rete (Open vSwitch virtuali) configurabili mediante l'utilizzo di un Controller esterno.

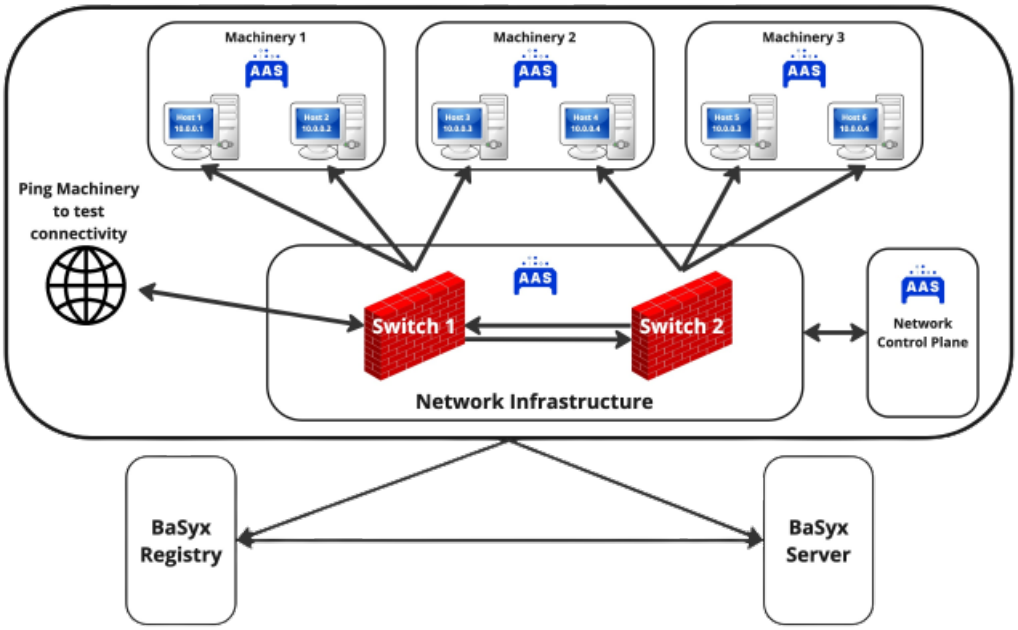


Figura 5.3: Caso d'uso supportato e descritto dal progetto in esame.

Ciascun Macchinario si compone di due interfacce verso la rete industriale (virtualizzata mediante Mininet), visibili come Host di rete (anch'essi virtuali) connessi ai rispettivi Switch.

Ciascuno Switch viene configurato mediante apposito Asset Administration Shell: Network Infrastructure. Mediante il Network Control Plane è invece possibile configurare le regole di inoltro del Controller selezionato.

L'obiettivo nell'implementazione del caso d'uso descritto è quello di consentire differenti configurazioni della rete industriale ottenuta, verificandone il funzionamento anche con l'utilizzo del comando Ping, dall'esterno dell'infrastruttura, così da simulare casi di utilizzo maggiormente realistici.

Il test delle configurazioni di rete disponibili all'utente finale avviene mediante Ping dei Macchinari industriali a disposizione (e relativamente agli host di cui si compongono). Come descritto nella sezione 5.1, il Ping avviene sempre tramite Host di rete esterni attraverso l'interfaccia NAT (nat0-eth0) messa a disposizione dallo Switch 1 (visibile in figura 5.3).

Capitolo 6

Conclusioni

L'obiettivo del progetto in questione era quello di fornire i mezzi software necessari all'Automazione di Rete, all'interno di un ambiente industriale 4.0. Lo sviluppo si è basato su alcuni concetti primari e fondamentali:

- Digital Twin come copia digitale di ciascun elemento industriale, fisico o logico;
- Asset Administration Shell come implementazione del concetto di Digital Twin, a oggi supportata e documentata da numerosi progetti in questo ambito.

Il risultato ottenuto favorisce l'utente nella personalizzazione e automazione del comportamento di una Rete industriale mediante la programmazione degli Switch presenti, finalizzato all'ottenimento di uno specifico comportamento.

Il comportamento viene basato su tre tipologie di Controller a disposizione: Simple Switch Controller, Closed Controller e Firewall Controller. Ciascuno di essi è stato accuratamente documentato nei precedenti capitoli di questo elaborato ed è a disposizione dell'utente, mediante un'interfaccia web completa e ben comprensibile.

Nel caso in esame, a scopo dimostrativo la rete viene stanziata mediante l'utilizzo dell'ambiente virtuale Mininet. Fondamentale è inoltre risultato l'utilizzo dell'infrastruttura BaSyx per la fornitura dell'API necessaria allo sviluppo degli AAS e relativi Submodels e per i componenti forniti mediante Docker Container.

Il risultato ottenuto raggiunge gli obiettivi preposti dal progetto stesso, fornendo un punto di partenza volutamente astratto nell'utilizzo del concetto

di Digital Twin e Asset Administration Shell allo scopo di automatizzare la programmazione di Rete all'interno di un ambiente Industriale 4.0.

Capitolo 7

Sviluppi futuri

Il progetto è stato volutamente realizzato in modo astratto, senza l'appoggio di casi d'utilizzo concreti e mediante tecnologie (quali ad esempio Mininet) che permettessero l'astrazione degli elementi utilizzati, dalla rete (host, switch, controller) ai macchinari utilizzati, alle metodologie con le quali verificarne il comportamento finale.

Possibili sviluppi futuri saranno perciò affrontati sotto differenti punti di vista. Il materiale a disposizione fornisce i mezzi necessari all'implementazione di casi d'uso maggiormente concreti e sempre meno astratti, seguendo le linee guida impostate.

Si potrebbe adottare l'utilizzo di Switch ed elementi fisici tra loro interconnessi e distribuiti nello spazio, esemplificando maggiormente un caso d'uso realistico e concreto. Inoltre, con il supporto delle aziende manifatturiere, si potrebbero ricreare dei casi d'uso totalmente realistici, tali da fornire indicazioni concrete sull'usabilità del sistema e delle tecnologie oggi a disposizione (Digital Twin, AAS, Switch SDN) all'interno di un ambiente industriale realistico e conseguentemente più complesso.

Un ulteriore punto di vista fondamentale riguarderà l'utilizzo e l'introduzione della tecnologie 5G nel progetto in esame (che si pone in questo ambito come elemento primordiale). L'introduzione di questa nuova tecnologia, risulterà nel principale elemento abilitante alla distribuzione del sistema in esame, ad esempio permettendo a un produttore il monitoraggio da remoto degli elementi industriali anche presso clienti terzi. Lo sviluppo di questa soluzione richiederà l'aggiunta di uno o più Asset Administration Shell, tali da permettere una modellazione accurata e sufficientemente semplificata della rete 5G

(descritta nel documento alla sezione 2.4).

Ciascun macchinario verrà rappresentato come un 5G User Equipment (UE), che a sua volta si interfaccerà con il 5G Core Network (CN). Quest'ultimo racchiude i differenti livelli e componenti che costituiscono la rete 5G, e ne dovrà permettere la configurazione ad-hoc per le casistiche affrontate dal settore industriale.

Infine, numerosi sono gli aspetti legati alla sicurezza degli apparati industriali 4.0 (sezione 2.3) soprattutto con l'avvento dei Digital Twin e degli Asset Administration Shell, che consentono e favoriscono la connessione e il raggiungimento delle linee di produzione anche dall'esterno (rete IT). Questo espone notevoli rischi data la non adeguatezza delle reti OT a oggi presenti, le quali non erano state inizialmente pensate per gestire connessioni e aspetti di sicurezza esterni, ma rappresentavano delle reti a loro stanti, non connesse e non raggiungibili dall'esterno della rete stessa. Numerose sono le iniziative presenti per la gestione degli aspetti indicati, ma a oggi non vi è uno standard unico con cui verificare il grado di adeguatezza di una determinata rete OT 4.0 rispetto ad aspetti di sicurezza informatica.

Bibliografia

[1] 5G-ACIA. Using digital twins to integrate 5g into production networks. In *Using Digital Twins to Integrate 5G into Production Networks*, 2021.

[2] André Bröring, Marco Ehrlich, Henning Trsek, and Lukaz Wisniewski. Secure usage of asset administration shells : an overview and analysis of best practises, 2021.

[3] A. Di Pinto Y. Dragoni and A. Carcano. *"Triton: The first ics cyber attack on safety instrument systems."*. Proc. Black Hat USA, 2018.

[4] URL: https: //github.com/aligungr/UERANSIM.

[5] Phani Kumar Garimella. It-ot integration challenges in utilities. In *2018 IEEE 3rd International Conference on Computing, Communication and Security (ICCCS)*, pages 199–204, 2018.

[6] Steven Hagner. Optimizing transmission asset health with it/ot integration. In *2016 Saudi Arabia Smart Grid (SASG)*, pages 1–6, 2016.

[7] Ali M. Hosseini, Thilo Sauter, and Wolfgang Kastner. Safety and security requirements in aas integration: Use case demonstration. In *2023 IEEE 19th International Conference on Factory Communication Systems (WFCS)*, pages 1–8, 2023.

[8] L. A. Macaulay. *Requirements engineering*. Business Media, 2012.

[9] Nick McKeown, Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry Peterson, Jennifer Rexford, Scott Shenker, and Jonathan Turner. Openflow: enabling innovation in campus networks. *ACM SIGCOMM computer communication review*, 38(2):69–74, 2008.

[10] https://github.com/mininet/mininet/wiki/Introduction-to-Mininet.

[11] URL: https:// open5gs.org/.

[12] Magnus Redeker, Jan Nicolas Weskamp, Bastian Rössl, and Florian Pethig. Towards a digital twin platform for industrie 4.0. In *2021 4th IEEE International Conference on Industrial Cyber-Physical Systems (ICPS)*, pages 39–46, 2021.

[13] Daniele Rossi, Giacomo Tontini, Davide Borsatti, and Franco Callegati. Integration of 5g connectivity with the asset administration shell in industry 4.0. In *2022 13th International Conference on Network of the Future (NoF)*, pages 1–3, 2022.

[14] https://osrg.github.io/ryu-book/en/html/rest_firewall.html.

[15] https://ryu.readthedocs.io/en/latest/app/ofctl_rest.html.

[16] Lucas Sakurada, Paulo Leitao, and Fernando De la Prieta. Towards the digitization using asset administration shells. In *IECON 2021 – 47th Annual Conference of the IEEE Industrial Electronics Society*, pages 1–6, 2021.

[17] Stephan Wein, Christian Fimmers, Simon Storms, Christian Brecher, Marlene Gebhard, Michael Schluse, and Jürgen Roßmann. Embedding active asset administration shells in the internet of things using the smart systems service infrastructure. In *2020 IEEE 18th International Conference on Industrial Informatics (INDIN)*, volume 1, pages 23–28, 2020.