Consiglio Nazionale delle Ricerche
Istituto di Calcolo e Reti ad Alte Prestazioni

Progetto, realizzazione e testing di un sistema per la gestione energetica efficiente in edifici intelligenti

Emilio Greco, Antonio Francesco Gentile

RT- ICAR-CS-21-01 **Gennaio 2021**

Consiglio Nazionale delle Ricerche, Istituto di Calcolo e Reti ad Alte Prestazioni (ICAR)
– Sede di Cosenza, Via P. Bucci 8-9C, 87036 Rende, Italy, URL: *www.icar.cnr.it*
– Sezione di Napoli, Via P. Castellino 111, 80131 Napoli, URL: *www.icar.cnr.it*
– Sezione di Palermo, Via Ugo La Malfa, 153, 90146 Palermo, URL: *www.icar.cnr.it*

Indice generale

Introduzione

I problemi di schedulazione che generalmente vengono trattati in letteratura sono spesso statici, ovvero le attività sono note in anticipo e i vincoli sono fissi, tuttavia i risultati prodotti anche se soddisfacenti non tengono conto del fatto che ogni programma di schedulazione nella vita reale è soggetto a eventi inaspettati: il numero di attività ed i vincoli che descrivono il problema potrebbero cambiare anche in maniera sostanziale. In questi casi, è necessaria una nuova soluzione in un tempo preferibilmente breve. Inoltre, questa nuova soluzione deve essere non troppo lontana da quella precedente in modo da non stravolgere l'evoluzione del sistema.

Approcci alla soluzione di questi tipi di problemi li troviamo in letteratura annoverati sotto diversi nomi: Job shop scheduling[1], Resource Constrained Project Scheduling Problem (RCPSP)[2], Dynamic Scheduling[3].

Questi problemi di ottimizzazione richiedono la programmazione di una serie di attività tenendo conto dei vincoli temporali e delle risorse adoperate in un certo istante. Questo insieme di problemi ha una elevata complessità computazionale ed appartengono alla classe di problemi chiamati "NP-Hard[4]" [Blazewicz et al., 1983]. L'utilizzo di algoritmi euristici [5] convenzionali per risolvere questa tipologia di problemi richiede elevati tempi di esecuzione che non consentirebbero l'applicabilità in un sistema real-time. L'approccio che si vuole utilizzare quindi è quello di applicare gli algoritmi del Reinforcement Learning (RL) per risolvere questa classe di problemi tenendo conto dei tempi di esecuzione e delle risorse impiegate.

Nella prima parte dell'elaborato viene introdotto il concetto di *"apprendimento per rinforzo"* come metodologia per la risoluzione di problemi complessi. Dopo una breve descrizione degli algoritmi classici del RL e delle loro caratteristiche viene affrontata la parte progettuale del sistema.

Nella seconda parte dell'elaborato, vengono messe in evidenza le caratteristiche tecniche e strutturali del problema e le scelte progettuali adottate per risolverlo. Particolare enfasi viene data alle scelte che hanno portato alla trasformazione da un algoritmo classico di schedulazione ad un algoritmo di schedulazione dinamica, in grado di sfruttare l'auto-apprendimento come elemento di forza per adattarsi all'ambiente. Vedremo come l'algoritmo realizzato sarà in grado di rispondere a più richieste provenienti da utilizzatori differenti e quindi da ambienti diversi, ma anche come la variabilità dei parametri che definiscono lo stesso problema (tempo massimo di differimento, tempo di esecuzione, potenza assorbita, etc.) ha portato all'implementazione di un modello dinamico che di volta in volta si adatta per risolvere un problema specifico. La necessità di realizzare un algoritmo real-time e stand-alone ha spinto la progettazione verso un sistema che ottimizza il più possibile le risorse computazionali e di memoria consentendone l'istallazione anche su dispositivi a basse prestazioni.

[1] Scheduling su macchine parallele scorrelate, dove n lavori devono essere processati da m macchine diverse disposte in parallelo, dove l'obiettivo è assegnare i lavori alle macchine in modo tale da minimizzare il tempo totale di completamento.

[2] Consiste nel ricercare uno schedule ottimo per le attività di un progetto che soddisfi non solo i vincoli di precedenza, ma che provveda anche ad una corretta allocazione delle risorse disponibili quali: manodopera, materie prime, etc.

[3] La pianificazione dinamica è una tecnica utilizzata dai moderni calcolatori, in cui le istruzioni non vengono eseguite in base all'ordine di arrivo, ma piuttosto alla disponibilità delle risorse computazionali disponibili.

[4] Nella teoria della complessità computazionale, descrivono quei problemi non deterministici in cui non esiste un algoritmo che è in grado di risolvere velocemente (in tempo polinomiale) il problema.

[5] l'algoritmo euristico riesce a ricavare una soluzione approssimativamente vicina a quella ottima in problemi altrimenti irrisolvibili con la matematica classica.

La maggior parte dei dispositivi IoT che troviamo in commercio usano una tecnologia on-cloud per elaborare i dati acquisiti. Tecnica che da una parte consente di abbattere i costi di produzione e dall'altra la fidelizzazione gli utilizzatori. In questo progetto vogliamo svincolarci da tecnologie proprietarie e realizzare un prodotto plug and play, installabile su dispositivi a basso costo ed open source. La scelta progettuale che ha consentito il raggiungimento di questo obbiettivo è stata l'accurata definizione di una funzione filtro in grado di discriminare le azioni ammissibili dalle azioni non ammissibili a partire da un dato stato del sistema. Quest'approccio ha consentito una riduzione sostanziale dei tempi di elaborazione dell'algoritmo, sia perché ha ridotto il numero delle scelte che l'algoritmo deve operare in ogni fase di esecuzione, e sia perché si è ottenuto una riduzione dei dati da archiviare a seguito della riduzione degli stati da visitare. Questo ha consentito l'uso di strutture di archiviazione dati come gli alberi al posto delle matrici con sostanziali benefici di ingombro di memoria.

Nella terza parte dell'elaborato vengono fornite le nozioni di base per l'utilizzo dell'ambiente di sviluppo e di addestramento[6]. Viene descritto in particolare l'ambiente Openai Gym, uno dei più diffusi ambienti di addestramento e vengono presentati i primi risultati prodotti in ambiente simulato. Risultati ottenuti a partire da due algoritmi classici del RL : SARSA e Q-Learnig. I test sono stati effettuati sulla base di tre casi studio, utilizzando dati assegnati staticamente. Gli stessi test poi sono stati riproposti sul prototipo realizzato, utilizzando questa volta parametri reali ricavati dai sensori a bordo del dimostratore.

Nell'ultima parte di questo lavoro viene presa in esame la progettazione e la realizzazione di un sistema IoT capace di rispondere ai requisiti progettuali richiesti e la realizzazione di un dimostratore. Vengono descritti i protocolli di comunicazione adoperati, gli hardware e i software scelti per realizzare il sistema, l'interfaccia grafica e l'elaborazione dei dati.

I risultati prodotti in questa attività multidisciplinare, sono stati:

- la realizzazione di una rete WLAN di supporto alla comunicazione tra i dispositivi IoT adoperati;
- la realizzazione di un sistema di messaggistica MQTT[7];
- un sistema di misura, elaborazione e rappresentazione dei consumi elettrici;
- un sistema di misura, elaborazione e presentazione dati per la produzione fotovoltaica;
- la realizzazione di un dimostratore di impianto fotovoltaico;
- la realizzazione di un dispositivo di controllo di rete IoT. Per rendere il sistema scalabile si è deciso di realizzare una unità di controllo dei dispositivi (hub IoT), replicabile su più nuclei abitativi ed un'unità centrale che elabora le richieste di schedulazione.;
- la realizzazione di una interfaccia utente per il monitoraggio ed il controllo del sistema.

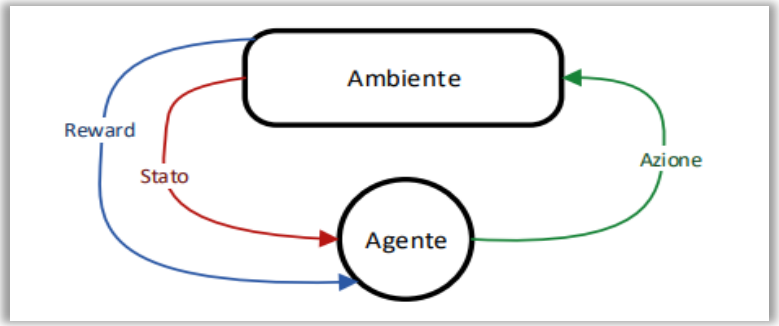
[6] Il processo attraverso cui i software di intelligenza artificiale sviluppano un certo grado di intelligenza viene chiamato addestramento.

[7] è un protocollo di trasmissione dati client-server leggero, semplice e realizzato per essere implementato rapidamente anche in progetti già finiti. Queste caratteristiche lo rendono ideale per essere usato dove è maggiormente richiesto un certo risparmio sia nel codice eseguito sui dispositivi sia nella banda di comunicazione per interscambio dati in condizioni critiche come la comunicazione in tempo reale.

Il Reinforcement Learning

L'obiettivo della ricerca in machine learning è fornire ai calcolatori l'abilità di apprendere automaticamente un comportamento sulla base di informazioni ed esempi, senza essere programmati esplicitamente per svolgere un determinato compito. In quest'ambito i metodi di Reinforcement Learning cercano di determinare come un agente razionale debba scegliere determinate azioni da eseguire a partire dalla conoscenza dello stato corrente del sistema, perseguendo l'obiettivo di massimizzare una sorta di ricompensa totale per raggiungere uno stato terminale a lui noto. La ricompensa totale è determinata sulla base di una sequenza di reward (ricompense) che l'agente ottiene nell'eseguire le singole azioni che portano al raggiungimento dello stato terminale. Il meccanismo fondamentale di tutti gli algoritmi noti del Machine Learning è il miglioramento automatico del comportamento mediante l'esperienza accumulata in fase di addestramento. Ciò vuol dire che un programma di RL dovrà scoprire, mediante ripetute prove, quali azioni permetteranno di ottenere la ricompensa maggiore.

L'idea di base che sta nella formulazione di un problema di RL è catturare gli aspetti più importanti di un problema reale, facendo interagire un agente, che è in grado di apprendere il problema in modo da fargli raggiungere l'obiettivo prefissato. Per ottenere ciò è opportuno che l'agente sia in grado di operare con l'ambiente e osservarne lo stato ad ogni istante[8]. L'agente dovrà essere in grado di eseguire azioni le quali avranno effetto sull'ambiente modificandone lo stato. Inoltre all'agente dovrà essere assegnato un obiettivo da perseguire o più obiettivi se si vuole che il sistema evolva attraversando alcuni stati desiderati e ne eviti altri indesiderati. La formulazione del problema di learning si basa dunque su questi tre aspetti: osservazione, azione e obiettivo.



La continua iterazione tra agente ed ambiente permette una continua acquisizione di conoscenza da parte dell'agente. Conoscenza che deve sfruttare in modo da massimizzare la ricompensa finale. La conoscenza acquisita non deve essere però preponderante sulle scelte dell'agente che allo stesso tempo deve esplorare nuove soluzioni in modo da scegliere azioni migliori nelle esecuzioni future. Bisogna considerare che l'agente interagisce con un ambiente stocastico, questo comporta che ogni azione dovrà essere provata più volte per ottenere una stima reale della ricompensa prevista.

[8] C'è una netta analogia con i sistemi di controllo retroazionati. Occorre individuare delle grandezze che governano il sistema, misurale, confrontarle con un valore obbiettivo e poi occorre intervenire con una attuazione per far evolvere il sistema.

Algoritmo Q_Learning

Uno degli algoritmi più importanti relativi alla risoluzione di problemi di apprendimento è il Q-Learning e si basa sull'utilizzo della funzione Q(s,a). Apprendere la funzione Q equivale ad apprendere la politica ottima. Per apprendere Q, occorre un modo per stimare i valori di addestramento per Q a partire solo dalla sequenza di ricompense immediate r in un lasso di tempo. La funzione Q viene aggiornata utilizzando la seguente formula:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)]$$

L'implementazione degli algoritmi appena trattati avviene attraverso l'utilizzo di una matrice avente un numero di righe pari al numero degli stati e un numero di colonne pari al numero delle possibili azioni. In ogni cella viene memorizzata una stima della funzione stato-azione Q(s,a).

Durante la risoluzione di un problema di Reinforcement Learning bisogna essere in grado di bilanciare la fase di esplorazione e di sfruttamento. L'esplorazione (exploration) dello spazio delle azioni permette di scoprire azioni che portano a ricompense migliori. Un agente che esplora solamente difficilmente riuscirà a convergere ad una policy ottima. Le azioni migliori vengono scelte ripetutamente (sfruttamento) perché garantiscono una ricompensa massima (reward). Il bilanciamento può essere ottenuto attraverso la tecnica ε-greedy. Questo metodo permette di scegliere l'opzione greedy per la maggior parte delle volte al fine di sfruttare l'informazione immagazzinata fino a quel momento e massimizzare la ricompensa totale e con probabilità ε di scegliere una azione in maniera casuale così da poter esplorare eventuali policy maggiormente produttive. Valori di ε troppo grandi portano a tempi di convergenza elevati mentre valori troppo piccoli non permettono di trovare la policy ottima. Il parametro α □ [0,1] influenza notevolmente le prestazioni degli algoritmi infatti per valori di α molto piccoli l'apprendimento è rallentato, mentre per valori di alpha troppo elevati l'algoritmo rischia di non convergere. Nell'implementazione degli algoritmi si inizia con un valore molto grande per poi farlo decrescere all'aumentare delle esplorazioni. Le prestazioni degli algoritmi presentati sono influenzati anche dalla inizializzazione della matrice Q(s,a) e dalla scelta delle ricompense nel caso di raggiungimento o non raggiungimento dell'obiettivo. Si è dimostrato come inizializzare una Q(s,a) con valori tutti diversi da zero porti ad avere una maggiore velocità di convergenza della soluzione rispetto a Q(s,a) con tutti valori nulli.

Algoritmo SARSA

Una caratteristica fondamentale dell'algoritmo è l'utilizzo della funzione stato-azione Q(s,a) al posto della value function V(s). In particolare SARSA deve stimare Q π (s,a) per una determinata policy π e per ogni stato s e azione a . Questo avviene utilizzando la formula descritta in seguito:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)]$$

Nell'algoritmo vengono considerate transizioni da una coppia stato-azione ad un'altra e l'aggiornamento avviene dopo ogni transizione indipendentemente se lo stato è terminale o no. Il nome SARSA deriva dal fatto che in realtà gli aggiornamenti si effettuano utilizzando una quintupla Q(s,a,r,s ′,a ′). Dove s e a rappresentano lo stato attuale e l'azione scelta nel passo corrente, invece s ′ e a ′ sono la nuova coppia stato-azione.
Il primo passo consiste nell'apprendere il valore di Q π per una determinata policy e successivamente, interagendo con l'ambiente, si può modificare e migliorare la policy.
I parametri γ e α rappresentano rispettivamente il discount factor e il learning rate. L'algoritmo SARSA converge con probabilità 1 ad una policy ottima e quindi ad un ottima funzione stato-azione Q(s,a) se le coppie stato-azione vengono visitate un numero infinito di volte.

Requisiti di progetto

Il progetto è stato svolto nell'ambito delle attività del progetto di ricerca PON ARS01_00836 denominato "COGITO – Sistema dinamico e cognitivo per consentire agli edifici di apprendere ed adattarsi".

La realizzazione del progetto consiste nel conseguimento dei seguenti macro obbiettivi:

1. formulazione del problema di schedulazione dei carichi domestici come problema di Reinforcement Learning;
2. sviluppo e test in ambiente simulato;
3. realizzazione di un prototipo con dispositivi IoT[9] ;
4. elaborato tecnico.

Relativamente al primo punto, viene richiesta la realizzazione di un modello [10] del problema utilizzando l'approccio o il paradigma del Reinforcement Learning[11]. Più in particolare si richiede di riformulare il noto problema di schedulazione[12] come un problema di RL attraverso l'uso di algoritmi afferenti alla classe denominata Temporal Difference (TD), ovvero algoritmi che basano l'apprendimento mediante differenza temporale e che adoperano la tecnica denominata bootstrap[13]

[9] Internet of Things (IoT) fa riferimento all'estensione alle cose dei benefici dell'uso di Internet finora limitati alle persone, permettendo agli oggetti di interagire con altri oggetti e quindi con le persone in modo sempre più digitale.
[10] rappresentazione formale e semplificata di un "pezzo" di realtà .
[11] è uno dei tre paradigmi principali dell'apprendimento automatico si occupa di problemi di decisioni sequenziali, in cui l'azione da compiere dipende dallo stato attuale del sistema e ne determina quello futuro. Questo tipo di apprendimento è solitamente modellizzato tramite i processi decisionali di Markov e può essere effettuato con diverse tipologie di algoritmi.
[12] problema decisionale che consiste nell'allocare risorse finite in modo tale che un dato obiettivo venga ottimizzato. Ambito della ricerca operativa che permette di risolvere problemi complessi e prendere decisioni tramite un modello matematico.
[13] è una tecnica statistica che permette di calcolare in modo approssimativo media e la varianza di uno stimatore senza conosce la distribuzione della statistica di interesse.

per la stima corrente della funzione obiettivo[14]. Tra questi, si richiede di adoperare la versione base degli algoritmi Q-Learning e SARSA in quanto l'uso degli stessi consentirà la verifica del processo decisionale e dei risultati raggiunti.

Parametri

Per definire completamente un problema decisionale di scheduling è necessario definire e valorizzare alcuni parametri o vincoli del problema. Tali parametri potranno essere ritenuti fissi e predeterminati nella fase di realizzazione del modello in ambiente simulato, successivamente dovranno essere misurati ed elaborati dalla strumentazione richiesta nel progetto.
Di seguito una breve descrizione dei vincoli/parametri del problema ed i requisiti richiesti da progetto:

a) i carichi possono essere interattivi: direttamente controllati dall'utente, oppure differibili: gestiti da una unità di controllo. Il numero massimo di carichi differibili è pari a 6;
b) per ogni dispositivo è noto il parametro beta. Beta è un parametro scelto dall'utente ed indica l'intervallo temporale entro cui il carico deve essere attivato.
c) per ogni dispositivo è noto il parametro alfa. Alfa è un parametro scelto dall'utente ed indica il tempo di esecuzione di ogni carico. Tempo in cui l'elettrodomestico resta in funzione senza subire interruzioni;
d) la schedulazione dei carichi avviene in una finestra temporale di 24 ore suddivise in step temporali o decisionali di 15 minuti, ovvero 96 time stamp;
e) è noto il profilo di consumo[15] dei dispositivi. Si possono utilizzare le informazioni contenute nei dati di targa[16] oppure da una stima ottenuta dagli appositi meter IoT[17];
f) la tariffa oraria dei prezzi dell'energia elettrica è nota sia nel caso di prelievo di energia dalla rete, sia nel caso di utilizzo di energia auto-prodotta da impianti a fonte rinnovabile[18];
g) la logica di attuazione che è la funzione obiettivo del problema decisionale, deve garantire un piano di schedulazione dei carichi tale da minimizzare la spesa energetica e ridurre i picchi di carico contenendoli sotto una soglia prefissata. Se la funzione obiettivo cercasse di minimizzare soltanto i costi, l'algoritmo cercherebbe di attivare tutti i carichi nello stesso istante di tempo. Ovvero nell'istante di minimo della curva dei costi di energia;

Per la realizzazione del prototipo è stato richiesto l'allestimento di un pannello dimostratore con le seguenti funzionalità:

- dispone di sei carichi controllabili[19] rappresentati da sei lampade con potenza differente;
- dispone di una presa di corrente per l'alimentazione dei carichi interattivi;
- consente l'attuazione dei singoli carichi attraverso tecnologia Wi-Fi;
- implementa una sottorete WLAN dedicata per gestire il traffico dati e garantire la sicurezza da attacchi informatici;
- dispone di un meter per ogni carico ed uno per l'intero sistema. I primi servono a caratterizzare il profilo energetico dei carichi controllati ed il secondo a misurare la

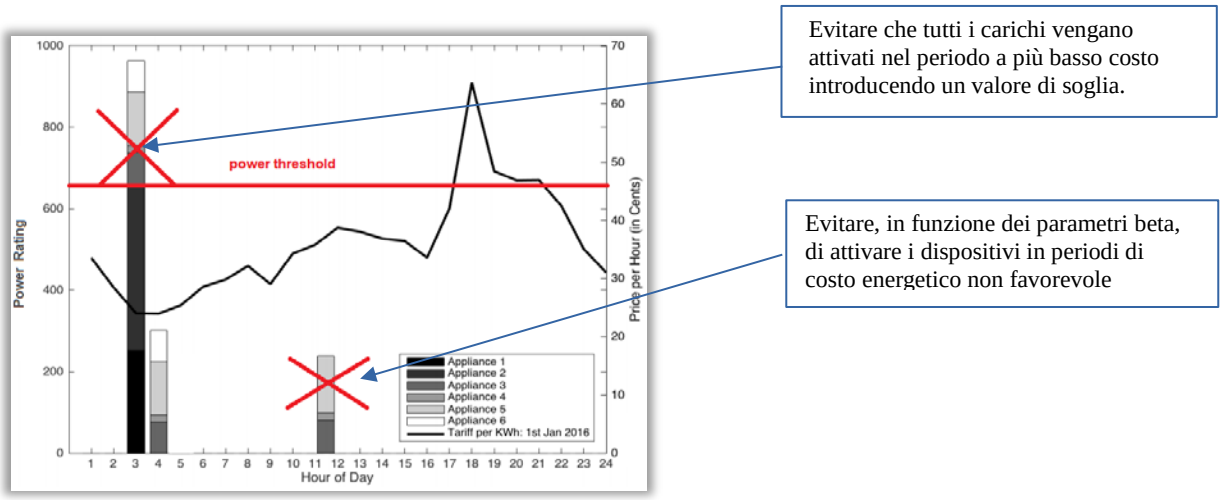
[14] la funzione obiettivo in un problema di scelta è una funzione di una o più variabili che esprime il fine in base al quale si intende effettuare la scelta.
[15] Identifica l'andamento di potenza assorbita da un apparecchio durante il suo utilizzo.
[16] Generalmente vengono riportati il consumo annuo normalizzato o la potenza nominale, che occorre convertire in potenza assorbita.
[17] In commercio sono sempre più diffusi attuatori IoT che eseguono anche un monitoraggio della potenza assorbita.
[18] Per gli impianti misti, composti da fonti rinnovabili e prelievo dalla rete elettrica, viene ricava una stima dell'andamento dei costi giornalieri, sottraendo al costo imposto dalla rete "il risparmio" ottenuto in previsione dell'energia prodotta dall'impianto fotovoltaico.
[19] Ovvero intermediati da un interruttore controllabile a distanza dal software di controllo.

potenza consumata dell'intero impianto. La finalità del secondo meter è quella di ricavare il prelievo energetico dei carichi interattivi e la dispersione dell'impianto;

- ingloba un server web per il controllo dei singoli dispositivi;
- ingloba un server web dedicato che espone i servizi di schedulazione verso le unità di controllo periferiche presenti nella rete (es. nel caso di utilizzo del sistema in un complesso abitativo condominiale dove sono presenti una o più unità di controllo per abitazione);
- integra un nodo per la misura e l'acquisizione dell'energia fotovoltaica incidente sul fabbricato.

Formulazione di un problema di schedulazione come problema di RL

Sintetizzando quando precedentemente esposto, l'obiettivo che ci prefiggiamo di raggiungere attraverso il metodo risolutivo adoperato è la generazione di una sequenza di attivazione di carichi domestici che vada a minimizzare i costi di consumo energetico e nel contempo vada ad evitare picchi di carico per l'attivazione simultanea di più dispositivi.



Per poter applicare gli algoritmi di RL al problema, innanzitutto occorre definire e strutturare il problema come un problema di MDP (Markov Decision Process), quindi occorrerà:

1) definire un insieme finito di stati;

2) definire un insieme finito di azioni di transizione tra gli stati;

3) definire una funzione di ricompensa che premi l'agente ad intraprende delle azioni;

Il problema così formulato deve godere della proprietà di Marckov, ovvero per ogni istante di tempo t T il sistema si troverà in un determinato stato appartenente ad un insieme numerabile di stati esaustivi e mutuamente esclusivi in cui l'agente potrà prendere la sua decisione guardando soltanto lo stato attuale e non lo stato precedente e/o l'evoluzione precedente effettuate.

Dalla valutazione del numero e della natura delle variabili che concorrono alla definizione della dinamica del sistema : stato dei dispositivi, tempo massimo di attivazione, tempo di esecuzione, potenza media dei dispositivi, potenza media di soglia, costo dell'energia, potenza prodotta dal fotovoltaico, numero di dispositivi da attivare e tempo di attivazione, occorre valutare accuratamente quali di queste concorrono alla definizione dello stato e quali concorrono alla definizione della funzione di reward per definire in tutte le sue parti il modello del sistema. Per superare questo scoglio, quello che generalmente si fa è considerare un sottoinsieme minimo delle variabili del sistema che concorrono alla definizione dello stato e successivamente inserire le variabili che vengono escluse all'interno della definizione della funzione di reward. Ricordiamo che quando si fa riferimento al modello del sistema, facciamo riferimento a tutte e tre gli elementi definiti poc'anzi: stato, azioni, reward. Concetto ben diverso invece è la definizione del problema di RL nella sua interezza. Quest'ultimo di fatti oltre alla definizione di un modello richiede la definizione di una policy. Tipicamente per la funzione di reward si trova una rappresentazione matematica che lega l'insieme delle variabili del problema con un numero reale. In questo lavoro viene data una definizione attraverso l'uso di pseudo-codice.

Definizione dello stato

Nel modello agente-ambiente, l'agente rappresentante "la parte decisionale" del sistema, deve essere in grado di immagazzinare un numero sufficiente di informazioni sul sistema tale da consentirgli di prendere decisioni in modo indipendente. Una decisione non è altro che l'azione migliore da scegliere tra quelle disponibili, tale da consentirgli il raggiungimento dell'obbiettivo prefissato.

L'azione è funzionale al raggiungimento di un nuovo stato del sistema. La scelta di inserire una variabile del sistema all'interno della rappresentazione dello stato è una scelta progettuale che ha forti implicazioni sul modello risultante. Il modello potrebbe non soddisfare a pieno la proprietà di Marckov oppure la cardinalità risultante degli stati potrebbe essere tale da rendere l'esecuzione degli algoritmi impraticabile. L'approccio quindi è aggiungere progressivamente un parametro del problema all'interno di una variabile di stato e capire se dal numero di informazioni inserite è possibile determinare l'intera evoluzione del sistema, mantenendo la proprietà di Marckov.

Nel nostro caso la scelta è stata di inglobare tre tipi di informazione nello stato. Queste rappresentano un sottoinsieme minimo di parametri del sistema in base al quale un'agente decisionale può intraprende delle scelte. Per poter effettuare una schedulazione temporale le informazioni minime da passare al decisore per noi sono le seguenti:

a) Il tempo di schedulazione. L'inserimento di questo parametro nello stato in realtà non è scontato. L'inserimento o l'esclusione determina il modus operandi dell'algoritmo. Se lo escludiamo, possiamo ipotizzare di effettuare delle schedulazioni "giornaliere" in cui si fissa un'ora di partenza ed una finestra temporale di esecuzione (ad esempio dalle 0:00 alle 24:00). Facendo questo presupposto ogni schedulazione inizierà allo step 0 (ore 0:00) e terminerà allo step 95 (ore 24:00). Quindi per collocare temporalmente un'attività all'interno della giornata sarà sufficiente moltiplicare la posizione dell'attività nella catena decisionale, per 15min. Il risultato dell'algoritmo sarà "il programma di schedulazione giornaliera". Tuttavia le sole informazioni di alfa e beta non sarebbero

sufficienti per collocare i carichi sull'asse temporale. In questo caso andrà indicato una finestra di preferenza per l'attivazione, altrimenti l'algoritmo va a schedulare il carico in qualsiasi ora della giornata. I requisiti del progetto richiedono invece di realizzare un sistema real-time di supporto alle decisioni, pertanto è importante determinare sia l'ora di inserimento di una nuova schedulazione che l'ora di terminazione in quando queste due informazioni definiscono la finestra di attivazione voluta dall'utente. L'operatore potrà richiedere una nuova schedulazione in ogni momento della giornata ed il completamento della richiesta potrà esaurirsi nella stessa giornata o nel giorno successivo. Inoltre nella stessa giornata possono susseguirsi più programmi di schedulazione o possono addirittura sovrapporsi senza generare ambiguità o malfunzionamenti.

b) Stato di accensione dei dispositivi. Senza dubbio la variazione dello stato dei dispositivi è l'elemento minimo indispensabile che determina un passaggio di stato. Siamo in un ambiente dinamico, il valore impostato per un dispositivo può non essere quello effettivo. E' importante quindi rilevare lo stato di attivazione dei dispositivi prima di richiederne la schedulazione;

c) Stato desiderato o Richiesta utente. Senza questa informazione non saremmo in grado di definire lo stato obbiettivo o terminale dell'algoritmo;

Fatte queste scelte progettuali possiamo ora procedere ad analizzare alcuni aspetti salienti. La prima osservazione che si può fare è che non è possibile definire uno stato iniziale del modello da cui partire per addestrare l'algoritmo.

Se consideriamo come stato iniziale ad esempio (000,111,0) ed addestriamo un modello nel giorno x, lo stesso modello non sarà replicabile per il giorno y. Questo è dovuto al fatto che i valori di reward calcolati nel giorno x, sulla base dei valori di auto produzione, costo, consumo, etc. non sono identici e replicabili per il giorno y. In altri termini avendo variabilità temporale sui vincoli del sistema, ci ritroveremo la stessa variabilità sui reward e quindi sulla value function calcolata dall'algoritmo. Possiamo affermare che la matrice della conoscenza Q che definisce il modello del sistema è tempo variante. Questo ha le seguenti ripercussioni:

a) se le condizioni a contorno (variabili che concorrono alla definizione della funzione di reward) cambiano col tempo, allora sarà necessario replicare la fase di apprendimento per ogni richiesta. Questo è dovuto al fatto che gli algoritmi di RL acquisiscono conoscenza eseguendo più episodi di apprendimento partendo dallo stesso nodo iniziale, verso i nodi terminali considerando "immutate" le condizioni a contorno.

b) il percorso markoviano generato così come la matrice di conoscenza costruita per ricavarlo, non sono riutilizzabili per altre schedulazioni. La variabilità dei costi energetici, dei parametri di settaggio dei dispositivi, della potenza media assorbita, fanno sì che partendo da uno stesso nodo iniziale ma in giorni differenti, il percorso ottimo generato e la matrice di conoscenza Q cambiano.

c) non è possibile adoperare un'unica matrice di conoscenza Q per "memorizzare" più percorsi tra i nodi iniziali e quelli terminali. La matrice Q è utilizzata dagli algoritmi di RL sostanzialmente come un'area di memoria dove conservare i valori stimati della value function per la ricerca della politica ottimale (percorso ottimo). La stessa cella della matrice Q individuata da una coppia stato azione potrebbe essere interessata da più percorsi presenti

nell’albero decisionale ed afferenti a nodi iniziali differenti. Questo porterebbe ad una sovrapposizione di valori.

Partendo da una prima rappresentazione dello stato definita come segue:

Possiamo determinare la quantità di memoria necessaria per il sistema e la complessità dell’algoritmo.

L’insieme degli stati avrà quindi una cardinalità pari ad: $S = \{96x2^{2n}\} = 393.216$

Mentre l’insieme delle azioni, definite come accensione o spegnimento di un dispositivo (0/1) avrà cardinalità pari ad: $A = \{2^n\} = 64$.

Pertanto la matrice Q avrà cardinalità pari ad: $S \times A = 25.165.824$

Anche se il numero degli stati attenuto non è un numero eccessivamente alto è possibile tuttavia effettuare altre semplificazioni.

Se consideriamo le ipotesi di progetto che se un carico viene attivato, questo rimarrà attivo ininterrottamente per un certo tempo prefissato allora lo “stato desiderato” e “stato corrente” possono essere ricondotti ad un’unica tupla che chiameremo “richiesta di attivazione”.

Il sistema dovrà quindi risolvere il problema di attivazione dei carichi senza preoccuparsi della disattivazione che avverrà in automatico (utilizzo di timer). Le azioni che consentiranno il passaggio da uno stato al successivo saranno quindi o istruzioni di attivazione o di differimento. All’interno dello stato a questo punto possiamo indicare con 0 il fatto che un dispositivo non è stato ancora attivato e con 1 dispositivo attivo o attivato.

Questa nuova rappresentazione dello stato non solo riduce la cardinalità degli stati da $96x2^{2n}$ a $96x2^n$, ma consente di scomporre il problema più generale di schedulazione di sei carichi, in sotto-problemi di minore complessità. Di fatti solo nel caso peggiore l’utente chiederà l'attivazione simultanea dei sei carichi, è molto più probabile che durante l’arco della giornata un utente chiederà l’attivazione dei carichi in tempi diversi ed al massimo di uno o due carichi per volta. Questa semplificazione del modello ci ha consentito di ricondurre il sistema, in un sistema real-time, poiché i tempi di risposta ora sono molto brevi così come l’ingombro di memoria richiesto per l'elaborazione.

Seguendo questa notazione, lo stato iniziale rappresentato dalla coppia (000,25) potrebbe identificare la richiesta di attivazione per il secondo, terzo e quarto dispositivo, oppure del primo, secondo e terzo dispositivo. È ovvio che nasce un problema di interpretazione.

Occorre quindi fare le seguenti ipotesi di lavoro:

a) il controllore che gestisce l’attuazione sui i dispositivi e riceve il comando dall’utente deve adottare una notazione posizionale per la gestione delle richieste verso lo scheduler. Nell’inviare la richiesta di schedulazione per lo stato iniziale (000,25) deve “ricordare” che il

primo bit rappresenta il dispositivo 2 ed il terzo bit il dispositivo 4. Dopo di ché, ricevuta la sequenza di schedulazione, tramuta questa informazione in azioni di "attivazione" verso i relativi dispositivi, ovvero tramuta la sequenza di schedulazione in timer programmati di accensione e spegnimento per i singoli dispositivi;

b) per identificare uno stato terminale (nodo foglia), il cui raggiungimento da parte dell'algoritmo di RL determina la conclusione di un ciclo di apprendimento, non è sufficiente testare i bit presenti nello stato ma occorrerà testare anche i tempi di esecuzione di ogni attività. Informazione non contenuta nello stato. L'algoritmo quindi terminerà se tutte le richieste sono state attivate (bit ad 1 nello stato) ed inoltre tutti i parametri alfa sono a zero.

c) L'inserimento del tempo all'interno dello stato evita la creazione di auto-anelli. Il differimento di attivazione di un dispositivo non genera un auto-anello sullo stato ma bensì un passaggio tra lo stato (xxx,t) allo stato (xxx,t+1).

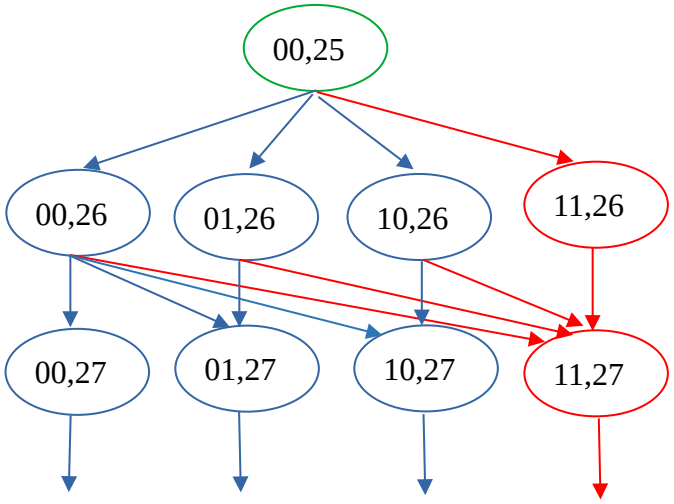
d) Lo spazio degli stati è un insieme ordinato secondo il parametro t (step). Pertanto la sequenza di schedulazione riporterà sempre valori di t crescente a partire dallo stato iniziale t0. In altri termini è possibile stabilire un ordinamento tra gli stati: (xxx,t+1) > (xxx,t).

e) Il tempo è ciclico, pertanto se deve essere processata una richiesta a partire dallo stato iniziale (000,94), raggiunto lo step temporale 95 l'elaborazione continuerà dallo stato con t =0. In altri termini è possibile stabilire questa proprietà: (xxx,0) > (xxx,95).

Albero decisionale – Marckov Decision Process Tree

Come anticipato nel paragrafo precedente, la matrice di conoscenza Q(s,a) che l'algoritmo di RL realizza per raggiungere una soluzione del problema assegnato, può essere rappresentata attraverso un albero decisionale. L'albero decisionale è una rappresentazione di un sistema costituito da uno stato iniziale, detto nodo di partenza, ed uno o più stati finali, detti nodo foglia. Le azioni o percorsi che collegano due stati del sistema sono detti archi. Il valore assegnato ad ogni cella della matrice Q(s,a), stimato dalla value function, rappresenta il peso di ogni arco. Trovare la politica ottima che risolve un problema di RL, equivale a trovare il percorso ottimo (percorso con peso massimo) tra il nodo iniziale ed il nodo finale, sull'albero decisionale. Gli algoritmi di TD-learning intervengono per "pesare" le scelte, ovvero gli archi del grafo, in funzione dei parametri che descrivono il sistema (Ci, Ki, αi e βi) al fine di determinare il percorso a guadagno massimo. Durante questa fase di ricerca si ipotizza che il sistema rimanga immutato, ovvero che le condizioni di partenza non vengono perturbate.

A questo punto è possibile rappresentare tutta la dinamica del sistema adoperando l'albero delle decisioni, chiamato **markov decision process (MDP)**.



Descrizione: l'MDP tree sopra rappresentato, descrive l'evoluzione di un sistema che opera una sequenza di attivazione di due dispositivi con l'obiettivo di minimizzare il costo e ridurre i picchi. Le azioni ammissibili per il nodo iniziale sono: attivare il primo dispositivo; attivare il secondo dispositivo; differire/attivare entrambi i dispositivi. L'algoritmo di RL assocerà un peso ad ognuno di questi archi (azioni) in funzione degli altri parametri del problema. Un nodo con valore (11,t) è terminale solo se tutti i parametri alfa (tempo di esecuzione attività) sono pari a zero. Raggiunto un nodo (11,t) l'algoritmo continuerà l'esecuzione passando dai nodi (11,t+1),(11,t+2),..fintanto che non verrà soddisfatta la condizione d'uscita. Il differimento delle attività non è infinito (00,26),(00,27),(00,28),...etc., in quanto ad ogni step viene decrementato il parametro beta (tempo massimo di differimento) ed al superamento di tale vincolo interviene una penalità (reward negativo) che obbliga l'algoritmo a convergere. Il numero delle azioni ammissibili per ogni stato decresce in funzione dei dispositivi già attivati.

Alla luce delle considerazioni fatte sulla generazione del MDP tree, appare evidente che se costruiamo l'albero considerando soltanto le sole azioni ammissibili per ogni stato, allora possiamo constatare che più scendiamo in profondità verso i nodi foglia e più il numero di stati raggiungibili si dimezza. L'utilizzo di una funzione che restituisce l'insieme delle azioni ammissibili a partire da un dato stato ci consente di ridurre i tempi di convergenza dell'algoritmo e migliorare le risorse di memoria. La strategia di ridurre il campo di ricerca fa parte dalle **policy** del nostro algoritmo.

Scegliendo questo approccio si è notato che soltanto una parte della matrice Q veniva popolata dall'algoritmo. Ciò ci ha indotto a considerare l'uso di un albero come struttura dati, anziché una matrice portando importanti benefici all'intero sistema.

Relativamente alla ricerca delle azioni ammissibili per ogni stato, occorre fare una precisazione: vengono considerate inammissibili solo le azioni che tentano di attivare un dispositivo già attivato nella schedulazione corrente o in una precedente schedulazione. Non vengono inserite in questo filtro le azioni che portano in uno stato che viola un parametro del sistema (alfa,beta o costo). Questo perché ci potrebbero essere dei casi in cui si considerano accettabili anche soluzioni sub ottime che violano alcuni vincoli del problema a favore di tempi di risposta più brevi.

Reward function

Il terzo elemento da definire per completare la definizione di un modello del sistema è appunto la definizione della funzione di reward.

In molti problemi semplici definire un reward positivo per i nodi terminali è sufficiente per far convergere l'algoritmo di RL agli obiettivi prefissati. Tuttavia può essere fatto qualcosa di più. Si può in aggiunta associare un reward negativo a nodi intermedi. In questo caso l'algoritmo per ogni step decisionale in più che eseguirà per raggiungere l'obiettivo perderà ricompensa, pertanto sarà indirizzato a trovare, non un percorso qualsiasi verso il nodo terminale, ma il "**percorso minimo**". Valorizzando i nodi intermedi o gli archi intermedi si riesce pertanto a ***perseguire un secondo obiettivo***, in questo caso il percorso più breve tra lo stato iniziale e lo stato finale.

La definizione di reward per i nodi intermedi descrive una metodologia valida per far convergere l'algoritmo verso un nodo terminale nel rispetto dei vari sub-obbittivi prefissati all'interno dello spazio degli stati.

Nel caso di **problemi con più sub-obbiettivi** abbiamo quindi la necessità di stabilire delle ricompense o penalità per i ***nodi intermedi*** in modo da correggere il percorso decisionale dell'algoritmo verso i nodi foglia durante l'esecuzione indirizzandolo a raggiungere degli stati per noi desiderati ed evitarne altri che invece porterebbero alla violazione di alcuni vincoli.

Vediamo pertanto quali sono i vincoli del nostro problema o i sub-obbiettivi:

a) accendere tutti i dispositivi voluti l'operatore (macro obiettivo);

b) evitare picchi di consumo;

c) minimizzare il consumo medio giornaliero;

d) rispettare le finestre temporali di utilizzo - preferenza utente

Per rispettare questi vincoli è indispensabile pertanto definire dei reward intermedi che indirizzino l'agente di controllo a scegliere un percorso che ne tenga conto.

Per la risoluzione del problema scegliamo quindi di procedere in maniera incrementale. Consideriamo un obiettivo per volta trascurando gli altri vincoli e scegliamo una funzione reward che soddisfi il singolo obbiettivo senza però penalizzare la ricerca degli altri. Questo ultimo problema può essere affrontato definendo un reward positivo per le scelte corrette ed un ***reward neutro o nullo per le scelte considerate errate***. Dobbiamo tenere in conto la probabilità che ci potrebbero essere dei casi in cui non tutti i sub-obbiettivi potranno essere rispettati simultaneamente, pertanto mettere dei reward negativi sulla base di una valutazione di un singolo sub_obbiettivo, porterebbe a delle ripercussioni anche sulla ricerca di una soluzione che soddisfi gli altri requisiti.

La dinamica di interazione tra agente e ambiente, come schematizzato in pag.4, prevede che ad ogni step decisionale l'agente scelga, seguendo una certa politica, un'azione tra quelle ammissibili per lo stato, e successivamente attuando quell'azione sull'ambiente, riceve un feedback negativo o positivo conseguente a quell'azione. All'interno dell'algoritmo che descrive o interfaccia l'ambiente dovremmo sostanzialmente definire la seguente funzione:

state2, reward, done, info = env.step(action1)

Le informazioni presenti all'interno del modello "ambiente" sono complete ed esaustive, viceversa nella parte del modello "agente" si hanno a disposizione soltanto le informazioni di feedback di ritorno dalla funzione di reward. Per il calcolo del reward quindi possiamo attingere alle informazioni contenute nello stato attuale ed in quello precedente più il valore di tutte le variabili che concorrono alla definizione del problema.

La funzione di reward può essere quindi realizzata attraverso una sequenza di check sullo stato. Di seguito viene mostrato lo pseudocodice adoperato:

- ricompensa al raggiungimento dello stato terminale:

```
if((stato[0][i]==1 and alfa[i]<=0)
        reward += 1
```

  Ricordiamo che lo stato è composto da due elementi: S(disp,timestamp), dove disp è una tupla binaria che descrive lo stato di attivazione dei dispositivi. Ad esempio 100, individua lo stato in cui è stato attivato il primo dispositivo (sugli N presenti sul controllore). Il valore 1 identifica il fatto che è stata eseguita una richiesta di attivazione, ma non ci dà informazioni sullo stato effettivo. Verificando il valore di alfa (tempo di esecuzione) ricaviamo il tempo trascorso e possiamo concludere se l'esecuzione del dispositivo è terminata oppure no.

- Ricompensa per l'attivazione di un dispositivo avvenuta in condizioni energetiche favorevoli:

```
if(statoprec[i]==0 and stato[i]==1):
        if(Cdisp[i,t] == min(Cdisp[i])):
                reward += 1
        else:
                reward -= 1
```

In questa funzione notiamo la ricerca su un vettore di costi: *min(Cdisp[i]).*

Questo vettore è ottenuto moltiplicando la potenza media stimata del dispositivo, il tempo di attività del dispositivo (alfa) ed il costo orario stimato di energia per quella fascia oraria.

Il costo orario stimato di energia è frutto anch'esso di una elaborazione ottenuta a partire dai dati raccolti dalla produzione fotovoltaica e dell'andamento dei costi giornalieri di energia fissati dall'operatore.

Il primo check effettuato nella funzione va a verificare se l'azione ha generato l'attivazione di un dispositivo. In caso affermativo si va a verificare se il costo generato dall'attivazione del dispositivo nell'intervallo "t - t+alfa" corrisponde al costo minimo previsto per le prossime 24 ore e per lo stesso intervallo di attivazione. Avendo presupposto il tempo ciclico, l'attivazione di un dispositivo può essere differita nel giorno successivo, pertanto la ricerca viene eseguita su tutto il vettore.

- Verifica del rispetto del parametro beta, tempo massimo di differimento:

  if (stato[i] == 0 and beta[i] <=0):

  reward -= 1

  Si effettua il check solo sul parametro beta e non su alfa in quanto quest'ultimo è gestito dal controllore attraverso l'uso di timer e quindi non è oggetto di decisione. Il controllo in questo caso consiste nel verificare se lo stato raggiunto dall'azione porta il dispositivo "i" in condizioni da non poter soddisfare il parametro beta.

- Verifica della soglia di potenza:

  ```
  if(stato[i]==1 and alfa[i]>0):
      potenza += potmed[i]
  if(potenza >soglia[t]):
      reward -= 1
  ```

  In questo caso occorre mantenere in memoria il valore della potenza impegnata dai dispositivi "accesi" e non "attivi" nella corrente schedulazione. La differenza è stata ampiamente discussa in precedenza.

  Questo valore va confrontato col valore di soglia. La soglia in questo caso non è un valore numerico fissato dall'utente ma viene calcolato sottraendo alla potenza massima consentita dall'impianto (es. 3kw) , la potenza non gestita (carichi non differibili) con la potenza "impegnata" da altri dispositivi differibili schedulati precedentemente e non presenti nell'attuale schedulazione.

o alla richiesta dopo 1000 iterazioni.

Realizzazione prototipo di sistema

La fase esecutiva del progetto ha richiesto lo studio e la definizione di molti elementi del sistema che nelle fasi precedenti erano stati approntati solo ad un alto livello di astrazione. Erano stati definiti i parametri da reperire e passare all'algoritmo di decisione, le modalità di elaborazione, il formato dei dati e del risultato, ma non era ancora chiaro come realizzare le misure, come trasmetterle, l'infrastruttura di rete, la sincronizzazione dei processi, il setup del sistema, etc.

E' stato innanzitutto necessario individuare e suddividere in più parti il sistema: una parte "Master" e una o più parti "Slave" o secondarie. Questa configurazione consente la scalabilità dell'architettura, in particolare:

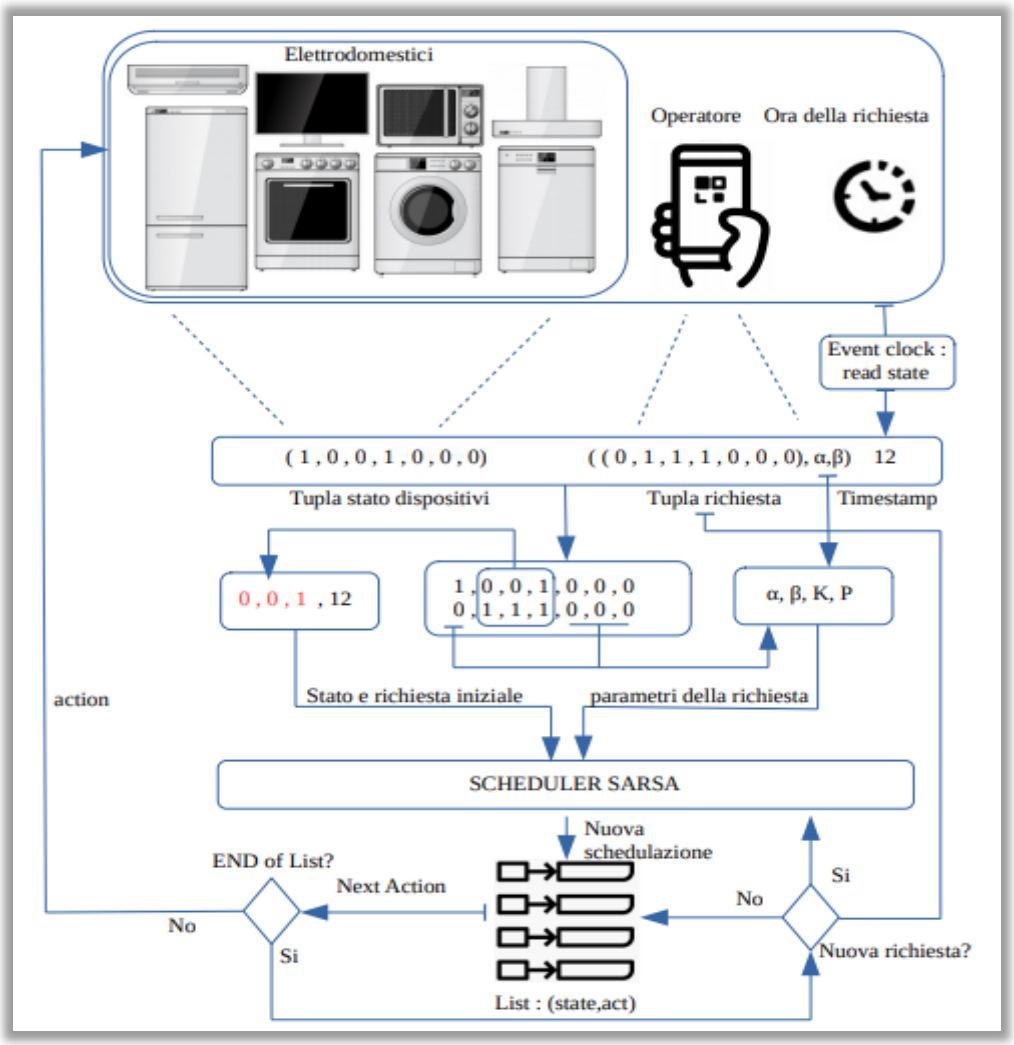
sul sistema master si rendono disponibili i seguenti servizi:

- schedulazione o algoritmi di reinforcment learning,
- web services
- server Mqtt
- Access Point per la rete Iot
- router
- DNS interno
- Dashboard

Sul sistema slave sono disponibili altri servizi afferenti alla sotto rete di controllo:

- Web server
- GUI
- Client Mqtt
- Cron jobs

In linea di massima uno schema di caso d'uso dell'intero sistema può essere così rappresentato dal seguente schema funzionale:



Descrizione: l'operatore attraverso una GUI esposta dalla parte slave del sistema, seleziona i dispositivi da voler attivare in un dato momento della giornata, settandone anche la finestra temporale di preferenza. Successivamente la richiesta viene presa in carico del web server slave che reperisce tutti gli altri parametri che definiscono una richiesta, come la potenza media dei dispositivi, la soglia di potenza dell'impianto, etc. li trasforma in un formato comprensibile dal web server Master (formato JSON) e li trasmette in attesa di una risposta.

Il web server master, ricevuta una richiesta di schedulazione, acquisisce e valida i dati trasmessi, successivamente esegue una chiamata di sistema passando i parametri ad uno script Python che realizza sia l'ambiente di addestramento che l'algoritmo di learning.

Terminata l'esecuzione della chiamata di servizio, il web server trasmette il risultato allo slave nello stesso formato della richiesta.

Da questo momento in poi l'esecuzione del programma di schedulazione è confinata al solo servizio slave.

Lo slavo non mette in produzione immediatamente la richiesta di schedulazione, ma elabora un calcolo sui costi e presenta all'utente un prospetto di schedulazione con eventuali vincoli non

soddisfatti ed il guadagno ottenuto. In questo modo l'operatore potrà decidere se rendere esecutivo il programma di schedulazione o richiederne un altro, variando il tipo di algoritmo o il numero di iterazioni di apprendimento.

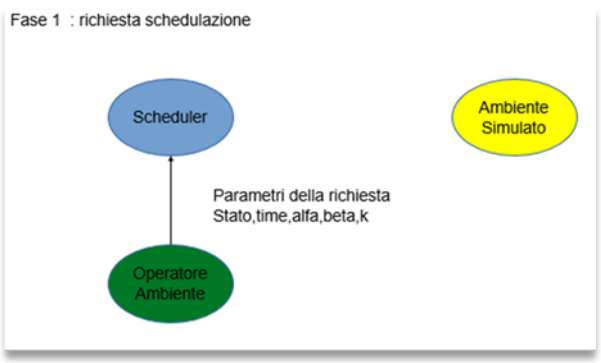
Ottenuto l'ok da parte dell'operatore, la sequenza di simboli che viene restituita dallo scheduler, viene trasformata in comandi verso i dispositivi. Le modalità di settaggio del programma di schedulazione possono essere due:

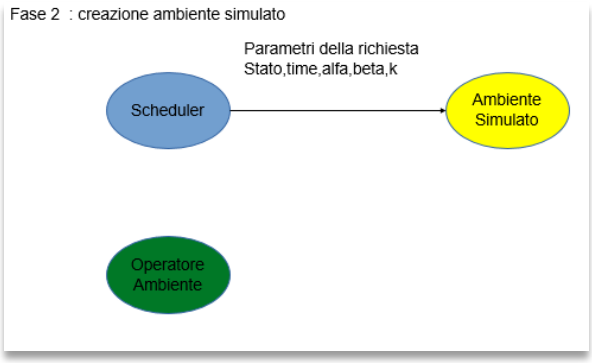
1. Viene inviato un messaggio di "setup timer "al singolo dispositivo con l'informazione del periodo di on e di off;
2. Viene generato un programma di cron job sul server slave. Il programma invierà i singoli comandi di on o di off ai dispositivi nei tempi prestabiliti.

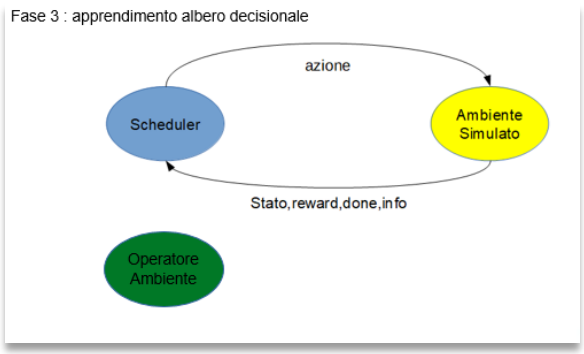
Le due soluzioni offrono dei pro e dei contro. La prima soluzione è più robusta in termini di default di rete, per cui una volta lanciati i comandi lo slavo potrebbe anche essere spento. Tuttavia decentralizzando il controllo si rischia che se l'utente interviene sui singoli dispositivi, il programma di schedulazione non verrebbe più rispettato.

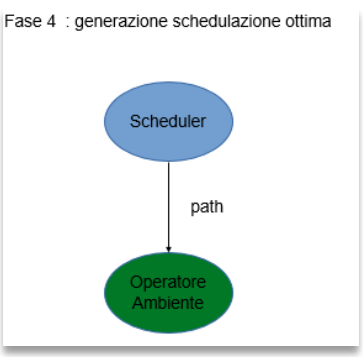
La seconda soluzione soffre dei rischi di default di un sistema di controllo centralizzato ma consente di tenere costantemente sotto controllo l'intero sistema. Per cui se un utente interviene in sostituzione del servizio su un dispositivo, il sistema di controllo lo riconosce e si riporta nelle condizioni programmate.

Di seguito la sequenza delle fasi di gestione di una richiesta di schedulazione:



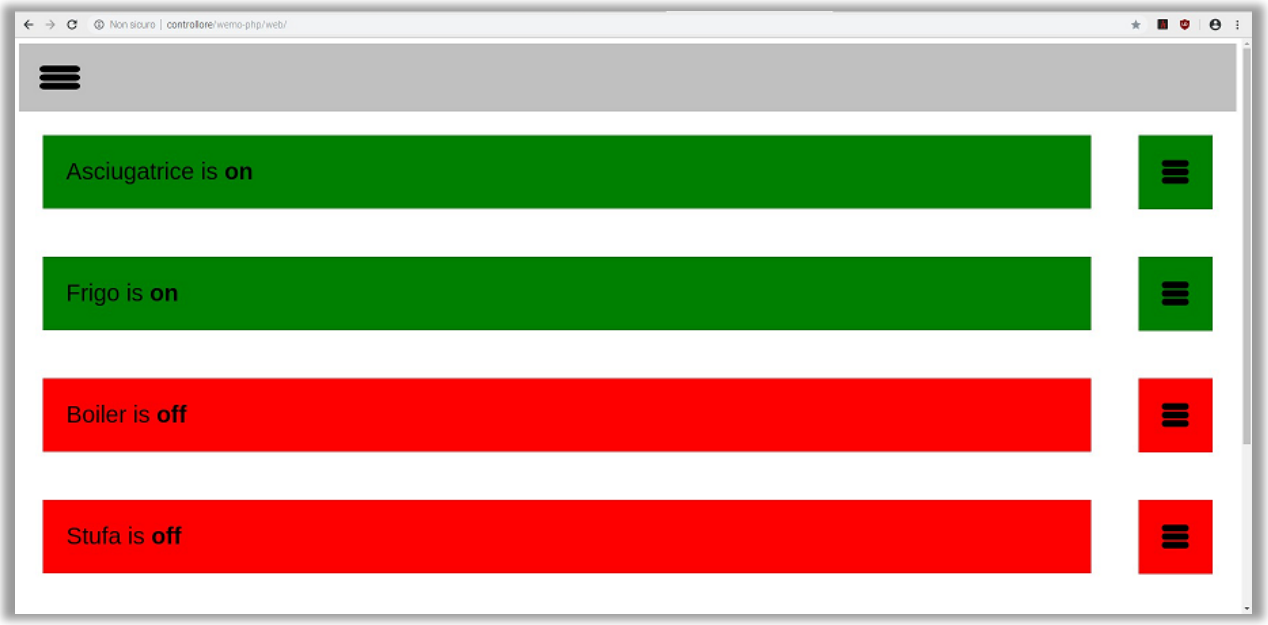






GUI di sistema

L’interfaccia grafica realizzata è di tipo responsive ovvero in grado di adattarsi graficamente in modo automatico al dispositivo coi quali vengono visualizzati i contenuti. E’ stata realizzata utilizzando la tecnologia PHP e Javascript al fine di rendere dinamici i contenuti. La home page della GUI è la seguente:



Presenta un menu in alto che permette la navigazione nelle altre pagine e dei menu laterali con cui è possibile accedere ai settaggi dei singoli dispositivi.

La pagina presenta una lista di dispositivi, corrispondente ad ogni singolo IP di rete, individuati attraverso il nome a loro assegnato. Il coloro verde, rosso e arancio ne indicano lo stato acceso, spento, non raggiungibile.

Tali informazioni vengono aggiornate utilizzando la messaggistica mqtt. Ogni evento di cambio stato dei dispositivi viene trasmesso al broker mqtt, ed un client presente sullo slave in ascolto dei messaggi, li interpreta e li traduce in informazioni grafiche.

Cliccando sul menu in alto è possibile accedere alla pagina dei servizi. Anch’essa realizzata con la stessa grafica della home page, presenta una lista di servizi con il relativo pulsante menù sulla destra.

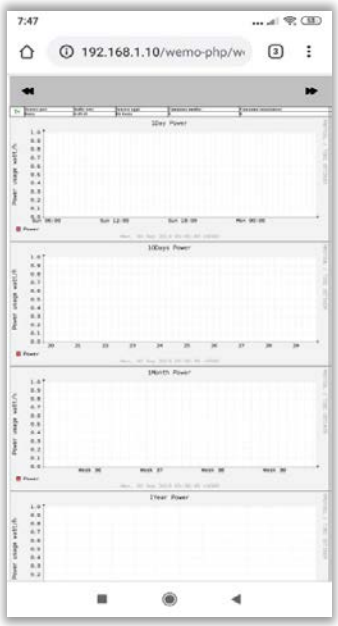
In questo caso cliccando sul nome del servizio si accede ad una pagina di visualizzazione dei dati, mentre cliccando sul menu a destra si accede ad una pagina di setup. I colori in questo caso non hanno un significato specifico.

In basso alcune immagini relative alle pagine dei servizi raffigurate su PC e su smartphone:


Schedulazione
Dettaglio Consumi
Parematri Ambientali
Configurazione Dispositivi


7:48
192.168.1.10/wemo-php/w
Dispositivi
Schedulazione
Consumi
Ambiente
Tv is on


7:51
192.168.1.10/wemo-php/w
Tv is off

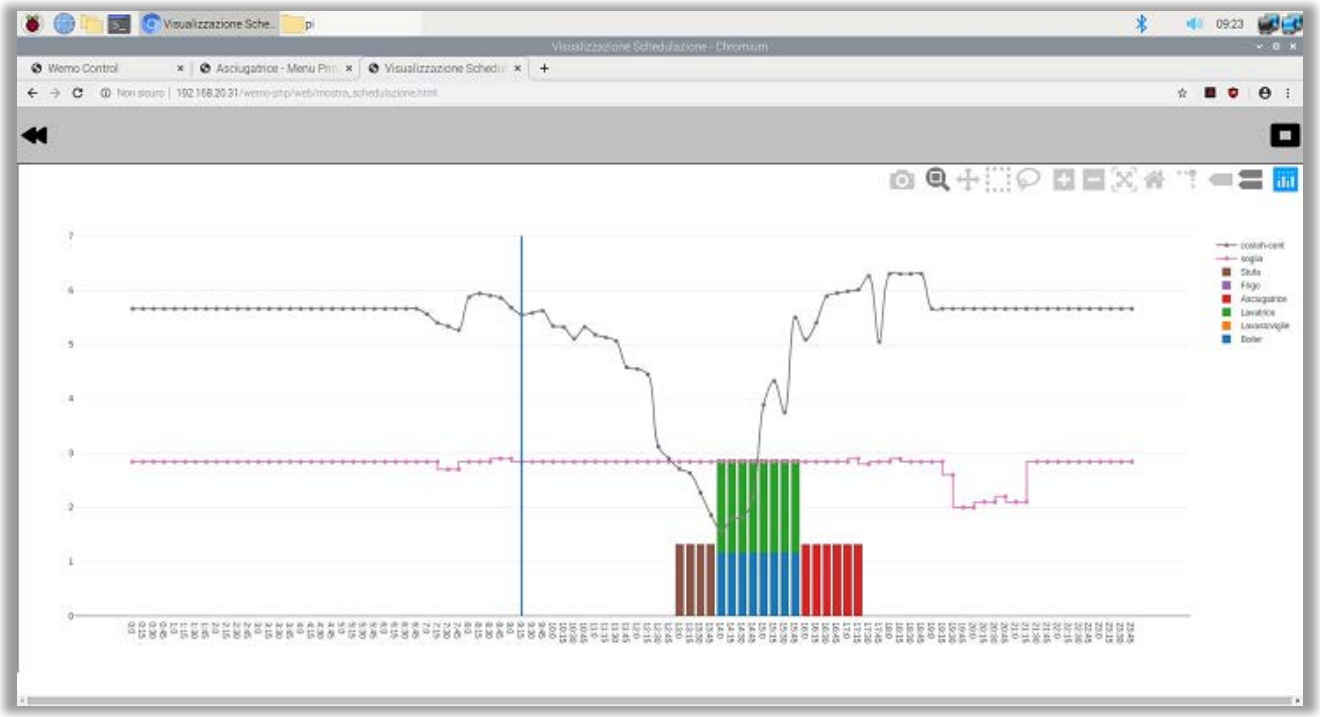

7:47
192.168.1.10/wemo-php/w


192.168.1.10/wemo-php/w
Parametri Dispositivo
SCEGLIERE DISPOSITIVO
Asciugatrice
TIPO DI PARAMETRO
Beta: tempo massimo di ritardo
Alfa: tempo minimo di accensione
SELEZIONARE VALORE (STEP MIN DI 15')
1500min - 3900min


192.168.1.10/wemo-php/w
Parametri Ambientali
SELEZIONARE PARAMETRO
Costo Energia
Invia


192.168.1.10/wemo-php/w
Parametri Dispositivo
SCEGLIERE DISPOSITIVO
TV
Invia

L’interfaccia di sistema che mostra l’andamento temporale del programma di schedulazione è rappresentata di seguito:



Le informazioni che vengono presentate sono le seguenti:

- Sull’asse delle ascisse vengono rappresentati gli slot temporali di schedulazione;
- Sull’asse delle ordinate vengono sovrapposti due tipi di grafico: uno rappresenta l’andamento della potenza disponibile durante la giornata, calcolata sulla base di stime della giornata precedente, ed un secondo che rappresenta l’andamento espresso in centesimi del costo dell’energia anch’esso calcolato sulla base di stime fatte sulle produzione della giornata precedente.
- Il grafico a barre raffigura l’impegno di potenza dei dispositivi schedulati e la loro collocazione temporale. L’altezza delle barre indica la potenza impegnata del carico, il colore identifica i diversi tipi di dispositivo;
- La barra verticale indica l’ora corrente e quindi lo stato di esecuzione del programma di schedulazione.

Dalla sovrapposizione dei tre tipi di grafico: potenza disponibile, curva dei costi, impegno di potenza dei carichi, è possibile avere un’informazione intuitiva sulla bontà di scelta del decisore. Nell’esempio possiamo verificare che:

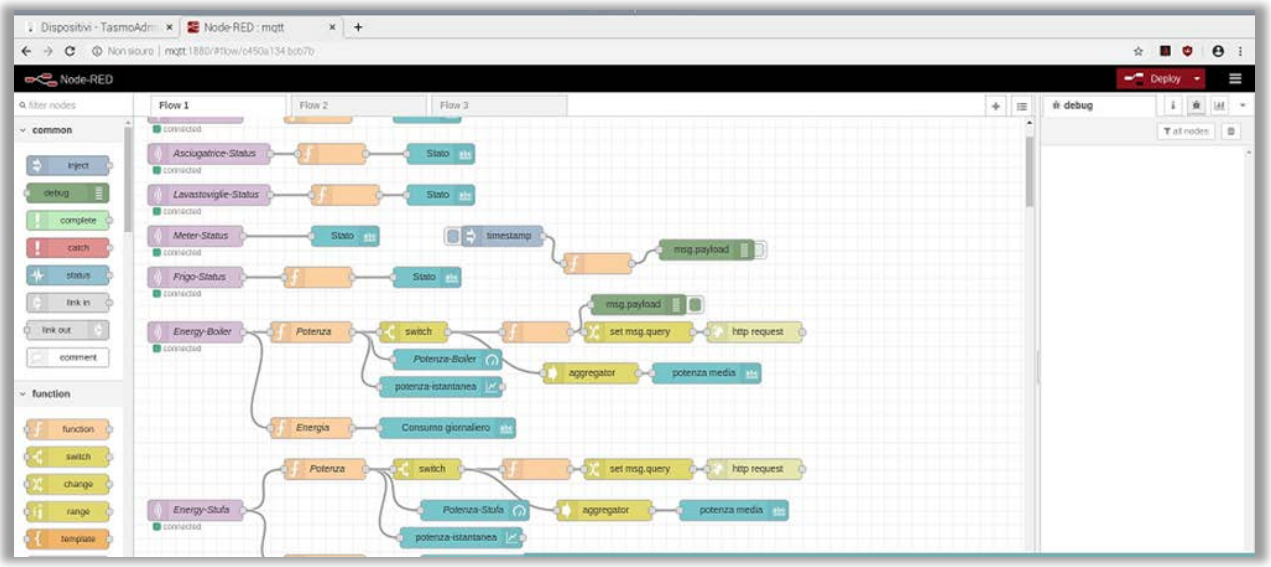
- è in esecuzione un programma di schedulazione dei seguenti carichi: stufa, boiler, lavatrice ed asciugatrice;

- la visualizzazione del programma, come mostrato dalla barra di avanzamento, è avvenuta tra le 9:15 e le 9:30;
- che è previsto un impegno di potenza dei carichi interattivi nelle fasce orarie: 7:15-9:00 ed 17:30-21:30, dove si denota un impegno maggiore nelle ore serali;
- la produzione da fotovoltaico avviene tra le 6:45 e le 17:00 con picco massimo alle 14:00. Tale produzione non è sufficiente a coprire la richiesta di 3 Kw/h, pertanto il costo medio stimato per un impegno di 3 kw/h raggiunge un minimo di 1,8 centesimi di euro.

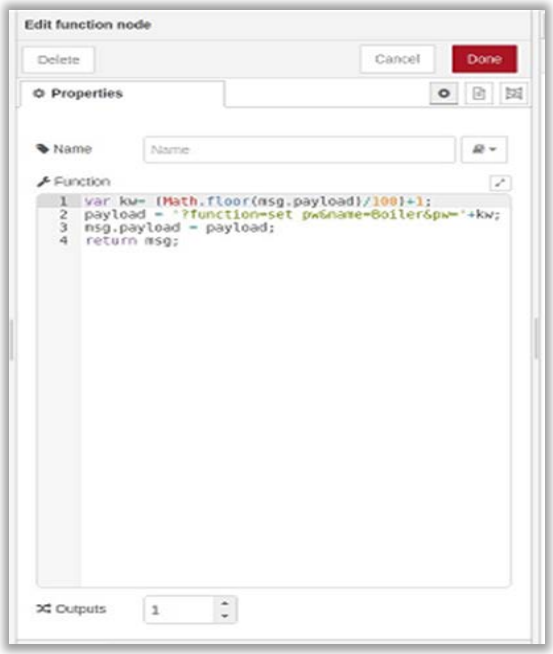
Server Mqtt e Dashboard

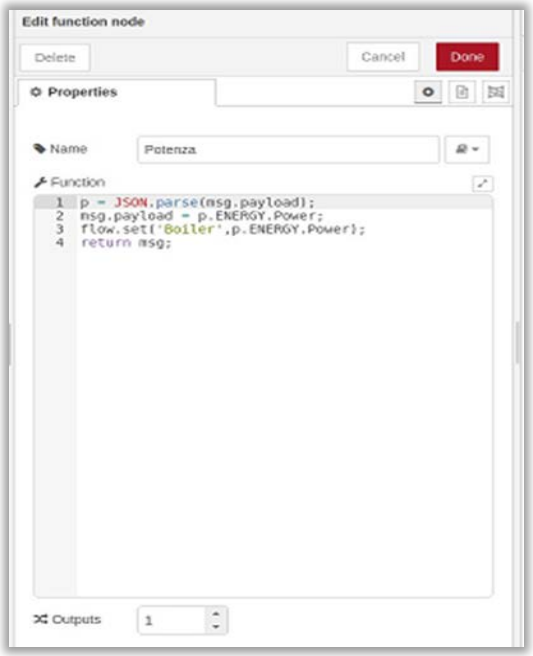
La gestione della messaggistica mqtt, dell'elaborazione dei dati come il calcolo della potenza media, della potenza prodotta dal fotovoltaivo e la presentazione dei consumi viene effettuata attraverso i tools node red.

Per ogni dispositivo viene creato un listner mqtt a cui segue un processamento dei messaggi. Scopo dell'elaborazione dei messaggi mqtt è la rappresentazione in una dashboard e il calcolo dei valori medi con aggiornamento dei parametri sullo slave attraverso una http request.

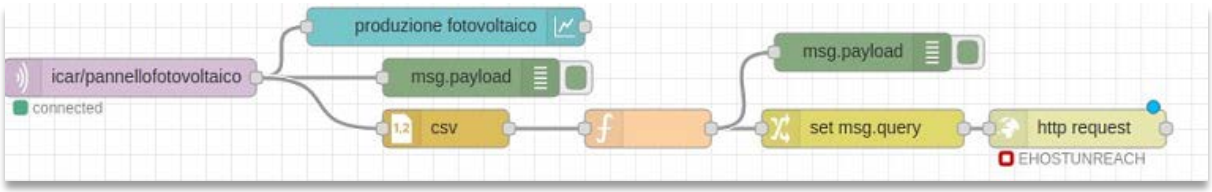


Di seguito due esempi di elaborazione dei messaggi mqtt. La prima funzione prepara un messaggio per una richiesta di aggiornamento http sul server slave relativamente alla potenza del dispositivo Boiler, la seconda funzione estrae l'informazione della potenza istantanea da un messaggio mqtt (dati in formato JSON), la memorizza in una variabile e la passa avanti nel flusso per successive elaborazioni.



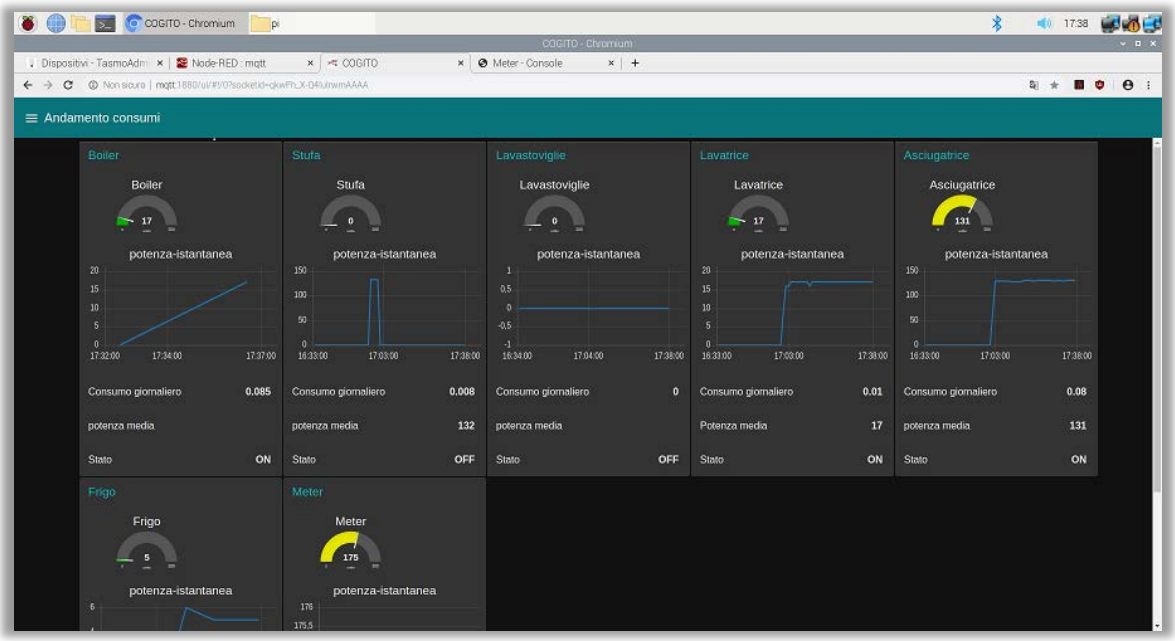


Per quando riguarda la produzione da impianto fotovoltaico, il flusso di elaborazione dei messaggi mqtt è il seguente:



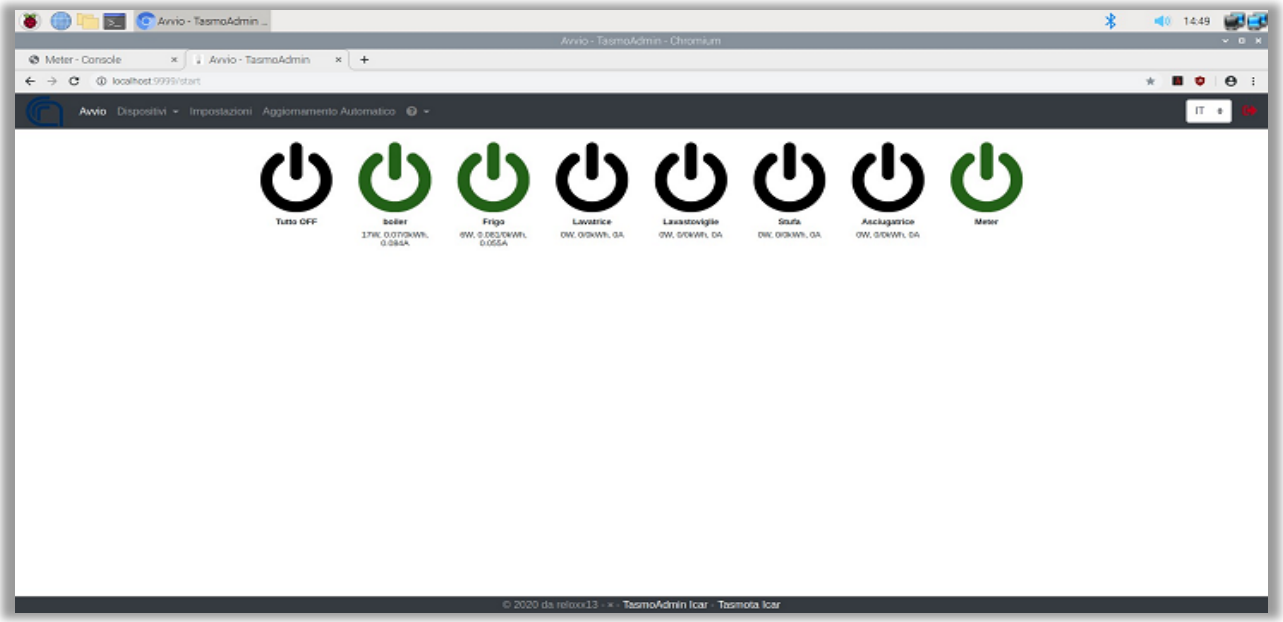
I dati vengono presentati in un grafico in formato grezzo e contestualmente vengono pre elaborati prima di inviarli allo slave per aggiornare i parametri dell'ambiente. Di seguito la funzione di aggiornamento:

La dashboard realizzata presenta due finestre, una relativa ai consumi dei dispositivi ed una relativa alla produzione giornaliera di fotovoltaico.

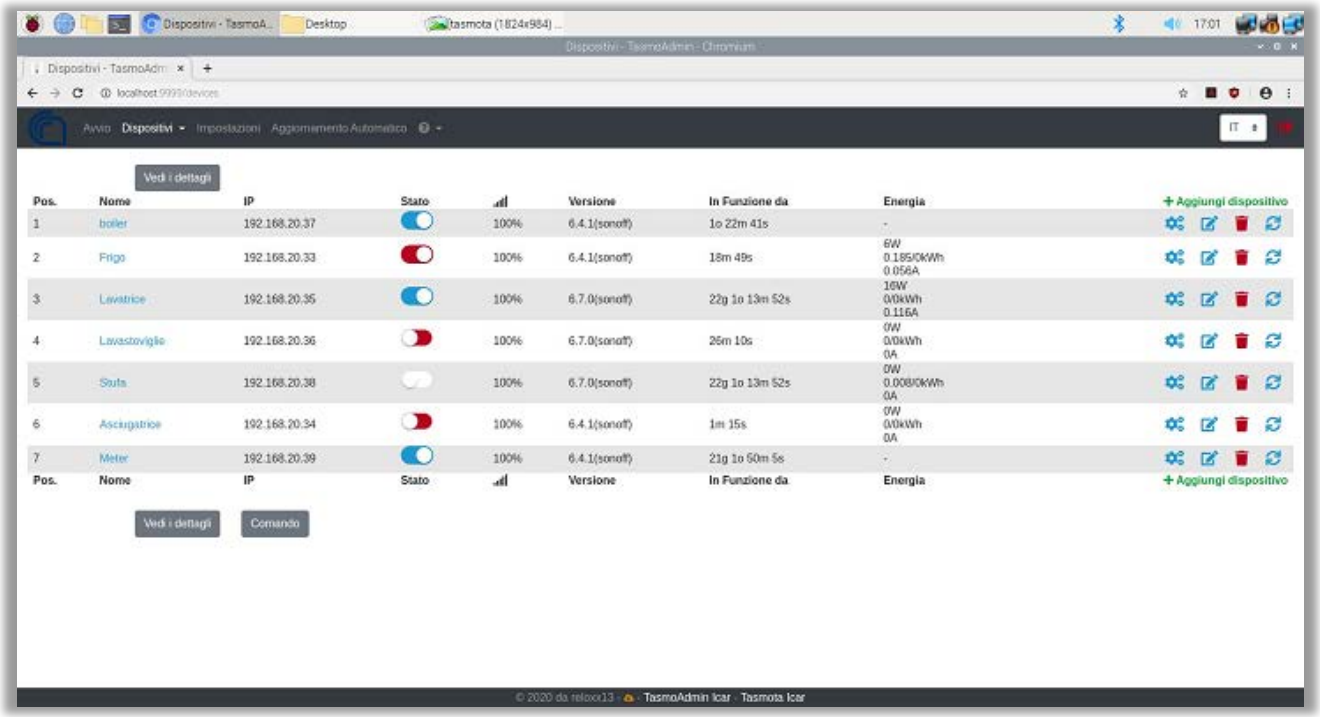


Gestione dispositivi

L’attuazione e la misura di potenza sui singoli carichi ha richiesto l’istallazione di dispositivi smart meter. Nello specifico è stato usato il dispositivo Sonoff POW su cui è stato installato il firmware opensource Tasmota. Questo ha permesso la qestione degli stessi sia attraverso interfaccia web che attraverso messaggistica mqtt. L’intera rete può essere gestita attraverso un’unica interfaccia web centralizzata come mostrato di seguito:

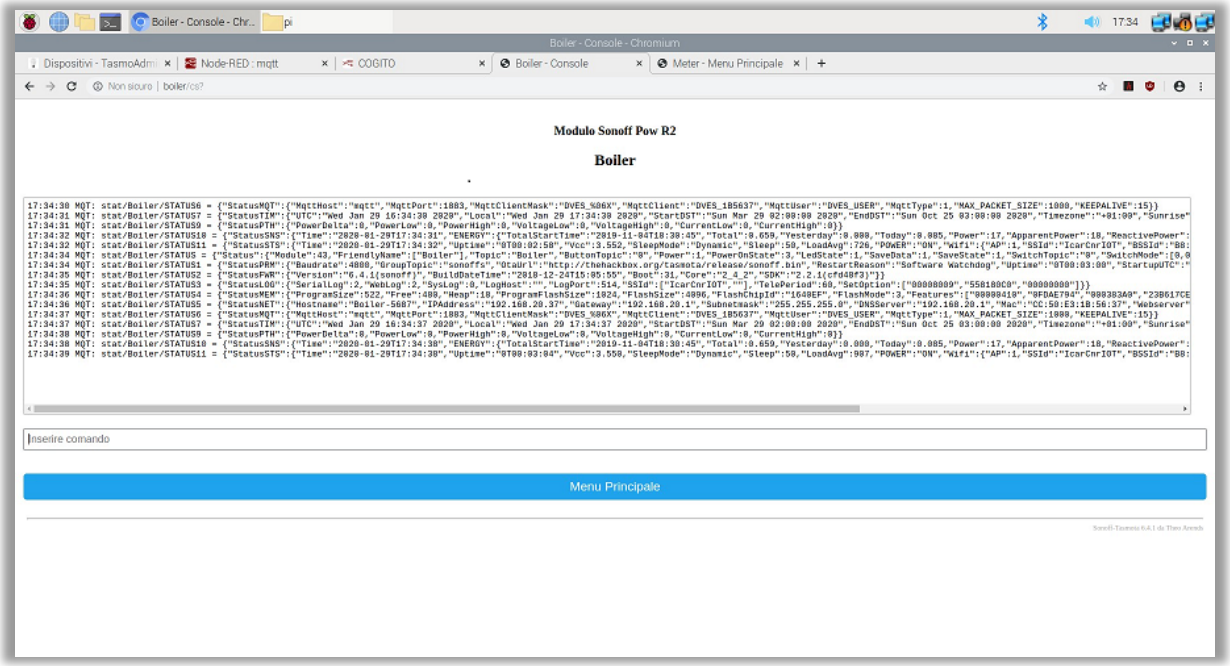


La home page del servizio di amministrazione della rete presenta lo stato dei dispositivi con informazioni relative al loro consumo. Per poter gestire gli aggiornamenti firmware o altre impostazioni di rete occorre accedere alla pagina “Dispositivi” come mostrato di seguito:



Un altro modo per effettuare le operazioni di settaggio sui parametri di rete dei singoli dispositivi è accedere direttamente al server web di ogni singolo dispositivo. Questo avviene digitando sul browser l’indirizzo ip assegnato al meeter. La pagina di setting dei dispositivi è mostrata di seguito:

Degno di nota è il menu “Console” che viene esposto dal servizio web. Accedendo a questo servizio è possibile monitorare il flusso di messaggi mqtt da e per il dispositivo, oltre alla possibilità di digitare i singoli messaggi direttamente sulla console per effettuare il testing dei messaggi.



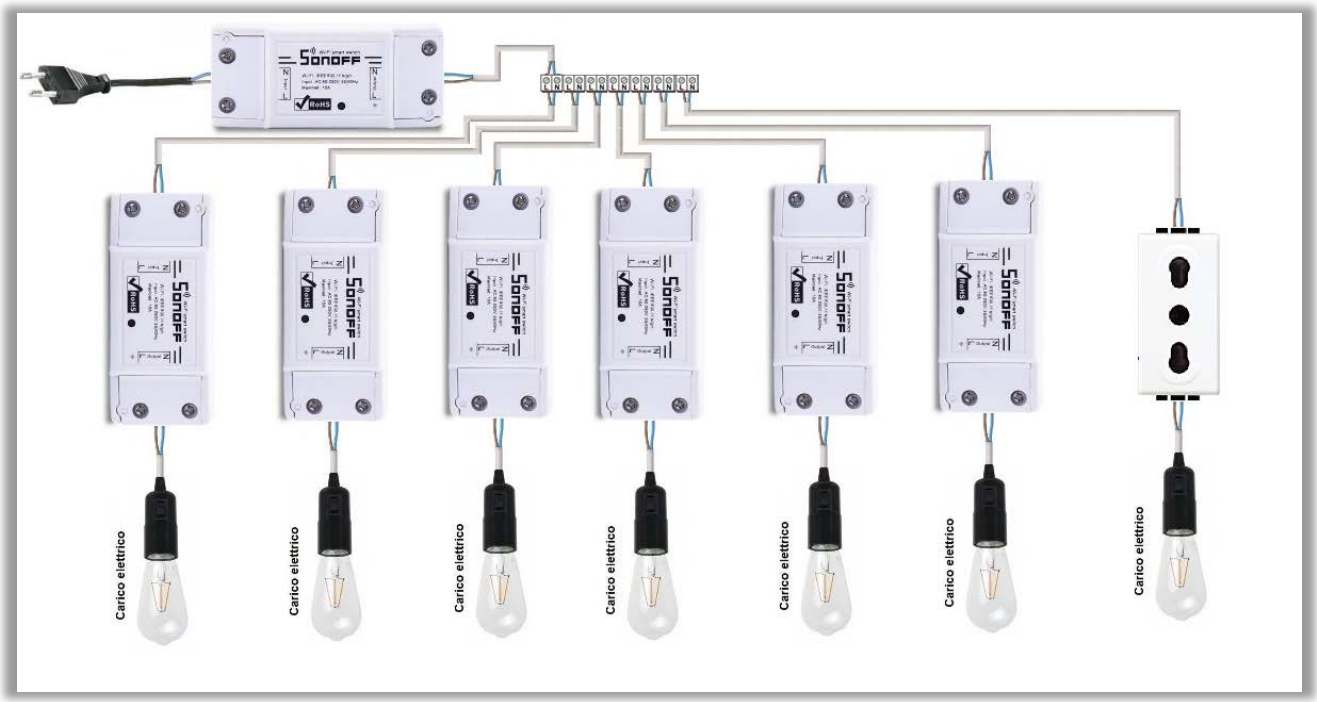
Impiantistica

Per la realizzazione del dimostratore sono stati predisposti sei carichi rappresentati da sei lampade di diverso vattaggio: da 18 Watt fino a 100 Watt per le lambade ad incandescenza di vecchia generazione. Per portare la potenza ai valori ti targa degli elettrodomestici che rappresentano è stato inserito un fattore moltiplicativo.

Oltre ai corpi luminosi è stata inserita una presa di corrente non controllabile da Sonoff che ha lo scopo di simulare i carichi interattivi.

In testa all'impianto abbiamo un meeter che rileva sia i consumi dei carichi interattivi che quelli controllati.

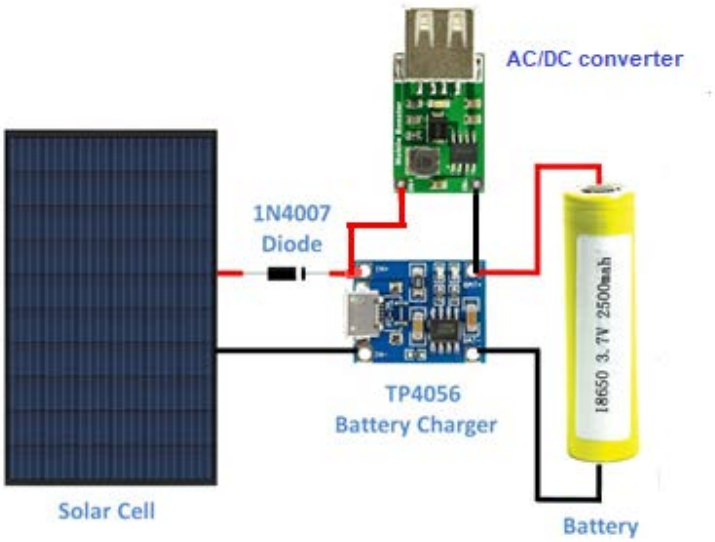
Lo schema di collegamento viene mostrato di seguito:



Solarimetro – pannello fotovoltaico

Di seguito una breve descrizione dell'ultimo elemento impiantistico del sistema. Al fine di testare il sistema in condizioni di utilizzo reali, si è deciso di integrare al sistema un nodo di acquisizione dell'energia solare incidente in modo da simulare la produzione di un impianto fotovoltaico di 3Kw.

Il nodo di acquisizione è costituito da un pannello fotovoltaico di 3Watt collegato attraverso un convertitore AC/DC che acquisisce la potenza prodotta dal pannello e la trasmette al mini pc raspberry, che ad intervalli di 1 minuto trasmette il dato in formato mqtt al server di rete.



Gestione del programma di attuazione dello scheduling giornaliero con task cron via bash

Il programma di scheduling prodotto dall'algoritmo di RL, deve essere rese esecutivo sui dispositivi di controllo della rete (smartplug IoT). Questo può avvenire in diversi modi, uno di questi ad esempio è il settaggio dei timer disponibili su ogni singolo dispositivo/smartplug della rete con firmware Tasmota. Il settaggio di timer sui dispositivi Tasmota può avvenire via Mqtt. Una volta impostati i timer, questi possono essere cancellati, resettati o modificati, ma in ogni caso la funzione del controller di rete termina la sua esecuzione. Un altro metodo è quello che abbiamo adottato nel nostro prototipo, ovvero il controllo di attuazione rimane per tutto il tempo della schedulazione al dispositivo di controllo della rete, che invia i singoli comandi di accensione e spegnimento nei tempi prestabiliti. Per fare ciò si è deciso di utilizzare i cron task messi a disposizione dal SO del controllore.

Quello descritto di seguito è un tipico scenario con diversi server o servizi che devono tutti avere un lavoro specifico da eseguire su base pianificata. Naturalmente, ci sono strumenti specializzati che possono automatizzare questo processo come Ansible, Puppet e Chef, ma a volte, necessita una

soluzione più rapida. In questi casi torna utilissima la shell Bash per portare a termine l'attività. Solitamente si procede così per creare manualmente un cron job:

crontab -e

Ad esempio per aggiornare l'ora del server col protocollo NTP ogni giorno a mezzanotte si può lanciare:

0 0 * * * ntpdate ntpsrv.domain.tld

Una gestione del genere va bene per pochi server, ma dovendone gestire diversi, potenzialmente centinaia, questo approccio non risulta praticabile.

Fortunatamente, cron ha un altro modo di accettare i lavori: basta usare il comando cron, seguito da un file di testo contenente i task da gestire. Ad esempio, se ho un file chiamato jobs.txt che contiene cron jobs, posso aggiungerlo a cron come segue:

cron jobs.txt

Tuttavia, quando si utilizza cron in questo modo, si sostituiscono tutti i processi cron con quelli presenti nel file. La soluzione è semplice, basta scaricare tutti i contenuti dei lavori crontab correnti in un file di dump, aggiungere qualsiasi lavoro a quel file, quindi utilizzare cron con quel file:

crontab -l> jobs.txt

Scarica i cron job esistenti in un file

echo "0 0 * * * ntpdate ntpsrv.domain.tld" >> jobs.txt

Aggiungi un nuovo lavoro al file

cron jobs.txt

Si possono anche modificare righe specifiche mediante l’uso del comando grep:

crontab -l | grep -v ntpdate> jobs.txt

La differenza qui è che abbiamo chiesto a Bash di visualizzare tutto il contenuto del file crontab escluso il lavoro che conteneva ntpdate. Abbiamo usato grep con l'opzione -v, che significa prendere tutto tranne ciò che viene dopo -v.

Tornando al nostro caso, abbiamo creato il seguente script (script.sh) che automatizza la creazione del cronjob:

crontab -l> /tmp/jobs.txt

echo "0 0 * * * ntpdate ntpsrv.domain.tld" >> /tmp/jobs.txt

cron /tmp/jobs.txt

Dnsmasq per la gestione dei servizi di rete (DHCP,DNS,TFTP)

Uno dei requisiti del prototipo è quello di realizzare un dispositivo di controllo di tipo plug&play. Ovvero il dispositivo deve fornire un Access Point di rete per tutte le smart plug collegate, deve fornire una interfaccia centralizzata per la configurazione e gestione di ogni elemento della rete.

Di seguito riportiamo alcuni passaggi e configurazioni necessarie per settare un access point di rete.

Tutti gli host di rete devono avere i propri nomi host ed essere configurati per ottenere le configurazioni di rete tramite DHCP (protocollo di configurazione host dinamico). Se si dispone di alcune macchine con indirizzi IP statici, anche Dnsmasq può incorporarli, quindi non è necessario cambiali.

Un server dei nomi autorevole serve a pubblicare gli indirizzi dei server pubblici. Se si dispone di un server con connessione a Internet come un sito Web, un server di posta o un server FTP, da qualche parte è presente un server autorevole che pubblicizza i propri indirizzi IP e nomi. Può trattarsi di un server DNS (Domain Name Services) autorevole nei tuoi locali o gestito da una terza parte come il tuo provider di servizi Internet o un servizio di hosting. Puoi interrogare qualsiasi server pubblico con il comando dig per vedere come il suo nome e indirizzo IP sono abbinati:

```
$ dig +nocmd www.linux.com +noall +answer
www.linux.com.  5276 IN  A  140.211.169.7
www.linux.com.  5276 IN  A  140.211.169.6
```

Un server DNS autorevole può essere pensato come alla rubrica principale di un dominio Internet. Questa rubrica viene copiata sui server DNS principali del mondo e quindi copiata da innumerevoli altri server in Internet. È un sistema distribuito che fornisce velocità e tolleranza ai guasti. Mantenere i server autorevoli completamente separati dagli altri tipi di server dei nomi. Quindi si va ad usare BIND, PowerDNS o MaraDNS per il tuo server autorevole e Dnsmasq per i servizi di nomi LAN privati e la memorizzazione nella cache.

Una cache DNS è una copia locale degli indirizzi dei siti visitati. Ciò accelera le prestazioni della rete perché le applicazioni di rete non devono attendere che i server remoti rispondano alle query DNS.

Un name server ricorsivo è quello che cerca l'indirizzo dei siti che si può visitare. Le funzioni ricorsive e cache sono spesso combinate nello stesso server. Ad esempio, quando si configura il DNS per l’account Internet, i server DNS del’ISP sono molto probabilmente server ricorsivi e di cache. I server DNS pubblici come Google Public DNS e OpenDNS sono server ricorsivi e di memorizzazione nella cache. Dnsmasq non è un server dei nomi ricorsivo, ma può essere configurato per interrogare qualsiasi server ricorsivo desiderato.

Trivial File Transfer Protocol (TFTP) è un server FTP molto semplice e insicuro utilizzato all'interno di reti private per l'avvio in rete di PC e dispositivi embedded come router ed endpoint VoIP (voice over IP).

Per questo lavoro abbiamo creato una piccola rete con due sottoreti: una cablata e una wireless, a 192.168.1.0 e 192.168.2.0. Dnsmasq è installato su un router LAN con interfacce sia cablate che wireless a 192.168.1.10 e 192.168.2.10. Per prima cosa, occupiamoci di alcune importanti impostazioni globali:

```
#/etc/dnsmasq.conf
```

```
domain-needed
bogus-priv

domain=mydomain.net
expand-hosts
local=/mydomain.net/

listen-address=127.0.0.1
listen-address=192.168.1.10
listen-address=192.168.2.10
bind-interfaces
```

L'aggiunta del dominio necessario blocca le richieste incomplete di lasciare la tua rete, come google invece di google.com. bogus-priv impedisce che gli indirizzi privati non instradabili vengano inoltrati fuori dalla rete.
Impostare il nome di dominio privato con dominio = miodominio.net, sostituendo miodominio con qualsiasi nome di dominio desiderato. Non è necessario registrarlo con un registrar di nomi di dominio perché è privato e non lascia la LAN.
La direttiva expand-hosts aggiunge il nome di dominio ai nomi host, in modo da ottenere nomi di dominio completi come hostname.mydomain.net. Ancora una volta, questi sono completamente arbitrari e possono essere scelti a piacere.
local = / miodominio.net / garantisce che le query per il dominio privato ricevano risposta solo da Dnsmasq, da / etc / hosts o DHCP.
La direttiva Listen-address dice a Dnsmasq su quale interfaccia o interfacce ascoltare. Usa sempre l'indirizzo di ascolto perché non si vuole che Dnsmasq sia esposto alle reti sbagliate, e soprattutto non a Internet e si deve includere sempre l'indirizzo di loopback. La direttiva bind-interfaces garantisce che Dnsmasq ascolterà solo gli indirizzi specificati con listen-address.

Ora vediamo la configurazione del DHCP per le nostre due sottoreti:

```
dhcp-range=lan,192.168.1.100,192.168.1.200
dhcp-range=wifi,192.168.2.100,192.168.2.200
```

Questo esempio fornisce un centinaio di indirizzi DHCP per sottorete. Nota che sono etichettati con i tag lan e wifi. Si tratta di un sistema semplice che semplifica la fornitura di servizi diversi a sottoreti diverse, come negli esempi seguenti:

```
#set default gateway
dhcp-option=lan,3,192.168.1.50
dhcp-option=wifi,3,192.168.2.50

#set DNS server
dhcp-option=lan,6,192.168.1.10
dhcp-option=wifi,6,192.168.2.10
```

La prima stanza imposta il percorso predefinito per ogni sottorete. Il tag numero 3 significa router. La seconda stanza dice ai nostri client LAN di ottenere il loro DNS dal server Dnsmasq.

Vediamo di seguito cosa occorre fare invece per configurare l’upstream name server.

È necessario indicare a Dnsmasq dove inoltrare le richieste DNS di Internet. Questo potrebbe essere il server dei nomi del ISP o qualsiasi servizio DNS. È bene utilizzare almeno due servizi completamente diversi. Questo esempio utilizza Google Public DNS e OpenDNS:

```
server=8.8.8.8
server=8.8.4.4
server=208.67.220.220
```

Vediamo ora come settare gli indirizzi IP statici.
Dnsmasq incorpora senza problemi host con indirizzi IP statici nel tuo DNS locale. Supponiamo di avere tre server con indirizzi statici; tutto ciò che si deve fare è aggiungerli al file / etc / hosts sul server Dnsmasq:

```
 127.0.0.1       localhost

192.168.1.15   server1
192.168.1.16   server2
192.168.1.17   server3
```

È possibile abilitare il server TFTP integrato di Dnsmasq aggiungendo questa riga a dnsmasq.conf:

```
dhcp-boot=pxelinux.0
```

E poi impostare la directory di avvio e la configurazione di pxelinux.
Esaminiamo la gestione avanzata dei file di configurazione, come testare le tue configurazioni, alcuni aspetti di sicurezza di base, caratteri jolly DNS e la configurazione DNS veloce.
Quando si testano nuove configurazioni, è necessario eseguire Dnsmasq dalla riga di comando, anziché come daemon. Questo esempio lo avvia senza avviare il demone, stampa l'output del comando e registra tutte le attività:

```
# dnsmasq --no-daemon --log-queries
dnsmasq: started, version 2.75 cachesize 150
dnsmasq: compile time options: IPv6 GNU-getopt
DBus i18n IDN DHCP DHCPv6 no-Lua TFTP conntrack
ipset auth DNSSEC loop-detect inotify
dnsmasq: reading /etc/resolv.conf
dnsmasq: using nameserver 192.168.0.1#53
dnsmasq: read /etc/hosts - 9 addresses
```

Per impostazione predefinita, Dnsmasq non dispone di un proprio file di registro, quindi le voci vengono scaricate in più posizioni in / var / log. Questo esempio cerca / var / log ricorsivamente, stampa i numeri di riga dopo i nomi dei file ed esclude / var / log / dist-upgrade:

```
# grep -ir --exclude-dir=dist-upgrade dnsmasq /var/log/
```

Dnsmasq è configurato in /etc/dnsmasq.conf. La tua distribuzione Linux può anche usare / etc / default / dnsmasq, /etc/dnsmasq.d/ e /etc/dnsmasq.d-available/.

/etc/dnsmasq.conf è il file principale. Dnsmasq lo legge per primo all'avvio. /etc/dnsmasq.conf può chiamare altri file di configurazione con l'opzione conf-file =, ad esempio conf-file = / etc / dnsmasqextrastuff.conf, e directory con l'opzione conf-dir =, ad es. conf-dir = / etc / dnsmasq.d.
Ogni volta che si apporta una modifica a un file di configurazione, è necessario riavviare Dnsmasq.
È possibile includere o escludere file di configurazione per estensione. L'asterisco significa includi e l'assenza dell'asterisco significa escludi:

```
conf-dir=/etc/dnsmasq.d/,*.conf, *.foo
conf-dir=/etc/dnsmasq.d,.old, .bak, .tmp
```

È possibile memorizzare le configurazioni host in più file con l'opzione --addn-hosts =.
Dnsmasq include un controllo della sintassi:

```
$ dnsmasq --test
dnsmasq: syntax check OK.
```

Vediamo ora la configurazione del DNS. In generale, i caratteri jolly DNS non sono una buona pratica perché invitano ad abusi. Ma ci sono momenti in cui sono utili, come all'interno dei bei confini protetti della tua LAN. Ad esempio, i cluster Kubernetes sono notevolmente più facili da gestire con il DNS jolly, a meno che non ti piaccia creare voci DNS per le tue centinaia o migliaia di applicazioni. In Dnsmasq un carattere jolly che risolve tutte le richieste a mehxample.com ha questo aspetto:

```
address=/mehxample.com/192.168.0.5
```

L'indirizzo da utilizzare in questo caso è l'indirizzo IP pubblico del tuo cluster. Questo risponde alle richieste di host e sottodomini in mehxample.com, ad eccezione di quelli già configurati in DHCP o / etc / hosts.

Configurazione di un Raspberry pi 3 come access point

Il motivo per usare la Raspberry Pi 3 come un access point è dato dal fatto di avere il pieno controllo della rete e del traffico del proprio custom AP Linux based.Ciò che occorre è:
1 Raspberry pi 3
1 microsd di almeno di 8 GB preferibilmente 32Gb
Installazione di alcuni tool

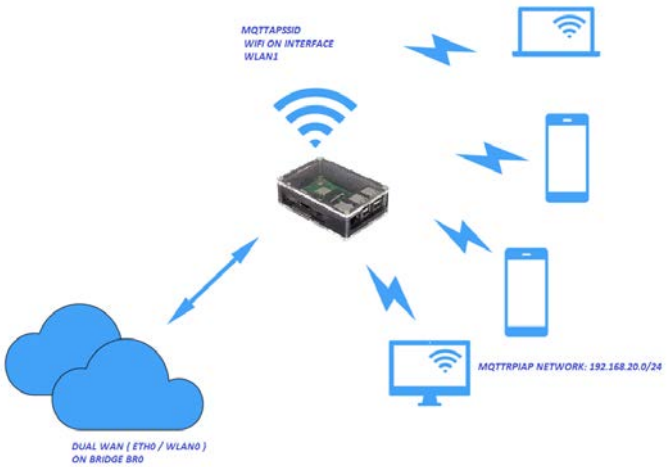
I software necessari per realizzare l'obiettivo sono due

- HostAPD (Host access point daemon) è un software per la gestione dell'interfaccia wireless e anche un server di autenticazione
- DNSMasq è un server DHCP che eroga anche servizi di cache DNS.

Installati i software, procediamo alla configurazione dei servizi installati. Lanciamo i seguenti comandi:

sudo systemctl stop dnsmasq
sudo systemctl stop hostapd

Per procedere alla fase di configurazione ipotizziamo che lo scenario di utilizzo sarà quello più semplice, costituito da un access point che eroga servizio di connettività ad una rete di client il cui indirizzamento sarà 192.168.20.0/24. Alla raspberry assegneremo l'indirizzo statico 192.168.20.1 che per convenzione identifica spesso l'ip del gateway. Come SSID (nome della rete wireless) utilizzeremo : TestReteWifi.



Per configurare l'indirizzo ip statico, da assegnare all'interfaccia wireless, sarà necessario, innanzitutto, identificarne il nome tra quelle disponibili. La raspberry pi 3 ha come caratteristiche di fabbrica una sola interfaccia wireless, che su linux viene indentificata generalemente come wlan0, ma per sicurezza , nel caso in cui desciste di utilizzare, anche, dei dongle wifi, basterà lanciare il comando seguente per assicurarsi di utilizzare l'identificativo di interfaccia corretta.

sudo ifconfig –a

Per configurare l'indirizzo IP statico, da assegnare alla interfaccia della raspberry, andremo a modificare il file di configurazione dhcpcd.
A questo punto sarà sufficiente editare il file dhcpcd.conf, lanciando il comando

sudo vi /etc/dhcpcd.conf

per aggiungere in coda la seguente configurazione:

```
interface wlan1
   static ip_address=192.168.20.1/24
   nohook wpa_supplicant
```

Il servizio DHCP come abbiamo già anticipato sarà erogato da dnsmasq. Di default, la configurazione contiene numerose informazioni, per cui, per semplificazione proveremo a riscrivere da zero un nuovo file, facendo prima un opportuno backup del file originale. Eseguiamo dunque i seguenti comandi per fare il backup e per editare il nuovo file di configurazione:

```
sudo mv /etc/dnsmasq.conf /etc/dnsmasq.conf.orig
sudo nano /etc/dnsmasq.conf
```

Inseririamo le seguenti informazioni nel file dnsmasq.conf e salviamo:

```
interface=wlan1      # Use the require wireless interface - usually wlan0
dhcp-range=192.168.20.2,192.168.20.20,255.255.255.0,24h
```

Il server dhcp a questo punto è già pronto ad erogare ai client gli indirizzi ip, inclusi nel range compreso tra 192.168.20.2 e 192.168.20.20

Per completare la configurazione editiamo il file /etc/hostapd/hostapd.conf ed aggiungiamo i parametri necessari partendo da un file vuoto.

sudo nano /etc/hostapd/hostapd.conf

La configurazione che faremo ipotizza di utilizzare :

- il canale 7 come banda di frequenza;
- SSID : MqttApWifi
- password ##########

La password dovrà avere una lunghezza compresa tra 8 e 64 caratteri, compatibilmente con le specifiche imposte dal protocollo di sicurezza scelto wpa2.
Di seguito, una piccola leggenda delle bande di frequenza dell'AP. Nel nostro esempio useremo la configurazione hw_mode=g.

- a = IEEE 802.11a (5 GHz)
- b = IEEE 802.11b (2.4 GHz)
- g = IEEE 802.11g (2.4 GHz)
- ad = IEEE 802.11ad (60 GHz)

Di seguito la configurazione del file /etc/hostapd/hostapd.conf

```
interface=wlan1
driver=nl80211
ssid= MqttApWifi
hw_mode=g
channel=7
wmm_enabled=0
macaddr_acl=0
auth_algs=1
ignore_broadcast_ssid=0
wpa=2
wpa_passphrase=############
wpa_key_mgmt=WPA-PSK
wpa_pairwise=TKIP
rsn_pairwise=CCMP
```

A questo punto possiamo attivare la configurazione creata, editando il seguente file:

sudo nano /etc/default/hostapd

cercate all'interno la linea contenente questa stringa : #DAEMON_CONF e sostituitela con questa:

DAEMON_CONF="/etc/hostapd/hostapd.conf"

Riavviamo il servizio di rete dhcpcd per riconfigurare l'indirizzo della wlan0:

sudo service dhcpcd restart

Abilitiamo e riavviamo hostapd:

```
sudo systemctl unmask hostapd
sudo systemctl enable hostapd
sudo systemctl start hostapd
```

A questo punto dovremmo già trovare, tra le reti wifi disponibili, quella configurata. Per una verifica sullo stato dei servizi potrete usare i seguenti comandi:

sudo systemctl status hostapd
sudo systemctl status dnsmasq

Sarà già possibile a questo punto far connettere dei client alla rete wifi creata, ma non sarà ancora possibile ad esempio connettersi al di fuori della rete stessa. Cerchiamo di capire come abilitare, dunque, i client alla navigazione internet via raspberry.
Per configurare il routing e il masquerade (NAT) editare /etc/sysctl.conf lanciando il comando:

sudo nano /etc/sysctl.conf

andare alla riga che contiene la stringa seguente e de commentare togliendo il #.

#net.ipv4.ip_forward=1

A questo punto il sistema sarà abilitato a ruotare i pacchetti da un'interfaccia ad un'altra, ma ancora non sarà possibile navigare in quanto gli ip dei client appartengono ad una rete privata, quindi sarà necessario mascherarli/nattarli per consentire il traffico in uscita sull'interfaccia eth0, che supponiamo sia connessa ad esempio ad un router abilitato alla navigazione Internet. Per nattare la rete verso il traffico su eth0 sarà sufficiente eseguire questo comando.

sudo iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE

e per rendere le modifiche definitive basterà creare un file di configurazione eseguendo il seguente comando:
sudo sh -c "iptables-save > /etc/iptables.ipv4.nat"

Aggiungiamo al file /etc/rc.local la seguente riga in modo che al reboot della raspberry la configurazione di iptables venga caricata opportunamente.
iptables-restore < /etc/iptables.ipv4.nat

La raspberry è adesso un access point.