Agenzia nazionale per le nuove tecnologie, l'energia e lo sviluppo economico sostenibile

Ministero della Transizione Ecologica

Ricerca di Sistema elettrico

Consolidamento dell'uso della Piattaforma PELL per i dati statici ed implementazione sezione per la raccolta dati dinamici

Cosma Damiano De Angelis, Pierfrancesco De Angelis

A4E

Report RdS/PTR2020/018

Consolidamento dell'uso della Piattaforma PELL per i dati statici ed implementazione sezione per la raccolta dati dinamici

Cosma Damiano De Angelis e Pierfrancesco De Angelis (arch4energy)

Aprile 2021

Report Ricerca di Sistema Elettrico

Accordo di Programma Ministero dello Sviluppo Economico - ENEA
Piano Triennale di Realizzazione 2019-2021 - II annualità
Obiettivo: Tecnologie
Progetto: Tecnologie per la penetrazione efficiente del vettore elettrico negli usi finali
Work package: Local Energy District
Linea di attività: .28 - Consolidamento dell'uso della Piattaforma PELL per dati statici ed implementazione sezione per la raccolta dati dinamici
Responsabile del Progetto: Claudia Meloni, ENEA
Responsabile del Work package: Claudia Meloni, ENEA

Il presente documento descrive le attività di ricerca svolte all'interno del Contratto "Progettazione software PELL Edifici
Responsabile Unico del Procedimento ENEA: Claudia Meloni (ENEA)
Responsabile del Contratto per il Contraente: Cosma Damiano De Angelis (arch4energy)

Indice

Sommario

La piattaforma PELL è una piattaforma di tipo smart city as-a-service, la cui architettura generale definisce il recupero dei dati da diverse infrastrutture e gestori e la creazione di una serie di servizi per gli utenti finali. Costituisce il punto di accesso per tutti gli utenti finali legati al progetto PELL (Public Energy Living Lab), fornendo l'entry point di registrazione, accesso al caricamento dati ed ai servizi correlati.
Il ruolo centrale della piattaforma è consentire il caricamento dei dati statici, ovvero le schede censimento relative allo stato corrente degli edifici, e dei dati dinamici, ovvero i dati relativi alle misure elettriche e termiche acquisite dai meters.
Le attività che sono state svolte sono due delle quattro previste dal contratto TERIN/2020/226, ovvero:

- *la progettazione del portale per consentire l'interfacciamento degli utenti al PELL Edifici e la fruizione dei servizi in esso integrati*
- *la progettazione e la realizzazione della struttura di comunicazione della piattaforma UBD*

Il primo task è consistito nella progettazione, quindi nello svolgimento delle varie analisi, del portale nei suoi componenti fondamentali. In considerazione delle applicazioni preesistenti, come il portale PELL IP, ed in considerazione delle condizioni al contorno sotto il profilo delle tecnologie software attuali nonché delle interazioni con le fonti dei dati, si è proceduto alla scelta di architetture congruenti. In conseguenza, si è operato la scelta delle tecnologie software in grado, sia di assicurare la realizzabilità di quanto delineato, sia di confermare la scalabilità in termini applicativi. Nel documento sono quindi esplicitate tutte le scelte progettuali effettuate, in modo da consentire alla parte realizzativa di operare in modo univoco.
Il secondo task è consistito nella progettazione della architettura necessaria a far fronte alla potenziale crescita esponenziale della richiesta di servizi di invio dati da parte dei partners, tenendo altresì conto delle architetture – in termini di protocolli di comunicazione – attuali della infrastruttura UBD.
Oltre alla progettazione, si è poi proceduto alla realizzazione di tali architetture in modo da consentire ad ENEA di porre in esercizio i sistemi dedicati a questa funzione. Nel documento quindi sono riportate sia le componenti progettuali, sia quelle realizzative, fino alle istruzioni operative per operare sui sistemi ENEA dedicati.

1 Introduzione

Fra i diversi ambiti in cui ENEA sta operando in vista della ottimizzazione dell'utilizzo dell'energia a livello nazionale, è previsto lo sviluppo di un sistema che ha come oggetto gli edifici di pubblica proprietà.
Facendo seguito al progetto PELL relativo alla pubblica illuminazione, grazie al quale è stata implementata una piattaforma Big Data denominata UBD, ed una serie di servizi applicativi verticali, è del tutto consistente pensare ad uno sviluppo di sistema, per gli edifici, che faccia tesoro di tale esperienza.
Arch4energy ha partecipato alla realizzazione del sistema attuale, nell'ambito di progetti precedenti, come riportato in dettaglio nel documento di gara richiesto per le esperienze precedenti, specifiche per il progetto PELL.
Il progetto PELL Edifici vedrà dunque, tra gli elementi costitutivi, quelli che si possono definire come i trasposti delle componenti PELL IP, utilizzando peraltro l'infrastruttura ICT, sia per gli aspetti di comunicazione che di storage ed elaborazione parallela, di cui è previsto il necessario potenziamento.
L'infrastruttura che attualmente accoglie i dati relavit al PELL IP è denominata Urband Big Data, costituita da un cluster Hadoop attualmente in esercizio.
Per accogliere la nuova applicazione Pell Edifici, è quindi necessario intervenire – tra l'altro – sul sistema UBD, per consentirne l'utilizzo e la separazione logica da quello che sinora è stata l'infrastruttura di PELL IP.
Il presente documento è relativo al rilascio dei due deliverables che seguono:

- Progettazione Portale PELL Edifici
- Infrastruttura di comunicazione

2 Progettazione Software Portale PELL edifici

Il portale PELL edifici - in modo analogo al PELL Illuminazione Pubblica - ha come scopo di fondo la conoscenza degli edifici pubblici dal punto di vista delle infrastrutture energetiche, al fine di consentire una gestione consapevole delle componenti energetiche. In realtà per arrivare alla conoscenza della parte energetica, si deve considerare la conoscenza dell'edificio dal punto di vista architettonico, che diventa dunque un ulteriore elemento di valore in termini di informazioni del patrimonio pubblico.
Tale conoscenza è articolata su due piani principali, il primo relativo alla architettura dell'edificio, il secondo sulla raccolta dati, in tempo reale, relativi ai consumi e produzione. Alcuni di queste informazioni saranno poi fondamentali per lo sviluppo della componente LENI, di cui in una ulteriore fase del progetto.
L'esperienza maturata da arch4energy nelle varie fasi di realizzazione del portale PELL IP assume una importanza fondamentale per la trasposizione verso il portale PELL edifici. Nella fase di progetto è stata cura, da parte di arch4energy, effettuare scelte di piattaforma che tengano conto della attuale piattaforma PELL IP.
Nel progettare il portale Edifici si è tenuto conto del fatto che non è possibile semplicemente "clonare" il sistema PELL IP verso quello Edifici. Ciò avviene non solo per le differenze formali fra i due ambiti, uno che tratta della Pubblica Illuminazione ed uno che tratta di Edifici civili - come scuole, ospedali, ecc. – ma anche perché gli strumenti di sviluppo software si evolvono nel tempo e dunque anche nella scelta progettuale ha tenuto conto di tali evoluzioni. L'obiettivo che arch4energy si è proposto è quello di progettare PELL Edifici in modo da porre le basi per arrivare ad un portale logicamente unico, che possa ospitare i diversi verticali PELL, attuali e futuri. Un aspetto importante di cui si è tenuto conto è l'integrazione del portale con basi di dati disponibili da altre amministrazioni, come ad esempio le regioni.
La progettazione del portale da parte di arch4energy è stata eseguita in relazione a diverse dimensioni e sotto i diversi profili sia nel metodo che, dove richiesto, nel merito.
Gli aspetti salienti considerati sono stati:

- Condizioni al contorno
- Framework sofware
- Gestione utenti
- Schede censimento

2.1 Condizioni al contorno

Il portale Pell Edifici deve, in continuità logica con Pell IP, provvedere ad una piattaforma dove possano convergere tutti gli attori che partecipano alla gestione di edifici di pubblica utilità. In tal senso è necessario ragionare non solo sugli aspetti implementativi immediati, ma anche sulla ricaduta che scelte tecniche specifiche possono avere in prospettiva. Stante la grande variabilità di edifici pubblici, si pensi ai Ministeri, agli Ospedali, alle Scuole di ogni ordine e grado, alle Caserme, a quelli relativi agli organi di sicurezza, e così via, pensare ad un portale Edifici significa in realtà pensare ad un portale di portali. Ogni portale sarà relativo ad una tipologia di edifici, con le sue caratteristiche, alcune peculiari ed alcune condivise con gli altri. Per quanto motivo ENEA inizia ad affrontare il tema partendo dal portale delle Scuole. Anche in questo caso si entra in una complessità non banale, in quanto le scuole sono in Italia un organismo complesso, afferente nei suoi componenti a parti diverse dello Stato e non, come nel caso delle scuole private. La progettazione del portale edifici-scuole, è quindi da vedere come inquadrato in un ambito più ampio in cui le scelte siano non preclusive allo sviluppo di portali in parte simili, in parte dissimili a quello scuole.
Una prima ricaduta di queste condizioni al contorno è la scelta della tecnologia software relativa al framework di realizzazione.

2.2 Framework software

Il portale Pell Edifici deve, in continuità logica con Pell IP, provvedere ad una piattaforma dove possano convergere tutti gli attori che partecipano alla gestione di edifici di pubblica utilità. In tal senso è necessario riflettere non solo sugli aspetti implementativi immediati, ma anche sulla flessibilità del framework per gli

sviluppi futuri. Poiché altri portali saranno implementati in relazione alle diverse tipologie di edifici pubblici, alcune scelte di fondo assicurano la espandibilità del sistema scuole alle future esigenze.

2.2.1 Front End

Per quanto attiene al sistema di Front End, si è optato per il sistema Angular (https://angular.io/), basandosi su framework Metronic. Questa scelta è stata operata in termini di confronto con le molte possibilità che la tecnologia software offre oggi. Delle tante possibili alternative è stato considerato il framework utilizzato nel portale PELL IP, ovvero linguaggio php insieme con gli strumenti ad esso correlati, Laravel o Code Igniter. La condizione al contorno che è stata considerata preminente è quella di una scelta orientata ad una modernità che il php, pur potentissimo ed ampiamente utilizzato, non può offrire, datando al sua nascita nel 1994. Questa scelta comporta ovviamente una soluzione di Back End congruente.

2.2.2 Back End

Il Back End che si sposa - di fatto - con gli elementi scelti per il FE è il Nodejs, già peraltro utilizzato in altri progetti ENEA e quindi immediatamente gestibile dai ricercatori coinvolti.

2.2.3 I Database

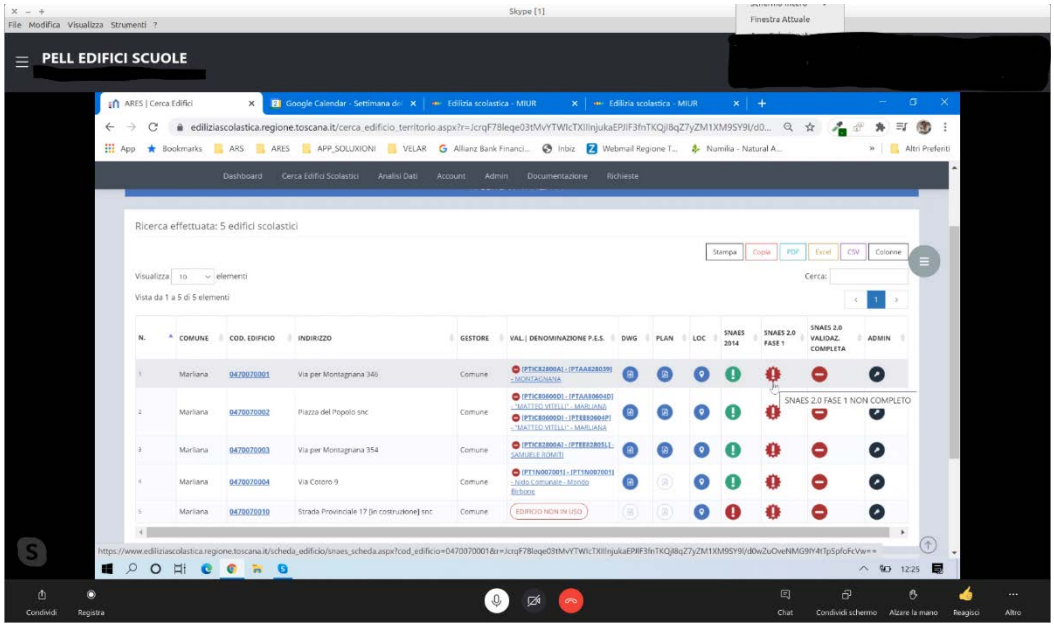
Dal punto di vista dei sistemi di gestione dati si è deciso di utilizzare i due componenti ormai standard nel mondo IT, ovvero un RDBMS (tipo mysql) per gli utenti e per i dati risultati di calcoli, mentre per l'ingestion dei dati massivi si è optato per la tecnologia NoSql. La complessità delle schede censimento delle scuole, così come degli altri edifici futuri, si sposa perfettamente con il concetto di "documento" NoSql, invece che con il concetto di tabelle in relazione fra loro. La struttura a lista concatenata tipica delle schede, che descrivono l'edificio scolastico – cosi come gli edifici in genere - in tutte le sue ramificazioni, è perfettamente gestita dalla logica NoSql, permettendo di "leggere" l'intero edificio tramite un file, come vedremo XML, in un solo statement.

2.3 Gestione Utenti

Il portale Pell Edifici deve, in continuità logica con Pell IP, provvedere ad una piattaforma dove possano convergere tutti gli attori che partecipano alla gestione di edifici di pubblica utilità. In tal senso è stato eseguito un lavoro di analisi che fa tesoro della esperienza maturata nel progetto PELL IP. Gli utenti che devono essere previsti sono di diverse tipologie: vanno dal cittadino, che si iscrive a livello personale, fino al gestore degli impianti tecnologici, dal rappresentate dell'ente locale fino al rappresentante di quello nazionale, e quanti altri si potranno individuare. In questa sede di progetto ciò che occorre è l'individuazione di una metodologia di creazione e gestione utenti che possa poi accogliere tutti quelli che si renderanno necessari in funzione del tipo di edificio che si andrà a gestire. Oggi consideriamo le scuole, domani gli ospedali o altro, per cui gli utenti saranno diversi. Infine si è considerato necessario prevedere un tipologia specifica di utente, ovvero il delegato. Qualsiasi siano i privilegi di utente, per privilegi si intende qui la lista dei permessi di creare, modificare, cancellare e visualizzare i dati, l'esperienza ha mostrato come spesso i titolari di privilegio debbano poter delegare qualcuno ad operare per loro conto. E' però necessario – per ovvi motivi di sicurezza e privacy - che rimanga evidente chi ha operato sul sistema. Quindi nella progettazione della fase "gestione utenti" si inserirà questa figura di "Utente delegato" che deve essere si creato, ma deve essere autorizzato dall'utente di riferimento. E' quindi previsto che nello sviluppo della sezione "Utenti" del portale edifici si possa definire sia utenti "titolare", che uno o più utenti da questo "delegati".

2.4 Schede censimento

La complessità delle schede censimento delle scuole è stata ampiamente valutata, anche grazie alla condivisione della esperienza ARES, Anagrafe Regionale Edilizia Scolastica, sulla edilizia scolastica regionale. A puro titolo di esempio viene riportata una scheda di ARES Toscana,



La possibilità di esaminare il portale realizzato per la gestione della edilizia scolastica ha portato alla evidenza della complessità notevolissima che gli edifici scolastici comportano, e di conseguenza di valutare la migliore struttura per le schede censimento. La struttura dei dati è riconducibile a quello che viene generalmente indicata come "lista concatenata" e ciò porta certamente in direzione dei database di tipo NoSql, invece che verso il concetto di tabelle in relazione fra loro, come nei classici RDBMS. La struttura a lista concatenata tipica delle schede, che descrivono l'edificio scolastico in tutte le sue ramificazioni, è perfettamente gestita dalla logica NoSql, permettendo di "leggere" l'intero edificio tramite un file un solo statement. Il formato del file di censimento è stato scelto – in continuità con il progetto PELL IL – essere XML. Nel caso di PELL IP la gestione dei files XML, sempre per gestire le consistenze, ha portato allo sviluppo di metodologie di verifica, sia formale che di merito, riutilizzabili nel caso delle scuole. Dal punto di vista delle tecnologie software, si è optato per l'utilizzo del package Elasticsearch – facente parte della suite Apache - già installato nella piattaforma UBD e idoneo sia in termini di capacità elaborativa sia in termini di scalabilità. La scalabilità è ormai un tema costante e fondamentale degli sviluppo ENEA, in quanto il PELL ha un orizzonte nazionale, sia già per la Pubblica Illuminazione, sia per gli edifici scolastici. I files XML assicurano grande facilità di conversione nei formati più utilizzati nell'ambito dei sistemi territoriali, come i Json e gli Shapefile.

2.5 Calcolo e gestione KPI

La complessità delle schede censimento delle scuole è stata ampiamente valutata, anche grazie alla condivisione della esperienza ARES, Anagrafe Regionale Edilizia Scolastica, sulla edilizia scolastica regionale. Il problema che si è posto è quello relativo alla individuazione degli insiemi intersezione fra le informazioni disponibili in ARES con quelli necessari ad ENEA per la completa descrizione delle consistenze. In questo senso, considerando che le schede che ARES può mettere a disposizione di PELL Scuole possono mancare dei dati necessari ai calcoli delle KPI individuate da ENEA, si è proposto di predisporre un meccanismo di gestione di valori paralleli delle stesse variabili. In altri termini, siccome può essere necessario consentire all'utente ENEA di valorizzare campi che dovrebbero essere già compilati da ARES, ma che sono invece mancanti, è necessario altresì gestire questi parallelismi ed individuare eventualmente un processo di "riconciliazione", come spesso avviene nelle grandi basi di dati. Si intende con riconciliazione, il processo – necessariamente off line – che ciclicamente esamina le schede e determina quale sia il valore da considerare valido in ultima analisi. Questa necessità comporta una scelta ulteriore in relazione al calcolo delle KPI. Poiché i valori delle variabili sulla base delle quali le KPI vengono calcolate può – per quanto su delineato – variare nel tempo, allora la soluzione è quella di creare processi batch iterativi i quali, partendo dalla lettura delle schede, possano consolidarle in ottica di riconciliazione, ed immediatamente appresso possano ricalcolare le KPI

opportune. Tutto ciò può certamente essere implementato pensando ad una lettura delle schede dal DB NoSql, operazione molto veloce perché sequenziale, ad un calcolo delle KPI ed infine ad una registrazione delle KPI stesse nel database Sql. Questo processo consente – dal punto di vista del software di Front End – di accedere velocissimamente a valori già pronti nel DB. Un tale operazione diventa velocissima, come deve essere una operazione on line nella quale l'utente effettua la richiesta per vederla immediatamente esaudita. In altri termini i task che portano dalla lettura/inserimento dei dati alla disponibilità delle KPI tramite interfaccia Web, non avvengono il real-time alla richiesta dell'utente, ma vengono preparati tramite processi batch off-line. Considerando la potenziale crescita del numero dei dati, sia in termini statici che dinamici, una simile architettura assicura la scalabilità assoluta del Portale Edifici.

3 Infrastruttura di Comunicazione

3.1 Stato attuale

Nell'ambito del progetto PELL IP, si sono realizzati due canali di comunicazione, uno relativo ai dati statici di Censimento, ed uno relativo ai dati dinamici costituiti dalle letture dei parametri elettrici in tempo reale. In relazione ai dati dinamici, Arch4enegy ha realizzato nei precedenti progetti PELL, una infrastruttura di comunicazione basata sul protocollo MQTT e quindi in termini di broker in logica publish/subscribe. Nell'ottica della politica ENEA, di ricorso a sistemi Open Source, è stato utilizzato il ben noto pacchetto "mosquitto" su sistema operativo Linux. Questo componente infrastrutturale diventa oggi critico in relazione alla attesa di crescita del traffico dovuto alla messa in esercizio sia di PELL IP che Pell Edifici.

3.2 Necessità evolutive

E' dunque necessario potenziare la capacità del canale, e soprattutto di predisporsi ad ulteriori aumenti laddove il carico dovesse aumentare oltre una certa soglia.
In altri termini è necessario realizzare una infrastruttura di comunicazione totalmente scalabile, armonizzandola alla infrastruttura UBD, capace anch'essa di scalabilità virtualmente illimitata.
E' necessario inoltre configurare il servizio in modo da renderlo congruente con le infrastrutture di autenticazione di ENEA.

3.3 Autenticazione

Per poter rendere dinamica e quindi integrabile con la piattaforma SCP (Smart City Platform), l'autenticazione degli utenti per l'invio dei dati al broker mosquitto si è scelto di integrare il plugin mosquitto-auth-plug disponibile in open source sulla piattaforma github, all'indirizzo https://github.com/jpmens/mosquitto-auth-plug.
Questo plugin può eseguire l'autenticazione, ovvero controllare il nome utente/password, e l'autorizzazione, ovvero concedere il permesso di iscriversi e/o pubblicare su argomenti specifici tramite ACL. Per quanto riguarda i back-end supportati si può fare riferimento alla tabella (semplificata) che segue:

Capability	**cdb**	**files**	**http**	**jwt**	**ldap**	**mongo**	**mysql**	**postgres**	**psk**	**redis**	**sqlite**
authentication	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
superusers			Y	Y		Y	Y	Y			
acl_checking		Y	Y	Y		Y	Y	Y		1	
static_superusers	Y	Y	Y	Y		Y	Y	Y		Y	Y

Si rimanda, per ogni ulteriore dettaglio, alla documentazione ufficiale del plugin.
Il plugin consente di utilizzare anche diversi metodi contemporaneamente.
La configurazione di mosquitto è costituita da un file di testo – come di consueto nei sistemi unix/linux - quindi in questo modo si può estendere il metodo di autenticazione ACL ad altre modalità, in linea con le metodologie disponibili.

3.3.1 Sviluppo in Laboratorio Arch4energy

Muovendosi, per scelta di fondo, nel mondo open source, il componente va usato "As is", ovvero così come'. Sta all'utilizzatore adattare quanto disponibili alle proprie necessità tecniche. Ciò comporta un lavoro di esame del codice e la individuazione di eventuali modifiche software per consentire la compilazione del plugin in modo congruente alla condizioni al contorno.
Per poter operare in piena sicurezza arch4energy ha messo a disposizione un proprio sistema di laboratorio configurato esattamente come il pellbroker di ENEA, sia dal punto di vista del sistema operativo CENTOS, sia in relazione alla versione mosquitto.
Le operazioni di analisi, troubleshooting e modifica, che sono state compiute sono le seguenti:

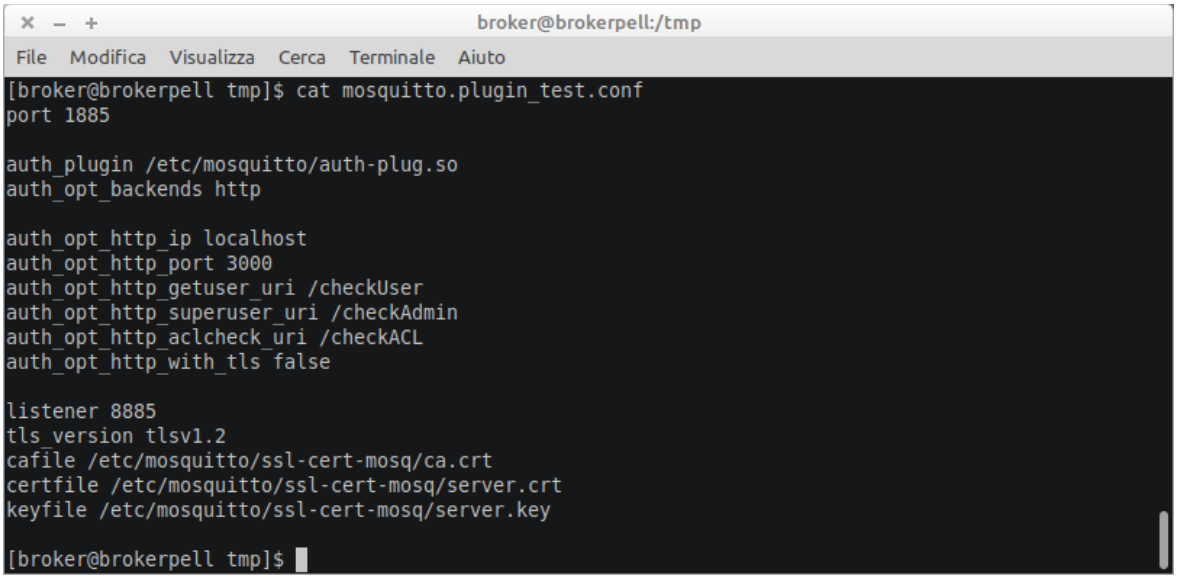
- Si è effettuata la compilazione da sorgente sia di mosquitto che del plugin;
- Si sono riscontrati diversi errori non recoverabili, per cui si è proceduto ad una analisi di tali errori per comprendere come superarli;
- Da qui è stato necessario modificare il Makefile, che si è mostrato fare riferimento a liberie openssl non corrette;
- Sono stati individuati pacchetti da aggiungere al sistema operativo, in particolare:
    - openssl-devel
    - c-ares-devel
    - libwebsockets-devel.x86_64
    - libwebsockets.x86_64
    - libuuid-devel
    - openssl-devel.x86_64
    - install openssl-libs.x86_64
    - install libcurl-devel.x86_64

- Si è verificato che per compilare il plugin ci fosse bisogno di ricompilare mosquitto ed installarlo, facendo si che il compilatore potesse trovare il file config.h e tutti i files necessari alla corretta compilazione
- Si è verificato che la versione di mosquitto capace di dare risposta correttamente alle richieste di compilazione, per motivi di compatibilità delle librerie openssl, era la 1.3.5;
- Si è quindi proceduto a scaricare la mosquitto-1.3.5 ed eseguire tutti i task di compilazione ed installazione;
- Solo a questo punto si è potuto compilare correttamente il plugin mosquitto-auth-plug;
- Al termine della compilazione si ottenuta la libreria che contiene il componente utilizzabilte di autenticazione, ovvero il file auth-plug.so.
- A questo punto diventa possibile configurare nel file di configurazione di mosquitto il richiamo al path della liberia appena compilata.

Una volta realizzata con successo la modifica in laboratorio, è stato possibile, senza alcun impatto, effettuare le stesse operazioni sul sistema in esercizio.

3.3.2 Configurazione brokerpell

Le attività di troubleshooting e di compilazione dei packages per l'autenticazione, hanno consentito di configurare il brokerpell in modo da utilizzare l'infrastruttura ENEA.
Nello screenshot un esempio di file di configurazione mosquitto che utilizza il plugin oggetto dello sviluppo.
La configurazione di mosquitto è estremamente flessibile, per cui si può utilizzare sia il plugin nella forma "auth_plugin /etc/mosquitto/auth-plug.so", oppure si può usare un normale file ACL (Access Control List) per operare in modo indipendente dalla infrastruttura di autenticazione centrale.
Questa seconda modalità è estremamente flessibile in una fase di implementazione e tuning come quella attuale.



3.4 Potenziamento del canale di comunicazione

3.4.1 Premesse

Il broker mqtt è un componente fondamentale della infrastruttura di comunicazione del progetto PELL. Tutti i dati dinamici dei portali, oggi di PELL IP, a breve di PELL edifici, e poi di tutti gli altri, vengono ricevuti tramite broker che utilizzano il protocollo mqtt.

I dati dinamici sono previsti in formato XML e possono essere di dimensioni fino a centinaia di MB. I files vengono inviati da chi rileva i dati, ad esempio i gestori degli impianti, in modalità completamente asincrona. L'infrastruttura UBD Urban Big Data – che è costituita da un cluster Apache Hadoop/Spark - è stata creata per accogliere e gestire una quantità di dati virtualmente il cui limite è dato soltanto dalla grandezza dello spazio disco a disposizione e dal numero di server di elaborazione parallela disponibili. Si richiede dunque un canale di comunicazione capace di gestire tale flusso di dati.

Come noto la logica del broker è quella del pub/subscribe, ovvero ogni utente si collega inviando i dati su un canale (topic) per il quale è autorizzato. Ogni qualvolta si effettua la connessione, i dati vengono scaricati per essere poi registrati ed elaborati.

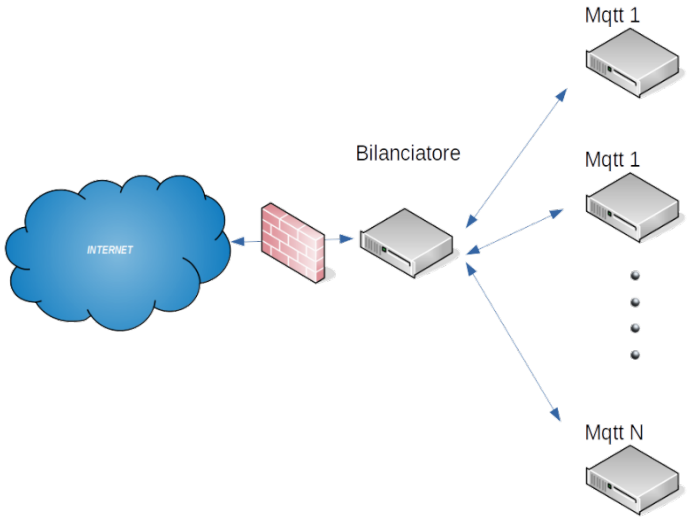
Il problema della scalabilità del canale si presenta immediatamente per il fatto che il momento dell'invio e la quantità di dati per ogni collegamento può essere qualsiasi.

Nelle reti tcp/ip le connessioni avvengono tramite socket, che possono essere aperti contemporaneamente, fino ad un numero definito dalla configurazione del sistema operativo. Ciò fa si che un solo broker possa inizialmente gestire un certo numero di connessioni contemporanee. Il problema è che la durata di tali connessioni dipende, fra l'altro, dalla quantità di dati passanti. Infine se il broker effettua anche un qualche tipo di elaborazione dei dati, si determina un rallentamento complessivo e quindi si raggiunge un limite oltre il quale le nuove richieste di connessione possono essere rifiutate.

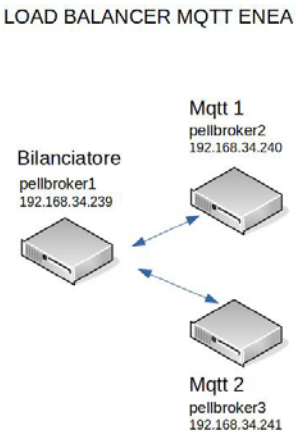
Il tema è dunque quello di creare un canale MQTT di capacità "scalabile", nel senso che si possa aumentarne la capacità ricettiva e computazionale al bisogno.

3.4.2 Architettura proposta: Mqtt Load Balancer

Attualmente esiste un unico broker, il sistema "pellbroker", con indirizzo interno 192.168.34.95, e con indirizzo pubblico "pellbroker.enea.it". Il sistema è configurato per ricevere dati da alcuni utenti autorizzati. Per ottenere la scalabilità richiesta, la soluzione proposta da Arch4energy è la realizzazione di una architettura di tipo "Load Balancer", la cui struttura logica è indicata nello schema in figura:



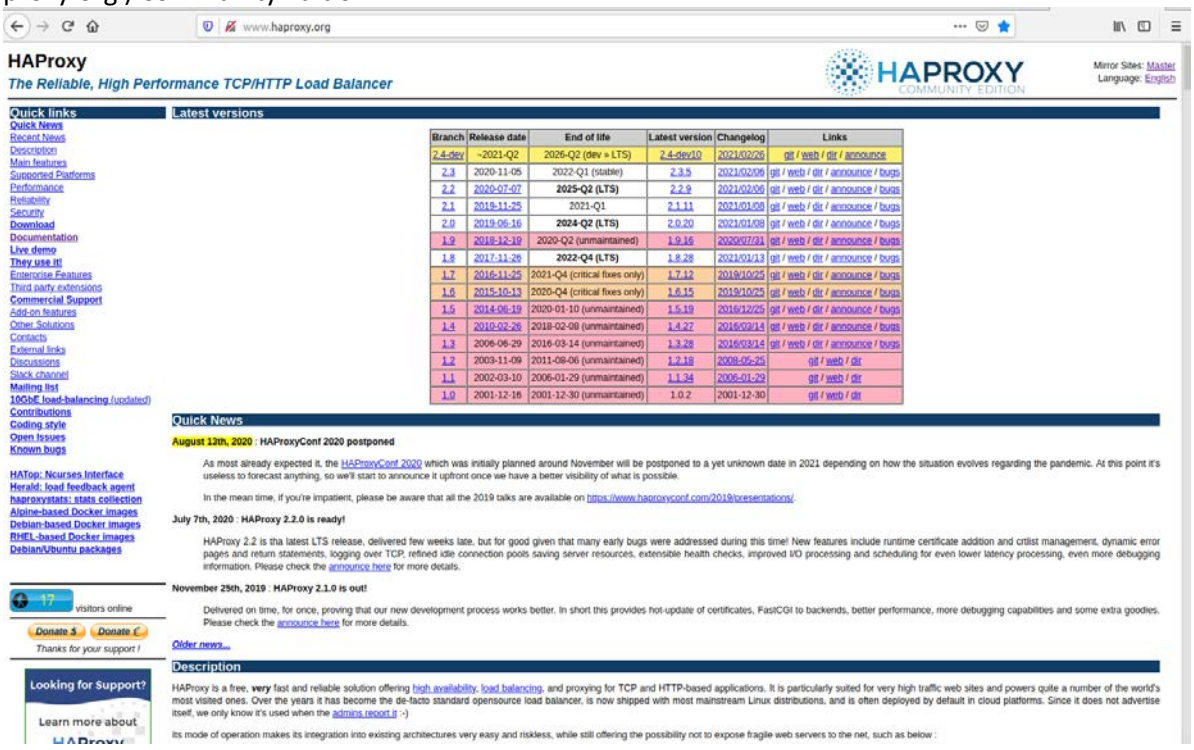
L'architettura prevede un server che riceve le connessioni – i socket tcp/ip – su un indirizzo IP e le smista ad un certo numero di server mqtt veri e propri. Il server di ricezione, anche detto "Bilanciatore di carico", espone la porta TCP che il client si aspetta, effettua la connessione socket, e gira immediatamente la connessione ad uno dei server di back-end.
L'algoritmo con cui il load balancer sceglie il server di back-end a cui girare la connessione può essere di diverso tipo. Ve ne sono di deterministici, ovvero basati su regole, oppure dinamici, basati sul carico istantaneo dei server di back-end. Gli algoritmi deterministici sono molto veloci, poiché non hanno bisogno di elaborare informazioni, quelli dinamici sono più lenti ma sono vantaggiosi quando il tempo di elaborazione richiesto ai server di back-end è rilevante.
Nel caso del protocollo mqtt, l'algoritmo più semplice – il cosiddetto "round robin" si mostra anche il più efficace, e per questo è stato scelto nella configurazione per ENEA. Esso consiste nel delegare a rotazione i server di back-end, ovvero passare la connessione da uno all'altro seguendo sempre lo stesso ordine.
Nel caso ENEA l'architettura prevede inizialmente due server di back-end, ma la peculiarità della soluzione realizzata consente l'aggiunta di server mqtt di back-end in tempi brevissimi.
Lo schema della architettura implementata in ENEA è la seguente:



I sistemi sono posizionati tutti sulla rete interna 192.168.34.0/24, e ciò conferma la flessibilità della soluzione che non obbliga ad una configurazione del bilanciatore su reti diverse.

3.5 *Realizzazione Load Balancer Base*

La funzionalità di Load Balancer può essere realizzata, come di consueto nel mondo ICT, con prodotti diversi. L'elemento centrale è il bilanciatore di carico per il quale esistono prodotti hardware e/o software di mercato ed Open Source. In considerazione della politica ENEA che privilegia l'uso di soluzioni Open, arch4energy ha realizzato l'architettura di bilanciamento di carico mqtt utilizzando il pacchetto open "haproxy" (haproxy.org) Community Edition.



Il pacchetto selezionato offre la funzionalità di bilanciatore di carico per qualsiasi protocollo tcp/ip, quindi in prospettiva potrebbe essere utilizzato anche per realizzare altri sistemi scalabili nell'ambito dei progetti UBD

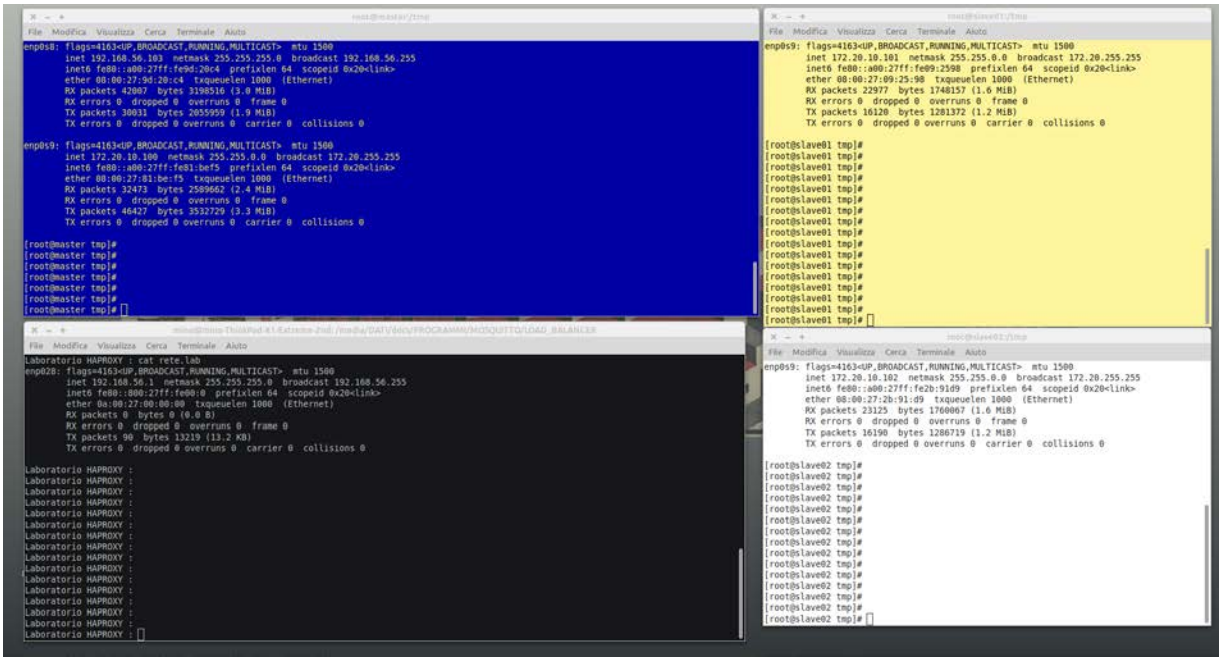
di ENEA. Prima di implementare la soluzione in ENEA, arch4energy ha operato nel proprio laboratorio. La prima configurazione realizzata è quella base, ovvero quella che lavora su protocollo mqtt non criptato. Dal punto di vista TCP ciò comporta l'utilizzo della porta 1883, (well known port) standard per il protocollo mqtt.

3.5.1 Configurazione di Laboratorio

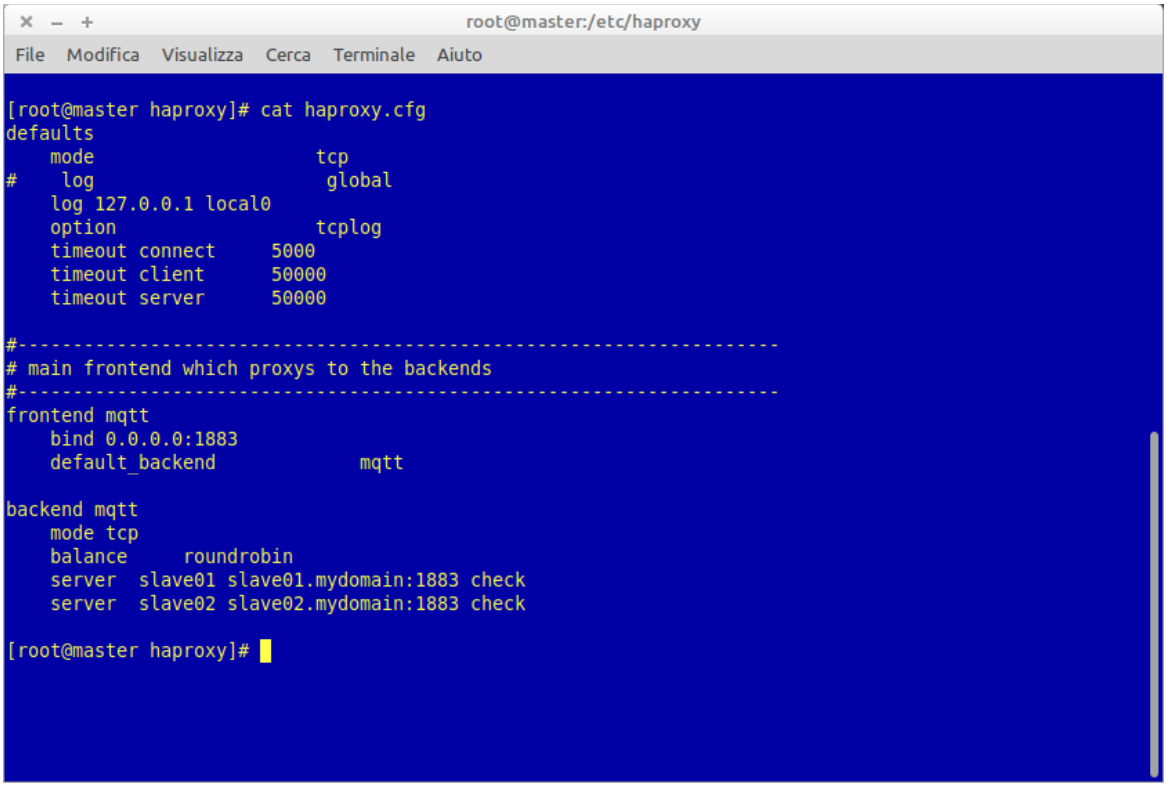
Come di consueto, per non operare direttamente sui sistemi ENEA, arch4energy ha installato e configurato tre server, con sistema operativo CENTOS identico a quello dei sistemi di destinazione. I sistemi coinvolti per l'installazione ed il test sono i seguenti:

- Client: "Laboratorio HAPROXY" con indirizzo 192.168.56.1
- Bilanciatore: "master" con doppio indirizzo 192.168.56.103 e 172.20.10.100
- Mqtt1: "slave01" con indirizzo 172.20.10.101
- Mqtt2: "slave02" con indirizzo 172.20.10.102

La configurazione prevede quindi il bilanciatore con doppia interfaccia, per mettersi in una condizione più complessa rispetto a quella di ENEA, in cui sia il bilanciatore che gli slave sono sulla stessa rete. In figura lo screenshot dei sistemi visti nei terminali di collegamento:



La configurazione di haproxy è situata in /etc/haproxy/haproxy.conf del master ed è la seguente:



Dal punto di vista mqtt, i due server slave01 e slave02 sono stati configurati con il software mosquitto che, come in ENEA, implementa il servizio mqtt di ricezione dei files dei dati dinamici di consumo elettrico.

3.5.2 Test di Laboratorio

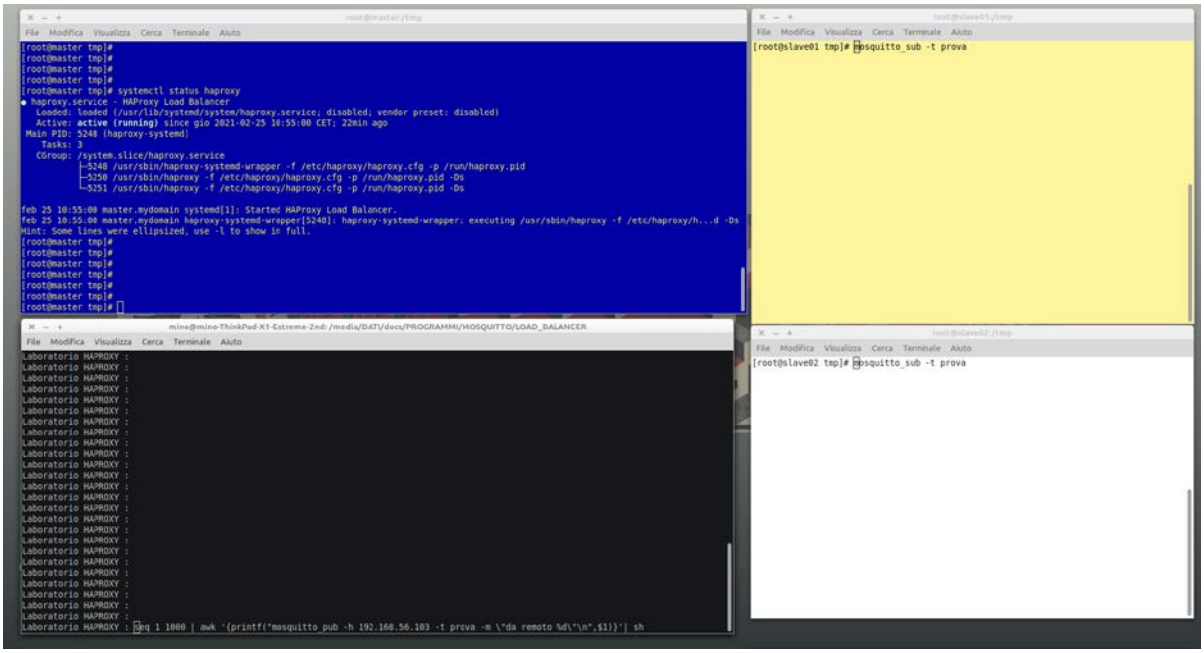
Il test di laboratorio, dopo la installazione e configurazione è consistito nella esecuzione dei passi seguenti:

- partenza del servizio haproxy sul master
- partenza dei servizi mqtt sugli slave
- messa in ascolto su ciascuno slave sul topic “prova”
- invio da parte del client di pacchetti dati numerati

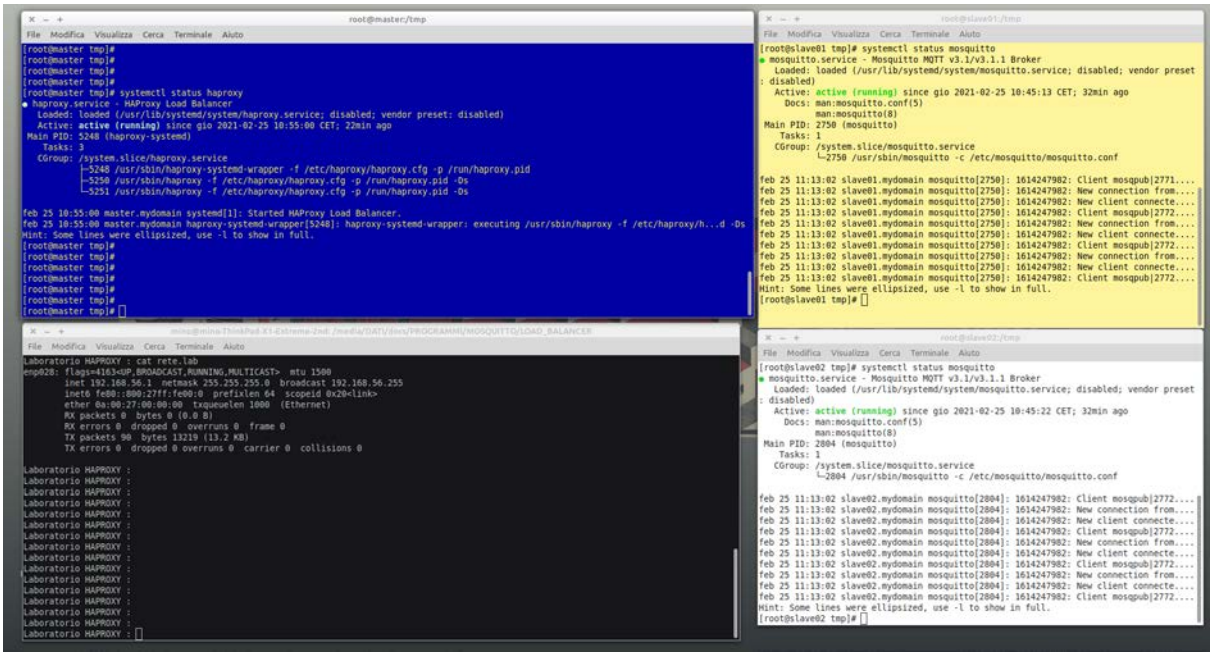
Risultato Atteso: ricezione di pacchetti pari su uno slave e dei dispari sull’altro slave

Negli screenshot che seguono sono riportati i momenti relativi alle varie fasi.

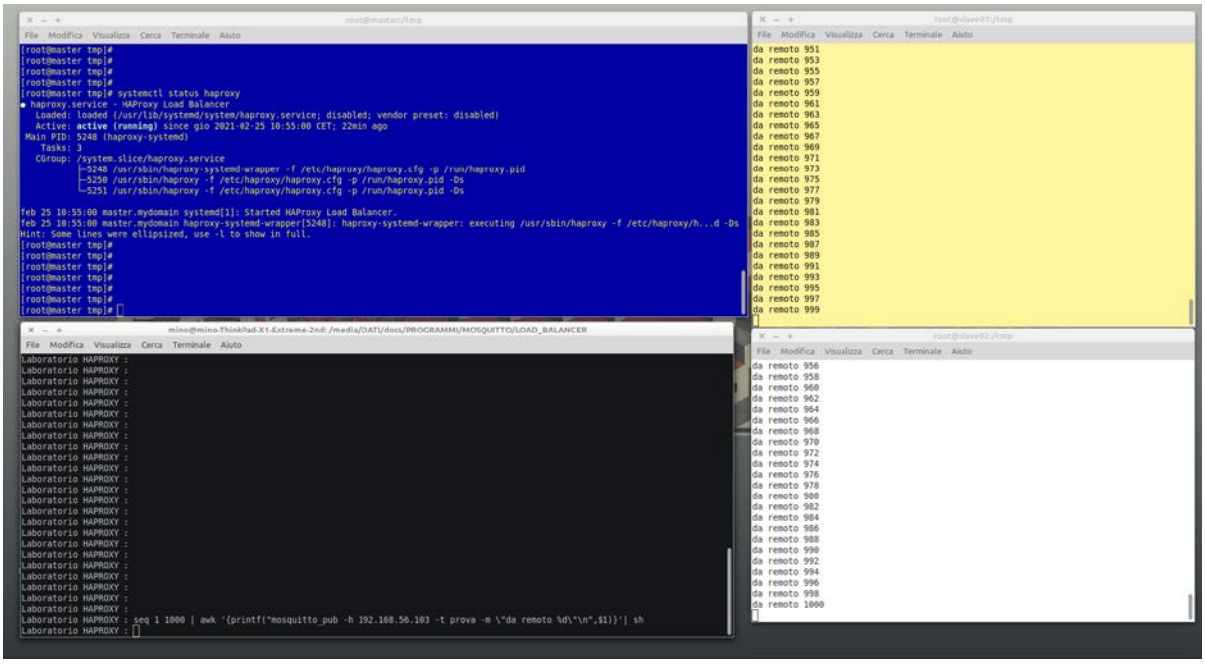
PARTENZA SERVIZI



VERIFICA DEI SERVIZI ATTIVI



INVIO PACCHETTI DAL CLIENT E VERIFICA RICEZIONE



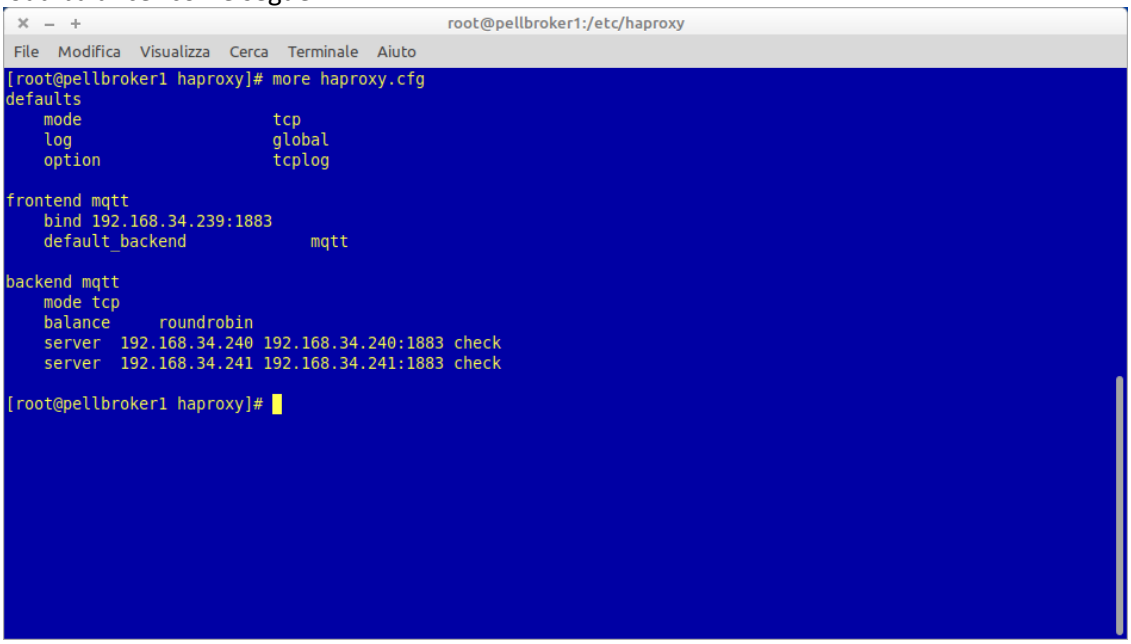
Da questo ultimo screenshot si verifica come i pacchetti vengano ricevuti e correttamente distribuiti sui due server in ricezione.
A questo punto si è proceduto alla implementazione sui sistemi ENEA.

3.5.3 Configurazione ENEA

I tre sistemi a disposizione sono stati installati in modo analogo al laboratorio.

- Bilanciatore: “pellbroker1” con 192.168.34.239
- Mqtt1: “pellbroker2” con indirizzo 192.168.34.240
- Mqtt2: “pellbroker3” con indirizzo 192.168.34.241

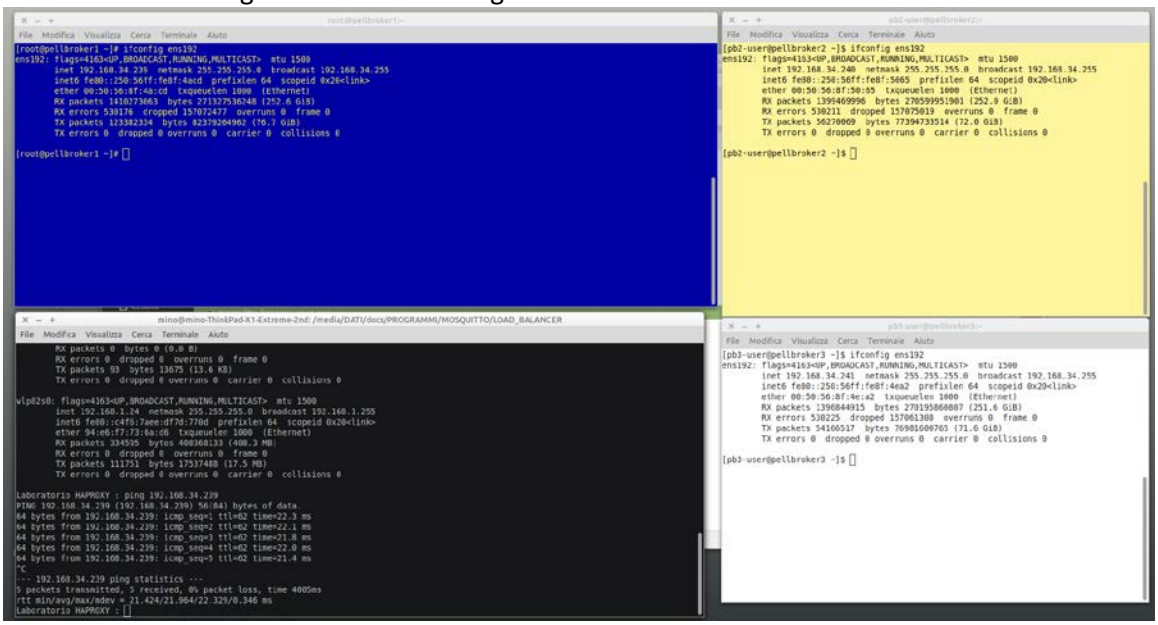
La configurazione di rete diversa, è più semplice in questa fase, del laboratorio, comporta una configurazione del load balancer come segue:



Anche in questo caso si vede come la porta esposta dal balancer sia la stessa, 1883 standard mqtt, esposta ovviamente anche dai server mosquitto.

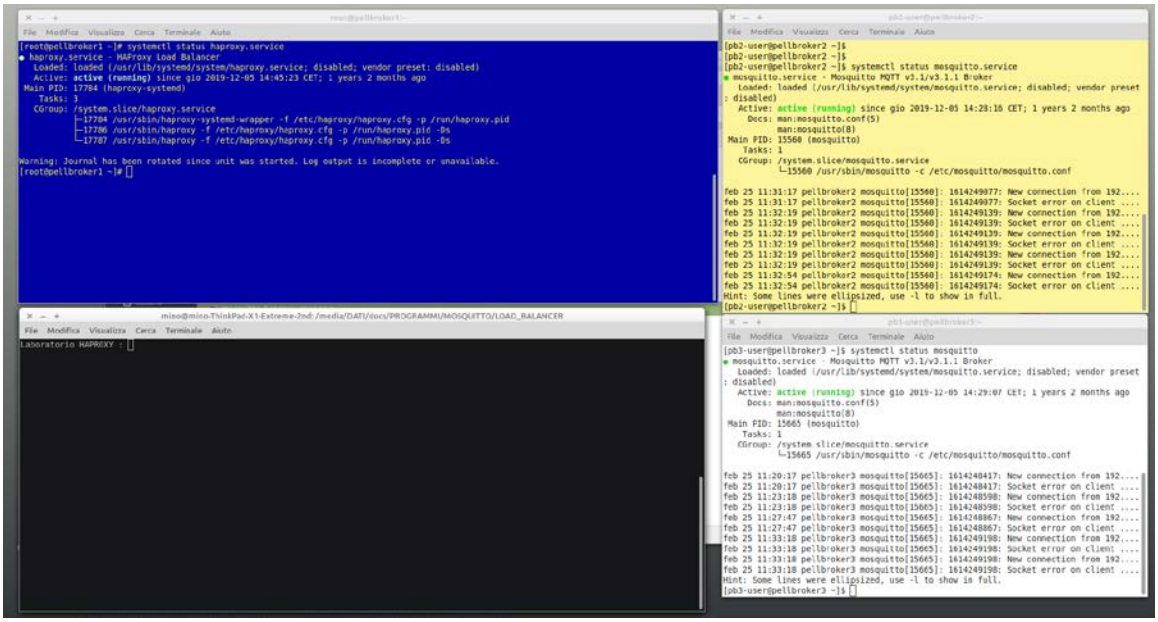
3.5.4 Test ENEA

Il test sui sistema ENEA, è consistito nella esecuzione degli stessi passi effettuati in laboratorio. La differenza sta negli indirizzi IP dei quattro sistemi coinvolti, in quanto si è operato in modalità VPN. Quindi il sistema client rimane quello di laboratorio Haproxy, ma con indirizzo diverso. Il risultato atteso è ovviamente lo stesso, ovvero vedere la ricezione sui due server mqtt di messaggi divisi in modo speculare.
Nello screenshot che segue sono evidenziati gli indirizzi IP dei sistemi:

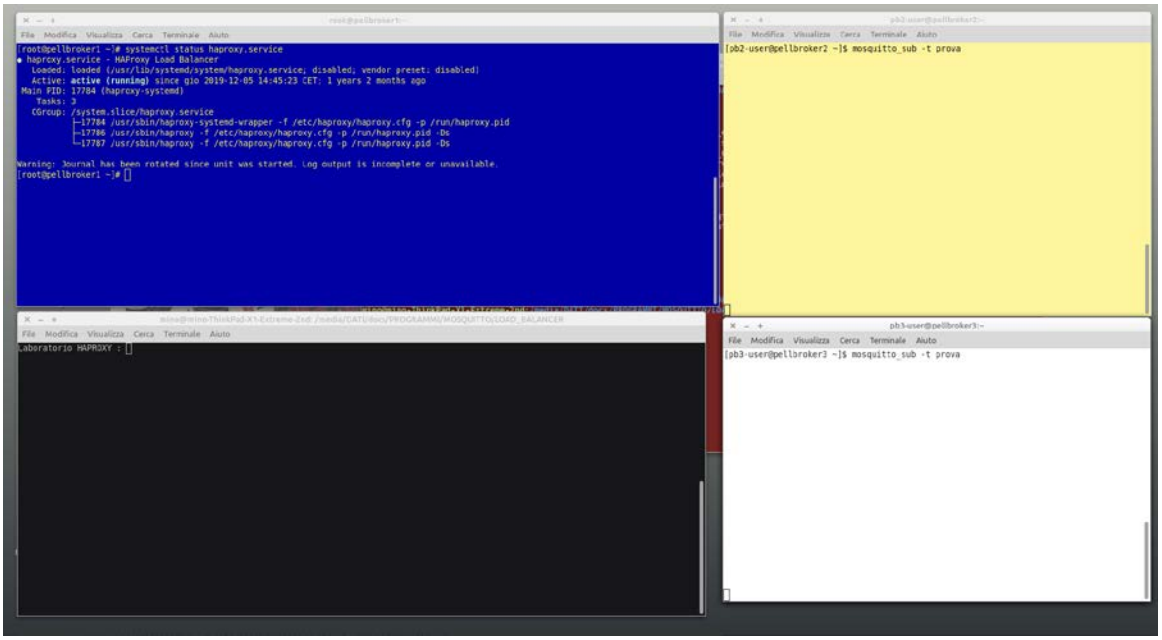


Vediamo di seguito le fasi:

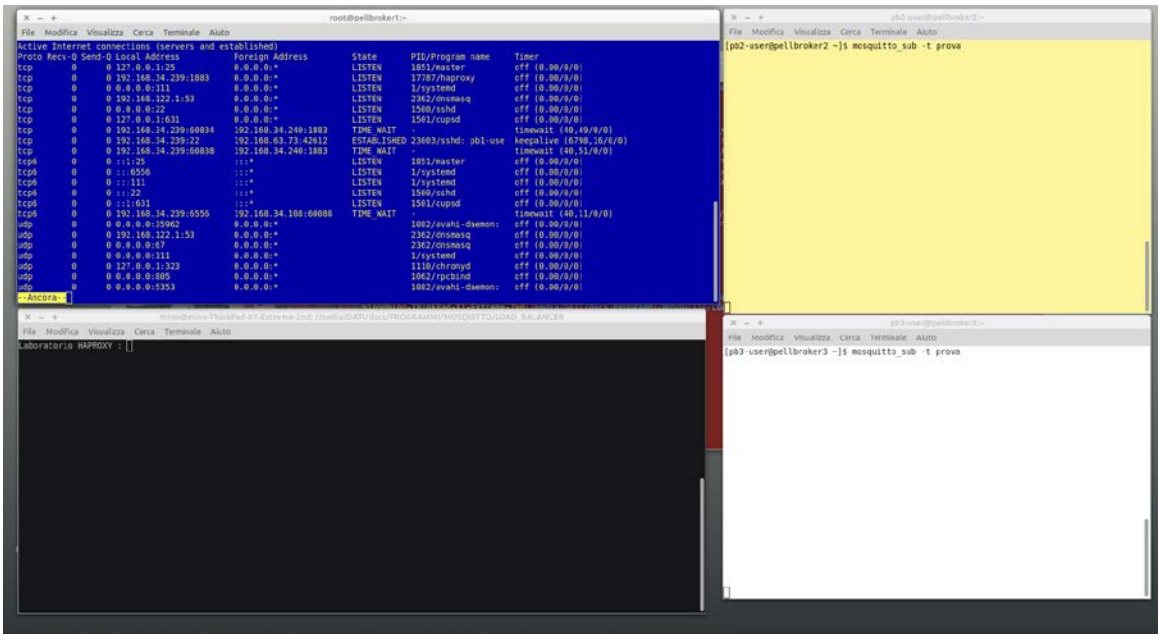
STATO DEI SERVIZI



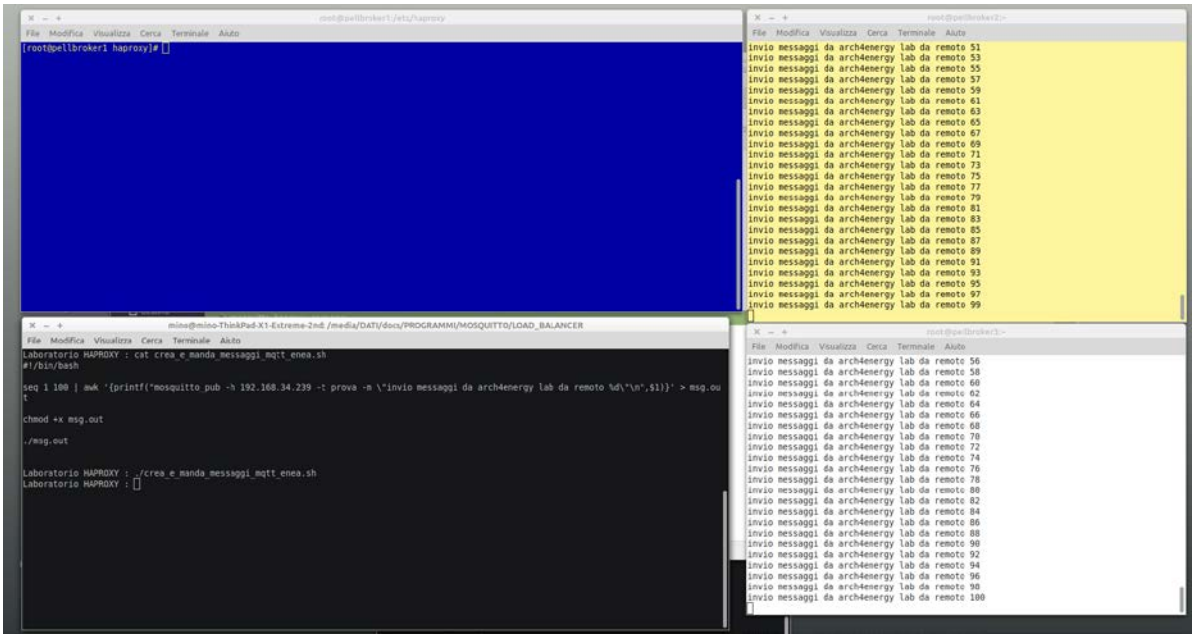
MESSA IN ASCOLTO TOPICS



VERIFICA ASCOLTO PORTE TCP



INVIO PACCHETTI DAL CLIENT E VERIFICA RICEZIONE



Si conferma dunque la funzionalità implementata.

3.6 Realizzazione Load Balancer SSL-TLS - Laboratorio

Quanto realizzato sinora è la base sulla quale di può implementare il protocollo TLS, che abilità la verifica tramite certificato che i client abbiano il permesso di connettersi al servizio. Ciò avviene tramite il metodo ben noto dei certificati. Dal punto di vista della architettura di colloquio fra il client, il proxy ed i server mqtt vi sono diverse combinazioni fra cui le più frequenti:

- 1 – implementazione del protocollo tls sul proxy e non sui server
- 2 – implementazione del protocollo tls sui server e non sul proxy

La scelta di progetto è caduta sul primo caso, che viene normalmente denominato “SSL Termination”. I motivi della scelta sono dovuti ai diversi vantaggi che essa offre:

- La gestione dei certificati viene eseguita solo sul proxy che si occupa i decriptare e criptare quando necessario. Se i server dovessero gestire il protocollo ssl-tls, avrebbero un aggravio di carico elaborativo che invece il proxy non ha. Il proxy infatti ha il compito di verificare la connessione, poi apre i pacchetti in chiaro e li passa ai server, che possono immediatamente processare la richiesta.
- Non occorre gestire i certificati dei diversi server di back-end;

3.6.1 Configurazione

Analogamente a quanto realizzato per la configurazione di base, si è proceduto in laboratorio sui sistemi clone di quelli ENEA. Per quanto detto sinora, i due server mosquitto, lo slave01 e slave02 sono rimasti nella configurazione base, ovvero non ssl-tls. Quello che ha subito le modifiche è invece il load balancer, sia per quanto attiene alla creazione dei certificati, sia per la configurazione di haproxy.

3.6.2 Creazione certificati

Tutti i comandi sono stati eseguiti sul master (192.68.56.103) come root:

 # openssl genrsa -des3 -out caproxy.key 2048
 # openssl req -new -x509 -days 1826 -key caproxy.key -out caproxy.crt
 # openssl genrsa -out serverproxy.key 2048
 # openssl req -new -out serverproxy.csr -key serverproxy.key
 # openssl x509 -req -in serverproxy.csr -CA caproxy.crt -CAkey caproxy.key -CAcreateserial -out serverproxy.crt -days 36

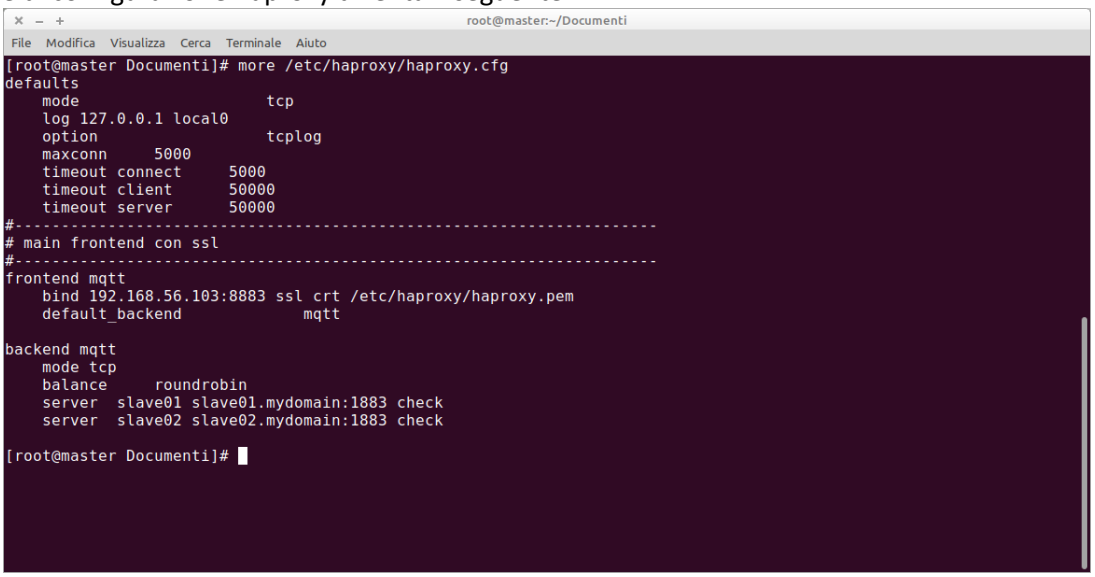
Nota tecnica importante: quando viene creato il certificato client e quello server, vengono richiesti dati sul dominio. Si deve, in tal caso, rispondere in modo differente, anche per un solo campo, nei due casi. Alla richiesta del "Common Name", si e risposto con l'indirizzo IP del proxy, ovvero 192.168.56.103 nel nostro caso.
Una volta terminata la generazione si troveranno, nella cartella di lavoro, i files seguenti:

- caproxy.crt
- caproxy.srl
- caproxy.key
- serverproxy.csr
- serverproxy.crt
- serverproxy.key

La configurazione di haproxy richiede un unico file, generalmente con estensione ".pem", che contiene le chiavi necessarie alla autenticazione.
Facendo riferimento ai files precedenti, si è creato il file "haproxy.pem" con i seguenti comandi:
 # cat serverproxy.crt > haproxy.pem
 # cat serverproxy.key >> haproxy.pem
 # cat caproxy.crt >> haproxy.pem
Il file così creato è stato posizionato nella directory /etc/haproxy

3.6.3 Configurazione haproxy

Il file di configurazione haproxy diventa il seguente:



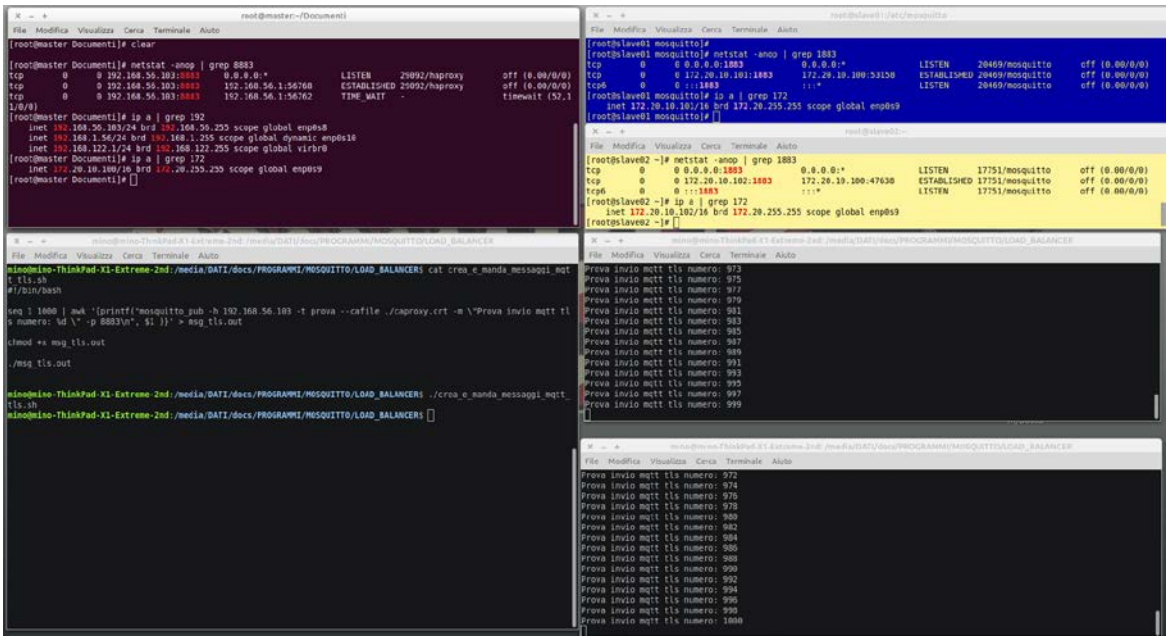
Da notare l'opzione "maxconn 5000" che indica il numero massimo di connessioni consentite. Questa opzione, pur non sollevando gli IDS aziendali dall'effettuare il detection di possibili attacchi tipo DDOS o simili, difende il proxy, e quindi i server, da un eccesso di richieste di connessione.

3.6.4 Test di Laboratorio

Il test consiste nelle seguenti operazioni:

- avvio e controllo dei server mqtt su slave01 e slave02
- verifica delle porte TCP disponibili alla connessione
- avvio e controllo di haproxy
- sottoscrizione su un topic denominato "prova" da due client situati sul sistema esterno (192.168.56.1)
- pubblicazione di 1000 messaggi dal client esterno
- verifica della distribuzione dei messaggi da stdout

Nello screenshot che segue sono visibili tutte le condizioni indicate:



Nella figura si vedono le informazioni salienti:

- gli indirizzi IP
- le porte in ascolto (1883 per mqtt senza ssl-tls e 8883 per haproxy)
- lo script utilizzato per l'invio massivo di messaggi
- la ricezione distribuita equamente sui due server di back-end.

3.7 Realizzazione Load Balancer SSL-TLS sistemi ENEA

La configurazione ENEA è stata effettuata replicando esattamente – salvo gli indirizzi IP ovviamente – quanto eseguito in laboratorio.

3.7.1 Configurazione

Il sistemi coinvolti sono stati:

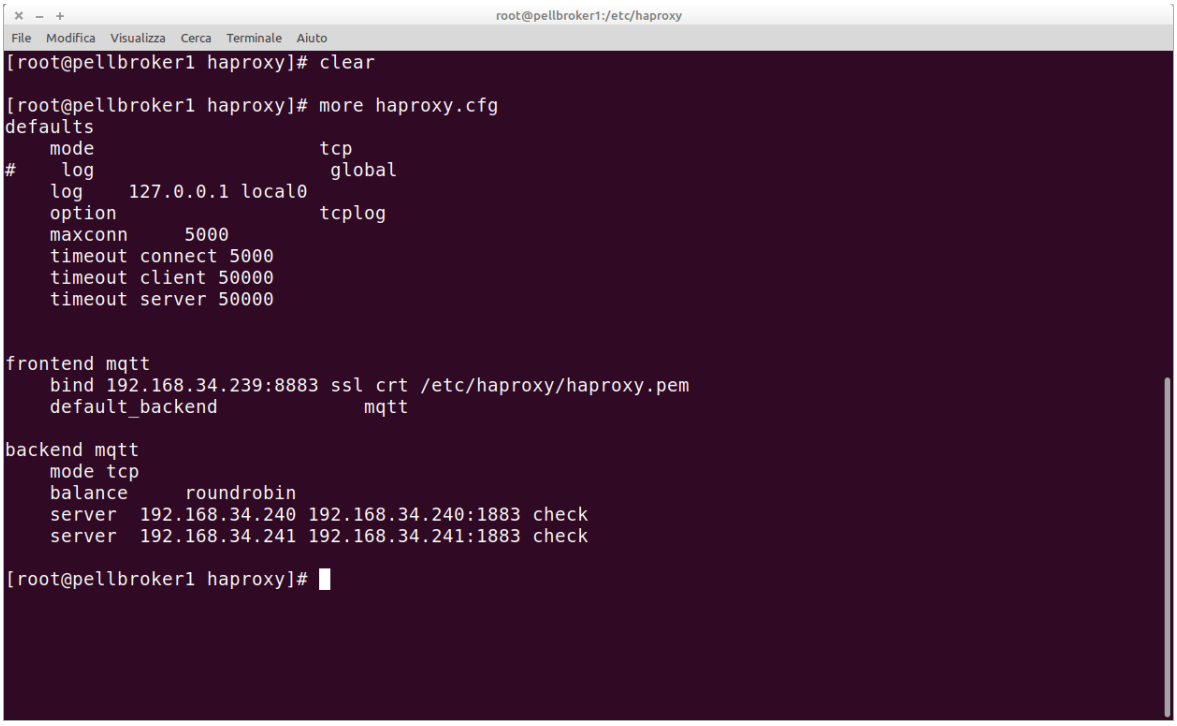
- pellbroker1 192.168.34.239 Haproxy
- pellbroker2 192.168.34.240 Mqtt server 1
- pellbroker3 192.168.34.241 Mqtt server 2
- brokerperll 192.168.34.95 come client mosquitto
- ENEA LB Lab: client esterno di laboratorio

3.7.2 Creazione certificati

Tutti i comandi sono stati eseguiti sul sistema brokerpell1 come root, in modo del tutto analogo a quello di laboratorio arch4energy. Una volta creato il file ".pem" è stato posizionato nel direttorio /etc/haproxy/.

3.7.3 Configurazione haproxy

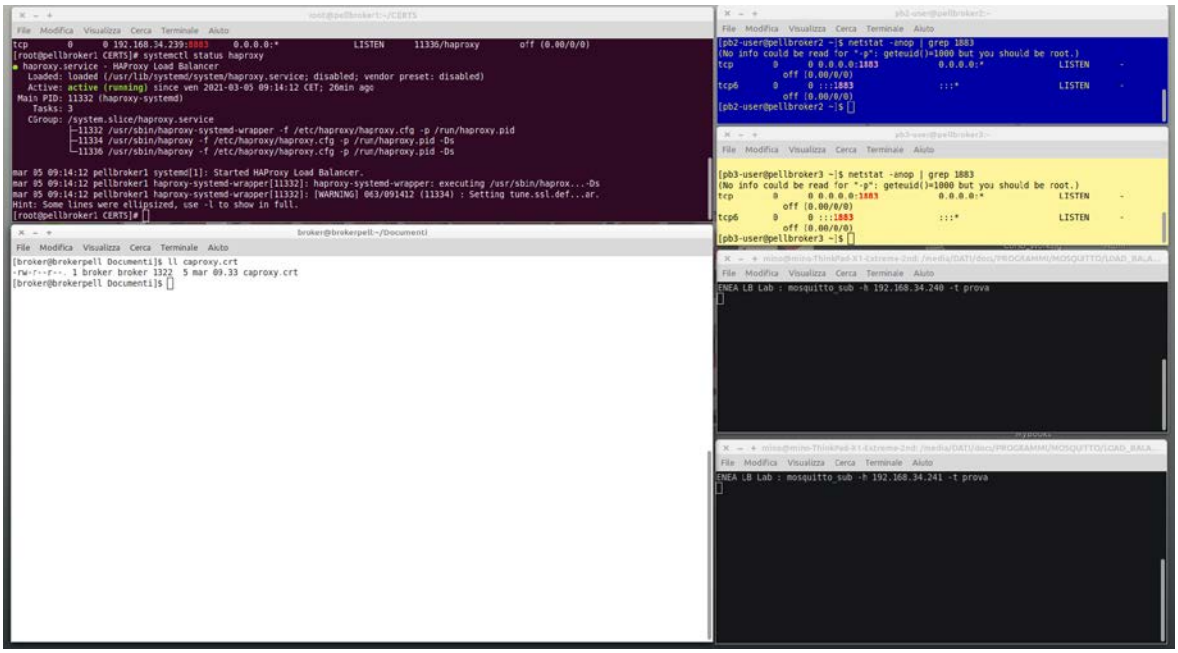
La configurazione di haproxy è stata modificata la configurazione di haproxy.cfg come segue:



3.7.4 Test ENEA

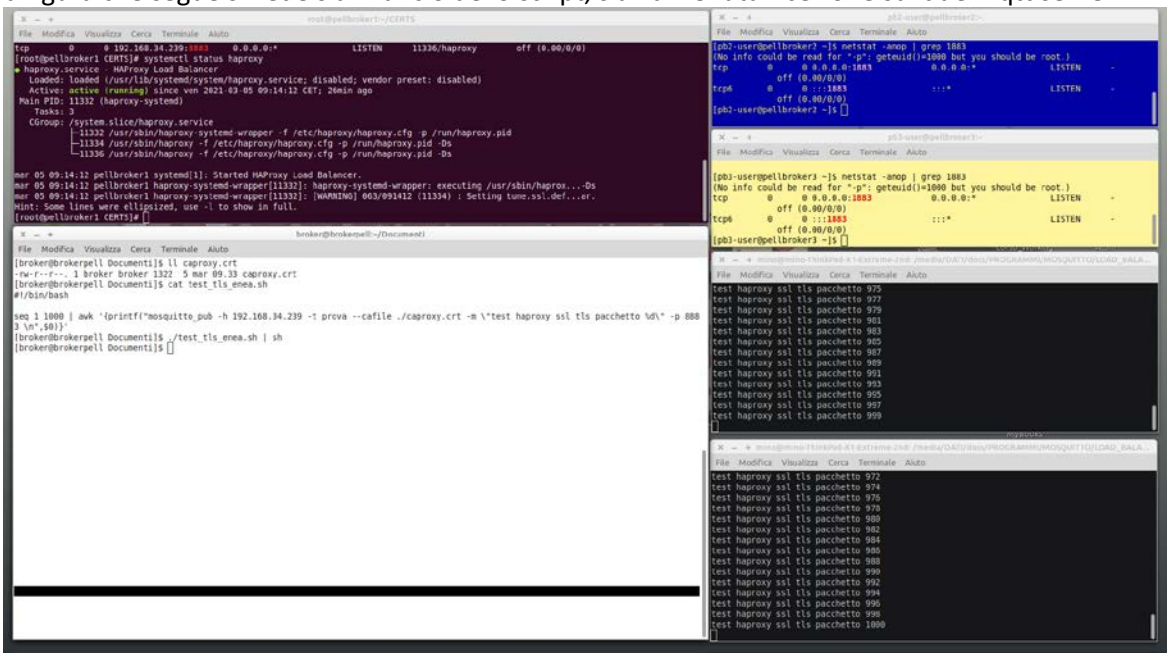
Il test è consistito nelle seguenti operazioni, riportate nello screenshot appresso:

- verifica dei server mosquitto mqtt su pellbroker2 e pellbroker3 su porta 1883
- verifica stato del servizio haproxy su pellbroker1 su porta 8883
- sottoscrizione da client esterno al topic “prova” sia su pellbroker2 che pellbroker3
- verifica certificato client brokerpell (client mosquitto)



Il test è consistito dunque nella pubblicazione (mosquitto_pub) di pacchetti di informazione verso il load balancer haproxy, tramite uno script che genera mille messaggi e le invia in continuo. L’effetto atteso è che nei terminali di sottoscrizione si vedano i messaggi in ricezione equamente distribuiti fra i due server mqtt.

Nella figura che segue si vede sia il lancio dello script, sia l'avvenuta ricezione sui due mqtt server.



Nello script è stato inserito un numero di msg che consente facilmente di verificare la corretta distribuzione effettuata dal load balancer.

3.8 Messa in opera del Load Balancer ENEA

Al momento del rilascio in esercizio, come rappresentato da questo documento, la configurazione del load balancer sui sistemi ENEA è quella del servizio mqtt su porta 1833, quindi senza configurazione ssl-tls.
Per passare alla messa in esercizio del load balancer in mqtt su porta 8883 – quindi in ssl-tsl – e viceversa, le operazioni da effettuare sono le seguenti:

3.8.1 Attivazione Mqtt su porta 8883

Per attivare il servizio mqtt criptato, si deve operare esclusivamente sul sistema pellbroker1.
Gi step da effettuare sono i seguenti:

- Collegarsi a pellbroker1 192.168.34.239
- Operare come super utente
- Posizionarsi nel direttorio /etc/haproxy
- Copiare il file haproxy.etc.tls in haproxy.cfg
- Riavviare il proxy con # systemctl restart haproxy
- Verificare il servizio con # systemctl status haproxy
- Verificare che la porta 8883 sia disponibilie tramite # netstat -anop | grep 8883
- A questo punto, da un sistema esterno come il brokerpell (192.168.34.95) si può sottoscrivere una topic, ad esempio "prova", sia sul pellbroker1 che sul pellbroker2 sulla porta 1883
    - mosquitto_sub -t prova -h 192.168.34.240
    - mosquitto_sub -t prova -h 192.168.34.241
- Da un altro sistema esterno si puo pubblicare sulla topic "prova" un qualsiasi messaggio sulla porta 8883
    - mosquitto_pub -t prova -h 192.168.34.239 –cafile ./cahaproxy.cfg -m "prova invio" -p 8883
- I messaggi arriveranno alternativamente sul primo e sul secondo broker, come visibile da terminale

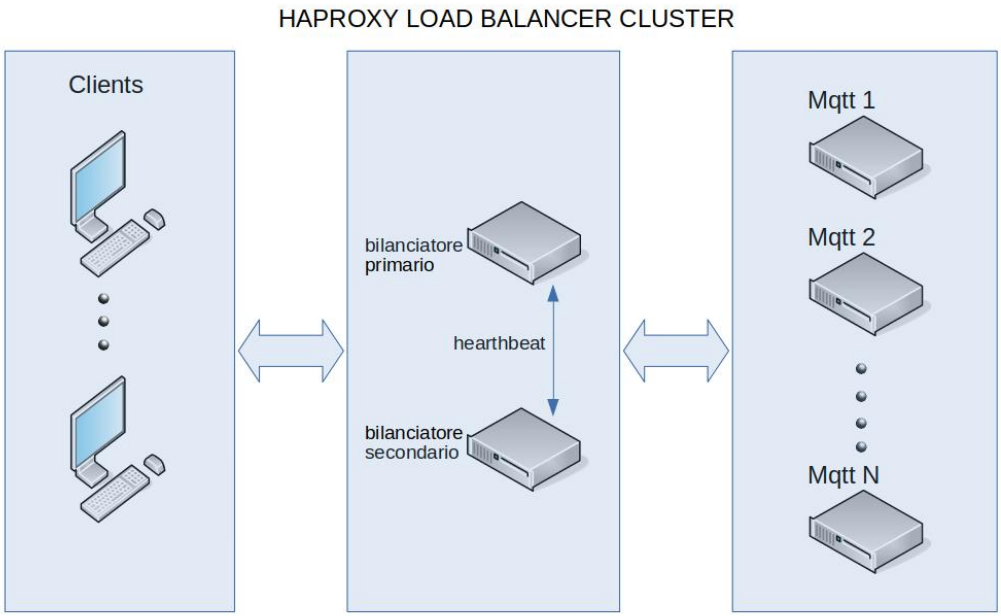
3.8.2 Attivazione Mqtt su porta 1883

Per attivare il servizio mqtt criptato, si deve operare esclusivamente sul sistema pellbroker1.
Gi step da effettuare sono i seguenti:

- Collegarsi a pellbroker1 192.168.34.239
- Operare come super utente
- Posizionarsi nel direttorio /etc/haproxy
- Copiare il file haproxy.etc.notls in haproxy.cfg
- Riavviare il proxy con # systemctl restart haproxy
- Verificare il servizio con # systemctl status haproxy
- Verificare che la porta 1883 sia disponibilie tramite # netstat -anop | grep 1883
- A questo punto, da un sistema esterno come il brokerpell (192.168.34.95) si può sottoscrivere una topic, ad esempio “prova”, sia sul pellbroker1 che sul pellbroker2 sulla porta 1883
    - mosquitto_sub -t prova -h 192.168.34.240
    - mosquitto_sub -t prova -h 192.168.34.241
- Da un altro sistema esterno si puo pubblicare sulla topic “prova” un qualsiasi messaggio sulla porta 1883
    - mosquitto_pub -t prova -h 192.168.34.239 -m “prova invio”
- I messaggi arriveranno alternativamente sul primo e sul secondo broker, come visibile da terminale

3.9 Possibili evoluzioni architetturali

L’architettura che prevede un bilanciatore di carico presente ovviamente un problema potenziale, ovvero il bilanciatore diventa uno SPOF (Single Point Of Failure).
Trattandosi però di una soluzione software che lavora sullo stack TCP/IP in modo nativo, è possibile in futuro – volendo - anche eliminare lo SPOF clusterizzando il bilanciatore stesso.
La clusterizzazione dei sistemi linux è orami uno standard consolidato. Una architettura molto utilizzata è quella del cluster in stand by, ovvero una coppia di server primario-secondario, in cui il primario è attivo e l’altro è acceso ed in grado di prendere in carico il servizio al momento della indisponibilità del primario. In riferimento al servizio haproxy, si può ipotizzare quindi una situazione come quella in figura:



Il funzionamento di una simile architettura è tale da rendere sostanzialmente impossibile la indisponibilità del servizio erogato dal sistema bilanciato, nel suo complesso. Nel caso attuale l’infrastruttura di ENEA risolve

il problema della continuità di servizio grazie alla piattaforma Vmware su cui sono installati i sistemi, per cui non si pone il problema dello SPOF. E' comunque fondamentale avere tali possibilità evolutive, in caso di future modifiche alle architetture di Data Center ENEA.