ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

DIPARTIMENTO DI INFORMATICA – SCIENZE E INGEGNERIA DISI

CORSO DI LAUREA MAGISTRALE IN
INGEGNERIA INFORMATICA (5826)

MIGRAZIONE DI APPLICAZIONI IN SCENARI 5G FEDERATI MULTI-ACCESS EDGE COMPUTING (MEC)

Tesi di laurea magistrale in MOBILE SYSTEM M

Relatore
Prof. Paolo Bellavista

Presentata da
Francesco Milione

Correlatore
Dott. Alessandro Calvio
Dott. Angelo Feraudo

Sessione Ottobre 2025

Anno Accademico 2024/2025

A Mamma, Papà
e Michele

Sommario

Abstract

L’avvento del 5G ha abilitato applicazioni distribuite con requisiti stringenti di latenza, affidabilità e scalabilità. Il paradigma del Multi-access Edge Computing (MEC), standardizzato da ETSI, consente di avvicinare le risorse computazionali all’utente finale, ma la mobilità e la coesistenza di domini multi-operatore pongono nuove sfide alla continuità del servizio. In questo contesto, la Federazione MEC è stata introdotta per abilitare interoperabilità e cooperazione tra sistemi distinti, rendendo possibile la migrazione trasparente delle applicazioni.

Questa tesi affronta il tema della migrazione applicativa in scenari federati MEC, con l’obiettivo di progettare e valutare un meccanismo che supporti discovery, decisione e trasferimento dello stato applicativo. Lo studio prende come riferimento gli standard ETSI e la letteratura di settore, integrando soluzioni progettuali originali per gli aspetti non ancora formalizzati.

La valutazione è stata condotta tramite simulazioni, considerando scenari intra-operator, federati e dinamici con mobilità veicolare. Sono state analizzate metriche come tempo di migrazione, latenza complessiva, scalabilità e impatto della dimensione dello stato trasferito.

I risultati dimostrano che la federazione consente di mantenere la continuità del servizio anche in contesti multi-operatore, con prestazioni comparabili a quelle intra-operator, pur introducendo nuove complessità gestionali. Il contributo principale del lavoro risiede nella definizione e validazione di un modello di migrazione federata conforme agli standard ETSI, utile come base per future evoluzioni di reti 5G e oltre.

1 Introduzione

Negli ultimi anni c'è stata un'evoluzione delle reti mobili significativa, con l'avvento del 5G, che introduce prestazioni superiori rispetto alle generazioni precedenti in termini di velocità, latenza e capacità di connessione massiva. Il 5G è stato progettato per supportare tre scenari fondamentali definiti dal 3rd Generation Partnership Project (3GPP): Enhanced Mobile Broadband (eMBB), che garantisce throughput molto elevati per servizi multimediali e applicazioni data-intensive; Ultra-Reliable Low Latency Communications (URLLC), dedicato a comunicazioni ultra-affidabili e con vincoli di latenza stringenti, essenziali per servizi mission-critical; e Massive Machine-Type Communications (mMTC), pensato per l'Internet of Things e la connettività di massa [1]. Queste nuove caratteristiche hanno aperto la strada a nuove applicazioni distribuite, come i sistemi veicolari connessi (V2X), le applicazioni di realtà aumentata e virtuale (AR/VR) e i servizi mission-critical. In particolare, queste applicazioni condividono il fatto che per funzionare correttamente debbano avere una latenza bassa. Cosa che il 5G ci permette di avere grazie al fatto che le funzioni posso essere posizionate nella rete vicino all'utente.
Perleremo di Multi-access Edge Computing (MEC), standard definito dall'European Telecommunications Standards Institute (ETSI), che offre funzionalità di cloud computing e un'erogazione di servizi vicino al terminale [2]. L'applicazione, dunque, potrà muoversi e sarà sempre vicina all'utente che si sta spostando nella rete. Proprio la mobilità rappresenta una sfida cruciale. Le connessioni variano rapidamente con lo spostamento degli utenti, i nodi computazionali possono entrare e uscire dalla rete in modo dinamico, e la continuità del servizio rischia di essere compromessa. Per questo motivo, i sistemi MEC devono prevedere meccanismi di migrazione delle applicazioni, così da mantenere inalterata l'esperienza utente anche quando il terminale attraversa aree di copertura differenti o cambia il dominio di appartenenza. le applicazioni possono essere spostate più vicino all'utente, il che introduce problematiche legate alla gestione della migrazione in reti appartenenti a operatori o domini differenti. Fin quando ci troveremo nella stessa rete lo standard MEC risolve i nostri problemi. Quando abbiamo questi nuovi scenari entra in gioco quella che viene chiamata Federazione MEC [3]. La federazione è stata recentemente formalizzata da ETSI con l'introduzione di nuove entità funzionali. Queste componenti hanno il compito di gestire la comunicazione tra orchestratori di

diversi domini, facilitando lo scambio di informazioni e la migrazione di un'applicazione anche in reti diverse. Ciò porta ad avere nuovi problemi legati alla gestione, scoperta e interoperabilità dei sistemi che appartengono a operatori diversi. Gli standard che parlano della federazione cercano di far adottare agli sviluppatori di app, rete e sistemi un approccio unico per integrare questa nuova capacità dei sistemi di collaborare per il trasferimento di un'applicazione. Studiare la federazione MEC in contesti reali non è banale: richiederebbe la collaborazione tra operatori diversi, l'accesso a infrastrutture eterogenee e l'implementazione di meccanismi di coordinamento complessi, difficilmente riproducibili in un laboratorio universitario. Per questo motivo, l'uso di piattaforme di simulazione come OMNeT++ e Simu5G è particolarmente rilevante: esse permettono di modellare scenari realistici di rete 5G integrati con sistemi MEC e di testare algoritmi e protocolli in un ambiente controllato, replicabile e facilmente estendibile [4]. Lo scopo della tesi, infatti, è quello di implementare un meccanismo di migrazione per le applicazioni in un ambiente federato. Facendo riferimento agli standard ETSI e utilizzando Simu5G come piattaforma di simulazione integrata nell'ambiente di simulazione OMNeT++. L'implementazione è stata fatta introducendo nuovi componenti e integrando quelli esistenti per supportare scenari di federazione.

Successivamente, è stata condotta una fase di raccolta dati eseguendo numerose simulazioni con differenti parametri di input. L'obiettivo era quello di confrontare il comportamento della migrazione delle applicazioni non solo in scenari intra-operator e federati. In questo modo è stato possibile analizzare le differenze in termini di prestazioni come latenza, tempi di handover e qualità percepita dall'utente finale, valutando l'impatto che le diverse architetture hanno sia sul mantenimento della Quality of Service (QoS) sia sulla Quality of Experience (QoE). Tale confronto ha permesso di evidenziare i vantaggi e le criticità introdotte dai diversi scenari, fornendo un quadro completo del comportamento del sistema in contesti eterogenei.

I contributi principali di questo lavoro sono riconducibili a tre aspetti: i) la progettazione e realizzazione di un meccanismo di decisione e gestione della migrazione tra sistemi MEC federati; ii) la raccolta di misure di prestazioni, con particolare attenzione ai tempi di handover e ai parametri di QoS; iii) l'analisi comparativa tra scenari intra-operator e federati.

Il resto della tesi è organizzato come segue: il Capitolo 2 introduce i concetti di background relativi al 5G e al MEC; il Capitolo 3 presenta i lavori correlati sul tema della

federazione; il Capitolo 4 descrive il system design, facendo riferimento alla reference architecture ETSI e a diagrammi di sequenza dei flussi di interazione; il Capitolo 5 illustra l'implementazione dei moduli in Simu5G; il Capitolo 6 discute i risultati ottenuti dalle simulazioni; infine, il Capitolo 7 raccoglie le conclusioni e le prospettive future.

2 Background

L'evoluzione delle reti mobili verso quello che è denominato paradigma 5G ha portato ad avere un'architettura con una bassa latenza e un supporto per il calcolo computazionale vicino all'utente. In particolare, parliamo di Multi-access Edge Computing, standard definito dall'European Telecommunications Standards Institute, che offre funzionalità di cloud computing e un'erogazione di servizi vicino al terminale. L'avvicinamento alla sorgente riduce drasticamente il percorso dei dati rispetto ad un'infrastruttura cloud centralizzata, diminuendo la latenza e la congestione di rete, aumentando l'esperienza e l'efficienza delle applicazioni real-time.
Il MEC è uno standard che permette il supporto ad applicazioni con alta interattività con ottimizzazione di banda e requisiti in Quality of Services. Le architetture MEC possono operare singolo operatore o in modalità federata, dove più operatori mettono a disposizione di altri i propri servizi e la migrazione di applicazioni. Questo è possibile attraverso regole di orchestrazione e federazione ben definite nei documenti ETSI.
Nelle sezioni seguenti viene descritta l'architettura proposta dallo standard ETSI, con particolare attenzione ai principali componenti coinvolti nei processi di discovery e migrazione delle applicazioni.

2.1 Evoluzione verso la rete 5G

L'evoluzione delle reti mobili è stata caratterizzata da generazioni successive (1G, 2G, 3G, 4G), ciascuna delle quali ha introdotto innovazioni sostanziali. La **prima generazione** (1G), nata negli anni '80, era basata su tecnologia analogica e permetteva esclusivamente comunicazioni vocali, si basava sulla tecnica chiamata FDMA (Frequency Division Multiple Access). Con la **seconda generazione** (2G), negli anni '90, si è passati al digitale, i segnali non sono più analogici e introduce gli SMS e primi servizi dati, seppur limitati in capacità. La **terza generazione** (3G) è stata una rivoluzione poiché ha reso possibile l'accesso a Internet via telefono simile a un pc collegato in rete, con velocità nell'ordine delle dei 2 Mbps, abilitando servizi multimediali e applicazioni mobili. Con il **4G/LTE** si è assistito a un aumento drastico delle prestazioni, raggiungendo velocità fino a centinaia di Mbps, abilitando lo streaming video, i social network e applicazioni di massa basate su dati [5].

Tuttavia, queste generazioni presentavano limiti evidenti: le latenze erano ancora elevate il supporto per dispositivi IoT massivi era limitato, e non era possibile garantire la continuità e affidabilità richiesta dai servizi mission-critical. Da queste necessità nasce il **5G** (Fifth Generation), concepito non solo come un incremento di banda e riduzione della latenza, ma come una piattaforma flessibile e programmabile in grado di supportare scenari molto eterogenei.
Ma per il 5G dovremmo aggiungere anche la parte delle funzionalità che supera di gran lunga le precedenti. Un'architettura basata su servizi (SBA), questa architettura consente al core 5G di essere disaggregato, cloud-native e distribuito. Uno dei vantaggi è avere proprio che le funzioni possono essere posizionate in modo ottimale per fornire le migliori prestazioni.

2.2 5G

Il documento 3GPP TS 23.501, pubblicato da ETSI e redatto dal 3rd Generation Partnership Project, un partenariato che riunisce gli enti di standardizzazione regionale (SDO) di Europa, America, Giappone, Cina, Corea e India, definisce **l'architettura del sistema 5G (5GS)** come i dispositivi mobili comunicano via radio con le antenne, ma specifica anche tutti i protocolli e le interfacce che permettono a una rete mobile di funzionare nel suo complesso: dalle chiamate, alla navigazione internet, alla gestione degli spostamenti degli utenti. Questo approccio consente alle reti mobili standardizzate dal 3GPP, come UMTS (3G), LTE (4G) e 5G System (5GS), di operare in ambienti eterogenei, garantendo l'interoperabilità sia tra dispositivi e componenti di diversi fornitori (inter-vendor), sia tra reti gestite da operatori differenti (inter-operator).
Il 5G migliora i servizi delle precedenti generazioni, in particolare migliora il 4G avendo una **Banda larga mobile avanzata, eMBB** che consente un volume di dati più elevati. Per il downlink, fino a 50 Mbps per esterni e 1 Gbps per interni (5GLAN), l'uplink la metà di questi. Un altro miglioramento è nelle **Comunicazioni critiche, CC e comunicazioni ultra-affidabili e a bassa latenza, URLLC** in alcuni contesti, ci si aspetta un'affidabilità estremamente elevata. Questo è fornito in particolare attraverso la capacità di Edge Computing. Il 5G si distingue per il supporto al **Massive Internet of Things, mIoT** supporti di densità di traffico molto elevate dei dispositivi. I requisiti di Massive Internet of Things includono gli aspetti operativi che si applicano all'ampia gamma di dispositivi e servizi IoT. Infine, servizi per **Operazioni di rete flessibili** che

copre aspetti come la slicing della rete, l'esposizione alle capacità di rete, la scalabilità e la mobilità diversificata, la sicurezza, la consegna efficiente dei contenuti e la migrazione e l'interazione [6]. Nei sistemi cellulari 4G/5G, chiameremo handover la procedura con cui un UE trasferisce la connessione radio da una cella sorgente a una cella target senza interrompere il servizio. Questa operazione è essenziale per garantire la continuità della comunicazione quando l'utente si sposta. Nel contesto del 5G, il processo di handover si articola in due fasi principali, la preparazione: la rete identifica la cella target e notifica l'UE e l'esecuzione: l'UE completa il passaggio, aggiornando i contesti di rete e ristabilendo il canale dati.

2.2.1 Architettura

Il sistema 5G (5GS) mantiene la struttura logica delle generazioni precedenti, basata su tre componenti principali come illustrato in Figura 1:

- l'User Equipment (**UE**), composto da terminale e USIM,
- la rete di accesso radio di nuova generazione (**NG-RAN**),
- la rete core 5G (**5GC**).

Figura 1 - Sistema 5G [1]

Nel NG-RAN l'elemento centrale è il gNB (Next Generation Node B), suddivisibile in unità centrale (gNB-CU) e unità distribuite (gNB-DU) collegate tra loro. La nuova interfaccia radio si chiama NR-Uu, dove NR indica New Radio, cioè la tecnologia di accesso radio 5G.

La rete core 5GC adotta un'architettura basata su servizi (**SBA - Service-Based Architecture**), in cui le componenti sono chiamate Funzioni di Rete (NF). Ogni NF

espone i propri servizi tramite interfacce comuni e può interagire con altri NF autorizzati, favorendo modularità e scalabilità.

Tra le funzioni principali della rete 5GC si trovano:

- **UPF (User Plane Function):** gestisce il traffico utente verso Internet o verso ambienti MEC.
- **AMF (Access and Mobility Management Function):** gestisce accesso, mobilità e registrazione dell'UE.
- **SMF (Session Management Function):** controlla le sessioni IP, collegandosi alla UPF.
- **PCF (Policy Control Function) e UDM (Unified Data Management):** per controllo di policy e **dati utente.**
- **NRF (Network Repository Function):** permette la scoperta e registrazione delle NF.

Questa architettura abilita direttamente scenari di edge computing, in quanto la funzione UPF può essere posizionata ai bordi della rete e il traffico utente può essere indirizzato verso applicazioni MEC con latenze ridotte.

L'UE rappresenta l'elemento terminale della rete, ossia il dispositivo utilizzato dall'utente finale che può essere un cellulare o anche un veicolo. È composto da: **ME (Mobile Equipment)**, cioè il terminale vero e proprio che integra capacità di elaborazione, radio e protocolli di rete, **USIM (Universal Subscriber Identity Module)**, che contiene le informazioni di autenticazione, le chiavi crittografiche e gli identificatori univoci dell'abbonato. L'UE non è un semplice utente passivo ma può svolgere un ruolo attivo nell'interazione con la rete, come il supporto alla selezione della rete e alla registrazione presso l'AMF, gestisce le procedure di handover e mobilità, comunicando eventi e misurazioni al NG-RAN e può interagire con funzioni di slicing per accedere a servizi differenziati.

2.2.2 Stack di Protocollo

Lo stack di protocollo del 5G è divisibile in due parti: piano di controllo (Control Pane) e piano utente (User Plane).

Il **Control Pane,** in Figura 2 si occupa della gestione necessaria per far si che l'UE si possa registrare, connettesi e muoversi nelle celle diversi mantenendo la sessione attiva.

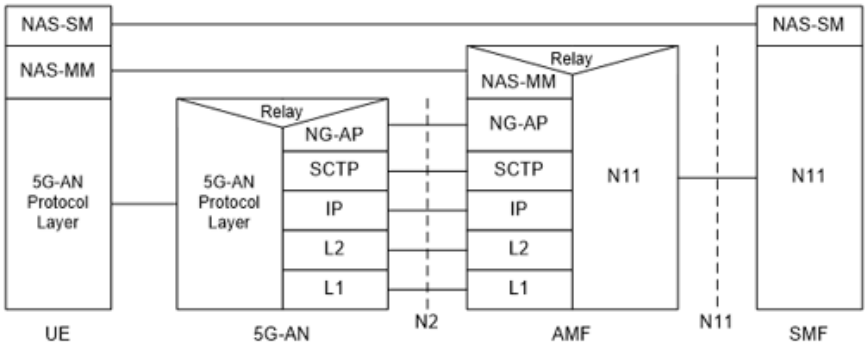

Figura 2 - Stack 5G Control Pane [1]

È diviso in **NAS-SM** e **NAS-MM**, dove NAS sta per Non-Access Stratum, SM Session Manager e MM Mobility Manager. Si occupano rispettivamente della creazione, modifica e rilascio delle sessioni dati e della connessione e mobilità dell'UE. Garantendo anche la sicurezza con la cifratura. **5G-AN** il livello di accesso radio. **NG-AP** Next Generation Application Protocol che consente il dialogo tra reti di accesso e l'AMF. **SCTP** Stream Control Transmission Protocol che assicura la consegna dei messaggi sull'interfaccia N2.

L'user plane, mostrato in Figura 3 invece è chi trasporta i veri e propri dati delle applicazioni è diviso in Livello PDU livello dove transitano i pacchetti che l'UE invia o riceve, possono essere sia di IPv4 che IPv6. GTP-U, GPRS Tunnelling Protocol – User Plane è il protocollo di tunneling che incapsula i pacchetti utente e li trasporta nella rete core. Stack del 5G-AN, Data Plane, che include protocolli per la gestione della trasmissione radio. UDP/IP, protocolli di rete standard.

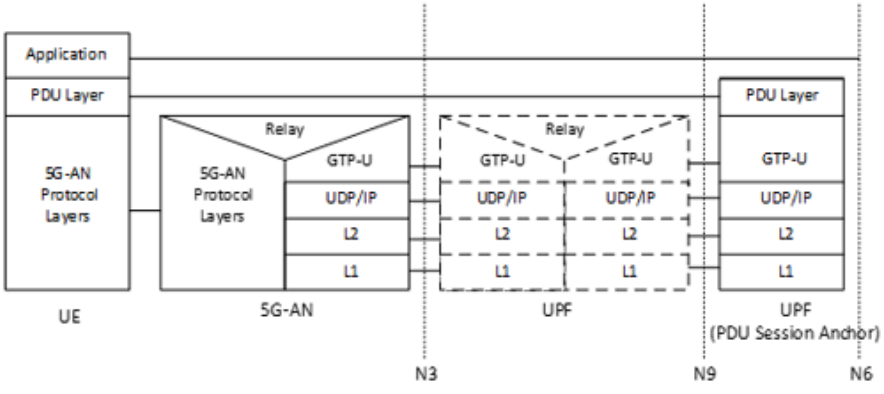

Figura 3 - Stack 5G User Plane [1]

2.2.3 Architettura Non-Stand Alone e Stand-Alone

Due opzioni di distribuzione sono definite per il 5G:

l'architettura "Non-Stand Alone" (NSA), in Figura 4, la 5G Radio Access Network (AN) e la sua interfaccia New Radio (NR) vengono utilizzate in combinazione con l'infrastruttura LTE ed EPC esistente Core Network, rendendo così disponibile la tecnologia NR senza sostituzione della rete. In questa configurazione, sono supportati solo i servizi 4G, ma godono delle capacità offerte dalla 5G New Radio.

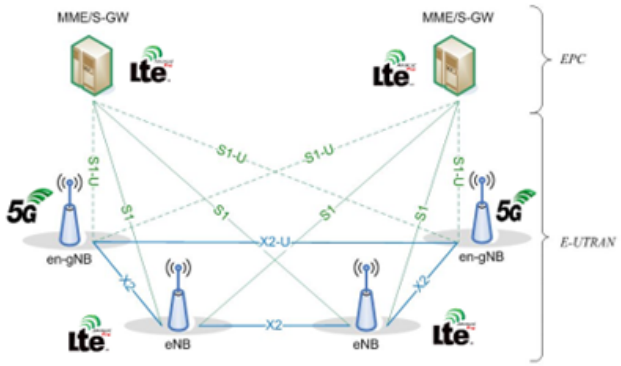

Figura 4 - Architettura Non-Stand-Alone [1]

l'architettura "Stand-Alone" (SA), rappresentata in Figura 5, dove l'NR è collegato al CN 5G. Solo in questa configurazione, sono supportate le funzionalità avanzate come: Network slicing, Ultra-reliable low-latency communications (URLLC), supporto completo per MEC con UPF distribuiti e
Dynamic policy e QoS control avanzati [1].

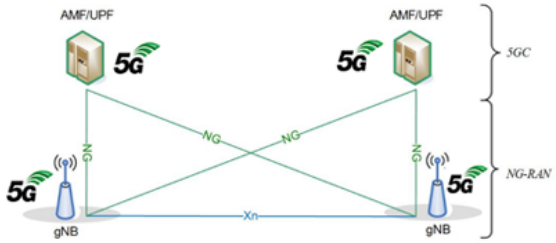

Figura 5 - Architettura Stand-Alone [1]

2.2.4 Funzionalità chiave

Il **Network Slicing** è una delle funzionalità distintive del 5G, che consente alla rete di essere suddivisa logicamente in più “fette” (slices), ognuna delle quali può essere configurata e ottimizzata per soddisfare specifici requisiti di servizio, utenti o applicazioni.

Ogni slice funziona come una rete virtuale autonoma, con una propria rete core (CN) e un proprio insieme di risorse, anche se condivide l'infrastruttura fisica sottostante. Questa architettura consente a un singolo operatore di fornire servizi diversificati con differenti caratteristiche di latenza, banda, priorità e sicurezza. Ad esempio: una slice può essere dedicata agli utenti consumer dell'operatore mobile tradizionale; un'altra slice può essere riservata a un operatore virtuale (MVNO) con proprie politiche di rete; una terza può essere configurata per applicazioni IoT/M2M come il monitoraggio di container o veicoli autonomi, con requisiti di bassa latenza e alta affidabilità [1].

Virtualizzazione delle Funzioni di Rete (NFV). Nel contesto del 5G, tutte le funzioni di rete (Network Functions - NF) sono implementate come moduli software virtualizzati, che comunicano attraverso una Service-Based Interface (SBI) comune. Questo approccio consente una estrema flessibilità nella distribuzione della rete, poiché le NF non sono più vincolate a una specifica posizione fisica e possono essere eseguite su qualunque nodo abilitato, anche in ambienti cloud o edge.

Un vantaggio importante di questa virtualizzazione è la semplificazione delle operazioni di manutenzione e gestione. È possibile, ad esempio, attivare rapidamente una funzione temporanea (es. una NF per bilanciare il carico in caso di traffico elevato) o sostituire dinamicamente una funzione guasta senza impatti significativi sul servizio [7].

Il 5G con le sue caratteristiche come la bassa latenza, minore di un millisecondo, la possibilità di connettere dispositivi a km di distanza ha reso evidente il problema di spostare le capacità di calcolo e di storage vicino agli utenti e quindi al margine della rete. Questa sfida è affrontata dall’**Edge Computing** che consiste nell'avvicinare fisicamente la potenza computazionale all’utente finale, spostando alcune risorse elaborative dal cloud centrale verso il bordo della rete. Alcuni scenari ci vengono in aiuto nel capire i

veri motivi dell'adozione dell'Edge Computing. Come nel caso della guida autonoma o della telemedicina che richiedono una bassa latenza inferiore ai millisecondi impossibile da garantire con un'elaborazione esclusivamente cloud. Un altro caso è la crescita esponenziale di dati IoT dove milioni di dispositivi generano flussi continui che, se inviati integralmente al cloud, saturerebbero la rete. L'edge consente filtraggio e preelaborazione locale. Inoltre, l'elaborazione locale offre una maggiore sicurezza e privacy dei dati.

In pratica, il concetto prevede che alcune repliche locali di server o servizi vengano posizionate in prossimità degli utenti o dei dispositivi (UE), così da ridurre i tempi di propagazione e migliorare le prestazioni delle applicazioni critiche [8].

2.3 Vehicular Edge Computing

L'introduzione del 5G ha abilitato scenari caratterizzati da bassa latenza, elevata capacità trasmissiva e supporto a un numero massivo di dispositivi connessi. Queste proprietà hanno avuto un impatto significativo anche nel settore dei trasporti, favorendo lo sviluppo delle Vehicular Networks, reti che consentono la comunicazione continua tra veicoli e con l'infrastruttura circostante. In particolare, le Vehicular Ad Hoc Networks (VANETs) rappresentano un'estensione delle reti mobili ad hoc (MANETs) al contesto veicolare: i veicoli diventano nodi mobili in grado di comunicare direttamente tra loro (Vehicle-to-Vehicle, V2V) o con unità di bordo strada (Vehicle-to-Infrastructure, V2I), supportando così servizi di sicurezza stradale, gestione del traffico e applicazioni infotainment.
Su questa base concettuale si è sviluppato il paradigma del **Vehicular Cloud Computing** (VCC), che considera l'insieme dei veicoli come un "cloud distribuito" capace di offrire risorse eterogenee di calcolo, memoria e connettività. In questo scenario, automobili, autobus o persino veicoli parcheggiati possono condividere parte della propria capacità di elaborazione per supportare applicazioni collettive, come l'elaborazione cooperativa dei dati provenienti dai sensori, il monitoraggio ambientale o l'analisi distribuita del traffico. Sebbene il VCC rappresenti un passo importante verso l'utilizzo cooperativo delle risorse veicolari, la sua natura fortemente dinamica, legata alla mobilità dei nodi e alla disomogeneità delle risorse disponibili, rende complesso garantire continuità di servizio e requisiti di qualità stringenti, specialmente per applicazioni critiche in tempo reale [9].

Per superare le criticità del VCC, si è affermato il paradigma del **Vehicular Edge Computing** (VEC), che estende e completa il VCC e può essere visto come la declinazione del Mobile/Multi-access Edge Computing al contesto veicolare. L'idea di base è spostare le funzionalità di calcolo, storage e servizio dall'interno dei veicoli verso l'infrastruttura di rete prossima all'utente, tipicamente costituita da Road Side Units (RSU) e stazioni base 5G (gNodeB) dotate di piattaforme MEC.

Attraverso questa architettura, i veicoli possono effettuare offloading computazionale, delegando parte dell'elaborazione di applicazioni intensive a nodi edge affidabili, riducendo così il consumo di risorse locali e garantendo tempi di risposta compatibili con i requisiti real-time. Questo approccio consente di superare i problemi del VCC legati alla natura transitoria e instabile delle risorse veicolari, beneficiando al tempo stesso della bassa latenza e della vicinanza all'utente che caratterizzano l'edge computing. [10]

Il VEC abilita scenari avanzati di mobilità intelligente, tra cui:

- Platooning cooperativo, in cui veicoli connessi mantengono velocità e distanze sincronizzate grazie a calcoli distribuiti sui nodi edge;
- Collision avoidance e safety applications, che richiedono latenze inferiori ai 10 ms per essere efficaci;
- Gestione intelligente del traffico, attraverso analisi in tempo reale di flussi veicolari e condizioni ambientali;
- Supporto alla guida autonoma, con elaborazione predittiva e coordinata di dati provenienti da sensori eterogenei;
- Infotainment e AR/VR on-board, che richiedono capacità di calcolo elevate e stabilità nella connettività.

Dal punto di vista architetturale, il VEC eredita i principi dell'ETSI MEC, adattandoli al contesto veicolare. I nodi edge fungono da MEC Hosts, situati in prossimità della rete di accesso, riducendo la distanza fisica e logica tra veicoli e risorse computazionali; Le applicazioni veicolari si basano su un'infrastruttura eterogenea e distribuita, che integra comunicazioni V2X (Vehicle-to-Everything) con le funzionalità di orchestrazione e gestione proprie del MEC;

L'interazione con i servizi MEC consente di ottimizzare il posizionamento e la migrazione delle applicazioni in funzione della mobilità dei veicoli.

Nonostante i vantaggi, il VEC introduce anche nuove sfide di ricerca: la gestione della mobilità e dell'handover tra edge server, la scalabilità delle piattaforme MEC in scenari

urbani ad alta densità veicolare, e le problematiche di sicurezza e privacy legate alla natura sensibile dei dati trattati.

In sintesi, il VEC rappresenta l'evoluzione naturale delle VANETs e del VCC, poiché sfrutta la robustezza e la continuità dell'infrastruttura edge per fornire servizi affidabili, a bassa latenza e ad alta disponibilità, aprendo la strada a un'integrazione più stretta con le architetture MEC definite da ETSI.

2.4 Architettura MEC

Se da un lato il VCC e il VEC consentono di valorizzare le risorse veicolari, dall'altro risulta necessario un quadro architetturale standardizzato che permetta di integrare tali risorse con l'infrastruttura di rete esistente e con i servizi di edge computing. In questa direzione si colloca il paradigma del **Multi-access Edge Computing** che abilita l'esecuzione di applicazioni MEC come entità software virtualizzate, distribuite su un'infrastruttura di virtualizzazione localizzata all'interno o nelle immediate vicinanze del bordo della rete. Il framework architetturale definito da ETSI per il MEC descrive l'insieme delle entità coinvolte nell'ecosistema, organizzandole in tre livelli principali come si può vedere dalla Figura 6: a livello di sistema (MEC System level), di host (MEC Host level) e di rete (Networks) [2].

2.4.1 Architettura Generica

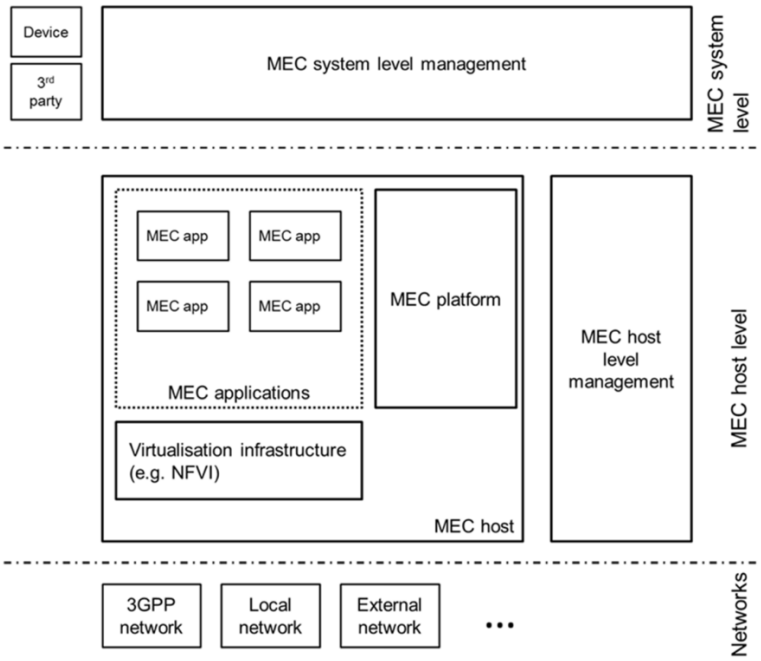

Figura 6 - Multi-access Edge Computing architettura [11]

I componenti principali sono:

- **MEC Host** composto da:
    - MEC Platform
    - MEC App
    - Virtualizzation Infrastructrure
- **MEC System Level Managment**
- **Entità a livello di rete** (es. Dispositivi Mobili)

Più nel dettaglio riporto l'illustrazione di Figura 7, l'architettura MEC con i componenti contenuti nei livelli di sistema e host [11].

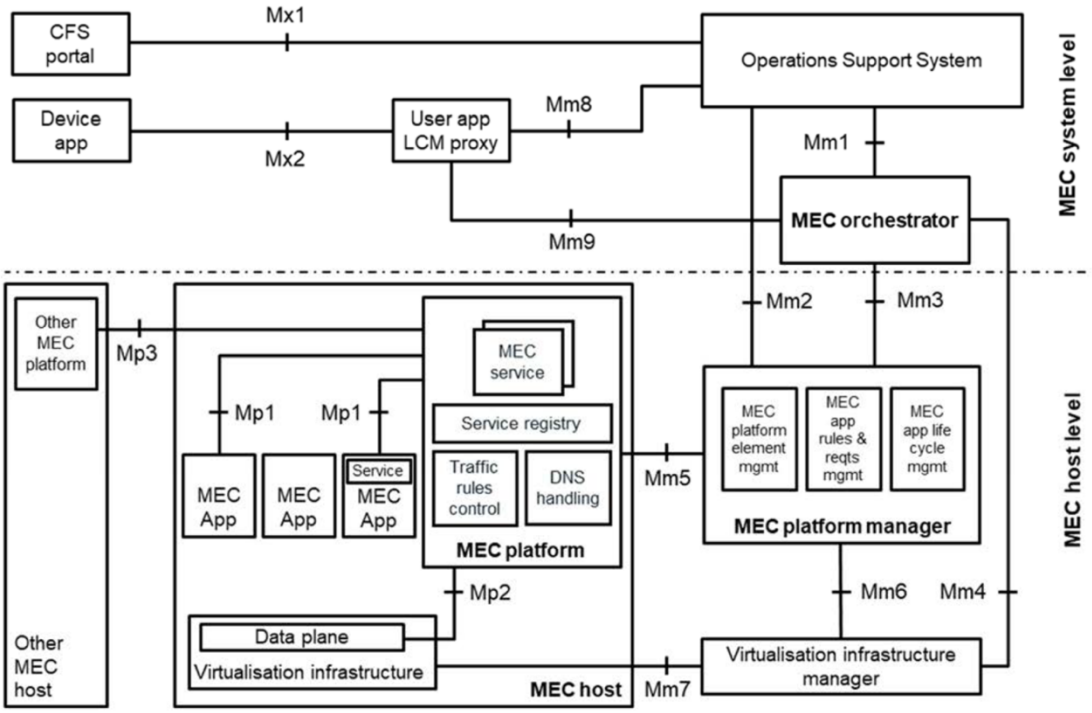

Figura 7 - Multi-access Edge Computing architettura [11]

2.4.2 Elementi Funzionali

L'architettura è composta da questi principali componenti:

Il MEC Host è un'entità fondamentale dell'architettura MEC che integra una piattaforma MEC e un'infrastruttura di virtualizzazione responsabile della fornitura di risorse

computazionali, di memorizzazione e di connettività di rete necessarie per l'esecuzione delle applicazioni MEC. Questa infrastruttura include un piano dati che ha il compito di eseguire le regole di instradamento del traffico definite dalla piattaforma MEC. In pratica, si occupa di dirigere correttamente i flussi di rete tra i vari elementi del sistema, come le applicazioni MEC, i servizi MEC, i server DNS o i relativi proxy, la rete mobile 3GPP, eventuali altre reti di accesso, le reti locali e le reti esterne.

La **piattaforma MEC** svolge un ruolo centrale nel sistema MEC, in quanto responsabile di un insieme eterogeneo di funzionalità fondamentali per l'esecuzione e la gestione delle applicazioni. Alcune di queste sono:

- la fornitura di un ambiente in cui le istanze delle applicazioni MEC possono registrarsi, scoprire, pubblicare, consumare e offrire servizi MEC, inclusi anche quelli esposti da piattaforme appartenenti ad altri sistemi MEC (Caso della federazione);

- la ricezione di regole di traffico dal gestore della piattaforma (MEPM), dalle applicazioni o dai servizi, e l'impostazione delle corrispondenti istruzioni per il piano dati;

- la configurazione di un server o proxy DNS, basata sui record ricevuti dal gestore della piattaforma;

- l'hosting diretto di servizi MEC, eventualmente inclusi quelli standard definiti nella clausola 8 delle specifiche ETSI;

- la fornitura di accesso a informazioni persistenti di archiviazione e dati relativi alla temporalità (es. timestamp);

- il supporto alla registrazione delle istanze applicative, consentendo a queste ultime di fornire e aggiornare dinamicamente le proprie informazioni di runtime all'interno della piattaforma.

A complemento, la piattaforma MEC può includere un gateway API, che abilita le istanze applicative all'accesso diretto alle API dei servizi MEC.

Un'applicazione MEC viene istanziata ed eseguita come componente software virtualizzato, tipicamente all'interno di una macchina virtuale (VM), forniti come immagini software all'interno del pacchetto applicativo. L'esecuzione avviene sopra l'infrastruttura di virtualizzazione messa a disposizione dal sistema MEC.

Una volta istanziata, l'applicazione può interagire con la piattaforma MEC per consumare o offrire servizi MEC, secondo quanto definito nelle specifiche. In aggiunta, l'applicazione può eseguire procedure di supporto legate al proprio ciclo di vita, quali la segnalazione della disponibilità, la gestione dello stato utente in vista di una possibile migrazione o ricollocazione.

L'istanza dell'applicazione può registrarsi presso la piattaforma MEC al fine di esporre le proprie informazioni di runtime. Tale registrazione facilita anche la fase di scoperta dell'applicazione, inclusi i casi in cui l'istanza non sia stata creata direttamente dal sistema di gestione MEC.

Ogni applicazione MEC può avere vincoli tecnologici per l'istanziazione. Ad esempio, risorse computazionali richieste, soglie di latenza massima ammissibile, o servizi MEC. Tali requisiti vengono valutati e convalidati dal sistema di gestione MEC, che può assegnare valori di default qualora essi non siano esplicitamente specificati.

E gestori, che si differenziano in gestori di sistema e di host.

La gestione a livello di sistema troviamo il **MEC Orchestrator**. Esso è incaricato di un insieme di funzioni critiche che garantiscono il corretto funzionamento e la coerenza dell'intero ecosistema MEC.
In particolare, il MEO:

- mantiene una visione globale del sistema MEC: distribuzione degli host MEC, stato delle risorse disponibili, servizi MEC offerti e la topologia del sistema;

- gestisce il processo di onboarding dei pacchetti applicativi, verifica dei requisiti applicativi. Mantiene inoltre un registro dei pacchetti caricati e predispone l'infrastruttura di virtualizzazione per la gestione delle relative applicazioni;

- seleziona l'host MEC più idoneo per l'istanza applicativa, tenendo conto di vincoli quali la latenza, la disponibilità di risorse e la presenza dei servizi MEC richiesti;

- supervisiona l'attivazione e la terminazione delle istanze applicative;

- abilita, se previsto, la ricollocazione delle applicazioni in funzione delle esigenze operative o dei vincoli dinamici della rete;

- coordina le operazioni legate al ciclo di vita delle applicazioni, in particolare l'istanziazione e la gestione evolutiva delle stesse.

La gestione a livello di host invece comprende il MEC Platform Manager e il Virtualisation Infrastructure Manager.

Il MEC Platform Manager (MEPM) ha la responsabilità della gestione operativa delle applicazioni MEC a livello di host. Le sue principali funzioni includono:

- la gestione del ciclo di vita delle applicazioni, dalla loro attivazione fino alla terminazione, con notifica al MEC Orchestrator di eventuali eventi.

- la fornitura delle funzionalità necessarie per il controllo e il monitoraggio delle applicazioni e dei servizi in esecuzione;

- la gestione delle regole e dei requisiti applicativi, che comprende le autorizzazioni di accesso ai servizi MEC, la configurazione delle regole di traffico, la gestione

del DNS e la risoluzione di eventuali conflitti legati alle risorse o alla configurazione.

Inoltre, il MEPM riceve dal gestore dell'infrastruttura di virtualizzazione informazioni relative a malfunzionamenti e metriche prestazionali delle risorse virtualizzate, che possono essere utilizzate per attività di ottimizzazione e manutenzione.

Il Virtualisation Infrastructure Manager è il componente responsabile della gestione delle risorse virtuali a livello fisico e logico. Le sue funzioni principali comprendono:

- l'allocazione, gestione e rilascio delle risorse virtualizzate;

- la preparazione dell'infrastruttura per l'esecuzione delle immagini software delle applicazioni;

- la raccolta di metriche relative alle prestazioni e la segnalazione di eventuali errori o malfunzionamenti delle risorse virtualizzate al MEPM;

2.4.3 Punti di riferimento

All'interno dell'architettura di riferimento i reference points, rappresentano interfacce logiche che permettono la comunicazione tra gli elementi funzionali del sistema. Essi sono fondamentali per garantire l'interoperabilità e l'orchestrazione dei servizi edge, consentendo lo scambio di informazioni e il coordinamento delle operazioni tra piattaforme MEC, orchestratori, applicazioni e infrastrutture di virtualizzazione. Secondo la specifica sono classificati in base al dominio a cui si riferiscono:

- piattaforma MEC (Mp)
- gestione (Mm)
- interfacce esterne (Mx)

Punti di riferimento della piattaforma MEC

- Mp1: Interfaccia principale tra la piattaforma MEC e le applicazioni. Gestisce la registrazione, scoperta e comunicazione dei servizi, attivazione delle regole DNS e supporto alla mobilità applicativa.
- Mp2: Consente alla piattaforma MEC di istruire il piano dati su come instradare il traffico.
- Mp3: Comunicazione di controllo tra piattaforme MEC, utile per la cooperazione tra sistemi in scenari di mobilità e V2X.

Punti di riferimento della gestione MEC

- Mm1: Attiva/termina le applicazioni.
- Mm2: configurazione della piattaforma MEC, la gestione degli errori e delle prestazioni.
- Mm3: Comunicazione tra MEO e MEC Platform Manager per la gestione delle applicazioni e delle regole.
- Mm4: Interfaccia tra MEO e VIM per il monitoraggio e la gestione delle risorse virtuali.
- Mm5: Utilizzata dal MEC Platform Manager per configurare la piattaforma e gestire la ricollocazione delle applicazioni.
- Mm6-Mm9: Supportano le comunicazioni tra i gestori della piattaforma, VIM e proxy di gestione. Alcuni non sono completamente specificati, ma contribuiscono alla gestione del ciclo di vita e della virtualizzazione.

Punti di riferimento verso entità esterne

- Mx1: Permette a terze parti di richiedere l'esecuzione di applicazioni MEC tramite l'OSS.
- Mx2: Usato dalle applicazioni dispositivo per richiedere lo spostamento o l'istanziazione di app nel sistema MEC.

2.4.4 Servizi MEC

In generale un servizio MEC è un servizio che viene messo a disposizione e utilizzato da una piattaforma MEC o da un'applicazione MEC. Quando il servizio è fornito da un'applicazione,

può essere registrato nel registro dei servizi alla piattaforma MEC tramite il punto di riferimento Mp1.

I principali servizi MEC standardizzati sono: Radio Network Information Service, Location Service, Traffic Management Services e Application Mobility Service.

2.4.4.1 Radio Network Information Service

Radio Network Information Service (RNIS). Fornisce alle applicazioni informazioni aggiornate sullo stato della rete radio, includendo:

- condizioni di rete radio attuali;
- statistiche e misurazioni del traffico utente;
- informazioni sugli utenti connessi UE servite dai nodi radio associati all'host MEC;
- notifiche relative a variazioni del contesto delle UE.

Le informazioni sono fornite con granularità adeguata, ad esempio per singola cella o singolo dispositivo.

2.4.4.2 Location Service

Location Service. Espone dati relativi alla posizione, tra cui:

- localizzazione delle UE servite dal nodo MEC;
- lista delle UE presenti in una specifica area;
- posizione dei nodi radio connessi all'host MEC.

La posizione può essere espressa come coordinate geografiche, ID di cella o altre modalità equivalenti.

2.4.4.3 Traffic Management Services

Traffic Management Services. Servizi opzionali per l'ottimizzazione del traffico:

- Bandwidth Management (BWM): consente l'allocazione e la priorità della larghezza di banda.

- Multi-access Traffic Steering (MTS): permette il bilanciamento, la duplicazione o la suddivisione del traffico su più interfacce di rete d'accesso in modo trasparente per l'applicazione.

2.4.4.4 Application Mobility Service

L'Application Mobility Service è uno dei servizi standardizzati e specificati nel documento ETSI MEC 021 [12] dalla piattaforma MEC ETSI per supportare la migrazione dinamica delle applicazioni MEC, in particolare in scenari con utenti mobili. Il suo scopo è fornire un'interfaccia uniforme che consenta alle applicazioni di:

- registrarsi presso un gestore della mobilità,
- ricevere notifiche su eventi rilevanti,
- sincronizzare il proprio stato con un'istanza migrata,
- gestire la continuità del servizio in modo trasparente per il client finale.

Ricordando che l'utente non percepisce interruzioni anche durante la migrazione. Che ci sia scalabilità cioè gestione indipendente dallo stack dell'applicazione. E infine la definizione ETSI assicura interoperabilità tra vendor/operatori diversi [12].

Il funzionamento dell'AMS si basa su un meccanismo di subscription e notifica. Le notifiche hanno dei tipi ben definiti per far capire in che stato si trovi l'applicazione, troviamo:

- INTERHOST_MOVEOUT_TRIGGERED, qui sappiamo che c'è in corso una migrazione di un'applicazione.
- INTERHOST_MOVEOUT_COMPLETED, sappiamo che l'applicazione ha completato la migrazione, il completamento si intende anche di passaggio di stato tra l'applicazione vecchia e la nuova.
- INTERHOST_MOVEOUT_FAILED, il trasferimento è fallito per qualche ragione.

In un caso generale possiamo dividere in fasi il funzionamento del servzio:

Registrazione e subscription: L'applicazione MEC (MEC-App) si registra presso l'AMS, indicando oltre alle proprie informazioni le preferenze di notifica, cioè che tipo

di notifiche. Successivamente si procede con la subscription che può essere fatta non solo da un'applicazione ma da qualsiasi entità che è interessata al monitoraggio di un'applicazione, si invia come sopra il tipo di notifiche a qui si è interessato, l'uri di callback e di quali applicazioni si vuole ricevere le notifiche. L'AMS sia per la registrazione che la subscription se avvenuta correttamente risponderà con un id univoco, questo id potrà essere usato poi per modificare e quindi fare un UPDATE o una DELETE sia della registrazione che della subscription.

Trigger della migrazione: Quando il sistema MEC rileva un evento di mobilità, invierà all'AMS l'intendo di un'applicazione di spostarsi, esso poi notificherà le app e i moduli iscritti a quell'applicazione che c'è una migrazione, attraverso lo stato INTERHOST_MOVEOUT_TRIGGERED, attivando il meccanismo di replica o spostamento dell'applicazione. A fine dello spostamento avremo due applicazioni, la vecchia e la nuova. L'utente sarà ancora collegato alla vecchia applicazione, in quando la nuova non è ancora pronta.

Context transfer: L' applicazione in corso di migrazione trasferisce il proprio stato interno alla nuova istanza tramite una comunicazione diretta tra MEC-App. Questa fase è cruciale per mantenere la QoS. Le info su porta e indirizzo della nuova applicazione sono inviate sempre tramite notifiche AMS, dove ci sarà un nuovo campo contenente le informazioni su il targetHost. Questo rende possibile la comunicazione diretta.

Completamento e unsubscription: Una volta che la nuova istanza è attiva e lo stato sincronizzato, l'applicazione che riceve lo stato invia la registrazione specificando che lo stato è stato trasferito. L'AMS notifica il completamento con INTERHOST_MOVEOUT_COMPLETED. La vecchia applicazione cancella la sottoscrizione all'AMS e procede con la sua eliminazione.

2.5 MEC Federation

Un ambiente MEC è composto da diversi attori che ne fanno parte, come, gli operatori di rete cioè proprietari delle infrastrutture, fornitori di servizi, sviluppatori di applicazioni e integratori di sistema. Alcune volte queste entità sono la stessa, ma nella maggior parte

dei casi, in scenari reali sono tutti diversi. Questo porta al ragionare per i sistemi MEC alla coordinazione e collaborazione per eseguire una applicazione e aver la possibilità di mantenere quelli che sono i loro requisiti come descritto nei capitoli precedenti.
Uno dei servizi principali da mantenere è quello della continuità dell'applicazione anche in scenari multi-operatore. Questa forma di cooperazione è detta **Federazione MEC.**

La Federazione MEC è un argomento ancora in sviluppo e in discussione, l'ETSI ha pubblicato alcuni documenti come ETSI GS MEC 002 [8], ETSI GS MEC 003 [11] dove parla in generale della Federazione e dei casi d'uso. In seguito, ha pubblicato nel ETSI GS MEC 040 [13] una guida per gli sviluppatori di applicazioni e sistemi MEC per implementare una Federazione conforme alle specifiche MEC.

ETSI GR MEC 035 [2] introduce la federazione MEC, come "un modello federato di sistemi MEC che consente l'uso condiviso di servizi e applicazioni MEC". Qui i componenti "federano" le proprie risorse di edge computing, offrendo/esponendo le proprie capacità di servizio MEC, non solo per il consumo reciproco, ma anche offrendole a sviluppatori di applicazioni e clienti finali.

2.5.1 Introduzione alla federazione

Per capire la federazione bisogna aver chiaro lo scenario reale. GSMA OPG ha pubblicato nel documento "Operator Platform Telco Edge Requirements" le opzioni di distribuzioni dell'Operator Platform (OP). L'OP è un componente che permette di mantenere la continuità delle applicazioni tra più operatori e quindi nel caso che operatori lavorino nel campo federato.

Esistono due opzioni, confrontabili visivamente in Figura 8, la prima che ogni operatore abbia un OP che può comunicare e federarsi con l'altro dell'altro operatore. Un altro che un operatore non ha un OP proprio ma lo condivide con altri. In seguito, sono riportate le illustrazioni di entrambe le opzioni.

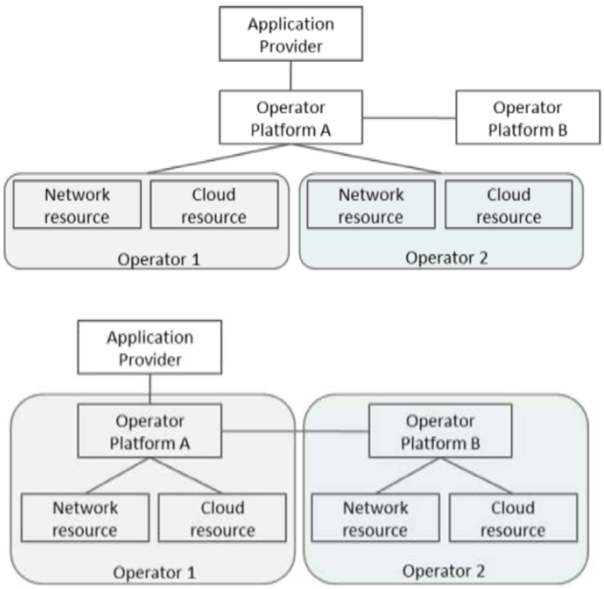

Figura 8 - Operatori con proprio OP e condiviso [3]

Da questa prospettiva è chiaro che un operatore che abbia un OP debba avere anche un orchestratore che gestista tutto e che in particolare permetti ad altri operatori nei casi federati di potersi connettere [3].

2.5.2 Architettura

In base all'architettura del sistema MEC descritta nei capitoli precedenti si espone la variante che consente la Federazione MEC, visibile in Figura 9.
Si aggiunge all'architettura il Federatore che potrebbe avere anche un broker (questa parte verrà discussa in seguito) e i punti di riferimento che portano ad avere nuove funzionalità come:

- Registrazione da parte dell'orchestratore delle informazioni sul sistema MEC
- Rilevamento del sistema MEC
- Capacità di broker per la scoperta di federatori (non sempre)
- Scambio di informazioni
- Gestione ciclo di vita dell'applicazione
- Monitoraggio dell'applicazione

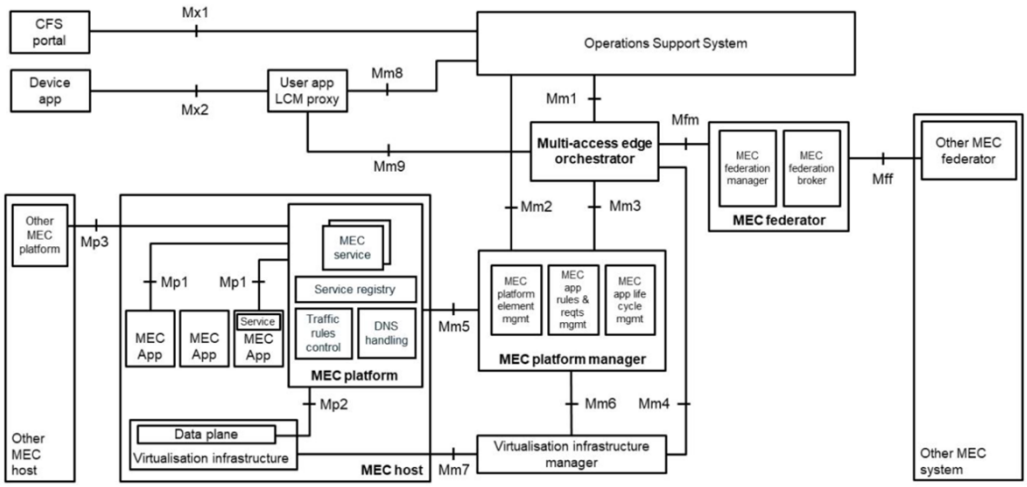

Figura 9 - Architettura con federazione [11]

Partendo dall'architettura generica descritta nei capitoli successivi vengono introdotti nuovi elementi:

- MEC Federator (MEF)
- MEC Federation Manager (MEFM)
- MEC Federation Broker (MEFB)

Il **MEC Federator MEF** è un'entità che permette di abilitare la federazione dei sistemi MEC.

Chiamiamo **MEFM** l'entità del Federator che scambia informazioni con un altro federator. In particolare, dopo essersi conosciuti si scambiano informazioni sul sistema MEC al quale fa riferimento. Inviando servizi e applicazioni contenuti nel sistema MEC, in modo da sapere prima cosa contiene o nel caso non dovesse saperli può chiederli al momento opportuno. Gestisce tutto lo scambio di messaggi di discovery di un'applicazione che della sua migrazione. Nello scenario in qui è presente il **MEC Federator Broker** il federator agisce anche da broker. Quindi abbiamo una prima fase di conoscenza, i federator conoscono il broker al quale si registrano. I messaggi sono tutti inviati al broker, sarà lui poi a inviare a tutti o solo al destinatario il messaggio. Questo permette un'aggiunta e rimozione dinamica dei federator e quindi dei sistemi MEC. In seguito, verrà discusso l'utilizzo o meno del broker. Il MEF è connesso direttamente al MEO tramite il punto di riferimento Mfm, anche qui abbiamo una scoperta del MEF da parte del MEO con registrazione, nelle specifiche questa interazione è stata implementata

tramite i diagrammi semplificando l'interazione e presumendo che il MEO conosce il MEF e invia una registrazione contenente le info del sistema al quale fa riferimento. Può cancellare la registrazione o aggiornarla con nuove info [11].

Gli altri elementi costituenti dell'architettura rimangono presso a poco simili, di alcune differenze solamente nelle interazioni e delle nuove funzionalità. Si cita il MEO che è l'elemento più variante dove ora considera non solo i MEC Host contenuti nel suo MEC System ma anche quelli esterni. Ha il collegamento al suo MEF al quale chiederà un'applicazione o un servizio del quale il suo MEC System non può rispondere oppure invierà direttamente una migrazione.

Punti di riferimento

- Mff: Il punto di riferimento Mff tra i MEF all'interno della federazione MEC viene utilizzato per la condivisione di informazioni (ad esempio informazioni sul sistema MEC).
- Mfm: Il punto di riferimento Mfm tra MEO e MEF consente la condivisione delle informazioni di un sistema MEC.

2.5.3 Broker e P2P

Finora si è dato per scontato che il sistema includesse un host Federator in grado di operare anche come **Broker Federator MEFB**. Infatti, questa opzione è una delle due possibili di collegamento tra Federator. Utilizzando il broker, come in Figura 10, si suppone di avere sempre e che non può venir meno l'entità del broker che permette il funzionamento di tutto il sistema federato.

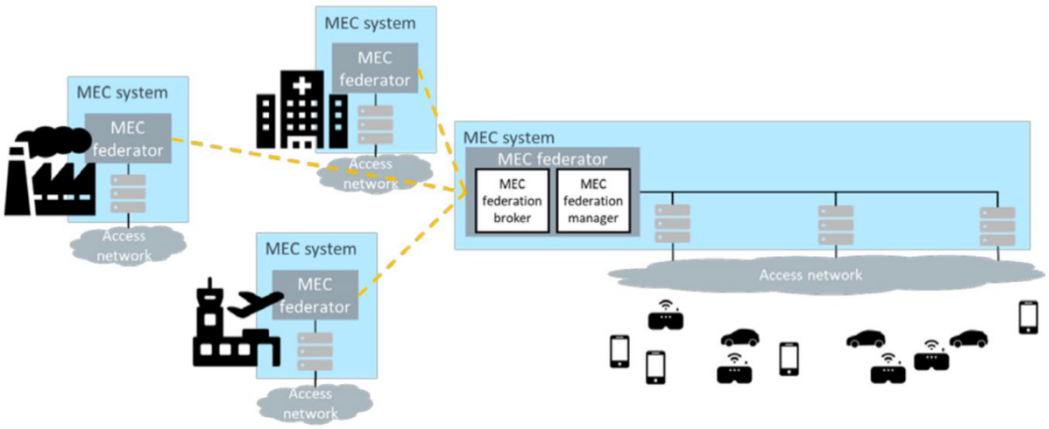

Figura 10 - MEF gestiti da Broker [3]

L'altra opzione, in Figura 11, prevede che ogni Federator sia indipendente e si colleghi direttamente a un altro, tramite la cosiddetta connessione **P2P** (peer-to-peer). Questo modello permette di non avere un punto singolo di guasto come accade per il broker, dove se il broker non funziona non funziona il sistema. E quindi abbiamo più flessibilità e autonomia da parte del Federator. D'altro canto, però l'utilizzo del P2P porta ad avere una gestione del sistema e delle connessioni più complessa. Il broker permette di avere costantemente aggiornate le info dei federator e quindi velocizzare anche la scoperta di servizi e applicazioni.

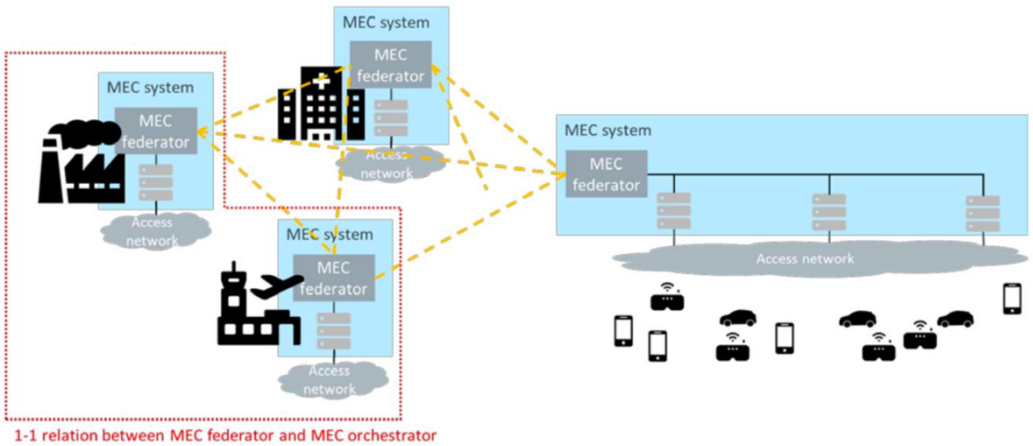

Figura 11 - MEF gestiti con architettura P2P [3]

La richiesta di un sistema MEC nel caso P2P potrebbe risultare anche poco performante se il federator richiedente non conoscesse il federator richiesto. Passando le info a chi conosce aspettando una risposta. Invece con il broker si può anche pensare a gerarchie più complesse, ad esempio, come la suddivisione in aree, regioni e continenti.

2.5.4 Casi federati a singolo operatore

Nel caso base si presuppone che ogni sistema MEC sia distribuito in una struttura che può essere un ospedale, una fabbrica ecc… In Figura 12 viene mostrato l'esempio. Ogni sistema mec ha il suo compito, servizi e applicazioni eseguite.

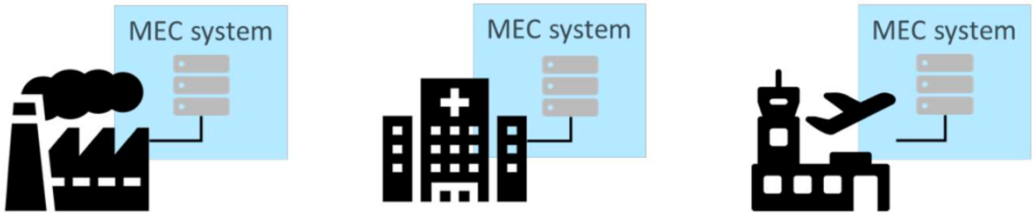

Figura 12 - MEC System nel caso reale [3]

Quello che accade nel singolo operatore che, come nelle Figure precedenti 10 e 11, i sistemi solo collegati tra loro con connessione o P2P o Broker tramite i federator. Possono comunicare e interagire come illustrato in Figura 13.

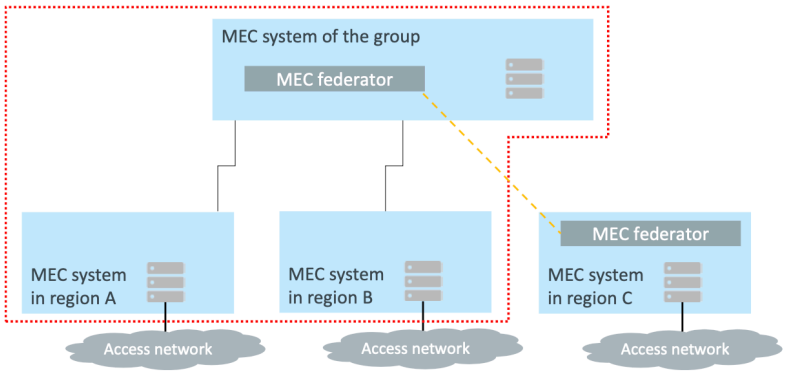

Figura 13 - Interazione tra MEC System [3]

Un operatore può anche decidere una divisione in gruppi o regione come mostrata in figura dove abbiamo sempre i federator che si occupano della gestione delle interazioni tra i mec system.

2.5.5 Casi federati a multi-operatore

Come descritto nel GSMA OP PRD, il principale vantaggio della federazione è quando abbiamo sistemi MEC in diverse regioni o con diverse reti. Anche se il consumatore si sposta si permette di mantenere la continuità del servizio applicativo senza problemi.

Inoltre, sappiamo anche che un sistema MEC non per forza è dello stesso operatore quindi abbiamo operatori diversi come in Figura 14. Quindi un utente collegato a un operatore può richiedere il servizio a un altro operatore che ha quel servizio nel suo MEC System. Oppure semplicemente un piccolo operatore che non può gestire i servizi si associa a uno più grande che li gestisce i servizi.

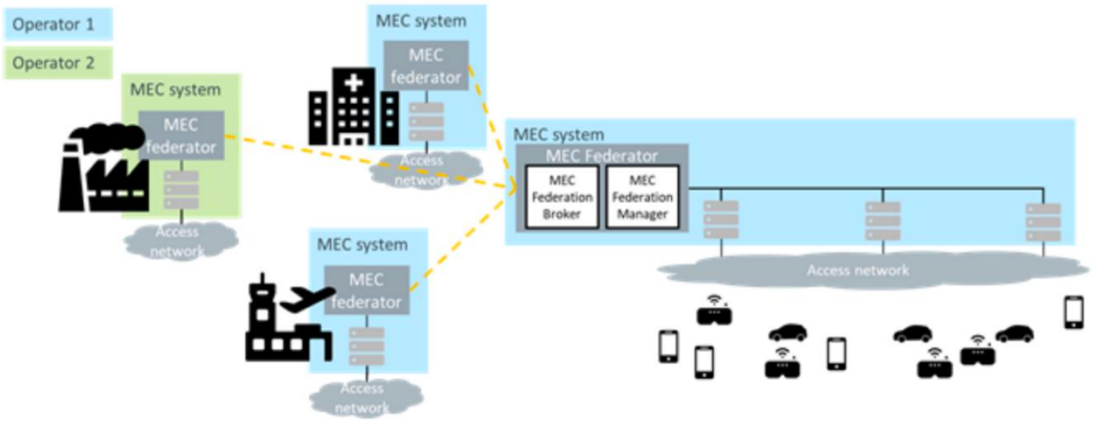

Figura 14 - Cooperazione MEC Multioperatore [3]

2.5.6 Ruolo dell'AMS in ambienti federati

In un contesto federato, dove abbiamo sistemi diversi e quindi anche **AMS** diversi, la gestione diventa ancora più critica poiché bisogna coordinare la migrazione anche tra nodi MEC appartenenti a domini diversi. Non solo nel contesto **multi-operatore** anche nel contesto **intra-mec**. Si parla di contesto intra-mec quando abbiamo più sistemi MEC appartenenti allo stesso operatore ma distribuiti o con tecnologie diverse come 5G e Wi-Fi oppure geograficamente diversi. Questo però mantenendo l'esperienza utente stabile, anche in presenza di cambiamenti infrastrutturali.

Discutendo questo aspetto sono emerse diverse possibili soluzioni, da un lato l'estensione dell'AMS con nuove funzionalità di notifica inter-dominio, dall'altro strategie che mantengono inalterate le specifiche ETSI e demandano parte della logica alle applicazioni. Nei capitoli successivi verranno discusse nel dettaglio le possibili soluzioni e le scelte progettuali effettuate in questa tesi.

2.6 OMNeT++

Per la parte sperimentale è stato utilizzato OMNeT++, un simulatore a eventi discreti modulare e componibile, ampiamente diffuso nella ricerca per la modellazione di reti di comunicazione. OMNeT++ è distribuito con licenza pubblica accademica e offre un IDE basato su Eclipse. [4]

La sua architettura si fonda su un kernel di simulazione e su una struttura gerarchica di moduli: moduli semplici, scritti in C++, che implementano il comportamento dei singoli componenti; moduli composti, definiti attraverso il linguaggio NED (Network Description Language), che consentono di combinare più moduli semplici o composti per costruire reti complesse.

Il codice NED rappresenta quindi la "mappa logica" della rete, specificando nodi, collegamenti e parametri. Grazie a questa astrazione, la progettazione della rete è separata dall'implementazione del comportamento dei singoli componenti (scritti in C++). Ciò favorisce modularità e riuso: lo stesso modulo può essere istanziato più volte in topologie diverse semplicemente modificando il file NED, senza intervenire sul codice.

3 Related Work

Nei capitoli precedenti sono stati introdotti i concetti fondamentali legati all'evoluzione delle reti mobili fino al 5G, al paradigma del Multi-access Edge Computing e alle sue estensioni in contesti veicolari. Sono state inoltre descritte le principali architetture e i servizi standardizzati dall'ETSI, con particolare attenzione alle problematiche di migrazione applicativa e di federazione multi-operatore.

A questo punto, risulta naturale collocare tali nozioni nel quadro delle ricerche esistenti. Il presente capitolo raccoglie e discute i principali lavori presenti in letteratura, con l'obiettivo di evidenziare le ricerche su VEC e MEC, che mostrano come il paradigma edge si adatti al contesto veicolare, superando le limitazioni del VCC e abilitando applicazioni critiche a bassa latenza; agli studi dedicati alla migrazione delle applicazioni MEC, nei quali sono state proposte diverse soluzioni architetturali e algoritmiche per ridurre downtime e latenza durante l'hand-over; i contributi sulla federazione MEC, che analizzano la cooperazione tra operatori diversi e i modelli di interoperabilità, successivamente recepiti e formalizzati negli standard ETSI.

Questo capitolo, dunque, rappresenta un ponte tra la parte introduttiva e di background e le sezioni successive di System Design e Implementazione: consente di comprendere quali approcci siano già stati esplorati in letteratura e quali lacune rimangano ancora aperte, fornendo così la base per la proposta sviluppata in questa tesi.

3.1 VEC e MEC

Il paradigma del Vehicular Edge Computing è stato introdotto per supportare i requisiti stringenti di latenza, affidabilità e capacità computazionale delle applicazioni veicolari in scenari V2X. A differenza del Vehicular Cloud Computing in cui l'elaborazione era delegata alle risorse eterogenee messe a disposizione direttamente dai veicoli, il VEC colloca le risorse di calcolo e storage in prossimità dell'infrastruttura stradale. In questo modo i veicoli possono eseguire lo scarico computazionale delle applicazioni verso l'edge riducendo la latenza end-to-end e migliorando l'affidabilità dei servizi critici, come il cooperative driving o la gestione di mappe HD [10];

La letteratura ha mostrato come il VEC offra vantaggi sostanziali rispetto al VCC, garantendo stabilità, maggiore capacità e QoS prevedibile come nel documento Hou et al. (2016) [14]. Tuttavia, come descritto da Mao et al., (2017) [15], emergono nuove sfide

legate a mobilità elevata dei veicoli, handover frequenti e necessità di migrazione dinamica delle applicazioni tra nodi edge diversi lungo il percorso. Queste problematiche collegano direttamente il VEC al paradigma standardizzato del MEC ETSI, che fornisce un'infrastruttura e un framework interoperabile per la gestione delle applicazioni edge, compresa la loro migrazione e federazione multi-operatore.

Il MEC è stato introdotto dall'ETSI come soluzione standard per portare risorse di calcolo e storage più vicino all'utente finale, sfruttando l'infrastruttura di rete già esistente [16] [17], sottolineando come il MEC non solo riduca la latenza e il carico, ma abiliti anche un nuovo ecosistema di applicazioni distribuite.

La letteratura ha evidenziato diverse problematiche che limitano l'efficacia del MEC in scenari reali:

- **Posizionamento delle applicazioni**: studiato in Abbas et al. (2018) [18] mostrano come il problema del posizionamento ottimale delle MEC App sia un problema NP-hard e richieda approcci euristici o basati su apprendimento per bilanciare carico e latenza.
- **Scalabilità e multi-tenant**: il MEC deve supportare più applicazioni e più provider simultaneamente. Questo solleva problemi di isolamento, orchestrazione e bilanciamento. Risultati ottenuti in Taleb et al. (2017) [16].
- **Interoperabilità multi-operatore**: in 5GAA (2021) [19] sottolineano come, in assenza di accordi di federazione, un'applicazione MEC sia confinata all'infrastruttura del singolo operatore, limitando la continuità del servizio in scenari di mobilità.

Queste limitazioni mostrano come il MEC, pur rappresentando una tecnologia chiave per ridurre la latenza e avvicinare le risorse all'utente, necessiti di meccanismi aggiuntivi per garantire continuità del servizio in mobilità. In particolare, la questione dell'interoperabilità multi-operatore mette in evidenza l'impossibilità, allo stato attuale, di mantenere attive le applicazioni quando l'utente si sposta al di fuori del dominio dell'operatore che ospita l'istanza MEC. La ricerca ha quindi individuato nella migrazione delle applicazioni e nei modelli di federazione tra sistemi MEC le soluzioni più promettenti per superare tali limiti, consentendo di riallocare dinamicamente le MEC App tra domini diversi e assicurare la service continuity anche in scenari multi-operatore.

3.2 Migrazione di applicazioni MEC

La migrazione delle applicazioni nel contesto del MEC è una tecnica chiave per garantire continuità del servizio e performance ottimali in presenza di mobilità o variazioni della domanda. Uno dei primi studi in questo ambito è il lavoro di Machen et al. (2017), che propone un framework a livelli per la migrazione live di servizi incapsulati in VM o container, riducendo significativamente il downtime grazie ad una stratificazione efficace [20]. Un passo avanti è rappresentato da Ngo et al. (2020), i quali introducono un meccanismo coordinato tra migrazione del container e handover della stazione base; la loro soluzione consente riduzioni del 30–40 % del downtime e del 13–22 % nella latenza end-to-end [21].

Più recentemente, Zhang et al. (2025) hanno fornito una survey sistematica delle strategie di "real-time service migration" nei sistemi edge: illustrano modelli architetturali, motivazioni, tecniche e scenari d'applicazione come smart city e smart manufacturing.

Queste ricerche evidenziano come la migrazione live di container sia oggi supportata da architetture multilivello, algoritmi di sincronizzazione avanzati e strategie intelligenti di orchestrazione, fondamentali per servizi con requisiti stringenti di latenza e disponibilità.

3.3 Federazione MEC

I sistemi MEC hanno come limite che operano singolo operatore e non in modalità di collaborare con più operatori, in merito a questo ci sono stati molti documenti e paper proposti che hanno puntato alla collaborazione tra i sistemi. In particolare, nello studio del mantenimento di un servizio o un'applicazione attiva. I primi lavori non citano la Federazione e propongono un'architettura di più sistemi MEC che collaborano tra loro con diversi metodi. Il primo metodo proposto da Hosseini et al. (2020) [22] parte dal problema che i veicoli in movimento attraversano paesi diversi e devono continuare a utilizzare un servizio, utilizzando il sistema MEC prevedono un collegamento in fibra diretto tra MEC System diversi e introdurre lo scambio di messaggi e autenticazione. L'applicazione riceve dal nuovo MEC System un token di autenticazione e può comunicare con il nuovo sistema. Il MEC System viene a conoscenza del dispositivo o veicolo tramite una notifica MQTT. In particolare, l'architettura prevede un broker MQTT con sottoscrizioni per geo zona attraversata. Il veicolo invierà info sulla sua posizione e chi è sottoscritto in quell'area riceverà la notifica. L'utilizzo del broker MQTT prevede una maggiore scalabilità del sistema soprattutto in condizioni sfavorevoli quando

un sistema non funziona. Inoltre, possiamo pensare a una replica dei messaggi MQTT in più domini.

Un altro documento Onjapera et al. (2018) [23], invece, parla della collaborazione tra sistemi MEC ancora tramite un collegamento diretto tra essi. Può avvenire in diversi modi: Direttamente tra stazioni base vicine, tramite la rete core tra operatori diversi attraverso le loro core network, via backend cloud se i dati non sono troppo urgenti. E il meccanismo di migrazione avviene secondo due modelli in base al tipo di applicazione. Esistono due tipi di applicazioni, una che gestisce i dispositivi e in particolare le auto a unirsi nei sistemi MEC. L'altra serve per la raccolta di dati sensoristici delle auto.

La prima formalizzazione del concetto di federazione è stata fatta da Cao et al. (2019) [24], dove si introduce il modello **dell'Edge Federation**, si introduce il problema della cooperazione tra diversi provider di infrastruttura e cloud. Il documento offre un'analisi dei costi per le risorse, in particolare una riduzione del circa 25% utilizzando il modello federato. Con il primo studio ufficiale ETSI si introduce il **MEC Federation** nello standard, il documento ETSI GR MEC 035 (2021) [2] introduce tutti quelli che sono i requisiti per gli operatori, sviluppatori da mantenere per poter cooperare. Per quanto riguarda la migrazione dell'applicazione ci sono molti studi a riguardo, in particolare si concentrano sul modo di trasferimento dello stato in modo sicuro, alcuni articoli parlano dell'utilizzo di proxy o API. Sempre dai documenti ufficiali, gli standard viene proposta la specifica dell'AMS Application Mobility Service, dove però non viene specificato il suo funzionamento all'interno della federazione [12].

Rasol et al. (2024) [25] ha formalizzato un'implementazione utilizzando il concetto di federazione basata sul modello Open Gateway NaaS e su API CAMARA. Viene implementata un'architettura dove i MEC Host non si collegano più tra loro, rimangono nell'interno e comunicano all'esterno con i federator. Il processo funziona con la richiesta di migrazione di un'applicazione, si inviano le info dell'applicazione e si crea l'applicazione nel nuovo sistema. Si trasferisce lo stato e si fa lo switch. Tutto questo attraverso l'utilizzo di API. Questo modello ha portato problemi soprattutto per la gestione del trasferimento dello stato. Per motivi di sicurezza e motivi di latenza per continuità del servizio. Per migliorare la latenza si sono studiate tecniche che prevedono

lo studio dei movimenti dei dispositivi, durata dei servizi, memoria e numero di repliche e dimensione della copertura.

Frangoudis et al. (2018) [26] ha visto che esistono due strade: spostare l'applicazione oppure replicare l'applicazione. La migrazione migliora la memoria ma porta un downtime e consumo della banda. L'altra peggiora la memoria, ma migliora la banda. Si è arrivati a due principali risultati che per i servizi lunghi è meglio usare la migrazione e per quelli brevi la replica. Un altro lavoro che studia le tempistiche e il downtime è stato proposto da Hathibelagal et al. (2023) [27] proponendo innanzitutto tre strategie di migrazione. SFB, rapida ma non preserva lo stato; FCCP, preserva lo stato; FullSP una migrazione live completa. Attraverso esperimenti soprattutto fatti con dati di movimento di veicoli reali è emerso che il miglior compromesso è adottare il FCCP. Le altre soluzioni o non salverebbero lo stato o sarebbero troppo lente.

Alcuni studi invece pongono l'attenzione alla sicurezza durante la migrazione; infatti, il trasferimento dello stato dovrebbe avvenire con una connessione affidabile, veloce e sicura. Nel documento Niewolski et al. (2022) [28] gli autori definiscono una procedura di migrazione con sicurezza in 7 passi: identificare la nuova area, stabilire la connessione, creare un contesto, stabilire un canale sicuro, duplicare i dati e infine eliminare il vecchio contesto. Inoltre, viene giustificato il motivo di questa sicurezza facendo riferimento ai casi reali di utilizzo come: comunicazioni di emergenza, banche, sanità e altro.

4 System Design

L'analisi dei documenti e dei requisiti ha portato alla definizione di una possibile architettura federata, capace di supportare la migrazione delle applicazioni in scenari multi-operatore. In questa fase di system design l'attenzione si è concentrata sull'adattamento del modello di riferimento definito dall'ETSI a un contesto sperimentale. L'obiettivo è stato quello di definire in maniera chiara i nuovi elementi e le interazioni. Sono stati analizzati e modellati i principali flussi descritti nella documentazione.

Dal punto di vista concettuale, la federazione si inserisce come livello superiore rispetto ai singoli MEC System. Ogni sistema mantiene le proprie funzionalità locali come il MEC Orchestrator, MEC Host, MEC Platform, AMS, ma è esteso con un nuovo elemento: il MEC Federator. Questo componente svolge il ruolo di interfaccia verso altri domini e abilita la cooperazione secondo procedure standardizzate di registrazione, discovery e migrazione di applicazioni.

L'architettura proposta prevede la presenza, in ciascun dominio, di un Federator collegato al rispettivo MEC Orchestrator. I Federator comunicano tra loro attraverso i reference point definiti da ETSI, scambiando informazioni su sistemi, servizi e applicazioni disponibili.

Durante la definizione dell'architettura, sono emerse alcune sfide principali:

- **Introduzione del Federator**: l'inserimento di questo nuovo elemento architetturale ha imposto una riflessione su come integrarlo con gli orchestratori esistenti, distinguendo chiaramente i ruoli di Federator Client e Federator Broker e il ciclo di vita delle loro interazioni.
- **Gestire le procedure di discovery**: è stato necessario modellare le interazioni attenendosi ai flussi previsti dalle API ETSI, in modo da mantenere compatibilità con la reference architecture e favorire una futura interoperabilità reale.
- **Gestione della migrazione applicativa**: estendere il meccanismo di Application Mobility Service al contesto federato ha richiesto di considerare scenari in cui lo stato dell'applicazione viene trasferito da un MEC Host a un altro appartenente a domini differenti.

4.1 Federator

Il MEC Federator rappresenta l'elemento principale dell'architettura della Federazione. Dalle specifiche ETSI, esso abilita l'interoperabilità tra MEC System appartenenti a domini diversi, permettendo la condivisione di servizi e applicazioni in scenari multi-operatore e multi-dominio. Nel capitolo 2.5.3 discutiamo del fatto che esistono due modelli principali di interconnessione: peer-to-peer e broker-based. Nel presente lavoro si è scelto di focalizzarsi sul secondo approccio, in quanto più semplice da implementare e scalare: l'introduzione di un Federator Broker riduce la complessità di gestione delle connessioni, evitando la necessità di stabilire legami diretti uno-a-uno fra tutti i domini coinvolti.

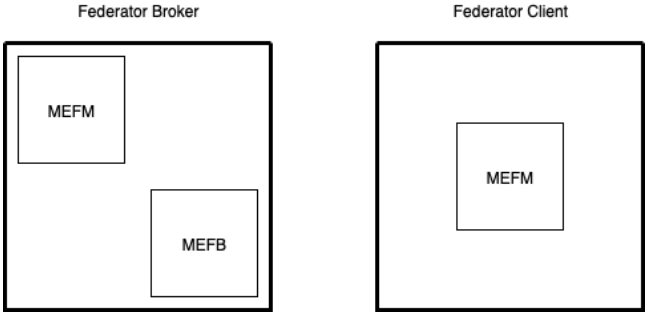

Figura 15 - Architettura logica del FederatorBroker e Federator Client

Distinguiamo dunque due tipi di Federator, il federator con le capacità di broker e il federator con capacità di client come in Figura 15. Nell'implementazione come da specifiche identifichiamo il Federator Client un modulo contenente un'applicazione che gestisce tutte le interazioni sia con il broker che con altri client federator per la gestione della federazione. Invece il Federator Broker come un modulo contenente due applicazioni, la prima uguale a quella client e la seconda l'applicazione per la gestione del broker, cioè la gestione centralizzata delle informazioni di federazione e al loro instradamento verso i vari MEC System.

Il ciclo di vita delle **interazioni fra client e broker** segue un modello di tipo request/response e sono: Registrazione, Aggiornamento, Eliminazione. I diagrammi sono in Figura 16.
All'avvio il client si registra al broker, inviando le informazioni sulla federazione, cioè le specifiche che descrivono il MEC System di riferimento del Federator.

Queste info possono essere aggiornate con un update inviando sempre le info. E l'eliminazione della registrazione.

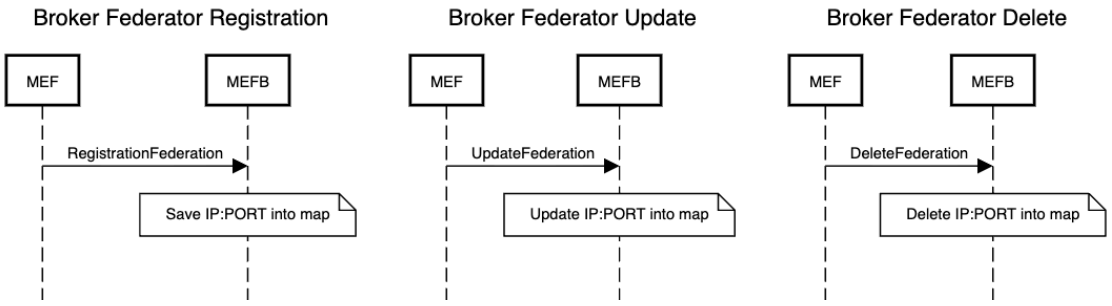

Figura 16 - Diagrammi di sequenza per Registrazione, Aggiornamento e Cancellazione

Il broker, in queste interazioni mantiene una tabella interna contenente i metadati dei sistemi federati (indirizzi IP, porte, descrittori di servizi) e li rende disponibili per procedure successive come il MEC system discovery o la MEC application/service discovery. In questo senso, il broker non si limita a un semplice archivio, ma svolge anche un ruolo di intermediazione attiva per inoltrare richieste e risposte tra federatori diversi. Questa architettura a due livelli riflette anche le considerazioni di ETSI GR MEC 035 [2], dove si distingue fra Federation Manager (gestione diretta e punto di contatto con l'orchestratore) e Federation Broker (hub per scenari multi-operatore). L'adozione del modello broker-based non solo semplifica le interazioni, ma migliora la scalabilità, permettendo l'integrazione incrementale di nuovi MEC System senza dover riconfigurare quelli esistenti.

4.2 Discovery del MEC System

Il Federator mantiene la conoscenza dei MEC System a esso associati, fungendo da punto di intermediazione fra domini diversi. Nella specifica ETSI GS MEC 040 [13] è previsto che un MEC Federator possa interagire con più MEC Orchestrator, quindi più MEC System, questo è uno scenario complesso, consentendo al federator di rappresentare simultaneamente più sistemi MEC. Questa estensione risulta particolarmente utile in contesti multi-operatore, soprattutto quando abbiamo una divisione dei sistemi sotto forma di domini e aree. Nell'implementazione si è scelto di ridurre la complessità logica e di predisporre il collegamento del Federator a un singolo MEO.

Il processo di scoperta dei MEC System si basa su procedure analoghe a quelle di registrazione tra federator, seguendo lo schema registrazione, aggiornamento e cancellazione come si può vedere dalla Figura 17.

Registrazione: ciascun MEO invia al federator le informazioni descrittive del proprio MEC System.

Aggiornamento: quando un MEC System modifica le proprie caratteristiche

Cancellazione: quando un MEC System non è più attivo o viene rimosso dalla federazione

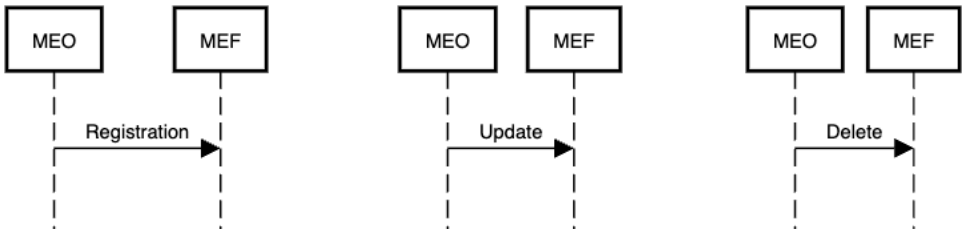

Figura 17 - Diagrammi di sequenza per Registrazione, Aggiornamento e Cancellazione

Grazie a questo meccanismo, il federator è in grado di pubblicare una vista unificata dei MEC System disponibili nella federazione, abilitando funzionalità come il system discovery, il service discovery e la MEC application instance discovery.

Un aspetto rilevante è che, in presenza di più orchestratori, il federator deve essere in grado di gestire conflitti e priorità, ad esempio quando due sistemi pubblicano servizi simili ma con politiche di accesso differenti. Questo implica la necessità di definire meccanismi di policy enforcement e di sicurezza, che sono demandati alle specifiche ETSI e agli accordi di federazione tra operatori.

Nella Figura 17 sono riportati i diagrammi che descrivono in dettaglio le interazioni di registrazione, aggiornamento e cancellazione per la scoperta del MEC System.

4.3 Migrazione di una applicazione

La mobilità degli utenti, specialmente in scenari veicolari, introduce la necessità di migrare le applicazioni da un MEC Host a un altro per garantire continuità di servizio e QoS. ETSI affronta questo problema tramite il servizio AMS, che abilita la relocation delle applicazioni all'interno di un MEC System. Nel contesto di questa tesi, tale meccanismo è stato esteso anche a scenari federati, in cui l'utente si sposta tra domini

appartenenti a operatori differenti. Di seguito si descrivono le principali fasi che caratterizzano la migrazione:

- handover e trigger di migrazione
- policy e logica di instradamento della migrazione: intra e federato
- migrazione e trasferiemento dello stato dell'applicazione.

4.3.1 Handover e trigger di migrazione

Dal punto di vista del Multi-access Edge Computing, l'handover radio nel 5G non riguarda solo il piano di controllo e utente della rete, ma ha un impatto diretto anche sulle applicazioni che devono rimanere vicine al terminale per rispettare i requisiti di bassa latenza. Per questo motivo, l'evento di handover diventa un trigger naturale per avviare una migrazione applicativa, sia intra-dominio sia in scenari federati, il digramma di sequenza è mostrato in Figura 18.

Tale evento è gestito dal modulo HandoverManager, solitamente contenuto nella gNodeB, che rileva il cambio di cella dell'UE e genera notifiche interne contenenti le informazioni sul nuovo collegamento. Queste informazioni vengono poi passate al servizio Radio Network Information, che ha il compito di raccogliere e diffondere i dati legati alla rete radio, come ad esempio gli spostamenti dei dispositivi tra una cella e l'altra. L'RNI funziona con un meccanismo di tipo publish/subscribe: in pratica, i moduli interessati si iscrivono al servizio e ricevono automaticamente una notifica ogni volta che avviene un cambiamento. In questo modo non è necessario che ciascun componente interroghi continuamente la rete, ma è l'RNI a distribuire in tempo reale gli aggiornamenti a chi ne ha bisogno, rendendo più semplice e veloce la gestione della mobilità.

In questo scenario, il VIM si sottoscrive alle notifiche di mobilità generate dall'RNI per tutti gli UE che ospitano applicazioni nel MEC Host. Quando l'RNI segnala lo spostamento di un determinato terminale verso una nuova cella, il VIM riceve la notifica e procede a verificare quali applicazioni, tra quelle attive sul MEC Host, sono associate a quell'utente. Abbiamo quindi che ora il sistema MEC è notificato dello spostamento. Il VIM invierà il messaggio di migrazione al MEC Platform Manager.

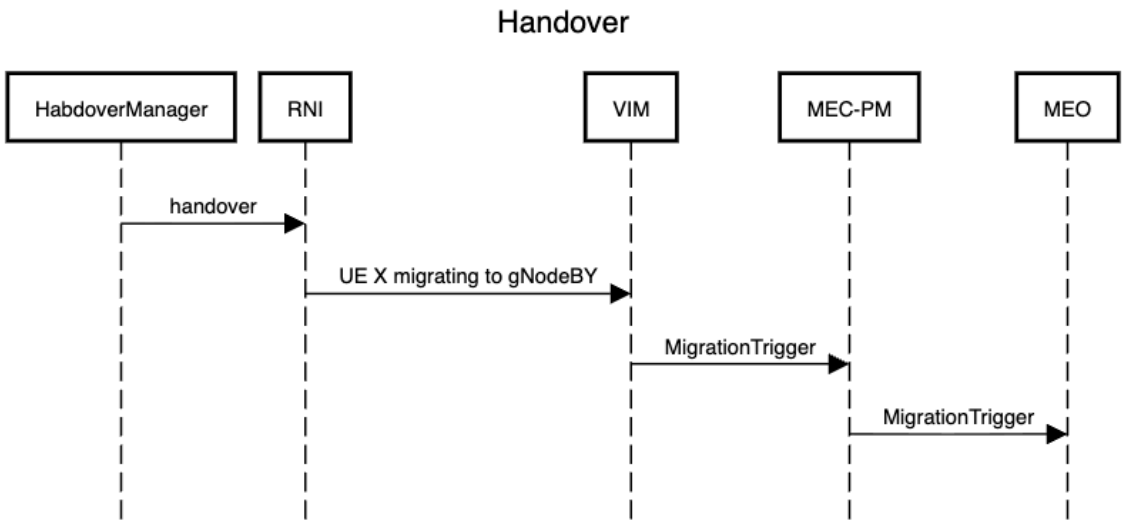

Figura 18 - Diagramma di sequenza Handover

Il MEC-PM inoltrerà la notifica al MEO, l'entità che poi dovrà capire se può gestire la migrazione oppure deve passare via federatore.

4.3.2 Policy e logica di instradamento della migrazione: intra e federato

Il sistema sa che un determinato UE si sta spostando verso la gNodeB X, questa informazione è arrivata ora al MEC Orchestrator, ora sarà lui a capire e decidere cosa fare e dove migrare l'applicazione. Le specifiche ETSI non descrivono in dettaglio la logica di routing o il meccanismo di selezione del nuovo MEC Host: tali aspetti sono lasciati all'implementazione dei singoli operatori. All'arrivo del messaggio di migrazione il MEO controllerà se la gNodeB di destinazione dell'UE è contenuta nel file, se sì allora la migrazione è locale altrimenti la migrazione è federata.

Per la migrazione locale si passa immediatamente alla gestione di avviamento della nuova applicazione, sempre internamente al MEO, Figura 19.

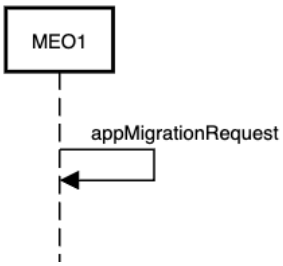

Figura 19 - Diagramma di sequenza migrazione interna

D'altro canto, invece la migrazione federata porta a interagire con il MEC Federator, come illustrato in Figura 20. Si inoltra il pacchetto di MigrationTrigger che da ora

chiameremo appMigrationRequest al nostro federatore di riferimento al quale ci siamo registrati. Il pacchetto conterrà le informazioni su chi e che applicazione deve muoversi.

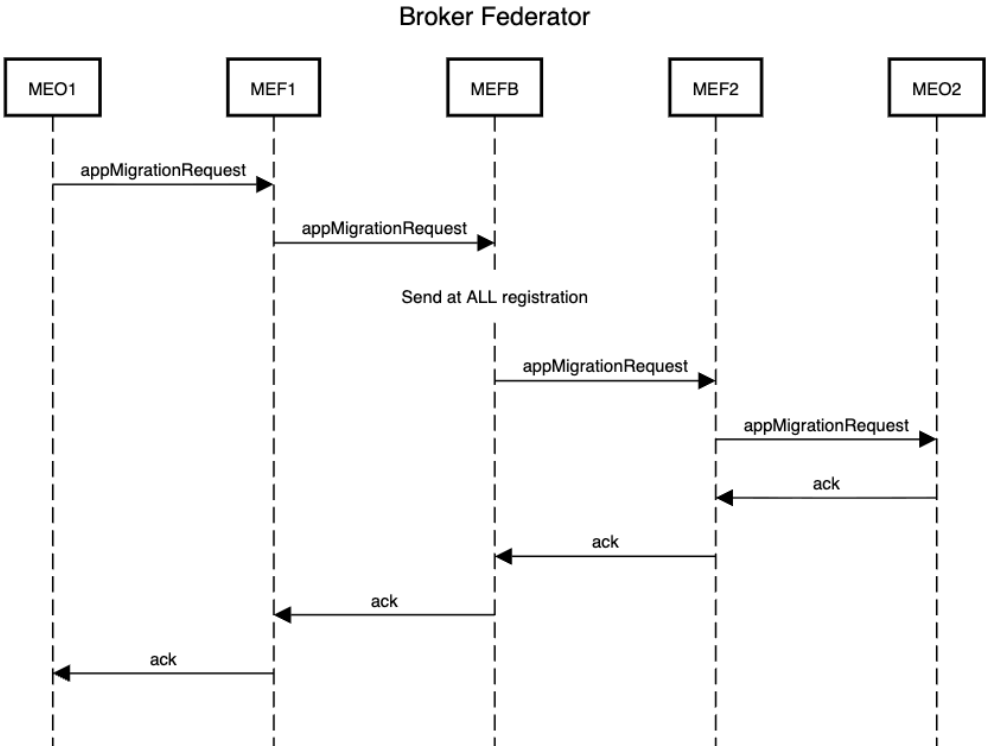

Figura 20 - Diagramma di sequenza MEF e MEFB nella migrazione

Il MEF aggiungerà nei campi del messaggio l'informazione di chi sta facendo la richiesta e in particolare la visione globale del sistema è determinata dal federator, quindi inserirà il suo indirizzo. Il messaggio viene inoltrato al MEFB, il broker invierà la richiesta di migrazione dell'applicazione a tutti i federator registrati, tranne al mittente. Gli altri MEF riceveranno la richiesta e la inoltreranno direttamente al MEO registrato. Il MEO a questo punto controllerà se può istanziare l'applicazione, controllerà che ha la gNodeB di destinazione. In caso positivo, il flusso prosegue come nel caso della migrazione locale: il MEO avvia la procedura di istanziazione presso il MEC Host corrispondente.

L'istanzazione della nuova applicazione parte appunto dal MEO che innanzitutto deve trovare un MEC Host che può ospitare l'applicazione, qui la scelta prosegue come sopra, si sceglie il MEC Host più vicino e quindi quella di riferimento alla gNodeB. Quindi, si inoltra al MEC Host giusto la richiesta di istanziazione dell'applicazione. Se il MEC Host invierà la possibilità di poter istanziare perché ha le risorse necessarie o perché soddisfa tutti i requisiti dell'applicazione allora si invia un ACK, che nel caso locale rimane al

MEO stesso ma nella la migrazione federata la notifica del successo deve essere inviata al MEO richiedente. Ciò è reso possibile proprio dall'informazione aggiuntiva inserita nel messaggio iniziale, che identifica il MEF originatore e ne permette il corretto instradamento della risposta.

4.3.3 Migrazione e trasferimento dello stato dell'applicazione

Come introdotto nel capitolo di Background, la gestione del servizio AMS diventa più complessa nel caso di una migrazione che avviene con la federazione, poiché ogni dominio MEC mantiene un proprio AMS. Per evitare modifiche invasive alle specifiche ETSI, la logica di gestione della mobilità è stata in questa tesi spostata a livello applicativo.

La soluzione adottata, illustrata in Figura 21, prevede che, al momento della migrazione, l'applicazione istanziata nel nuovo MEC host riceva nei messaggi di migrazione i parametri necessari alla configurazione iniziale. Tra questi figurano indirizzo IP e porta dell'AMS associato all'istanza precedente. Ciò consente all'applicazione di collegarsi direttamente al "vecchio" AMS, evitando la richiesta al service registry del MEC host locale che, in condizioni normali, sarebbe responsabile della discovery.

In questa fase di transizione esistono temporaneamente due istanze della stessa applicazione: la vecchia applicazione, ancora attiva sul MEC host sorgente e la nuova applicazione, in fase di avvio sul MEC host di destinazione.

La nuova applicazione utilizza le informazioni ricevute per connettersi all'AMS, registrarsi e sottoscriversi. Questa operazione innesca il trigger INTERHOST_MOVEOUT_TRIGGERED, che viene notificato a tutti gli iscritti. Entrambe le istanze ricevono la notifica, la quale contiene anche le informazioni di host e porta necessarie per instaurare la connessione diretta tra le due applicazioni.

Una volta stabilita la connessione, ha luogo il trasferimento dello stato: la vecchia applicazione invia i dati necessari alla nuova, in modo da ripristinare il contesto operativo. Terminato questo scambio, l'applicazione migrata procede alla fase di pulizia, deregistrandosi dal vecchio AMS e completando l'eliminazione dell'istanza sorgente.

Questo flusso, che sarà illustrato nei diagrammi di sequenza riportati nei paragrafi successivi, consente di garantire la continuità del servizio applicativo senza modificare il

comportamento standard dell'AMS, mantenendo quindi piena compatibilità con le specifiche ETSI.

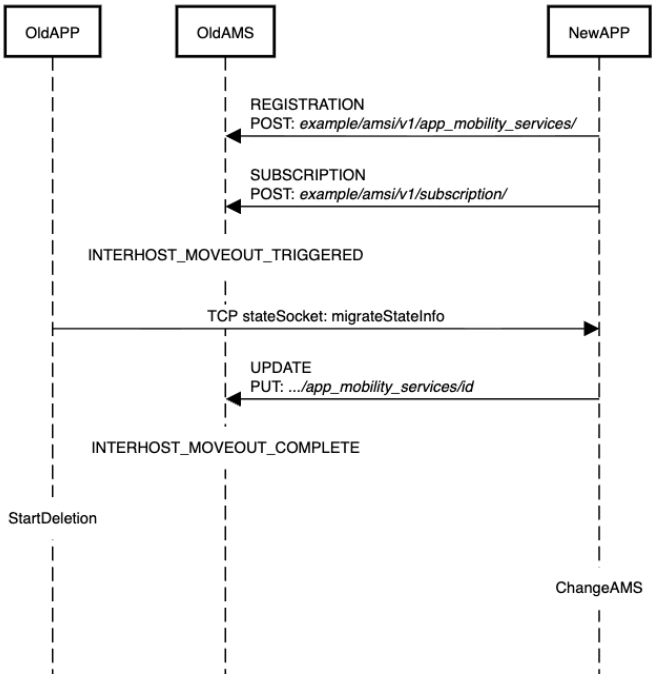

Figura 21 - Diagramma di sequenza Migrazione applicazione con AMS nel caso federato

La nuova applicazione riceve lo stato e con un PUT fa l'update dello stato dell'applicazione in USER_CONTEXT_TRASFERED, questo fa scattare la notifica di INTERHOST_MOVEOUT_COMPLETE. Il trasferimento è completo, questa notifica fa scattare due procedure, nella vecchia applicazione l'eliminazione e nella nuova il passaggio al nuovo AMS, quello del sistema di cui fa parte.

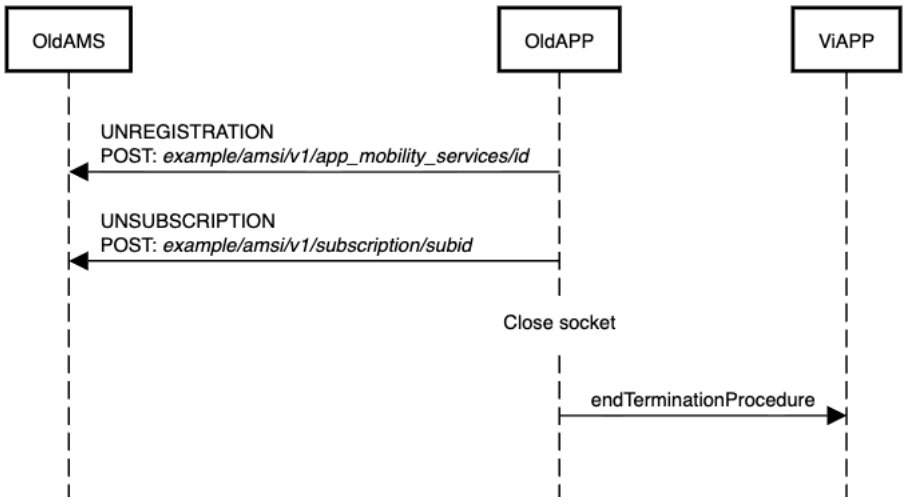

Figura 22 - Diagramma di sequenza Migrazione app con AMS nel caso federato

Nella Figura 22 vediamo che la vecchia applicazione riceve il trigger che segnala il completamento delle operazioni e dopo essersi cancellata e annullamento della sottoscrizione presso l'AMS. Invia il messaggio endTerminationProcedure per far si che la sua istanza viene eliminata.

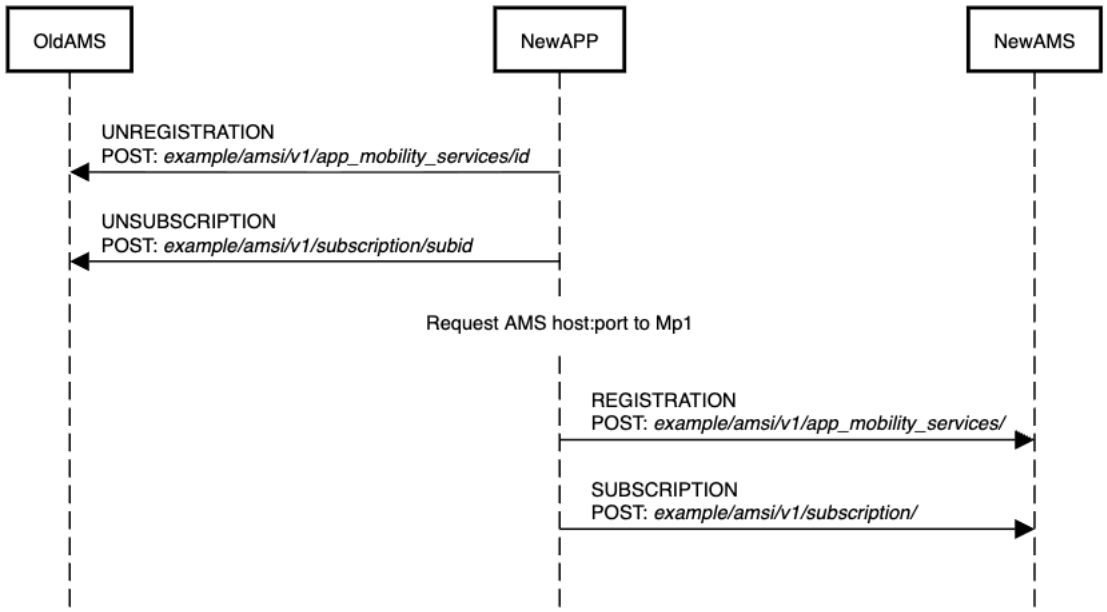

Figura 23 - Diagramma di sequenza Migrazione applicazioni con AMS nel caso federato

Invece in Figura 23, la nuova applicazione anche lei cancellata e annulla della sottoscrizione presso l'AMS di riferimento alla vecchia applicazione, attraverso il MEC Platform richiede l'AMS di riferimento del MEC Host dov'è allocata e si registra e sottoscrive.

4.4 Richiesta di un'applicazione nel contesto federato

La richiesta di un'applicazione rappresenta un meccanismo fondamentale nei sistemi MEC, in quanto abilita un'applicazione a interagire con i servizi offerti da un'altra applicazione già istanziata all'interno della piattaforma. In termini pratici, ciò si traduce nella possibilità, per un'applicazione, di ottenere le informazioni necessarie, tipicamente indirizzo IP e porta, per instaurare una comunicazione diretta tramite API. Questo processo non si limita a una semplice lookup di endpoint: implica invece un insieme di operazioni standardizzate di discovery e scambio di informazioni.

Nel caso in cui l'applicazione richiesta sia già disponibile all'interno del MEC System locale, l'interazione risulta relativamente semplice e confinata all'interno del sistema: il

sistema, individua l'applicazione e restituisce i dati di accesso al richiedente. Diversa è la situazione in cui l'applicazione non sia presente nel dominio locale: in questo scenario entra in gioco la federazione, cioè il meccanismo che permette a sistemi MEC appartenenti a domini diversi di cooperare per soddisfare la richiesta.

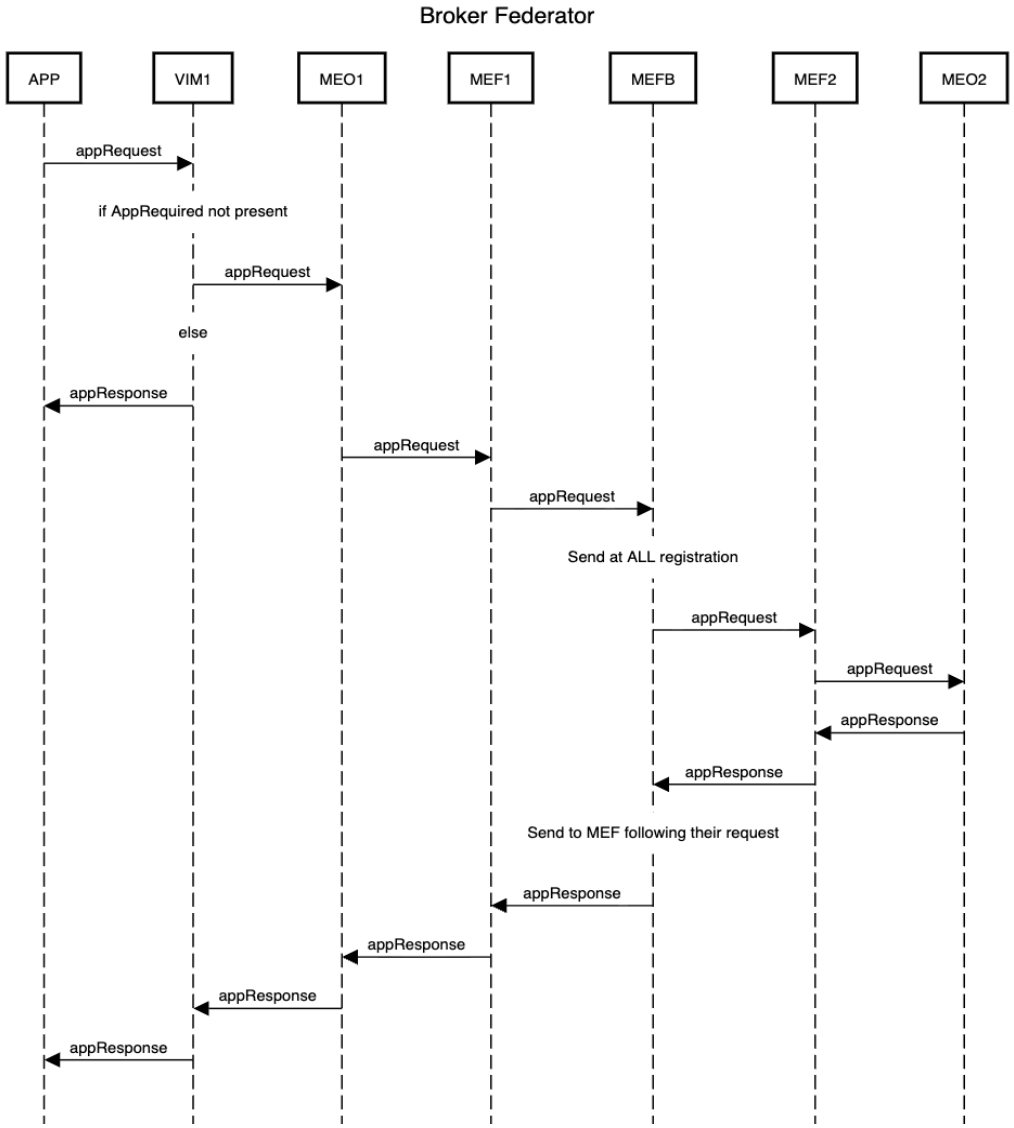

Figura 24 - Diagramma di sequenza Richiesta di un'applicazione nel caso federato

Nella Figura 24 è possibile notare come la procedura di richiesta di un'applicazione segua, a grandi linee, uno schema simile a quello previsto per la migrazione. Le specifiche ETSI descrivono infatti flussi con un'analoga sequenza di attori e passaggi; la differenza principale risiede nel tipo di gestione e nell'obiettivo finale del processo.

In primo luogo, l'applicazione richiedente invia un messaggio al sistema per ottenere informazioni su un'altra applicazione o servizio. Tale richiesta può contenere il nome o l'identificatore dell'applicazione desiderata. Il messaggio viene inoltrato al VIM, che

effettua una verifica a livello locale, ossia all'interno del MEC Host. Se l'applicazione è presente, vengono restituite immediatamente le informazioni di accesso. In caso contrario, la richiesta viene propagata al MEC Orchestrator, che ha visibilità su tutti i MEC Host del sistema e può quindi estendere la ricerca.

Se neppure a livello di MEO l'applicazione è disponibile, entra in gioco la federazione. In questo scenario, la richiesta viene inoltrata al MEC Federator. Quest'ultimo potrebbe già conoscere la disponibilità di determinati servizi o applicazioni presso altri sistemi, grazie alle informazioni raccolte in precedenza nelle fasi di discovery. Tuttavia, per garantire l'aggiornamento e ridurre il rischio che informazioni obsolete generino inconsistenze, come applicazioni istanziate in passato ma non più attive e non aggiornato il discovery, la richiesta viene comunque inoltrata al Broker di Federazione, che la distribuisce a tutti i MEF coinvolti.

Ogni MEF riceve la richiesta e la inoltra al suo MEO che controllerà i paramentri di richiesta. Nel caso in cui l'applicazione sia effettivamente disponibile come per la migrazione la risposta è diretta al richiedente perché il campo di richiesta viene arricchito delle info sul MEC System e in particolare il Federator che è l'entità conosciuta nella federazione. L'applicazione ha le info sull'applicazione richiesta e può avviare una comunicazione diretta e utilizzare i suoi servizi.

5 Implementazione

Dopo aver delineato nel capitolo precedente il disegno architetturale e i principi di funzionamento della federazione in ambienti MEC, questo capitolo descrive nel dettaglio le scelte implementative effettuate e le estensioni necessarie per supportare la migrazione delle applicazioni in scenari multi-operatore. L'obiettivo è tradurre le specifiche ETSI, e in particolare i meccanismi previsti dal MEC Federator e dal servizio AMS, in componenti concreti integrati nell'ambiente di simulazione Simu5G/OMNeT++.

La sezione si concentra inizialmente sugli strumenti e sulle tecnologie utilizzate. Successivamente vengono presentati i moduli sviluppati e le modifiche introdotte a quelli esistenti, evidenziando come tali componenti interagiscono per garantire il corretto funzionamento della migrazione. Particolare attenzione è posta sull'estensione del servizio AMS, necessaria per gestire sia il caso di istanziazione originaria sia quello di istanza migrata.

In questo modo si fornisce una visione completa del passaggio dal livello teorico al livello pratico, permettendo di comprendere come l'architettura proposta sia stata effettivamente modellata.

5.1 Framework adottati

Negli anni OMNeT++ è stata arricchita da framework e modelli di simulazione, la sua libreria principale è **INET Framework**, che implementa i modelli di protocollo standard di OMNet++. Con protocolli e modelli di rete, permettendo quindi di modellare sia la parte di accesso radio sia la parte di trasporto IP necessaria per supportare le applicazioni MEC. Esistono altri framwork come Venis che permette l'implementazione di reti veicolari.

Per quanto riguarda il dominio specifico delle reti mobili 5G, il framework utilizzato è **Simu5G**, sviluppato come estensione di OMNeT++ e derivato dal progetto SimuLTE. Simu5G fornisce un insieme di moduli per la modellazione delle componenti principali del 5G System, in linea con le specifiche 3GPP: NG-RAN: modellazione del gNodeB, gestione delle risorse radio e delle procedure di handover. UE: terminali mobili connessi tramite interfaccia NR-Ue, configurabili con diverse caratteristiche di traffico e mobilità. Simu5G integra inoltre un modello Multi-access Edge Computing conforme agli standard ETSI. Questo modello include:

- MEC Host, con piattaforma MEC e Virtualisation Infrastructure Manager.
- MEC Orchestrator, responsabile della gestione a livello di sistema e della decisione di migrazione intra-MEC.
- MEC Platform Manager, che gestisce localmente le applicazioni e le interazioni con il VIM.
- Servizi MEC standard come Radio Network Information Service e Application Mobility Service.

Nel presente lavoro non è stata utilizzata la versione standard di Simu5G, ma una **versione estesa** sviluppata dai correlatori e presentata in Feraudo et al. (2023) [29], che introduce un modello avanzato di gestione del VIM e meccanismi di migrazione intra-host. Questa estensione è disponibile pubblicamente su GitHub al link Simu5G[1]. A partire da questa base, il simulatore è stato ulteriormente **customizzato nell'ambito di questa tesi**, introducendo nuovi moduli per supportare scenari federati multi-operatore e in particolare la gestione della **migrazione inter-sistema** tramite un **Federator** disponibile sempre su GitHub al link Simu5G Federation[2].

5.2 Estensione moduli MEC

Per supportare la migrazione delle applicazioni in scenari sia intra-sistema sia federati, è stato necessario estendere e in alcuni casi modificare diversi moduli MEC presenti in Simu5G, mantenendo l'aderenza alle specifiche ETSI. Le modifiche principali hanno riguardato i componenti coinvolti nella gestione del ciclo di vita delle applicazioni e nella federazione tra domini. In particolare: LTEHandoverManager, RNIService, VIM, MEC-PM e MEO.

5.2.1 Handover Management: LTEHandoverManager & RNI Service

Il funzionamento di questa estensione è rappresentato nel System Design (Sezione 4.3.1, Figura 18). Il modulo **HandoverManager** è responsabile del rilevamento degli eventi di handover, sia in fase di avvio sia al termine della procedura. Nella nostra estensione, al momento in cui viene catturato l'inizio di un handover, il modulo invia una segnalazione al **RNI**, indicando che un determinato UE si sta spostando verso una nuova cella, gNodeB

[1] Disponibile al repository GitHub: https://github.com/aferaudo/Simu5G
[2] Disponibile al repository GitHub: https://github.com/aferaudo/Simu5G/tree/federation

di destinazione. Il servizio RNI, previsto dalle specifiche ETSI come punto di raccolta e distribuzione delle informazioni radio, funge da broker di notifica: tutti i componenti che necessitano di monitorare la mobilità degli utenti possono sottoscriversi per ricevere aggiornamenti. In questo contesto, i VIM risultano tra i principali sottoscrittori, poiché devono essere informati sugli spostamenti degli UE che eseguono applicazioni in quel MEC Host.

Quando riceve la comunicazione dall'LTEHandoverManager, l'RNI provvede quindi a notificare i VIM interessati, includendo l'identificativo dell'UE e la cella di destinazione. In questo modo, ogni VIM dispone delle informazioni necessarie per valutare se è richiesta una procedura di migrazione applicativa e può attivare le logiche successive di interazione con il MEC Platform Manager e l'AMS.

5.2.2 VIM

Il **VIM**, oltre alle funzionalità già previste dalle specifiche standard, è stato esteso per interagire direttamente con il RNI. In particolare, al momento della ricezione di una richiesta di InstantiateApplicationRequest all'interno della funzione instantiateMEApp, il VIM invia al servizio RNI un messaggio di sottoscrizione. Tale sottoscrizione è specifica per l'UE che ha richiesto l'istanza applicativa: infatti, il messaggio contiene anche le informazioni relative all'utente, incluse l'identità e l'indirizzo IP, che vengono utilizzate come riferimento per la registrazione alla notifica.

```
nlohmann::json jsonBody = nlohmann::json::parse(request->getBody());

//NOTIFICA DA RNI
std::string addressUEstr = jsonBody["associateId"]["value"];
int cellId = jsonBody["cellId"];

inet::L3Address addressUE = inet::L3Address(addressUEstr.c_str());
bool found = false;

//Cerco se quell UE ha un app istanziata
for (auto meAppEntry = handledApp.begin(); meAppEntry != handledApp.end(); ++meAppEntry) {
    const inet::L3Address &currentEndpoint = meAppEntry->second->ueEndpoint;
    if (currentEndpoint == addressUE) {

        //UE ha un app istanziata

        //Invio messaggio di migrazione
        mobilityTriggerFederation(meAppEntry->second->appInstanceId, meAppEntry->second->ueAppID, cellId, addressUEstr.c_str());

        //Elimino la sottoscrizione per questo UE
        int id = ueToSubscriptionId[addressUEstr];
        deleteRNISubscription(id);

        found = true;
        break;
    }
}
```

Figura 25 - Codice, gestione notifica RNI nel VIM

È stata aggiunta inoltre al VIM una funzione dedicata alla gestione dei messaggi provenienti dall'RNI. Ogni volta che un UE, per il quale è stata effettuata la sottoscrizione, effettua un handover, il RNI inoltra la notifica corrispondente. Alla ricezione di questa notifica, parte del codice è illustrato in Figura 25, il VIM genera e trasmette un messaggio di MigrationTrigger al MEC Platform Manager, includendo tutte le informazioni necessarie al processo di migrazione: l'identificativo dell'UE, i dettagli dell'applicazione associata e la cella di destinazione.
Per evitare di mantenere sottoscrizioni non più necessarie, il VIM procede immediatamente anche alla cancellazione della sottoscrizione relativa all'UE migrato. La stessa operazione di eliminazione viene eseguita in caso di ricezione di un messaggio di terminazione dell'applicazione, così da mantenere sempre coerente lo stato delle sottoscrizioni attive.

5.2.3 MEC Platform Manager

L'implementazione del **MEC Platform Manager** è stata estesa per includere la gestione del messaggio MigrationTrigger, inviato dal VIM. Non appena questo messaggio viene ricevuto, il MEC-PM provvede a inoltrarlo direttamente al MEC Orchestrator, garantendo così la continuità del flusso di segnalazione nel processo di migrazione applicativa.

5.2.4 MEC Orchestrator

Nel **MEC Orchestrator** per abilitare le funzionalità di interfacciarsi con il Federator, si è resa necessaria una modifica al suo codice, introducendo il supporto nativo alla federazione. In fase di configurazione (file .ini), il MEO riceve l'indirizzo e la porta del Federator e, sulla base di tali parametri, gestisce lo scambio di messaggi secondo le specifiche ETSI.
Il MEO è quindi l'unica entità del sistema MEC autorizzata a comunicare con il Federator: supporta sia messaggi in ricezione sia in inoltro, e svolge un ruolo cruciale nella gestione dei flussi di migrazione inter-sistema. In particolare, può ricevere dal Federator messaggi come:

- AppMigrationRequest, inviato quando un altro MEO non è in grado di soddisfare una richiesta di migrazione e, tramite federazione, ne richiede l'esecuzione;

- AppRequest, utilizzato da un MEO remoto per ottenere informazioni su applicazioni necessarie a garantire la continuità dei servizi.

In entrambi i casi, il MEO locale risponde direttamente o inoltra la richiesta al Federator, che provvede a ritrasmetterla verso il sistema corretto. Sarà sempre un messaggio di tipo AppMigrationRequest.

Un aspetto fondamentale di questa estensione riguarda la gestione delle informazioni di rete: il MEO mantiene, sempre tramite configurazione, una mappa che associa ciascun MEC Host del sistema alla gNodeB a cui è collegato. Tale mappa diventa cruciale quando arriva un MigrationTrigger da un'applicazione di questo sistema, in particolare seguendo il flusso di migrazione l'ultima entità è il MECPM. In quel momento, il MEO deve stabilire la destinazione della migrazione: se l'host di destinazione appartiene al proprio dominio, la richiesta viene gestita localmente; altrimenti, viene inoltrata al Federator, che ne assicura la propagazione verso l'MEO corretto. La gestione del messaggio di MigrationTrigger è illustrata nel codice in Figura 26.

```
void MecOrchestratorApp::handleAppMigrationRequest(inet::Packet *contAppMsg){
    auto data = contAppMsg->peekData<MigrationTrigger>();

    inet::Packet *pkt = new inet::Packet("appMigrationRequest");
    //Preparo pacchetto con le info dell'applicazione da migrare
    //...

    //carico da file gbNodeToAddressVimPort, tupla contenente per ogni gbNode - MECPlatform address:port

    gbNode = data->getGbNode();
    if (gbNodeToAddressVimPort.count(gbNode)) {
        //Se la gbNode appartiene al MEO

        const auto& entry = gbNodeToAddressVimPort[gbNode];
        std::string destAddr = std::get<0>(entry);
        int destPort = std::get<1>(entry);

        socket.sendTo(pkt, inet::L3AddressResolver().resolve(destAddr.c_str()), destPort);
    } else {
        //Se non appartiene allora invio al federatore
        socket.sendTo(pkt, federatorAddress, federatorPort);
    }
}
```

Figura 26 - Codice, gestione messaggio MigrationTrigger nel MEO

Finita questa fase o abbiamo mandato il messaggio via federator e arriverà sempre a un MEO o in locale, quindi possiamo procedere alla migrazione, verifichiamo che l'applicazione si può istanziare e faremo **findMecHostByTargetId,** evoluzione della precedente findHost, invece di inviare a tutti gli host del sistema la richiesta e il primo che risponderà sarà chi allocherà l'applicazione. Ora invece ricerchiamo l'host per id, nel messaggio di migrazione oltre ai dati dell'applicazione avremo i dati della gNodeB di destinazione, questo ci permette di controllare di nuovo la mappa, vedere quale host è il migliore e inviare la richiesta di istanzazione. In questo modo si riduce la complessità, si

ottimizza il tempo di instanziazione e si allinea l'operazione con le procedure previste dagli standard di federazione.

5.3 Implementazione nuovi moduli

Per poter completare l'architettura federata è stato necessario introdurre nuovi moduli che nello scenario intra-operator non erano presenti. In particolare, è stato creato il MEC Federator e un'applicazione MEC che possa gestire la migrazione federata.

5.3.1 MEC Federator

Come introdotto nella Capitolo 4.1, il **MEC Federator** rappresenta l'elemento centrale della federazione. In base allo standard, esso può operare con funzionalità di manager, broker oppure come client che instaura connessioni dirette con altri federator. Nell'implementazione è stato seguito l'approccio con Broker, mentre non è stato sviluppato il modello Peer-to-Peer.

5.3.1.1 Federator Broker e client

Nell'implementazione realizzata, il Federator può assumere due configurazioni principali: **Federator Client**, che svolge unicamente le funzioni di Federation Manager (MEFM) e **Federator Broker**, che integra sia il Federation Manager sia il Federation Broker App, in grado di smistare le richieste provenienti da più sistemi MEC.

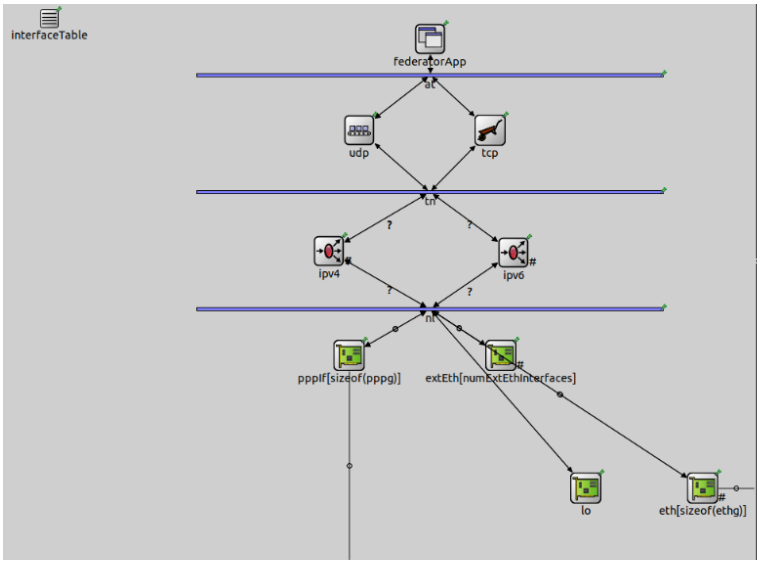

Figura 27 - FederatorClient, modulo OMNeT++

Il Federator Client è quindi composto esclusivamente dal modulo FederatorApp come illustrato il modulo nella Figura 27, che implementa la logica del MEFM. In questa modalità, il federator è responsabile soltanto della gestione delle interazioni con il MEC Orchestrator locale e dello scambio di messaggi con un federator remoto o con un broker.

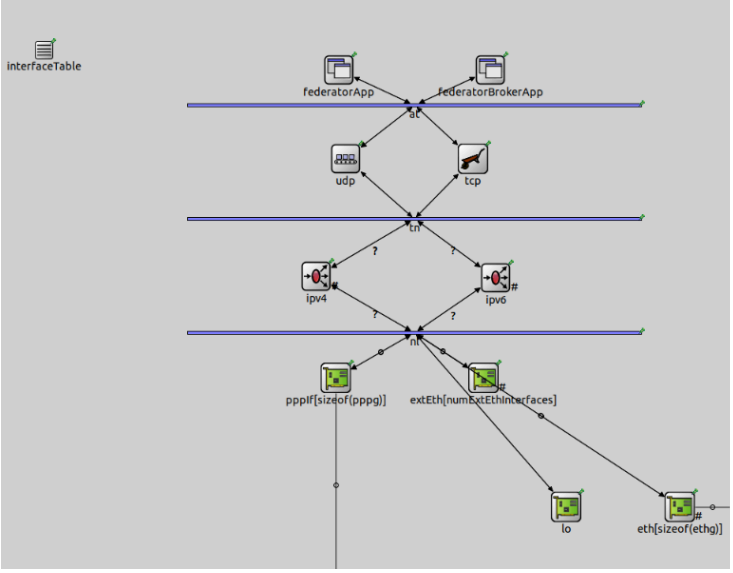

Figura 28 - FederatorBroker, modulo OMNeT++

Il Federator Broker, in Figura 28 invece, oltre al FederatorApp include anche il modulo FederatorBrokerApp, che implementa le funzioni di instradamento e aggregazione delle richieste di federazione. In questa configurazione, il federator è in grado di gestire più collegamenti contemporaneamente e svolge il ruolo di nodo centrale di una topologia hub-and-spoke, come previsto dalle specifiche ETSI.

La differenza la si può notare nell'illustrazione nel System Design (Sezione 4.1, Figura 15).

Dal punto di vista modellistico, entrambi i moduli sono definiti come nodi con stack TCP/IP completo. L'interfaccia di rete è esposta attraverso collegamenti Ethernet (Eth) o PPP, così da poter essere integrata in scenari sia cablati che punto-punto. Lo stack è configurato per consentire alle applicazioni (FederatorApp e FederatorBrokerApp) di aprire socket TCP/IP e gestire sessioni di comunicazione affidabili verso i federator remoti.

In questo schema, le applicazioni sono connesse direttamente allo stack tramite socket, mentre il nodo mette a disposizione porte ethg e ppp come interfacce di collegamento con la rete simulata. Questo approccio permette di mantenere l'aderenza al modello OMNeT++/INET, rendendo i moduli del Federator interoperabili con gli altri elementi della simulazione.

Il Federator Broker è stato progettato con una socket in ascolto per accettare nuove connessioni da parte dei federator client. Al momento della registrazione di un nuovo client, le relative informazioni vengono salvate in una tabella, che può essere aggiornata o svuotata in caso di modifiche o cancellazioni. L'interazione con questa tabella è gestita direttamente dai messaggi inviati dai client stessi.
Oltre alla gestione delle iscrizioni, il broker è in grado di ricevere messaggi dai federator relativi a richieste di migrazione di applicazioni o richieste di scoperta di un'applicazione. In questi casi, il broker agisce come punto di smistamento centralizzato: il messaggio ricevuto viene inoltrato in modalità broadcast a tutti i client registrati, così da garantire che la richiesta raggiunga ogni sistema federato potenzialmente in grado di rispondere.
Diverso è il caso dei messaggi di acknowledgment o di risposta. Qui il broker non effettua un inoltro massivo, ma svolge il ruolo di intermediario intelligente: utilizzando le informazioni sul destinatario contenute nel messaggio, provvede a recapitarlo in maniera selettiva al federator corretto. In questo modo, il broker si limita a garantire l'instradamento dei messaggi senza alterarne il contenuto, fungendo da nodo trasparente nella comunicazione punto-punto.

5.3.1.2 Federator Manager

L'applicazione FederatorApp implementa il ruolo di MEC **Federation Manager**, ossia il componente logico che rappresenta il federator vero e proprio. Al suo avvio, il manager legge da file di configurazione l'indirizzo e la porta del Federator Broker e provvede a stabilire una connessione iniziale inviando le informazioni di identificazione del proprio sistema. In questo modo, il broker è in grado di registrare e riconoscere il nuovo federator all'interno della federazione.
Dal punto di vista funzionale, il manager riceve messaggi dal MEC Orchestrator, che possono consistere in: richieste di scoperta di un'applicazione, richieste di migrazione di un'applicazione. In entrambi i casi, il FederatorApp inoltra i messaggi al broker, che a sua volta provvede a diffonderli agli altri federator della rete.

```
void FederatorApp::handleAppMigrationRequestMEOtoMEF(inet::Packet *contAppMsg){
    auto data = contAppMsg->peekData<AppMigrationRequest>();

    //Copio la richiesta in un nuovo pacchetto
    inet::Packet* pktdup = new inet::Packet("appMigrationRequestMEFtoMEFB");
    auto request = inet::makeShared<AppMigrationRequest>();

    request->setAppName(data->getAppName());
    request->setAppPackageSource(data->getAppPackageSource());
    request->setAppIsOnboarded(data->getAppIsOnboarded());
    request->setDevAppId(data->getDevAppId());
    request->setAppAddress(data->getAppAddress());
    request->setAppPort(data->getAppPort());
    request->setGbNode(data->getGbNode());
    request->setUeIpAddress(data->getUeIpAddress());

    //Aggiungo l'ip del federatore richiedente
    request->setIpMefRequest(localAddress);

    request->setChunkLength(inet::B(64));
    pktdup->insertAtBack(request);

    //INVIA Al BROKER
    socket.sendTo(pktdup, MEFBrokerAddress, MEFBrokerPort);
}
```

Figura 29 - Codice, handleAppMigrationRequest

Nella Figura 29 è illustrata la funzione in FederatorApp che gestisce l'arrivo del messaggio di migrazione dal MEO. Copiamo la richiesta, inseriamo l'ip per gestire la risposta e inviamo al broker. Lo stesso procedimento avviene con la richiesta di applicazioni.

Infine, il manager gestisce anche la ricezione dei messaggi di ritorno, come acknowledgment e risposte contenenti le informazioni sulle applicazioni richieste, garantendo che vengano correttamente inoltrati al MEO locale.

5.3.2 MEC App

Per valutare il comportamento del sistema federato, è stata implementata un'applicazione generica in grado di supportare la migrazione inter-MEC con il meccanismo esteso dell'Application Mobility Service. L'obiettivo principale è consentire all'applicazione di mantenere continuità operativa anche durante lo spostamento da un MEC Host all'altro, preservando la coerenza dello stato e la compatibilità con le specifiche ETSI.
All'atto dell'istanza, l'applicazione inizializza una socket in ascolto per ricevere messaggi e determina il proprio stato operativo, distinguendo tra:

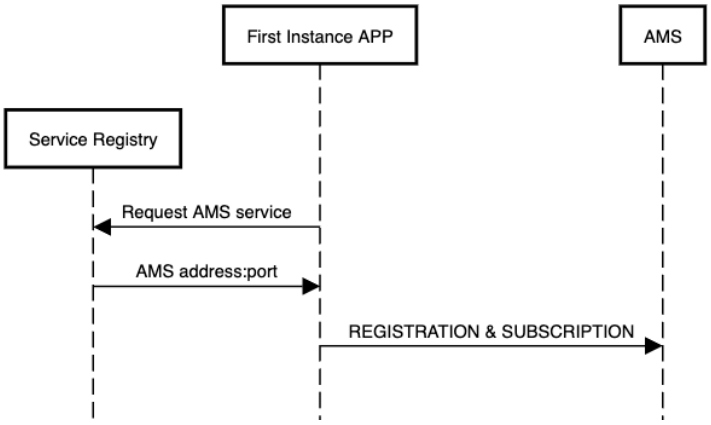

Figura 30 - Prima istanza, connessione AMS

Prima istanza (istanza normale): come illustrato in Figura 30, in questo caso l'applicazione non riceve come parametro iniziale l'indirizzo e la porta del servizio Application Mobility Service. Per ottenere queste informazioni si connette quindi al Service Registry, dal quale richiede l'endpoint dell'AMS. Una volta ricevuta la risposta, procede alla connessione con l'AMS, registrandosi e sottoscrivendosi per le notifiche relative alla propria applicazione, identificate dal suo ID.

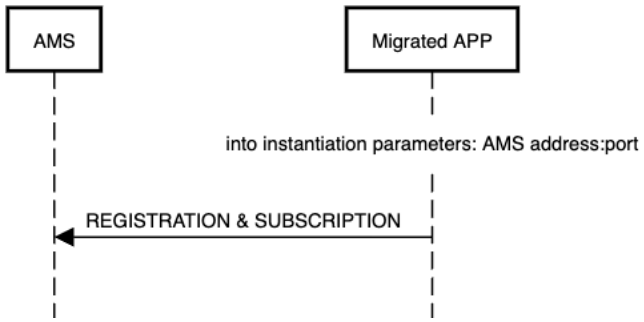

Figura 31 - Istanza Migrata, connessione AMS

Istanza migrata: in Figura 31, se l'applicazione viene avviata in seguito a una migrazione, riceve già come parametro l'indirizzo e la porta dell'AMS di origine. In questo caso, salta la fase di discovery e si connette direttamente all'AMS indicato, completando le stesse operazioni di registrazione e sottoscrizione. Questa modalità è essenziale perché, come previsto dal funzionamento dell'AMS in contesti federati, un'applicazione migrata deve continuare a ricevere le informazioni dalla sua istanza originale finché non è completato il trasferimento dello stato.

L'applicazione gestisce quindi tutti i messaggi provenienti dall'AMS, con particolare attenzione alle notifiche di tipo INTERHOST. Al momento della ricezione di una

INTERHOST_MOVEOUT_TRIGGERED, l'applicazione entra nello stato di migrazione e legge il parametro target host. Sono possibili due casi:

- Target host coincide con l'istanza corrente: la notifica viene ignorata.
- Target host differente: l'indirizzo corrisponde alla nuova istanza della stessa applicazione su un altro host. In questo caso, l'applicazione trasferisce il proprio stato alla nuova applicazione, utilizzando le informazioni ricevute.

Una volta completato il trasferimento, l'applicazione invia un update all'AMS, che scatena la notifica INTERHOST_MOVEOUT_COMPLETE. Anche qui sono previsti due comportamenti:

- Per la nuova applicazione (target host uguale a sé stessa): si procede al cambio di AMS. L'applicazione si disconnette dal vecchio AMS, interroga nuovamente il Service Registry e si registra al nuovo AMS del MEC Host di destinazione.
- Per la vecchia applicazione (target host diverso): viene completata la dismissione, con la conseguente eliminazione dell'istanza.

Oltre a queste funzionalità di mobilità, l'applicazione integra anche una logica di test applicativo. In particolare, segue le procedure previste dalla documentazione per la richiesta di un'applicazione al MEO: se l'applicazione richiesta non è disponibile localmente, la richiesta viene inoltrata tramite il Federator.

5.4 Implementazione nuovi messaggi

Nella discussione di questi nuovi moduli e dell'estensione di quelli già presenti si sono nominati alcuni messaggi che non erano previsti nell'ambiente intra-operator. I Messaggi principali che sono stati implementati riguardano quelli che permettono la migrazione e la richiesta di una nuova applicazione.
Per la migrazione innanzitutto è stato esteso il messaggio **MigrationTrigger**, messaggio che deve contenere le info su quale dispositivo si sta muovendo e dove.

Il primo componente è il VIM che creerà questo tipo di messaggio e lo arricchirà delle info ricevute dalla notifica di migrazione.

- **UeIpAddress**, indirizzo dell'UE che si sta muovendo
- **gNodeB**, base station cellulare dove si sta spostando

- **AppName**, nome dell'applicazione "PingApp"
- **AppId**, id unico del sistema dell'applicazione
- **TargetAddress**, indirizzo della vecchia applicazione
- **TargetPort**, porta della vecchia applicazione

Nello specifico dalla notifica di migrazione avrà solo l'indirizzo dell'UE e della gNodeB, tutte le info sull'app le cercherà il VIM grazie a una ricerca per indirizzo UE, si precisa che se è presenta più di un'applicazione dello stesso dispositivo vengono inviate più notifiche di MigrationTrigger e non viene inviato un array.

Il messaggio ora è passato al MEPM che provvederà a inviarlo al MEO, copia le info passate e ne aggiunge altre due:

- **AddressAMS**, indirizzo dell'AMS di quel sistema
- **PortAMS**, porta dell'AMS di quel sistema

utili nel caso di migrazione a un altro host.

Questo messaggio arriva al MEO e si trasforma se nel caso di migrazione federata in **AppMigrationRequest**, così definito:

- **gNodeB**
- **AppName**
- **AppId**
- **TargetAddress**
- **TargetPort**
- **AddressAMS**
- **PortAMS**
- **AppPackageSource**, path del descrittore applicazione

Molto simile al primo, al passaggio al federator, verrà arricchito dell'indirizzo e porta di esso, in modo da poter poi gestire la risposta.

- **AddressFederator**, indirizzo del federator richiedente
- **PortFederator**, porta del federator richiedente

L'ack di risposta invece mantiene solamente indirizzo del sistema che aveva fatto richiesta e l'applicazione migrata.

Per quanto riguarda i messaggi di discovery da parte dei federator al broker sono messaggi contenente id e nome del sistema, ugualmente per il discovery per il MEO al MEF.

Per la richiesta di un'applicazione o un servizio i messaggi sono composti da:

- **AppId**
- **AppName**
- **AddressFederator**
- **PortApp**
- **HostId**

Dove come prima AddressFederator serve a inoltrare correttamente la risposta, qui viene arricchita anche da HostId e PortApp, perché non solo dobbiamo inoltrare la risposta al sistema giusto ma anche al giusto host di quel sistema (HostId) e anche alla giusta applicazione, quindi la porta dell'applicazione.

Il messaggio di risposta avrà solo due campi in più con:

- **AddressApplication**
- **PortApplication**

Che sono le info dell'applicazione richiesta per poter comunicare direttamente.

6 Risultati

Dopo aver descritto nel capitolo precedente l'implementazione dei moduli necessari a supportare la migrazione in un contesto federato, in questa sezione vengono presentati i risultati ottenuti dalle simulazioni. L'analisi considera diversi scenari: due scenari controllati, intra-MEC e federato, e uno scenario dinamico di Smart City, realizzato con SUMO, in cui i veicoli seguono il flusso del traffico per riprodurre un caso realistico. Vengono valutate le prestazioni del meccanismo di migrazione in funzione del numero di migrazioni contemporanee e della dimensione dello stato da trasferire, analizzata l'efficienza del federatore e confrontati i risultati ottenuti nei vari scenari.

6.1 Ambiente di simulazione

Le simulazioni sono state eseguite su una macchina con sistema operativo **Ubuntu Linux**, scelta in quanto ambiente stabile, open-source e pienamente compatibile con OMNeT++ e Simu5G.

La configurazione hardware e software utilizzata è la seguente:

- Sistema operativo: Ubuntu 22.04.4 LTS (kernel Linux 5.15)
- Processore: CPU virtualizzata QEMU Virtual CPU version 2.5+ 4 core @ 2.2 GHz
- Memoria RAM: 8 GB
- Compilatore: GCC 9.4

Ambiente di simulazione: OMNeT++ 6.0, con INET Framework e libreria Simu5G integrata.

Questa configurazione è sufficiente per supportare scenari con più nodi MEC e utenti mobili, garantendo la possibilità di eseguire esperimenti di migrazione intra-sistema e inter-sistema. L'impiego di una macchina virtuale consente inoltre di isolare l'ambiente di sviluppo e di semplificare la replicabilità dei risultati in contesti differenti.

6.2 Configurazione scenari

Per valutare l'implementazione è stato necessario definire una serie di scenari di simulazione differenziati in base alla topologia della rete, numero dispositivi e numero di federator. Sono state create più reti definite nei file .ned e diverse configurazioni definite

nel file .ini di OMNeT++. Tutte le simulazioni sono state ripetute 5 volte, per avere un dato più preciso possibile.

Le reti sono state divise in quattro scenari principali:

Scenario intra-operator, illustrato in Figura 32 ha un singolo MEC System, composto da due MEC Host, ogni host è collegato a una gNodeB e identificano due celle di rete. Tutto coordinato da un unico MEC Orchestrator. Gli UE si associano ai gNodeB e generano traffico verso applicazioni MEC. Questo scenario è usato per valutare la migrazione intra-sistema, dove il MEO gestisce direttamente la decisione e il VIM coordina il trasferimento delle applicazioni tra host dello stesso dominio.

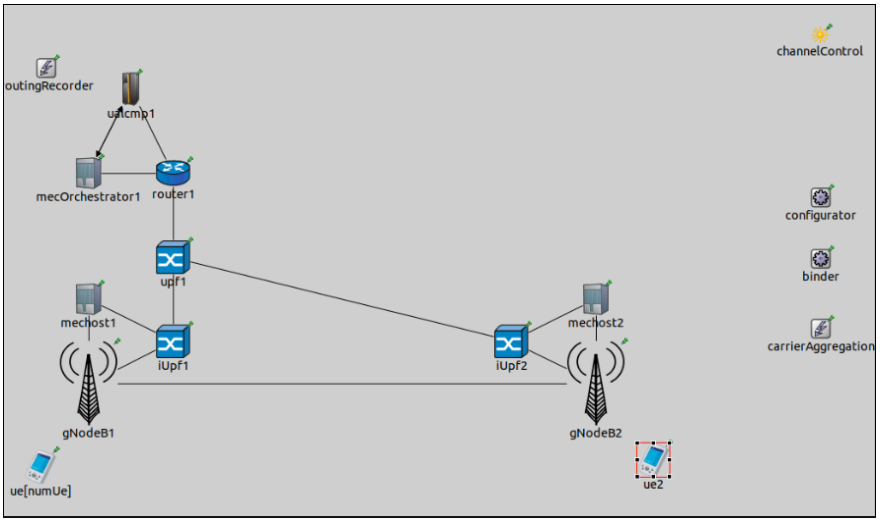

Figura 32 - Scenario Inter-Operator

Scenario federato, illustrato in Figura 33 con due MEC System di domini differenti, ogni sistema ha un MEC Federator, un MEC Orchestrator e un MEC Host. Si identifica il *federator1* anche con le capacità di Federator Broker. Ogni host ha la sua gNodeB con la cella di riferimento. Questa topologia permette di valutare lo scenario multi-operatore, dove il coordinamento avviene attraverso il MEC Federator secondo quanto previsto dalle specifiche ETSI.

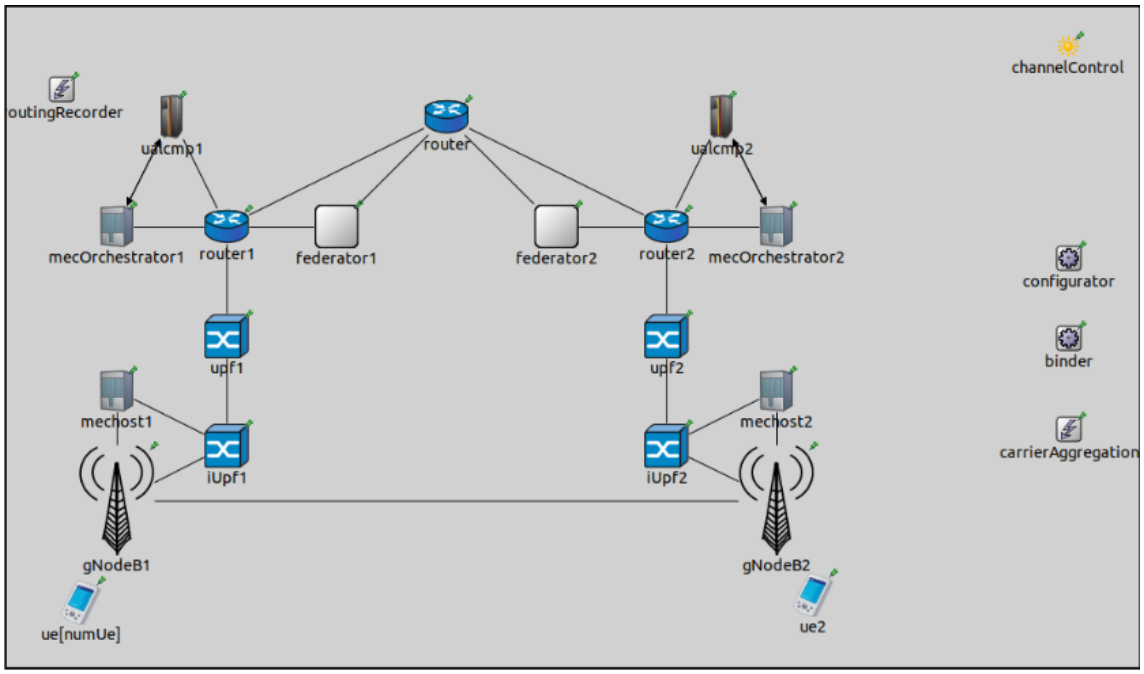

Figura 33 - Scenario Multi-operator Federato

Le reti sopra descritte sono composte da dei componenti principali come:

- UE, rappresentano i dispositivi e si connettono ai gNodeB.
- gNodeB, nodi di accesso 5G che gestiscono la connessione radio.
- UPF, funzione di core che instrada i pacchetti di rete.
- Collegamenti EthG10, Ethernet 10 Giga

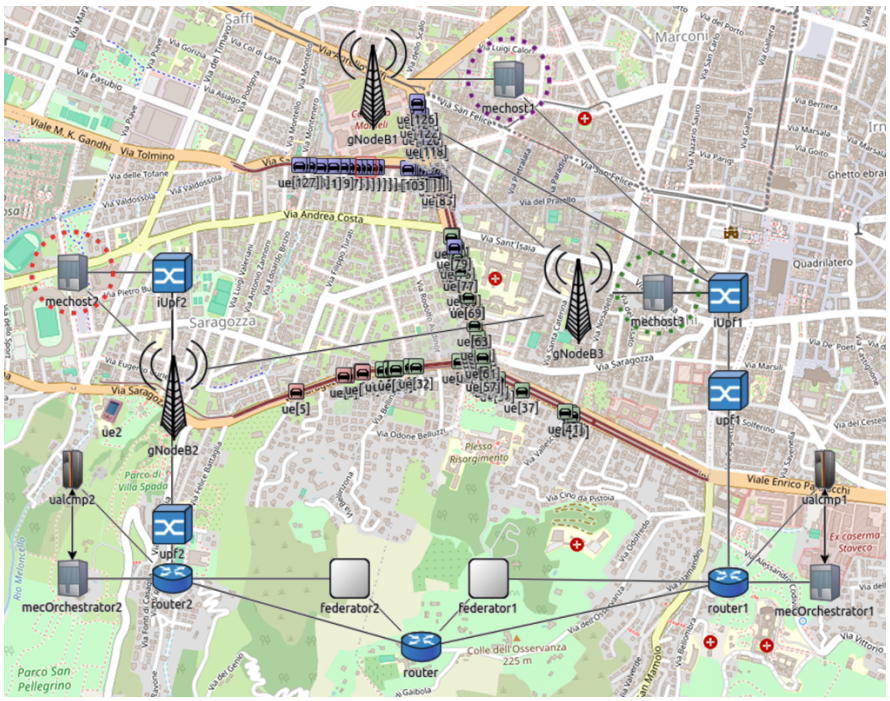

Figura 34 - Scenario Dinamico

Scenario dinamico. Illustrato in Figura 34, a livello architetturale questo scenario prevede tre MEC Host e due MEC System in federazione, ma con una differenza rispetto al caso precedente: uno dei due MEC System è composto da due MEC Host. Ogni MEC Host è associato a una gNodeB differente. L'obiettivo di tale configurazione è quello di validare l'efficacia della soluzione anche in un contesto di mobilità più realistico.
Per riprodurre la mobilità veicolare è stato utilizzato l'ambiente SUMO (Simulation of Urban MObility) [30], integrato con Veins [31], che consente il collegamento diretto tra simulatori di rete e simulatori di traffico. Questa integrazione permette di generare scenari dinamici e verosimili, in cui le entità veicolari non si muovono secondo traiettorie predefinite, ma seguono rotte generate su una mappa reale con logiche di traffico e incroci regolati. In particolare, è stato scelto come caso di studio l'incrocio tra Via Saragozza e i Viali di Bologna, caratterizzato da elevata densità di traffico.
La disposizione delle gNodeB è stata configurata come segue: lungo i Viali sono state collocate due gNodeB appartenenti allo stesso MEC System, mentre una terza gNodeB è stata posizionata su Via Saragozza. La scelta di tali posizioni non è casuale, ma riproduce la collocazione reale di tre antenne presenti in quell'area urbana, individuate attraverso piattaforme online di mappatura delle infrastrutture radiomobili [32]. In questo modo è stato possibile identificare tre aree di copertura distinte. Come mostrato in Figura 32, le automobili vengono colorate in maniera differente in base alla gNodeB a cui risultano connesse, rendendo immediatamente visibile la suddivisione delle zone di aggancio. In particolare, la gNodeB in alto è associata al colore blu, quella in basso a destra (appartenente allo stesso MEC System) al verde, mentre la gNodeB dell'altro sistema è rappresentata in rosso. Le rotte dei veicoli sono state generate in maniera casuale, la simulazione è stata posta con la velocità dei veicoli massima quella consentita dalle strade interessate, in particolare 50 Km/h lungo i viali e 30 Km/h nelle altre strade interessate alla simulazione. Mentre la topologia di rete e i collegamenti logici tra i nodi restano coerenti con quelli degli altri scenari sviluppati.

Discovery dei federator, un ultimo scenario illustrato in Figura 35, è stato proposto per verificare i tempi di discovery quindi, si è proceduto a creare una rete con un broker e un numero variabile di client.

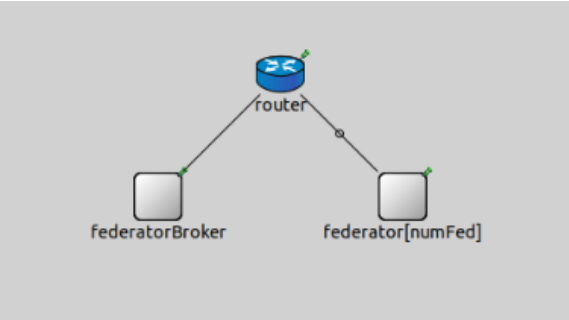

Figura 35 - Scenario test multi-federator

È stata realizzata anche un'applicazione di test, in continuità con l'estensione delle MEC App già implementate. L'applicazione, denominata **PingApp**, è stata progettata con capacità di migrazione e segue le logiche descritte nei documenti di riferimento sul sistema MEC e sulla federazione.

L'applicazione PingApp richiede nello specifico la presenza della controparte PongApp (anch'essa implementata, ma senza supporto alla migrazione). Una volta stabilita la connessione, PingApp invia messaggi di ping, a cui PongApp risponde con i corrispondenti pong, realizzando così un semplice scambio end-to-end utile a verificare il corretto funzionamento del sistema.

6.3 Metriche di valutazione

L'analisi dei risultati di una sperimentazione, specialmente quando riguarda la migrazione di applicazioni in un contesto complesso come quello del Multi-Access Edge Computing federato, richiede la definizione di metriche chiare e condivise. Le metriche non hanno solo la funzione di rendere confrontabili scenari e approcci diversi, ma soprattutto permettono di mettere in luce i punti di forza e le debolezze della soluzione implementata. Nel caso specifico, l'obiettivo principale era valutare l'impatto della federazione sulla continuità e sulla qualità del servizio durante la migrazione di un'applicazione tra nodi appartenenti a domini diversi. Per questo motivo è stato necessario individuare un insieme di parametri in grado di descrivere in maniera esaustiva il comportamento del sistema e i fenomeni osservati durante le simulazioni. Le metriche scelte derivano sia dai requisiti individuati all'interno della documentazione ETSI sul tema della federazione [2] sia dalle necessità pratiche emerse durante l'implementazione dei moduli e l'esecuzione delle simulazioni. Viene riportata in Tabella 1 i requisiti e raccomandazioni da parte di ETSI dei vari use case estratta da ETSI GR MEC 035 [2].

Use Case	Requisiti ETSI (Requirements / Recommendations)
UC #1 – MEC federation scenario of V2X services	• MEC system discovery (con sicurezza: autenticazione, autorizzazione, gestione identità, charging, monitoring) • MEC platform discovery (identità, servizi offerti, politiche di accesso) • Information exchange tra piattaforme e applicazioni MEC di domini diversi
UC #2 – Multi-operator agreements enabling MEC Federation	• Stessi requisiti di UC #1 (discovery, sicurezza, information exchange) applicati a scenari con accordi multi-operatore
UC #3 – Application instance transfer tra MEC e Cloud	• Supporto alla discovery delle MEC platform da parte del Cloud- Esposizione disponibilità dei servizi prima della migrazione- Distribuzione e validazione pacchetti applicativi • Trasferimento del contesto utente (user context, subscription, eventi) • Supporto instanziazione di applicazioni su richiesta esterna • Supporto (ri)avvio istanze sul Cloud- Switch dinamico del path di comunicazione tra MEC e Cloud
UC #4 – Inter-system communication in un MNO	• Esposizione disponibilità servizi tra sistemi- Distribuzione e validazione pacchetti su più MEC- Trasferimento del contesto utente • Instanziazione applicazioni cross-MEC • Switching del path di comunicazione • Supporto alla connessione del device verso app su MEC host sorgente tramite altri sistemi
UC #5 – MEC federation per connettere servizi diversi	• Registrazione informazioni presso i Federation Management Entities (risorse di calcolo/rete, info applicative) • Scambio di info tra membri della federazione • Supporto comunicazione app-to-app multi-MEC • Discovery della piattaforma appropriata considerando parametri come posizione utente
UC #6 – MEC federation per giochi AR immersivi	• Discovery tra istanze applicative della stessa app • Esposizione KPI (es. latenza) durante la discovery • Supporto instanziazione app basata su KPI richiesti • Selezione della migliore istanza di server/app per gruppi multi-operatore • Valutazione KPI raggiungibili e traffico instradato in base a regole UPF/SMF
UC #7 – MEC federation per disponibilità Edge Service in roaming	• Gestione autenticazione e autorizzazione tramite MEC di origine, con possibilità di trasferire credenziali

	• Supporto permanenza utente su rete visitata fino a cambio di rete • Configurazioni di local breakout e interconnessione tra operatori
UC #8 – MEC federation per edge node sharing	• Connettività diretta tra MEC platform e gateway di rete di operatori diversi • Stesse considerazioni del roaming (UC #7) • Pubblicazione/discovery a priori delle applicazioni disponibili per instradamento veloce • Supporto a requisiti stringenti di latenza (es. scenari V2X)

Tabella 1 - Requisiti ETSI per scenari di federazione MEC

Di seguito vengono presentate in maniera dettagliata le metriche adottate, insieme alle motivazioni della scelta e al modo in cui sono state calcolate nel contesto sperimentale.

6.3.1 Tempo complessivo di migrazione

La metrica più importante è il **tempo totale richiesto per completare una migrazione**. Per tempo di migrazione si intende l'intervallo che intercorre tra il momento in cui si verifica un evento di handover di un dispositivo, e l'istante in cui l'applicazione sul nodo di destinazione è pienamente operativa e in grado di erogare il servizio senza perdita di funzionalità.

Questa misura è fondamentale perché rappresenta la percezione diretta dell'utente finale: più il tempo di migrazione è breve, maggiore è la probabilità che il servizio risulti trasparente e che non si verifichino interruzioni percepibili. Viceversa, valori elevati comportano degrado della qualità del servizio, possibili disconnessioni e perdita di dati in transito.

Dal punto di vista metodologico, il tempo complessivo è stato calcolato registrando la differenza di tempo da inizio e fine migrazione all'interno del simulatore con `emit`. L'inizio coincide con il messaggio di notifica generato dal servizio di gestione della mobilità a seguito di un cambiamento di cella; la fine viene identificata quando la nuova istanza dell'applicazione conferma il corretto trasferimento dello stato.

Per comprendere in maniera più approfondita le dinamiche della migrazione, il tempo complessivo è stato scomposto nelle sue componenti principali. Tale approccio ha permesso di identificare i colli di bottiglia e di valutare quali fasi incidano maggiormente sul ritardo totale. Le fasi considerate sono le seguenti:

- **Tempo di segnalazione della migrazione**. In scenari federati, il messaggio che innesca la migrazione deve attraversare i diversi domini, passando attraverso i moduli di federazione. Questo introduce un overhead che non è presente nello scenario intra-operatore. La misura di questo intervallo permette di quantificare l'impatto della federazione sulla segnalazione. Questo tempo è calcolato come la differenza dell'invio dell'appMigrationRequest da parte del MEO1 e l'arrivo al MEO2 in Figura 20 (Sezione 4.3.2).
- **Tempo di istanziazione dell'applicazione**. Il tempo che impiega il sistema a istanziare l'applicazione. Calcolato come la differenza dell'istanzazione dell'applicazione migrata in Figura 21 (Sezione 4.3.3) e il trasferimento dello stato avvenuto.
- **Tempo di registrazione e sottoscrizione all'AMS**. Questo tempo indica il tempo che impiega l'applicazione per iniziare a ricevere le info dall'AMS. Misurare questo valore è utile per capire se nel contesto federato utilizzare il "vecchio" AMS comporta problemi di latenza. Tempo di registrazione all'AMS in Figura 21 (Sezione 4.3.3).
- **Tempo di trasferimento dello stato**. La componente più critica riguarda il trasferimento dello stato dell'applicazione dal nodo sorgente al nodo di destinazione. In Figura 21 (Sezione 4.3.3).

Grazie a questa scomposizione è stato possibile confrontare non solo il tempo totale ma anche il contributo relativo di ciascuna fase, ottenendo una visione più granulare e utile per ottimizzare l'architettura.

6.3.2 Scalabilità e migrazioni concorrenti

Uno degli aspetti più significativi riguarda la capacità del sistema di gestire più migrazioni contemporanee. In scenari reali, infatti, è altamente probabile che più utenti in movimento attraversino le stesse aree di copertura nello stesso momento, innescando procedure di migrazione simultanee.

Per analizzare questo fenomeno, sono state eseguite simulazioni in cui il numero di applicazioni da migrare veniva progressivamente aumentato. Le metriche osservate sono quelle definite in 6.3.1.

Le migrazioni simultanee, intese come i passaggi da una cella a un'altra di un dispositivo nello stesso momento, sono state simulate incrementando progressivamente il numero di

dispositivi, tutti caratterizzati dalla stessa mobilità lineare. Negli scenari intra e federati sono stati condotti esperimenti inizialmente con una singola migrazione, per poi aumentare gradualmente a 10, 50, 100 e 200 migrazioni. Inoltre, nella simulazione è stato previsto che, una volta completata la migrazione, il dispositivo attendesse un intervallo di tempo per poi percorrere nuovamente la traiettoria inversa, così da innescare una seconda migrazione dell'applicazione. Questa analisi ha permesso di stimare la scalabilità della soluzione implementata e di individuare possibili limiti dovuti al carico.

6.3.3 Dimensione dello stato applicativo e tempo di trasferimento

Una delle osservazioni più rilevanti emerse durante le simulazioni riguarda la relazione tra dimensione del pacchetto di stato da trasferire e tempo complessivo di migrazione.
Lo stato applicativo non è sempre uguale: in certi casi può trattarsi di poche centinaia di byte in altri può raggiungere dimensioni molto più elevate. È evidente che la quantità di dati da trasferire ha un impatto diretto sul tempo necessario per completare la procedura. Durante le simulazioni sono stati presi in esame pacchetti di stato di dimensioni crescenti, a partire da 1 KB fino a valori superiori al megabyte. I risultati hanno mostrato un incremento quasi lineare del tempo di trasferimento con l'aumentare della dimensione, ma con alcune variazioni dovute al carico della rete e al meccanismo di segmentazione dei pacchetti.
Questa metrica assume particolare importanza perché mette in luce un compromesso progettuale: mantenere lo stato dell'applicazione ricco di informazioni permette una ripresa più veloce e completa sul nodo di destinazione, ma allo stesso tempo allunga il processo di migrazione. Al contrario, minimizzare lo stato riduce il tempo di trasferimento ma può comportare perdita di contesto e necessità di ricostruire parte delle informazioni. L'analisi di questa metrica ha evidenziato come la scelta del livello di granularità dello stato da trasferire debba essere calibrata in base al tipo di applicazione e ai requisiti di continuità del servizio.

6.4 Risultati scenari controllati

Dopo la definizione degli scenari e delle metriche, si procede con l'analisi dei risultati relativi agli scenari controllati, caratterizzati da una mobilità lineare degli utenti, interamente gestita. L'obiettivo di questa fase è duplice: da un lato, isolare il

comportamento del meccanismo di migrazione in condizioni semplificate e prive di variabili esterne, così da ottenere un riferimento chiaro e confrontabile; dall'altro, fornire una base di confronto necessaria per valutare in seguito le prestazioni in scenari più complessi e realistici.

L'analisi viene condotta inizialmente sui singoli scenari, distinti in intra-operator e federato, per poi proseguire con un confronto diretto che consente di evidenziare le differenze principali e di comprendere in quali condizioni l'approccio federato introduce vantaggi o criticità. Per quanto riguarda il trasferimento dello stato si è previsto uno stato di 1KB.

6.4.1 Risultati scenario intra-operator

Si presentano innanzitutto i risultati relativi allo scenario intra-MEC rappresentato in Figura 32, ossia il caso in cui la migrazione avviene esclusivamente tra nodi appartenenti allo stesso sistema. Tale configurazione rappresenta la condizione di riferimento più semplice e costituisce un punto di confronto essenziale con lo scenario federato, nel quale entrano in gioco ulteriori complessità dovute all'interconnessione tra domini gestiti da operatori differenti.

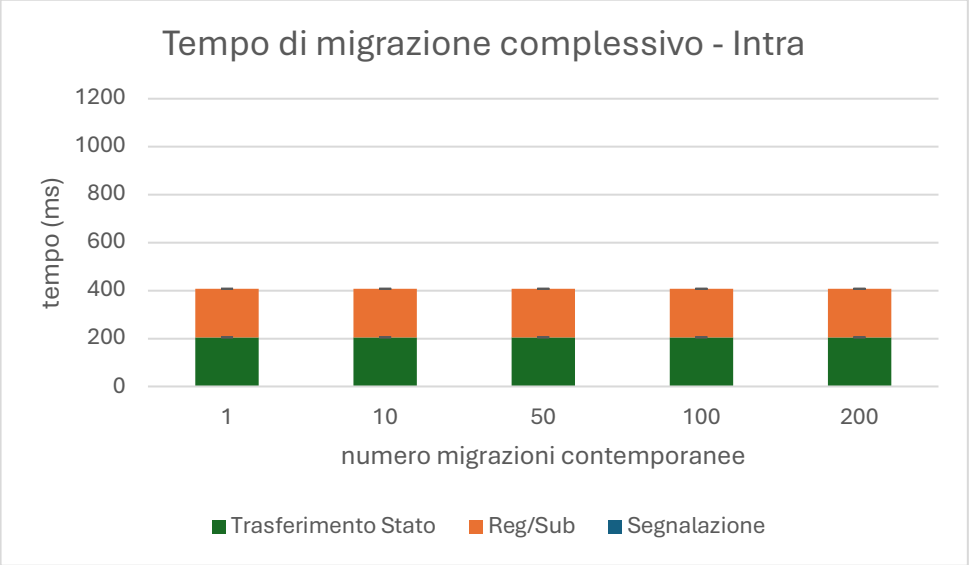

Figura 36 - Tempo di migrazione complessivo intra-MEC

L'analisi riportata in Figura 36 mostra il tempo di migrazione complessivo, misurato tra l'istante in cui si verifica l'handover e il completamento del trasferimento dello stato verso la nuova istanza applicativa. I risultati evidenziano che, anche al crescere del

numero di migrazioni contemporanee, ovvero nel caso in cui più dispositivi cambiano cella nello stesso momento, il tempo complessivo non subisce variazioni significative. Ciò è spiegabile osservando la composizione della misura:
Il tempo di segnalazione è nullo, poiché il passaggio dei messaggi avviene nello stesso sistema e il tempo di computazione è trascurabile in un sistema di simulazione. Per quanto riguarda il trasferimento dello stato notiamo come il numero di migrazioni non influisce, questo perché le migrazioni prevedono un collegamento diretto tra l'applicazione vecchia e nuova, utilizzando la rete internet, senza passare dai componenti del sistema. Invece per quanto riguarda il tempo di Registrazione/Sottoscrizione al servizio AMS non notiamo cambiamenti, questo dovuto alla ottima gestione del servizio con il sistema delle notifiche.

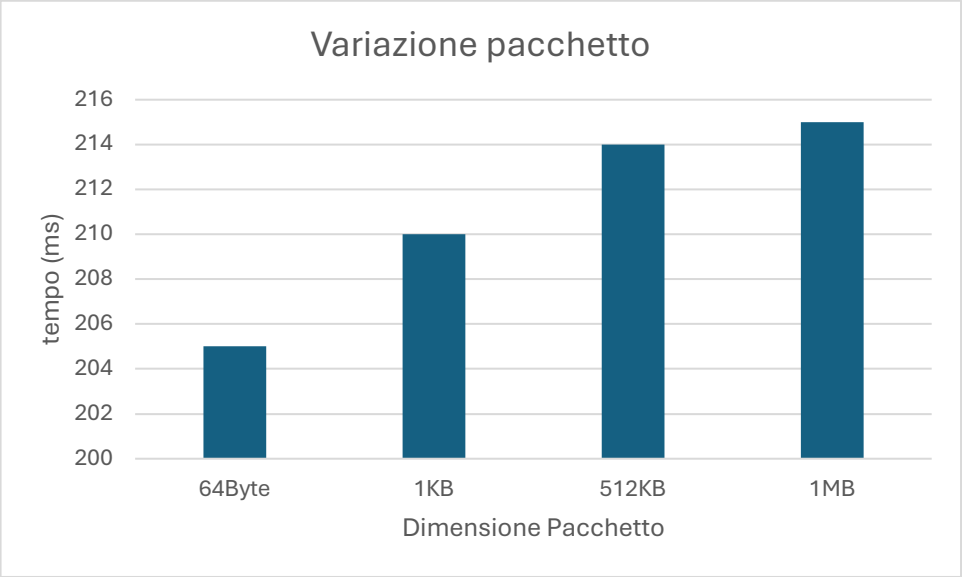

Figura 37 - Tempo di trasferimento stato al variare della dimensione del pacchetto

Si è quindi confrontato il tempo di trasferimento dello stato al variare della dimensione. Dal grafico di Figura 37 notiamo come il tempo di trasferimento rimane di media intorno ai 210 ms, una piccola variazione di 10 ms tra 64 Byte e 1 MB.

Un ulteriore elemento di solidità dei risultati è dato dalla deviazione standard tendente a 0, che conferma l'uniformità delle misurazioni raccolte. In generale, il comportamento osservato mette in luce l'efficienza del sistema MEC in contesto intra, dove anche scenari di mobilità intensiva non comportano un aumento della latenza associata al processo di migrazione applicativa. Questi risultati ci serviranno per confrontare nel complessivo la differenza con lo scenario federato.

6.4.2 Risultati scenario federato

I risultati relativi allo scenario federato di Figura 33 portano a considerazioni di maggiore rilievo rispetto al caso intra-operator.

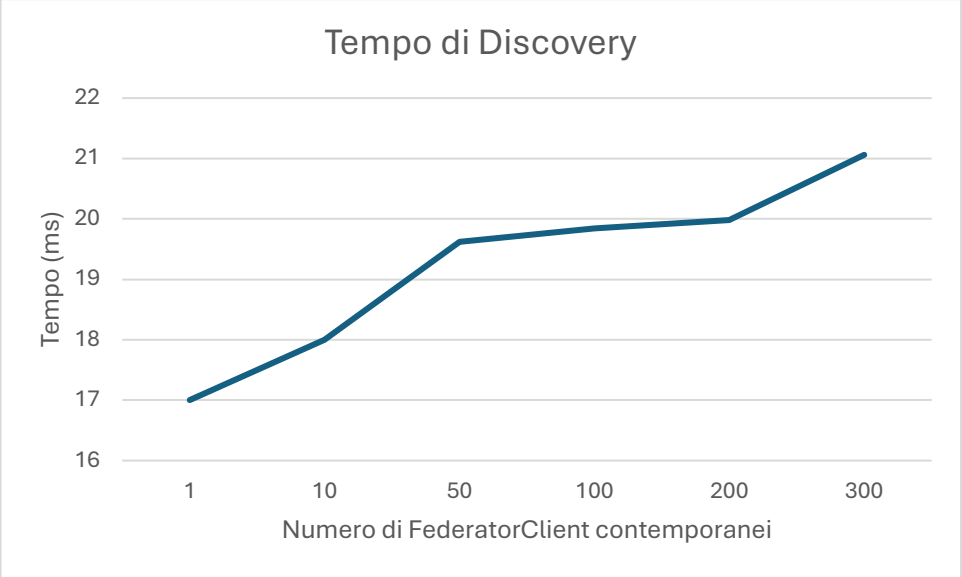

Figura 38 - Tempo di discovery federatori

Il grafico riportato in Figura 38 mostra il tempo medio di discovery in funzione del numero di registrazioni contemporanee effettuate dai Federatori verso il Broker. I risultati indicano che, pur registrandosi un incremento progressivo con l'aumentare del numero di client, il tempo rimane comunque contenuto: anche nel caso di 200 registrazioni simultanee non supera i 20 ms.

Questo comportamento è riconducibile all'efficienza del meccanismo di gestione del Broker, che all'arrivo di una nuova richiesta si limita a inserire l'informazione di registrazione, rendendo il client immediatamente disponibile per successive operazioni di migrazione.

Analogamente a quanto fatto in precedenza, è stato calcolato il tempo complessivo di migrazione nelle condizioni in cui più dispositivi effettuano un cambio di cella in maniera simultanea.

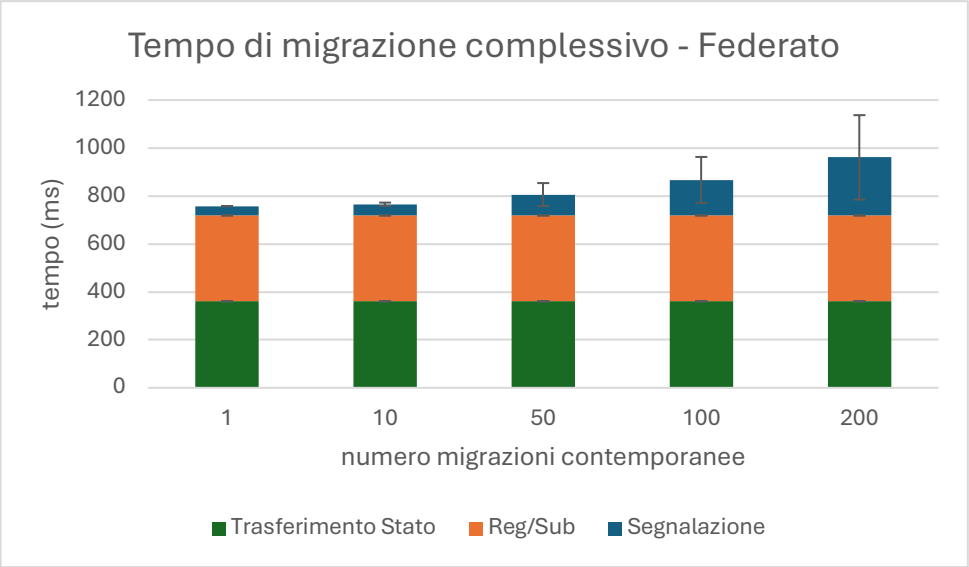

Figura 39 - Tempo di migrazione complessivo federato

Notiamo subito nel grafico di Figura 39 che all'aumentare delle migrazioni contemporanee varia il tempo di migrazione complessiva. Se analizziamo meglio scomponendo il tempo di migrazione nei suoi intermedi notiamo immediatamente che i tempi per il trasferimento dello stato e quelli della registrazione e sottoscrizione all'AMS, rimangono stabili all'aumentare delle migrazioni. Notiamo che invece l'unico valore che cambia è il tempo di segnalazione. Abbiamo la federazione, infatti, la richiesta di migrazione deve attraversare una catena di attori aggiuntivi MEO richiedente, Federatore richiedente, FederatoreBroker, Federatore ricevente e MEO ricevente. Questo passaggio avviene nella rete internet e comporta di per sé un aumento della latenza. L'altro fattore che fa aumentare questo valore è la gestione delle richieste, il Federatore e il Broker devono gestire tutte le richieste ottenute nello stesso momento, mettendole in coda.
In generale però notiamo che questa differenza non è molto significativa si parla di circa 200ms tra il caso di 1 migrazione e 200 contemporanee.
La deviazione standard, pur mostrando un leggero incremento, rimane bassa; ciò conferma non solo la consistenza delle misure, ma anche la presenza del meccanismo di accodamento che regola l'ordine di esecuzione delle migrazioni.

Come osservato nelle analisi precedenti, il tempo di trasferimento dello stato rimane costante al crescere del numero di migrazioni simultanee. Ciò accade perché le applicazioni comunicano direttamente tra loro, senza coinvolgere componenti intermedi del sistema. Per approfondire il fenomeno, lo studio si è quindi concentrato sulla

variazione del tempo di trasferimento al variare della dimensione del pacchetto di stato, così da avere un confronto più realistico rispetto a scenari applicativi eterogenei.
Infatti, mentre alcune applicazioni si limitano a scambiare informazioni di contesto o piccoli metadati, altre necessitano di trasferire quantitativi significativi di dati. Nella Figura 40 è presente il grafico del tempo di trasferimento dello stato dalla vecchia applicazione alla nuova, in particolare è stato preso il tempo nel momento di notifica di inizio trasferimento fino al trasferimento completato. Il pacchetto è stato fatto variare da 64 Byte, 1KB, 512KB e 1MB. Nella simulazione è stato preso il caso di 50 migrazioni contemporanee. La scelta di questi valori non è stata casuale ma è dovuta alle capacità dell'ambiente di simulazione.

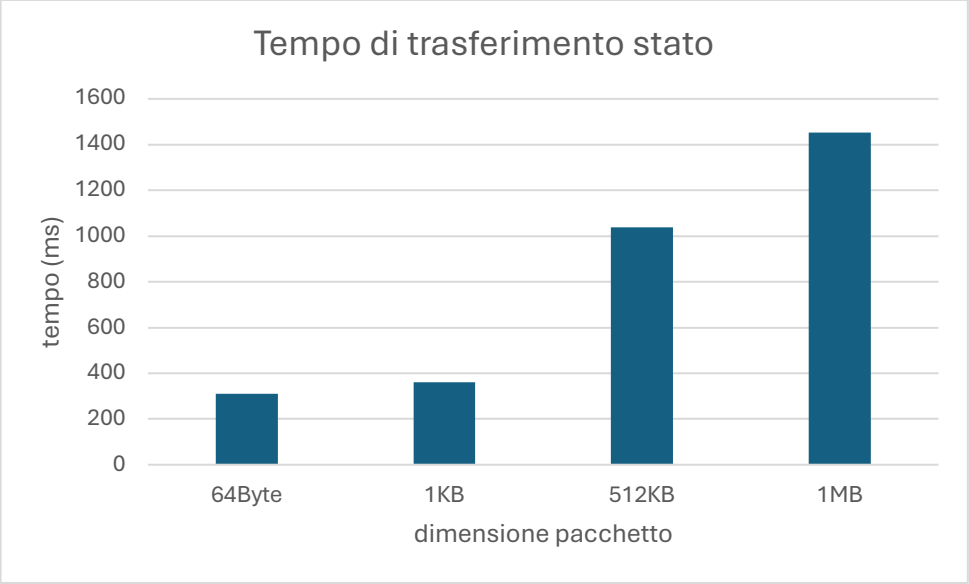

Figura 40 - Tempo di trasferimento stato al variare della dimensione del pacchetto

Dai risultati emerge chiaramente che la dimensione del pacchetto ha un impatto diretto sul tempo di trasferimento. Questo incremento è spiegabile attraverso diversi fattori come la frammentazione dei pacchetti di rete, all'aumentare della dimensione i dati devono essere suddivisi; Latenza di serializzazione e deserializzazione, pacchetti più grandi hanno bisogno di tempi maggiori di elaborazioni e gestione del buffer di trasmissione e ricezione; Congestione della rete interna in quando le migrazioni avvengono in parallelo.

Dalle analisi precedenti è emerso come l'interazione tra componenti distribuiti in rete introduca un incremento dei tempi di migrazione. Per approfondire questo aspetto, è stato eseguito uno studio specifico sui tempi di discovery dei Federatori presso il Broker di

federazione. Questo dato risulta particolarmente rilevante per valutare se un sistema MEC, una volta registrato, sia immediatamente disponibile a ricevere e gestire migrazioni in un contesto federato.

6.4.3 Discussione risultati scenari controllati

Valutando i diversi scenari e le misurazioni possiamo innanzitutto confermare l'efficacia dell'utilizzo del sistema MEC, per capire le differenze tra lo scenario intra-operator e federato dobbiamo confrontare le metriche ottenute.

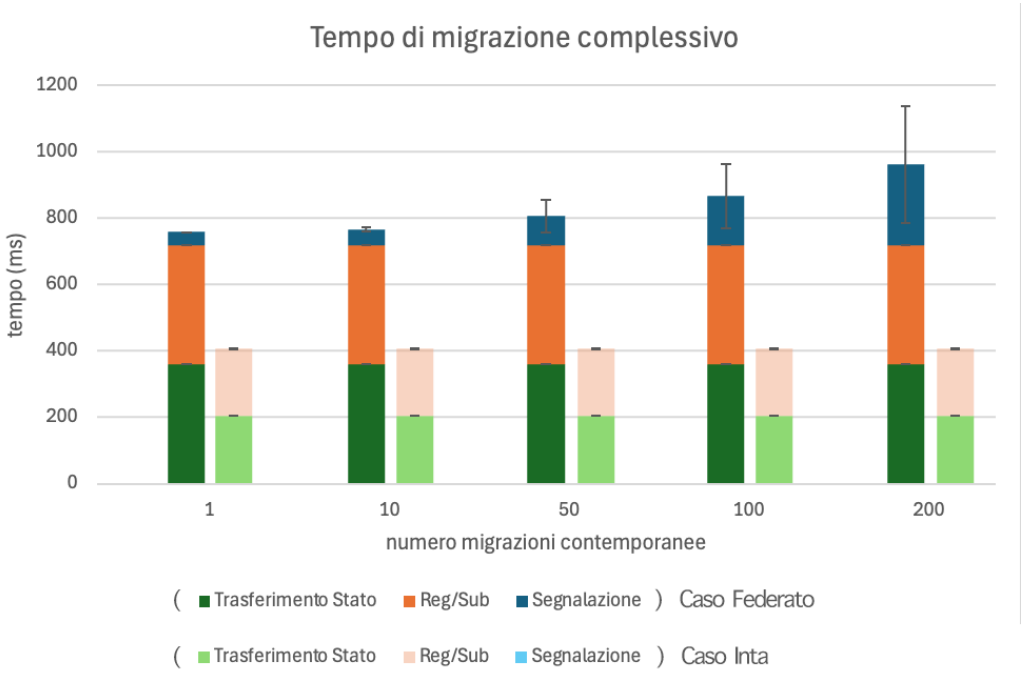

Figura 41 - Tempo di migrazione complessivo, confronto tra scenari

Nella Figura 41 possiamo notare come l'avere uno scenario federato aumenta i tempi per la migrazione, in particolare come già comparato prima nello scenario federato abbiamo il tempo di Segnalazione che nel caso intra è nullo. L'altra differenza che risalta all'occhio è che i tempi di registrazione e sottoscrizione e di trasferimento stato sono più alti nel caso federato. Nel caso intra abbiamo 200 ms circa sia per la registrazione che la sottoscrizione; invece, nel caso federato abbiamo rispettivamente circa 400 ms ognuno. Il che è sempre dovuto al fatto che nella nostra soluzione l'applicazione ha un'interazione con l'AMS del sistema da cui avviene la migrazione, che si trova nell'altro sistema, aumentando così il tempo di latenza dovuta alla distanza. Stesso motivo, la distanza è quello che porta all'aumento del tempo di trasferimento dello stato. In generale quindi possiamo dire che l'unico difetto è che la distanza tra i sistemi comporta una piccola

latenza. E che la gestione contemporanea delle migrazioni comporta una coda nella logica applicativa del federator.

Nell'analisi della soluzione proposta per il funzionamento della migrazione con l'utilizzo di entrambi gli AMS dei due sistemi nel caso federato ha portato alla luce il problema dell'aumento del tempo di trasferimento dei messaggi con l'AMS del sistema che ha l'applicazione migrata e l'aumento significativo del tempo di trasferimento stato nel caso in cui un'applicazione avesse uno stato di dimensioni maggiori.

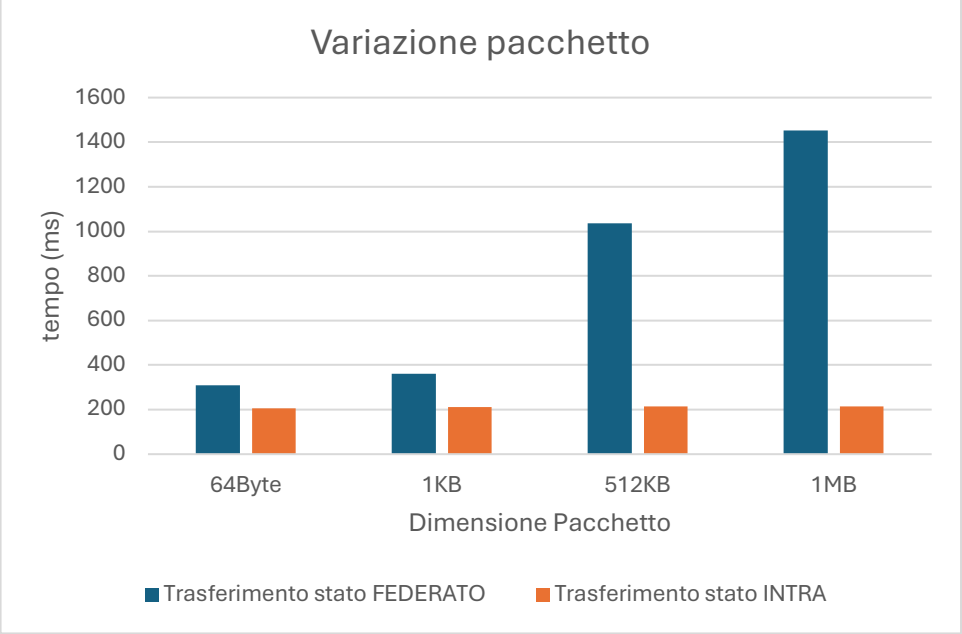

Figura 42 – Variazione dimensione stato

Confrontando i risultati ottenuti dallo scenario intra e federato sulla variazione della dimensione dello stato, nella Figura 42, notiamo come nel caso interno dove le applicazioni sono vicine, la dimensione non incide molto sul tempo di trasferimento a differenza del caso federato.

In generale, possiamo concludere che il principale svantaggio riscontrato è legato alla distanza e all'interconnessione tra i sistemi, che introducono una latenza addizionale rispetto al caso intra-operator, ma che in generale sono un buon compromesso nell'avere una migrazione nel caso di sistemi MEC differenti.

Inoltre, i risultati confermano le indicazioni presenti negli studi ETSI GR MEC 035 [2] che sottolineano come la continuità del servizio in ambienti multi-operatore dipenda più dall'efficienza del signalling e della gestione dei contesti applicativi che dal numero assoluto di migrazioni concorrenti.

Se estendiamo l'analisi a un contesto più ampio, caratterizzato dalla presenza non soltanto di due ma di molteplici operatori, i risultati ottenuti sul discovery dei federatori e dalle simulazioni a due operatori consentono di trarre una considerazione importante. L'interazione tra i federatori rimane infatti molto rapida anche all'aumentare del numero di domini coinvolti. Questo perché la logica alla base del meccanismo di federazione è stata progettata in modo semplice ed efficiente: il federatore che riceve la richiesta non esegue operazioni complesse di elaborazione, ma si limita a inoltrare il messaggio.

Di conseguenza, il tempo di discovery non cresce in maniera significativa con il numero di operatori federati: che siano due o centinaia, solo il federatore responsabile della zona di copertura interessata gestirà effettivamente la richiesta. Tale comportamento dimostra la scalabilità del meccanismo di federazione e la sua idoneità a scenari di Smart City con più operatori, inoltre in contesti di Smart City si potrebbe pensare alla gestione del Federatore Broker in domini geografici, in modo da velocizzare la ricerca.

6.5 Risultati scenario dinamico

Nello scenario dinamico la simulazione non è più condotta in maniera controllata come nei casi precedenti: non si impone un numero prefissato di migrazioni contemporanee, ma si lascia che il sistema riproduca dinamiche di traffico realistico. Sono quindi state condotte tre simulazioni con intensità crescente: nella prima è stato generato un veicolo ogni 12 secondi per ciascuna strada, nella seconda uno ogni 6 secondi, mentre nella terza uno ogni 3 secondi.

Il primo confronto, in continuità con quanto fatto negli altri scenari, riguarda il tempo complessivo di migrazione. Come mostrato in Figura 43, l'aumento del traffico non incide in maniera significativa sul tempo di migrazione, sia nel caso di migrazione federata con una media del tempo di migrazione complessiva di circa 750 ms, che in quello intra-operator, con una media di circa 450 ms.

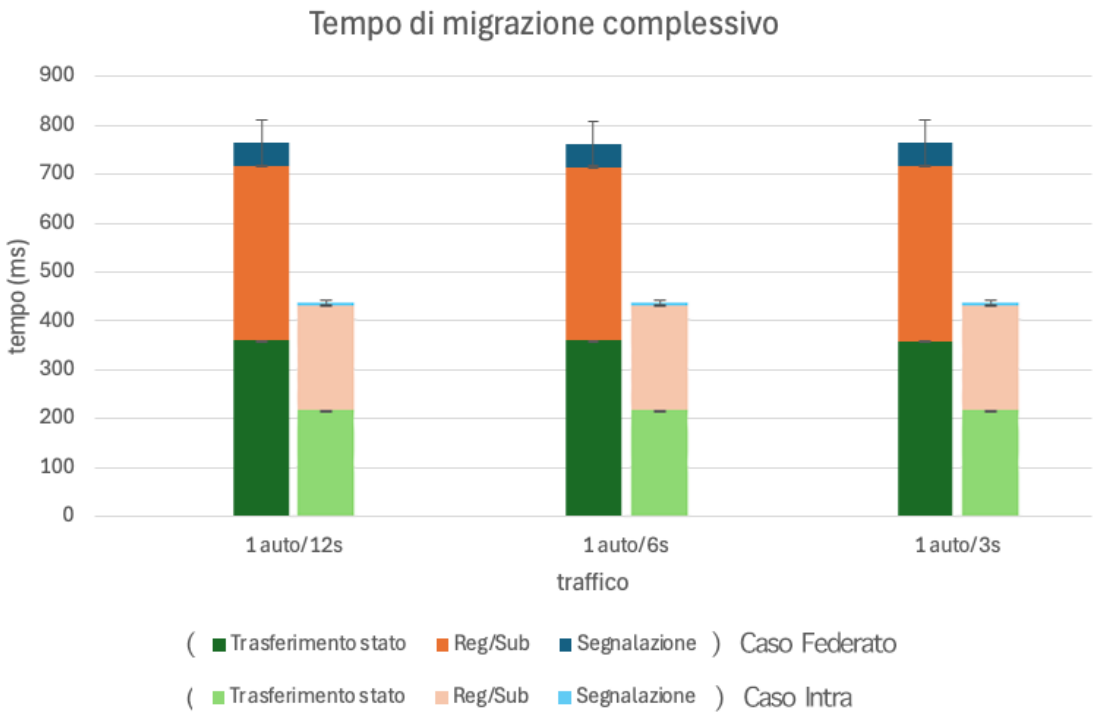

Figura 43 - Tempo di migrazione complessivo Scenario Dinamico

Un aspetto di rilievo riguarda invece il numero di migrazioni effettivamente eseguite durante i cinque minuti di osservazione del traffico all'incrocio. In Figura 44 si nota come tale numero tenda a diminuire al crescere dell'intensità veicolare, con l'aumentare del traffico da 1 auto ogni 12s a 1 auto ogni 3s, si passa da 300 migrazioni a 250 migrazioni in 5 minuti di simulazione. Questo risultato è spiegabile considerando che un traffico più elevato comporta un maggior tempo di permanenza dei veicoli in coda, con conseguente riduzione delle occasioni di handover e, quindi, delle migrazioni applicative.

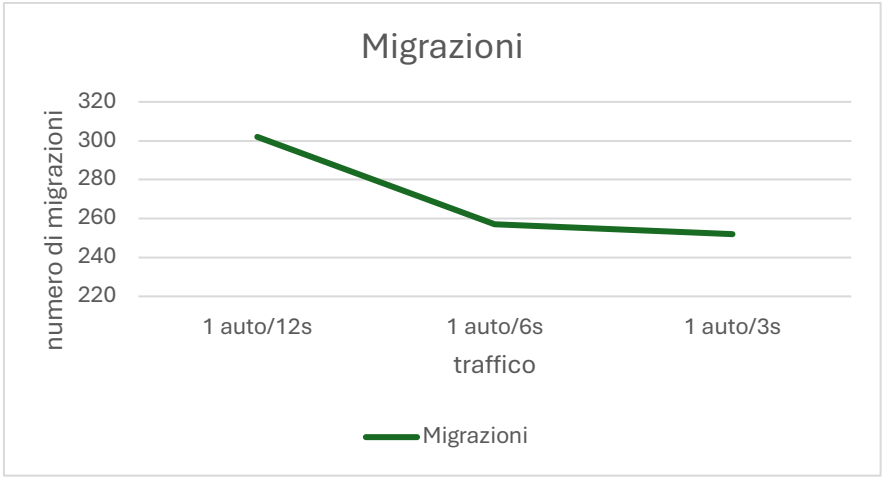

Figura 44 - Migrazioni Scenario Dinamico

6.5.1 Discussione risultati scenario dinamico

Confrontando i risultati dello scenario controllato con quelli dello scenario reale, si osserva, analizzando le Figure 41 e 43, come il tempo complessivo di migrazione sia sostanzialmente allineato con il caso di 10 migrazioni contemporanee nello scenario controllato. Questo risultato suggerisce che, anche in un contesto più dinamico e vicino alla realtà urbana, l'overhead introdotto dal meccanismo di federazione non porta a un degrado significativo delle prestazioni.

In altre parole, negli scenari di Smart City caratterizzati dalla presenza di due operatori, il numero di migrazioni simultanee che effettivamente si verificano non raggiunge valori tali da produrre latenze eccessive. Tale evidenza è coerente con il fatto che la mobilità veicolare, condizionata dal traffico e dalle soste ai semafori o agli incroci, limita naturalmente la concentrazione di handover nello stesso istante.

Questo aspetto riveste particolare importanza, poiché dimostra che la federazione tra sistemi MEC può essere applicata in scenari urbani complessi senza comportare penalizzazioni critiche in termini di QoS.

7 Conclusioni

In questo lavoro di tesi è stato affrontato lo studio degli scenari federati nel contesto del Multi-access Edge Computing (MEC), con particolare attenzione alla migrazione delle applicazioni in ambienti multi-operatore 5G. L'obiettivo è stato quello di valutare le potenzialità e i limiti della federazione MEC, analizzando metriche chiave quali tempo di migrazione, latenza, scalabilità e continuità del servizio in condizioni sia controllate sia realistiche.
Per rendere possibile questa analisi, è stato adottato e opportunamente esteso il simulatore Simu5G/OMNeT++, che ha fornito l'ambiente necessario per modellare gli scenari federati, implementare le logiche di discovery e migrazione, e riprodurre la mobilità veicolare tramite integrazione con SUMO/Veins.

Il percorso seguito è stato suddiviso in più fasi:

- Analisi teorica (Capitolo 2 e 3), dove sono stati introdotti i concetti di 5G, MEC, VEC e Federazione, richiamando i principali studi accademici e gli standard ETSI/3GPP di riferimento.
- System Design (Capitolo 4), in cui sono state definite le entità funzionali necessarie per abilitare la federazione e descritte le interazioni con gli orchestratori MEC.
- Implementazione (Capitolo 5), che ha previsto l'estensione di moduli esistenti di Simu5G e lo sviluppo di nuovi componenti per supportare discovery, migrazione e scambio di stato applicativo.
- Analisi sperimentale (Capitolo 6), condotta in tre scenari distinti: intra-operator, federato e dinamico con integrazione di SUMO/Veins per la mobilità veicolare realistica.

I risultati hanno permesso di evidenziare i trade-off tra intra-MEC e federazione, nonché di validare la bontà del modello proposto in termini di scalabilità e robustezza.

Dall'analisi dei risultati emergono alcune osservazioni chiave:

Tempo di migrazione. In scenari intra-operator è generalmente inferiore rispetto a scenari federati, poiché non vi è signalling aggiuntivo tra federatori. Tuttavia, anche nei casi federati il tempo rimane entro valori accettabili (< 100 ms in media), confermando

la fattibilità del modello ETSI. I tempi di registrazione, discovery e sottoscrizione aumentano in ambiente federato, ma non compromettono la continuità del servizio.

Scalabilità. Il discovery dei federatori tramite broker si è dimostrato altamente efficiente: anche con 200 registrazioni simultanee i tempi si mantengono nell'ordine di poche decine di millisecondi. Ciò dimostra che la soluzione è adatta a scenari multi-operatore con numerosi attori.

Dimensione dello stato applicativo. L'aumento della dimensione del pacchetto di stato influisce linearmente sul tempo di trasferimento, passando da ~300 ms (64 B) a oltre 1400 ms (1 MB). Questo risultato sottolinea l'importanza di strategie di trasferimento.

Scenario Dinamico con SUMO/Veins

L'integrazione con la mobilità veicolare ha permesso di dimostrare che il modello rimane valido anche in contesti realistici. Le automobili, collegate a differenti gNodeB, hanno subito handover e migrazioni senza interruzione significativa, validando la continuità del servizio.

Tra le prospettive future si individuano:

- **Ottimizzazione del trasferimento dello stato.** Si apre la discussione su come possa avvenire, in maniera ottimizzata, il trasferimento dello stato applicativo. Una possibile direzione consiste nell'adottare approcci gerarchici, migrando inizialmente solo i dati critici indispensabili alla continuità del servizio e rinviando il trasferimento di quelli secondari, così da ridurre il downtime percepito dall'utente. Un'alternativa complementare è l'impiego di repliche parziali o totali dell'applicazione su più MEC Host, in modo da avere un'istanza già pronta e sincronizzata al momento del trigger di migrazione. In entrambi i casi, la scelta tra copia completa, trasferimento incrementale o replica dipende dalla tipologia di applicazione e dai requisiti di latenza. I risultati ottenuti mostrano quindi che il meccanismo di trasferimento dello stato resta un aspetto critico su cui concentrare studi futuri, per bilanciare tempi di migrazione, risorse e continuità del servizio.

- **Suddivisione geografica**. Un'ulteriore estensione potrebbe riguardare la gestione dei Federator tramite Broker con una suddivisione geografica: in questo modello ciascun Federator locale coordinerebbe i nodi MEC della propria area (quartiere, città, regione), mentre il Broker si occuperebbe dell'interazione tra aree diverse. Questa organizzazione consentirebbe di ridurre la latenza intra-area e migliorare la scalabilità complessiva del sistema.
- **Integrazione di meccanismi di sicurezza avanzati**, non solo legati al passaggio dello stato ma in generale alla migrazione dell'applicazione. Tecniche con l'utilizzo di Blockchain o autenticazione tra federatori.
- **Analisi avanzate,** con scenari utilizzando applicazioni reali real time, con scenari di federazione su larga scala con centinaia di operatori e migliaia di nodi edge.

In conclusione, il lavoro di tesi ha mostrato come la federazione MEC, sebbene complessa, rappresenti una via necessaria per garantire continuità, scalabilità e interoperabilità in ambienti 5G multi-operatore. L'implementazione realizzata in Simu5G costituisce un primo passo verso la sperimentazione di soluzioni realistiche, offrendo un banco di prova riproducibile per la comunità scientifica.

La federazione, resa possibile dall'introduzione del Federator e dal coordinamento tra orchestratori, permette di superare i limiti del MEC tradizionale, aprendo la strada a scenari in cui la mobilità e la collaborazione tra operatori non rappresentano più un ostacolo, ma un'opportunità.

Bibliografia

[1] 3GPP, «TS 23.501 v16.6.0 - System architecture for the 5G System (5GS),» 2020.

[2] ETSI, «GR MEC 035 v3.1.1 - Multi-access Edge Computing (MEC); Study on Inter-MEC systems and MEC-Cloud systems coordination».

[3] ETSI, MEC federation: deployment considerations.

[4] A. Varga, «THE OMNET++ DISCRETE EVENT SIMULATION SYSTEM,» 2001.

[5] N. Lan, «Evolution of Wireless Technology: From 1G to 5G,» *Asian Journal of Applied Science and Technology,* pp. 68-73, 2023.

[6] 3GPP, «TS 22.261 v16.14.0 - Service requirements for the 5G system,» 2020.

[7] ETSI, «GS NFV-002, V1.2.1 - Network Functions Virtualisation (NFV); Architectural Framework,» 2014.

[8] ETSI, «GS MEC 002 – Use Cases and Requirements,» 2023.

[9] M. Gerla, «Vehicular Cloud Computing,» *2012 The 11th Annual Mediterranean Ad Hoc Networking Workshop (Med-Hoc-Net),* pp. 152-155, 2012.

[10] C. C. Q. P. S. M. Y. Z. Lei Liu, «Vehicular Edge Computingand Networking: ASurvey,» 2020.

[11] ETSI, «GS MEC 003 - Multi-access Edge Computing (MEC); Framework and Reference Architecture,» 2024.

[12] ETSI, «GS MEC 021 v2.2.1 - Multi-access Edge Computing (MEC); Application Mobility Service API,» 2022.

[13] ETSI, «GS MEC 040 - Multi-access Edge Computing (MEC); Federation enablement APIs,» 2024.

[14] M. A. G. ,. (. M. I. A. S. L. LING HOU, «A Survey of Multi-Access Edge Computing and Vehicular Networking,» 2022.

[15] «A Survey on Mobile Edge Computing: The Communication Perspective,» *Yuyi Mao, Student Member, IEEE, Changsheng You, Student Member, IEEE, Jun Zhang, Senior Member, IEEE, Kaibin Huang, Senior Member, IEEE, and Khaled B. Letaief, Fellow, IEEE,* 2017.

[16] S. M. I. K. S. B. M. H. F. S. D. a. D. S. Tarik Taleb, «On Multi-Access Edge Computing: A Survey of the Emerging 5G Network Edge Cloud Architecture and Orchestration,» 2017.

[17] M. I. a. Z. B. M. I. Pavel Mach, «Mobile Edge Computing: A Survey on Architecture and Computation Offloading,» 2017.

[18] Y. Z. S. M. I. A. T. M. I. a. T. S. Nasir Abbas, «Mobile Edge Computing: A Survey,» 2018.

[19] 5GAA, «MEC for Automotive in Multi-Operator Scenarios,» 2021.

[20] A. a. W. S. a. L. K. K. a. K. B. J. a. S. T. Machen, «Live Service Migration in Mobile Edge Clouds,» *IEEE Wireless Communications,* vol. 25, n. 1, pp. 14-147, 2018.

[21] M. V. e. L. T. e. H. H. T. e. O. T. Q. Ngo, «Mograzione coordinata di container e passaggio di consegno della stazione base nel Mobile Edge Computing,»

GLOBECOM 2020 - 2020 IEEE Global Communications Conference, pp. 1-6, 2020.

[22] J. M. R. D. A. J. B. P. F. J. R. C. a. M. M. Hosseini SM, Cooperative, Connected and Automated Mobility Service Continuity in a Cross-Border Multi-Access Edge Computing Federation Scenario, 2022.

[23] H. v. d. B. W. I. R. d. S. S. a. M. D. Tiia Ojanper, «Application Synchronization among Multiple MEC Servers in Connected Vehicle Scenarios,» 2018.

[24] X. a. T. G. a. G. D. a. L. Y. a. Z. W. Cao, Edge Federation: Towards an Integrated Service Provisioning Model, 2020.

[25] E. C.-C. A. e. C. a. J. N. A. A. ,. M. A. S. M. D. J. M. R. P. S. P. A. D. R. V. S. S. Kurdman Rasol, «MEC Federation for Seamless Service Continuity in Cross-Border Mobility Scenarios,» 2024.

[26] P. A. F. a. A. Ksentini, «Service migration versus service replication in Multi-access Edge Computing,» 2018.

[27] R. G. G. G. N. Mohammed A. Hathibelagal, «Experimental comparison of migration strategies for MEC-assisted 5G-V2X applications,» *Computer Communications,* vol. 197, pp. 1-11, 2023.

[28] W. Niewolski, T. Nowak, M. Sepczuk, Z. Kotulski, R. Artych, K. Bocianiak e J.-P. Wary, «Security Context Migration in MEC: Challenges and Use Cases,» 2022.

[29] A. C. P. B. Angelo Feraudo, «Simulating and Validating Vehicular Cloud Computing Applications in MEC-enabled 5G Environments,» *Journal of Simulation Engineering,* 2024.

[30] P. A. L. a. M. B. a. L. B.-W. a. J. E. a. Y.-P. F. a. R. H. a. L. L. a. J. R. a. P. W. a. E. Wie{\ss}ner, «Microscopic Traffic Simulation using SUMO,» *The 21st IEEE International Conference on Intelligent Transportation Systems,* 2018.

[31] R. G. a. F. D. Christoph Sommer, «Bidirectionally Coupled Network and Road Traffic Simulation for Improved IVC Analysis,» *IEEE Transactions on Mobile Computing (TMC),* 2011.

[32] Cellmapper, «Cell Tower Maps,» [Online]. Available: https://www.cellmapper.net/.

[33] W. S. Sidi Lu, «Vehicle Computing: Vision and challenges,» *Journal of Information and Intelligence,* pp. 23-35, 2023.

[34] M. A. H. F. Geoffrey Wilhelm, «A novel cloud approach for connected vehicles. Applied Sciences,» *Special Issue Cooperative-Intelligent Transport Systems: New Challenges,* 2023.

[35] M. a. Z. R. a. X. W. a. Q. W. a. Z. A. Zhou, «Security and Privacy in Cloud Computing: A Survey,» *Semantics Knowledge and Grid (SKG), 2010 Sixth International Conference On,* 2010.

[36] F. H. P. &. S. C. Dressler, Towards a vehicular cloud - using parked vehicles as a temporary network and storage infrastructure, 2014.