ETSI GR MEC 059 V4.1.1 (2026-06)

GROUP REPORT

Multi-access Edge Computing (MEC): Exploiting Edge Computing Resources

Disclaimer

The present document has been produced and approved by the Multi-access Edge Computing (MEC) ETSI Industry Specification Group (ISG) and represents the views of those members who participated in this ISG.
It does not necessarily represent the views of the entire ETSI membership.

Reference
DGR/MEC-0059v411EdgeComputeRes

Keywords
MEC, network scenarios

ETSI

650 Route des Lucioles
F-06921 Sophia Antipolis Cedex - FRANCE

Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Siret N° 348 623 562 00017 - APE 7112B
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° w061004871

Contents

Intellectual Property Rights

Essential patents

IPRs essential or potentially essential to normative deliverables may have been declared to ETSI. The declarations pertaining to these essential IPRs, if any, are publicly available for **ETSI members and non-members**, and can be found in ETSI SR 000 314: *"Intellectual Property Rights (IPRs); Essential, or potentially Essential, IPRs notified to ETSI in respect of ETSI standards"*, which is available from the ETSI Secretariat. Latest updates are available on the ETSI IPR online database.

Pursuant to the ETSI Directives including the ETSI IPR Policy, no investigation regarding the essentiality of IPRs, including IPR searches, has been carried out by ETSI. No guarantee can be given as to the existence of other IPRs not referenced in ETSI SR 000 314 (or the updates on the ETSI Web server) which are, or may be, or may become, essential to the present document.

Trademarks

The present document may include trademarks and/or tradenames which are asserted and/or registered by their owners. ETSI claims no ownership of these except for any which are indicated as being the property of ETSI, and conveys no right to use or reproduce any trademark and/or tradename. Mention of those trademarks in the present document does not constitute an endorsement by ETSI of products, services or organizations associated with those trademarks.

DECT™, **PLUGTESTS™**, **UMTS™** and the ETSI logo are trademarks of ETSI registered for the benefit of its Members. **3GPP™**, **LTE™** and **5G™** logo are trademarks of ETSI registered for the benefit of its Members and of the 3GPP Organizational Partners. **oneM2M™** logo is a trademark of ETSI registered for the benefit of its Members and of the oneM2M Partners. **GSM**® and the GSM logo are trademarks registered and owned by the GSM Association.

Modal verbs terminology

In the present document "**should**", "**should not**", "**may**", "**need not**", "**will**", "**will not**", "**can**" and "**cannot**" are to be interpreted as described in clause 3.2 of the ETSI Drafting Rules (Verbal forms for the expression of provisions).

"**must**" and "**must not**" are **NOT** allowed in ETSI deliverables except when used in direct citation.

Foreword

This Group Report (GR) has been produced by ETSI Industry Specification Group (ISG) Multi-access Edge Computing (MEC).

1 Scope

The present document studies the efficient utilization and exploitation strategies of edge computing resources in 3GPP architecture's Data Network (DN), taking into account the underlying 3GPP 5G system architecture ETSI TS 123 501 [i.2], ETSI and 3GPP existing standards.

The scope of the present document includes:

- to explore pertinent use cases, and to study the benefits and issues, via considering computing resources in the MEC system deployed in DN;
- to study the impact of fluctuations in computing resources, as well as corresponding alternative solutions against this impact, especially through redundant deployment of applications in the MEC system deployed in DN, considering prediction and proactive deployment of applications, and timing for application uninstallation;
- to discuss potential requirements and enhancements for MEC system architecture and functions.

2 References

2.1 Normative references

Normative references are not applicable in the present document.

2.2 Informative references

References are either specific (identified by date of publication and/or edition number or version number) or non-specific. For specific references, only the cited version applies. For non-specific references, the latest version of the referenced document (including any amendments) applies.

NOTE: While any hyperlinks included in this clause were valid at the time of publication, ETSI cannot guarantee their long-term validity.

The following referenced documents may be useful in implementing an ETSI deliverable or add to the reader's understanding, but are not required for conformance to the present document.

[i.1] ETSI GR MEC 001: "Multi-access Edge Computing (MEC); Terminology".

[i.2] ETSI TS 123 501: "5G; System architecture for the 5G System (5GS) (3GPP TS 23.501)".

[i.3] The SNIA Dictionary.

[i.4] ETSI TS 123 548 (V18.9.0): "5G; 5G System Enhancements for Edge Computing; Stage 2 (3GPP TS 23.548 version 18.9.0 Release 18)".

[i.5] ETSI GS MEC 010-2: "Multi-access Edge Computing (MEC); MEC Management; Part 2: Application lifecycle, rules and requirements management".

[i.6] ETSI GS MEC 011: "Multi-access Edge Computing (MEC); Edge Platform Application Enablement".

[i.7] ETSI TS 123 502: "5G; Procedures for the 5G System (5GS) (3GPP TS 23.502)".

[i.8] ETSI GS MEC 016: "Multi-access Edge Computing (MEC); Device application interface".

[i.9] ETSI GS MEC 048: "Multi-access Edge Computing (MEC); Enablement API for Customer Self-Service".

[i.10] ETSI GS MEC 003: "Multi-access Edge Computing (MEC); Framework and Reference Architecture".

[i.11] ETSI GS MEC 002: "Multi-access Edge Computing (MEC); Use Cases and Requirements".

3 Definition of terms, symbols and abbreviations

3.1 Terms

For the purposes of the present document, the terms given in ETSI GR MEC 001 [i.1] and the following apply:

active-active: pair of components that share a task or class of tasks when both are functioning normally, where one takes on the entire task or tasks when the other component fails

NOTE: As defined in The SNIA Dictionary [i.3].

central DNS resolver/server: DNS resolver/server centrally deployed by the 5GC operator or 3rd party and is responsible for resolving the UE DNS Queries into suitable Edge Application Server (EAS) IP address(es)

NOTE: As defined in ETSI TS 123 548 [i.4].

consensus protocol: protocol that provides a shared and common view about the data published by every currently bidirectionally reachable node in a network

local DNS resolver/server: DNS resolver/server that may be locally deployed by 5GC operator or 3rd parties within the Local DN, and is responsible for resolving UE DNS Queries into suitable EAS IP address(es) within the local DN

NOTE 1: As defined in ETSI TS 123 548 [i.4].

NOTE 2: The L-DNS resolvers/servers may or may not have connectivity with C-DNS depending on the deployment.

3.2 Symbols

Void.

3.3 Abbreviations

For the purposes of the present document, the abbreviations given in ETSI GR MEC 001 [i.1] and the following apply:

AF	Application Function
AMF	Access and Mobility Management Function
BP	Branching Point
CCMF	Computing Coordination and Management Function
CRAM	Computing Resource Awareness and Management
EASDF	Edge Application Server Discovery Function
EV	Electric Vehicle
FAR	Forwarding Action Rules
IOPS	Input/output Operations Per Second
P2P	Peer-to-peer
PCO	Protocol Configuration Options
PDR	Packet Detection Rules
SFC	Service Function Chaining
TPS	Transactions Per Second
UDR	Unified Data Repository
UL CL	Uplink Classifier

4 Overview

The present document describes the key study areas within the MEC system that are necessary to support the efficient utilization and exploitation of computing resources across edge data networks.

Clause 5 presents use cases illustrating the effective exploitation of edge computing resources to support enhanced resource utilization and service delivery.

Clause 6 proposes all identified key issues and their related solution proposals and evaluation.

Based on identified gaps, clause 7 contains recommendations for further work.

5 Use cases

5.1 Use case #1: MEC computing resources dynamic scaling

5.1.1 Description

Dynamic scaling of MEC computing resources demonstrates significant potential across various business sectors, especially in industries with fluctuating computing resource demands, high requirements for low-latency processing, and a strong need for edge computing capabilities. Through dynamic scaling, MEC systems can significantly improve service quality, accelerate system response times, and reduce operational costs. The principles of MEC computing resources dynamic scaling are as follows:

- When the workload exceeds the maximum capacity of a MEC application instance in a MEC host, a significant degradation in the quality of edge computing services may occur. To address this, it may be required to instantiate the MEC application in a different MEC host, while ensuring the continuity and quality of service.
- When MEC application instances are detected to be idle or operating under low load conditions, the system can terminate, migrate, consolidate some of these low-load instances to release the resources for other uses, thereby optimizing resource utilization.

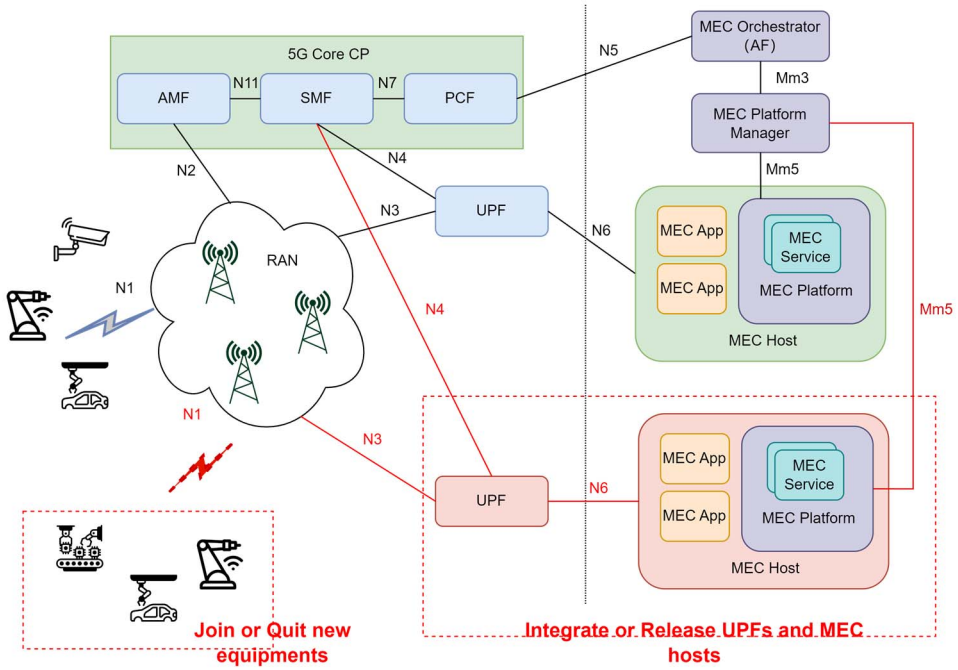

Figure 5.1.1-1: MEC resources dynamic scaling in Smart Factory

Figure 5.1.1-1 illustrates a smart factory setup where MEC computing resources are managed dynamically. By adjusting UPF and MEC host and/or application deployments, the system can respond to changing demands, such as peak processing needs or reduced workload periods, maintaining efficient operations and resource utilization within the factory's 5G network.

Table 5.1.1-1 summarizes various application scenarios where edge computing is utilized, their roles, and the associated benefits of dynamically scaling edge computing resources based on demand.

Table 5.1.1-1: Application scenarios for MEC resource scaling

Application Scenario	Requirements for Edge Computing	Dynamic Scaling Benefits
Event Streaming & Real-Time Interaction Management	Manage traffic spikes during live event streaming (e.g. sports events, concerts), with increased video stream processing demand.	Dynamic resource expansion to support large numbers of simultaneous viewers and resource reduction post-event.
	Real-time data from sensors, cameras, and other sources is collected during the game and transmitted to edge computing host for immediate processing and analysis. The system then uses data algorithms to predict game trends, identify key moments, and generate real-time commentary, such as highlighting a player's performance, which is quickly delivered to the audience.	Support for sudden increases in viewership during critical event moments, ensuring smooth streaming and real-time data processing.
Smart Manufacturing & Quality Control	Support real-time machine control, sensor data analysis, and production line monitoring.	Dynamic scaling of MEC resources based on production load and resource demands during peak production periods.
	Perform quality inspection and predictive maintenance using edge computing.	Resource expansion to handle large-scale data processing during high production periods; reduction of resources during off-peak times.
Retail Logistics & Personalized Shopping	Support robotics, drones, and inventory management in warehouses.	Resource scaling during high demand periods (e.g. shopping festivals, holidays) to handle real-time order processing.
	Process edge data for personalized shopping experiences, including video analysis, customer behaviour analysis, and product recommendations.	Scaling to handle large customer foot traffic; shrinking resources during non-peak times to reduce waste.
Smart Healthcare Monitoring & Emergency Response	Support real-time remote diagnostics, patient data processing, and medical image analysis.	System expansion during high-demand periods (e.g. large medical conferences or emergencies) to accommodate increased healthcare services.
	Coordinate smart medical devices and process large amounts of patient and equipment data.	Scaling of MEC resources during peak times (e.g. flu season or health emergencies) to handle large volumes of medical data and patient information.
Gaming & AR/VR Experience Management	Handle real-time game data, graphic rendering, and physics calculations for multiplayer online games.	Scale MEC resources during peak game participation to handle large data, rendering, and compute tasks, then reduce resources as players decrease.
	Process large amounts of AR/VR interaction data in malls, tourist attractions, and museums.	Expand resources during peak tourist seasons or high visitor traffic to support immersive experiences.

Dynamic scaling of computing resources can be achieved through various strategies, emphasizing flexibility and efficiency in expanding capacity as needed. Key approaches include dynamically scheduling and adjusting existing resources or activating pre-deployed but inactive MEC hosts to accommodate increased workloads and meet demand efficiently. These strategies include:

- **Utilizing lightly loaded MEC hosts in the nearby edge and/or core network:** By dynamically instantiating or activating MEC application instance in the lightly loaded MEC hosts in nearby edge locations and/or the core network, these resources can be effectively leveraged to meet computing demands during peak periods or unexpected surges. This ensures efficient resource utilization and continuous service operation while minimizing latency and maintaining optimal performance.

- **Adjusting MEC host tasks to free up computing resources:** By migrating or terminating MEC application instances related to certain low-priority computational tasks and appropriately adjusting the load on local or remote MEC hosts, new computing resources are freed up within active hosts. This helps optimize resource utilization and load balancing, ensuring system performance and service quality.

- **Activating pre-provisioned MEC hosts to expand computing resources:** During demand surges, pre-deployed but idle MEC hosts can be swiftly activated at Local UPF or other strategic locations to scale computing capacity. This approach leverages existing infrastructure to quickly meet user needs and maintain a positive user experience, aligning with the principles of dynamic scaling by optimizing resource allocation without introducing new hardware.

In dynamic MEC environments, MEC systems can reclaim and optimize resources in real time based on fluctuating computational demands, prioritizing efficient resource utilization and system stability. Typical scenarios of resource downsizing and optimization include consolidating workloads onto fewer resources during off-peak times, releasing the resources allocated for idle MEC hosts. The following are typical strategies of resource reclamation and optimization:

- **Regular Resource Usage and Health Checks:** The system performs regular resource usage and health checks for all MEC hosts, ensuring resources are not unnecessarily occupied and maintaining the health status of MEC hosts. Through periodic resource usage and health checks, the system can maintain efficient operation of MEC resources, avoid resource waste, and mitigate potential failure risks.

- **Termination of Inactive MEC Application Instances After Peak Demand:** After peak periods, the system detects MEC application instance that are no longer actively serving user demands. These unused MEC application instances can be terminated to free up resources and reduce power consumption.

- **Consolidation and Release of MEC Resources:** Once workloads subside, the system consolidates MEC application instances from multiple lightly loaded MEC hosts onto fewer hosts, freeing up resources for other uses. Priority is given to deactivating temporarily added resources to reduce unnecessary consumption. This approach ensures efficient resource utilization, optimizes computational resource allocation, and maintains flexibility as demand decreases.

5.1.2 Analysis

When considering the performance and effectiveness of MEC computing resources dynamic scaling, the following information items can be described for the use case.

Table 5.1.2-1: Information items to support the use case

Information item	Description
Scaling Frequency	The frequency of scaling operations (in or out) within a certain time period in the MEC system.
Scaling Response Time	The time taken from the request to scaling operations (in or out) to the completion of resource adjustment.
Task Migration Time	The time taken to migrate MEC application instance from one MEC host to another host during scaling operations.
Resource Utilization	The ratio of actual usage of computational resources (CPU, memory, storage, etc.) by specific MEC application instances on a MEC host to the total available resource quota allocated to those instances.
Service Quality Impact	The changes in service quality parameters due to scaling operations, including latency, jitter, throughput, response time, can be measured by comparing the performance parameters before, during, and after the scaling operations.

MEC resource dynamic scaling is essential for optimizing system performance by effectively responding to fluctuating workloads. Information items such as Scaling Frequency, Scaling Response Time, and Task Migration Time is used to assess the performance pertaining to scaling in/out processes, and provide insights into the efficiency and effectiveness of the scaling operations themselves. Resource Utilization and Service Quality Impact serve to assess the extent to which scaling activities affect overall resource optimization and maintain service quality level. These information items enable proactive monitoring and optimization, allowing network operators to detect potential issues, take corrective actions promptly, and assess the performance of strategies and orchestration algorithms.

5.2 Use case #2: MEC computing resources redundant deployment

5.2.1 Description

In a MEC environment, ensuring high availability and continuity of applications is crucial. With the widespread adoption of MEC technology, particularly in critical business scenarios, any node failure or resource overload could lead to service interruptions, impacting user experience and business continuity. In the MEC system, computing resources or services are provided through MEC application instances. Therefore, redundant MEC application deployment have become a key means of ensuring the stability and reliability of MEC systems.

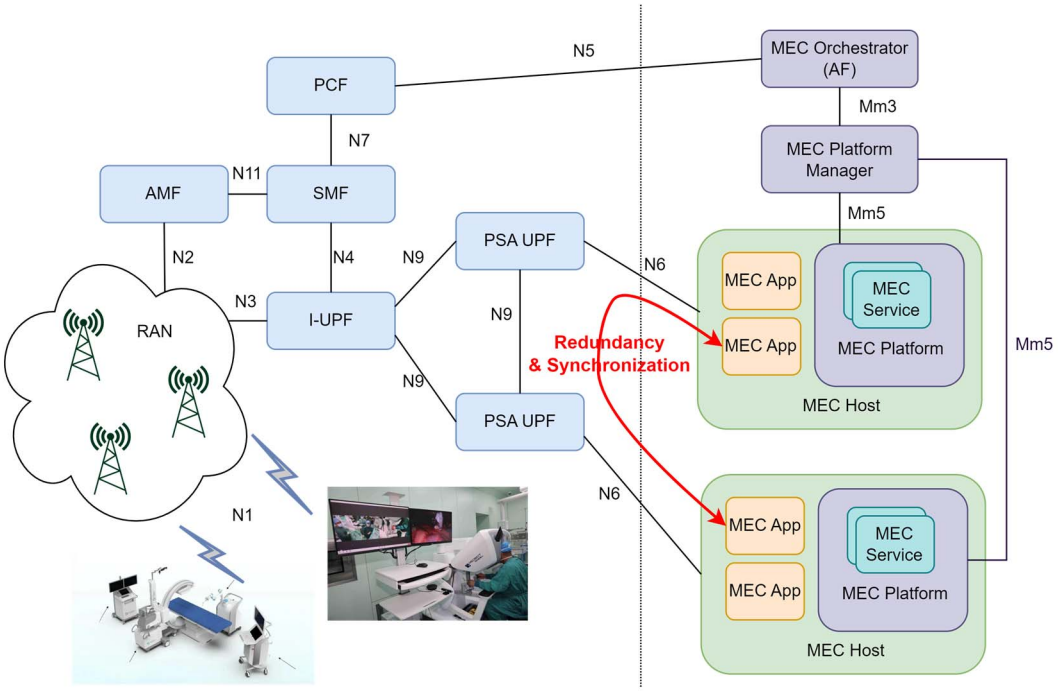

Figure 5.2.1-1: MEC application redundancy deployment in Remote Surgery

Figure 5.2.1-1 illustrates a redundant MEC application deployment architecture designed for remote surgery. In this setup, the MEO and MEPM coordinate to manage the redundancy of MEC application instances. The Mm3 and Mm5 reference points are used for the management of the lifecycle of redundantly deployed applications within the MEC system. The architecture integrates core network elements - AMF, SMF, PCF, and UPF - to handle data routing, policy control, and mobility support. This setup ensures a stable communication pathway that supports the low-latency and reliability requirements of remote surgical operations, while also providing connections between MEC applications for real-time data synchronization and state consistency.

Table 5.2.1-1 summarizes the application scenarios for MEC application redundant deployment and the benefits in each use case.

Table 5.2.1-1: Application scenarios for MEC application redundant deployment

Application Scenario	Requirement for Edge Computing	Redundant Deployment Benefits
Autonomous Driving	Process sensor data, optimize path planning, enable collaborative driving, and support V2X communication.	Ensures system stability under node failure, network fluctuations, or high load; supports real-time data processing and low-latency communication; improves decision-making speed and accuracy.
Remote Surgery	Process video streams, sensor data, and medical equipment data in real time; monitor equipment and patient status.	High reliability for uninterrupted surgery; immediate takeover by redundant nodes in case of primary node failure; ensures continuity and safety.
Intelligent Traffic Management	Process real-time data, monitor traffic, and optimize routes.	Ensures continuous real-time analysis and response to traffic data despite edge node failure; improves stability and efficiency of traffic management.
Industrial IoT	Monitor and control equipment, analyse production data, and execute fault warnings.	Improves reliability of production line monitoring; enables quick failover to backup nodes; optimizes resource utilization and speeds up fault detection.
Smart City Security	Real-time video surveillance, anomaly detection, and alarm notifications.	Ensures monitoring and alarm functions continue even during system failure; reduces data loss risk; improves overall system robustness.

By implementing effective redundancy mechanisms and intelligent resource management, the system's reliability and continuity can be maximized. The following are key designs and implementation methods for redundant deployment:

- **Redundancy Strategy:** Achieve high availability and application continuity in the MEC environment through redundant deployment strategies. Architectures like active-active [i.2] and primary-backup effectively mitigate the impact of node failures and resource overload, ensuring reliable system performance.
- **Transmission Path:** Establish an efficient path between UPFs for real-time data transmission and state synchronization between redundant MEC applications. This high-bandwidth, low-latency connection ensures that redundant MEC nodes can quickly share information, maintain data consistency, and improve overall system performance and reliability.
- **Load Balancing:** Use load balancers to distribute traffic among multiple MEC nodes, optimizing resource utilization, avoiding overload on any single node, and improving system response speed and reliability.
- **Regular Snapshots and Backups:** Regularly generate application snapshots and synchronize them to backup nodes or cloud storage to ensure rapid recovery to the most recent state in the event of a failure, reducing the risk of data loss.
- **Data & State Synchronization:** Data and state synchronization can be achieved using common methods such as P2P synchronization, consensus protocols (like Raft or Paxos), and event-driven messaging systems, ensuring consistency across all redundant MEC applications and enhancing system stability and fault tolerance.
- **Automatic Switching Mechanism:** Continuously monitor the health of MEC applications. Upon detecting a failure, the system can quickly switch to a backup application and redeploy another active application to ensure service continuity.

5.2.2 Analysis

When considering the performance and effectiveness of MEC computing resources redundant deployment, the following network information items can be described for the use case.

Table 5.2.2-1: Information items to support the use case

Information item	Description
Failover Service Continuity Ratio	The ratio of successfully processed requests to total requests when partial failures occur in a redundant deployment scenario for MEC application instances. It reflects the system's fault tolerance capability in the face of hardware or software failures.
Failover Time	The time taken for the MEC system to detect failure and switch to a backup MEC application instance.
Synchronization Latency	The time delay of synchronization by tracking timestamps of state updates on source and target MEC application instances.
Synchronization Overhead	The amount of data used for synchronization purposes between MEC application instances.
Service Quality Impact	The changes in service quality parameters due to MEC application instance switch operations, including latency, jitter, throughput, response time, can be measured by comparing the performance parameters before, during, and after the switch operations.

MEC resource redundancy plays a significant role in supporting mission-critical applications by enhancing system resilience, ensuring uninterrupted service, and optimizing performance. Exposed information items such as Failover Service Continuity Ratio and Failover Time provide a detailed assessment of the system's robustness and responsiveness during failure scenarios. Synchronization Latency and Synchronization Overhead highlight the system's efficiency in managing data synchronization. Service Quality Impact serves to evaluate the impact of these operations on overall service performance, offering insights into potential efficiency losses or performance degradation. These information items offer a comprehensive overview of the system's performance and effectiveness during application instance switches. They facilitate proactive monitoring and optimization, enabling network operators to detect potential issues, implement corrective actions in a timely manner, and assess the effectiveness of strategies and orchestration algorithms.

5.3 Use case #3: Cross-MEC host collaborative edge computing

5.3.1 Description

Cross-MEC Host Collaborative Edge Computing is a computing model where MEC application instances, deployed across multiple MEC hosts, collaborate to process tasks. The primary goal is to handle high computational demands that exceed the capacity of a single MEC host while efficiently processing distributed, multi-source data across different locations. By utilizing computing resources across multiple MEC hosts, this model enables the processing of large, distributed data flows, reduces latency, and improves system fault tolerance, offering more efficient, reliable, and flexible solutions for a wide range of application scenarios.

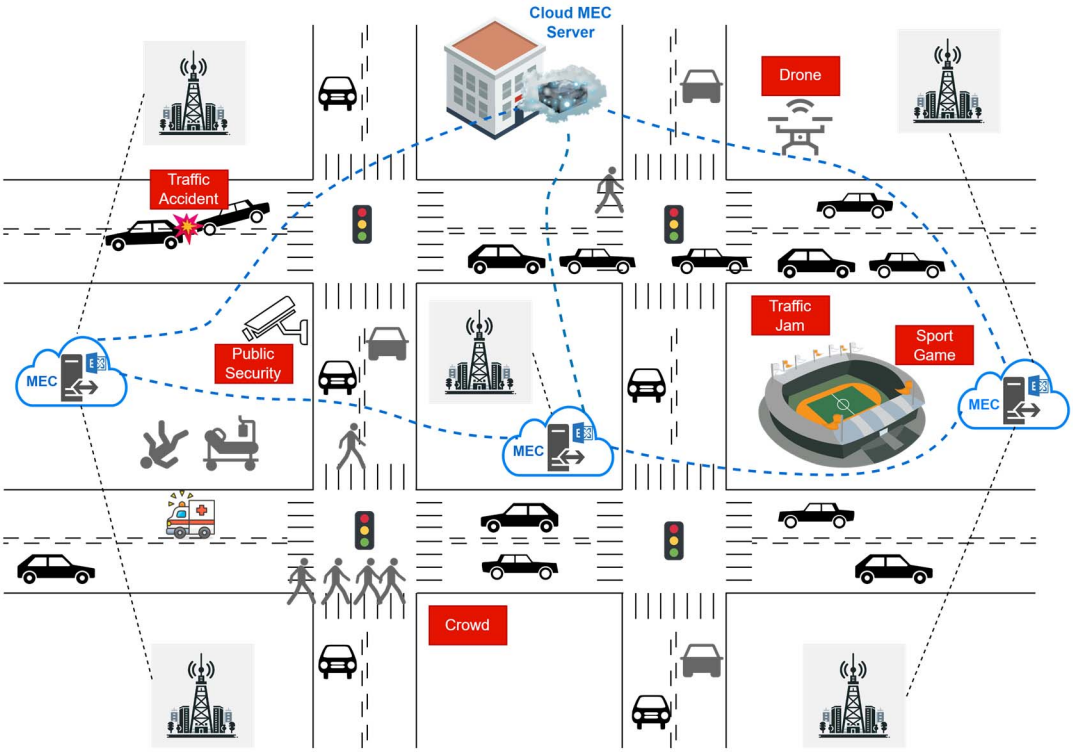

Figure 5.3.1-1: Cross-MEC Host Collaborative Edge Computing in Smart City

Figure 5.3.1-1 illustrates a Cross-MEC Host Collaborative Edge Computing scenario in a smart city, where MEC applications in multiple MEC hosts collaborate to process diverse urban tasks. From traffic management and public security to event support and emergency response, MEC applications work together to share computational loads, reduce latency, and ensure seamless data flow. By dynamically coordinating resources and computation across MEC hosts in different locations, this collaborative approach ensures efficient, reliable, and real-time processing for complex and critical scenarios.

Table 5.3.1-1 summarizes the application scenarios for cross-MEC host collaborative edge computing in smart city and the benefits in each use case.

Table 5.3.1-1: Application scenarios for cross-MEC host collaborative edge computing

Application Scenario	Role of Edge Computing	Cross-MEC Host Collaborative Edge Computing Benefits
Edge AI Model Training	Smart cities rely on video surveillance, sensor data, and V2X information to train AI models. MEC applications in multiple MEC hosts collaboratively train traffic flow prediction models using Federated Learning, where each MEC application processes local data and shares only model parameters instead of raw data. This ensures data privacy while enabling optimized AI models across MEC hosts, enhancing traffic flow prediction, incident forecasting, and crowd analysis.	Distributed Cross-MEC Host Learning: Allows AI models to be continuously optimized using diverse datasets from different locations without requiring data centralization. Collaborative Edge Computing: Reduces network congestion by limiting raw data transmission, enhances AI model generalization across regions, and ensures rapid local adaptation to environmental changes.
Smart Grid Scheduling	Distributed MEC applications in multiple MEC hosts, deployed at substations, microgrids, and EV charging stations, monitor grid loads in real time. During peak load conditions, MEC applications coordinate to intelligently dispatch renewable energy and storage resources, optimizing power distribution. Predictive analysis of electricity consumption trends allows MEC applications to collaborate with cloud MEC servers to ensure grid stability and minimize energy waste.	Distributed Cross-MEC Host Load Balancing: Ensures decentralized energy resource allocation, preventing overload in any single location and improving power grid efficiency. Collaborative Edge Computing: Enhances the resilience of power distribution by dynamically adjusting energy flow across multiple grid segments, reducing power fluctuations and blackouts.
Public Safety and Emergency Response	During natural disasters or emergencies, MEC applications in multiple MEC hosts process real-time data from cameras and sensors to assess the impact zone and coordinate rescue operations. MEC applications in affected areas gather local information and share it with other MEC applications, while cloud MEC servers integrate global data to optimize emergency response strategies, pre-plan evacuation routes, and enhance rescue efficiency.	Distributed Cross-MEC Host Situational Awareness: Enables each MEC application to locally process emergency data, reducing response time while synchronizing information with other MEC hosts for a broader, coordinated rescue strategy. Collaborative Edge Computing: Allows emergency services to dynamically adapt their response based on real-time environmental changes, ensuring optimized allocation of rescue resources across affected regions.
Traffic Flow Optimization and Intelligent Signal Control	MEC applications in multiple MEC hosts at intersections monitor vehicle and pedestrian movement, dynamically adjusting traffic light durations. Collaborating with cloud MEC servers, historical data is utilized to optimize citywide traffic management. In case of accidents, MEC applications near the affected area analyse video streams, assess severity, and notify adjacent MEC applications to modify traffic signals for congestion mitigation. Cloud MEC servers further optimize rerouting strategies, preventing cascading congestion and improving incident response efficiency.	Distributed Cross-MEC Host Traffic Control: Allows real-time signal adjustments at different intersections, preventing localized congestion from escalating into citywide gridlock. Collaborative Edge Computing: Improves overall traffic efficiency by enabling multi-region coordination, optimizing traffic flow dynamically across multiple road networks rather than in isolated segments.
Intelligent Waste Management System	Smart waste management systems use sensor-equipped trash bins and street cameras to collect waste level and pollution data. MEC applications in multiple MEC hosts analyse real-time data and coordinate waste collection logistics. Cloud MEC servers integrate data from multiple MEC applications, predict waste accumulation patterns, and compute optimal garbage collection routes, reducing energy consumption and improving operational efficiency. After large events, MEC applications process crowd density data to predict waste accumulation hotspots and pre-emptively deploy sanitation teams.	Distributed Cross-MEC Host Waste Monitoring: Enables real-time tracking of sanitation needs across different areas, ensuring localized waste issues are addressed efficiently without relying on centralized reporting. Collaborative Edge Computing: Optimizes waste collection logistics by synchronizing collection routes across multiple regions, reducing redundant trips and improving service efficiency.

Cross-host edge computing collaboration can typically be divided into three types: horizontal collaboration, vertical collaboration, and hybrid collaboration. The appropriate collaboration model can be selected based on the complexity of the application scenario, latency requirements, and computation load:

- **Horizontal Collaboration:** Refers to MEC application instances deployed on MEC hosts at the same level collaborating in computation, where tasks or data are shared between hosts to balance the load and improve processing efficiency.
- **Vertical Collaboration:** Refers to hierarchical collaboration between terminal devices, edge hosts, and the cloud. Terminal devices handle simpler tasks, edge hosts process tasks of moderate complexity, and the cloud handles the most complex tasks while integrating global data.
- **Hybrid Collaboration:** Combines the advantages of horizontal and vertical collaboration, supporting cooperation between MEC application instances at the same level and enabling cross-level cooperation between end devices, edge, and cloud, to meet more complex task requirements.

In Cross-MEC Host Collaborative Edge Computing, the collaboration between MEC application instances deployed across multiple MEC hosts may involve various computation models depending on the application requirements, network topology, data processing capabilities, and computing resources. The primary computation models include:

- **Task Partition Mode:** Complex tasks are broken down into several sub-tasks, with MEC application instances on different MEC hosts processing different sub-tasks or task partitions. The final results are aggregated on one host or in the cloud.
- **Data Partition Mode:** Raw data is partitioned and allocated to different MEC application instances on MEC hosts. Each host processes its respective data partition, while the final results are aggregated on one host or in the cloud. This mode is suitable for scenarios requiring parallel data processing.
- **Hybrid Mode:** Combines data distribution, model distribution and task distribution, enabling dynamic collaboration based on the specific application scenario. It supports collaboration between MEC application instances on the same level and allows cross-level collaboration between end-devices, edge, and cloud, balancing computation load, reducing latency, and improving system elasticity and fault tolerance.

To implement Cross-MEC Host Collaborative Edge Computing, dynamic resource orchestration and scheduling and rapid recovery in case of failures are essential to ensure stable and reliable operation under complex computational and network conditions. The key technologies and methods required to achieve these objectives include:

- **Task Orchestration and Scheduling:** In collaborative edge computing, task orchestration and scheduling involve selecting appropriate MEC hosts to deploy application instances based on collaboration and computation models. The orchestration and scheduling strategies are designed to dynamically adapt to changes in application instances, such as resource availability, network conditions, and workload fluctuations.
- **Dynamic Communication Path Management:** Dynamic communication paths can be established and maintained between MEC application instances. As participants may join or leave dynamically, adjustments to PDU session management and UPF traffic steering rules are essential to ensure efficient and reliable data transmission.
- **Coordinator Deployment:** In collaborative edge computing, a dedicated coordinator is essential to manage task scheduling, execution, and state monitoring across multiple MEC application instances deployed on different hosts. The coordinator ensures seamless coordination among distributed application instances, enhancing the overall performance and reliability of the system.
- **Monitoring and Status Reporting:** The coordinator proactively monitors or passively receives status reports from MEC applications and MEC platform, collecting performance data, operational states, and fault information. It then reports this data to the MEC Platform Manager and MEC Orchestrator to ensure timely detection of errors and failures in collaborative edge computing, supporting resource scheduling, optimization, and fault recovery.

5.3.2 Analysis

When considering the performance and effectiveness of cross-MEC host collaborative edge computing, the following network information items can be described for the use case.

Table 5.3.2-1: Information items to support the use case

Information item	Description
Application Instance Throughput	The total volume of data successfully processed by one MEC application instance per unit of time.
Collaborative Processing Throughput	The total volume of data successfully processed by multiple collaborating MEC application instances per unit of time, reflecting the collective computational efficiency of the distributed system.
Coordination Delay	The time difference between the completion of the fastest and slowest MEC applications within a synchronization round in distributed computing or training, reflecting the synchronization overhead introduced by lagging nodes.
Inter-MEC Application Communication Overhead	The amount of data transferred between any two communicating MEC applications among multiple MEC applications engaged in collaborative edge computing. This item is statistically analysed to derive the minimum, maximum, and average values.
Inter-MEC Application Communication Delay	The delays encountered during data transmission between any two communicating MEC applications among multiple MEC applications engaged in collaborative edge computing. This item is statistically analysed to derive the minimum, maximum, and average values.
Interruption Rate	The number of interruptions occurring per unit of time due to the failure of certain application instances in collaborative edge computing across multiple MEC applications, resource unavailability, or other operational issues.
Application Instance Redeployment Duration	The time required to redeploy application instances across MEC hosts until services are restored, due to anomalies, failures, or low processing performance, which affects the stability of collaborative operations.

These information items can be used to establish system performance baselines, monitor real-time operational status, identify performance bottlenecks, and guide the optimization of service scheduling and resource allocation strategies. Application Instance Throughput and Collaborative Processing Throughput are used to measure the data processing capabilities of collaborative edge computing. Coordination Delay is used to analyse the differences in data processing capabilities across different MEC hosts. Inter-MEC Application Communication Overhead and Inter-MEC Application Communication Delay are used to analyse the communication overhead and performance between hosts or applications. Application Instance Redeployment Duration and Interruption Rate are used to assess the ability of collaborative edge computing to handle MEC host and application failures, anomalies, and self-recovery. By analysing these information items in real time, system load distribution can be dynamically adjusted to optimize efficiency and quickly detect and address anomalies. Moreover, these information items are used to verify whether QoS meets SLA demands, assess the system's recovery capabilities during failures or anomalies, and provide a basis for assessing new functionalities and continuous system improvement, ensuring efficient operation in complex service scenarios.

5.4 Use case #4: MEC Application deployment with computing resource awareness

5.4.1 Description

With the widespread deployment of 5G networks and edge computing technologies, the computing resource demands of MEC applications are becoming increasingly diverse and complex. Different MEC applications require varying levels of CPU, GPU, memory, and storage resources, which are often distributed across multiple geographically dispersed edge networks. To ensure efficient deployment, the MEC Orchestrator may possess global computing resource awareness, dynamically manage resource allocation, and optimize the placement of MEC applications based on real-time resource availability.

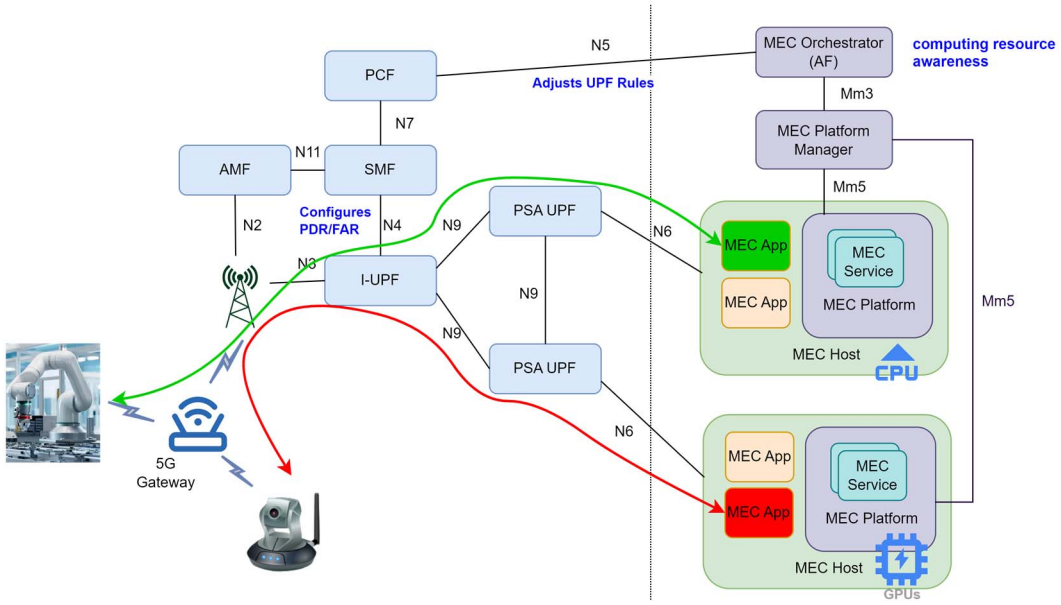

Figure 5.4.1-1: MEC Application Deployment with Computing Resource Awareness

Figure 5.4.1-1 illustrates MEC applications are deployed across different edge networks with computing resource awareness. In industrial networks and similar scenarios, multiple terminal devices may share a single 5G gateway, which is responsible for distributing traffic for various types of client applications. The corresponding MEC applications may be deployed in edge networks connected to different UPFs, requiring flexible traffic routing strategies for UPFs to ensure accurate service transmission.

To enable intelligent deployment, the MEC Orchestrator should have system-wide computing resource awareness, allowing it to dynamically allocate resources across multiple edge network regions. During deployment, MEC applications should be assigned to the most suitable MEC hosts based on available computing resources, network bandwidth, storage capacity, and specific application requirements, for example, whether a given MEC platform has sufficient GPU capacity to support AI model training.

In the 5G core network architecture, the MEC Orchestrator can function as an AF, registering with the 5GC via the NEF and interacting with the PCF. MEC Orchestrator can instruct the SMF to issue PDR and FAR rules to the UPF. The UPF can identify different application data flows from the UE or 5G gateway using various methods, including IP 5-tuple filtering, QoS Flow identification, and Deep Packet Inspection techniques. By leveraging UL CL and multiple homing technologies, the UPF ensures that traffic from client applications is correctly distributed to the corresponding MEC applications. Additionally, the 5GC QoS framework enables dynamic traffic prioritization, ensuring that different MEC applications meet their respective Service Level Agreements while optimizing network performance.

The MEC Orchestrator continuously monitors the status of computing resources, ensuring that applications can seamlessly adapt to changing conditions while maximizing overall resource utilization efficiency. By continuously interacting with the MEC Platform Manager, the MEC Orchestrator can retrieve real-time information on MEC platform status, available resources. This enables intelligent decision-making, ensuring MEC applications are deployed in suitable MEC hosts.

Fluctuations in client application workloads can trigger dynamic reallocation of computing resources, directly impacting UPF traffic routing strategies. The lifecycle of MEC applications plays a crucial role in computing resource management. The MEC Orchestrator supports efficiently handling of application onboarding, termination, and migration events, along with updates of the PDU session rules of UPFs. This ensures that traffic is always directed to the appropriate MEC application, maintaining seamless service continuity and optimizing network performance.

5.4.2 Analysis

When considering the performance and effectiveness of MEC Application deployment with computing resource awareness, the following network information items can be described for the use case:

Table 5.4.2-1: Information items to support the use case

Information item	Description
MEC Host Resource Utilization	The usage percentage of CPU, GPU, and Memory resources on a single MEC host, measuring computing resource load.
Deployment Success Rate	The percentage of successful MEC application instance deployments to MEC hosts that meet performance requirements.
Deployment Latency	The time from application deployment request to actual deployment completion.
Traffic Steering Success Rate	The percentage of bidirectional client traffic (uplink & downlink) that is properly routed to/from the target MEC application instance in accordance with traffic steering rules.
SLA Compliance Rate	The percentage of data packets between the MEC client applications and the MEC application that meet the SLA requirements, including latency, throughput, and packet loss rate.

For the MEC application deployment with computing resource awareness use case, the capabilities of the MEC Orchestrator are essential. It can accurately sense the status and dynamic fluctuations of computing resources across edge networks, efficiently deploy MEC applications, and establish stable, adaptive communication paths between MEC client applications and MEC applications.
Among these information items, MEC Host Resource Utilization helps analyse resource usage on MEC hosts, ensuring optimal allocation while preventing both overload and underutilization. Deployment Success Rate and Deployment Latency evaluate the MEC Orchestrator's ability to dynamically perceive computing resources and its efficiency in orchestrating and scheduling MEC applications. Traffic Steering Success Rate assesses whether application traffic is successfully transmitted between the client and the MEC application instance according to the predefined traffic steering rules. SLA Compliance Rate measures network service quality, reflecting the availability and reliability of MEC applications.
By analysing these information items in real time, the MEC Orchestrator can enhance its dynamic resource awareness, optimize intelligent orchestration and scheduling, and improve its ability to manage and configure UPF traffic rules. This ensures efficient computing resource utilization, precise traffic routing, and high-quality network service, ultimately supporting the seamless and reliable operation of MEC applications.

5.5 Use case #5: Edge Computing Resource Defragmentation

5.5.1 Description

Edge computing resource fragmentation refers to a situation in MEC environments where some MEC hosts run only a small number of low-load MEC application instances. Each instance uses only a small amount of computing, storage, or network resources, resulting in an overall host resource utilization rate that is lower than expected. The core cause of this fragmentation phenomenon lies in the dynamic changes in service demands, such as large differences in application instance loads during different time periods.

Edge computing resource defragmentation addresses this issue by dynamically migrating or consolidating MEC application instances that are dispersed across multiple underutilized hosts. Through intelligent scheduling and resource reorganization, the system can vacate certain MEC hosts. These idle hosts can then be repurposed to run resource-intensive application instances or transitioned into a low-power sleep mode to reduce energy consumption.

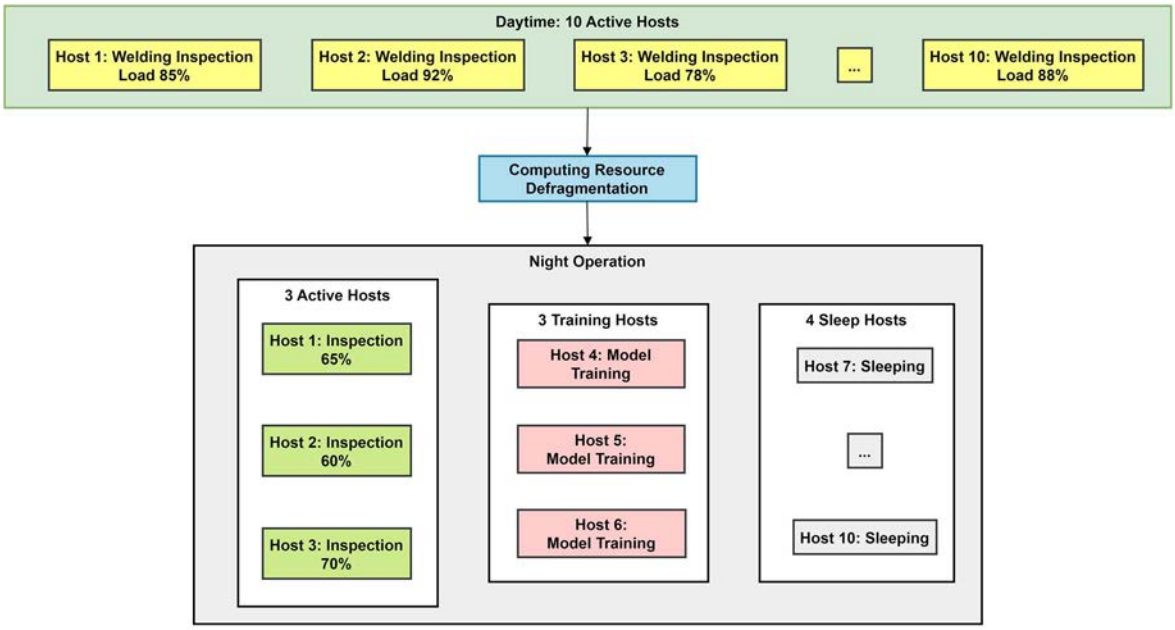

Figure 5.5.1-1: Smart Factory MEC Computing Resource Defragmentation

Figure 5.5.1-1 illustrates the process of edge computing resource defragmentation in action. The example is based on a smart factory scenario, where welding robots on an automotive production line require real-time visual quality inspection. This service relies on 10 MEC hosts to support the intensive AI inference workload. During peak daytime production, all MEC hosts and their deployed application instances operate at full capacity. However, during the night shift, the service load decreases significantly, resulting in some application instances running at low utilization or becoming idle. Through resource defragmentation, the system dynamically consolidates these low-load application instances, which were previously distributed across 10 MEC hosts, into just 3 hosts to improve resource efficiency. As a result, the remaining 7 MEC hosts can either enter a low-power sleep mode to reduce energy consumption or be repurposed to run compute-intensive tasks such as AI model training, thereby enhancing overall hardware utilization.

The key to achieving edge computing resource defragmentation lies in real-time monitoring and dynamic scheduling. Virtualization infrastructure resource utilization is tracked by the VIM, which collects detailed performance metrics such as CPU and memory usage, GPU utilization, and storage IOPS via the Mm6 interface within the virtual infrastructure. For client connection monitoring, the MEC platform adopts two complementary approaches:

- The passive approach relies on data plane traffic statistics to capture the number of active connections and traffic rates for each MEC application instance.
- The active approach involves MEC application instances periodically reporting client session data such as the number of client connections, TPS, and queue lengths through the Mp1 interface.

This information is aggregated by the MEPM and reported to the MEO via the Mm3 interface, enabling the construction of a global, real-time view of resource status across the edge infrastructure.

In the MEC environment, the MEO is responsible for determining whether defragmentation should be performed, typically through two mechanisms: periodic triggering and event-driven triggering:

- Periodic triggering involves assessing resource usage at predefined intervals and initiating defragmentation based on the observed degree of fragmentation.
- Event-driven (passive) triggering activates defragmentation in response to specific conditions, such as resource utilization exceeding predefined thresholds or significant delays in task processing.

When executing defragmentation, the MEO need to comprehensively evaluate factors such as task priority, resource demands, and MEC host load conditions to select suitable application instances for consolidation or resource reclamation. The defragmentation process typically involves three steps:

- Identifying application instances that can be merged or co-located.
- Migrating these instances to the fewer selected MEC host to optimize resource usage.

- Idle MEC hosts can either be transitioned into a low-power sleep mode, or allocated to execute newly scheduled or high-priority tasks.

This approach helps enhance overall resource utilization efficiency and ensures consistent service performance across the edge infrastructure.

Defragmentation may impact the data plane, particularly in terms of traffic routing policies. During the process, client requests might need to be redistributed across different MEC hosts to maintain efficient task execution. At the MEC host level, the MEC platform can issue updated traffic steering rules to the data plane via the Mp2 interface, ensuring that requests are directed to the correct application instances post-migration. When instance migration spans different UPF service areas, the MEO may need to coordinate with the 5G core and UPF to modify traffic forwarding rules, including adjusting PDU session anchor points, inserting uplink classifiers for path optimization, and reconfiguring QoS policies to align with the new service paths.

5.5.2 Analysis

When considering the performance and effectiveness of edge computing resource defragmentation, the following information items can be described for the use case:

Table 5.5.2-1: Information items to support the use case

Information item	Description
Resource Utilization	The overall usage of MEC host resources such as CPU, memory, and GPU.
Average Client Request Queue Time	The average time client requests spend waiting in queue for processing by MEC applications, reflecting service handling capacity and load balancing effectiveness.
Average Instance Migration Time	The average time required to migrate a single MEC application instance during defragmentation, affecting service continuity and scheduling efficiency.
Session Interruption Rate	The percentage of active sessions between Client Applications and MEC Applications (as Servers) that are interrupted due to redirection or disconnection, typically during processes such as MEC application instance migration.

NOTE: MEC Host Consolidation Ratio is for future study.

These information items are critical for evaluating the effectiveness and impact of edge computing resource defragmentation. By monitoring resource utilization, operators can assess how efficiently MEC hosts are being used before and after defragmentation. The average client request queue time provides insights into service responsiveness and helps identify performance bottlenecks. Average instance migration time reflects the efficiency of the defragmentation process and its potential impact on service continuity. Session interruption rate is essential for evaluating user experience and ensuring service reliability during application instance migration.

Together, these items support informed decision-making, help fine-tune scheduling strategies, and enable continuous optimization of MEC infrastructure operations.

6 Key issues and potential solutions

6.1 Key issue #1: MEC Application State Monitoring

6.1.1 Description

6.1.1.1 Introduction

In MEC system, the VIM is responsible for allocating, managing, and releasing virtualized resources (compute, storage and networking) within the virtualization infrastructure. The state of the virtualization infrastructure, which refers to the physical resource consumption involved in providing virtualized infrastructure, is primarily monitored and managed by the VIM.

A MEC application is instantiated and run as a virtualized software application, within for instance a VM or OS containers provided as part of the application package as a software image(s), on top of the virtualization infrastructure of the MEC system. The state of a MEC application reflects its operational state and serving capabilities. This includes, but is not limited to, the number of active sessions, requests per second, request latency, thread or connection pool usage, error rates, and request queue lengths.

Relying solely on metrics from the virtualization infrastructure (e.g. CPU and memory usage reported by the VIM) is insufficient to provide a comprehensive view of the actual operating conditions and processing stress of MEC applications. In MEC scenarios, the application's state is influenced not only by the availability of compute resources but also by factors such as the complexity of application logic, external dependencies, and service responsiveness. In some cases, a low computing resources usage may be observed even when the application is under heavy operational pressure. The reasons for such mismatches include:

- A sudden influx of user requests may not immediately saturate the CPU but can trigger task queuing and increased waiting times, ultimately leading to significant response delays.
- Internal threads may be blocked due to lock contention, disk I/O, or waiting for external services. While CPU usage remains low, the application's throughput is heavy.
- Some MEC applications with complex processing workflows (such as dynamic trajectory planning in vehicle-to-infrastructure collaboration, or multi-level scheduling in industrial control), despite having a light computational load, feature tight processing timing, numerous dependencies.
- VM or OS containers often impose strict resource limits. When an application reaches its quota and cannot acquire more resources despite growing demand, CPU usage may appear below threshold, while the system is actually overloaded.

Clause 5 discusses several representative use cases regarding exploiting computing resource, including dynamic scaling, redundant deployment, collaborative edge computing, and resource defragmentation. All four use cases heavily rely on effective monitoring of MEC application states:

- Dynamic scaling adjusts resource allocation flexibly based on periodic checks of MEC host resource utilization and health, in combination with application load changes.
- Redundant deployment depends on continuous monitoring of MEC application runtime status to enable fast failover and automated redeployment in the event of faults or performance degradation.
- Collaborative edge computing requires the coordinator to proactively or reactively collect performance metrics, runtime states, and fault information from MEC applications and platforms to support cross-MEC host scheduling, task collaboration, and failure recovery.
- Resource defragmentation builds a real-time global resource view by combining VIM-level monitoring of virtualized resource usage with MEC platform-collected client session data, enabling cross-host resource consolidation and dynamic optimization.

Therefore, comprehensive and timely awareness of MEC application state is a common and foundational capability for the successful implementation of these use cases.

6.1.1.2 MEC Application State Monitoring in MEC Architecture

In the implementation of MEC application state monitoring based on the MEC reference architecture, it is essential to integrate the core functional components and standardized interfaces defined in the architecture.

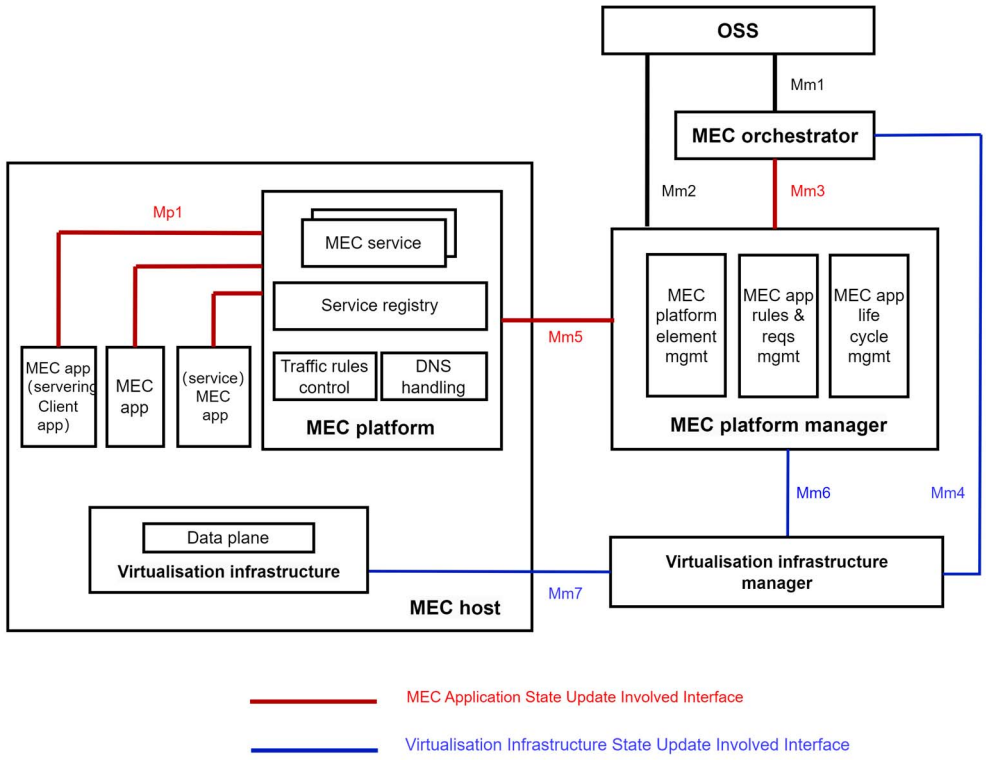

Figure 6.1.1.2-1: MEC Application State Monitoring in MEC Architecture

Figure 6.1.1.2-1 depicts the functional elements and information exchange flows required for MEC application state monitoring in the MEC reference architecture. Regarding MEC application state monitoring, the involved functional elements are as follows:

1) MEC Host Level:

   - MEC Platform (MEP): Responsible for actively collecting or passively receiving runtime state information from MEC applications (e.g. session count, response latency, error rate). It enables local application load awareness and status caching, and can control collection frequency and metrics of interest via configurable policies.

   - Virtualization Infrastructure Manager (VIM): Continuously monitors and manages virtualized resources (compute, storage, network), including resource allocation, utilization rates, and alarms. It ensures resource availability at the virtualization layer.

   - MEC Platform Manager (MEPM): Aggregates application status data from the MEP and resource usage information from the VIM to form a consolidated host-level load view.

2) MEC System Level:

   - MEC Orchestrator (MEO): Maintains a global view of MEC system resources. Based on the collected insights, the MEO can drive application scaling, redundancy deployment, fault recovery, and inter-host migration, thereby enabling system-wide resource optimization and high availability.

   - Operations Support System (OSS): Operates at the operations and management level, providing higher-level system monitoring and service assurance. The MEO may report aggregated status data to the OSS, or the OSS can directly retrieve the necessary monitoring data through standard interfaces with the MEO or MEPM, enabling automated operations and policy enforcement at the operational layer.

The State Update Information Flow Paths are as follows:

- Virtualized Infrastructure State Monitoring Path: The VIM collects and manages resource utilization via the Mm7 interface, and may report data to the MEPM via the Mm6 interface. In some implementations, it can also report directly to the MEO through the Mm4 interface for orchestration decisions.

- MEC Application State Monitoring Path: MEC applications interact with the MEP via the Mp1 interface, reporting their runtime status or exposing monitoring endpoints for data collection. The MEP sends the collected or received data to the MEPM via the Mm5 interface, and the MEPM forwards the aggregated monitoring data to the MEO via the Mm3 interface, forming a system-wide state view.

6.1.2 Solution proposal #1-1: MEC Application-Driven State Reporting

In this solution, the MEC application is responsible for actively reporting its operational status to the MEC Platform. The process begins with a configuration handshake, during which the MEP provides the application with parameters such as the reporting interval, metrics list, sampling frequency, and optional event-driven reporting triggers.

Once configured, the application pushes state reports to the MEP in two primary modes:

- Periodic Reporting: Status is sent at regular intervals, ensuring basic liveness and visibility.
- Event-Driven Reporting: In response to significant state changes or internal events, the application immediately pushes an update to the MEP, enabling fast reaction to critical conditions.

Additionally, liveness monitoring is embedded. If a MEC application fails to report for N consecutive intervals, the MEP raises an alert to the MEPM and MEO. This heartbeat mechanism improves fault detection even when the application is unable to push data due to internal failure.

MEC Application-Driven State Reporting offers high timeliness and event awareness by allowing applications to proactively send status updates, reducing the MEC platform's polling load; however, it increases application complexity by requiring them to manage metric collection and transmission.

Figure 6.1.2-1 shows the message flow for MEC Application-Driven state reporting.

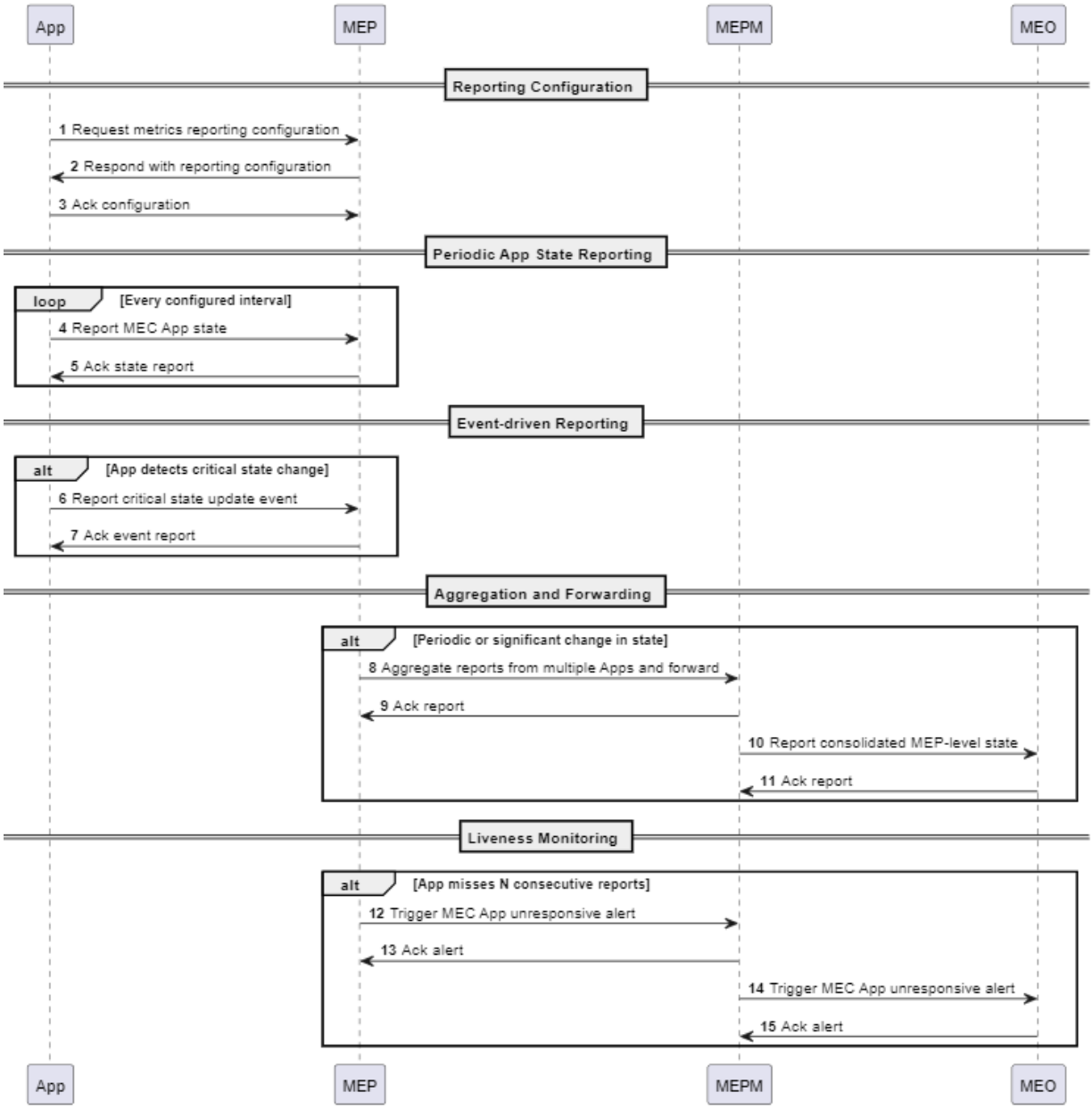

Figure 6.1.2-1: MEC Application-Driven State Reporting Flow

1) The MEC application initiates the setup by sending a metric reporting configuration request to the MEP. This request indicates the application's readiness to start reporting metrics.

2) The MEP responds with a full configuration profile. This includes:

   - **Metrics list:** the expected data fields and formats.
   - **Endpoint:** the target address to which state data should be pushed, such as a REST API.
   - **Sampling interval:** how often the app samples internal state metrics.
   - **Max interval:** how often at minimum the app should report, even if nothing changed.
   - **Event triggers:** definitions of which internal events require immediate reporting (e.g. crash, threshold breach).

3) The MEC application sends to the MEP an acknowledgment to the reporting configuration.

4) At each configured interval, the MEC application pushes its collected status to the MEP. This includes the number of active sessions, requests per second, request latency, thread or connection pool usage, error rates, and request queue lengths. This keeps the system updated even without major state changes.

5) The MEP sends to the MEC application an acknowledgment to the state report.

6) When a significant internal event occurs (such as an error, state transition, or threshold crossing), the MEC application immediately sends a state report to the MEP, bypassing the periodic schedule. This improves responsiveness for operations like alerting or autoscaling.

7) The MEP sends to the MEC application an acknowledgment to the critical state update event report.

8) Upon receiving either a periodic or event-driven report from any MEC application, the MEP aggregates data across all MEC applications under its management and forwards a consolidated view to the MEPM. This aggregation reduces message overhead and centralizes processing.

9) The MEPM sends an acknowledgement to the MEP upon receiving the aggregated state/event reports from MEC applications.

10) The MEPM receives reports from multiple MEPs, aggregates them, and forwards a comprehensive MEP-level applications status to the MEO. The MEO thus maintains a full real-time view of the MEC system, which is useful for orchestration, scaling, and policy enforcement.

11) The MEO sends an acknowledgement to the MEPM upon receiving the MEP-level state/event reports from MEC applications.

12) If a MEC application misses N consecutive reporting cycles (no periodic nor event-driven messages), the MEP identifies it as unresponsive and sends an alert to the MEPM, flagging a potential application failure or network issue.

13) The MEPM sends an alert acknowledgement to the MEP in response to the unresponsive MEC application notification.

14) The MEPM forwards this alert to the MEO, ensuring central visibility of application-level health issues. This enables orchestration logic (like auto-recovery, restart, or migration) to be triggered as necessary.

15) The MEO sends an alert acknowledgement to the MEPM in response to the unresponsive MEC application notification.

6.1.3 Solution proposal #1-2: MEC Platform-Driven Collection of Application State

In this solution, the MEP plays an active role in collecting the operational status of MEC applications. The MEP initiates a process to discover the application's available monitoring endpoints and supported metrics. The MEP periodically polls the application for its current state. This method ensures centralized control, standardized data formats. Aggregated metrics are forwarded to the MEPM and then to the MEO. The system also monitors liveness by detecting unresponsive applications when multiple polling attempts fail. This platform-driven method offers lower intrusion into application business logic, reduces development complexity, facilitates easier forward integration of features, and results in a lower load on the MEC application. However, it may introduce latency, increase MEC platform load, and miss short-term issues between polling intervals.

Figure 6.1.3-1 shows the message flow for MEC Platform-Driven collection of application state.

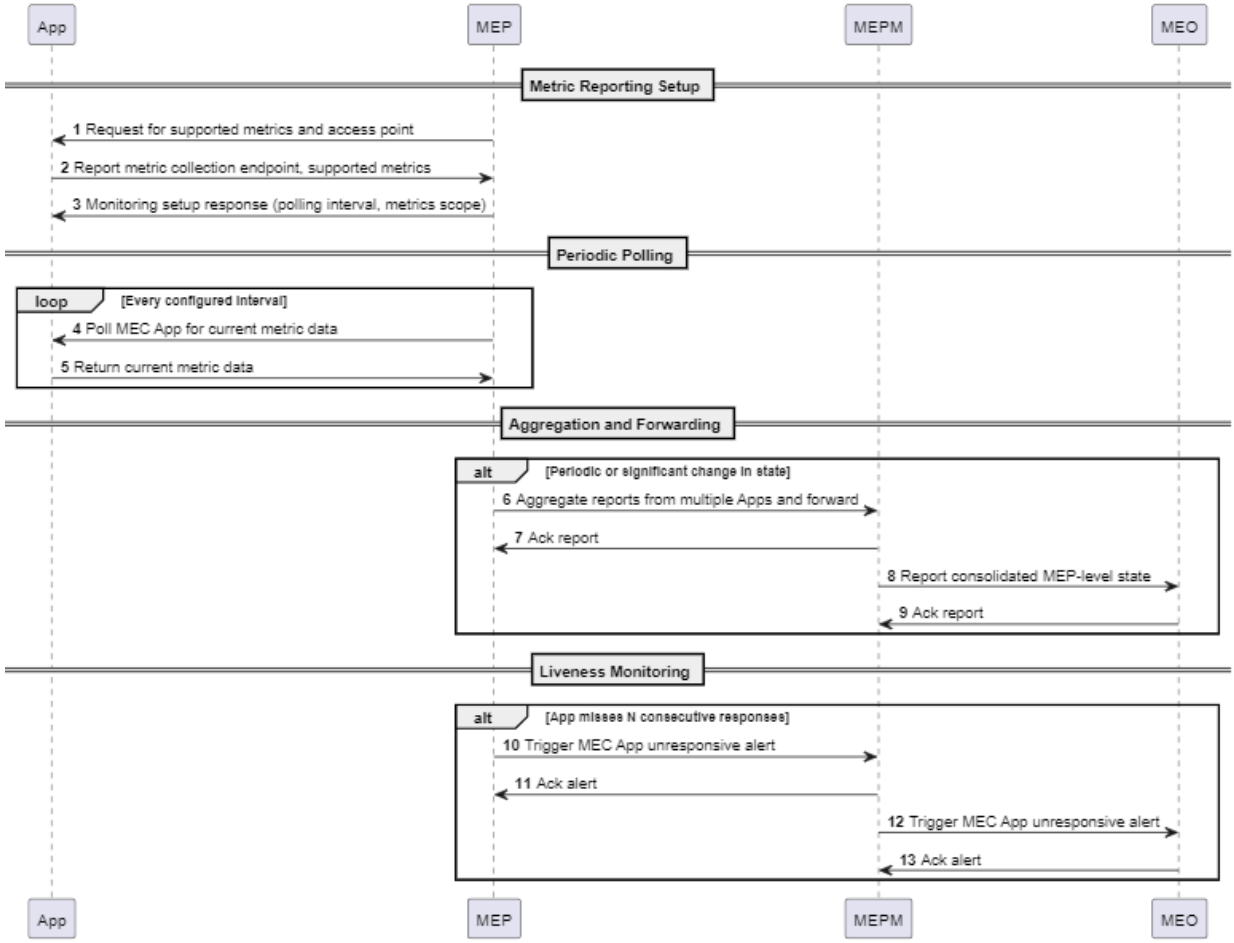

Figure 6.1.3-1: MEC Platform-Driven Collection of Application State Flow

1) MEP initiates a monitoring metrics inquiry to the MEC application, asking it to provide the accessible monitoring endpoint and a list of metrics it supports reporting. This is the initial discovery phase that enables the MEP to understand how to interact with the application for state retrieval.

2) The MEC application responds to the MEP, supplying its metric collection endpoint and the list of available application metrics (e.g. active sessions, error rate, request latency). This helps the platform determine what kinds of operational data can be queried.

3) MEP sends a monitoring setup confirmation back to the MEC application, confirming the polling parameters such as how often the MEP will collect data (polling interval) and which metrics it will request. This sets the monitoring behaviour between the platform and the application.

4) At every polling interval, the MEP actively sends a request to the MEC application to fetch its current metric data. This request includes the required metric scope as defined earlier.

5) The MEC Application responds to the polling request, returning the latest values of the requested metrics. This enable the MEP to track how the application is operating in real time.

NOTE: The processing procedures for steps 6) to 13) are the same as those for steps 8) to 15) in clause 6.1.2.

6.1.4 Evaluation

The two solution proposals are technically feasible under the following conditions:

- MEC applications are capable of self-monitoring, including collecting and formatting relevant runtime metrics.
- Standardized data models for application-level metrics are required to normalize the format and semantics of the reported state data, ensuring consistency and interoperability across MEC applications from different vendors.

Especially, the solution proposal "MEC Platform-Driven Collection of Application State" may present limitations in the following cases:

- The polling-based mechanism may increase MEC platform-side load and introduce latency in high-frequency state changes.
- The MEC platform may face challenges in detecting transient or fine-grained events between polling intervals.
- Not all applications would support to the mechanism to respond to MEP.
- The above limitations could be instead mitigated in case of considering other solutions, e.g. clause 6.1.2.

6.2 Key issue #2: Dynamic MEC Application Relocation and Discovery

6.2.1 Description

The use cases described in clause 5 illustrate several scenarios in which the deployment location and operational state of MEC applications may change dynamically. Such changes may result in relocation of MEC applications and modifications to the IP address:

- Dynamic scaling may lead to multiple MEC Application instances being created to support the same service, with each instance serving a subset of Client Applications. Scaling up requires deploying new MEC Application instances, while scaling down involves migrating Client Applications to remaining active MEC Application instances. This results in changes to instance associations and potential relocation.
- Computing resource defragmentation consolidates MEC Application instances from multiple hosts onto fewer hosts in order to free up MEC hosts for other purposes. This leads to MEC Application instance changes and redeployment, affecting the connection between Client Applications and MEC Applications.
- Failure recovery, where a faulty MEC Application is detected and automatically redeployed by the MEC platform, may lead to instance changes and disruption of ongoing sessions with Client Applications.

In a MEC system, the device application can request the on-boarding, instantiation, termination, and when supported, relocation of MEC applications via the User Application Lifecycle Management Proxy (UALCMP). The Device Application can receive notification events from the UALCMP in order to monitor the state of MEC applications. Consequently, the Device Application need to maintain a continuous connection with the MEC system-level components.

The current MEC reference architecture does not explicitly specify that a Client Application is deployed together with the Device Application. However, the separation of the Device Application and the Client Application enhance the flexibility of the MEC system. With this separation, the Client Application is not required to be specifically adapted to the MEC environment. The Client Application is agnostic to the deployment location of the Server Application, whether it resides in the cloud or at the edge. It is also agnostic to the MEC Application lifecycle, consuming the MEC Application solely as a remote service and requiring only the service API URI for invocation.

The key issue is that appropriate mechanisms are required to dynamically establish and maintain the communication path between the Client Application and the MEC Application in a manner that is transparent to the Client Application and preserves service continuity. This requires:

- Prompt acquisition by the 5G system of MEC Application deployment information.
- User plane re-establishment that takes into account both the Clients' location and the MEC Application's location.
- Clearly defined triggering mechanisms for initiating user plane re-establishment.

The potential solution is to leverage the 3GPP Edge Application Server Discovery Function (EASDF). During PDU Session Establishment or Modification, the SMF delivers DNS server configuration to the UE via PCO. The configured DNS server may be the EASDF, and the UE will send its DNS queries to the EASDF. According to ETSI TS 123 548 [i.4], clause 5.1, the EASDF handles DNS messages, including the following:

- Receiving DNS message handling rules and/or BaselineDNSPattern from the SMF.
- Exchanging DNS messages from the UE.
- Forwarding DNS messages to C-DNS or L-DNS for DNS Query.
- Adding EDNS Client Subnet (ECS) option into DNS Query for an FQDN.
- Reporting to the SMF the information related to the received DNS messages.
- Buffering/Discarding DNS messages from the UE or DNS Server.
- Providing a DNS response with a specific IP address to a DNS query.
- Constructing and sending DNS Query messages with specific FQDN and ECS option.

The EASDF is responsible for processing UE DNS messages based on DNS handling rules or a BaselineDNSPattern provisioned by the SMF. In this context, DNS queries initiated by the UE may be directed to the EASDF, which can subsequently notify the SMF. Upon receiving such notification, the SMF may initiate session re-establishment procedures, e.g. the insertion of an UL-CL/BP, or provide updated URSP rules to the UE. At the same time, the UE obtains the latest MEC Application destination address through the DNS response, ensuring that the user plane path and service invocation remain aligned.

6.2.2 Solution proposal #2-1: Exposure of MEC Application Deployment Information to EASDF

Figure 6.2.2-1 copied from ETSI TS 123 548 [i.4], Figure 6.2.3.4.2-1, illustrates the procedure in which the AF provides non-PDU Session specific EAS Deployment Information to the 5GC.

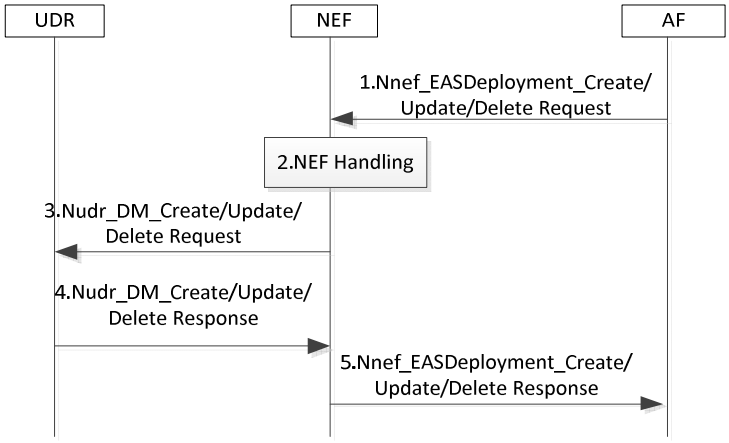

Figure 6.2.2-1: EAS Deployment Information management in the AF procedure

According to ETSI TS 123 548 [i.4], clause 6.2.3.4, the AF provides the EAS Deployment Information to the NEF, which stores it in the UDR. The SMF can then obtain this information by subscribing to the NEF, and use it as the basis to instruct the EASDF on DNS query handling.

In the MEC system, a MEC Application may appear as an EAS, while the MEC orchestrator can act as an AF towards the 5G system. The MEC platform may also configure a local DNS server. When a MEC Application is relocated or its IP address changes, the updated Deployment Information needs to be reported to the EASDF.

Based on the EAS Deployment Information, the SMF configures the EASDF with DNS message handling rules and/or BaselineDNSPattern. The BaselineDNSPattern and DNS message handling rules together define how the EASDF processes DNS messages from UEs:

- A BaselineDNSPattern contains:
  - Detection templates specifying the DNS message type, FQDN ranges, and/or EAS IP addresses.
  - Handling actions specifying ECS options, local DNS server IPs, or other processing parameters.
- DNS message handling rules may reference one or more BaselineDNSPattern items or define additional message-specific templates and actions.

In essence, BaselineDNSPattern provides a reusable template set, while DNS message handling rules apply these templates and actions dynamically to DNS messages, ensuring correct processing, forwarding, or reporting as instructed by the SMF.

ETSI TS 123 548 [i.4], clause 6.2 introduces the procedure for EAS discovery procedure with EASDF. When the UE sends a DNS Query to the EASDF, the EASDF checks it against the configured BaselineDNSPattern. If reporting is required, the EASDF notifies the SMF, which may update DNS handling rules. The EASDF then processes the query by interacting with the Central DNS server or Local DNS server and, if the DNS Response also matches reporting rules, it reports to the SMF and buffers the response. Based on the reported EAS information, the SMF may trigger UL-CL/BP insertion for optimized routing to the EAS, and finally instruct the EASDF to release the DNS Response to the UE. As a result, the Client Application can successfully access to the MEC Application, preserving service continuity.

6.2.3 Evaluation

The solution proposal is technically feasible under the following conditions:

- When access fails, the Client Application supports retrying DNS resolution.
- The MEC Orchestrator exposes all active MEC Application instances deployment information belonging to the same Application Package to the 5GC/NEF, allowing the SMF in coordination with the EASDF to select the optimal instance (e.g. based on UE location or operator policy).

6.3 Key issue #3: Computing Coordination and Management for Cooperative MEC Applications

6.3.1 Description

According to clause 5 of ETSI GS MEC 010-2 [i.5], the lifecycle management process of a MEC Application instance including Instantiate, Operate, and Terminate follows a defined message flow. This process starts with the OSS or the MEO through a granting procedure and continues with coordination among multiple entities such as the MEPM, MEP, and VIM, thereby completing the lifecycle management of the MEC Application.

Clause 5.2 of ETSI GS MEC 011 [i.6] describes how MEC Applications and/or MEC Services are supported by the MEC Platform. This includes mechanisms for Application start-up, stop, configuration management, and event notification, establishing a platform-centric control framework for application runtime management.

In scenarios such as dynamic scaling, redundant deployment, and collaborative edge computing (as discussed in clause 5), a group of interdependent MEC Applications often need to operate as a coordinated unit rather than as isolated instances. These application groups require continuous awareness of each member's runtime state, enabling autonomous lifecycle actions such as scaling, migration, synchronization, and recovery based on real-time performance and availability conditions.

However, the current MEC reference architecture exhibits the following limitations:

- The lifecycle management of MEC Applications, including instantiation, operation, and termination, is entirely driven by explicit operations initiated by the OSS or MEO through predefined workflows defined in ETSI GS MEC 010-2 [i.5]. The MEO primarily acts upon user or OSS requests and has limited capability to dynamically adjust deployments based on the runtime information performance or operational conditions of MEC Application instances.
- Since MEC Applications have no direct reference points to the MEO or OSS, they are unable to trigger lifecycle management procedures for related or dependent MEC Application instances. Even when a MEC Application detects significant changes in the state of an interdependent peer (e.g. performance degradation, malfunction, or failure), it is unable to proactively request peer-related scaling, role switching, or recovery actions.

To achieve group-level coordination, including peer monitoring, state synchronization, role switching, performance optimization, and fault recovery, a specific functional module is introduced to act as the runtime coordination and control plane among MEC Application instances.

6.3.2 Solution proposal #3-1: Computing Coordination and Management Function

6.3.2.1 Introduction

To overcome the limitations of the current MEC reference architecture, particularly in terms of flexibility, scalability, and autonomous coordination, a new functional module called **Computing Coordination and Management Function (CCMF)** is introduced.

The CCMF enables adaptive runtime coordination, autonomous fault recovery, and dynamic resource optimization across distributed MEC Application groups. Rather than replacing the MEC Orchestrator, the CCMF complements it by providing a computing-aware control plane. The MEO continues to handle lifecycle management as specified in ETSI GS MEC 010-2 [i.5], while the CCMF delivers fine-grained runtime coordination and management that is especially valuable for multi-instance, replicated, and collaborative MEC Application deployments.

As illustrated in Figure 6.3.2.1-1, the CCMF is integrated into the existing MEC reference architecture as a computing-aware control layer positioned between the OSS, the MEO, and the MEC Applications. It interacts with these entities through two reference points: Mc1 (CCMF–OSS), Mc2 (CCMF–MEO).

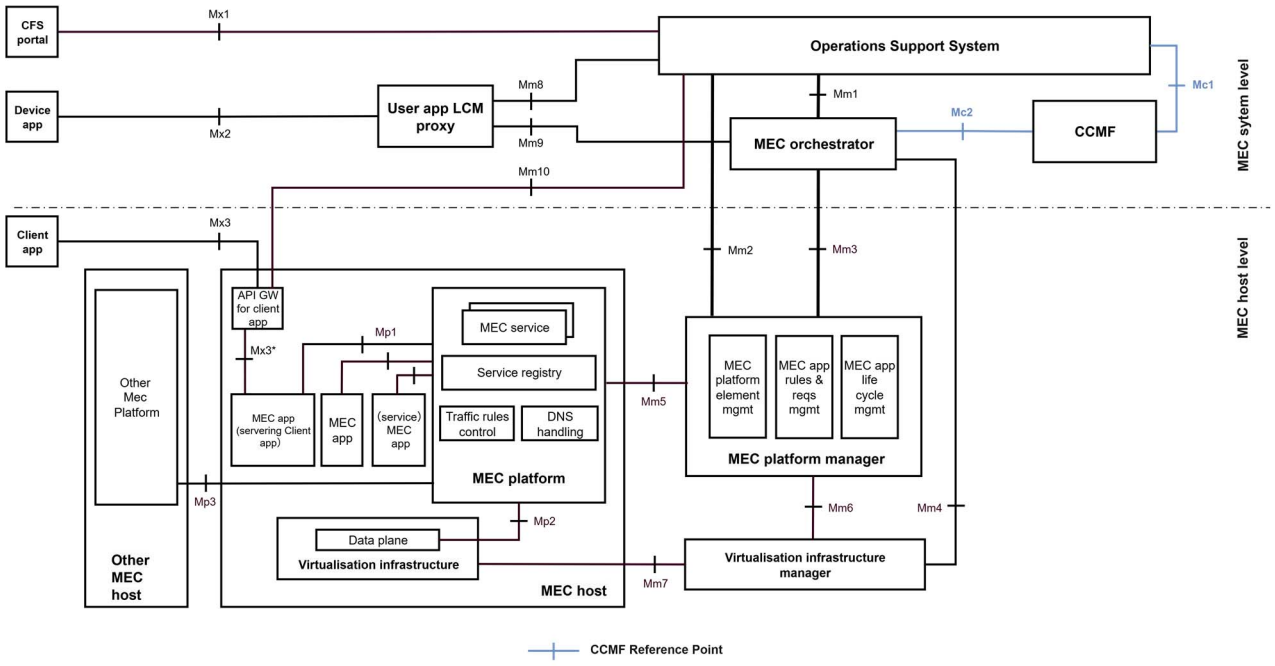

Figure 6.3.2.1-1: Extended MEC Reference Architecture with CCMF

The CCMF introduces a new layer of runtime computing management and coordination logic that operates in synergy with existing MEC management functions. Its responsibilities can be summarized across three key dimensions:

1) Dynamic Scaling:

   - Receive scaling policies, threshold definitions, and performance objectives from the OSS via Mc1.
   - Analyse runtime metrics (CPU, memory, latency, throughput) via Mc2, and request instantiation, termination, or migration of MEC Application instances as needed to implement scaling decisions.
   - Decide when to scale, which instance to extend or shrink, and how to balance load.
   - Trigger pre-warming and replication before scaling operations via Mc2, with the message originating from CCMF and propagating through MEO, MEPM, and MEP before finally reaching the MEC application instances.

2) Redundant Deployment (Primary–Backup / Active-Active Replication):

   - Interact with the OSS via Mc1 to receive redundancy configuration, policy input, and failure recovery intent.
   - Handle liveness monitoring for application instances via Mc2, and request instantiation, termination, or migration as needed to maintain redundancy and availability.
   - Coordinate leader election, role switching, and state synchronization between replicas.
   - Trigger seamless state handover and continuous service availability vis Mc2, with the message originating from CCMF and propagating through MEO, MEPM, and MEP before finally reaching the MEC application instances.

NOTE 1: Seamless state handover is for future study.

3) Collaborative Computing:

   - Support distributed or cooperative workloads such as MapReduce, distributed training, and multiple instances inference by obtaining job configuration, scheduling intent, and collaboration policies from the OSS via Mc1.
   - Execute runtime fault detection, worker health assessment, and role rescheduling (e.g. coordinator/worker reassignment) using liveness monitoring, heartbeat collection, and cluster health probes provided via Mc2, and request instance-level orchestration actions such as instantiation, termination, or migration as necessary.
   - Perform resource-aware and topology-aware scheduling across distributed MEC Application instances, taking into account MEC host metrics (e.g. CPU/GPU type, memory capacity), network metrics (e.g. throughput, latency), and other affinity/anti-affinity constraints.
   - Coordinate runtime collaboration behaviours across MEC Application instances via Mc2, including leader election, role assignment, consistency verification, distributed task scheduling, job lifecycle management, and instance-level recovery or failover, to ensure correct operation of distributed workloads, with the message originating from CCMF and propagating through MEO, MEPM, and MEP before finally reaching the MEC application instances.

The CCMF uses standardized reference points to interface with the OSS, the MEO respectively. Table 6.3.2.1-1 provides a summary of the reference points that support the CCMF.

Table 6.3.2.1-1: Summary of CCMF Reference Points and Functional Roles

Interface	Peer Entities	Function Summary
Mc1	CCMF ↔ OSS	Used for association declaration and policy input. The OSS informs the CCMF of the relationships, dependencies, and policies among groups of MEC Application instances (e.g. replication groups or cluster memberships). The CCMF registers itself as a service module available for multiple application groups.
Mc2	CCMF ↔ MEO	Used for telemetry access, granting procedures, and runtime orchestration requests. The CCMF obtains information from the MEO regarding application deployment status, inter-application dependencies, and performance metrics, and may request the MEO to instantiate, terminate, or migrate MEC Application instances in accordance with approved policies.

NOTE 2: The CCMF functionality is optional and is activated only when a group of interdependent MEC Applications requires runtime coordination.

6.3.2.2 Coordination Configuration and Runtime Information Exchange

Figure 6.3.2.2-1 illustrates the flow where the OSS interacts with the MEO and the CCMF to provide coordination-related configuration and to enable the CCMF to obtain runtime status, deployment, and performance information for a group of MEC Application instances. This procedure supports lifecycle management, policy provisioning, and runtime contextualization for coordinated multi-instance or interdependent MEC Application deployments.

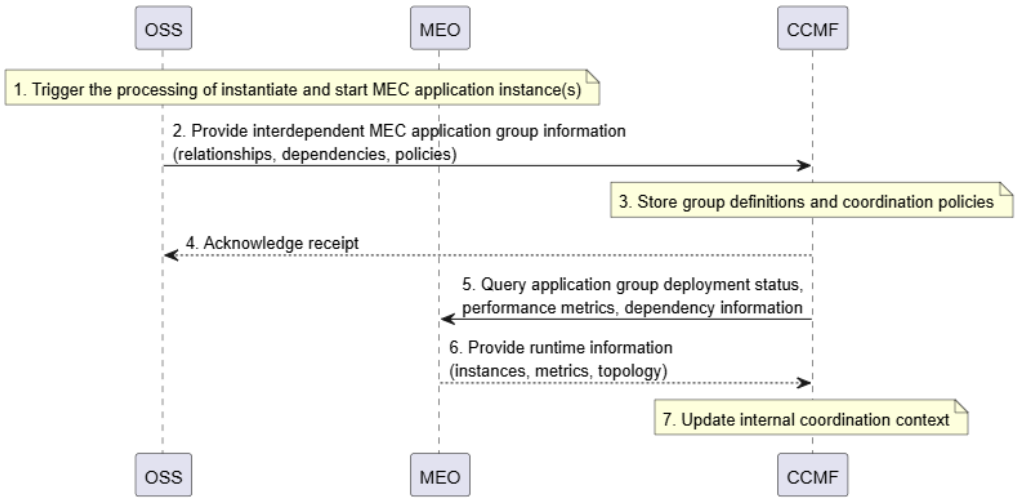

Figure 6.3.2.2-1: Coordination Configuration and Runtime Information Exchange Flow

The coordination configuration and runtime information exchange consists of the following steps:

1) The OSS triggers the processing required to instantiate and start a group of interdependent MEC application instances through the MEO.

NOTE 1: The above procedures reference Figure 5.3.1-1 and Figure 5.3.3-1 in ETSI GS MEC 010-2 [i.5], which illustrate the application instantiation flow and the application operation flow, respectively.

2) The OSS provides coordination configuration information to the CCMF. This includes group-level definitions of interdependent MEC Application instances, such as relationships, dependencies, replication groups, cluster memberships, and coordination policies.

3) The CCMF stores the received group definitions and coordination policies. The CCMF updates its internal policy and group membership database to support subsequent runtime coordination and management.

4) The CCMF acknowledges receipt of the configuration information to the OSS. This acknowledgment confirms that the CCMF has successfully registered the group configuration and is available to support coordination for the referenced application groups.

5) The CCMF queries the MEO for runtime information regarding the MEC Application group, including deployment status, performance metrics, and dependency-related information.

6) The MEO responds to the CCMF with runtime information, including instance lists, deployment locations, performance metrics, and application topology or dependency information relevant to runtime coordination.

NOTE 2: The MEO is capable of monitoring and perceiving the operational state of MEC Application instances. The procedure references clause 6.1, which specifies the procedures and interfaces for MEC Application State Monitoring

7) The CCMF updates its internal coordination context based on the received runtime information. This includes updating runtime state, resource availability, dependency graphs, and coordination logic in preparation for potential scaling, redundancy handling, or collaborative computing operations.

6.3.2.3 CCMF Event Subscription and Processing

Figure 6.3.2.3-1 illustrates the flow for CCMF event subscription and processing, showing how CCMF subscribes to group-level MEC application events via the MEO, receives event notifications, updates its coordination context, and determines the required lifecycle or operational actions based on the received events.

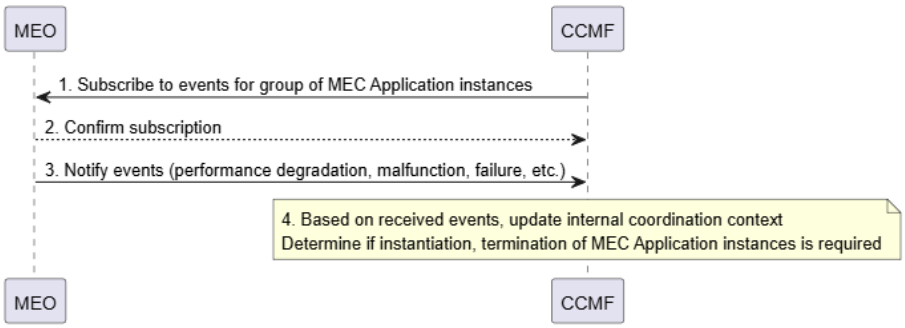

Figure 6.3.2.3-1: CCMF Event Subscription and Processing Flow

The CCMF event subscription and processing procedure consists of the following steps:

1) The CCMF subscribes to events for a group of MEC Application instances via the MEO. This subscription enables the CCMF to receive notifications regarding application performance, deployment status, and runtime events.

2) The MEO confirms the subscription to the CCMF. The confirmation indicates that the CCMF is now registered to receive relevant events for the specified MEC Application group.

3) The MEO notifies the CCMF of runtime events. Such events may include performance degradation, malfunctions, failures, or other state changes affecting MEC Application instances.

4) The CCMF updates its internal coordination context based on the received events. This includes analysing the event information to determine whether any orchestration actions, such as instantiation or termination of MEC Application instances, are required to maintain service availability, redundancy, and performance objectives.

6.3.2.4 CCMF Coordination Actions Execution

In clause 6.3.2.2, when CCMF detects the need to perform dynamic scaling, role switching, failover, or recovery of MEC application instances, it triggers the Coordination Actions Execution.

Figure 6.3.2.4-1 illustrates the flow for CCMF coordination actions execution, describing how CCMF triggers the instantiation, termination, and operational processing of MEC application instances via the MEO, and coordinates their runtime behaviours such as task scheduling, leader election, role switching, recovery, and failover.

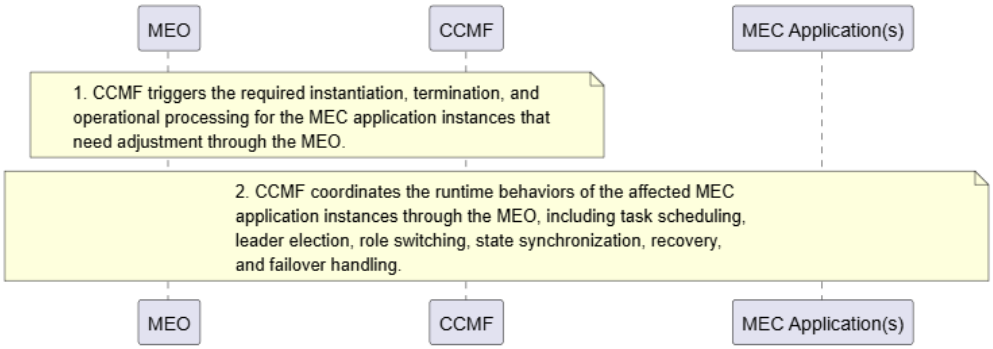

Figure 6.3.2.4-1: CCMF Coordination Actions Execution Flow

The CCMF coordination actions execution procedure consists of the following steps:

1) CCMF triggers the instantiation, termination, and operational processing of the specific MEC application instances within the group that CCMF determines require adjustment, and executes the necessary lifecycle actions through the MEO.

2) CCMF coordinates the runtime behaviours of the identified MEC application instances and any related instances affected by the required adjustments through the MEO. The coordination actions include distributed task scheduling, leader election, role switching, state synchronization, recovery procedures, and failover handling.

6.3.3 Solution proposal #3-2: Topology-Aware Inter-MEC-Application Communication

In MEC scenarios, a group of interdependent MEC application instances deployed across different Data Networks (DNs) may require application-to-application communication. The logical communication topology among MEC applications (e.g. star, tree, mesh) is application-specific and may evolve dynamically at runtime. Existing MEC specifications primarily focus on the lifecycle management of individual MEC applications, whereas inter-application communication topology and its mapping to 5G user plane paths are not explicitly addressed.

This solution proposes the CCMF acts as an Application Function (AF) to enable topology-aware orchestration of inter-MEC-application communication. The CCMF:

- Maintains a real-time view of MEC application context and their logical communication topology via Mc1 and Mc2.
- Translates application-level topology requirements into traffic influence policies toward the 5GC.
- Acts as an AF toward the 5G Core, leveraging standardized interfaces and procedures, enabling the 5GC to dynamically establish and adapt UPF-to-UPF forwarding paths that reflect the collaborative topology of MEC applications.

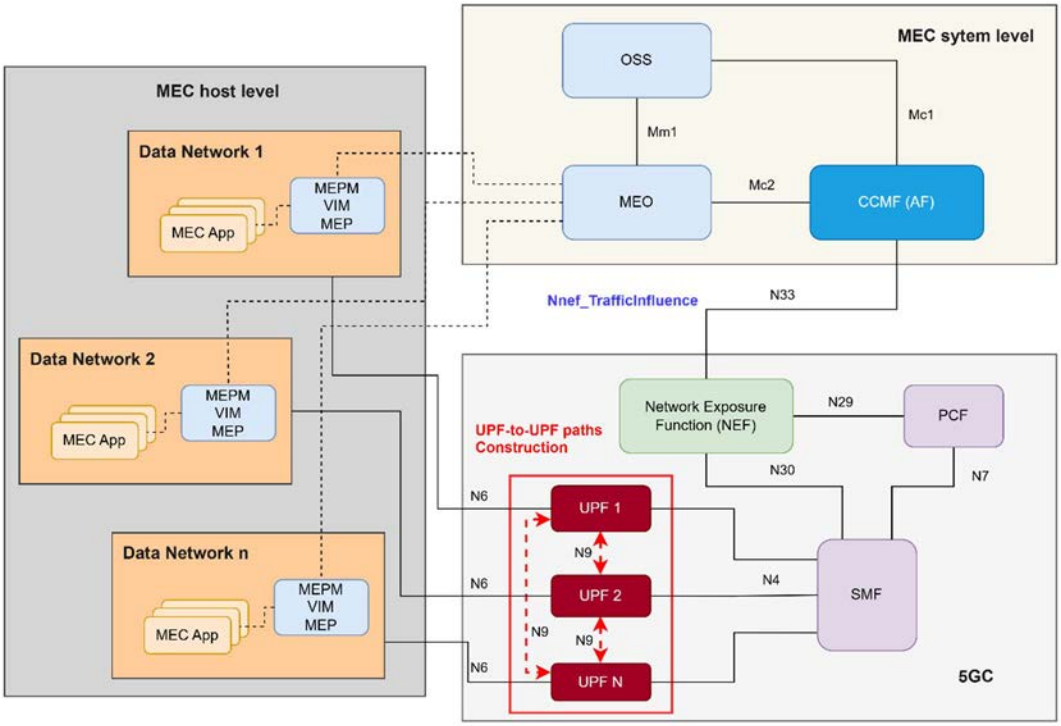

Figure 6.3.3-1: UPF-to-UPF Path Construction Assisted by CCMF (AF)

Figure 6.3.3-1 illustrates the MEC system-level architecture and the interactions between the CCMF, 5GC control functions, and multiple UPFs, enabling dynamic construction of UPF-to-UPF forwarding paths based on application-level traffic influence across multiple Data Networks supporting distributed MEC applications.

NOTE: While the MEO can also act as an AF, allowing the CCMF to act directly avoids unnecessary forwarding and reduces latency. The CCMF requires only the NEF service endpoint and authentication tokens, which may be shared with the MEO within the same trust domain.

As a MEC-domain component, the CCMF does not have direct visibility of 5GC user plane functions. Instead, it derives application-to-host and application-to-DN mappings based on MEC system information, and provides locality and connectivity constraints. To map application-level collaboration onto the 5G user plane, the CCMF:

- Identifies MEC application instances involved in collaborative execution.
- Obtains the corresponding MEC host, DN, DNAI, and logical correlation information for each application instance.
- Generates traffic influence policies that reflect application-level connectivity constraints.

Different logical communication topologies among MEC applications impose different requirements on application-to-application communication and corresponding user plane realization. Star topologies emphasize centralized traffic aggregation and stable UPF anchoring, tree topologies favour hierarchical and region-aware forwarding, while mesh topologies require decentralized and flexible UPF-to-UPF connectivity.

To realize topology-aware inter-MEC-application communication, the CCMF translates application-level topology characteristics into traffic influence policies as defined in clause 5.2.6.7 of ETSI TS 123 502 [i.7]. Depending on whether a star, tree, or mesh topology is applied, the CCMF provides different combinations of traffic identification, logical correlation, routing constraints (e.g. DN, DNAI, SFC indications), and routing preferences (e.g. locality, latency, relocation support).

Clause 4.3.6.2 of ETSI TS 123 502 [i.7] specifies the processing of AF requests to influence traffic routing and/or service function chaining for sessions not identified by a UE address. This procedure enables the 5G Core to apply application-specific traffic handling to UE-independent sessions, such as network-internal or application-to-application traffic. Acting as an AF, the CCMF expresses application-specific traffic influence policies via the NEF. In the Nnef_TrafficInfluence service, it provides traffic identification information, user plane selection constraints and routing preferences. These policies enable the PCF and SMF to select appropriate UPFs, insert branching or chaining functions if needed, and construct UPF-to-UPF forwarding paths for UE-independent application-to-application communication.

6.3.4 Evaluation

The solution proposal #3-2 is technically feasible under the following conditions:

- The CCMF provides traffic identification, logical correlation information, and routing constraints via Traffic Influence Policies, without selecting UPFs or constructing user plane paths, and the 5G Core Network supports the application of these policies to establish UPF-to-UPF forwarding paths that reflect application-level communication topology requirements.
- The CCMF acts as the core entity for runtime coordination of interdependent MEC Application instance groups. It receives objectives, policies, and configuration inputs from the OSS via Mc1, and obtains instance-level dynamics and execution status from the MEO via Mc2. It maintains real-time topology and dependency awareness, determines scaling and redundancy deployment actions, coordinates replica roles and state synchronization, and provides topology-aware communication support across distributed MEC Application instances.

6.4 Key issue #4: Computing Resource Awareness and Management

6.4.1 Description

This key issue describes a standardized architectural capability in the MEC system for global computing resource awareness and coordinated computing resource management across distributed edge environments.

As MEC evolves from hosting normal MEC applications to supporting AI-intensive, GPU-accelerated, and latency-sensitive workloads, computing resources at the edge are becoming increasingly heterogeneous and dynamic. MEC hosts may differ significantly in CPU capacity, memory size, accelerator availability (e.g. GPU, NPU, FPGA), storage performance, and power characteristics. Moreover, these resources are often geographically distributed across multiple edge network regions, connected to different UPFs and serving different traffic domains.

In current MEC deployments, computing resource management is largely confined to local infrastructure domains, typically handled by the VIM or platform-specific mechanisms. While the MEC Orchestrator (MEO) can trigger application lifecycle operations, it lacks native, system-wide awareness of real-time computing resource status and does not provide standardized mechanisms to optimize resource usage across multiple MEC hosts and regions. This limitation becomes increasingly problematic in scenarios where:

- MEC applications have explicit and diverse computing requirements (e.g. GPU-based AI inference or training) that may span cloud, edge, and device-side execution environments.
- Application workloads fluctuate significantly over time.
- Multiple MEC applications collaborate across different edge locations.
- Energy efficiency and hardware utilization are critical operational objectives.

Without an architectural solution, computing resource allocation decisions may be suboptimal, leading to inefficient application placement, uneven resource utilization, and reduced service quality.

6.4.2 Solution proposal #4-1: Computing Resource Awareness and Management within the MEO

The proposed solution introduces a Computing Resource Awareness and Management feature within the MEC architecture to provide a global, real-time view of heterogeneous computing resources and support resource-aware application placement and resource defragmentation. With these capabilities, MEC can efficiently handle AI-intensive workloads, collaborative edge applications, and energy-efficient operations without affecting existing application lifecycle or service exposure mechanisms.

At the foundation of the proposed framework is a standardized capability within the MEO to maintain system-wide computing resource awareness across distributed MEC environments. This capability is implemented through a **Computing Resource Awareness and Management (CRAM) module within the MEO**.

- The CRAM maintains a consolidated view of computing resources available at MEC hosts, including CPU, memory, storage, accelerator resources (e.g. GPU), network capacity, and power or energy state.
- Runtime computing resource information is collected from the MEC platforms, VIM and MEPM through existing management interfaces (e.g. Mm3, Mm4, Mm5).

Building on global resource awareness, the CRAM enables resource-aware decision-making during MEC application instances deployment and scaling. The key functions include:

- Computing resource capability exposure may leverage the NBI defined in the GSMA Operator Platform architecture, which can be aligned with the Mx1 and Mx2 reference points, as described in clause C.3 of ETSI GS MEC 003 [i.10], to enable operation across cloud–network–edge–device architectures, including support for cooperative execution models.
- When a MEC application is instantiated or scaled, its computing resource requirements (e.g. minimum CPU capacity, memory footprint, accelerator dependency such as GPU, and latency sensitivity) are supported and evaluated against the runtime resource availability across MEC hosts. The CRAM module provides resource suitability information to the MEO's application placement and scaling logic, enabling the MEO to select an appropriate MEC host or region for application instantiation.
- The CRAM continuously analyses resource utilization patterns across MEC hosts to detect underutilized or fragmented resource states. Based on predefined policies or thresholds, the framework can recommend or trigger defragmentation actions by identifying and consolidating low-load application instances onto fewer MEC hosts. This enables idle or lightly loaded MEC hosts to be reclaimed for high-priority workloads or transitioned into low-power states, thereby improving resource utilization and energy efficiency.
- The CRAM provides input to the MEO to support coordination of traffic steering and routing updates with underlying network functions when application instances are relocated.

6.4.3 Evaluation

The solution proposal is technically feasible under the following conditions:

- The Mx1 and Mx2 reference points support the exposure of computing resource capabilities and energy consumption information, enabling northbound access to information related to available computing resources at MEC hosts.
- The Mm3, Mm4, and Mm5 reference points support the collection of computing resource information from VIM and MEC platforms, including both static resource capabilities (e.g. CPU, memory, accelerator resources such as GPU) and runtime computing resource information (e.g. utilization, load, and availability).

7 Gap analysis and recommendations

The mapping of the key issues, identified in clause 6, to their associated solutions is provided in Table 7-1. This includes highlighting any identified gaps and external dependencies.

Table 7-1: Key issue and solution evaluation

Key issues	Clause #	Solution	Gap	External dependency
#1: MEC Application State Monitoring	6.1	#1: MEC Application-Driven State Reporting	Yes, ETSI GS MEC 010-2 [i.5] ETSI GS MEC 011 [i.6]	No
		#2: MEC Platform-Driven Collection of Application State		No
#2: Dynamic MEC Application Relocation and Discovery	6.2	#1: Exposure of MEC Application Deployment Information to EASDF	No	ETSI TS 123 548 [i.4]
#3: Computing Coordination and Management for Cooperative MEC Applications	6.3	#1: Computing Coordination and Management Function	Yes ETSI GS MEC 003 [i.10]	No
		#2: Topology-Aware Inter-MEC-Application Communication	No	ETSI TS 123 501 [i.2] ETSI TS 123 502 [i.7]
#4: Computing Resource Awareness and Management	6.4	#1: Computing Resource Awareness and Management	Yes, ETSI GS MEC 003 [i.10], ETSI GS MEC 016 [i.8], ETSI GS MEC 048 [i.9]	No

Taking into account the gap analysis presented in Table 7-1, extensions to the MEC requirements, architecture, and certain reference points are identified as necessary to address the identified gaps. Accordingly, the following topics could be considered for further study within ETSI MEC:

- Requirements and possibly related use-cases may be added to ETSI GS MEC 002 [i.11].
- The reference points Mp1 (specified in ETSI GS MEC 011 [i.6]), and Mm3 (specified in ETSI GS MEC 010-2 [i.5]) are to be enhanced to support MEC application state monitoring.
- The MEO are to be enhanced to support the optional Computing Coordination and Management Function, as specified in ETSI GS MEC 003 [i.10].
- The MEO is to be enhanced to support computing resource awareness, exposure (via Mx1 and Mx2, as specified in ETSI GS MEC 016 [i.8] and ETSI GS MEC 048 [i.9]), and resource defragmentation capabilities, as specified in ETSI GS MEC 003 [i.10].

Annex A:
Change history

Date	Version	Information about changes
September 2024	V4.0.1	Initial skeleton proposal for the study
May 2025	V4.0.2	Contributions included: • MEC(24)000452r5_MEC059_Description_of_Dynamic_Scaling_of_MEC_Resources_for_UseCase1 • MEC(24)000453r4_MEC059_Analysis_of_MEC_resources_dynamic_scaling_for_UseCase1 • MEC(24)000469r3_MEC059_Description_of_MEC_computing_resources_redundant_deployment_for_UC2 • MEC(24)000470r4_MEC059_Analysis_of_MEC_computing_resources_redundant_deployment_for_UC2 • MEC(25)000008r2_MEC059_Description_of_Cross-MEC_Host_Collaborative_Edge_Computing_for_UC3 • MEC(25)000009r3_MEC059_Analysis_of_Cross-MEC_Host_Collaborative_Edge_Computing_for_UC3 • MEC(25)000107r1_MEC059_Description_of_MEC_application_deployment_with_computing_resource_awareness_for_UC4 • MEC(25)000108r1_MEC059_Analysis_of_MEC_application_deployment_with_computing_resource_awareness_for_UC4 • MEC(25)000176_MEC059_Description_of_Edge_Computing_Resource_Defragmentation_for_UC5 • MEC(25)000177r2_MEC059_Analysis_of_Edge_Computing_Resource_Defragmentation_for_UC5
December 2025	V4.0.3	Contributions included: • MEC(25)000225r1_MEC059_Clause_6_1_1_Gap_Analysis_of_MEC_Application_State_Monitoring • MEC(25)000226r1_MEC059_Clause_6_1_2_Solution_Proposal_1_of_MEC_Application_Driven_State_Reporting • MEC(25)000251r2_MEC059_Clause_6_1_3_Sol_2_MEC_Platform-Driven_Collection_of_Application_State • MEC(25)000252r1_MEC059_Clause_6_1_4_Evaluation_of_the_MEC_Application_State_Monitoring_Solutions • MEC(25)000339_Dynamic_MEC_Application_Relocation_and_Discovery
December 2025	V4.0.4	Contributions included: • MEC(25)000398r4_MEC059_Clause_6_3_Computing_Coordination_and_Management • MEC(25)000417r2_MEC059_Clause_6_3_3_CCMF_Sequence_Diagrams
February 2026	V4.0.5	Contributions included: • MEC(26)000003r2_MEC059_updated_Clause_6_3_3_Topology-Aware_Inter-MEC-Applica • MEC(26)000014_MEC059_Clause_6_4_Computing_Resource_Awareness_and_Managemen • MEC(26)000020_MEC059_Clarification_of_CCMF_Integration_Impacts
March 2026	V4.0.6	Contributions included: • MEC(26)000032r1_MEC059_updated_Clause_7_Gap_analysis_and_Recommendations • MEC(26)000033_MEC059_updated_Clause_4_Overview
March 2026	V4.0.7	Revised in accordance with edit help comments.
April 2026	V4.0.8	Clause 5.2.2 was accidentally moved and has been restored to its correct position.
April 2026	V4.0.9	Final draft similar to Stable draft V4.0.8 as no further comments were raised. This Final draft is ready to go to MEC RC for review.
May 2026	V4.0.10	Final draft updated following editorial comments raised during the MEC Remote Consensus (RC) review, including clarification and alignment of abbreviations. This Final draft is ready for MEC RC approval and ETSI publication.

History

Version	Date	Status
V4.1.1	June 2026	Publication