SN Computer Science (2026) 7:300
https://doi.org/10.1007/s42979-026-04869-x

ORIGINAL RESEARCH

CEC-Trust: Using Trust to Improve the QoS in Collaborative Edge Computing

Fabrizio Messina[1] · **Domenico Rosaci**[2] · **Giuseppe Sarnè**[3]

Received: 29 September 2025 / Accepted: 27 February 2026

Abstract
In the challenging and dynamic context of Edge Computing, we propose a distributed trust-based approach, which includes a trust model and a nodes pooling algorithm, both validated through simulation, where edge servers collaborate to manage task offloading requests. Our model analyzes the past behavior of edge domains by considering both the resources they have shared and requested from other domains, as well as the characteristics of the tasks involved. In addition, we characterize the trust relationships established between edge servers. As we explain in our work, such trust relationship is directly connected to the measured QoS of task offloading. These two aspects (i.e., historical behavior and trust) are then combined into a unified metric, termed Collaborative Edge Computing Trust (CEC-Trust), which serves to assess the benefits (in terms of QoS) of joining a pool of edge domains for collaborative task offloading. To this end, we designed a fully decentralized procedure, guided by the CEC-Trust measure to form pools of edge domains. A set of simulation results are reported and prove that, by the designed procedure and along with the exploitation of the CEC-Trust measure, the overall performance of the edge servers in the collaborative scenario are notably improved.

Keywords Edge computing · Trust · Internet of things · Mobile computing · QoS

Introduction

Edge Computing [1, 2] enables user devices (UDs) to leverage computing resources deployed at the network edge: hence, smartphones, tablets, laptops, and other devices can offload computationally intensive tasks-either partially or fully-to nearby edge servers, improving performance and reducing latency. In particular, the offloading mechanism enables UDs to address several constraints, such as energy consumption, limited on-board computational capacity, and task completion time. In other words, the partial "relocation" of Cloud Computing power to the edge [3] supports resource-demanding applications through localized execution. Consequently, the Edge Computing paradigm is now widely adopted in critical scenarios-such as smart cities and real-time systems-where offloading to centralized, remote servers would introduce prohibitive delays. Representative case studies include online gaming [4], video and image processing [5], augmented reality [6], and accelerated web browsing [7], among others.

The Collaborative Edge Computing (CEC) paradigm [8–11] extends the classical Edge Computing scenario by introducing collaboration among multiple edge nodes that cooperate to process tasks offloaded by the UDs. In this collaborative setting, additional requirements emerge to support UDs in executing complex tasks. As in the conventional Edge Computing scenario, resources located at the edge can be grouped into Edge Domains (ED nodes), which represent clusters of edge servers at the same location that share uniform offloading policies (e.g., costs, CPU-intensive tasks, delay-constrained tasks, latency-sensitive

✉ Fabrizio Messina
fabrizio.messina@unict.it

Domenico Rosaci
domenico.rosaci@unirc.it

Giuseppe Sarnè
giuseppe.sarne@unimib.it

[1] Department of Mathematic and Computer Science, University of Catania, Via S. Sofia 6, Catania 95126, CT, Italy

[2] DIES, University of Reggio Calabria, Loc. Feo di Vito, Reggio Di Calabria 89122, RC, Italy

[3] Department of Psychology, University of Milano-Bicocca, Milano, Italy

Published online: 24 March 2026

tasks) as well as similar computational characteristics (e.g., available processing power, GPU availability, or specialized software). For example, some ED nodes may be optimized for specific real-time applications such as online gaming, while others may specialize in computationally intensive or AI-driven workloads. To handle offloading requests that exceed the capabilities of a single ED node, CEC relies on advanced techniques such as resource allocation strategies and shared decision-making. In this context, where ED nodes collaborate to provide optimal task offloading for UDs, an important question arises: given an ED node, denoted ED_1, and another ED node, denoted ED_2, how can we objectively quantify the benefit that ED_1 gains by collaborating with ED_2 to support a complex or composite offloading task? Conversely, how can we determine whether it is more advantageous for ED_1 to avoid collaborating with ED_2? To address these questions, we reformulate the problem as "dynamically establishing collaboration links among ED nodes with the goal of optimizing the Quality of Service (QoS) delivered by a pool of ED nodes to end users". For convenience, we refer to this problem as the ED nodes pool formation, and our approach to its solution is described hereinafter.

Main Contributions

In this paper, we propose a distributed approach to assist ED nodes in automatically selecting an appropriate pool of ED nodes to join, and to help cluster administrators choose which ED nodes to accept into their own pools.

Our approach considers both past ED behaviors and their mutual trustworthiness. Specifically, we take into account *(i)* the historical behavior of each ED node in terms of resource sharing (i.e., resources offered and requested from peers) and *(ii)* the trust of the ED nodes, obtained by collecting mutual feedback after collaborations for task offloading. The trust model introduced in this work is designed to consider reliability and reputation criteria, weighted by the knowledge that each ED node has acquired about other ED nodes by previous interactions, as well as the freshness of this information. Furthermore, the proposed trust system is capable of identifying malicious behaviors, as demonstrated by the results of the discussed experiments. By combining trust and historical behavior, we derive a unified metric, named CEC-Trust (Collaborative Edge Computing–Trust), to quantify the potential benefit for a given ED node from joining with a particular pool and vice versa. At the same time, the CEC-trust measure is exploited by a simple and fully decentralized greedy procedure, called Pool Formation (PF), to associate ED nodes with pools with a bi-directional link. In other words, the PF procedure formulates pool association operating a matching problem, leveraging the CEC-Trust metric in a distributed manner.

The key contributions of this work are summarized as follows:

- A computational framework that captures the past behaviors of ED nodes to compute trust between pairs of ED nodes.
- A trust model tailored for a CEC context which, when combined with behavioral measures, yields the unified metric CEC-Trust.
- A decentralized algorithm that exploits the CEC-Trust metric to drive the formation of pools of ED nodes, promoting an asymptotic improvement of the overall system performance in terms of trust-aware Quality of Service, without relying on a centralized global optimization function.
- A set of experimental results allowed us to evaluate the *(i)* resilience of the trust system to malicious activities and the *(ii)* contribution of the decentralized PF procedure with respect to the composition of the pools of ED nodes. These last results have shown that, within a few iterations, the PF algorithm introduces a relevant increase of about 40% of the CEC-Trust measure of the pools compared to a purely random assignment of ED nodes to pools. Moreover, since the CEC-Trust measure also reflects the measured level of QoS of the collaborations among the Edge servers, the overall performance of the pools, in terms of QoS, are significantly improved.

Organization of the Manuscript

The remainder of this manuscript is organized as follows. Section "The Collaborative Edge Computing Scenario" presents the scenario of the CEC. In "Behavioral and Trust Measures in Pools of Edge Domains", we describe the behavioral and trust measures designed for the collaborative edge computing scenario. Section "Pool Formation (PF) Algorithm" details the design of the decentralized PF algorithm, which forms pools of ED nodes for collaborative task offloading. In "Experiments", we present and analyze the experimental results, while Sect. "Related Work" discusses related work. Finally, Sect. "Conclusion" provides our conclusions.

The Collaborative Edge Computing Scenario

In our scenario, an ED (Edge Domain) node consists of a set of computational resources deployed at the network edge. The reference context is depicted in Fig. 1, where EDs

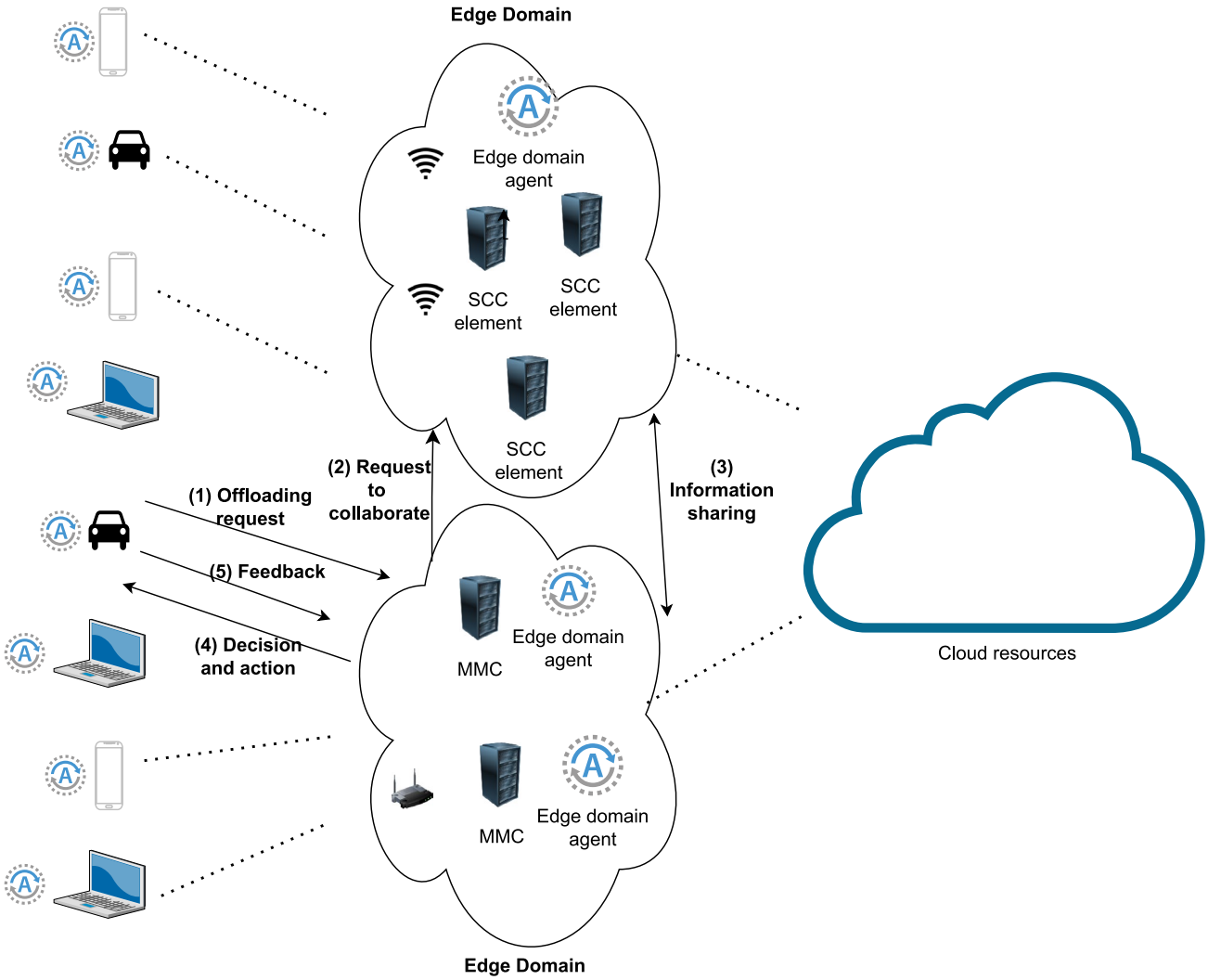

Fig. 1 Typical mobile collaborative edge computing scenario

are interconnected via various access points (e.g., mobile networks or Wi-Fi). In order to give a precise technical description of the scenario under consideration, we assume that an ED node can be represented as a Mobile Micro-Cloud [12] (MMC, shown in the lower part of Fig. 1), or as a group of resources overseen by a Small Cell Manager; these resources are identified as Small Cell Cloud (SCC) elements [13] in Fig. 1. For instance, the upper part of Fig. 1 shows an ED agent responsible for coordinating several SCCs-this software agent is logically linked to the small cell manager. The lower portion of the figure depicts an ED node agent acting on behalf of a single MMC, which functions as an independent pool of computing resources. As shown in Fig. 1, UDs (i.e., the corresponding agents) connect to nearby CEC resources through different access technologies, such as Wi-Fi networks and mobile operators. In this work, as illustrated in Fig. 1, we assume the existence of software agents, referred to as ED agent and UD agents, representing ED nodes and UDs, respectively.

Deployment Assumptions

The proposed framework relies on the following assumptions, which are consistent with typical Collaborative Edge Computing environments:

1. Each ED node is capable of locally monitoring task offloading outcomes and extracting feedback used for trust updates.
2. Edge resources and task requirements can be abstracted as quantitative resource units (e.g., CPU, memory, bandwidth), without loss of generality.
3. ED nodes operate in a decentralized manner, without relying on a global coordinator or centralized authority.

4. Interactions among ED nodes are local and bounded, as enforced by the Pool Formation algorithm through parameters such as N_{MAX} and K_{MAX}.
5. Trust and reputation information is exchanged only among a limited subset of peers, preventing global knowledge assumptions.
6. The system tolerates heterogeneous ED capabilities and dynamic workloads, including unreliable and malicious behaviors.
7. Network-level properties (e.g., latency or packet loss) are abstracted and reflected indirectly through task execution feedback.
8. Pools and trust relationships may evolve over time as ED nodes join or leave collaborative configurations.

An Example of Collaborative Task Offloading

Below we provide a brief case study of CEC, where task offloading is utilized within an Intelligent Transportation System (ITS) [14]. ITS offers a representative context in which a self-driving vehicle produces a very large amount of data from its numerous sensors. This information requires rapid processing to prevent accidents and, more generally, to make timely decisions regarding driving behavior and route planning. Task offloading becomes essential here because the vehicle can decide to delegate parts of this computation-such as visual data analysis or route calculation-to a nearby CEC server. If the primary CEC server associated with a given ED node is overloaded or unavailable, another CEC server belonging to a different ED node can take over part of the workload, or can be asked by the original ED node to assist with processing.

The main steps of this collaborative process are detailed below; the reader may also refer to the central portion of Fig. 1:

Data generation. A self-driving vehicle (e.g., a car) produces a substantial volume of information from its cameras and sensors, including video streams and environmental measurements.

Offloading request. Because the vehicle's onboard computational capacity is limited, it transfers demanding tasks-such as traffic analysis or hazard detection-to a nearby CEC server of a specific ED node, denoted as ED_1.

Decision to collaborate with other edge Servers. ED_1 attempts to allocate the resources required for task execution; if multiple vehicles generate excessive load, or for any other reason resources become insufficient, ED_1 will request support from a neighboring ED node, say ED_2, which belongs to a pool that ED_1 previously joined.

Request for collaboration. At this stage, ED_1 communicates with ED_2 to request assistance with the specific offloaded task. Some portions of the workload may then migrate to servers in ED_2 that have additional computational capacity.

Information sharing. The Edge Domain ED_1 may also share data on local traffic conditions or potential hazards with the servers of ED_2.

Decision and action. The collaborative CEC system, represented by the pool(s) that processed the overall task, can now take rapid and critical decisions-for example, initiating an emergency braking maneuver-and communicate this decision back to the vehicle through ED_1's server.

Feedback. Finally, ED_1, which was responsible for managing the entire offloading operation, collects detailed feedback from the vehicle to update its trust value (reliability) regarding ED_2.

This cooperative method represents a time-sensitive, real-time system that mitigates failures by distributing task offloading among multiple EDs. In this simple example, a key element is the choice of the collaborating Edge Domain ED_2, which must be based on relevant criteria (e.g., historical performance, resource characteristics, trust, and similar factors). In our work, this selection relies on mutual trust established with other ED nodes. As we will elaborate later, our proposal is designed to support such partner selection for collaborative task offloading.

We highlight that the proposed approach is particularly suited for realistic Collaborative Edge Computing scenarios such as smart cities, IoT-enabled intelligent transportation systems, and large-scale mobile edge infrastructures. In these contexts, edge domains exhibit heterogeneous resource capabilities and dynamic collaboration patterns, while centralized coordination is often impractical. The abstraction adopted in this work allows CEC-Trust to be applied across different deployment scenarios without relying on application-specific assumptions. Concrete instantiations may map resource units to CPU cycles, memory, or bandwidth, and task offloading interactions to application-level services, such as traffic analysis, video processing, or sensor data aggregation.

Pool Repository (PR) Implementation and Protection

A practical real instantiation of the solution presented in this work may use the Pool Repository (PR), defined as a logical directory service that stores pool descriptors (e.g., pool identifier, member list summary, and metadata) and supports discovery of candidate pools. In practical deployments, PR can be implemented as a replicated and distributed directory (e.g., a DHT-based lookup service or a federated registry across edge domains), thereby avoiding a single point of failure. To reduce the PR attack surface, pool registration and updates should be authenticated and access-controlled.

Rate limiting and replication mitigate flooding and availability attacks, while integrity can be ensured by signed updates and cross-checking across replicas. These deployment choices preserve the decentralized nature of PF, as PR provides only discovery support and does not make allocation decisions.

Illustrative Scenario

Consider a collaborative pool p_j where most members currently have a consumption-heavy behavior, resulting in a pool profile PB_j that reflects high demand. Adding another ED node n_i with $BT_i \approx PB_j$ increases aggregate demand without increasing supply, which may reduce the probability of successful offloading for the pool. Conversely, selecting a node with complementary behavior (large $|BT_i - PB_j|$) provides additional resources that balance demand, improving collaborative effectiveness and stability of task execution.

Behavioral and Trust Measures in Pools of Edge Domains

Let us suppose having a collaborative CEC scenario with a certain number of ED nodes. We have introduced, in the previous section, the concept of pool, as a group of ED nodes that can interact in the process of collaborative tasks offloading. It is assumed that each ED node will provide a set of computational task offloading services and resources according to the main characteristics of the edge resources grouped in the specific ED node. Furthermore, we assume that decision processes aimed at joining to or leaving any pool are supported, on each ED node, by a software agent, let be a_i. Similarly, we can assume that the "reasoning capability" of each pool of ED nodes will be provided by a software agent [15].

As we explain later in this section, we designed a number of different numerical measures that are combined in a single measure named CEC-Trust: in "he Behavioral Measures" we discuss the behavioral measures, in "The Trust Measures" we illustrate the trust measure and, finally, in "The CEC-Trust Measure" we illustrate the rational of the CEC-trust measure, which is exploited by the PF algorithm presented in "The PF Procedure Performed by the Node Agents".

The Behavioral Measures

Each ED node is also characterized by a given number of User Devices (UD) submitting task offloading requests. The set of resources requested for the specific task offloading by the UDs of a specific ED node will be denoted as "required" resources. Vice-versa, offered resources are those "shared" by the specific ED node within a pool.

The rationale behind the behavioral measures is represented by the fact that ED nodes receiving specific task offload requests – characterized by resource requirements that are not completely satisfied by the ED node itself – are interested in joining certain pools. The same concept will hold for the pool of ED nodes that do not include ED nodes with certain specific capabilities in terms of resources, connectivity, and so on.

In order to model the aspect above, we define the *Resource Impact* for the generic ED node n_x (i.e., RI_x) as the actual impact (e.g., the cost) of the task offloading service w.r.t. the amount of resources offered/required by ED node *x*:

$$RI_x = \sum_{j=1}^{n} r_j \cdot c_j \tag{1}$$

where r_j is the *j*-th offered/required resource expressed in resource units (CPU, storage, bandwidth and so on) and c_j is its associated unitary impact.

We define the Behavioural Trend (BT) of the generic ED node n_x to share or consume resources for task offloading within a pool of ED node:

$$BT_x^{(t)} = \alpha \cdot BT_x^{(t-1)} + (1-\alpha) \cdot \frac{RI_{x,req}^{(t)}}{RI_{x,req}^{(t)} + RI_{x,off}^{(t)}} \tag{2}$$

where $RI_{x,off}$ is the impact of offered resources, while $RI_{x,req}$ is the impact of requested resources. In particular, the resources requested by the ED node n_x represent the amount of resources requested/consumed by ED node n_x that asks for partial task offloading to some other nodes. The value of $\alpha \in [0,1] \in \mathbb{R}$ represent the relevance of the contributions. In particular, every value of $BT \in [0,1] \in \mathbb{R}$ is computed, at time *t*, by weighting the value computed at time $t-1$, and the new contribution computed at time *t* weighted by $(1-\alpha)$.

The second contribution in equation 2 is computed as the ratio of requested amount of resources over the total (requested and offered). Therefore, a ratio close to zero indicates a trend to export more resources than those requested, and viceversa.

The Pool Behaviour is a similar measure aimed at evaluating the global behaviour of a pool of ED node s:

$$PB_j^{(t)} = \frac{1}{||p_i||} \sum_{k=1}^{||p_i||} BT_k^{(t)} \tag{3}$$

Clearly PB_j assumes a value in [0, 1] and represents the tendency of the whole pool p_j to offer or require resources.

PB represents a behavioral trend metric designed to capture long-term resource consumption and provisioning tendencies, rather than instantaneous behavior. The use of the absolute difference $|BT_i - PB_j|$ reflects the intuition that collaborative efficiency is maximized when complementary behaviors are matched, thus favoring load balancing and efficient resource sharing.

The Trust Measures

Preliminarly, we highlight that in this work, Quality of Service is considered at the service and collaboration level, rather than at the network protocol level. Accordingly, the proposed CEC-Trust metric is designed to act as a proxy for QoS, capturing the reliability, effectiveness, and stability of collaborative task offloading through trust relationships and historical behavior, independently from specific network technologies.

In our model, a truster is any generic ED node n_x (agent a_x), on the behalf of its own pool user, with respect to n_y (agent a_r).

The reliability is defined as a measure of perceived trust computed on the basis the *interactions* occurred in the past, while the reputation measure represents an expected behavior based on the indirect knowledge represented by the past interactions occurred with other counterparts [16]. For example, in the scenario of subsection "An Example of Collaborative Task Offloading", let n_p be the ED node that receives the task offloading request and n_r the ED node that executes part of the task. After offloading, n_p must compute reliability indexes that reflect n_r's performance. After the task offloading process, ED node n_p has to extract reliability indexes reflecting the performance of the ED node n_r.

We denote the trust measure as $\tau_{x,y}$, the reliability as $\eta_{x,y}$ and the reputation as $\rho_{x,y}$ assuming that the generic ED node n_x (i.e., agent a_x) computes these measures w.r.t. the ED node n_y (i.e., agent a_y). The trust measure $\tau_{x,y}$ is computed by weighting the reliability ($\eta_{x,y}$) and the reputation ($\rho_{x,y}$) measures by a coefficient $\beta_{x,y} \in [0, 1] \in \mathbb{R}$:

$$\tau_{x,y} = \begin{cases} 0.5 & t = 0 \\ \beta_{x,y} \cdot \eta_{x,y} + (1 - \beta_{x,y}) \cdot \rho_{x,y} & t > 0 \end{cases} \tag{4}$$

The first part of the equation 4 represents the fact that every new ED node will hold an initial "neutral" trust value [17, 18].

Moreover, the coefficient $\beta_{x,y}$ will increase according to the number of interactions between the ED nodes as well as their freshness, as follows:

$$\beta_{x,y} = \frac{I_{x,y}}{I_{max}} \tag{5}$$

where I_{max} represents the maximum allowed number of interactions, i.e. after which the "knowledge" of n_x w.r.t. n_y is considered maximum, and, furthermore $I_{x,y}$ ise defined as follows:

$$I^t_{x,y} = \begin{cases} max(1, I^{(t-1)} - 1) & \text{if } \Delta T_{x,y} > \Delta T \\ min(I_{max}, I^{(t-1)} + 1) & \text{if } \Delta T_{x,y} \leq \Delta T \end{cases} \tag{6}$$

where ΔT is a threshold that allow us to limit the amount of time between the current interaction and the last one.

Therefore $\beta_{x,y}$ is computed to assign a different weight to the reliability according to the experience of n_x about n_y as well as the freshness of such an experience. In this way the larger the number of the interactions occurred between the ED nodes, the bigger the weight assigned to the reputation in computing the trust value. At the same time, whenever the last interaction with n_x is older than ΔT, the reputation increases in relevance, or the reliability decreases in relevance, (first row of Eq. 6), and vice versa (second row of 6). We remark that a proper calibration of ΔT and I_{max} will enable a fine tuning of the (i) relevance of reliability and reputation and the level of (ii) resilience to unexpected behaviors changes (possible malicious ED nodes).

We highlight that the parameters ΔT and I_{max} are introduced to control the temporal relevance of interactions and to bound the influence of accumulated experience. This prevents outdated information from dominating trust evaluation and improves resilience against sudden or strategic behavior changes.

Computation of reliability. Now we define the reliability $\eta_{x,y} \in [0, 1] \subset \mathbb{R}$ as

$$\eta^{(t)}_{x,y} = \vartheta_{x,y} \cdot \sigma_{x,y} + (1 - \vartheta_{x,y}) \cdot \eta^{(t-1)}_{x,y} \tag{7}$$

where $\vartheta_{x,y}$ weights *(i)* the feedback parameter $\sigma_{x,y} \in [0, 1] \in \mathbb{R}$ computed by n_x with respect to the task offloading provided by n_y at time-step *t* and *(ii)* the value of $\eta_{x,y}$ at time $(t - 1)$. Parameter $\vartheta_{x,y}$ is calculated on the basis of the relevance measure (ψ) assigned to the task offloading service $s_{x,y}$ and the average ($\bar{\psi}$) of the relevance measures of all task offloading services provided by n_y to n_x in the past, as follows (Eq. 8):

$$\vartheta_{x,y} = \text{clip}_{[0,1]}\left(0.5 - \frac{\psi_{x,y} - \bar{\psi}_{x,y}}{2}\right), \tag{8}$$

where $\psi_{x,y} \in [0, 1]$ denotes the relevance of the current task offloading service provided by n_y to n_x, and $\bar{\psi}_{x,y} \in [0, 1]$ denotes the historical average relevance computed over

past interactions between n_x and n_y. The clipping operator $\text{clip}_{[0,1]}(\cdot)$ ensures $\vartheta_{x,y} \in [0,1]$.

The effect of Eq. 8 is to limit the malicious behaviour of some ED nodes that aim at acquiring positive feedback by providing services that assume marginal importance.

Computation of reputation. The reputation is computed due to the partial view of the single agent, which will compute its own trust measure in a different way from the other agents of the community. Therefore the ED node n_x should calculate the reputation of the ED node n_y and, to this end, it will ask an opinion about n_y to an ED node (agent) n_q. Then the agent a_q will provide its opinion to a_x consisting of its own trust measure about n_y (i.e., $\tau_{q,y}$).

We denote the reputation as $\rho_{x,y}$, which is computed by an ED node n_x with respect to a given ED node n_y as a value ranging in $[0,1] \subset \mathbb{R}$:

$$\rho_{x,y} = \frac{\sum_{q=1}^{||R_{x,y}||} (\tau_{q,y})}{||R_{x,y}||} \tag{9}$$

where $||R_{x,y}||$ is the cardinality of the set of agents which provided an opinion about n_y .

Note that, in the computation of reputation, opinions are collected from a bounded subset of ED nodes. In the experiments, $\tau_{sample} = 125$ peers are selected uniformly at random from the population. This sampling strategy provides a balance between robustness and scalability, limiting communication overhead while mitigating the influence of collusive or biased recommendations. The parameter τ_{sample} is configurable and can be adapted to the size and characteristics of the deployment environment.

Numerical example (one trust update epoch)

Consider n_x interacting with n_y at epoch t. Assume $\eta_{x,y}^{(t-1)} = 0.60$, $\rho_{x,y}^{(t)} = 0.70$, and the current feedback is $\sigma_{x,y}^{(t)} = 0.80$. Let the current relevance be $\psi_{x,y}^{(t)} = 0.90$ and the historical average relevance be $\bar{\psi}_{x,y} = 0.60$. Using Eq. (8), we obtain $\vartheta_{x,y}^{(t)} = 0.5 - (\psi_{x,y}^{(t)} - \bar{\psi}_{x,y})/2 = 0.35$. Then, by Eq. (7),

$$\begin{aligned}\eta_{x,y}^{(t)} &= \vartheta_{x,y}^{(t)} \cdot \sigma_{x,y}^{(t)} + \left(1 - \vartheta_{x,y}^{(t)}\right) \cdot \eta_{x,y}^{(t-1)} \\ &= 0.35 \cdot 0.80 + 0.65 \cdot 0.60 = 0.67.\end{aligned}$$

Assume $I_{max} = 10$ and the interaction counter is $I_{x,y}^{(t)} = 6$, hence $\beta_{x,y}^{(t)} = I_{x,y}^{(t)}/I_{max} = 0.60$. Finally, using Eq. (4),

$$\begin{aligned}\tau_{x,y}^{(t)} &= \beta_{x,y}^{(t)} \cdot \eta_{x,y}^{(t)} + \left(1 - \beta_{x,y}^{(t)}\right) \cdot \rho_{x,y}^{(t)} \\ &= 0.60 \cdot 0.67 + 0.40 \cdot 0.70 \approx 0.68.\end{aligned}$$

This example shows how a single interaction updates reliability and trust based on feedback, relevance, and interaction history.

The CEC-Trust Measure

Here we define the CEC-Trust measure computed by the generic ED node n_i. This measure will indicate the potential advantages of any ED node n_j to join with the pool p_j. We denote the CEC-Trust as $C_{i,j}$, which is computed as follows:

$$C_{i,j} = |BT_i - PB_j| \cdot \frac{\sum_{n_k \in p_j} \tau_{i,k}}{||p_j||} \tag{10}$$

where $||p_j||$ is the number of ED nodes affiliates with p_j; therefore the larger is the difference between the behaviors of n_i and p_j, the larger is $C_{i,j}$; moreover, the larger is the average trust computed by n_i for all the ED nodes belonging to p_j, the larger the CEC-trust measure.

We also measure the CEC-trust for a pool p_j, in order to give an indication whether to accept the request of any ED node n_i to join with p_j:

$$C_{j,i}^{*} = |BT_i - PB_j| \cdot \frac{\sum_{n_k \in p_j} \tau_{k,i}}{||p_j||} \tag{11}$$

In defining $C_{j,i}^{*}$ we considered the behaviors of n_i and p_j, on the one hand, and the trust computed by all the ED nodes $n_k \in p_j$ for n_i .

Note that, in general, $C_{i,j}^{*} \neq \eta_{j,i}$, in other words the trust computed by the ED node n_i to the ED node n_j is not the same of that computed by n_j for n_i, therefore Eq. 10 and 11 assume different values for the ED nodes n_i and n_j .

The reader may refer to Table 1 which describes all the used notation, as well as the introduced acronyms.

Behavioral Complementarity

The term $|BT_i - PB_j|$ is intentionally introduced to capture *behavioral complementarity* between an ED node n_i and a pool p_j. In Collaborative Edge Computing, complementarity promotes diversity of resource tendencies within pools, improving load balancing and reducing the risk of forming demand-heavy (or supply-heavy) groups that are less effective for collaborative task offloading.

Resilience to Byzantine and Collusive Behaviors

The proposed trust update mechanism does not rely on global consensus and therefore does not provide Byzantine fault tolerance in the classical sense. Instead, resilience emerges from bounded reputation sampling, interaction-weighted

Table 1 Notation used in the trust and pool formation model

Symbol	Meaning
n_x, n_y	ED nodes (truster x, trustee y)
t	Discrete time index (epoch)
$\tau_{x,y}^{(t)} \in [0,1]$	Trust of n_x toward n_y at time t
$\eta_{x,y}^{(t)} \in [0,1]$	Reliability of n_x toward n_y at time t
$\rho_{x,y}^{(t)} \in [0,1]$	Reputation of n_y as computed by n_x at time t
$\beta_{x,y}^{(t)} \in [0,1]$	Weight between reliability and reputation
$I_{x,y}^{(t)}$	Interaction counter (freshness-aware) between n_x and n_y
I_{max}	Maximum interaction level used to bound $\beta_{x,y}^{(t)}$
ΔT	Freshness threshold for last interaction
$\sigma_{x,y}^{(t)} \in [0,1]$	Feedback score from n_x about service from n_y at t
$\psi_{x,y}^{(t)} \in [0,1]$	Relevance of the current interaction at t
$\bar{\psi}_{x,y}$	Historical average relevance between n_x and n_y
$\vartheta_{x,y}^{(t)} \in [0,1]$	Weight of current feedback in reliability update
$BT_x^{(t)}$	Behavioral Trend of ED node n_x
$PB_j^{(t)}$	Pool Behavior of pool p_j
$C_{i,j}$	CEC-Trust of node n_i toward pool p_j
$C_{j,i}^*$	Pool-side CEC-Trust of pool p_j toward node n_i
N_{MAX}, K_{MAX}	Bounds on pools joined per node / nodes per pool
CEC	Collaborative Edge Computing
ED	Edge Domain
PF	Pool Formation (algorithm)
MMC	Mobile Micro Cloud
SCC	Small Cell Cloud

updates, and trust decay. Equations 10–11 progressively reduce the influence of malicious or colluding ED nodes by favoring direct experience over indirect opinions and by limiting the impact of coordinated false feedback. This approach provides probabilistic robustness against Byzantine behaviors while preserving scalability and decentralization.

Additional Considerations

Note that, although all ED nodes are initialized with a neutral trust value, malicious and unreliable behaviors are explicitly introduced from the early stages of the simulations, including collusive and adaptive attack strategies. This allows us to evaluate the capability of the proposed trust model to identify and mitigate untrustworthy behavior without assuming an initial honest-only phase. The parameters of the trust model are intended to be configurable in order to adapt the system to different Collaborative Edge Computing scenarios. In particular, α regulates the balance between historical behavior and recent observations, and can be tuned to favor stability in relatively static environments or responsiveness in highly dynamic ones. Similarly, the interaction-based coefficient $\beta_{x,y}$ allows the trust computation to progressively rely more on direct experience as the number and freshness of interactions increase. Although fixed parameter values are used in this study, future work will investigate adaptive and context-aware tuning strategies, as well as sensitivity analyses, to further support real-world deployments. Furthermore, it is worth noting that resilience to malicious and unreliable behavior emerges from the dynamic interaction between reliability, reputation, and freshness-aware trust updates, rather than from explicit attack detection mechanisms. In particular, sophisticated behaviors such as collusion or reputation-building followed by service degradation are implicitly captured by the trust model, as demonstrated by the simulated scenarios in which malicious ED nodes adapt their behavior over time. The experimental results confirm that such nodes are progressively identified and penalized through trust decay, even in the presence of coordinated or strategic attacks.

Pool Formation (PF) Algorithm

Here we illustrate the decentralized procedure named PF (Pool Formation), which is based on the measures defined in "Behavioral and Trust Measures in Pools of Edge Domains". The algorithm is composed by two parts executed by ED agents and pool agents, respectively.

Let *T* be the time elapsed between two consecutive runs of the PF procedure. We also assume that agents are able to query a distributed repository *PR* (Pool Repository) on which the list of the pools is maintained.

The PF Procedure Performed by the Node Agents

Let X_n be the set of the pools of ED nodes which the ED agent a_n is affiliated to, where $||X_n|| \leq N_{MAX}$, and N_{MAX} is the maximum number of pools which an ED agent can join with. We suppose that a_n can hold a cache of the pool profile $prof_p$ of each pool contacted in the past and the timestamp d of the execution of the PF procedure for that pool of ED agents. Finally, let ξ_n (a timestamp) and $\chi_n \in [0,1]$ be some thresholds fixed by the agent a_n .

The PF procedure performed by the ED agent a_n is represented by the following, ordered steps:

1. A set Y of nearest N_{max} pools so that $X_n \bigcap Y = \{0\}$ is randomly selected.
2. For each pool $p \in Z = (X_n \bigcup Y)$ such that $d_p > \xi_n$, a message to the agent a_p for asking its profile $prof_p$ is sent.
3. For each received $prof_p$, the CEC-Trust measure $C_{n,p}$ (see "The CEC-Trust Measure") between the profiles of the ED agent a_n and the pool *p* is computed.
4. The list L_{good}, with all the pools $p \in Z$ such that $C_{n,p} > \chi_n$, is updated.
5. A second list L'_{good} is built by inserting a number $k = min(N_{max}, ||L_{good}||)$ pools of L_{good} with the greater value of $C_{n,p}$.
6. For each pool $p \in L'_{good}$, if $p \notin X_n$, a join request to a_p together with the profile p_n is sent.
7. For each $p \in X, p \notin L'_{good}$, then the agent a_n deletes its ED node *n* from *p* (i.e., a message to a_p in order to leave the pool *p* is sent).

Definition of "Nearest Pools" and Candidate Selection

In Step 1 of the PF procedure, the term "nearest pools" refers to pools that are close to an ED node *n* according to a proximity function *d*(*n*, *p*). This function captures the expected communication cost between *n* and pool *p* and can be instantiated using measurable signals such as RTT/latency, hop count, or geographic distance, depending on the deployment context. Let $\mathcal{P}$ denote the set of available pools in the pool repository. The candidate set *Y* is selected as

$$Y = \arg \min_{Y \subseteq \mathcal{P} \setminus X_n,\ |Y| = N_{MAX}} \sum_{p \in Y} d(n,p),$$

i.e., the N_{MAX} pools minimizing *d*(*n*, *p*) among those not already joined by *n*. When network conditions are abstracted (as in our simulation setting), the bounded candidate selection is implemented by uniformly sampling N_{MAX} pools from $\mathcal{P} \setminus X_n$. This preserves scalability and decentralization, while avoiding assumptions on a specific network technology.

The PF Procedure Performed on the Pool Agent Side

We denote as K_p the set of the ED nodes of the pool *p*, with $||K_p|| \leq K_{MAX}$, where K_{MAX} represents the maximum number of ED nodes of the specific pool. The pool agent a_g holds a cache with the profile p_u of each ED node $u \in K_g$ and the timestamp d_u of its acquisition. Moreover, let ω_p (a timestamp) and $\pi_p \in [0,1] \subset \mathbb{R}$ be thresholds fixed by the agent a_p.

The PF procedure on the pool agent side is triggered whenever a join request by an agent a_n is received by a_p:

1. For each ED node $u \in K_p$, such that $d_u > \omega_g$, a message is sent to the ED node agent a_u to retrieve the profile p_u associated with *u*.
2. The CEC-Trust measure $C^*_{p,u}$ (see "The CEC-Trust Measure") for each ED node $u \in K_p \bigcup \{n\}$ – where *n* is the ED node which has asked to join with the pool *p* – and the profile of the pool *p* is computed.
3. The list of best candidates K_{good} that stores the ED nodes *u* such that $\eta_{p,u} > \pi_p$ is updated.
4. A second list K'_{good} stores a number $s = min(K_{max}, ||K_{good}||)$ of ED nodes belonging to K_{good} having the greater value of $C^*_{p,u}$.
5. For each ED node $u \in K_p$, if $u \notin K'_{good}$, the pool agent a_p deletes *u* from *p* and notifies *u*. Clearly, if $n \in K'_{good}$, its request to join with *p* is accepted.

The PF algorithm is designed to be scalable by construction. Both ED agents and pool agents operate using local information, bounded candidate sets, and cached profiles, which limits computational and communication overhead. In particular, trust values are updated incrementally after interactions, while pool selection is restricted by configurable thresholds (e.g., N_{MAX} and K_{MAX}), preventing

global coordination and ensuring scalability in large-scale CEC environments. The parameters N_{MAX} and K_{MAX} are used to explicitly bound the size of the search space for pool selection and membership, ensuring scalability and limiting computational and communication overhead in large-scale Collaborative Edge Computing environments.

PF workflow overview

Fig. 2 provides a high-level workflow diagram of the Pool Formation (PF) algorithm. The diagram summarizes the sequence of operations performed by an ED node in each epoch, from candidate pool discovery to trust and behavioral evaluation, pool selection, and membership update. This visual overview complements the algorithmic description and helps clarify the distributed and iterative nature of the PF procedure.

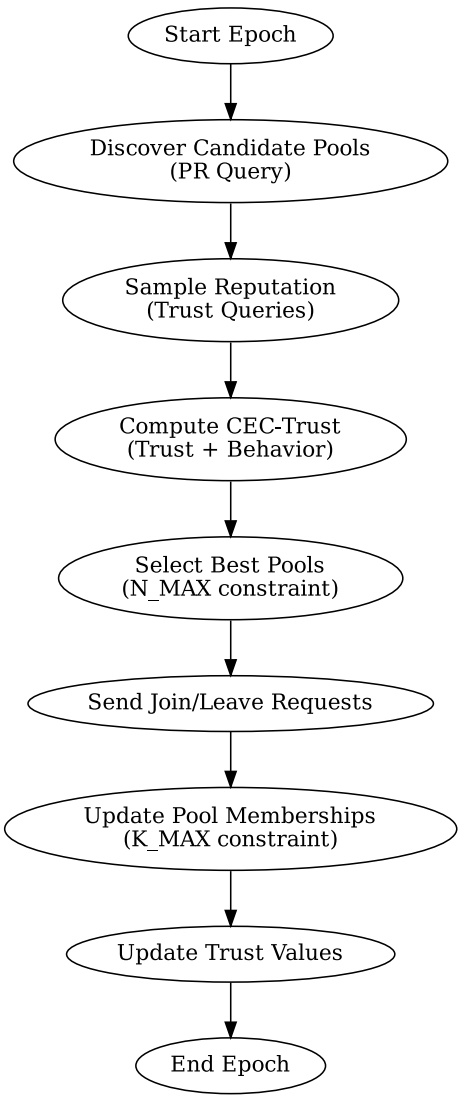

Fig. 2 Workflow diagram of the distributed Pool Formation (PF) algorithm. At each epoch, an ED node discovers candidate pools, evaluates them using the CEC-Trust metric, selects the most suitable pools, and updates its memberships accordingly

Logical Communication Protocol

The proposed framework can be implemented by including a lightweight, asynchronous, message-based protocol among ED nodes and pools. The main message types include *TaskOffloadRequest*, *TaskOffloadResponse*, *TrustQuery*, *TrustReply*, *PoolJoinRequest*, and *PoolJoinReply*. Trust updates are triggered only after task execution feedback is received, ensuring causal consistency. Potential conflicts, such as concurrent pool join attempts or inconsistent reputation opinions, are resolved locally through trust-based ranking and bounded candidate selection. This decentralized protocol avoids global synchronization and ensures scalability in large-scale Collaborative Edge Computing environments.

Computational Complexity Issues

From a complexity perspective, the proposed PF algorithm is scalable by construction. Each ED node evaluates a bounded number of candidate pools and maintains trust values only for interacted peers. Therefore, the per-node computational complexity is $\mathcal{O}(N_{MAX} + K_{MAX})$, independent of the total number of ED nodes in the system. This design choice ensures scalability in large-scale Collaborative Edge Computing environments.

Termination and Convergence Discussion

The PF procedure terminates within each epoch because all node actions are bounded: each ED node evaluates at most N_{MAX} candidate pools, each pool enforces a maximum capacity K_{MAX}, and reputation sampling is bounded by the parameter τ_{sample}. Therefore, each epoch involves a finite number of join/leave operations and a finite amount of computation and messaging per node. Although we do not claim a formal proof of global optimality, PF can be interpreted as a distributed best-response process where each ED node locally maximizes its pool desirability score (CEC-Trust) under bounded constraints. Convergence can be expected when system dynamics are stationary or slowly varying, and when trust updates exhibit diminishing changes over time due to interaction saturation and temporal decay. Under these assumptions, the system tends to reach a stable configuration in which unilateral pool changes do not provide significant improvement. In the worst-case highly non-stationary settings (e.g., extreme churn or adversarial perturbations), oscillations may occur. Nevertheless, per-epoch overhead remains bounded by N_{MAX}, K_{MAX}, and τ_{sample}. Practical stabilization can be further improved through standard mechanisms such as hysteresis (e.g., minimum residence time in a pool) or adaptive PF execution intervals.

Communication Overhead per Epoch

The proposed framework operates through a lightweight, asynchronous control protocol. Let N_{MAX} be the maximum number of candidate pools evaluated by an ED node, K_{MAX} the maximum pool size, and τ_{sample} the bounded number of peers queried for reputation. In one PF epoch, an ED node exchanges a bounded number of control messages: *(i)* one *PoolQuery* and one *PoolReply* to obtain candidate pools, *(ii)* up to N_{MAX} *PoolJoinRequest*/*PoolJoinReply* pairs, and *(iii)* up to τ_{sample} *TrustQuery*/*TrustReply* pairs for reputation sampling. Therefore, the control-message complexity per node per epoch is $\mathcal{O}(N_{MAX} + \tau_{sample})$, independent of the total number of ED nodes. The corresponding byte overhead is proportional to the message payload sizes and remains bounded under fixed N_{MAX} and τ_{sample}.

Operational Framework

The proposed CEC-Trust framework operates through a continuous and decentralized lifecycle composed of the following phases.

System Initialization

Each ED node initializes its local trust values to a neutral state and joins a small set of initial pools satisfying the minimum size constraints. Initial pool memberships and behavioral measures are established using local resource availability and bootstrap interactions.

Monitoring and Trust Update

During operation, ED nodes continuously monitor task offloading outcomes and interaction feedback. Trust values are updated incrementally after each interaction using the defined trust update rules, allowing the system to adapt to dynamic behavior changes.

Incremental Rollout and Pool Adaptation

The Pool Formation algorithm is executed periodically or upon significant events (e.g., performance degradation or churn). Pools evolve incrementally, enabling ED nodes to join or leave collaborative configurations based on local CEC-Trust optimization.

Failure Handling and Recovery

When failures, malicious behavior, or performance degradation are detected through trust decay, ED nodes autonomously reduce reliance on affected pools and re-execute the PF procedure. This local recovery mechanism allows the system to maintain stable operation without requiring global coordination or rollback procedures.

Experiments

In this Section we describe the results of a set of simulations, which consists of two parts. The former contains experiments aimed at testing the effectiveness of the trust measure we have introduced in "Behavioral and Trust Measures in Pools of Edge Domains", while the latter part represents a simulation of the execution of the PF algorithm discussed in "Pool Formation (PF) Algorithm".

Simulation Setup

In the simulations, each ED node is characterized by a heterogeneous amount of computational resources, abstracted as resource units representing CPU, storage, or bandwidth capacity. Task offloading requests generated by user devices are modeled as variable resource demands, allowing both single-node execution and collaborative offloading when local resources are insufficient. Network conditions are not explicitly simulated at the protocol level; instead, their effects are implicitly captured through interaction frequency, collaboration probability, and trust update dynamics. This abstraction allows us to focus on the impact of trust relationships and pool formation on the overall Quality of Service, independently from specific network technologies or deployment assumptions.

The Trust Model

A set of experiments was carried out by simulating a population consisting of 16,000 EDs which interact with a number of UD to provide task offloading services. Each ED is initially joined with a random number of pools of ED comprised between 5 and 10. Each experiment consisted of 50000 epochs, where in each epoch about 250 UD send task offloading requests which require, in average, resources belonging to other ED nodes. This particular aspect was simply modeled, without any loss of generality, by representing any request by a number representing the "quantity" of requested resources (*rt*) and, at the same time, by characterizing any ED by a number representing the quantity of resources which can be allocated for any given task (*r*). When $r > rt$, then the ED will ask to another ED node of a pool to get part of the computation by assigning the remaining part of the task offloading. We tuned the simulation such that almost the 70% of the task offloading requests involved two different EDs.

Regarding reputation values, i.e., the opinion asked by the EDs to the other ED nodes, each ED node was allowed to request opinions from up to $\tau_{sample} = 125$ randomly selected ED nodes within the population.

Table 2 Scenarios simulated for the trust system

Scenario	Ratio of unreliable	Malicious	Behavior
A	0.1 or 0.5	No	Totally reliable or unreliable
B	0.1 or 0.5	0.1 or 0.5	Totally reliable and honest or totally unreliable and malicious
C	0.1 or 0.5	0.1 or 0.5	Building positive reputations on services for about 1/4 of the task offloading services

When a simulation starts, all the ED nodes are assumed to be not malicious and trusted, i.e. they receive an initial trust value of 0.5 (see "Behavioral and Trust Measures in Pools of Edge Domains").

Since the trust model was designed also to identify the malicious ED nodes, we simulated both *(i)* a variable number of malicious ED nodes, and *(ii)* a variable number of unreliable ED nodes. In particular, we simulated two different scenarios, *A*, *B* and *C*, each identifying a separate experiment, as described in Table 2, where in the *(i)* scenario *A* 10% or 50% of the ED nodes have an unreliable behavior; *(ii)* scenario *B* is designed such that unreliable ED nodes show also a malicious/dishonest behavior when they provide recommendations to other ED nodes; *(iii)* scenario *C* is designed as the scenario *B*, but malicious ED nodes show a sophisticated strategy in order to build a positive reputation for about 1/4 of the provided task offloading services, and afterwards they provide unreliable task offloading services for the remaining.

Parameter Settings and Sensitivity Analysis

Table 3 summarizes the parameters used in the proposed model and in the PF algorithm, reporting both the default values used in the simulations and the ranges considered in sensitivity tests.

We performed a sensitivity analysis by varying α, ΔT, and I_{max} within the ranges reported in Table 3. The results confirm that the PF procedure consistently improves the Average CEC-Trust and preserves the main trends of the evaluation across different parameter settings.

Reproducibility of the SImulation Setup

All simulation parameters and stochastic components are generated using uniform random distributions within the bounds specified in Table 2. Initial ED node behaviors, resource capacities, pool sizes, and interaction partners are randomly assigned at the beginning of each simulation run. To ensure reproducibility, all experiments are executed using a fixed random seed. The Pool Formation algorithm described in Sect. "The PF Procedure Performed by the Node Agents" is executed periodically within the simulation loop, and trust values are updated incrementally after each task offloading interaction according to the defined trust update rules.

Simulation-to-Deployment Adaptation

Table 4 summarizes how key real-world factors are abstracted in our simulations and how they can be instantiated in practical deployments.

The above adaptations do not change the core PF logic and preserve decentralization, while allowing CEC-Trust to be instantiated with measurable deployment signals in realistic Collaborative Edge Computing environments.

Simulation Results

To analyze experimental results, we measured the ratio of ED nodes which correctly recognize unreliable ED nodes by means of the trust measure presented in "Behavioral and Trust Measures in Pools of Edge Domains". We denote this measure as *LTE* (Low-Trust ED node Estimation).

Figure 3 shows the results of a number of experiments carried out by simulating the scenarios A, B, while Fig. 4 reports the experimental results of the scenario C. The results named *A10* and *A50* represent the trend of the LTE over the various simulation epochs, with *A10* representing the case where the 10% of the ED nodes assume an unreliable behavior, and *A50* the case with the 50% of the ED nodes assuming an unreliable behavior, as already illustrated in Table 2. Moreover, the bars *B10* and *B50* represent the trend of LTE with the 10% and 50% of the ED nodes

Table 3 Model and algorithm parameters used in the experiments

Parameter	Default	Range tested	Meaning
α	0.5	0.2–0.8	Behavioral memory factor in $BT_x^{(t)}$
ΔT	10 epochs	1–100 epochs	Freshness threshold for interaction recency
I_{MAX}	20	10–100	Maximum interaction level used to bound $\beta_{x,y}$
N_{MAX}	10	5–20	Maximum number of pools an ED node can join
K_{MAX}	80	20–200	Maximum number of ED nodes per pool
N_{MIN}	2	1–10	Minimum number of pools per ED node at initialization
K_{MIN}	10	20–200	Minimum pool size at initialization

Table 4 Simulation-to-deployment adaptation guidelines for CEC-Trust

Factor	Simulation abstraction	Deployment signals	Adaptation in CEC-Trust / PF
Latency/network cost	Not simulated at protocol level; reflected indirectly via interaction dynamics	RTT estimates, hop count, link quality, geographic distance	Define a proximity function $d(n, p)$ to select "nearest pools"; optionally penalize high $d(n, p)$ in candidate selection
Churn/mobility	Pools and memberships evolve by periodic PF runs	Join/leave events, session duration, availability monitoring	Increase PF frequency under high churn; shorten ΔT to discount stale interactions; keep bounded N_{MAX}, K_{MAX}
Feedback quality and delay	Immediate feedback after task completion via $\sigma_{x,y}$	Application-level success/failure, deadline miss, completion time, SLA violations	Compute $\sigma_{x,y}$ from service-level outcomes; apply delayed updates when feedback arrives; use trust decay to handle missing feedback
Workload variability	Task demand modeled as variable resource units	Request traces, CPU/memory usage, queue length, arrival rate	Map resource units to monitored resources; adapt α to trade off stability vs. responsiveness under non-stationary workloads
Adversarial behavior/attacks	Unreliable, collusive, and strategic behaviors simulated (scenarios A–C)	False recommendations, intermittent misbehavior, collusion clusters	Limit reputation via bounded sampling τ_{sample}; favor direct evidence via interaction-weighted $\beta_{x,y}$; reduce influence of stale info via ΔT
Heterogeneous hardware	ED capability modeled as heterogeneous resource capacity	CPU type, accelerators, memory, energy constraints	Encode capability into resource units and costs c_j; update behavioral measures accordingly

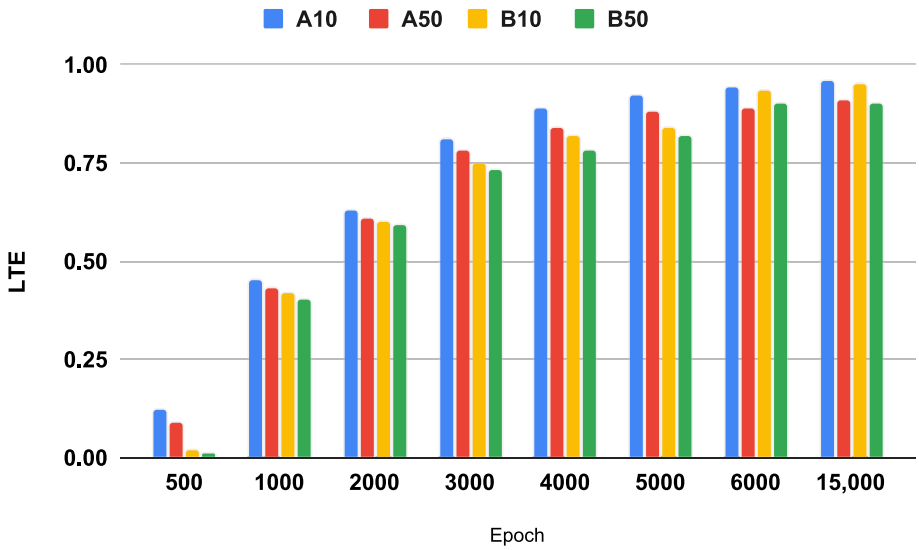

Fig. 3 LTE vs epochs that for the scenarios: **A** uniform behavior, **B** alternate behavior

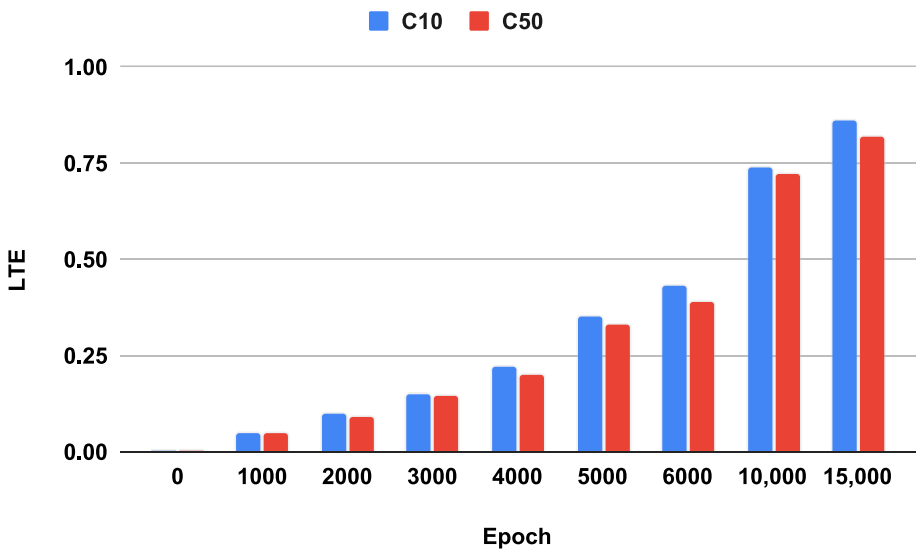

Fig. 4 LTE vs epochs that for scenario C

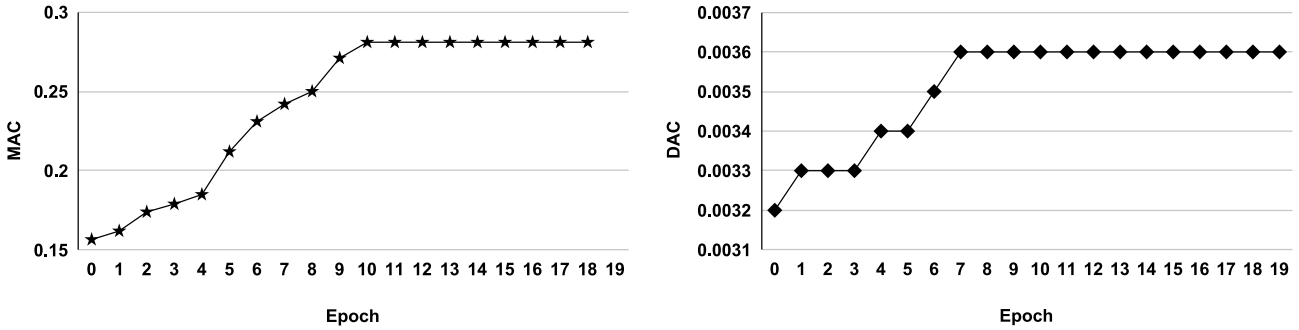

Fig. 5 Experiment 1, 40 pools: Variation of MAC (left) and DAC (right) vs epochs in the scenario A, with 4000 ED nodes

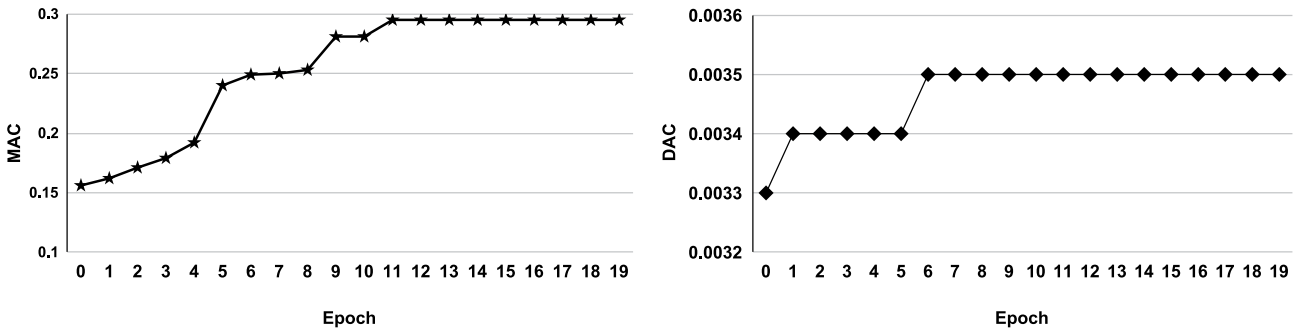

Fig. 6 Experiment 1, 40 pools: Variation of MAC (left) and DAC (right) vs epochs in the scenario B, with 8000 ED nodes

having the behavior of the scenario *B*. The same applies to the figure 4, which shows the results for *C*10 and *C*50.

These experiments highlight the resilience of the system to malicious and unreliable ED nodes. For example, under uniform behavior *A10* and *A50*, about 90% of the ED nodes correctly identified all malicious nodes within 1500 epochs. In the case where collusive activities are introduced (*B10* and *B50*), the results show a very little gap at the beginning of the simulation (it is the effect of the wrong values of the malicious opinions), but such a gap becomes equal to 0 around the 4, 500-th epoch. We can observe that the above gap is high when the simulation starts (see Sect. "The Collaborative Edge Computing Scenario"), but it quickly disappears as the simulation reaches about 1000 epochs. Moreover, when dynamic behavior is simulated (curves labeled *C10* and *C50*) the influence of trouble behaviors is clearly evident.

The PF Algorithm

Here we discuss the experimental results related to the effectiveness of the *PF* algorithm (Sect. "Pool Formation (PF) Algorithm").

First of all, we performed experiments related to three scenarios consisting of 4000, 8000 and 16000 ED nodes respectively, and a fixed number of 40 pools of EDs. Moreover, 10% of ED nodes was simulated as having an unreliable behavior. All the parameters have been randomly generated by a uniform distribution in their respective domains. To start the simulation we randomly generated the behavior of each ED node in terms of resources offered or requested in collaborative task offloading as well as the honest or malicious behavior. Moreover, each ED node could join a random number of pools between $N_{MIN} = 2$ and $N_{MAX} = 10$; the number of ED nodes for any single pools was randomly assigned in the range $K_{MIN} = 10$, $K_{MAX} = 80$ (see Table 3).

To measure the internal trust within a pool p_j, we introduced the concept of *Average CEC-Trust* (*AC*) computed on the basis of the CEC-Trust measure described by Eq. 11 in "The Collaborative Edge Computing Scenario". More in detail, the Average CEC-Trust AC_j is the average of all the measures $C^*_{j,i}$ computed by the pool $p_j \in \mathcal{P}$ for all its affiliated ED nodes $n_i \in p_j$. Therefore, to measure the global CEC-Trust of all the pools belonging to the set of pools $\mathcal{P}$ in our simulated scenario, we computed the mean (denoted as *MAC*) and the standard deviation (denoted as *DAC*) of all the AC_j:

$$MAC = \frac{\sum_{p_j \in G} AC_j}{||P||} \qquad DAC = \sqrt{\frac{\sum_{p_j \in G} (AC_j - MAC)^2}{||P||}}$$

Average CEC-Trust. The results of the simulations related to the three scenarios (Table 2, A, B, and C), in terms of numerical index *MAC* and *DAC* are shown in Figs. 5, 6 and 7, left (MAC) right (DAC).

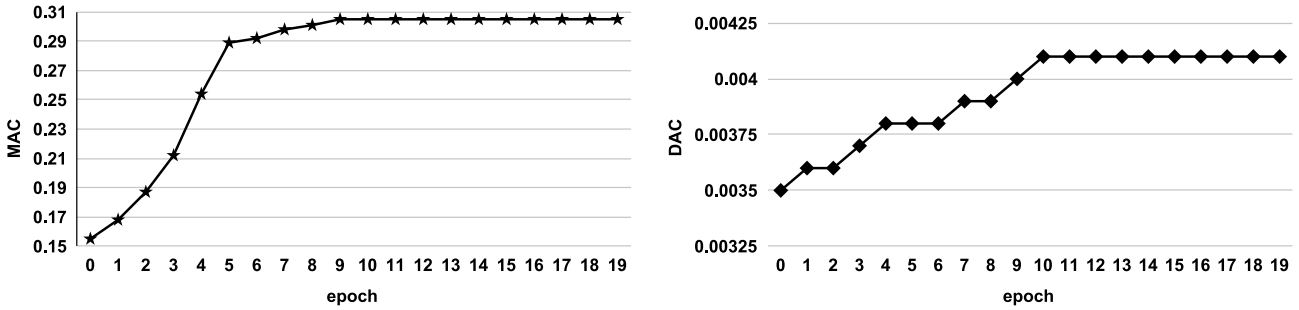

Fig. 7 Experiment 1, 40 pools: Variation of MAC (left) and DAC (right) vs epochs in the scenario C, with 16000 ED nodes

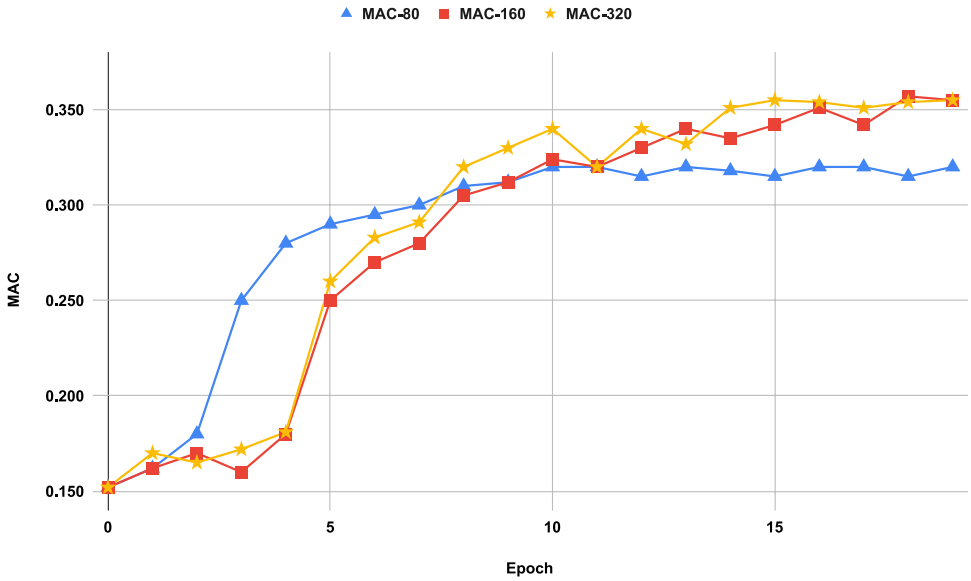

Fig. 8 Experiment 2, 16000 ED nodes: Variation of MAC vs epochs in the 1st scenario (80 pools), 2nd scenario (160 pools), 3rd scenario (320 pools)

The results indicate that the designed algorithm introduces a relevant increment of the CEC-Trust in the pools; in particular, after only 10 epochs it achieves a good configuration, with $MAC = 0.281$ and $DAC = 0.0035$ in the first scenario, $MAC = 0.295$ and $DAC = 0.0036$ in the second scenario and $MAC = 0.305$ and $DAC = 0.0041$ in the third scenario. The results show an improvement of about 44% of the average CEC-Trust measure of the set of pools in the first scenario; a 47% and 49% in the second and third scenario, while the standard deviation is very small. We also observe that the bigger the the number of ED nodes, the better is the performances of the algorithm.

Average CEC-Trust vs no. of pools. In a second experiment we investigated on the effect of the number of pools on the algorithm effectiveness, therefore we simulated the three different scenarios A, B and C (Table 2) with a fixed number of 16, 000 ED nodes and a variable numbers of pools, namely 80, 160 and 320, respectively. Also in this case we have simulated a random configurations for the pool.

The results of the simulations for the three scenarios, in terms of MAC with respect to the epochs, are shown in Fig. 8. We observe that the PF algorithm improves its performances in term of CEC-trust as the number of pools increases: just after 10 epochs, it achieves a stable configuration with $MAC = 0.321$ in the first scenario (80 pools), $MAC = 0.327$ in the second scenario (160 pools) and $MAC = 0.350$ and in the third scenario (320 pools). As a result, we have obtained an improvement of about a 49% in the average CEC-trust of the pools in the first scenario, and 51% and 43% in the second and third scenario; also in this case we measured a small standard deviation (about $0.0043 - 0.0052$).

Baselines and Ablation Study

To strengthen the experimental validation, we extended the evaluation by introducing additional baseline strategies and an ablation study. In particular, we compare the proposed PF approach driven by the full CEC-Trust metric against: *(i)* random pool assignment, *(ii)* a trust-only strategy based solely on reliability and reputation, and *(iii)* a behavior-only strategy based exclusively on behavioral complementarity. As an additional heuristic baseline, we consider a greedy pool selection strategy based on trust aggregation at the pool

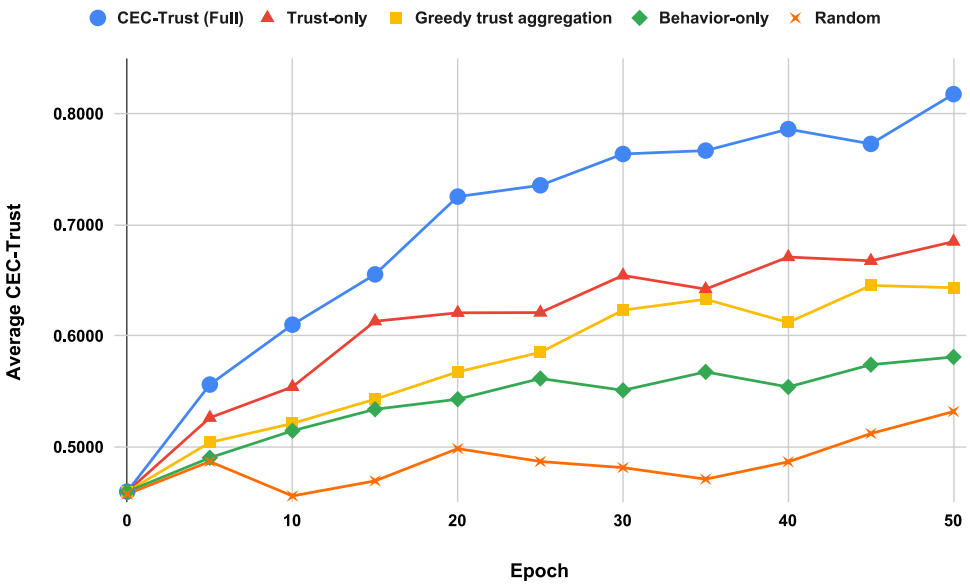

Fig. 9 Comparison of Pool Formation strategies in terms of Average CEC-Trust over time. The proposed CEC-Trust-driven PF algorithm consistently outperforms all baseline strategies, including trust-only, behavior-only, greedy trust aggregation, and random assignment

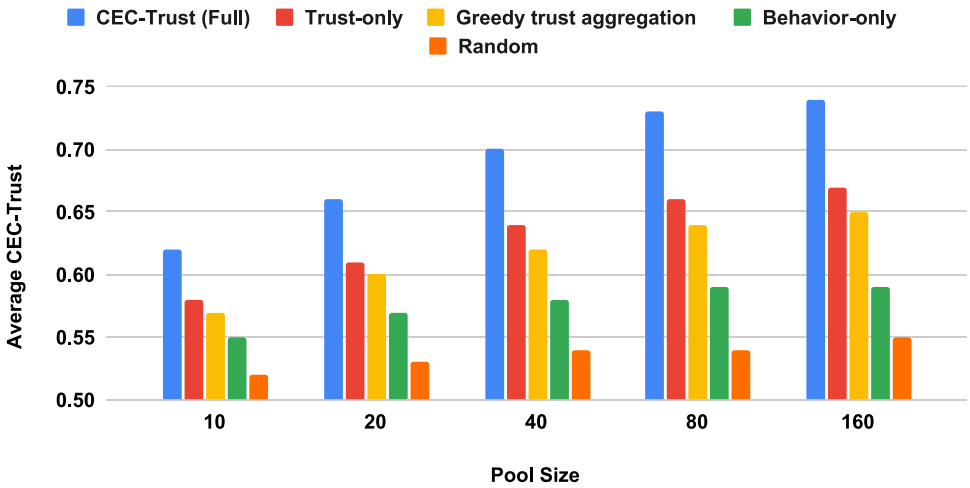

Fig. 10 Impact of pool size on Average CEC-Trust for different Pool Formation strategies. The full CEC-Trust approach scales more effectively than trust-only, behavior-only, greedy trust aggregation, and random baselines, maintaining higher trust levels as pool size increases

level. In this baseline, each ED node n_i evaluates a candidate pool p_j by computing the average trust toward its current members:

$$G_{i,j} = \frac{1}{|p_j|} \sum_{n_k \in p_j} \tau_{i,k}.$$

The ED node then selects the N_{MAX} pools with the highest $G_{i,j}$ values. This strategy represents a simple and intuitive heuristic that relies solely on trust aggregation, without considering behavioral complementarity or dynamic pool adaptation. These baselines allow us to isolate and quantify the contribution of each component of the proposed model. Figure 9 reports the evolution of the Average CEC-Trust over time. The results show that the full CEC-Trust-driven PF algorithm consistently outperforms all baseline strategies, achieving faster convergence and higher steady-state values. The trust-only baseline exhibits moderate improvement, confirming the importance of trust dynamics in collaborative environments, while the behavior-only strategy shows limited effectiveness when used in isolation. As expected, random assignment yields the lowest and least stable performance. The greedy trust aggregation baseline performs better than random assignment, but is consistently outperformed by the proposed CEC-Trust-driven PF approach, which benefits from the combined use of trust dynamics and behavioral complementarity. Figure 10 illustrates the impact of pool size on the Average CEC-Trust. The proposed approach demonstrates superior scalability, maintaining higher trust levels as pool size increases. This indicates that combining trust evaluation with behavioral complementarity helps form pools that remain effective even in larger and more heterogeneous collaborative settings. In contrast, baseline strategies degrade more rapidly as pool size grows, highlighting the limitations of relying on a single component. Overall, these results confirm that neither behavioral measures nor trust dynamics alone are sufficient to achieve optimal pool formation. Instead, their

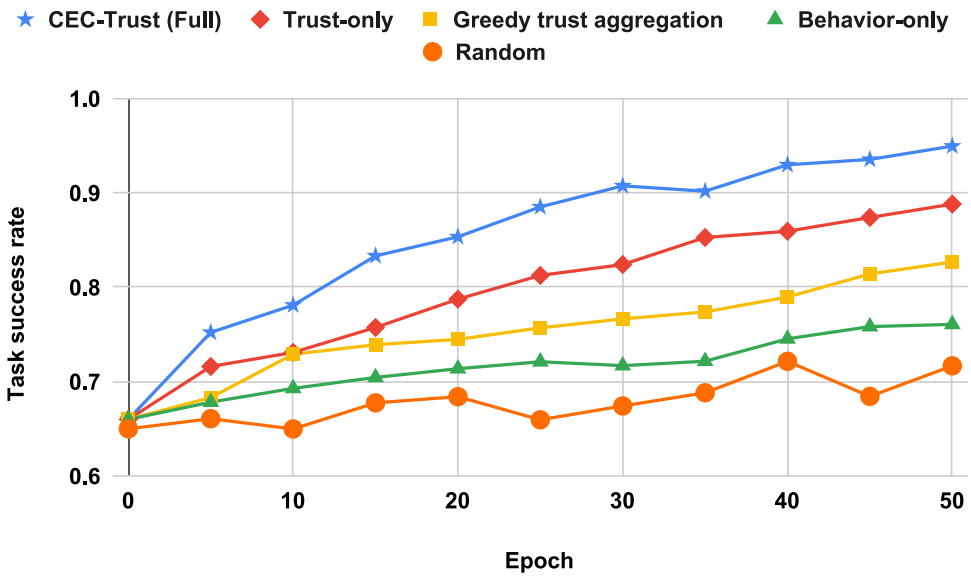

Fig. 11 Task success rate achieved by different Pool Formation strategies over time. The proposed CEC-Trust-driven PF algorithm achieves the highest success rate, outperforming trust-only, greedy trust aggregation, behavior-only, and random baselines

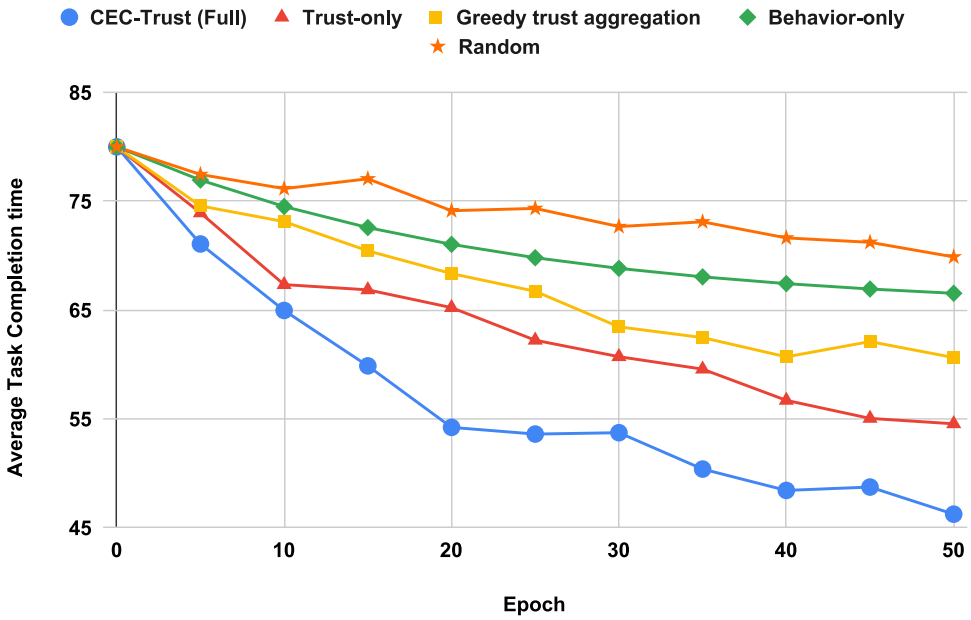

Fig. 12 Average task completion time over time for different Pool Formation strategies.

integration within the CEC-Trust metric is crucial to balance reliability, diversity, and scalability, leading to more stable and effective collaborative edge computing environments.

QoS Metrics and Trust-QoS Relationship

While CEC-Trust is designed as a QoS-aware metric, we complement its evaluation with explicit service-level QoS indicators to better assess the practical impact of the proposed approach. Specifically, we consider the task success rate and the average task completion time, which are standard metrics in Collaborative Edge Computing. Figure 11 shows the task success rate achieved by different Pool Formation strategies. The results indicate that pools formed using the full CEC-Trust metric achieve a higher success rate compared to trust-only, greedy trust aggregation, behavior-only, and random baselines. While the behavior-only strategy provides limited improvement over random assignment, the combination of trust dynamics and behavioral complementarity leads to significantly more reliable collaborative task execution. This improvement reflects more reliable collaborations and fewer failed task offloading attempts. Figure 12 reports the average task completion time. The proposed approach consistently yields lower completion times than all baseline strategies. Trust-only and greedy trust aggregation provide moderate improvements, whereas the behavior-only strategy shows only marginal gains over random assignment, confirming that trust-aware pool selection is essential to effectively reduce execution delays and contention. Overall, these results show a clear correlation between Average

CEC-Trust and service- level QoS metrics. Higher trust values, when combined with behavioral complementarity, are associated with more reliable and efficient task execution, supporting the claim that CEC-Trust-driven Pool Formation leads to tangible QoS improvements.

Experimental Scope and Limitations

The experimental evaluation presented in this work is intended to validate the effectiveness and robustness of the proposed CEC-Trust metric and Pool Formation algorithm, rather than to provide an exhaustive comparative benchmark. Given the decentralized nature of the approach and the heterogeneity of existing trust and collaboration frameworks, direct comparisons and ablation studies are left as future work. Future experimental extensions will include comparative evaluations with alternative trust-based approaches, sensitivity analysis of key parameters, and ablation studies to further quantify the contribution of individual components of the model. It is worth noting that the experiments involving up to 16,000 ED nodes implicitly validate the scalability of the proposed approach. Despite the large system size, the PF algorithm converges within a limited number of epochs and maintains a low variance of CEC-Trust across pools, confirming its suitability for large-scale and heterogeneous edge environments. It is also worth highlighting that the experimental evaluation focuses on service-level QoS, as reflected by trust-based performance indicators, rather than on explicit network-layer metrics such as latency or throughput. This choice allows the analysis of collaborative behavior and pool formation independently from specific deployment technologies.

Control Traffic and Convergence

To characterize operational overhead, we measure control traffic and convergence of the PF procedure. For each epoch, we record: *(i)* the average number of control messages per ED node, *(ii)* the estimated bytes exchanged, computed from message payload definitions and their estimated sizes reported in Table 5, and *(iii)* convergence time, defined as the minimum number of epochs after which the Average CEC-Trust changes by less than ϵ for w consecutive epochs. We report these measurements for different system sizes and pool configurations. The results confirm that control traffic per node remains bounded and that PF converges within a limited number of epochs, supporting the scalability of the distributed protocol.

Limitations and Robustness Against Advanced Adversarial Behaviors

The experimental evaluation focuses on unreliable and malicious behaviors at the individual ED node level. More sophisticated adversarial strategies, such as Sybil attacks, coordinated collusion, and targeted attacks against the Pool Repository (PR), are not explicitly simulated. In the following, we discuss the expected behavior of the proposed framework under these threats and outline practical countermeasures.

Sybil attacks

In a Sybil attack, a single adversary creates multiple identities to manipulate trust or reputation. CEC-Trust mitigates such attacks through bounded reputation sampling (τ_{sample}), interaction-based trust weighting, and temporal decay, which limit the influence of newly created or short-lived identities. In practical deployments, Sybil resistance can be further strengthened by introducing identity costs (e.g., registration fees, hardware-backed identities, or rate-limited admission), which are orthogonal to the trust model and compatible with the proposed framework.

Coordinated collusion

Colluding ED nodes may attempt to inflate mutual trust through coordinated positive feedback. The proposed trust model reduces the impact of such behavior by weighting direct experience over indirect reputation and by incorporating behavioral consistency through time. Moreover, bounded sampling prevents a colluding group from dominating the reputation space unless it controls a significant fraction of the system. Detecting highly sophisticated collusion patterns (e.g., long-term coordinated strategies) remains a challenging problem and represents an important direction for future work.

Attacks against the Pool Repository

The PR represents a potential attack surface if implemented as a single centralized service. As discussed in the system model, PR is intended as a logical component that can be realized through replication, authentication of

Table 5 Estimated control-message payload sizes used for byte overhead estimation

Message type	Main fields	Estimated bytes
PoolQuery	Requester id (8), optional filters (e.g., max size) (8), timestamp (8), header (24)	≈ 48
PoolReply	k pool ids ($8k$), optional metadata summary per pool (e.g., avg trust, size) ($8k$), timestamp (8), header (24)	$\approx 32 + 16k$
PoolJoinRequest	Requester id (8), pool id (8), profile summary (e.g., BT_i) (8), timestamp (8), header (24)	≈ 56
PoolJoinReply	Pool id (8), accept/reject (1), optional role (1), padding (6), timestamp (8), header (24)	≈ 48
TrustQuery	Requester id (8), target id (8), timestamp (8), header (24)	≈ 48
TrustReply	Opinion value (8), confidence (4), padding (4), timestamp (8), header (24)	≈ 48

updates, and access control. Poisoning attacks can be mitigated by requiring signed pool updates and cross-checking information across replicas, while denial-of-service attacks can be addressed through rate limiting and federated deployment. Importantly, PR does not make allocation decisions, so its compromise does not directly affect trust computation or task execution. Overall, while the proposed framework does not claim complete resistance to all advanced adversarial strategies, its design incorporates several mechanisms that limit their effectiveness. A comprehensive experimental evaluation of Sybil and large-scale collusion attacks is left as future work.

Related Work

Edge Computing is a well-established key solution for reducing latency and improving the quality of application services, by pushing computational and storage capabilities closer to the end users [30, 31]. However, given the heterogeneous and distributed nature of edge resources, it could be necessary to coordinate several nodes and realize forms of cooperative mechanisms in order to realize individual goals [32]. In such a perspective, it is necessary to build a comfortable and/or collaborative space where to interact and to collaborate with reliable actors: therefore, trust and reputation systems, as well as group formation processes, represent foundational enablers to tackle all those problems rising from the presence of malicious nodes, low quality services, communication network overload, and computational, storage or power resources unavailability typically affecting edge servers or mobile nodes. A consolidated solution for obtaining more stable and reliable performance, as well as for making systems more resilient to adverse situations, is just to consider together the trustworthiness of actors and forming groups of participants sharing the same goals [33]. In particular, the relevance of cooperation with reliable counterparts in bringing services closer to the end user, allowing to limiting latency and alleviating network congestion issues, has been well highlighted by Li et al. [19].

The creation of reliable collaborative space in CEC environments can take various forms. One of this approaches considers the coalition game theory [34], in which the interactions occurring among nodes are modeled as cooperative competitions and where the decision to join or not with a group is ruled from the expected benefits in terms of service quality, which, in our context, often means accessibility to computational resources and load balancing between the nodes belonging to the group. For instance, Chen and Xu [20] designed an ultra-dense cellular network scenario in which Small Base Stations, acting as both buyers and sellers, are encouraged to form coalitions and cooperate to maximize the overall utility. The authors applied coalition game theory, implementing a distributed algorithm to form coalition based on a trusted social network, where payments serve as incentives. Incorporating trust contributes to enhance coalition stability, as also confirmed in [35]. The experimental results reported by Chen and Xu demonstrated that, within the SBS network, the proposed solution is able in both balancing load distribution and mitigating safety risks.

In group formation processes, reliability and reputation metrics, often joined in a synthetic measure simply named trust, are commonly used to support the selection of trustworthy nodes (i.e., services, groups, etc., in a cooperative fruition [36–38]). In fact, in highly dynamic systems like mobile environments, it is frequently necessary to evaluate past behaviors or feedback through local or global approaches, for instance, before of a node affiliation to a group, also in order to assess the mutual benefit of both the node and the group. Approaches based on fuzzy logic or multi-criteria aggregation techniques have resulted effective in integrating heterogeneous parameters, such as performance, SLA compliance, and security issues, into a single trustworthiness value that can be exploited in decision-making processes.

In detail, in the formation of collaborative groups in Edge Computing environments, fuzzy logic-based models have been proposed to effectively manage uncertainty, which is an intrinsic factor in behavioral evaluation and QoS measures to constitute cooperative groups. Xia et al. [21] proposed the Lightweight Fuzzy Collaborative Trust Evaluation Model (LFCTEM), which aggregates a few multiple fuzzy factors to evaluate the reliability of edge nodes via group nodes. Simulations show that this solution limits evaluation errors and improves cooperation efficiency. However, in the Edge/Fog domain many studies relying on fuzzy logic aim to identify the most suitable node or server for service delivery based on trust and reputation measures. Although not always explicitly stated, these same measures could also be used to form cooperative groups of nodes or clusters. For instance, Kesarwani and Khilar [22] introduce a fuzzy system for calculating trust that includes factors such as performance and scalability, and uses fuzzy clustering to aggregate node ratings, a principle that can be transferred to CEC in a relatively easy way. Similarly, Monir et al. [23] describe a three-level framework that integrates SLAs, computational performance, and violations into trust calculation using fuzzy logic.

In a recent paper, Messina et al. [24] proposed to form teams of smart objects in a Mobile Edge Computing scenario involving IoT-based Connected Vehicles. Teams formation takes into account quality measures, referring to the

quality of services provided by CVs and their trustworthiness. The experiment witnessed the advantages of adopting the designed mechanism for the framework. In domains such as Cloud, IoT, and Fog trust and reputation plays a key role in forming reliable groups. Fortino et al. in several studies, among which [39–41], investigated different scenarios to form agent-based groups by adopting trust and reputation metrics to analyze different properties as competitiveness, clustering, and resilience. To improve the service quality in mobile cloud-edge computing networks, most of the existing approaches do not consider the threat posed by dishonest edge servers. To address this issue, in [25] has been designed a Trust-based Computation Offloading framework. This framework evaluates costs and revenues, and employs a non-linear decomposition method to efficiently derive near-optimal solutions. In addition, a two-tier trust evaluation mechanism is introduced to obtain accurate trust values by adopting an association scheme. Experimental results show that the proposed approach outperforms benchmarks, with a marginal loss in terms of performance. The authors of [26] describe a transport protocol, for distributed and secure public CVs infrastructures, which has been designed together with a trust model combining indirect and direct measures. RTSP involves decentralized and centralized nodes, and in a modular architecture a cluster/group formation process by trust, in order to enable CV to different services avoiding the risks due to safety vulnerabilities.

In the latest years some researchers have considered the integration of blockchain with Federated Learning (FL) and Edge Computing to realize trust-centric, decentralized, and resilient systems. In the detail, in these systems the role of a blockchain is due to its nature of chronologically ordered chain of encrypted data blocks, which are validated and verified through distributed consensus, and replicated across multiple peer-to-peer hosts in a permanent, immutable, and transparent manner [42]. Because all local ledger copies remain synchronized, the blockchain cannot be easily controlled, altered, or deleted by any single entity. By adopting a distributed coalition formation architecture and a reputation-based coalition game model, in [27] a secure and scalable sharding is created in blockchain networks supported by Edge/IoT infrastructures without requiring a central control. Node's reputation guides the selection of shard participants and the assignment of their roles. The reputation update mechanism mitigates Sybil attacks and selects trusted groups for consensus. Experimental results show improvements in consensus security, throughput, and resilience compared to schemes without reputation, demonstrating the feasibility of the approach in such scenarios with distributed computational capabilities. The authors of [28] propose an editable blockchain scheme for Mobile Edge Computing environments, which incorporates reputation measures and reputation consensus to allow legitimate blockchain nodes' activities. This system uses one-way trapdoor functions and a modified Merkle tree structure to maintain chain integrity after each operation. In this framework coalitions, where leaders and validator nodes are chosen by their reputation are formed, it is similar to forming trusted subgroups within the network. Experimental results show that modification/deletion operations can take place without breaking the blockchain structure and with accurate verification through consensus. Finally the authors of [29] propose a mechanism for Federated Learning that integrates reputation metrics and fairness criteria to select trustworthy clients and aggregate updates with weights proportional to reputation. Rotation among clients is enforced to avoid monopoly by highly trustworthy clients. Reputation guides the formation of hierarchical groups (or client clusters), aimed at improving the quality of the FL service, which perform local aggregations to reduce traffic with the central server, thereby reducing the load on the network, improving robustness, and reducing the overall latency of training, as demonstrated by their experiments.

In summary, the presented literature highlights several paradigms to enhance effective forms of collaboration in Edge Computing environments based on trust and/or reputation measures, and/or group formation processes, among those that, in our opinion, coming most interesting, with respect to the material presented in this paper. This overview gives evidence that, for ensuring robust collaborative decision-making in heterogeneous and dynamic environments, *(i)* architectures should implement distributed collaboration mechanisms for load management and QoS maintenance; and *(ii)* trust and reputation are key components in selecting reliable nodes and creating stable and secure groups. These aspects (i.e., approach, context, goals, and main differences with respect to our work) are condensed in Table 6.

We highlight that, differently from most existing trust and reputation models for Edge Computing, the proposed approach explicitly integrates trust evaluation with behavioral complementarity and decentralized pool formation. While several works focus on trust-based node selection, security enforcement, or centralized coalition management, CEC-Trust is designed to directly support collaborative task offloading by quantifying the mutual QoS benefits of cooperation between edge domains. Moreover, the proposed model jointly considers historical resource sharing behavior and dynamically evolving trust relationships, enabling edge domains to autonomously form QoS-oriented pools without relying on centralized control or static group configurations. This makes CEC-Trust particularly suitable for large-scale, heterogeneous, and dynamic collaborative edge environments, where both trustworthiness and service quality must be continuously assessed and balanced. Table 6

Table 6 Comparison of representative trust and reputation approaches for group formation in Edge Computing environments

Work	Approach	Context / goal	Main differences with our work
Cao et al. [19]	Multi-criteria	Minimizing latency and energy consumption	Peers' trust is considered only implicitly
Chen and Xu [20]	Coalitional game + payment scheme	Coalition formation to maximize efficiency and utility	Coalitions built on trust measures, costs, and fees data
Xia et al. [21]	Fuzzy logic + trust decay mechanism	Edge, Cloud/ Trust measures to promote cooperation with reliable mobile devices	Group affiliation relies on device position without considering trust
Kesarwani and Khilar [22]	Fuzzy logic	Cloud/Improving access security by evaluating actors' trust	Only two groups of honest and malicious actors are formed
Mansour et al. [23]	Multi-criteria + three tiers trust method	Node trustworthiness computation based on SLA compliance, processing performance and violations measures	Centralized approach supported by a trusted cloud service manager
Messina et al. [24]	Multi-criteria	Dynamic formation of temporary trust and reputation based teams	Service and resource quality considered in trust computation
Wang et al. [25]	Multi-criteria + two tiers trust method	Cloud/Assigning offloading tasks to honest edge servers	Trust is referred to the only edge servers
Peter and Rani [26]	Direct and Indirect trust	Cloud, IoT/Vehicle traffic control and emergency assistance evaluation	Trust is not used in the group formation process
Asheralieva and Niyato [27]	Reputation + coalition game	IoT/Secure and scalable sharding in blockchain networks	Reputation guides the assignment of nodes' role + blockchain
Tang et al. [28]	Reputation + one-way trapdoor functions + Merkle tree	Choosing leaders and validators in coalitions	Trust is not used in the group formation process
Guo et al. [29]	Reputation + fairness criteria	FL/Selection of trustworthy clients and aggregation ruled by reputation	Groups have a hierarchical Structure

summarizes representative trust- and reputation-based approaches for collaborative Edge Computing, highlighting their main objectives and limitations. Most existing solutions focus either on trust evaluation for security purposes or on centralized coalition and task allocation mechanisms. In contrast, the proposed CEC-Trust framework explicitly targets decentralized pool formation driven by QoS-aware trust and behavioral complementarity. Furthermore, unlike approaches that rely on static coalitions, centralized control, or application-specific assumptions, CEC-Trust is designed to operate in large-scale and heterogeneous environments through local and incremental trust updates. This comparative analysis clarifies the unique contribution of the proposed approach and its suitability for dynamic Collaborative Edge Computing scenarios.

Conclusions

In this work, we presented a distributed, trust-based approach to support Collaborative Edge Computing scenarios, where task offloading requests are handled by multiple edge servers belonging to different edge domains. Because selecting a suitable collaboration partner is crucial for each edge domain, our approach analyzes past behaviors of edge domains (considering resources shared and requested, as well as task characteristics) and models trust relationships among edge servers. We combined these aspects into an integrated metric, named *CEC-Trust*, which quantifies the advantages of joining a pool of edge domains for collaborative task offloading. Building on this metric, we designed a fully decentralized procedure, driven by CEC-Trust, to form and maintain pools of edge domains. Simulation-based experiments demonstrated that our procedure, together with the proposed measures, significantly improves the overall performance and robustness of edge servers in CEC scenarios. In particular, we measure the contribution of the decentralized PF procedure to the composition of the pools of ED nodes. The collected results have shown that, within a few iterations, the PF algorithm introduces a relevant increase of about 40% of the CEC-Trust measure of the pools compared to a purely random assignment of ED nodes to the pools. Since the CEC-Trust measure indirectly reflects the measured level of QoS within the collaborations among the ED nodes, these results highlight the effectiveness of leveraging trust and historical behavior to guide dynamic and decentralized pool formation in heterogeneous edge environments.

At the same time, although very promising, to more comprehensively validate the effectiveness of the proposed approach, some work still needs to be done.

Although the simulation-based evaluation demonstrates the effectiveness and robustness of the proposed approach under controlled and large-scale scenarios, we acknowledge that real-world deployments may introduce additional challenges, such as network instability, hardware heterogeneity, and workload variability. Therefore, as part of our future work, we plan to validate the proposed CEC-Trust model

Table 7 Simulation parameters and configuration values

Parameter	Value	Description
N	$\{1{,}000, 5{,}000, 16{,}000\}$	Number of ED nodes
N_{MAX}	10	Max pools joined per ED node
N_{MIN}	2	Max pools joined per ED node
K_{MAX}	80	Max ED nodes per pool
K_{MIN}	10	Max ED nodes per pool
τ_{sample}	125	Reputation sampling size
ΔT	10	Freshness threshold (epochs)
I_{max}	20	Interaction saturation threshold
α	0.5	Reliability update weight
β	Dynamic	Trust mixing weight (Eq. X)
Initial trust	0.5	Initial trust for unknown nodes
Epochs	50, 000	Simulation length
Task arrivals	Poisson (λ)	Task generation process
Behavior BT_i	Uniform [0, 1]	Behavioral tendency distribution
Malicious ratio	$\{10\%, 50\%\}$	Fraction of unreliable/malicious ED nodes
Random seed	Fixed	Fixed seed per experiment

and the PF algorithm using real-world datasets and experimental testbeds, including IoT-enabled and smart city Edge Computing infrastructures. This will allow us to further assess the applicability and performance of the approach in realistic operational environments.

Future work will focus on a progressive real-world validation of the proposed approach. An initial phase will exploit publicly available Edge and IoT datasets to map task execution outcomes and resource usage to trust feedback, enabling the calibration of the CEC-Trust metric under realistic workload traces. A second phase will involve pilot deployments on small-scale edge testbeds, such as container-based platforms or campus-level infrastructures, where ED nodes correspond to virtualized services or edge gateways. These deployments will allow the evaluation of service-level QoS metrics, including task completion time, success rate, and control traffic, under realistic network conditions, node churn, and heterogeneous resource capacities. Finally, scenario-aligned validation in smart city and IoT-enabled transportation contexts will be considered to assess long-term service-level QoS, collaboration stability, and trust dynamics in the presence of noisy feedback and partial observability. These experimental phases will complement the simulation-based results presented in this work and provide empirical evidence of the applicability of the proposed CEC-Trust-driven Pool Formation mechanism.

Appendix A Reproducibility and Simulation Configuration

This appendix provides all the information required to reproduce the experimental results presented in the paper. We report simulation parameters, random seeds, and behavioral distributions used throughout the experiments.

A.1 Simulation Parameters

Table 7 summarizes all the parameters used in the simulation environment, including trust model parameters, PF constraints, and workload-related settings.

A.2 Randomization and Seeds

All stochastic components of the simulation (node behavior generation, malicious behavior activation, and interaction scheduling) are driven by pseudo-random number generators initialized with fixed seeds. Unless otherwise stated, the same seed is used across all baseline comparisons to ensure fair and repeatable results.

A.3 Behavioral and Adversarial Models

Normal ED node behaviors are generated according to predefined distributions described in Table 7. Malicious behaviors follow scenario-specific strategies (e.g., unreliable execution, selective misbehavior) activated with configurable probabilities. More sophisticated adversarial strategies are discussed in Section Related Work and considered as future work.

Funding Open access funding provided by Università degli Studi di Catania within the CRUI-CARE Agreement. This work has been partially supported by the project "Pia.ce.ri linea 2 2024-2026", granted by the University of Catania, Italy.

Data Availability No new data were created in this study. Data sharing is not applicable to this article.

Declarations

Conflict of interest The authors declare that they have no Conflict of interest.

Ethical Approval Not applicable.

References

1. Mach P, Becvar Z. Mobile edge computing: A survey on architecture and computation offloading. IEEE Communications Surveys & Tutorials. 2017a;19(3):1628–56. https://doi.org/10.1109/COMST.2017.2682318.
2. Haibeh LA, Yagoub MC, Jarray A. A survey on mobile edge computing infrastructure: Design, resource management, and optimization approaches. IEEE Access. 2022;10:27591–610.
3. Shi W, Cao J, Zhang Q, Li Y, Xu L. Edge computing: Vision and challenges. IEEE IoT Journal. 2016;3(5):637–46.
4. Zhang X, Chen H, Zhao Y, Ma Z, Xu Y, Huang H, et al. Improving cloud gaming experience through mobile edge computing. IEEE Wirel Commun. 2019;26(4):178–83.
5. Ma W, Mashayekhy L. Video offloading in mobile edge computing: Dealing with uncertainty. IEEE Transactions on Mobile Computing (2024)
6. Siriwardhana Y, Porambage P, Liyanage M, Ylianttila M. A survey on mobile augmented reality with 5g mobile edge computing: Architectures, applications, and technical aspects. IEEE Communications Surveys & Tutorials. 2021;23(2):1160–92.
7. Takahashi N, Tanaka H, Kawamura R. Analysis of process assignment in multi-tier mobile cloud computing and application to edge accelerated web browsing. In: 2015 3rd IEEE International Conference on Mobile Cloud Computing, Services, and Engineering, pp. 233–234 (2015). IEEE
8. Tran TX, Hajisami A, Pandey P, Pompili D. Collaborative mobile edge computing in 5g networks: New paradigms, scenarios, and challenges. IEEE Commun Mag. 2017;55(4):54–61.
9. Wang L, Jiao L, He T, Li J, Bal H. Service placement for collaborative edge applications. IEEE/ACM Trans Networking. 2020;29(1):34–47.
10. Wang K, Yin H, Quan W, Min G. Enabling collaborative edge computing for software defined vehicular networks. IEEE Network. 2018;32(5):112–7.
11. Zhen Q, Shoushuai H, Hai W, Yuben Q, Haipeng D, Fei X, et al. Air-ground collaborative mobile edge computing: Architecture, challenges, and opportunities. China Communications. 2024;21(5):1–16.
12. Wang S, Tu G.-H, Ganti R, He T, Leung K, Tripp H, Warr K, Zafer M. Mobile micro-cloud: Application classification, mapping, and deployment. In: Proc. Annual Fall Meeting of ITA (AMITA) (2013)
13. Lobillo F, Becvar Z, Puente M.A, Mach P, Presti F.L, Gambetti F, Goldhamer M, Vidal J, Widiawan A.K, Calvanesse E. An architecture for mobile computation offloading on cloud-enabled lte small cells. In: 2014 IEEE Wireless Communications and Networking Conference Workshops (WCNCW), pp. 1–6 (2014). IEEE
14. Zhou X, Ke R, Yang H, Liu C. When intelligent transportation systems sensing meets edge computing: Vision and challenges. Appl Sci. 2021;11(20):9680.
15. Wooldridge M, Jennings N. Intelligent agents: Theory and practice. The knowledge engineering review. 1995;10(02):115–52.
16. Abdul-Rahman A, Hailes S. Supporting trust in virtual communities. In: HICSS 2000: Proc. of the 33rd Hawaii Int. Conf. on System Sciences, vol. 6. IEEE Computer Society., ??? (2000)
17. Ramchurn S, Huynh D, Jennings N. Trust in multi-agent systems. Knowl Eng Rev. 2004;19(1):1–25.
18. Zacharia G, Maes P. Trust management through reputation mechanisms. Applied Artificial Intell. 2000;14(9):881–907.
19. Li T, Wang K, Xu K, Yang K, Magurawalage CS, Wang H. Communication and computation cooperation in cloud radio access network with mobile edge computing. CCF Transactions on Networking. 2019;2(1):43–56.
20. Chen L, Xu J. Socially trusted collaborative edge computing in ultra dense networks. In: Proceedings of the Second ACM/IEEE Symposium on Edge Computing, pp. 1–11 (2017)
21. Xia G, Yu C, Chen J. A fuzzy-based co-incentive trust evaluation scheme for edge computing in ceec environment. Appl Sci. 2022;12(23):12453.
22. Kesarwani A, Khilar PM. Development of trust based access control models using fuzzy logic in cloud computing. Journal of King Saud University - Computer and Information Sciences. 2022;34(5):1958–67. https://doi.org/10.1016/j.jksuci.2019.11.001.
23. Mansour MB, Abdelkader T, Hashem M, El-Horbaty ESM. An integrated three-tier trust management framework in mobile edge computing using fuzzy logic. PeerJ Computer Science. 2021;7:700.
24. Messina F, Rosaci D, Sarnè GML. Forming teams of smart objects to support mobile edge computing for iot-based connected vehicles. Appl Sci. 2025;15(17):9483.
25. Wang J, Li Z, Liu H, Qiu T, Luo H. A trust-based computation offloading framework in mobile cloud-edge computing networks. IEEE Transactions on Mobile Computing (2025)
26. Peter MN, Rani MP. V2v communication and authentication: the internet of things vehicles (iotv). Wireless Pers Commun. 2021;120(1):231–47.
27. Asheralieva A, Niyato D. Reputation-based coalition formation for secure self-organized and scalable sharding in iot blockchains with mobile-edge computing. IEEE Internet Things J. 2020;7(12):11830–50.
28. Tang Y, Wu S, Wang X. Redactable blockchain trust scheme based on reputation consensus for mec. Comput Intell Neurosci. 2022;2022:3269445. https://doi.org/10.1155/2022/3269445.

29. Guo C, Zhang X, Zhang L, Gong C, Xu H, Han Z. Reputation-based federated learning algorithm for fairness and security in internet of vehicles. IEEE Internet of Things Journal (2025)
30. Spinelli F, Mancuso V. Toward enabled industrial verticals in 5g: A survey on mec-based approaches to provisioning and flexibility. IEEE Communications Surveys & Tutorials. 2020;23(1):596–630.
31. Abbas N, Zhang Y, Taherkordi A, Skeie T. Mobile edge computing: A survey. IEEE Internet Things J. 2017;5(1):450–65.
32. Cao X, Wang F, Xu J, Zhang R, Cui S. Joint computation and communication cooperation for mobile edge computing. In: 2018 16th International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt), pp. 1–6 (2018). IEEE
33. Mach P, Becvar Z. Mobile edge computing: A survey on architecture and computation offloading. IEEE communications surveys & tutorials. 2017b;19(3):1628–56.
34. Hajduková J. Coalition formation games: A survey. International Game Theory Review. 2006;8(04):613–41.
35. De Meo P, Messina F, Rosaci D, M. L. Sarné GML. Forming time-stable homogeneous groups into online social networks. Information Sciences **414**, 117–132 (2017)
36. Fortino G, Fotia L, Messina F, Rosaci D, Sarné GML. Trust and reputation in the internet of things: State-of-the-art and research challenges. IEEE Access. 2020;8:60117–25.
37. Shao T, Yang X, Wang F, Yan C, Luhach AK. Trusted service evaluation for mobile edge users: Challenges and reviews. Complexity. 2021;2021(1):2227459.
38. Fotia L, Delicato F, Fortino G. Trust in edge-based internet of things architectures: state of the art and research challenges. ACM Comput Surv. 2023;55(9):1–34.
39. Fortino G, Fotia L, Messina F, Rosaci D, Sarnè GML. A social edge-based iot framework using reputation-based clustering for enhancing competitiveness. IEEE Transactions on Computational Social Systems. 2022;10(4):2051–60.
40. Fortino G, Fotia L, Messina F, Rosaci D, Sarné GML. Trusted object framework (tof): A clustering reputation-based approach using edge computing for sharing resources among iot smart objects. Computers & Electrical Engineering. 2021;96:107568.
41. Fortino G, Messina F, Rosaci D, Sarnè GML. Resiot: An iot social framework resilient to malicious activities. IEEE/CAA Journal of Automatica Sinica. 2020;7(5):1263–78.
42. Pilkington M. Blockchain technology: principles and applications. In: Research Handbook on Digital Transformations, pp. 225–253. Edward Elgar Publishing, ??? (2016)