Coalitional Federated Learning: Improving Communication and Training on Non-IID Data With Selfish Clients

Sarhad Arisdakessian, Omar Abdel Wahab, Azzam Mourad, *Senior Member, IEEE*, and Hadi Otrok, *Senior Member, IEEE*

***Abstract*—In this article, we propose a new paradigm of Federated Learning (FL) for Internet of Things (IoT) devices called *Coalitional Federated Learning*. The proposed paradigm aims to address the challenges of (1) non-independent and identically distributed (non-IID) data across clients; (2) communication overhead due to the large number of messages exchanged between the server and clients; and (3) selfish clients that seek to obtain the latest global models without efficiently contributing to the training of the FL model. Our novel paradigm consists of three main components, i.e., (1) client-to-client trust establishment mechanism that relies on subjective and objective sources to enable clients to establish credible trust relationships toward one another; (2) trust-enabled coalitional game to enable clients to autonomously form harmonious coalitions of FL trainers; and (3) coalitional federated learning in which multiple local aggregations take place at the level of each coalition to mitigate the problems of non-IID data and communication bottleneck. Extensive experiments suggest that our solution outperforms both the standard vanilla FL approach and one state-of-the-art trust-based FL approach in terms of increasing the accuracy of the global FL model and decreasing the presence of selfish devices participating in the training.**

***Index Terms*—Client selection, communication efficiency, federated learning, non-IID data, security, selfish client.**

I. Introduction

FEDERATED learning (FL) has recently been brought forward to enable privacy-oriented machine learning processes over a large set of edge devices, often referred to as *clients*. Instead of requiring these devices to offload their data to a central cloud server, FL proposes to equip these devices with machine learning processing capabilities, enabling them to run the data analytics locally. In FL, two types of machine learning models can be distinguished, i.e., local and global. The *local model* is the result of the local training that happens at the level of each device. On the other hand, the *global model* is the result of aggregating the set of local models by the federated server.

A. Problem Statement

Despite the enormous advantages of federated learning in terms of privacy preservation and communication efficiency, this emerging paradigm faces some serious challenges [1]. In fact, the data across the client devices in FL are highly non-independent and identically distributed (non-IID). This is because each device usually stores the specific activities of its owner, meaning that the local data of an individual device does not represent the overall data distribution of the environment. Several approaches have been proposed to combat the non-IID data challenge in FL such as: flexible client participation, data augmentation, federated transfer learning, parameter tuning and client clustering [2]. Despite their importance, most of these approaches require updating the internal structure and/or optimization functions in FL, which might entail additional complexities.

Another fundamental challenge of FL is the communication bottleneck. In fact, communication is essential in federated learning and can take two forms, i.e., downstream communication and upstream communication. The first type of communication happens when clients download the current training model from the central server. The second type of communication describes the action of clients uploading the locally updated model to the central server. Nonetheless, the limited bandwidth of the clients, coupled with their relatively far distance from the federated server are the main reasons for communication bottleneck. To address the communication challenges in FL, several approaches such as periodic aggregation, dynamic aggregation, compression, over-the-air-computation have been proposed [2]. In spite of the importance of these approaches in reducing the communication rounds, they might still lead to communication bottlenecks at the central server per communication round. This is because the server still experiences heavy model weight updates during some training epochs.

To address the statistical and communication challenges, we propose in this work a new paradigm for FL, which we refer to as *Coalitional Federated Learning*. The main idea of our new

Manuscript received 14 August 2022; revised 22 January 2023; accepted 15 February 2023. Date of publication 28 February 2023; date of current version 8 August 2023. This work was supported in part by the Natural Sciences and Engineering Research Council of Canada under Grant RGPIN-2020-04707, in part by Polytechnique Montréal, in part by the Lebanese American University, and in part by Khalifa University. Recommended for acceptance by J. Kolodziej. *(Corresponding author: Omar Abdel Wahab.)*

Sarhad Arisdakessian and Omar Abdel Wahab are with the Department of Computer Engineering and Software Engineering, Polytechnique Montréal, Montréal, QC H3T 1J4, Canada (e-mail: sarhad.arisdakessian@polymtl.ca; omar.abdul-wahab@polymtl.ca).

Azzam Mourad is with the Department of Computer Science and Mathematics, Lebanese American University, Beirut 1102-2801, Lebanon (e-mail: azzam.mourad@lau.edu.lb).

Hadi Otrok is with the Department of EECS, Khalifa University, Abu Dhabi 127788, UAE (e-mail: hadi.otrok@ku.ac.ae).

Digital Object Identifier 10.1109/TSC.2023.3246988

paradigm is to enable the client devices to autonomously form coalitions. At the level of each coalition, a coalition master (CM) is elected. This allows us to carry out multiple local aggregations at the level of each coalition by the coalition master. Only locally aggregated models are shared with the federated server by each coalition master. This minimizes the communications between the client devices and the federated server. To mitigate the severity of the non-IIDness of the data, we enable in our solution each coalition master to share a small subset of data with the clients belonging to its coalition as proposed in [3]. This coalition-level data, added to each client's local data, would make these local data less skewed and biased. A key enabler for this idea is our coalition formation solution which allows the client devices to autonomously select their work partners with which they are willing to work.

Yet, one major challenge to the *Coalitional Federated Learning* architecture and to FL in general is the presence of malicious clients. These malicious clients can be either *active* or *passive*. Active malicious clients are those devices that actively try to harm the FL environment using some active attacks. On the other hand, passive malicious clients do not seek to intentionally harm the system, but are rather selfish parties that prefer not to waste their own resources contributing to the FL model. Such clients would train the FL model only on some small subsets of their local data. The objective is to save on their own resources such as CPU, RAM and bandwidth, while still receiving the latest versions of the global model. Such a misbehavior would make the convergence of the global model to the desired accuracy quite slower. It would hence considerably increase the number of communication rounds that are needed to reach the desired convergence.[1] In this work, we are interested in the *passive malicious* client devices and propose a mechanism to identify and avoid them. Such devices represent a big threat to our architecture as they would seek to join strong coalitions to benefit from the powerful models that are generated by these coalitions without significantly contributing to the training of these models.

B. Contributions

In this paper, we propose a three-fold federated learning solution to address the challenges of non-IID data, communication bottleneck and presence of selfish client devices. In the first phase, we propose a client-to-client trust established mechanism which utilizes both subjective and objective sources of trust to further improve the credibleness of the trust scores and minimize biases. In order to achieve so, we first model the relationships among client devices in FL as a social network and propose a discovery algorithm based on which the devices can enquire about other's behavior. This phase is then complemented with an objective trust establishment method which monitors the resource utilization of the client devices. Under-utilization of resources could be an indication that the underlying device is not using its entire local data for the training or that the device is not performing appropriate computations on the data to save on resources. We aggregate both the subjective and objective trust metrics using the Bayesian inference approach.

In the second phase, we propose a trust-augmented coalition formation game which enables the clients to form trustworthy coalitions. Our method is entirely autonomous in the sense that it enables each client to join the coalition that maximizes its trust belief. Devices can also choose to leave their current coalitions and join other ones in order to maximize their trust. The objective is to create harmonious coalitions of clients whose collaboration would contribute to increasing the performance of the FL process.

Having formed coalitions of clients, the last phase of our solution is the implementation of the *Coalitional Federated Learning* paradigm. Instead of selecting individual clients as is the case in traditional FL, our solution enables the server to select coalitions of clients enjoying high trust and minimizing the number of selfish members. Each coalition is represented by a coalition master which is responsible for periodically sharing a small subset of data with its coalition members to reduce the non-IIDness of the data across the the members of the same coalition. The master is also responsible for performing, from time to time, local aggregations for the local models generated by the clients in its coalition. This master then serves as the representative of its coalition through periodically sharing coalition-level aggregate model with the server for further global aggregation. This greatly contributes to reducing the communication bottleneck through avoiding pairwise communications among the federated server and each individual client.

Hereafter, the main contributions of this paper can be summarized as follows:

- We propose a trust-enabled coalitional federated learning solution. Our solution aims to mitigate the problems of non-IID data and communication bottleneck in FL, while detecting and avoiding the selfish client devices whose presence negatively affects the overall performance of FL. To the best of our knowledge, our solution is the first that jointly addresses the problems of non-IID, communication efficiency and robust client selection mechanism in federated learning.
- We devise a client-to-client trust establishment solution. We argue that client-to-client trust relationships are key enablers for any distributed communication architecture in FL. Although some trust approaches have been proposed for FL, these approaches mainly focus on the trust relationships between the clients and the server. Therefore, we believe that our trust solution would substantially advance the state-of-the-art in field of trust establishment in FL.
- We leverage a trust-enabled coalition formation game that enables autonomous client devices to form trustworthy and harmonious coalitions of workers. The proposed coalition formation method converges to both individual as well as Nash stable partitions of devices.

C. Paper Outline

In Section II, we review some relevant literature on client selection mechanisms in federated learning and explain the unique

1.In the rest of this paper, we use the term *selfish devices* to refer to passive malicious clients.

features of our solution. In Section III, we discuss the adversary model considered in this paper. In Section IV, we explain the system model and trust establishment mechanism among client devices. In Section V, we formulate the coalition formation game and discuss the corresponding preference relations among client devices. In Section V-D, we show how our proposed *Coalitional Federated Learning* approach can be implemented and provide the underlying algorithm. In Section VI, we discuss our simulation environment and the simulation results. Finally, in Section VII we derive our conclusions and highlight potential future directions.

II. Related Work

The notion of client selection in the realm of FL has been an ongoing research topic that has been addressed by the research community. In this section, we review relevant literature on the challenges of non-IID data, communication bottleneck and selfish devices as well as we survey the recent state-of-the-art in client selection in FL and highlight the key differences of our work.

A. non-IID Data

Given the heterogeneous of client devices in FL, data generated by these devices will also be heterogeneous (i.e., non-IID). Such type of data plays against the training of robust FL models as it causes delays in the desired accuracy convergence of the global model. Several works in the literature have discussed and tackled the issue of non-IID data. For example, the authors of [4] study the disadvantages of non-IID data in FL environments by conducting extensive experiments. They reiterate the fact that despite the state-of-the-art approaches that try mitigating this, non-IID data still causes a challenge for the learning accuracy in FL. The notion of non-IIDness is further divided into subcategories in [2] where the authors discuss three different sets of imbalance that the non-IIDness of the data can incur which are (1) class imbalance, (2) distribution imbalance, (3) size imbalance. Authors of [5] survey and discuss about possible ways to combat non-IIDness such as: Data-sharing, data-augmentation, knowledge-distillation, local fine-tuning, adding personalization layers, and applying multi-task learning. In [6] the authors try to combat non-IIDness by proposing a knowledge-distillation approach that fine-tunes the global model in the server by exploring the state of the local model using a generator and accordingly apply on it the knowledge-transfer towards the global model. Similarly, the authors of [7] combat non-IIDness by a proposal which forms a unified feature learning. Their proposal includes two parts. First, an adversary module which helps in decreasing the differences in feature representations between the clients, and second, a consensus losses approach which reduces the inconsistencies of the optimization functions in the proposal. Finally, the authors of [8] perform a deep experimental analysis on how non-IIDness can effect each layer pertaining to the underlying deep learning batches in FL. Their experiments highlight that by post-calibrating the classifier after the FL round, the classification performance can be enhanced.

B. Communication Bottleneck

Since FL entails many connected nodes (i.e., central server, clients) that share updates in both directions (i.e., upward, downward), FL environments are prone to communication overhead. Several works in the literature study and discuss the issue of communication bottlenecks in FL and offer improvement strategies. For example, the authors in [9] dedicate a survey entirely for this matter. They highlight possible causes of such bottlenecks and showcase possible solutions such as local updating, better client selection, reducing update rate, decentralized training, peer-to-Peer Learning, compression, quantization, and sparsification. The authors of [10] propose a compression technique that is based on standard scalar quantization as well as pruning methods which enables quantization and extreme compression for uplink during the learning rounds. Their experiments show that the proposed method can secure up to 40x compression rate versus uncompressed methods. In [11], the authors propose a novel FL approach called ProgFed that is based on progressive learning. The approach expands the type of the deep layers from shallow into a more complete model and can achieve up to 50x reduction in network communication costs. In [12], the authors propose a model-compressed technique in called GWEP based on joint quantization and pruning. Through evaluations, the authors show that their approach can achieve high convergence results in 11 times lesser communication rounds compared to baseline methods.

C. Selfish Devices

The presence of selfish nodes (also referred to as free-riders in the literature) in any federated learning environment affects negatively to the overall global model. Their presence in FL environments is not fair for other clients that contribute with their entire datasets and resources to the learning rounds. To overcome this problem, several works in the literature study and discuss the notion of selfish clients in FL and propose different possible solutions. For example, in [13], the authors evaluate and perform a theoretical and experimental analysis pertaining to the presence of selfish clients in FL. In [14], the authors highlight the detrimental effects of selfish clients and their impact on other client's long term participation in the FL rounds. They propose a method to combat this by treating the selfish behavior of the clients as infinitely repeated game and derive the optimal Nash Equilibrium. Simulation results shows that their approach can decrease the number of selfish devices tremendously. In [15], the authors propose a a defensive mechanism against selfish clients in FL called PASS (Parameter Audit-based Secure and fair federated learning Scheme). The approach relies on an contribution evaluation where non-contributing clients are left out from the learning rounds if they do not contribute above a certain threshold. Finally, in [16], the authors propose a framework that analyzes the behavior of clients in FL. They highlight the importance of sharing of data, as well as the negative impact of selfish clients in the learning rounds. Based on their findings, they propose an accuracy-shaping based mechanism; inspired from contact theory; which maximizes the data generation, while decreasing the selfishness of the clients in the environment.

D. Client Selection

Several works in the literature propose different methods of client selection in FL. For example, the authors of [17] use the devices' geographical locations and resource capabilities to derive a list of devices that can handle the tasks at a given FL round. They treat this as a maximization problem and work on a solution using heuristic approaches in which the server tries to maximize the number of devices per round. The authors of [18] propose a scheduling approach between the federated server and clients by taking into account the trust factor between them. To do this, they use the resource capabilities of the devices coupled with a Double Deep Q Learning (DDQN) based scheduling approach. However, this approach takes into account only the trust between the federated server and individual client devices, thus overlooking client-to-client trust relationships. The authors of [19] also investigate the problem of client selection inside wireless networks. They base their approach on prioritizing the clients that would have a more accurate models for the overall global model. Once clients are selected, they are given resource blocks (RB) to communicate with the federated server. The authors also make use of Artificial Neural Networks (ANN), especially with devices not selected for the learning rounds as they try to extract possible gradients from them. These gradients further help coming up with better global models. In [20], the authors highlight client selection in live and dynamic environments as they propose an on-demand client selection scheme. Their approach relies on containerization technologies to construct the environments as they tackle the problem of availability, especially in FL areas which lack enough available clients with adequate resources needed to participate in the learning rounds. Finally, in [21], the authors base their client selection mechanism on the data found at each client. They argue that the varying degrees of non-IID data at each client play a big role in the accuracy degradation of the global model. Therefore, they use weight divergence to estimate the non-IID degree of the clients. They accordingly propose a new FL algorithm called *CSFedAvg* where clients that have lower degrees of non-IID data have a preference by the federated server to get selected to train models.

Based on the above literature review, we notice that the reviewed works mainly focus on a single aspect or a component, while overlooking others, especially during the client selection process. For example, the approaches that focus on the hardware and resource aspect at the clients, neglect the quality of the data within these clients which can negatively influence the global model. Other approaches that focus on the fairness and the presence of selfish clients do not take into account the communication bottlenecks. Different from these approaches, our approach is more holistic in the sense that it tries to jointly solve the problems of non-IIDness, communication efficiency and presence of selfish clients. Contrary to current works in the literature that only consider the trust aspect between the clients and the server, our work highlights the notion of client-to-client trust in the domain of FL. We believe this is an important enabler that can derive more fair and less biased trust recommendations in distributed system environments. Therefore, we believe that these differentiating factors makes our solution more appropriate for practical federated learning scenarios.

TABLE I
LIST OF SYMBOLS AND DEFINITIONS USED IN THE PAPER

Abbreviations	
D	Dataset of a client
Q	Set of all works/tasks
q	A work/task from Q
W	Set of all weights
w	A weight from W
T	Time Slots
t	A specific time slot from T
I	Set of IoT devices
i	An IoT Device from I
Z	A coalitions structure
C	A coalition in Z
E	Set of edges between nodes
Rec	Recommendation of trust towards a device i
CM	Coalition Master
FS	Federated Server
FL	Federated Learning
P_i	Preference function of i
R	Set of all resource metrics {cpu, ram, bandwidth}
BW	bandwidth of a device
LC	Communication latency
Q1	1st quartile or 25th percentile
Q3	3rd quartile or 75th percentile
IQR	Interquartile Range
p	A value of current resource utilization of i
LL	Lower limit resource threshold
α	Total of under-utilized resources
ρ	Number of times a resource was under-utilized
ϕ	Average of under-utilizing a resource
η	Probability of under-utilizing a resource
δ	Count of type of resource under-utilized
τ_i	Objective trust value towards device i
$h_i(t)$	Historical set of coalitions for i before t

III. ADVERSARY MODEL

In this section, we explain the notion of selfish devices, explain their behavior and discuss the consequences of this behavior on the FL process. We also summarize the annotations and symbols used in this paper in Table I.

Federated learning heavily relies on clients' data to derive robust machine learning models. This implies that systems with abundant high quality data, will have better federated global models. The default federated learning approach [22] assumes that the devices selected to participate in the learning rounds are honest and willing to train on their full local datasets (or at least with the volume of data that they have advertised). In a federated learning scenario, an entire dataset is denoted by $D^Q = \{D_1^q...D_n^q\}$ where D_n^q refers to the pieces of information a client owns pertaining to a specific work or a task $Q = \{1, 2, 3..., n\}$. Therefore, FL relies completely on the dataset of the underlying devices in the learning rounds. Each device does its best to train the local model with the data it possesses by dedicating resources to it such as CPU, memory, and bandwidth. In return, these devices receive an enhanced version of the trained model, which is an aggregation of several local models into a better global one. Thus, devices benefit from each other's models in a cooperative way since each device's local model is expected to enrich the global model and make it more accurate.

However, in practical scenarios, the FL process is likely to be challenged by the existence of selfish devices, which are not willing to perform the training with their full datasets. Such selfish devices would still want to reap the benefits of the learning done by others through constantly receiving the updated versions of the global model. Selfish devices are different from *straggler devices* in FL [2], which take longer than usual to submit their model updates to the federated servers. Such devices cause significant delays in the model aggregation and can have detrimental consequences in time-critical applications, such as in the domain of autonomous vehicles, where FL can assist in providing real-time information and updates about traffic and road conditions. The main difference between straggler devices and selfish devices is that the behavior of the former devices stems from some lack in computational and/or network resources, while the behavior of the latter devices is to obtain the federated learning model without wasting their resources. Hereafter, throughout this work, we mainly focus on selfish/free-rider devices.

There are several reasons why some devices might act in a selfish manner such as:

- Some devices simply may not prioritize the federated learning tasks and prefer to dedicate their resources on other tasks (e.g., browsing, uploading videos).
- Some devices might be owned by some lazy or indifferent clients that bypass some of the FL rules, while still being able to gett the latest global model updates.
- Some devices might not be receiving proper incentives to continue their participation with their full datasets or computational/network resources.

The presence of such devices can entail negative consequences on the whole FL process, which are listed below:

- First and foremost, it would be unfair for the honest devices in the environment that are participating with their entire datasets and dedicating their resources to receive the same global model as the selfish devices. This might demotivate honest devices from participating in further rounds and might even push them to entirely leave the network.
- The convergence of the global model will be delayed, since the selfish devices are not participating with their entire datasets. Therefore, it would take longer for the global model to converge to a certain desired accuracy. This will require the honest devices to spend more resources over a larger number of communication rounds to attain the desired accuracy.

To study the negative consequences of having selfish devices, we simulate in Fig. 1 an FL environment in which we inject and

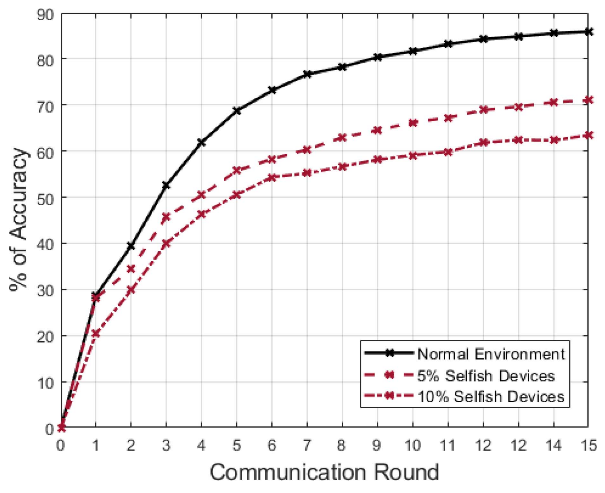


Fig. 1. Effect of selfish device on the accuracy of the global model.

vary the percentage of selfish devices. We compare the accuracy of such an environment against a fully honest FL environment wherein all clients use their entire datasets to train the FL model.

We notice from Fig. 1 how only a small percentage of 5% of selfish devices can cause a significant drop in the global model's accuracy, causing considerable convergence delays. For example, the fully honest environment consisting of 50 honest devices reached an accuracy level of 85% in 15 rounds, while the environment with 5% of selfish devices was able to reach only 70% of accuracy with the same number of rounds, thus a gap of 15% between the two environments. This gap becomes larger as the percentage of selfish devices in the environment increases. With 10% of selfish devices, the global model's accuracy was approximately 64% over 15 communication rounds, a gap of 22% with the fully honest environment. Motivated by these results, we propose in the following sections a three-fold federated learning solution to address the challenges of non-IID data, communication bottleneck in the presence of selfish client devices. We showcase a high-level architecture of our approach in Fig. 2.

IV. Client-to-Client Trust Model

In general, in the domain of federated learning, usually the central server handles the client selection process. This would mean that the server relies on its own trust knowledge of the clients for making the selections. However, this knowledge can be limited, or biased towards a certain client that can mislead the server by posing as honest while being selfish in practice. Most works in the literature mainly focus on the server-client trust aspect. This is where the notion of client-to-client trust is highlighted, as these relationships are key enablers for any distributed communication architecture in FL. This is because in addition to server's feedback regarding a client, we take the opinion of other clients in the environment that have dealt with the node in question and aggregate both trusts inputs accordingly. This decreases the possible bias towards classification of clients as honest or not as now more entities are involved

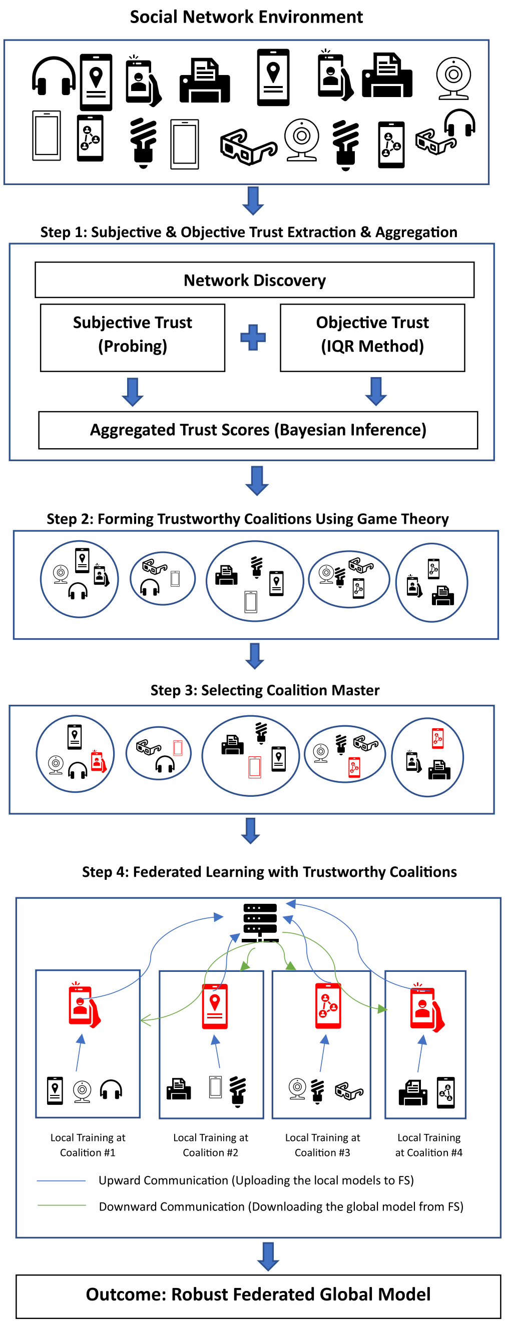

Social Network Environment
Step 1: Subjective & Objective Trust Extraction & Aggregation
Network Discovery
Subjective Trust (Probing)
Objective Trust (IQR Method)
Aggregated Trust Scores (Bayesian Inference)
Step 2: Forming Trustworthy Coalitions Using Game Theory
Step 3: Selecting Coalition Master
Step 4: Federated Learning with Trustworthy Coalitions
Local Training at Coalition #1
Local Training at Coalition #2
Local Training at Coalition #3
Local Training at Coalition #4
Upward Communication (Uploading the local models to FS)
Downward Communication (Downloading the global model from FS)
Outcome: Robust Federated Global Model

Fig. 2. A high level representation of our architecture.

in the classification process. Each client can have a separate and independent trust perspective regarding the other device. For example, a device 'A' can be highly trusted by a device 'B' because of a good past collaboration, yet not so much by another device 'C'. Hereafter, in this Section, we explain the details of our trust model which allows the clients to autonomously establish trust relationships toward one another.

A. System Model

Our system is composed of a social network of IoT devices that can be represented using a directed graph. Let $I = \{i_1 \ldots i_n\}$

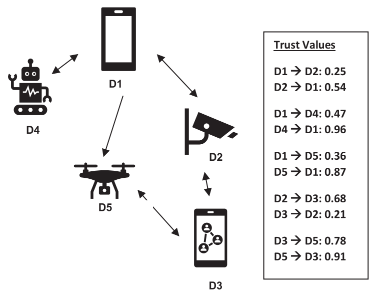


Fig. 3. A directed graph showing the trust values between the devices.

denote the finite set of the IoT devices in which each device is characterised by a set of resources such as CPU, RAM, and network bandwidth represented by $R = \{CPU, RAM, BW\}$. We model the IoT network using a directed network graph where $G = (I, E, Trust)$ represents the relationships between these devices. Each edge $(i_k, i_j) \in E$ indicates a relationship between devices i_k and i_j. Furthermore, a trust value $Trust(i_k, i_j) \neq Trust(i_j, i_k)$ is associated with each edge (i_k, i_j) to represent device i_k's trust opinion on the other device i_j. These values can exist initially or not, and can be gathered by various ways such as previous knowledge and interaction, asking other devices that have interacted with a certain device, and being neighbors. We provide an example of a social network of IoT devices with their corresponding trust values in Fig. 3.

B. Subjective Trust: Recommendation Collection

To establish trust between the devices in the network, we propose first to collect recommendations on the behavior of the devices. This is done by consulting the devices that have a certain connection and have formed a certain sense of trust (or the lack of) in regards to their neighbors with which they have interacted in the past. The advantage of such an approach is its ability to take into account the recommendations of several devices rather than a single entity that can hold a false or biased recommendation about some other peers. This type of trust calculation is known as *subjective trust*. To accomplish this, we devise a discovery algorithm (Algorithm 1) that allows devices to inquire about others from their adjacent neighbors. This is done by consulting devices that have dealt with the device in question in the past and have a certain knowledge about its behavior. The proposed Algorithm is inspired by the DFS algorithm [23], which has the advantage of being memory-efficient, considering the dense network of IoT devices that are present in an FL scenario.

The method $FIND - PATHS$ takes as inputs the a social network of IoT devices, the origin device that is enquiring about the behavior of another device, and the target device that is being inquired upon (Line 1). It returns the trust scores of the target based on the recommendations collected by the target's direct neighbours (Line 2). Initially, a stack data structure is created. The stack stores the target device as well as the path through which this device can be reached (Line 7). As long as the stack is not empty, the algorithm pops out its last element (Lines $8-9$) and loops over this element's neighbors (Line 10). If a certain

Algorithm 1: Trust Recommendation Extraction.

```
1: Input: $network$: A Social network of IoT devices
2: Input: $origin$: The IoT device asking for recommendation
3: Input: $target$: The IoT device on which $origin$ seeks to
   gather recommendations
4: Output: Trust scores of $target$
5:
6: Method: FIND-PATHS
7:      Initialize $stack$ with [($origin$, [$path$])]
8:      while $stack$ is not empty then
9:        $device$ = pop last $device$ from $stack$
10:       $path$ = pop last $path$ from $stack$
11:       for each neighbor $n$ in $network$
12:         if $n \in path$ then
13:           Continue without looping inside $path$
14:       Record the path to reach there
15:       if $n$==$target$ then
16:         Return $nextPath$
17:         Continue with the rest of $stack$
18:       Append ($n$, $nextPath$) to $stack$
19: End method
20:
21: Method: GET-TRUST-VALUES
22:     neighbours = []
23:     $allPaths$ = FIND-PATHS($origin$, $target$, $network$)
24:     for each $path$ in $allPaths$:
25:       Get neighbors via $path$
26:     return Trust-scores/recommendation from neighbors
27: End method
```

Algorithm 2: Objective Trust Calculation Algorithm.

```
1: Input: Historical values of different resources $R$ from
   the devices at previous time slots $T = \{t_1 ... t_k\}$.
2: Output: $\tau_i$: Objective trust value towards IoT device $i$
3:
4: Method: CALCULATE-OBJECTIVE-TRUST
5:     For each item in $R = [CPU, RAM, BW]$ do
6:         Calculate $Q1$, $Q3$, $IQR$
7:         Calculate $LL = Q1$ - $IQR$ x 1.5
8:       For each value point $p$ at $t$ do
9:         if $p < LL$ then
10:          $\alpha$+=$p$
11:          $\rho++$
12:      if $\rho > 0$ then
13:        $\phi$=$\alpha$ / $\rho$
14:        $\eta$=$LL$ / $\phi$
15:        $\delta++$
16:    if $|\delta| == 0$ then
17:      $\tau_i = 1$
18:    else then
19:
20:      $\tau_i = \frac{\sum_{r \in R} \eta}{\delta}$
21: End method
```

neighbor is already in the path to the device, the algorithm stops the loop (Lines 12-13). Otherwise, it stores the path on how the neighbor device was reached (Line 14). If the neighbor was the intended target (Line 15), the algorithm gets the path that enabled reaching the target (Line 16), goes back to the loop of Line 8 and continues with the rest of the elements in the stack (Line 17). Then, the algorithm appends the underlying neighbor to the stack (Line 16), so that it gets processed again using the while loop in Line 8.

In its turn, the method $GET-TRUST-VALUES$ uses the method $FIND-PATHS$ to get the reachable paths between an origin and a target device (Line 22) and accordingly retrieves the devices composing this path (Lines $23-24$). Finally, finally, the method outputs the trust values between the devices (Line 21) as outputs trust value of each target device.

C. Objective Trust: Resource Monitoring

To complement our subjective trust establishment approach in terms of minimizing the risks of collusion attacks (i.e., promoting and slandering) against certain devices [24], we propose an objective trust approach.

Our objective trust approach is based on monitoring the resource utilization of the client devices for certain FL tasks and comparing this utilization with historical utilization on similar tasks. Technically speaking, our approach capitalizes on the Interquartile Range (IQR) technique to check for any abnormal utilization of resources using two disjoint quartiles referred to as $Q1$ and $Q3$. Our motivation for choosing IQR is due to its lightweight nature as well as its minimal time and space complexity [25]. The aim here is to find those devices that are under-utilizing their resources throughout the FL process, indicating that they might be acting in a selfish manner by not dedicating enough resources to this process.

We explain the objective trust establishment process in Algorithm 2. The Algorithm takes as an input the values of the different resources metrics that are necessary for FL training such as CPU, RAM and bandwidth (Line 1). It returns τ_i, the objective trust value towards device i (Line 2). The algorithm loops over each of the resource metrics (Line 5) and computes, based on the previous utilization history of that resource metric on FL tasks with similar resource requirements as the underlying task, the values of $Q1$ (the first quartile), $Q3$ (third quartile), IQR (interquartile range), and LL (lower utilization limit) (Lines 6-7). Multiplying the IQR by a factor of 1.5 for calculating LL is inspired by Tukey analysis as explained in [26]. Any value that is found to be lower than LL is considered to be an outlier, indicating that the concerned device might be selfish. Once these values are populated, the algorithm loops over each specific value point p representing the current resource utilization of each device, and checks whether that value is lower than LL (Lines 8-9). This step is done to check whether a device is currently under-utilizing its resources. If such a misbehavior is detected, the value is added to the variable α which denotes the total under-utilization of resources detected at the device being inquired (Line 10). During this check, the algorithm also saves the number of times a device has under-utilized a certain resource metric in ρ (Line 11). This step is important as some devices can under-utilize their resources more often than others, a fact that is considered when computing the objective trust toward the devices. For example, a device that has under-utilized its resources almost at all the time is more likely to be selfish compared to a device that has done this only once. This is taken into account by computing the average under-utilization ϕ in terms of the number of times this under-utilization happens (Line 12). This step is important to ensure fairness among the devices, especially for the devices that relatively exhibit less selfish behavior compared to others. Then, the probability of under-utilization is computed by dividing the lower utilization limit by the overall under-utilization average and stored in η (Line 14). The algorithm also keeps track of the number of resource metrics that are being under-utilized in δ (Line 15). Finally, if no resource metric was under-utilized, the algorithm assigns a value of 1 to the variable τ, implying full objective trust toward the underlying device (Line 18). Yet, if one or more metrics have been under-utilized, the algorithm computes τ by dividing the sum of average under-utilization over all the resource metrics with the quantity of the metrics that have been under-utilized by the device (Line 20).

D. *Trust Aggregation*

After having derived the subjective and objective trust values for the devices, we propose an aggregation formula to aggregate both trust sources. The formula is inspired by the aggregation models discussed in [27], [28]. It is based on Bayesian inference [29] from the domain of subjective probability theory. This approach utilizes the laws of conditional probability which are related to as the Bayes theorem [30]. The motivation behind this is to decrease bias and provide more fair recommendations. Unlike most of the works in FL which only study trust in the aspect of client-server relationship, we also study it from the client-to-client perspective. This is because the server can sometimes be biased towards a certain client, or the clients can be dishonest and pose as trustworthy just to be selected by server and later turn mischievous and act selfish. Therefore, using Bayesian inference, we model this aggregation as a move from prior distribution where subjective trusts are accompanied with objective trusts towards a less biased posterior distribution. The Bayesian estimation of the belief in IoT device $i's$ trustworthiness is given by (1)

$$Trust_i = \sum_{t=1}^{n} \frac{n\tau_i + Rec_i}{2n}. \tag{1}$$

Where $Trust_i$ is the belief in IoT device $i's$ trustworthiness, Rec_i is each recommendation given on device i in the subjective trust phase derived using Algorithm 1, n is the number of recommendations collected on i, and τ_i is the objective trust value towards device i derived using Algorithm 2.

V. Coalitional Federated Learning

In this Section, we first formulate the coalitional game, including the preference function, that enables the client devices to form efficient coalitions and then we explain the practical implementation of our coalitional federated learning paradigm.

A. *Utility and Preference Functions*

A Hedonic coalitional game is a type of cooperative game that analyzes the interactions between the players where they seek to form coalitions. In such games, the players have the privilege to choose which coalition to join as they have full control over their preference rankings. In such games, the players are rational and self-interested, accordingly they are concerned about the identity of other players in their coalition as it might affect their utility.

Based on the above, an important part of the coalitional game is the utility on which the players rely to adjust their preferences. Let $U_i(C)$ denote the utility of the IoT device i vis-à-vis a given coalition C. The utility of i is calculated by summing up i's beliefs in the trustworthiness of each member j of C using (2).

$$U_i(C) = \sum_{j\in C} Trust_i^j. \tag{2}$$

Intuitively, client devices prefer to join coalitions that exhibit a higher level of trust, thus minimizing the number of selfish members. The objective is to be part of strong and honest communities, where each of the devices effectively contributes to the training of the FL model. In this context, the IoT devices that join coalitions consisting of a large number of selfish members would end up having low trust scores as a coalition from the side of the federated server. To preserve their trust, honest devices in a coalition with a large number of selfish members might even need to spend more resources and participate in more communication rounds to compensate the harm caused by their selfish counterparts.

For every IoT device $i \in I$, a preference relation denoted by $(\succeq_{i\in I})$ is a complete, reflexive, and transitive binary relation over the set of all possible coalitions that i could join. Using this relation, $C \succ_i C'$ implies that device i strictly prefers to be member of coalition C over C', while $C \succeq_i C'$ implies that i prefers to be member of coalition C over C' or has no preference between the two coalitions. Based on this definition, the preference relation of each IoT device is given as follows:

$$C \geq_i C' \iff P_i(C) \geq P_i(C'), \tag{3}$$

where C and C' being any two coalitions of which device i could be a member, and P_i being the preference function for any device i such that

$$P_i(C) = \begin{cases} -\infty, & \text{if } i \text{ believes that } C \text{ contains at} \\ & \text{least one selfish member} \\ 0, & \text{if } C \in h_i(t) \\ U_i(C) & \text{otherwise} \end{cases}, \tag{4}$$

where $h_i(t)$ represents the history set of device i at time t, containing the coalitions that i has joined previously and left at a time $t' < t$ prior to the establishment of the present coalition structure [31]. The preference function defined in (4) enables each client IoT device i to join the coalition that maximizes its trustworthiness. At the same time, it enables the devices to avoid the coalitions that they believe contain selfish devices, based on our proposed client-to-client trust framework. This is why the function assigns a $-\infty$ preference to any coalition that the device believes contains selfish devices. Furthermore, each

Algorithm 3: Coalition Formation.

```
 1: Input: Initial partition of IoT devices $Z(t)$ at time $t$
 2: Output: Final coalition structure $Z^*(t_f)$ at time $t_f$
 3:
 4: Method: COALITION-FORMATION
 5: initialize $t = 0$
 6: initialize $Z(t) = \{C_1(t)...C_i(t)\}$
 7: initialize $h_i(t) = C_l^i(t)$
 8: initialize $C_i$ as the current coalition of device $i$
 9: repeat
10:     for each IoT device $i \in Z(t)$ do:
11:         Select a coalition $C_l$ where $i$ is not member
12:         Calculate $P_i(C_i(t))$ using (4)
13:         Calculate $P_i(C_l(t))$ using (4)
14:         Compare $P_i(C_l(t) \cup \{i\})$ with $P_i(C_i(t))$
15:         if $P_i(C_l(t)) > P_i(C_i(t))$ then
16:             Leave $C_i$ and Join $C_l$
17:             Update history $h_i(t)$
18:         else
19:             $Z(t+1) = Z(t)$
20: $t = t + 1$
21: until no change happens in the partition
22: $Z^*(t_f) = Z(t)$
23: return $Z^*(t_f)$
24: End method
```

device tries as much as possible to avoid joining any previously joined coalition in which there have not been any changes in terms of its members. This is possible by assigning the value 0 to any coalition that is part of the device's history set. Lastly, the device would prefer to join a coalition that maximizes its utility, which quantifies the overall device's trust toward the coalition's members.

B. Federated Learning Coalition Formation

To derive the equilibrium of the game, we first formally define the coalition structure and propose a distributed coalition formation algorithm for the client IoT devices (Algorithm 3). This algorithm enables the IoT devices to decide about the coalitions to join so as to maximize the overall trust.

Definition 1. A coalition structure is a set of coalitions $Z = \{C_1, \ldots, C_j\}$ that divides the set I of IoT devices into disjoint coalitions in a way that $\forall i \neq i,' C_i \cup C'_{i'} = \emptyset$. We denote by C_i the coalition of which device i is member.

We explain in Algorithm 3 the distributed client devices coalition formation process. The algorithm takes as an input the initial partition of the coalition structure (Line 1) and outputs the final coalitional structure (Line 2). Lines $4-7$ are used for initialize the different variables. Thereafter, the algorithm loops over each IoT device in the coalition structure (Line 9), saving the coalition of which they currently are member (Line 10). It then arbitrarily selects a coalition of which the underlying device is not member (Line 11). Then, based on (4), the algorithm calculates the preference P_i of the coalition that contains i (Line 12) and that of the other arbitrarily chosen coalition that i can potentially join (Line 13). Based on the computed trust values, the algorithm compares the utility values of both coalitions (Line 14). In case the new coalition has a higher utility compared to the current one (Line 15), the device would prefer to leave its current coalition and join the new one, while updating its history set (Lines $16-17$). If this is not the case, the coalition structure remains the same at the current time moment, and that coalition structure is assigned to the next time moment (Line 19). The algorithm is repeated until no change in the partitioning of the coalitions occurs (Line 21), meaning that the algorithm has converged to a Nash and individual stable points. Note that, according to the proposed algorithm, the selfish devices will end up in singleton coalitions (due to the preference function proposed in (4)). This greatly helps the federated server avoid the selfish devices when making its selections.

Definition 2. Nash Equilibrium: A coalition structure Z is said to be in a Nash equilibrium state (also known as Nash stability) if there is no device having incentives to leave its current coalition to (1) either form a singleton coalition in which it is the sole member, or (2) join another existing coalition, i.e., $\forall i \in I, C_i \succeq_i C \cup \{i\} \forall C_i \in Z$.

Definition 3. Individual Stability: A coalition structure Z is said to be individually stable if no players in Z can benefit from moving out of their current coalition and joining any other one, without making the total trust of the members of the newer coalition worse, i.e, $i \in I$ and $C_l \cup Z \in \emptyset$ such that $C_l \cup \{i\} \succ_i C_m^i$ and $C_l \cup \{j\} \succeq_j C_l, \forall j \in C_l$.

Based on these two definitions along with the proofs provided in [24], we argue that Algorithm 3 converges to a final coalition structure $Z^*(t_f)$ which consists of several disjoint coalitions that are both Nash and individually stable.

C. Coalition Master Election

We discuss in this Section a Coalition Master (CM) election mechanism that is inspired by the work proposed in [32] in the context of Vehicular Ad-Hoc Networks. Once the coalitions are formed, the devices belonging to the same coalition share resource information with one another. This information is used to elect a Coalition Master (CM) for each coalition. The master serves as the point of contact between its coalition members and the federated server. It is mainly responsible for (1) aggregating the local models trained by its coalition members; (2) periodically sending the aggregate coalition-level models that are locally trained in its coalition to the federated server; and (3) downloading the latest version of the global model from the federated server and sharing it with its coalition members. This is important to avoid pairwise communications between each client and the federated server, which are the main cause for communication bottlenecks in FL.

The resource information shared by devices are focused on four resource metrics which are crucial for FL training, i.e., available CPU, RAM, bandwidth (BW) and latency (LC) between each device and the federated server. Formally, let CPU_i be the amount of CPU available at device i, RAM_i be the amount of RAM available at i, BW_i be the amount of bandwidth available at i and $LC_{i,CS}$ be the latency between device i and the

Algorithm 4: Coalition Master Election.

```
 1: Input: A coalition $C$
 2: Input: Set of IoT devices in $C$
 3: Input: Set of weights for the resource metrics
 4: Output: Master of coalition $C$
 5:
 6: Method COALITION-MASTER-ELECTION
 7:     Devices in $C$ share their resource information
 8:     For each device $i \in C$ do
 9:         Compute $Score(CMS_i)$ using (5)
10:         Store $Score(CMS_i)$ in list $ListMaster$
11:         End For
12:         Sort $ListMaster$ from highest to lowest
13:         Return First element of $ListMaster$
14: End method
```

federated server CS. Also, let $W = \{w_{cpu}, w_{ram}, w_{BW}, w_{LC}\}$ be the set of weights representing the different weights for the metrics that are taken into account for the master's election. Let $Score(CMS_i)$ be the score obtained by each device i to quantify its adequacy in being the master of its coalition, and is calculated according to (5).

$$Score(CMS_i) = (w_{cpu} \times CPU_i) + (w_{ram} \times RAM_i) + (w_{BW} \times BW_i) + (w_{LC} \times (1 - LC_{i,CS})). \tag{5}$$

The CM election algorithm is described in Algorithm 4. The algorithm takes as input a certain coalition, the set of IoT devices inside that coalition, and the set of weights for the resource metrics (Lines $1-3$) and returns the elected coalition master (Line 4). First, the devices inside the coalition share with one another their resource information in terms of available CPU, RAM, bandwidth, and latency from the federated server (Line 7). This information is then used alongside the input weights to compute the score $Score(CMS_i)$ of each device using (5) (Line 9). The scores of all devices belonging to the same coalition are stored in a list $ListMaster$. This list is then sorted from the highest to the lowest (Line 12) and the device with the highest score is then selected to be the coalition master (Lines 13).

D. Coalition-Based Federated Learning Training

Having formed the client IoT devices' coalitions, we explain in Algorithm 5 how the practical FL training takes place using the proposed *Coalitional Federated Learning* architecture.

Algorithm 5 takes as an input the IoT devices in the environment, as well as the federated server which will be performing the client selection (Line 1). It outputs the final model parameters of the federated learning model trained by the selected IoT devices (Line 2). During phase 1, algorithms 1 and 2 are invoked to respectively derive the subjective and objective trust scores of the IoT devices (Lines 7 and 8). Equation (1) is then used to aggregate the subjective and objective trust sources (Line 9). In Phase 2 (Line 10), Algorithm 3 is invoked to enable IoT devices to autonomously form trustworthy coalitions using

Algorithm 5: Trust-Enabled Coalitional Federated Learning.

```
 1: Input: Set of IoT devices, federated Server (FS)
 2: Output: Final model parameters of FL
 3:
 4: Method: TeC-FL:
 5: Repeat
 6:     Phase 1: Client-to-client trust establishment
        mechanism:
 7:         Run Algorithm 1 to derive IoT devices'
            subjective trust
 8:         Run Algorithm 2 to derive IoT devices'
            objective trust
 9:         Use (1) to aggregate both sources of trust
10:     Phase 2: Trust-Enabled Coalition Formation
11:         Run Algorithm 3 to form client IoT devices
            coalitions
12:     Phase 3: Coalition Master Selection
13:         Run Algorithm 4 to elect the CM of each
            coalition
14:     Phase 4: Coalitional Federated Learning
15:             While Desired accuracy is not attained do
16:               CS selects non-singleton coalitions for
                  training
17:               For each non-singleton coalition do
18:                 CM shares a subset of data with its
                    members
19:                 Members merge local and shared data
20:                 Members start the training process
21:                 Members share model parameters with CM
22:                 CM aggregates local results
23:               End For
24:                 CM sends aggregate local models to CS
25:                 FS performs aggregation of local models
26:                 FS sends global model to the coalitions
27:             End While
28: Until $\epsilon$ elapses
29: End method
```

the aggregate trust scores. Once the coalitions are formed, the election of a Coalition Master (CM) at the level of each coalition takes place in Phase 3 (Lines 12). In Phase 4, the federated learning training takes place on the formed coalitions instead of individual devices. In fact, the federated server selects all non-singleton coalitions to participate in the FL training (Line 17). This is because only selfish devices will end up in singleton coalition as per our proposed coalition formation algorithm (Algorithm 3).

At the level of each selected coalition, the master of that coalition shares a small subset of shared data with its members to reduce non-IIDness problem (Line 18). Each coalition member combines its own local data with the shared subset to perform the training of the FL model (Line 19), and starts the learning process (Line 20) and then shares its updated set of parameters with the coalition master (Lines 21). The master then

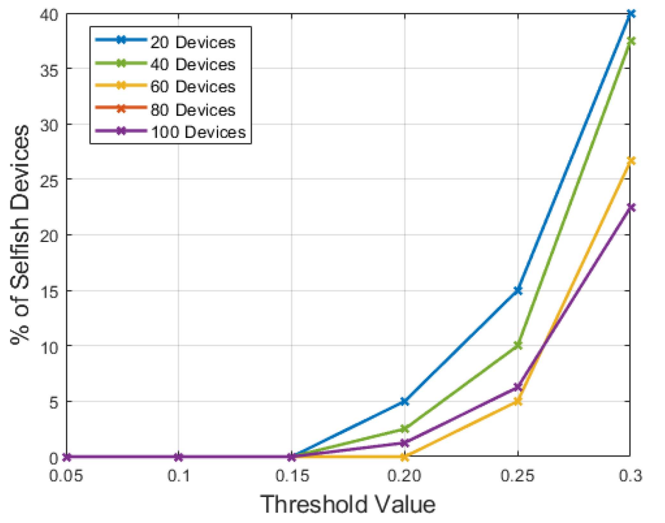


Fig. 4. The effect of various threshold values on the overall final selfish device detection in different environments.

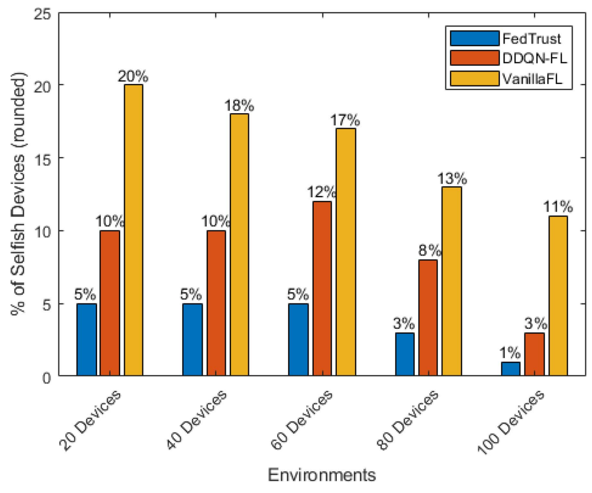


Fig. 5. Percentage of selfish devices in the final coalitional structure.

aggregates the local models from all its coalition members (Line 22) and only periodically shares the aggregation results with the federated server (Line 24). In its turn, the federated server collects the aggregate local models from the different coalition masters and aggregates them to generate the global model (Line 25) which is then sent to each coalition master for the next communication round (Line 26). By restricting the upstream (uploading local models to the federated server) and downstream communications (downloading the latest global model from the federated server) to only the coalition masters, our solution significantly improves the communication efficiency of the FL process compared to conventional FL approaches.

The coalition-based training is repeated until a desired accuracy level for the global model is reached (Line 15). Moreover, the whole phases (i.e., trust establishment, coalition formation, master election, and coalitional FL) are repeated after a certain fixed period of time ϵ (Line 28) to catch and reflect the changes that happen in the environment such as the arrival/leave of new or existing IoT devices and dynamism in the devices' trust values.

VI. Empirical Results and Discussion

In this section, we describe the environment set up to perform our experiments, present our results, and discuss the main findings.

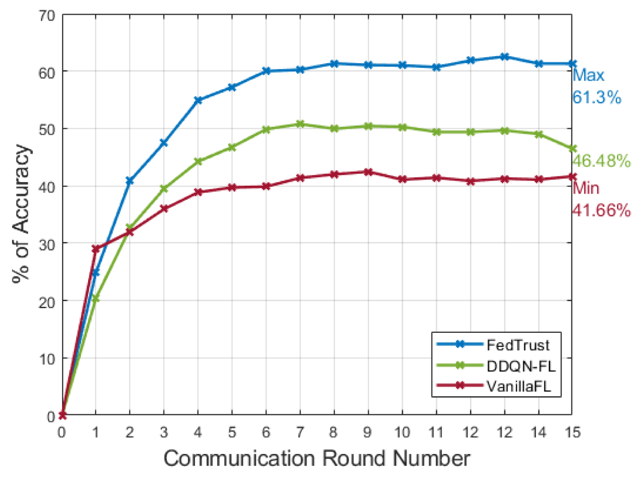
A. Experimental Setup

To perform the simulations, we build our own environment in which each IoT device is equipped with a CPU, RAM, and network bandwidth capacity. We utilize Python 3 alongside several libraries such as Numpy, Torch, Torchvision, and Syft. We run the simulations on a workstation that is equipped with an Intel i7 processor with 16 GB of RAM. The operating system on which we base our simulations is Microsoft Windows 10. We use the MNIST (Modified National Institute of Standards and Technology database) dataset [33] which can be downloaded as part of the Torchvision datasets. MNIST contains 60,000 training and 10,000 testing images of $28x28$ pixel each. The images represent handwritten digits ranging between 0 and 9 in black and white alongside their labels. In the federated learning part, we use the *FedAvg* traditional local model aggregation method proposed in [22]. We change the number of IoT devices between 20, 40, 60, 80, and 100. Furthermore, to classify a client as either selfish or not, we experimentally derive a threshold (Section VI-B1), where any IoT device having a trust of value (computed as per (1)) below this threshold is considered to be selfish. This methodology for deriving a trust threshold has been adopted in [34] where the authors experiment with different threshold values in the context of trust management using a Bayesian approach. Furthermore, for simplicity, in our experimental environments we assume that devices do not straggle and focus mainly on selfish devices. We compare our solution with the two following existing approaches:

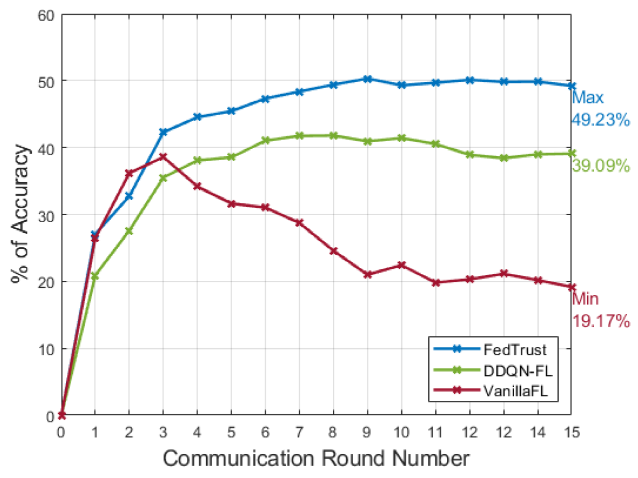
1) Vanilla-FL: This is the de facto approach that assumes a fully honest client environment in the sense it does not take into account the presence of selfish clients that try to harm the FL process.
2) A State-of-the-art approach [18] that relies on trust between the federated server and the devices individually. This approach utilizes the resources of devices such as CPU and Memory resources coupled with some DDQN (Deep Reinforcement Learning with Double Q-learning) techniques to derive the trust scores between the two entities and. Hereafter, in our writings we refer to this approach as 'DDQN-FL'

B. Experimental Results

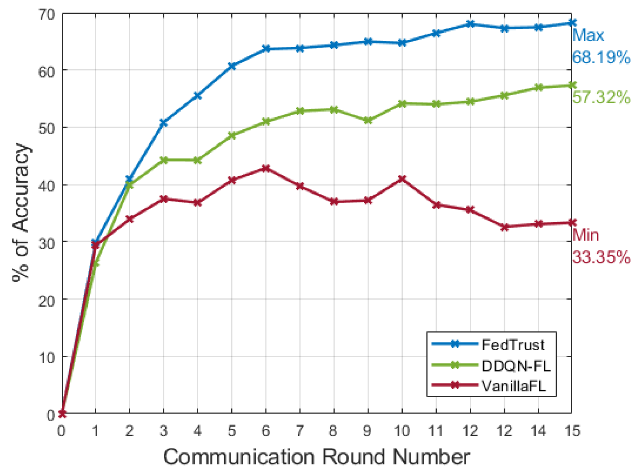
1) Threshold Value Determination: For the devices to evaluate each other's trustworthiness and select the coalitions that maximize their utility, a trust threshold should be derived. That is, if a device's aggregate trust value (using (1)) is above that threshold, the device would be considered to be trustworthy; otherwise, the device would be considered to be selfish. Therefore, selecting an appropriate trust threshold is of utmost importance. Selecting an arbitrarily high threshold value would make the



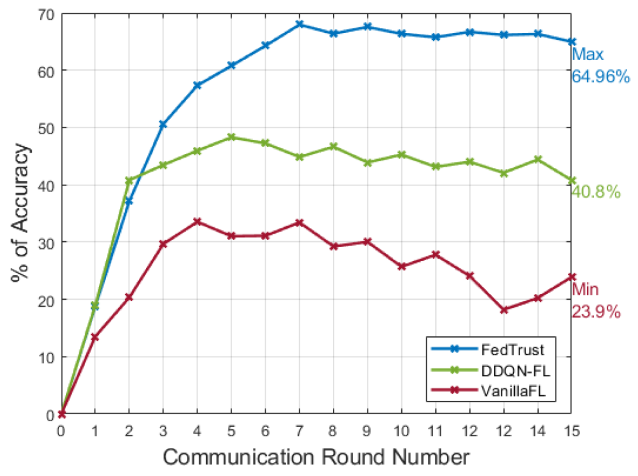
(a) 20 devices environment



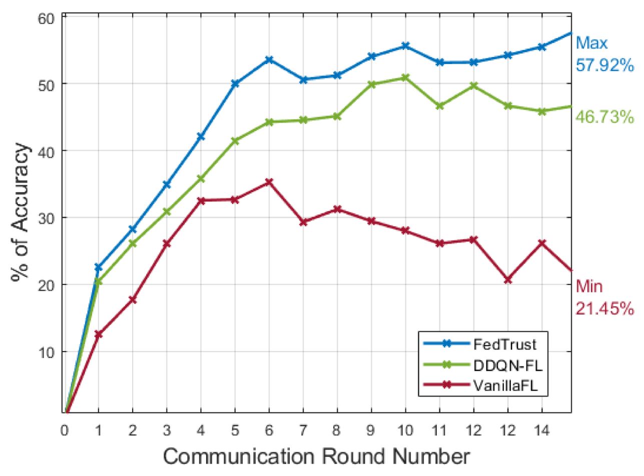
(b) 40 devices environment



(c) 60 devices environment



(d) 80 devices environment



(e) 100 devices environment

Fig. 6. *Accuracy:* The effect of accuracy in various environments with different number of input devices.

environment severe and would result in a large number of selfish devices in the network. On the other hand, selecting an arbitrarily low threshold value would result in an almost fully honest environment in which the number of selfish devices is minimal. To avoid these problems, we apply an experimental approach to derive a fair trust threshold. Specifically, we study the impact of several threshold values (i.e., 0.05, 0.10, 0.15, 0.20, 0.25, and 0.3) on the percentage of selfish devices in the environment while also varying the number of devices, as highlighted in Fig. 4. We notice from the figure that with the values of 0.05, 0.1, and 0.15 for the threshold and for the different studied number devices (i.e., 20, 40, 60, 80 and 100), the algorithm fails to detect selfish devices in the final coalitions. A threshold value of 0.2 can detect a small number selfish devices for an input number of devices of 40, 60, 80 and 100. On the contrary, a threshold value of 0.3 results in a large percentage of selfish devices. For example, for an environment with 20 devices, a threshold value of 0.3 results in 40% of the environment being selfish. To avoid these two extreme scenarios, we believe that a threshold value of 0.25 is a reasonable choice. Thus, in the rest of the experiments, the value of 0.25 will be used for the trust threshold.

2) Percentage of Selfish Devices in the Coalitional Federated Learning Environment: In this set of experiments, we compare the effectiveness of our approach against the other two studied approaches in terms of minimizing the number of selfish devices in the coalitional FL environment. This, however, does not include the selfish devices that end up in singleton coalitions (i.e., the devices that do not get chosen by any other devices based on the preference function in (4)) as these singletons will be automatically avoided by the federated server.

We notice in Fig. 5 that our solution is successful in minimizing the number of selfish devices in final structure of the coalitional FL across the different input number of devices. On the contrary, the Vanilla-FL client selection approach entails the highest percentage of selfish client devices. It can be observed that our solution can reduce the percentage of selfish devices up to $4x$ to $5x$ more than Vanilla-FL approach. For example, in an environment consisting of 20 devices, our approach results in only 5% of selfish devices in the final coalition structure as opposed to 20% in the case of Vanilla-FL. Furthermore, as the number of devices increases, our solution continues to outperform the Vanilla-FL at a larger scale. Compared to the DDQN-FL approach as well, our approach reduces the percentage of selfish devices. For example in environments with 20, 40, and 60, we can see that our approach minimizes the percentage of selfish devices by $2x$ compared to the DDQN-FL approach and more than $2x$ in environments with 80 and 100 devices. This is because our solution takes into account, not only the CPU and RAM as is the case in DDQN-FL, but also the bandwidth when deriving the trust values. In fact, in a federated learning environment, communication efficiency between the clients and the federated server is of utmost importance and should not be neglected when computing the trust values. Moreover, the DDQN-FL approach only considers the trust relationships between the federated server and clients, but does not study the trust between the clients themselves. Thus, this approach cannot guarantee the harmony among the clients during the training.

3) FL Training Accuracy: In Fig. 6, we study the performance of the different comparison approaches in terms of improving the accuracy of the overall global FL model over 15 communication rounds. In this set of experiments, We vary the number of client IoT devices from 20 to 100.

We notice from Fig. 6 that our approach scores the best (maximum) in terms of accuracy compared to the other approaches. The results also indicate that the Vanilla-FL approach yields the least (minimum) accuracy, while the DDQN-FL method, although performs better than the traditional Vanilla-FL method, still falls short compared to our approach. Particularly, approach attains maximum accuracy of 68% in the 80 devices environment which is around 24% higher compared to DDQN-FL, and around 41% higher than the vanilla approach. On the other hand, our approach scores the minimum in terms of accuracy in the 40 devices environment with an accuracy of 49% which is still around 10% higher than the DDQN-FL and 30% higher than the vanilla approach. The results highlight that in different and various environments, our approach keeps on scoring better results in terms of accuracy compared to the other approaches. An accuracy gap ranging between around 10% to 25% (Fig. 6(b), (c), (e), (a), (d) respectively) between our approach and the DDQN-FL approach can be noticed. This gap stems from the fact that our approach takes both subjective and (improved) objective trust metrics into account and integrates a robust aggregation method for these trust metrics. Moreover, our solution enables the client devices to form trustworthy coalitions where the number of selfish devices is minimal using client-to-client trust relationships, while on the other hand, the DDQN-FL approach does not take into account the cross-client trust relationships. On the other hand, the difference in accuracy between our approach and the Vanilla-FL approach varies between around 20% and 40% and more (Fig. 6(a), (b), (c), (e), (d) respectively). This large gap is attributed to the fact that the later approach does not propose any mechanism to filter out any possible selfish device as the client selection is purely random. This causes the selection of a large number of selfish devices to participate in the FL training.

VII. Conclusion & Future Directions

In this paper, we proposed a new federated learning paradigm called *Coalitional Federated Learning*. Our solution first proposes a comprehensive trust mechanism to establish client-to-client relationships. Based on these relationships, we propose a coalition formation algorithm that enables the client devices to form harmonious team works. The benefits of our solution is that it (1) reduces the non-IIDness of the data across clients by enabling periodic sharing of small subsets of data within coalitions; (2) improves the communication efficiency by avoiding pairwise communications between the server and individual clients; and (3) minimizes the number of selfish clients that seek to join the FL process to obtain the global model without actively participating in the training. Simulations in comparison with two state-of-the-art approaches suggest that our solution outperforms the Vanilla-FL and DDQN-FL in terms of (1) minimizing the percentage of selfish devices in the coalitional

FL environment by up to 5 times and (2) improving the accuracy between 10% and 40% for the same number of communication rounds.

We have considered the notion of selfish devices in our experiments and their effect on the overall accuracy of the global model. In future works, the notion of straggler devices can also be studied in addition to selfish devices to highlight their effect on the accuracy of the global model and its convergence speed. Furthermore, our work can be used to inspire the evaluation of other active type of attacks such as poisoning attacks where misbehaving clients inject false or poisoned data into the training datasets to decrease the accuracy of the global model.

References

[1] P. Kairouz et al., "Advances and open problems in federated learning," *Found. Trends Mach. Learn.*, vol. 14, no. 1/2, pp. 1–210, 2021.

[2] O. A. Wahab, A. Mourad, H. Otrok, and T. Taleb, "Federated machine learning: Survey, multi-level classification, desirable criteria and future directions in communication and networking systems," *IEEE Commun. Surveys Tuts.*, vol. 23, no. 2, pp. 1342–1397, Second Quarter 2021.

[3] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra, "Federated learning with non-iid data," 2018, *arXiv:1806.00582*.

[4] Q. Li, Y. Diao, Q. Chen, and B. He, "Federated learning on non-iid data silos: An experimental study," in *Proc. IEEE 38th Int. Conf. Data Eng.*, 2022, pp. 965–978.

[5] H. Zhu, J. Xu, S. Liu, and Y. Jin, "Federated learning on non-iid data: A survey," *Neurocomputing*, vol. 465, pp. 371–390, 2021.

[6] L. Zhang, L. Shen, L. Ding, D. Tao, and L.-Y. Duan, "Fine-tuning global model via data-free knowledge distillation for non-iid federated learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 10174–10183.

[7] L. Zhang, Y. Luo, Y. Bai, B. Du, and L.-Y. Duan, "Federated learning for non-iid data via unified feature learning and optimization objective alignment," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2021, pp. 4420–4428.

[8] M. Luo, F. Chen, D. Hu, Y. Zhang, J. Liang, and J. Feng, "No fear of heterogeneity: Classifier calibration for federated learning with non-iid data," in *Proc. Adv. Neural Inf. Process. Syst.*, M. Ranzato, A. Beygelzimer, Y. Dauphin, P. S. Liang, and J. Wortman Vaughan, Eds., Curran Associates, Inc., 2021, vol. 34, pp. 5972–5984.

[9] O. Shahid et al., "Communication efficiency in federated learning: Achievements and challenges," 2021, *arXiv:2107.10996*.

[10] K. Prasad, S. Ghosh, G. Cormode, I. Mironov, A. Yousefpour, and P. Stock, "Reconciling security and communication efficiency in federated learning," 2022, *arXiv:2207.12779*.

[11] H.-P. Wang, S. Stich, Y. He, and M. Fritz, "ProgFed: Effective, communication, and computation efficient federated learning by progressive training," in *Proc. 39th Int. Conf. Mach. Learn.*, K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvari, G. Niu, and S. Sabato, Eds., PMLR, 2022, pp. 23034–23054.

[12] P. Prakash et al., "IoT device friendly and communication-efficient federated learning via joint model pruning and quantization," *IEEE Internet Things J.*, vol. 9, no. 15, pp. 13638–13650, Aug. 2022.

[13] Y. Fraboni, R. Vidal, and M. Lorenzi, "Free-rider attacks on model aggregation in federated learning," in *Proc. 24th Int. Conf. Artif. Intell. Statist.*, A. Banerjee and K. Fukumizu, Eds., PMLR, 2021, pp. 1846–1854.

[14] N. Zhang, Q. Ma, and X. Chen, "Enabling long-term cooperation in cross-silo federated learning: A repeated game perspective," *IEEE Trans. Mobile Comput.*, to be published, doi: 10.1109/TMC.2022.3148263.

[15] J. Wang, "PASS: Parameters audit-based secure and fair federated learning scheme against free rider," 2022, *arXiv:2207.07292*.

[16] S. P. Karimireddy, W. Guo, and M. I. Jordan, "Mechanisms that incentivize data sharing in federated learning," 2022, *arXiv:2207.04557*.

[17] S. AbdulRahman, H. Tout, A. Mourad, and C. Talhi, "FedMCCS: Multicriteria client selection model for optimal IoT federated learning," *IEEE Internet Things J.*, vol. 8, no. 6, pp. 4723–4735, Mar. 2021.

[18] G. Rjoub, O. A. Wahab, J. Bentahar, and A. Bataineh, "A trust and energy-aware double deep reinforcement learning scheduling strategy for federated learning on IoT devices," in *Proc. Int. Conf. Service-Oriented Comput.*, Springer, 2020, pp. 319–333.

[19] M. Chen, H. V. Poor, W. Saad, and S. Cui, "Convergence time optimization for federated learning over wireless networks," *IEEE Trans. Wireless Commun.*, vol. 20, no. 4, pp. 2457–2471, Apr. 2021.

[20] M. Chahoud, S. Otoum, and A. Mourad, "On the feasibility of federated learning towards on-demand client deployment at the edge," *Inf. Process. Manage.*, vol. 60, no. 1, 2023, Art. no. 103150.

[21] W. Zhang, X. Wang, P. Zhou, W. Wu, and X. Zhang, "Client selection for federated learning with non-iid data in mobile edge computing," *IEEE Access*, vol. 9, pp. 24462–24474, 2021.

[22] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. 20th Int. Conf. Artif. Intell. Statist.*, PMLR, 2017, pp. 1273–1282.

[23] R. Tarjan, "Depth-first search and linear graph algorithms," *SIAM J. Comput.*, vol. 1, no. 2, pp. 146–160, 1972.

[24] O. A. Wahab, J. Bentahar, H. Otrok, and A. Mourad, "Towards trustworthy multi-cloud services communities: A trust-based hedonic coalitional game," *IEEE Trans. Serv. Comput.*, vol. 11, no. 1, pp. 184–201, Jan./Feb. 2018.

[25] D. F. Groebner, P. W. Shannon, P. C. Fry, and K. D. Smith, *Business Statistics*. London. U.K.: Pearson Education, 2013.

[26] J. W. Tukey et al., *Exploratory Data Analysis*, vol. 2. Reading, MA, USA: Addison-Wesley, 1977.

[27] H. Wang, B. Zou, G. Guo, D. Yang, and J. Zhang, "Integrating trust with user preference for effective web service composition," *IEEE Trans. Serv. Comput.*, vol. 10, no. 4, pp. 574–588, Jul./Aug. 2017.

[28] O. A. Wahab, J. Bentahar, H. Otrok, and A. Mourad, "Optimal load distribution for the detection of VM-based DDoS attacks in the cloud," *IEEE Trans. Serv. Comput.*, vol. 13, no. 1, pp. 114–129, Jan./Feb. 2020.

[29] M. S. Hamada, A. Wilson, C. S. Reese, and H. Martz, *Bayesian Reliability*. Berlin, Germany: Springer Science & Business Media, 2008.

[30] V. N. Vapnik, "An overview of statistical learning theory," *IEEE Trans. Neural Netw.*, vol. 10, no. 5, pp. 988–999, Sep. 1999.

[31] W. Saad, Z. Han, T. Basar, M. Debbah, and A. Hjorungnes, "Hedonic coalition formation for distributed task allocation among wireless agents," *IEEE Trans. Mobile Comput.*, vol. 10, no. 9, pp. 1327–1344, Sep. 2011.

[32] O. A. Wahab, H. Otrok, and A. Mourad, "VANET QoS-OLSR: QoS-based clustering protocol for vehicular ad hoc networks," *Comput. Commun.*, vol. 36, no. 13, pp. 1422–1435, 2013.

[33] L. Deng, "The MNIST database of handwritten digit images for machine learning research [best of the web]," *IEEE Signal Process. Mag.*, vol. 29, no. 6, pp. 141–142, Nov. 2012.

[34] M. K. Denko, T. Sun, and I. Woungang, "Trust management in ubiquitous computing: A Bayesian approach," *Comput. Commun.*, vol. 34, no. 3, pp. 398–406, 2011.

Sarhad Arisdakessian received the master's degree in computer science from Lebanese American University (LAU), Beirut. He is currently working toward the PhD degree with the Department of Computer and Software Engineering, Polytechnique Montréal, Canada. His main research interests are in the areas of federated learning, game theory, cybersecurity, and Internet of Things.

Omar Abdel Wahab received the Msc degree in computer science from Lebanese American University (LAU), Lebanon, in 2013, and the PhD degree in information and systems engineering from Concordia University, Montreal, Canada. He is assistant professor with the Department of Computer Engineering and Software Engineering, Polytechnique Montréal, Canada. From 2017 to 2018, he was postdoctoral fellow with the École de Technologie Supérieure (Canada), where he worked on an industrial research project in collaboration with Rogers and Ericsson. His main research interests are in the areas of cybersecurity, artificial intelligence, Internet of Things and Big Data analytics.

Azzam Mourad (Senior Member, IEEE) received the MSc degree in CS from Laval University, Canada, in 2003, and the PhD degree in ECE from Concordia University, Canada, in 2008. He is currently a professor of computer science with the Lebanese American University, a visiting professor of computer science with New York University Abu Dhabi and an affiliate professor with the Software Engineering and IT Department, Ecole de Technologie Superieure (ETS), Montreal, Canada. He published more than 100 papers in international journal and conferences on *Security, Network and Service Optimization and Management targeting IoT*, *Cloud/Fog/Edge Computing*, *Vehicular and Mobile Networks*, and *Federated Learning*. He has served/serves as an associate editor for *IEEE Transaction on Network and Service Management*, *IEEE Network*, *IEEE Open Journal of the Communications Society*, *IET Quantum Communication*, and *IEEE Communications Letters*, the general chair of IWCMC2020, the general co-chair of WiMob2016, and the track chair, a TPC member, and a reviewer for several prestigious journals and conferences.

Hadi Otrok (Senior Member, IEEE) received the PhD degree in ECE from Concordia University. He holds an associate professor position with the Department of EECS, Khalifa University, Abu Dhabi, UAE. He is a associate editor with *Ad-Hoc Networks* (Elsevier) and *IEEE Network*. He co-chaired several committees at various IEEE conferences. His research interests include the domain of computer and network security, crowd sensing and sourcing, ad hoc networks, and cloud security.