October 6 University

Info Sys & Comp Science

Paper Type: Survey

federated learning

Kareem Abd Eltawab Hamed, and Ahmed Osama Mohamed, and Mohamed ibrahim Hussein

Code :https://github.com/kareem5T9/Federated-Learning.git

Abstract

The amount of data generated owing to the rapid development of the Smart Internet of Things is increasing exponentially. Traditional machine learning can no longer meet the requirements for training complex models with large amounts of data. Federated learning, as a new paradigm for training statistical models in distributed edge networks, alleviates integration and training problems in the context of massive and heterogeneous data and security protection for private data. Edge computing processes data at the edge layers of data sources to ensure low-data-delay processing; it provides high-bandwidth communication and a stable network environment, and relieves the pressure of processing massive data using a single node in the cloud center. A combination of edge computing and federated learning can further optimize computing, communication, and data security for the edge-Internet of Things. This review investigated the development status of federated learning and expounded on its basic principles. Then, in view of the security attacks and privacy leakage problems of federated learning in the edge Internet of things, relevant work was investigated from cryptographic technologies (such as secure multi-party computation, homomorphic encryption and secret sharing), perturbation schemes (such as differential privacy), adversarial training and other privacy security

protection measures. Finally, challenges and future research directions for the integration of edge computing and federated learning are discussed.

Keywords :(Federated learning · Edge computing · Data security · Privacy-preserving · Internet of Things)

1 | Introduction

The Artificial intelligence (AI) is a technology that enables machines to realize human learning and reasoning abilities. This technology has been rapidly advancing and playing a significant role in our daily lives. In AI technology, data acquisition is crucial because AI technologies require model training using a certain amount of data for reliable AI based services, and the performance of AI-based services is considerably affected by the training data quality. However, there are difficulties in data collection because the data may contain sensitive private information.

In FL, a centralized server sends a global model for AI learning to many distributed devices, which return local model parameters to the centralized server after training the model with local data. The centralized server updates the global model parameters using the locally trained model parameters from the distributed devices and sends the updated global model parameters to the distributed devices. This procedure is repeated until convergence is achieved. FL has the advantage of preventing the leakage of sensitive private information because it does not require local data sharing.

Homomorphic encryption (HE) is a technology that enables arithmetic operations on ciphertexts without decryption.

Using the HE scheme, the centralized server can update the global model parameters with the encrypted local gradients based on the homomorphic operation. Therefore, the distributed devices participating in FL, which we refer to as clients, do not have to concern about data leakage through local gradients because they deliver encrypted local gradients to the server. However, the clients must share the same private key in the FL-based system because homomorphic operations can only be performed between values encrypted with the same public key. In FL-based systems where many distributed devices, such as smartphones and Internet

of Things (IoT) devices participate, the same private key for decryption can be distributed to many clients. Suppose The same private key is shared with many clients. Then, the probability of the private key being leaked or a malicious participant accessing other participants' data increases, which can weaken privacy protections in FL-based systems. As the result, stealing one client's private key can nullify the data privacy protection of all clients participating in FL systems. To overcome this vulnerability, this paper proposes a privacy-preserving federated learning (PPFL) algorithm that allows a cloud server to update global model parameters by aggregating local parameters encrypted by different HE keys in the same FL-based system using homomorphic operations based on a distributed cryptosystem.

2| Federated Learning Background

Federated Learning was introduced by Google in 2017 and came to solve ideal problems

with following properties :

1) Training on real-world data from mobile devices provides a distinct advantage over training on proxy data that is generally available in the data center.

2) This data is privacy sensitive or large in size (compared to the size of the model), so it is preferable not to log it to the data center purely for the purpose of model training (in service of the focused collection principle).

3) For supervised tasks, labels on the data can be inferred naturally from user interaction.

To name the solution the authors of investigated a learning technique that allows users to collectively reap the benefits of shared models trained from this rich data, without the need to centrally store it. They called their approach Federated Learning, since the learning task is solved by a loose federation of participating devices (we refer to them as clients) which are coordinated by a central server. Each client has a local training dataset which is never uploaded to the server. Instead, each client computes an update to the current global model maintained by the server, and only this update is communicated.

The primary motivation behind FL is direct implementation of the principle of focused collection or

data minimization proposed by the 2012 White House report on privacy of consumer data.

FL focuses on preserving privacy and enabling on-device model training, making it suitable for low-latency applications with limited server connectivity. It works well in scenarios where data is spread across multiple devices, eliminating the need for centralized data storage and reducing communication overhead by transmitting model updates instead of raw data. However, it faces challenges such as effective communication between devices, handling heterogeneous data distribution, and security risks like model inversion attacks.

Despite these limitations, FL holds promise for revolutionizing collaborative machine learning by addressing privacy concerns, improving bandwidth efficiency, and enhancing security. It is particularly beneficial for real-world mobile device data, mitigating privacy risks associated with centralized storage.

It can also play a vital role where we are handling large-scale data. To manage the abundant data generated by smart devices within cyber-physical systems, IoT frameworks necessitate smart and safe big data analysis methodologies.

Experimental results across various tasks like image classification and language modeling demonstrate its effectiveness in reducing communication rounds and improving generalization performance. FL tackles challenges such as non-representative client data and

limited mobile device communication. In conclusion, FL is a viable approach for training machine learning models on decentralized mobile device data, offering significant benefits over centralized training methods.

2.1| Horizontal and Vertical FL

Horizontal FL (also known as sample-based FL) is where features are similar but vary in terms of data. The benefit of this architecture is its independence, and its focus is on security.

Vertical FL (also known as feature-based FL) is where datasets can have similar sample IDs but differ in their features. The benefit of this architecture is encryption, and itsfocus is on privacy.

2.2| IID and non-IID

Data in FL is either IID or non-IID.

IID stands for Independent and Identically Distributed:

Independent: Each data point is unrelated to and not influenced by any other data point in the dataset. The occurrence or value of one data point has no effect on the occurrence or value of another.

Identically Distributed: The data points are drawn from the same probability distribution.

This means that each data point has the same underlying statistical properties.

Non-IID stands for Non-Independent and Non-Identically Distributed:

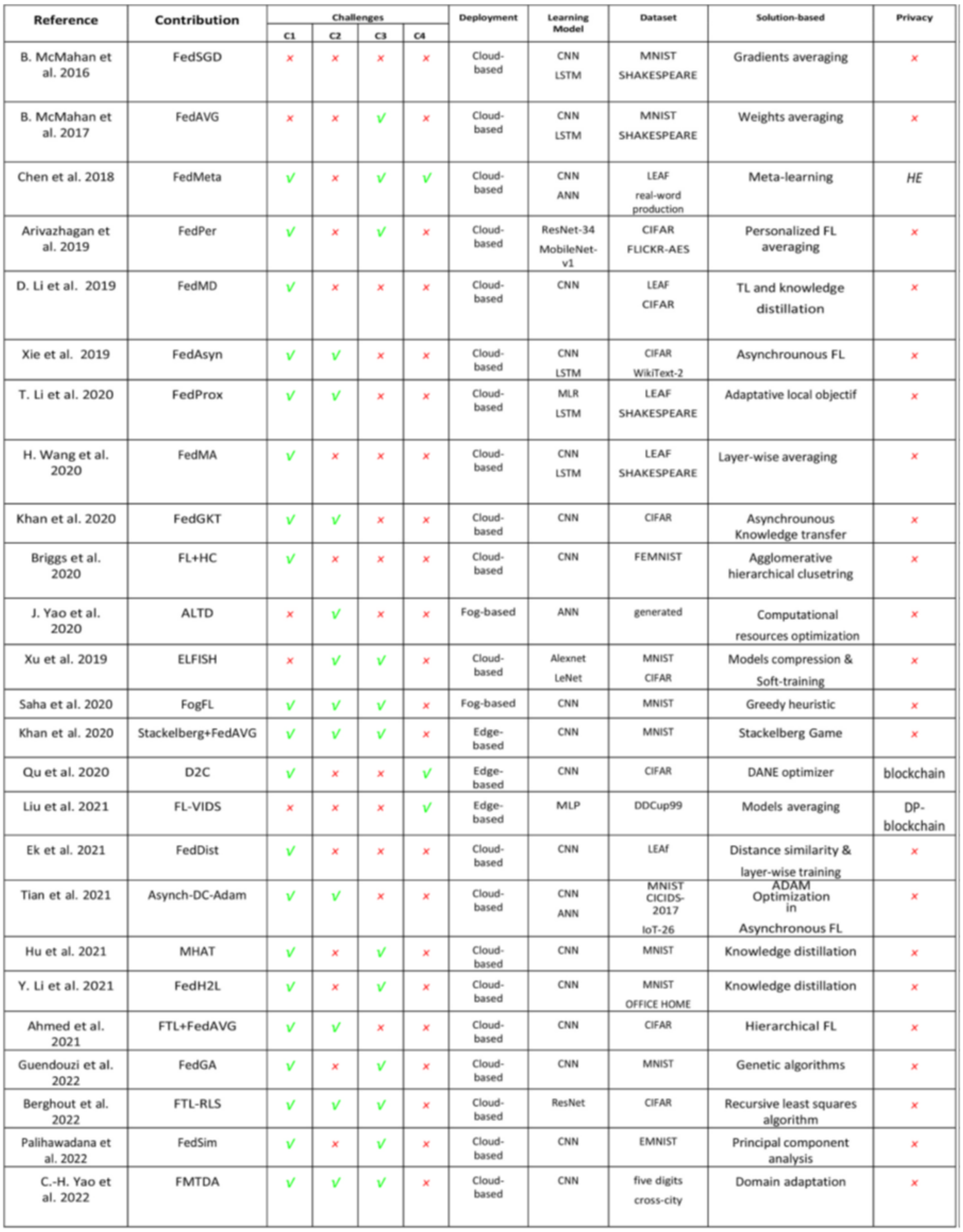
Non-Independent: The data points in the dataset may have some form of dependence or correlation between them. The occurrence or value of one data point may be influenced by or correlated with the occurrence or value of another.

Non-Identically Distributed: The data points are not drawn from the same probability distribution. There may be variations in statistical properties across different subsets or groups of the data.

Non-IID datasets are common, especially in FL or decentralized systems where data is distributed across multiple source.

2.3| FL Aggregation Methods

In this table we can see different aggregation methods and how Fedavg serves our purpose in further analysis of communication in FL.

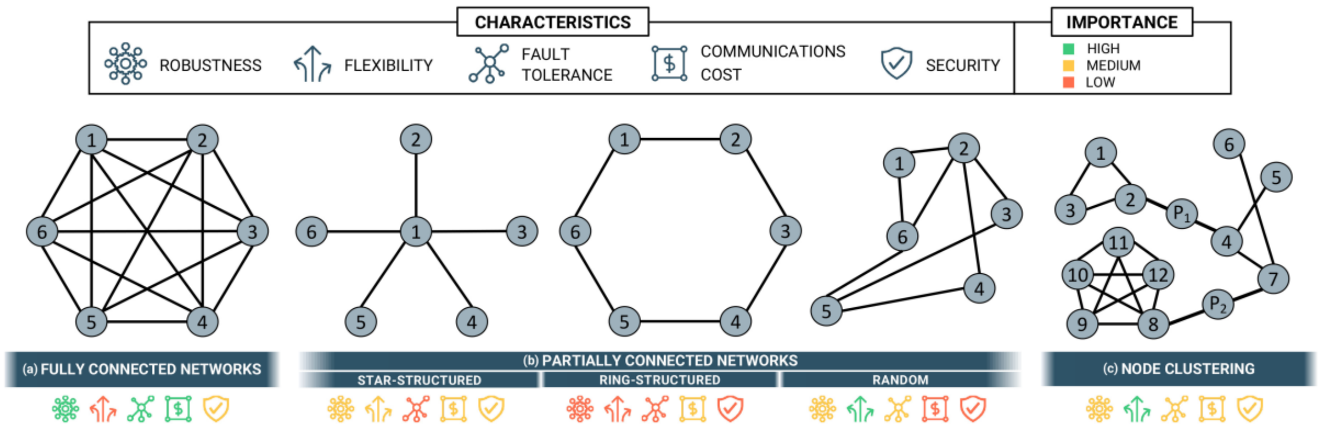


Reference	Contribution	Challenges				Deployment	Learning Model	Dataset	Solution-based	Privacy
		C1	C2	C3	C4					
B. McMahan et al. 2016	FedSGD	✗	✗	✗	✗	Cloud-based	CNN LSTM	MNIST SHAKESPEARE	Gradients averaging	✗
B. McMahan et al. 2017	FedAVG	✗	✗	✓	✗	Cloud-based	CNN LSTM	MNIST SHAKESPEARE	Weights averaging	✗
Chen et al. 2018	FedMeta	✓	✗	✓	✓	Cloud-based	CNN ANN	LEAF real-word production	Meta-learning	*HE*
Arivazhagan et al. 2019	FedPer	✓	✗	✓	✗	Cloud-based	ResNet-34 MobileNet-v1	CIFAR FLICKR-AES	Personalized FL averaging	✗
D. Li et al. 2019	FedMD	✓	✗	✗	✗	Cloud-based	CNN	LEAF CIFAR	TL and knowledge distillation	✗
Xie et al. 2019	FedAsyn	✓	✓	✗	✗	Cloud-based	CNN LSTM	CIFAR WikiText-2	Asynchrounous FL	✗
T. Li et al. 2020	FedProx	✓	✓	✗	✗	Cloud-based	MLR LSTM	LEAF SHAKESPEARE	Adaptative local objectif	✗
H. Wang et al. 2020	FedMA	✓	✗	✗	✗	Cloud-based	CNN LSTM	LEAF SHAKESPEARE	Layer-wise averaging	✗
Khan et al. 2020	FedGKT	✓	✓	✗	✗	Cloud-based	CNN	CIFAR	Asynchrounous Knowledge transfer	✗
Briggs et al. 2020	FL+HC	✓	✗	✗	✗	Cloud-based	CNN	FEMNIST	Agglomerative hierarchical clusetring	✗
J. Yao et al. 2020	ALTD	✗	✓	✗	✗	Fog-based	ANN	generated	Computational resources optimization	✗
Xu et al. 2019	ELFISH	✗	✓	✓	✗	Cloud-based	Alexnet LeNet	MNIST CIFAR	Models compression & Soft-training	✗
Saha et al. 2020	FogFL	✓	✓	✓	✗	Fog-based	CNN	MNIST	Greedy heuristic	✗
Khan et al. 2020	Stackelberg+FedAVG	✓	✓	✓	✗	Edge-based	CNN	MNIST	Stackelberg Game	✗
Qu et al. 2020	D2C	✓	✗	✗	✓	Edge-based	CNN	CIFAR	DANE optimizer	blockchain
Liu et al. 2021	FL-VIDS	✗	✗	✗	✓	Edge-based	MLP	DDCup99	Models averaging	DP-blockchain
Ek et al. 2021	FedDist	✓	✗	✗	✗	Cloud-based	CNN	LEAf	Distance similarity & layer-wise training	✗
Tian et al. 2021	Asynch-DC-Adam	✓	✓	✗	✗	Cloud-based	CNN ANN	MNIST CICIDS-2017 IoT-26	ADAM Optimization in Asynchronous FL	✗
Hu et al. 2021	MHAT	✓	✗	✓	✗	Cloud-based	CNN	MNIST	Knowledge distillation	✗
Y. Li et al. 2021	FedH2L	✓	✗	✓	✗	Cloud-based	CNN	MNIST OFFICE HOME	Knowledge distillation	✗
Ahmed et al. 2021	FTL+FedAVG	✓	✓	✗	✗	Cloud-based	CNN	CIFAR	Hierarchical FL	✗
Guendouzi et al. 2022	FedGA	✓	✗	✓	✗	Cloud-based	CNN	MNIST	Genetic algorithms	✗
Berghout et al. 2022	FTL-RLS	✓	✓	✓	✗	Cloud-based	ResNet	CIFAR	Recursive least squares algorithm	✗
Palihawadana et al. 2022	FedSim	✓	✗	✓	✗	Cloud-based	CNN	EMNIST	Principal component analysis	✗
C.-H. Yao et al. 2022	FMTDA	✓	✓	✓	✗	Cloud-based	CNN	five digits cross-city	Domain adaptation	✗

A comprehensive analysis of recent contributions sheds light on how they address prevalent challenges in FL, including statistical heterogeneity (c1), system heterogeneity (c2), expensive communication (c3), and privacy concerns (c4). The surveyed contributions, particularly those related to FL aggregation methods, are systematically organized and contrasted based on these differentiating criteria. Additionally, various aspects such as the deployment setting of FL (Cloud, Fog, or Edge), the learning model, the dataset employed for simulation, foundational solution strategies, and techniques to ensure privacy are considered.

FL frameworks are equipped to handle a vast array of remote participant devices, spanning millions of intelligent entities. These entities collaborate by sharing extensive AI models, like neural networks, with the central aggregator. Such models, comprising millions of parameters, demand regular updates to achieve optimal convergence. However, they frequently contend with constrained network bandwidth, presenting a substantial communication cost challenge in FL. This challenge directly impacts the efficiency, scalability, and overall performance of the learning process, making it a pivotal focus area.

2.4 | Network Topologies in FL



(DFL network topologies)

Learning (DFL) networks, highlighting their characteristics and the relative importance of these characteristics. Starting from left to right we have: (a) Fully connected networks (b) Partially connected networks (c) Node clustering Understanding these topologies is crucial for selecting the most suitable network structure based on specific system requirements. Fully Connected Networks

are characterized by every node being connected to every other node, forming a highly robust and flexible network. These networks exhibit high fault tolerance and medium security due to their extensive interconnections. However, this comes at the cost of high communication overhead, making them resource-intensive. Despite their advantages, the high communication cost can be a significant drawback in resource-constrained environments. Partially Connected Networks encompass several topologies, including star-structured, ring-structured, and random configurations. In Star-Structured Networks, a central hub node

connects to all other nodes, which are not interconnected. This topology offers low fault tolerance since the failure of the central node can disrupt the entire network. However, it benefits from medium communication costs, security, flexibility and robustness. This structure is simple but not ideal for applications requiring high reliability and fault tolerance.

2.5 | Variables and Parameters

This section outlines the key variables and parameters fundamental to understanding the subsequent discussions. It covers concepts such as accuracy (our ruler to verify how promising the framework is), the Adam optimizer update rule, and the cross-entropy loss function, providing a foundation for the technical intricacies of the following content. The section elucidates various elements including the number of samples,, model parameters, learning rate, and other essential components integral to machine learning and optimization processes.

Accuracy:

$$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}}$$

Adam Optimizer Update Rule:

• Initialize parameters:

– m0 = 0 (initial 1st moment vector)

– v0 = 0 (initial 2nd moment vector)

– t = 0 (time step)

• Update biased first moment estimate:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t$$

- **Update biased second moment estimate:**

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$$

- **Compute bias-corrected first moment estimate:**

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t}$$

- **Compute bias-corrected second moment estimate:**

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t}$$

- **Update parameters:**

$$\theta_t = \theta_{t-1} - \alpha \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}$$

Cross-Entropy Loss Function:

$$\text{Cross Entropy Loss} = -\frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{C} y_{ij} \log(p_{ij})$$

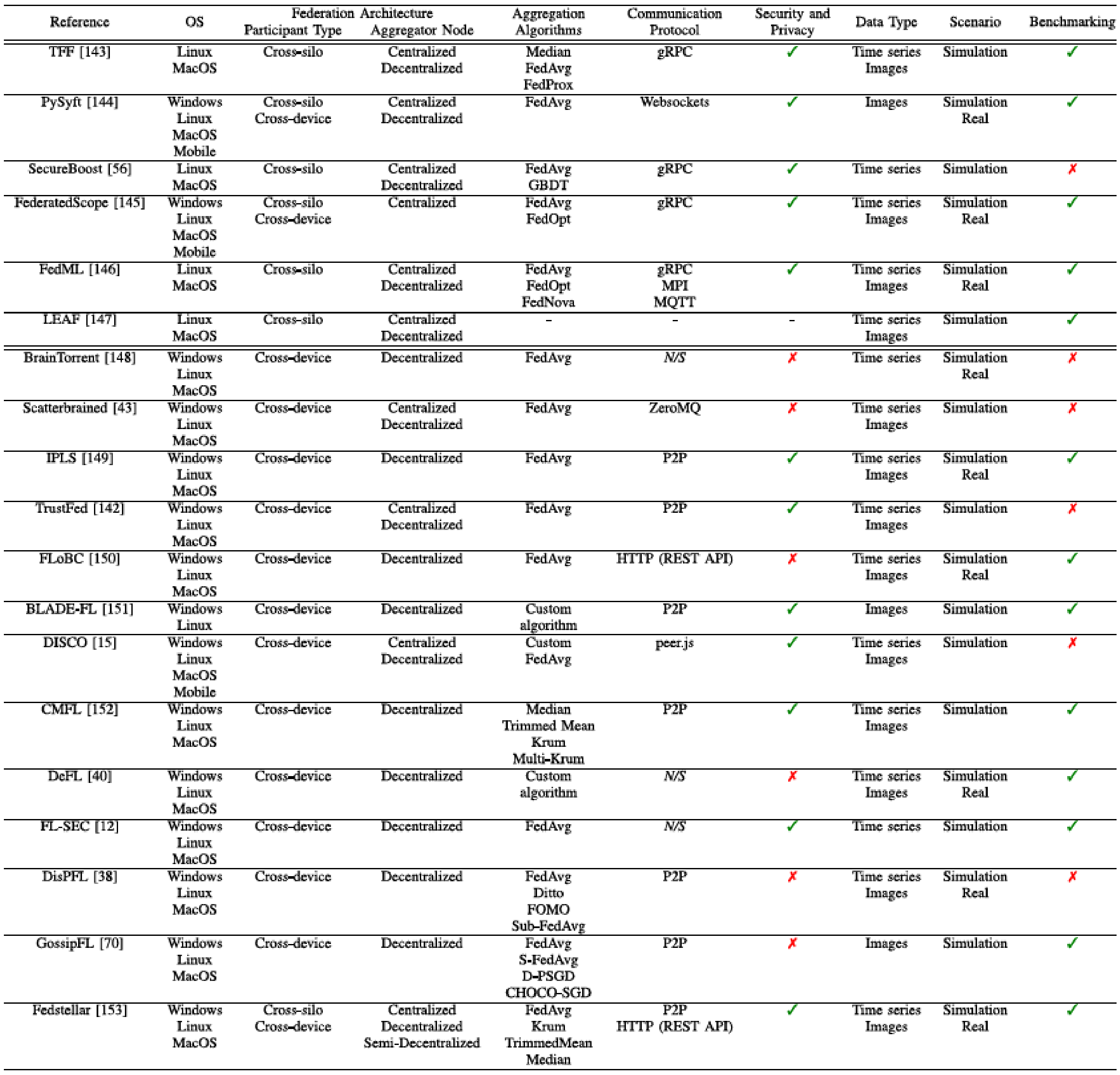
Here, N represents the number of samples, C is the number of classes, Yij is the true label

for sample i and class j, Pij is the predicted probability for sample i and class j, β1 and β2 are the exponential decay rates for the moment estimates, gt is the gradient at time step t, θ are the model parameters, η is the learning rate, ϵ is a small constant to prevent division by zero, and mt and vt are the first and second moment estimates, respectively.

3| Communication Protocols in Federated Learning

The analysis of available literature reveals a rich array of protocols utilized across various domains, as depicted in the table of Comparison of Mature and Incipient DFL Frameworks The entire list is gRPC, WebSocket, P2P, MQTT, ZeroMQ, MPI, and HTTP (REST API).

Moreover, upon further consideration with the inclusion of additional researches such as FedComm, UDP, and TCP will be added to this list.
The pattern observable here is that gRPC is favored by many while others might opt for P2P connections. While some are more innovative like others might not be a secure choice for FL frameworks.
Documented sources indicate the simultaneous deployment of protocols is seldomly seen.



Reference	OS	Federation Architecture		Aggregation Algorithms	Communication Protocol	Security and Privacy	Data Type	Scenario	Benchmarking
		Participant Type	Aggregator Node						
TFF [143]	Linux MacOS	Cross-silo	Centralized Decentralized	Median FedAvg FedProx	gRPC	✓	Time series Images	Simulation	✓
PySyft [144]	Windows Linux MacOS Mobile	Cross-silo Cross-device	Centralized Decentralized	FedAvg	Websockets	✓	Images	Simulation Real	✓
SecureBoost [56]	Linux MacOS	Cross-silo	Centralized Decentralized	FedAvg GBDT	gRPC	✓	Time series	Simulation	✗
FederatedScope [145]	Windows Linux MacOS Mobile	Cross-silo Cross-device	Centralized	FedAvg FedOpt	gRPC	✓	Time series Images	Simulation Real	✓
FedML [146]	Linux MacOS	Cross-silo	Centralized Decentralized	FedAvg FedOpt FedNova	gRPC MPI MQTT	✓	Time series Images	Simulation Real	✓
LEAF [147]	Linux MacOS	Cross-silo	Centralized Decentralized	-	-	-	Time series Images	Simulation	✓
BrainTorrent [148]	Windows Linux MacOS	Cross-device	Decentralized	FedAvg	N/S	✗	Time series	Simulation Real	✗
Scatterbrained [43]	Windows Linux MacOS	Cross-device	Centralized Decentralized	FedAvg	ZeroMQ	✗	Time series Images	Simulation	✗
IPLS [149]	Windows Linux MacOS	Cross-device	Decentralized	FedAvg	P2P	✓	Time series Images	Simulation Real	✓
TrustFed [142]	Windows Linux MacOS	Cross-device	Centralized Decentralized	FedAvg	P2P	✓	Time series Images	Simulation	✗
FLoBC [150]	Windows Linux MacOS	Cross-device	Decentralized	FedAvg	HTTP (REST API)	✗	Time series Images	Simulation Real	✓
BLADE-FL [151]	Windows Linux	Cross-device	Decentralized	Custom algorithm	P2P	✓	Images	Simulation	✓
DISCO [15]	Windows Linux MacOS Mobile	Cross-device	Centralized Decentralized	Custom FedAvg	peer.js	✓	Time series Images	Simulation	✗
CMFL [152]	Windows Linux MacOS	Cross-device	Decentralized	Median Trimmed Mean Krum Multi-Krum	P2P	✓	Time series Images	Simulation	✓
DeFL [40]	Windows Linux MacOS	Cross-device	Decentralized	Custom algorithm	N/S	✗	Time series Images	Simulation Real	✓
FL-SEC [12]	Windows Linux MacOS	Cross-device	Decentralized	FedAvg	N/S	✓	Time series	Simulation	✓
DisPFL [38]	Windows Linux MacOS	Cross-device	Decentralized	FedAvg Ditto FOMO Sub-FedAvg	P2P	✗	Time series Images	Simulation Real	✗
GossipFL [70]	Windows Linux MacOS	Cross-device	Decentralized	FedAvg S-FedAvg D-PSGD CHOCO-SGD	P2P	✗	Images	Simulation	✓
Fedstellar [153]	Windows Linux MacOS	Cross-silo Cross-device	Centralized Decentralized Semi-Decentralized	FedAvg Krum TrimmedMean Median	P2P HTTP (REST API)	✓	Time series Images	Simulation Real	✓

(Comparison of Mature and Incipient DFL Frameworks)

3.1 | REST

Representational State Transfer (REST) stands as a cornerstone in the architectural landscape of networked applications. This paradigm offers a set of principles and constraints that facilitate the creation of scalable and stateless web services. At its core, REST serves as the foundation for modern web services, fostering seamless communication between clients and servers. The crux of RESTful services revolves around the notion of resources, each uniquely identified by Uniform Resource Identifiers (URIs). Communication, in this paradigm, unfolds through standard HTTP methods, including GET, POST, PUT, and DELETE.

One of the foundational principles of REST is statelessness. Servers refrain from maintaining client-specific states, enhancing scalability and operational efficiency. Each client request carries all the necessary information for the server to comprehend and process the request, eliminating the need for the server to store any state about the client session between requests. This stateless nature not only streamlines communication but also contributes to a more resilient and scalable architecture.

REST's embrace of a client-server architecture further solidifies its principles. The separation of concerns between the client and server entities, communicating over a network, establishes a clear and efficient interaction model. Clients initiate requests, and servers respond, fostering improved scalability and flexibility. This architecture also allows for the independent evolution of both the client and server components, contributing to a more modular and maintainable system.

The concept of a uniform interface is pivotal to REST, whe re resources are identified using URIs, and interactions occur through standardized HTTP methods. This uniform and consistent interface simplifies the overall architecture and enhances the visibility of interactions. This includes the utilization of standard methods, resource identification, and representation. Such consistency promotes interoperability and makes the system more comprehensible.

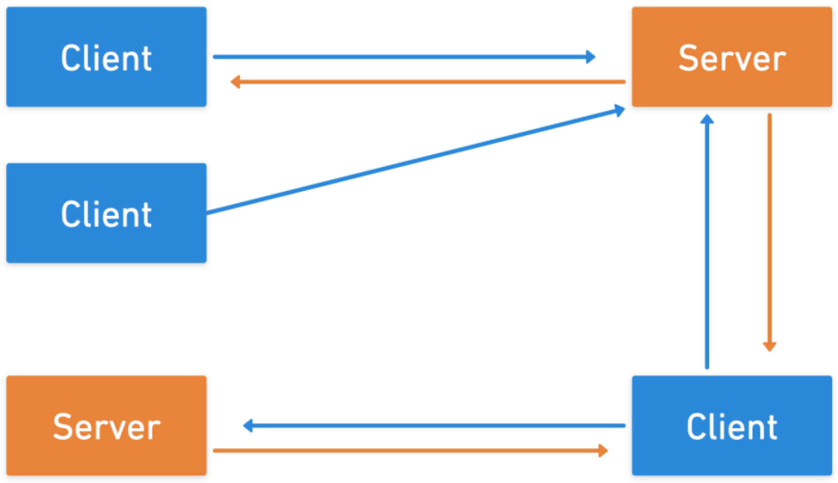
Cacheability is another key principle in REST, allowing clients to cache responses. This feature significantly improves response times and reduces the load on servers, contributing to a more efficient and responsive system. The layered system architecture, a characteristic of REST, accommodates complexity and enhances flexibility by allowing the addition of architectural layers.

In addition to these principles, REST incorporates key features, such as resources that

represent entities in the system and are identified by URIs, standard HTTP methods for various operations, statelessness where each request is independent, and the representation of resources in formats like JSON or XML.
REST finds extensive application in various scenarios. Web APIs, a prevalent use case, leverage REST for enabling communication between web services and client applications. The simplicity and scalability of RESTful services also make them a popular choice for building microservices architectures. Despite its strengths, REST does face certain challenges, such as over-fetching and under-fetching of data, leading to inefficiencies, and limited discoverability of available endpoints and operations without additional documentation. In the realm of applications, RESTful APIs have become ubiquitous in modern web applications, powering diverse platforms such as e-commerce websites, content management systems, and social media networks. In summary, RESTful architectures offer a scalable and straightforward approach to building web services, particularly suited for scenarios where statelessness and simplicity are paramount, as seen in the widespread adoption in web APIs and microservices.

3.2| CoAP

Constrained Application Protocol (CoAP) emerges as a specialized web transfer protocol meticulously crafted to cater to resource-constrained devices and environments, particularly in the context of low-power, low-bandwidth networks like the Internet of Things (IoT). At its core, CoAP is engineered to be both lightweight and efficient, addressing the unique challenges posed by devices with limited processing power and constrained bandwidth.
One of the fundamental principles that define CoAP is its adherence to a Representational State Transfer (RESTful) design, akin to HTTP. This design choice not only simplifies its structure but also ensures ease of use, aligning with the overarching goal of making communication seamless in resource-constrained settings. CoAP mirrors the familiar resource model present in HTTP, enhancing its suitability for devices with limitations.



(CoAP)

CoAP leverages the User Datagram Protocol (UDP) as its preferred transport protocol. This strategic choice allows for a reduction in overhead, a critical consideration given the constraints inherent in the target environments. The utilization of UDP contributes to the protocol's efficiency in terms of both bandwidth utilization and processing power. The lightweight nature of CoAP is a key feature that sets it apart. Tailored to operate efficiently in environments where resources are at a premium, CoAP strives to minimize the burden on devices with limited capabilities. The protocol's emphasis on efficiency extends beyond its design principles to the choice of transport protocol, as it relies on UDP for communication.

Resource discovery is another noteworthy feature of CoAP. The protocol incorporates mechanisms that facilitate the discovery and interaction with resources on devices. This capability enhances the flexibility and usability of CoAP in dynamic environments where the availability and characteristics of resources may change over time.

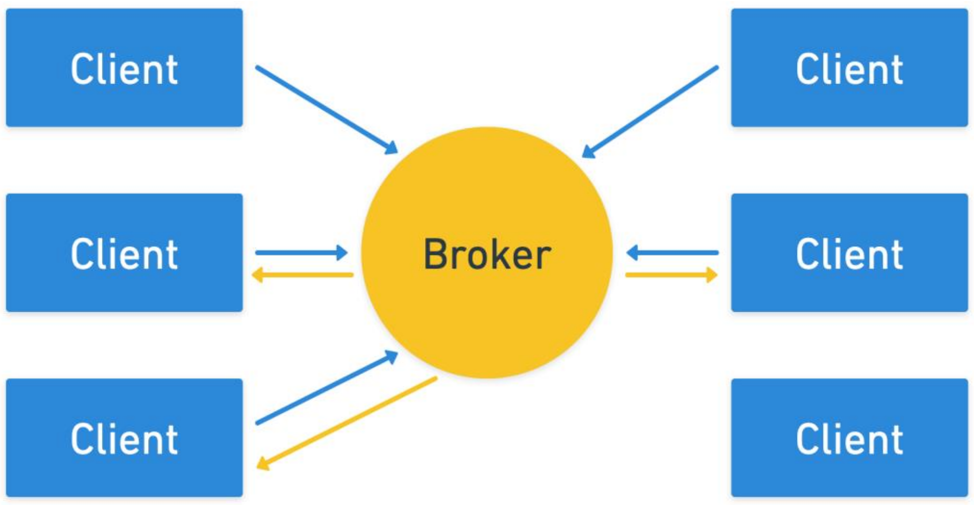
In terms of use cases, CoAP finds a natural fit in scenarios involving IoT devices. Its efficiency in communication aligns with the constraints often present in IoT deployments. Additionally, CoAP is well-suited for wireless sensor networks, offering a lightweight communication solution for scenarios where resources are limited. Its application extends to home automation systems, where devices need to communicate efficiently and effectively.

Despite its strengths, CoAP faces certain challenges. While it is gaining traction, the adoption of CoAP may not be as widespread as HTTP for certain use cases, posing interoperability challenges in diverse ecosystems. Furthermore, as with any IoT protocol, ensuring secure implementations is crucial, and CoAP is no exception to the broader security considerations inherent in the IoT landscape. Also congestion is an important

issue in IoT networks with constrained devices and a growing number of applications.
In summary, CoAP stands out as a protocol meticulously crafted for resource-constrained environments. Its lightweight design, RESTful principles, utilization of UDP, and resource discovery mechanisms make it an ideal choice for communication in IoT scenarios and other contexts where resources are limited. However, challenges such as limited adoption and security considerations underscore the need for thoughtful implementation and consideration of specific use cases.

3.3 | MQTT

Message Queuing Telemetry Transport (MQTT) emerges as a lightweight and highly efficient messaging protocol, purposefully designed for resource-constrained devices operating in environments characterized by low bandwidth, high latency, or unreliable networks.
MQTT has found widespread application in scenarios where devices or applications necessitate the exchange of messages with minimal overhead. Particularly prevalent in the Internet of Things (IoT), MQTT facilitates seamless communication between sensors, actuators, and cloud-based services.

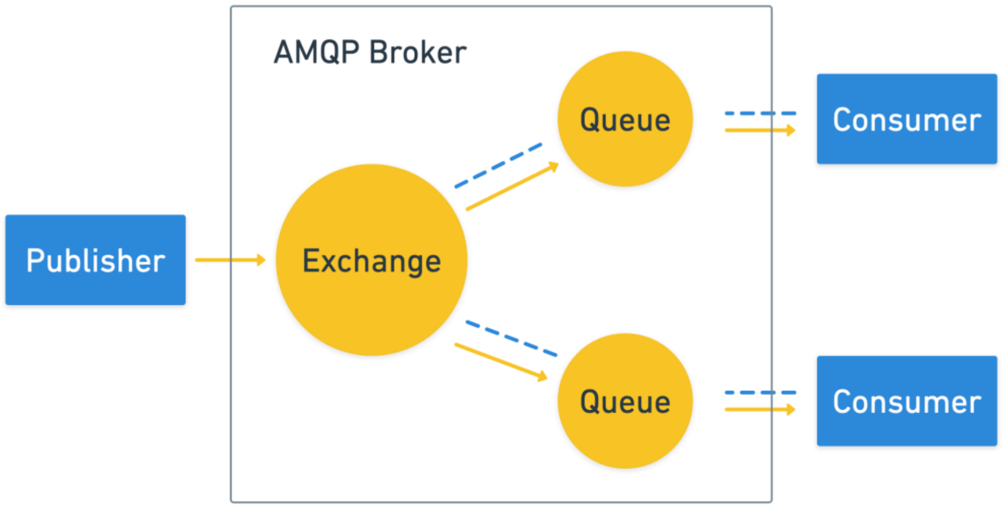


(MQTT)

At the heart of MQTT is the publish-subscribe model, a communication pattern that allows multiple clients to engage in decoupled communication by subscribing to specific topics and receiving relevant messages. This model enhances flexibility and scalability, making MQTT well-suited for environments where components need to exchange information without tight coupling.
MQTT's key features align with its design principles, ensuring optimal performance in challenging network conditions. The protocol is inherently lightweight, catering to the constraints imposed by low-bandwidth and high-latency networks commonly encountered in IoT scenarios. Leveraging the publish-subscribe mechanism, MQTT fosters efficient communication between components using topics, allowing for organized and targeted message distribution.
Quality of Service (QoS) is a pivotal aspect of MQTT, offering three levels for message delivery assurance. This flexibility enables users to choose the appropriate QoS level based on their specific application requirements, balancing factors such as reliability and latency. Additionally, MQTT incorporates a "Last Will and Testament" feature, allowing clients to specify a message to be sent in the event of an unexpected disconnection. This feature enhances the resilience of the communication system by providing a mechanism to convey information about a client's status in case of abrupt disruptions.
Common use cases for MQTT highlight its adaptability to diverse communication scenarios. In the IoT domain, MQTT serves as a cornerstone for efficient and scalable device communication. Its lightweight nature and publish-subscribe model make it suitable for real-time messaging and notifications. Moreover, MQTT is extensively employed in machine-to-machine communication, facilitating seamless interaction between various components or machines.
Despite its strengths, MQTT faces certain challenges. Security considerations are paramount, and while secure versions of MQTT exist, the proper implementation of security measures is crucial to safeguard communication. Additionally, MQTT itself does not manage state, leaving state management responsibilities to be addressed at the application level.
In summary, MQTT stands out as a lightweight and efficient messaging protocol, tailormade for scenarios where low bandwidth, high latency, and reliable messaging are essential. Its dominance in the IoT landscape underscores its ability to meet the specific communication requirements of resource-constrained devices operating in challenging network conditions.

3.4 | AMQP

The Advanced Message Queuing Protocol (AMQP) emerges as a powerful messaging protocol meticulously designed to cater to high-throughput, low-latency messaging systems.
As an open standard application layer protocol for message-oriented middleware, AMQP finds its niche in enterprise applications where the reliable and efficient delivery of messages stands as a critical requirement. This protocol serves as the linchpin in enabling communication between applications or services by delineating a set of messaging patterns.
At its core, AMQP introduces key concepts that underpin its functionality. Message queues play a central role in facilitating the exchange of messages between applications.
This mechanism not only enhances the decoupling of components but also ensures efficient communication in a distributed system. Exchanges, another foundational concept, define how messages are routed to queues, allowing for the implementation of various communication patterns such as fanout, header, and topic-based routing. Complementing these, brokers, acting as intermediaries, facilitate the routing and delivery of messages between producers and consumers, forming a robust foundation for the messaging infrastructure.



(AMQP)

AMQP boasts several key features that contribute to its effectiveness in diverse scenarios.
Asynchronous messaging is a prominent aspect, supporting communication between applications in an asynchronous manner. This not only enhances system responsiveness but also enables components to operate independently, fostering scalability.

Reliable message delivery is another critical feature, ensuring the acknowledgment of messages and, consequently, minimizing the risk of data loss or communication failures. The protocol also excels in providing flexible message routing mechanisms through exchanges, allowing for the adaptation of communication patterns to specific application requirements.
Furthermore, AMQP showcases its cloud-friendliness by being deployable on platforms like Docker, aligning with contemporary cloud-native architectures. It can also provide QoS the same way it was offered by MQTT.
The common use cases of AMQP underscore its versatility and applicability in building robust and scalable systems. Message queues, facilitated by AMQP, become a cornerstone for constructing scalable and decoupled architectures, laying the groundwork for efficient communication between components. In the realm of microservices architectures, AMQP plays a pivotal role, enabling seamless communication between services and contributing to the overall agility of the system. Additionally, AMQP serves as integration middleware, bridging the gap between disparate systems and facilitating interoperability.
However, like any technology, AMQP is not without challenges. Depending on the specific use case, the overhead associated with message queuing may be a consideration, and organizations need to weigh the benefits against the potential complexities.
In summary, AMQP stands as a versatile and powerful protocol, particularly well-suited for building distributed and decoupled systems based on message queues. Its widespread use in scenarios where reliable and asynchronous messaging is paramount attests to its effectiveness in addressing critical communication requirements in contemporary enterprise architectures.

3.5 | TCP Socket

Transmission Control Protocol (TCP) stands as a fundamental and reliable communication protocol, offering a connection-oriented channel between two devices over a network.
In contrast to User Datagram Protocol (UDP), which prioritizes low latency and high throughput, TCP ensures the ordered and error-free delivery of data, making it a cornerstone in various network communication scenarios.
The connection-oriented nature of TCP is a key characteristic that distinguishes it from UDP. TCP initiates a connection before data transfer, employing a three-way handshake

to establish a reliable communication link between the sender and receiver. Once the connection is established, TCP guarantees the reliable delivery of data, addressing concerns related to data integrity, loss, or corruption.

A hallmark feature of TCP is its emphasis on reliable data transfer. The protocol ensures that data sent over the network is received correctly and in the order it was transmitted. This reliability is achieved through acknowledgments, checksums, and sequence numbers, which collectively detect and recover from errors in data transmission.

TCP incorporates flow control mechanisms to prevent overwhelming the receiving device. With windowing and acknowledgments, TCP regulates the pace of data transmission, optimizing the communication process and preventing congestion at the receiving end. Additionally, TCP dynamically adjusts transmission rates based on network conditions, implementing congestion control to optimize overall network utilization.

In terms of features, TCP guarantees both the reliability and ordered delivery of data. It establishes connections through a three-way handshake, ensuring a secure and synchronized communication channel. These characteristics make TCP well-suited for applications where data integrity is paramount, such as file transfer protocols like FTP,web browsing using HTTP, and email communication through protocols like SMTP.

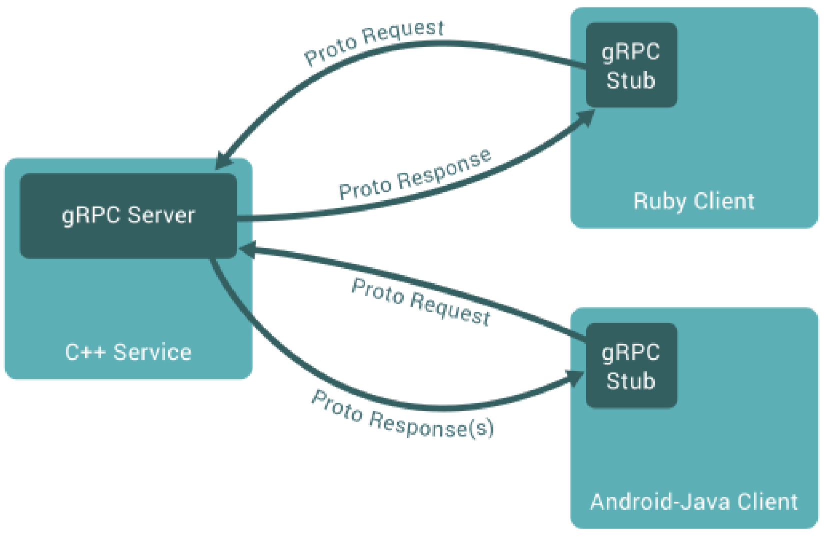
However, TCP does present certain challenges. The reliability mechanisms and connectionoriented nature introduce overhead, making TCP less efficient in scenarios where minimal latency is crucial. In such cases, UDP might be preferred due to its emphasis on low latency and high throughput.

To define TCP socket first we need to define socket; A socket is one endpoint of a two-way communication link between two programs running on the network. The TCP connection available in transport layer can now recieve a socket number.

In summary, TCP serves as a foundational protocol, providing a reliable and ordered communication channel. Its connection-oriented approach and emphasis on data integrity make it an ideal choice for applications where reliability is critical, such as file transfer, web browsing, and email communication. While TCP introduces some overhead, its reliability mechanisms make it well-suited for scenarios where the accurate and ordered delivery of data takes precedence.

3.6 | Remote Procedure Calls: gRPC

gRPC, an open-source RPC (Remote Procedure Call) framework developed by Google, utilizes HTTP/2 for transport and Protocol Buffers as the interface definition language. It enables efficient and structured communication between services, offering a high-performance solution for RPC needs. Unlike tRPC, gRPC is more intricate to employ but delivers superior performance, particularly for large data transfers. Widely adopted in microservices architectures and cloud-based applications, gRPC serves as a cornerstone for modern service-oriented communication.
Protocol Buffers (Protobuf) define the interface and serialization format, facilitating data exchange between client and server. Meanwhile, HTTP/2 leverages the protocol's advantages such as multiplexing and header compression, optimizing communication efficiency. Bidirectional streaming further enhances interaction, supporting simultaneous data streaming between clients and servers. Language-Agnostic support allows developers to work across multiple programming languages, enhancing versatility for building crosslanguage services. Bidirectional Streaming enables data streaming in both directions concurrently, enhancing real-time communication. Additionally, gRPC facilitates code generation for both client and server components based on the defined service structure.



(gRPC Overview)

In gRPC, a client application can directly call a method on a server application on a
different machine as if it were a local object, making it easier for you to create distributed
applications and services. As in many RPC systems, gRPC is based around the idea of
defining a service, specifying the methods that can be called remotely with their parameters
and return types. On the server side, the server implements this interface and runs
a gRPC server to handle client calls. On the client side, the client has a stub (referred to
as just a client in some languages) that provides the same methods as the server.1
gRPC clients and servers can run and talk to each other in a variety of environments -
from servers inside Google to your own desktop - and can be written in any of gRPC's
supported languages. So, for example, you can easily create a gRPC server in Java [3]
with clients in Go, Python, or Ruby. In addition, the latest Google APIs will have
gRPC versions of their interfaces, letting you easily build Google functionality into your
applications.
gRPC finds widespread application in microservices communication due to its efficiency
and suitability for polyglot environments where services are implemented in various programming
languages. Its efficiency and low latency make it particularly suitable for
projects requiring fast and reliable communication between distributed components.
Despite its benefits, developers may encounter a learning curve, especially when grappling
with concepts like Protocol Buffers and bidirectional streaming for the first time. Additionally,
while tooling support for gRPC is growing, some ecosystems may still offer less
mature tooling compared to other RPC frameworks.

gRPC emerges as a robust RPC framework, well-suited for microservices architectures,
particularly in polyglot environments where services span multiple programming languages.

4| State of the Art

Prior to initiating our research on FL, we conducted a thorough review of existing studies.
This allowed us to grasp the current landscape of knowledge, identify gaps, and formulate
a strategic approach to contribute meaningfully to the field.
While some existing research has touched upon certain aspects of communication in FL,

no survey or review has comprehensively examined both network topology and network protocols in the context of FL.

One of the primary challenges in the field of FL is the absence of a comprehensive survey or review that addresses all aspects of communication in FL.This gap in the literature makes it difficult to obtain a holistic understanding of the current state of research and to identify areas that require further investigation.

Existing studies tend to focus on specific components of FL rather than providing a broad overview that integrates various perspectives and findings. Additionally, as research progresses, this challenge is becoming more pronounced; many surveys tend to overlook crucial aspects such as communication efficiency. This oversight further complicates the effort to create a well-rounded understanding of FL, as communication plays a vital role in the efficiency and effectiveness of FL systems. Currently, the only survey making contributions both to network protocols and topologies at the same time is "Decentralized Federated Learning: Fundamentals, State-of-the-art, Frameworks, Trends, and Challenges" which still could contribute more on the protocol aspect.

Despite the challenges, significant opportunities exist for advancing research in FL. Notably, there are several papers that concentrate on specific aspects of FL, in our case communication efficiency can be exemplified by studies like FedComm.

Firstly, we identified several shortcomings in the FedComm framework. One major issue is the assignment of static IP addresses to clients, which restricts their connectivity options. While dynamic IP handling could have been imployed allowing clients to connect through various means and adapting to real-world scenarios where IP addresses are not always static.

5| FedMingle

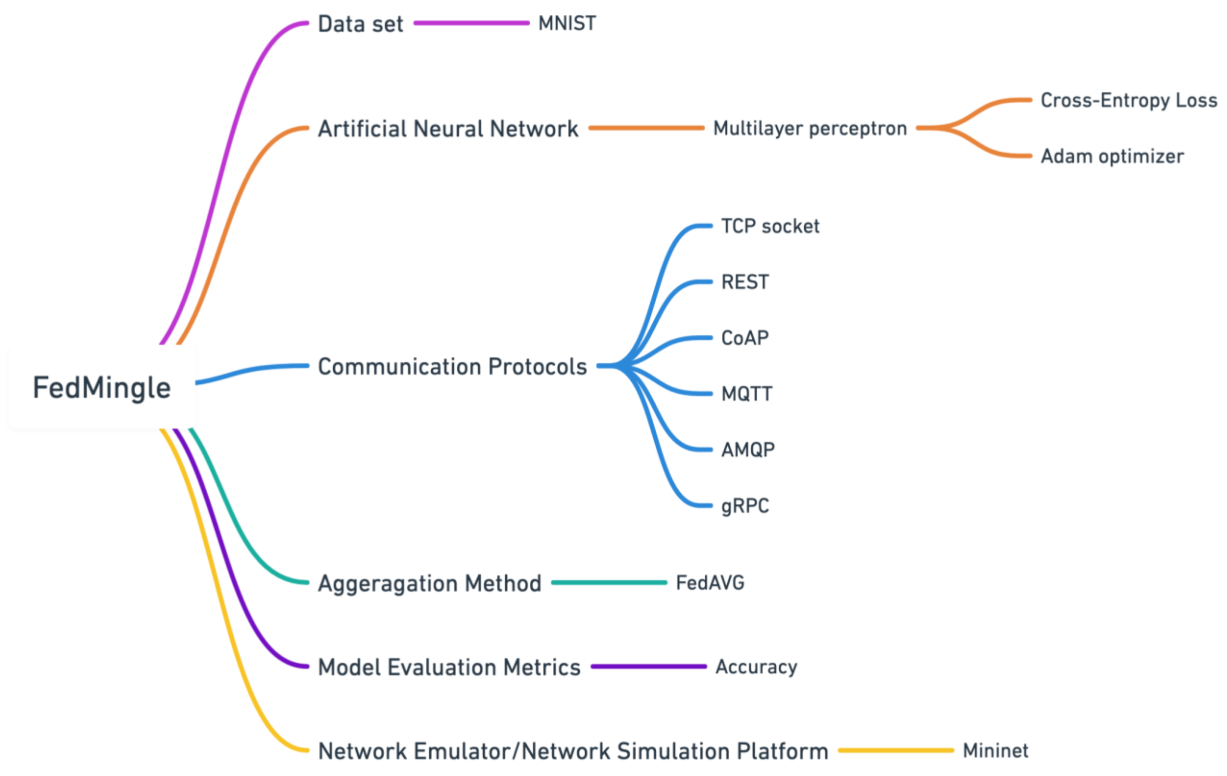
Merriam-Webster Dictionary defines "Mingle" as "to bring or mix together or with something else usually without fundamental loss of identity." In line with our objective to enhance communication between devices in FL, we developed this new framework called FedMingle. "FedMingle is a FL framework designed for collaborative machine learning across distributed systems."

Communication protocols offered in FedMingle such as TCP socket, REST, CoAP, MQTT, AMQP, and gRPC facilitate seamless interaction among distributed nodes. Protocols like gRPC are getting more and more attention in FL frameworks, few examples are FedML, TFF, Flower and APPFL.

The framework utilizes the FedAvg aggregation method to consolidate model updates from participating devices. Evaluation of model performance primarily relies on accuracy metrics, leveraging the MNIST dataset, we employ a Multilayer Perceptron model trained using the Cross-Entropy Loss function and optimized with the Adam optimizer.

We also implemented a complete learning possibility. For instance, some popular frameworks such as Flower and FLUTE only support one local epoch. Our framework is capable of accepting desired number of local epochs. This will help us to check how FedMingle will be performing in action. To simulate real-life scenarios, FedMingle is tested within a network emulation environment provided by Mininet, ensuring realistic simulation of network conditions for comprehensive testing and development.

It is important to note that FedMingle can be installed as a Python package. Moreover, a Dockerized version is available, providing an alternative method for deployment.



(FedMingle Overview)

4.1.1 | Research Design

The research design for this study is centered around the evaluation and comparison of communication protocols used in FL. The primary goal is to assess the performance of different communication protocols in the context of server-client communication within FL systems.

4.1.2 | Data Sources

Through the application of ML techniques, the big amount of data generated by IoT devices can unlock intelligence into different domains, spanning from transportation to industries, from healthcare to agriculture.[Considering scenarios where FL is utilized, we can observe that various entities, each with their own devices, perform ML on their individual datasets, often facilitated by IoT technology.

For example, Smart City is a subdomain of IoT where urban areas are equipped with several sensors and electronic devices that gather data from the environment for different purposes. These collected data are used to monitor metropolitan resources and improve the public services of the cities that directly affect the quality of our lives. A typical Smart City IoT application consists of several wireless sensors as publishers, gateways to exchange the protocols and central servers to analyze and store the data. [10]
For FedMingle framework the primary evaluation dataset choosen will be sourced from real-world dataset such as MNIST which is employed to evaluate the FedMingle's performance in practical applications. This dataset will be divided into portions without any overlap as if each client has its own data to learn from.

6| Fundamentals of federated learning

This section introduces the concepts and working principles of EC and FL. We also summarize the data security and privacy leakage attacks faced by FL systems in current edge- IoT environments.

6.1| Related technologies

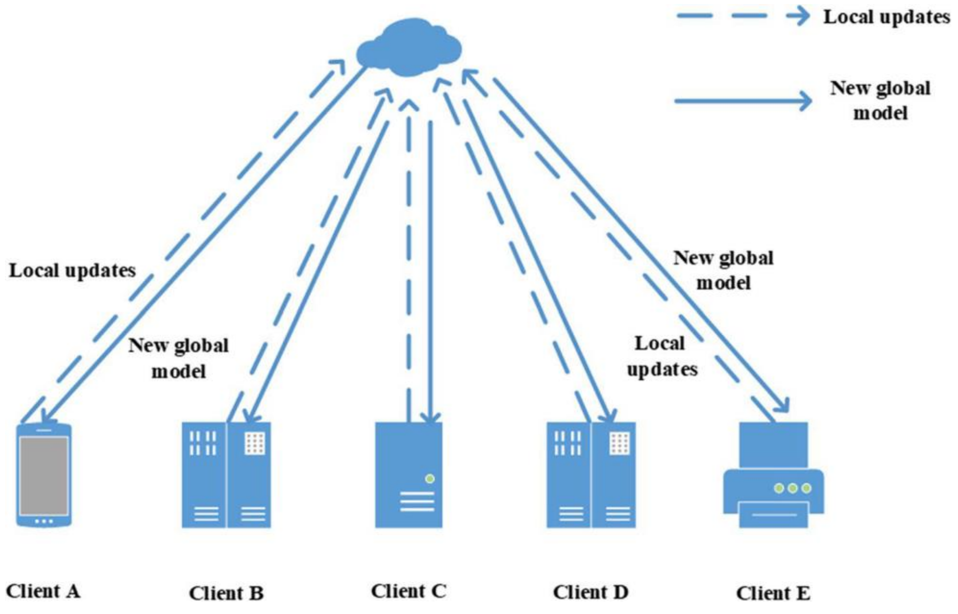
6.1.1| Edge computing

At present, ML training data mainly comes from edge-IoT devices, which not only have a large amount of data, but also a high degree of data heterogeneity. With increasing attention being paid to the security protection of private data, the traditional architecture model of centralized processing in a cloud center can no longer meet the needs of the current technological developments. As a new type of distributed computing technology, the core of the EC technology involves loading raw data into edge network devices (such as edge servers) for storage, processing, and computing. In the edge-IoT, the breakthroughs in EC technology have meant that many ML model trainings can be implemented locally without having to be delivered to a cloud center. The EC processes and analyses data in real time near the data source, providing the advantages of high data processing efficiency, strong real-time performance, and low network latency. This mode is closer to the user and solves requirements at the edge node. It effectively improves the computing processing efficiency, reduces channel pressure, and protects the security of private data.

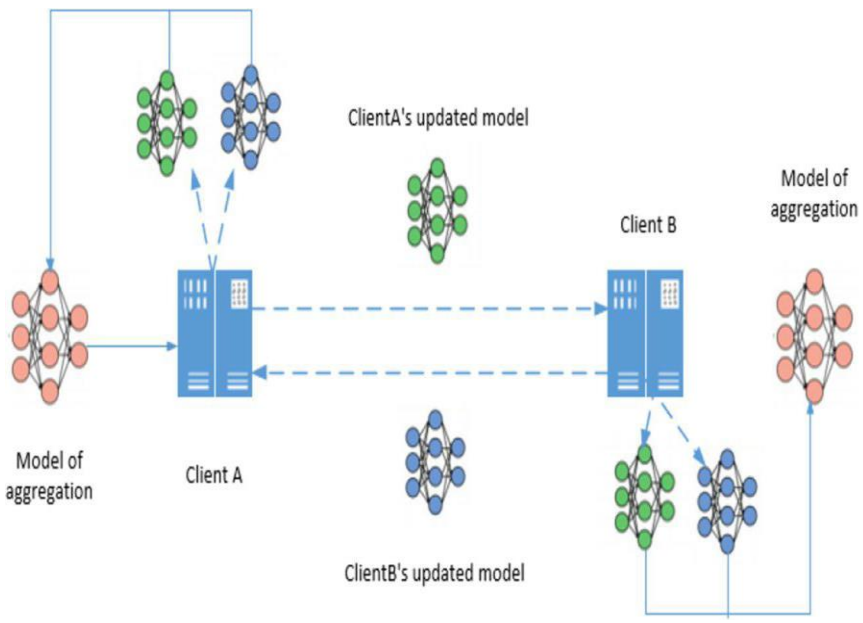
6.1.2| Deep Learning

DL is currently used in a wide range of applications, including computer vision and natural language processing. Terminal devices such as smartphones and IoT sensors generate data to be analyzed in real time using DL and to train DL models. However, DL inferences and training require significant computing resources for quick execution. EC's fine grid of computing nodes placed close to terminal devices is a viable way to meet the high computing and low latency requirements of edge

device DL and also provides privacy technology protection, bandwidth efficiency, and scalability. However, there is a risk that the sensitive information of data owners in the EC environment will be leaked when the data leaves the edge server node during local upload. Service providers are usually honest, but may be curious. FL was initially proposed to provide a collaborative data-training solution. It provides considerable privacy enhancements by coordinating multiple client devices to train a shared ML model without directly exposing the underlying data.



(Federated learning system architecture: client–server)



(Federated learning system architecture: Peer-to-peer network architecture)

6.2| FL data security and privacy leakage

In edge-IoT, a large number of edge-IoT devices are often connected to the Internet, which undoubtedly significantly increases the security risks. Currently, most of the various security attack methods are conducted through the Internet, and the success rate is generally high. In this environment, there are a variety of data security and privacy leakage threats in FL systems. This paper mainly discusses the data security attacks and privacy leakage attacks involved in FL in edge-IoT.

6.2.1| FL data security attacks

FL under EC utilizes the computational power of edge devices to collaboratively train models. However, insecure network communication environments and design flaws of traditional FL structures lead to many data security issues facing the FL process. At the same time, current FL security techniques are difficult to defend against constantly innovative and sophisticated security attacks. Malicious adversaries often use vulnerable edge-IoT devices to intrude into internal targets, destroying the global model convergence as well as the model performance, thus posing security threats to the entire FL process. In the following, the more common data security attacks are described and categorized in detail.

Poisoning attack: In the FL system, a poisoning attack aims to use poisoned data to disrupt the model training and reduce the federated model accuracy. Poisoning attacks can be divided into model and data poisoning.

Model Poisoning attacks are injected into the model trained by the edge nodes. Data poisoning can disrupt the model training by injecting poisoned data into a local dataset of an edge device. Both methods will produce malicious updates to the model training of FL systems, and it is difficult to detect such attackers In the edge-IoT environment, the large and complex number of networked devices participating in the FL system and degrees of device trustworthiness are unknown, posing a potential threat to the security of the FL system. Therefore, it is necessary to develop defensive measures to protect FL systems from poisoning

Sybil attacks: In FL-distributed networks, attackers can use a single node to forge multiple identities. Attackers use these forged identities to control or affect other normal nodes in a distributed network, resulting in network robustness losses intelligent applications need to exchange a large amount of data and information to ensure the efficient operation of different programs, and the distributed network environment provides conditions for Sybil attacks, where attackers can create multiple identities to interfere with the normal operation of the system or steal services for profit .

Traditional FL allows distributed clients to join and exit the system, and attackers can use multiple colluding aliases to join the system to execute attacks, for this reason, some researchers have set up a trusted third party for centralized FL systems to verify false node

identities or use blockchain technology to implement decentralized and reliable FL in untrustworthy networks, and utilize the structural properties of the blockchain and cryptography to defend against Sybil attacks

Backdoor attack: Backdoor attack is when an attacker intentionally inserts a malicious message or "backdoor" into a system in order to trigger malicious behavior under certain conditions. Backdoor attack may cause a system to perform well under normal conditions, but perform illegal and improper tasks when certain conditions are met. Since FL involves multiple participants, an attacker can attempt to insert a backdoor into the local model of some trusted participants and then disrupt the FL process by tampering with gradient updates, sending malicious update parameters. Meanwhile, after multiple iterations of FL, the model parameter aggregation may change and the embedded backdoor may gradually fail.

Byzantine attack: As the most typical attack method in FL, the attacker tries to tamper with the model update parameters submitted by trusted nodes so that the actual model aggregation deviates far from the model convergence direction, resulting in a decrease in the FL model accuracy and serious deviations in the predicted values. This type of security

attack is the most common and is highly effective. Correspondingly, the FL security systems for Byzantine attacks are also being updated. The use of secure robust aggregation is currently recognized as an

effective means for defending against Byzantine attacks Free-riding Attack: In FL, free-riding attack is the process of a free-rider generating false model update parameters to report to the parameter server. Then, the free-rider uses the global model parameters to update its local model but does not contribute its own local data to the global model aggregation. Free-riding attacks reduce the amount of data involved in global model training. Spurious parameter updates can also affect the global

model accuracy .

Adversarial attack: Adversarial attack in FL deceive federated models by adding subtle perturbations to the local raw data to generate adversarial samples. Adversarial attacks can be divided into white-box, black-box, targeted, and non-directed attacks. In a white-box attack, the adversary masters the model and training data information; however, this is generally inconsistent with actual situations. Under a black-box attack, the adversary has little knowledge of the model and training set information. This is in line with the actual scenarios, and is currently the main research direction. Under a targeted attack, a multi-classification ML model classifies and outputs input samples into specified categories. Under a non-directional attack, the adversary uses generative adversarial samples to deceive the FL model. Adversarial attacks help adversaries evade FL system security detection and can

generate poisoned samples to undermine the FL system accuracy.

6.2.2 | FL privacy leakage attacks

In terms of privacy leakage, during the entire training cycle of the FL model, information such as weight updates and gradient updates may be leaked as sensitive private data. During this period, there may be a risk of privacy leakage, whether from the participating clients

of the FL, central servers, or third-party servers. Recent studies have found that even some model gradient information can be leaked as local private data samples or features.

Malicious attackers can also steal the training datasets of a local client in an FL iteration, or rebuild the training datasets through an FL global model inversion.

Model inversion attack: This type of attack method obtains the training set information from a target model that has completed the training. A model inversion attack infers the training set information through a reverse analysis. This information can be the information of members participating in the training and/or certain statistical characteristics of the training set data. For example, an attack method that uses a model inversion attack

to infer actor identity information is called a member inference attack. Member inference

attacks are designed to obtain information by checking for the presence of raw data in the training set. In an FL member inference attack, with each iteration, the data contributor uploads its own model update parameters to the parameter server. At this

time, the server understands the model characteristics and global model parameters of each party. Thus, it can easily be determined whether a specific data sample is in the local training dataset. In some cases, parameter servers can become malicious adversaries and conduct member inference attacks. The remaining participants in the FL system may be potential adversaries. They can determine whether the estimated data originates from the target model training set. Subsequently, the adversaries judge whether the data are member data. During the entire attack process, the FL model parameter contributors don't know the opponent's inference attack behavior, leading to the privacy data of the model contributors being leaked unknowingly .

Refactoring attack: This type of attack focuses on reconstructing all of the training set information of the model contributors or certain sensitive feature categories of the trainingset (e.g., attribute category labels). Several reconstruction attack methods have been proposed, such as generalized adversarial network (GAN) attacks. First proposed a GAN-based data reconstruction attack to steal the private data of model contributors.

When using the GAN to attack the FL, an adversary generates a prototype of the training data of the target by training the GAN. By injecting fake training data samples into the model server, the model contributor is tricked into contributing more local training data. Eventually, the adversary may use this information to restore the local original data of the model contributor and thereby steal sample data. Through GAN attacks, adversaries can also complete data category ,Because the FL system server does not know much regarding the reputation and honesty of the various parties, it is difficult to distinguish a GAN-based data-inference attack. Therefore, when building and maintaining FL security systems, it is necessary to strengthen the identification of and defend to such reconstruction attack methods.

Model extraction attack: In 2016,proposed a model extraction

attack method focused on reconstructing alternative models similar to the target model. In the FL context, the adversary obtains black-box access to the target model. It obtains the return result by sending data in a loop and uses these return values to steal the FL model update parameters or model functions. It restores the FL target model or reconstructs a similar target FL model as much as possible. The above data security and privacy leakage attacks are common attack types in FL.

Table 1 and 2 compare and analyse the methods and effects of the security and privacy attack methods:

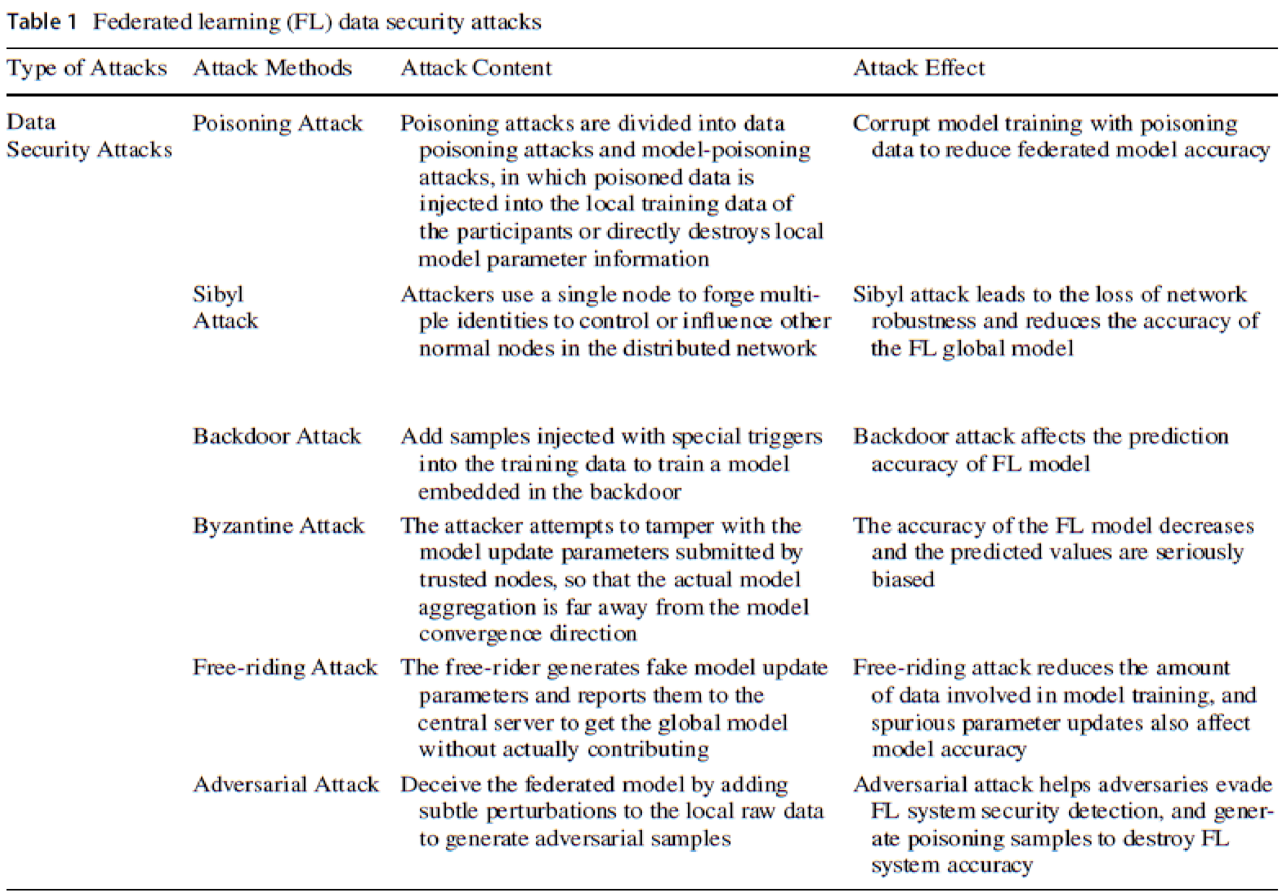


Table 1 Federated learning (FL) data security attacks

Type of Attacks	Attack Methods	Attack Content	Attack Effect
Data Security Attacks	Poisoning Attack	Poisoning attacks are divided into data poisoning attacks and model-poisoning attacks, in which poisoned data is injected into the local training data of the participants or directly destroys local model parameter information	Corrupt model training with poisoning data to reduce federated model accuracy
	Sibyl Attack	Attackers use a single node to forge multiple identities to control or influence other normal nodes in the distributed network	Sibyl attack leads to the loss of network robustness and reduces the accuracy of the FL global model
	Backdoor Attack	Add samples injected with special triggers into the training data to train a model embedded in the backdoor	Backdoor attack affects the prediction accuracy of FL model
	Byzantine Attack	The attacker attempts to tamper with the model update parameters submitted by trusted nodes, so that the actual model aggregation is far away from the model convergence direction	The accuracy of the FL model decreases and the predicted values are seriously biased
	Free-riding Attack	The free-rider generates fake model update parameters and reports them to the central server to get the global model without actually contributing	Free-riding attack reduces the amount of data involved in model training, and spurious parameter updates also affect model accuracy
	Adversarial Attack	Deceive the federated model by adding subtle perturbations to the local raw data to generate adversarial samples	Adversarial attack helps adversaries evade FL system security detection, and generate poisoning samples to destroy FL system accuracy

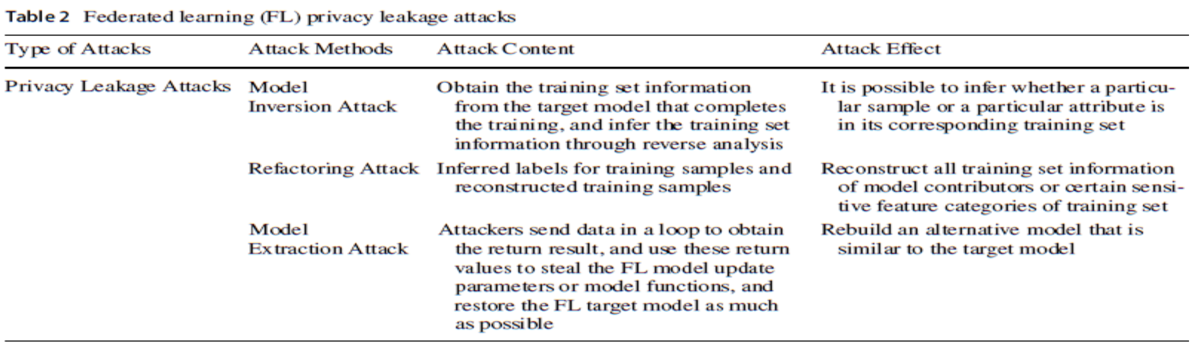


Table 2 Federated learning (FL) privacy leakage attacks

Type of Attacks	Attack Methods	Attack Content	Attack Effect
Privacy Leakage Attacks	Model Inversion Attack	Obtain the training set information from the target model that completes the training, and infer the training set information through reverse analysis	It is possible to infer whether a particular sample or a particular attribute is in its corresponding training set
	Refactoring Attack	Inferred labels for training samples and reconstructed training samples	Reconstruct all training set information of model contributors or certain sensitive feature categories of training set
	Model Extraction Attack	Attackers send data in a loop to obtain the return result, and use these return values to steal the FL model update parameters or model functions, and restore the FL target model as much as possible	Rebuild an alternative model that is similar to the target model

6.3| FL privacy security threats in EC

The traditional cloud-centric computing approaches are gradually proving inadequate in addressing the security challenges posed by the vast amounts of privacy-sensitive data generated by edge terminal devices in the intelligent Internet of Things landscape. EC technology, by processing and analyzing data near its source in real-time, not only caters to the demand at the edge nodes but also significantly enhances processing efficiency, alleviates communication overhead, and safeguards data privacy. Concurrently, FL enables model training on edge devices, making EC an apt environment for FL deployment. Nevertheless, EC, being a nascent distributed computing paradigm, harbors distinctive privacy security risks. The FL model training process within EC confronts similar privacy security threats.

The intricate service model of edge computing, coupled with real-time application requisites and the resource constraints of edge terminal devices, alongside the heterogeneous nature of edge user privacy data from multiple sources, may exacerbate privacy security

concerns during the FL training process. For instance, the exigency for refined security authentication mechanisms at the edge nodes, coupled with the inadequacy of traditional encryption technologies for edge computing environments, poses risks of privacy security breaches, particularly in untrusted execution environments. The primary data security threats and privacy protection challenges encountered by FL within EC encompass:

• Secure sharing and storage of privacy data: The collection of user data by edge terminal devices encompasses sensitive information such as personal location, health data, and identity details. Storing such privacy data in third-party servers, like edge servers, within edge computing environments raises concerns regarding data leakage and unauthorized tampering. Moreover, the presence of numerous unknown trust nodes within the intricate edge network environment poses additional security risks. The connection of FL model learning and training tasks to the network exposes vulnerabilities that malicious adversaries can exploit through various security attack methods. Common network security threats in the EC environment include denial of service attacks,information injection, malicious code attacks, gateway forgery, and man-in-the-middle attacks. The fundamental challenge persists in securely uploading local data to the network or entrusting it to third-party servers for storage and processing.

• Fine-grained authentication access control: EC, being a distributed computing system, operates across multiple trust domains. However, even authorized nodes within

Untrusted network environments face trust issues. Establishing authentication identities

for network nodes across diverse trust domains and ensuring secure access verification between them imposes elevated requirements and challenges on the design of fine grained access control mechanisms within complex edge networks.

7 | Challenges and future directions

We discussed the major security and privacy issues of FL under edge networks. These include the major mechanisms, attacks, and possible countermeasures. However, there are still emerging privacy security challenges and issues that have yet to be explained or require further exploration from the perspective of the EC-assisted IoT paradigm. This section

explains some of these challenges, and provides insights into promising future research directions.

7.1 | Challenges

The rapid development of the SIoT not only promotes the wide application of FL; it also results in higher requirements for FL. This section summarizes the challenges facing the

future development of FL by combining the current development status of FL with the development needs of edge-IoT.

7.1.1 | Secured and efficient edge FL

Currently, numerous research endeavors underscore the compromise of FL model training

efficiency in favor of bolstering privacy safeguards within the FL framework. Specifically, privacy-preserving FL that relies on encryption techniques confronts a delicate balance between model training efficiency and privacy assurance. For instance, the utilization of encryption to secure model parameters introduces a noteworthy consideration: escalating the encryption level leads to an augmented computational overhead. Consequently, with the substantial influx of client interactions in the domain of Edge- IoT, the task of enhancing both model training efficiency and privacy security performance within Edge FL becomes a substantial challenge. This challenge is particularly pertinent as the goal is to establish a dependable FL system within the context of an inherently unreliable edge network environment.

7.1.2| Lightweight byzantine robust FL

A large number of client devices exist in the mobile edge-IoT scenario. FL may not be able to safely and efficiently integrate various model trainers with varying access, identities, purposes, and requirements. Secure authentication authorization is required for participants who access FL systems anytime and anywhere. The effectiveness and safety of training data with complex and varied structures must also be evaluated. The training process must be resilient to malicious enemy poisoning damage and other privacy security threats. The above issues require us to design a set of Byzantine robust security FL architectures for future edge-IoT environments. These architectures can accommodate a large user base and provide lightweight security against hostile Byzantine adversaries.

7.1.3| FL universal safety mechanism design

The research and development of privacy protection mechanisms should be applicable to different FL classification scenarios. The current federal privacy and security protocols are mainly developed for HFL. When applied to VFL and federated transfer learning, they are not only inefficient, but also insecure. Therefore, the development of privacy

protection mechanisms applicable to various FL classification scenarios remains a major challenge for the development of privacy-preserving FL.

7.1.4| Stringent latency demands

Contemporary intelligent IoT applications at the edge impose significant requirements on FL to satisfy the stringent latency prerequisites of client services. Examples encompass intelligent driving reliant on edge terminals and real-time medical diagnosis and analysis.

Nonetheless, the issue of delay stemming from repeated communication rounds in FL curbs its convergence velocity. This delay arises due to FL clients awaiting the aggregation of all local models before initiating a new global model round and commencing the subsequent training phase. Consequently, the delay encountered in the FL communication

process becomes a substantial impediment, substantially constraining the integration and application of FL within the domain of edge-based intelligent IoT networks.

7.2 | Future research directions

This section combines the proposed future challenges for FL in the edge-IoT context as a

direction guide. We propose meaningful research directions for the development of future FL based on new technologies with broad development prospects.

7.2.1 | Secure and efficient FL based on over-the-air computing

The proliferation of edge Internet of Things devices has prompted the necessity for expedited processing of extensive sensor data, resulting in protracted data processing delays.

Over-the-air computing, which merges communication and computation, presents a solution by accomplishing computing tasks during data transmission, potentially mitigating the data processing delay conundrum. Rooted in the concept of "communication and computing integration", over-the-air computing harnesses the signal waveform superposition attribute during transmission to facilitate rapid data aggregation.

The fusion of over-the-air computing with FL involves executing computational tasks while transmitting model updates. By adopting a collaborative design strategy that integrates computation and communication, the expedited aggregation of the FL global model is achieved. Concurrently, the computation for updating FL's local models takes place during the communication phase. This method adeptly alleviates the possible vulnerability of the local model computation process to semi-honest or malicious central aggregation servers,

resulting in a notable enhancement of data confidentiality. In the forthcoming landscape of edge-intelligent networks teeming with terminal devices, over-the-air computing

emerges as a promising avenue to enhance the performance of distributed model training. Exploiting this technology to expedite FL model aggregation warrants comprehensive exploration. Additionally, over-the-air computing leverages the wireless channel's superposition characteristics to thwart privacy breaches during data communication. The optimization of radio resource allocation to ensure heightened confidentiality throughout the FL process in over-the-air computing also necessitates extensive investigation. Hence, the incorporation of over-the-air computing for secure and efficient FL design constitutes a compelling avenue for future research, holding considerable potential in the context of advancing both privacy-preserving and model efficiency.

7.2.2 | Federated transfer learning based on knowledge distillation

The massive data structures in edge-IoT are heterogeneous and cannot be effectively integrated.

Consequently, FL always faces heterogeneity challenges caused by the distribution of non-identically independently distributed data from different clients with different computing and communication

capabilities. Severe data heterogeneity leads to client-side drift, resulting in unstable model convergence and poor performance. Federated transfer learning comprises a combination of FL and transfer learning techniques for allowing knowledge to be shared while protecting private data. This is set up for FL not only for different sample spaces but also for different feature spaces, which can help build effective and accurate ML models for applications with only a small amount of data and weak supervision. KD is a teacher-student training structure. Usually, the teacher model provides knowledge and the student model extracts the knowledge of the teacher model through distillation training.

Knowledge from the complex teacher model is then transferred to the student model at a small cost.

As an attractive option, federated transfer learning employing KD addresses the problems of data heterogeneity and small amounts of training data. By employing KD to perform federated transfer learning from large models to compact models, the FL models can be compressed. Simultaneously, KD can reduce the bandwidth occupied by the FL training and improve the communication efficiency in FL. Therefore, as an effective method to solve data heterogeneity, federated transfer learning with KD can alleviate the costs in large-scale FL training and communication to a certain extent.

7.2.3 | Quantum technology designs lightweight robust FL

An FL security protocol designed using encryption technology can provide a large degree of defense against illegal attacks and prevent privacy leakages. However, the high computing overhead and heavy communication costs of encryption technology in large FL systems render FL less suitable. Therefore, we consider that the high encryption rate, high execution rate, and high security of quantum encryption (QE) technology may be suitable for an FL lightweight secure and robust protocol design. QE technology mainly uses quantum properties and principles. It comprises a series of encryption technologies such as key generation, plaintext obfuscation encryption, ciphertext restoration and decryption, key preservation and transmission, and anti-eavesdropping. Eavesdropping by an intermediate adversary, copying, or tampering may cause a quantum-state change that exposes such eavesdropping. The key to the QE is generated randomly in the process of communication.

This ensures that the key cannot be eavesdropped upon or cracked. In addition, quantum key distribution technology ensures that the communication parties only need to share the secret key once, thereby ensuring the security of transmitting the encrypted information in the open channel.

In the design of a secure and robust FL aggregation protocol, QE technology and applications can be mixed with classical encryption technology and applications. For example, for key management, we can design a

quantum key pool to realize fast key distribution and reduce the waiting time for the key exchange for a large number of edge devices. We can combine quantum key distribution technology to ensure the random and secure generation and sharing of keys. Compared with traditional encryption methods, QE technology can effectively improve the encryption and decryption efficiency in large FL scenarios and simplify the complex key generation and differentiation processes. As a hot research direction in the future, quantum technology can not only be used to design robust edge FL security protocols that are more secure and reliable, but can also help realize lightweight communication, encryption, and decryption.

7.2.4 | Asynchronous FL based on trusted blockchain

Considering the integration of blockchain and FL technologies, there may still be problems caused by blockchain itself. For example, the traditional blockchain consensus mechanism and network structure cause problems such as long transaction confirmation times, limited throughput, and complex communication structures. These problems may lead to an increase in the model update parameter aggregation delay in the blockchain network for each round of the FL process. Each participant in the federation has a different device:

when models are uploaded to the blockchain network separately, the time delay of each device may not be uniform. This could also result in a decrease in the

prediction accuracy of the trained global model. In asynchronous FL, the central aggregation server undertakes global aggregation shortly after amassing a limited set of local models. This prompt aggregation

strategy mitigates the influence of underperforming clients on the overall efficiency of FL global model training.

Asynchronous FL solves the problem of inefficient FL systems caused by the different performances of the edge devices. However, the times of uploading or downloading the model information by each device node are not synchronized. This may cause gradient

delays at some nodes. Asynchronous FL introduces a trusted blockchain as an FL server.

The child blockchains are used for partial model parameter updates, and the main blockchain is used for global model parameter updates. The blockchain is decentralized, transactions are conducted in real time, and the nodes in the various blocks communicate with

each other in a timely manner. These features can also alleviate the problem of gradient staleness in the model training process of asynchronous FL, resulting in degradation of the accuracy of the global model. The trusted identity authentication mechanism of blockchain can filter out unreliable devices that apply to access the FL in the edge-IoT, thereby reducing FL privacy and security risks from the source.

8 | Conclusion

This survey provides an in-depth exploration of federated learning (FL) and its integration with edge computing as a transformative solution to the challenges of training AI models on privacy-sensitive distributed data. By shifting the focus from centralized data collection to local model training, federated learning not only mitigates the risks associated with data breaches but also addresses the scalability challenges inherent in managing the massive data streams generated by smart IoT devices. Comprehensive review of the core principles of FL—including different architectures, data distribution models (IID vs. non-IID), and aggregation methods—emphasized the balance between efficiency and security. The survey addressed the multifaceted nature of data security threats in scalable mesh environments, detailing attacks ranging from poisoning and Sybil attacks to sophisticated model inversion and refactoring techniques. These discussions highlight persistent vulnerabilities and the need for robust and innovative countermeasures.Furthermore, the integration of edge computing into the cloud computing framework presents both an opportunity and a challenge. While edge computing enhances real-time data processing and reduces communication latency, it also introduces new security concerns. The survey identified several promising future research directions, such as over-the-air computing, federated transfer learning with knowledge distillation, and the application of quantum cryptography techniques. These avenues aim to increase the security of machine learning systems while improving their computational efficiency and scalability in dynamic, resource-constrained environments.In short, while federated learning represents a major advance in privacy-preserving AI, its full potential will only be realized through continued innovation in securing and improving edge networks. As we move toward an increasingly interconnected world, addressing these challenges will be critical to developing flexible and efficient AI systems that can meet the demands of modern data-driven applications.

References

Foundational Works

1. **McMahan,H.B.,Moore,E.,Ramage,D.,Hampson,S.,&yArcas,B.A. (2017).**

   Communication-EfficientLearnin gof Deep Networks from Decentralized Data. **AISTATS.** arXiv:1602.05629
   Introduces Federated Averaging (FedAvg), a foundational algorithm for federated learning.

2. **Konečný,J.,McMahan,H.B.,Ramage,D.,&Richtárik,P. (2016).**

   Federated Optimization: Distributed Machine Learning for On-Device Intelligence. **arXiv.** arXiv:1610.02527
   Early work on optimization challenges in federated settings.

Surveys and Reviews

3. **Yang,Q.,Liu,Y.,Chen,T.,&Tong,Y. (2019).**

Federated Machine Learning: Concept and Applications.
ACM Transactions on Intelligent Systems and Technology (TIST). [DOI]
A comprehensive survey on federated learning concepts and applications.

4. **Kairouz, P., McMahan, H. B., Avent, B., et al. (2021).**
Advances and Open Problems in Federated Learning.
Foundations and Trends® in Machine Learning. [arXiv:1912.04977]
Detailed review of challenges, methods, and future directions.

Privacy and Security

5. **Zhu,L.,Liu,Z.,&Han,S. (2019).**

Deep Leakage from Gradients.
NeurIPS. [arXiv:1906.08935]
Demonstrates privacy risks via gradient inversion attacks.

6. **Bonawitz,K.,Ivanov,V.,Kreuter,B.,etal. (2017).**
Practical Secure Aggregation for Federated Learning on User-Held Data.
CCS. [DOI]
Proposes secure aggregation protocols for privacy preservation.

7. **Geyer,R.C.,Klein,T.,&Nabi,M. (2017).**

Differentially Private Federated Learning: A Client-Level Perspective.
arXiv. [arXiv:1710.06963]
Integrates differential privacy in federated learning.

Optimization and Heterogeneity

8. **Li,T.,Sahu,A.K.,Zaheer,M.,etal.** (2020).

Federated Optimization in Heterogeneous Networks.
MLSys. [arXiv:1812.06127]
Introduces FedProx for handling non-IID data.

9. **Zhao,Y.,Li,M.,Lai,L.,etal.** (2018).

Federated Learning with Non-IID Data.
arXiv. arXiv:1806.00582
Analyzes challenges of data heterogeneity.

Frameworks and Systems

10. **Bonawitz,.,Eichner,H.,Grieskamp,W.,etal.** (2019).

    Towards Federated Learning at Scale: System Design.
    MLSys. arXiv:1902.01046
    Describes TensorFlow Federated (TFF) framework.

11. **Caldas,S.,Duddu,S.M.K.,Wu,P.,etal.** (2018).

    LEAF: A Benchmark for Federated Settings.
    MLSys Workshop. OpenReview
    Benchmark suite for federated learning (e.g., FEMNIST).