Self-adaptive Federated Learning in Internet of Things Systems: A Review

ABDULAZIZ ALJOHANI, Cardiff University, UK
OMER RANA, Cardiff University, UK
CHARITH PERERA, Cardiff University, UK

In recent years, Federated Learning (FL) and the Internet of Things (IoT) have enabled numerous Artificial Intelligence (AI) applications. FL offers advantages over traditional Machine Learning (ML) and Deep Learning (DL) by shifting model training to the edge. However, the dynamic nature of IoT environments often interferes with FL's ability to converge quickly and deliver consistent performance. Therefore, a self-adaptive approach is necessary to react to context changes and maintain system performance. This paper provides a systematic overview of current efforts to integrate self-adaptation in FL for IoT. We review key computing disciplines, including Self-Adaptive Systems (SAS), Feedback Controls, IoT, and FL. Additionally, we present (i) a multidimensional taxonomy to highlight the core characteristics of self-adaptive FL systems and (ii) a conceptual architecture for self-adaptive FL in IoT, applied to Anomaly Detection (AD) in smart homes. Finally, we discuss the motivations, implementations, applications, and challenges of self-adaptive FL systems in IoT contexts.

CCS Concepts: • **Computing methodologies → Multi-agent systems** ; • **Computing methodologies → Anomaly detection** ;

Additional Key Words and Phrases: IoT, Smart Home, Federated Learning, Resilience, Adaptation

ACM Reference Format:
Abdulaziz Aljohani, Omer Rana, and Charith Perera. 2025. Self-adaptive Federated Learning in Internet of Things Systems: A Review. *J. ACM* 37, 4, Article 111 (2025), 36 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

The Internet of Things (IoT) transforms cities, homes, and workplaces into interconnected, efficient, and responsive environments [78]. By 2030, the IoT ecosystem is projected to encompass 125 billion devices [24]. This explosive growth has enabled numerous Artificial Intelligence (AI) applications that leverage previously unavailable massive datasets [86]. However, data privacy concerns prevent the direct use of this data for AI applications. As a result, Google introduced Federated Learning (FL) to allow AI systems to learn from distributed data without compromising privacy. FL has found applications across diverse domains. In healthcare, medical organisations can collaborate on training Machine Learning (ML) diagnostic models while keeping patient records secure. Manufacturing companies can jointly develop early fault detection systems through shared model training. Even domestic users benefit from FL in preventing cyber-physical attacks on IoT devices. This convergence of IoT and FL has led to a new paradigm called the Internet of Federated Things (IoFT) [51]. IoFT enables devices to build intelligent analytics and models collaboratively

Authors' addresses: Abdulaziz Aljohani, Cardiff University, P.O. Box 1212, Cardiff, Wales, UK, 43017-6221; Omer Rana, Cardiff University, P.O. Box 1212, Cardiff, Wales, UK, 43017-6221; Charith Perera, Cardiff University, P.O. Box 1212, Cardiff, Wales, UK, 43017-6221.

while maintaining data stored locally. Additionally, IoFT offers multiple advantages beyond privacy preservation, including cost-effectiveness, diversity, and reduced computational requirements. Nevertheless, this new paradigm faces rising complexity due to unpredictable operating environments (e.g., weather variations and service delivery uncertainties). As a result, the Self-adaptive System (SAS) has appeared as a solution, enabling autonomous responses that maintain service quality [40]. This paper examines how the research community has leveraged SAS properties to overcome IoFT challenges. Table 1 provides a reference for all abbreviations used in this article.

1.1 Existing Surveys

Numerous studies have demonstrated the viability of SASs across various contexts. Table 2 summarises the analysis of prior research on SASs and self-adaptive properties in IoFT systems. Interest in SASs increased after IBM introduced the vision of autonomic computing in 2001 [40] that emphasised the need for systems capable of autonomous operation to manage complexity. Early reviews of autonomic computing explored motivations, methodologies, and applications but did not include detailed discussions on SAS evaluation [44]. Salehie and Tahvildari [96] provided a more extensive analysis, using a question-based framework to examine self-adaptive properties. Building on this, Krupitzer et al. [53] developed a taxonomy of self-adaptation mechanisms, offering a comprehensive classification. Elhabbash et al. [28] extended these discussions to software engineering, covering definitions and engineering practices for self-awareness in software systems. While these surveys provide comprehensive insights into SAS, few explicitly examine the uniqueness of self-adaptive properties in IoFT systems. Some studies partially address these aspects. For instance, Abdulrahman et al. [1] explore self-optimisation for resource management in federated environments, Zhang et al. [117] propose solutions for privacy and statistical heterogeneity, and Bellavista et al. [10] emphasise communication efficiency and privacy. However, these efforts do not explicitly focus on integrating SAS within the IoFT domain. Petri et al. [84] work shares a few similarities to our work, yet the main focus of their work is to apply resource automation to reduce the complexity requirements of industrial workflows for edge native applications.

1.2 Contributions

Despite the availability of systematic reviews on both SAS and FL, most existing studies treat these topics independently. Additionally, no comprehensive study has addressed self-adaptive IoFT systems. Therefore, our contributions are outlined as follows:

- We have thoroughly evaluated literature related to the implementation of self-adaptive IoFT systems.
- We briefly define SAS, FL, and IoT and their unique characteristics and relationships.
- We introduce our taxonomy, highlighting the main properties of self-adaptive IoFT systems.
- We cover aspects of self-adaptive IoFT systems, such as the context, motivations, implementations and their use in key application areas, research challenges and future directions.
- We introduce a conceptual architecture with MAPE-K feedback loop for self-adaptive IoFT, demonstrated through Anomaly Detection (AD) in smart home environments.

1.3 Paper Structure

The rest of this paper is structured as follows: Section 2 introduces key topics and terminology for IoFT. Section 3 highlights the methodology for this survey. Section 4 presents the definition, motivation, implementation, and application of self-adaptive IoFT. In this section, we also introduce feedback control for self-adaptive IoFT and demonstrate how it works. An AD classification for

Table 1. List of Abbreviations

Phrase	Acronym	Phrase	Acronym
Internet of Federated Things	IoFT	Research Question	RQ
Federated Learning	FL	Inclusion Criteria	IC
Internet of Things	IoT	Exclusion Criteria	EC
Industrial Internet of Things	IIoT	Transmission Control Protocol	TCP
Self-adaptive System	SAS	User Datagram Protocol	UDP
Machine Learning	ML	Internet Protocol	IP
Reinforcement Learning	RL	Remote Procedure Call	RPC
False Positive	FP	True Positive	TP
Deep Learning	DL	Software-Defined Networking	SDN
Anomaly Detection	AD	Virtual Machine	VM
Artificial Intelligence	AI	Systematic Literature Review	SLR
Independent and Identically Distributed	IID	Stochastic Gradient Descent	SGD
Non-Independent and Identically Distributed	Non-IID	Neural Network	NN
Network Functions Virtualization	NFV	Systematic Literature Review	SLR
Supervisory Control and Data Acquisition	SCADA	Programmable Logic Controller	PLC

Table 2. Previous Survey Comparison

|Paper	|Year	|Method	|Domain	|Focus	|Studies		|Self-adaptation System					
					considered	analysed	definitions	motivations	methods	evaluations	applications	IoFT
[44]	2008	Adh	AC	C|H|O|P	N/A	N/A	Ex	✓	✓	x	✓	x
[96]	2009	Adh	SE	C|H|O|P|A|D|CA	N/A	N/A	Ex	✓	✓	✓	✓	x
[53]	2015	Adh	AC	C|H|O|P|A|D|CA	N/A	N/A	Ex	✓	✓	✓	✓	x
[28]	2019	SLR	SE	H|P|A|D|CA	865	74	Ex	✓	✓	✓	✓	x
[3]	2022	SLR	CPS	H|P|A|D|CA	266	30	Ex	✓	✓	✓	✓	x
[1]	2021	Adh	FL	O	N/A	N/A	x	P/C	P/C	P/C	✓	✓
[117]	2021	Adh	FL	D|P|L|CA	N/A	N/A	Im	P/C	P/C	P/C	✓	✓
[10]	2021	Adh	FL	D|P|CA	Ex	N/A	Im	P/C	P/C	P/C	✓	✓
This Paper		SLR	FL	C|H|O|P|A|D|CA	290	17	Ex/Im	✓	✓	✓	✓	✓

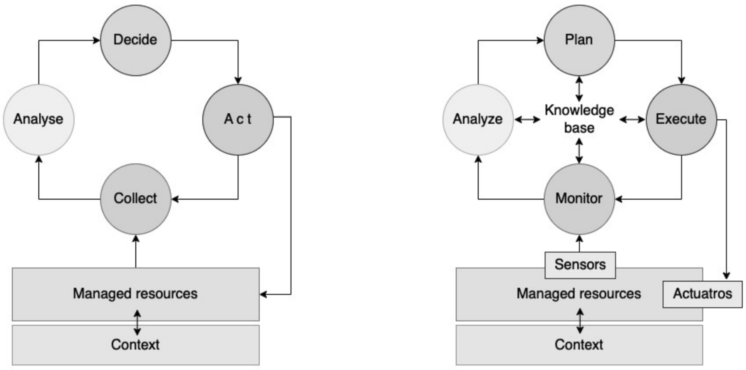
C: Self-configuring **H**: Self-healing **O**: Self-optimization **P**: Self-protecting **A**: Self-awareness, **D**: Self-adaption **M**: Self-managing **L**: Self-learning, **CA**: Context-awareness **SLR**: Systematic literature review **Adh**: Ad hoc **Ex**: Explicit **Im**: Implicit **IoFT**: Internet of Federated Things, **FL**: Federated learning **SE**: Software engineering **AC**: Autonomic computing **CPS**: Cyber–physical system **P/C**: Partial coverage, **N/A**: Not available

a smart home use case is presented in Section 4.8 to apply the conceptual architecture for self-adaptive IoFT. We discuss challenges and future research directions for self-adaptive IoFT in Section 5. Finally, we provide concluding comments in Section 6.

2 BACKGROUND

2.1 Self-adaptive IoT Systems

SASs address the challenges of managing complex systems under runtime uncertainty. The research community introduced the concept of SAS [96] [49] [31], describing a system that can configure itself using a variety of mechanisms to preserve system quality within an unstable environment. Uncertainties can have different forms, such as introducing a new device to the current network, adapting to a new user behaviour, or responding to an unknown event in cloud resources. Therefore, a system that can adapt to the dynamic environment is crucial. The core design philosophy of a SAS is to distinguish between the adaptation logic that maintains or enhances certain system qualities and the managed resources that executes the domain-specific functions of the application. This architectural design principle offers valuable insights for developing self-adaptive IoT systems which operate in dynamic and heterogeneous environments. A number of system architectures and approaches to adaptation have been proposed in the last two decades for designing SASs. We briefly discuss the key conceptual designs for developing and using SAS in the literature.



(a) Autonomic feedback loop

(b) MAPE-K model

Fig. 1. Conceptual designs of autonomic feedback loop and original MAPE-K model

2.1.1 Autonomic feedback loop . Driven by autonomic communication and distributed computing paradigms, Dobson et al. [26] proposed an autonomic control loop as shown in Figure 1a to enable self-adaptation to autonomic communication. Their suggestion states that the system begins gathering information from various sources, such as network traffic, sensor readings, user context, and application requirements. The data is analysed using different techniques, such as decision theory and risk assessment. This analysis is used to capture the system's current state, subsequently used to make decisions. The decision-making stage is followed by an action phase, which provides instructions to a system administrator or another actuation mechanism.

2.1.2 Traditional and MAPE-K Feedback Loop. The MAPE feedback control loop is a widely recognized engineering approach that enables self-adaptation. It consists of four computational components: (M)onitor, (A)nalyze, (P)lan, and (E)xecute [49]. The monitoring stage initiates the control loop by gathering relevant data from the environment to represent the system's current state. Following this, the system enters the analysis stage, which examines the monitoring phase output. Various methods can be used during this stage to organize and interpret the information, which will be discussed further in Section 4.3. Next, in the planning stage, the system defines a set of actions to adapt the managed resources. This allows for a reactive response to events that may arise over time. Finally, in the execution stage, the planned actions and responses are implemented to enable adaptation within the system. Additionally, the MAPE-K feedback loop enhances the original MAPE model by incorporating a knowledge base, which facilitates data sharing across all computational components, as illustrated in Figure 1b.

2.1.3 Architecture-based Self-adaptation. Oreizy et al. [80] present one of the earliest architecture for self-adaptation. Unlike other self-adaptation methods, Oreizy's approach features a two-phase model for SASs. The two phases are evolution management and adaptation management, both of which enclose key processes essential for achieving the system's goals and objectives. In the evolution management phase, various components are utilized to implement changes and consistently gather observations. The primary purpose of this phase is to minimize changes and accidental errors during system operation. Evolution management focuses on applying changes over time while mitigating associated risks. In contrast, the adaptation management phase assesses system behaviour to respond appropriately and determine the necessary adaptations. Critical practices in this phase include monitoring evaluations and planning for change. Overall, this model addresses the challenges of unexpected inconsistencies and errors.

2.1.4 Rainbow Framework. Garlan et al. [31] introduced the rainbow framework, which integrates a system's architectural model in its runtime system in contrast to conventional applications of software architecture as a purely design-time artefact. Developers of self-adaptation capabilities specifically use the software architecture model of a system to track and analyse the system. Therefore, the rainbow framework design consists of two layers: the System layer and the Architecture layer. Separating the system design reduces the development cost and increases the framework's usability since it can be used with any system, such as a legacy system. Rainbow realises self-adaptation by using the model manager to support the constraint evaluator, which provides reasoning features for the current system and identifies the current system's behaviour. The reasoning features will support the planning for the following action through the adaptation engine, which will later be translated to action via the adaptation executor.

2.1.5 The Common Features of SAS. We listed some of the approaches in the literature to enable SASs. These self-adaptation techniques share some common features, which can be summarised into three main observations. First, one crucial feature is separating the control logic and managed resources. This separation enhances the usability and portability of these approaches, as it allows for easy adaptation to new or existing systems. Moreover, it reduces the system's complexity by splitting it into small subsystems. From a system design perspective, the separation allows the engineers to focus more on system operation and maintenance of system quality rather than on the system interface with the adaptation logic component. Second, reducing the number of adaptation logic and managed resource interactions is one of the main objectives. As a result, the system permits costly operations on the adaptation logic component. It returns feedback as one or more actions to apply on managed resources or notify the system administrator. Third, the main structure of the adaptation cycle in these approaches contains several steps. All these steps involve state management, sharing common variables and system configurations to synchronise data across all the components.

The observations above are essential for designing and operating self-adaptive IoFT systems. In the context of IoFT, separating control logic and managed resources aligns with the distributed nature of FL, where adaptation logic typically resides at the FL server, and IoT devices act as managed resources. This separation supports scalability and facilitates the integration of heterogeneous devices. Additionally, minimising interactions between adaptation logic and managed resources is essential for reducing communication overhead, where excessive communication in FL between the FL server and client is undesirable. These principles collectively enable self-adaptive IoFT systems to deliver efficient and scalable performance in diverse, real-world applications.

2.2 Federated Learning in IoT Systems

2.2.1 Definition. Machine Learning use across IoT devices can be both centralised and decentralised, i.e. a central server to carry out learning vs. use of distributed learning – each has its advantages and disadvantages [6]. The FL paradigm for IoT is a new addition to these methods. FL applications can be seen in a variety of fields, including smart city [64], healthcare [107], recommendation systems [109], edge network [114], electric grid [100], vehicular ad hoc network [58] and blockchain [54]. In contrast to centralised ML approaches, FL inherently enhances security and privacy by keeping data localised at the edge. In this framework, data generated on edge devices is used for local training of ML models rather than being transmitted to a central server. Consequently, only model

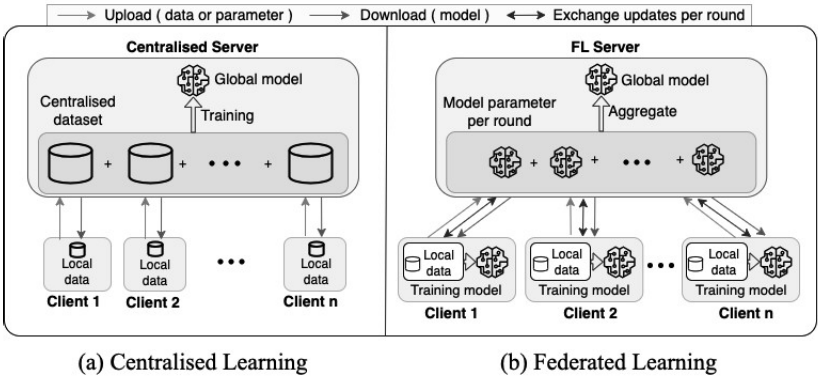


Fig. 2. Architectural design for (a) centralised learning vs. (b) federated learning

parameters are exchanged between the edge devices and the cloud server, as shown in Figure 2. The federated learning process generally includes three main steps:

- Step 1: A FL server initiates the training task and creates an initial global ML model. The FL server also determines the list of contributing clients, the number of rounds and the aggregation process for all incoming parameters.
- Step 2: After receiving the model from the FL server, the FL clients engage in training using their local data. After completing this training phase, they transmit the newly refined model back to the federated server.
- Step 3: After receiving the refined models from FL clients, the FL server aggregates the received model parameters to create an updated global model. Subsequently, the global model is redistributed to all participants, initiating another training round.

These processes continue until the FL server reaches the maximum number of rounds or converges to a known error bound. In the literature, the terms generalisation and personalisation are used to describe different aspects of model performance. In the context of FL, generalisation refers to the model's ability to perform well on unseen data drawn from a distribution similar to the clients' training data, as highlighted in Mora et al. [74]. Personalisation, on the other hand, denotes the model's capacity to quickly adapt to the local data of individual clients, as discussed in Tan et al. [102] and Wang et al. [106]. Additionally, communication efficiency in FL is closely tied to the model's convergence speed. A model that generalises well typically requires fewer communication rounds to reach a convergence point in test accuracy or loss. Therefore, efficient aggregation methods and other optimisation strategies are often employed to improve convergence and reduce communication overhead. The literature introduces many FL aggregation approaches. The most common method is Federated Averaging – FedAvg [72]. In this approach, the algorithm tries to reduce the loss function and reach convergence by averaging the received model's weights from the client, making it the baseline of FL. The FedSGD algorithm is commonly used in Deep Learning (DL), as it aggregates gradients and performs one step of gradient descent [72]. However, FedAvg distributes updated weights instead of gradients, allowing FL clients to perform multiple batch updates on local data. If all FL clients have the same initialisation model parameters, averaging the gradients in FedSGD is equivalent to averaging the weights in FedAvg.

2.2.2 Data Partitioning. An essential stage when designing a FL system is to analyse data distribution over samples and feature spaces. Therefore, the FL system can be typically categorised into horizontal FL, vertical FL, and hybrid FL, as shown in Figure 3.

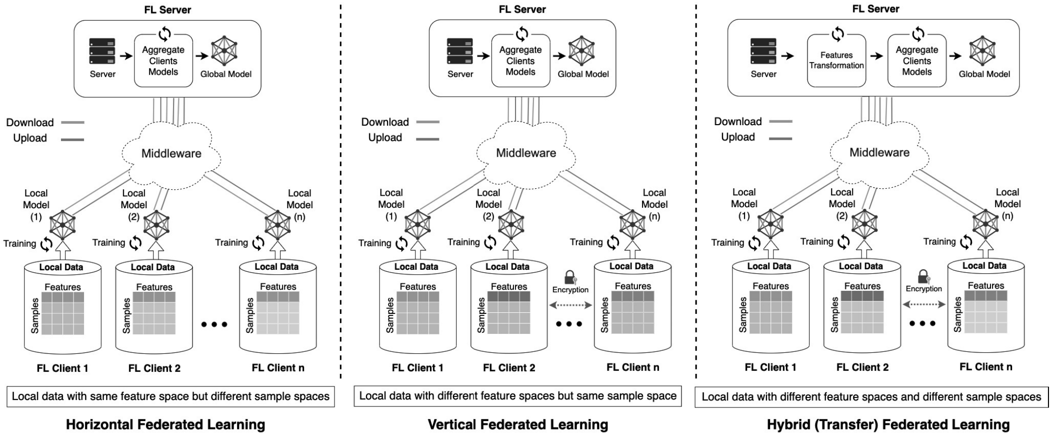

FL Server
Server
Aggregate Clients Models
Global Model
Features Transformation
Download
Upload
Middleware
Local Model (1)
Local Model (2)
Local Model (n)
Training
Local Data
Features
Samples
Encryption
FL Client 1
FL Client 2
FL Client n
Local data with same feature space but different sample spaces
Local data with different feature spaces but same sample space
Local data with different feature spaces and different sample spaces
Horizontal Federated Learning
Vertical Federated Learning
Hybrid (Transfer) Federated Learning

Fig. 3. Data partitioning types in FL

Horizontal / Example-based FL. The datasets of participants may be different or have a limited intersection, yet the feature space is identical. Horizontal FL for partitioning data occurs when multiple participants attempt to enhance ML model performance on similar tasks – one of the most widely used approach in FL research. As *local* data is in the same feature space, participants can train their local models using the same ML model architecture and their local data. Therefore, FedAvg has been used in horizontal FL to average all local models. A typical application is a Wake-word detector [56] and a word prediction application [99].

Vertical / Feature-based FL. The data space between the participants is the same, but it differs in the feature space. For example, assuming we want to make a generalised model that works across different applications, we can combine features belonging to different applications to train one global model. This type of data partitioning requires a different approach to achieve the FL task. For instance, entity alignment techniques can acquire knowledge between participants and identify the overlapping samples between participants [119]. Cheng et al. [22] proposed a new method for vertical FL that allows participants to train gradient-boosting decision trees collaboratively without losing information. Their approach finds commonalities between the data of different FL clients that join the decision tree process, while still preserving FL client's privacy.

Hybrid / Transfer FL. This involves a combination of both horizontal and vertical FL data partition. Another name used in literature is Federated Transfer Learning. The main reason for introducing this hybrid data partitioning is that if we have two or more FL clients with overlapping sample data and features, the FL will perform poorly due to the heterogeneity of data and feature spaces. An example of an application that could benefit from hybrid FL is a marketing company that collects customer data to launch a marketing campaign in different countries. Due to cultural variations, some countries may have more features than others that work well to predict the marketing campaign's success within a country. However, there is a small intersection in the feature space, such as participant age, which can be generalised across countries. Furthermore, due to the different geographic locations of the two marketing campaigns, the data overlap would be negligible. As a result, hybrid FL provides benefit from both data partitioning techniques by transferring learning [81] to another FL client. Liu et al. [65] provide a secure transfer FL system that uses shared features and samples to learn a representation of FL clients.

2.2.3 Federation Scale. The classification of FL scaling encompasses cross-device [111] and cross-application domain (referred to as "cross-silo") [42] based on the number of participants and the volume of data distributed within the federation.

Cross-Silo. It is usually used in cross-domain FL, such as banking or medicine, or for different geographically distributed data centres. Cross-Silo FL leverages the extensive datasets from a small number of FL clients to enhance the training of a global model. The number of FL clients is usually small, typically encompassing companies or organisations. An example of cross-silo includes YouTube, which shows targeted advertisements by training models using data collected from different geographical locations and storing them in the nearest Google data centre. Due to the flexibility of cross-silo FL, data partitioning could be either example-based (horizontal) or feature-based (vertical). Moreover, many FL clients are incentivised to train a model using all their data in the cross-silo context. However, they cannot exchange their data directly due to privacy constraints, regulatory limitations, or even when they cannot organise their data to meet FL's initial data requirements.

Cross-Device. There are potentially a vast number of FL clients, but only a tiny fraction are available at any given time. The type of FL client can vary due to the heterogeneity of devices such as smartphones and IoT devices. Most of these devices have limited computational resources, making them ineffective for performing the training task. In this case, the FL server must be able to handle all revised local models to develop a global training model. For example, Google suggested an FL-based keyboard suggestion using end-user devices to train the keyboard suggestion model locally [36]. Similarly, Apple utilises cross-device settings to teach Siri to recognise various voices [12]. Cross-device scaling is highly prevalent in IoT applications, often involving large-scale, distributed FL clients. However, this setup introduces additional challenges, such as client availability fluctuations, inconsistent local training performance, and communication overhead due to intermittent connectivity and limited device resources. Moreover, the inherent nature of the cross-device FL setting makes it nearly impossible to directly address or index all participating clients. This limitation reduces the system's reliability compared to cross-silo FL, where each client can be easily identified and accessed [42]. These challenges emphasise the need for a self-adaptive IoFT approach that dynamically reduces constraints, ensuring a balance between model accuracy and overall system performance across all FL clients.

2.2.4 FL with Heterogeneous IoT Systems. The heterogeneity of IoT systems poses a significant challenge in FL. Data in these environments are often collected from diverse sources, such as smartphones, cameras and smart sensors, each with varying characteristics like data quality and volume. This results in Non-Independent and Identically Distributed (non-IID) data, a common challenge in FL systems. Such non-IID data can lead to model divergence, slower convergence, and an overall reduction in performance [120] [59]. The variations among FL clients in IoFT systems are more noticeable in cross-device settings compared to cross-silo settings, as shown in Section 2.2.3. This is because the heterogeneity problem extends beyond the data collected by the FL clients. Ye et al. [113] categorised the heterogeneous aspects of FL into four types: statistical, model, communication, and device heterogeneity.

Statistical Heterogeneity. It refers to the non-IID nature of data across clients, where local datasets can differ significantly in terms of features, labels, or distributions. This disparity leads to biased model updates, preventing the FL system from converging effectively and limiting its ability to generalise beyond the training data. To address these issues, optimisation frameworks such as *FedOpt* propose adaptive approaches where FL clients are not restricted to SGD but can utilise alternative optimisers (e.g., *Adam*, *Yogi*). To enhance local training and encourage more unbiased aggregation amongst nodes[87]. Additionally, server-side optimisation techniques can be integrated. For example, Hsu et al. [41] introduce *FedAvgM*, which incorporates *server momentum* to reduce the effects of non-IID data that are common in conventional FedAvg implementations.

Model heterogeneity. It arises when FL participants use different model architectures due to variations in computational capabilities or application-specific requirements. This variation makes it challenging to aggregate model parameters effectively. To address this, methods such as knowledge distillation [75] and personalised FL approaches [102] align shared knowledge across participants, while also allowing local models to retain features tailored to their specific contexts.

Communication heterogeneity. It reflects IoT device network connectivity, bandwidth, and latency differences. Some clients may experience delays, limited connectivity, or even drop out during training, causing disruptions to the FL process. Asynchronous FL [21] and dropout-resilient aggregation [66] strategies are commonly used to overcome these issues and maintain robust training despite communication variability.

Device heterogeneity. It refers to variations in hardware capabilities, such as processing power, memory, and energy efficiency. These differences can restrict some IoT devices from participating effectively in FL due to resource constraints. To address this, lightweight models, model compression techniques [35], and resource-aware adaptation methods [108] are employed to ensure that even resource-constrained devices can contribute effectively to the training process.

2.2.5 ML Techniques. IoFT can utilse several ML techniques. The selection of techniques is based on the goal that needs to be achieved and the type of data available. Many domains utilise different ML techniques along with SASs. ML techniques can make use of Bayesian Theory [43][68][43], Clustering [48], Fuzzy Learning [98], Genetic Algorithms [4] [20], Neural Networks (NN) [25][39], and Decision Trees [18][45][92]. Saqutri and Lee [97] provide a comprehensive review of ML for SASs. A recent analysis of both qualitative and quantitative synthesis of 231 studies that reflect the state-of-the-art in federated machine learning can be found in [67].

2.2.6 FL Frameworks. Many frameworks are used in the FL research community. These frameworks were primarily created to be implemented on real-world systems. Some well-known frameworks are PySyft [94], FedML [37], LEAF [17], Flower [11], Clara [79], PaddleFL [105], Open FL [89], TensorFlow-Federated [103], FATE [110]. Burlachenko et al. [16] present FL_PyTorch, an FL simulator, to deduce the preliminary required to implement FL without expert knowledge.

2.2.7 The Relationship Between FL and IoT. Several limitations regarding the current IoT ecosystem paradigm for implementing ML tasks must be noted. These limitations include lack of data availability, violation of end-user privacy, high communication costs, heterogeneity of IoT devices, and challenges in scalability, availability, and reliability of ML functionality [55]. FL plays an essential role in addressing the limitations of ML in the IoT domain. FL can be used to overcome the lack of data availability by allowing several parties to join the collaboration and share their model parameters trained within other parties' local data. The conventional approach for training ML in the IoT domain requires all benefited parties to share their data with a central server to perform the model training. Many organisations, particularly in healthcare, may be hesitant to embrace the idea of collaboration due to concerns about privacy violations related to sharing end-user data. FL discourages all parties from sharing their local data to maintain privacy. Instead, participants can share model parameters, such as gradients and weights. Moreover, the communication costs associated with offloading local data to the cloud are critical for the traditional ML in IoT systems. Additionally, the heterogeneity of the IoT introduces new challenges due to the capabilities of IoT devices, which can affect the overall training process and the performance of the global model. The architecture of FL provides a novel solution for both communication costs and the diversity of IoT devices by taking advantage of the distributed nature of FL and the type of data required

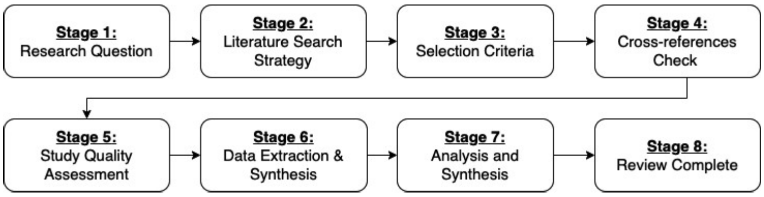


Fig. 4. Research methodology stages [23].

from joined parties. Each FL client is only required to share a small amount of data, processing less payload and improving network bandwidth. Moreover, the FL server's aggregation process can efficiently be utilised to handle the diversity of IoT devices by leveraging different incentive mechanisms to reward FL clients based on their contributions, such as data quality and quantity. One major challenge of conventional ML systems in the IoT domain is their scalability, availability, and reliability due to centralised training on a single server, which increases vulnerability to threats. Additionally, as IoT devices increase in number, handling scalability is overwhelmingly complex. In the FL architecture, the FL client is responsible for training a global model with its local data, and the FL server will be responsible for global model initiation, including aggregating all FL client contributions. FL clients are not required to join the training round and can be disconnected anytime. However, they will lose the benefit of FL by obtaining the most updated model. In FL, aggregation techniques are designed to manage many FL clients and measure their contribution to the quality of the overall (global) model. One key objective of adapting IoFT is to overcome the limitations of conventional ML systems when applied to continuously changing environments.

3 REVIEW METHODOLOGY

The systematic literature review (SLR) is valuable for assessing papers that meet pre-defined eligibility criteria [73]. In addition, the SLR is concerned with identifying, analysing, and assessing research findings relevant to specified research questions. To conduct this work, we performed both manual and automatic searches. We explored the implementation of self-adaptive IoFT in edge, fog, and cloud computing scenarios. We used backwards and forward reference searching methods to identify the most relevant results. We followed the research methodology outlined in Figure 4.

- *Stage 1 : Research Questions*: We intend to present a comprehensive and structured overview of all significant self-adaptive IoFT articles related to the following research questions.
  - RQ1: What is the definition of self-adaptive IoFT systems?
  - RQ2: What are the characteristics of self-adaptive IoFT systems?
  - RQ3: What is the feedback loop architecture for self-adaptive IoFT systems?
  - RQ4: What are the primary motivations for using self-adaptive IoFT systems?
  - RQ5: What technical considerations for implementing self-adaptive IoFT systems?
  - RQ6: How are self-adaptive IoFT systems evaluated?
  - RQ7: What is the reality of self-adaptive IoFT systems?

  RQ1 addresses the definition of a self-adaptive IoFT in the literature and its different interpretations in Section 4.1. RQ2 is motivated by the need to define and characterise self-adaptive IoFT capabilities. Additionally, we summarise several existing taxonomies used in the literature to identify the primary characteristics of self-adaptive IoFT in Section 4.2. RQ3 addresses the adaptation logic and the conceptual architecture of feedback loop control for Self-adaptive IoFT in Section 4.3. RQ4 explains the motivation for utilising self-adaptive IoFT in real-world application in Section 4.4. To conduct a thorough analysis, RQ5 will examine various engineering approaches for developing self-adaptive IoFT systems in Section 4.5. The purpose of RQ6 is to identify various evaluation techniques, criteria, and metrics proposed in the literature for assessing the performance and reliability of IoFT in Section 4.6. Finally, RQ7 demonstrates the reality of self-adaptive IoFT by examining real-world applications and domains that employ this paradigm in Section 4.7.

- *Stage 2 : Literature Search Strategy*: Our search technique begins with identifying data sources and a search query. The study's search strategy is based on an automated search in multiple globally known databases and indexing systems such as Google Scholar, Scopus, ScienceDirect, AMC Digital Library, IEEE Xplore and SpringerLink to collect relevant information from published sources. We searched for keywords against lists of databases and scientific citation indexing services, as shown in Table 3. Our automatic search technique was focused on the keywords "Federated learning", "Internet of Federated Things", "Autonomic computing", and "Context-aware". Due to the ambiguity surrounding the definition of self-adaptation, we used a wildcard search to get all relevant results that contain terms such as "Self-configuration", "Self-learning", and "Self-optimisation".The automatic search results before snowballing search strategy are reported in Table 4.
- *Stage 3 : Selection Criteria*: We conducted two rounds of study selection against the findings of automatic searching to determine the primary study. We carefully filter the papers in the first round according to their title, abstracts, and keywords. Additionally, we eliminate duplicate data from a variety of data sources. In the second round, We filtered the primary studies using well-defined inclusion (IC) and exclusion criteria (EC).
  - IC1: FL was first proposed in 2016 by Google as an alternative setting for centralized ML approaches[72]. Therefore, we limited our search to including papers published after 1 January 2016 demonstrating the self-adaptive IoFT.
  - IC2: The study suggests an ML-based approach for IoFT and property. However, any study incorporating a self-adaptive technique that explicitly or explicitly mentions the IoFT, such as federation in cloud computing, software, or blockchain, will be included.
  - IC3: The study discusses self-* logic in general. Since this study focuses on IoFT, we include only research that applies the self-adaptive IoFT environment for self-adaptation.
  - EC1: The study should not be an abstract or limited to one or two pages. These studies are eliminated because they often need more information.
  - EC2: The study should not use self-adaptation in contexts other than FL and IoT. This study does not contribute to answering the main research question since we mainly focus on the IoFT.
  - EC3: The study that focuses on theory without proof of concept will not be included because this study needs to meet our quality assessment criteria.
- *Stage 4 : Cross-references Check*: To ensure we do not miss any relevant research, we use a cross-referencing methodology and identify potentially relevant papers through the "snowballing" search strategy. This involves recording the references found in the "References" section of each primary study [13] [73].
- *Stage 5 : Quality Assessment Criteria*: Kitchenham et al. [13] stated that the quality of research is related to its ability to minimise bias and maximise internal and external validity. Accordingly, our primary studies are evaluated according to the pre-defined quality assessment criteria to assess the quality of the studies. In addition, we employed the checklists provided by reference [13]. We included each paper that defined the problem

Table 3. Search Queries

Database or Indexing Services	Search Queries
Google Scholar	intitle:"federated learning" OR intitle:FL AND IoT OR "Internet of Things" AND intitle:self-* AND -intitle:review AND -intitle:survey AND -intitle:systematic
Scopus	TITLE-ABS-KEY(("federated")AND(self*) AND ("IoT" OR "Internet of Things")) AND PUBYEAR > 2015
ScienceDirect	("federated") AND ("IoT" OR "Internet of Things")
ACM	[Abstract: "federated"] AND [Abstract: self*] AND [Publication Date: (01/01/2016 TO 31/12/2022)]
IEEE	("Abstract":"federated") AND ("Abstract":"self*") AND ("Abstract":IoT OR "Abstract":"Internet of Things")
Springer Link	"federated learning" OR "Internet of Federated Things" AND self-* AND (review OR survey OR systematic)

Table 4. Search Results Before Snow-balling

Google Scholar	IEEE	ACM	ScienceDirect	Scopus	Springer Link
64	21	50	23	104	28

Table 5. Data Item Collection Form

Data item	Description	Relevant QR
Title	Paper's reference	Documentation
Year	The publication year of the primary study	Documentation
Publication source	The publication metadata and type of primary study	Documentation
Definition	The definition of self-adaptive IoFT	RQ1
Self-* property	The properties of self-adaptive IoFT	RQ2, RQ3
Self-* taxonomy	The self-* taxonomy on different in the field of a self-adaptive system	RQ2, RQ3
Feedback loop control	The feedback loop architecture concept	RQ3
Motivation	The primary motivations for using self-adaptive IoFT	RQ4
Technical aspect	The technical considerations for implementing self-adaptive IoFT s	RQ5
Assessment method	The various evaluation techniques, criteria, and metrics for assessing self-adaptive IoFT	RQ6, RQ7
Applications	The real-world applications and domains that employ the self-adaptive IoFT	RQ7

statement and contribution, presented background and context, clearly described the research method and evaluation, and reported on the findings.

- *Stage 6 : Data Extraction Item*: We reviewed all selected primary studies to gather data to help answer the research questions. Table 5 describes the data items to retrieve and their corresponding research questions.
- *Stage 7 : Analysis and Synthesis*: Stages three, four, and six of the research method used in this work examined the analysis and synthesis of research publications. Data items defined earlier in stage six were extracted and recorded in a spreadsheet for each study. Additionally, we used various software tools such as Nvivo and Excel to analyse and visualise the findings of the selected primary studies.
- *Stage 8 : Reporting The Review*: In this section, we present the results of our analysis of the 17 selected primary study data after completing the filtering process conducted in all previous stages, along with the answers to our research questions.

4 SELF-ADAPTIVE INTERNET OF FEDERATED THINGS (IOFT)

4.1 The Definition of Self-adaptive IoFT

To define a self-adaptive IoFT, we first need to understand two terms: context and context-aware systems. These terms form the foundation of every SASs.

4.1.1 Context and Context-aware Computing. The primary source of data generation is the context. However, in the literature, many researchers define the term "Context" differently based on their domain expertise. For instance, Brown [14] defines context as the components of the user's surroundings that are known to the computer. Salehie et al.[96] extend Brown's definition, so the context is everything in the operating environment that influences the system's attributes and behaviour. However, the most accurate definition was introduced by Abowd et al. [2]. These authors refer to context as:

> *"Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves."*

Abowd et al. [2] state that the context compresses three entities, which are people (individuals, groups), places (buildings, offices), and things (sensors, actuators). These entities are characterised by several attributes across four categories: identity, location, status, and time. Abowd et al. [2] also define these categories as the primary context. In contrast, secondary context can only be obtained from the primary context with additional information, such as weather conditions based on the user's location. However, in IoT applications, this definition is only partially applicable. Perera et al. [83] argue that Abowd et al.'s [2] categorisation scheme did not consider ordinary IoT contexts. For example, calculating the distance between two GPS sensors involves processing the locations obtained by both sensors. This type of data processing commonly occurs in IoT applications as most data interpretation is derived from a group of sensor readings. Abowd et al.'s [2] definition does not cover this common scenario in IoT applications. Therefore, Perera et al. [83] extend the original definition to include an operational perspective. Their definition highlights the challenges in data acquisition in the IoT application and the conceptual perspective to understand the relationships between different contexts. Our research adopts the categorisation scheme by Perera et al. [83], as it is better suited for the IoT paradigm and capturing both the operational and conceptual attributes of context-aware IoT applications. Context-aware computing is derived from the desire to make use of contextual information. In the literature, numerous viewpoints on how systems should consider context have been provided [83] [2] [14] [96]. However, the primary intent of context-aware computing is to evaluate the context and respond to dynamic environment changes to meet a specific goal based on relevant information. In the IoT setting, context-aware systems connect context information to sensor data to provide insight for interpretation.

4.1.2 Self- IoFT Systems.* An early definition of the term 'self' emerged in the late 1890s within psychology, where Baker [7] described 'self' as a process of identification, marking the beginning of its scholarly exploration. Goffman [32] further extended this conceptualization of 'self' in sociology, illustrating 'self' as a dynamic entity influenced by varying circumstances and contexts. The transition from these foundational ideas to the technological domain was marked by IBM's introduction of 'autonomic computing,' aimed at developing systems capable of self-management [49]. This manifesto led to the emergence of 'self-*' systems, encapsulating behaviours such as self-configuration, self-optimization, self-healing, and self-protection. The concept of 'self-*' systems has since evolved rapidly, prompting efforts to define it broadly, despite the lack of a universally accepted definition as highlighted in [28][52]. The literature presents two opinions on the definition of 'self-*'. Initially, the definition is influenced by the author's perspective and the context in which it is applied. Alternatively, it is shaped by the particular domain that adopts the 'self-*' method. This variation underscores the adaptability of 'self-*' concepts across different scientific fields, including the IoFT, where such systems play a crucial role. Table 6 shows explicit and implicit definitions of 'self-*' in IoFT systems as found in primary studies.

Table 6. Definitions of Self-* Property in IoFT System

Study	Express	Property	Definition
[116]	Implicit	Self-organisation	*Ability of federated learning clients to autonomously enhance the uploaded weight of parameters to the federated learning server.*
[93]	Implicit	Self-organisation	*The ability of federated sensor network to autonomously detect resources, optimise the network routing protocol and algorithm*
[77]	Implicit	Self-learning	*The ability of federated learning to learn the device's type and anomalies autonomously.*
[82]	Implicit	Self-organisation	*The ability of federated learning to autonomously create different collaboration schemes to identify the heterogeneity hidden in the federation.*
[50]	Implicit	Self-organisation	*The ability of federated learning to autonomously formalities device clusters, join devices, and allocate resources.*
[30]	Explicit	Self-adaption	*”In the context of self-adaptation, the roles are monitor, analyse, plan and execute over a shared knowledge ”*
[8]	Explicit	Self-adaption	*”FedML solutions configure the system, that is, set its parameters, and allocate resource dynamically”*
[33]	Explicit	Self-awareness	*”we refer to a SASO system S as a collection A of autonomous subsystems ai that are able to adapt their behaviour based on self-awareness of the internal and external conditions”*
[112]	Explicit	self-attention	*”By using the self-attention mechanism, we can optimize both the server-to-client and the client-to-client parameter divergence and increase the model's performance to Non-IID data. ”*
[63]	Explicit	self-revealing	*”The self-revealing mechanism of the contract theory approach enables workers to be rewarded based on their specific types even in the presence of information asymmetry, i.e., when the worker types are not known by a model owner.”*

4.2 The Characteristics of Self-adaptive IoFT

There has been a significant effort in the literature to develop various taxonomies to identify the key characteristics of SASs over the years. A notable contribution to this field is the comprehensive empirical analysis conducted by Krupitzer et al. [53], which spans early studies from the 2000s. This work aimed to establish a uniform taxonomy for self-adaptation by addressing the 5W+1H questions, a concept first introduced by Salehie et al. [96]. The taxonomy presented by Krupitzer et al. is summarized into five dimensions: time, level, technique, reason, and adaptation control, providing a general overview of the landscape of SASs. Other researchers, such as Andersson et al. [5], have proposed alternative dimensions, viewing self-adaptation through system goals, triggers, mechanisms, and adaptation outcomes. This perspective was further refined by Brun et al. [15], who proposed five dimensions specifically for the software engineering domain, including adaptation targets, effects, actions, states, and environment. While these contributions are pivotal, they predominantly reflect the viewpoints and requirements of the software engineering community without directly addressing the unique aspects of the IoFT. Therefore, we have formed IoFT taxonomy in Figure 5, drawing inspiration from the studies above. The multi-dimensional taxonomy is organised to encapsulate the distinctive nature of IoFT, offering a novel framework that complements existing research while addressing the specificities of IoFT applications. Additionally, in Table 8, we present the evaluation of self-adaptive IoFT characteristics taxonomy against the primary studies we collected.

4.2.1 Time . In an ideal situation, *Proactive* adaptation is preferred over *Reactive* adaptation to maintain consistent performance.*Proactive* adaptation depends on precise prediction capabilities, which require continuous monitoring and advanced learning techniques [53]. In contrast, *Reactive* adaptation starts after a change need is identified, responding to unexpected environmental patterns as they arise. Therefore, the adaptation process in the *Reactive* model is initiated by detecting these irregularities. On the other hand, *Proactive* adaptation seeks to anticipate potential environmental changes before any performance degradation occurs, focusing on predicting and preparing for future environmental shifts [96] [53].

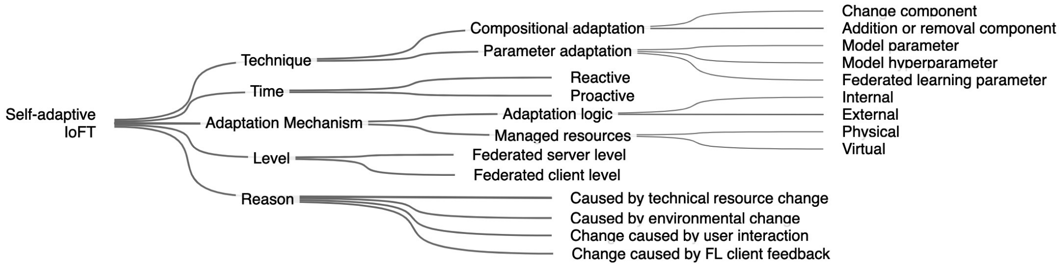


Fig. 5. A taxonomy of characteristics for self-adaptive IoFT

4.2.2 Reason . Adaptation action is often a response to a change. Consequently, the type and effect of the change must be defined to determine whether a response is required. Additionally, it is essential to identify the cause of the change and define the criteria for an appropriate method to respond to that change. The main reasons for the adaptation in IoFT are: 1) *change in the technical resources,* 2) *change in the environment,* 3) *change due to user interactions,* and 4) *downgraded feedback from FL clients.* Firstly, a *change in the technical resources* includes all the tangible (e.g., IoT devices failure and power outage from hardware components) and intangible (e.g., software failure, FL server model divergence, FL client aggregation errors, and unstable network connection) assets. Secondly, the *environment changes* when the context of the FL clients changes from one context to another (e.g., smart home, smart building, smart manufacturing). Third, a *change due to user interactions* can be either by altering user behaviour using IoT devices (e.g., fire alarm leading to evacuation) or changing user preferences by the user to interact with IoT devices and sensors. Finally, the FL server may trigger the adaptation strategy if the *FL Client's feedback* does not meet the current service requirement (e.g., lack of availability to join the training, poor model training due to data availability or model poisoning attack).

4.2.3 Level . The implementation of adaptation can exist at a different level. Various existing taxonomies try to answer the question, "Which layer of the system can be changed?" [96] and "Where do we have to implement changes" [53]. These questions were answered in the literature. However, there is no explicit declaration of the adaptation levels in IoFT. There are two levels in IoFT, which are the *FL server* and the *FL client.* They provide a high-level abstraction of where the managed resources and adaptation logic exist. An *FL server* has many adaptation mechanisms, such as client selection, global model selection, and model aggregation optimisation. Adaptation also occurred in the *FL client*, where different actions such as local model optimisation, automatic sensor configuration, and automatic delivery of actuation orders can be presented.

4.2.4 Technique . The literature employs several adaptation techniques from different fields. For instance, McKinley et al. [71] mentioned two approaches for adaptation in the field of software engineering: 1) parameter adaptation and 2) compositional adaptation. Parameter adaptation is the most straightforward technical approach since it is achieved by simply identifying the contributed system's parameters and changing them according to the adaptation policies. Compositional adaptation is a dynamic approach to changing algorithms or system components to reduce performance degradation. Therefore, we leverage these two adaptation techniques for self-adaptive IoFT. In self-adaptive IoFT, the *Parameter* adaptation needs to consider three general FL system design tiers: 1) *FL server parameters,* 2) *ML model parameters,* and 3) *ML model hyperparameters. FL server parameters* adaptation achieves the adaptation behaviour by adjusting the global configuration setting for the FL server and aggregator parameters, which vary depending on the aggregation

Table 7. Configurable Parameters in FL Systems

Category	Parameter/Algorithm	Definition/Parameter Details
Global FL Parameters	fraction_fit	The proportion of randomly selected clients participating in each training round reflects the system's strategy.
	fraction_evaluate	The proportion of clients selected for model evaluation after training.
	min_fit_clients	The minimum number of clients required to proceed with a training round.
	min_evaluate_clients	The minimum number of clients needed to validate the model's performance.
	min_available_clients	The minimum number of clients that must be online and available for the FL system to function.
	accept_failures	A policy determining whether training rounds that experience client failures are accepted or discarded.
	initial_parameters	The initial global model parameters serve as a baseline for subsequent iterations.
	client_learning_rate	The learning rate used by individual clients to update their local models during training. This parameter controls the size of the step taken during each gradient descent update.
	batch_size	The number of samples used in one forward and backward pass of the training process. It impacts the stability and speed of local training.
	local_epochs	The number of passes over the local dataset performed by each client before sending model updates to the server.
Aggregation Parameters*	FedAvg [72]	(c) Number of clients participating in each training round, (e) Number of training epochs each client makes over its local dataset on each round, (b) The local mini-batch size used for training in client side
	FedSGD [72]	(w) Model weights on communication selected round , (c) Number of clients participating in each training round, (e) Number of training epochs each client makes over its local dataset on each round, (b) The local mini-batch size used for training in client side, (eta) The learning rate
	FedAdam, FedYogi [88]	(eta) Server-side learning rate, (eta_l) Client-side learning rate, (beta_1) Momentum parameter, (beta_2) Second moment parameter, (tau) Controls the degree of adaptability
	FedAdagrad [88]	(eta) Server-side learning rate, (eta_l) Client-side learning rate, (tau) Controls the degree of adaptability
	FedProx [60]	(proximal_mu) The weight of the proximal term used in optimization
	FedTrimmedAvg [115]	(beta) Fraction to cut off of both tails of the distribution
	q-FedAvg [61]	(eta) learning rate, (q) Tune the amount of fairness

**Note: The aggregation methods listed in this table are not exhaustive. For a comprehensive review of aggregation methods and their impact on federated learning performance, see [34].*

algorithms. Table 7 shows the literature's most commonly used aggregation algorithms and their configurable parameters. *ML model parameters* allow the changes of ML algorithm parameters in which it obtains a better starting model. Depending on the pre-defined strategy, the adaptation can occur on the FL server or client side. Changing model parameters may require entirely reinitialising the FL system to deploy the changes. Unlike *ML model parameters*, the *model hyperparameters* enable changing at runtime in any part of the FL system, including the FL server and client. Two methods can archive *compositional adaptation*: 1) *Change component* by replacing the ML algorithms or adding more models in the case of ensemble ML, and 2) *Add or remove component*, such as including or excluding FL clients, based on their contributions and effectiveness to global model convergence. Finally, the adaptation triggers are based on specific pre-defined policies or criteria where the adaption is required to prevent performance loss or to get better overall performance.

4.2.5 Adaptation Mechanism. Adaptation Mechanism is the core element of any self-adaptive system [53][96]. Krupitzer et al. [53] highlight that adaptation control is compressed into two main components: *Managed Resources* and *Adaptation Logic*. The same components are also applied to the self-adaptive IoFT domain but with different interpretations.

Managed resources. The IoFT ecosystem encompasses *Physical* and *Virtual* resources. *Physica*l resources within FL clients vary to their specific domain, such as motion sensors in smart homes or heart-rate monitors in healthcare. Despite the diversity across domains, these resources universally function as sensors, actuators, small devices, or gateways. For instance, a sensor translates physical

Table 8. The Evaluation of Self-adaptive IoFT Characteristics Taxonomy Against The Primary Studies

	Level		**Reason**				**Technique**		**Time**		**Adaptation Mechanism**	
Paper	Server	Client	Context	Technical Resources	FL Client feedback	User Action	Compositional	Parameter	Proactive	Reactive	Approach	Managed Resources
[77]	×	✓	×	✓	×	×	Addition/ Removal component	×	×	✓	Internal/ External	(Physical) Gateway
[82]	✓	×	×	×	✓	×	×	Federated	✓	×	External	(Physical) FL server
[50]	✓	×	×	×	×	✓	Change component	×	✓	×	External	(Physical) Small device
[95]	×	✓	✓	×	×	×	Change component	Model	×	✓	External	(Physical) Gateway
[27]	✓	×	✓	×	×	×	Change component interaction	Federated	×	✓	Internal/ External	(Physical) FL server
[62]	×	✓	×	×	✓	×	Change component	Federated	×	✓	External	(Physical) Small device
[38]	×	✓	✓	×	×	×	×	Model	×	✓	Internal	(Physical) FL server
[58]	×	✓	✓	×	×	×	×	Federated	×	✓	External	(Physical) FL server
[112]	✓	✓	✓	×	×	×	×	Model	×	✓	Internal/ External	(Physical) FL server
[107]	×	✓	✓	×	×	×	Change component interaction	×	×	✓	Internal	(Physical) FL server
[101]	×	✓	×	×	✓	×	×	Federated	×	✓	External	(Physical) FL server
[57]	×	✓	✓	×	×	×	×	Model	×	✓	External	(Physical) FL server
[116]	✓	×	✓	×	×	×	Change component	×	×	✓	External	(Physical) FL server
[30]	×	✓	×	×	✓	×	Change component interaction	×	×	✓	Internal/ External	(Physical) Gateway
[8]	✓	×	×	×	×	✓	Change component interaction	×	×	✓	External	(Virtual) Resource capability
[121]	✓	×	×	×	✓	×	×	Federated	×	✓	External	(Physical) FL server
[19]	✓	×	✓	×	×	×	Change component interaction	×	×	✓	External	(Physical) FL server

events into electrical signals, whereas an actuator executes physical actions based on electrical inputs. Small devices like mobile phones or Raspberry Pi and any control system device with limited computational power. A gateway connects the IoT devices to the Internet, facilitating data transmission between FL client devices and the FL server. On the other hand, *Virtual* resources represent the capabilities of these *Physical* resources, such as the accuracy of sensor data or the storage capacity of devices. These intangible factors are crucial for the efficiency of IoFT ecosystems. In the FL server, resources are represented differently, aligning more with cloud and fog computing paradigms. *Physical* resources here refer to the devices hosting FL orchestration mechanisms, ranging from PCs to virtual machines in the cloud. *Virtual* resources in the FL server enclose performance metrics like computing power and autoscaling capabilities.

Adaptation logic The self-adaptation performed by the *Adaptation Logic* component, which can be either *Internal* or *External* [53][96]. The main objective of *Adaptation Logic* is to explain how we can adapt within a given context. Based on the nature of IoFT, *Internal* and *External* adaptation can occur on the FL server and client since they are not tightly coupling systems. Therefore, the *Internal* approaches integrate application and the logic of adaptation. This approach aims to change the *Internal* configuration of IoFT. For example, an FL server hosted on the cloud internally configures its virtual resources for provisioning autoscaling infrastructure based on a specific threshold [118]. Additionally, an FL server may change the ML model during the run time due to a drop in the overall performance. Similarly, an FL client may change the local ML model hyperparameters based on concept drift [9]. The *Internal* approach has two main drawbacks: the cost of testing and maintaining the system, and (ii) the knowledge required to perform the *Internal* adaptation is only sometimes available [53][96]. Therefore, the *External* approaches overcome these limitations by separating the *Adaptation Logic* and the *Managed Resource* and connecting them via different interfaces. Thus, the *Adaptation Logic* uses an interface to acquire knowledge from the context and to interact with *Managed Resource* [49]. The modularity of the *External* approach makes it more suitable for the IoFT. An example of an *External* approach is an FL server that uses TCP/IP communication protocol to get the FL client's local model parameters and optimise the global FL model according to the client's feedback. Therefore, The communication protocol acts as the interface that enables the *Adaptation Logic* in the FL server to gather contextual knowledge from the FL client. In this context, the FL client utilises the gateway as a physically *Managed Resource* to facilitate responses to the FL server. The *External* approach can also be implemented in FL clients. The *Adaptation Logic* in FL clients collects knowledge from the context via sensors and enables adaptation via actuators. For example, in federated smart homes with "anomaly detection" scenarios, the adaptation logic gets humidity readings from the sensor, which gives the *Adaptation Logic* the contextual information to decide whether to open or close windows using an actuator. In summary, *Managed Resource* and *Adaptation Logic* are the two main pillars of SAS development, including IoFT. Therefore, it is essential to define them to implement self-adaptive IoFT systems. In section 4.3, we deeply explain the Adaptation feedback loop for the self-adaptive IoFT.

4.3 Feedback Control Loop for Self-adaptive IoFT

Because of the characteristics of SASs, it is crucial to comprehend how self-adaptive IoFT is used in a MAPE-K loop. Furthermore, the feedback control loop mechanism should be able to handle the growing complexity of IoFT systems, as we discussed in Section 2.1.5. Therefore, the MAPE-K loop should contain an adaptation logic to manage different internal and external resources to comply with these standards. Unlike the traditional design of standalone and distributed systems, the IoFT systems design must contain two modules, FL server and client, to make the IoFT system work as discussed in 2.2.1. To differentiate FL from the traditional client/server system, The minimum number of FL clients is set to two, each as an independent module within the FL system. As a result, The FL client will essentially be equivalent to what is found in a standalone IoT system. The adaptation logic could be in the system gateway, and the managed resources could be virtual or physical within the IoT environment, as discussed in 4.2.5. On the other hand, the FL server has different characteristics and managed resources to perform self-adaptation. To explain and analyse the self-adaptation in the context of IoFT, we utilised an architecture-based self-adaptation [80] using the MAPE-K framework [49] to explain several aspects of the adaptation mechanism. First, we will analyse each level of IoFT in section 4.2.3, including the adaptation logic and the managed resources for both the FL server and client. Finally, We conclude with a generalised interpretation of MAPE-K for the entire FL system. Figure 6 shows our conceptual architecture overview for self-adaptive IoFT.

4.3.1 FL Client as Managed Resource. On the FL client side, we categorise the devices into two types, including non-adaptive and adaptive IoT clients. First, the non-adaptive IoT client includes all small and large devices such as smartphones, tablets, PCs, microcontrollers, microprocessors, and other embedded systems. One important criterion in non-adaptive devices is the ability to participate in the FL collaboration. Sensors and devices unable to perform computation power

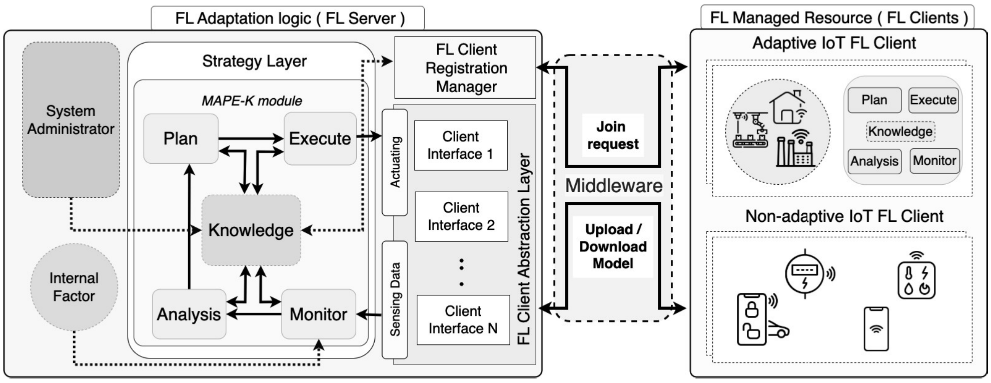


Fig. 6. Conceptual architecture overview for self-adaptive IoFT

will not be included unless they use an adapter to extend their capabilities, such as humidity and temperature sensors connected to a microcontroller to join in FL rounds. Another factor distinguishing non-adaptive IoT clients is that they only train the local model and share their updated version. They did not adapt to the context or modify the model parameters based on some factors, such as concept drift. On the other hand, adaptive IoT clients are any devices that can join the FL collaboration and are also able to adapt to the context. These FL clients use one of the self-adaptation approaches mentioned in 2.1. First, they receive the global model from the FL server, and then they start training the local model using their local data. Asynchronously, they adapt the training behaviour based on the sensing layer, which reflects the local model training. At the end of each round, all non-adaptive and adaptive IoT clients will upload their trained local model to the FL server to join another round of training.

4.3.2 Middleware. The middleware is responsible for communication between the FL client and the server for 1) uploading or downloading the FL model, 2) registering or removing an FL client, and 3) modifying an FL client's parameters. The middleware uses communication protocols to maintain the orchestration of FL, such as Hypertext Transfer Protocol Secure (HTTPS) or Remote Procedure Call (RPC). HTTPS is based on a TCP/IP, which leads to more reliable and stable communication among the FL server and clients. However, RPC uses TCP for reliability and order, while it uses UDP for lower latency and reduced overhead, depending on the application's requirements or system configuration. For example, it may use the TCP protocol when the amount of data to be sent cannot fit into a single UDP datagram. Otherwise, RPC will use UDP initially for small data.

4.3.3 FL Server as Adaptation Logic . The workflow of the self-adaptive feedback loop in the FL server has different characteristics and features than the one in the FL client. The overall objective of the self-adaptive FL server is to maintain system quality by managing the orchestration of IoFT systems. Therefore, the FL server is responsible for the adaptation logic. The FL server also can be installed on physical devices such as computers, microcontrollers, or virtual devices like containers and VMs running in the cloud. Additionally, it consists of several layers of independent and dependent components that form the IoFT adaptation logic.

The *FL Client Abstraction Layer* is the interface between the FL server and clients, where each FL client is connected to a client interface to facilitate communication. The FL client proxy is changeable based on the communication protocol used by the FL system. The received data from the FL clients via the client interface is used as contextual information for the MAPE-K module within the *Strategy Layer*. *FL Client Registration Manager* component provides a client registration or elimination mechanism to allow FL clients to join the FL collaboration. The data will be available in the *knowledge base* of the MAPE-K module for later use during the MAPE-K cycle. The *System Administration* component allows the administrator to interact with the FL system and modify the FL parameters within the *knowledge base*. The *Strategy Layer* is the core layer that contains the MAPE-K module that helps the FL server to implement model aggregations and maintain system performance. The MAPE-K cycle starts at the *monitoring phase* to collect data from the context, as discussed in Section 4.1.1. From the FL server's perspective, the context can be either physical resources (e.g., adaptive or non-adaptive IoT clients) or internal factors as virtual resources (e.g., CPU and RAM usage), as discussed in 4.2.5. Collecting contextual information involves two modes, which vary depending on the data source. Data generated from virtual resources and updated by the system administrator in the *knowledge base* is monitored in real-time. In contrast, data received from the *FL client abstraction layer* is monitored after each FL round.

All received data is preprocessed in the *monitor phase* using the assumptions and system rules predefined within the *knowledge base* component. This preprocessing includes extracting data from sources (e.g., network instrumentation, environmental sensors, application requirements), selecting a model (e.g., NN model, ensemble ML model, or Reinforcement Learning (RL) model), and identifying FL clients using participant information and other metadata. After preprocessing, the sensing data is passed on to the *analysis phase* for data diagnosis and model aggregation. The sensing data received from the *monitoring phase* is used to analyse the quality of FL client contributions. Various reasoning algorithms can be used in the *analysis phase*, depending on the system requirements, FL client configuration type, and targeted application. The *knowledge base* can provide the required data for reasoning, training, and model aggregating tasks. The analysis task can also be extended to include multiple tasks simultaneously, such as fault detection and classification, as shown in [77]. The result of the data analysis is stored in the shared *knowledge base*, making it available across all the MAPE-K framework components. If adaptation is required, an adaptation request is sent to the *plan phase*, which creates a workflow of adaptation actions required to maintain the system's performance. This workflow may include removing weak FL clients' contributions, reducing the weight of their contributions, changing the aggregation model approach, and changing FL global configuration (e.g., the number of FL clients or the training rounds). The *execution phase* then carries out these workflows produced by the *planning phase* through the actuators of the managed resources.

Finally, the *FL Client Registration Manager* component serves as the primary entry point for any FL client looking to participate in the collaboration. This component is responsible for initializing and profiling newly added FL clients and reporting them to the *knowledge base* within the *Strategy Layer*. Additionally, It provides the initial FL model and any other FL configuration settings to the FL client. During the joining process, a new FL client has a chance to provide any extra information regarding their context, which may help FL to be aware of the context of their FL client. By introducing this component, the system designer will have extra flexibility to manage the initial data exchange between the FL client and server before allowing the new FL client to join the training round.

4.4 Motivation of Self-adaptive IoFT

This section examines the motivation behind studies focused on self-adaptive IoFT. A significant portion of these studies explicitly articulate motivation as a benchmark for evaluating the effectiveness of their proposals. Nonetheless, a few studies have not overtly acknowledged their use of self-adaptive IoFT. Therefore, we reviewed each study, extracting the inherent "self-*" feature to bridge the connection between the intended purpose and the motivation, as shown in Table 9. We

Table 9. Motivation of Self-adaptive IoFT Systems

Motivation	Primary Studies
Improve FL performance	[77] [82] [95] [27] [38] [58] [112] [107] [57] [116] [30] [8] [121] [19]
Data Quality	[112] [107] [19] [27]
System Complexity and Resource Consumption	[50] [27] [62] [101] [30] [8] [19]
IoT Heterogeneity	[77] [82] [95] [62] [38] [58] [101] [116] [30] [8] [121] [19]

Table 10. Implementation of Self-adaptive IoFT Systems

Implementation	Primary Studies
Context-based approach	[77] [121]
ML-based approach	[95] [27] [38] [58] [112] [107] [57] [116] [8] [19]
Nature-based approach	[50] [62] [30]
Agent-based approach	[82] [101]

enumerate the primary motivations that have driven researchers toward exploring the domain of self-adaptive IoFT.

4.4.1 IoFT Performance . One of the core components of FL is predictive modelling. In contrast to the traditional method of training and evaluating a machine learning model in a centralised ML application, an FL model is trained and evaluated in a distributed environment, which can introduce various challenges, as discussed in Section 2.2.1. Some work in the literature has been done to improve IoFT performance using a self-adaptation approach.

4.4.2 Data Quality. One of the challenges of the IoFT is the data quality used to train the global model. For example, IoFT based on stochastic gradient descent (SGD) is highly sensitive to the data used to train the model. Therefore, The independent and identically distributed (IID) data sample is required to ensure that the stochastic gradient is an unbiased estimate of the entire gradient. The Non-IID data samples must be addressed to obtain better performance [112]. To overcome this issue, many researchers have developed self-adaptive IoFT to minimise the effect of non-IID impacts.

4.4.3 System Complexity and Resource Consumption. Due to the complexity of implementing IoFT systems in a real-world scenario, many researchers try to facilitate the deployment and reduce the resource consumption of IoFT systems. That includes reformatting the structure of the IoFT system [50] or automatically altering the training behaviour [19].

4.4.4 IoT Heterogeneity. Another motivation driven by the literature is dealing with FL client heterogeneity. In real-world applications, IoT devices are diverse and have heterogeneous computing resources. Also, the wireless network is unstable, making it difficult to guarantee consistent connection as communication conditions change. Therefore, to implement FL training processes on heterogeneous FL client devices, researchers use the self-learning approach to overcome this challenge in the IoFT system [77][101] [19].

4.5 Implementation of Self-adaptive IoFT

Engineering self-adaptive IoFT aims to integrate "self-*" properties into the system design. This integration enables the system to reactively or proactively manage its state, knowledge, and execution environment. This section outlines how self-adaptation techniques have been implemented in IoFT systems. In Table 10, we present the implementation of self-adaptive IoFT based on the primary studies.

4.5.1 Context-based Approach. As discussed in section 4.1.1, The primary responsibility of context-aware systems is to assess the context and respond to dynamic environment changes to accomplish a specific objective based on relevant data. Nguyen et al. [77] use a self-learning approach to improve the overall AD classifier within an IoFT environment. They use a device-type identification approach to identify the device type and extract feature representations of normality. As the authors claim, they achieved no false alarms in a real-world smart home environment. Similarly, Zhaohang et al. [121] improve the quality of FL performance by introducing an adaptive asynchronous FL for participants. They intend to solve the issue of slower devices that decrease convergence speed and worsen the global model performance.

4.5.2 ML-based Approach. Learning and self-adaptation are closely related in the IoFT systems. A self-adaptive IoFT system continuously adjusts its structure, settings, or algorithms to become more effective, as discussed in Section 4.2.4. Therefore, various learning techniques were used in the literature to achieve different system goals. Firstly, Improving ML performance in IoFT is one of the motivations for utilising the self-adaptation technique. For example, Li et al. [58] take advantage of information about the device's historical training tasks to constrain future workflow and combine it with active learning to select participants adaptively. They also integrate spatiotemporal information with a self-attention approach to mixing the local and global models based on the differences between the local, global, and personalised models [57]. Zhu et al. [116] suggest a mechanism to improve the uploaded weight of parameters to the FL server. They also achieve faster convergence with higher accuracy by improving the sparsity of the global model by continuously updating the online optimisation function in their proposal algorithm. Whereas, Baresi et al.[8] explore the self-adaptive IoFT system by optimising clients' resources at runtime considering network overhead and model accuracy. Secondly, there are some works in the literature that improve the quality of the data collected in the FL client. For example, Duan et al. [27] use Kuullback-Leibler divergence to solve the unbalanced data and improve the FL performance. Similarly, Wang et al. [107] introduced self-paced learning to keep the high-confidence samples and drop the high-noise samples. As most FL algorithms are severely affected by the distribution of data, A self-attention approach was proposed by Xu et al. [112] to set up a communication strategy for FL effectively. Their approach applies parameter optimisation for server-to-client and client-to-server to overcome the problem of non-IID in an FL setting. Finally, some work has been done to address unlabelled data collected from the FL client. Saeed et al. [95] leverage the scalogram signal correspondence learning on wavelet transform to self-learn valuable representations from unlabelled sensor inputs. He et al. [38] propose a self-supervised and personalised FL framework that uses algorithms for collaborative training of global and personalised models. Moreover, Chen et al. [19] propose a federated graph learning framework to optimise a global self-supervision model to create global pseudo-label discovery and graph construction that will be shared with a federated client to label the data.

4.5.3 Nature-based Approach. A nature-inspired approach optimises the given system to achieve specific goals. The main idea is that each system component has limited information and acts according to the tasks the system administrator assigns. The overall system behaviour can be predicted by the collective behaviour of each individual participating within the population. Reducing training time in ML is a significant area of interest within nature-inspired computing. Khan et al. [50] propose a novel FL scheme that uses a formulated optimisation problem to minimise global FL training time. They achieve that with the help of cluster formalisation, joint device frequency selection and resource allocation. Another work was introduced by Lim et al. [62] to select an FL server and resource allocation based on the evolutionary game. Their approach provides two levels. The lower level uses an evolutionary game to represent the workers' edge association strategies.

With the edge server's bandwidth allocation control method, the upper level uses a Stackelberg differential game in which the model owner selects the optimal reward structure. Franco et al. [30] provide a self-adaptive architecture for IoFT in industrial automation systems. Their method considers the participating parties at various levels of the industrial ecosystem. Each factory internally trains the model in a self-adaptive way and delivers it to the centralised cloud server for global aggregation. As a result, they fulfil the goals of global model optimisation and reduction of communication cycles within the IoFT system. Furthermore, to overcome a multi-assignment optimisation issue, they split the dataset into as many subsets as are equivalent to the number of participants. At each local iteration, each device selects the appropriate subset to optimise the model.

4.5.4 Agent-based Approach. The agent-based approach contains different agents who share similar goals. Those agents coordinate and communicate with each other, which opens the door to working with the decentralised systems more efficiently. This approach can be helpful for the IoT environment since different IoT devices can be considered different agents. Pang et al. [82] use the agent-based approach in IoFT for finding cooperation plans based on RL. They also adjust system weight based on user feedback. Similarly, Tam et al. [101] suggested implementing a self-learning agent engaging with a network functions virtualisation (NFV) orchestrator and software-defined networking(SDN) architecture by incorporating a deep q-learning algorithm.

4.6 Evaluation of self-adaptive IoFT

This section examines the evaluation methodologies used in the literature for self-adaptive IoFT and the testbeds involved.

4.6.1 IoFT Testbed. IoFT systems can be developed using various approaches, including case studies, physical deployments, local lab networks, simulations, and publicly available datasets. Case studies provide in-depth examinations of real-life contexts. For example, Khan et al. [50] utilized a smart tourism scenario to automatically formalise IoFT device clusters and resource allocation. Moreover, physical deployments allow for testing in realistic environments, while local laboratory networks, including virtual machines (VM) and physical devices, can provide valuable data. An example is the work by Nguyen et al. [77], where they detected Mirai malware in small office settings by creating profiles for 33 IoT devices. Simulations can also replicate real-world behaviour. For instance, Tam et al. [101] used Mininet to emulate IoFT for resource optimization. Additionally, researchers often rely on publicly available datasets, such as MNIST [8][121][112][27], CIFAR10 [121], CINIC-10 [27][38][112], BAL1 [27], FEMNIST [58], Tweets [57], and Cora [19].

4.6.2 IoFT Evaluation. In recent years, evaluating SASs has become increasingly complex, especially when SAS adopt the distributed nature of FL and the inherent heterogeneity of IoT environments. Researchers from various domains have undertaken considerable efforts to address this complexity, focusing on different aspects of SAS evaluation through multiple perspectives and mechanisms. Broadly, these efforts align with two principal paradigms: the evaluation of the adaptation mechanism (i.e., architectural evaluation, as referred to in the literature) and the evaluation of the impact of embedding self-* properties into underlying systems. The motivation for these evaluations, as discussed in Section 4.4, is to clarify the overall goal, whether to assess the adaptation mechanism or the performance impact of the SAS.

In adaptation mechanism evaluations, system architects primarily evaluate the design attributes of all stages in the SAS architecture, which we discussed in Section 2.1. McCann et al. [70] propose nine perspectives for assessing SAS, encompassing *quality of service*, *cost*, *granularity/flexibility*, *failure avoidance (robustness)*, *degree of autonomy*, *adaptivity*, *time to adapt/reaction* *time*, *sensitivity*, and *stabilization*. These perspectives help evaluate the managed resources' runtime performance and the adaptation mechanism. While the authors suggest example metrics for each perspective, they note that actual measurement strategies depend strongly on factors such as system domain, adaptation goals, and design choices. They also highlight that some aspects, such as cost, granularity and flexibility, can be evaluated at both runtime and design time. Moreover, Villegas et al. [104] propose a quality-driven framework for evaluating SASs. They identify key *analysis dimensions*, such as adaptation goals, reference inputs, measured outputs, and computed control actions. Then, the authors focus on the adaptation goal by introducing four main *quality attributes*, which are performance, dependability, security, and safety. These attributes map to self-* properties in SAS for both the adaptation mechanisms and the managed resources. These evaluations of adaptation mechanisms aim to assess the quality of architectural design in SASs and how they function across various use cases. However, none of those mentioned above provides a quantitative evaluation to precisely measure these self-* properties' benefits for targeted systems.

On the other hand, a group of researchers adopt different approaches to evaluating the impact on the performance of utilising self-* properties on the underlying systems. Most of these contributions use quantitative methods to make these evaluations comparable. For example, Kaddoum et al. [47] presents 31 metrics for evaluating SAS that are categorised into *methodological*, *architectural*, *intrinsic*, and *runtime dimensions*. Methodological criteria address the development process, while architectural criteria focus on system growth and separation of concerns. Intrinsic criteria evaluate computational complexity, decentralisation, and external factors, while runtime criteria assess performance metrics such as latency, communication load, and robustness. The authors demonstrate the applicability of the proposed criteria by analysing different case studies from various domains that use self-* proprieties. Similarly, Reinecke et al. [90] present a method for defining an adaptivity metric that facilitates the comparison of SAS according to their adaptive performance. The proposed metric measures how closely the benefits accumulated by a system match those of the optimal system. Additionally, the metric is intentionally independent of the system's internal design. So, It focuses solely on the system's adaptive behaviour and quantitatively measures how effectively the system adapts. The process for establishing an adaptivity metric involves four key steps. 1) Clearly define the system that needs to be evaluated. This includes all adaptation logic and managed resources and their interfaces. At this stage, it is also essential to identify whether the system is fully adaptive or partially adaptive. 2) Identify all the performance metrics representing the system's value and usefulness to its intended purpose. 3) Using the performance metrics to calculate a payoff metric. 4) Observe the performance of SAS in a real-world scenario or the testbed. 5) Calculate the adaptivity metrics using the given formula. The author explains the adaptation process as a sequence of trials where the SAS experiences multiple iterations over time. During these iterations, the system may make positive, neutral, or negative decisions, particularly when its observable performance declines.

Despite all the effort, the complexity of the evaluation of SAS comes from the fact that most of these systems are domain-dependent and customised toward specific goals specified by the system architect. Therefore, a good evaluation approach requires a high level of abstraction and is easily customised and interpreted despite the domain that adapts it. In the literature, we found that the work proposed by Reinecke et al.[90] can be easily adapted for any domain. Yet, it provides quantitative metrics that can easily be understood and compared. In table 11, we have explained and customised each step of Reinecke et al.[90] approach to evaluate self-adaptive IoFT systems with its distinguish distributed nature and the heterogeneity of IoT system.

Table 11. Evaluation of Self-adaptive IoFT Systems

Step 1	Step 2	Step 3	Step 4	Step 5
IoFT Characteristics	**Performance Metrics**	**Payoff Calculation**	**IoFT Testbed**	**Adaptivity Metrics Calculation**
Technique (Sec. 4.2.4), **Time** (Sec. 4.2.1), **Adaptation mechanism** (Sec. 4.2.5), **Level** (Sec. 4.2.3), **Reason** (Sec. 4.2.2)	**Accuracy** [82] [27] [38] [58] [112] [107] [101] [57] [116] [30] [8] [121] [19], **FP/TP** [77], **F-score** [95] [107], **Kappa** [95], **AUC** [107], **Loss** [38] [58] [112] [101] [8], **Recall** [107], **Precision** [107], **Negative log-likelihood** [57], **Average percentage ranking** [57]	$\mathrm{P} : \mathrm{M}_1 \times \mathrm{M}_2 \times \cdots \times \mathrm{M}_n \rightarrow [0, 1]$ where $\mathrm{M} = \{\mathrm{M}_1, \mathrm{M}_2, \ldots, \mathrm{M}_\mathrm{N}\}$, which represents the set of N metrics. $\mathrm{D}_- = \{\mathrm{i} \mid \mathrm{p}_{\mathrm{i}-1} > \mathrm{p}_\mathrm{i}\}$ (Negative) $\mathrm{D}_\Theta = \{\mathrm{i} \mid \mathrm{p}_{\mathrm{i}-1} = \mathrm{p}_\mathrm{i}\}$ (Neutral) $\mathrm{D}_+ = \{\mathrm{i} \mid \mathrm{p}_{\mathrm{i}-1} < \mathrm{p}_\mathrm{i}\}$ (Positive) Here, $\mathbf{i} \in \{2, 3, \ldots, \mathrm{N}\}$, and p_i denotes the probability at trial **i**. The value of p_i must be normalized using the reference scale from the IoFT system in its optimal state.	**Real-world scenario**, **Physical Testbed**, **Laboratory Testbed**, **Simulation** (Sec. 4.6.1)	$\mathrm{Ad} := \frac{\sum_{\mathrm{i} \in \mathrm{D}_+} \Delta_\mathrm{i} + \sum_{\mathrm{i} \in \mathrm{D}_\Theta} \mathrm{p}_\mathrm{i}}{\mathrm{N} - 1}$ $\Delta_\mathrm{i} := \frac{\mathrm{p}_\mathrm{i} + \mathrm{p}_{\mathrm{i}-1}}{2}$ Where Δ_i defines the benefit of a decision, and p_i is the benefit in a trial **i**. $\mathrm{Ad} \in [0, 1]$ where 1 represents the maximum benefit the IoFT system can obtain from employing self-* properties.

4.7 Application of Self-adaptive IoFT

This section shows the recognised application domains in the literature that may benefit from using the concept of self-adaptive IoFT.

4.7.1 Healthcare . The domain of Healthcare faces significant challenges, including patient privacy concerns, data availability issues, and system complexities. For instance, IoT devices collecting patient metrics encounter restrictions on data sharing across organisations for further analysis. Limitations such as obtaining patient information restrict ML-based healthcare applications from scaling effectively. As a result, a self-adaptive IoFT offers a solution by enabling self-adaptive collaborative model training across organisations without sharing patient data to preserve privacy and governance [91]. A self-adaptive IoFT entails handling the issues associated with data collection and energy efficiency-related issues and developing an unbiased model for better diagnosis and treatment. For example, Wang et al. [107] aim to demonstrate self-adaptation in patients' data to diagnose diseases while preserving patients' privacy.

4.7.2 IoFT for Dynamic Context . Self-adaptive IoFT can be implemented in unstable environments, allowing for the dynamic replacement or adjustment of IoFT components in response to changing contexts. Additionally, the techniques outlined in Section 4.2.4 may yield promising results in maintaining system performance and availability. For example, Khan et al. [50] present a self-organizing FL system over a wireless network in smart tourism, which substitutes an unavailable FL server with an alternative candidate. Moreover, Lim et al. [62] propose strategies for FL client associations and resource allocation that help reduce communication overhead for large-scale implementations.

4.7.3 Industrial Internet of Things (IIoT) . Many IIoT systems utilise distributed control systems such as Supervisory Control and Data Acquisition (SCADA) and Programmable Logic Controllers (PLCs) to enable real-time collaboration among sensors, actuators, and controllers. Effective communication between these components is essential to maintain quality of service. Moreover, real-time feedback is critical in applications like cyber-physical systems. Consequently, many of these systems implement inference ML models at the edge. Franco et al. [30] provide an example from the IIoT sector, where they propose a self-adaptive IoFT architecture specifically designed for industrial automation systems.

4.7.4 Smart- .* Many smart applications (e.g., smart places, cities, and homes) utilise a variety of IoT devices to meet user needs. These needs may include ensuring user comfort and privacy and protecting against cyberattacks on smart applications. For instance, Nguyen et al. [77] introduce a self-learning anomaly detection model for the IoFT, designed to identify Mirai malware within smart office environments.

4.8 Self-adaptive IoFT for AD in Smart Home Use Case

This section will apply the conceptual architecture for self-adaptive IoFT for AD classification, as discussed in Section 4.3. AD is finding unexpected items or events that deviate from the norm. It has received much attention from the research community. Nguyen et al. [77] propose DÏoT, a self-learning FL-based AD system for the IoT. The definition of self-adaptation for their work was introduced in Table 6. Nguyen et al.'s [77] system successfully utilised FL to create AD that could identify various IoT devices and build detection profiles based on the device's type for small and home offices. Their system is capable of functioning without human intervention or labelled data. Therefore, it is essential to understand how IoFT for AD works in typical smart places. Figure 7 shows the architecture of using self-adaptive IoFT for AD in smart homes, and Table 12 presents the characteristics of the two scenarios that we used based DÏoT system. This use case has two main components: 1) a security *gateway* as FL client and 2) an *AD service provider* as FL server. The security *gateway* has access to the internet and is responsible for running local AD to identify any infected IoT device. Additionally, it detects the type of device for any newly connected IoT device to the network to create a device-type profile. As part of FL participation, security *gateway* will train the received global model with the local data and share the model weight and the updated *device-type profile* at the end of each round. The AD service provider periodically supports the security *gateway*. Each device type has its AD model stored in a model repository within the *AD service provider*. The security *gateways* within this proposed system are FL clients, whereas the IoT service provider is an FL server.

The proposed architecture includes security *gateways* that act as *adaptive IoT clients*. Their dedicated MAPE-K module enables them to self-learn the type of IoT device based on their communication pattern with the security *gateways*. This self-learning AD is located on the client side of the IoFT paradigm. However, the *AD service provider*, which acts as an FL server, also automatically identifies one of the challenges associated with the AD model based on device-type behaviour. This challenge is the rise of false alarms due to regular device firmware updates provided by one of *IoT service providers* in the market, such as Amazon, Samsung, and Philips, amongst others. This behaviour can affect the overall AD performance as it gets aggregated to a global model, which has a similar impact as a model poisoning attack [29]. Nguyen et al. [77] highlight this issue and suggest that an FL server must track the rising alarm across FL clients to determine the legitimacy of the alarm. We incorporate this aspect in our use case; therefore, the *AD service provider* has three tasks: 1) aggregating the FL client's module, 2) determining if a new *device-type profile* does not match any existing records in the *device-type profile* repository, and 3) dealing with IoT firmware updates that arise from FL clients with similar *device-type profiles*. The two use cases below demonstrate *FL adaptation logic* interpretation within the proposed conceptual architecture of the self-adaptive IoFT, automatic enrolment of FL clients, and address IoT firmware updates for self-adaptive IoFT for AD in smart homes.

FL Client Automatic Enrolment Figure 8a illustrates the *FL adaptation logic* for enrolling new or existing device-type FL client profiles.

- Initial component: The *FL Registration Manager* updates the *knowledge base* with the new client profile, prompting the *Monitor* component to send an analysis request.

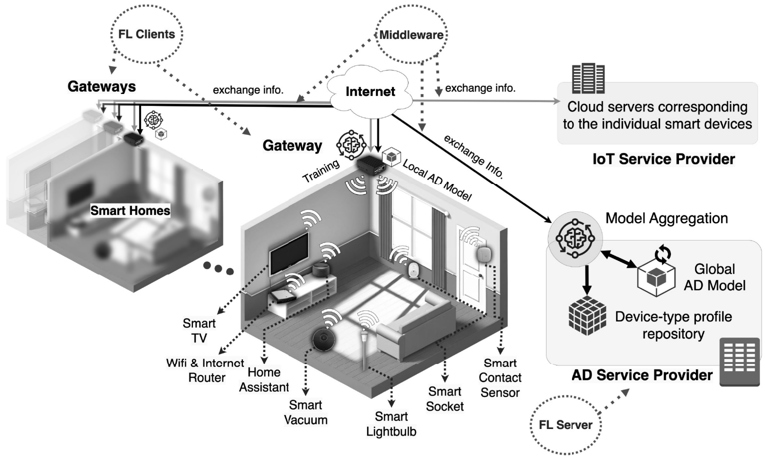
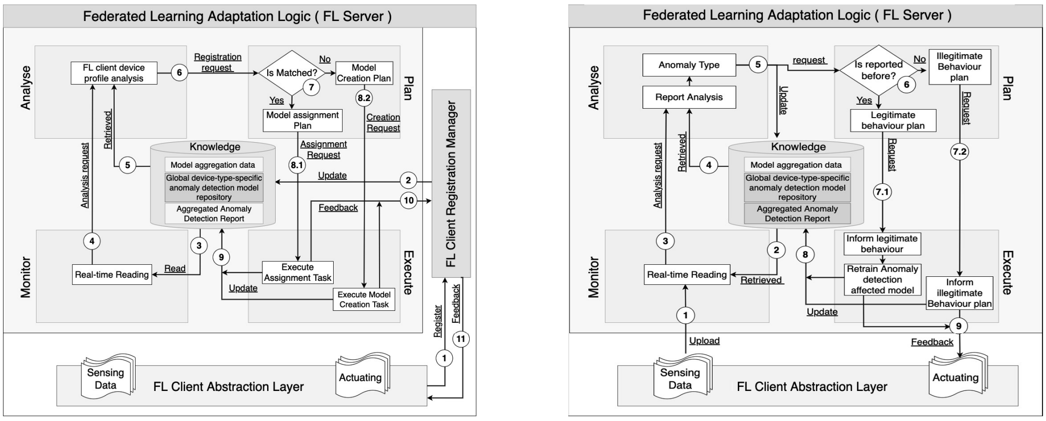


Fig. 7. IoFT architecture for AD in smart homes



(a) Automatic client enrolment

(b) IoT device firmware update spotting

Fig. 8. Two case studies of MAPE-K-based self-adaptive IoFT

- Monitor component: Detects knowledge base changes and forwards the registration request to the *Analyse* component.
- Analyse component: Retrieves device-type AD models from the *Knowledge Base*, analyzes the new profile, and sends a registration request to the *Plan* component.
- Plan component: Decides whether to assign an existing AD model or create a new one. If a match is found, a model assignment request is sent to the *Execute* component; otherwise, a new FL client request is initiated.
- Execute component: Either creates a new random device-type-specific AD model or assigns an existing model to the new FL client. The resulting model is reported to the *FL Registration Manager* and stored in the *Knowledge Base*.

IoT Device Firmware Update Spotting Figure 8b shows the *FL adaptation logic* for firmware updates, focusing on AD during training rounds.

Table 12. Characteristics of Self-Adaptive IoFT in Two DÏoT Use Cases [77] [69]

Use Case	Motivation	Implementation	Evaluation Metrics	Evaluation Testbed	Application
IoT Device Firmware Update Spotting	Improve IoFT Performance	Context-based Approach	FP/TP Rate	Laboratory Testbed	Smart Offices
FL Client Automatic Enrolment	Overcome IoT Device Heterogeneity	Context-based Approach	Precision, Recall, F1-Score	Laboratory Testbed	Smart Homes

- Monitor Component: FL clients submit anomaly reports at the end of each training round, which are forwarded to the *Analyse* component.
- Analyse Component: Detects and categorizes anomalies by comparing them with known cases in the *Knowledge Base*, then sends findings and action requests to the *Plan* component.
- Plan Component: Determines if the anomaly is legitimate or illegitimate based on prior knowledge.
- Execute Component: For legitimate anomalies, false alarms are cancelled and the AD model is retired. For illegitimate anomalies, a verification process is triggered. Outcomes are returned to the FL client via an actuator.

5 CHALLENGES AND FUTURE DIRECTION

Utilising self-adaptive IoFT introduces new features to existing FL and IoT systems. The main goal of this application is to provide enhanced resiliency to an IoFT. Nevertheless, certain challenges must be addressed to fully realise the benefits of adopting this new design paradigm.

5.1 Challenges in Designing Self-adaptive IoFT Systems

One of the biggest challenges is identifying the requirement of the IoFT system that needs to be maintained and how to build a self-adaptive IoFT that fulfils the adaptation requirement. In Section 4.2, we explain the main properties of designing self-adaptive IoFT. As a part of *Technique* used, Parameter adaptation in IoFT can be challenging. An engineer who designs self-adaptive IoFT needs to clearly state which parameter must be optimised automatically (i.e., FL global parameter, Model initial parameter, and Model hyperparameter). An optimisation technique based on a nature-based approach can be used to modify these parameters as discussed in Section 4.5.3. However, attempting to change all the parameters in the IoFT paradigm together is a resource-intensive task. Moreover, performing self-adaptation *proactively* is limited by the need to establish a strict policy to drive the adaptation towards a specific direction – therefore making *reactive* adaptation more appealing for system designers, as it triggers the adaptation process when these policies are violated. Therefore, more work on *proactive* adaptation is needed, especially for critical applications (healthcare, finance, IIoT) where we must start the adaptation process before the violation occurs. Qi et al. [85] follow this direction by introducing proactive handover using FL to maintain quality of service for mobile users in vehicular networks.

5.2 Challenges in Distinct Domains of IoFT Systems

Self-adaptive IoFT systems are exposed to the same limitations that affect FL and IoT systems, as outlined in Section 2.2.7. Additionally, the discussion in Section 2.2.4 highlights that the diversity of IoT systems among FL clients complicates the definition of adaptation logic and the allocation of suitable resources. Furthermore, communication and bandwidth limitations constrain the decision-making process in the design of self-adaptive IoFT systems by limiting the interaction between FL clients and servers. This challenge arises because the adaptation logic requires a reliable interface to manage resources effectively. Another challenge related to IoT devices is their limited computational capacity, which restricts their ability to participate in IoFT systems. Likewise, the availability of FL clients is highly dependent on the IoFT architecture. In cross-device FL, as discussed in section 2.2.3, there can be numerous clients, many of whom may be intermittently available for participation in the training process. Khan et al. [50] proposed a self-organising FL approach for wireless networks to address this challenge. This method improves client availability by forming clusters, registering devices and allocating resources in a completely autonomous technique. Moreover, data privacy requirements imposed by FL clients often limit the degree of context-sharing across FL clients. In Section 4.8, we demonstrated the advantages of sharing device-type profiles across FL clients to detect false alarms resulting from firmware updates of IoT devices. However, the main advantage of FL lies in its ability to maintain privacy across participants. Sharing additional information could accidentally lead to data leakage, thus exposing the IoFT systems to various security threats. Therefore, strict privacy and security requirements for IoFT systems pose significant challenges when designing adaptation logic. Mothukuri et al. [76] present a classification of the security and privacy issues in general FL systems. In Figure 9, we expand upon their proposed threat categorisation by incorporating the unique implementation challenges for adopting various defence techniques specific to self-adaptive IoFT systems. These challenges are further magnified in real-time processing scenarios, where decisions must meet strict latency constraints to maintain system responsiveness and effectiveness. Real-time data streams in IoFT systems require continuous monitoring, updating the predictive models to adapt to new data, and providing rapid feedback mechanisms to respond quickly to changes in the operating environment. The limited memory and processing power of edge devices and the need for real-time data processing add more complexity to implementing self-adaptive IoFT. Therefore, there is a need to design efficient ML model architectures, robust communication protocols, and proactive adaptation strategies to prevent system degradation and preserve privacy for real-time self-adaptive IoFT systems.

5.3 Challenges in Adaptation Logic

The MAPE-K framework is the main engine that drives self-adaptive IoFT towards different states. To utilise them effectively, we need to highlight the difficulty of each stage of the framework. For instance, Which part of the IoFT system needs to be *monitored*? What are the states and behaviours of a self-adaptive IoFT system that need to be *analysed* thoroughly to identify the system's status? What is the best adaptation *plan* for the underlying system to make effective decisions? What is the correct *execution* approach that assures the adaptation is performed correctly? Answering these questions required a comprehensive understanding of the specific IoFT system being utilised to monitor, analyse, plan, and execute adaptations effectively. Also, it requires in-depth knowledge of the system's components, functionalities, and potential challenges that may arise during each framework stage. Jahan et al. [46] present a security-focused feedback control loop that interacts with the MAPE-K framework to dynamically manage runtime adaptation for the changes in functional and security conditions. Their approach provides two control loops that interact with each other to reduce the effort undertaken at the design stage for building a reliable MAPE-K framework. Although there have been efforts to tackle the challenges of designing the MAPE-K framework in various domains, this issue warrants further focus when implemented in a self-adaptive IoFT system.

5.4 Challenges in Evaluating self-adaptive IoFT

Evaluating self-adaptive IoFT systems presents significant challenges due to their reliance on FL and IoT infrastructures. Previous evaluation frameworks for SASs, such as those by McCann and

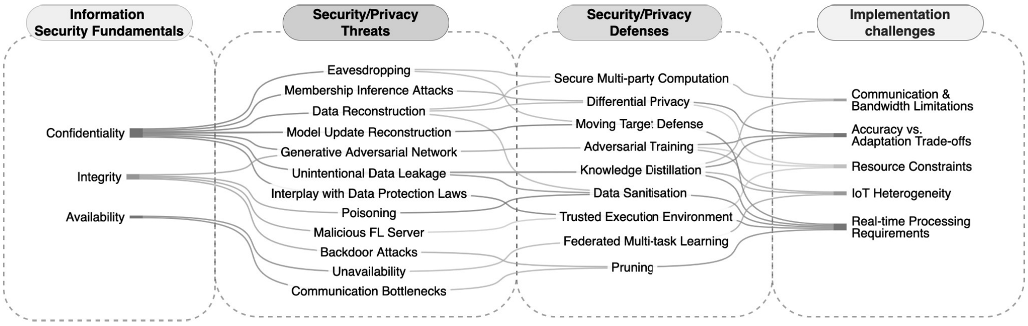

Information Security Fundamentals
Security/Privacy Threats
Security/Privacy Defenses
Implementation challenges
Confidentiality
Integrity
Availability
Eavesdropping
Membership Inference Attacks
Data Reconstruction
Model Update Reconstruction
Generative Adversarial Network
Unintentional Data Leakage
Interplay with Data Protection Laws
Poisoning
Malicious FL Server
Backdoor Attacks
Unavailability
Communication Bottlenecks
Secure Multi-party Computation
Differential Privacy
Moving Target Defense
Adversarial Training
Knowledge Distillation
Data Sanitisation
Trusted Execution Environment
Federated Multi-task Learning
Pruning
Communication & Bandwidth Limitations
Accuracy vs. Adaptation Trade-offs
Resource Constraints
IoT Heterogeneity
Real-time Processing Requirements

Fig. 9. Overview of security and privacy threats in IoFT systems

Huebscher [70], address core concerns related to scalability and responsiveness but lack detailed case studies or practical tool support. Similarly, Villegas et al. [104] offer a quality-driven framework without practical implementation. Moreover, the quantitative evaluation proposed by Reinecke et al. [90] does not include the architectural properties often implicit or distributed across the system. These challenges are observable in IoFT systems due to device heterogeneity, non-IID data, limited client availability, and concerns regarding privacy and security. Current evaluation approaches do not adequately capture the unique interplay of FL and IoT constraints. As a result, the lack of a suitable evaluation approach emphasises the need for a generalisable framework to address these challenges. Addressing this gap could open new research directions that enhance the scalability and reliability of SASs in general and IoFT systems specifically.

6 CONCLUSION

Self-adaptive IoFT combines three distinctive domains (i.e., SAS, FL, and IoT systems). This work presents a systematic review establishing a comprehensive framework for understanding and implementing self-adaptive IoFT systems. Moreover, we propose a conceptual architecture that combines MAPE-K feedback loops with FL server-client interactions. This architecture provides a foundation for implementing self-* properties while maintaining privacy by separating adaptation logic and managed resources. Our analysis identified four main approaches to implementing self-adaptive IoFT systems (i.e., context-based, ML-based, nature-based, and agent-based), each with distinct advantages. Additionally, these implementation techniques illustrate the applicability of self-adaptive IoFT systems in various application domains (e.g., healthcare, industrial Internet of Things, and smart places), where collaborative learning and adaption in real time are essential. On the other hand, additional efforts are required to implement a successful self-adaptive IoFT system reliably. This review highlights a few challenges affecting the reliability of the adaptation mechanism, including privacy preservation versus adaptability, heterogeneous IoT devices, and communication overhead. Moreover, the literature emphasises the importance of exploring proactive and efficient adaptation strategies for resource-constrained IoT devices and standardised evaluation frameworks, which will benefit researchers and developers of self-adaptive IoFT systems.

ACKNOWLEDGMENTS

This work has been partially supported by the PETRAS National Centre of Excellence for IoT Systems Cybersecurity (Resilient Built Environments (ResBE) Project) EP/S035362/1, Digital Catapult Researcher-in-Residence program (EP/T517203/1), Innovate UK CyberASAP, GCHQ National Resilience Fellowship, and EPSRC National Edge Artificial Intelligence Hub (EP/Y028813/1).

REFERENCES

[1] Sawsan Abdulrahman, Hanine Tout, Hakima Ould-Slimane, Azzam Mourad, Chamseddine Talhi, and Mohsen Guizani. 2021. A Survey on Federated Learning: The Journey From Centralized to Distributed On-Site Learning and Beyond. *IEEE Internet of Things Journal* 8, 7 (apr 2021), 5476–5497.

[2] Gregory D. Abowd, Anind K. Dey, Peter J. Brown, Nigel Davies, Mark Smith, and Pete Steggles. 1999. Towards a Better Understanding of Context and Context-Awareness. In *Handheld and Ubiquitous Computing*, Hans-W. Gellersen (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 304–307.

[3] Rima Al-Ali, Lubomír Bulej, Jan Kofroň, and Tomáš Bureš. 2022. A guide to design uncertainty-aware self-adaptive components in Cyber–Physical Systems. *Future Generation Computer Systems* 128 (mar 2022), 466–489.

[4] Ivan Dario Paez Anaya, Viliam Simko, Johann Bourcier, Noël Plouzeau, and Jean-Marc Jézéquel. 2014. A Prediction-Driven Adaptation Approach for Self-Adaptive Sensor Networks. In *Proceedings of the 9th International Symposium on Software Engineering for Adaptive and Self-Managing Systems* (Hyderabad, India) *(SEAMS 2014)*. Association for Computing Machinery, New York, NY, USA, 145–154.

[5] Jesper Andersson, Rogério De Lemos, Sam Malek, and Danny Weyns. 2009. Modeling dimensions of self-adaptive software systems. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 5525 LNCS (2009), 27–47.

[6] Muhammad Asad, Ahmed Moustafa, and Takayuki Ito. 2021. Federated Learning Versus Classical Machine Learning: A Convergence Comparison. arXiv:2107.10976 [cs.LG]

[7] Smith Baker. 1897. The identification of the self. *Psychological Review* 4 (5 1897), 272–284. Issue 3.

[8] Luciano Baresi, Giovanni Quattrocchi, and Nicholas Rasi. 2021. Federated Machine Learning as a Self-Adaptive Problem. *Proceedings - 2021 International Symposium on Software Engineering for Adaptive and Self-Managing Systems, SEAMS 2021* (may 2021), 41–47.

[9] Firas Bayram, Bestoun S. Ahmed, and Andreas Kassler. 2022. From Concept Drift to Model Degradation: An Overview on Performance-Aware Drift Detectors. arXiv:2203.11070 [cs.LG]

[10] Paolo Bellavista, Luca Foschini, and Alessio Mora. 2021. Decentralised Learning in Federated Deployment Environments: A System-Level Survey. *ACM Comput. Surv.* 54, 1, Article 15 (Feb. 2021), 38 pages.

[11] Daniel J. Beutel, Taner Topal, Akhil Mathur, Xinchi Qiu, Javier Fernandez-Marques, Yan Gao, Lorenzo Sani, Kwing Hei Li, Titouan Parcollet, Pedro Porto Buarque de Gusmão, and Nicholas D. Lane. 2020. Flower: A Friendly Federated Learning Research Framework.

[12] Abhishek Bhowmick, John Duchi, Julien Freudiger, Gaurav Kapoor, and Ryan Rogers. 2019. Protection Against Reconstruction and Its Applications in Private Federated Learning.

[13] Pearl Brereton, Barbara A. Kitchenham, David Budgen, Mark Turner, and Mohamed Khalil. 2007. Lessons from Applying the Systematic Literature Review Process within the Software Engineering Domain. *J. Syst. Softw.* 80, 4 (apr 2007), 571–583.

[14] P J Brown. 1996. The stick-e document: a framework for creating context-aware applications. *Electronic Publishing 96* 8 (1996). Issue SEPTEMBER 1995.

[15] Yuriy Brun, Ron Desmarais, Kurt Geihs, Marin Litoiu, Antonia Lopes, Mary Shaw, and Michael Smit. 2013. A Design Space for Self-Adaptive Systems. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 7475 LNCS (2013), 33–50.

[16] Konstantin Burlachenko, Samuel Horváth, and Peter Richtárik. 2021. FL_PyTorch. In *Proceedings of the 2nd ACM International Workshop on Distributed Machine Learning*. ACM.

[17] Sebastian Caldas2018, Sai Meher Karthik Duddu, Peter Wu, Tian Li, Jakub Konečný, H. Brendan McMahan, Virginia Smith, and Ameet Talwalkar. 2018. LEAF: A Benchmark for Federated Settings.

[18] Bihuan Chen, Xin Peng, Yijun Yu, Bashar Nuseibeh, and Wenyun Zhao. 2014. Self-Adaptation through Incremental Generative Model Transformations at Runtime. In *Proceedings of the 36th International Conference on Software Engineering* (Hyderabad, India) *(ICSE 2014)*. Association for Computing Machinery, New York, NY, USA, 676–687.

[19] Chuan Chen, Weibo Hu, Ziyue Xu, and Zibin Zheng. 2021. FedGL: Federated Graph Learning Framework with Global Self-Supervision. (may 2021). arXiv:2105.03170

[20] Xing Chen, Junxin Lin, Bing Lin, Tao Xiang, Ying Zhang, and Gang Huang. 2019. Self-learning and self-adaptive resource allocation for cloud-based software services. *Concurrency and Computation: Practice and Experience* 31, 23 (2019), e4463. arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/cpe.4463 e4463 CPE-17-0360.

[21] Yujing Chen, Yue Ning, Martin Slawski, and Huzefa Rangwala. 2020. Asynchronous Online Federated Learning for Edge Devices with Non-IID Data. arXiv:1911.02134 [cs.DC]

[22] Kewei Cheng, Tao Fan, Yilun Jin, Yang Liu, Tianjian Chen, Dimitrios Papadopoulos, and Qiang Yang. 2021. SecureBoost: A Lossless Federated Learning Framework. *IEEE Intelligent Systems* 36, 6 (2021), 87–98.

[23] D Denyer and D Tranfield. 2009. The Sage Handbook of Organizational Research Methods. , 671-689 pages.

[24] Semiconductor Digest. 2017. Number of connected IOT devices will surge to 125 billion by 2030.

[25] Nathalia Moraes do Nascimento and Carlos José Pereira de Lucena. 2017. FIoT: An agent-based framework for self-adaptive and self-organizing applications based on the Internet of Things. *Information Sciences* 378 (2017), 161–176.

[26] Simon Dobson, Spyros Denazis, Antonio Fernández, Dominique Gaïti, Erol Gelenbe, Fabio Massacci, Paddy Nixon, Fabrice Saffre, Nikita Schmidt, and Franco Zambonelli. 2006. A survey of autonomic communications. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)* 1, 2 (dec 2006), 223–259.

[27] Moming Duan, Duo Liu, Xianzhang Chen, Yujuan Tan, Jinting Ren, Lei Qiao, and Liang Liang. 2019. Astraea: Self-balancing Federated Learning for Improving Classification Accuracy of Mobile Deep Learning Applications. *Proceedings - 2019 IEEE International Conference on Computer Design, ICCD 2019* (jul 2019), 246–254. arXiv:1907.01132v2

[28] Abdessalam Elhabbash, Maria Salama, Rami Bahsoon, and Peter Tino. 2019. Self-awareness in Software Engineering. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)* 14, 2 (oct 2019).

[29] Minghong Fang, Xiaoyu Cao, Jinyuan Jia, and Neil Gong. 2020. Local Model Poisoning Attacks to Byzantine-Robust Federated Learning. In *29th USENIX Security Symposium (USENIX Security 20)*. USENIX Association, 1605–1622.

[30] Nicola Franco, Hoai My Van, Marc Dreiser, and Gereon Weiss. 2021. Towards a Self-Adaptive Architecture for Federated Learning of Industrial Automation Systems. *Proceedings - 2021 International Symposium on Software Engineering for Adaptive and Self-Managing Systems, SEAMS 2021* (may 2021), 210–216.

[31] David Garlan, S.-W. Cheng, A.-C. Huang, Bradley Schmerl, and Peter Steenkiste. 2004. Rainbow: Architecture-based self-adaptation with reusable infrastructure. *Computer* 37, 10 (oct 2004), 46–54.

[32] Goffman, Erving, , and University of Edinburgh. 1956. *The presentation of self in everyday life*. University of Edinburgh, Social Sciences Research Centre Edinburgh. 162p. pages.

[33] Martin Goller and Sven Tomforde. 2021. On the stability of (self-)adaptive behaviour in continuously changing environments: A quantification approach. *Array* 11 (sep 2021), 100069.

[34] Badra Souhila Guendouzi, Samir Ouchani, Hiba EL Assaad, and Madeleine EL Zaher. 2023. A systematic review of federated learning: Challenges, aggregation methods, and development tools. *J. Netw. Comput. Appl.* 220, C (Nov. 2023), 28 pages.

[35] Song Han, Huizi Mao, and William J. Dally. 2016. Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding. arXiv:1510.00149 [cs.CV]

[36] Andrew Hard, Chloé M Kiddon, Daniel R Ramage, Françoise Simone Beaufays, Hubert Eichner, Kanishka Rao, Rajiv Mathews, Sean Augenstein, and Swaroop Ramaswamy. 2019. Federated Learning for Mobile Keyboard Prediction.

[37] Chaoyang He, Songze Li, Jinhyun So, Mi Zhang, Hongyi Wang, Xiaoyang Wang, Praneeth Vepakomma, Abhishek Singh, Hang Qiu, Li Shen, Peilin Zhao, Yan Kang, Yang Liu, Ramesh Raskar, Qiang Yang, Murali Annavaram, and Salman Avestimehr. 2020. FedML: A Research Library and Benchmark for Federated Machine Learning. *Advances in Neural Information Processing Systems, Best Paper Award at Federate Learning Workshop* (2020).

[38] Chaoyang He, Zhengyu Yang, Erum Mushtaq, Sunwoo Lee, Mahdi Soltanolkotabi, and Salman Avestimehr. 2021. SSFL: Tackling Label Deficiency in Federated Learning via Personalized Self-Supervision. (oct 2021). arXiv:2110.02470

[39] Ryan Heartfield, George Loukas, Anatolij Bezemskij, and Emmanouil Panaousis. 2021. Self-Configurable Cyber-Physical Intrusion Detection for Smart Homes Using Reinforcement Learning. *IEEE Transactions on Information Forensics and Security* 16 (2021), 1720–1735.

[40] Petr Jan Horn. 2001. Autonomic Computing: IBM's Perspective on the State of Information Technology.

[41] Tzu-Ming Harry Hsu, Hang Qi, and Matthew Brown. 2019. Measuring the Effects of Non-Identical Data Distribution for Federated Visual Classification. arXiv:1909.06335 [cs.LG]

[42] Chao Huang, Jianwei Huang, and Xin Liu. 2022. Cross-Silo Federated Learning: Challenges and Opportunities. arXiv:2206.12949 [cs]

[43] Shihong Huang and Pedro Miranda. 2015. Incorporating Human Intention into Self-Adaptive Systems. In *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, Vol. 2. 571–574.

[44] Markus C. Huebscher and Julie A. McCann. 2008. A Survey of Autonomic Computing—Degrees, Models, and Applications. *ACM Comput. Surv.* 40, 3, Article 7 (aug 2008), 28 pages.

[45] Matthias Hölzl and Thomas Gabor. 2015. Continuous Collaboration: A Case Study on the Development of an Adaptive Cyber-physical System. In *2015 IEEE/ACM 1st International Workshop on Software Engineering for Smart Cyber-Physical Systems*. 19–25.

[46] Sharmin Jahan, Ian Riley, Charles Walter, Rose F. Gamble, Matt Pasco, Philip K. McKinley, and Betty H.C. Cheng. 2020. MAPE-K/MAPE-SAC: An interaction framework for adaptive systems with security assurance cases. *Future Generation Computer Systems* 109 (2020).

[47] Elsy Kaddoum, Claudia Raibulet, Jean-Pierre Georgé, Gauthier Picard, and Marie-Pierre Gleizes. 2010. Criteria for the Evaluation of Self-* Systems. In *Proceedings of the 2010 ICSE Workshop on Software Engineering for Adaptive and Self-Managing Systems (SEAMS '10)*. Association for Computing Machinery, New York, NY, USA, 29–38.

[48] Dhrgam AL Kafaf and Dae-Kyoo Kim. 2017. A web service-based approach for developing self-adaptive systems. *Computers & Electrical Engineering* 63 (2017), 260–276.

[49] Jeffrey O. Kephart and David M. Chess. 2003. The vision of autonomic computing. *Computer* 36, 1 (2003), 41–50.

[50] Latif U. Khan, Madyan Alsenwi, Zhu Han, and Choong Seon Hong. 2020. Self Organizing Federated Learning Over Wireless Networks: A Socially Aware Clustering Approach. *International Conference on Information Networking* 2020-January (jan 2020), 453–458.

[51] Raed Kontar, Naichen Shi, Xubo Yue, Seokhyun Chung, Eunshin Byon, Mosharaf Chowdhury, Judy Jin, Wissam Kontar, Neda Masoud, Maher Noueihed, Chinedum E. Okwudire, Garvesh Raskutti, Romesh Saigal, Karandeep Singh, and Zhisheng Ye. 2021. The Internet of Federated Things (IoFT): A Vision for the Future and In-depth Survey of Data-driven Approaches for Federated Learning. arXiv:2111.05326 [cs.LG]

[52] Samuel Kounev, Peter Lewis, Kirstie L. Bellman, Nelly Bencomo, Javier Camara, Ada Diaconescu, Lukas Esterle, Kurt Geihs, Holger Giese, Sebastian Götz, Paola Inverardi, Jeffrey O. Kephart, and Andrea Zisman. 2017. The Notion of Self-aware Computing. *Self-Aware Computing Systems* (2 2017), 3–16.

[53] Christian Krupitzer, Felix Maximilian Roth, Sebastian Vansyckel, Gregor Schiele, and Christian Becker. 2015. A survey on engineering approaches for self-adaptive systems. *Pervasive and Mobile Computing* 17, PB (feb 2015), 184–206.

[54] Rajesh Kumar, Abdullah Aman Khan, Jay Kumar, Zakria, Noorbakhsh Amiri Golilarz, Simin Zhang, Yang Ting, Chengyu Zheng, and Wenyong Wang. 2021. Blockchain-Federated-Learning and Deep Learning Models for COVID-19 Detection Using CT Imaging. *IEEE Sensors Journal* 21, 14 (2021), 16301–16314.

[55] Sachin Kumar, Prayag Tiwari, and Mikhail Zymbler. 2019. Internet of Things is a revolutionary approach for future technology enhancement: a review. *Journal of Big Data* 6, 1 (dec 2019).

[56] David Leroy, Alice Coucke, Thibaut Lavril, Thibault Gisselbrecht, and Joseph Dureau. 2019. Federated Learning for Keyword Spotting. In *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 6341–6345.

[57] Anliang Li, Shuang Wang, Wenzhu Li, Shengnan Liu, and Siyuan Zhang. 2020. Predicting Human Mobility with Federated Learning. In *GIS: Proceedings of the ACM International Symposium on Advances in Geographic Information Systems*. ACM, New York, NY, USA, 441–444.

[58] Li Li, Moming Duan, Duo Liu, Yu Zhang, Ao Ren, Xianzhang Chen, Yujuan Tan, and Chengliang Wang. 2021. FedSAE: A Novel Self-Adaptive Federated Learning Framework in Heterogeneous Systems. In *2021 International Joint Conference on Neural Networks (IJCNN)*, Vol. 2021-July. IEEE, 1–10. arXiv:2104.07515

[59] Qinbin Li, Yiqun Diao, Quan Chen, and Bingsheng He. 2022. Federated Learning on Non-IID Data Silos: An Experimental Study. In *2022 IEEE 38th International Conference on Data Engineering (ICDE)*. 965–978.

[60] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. 2020. Federated Optimization in Heterogeneous Networks. arXiv:1812.06127 [cs.LG]

[61] Tian Li, Maziar Sanjabi, Ahmad Beirami, and Virginia Smith. 2020. Fair Resource Allocation in Federated Learning. arXiv:1905.10497 [cs.LG]

[62] Wei Yang Bryan Lim, Jer Shyuan Ng, Zehui Xiong, Dusit Niyato, Chunyan Miao, and Dong In Kim. 2021. Dynamic Edge Association and Resource Allocation in Self-Organizing Hierarchical Federated Learning Networks. *IEEE Journal on Selected Areas in Communications* 39, 12 (dec 2021), 3640–3653.

[63] Wei Yang Bryan Lim, Zehui Xiong, Chunyan Miao, Dusit Niyato, Qiang Yang, Cyril Leung, and H. Vincent Poor. 2020. Hierarchical Incentive Mechanism Design for Federated Machine Learning in Mobile Networks. *IEEE Internet of Things Journal* 7, 10 (oct 2020), 9575–9588.

[64] Chang Liu, Shaoyong Guo, Song Guo, Yong Yan, Xuesong Qiu, and Suxiang Zhang. 2022. LTSM: Lightweight and Trusted Sharing Mechanism of IoT Data in Smart City. *IEEE Internet of Things Journal* 9, 7 (2022), 5080–5093.

[65] Yang Liu, Yan Kang, Chaoping Xing, Tianjian Chen, and Qiang Yang. 2020. A Secure Federated Transfer Learning Framework. (2020). arXiv:1812.03337v2

[66] Ziyao Liu, Jiale Guo, Kwok-Yan Lam, and Jun Zhao. 2023. Efficient Dropout-Resilient Aggregation for Privacy-Preserving Machine Learning. *IEEE Transactions on Information Forensics and Security* 18 (2023), 1839–1854.

[67] Sin Kit Lo, Qinghua Lu, Chen Wang, Hye-Young Paik, and Liming Zhu. 2021. A Systematic Literature Review on Federated Machine Learning: From a Software Engineering Perspective. *ACM Comput. Surv.* 54, 5, Article 95 (may 2021), 39 pages.

[68] Huihua Lu and Bojan Cukic. 2012. An Adaptive Approach with Active Learning in Software Fault Prediction. In *Proceedings of the 8th International Conference on Predictive Models in Software Engineering* (Lund, Sweden) *(PROMISE '12)*. Association for Computing Machinery, New York, NY, USA, 79–88.

[69] Samuel Marchal, Markus Miettinen, Thien Duc Nguyen, Ahmad-Reza Sadeghi, and N. Asokan. 2019. AuDI: Toward Autonomous IoT Device-Type Identification Using Periodic Communication. *IEEE Journal on Selected Areas in Communications* 37, 6 (June 2019), 1402–1412.

[70] Julie A. McCann and Markus C. Huebscher. 2004. Evaluation Issues in Autonomic Computing. In *Grid and Cooperative Computing - GCC 2004 Workshops*, Hai Jin, Yi Pan, Nong Xiao, and Jianhua Sun (Eds.). Springer, Berlin, Heidelberg, 597–608.

[71] P.K. McKinley, S.M. Sadjadi, E.P. Kasten, and B.H.C. Cheng. 2004. Composing adaptive software. *Computer* 37, 7 (2004), 56–64.

[72] H. Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. 2016. Communication-Efficient Learning of Deep Networks from Decentralized Data. (2016).

[73] Wondimagegn Mengist, Teshome Soromessa, and Gudina Legese. 2020. Method for conducting systematic literature review and meta-analysis for environmental science research. *MethodsX* 7 (2020), 100777.

[74] Alessio Mora, Armir Bujari, and Paolo Bellavista. 2024. Enhancing generalization in Federated Learning with heterogeneous data: A comparative literature review. *Future Gener. Comput. Syst.* 157, C (July 2024), 1–15.

[75] Alessio Mora, Irene Tenison, Paolo Bellavista, and Irina Rish. 2022. Knowledge Distillation for Federated Learning: a Practical Guide. arXiv:2211.04742 [cs.LG]

[76] Viraaji Mothukuri, Reza M. Parizi, Seyedamin Pouriyeh, Yan Huang, Ali Dehghantanha, and Gautam Srivastava. 2021. A Survey on Security and Privacy of Federated Learning. *Future Generation Computer Systems* 115 (Feb. 2021), 619–640.

[77] Thien Duc Nguyen, Samuel Marchal, Markus Miettinen, Hossein Fereidooni, N. Asokan, and Ahmad Reza Sadeghi. 2018. DIoT: A Federated Self-learning Anomaly Detection System for IoT. *Proceedings - International Conference on Distributed Computing Systems* 2019-July (apr 2018), 756–767. arXiv:1804.07474

[78] Sandro Nižetić, Petar Šolić, Diego López-de-Ipiña González-de-Artaza, and Luigi Patrono. 2020. Internet of Things (IoT): Opportunities, issues and challenges towards a smart and sustainable future. *Journal of Cleaner Production* 274 (nov 2020), 122877.

[79] NVIDIA. 2016. NVIDIA Clara - NVIDIA Developer. https://developer.nvidia.com/clara, Last accessed on 2023-08-08.

[80] Peyman Oreizy, Michael M. Gorlick, Richard N. Taylor, Dennis Heimbigner, Gregory Johnson, Nenad Medvidovc, Alex Quilici, David S. Rosenblum, and Alexander L. Wolf. 1999. An architecture-based approach to self-adaptive software. *IEEE Intelligent Systems and Their Applications* 14, 3 (1999), 54–62.

[81] Sinno Jialin Pan and Qiang Yang. 2010. A Survey on Transfer Learning. *IEEE Transactions on Knowledge and Data Engineering* 22, 10 (2010), 1345–1359.

[82] Junjie Pang, Yan Huang, Zhenzhen Xie, Qilong Han, and Zhipeng Cai. 2021. Realizing the Heterogeneity: A Self-Organized Federated Learning Framework for IoT. *IEEE Internet of Things Journal* 8, 5 (mar 2021), 3088–3098.

[83] Charith Perera, Arkady Zaslavsky, Peter Christen, and Dimitrios Georgakopoulos. 2014. Context Aware Computing for The Internet of Things: A Survey. *IEEE Communications Surveys & Tutorials* 16, 1 (2014), 414–454.

[84] Ioan Petri, Omer F. Rana, Luiz F. Bittencourt, Daniel Balouek-Thomert, and Manish Parashar. 2021. Autonomics at the Edge: Resource Orchestration for Edge Native Applications. *IEEE Internet Computing* 25, 4 (July 2021), 21–29.

[85] Kaiqiang Qi, Tingting Liu, and Chenyang Yang. 2021. Federated Learning Based Proactive Handover in Millimeter-wave Vehicular Networks. arXiv:2101.07032 [eess.SP]

[86] Kaiqiang Qi and Chenyang Yang. 2020. Popularity Prediction with Federated Learning for Proactive Caching at Wireless Edge. In *2020 IEEE Wireless Communications and Networking Conference (WCNC)*. 1–6.

[87] Sashank Reddi, Zachary Charles, Manzil Zaheer, Zachary Garrett, Keith Rush, Jakub Konečný, Sanjiv Kumar, and H. Brendan McMahan. 2021. Adaptive Federated Optimization. arXiv:2003.00295 [cs.LG]

[88] Sashank Reddi, Zachary Charles, Manzil Zaheer, Zachary Garrett, Keith Rush, Jakub Konečný, Sanjiv Kumar, and H. Brendan McMahan. 2021. Adaptive Federated Optimization. arXiv:2003.00295 [cs.LG]

[89] G Anthony Reina, Alexey Gruzdev, Patrick Foley, Olga Perepelkina, Mansi Sharma, Igor Davidyuk, Ilya Trushkin, Maksim Radionov, Aleksandr Mokrov, Dmitry Agapov, Jason Martin, Brandon Edwards, Micah J. Sheller, Sarthak Pati, Prakash Narayana Moorthy, Shih han Wang, Prashant Shah, and Spyridon Bakas. 2021. OpenFL: An open-source framework for Federated Learning. arXiv:2105.06413 [cs.LG]

[90] Philipp Reinecke, Katinka Wolter, and Aad van Moorsel. 2010. Evaluating the Adaptivity of Computing Systems. *Performance Evaluation* 67, 8 (Aug. 2010), 676–693.

[91] Nicola Rieke, Jonny Hancox, Wenqi Li, Fausto Milletarì, Holger R. Roth, Shadi Albarqouni, Spyridon Bakas, Mathieu N. Galtier, Bennett A. Landman, Klaus Maier-Hein, Sébastien Ourselin, Micah Sheller, Ronald M. Summers, Andrew Trask, Daguang Xu, Maximilian Baust, and M. Jorge Cardoso. 2020. The future of digital health with federated learning. *npj Digital Medicine* 3, 1 (Sept. 2020).

[92] Arthur Rodrigues, Genaína Nunes Rodrigues, Alessia Knauss, Raian Ali, and Hugo Andrade. 2019. Enhancing context specifications for dependable adaptive systems: A data mining approach. *Information and Software Technology* 112 (2019), 115–131.

[93] L. Rodrigues, J. Guerreiro, and N. Correia. 2021. Resource design in federated sensor networks using RELOAD/CoAP overlay architectures. *Computer Communications* 179 (nov 2021), 11–21.

[94] Theo Ryffel, Andrew Trask, Morten Dahl, Bobby Wagner, Jason Mancuso, Daniel Rueckert, and Jonathan Passerat-Palmbach. 2018. A generic framework for privacy preserving deep learning.

[95] Aaqib Saeed, Flora D. Salim, Tanir Ozcelebi, and Johan Lukkien. 2020. Federated Self-Supervised Learning of Multi-Sensor Representations for Embedded Intelligence. *IEEE Internet of Things Journal* 8, 2 (jul 2020), 1030–1040. arXiv:2007.13018

[96] Mazeiar Salehie and Ladan Tahvildari. 2009. Self-adaptive software: Landscape and research challenges. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)* 4, 2 (may 2009).

[97] Theresia Ratih Dewi Saputri and Seok-Won Lee. 2020. The Application of Machine Learning in Self-Adaptive Systems: A Systematic Literature Review. *IEEE Access* 8 (2020), 205948–205967.

[98] Bogdan Solomon, Dan Ionescu, Marin Litoiu, and Mircea Mihaescu. 2007. A Real-Time Adaptive Control of Autonomic Computing Environments. In *Proceedings of the 2007 Conference of the Center for Advanced Studies on Collaborative Research* (Richmond Hill, Ontario, Canada) *(CASCON '07)*. IBM Corp., USA, 124–136.

[99] Joel Stremmel and Arjun Singh. 2020. Pretraining Federated Text Models for Next Word Prediction.

[100] Su, Zhou, Wang, Yuntao, Luan, Tom H., Zhang, Ning, Li, Feng, Chen, Tao, Cao, and Hui. 2022. Secure and Efficient Federated Learning for Smart Grid With Edge-Cloud Collaboration. *IEEE Transactions on Industrial Informatics* 18, 2 (2022), 1333–1344.

[101] Prohim Tam, Sa Math, Chaebeen Nam, and Seokhoon Kim. 2021. Adaptive Resource Optimized Edge Federated Learning in Real-Time Image Sensing Classifications. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* 14 (2021), 10929–10940.

[102] Alysa Ziying Tan, Han Yu, Lizhen Cui, and Qiang Yang. 2023. Towards Personalized Federated Learning. *IEEE Transactions on Neural Networks and Learning Systems* 34, 12 (2023), 9587–9603.

[103] TensorFlow. 2016. TensorFlow Federated. https://www.tensorflow.org/federated, Last accessed on 2023-08-08.

[104] Norha M. Villegas, Hausi A. Müller, Gabriel Tamura, Laurence Duchien, and Rubby Casallas. 2011. A Framework for Evaluating Quality-Driven Self-Adaptive Software Systems. In *Proceedings of the 6th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS '11)*. Association for Computing Machinery, New York, NY, USA, 80–89.

[105] Wang and Yanjun Ma anddianhai Yu andtian. 2019. PaddlePaddle: An Open-Source Deep Learning Platform from Industrial Practice. *Frontiers of Data and Domputing* 1, 1 (Jan. 2019), 105–115.

[106] Kangkang Wang, Rajiv Mathews, Chloé Kiddon, Hubert Eichner, Françoise Beaufays, and Daniel Ramage. 2019. Federated Evaluation of On-device Personalization. arXiv:1910.10252 [cs.LG]

[107] Qingyong Wang and Yun Zhou. 2022. FedSPL: federated self-paced learning for privacy-preserving disease diagnosis. *Briefings in Bioinformatics* 23, 1 (jan 2022).

[108] Shiqiang Wang, Tiffany Tuor, Theodoros Salonidis, Kin K. Leung, Christian Makaya, Ting He, and Kevin Chan. 2019. Adaptive Federated Learning in Resource Constrained Edge Computing Systems. *IEEE Journal on Selected Areas in Communications* 37, 6 (2019), 1205–1221.

[109] Yang, Liu, Zhang, Junxue, Chaiand Di, Wangand Leye, Guo, Kun, Chen, Kai, Yang, and Qiang. 2021. Practical and Secure Federated Recommendation with Personalized Masks.

[110] Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. 2019. Federated Machine Learning: Concept and Applications. (2019).

[111] Timothy Yang, Galen Andrew, Hubert Eichner, Haicheng Sun, Wei Li, Nicholas Kong, Daniel Ramage, and Françoise Beaufays. 2018. Applied Federated Learning: Improving Google Keyboard Query Suggestions.

[112] HengJie Yawen Xu, Xiaojun Li, Zeyu Yang, Yawen Xu, and Hengjie Song. 2020. Robust communication strategy for federated learning by incorporating self-attention. *https://doi.org/10.1117/12.2581491* 11584, 10 (nov 2020), 336–341.

[113] Mang Ye, Xiuwen Fang, Bo Du, Pong C. Yuen, and Dacheng Tao. 2023. Heterogeneous Federated Learning: State-of-the-art and Research Challenges. arXiv:2307.10616 [cs.LG]

[114] Yunfan Ye, Shen Li, Fang Liu, Yonghao Tang, and Wanting Hu. 2020. EdgeFed: Optimized Federated Learning Based on Edge Computing. *IEEE Access* 8 (2020), 209191–209198.

[115] Dong Yin, Yudong Chen, Kannan Ramchandran, and Peter Bartlett. 2021. Byzantine-Robust Distributed Learning: Towards Optimal Statistical Rates. arXiv:1803.01498 [cs.LG]

[116] Meng yuan Zhu, Zhuo Chen, Ke fan Chen, Na Lv, and Yun Zhong. 2022. Attention-based federated incremental learning for traffic classification in the Internet of Things. *Computer Communications* 185 (mar 2022), 168–175.

[117] Chen Zhang, Yu Xie, Hang Bai, Bin Yu, Weihong Li, and Yuan Gao. 2021. A survey on federated learning. *Knowledge-Based Systems* 216 (2021), 106775.

[118] Qi Zhang, Lu Cheng, and Raouf Boutaba. 2010. Cloud computing: State-of-the-art and research challenges. *Journal of Internet Services and Applications* 1 (5 2010), 7–18. Issue 1.

[119] Rui Zhang, Bayu Distiawan Trisedy, Miao Li, Yong Jiang, and Jianzhong Qi. 2021. A Benchmark and Comprehensive Survey on Knowledge Graph Entity Alignment via Representation Learning.

[120] Yue Zhao, Meng Li, Liangzhen Lai, Naveen Suda, Damon Civin, and Vikas Chandra. 2018. Federated Learning with Non-IID Data. (2018).

[121] Wang Zhaohang, Xia Geming, Chen Jian, and Yu Chaodong. 2021. Adaptive Asynchronous Federated Learning for Edge Intelligence. *Proceedings - 2021 International Conference on Machine Learning and Intelligent Systems Engineering, MLISE 2021* (2021), 285–289.