Outline of the Paper

> Curriculum: Computational communication theory
>
> Title： ArXiv19_Federated Learning in Mobile Edge Networks
>
> Speaker: Wu Ningyuan

Brief Outline

1. A tutorial on FL implementation
2. Unique Features of the FL and the resulting implementation
3. FL as an enabling technology for mobile edge network optimization

Advantages

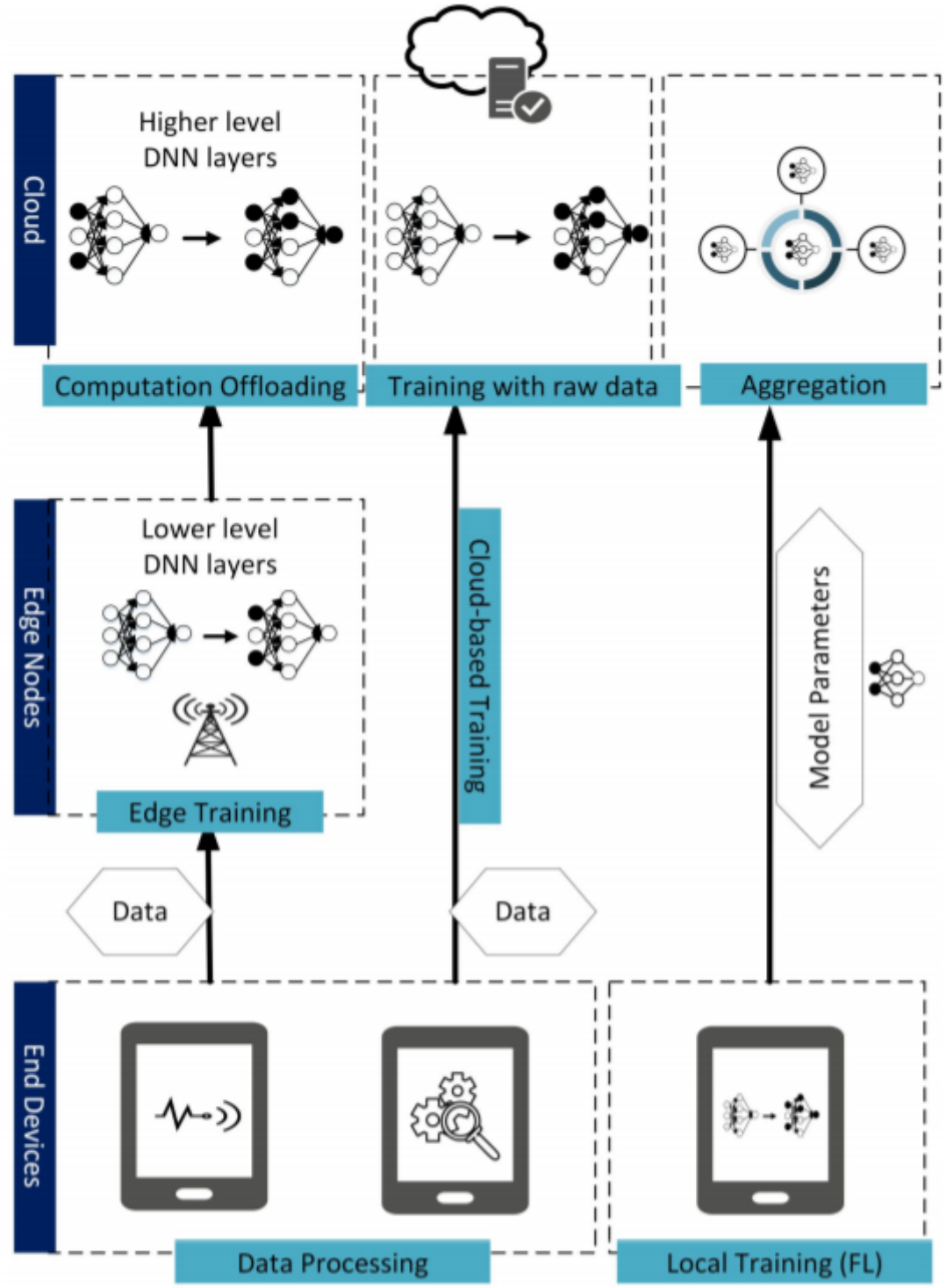

Fig. 1: Edge AI approach brings AI processing closer to where data is produced. In particular, FL allows training on devices where the data is produced.

1. Highly efficient use of network bandwidth
2. Privacy
3. Low latency
4. An enabling technology for mobile edge network optimization

Classification

1. FL at mobile edge network

2. FL for mobile edge network

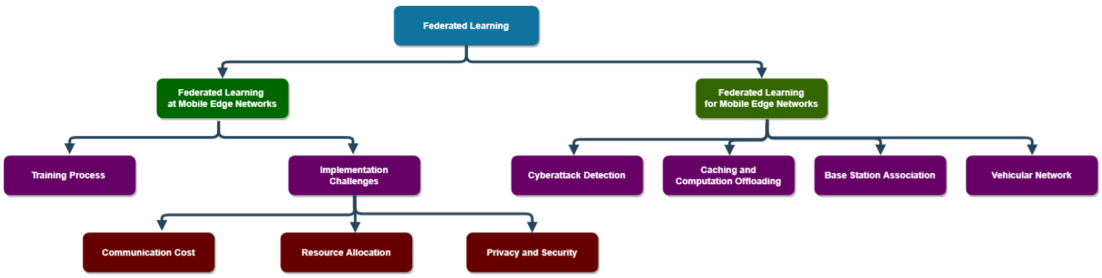

Fig. 2: Classification of related studies to be discussed in this survey.

Principle of FL

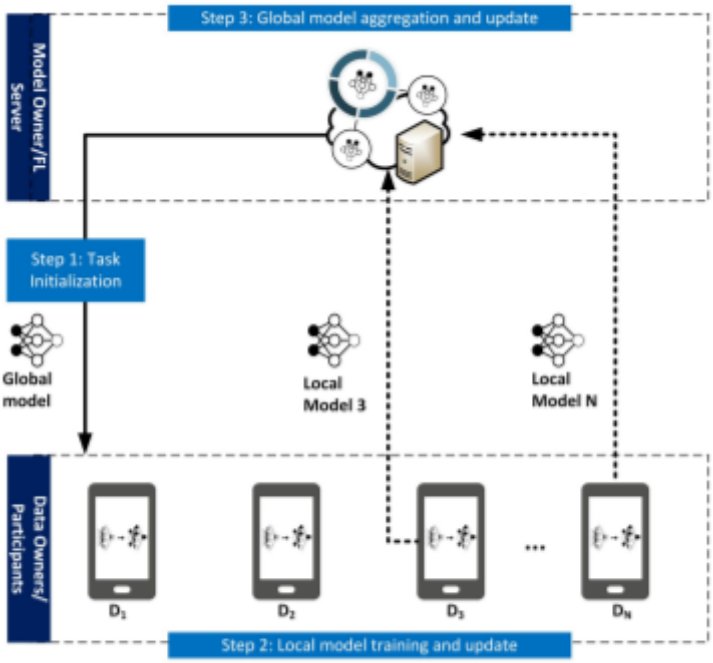

Fig. 4: General FL training process involving N participants.

Step 1 (Task Initialization) :

Server: a. decides the Training Task & Corresponding Data Requirements

b. specifies the hyper-parameters of the global model(learning rate, epoch etc.)

c. broadcasts the global model W_g^0 and task to selected participants

Step 2 (Local model training and update) :

The goal of participant i in iteration t is to find optimal parameters that minimize the loss function $L(W_i^t)$.

$$W_i^{t*} = \underset{W_i^t}{arg\,min}\, L(W_i^t)$$

The updated local model parameters are subsequently to the server (the difference).

Step 3 (Global model aggregation and update) :

Server: aggregates the local models and sends the updated global model parameters W_G^{t+1} to the data owners.

$$L(W_G^t) = \frac{1}{N}\sum_{i=1}^{n} L(W_i^t)$$

Steps 2 and 3 are repeated until the global loss funtion converges or a desirable trianing accuracy is achieved.

Algorithm

Algorithm 1 Federated averaging algorithm [23]

Require: Local minibatch size B, number of participants m per iteration, number of local epochs E, and learning rate η.
Ensure: Global model $\mathbf{w}_G$.

[Participant i]
LocalTraining(i, $\mathbf{w}$):
Split local dataset D_i to minibatches of size B which are included into the set $\mathcal{B}_i$.
for each local epoch j from 1 to E **do**
  for each $b \in \mathcal{B}_i$ **do**
    $\mathbf{w} \leftarrow \mathbf{w} - \eta \Delta L(\mathbf{w}; b)$ (η is the learning rate and ΔL is the gradient of L on b.)
  end for
end for

[Server]
Initialize $\mathbf{w}_G^0$
for each iteration t from 1 to T **do**
  Randomly choose a subset $\mathcal{S}_t$ of m participants from $\mathcal{N}$
  for each partipant $i \in \mathcal{S}_t$ **parallely do**
    $\mathbf{w}_i^{t+1} \leftarrow$ **LocalTraining**(i, $\mathbf{w}_G^t$)
  end for
  $\mathbf{w}_G^t = \frac{1}{\sum_{i\in\mathcal{N}} D_i} \sum_{i=1}^{N} D_i \mathbf{w}_i^t$ (Averaging aggregation)
end for

Statistical Challenges

1. (分布不均匀)Local datasets follow different distributions: i.e. The datasets of participants are non-IID

solution: Private data with public shared data (30% accuracy increased with 5% shared data)

Alternative Solution: ???

2. (分布类型不均匀)Global imbalance

solution: Astraea framework

每个participant首先将自己的数据分布上传至server，然后服从分布在全部分布类型中占比较小的进行数据扩充，然后mediator选择对均匀分布贡献最佳的数据进行拟合。

2. (数据量不均匀)

   solution: MOCHA 算法 (cannot be applied to non-convex DL models)

3. 算法的收敛性

solution: FedProx;

它修改了损耗函数，使其包含一个可调参数用于限制了本地更新对初始模型参数的影响程度。FedProx算法可以自适应调整，例如，当训练损失增加时，可以调整模型更新以减少对当前参数的影响。类似地，[76]中的作者还提出了LoAdaBoost-FedAvg算法来补充上述医学数据共享方法[66]。在LoAdaBoost-FedAvg中，参与者根据本地数据训练模型，并将交叉熵损失与上一轮训练的中值损失进行比较。如果当前交叉熵损失较大，则在全局聚集之前对模型进行再训练，以提高学习效率。仿真结果表明，该算法具有较快的收敛速度。

FL protocols and frameworks

Protocols

Three phases:

1. Selection: the server chooses a subset of connected devices to participate in a training round.
2. Configuration
3. Reporting

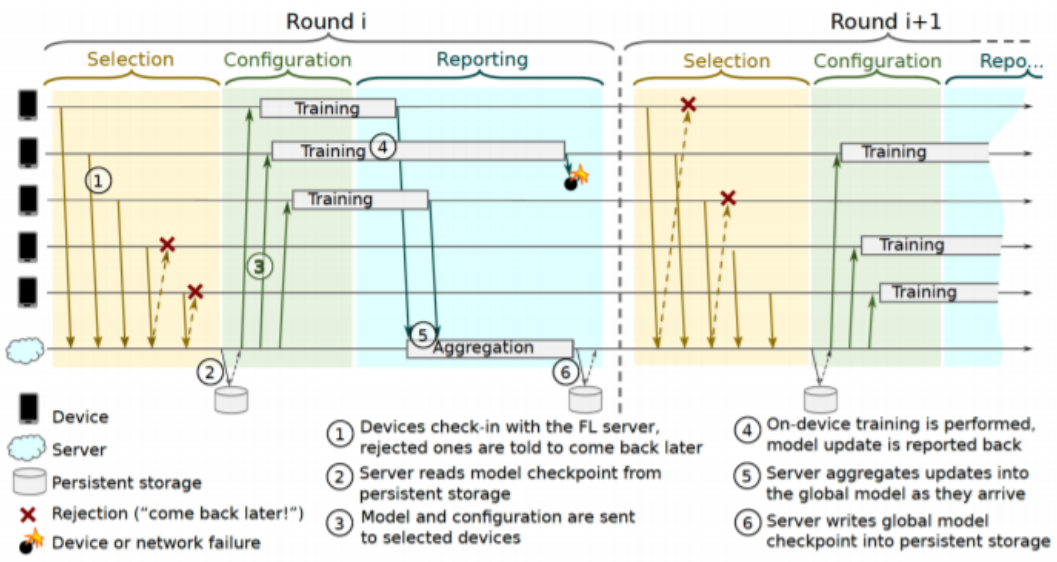

Fig. 5: Federated learning protocol [77].

适应性：

1. Population: Pace Steering(ensure sufficient number of devices)
2. Privacy: Prevent the local updates from being traced and utilized to infer the identity of the fl participant.(Third party server; Noise)

Frameworks

1. Tensorflow Federated (TFF)
2. PySyft (Based on PyTorch)
3. LEAF (An open source framework of datasets which can be used as benchmarks in FL)

Unique Characteristics and issues of FL

1. Slow and unstable communication（上传速度慢且不稳定，设备鱼龙混杂）
2. Heterogeneous devices（各个设备有不同的处理性能以及不同的参与意愿）
3. Privacy and security concerns（可能有恶意参与participants）

以下主要讨论三个问题：

1. **Communication costs**
2. **Resource allocation**
3. **Privacy and allocation**

Communication Cost

如遇到复杂模型（CNN），常常需要训练上百万个参数，导致巨大的传输损耗和训练瓶颈。

一般如下两种情况会导致情况更加糟糕：

1. 参与者网络状况不稳定
2. 上传速度远小于下载速度，导致参数更新的时间将大大延长

解决方案：

总结：

TABLE III: Approaches to communication cost reduction in FL.

Approaches	Ref.	Key Ideas
Edge and End Computation	[23]	More local updates before communication
	[97]	Reference to global model for faster convergence
	[98]	Intermediate edge aggregation before FL server aggregation
Model Compression	[88]	Structured and sketched updates for participant-to-server communication
	[93]	Lossy compression and federated dropout for server-to-participant communication
Importance-based Updating	[94]	eSGD to selectively communicate parameters that reduce training loss
	[90]	CMFL to selectively communicate parameters based on signs of parameters compared to global parameters

1. Edge and End Computation

在end nodes和end devices进行尽可能多的计算以减少传输的轮数从而减少传输带来的时延。

以增加计算量为代价，减少传输的轮数。

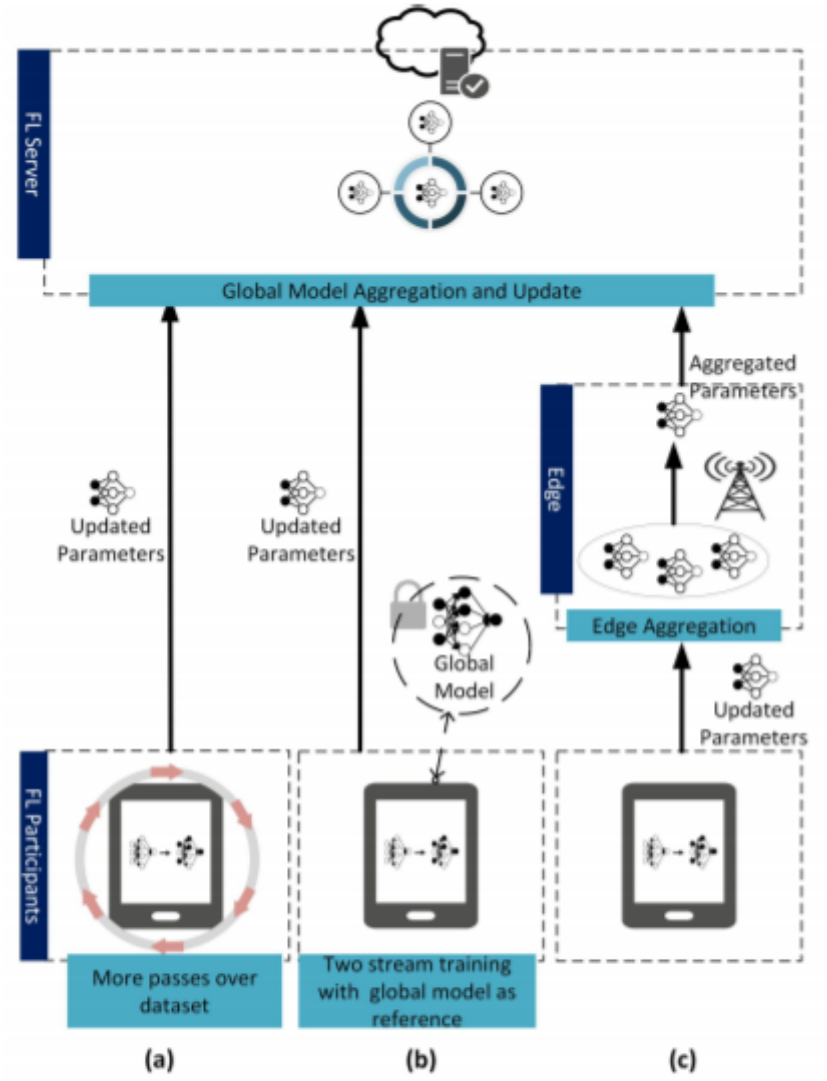

Fig. 6: Approaches to increase computation at edge and end devices include (a) Increased computation at end devices, e.g., more passes over dataset before communication [23] (b) Two-stream training with global model as a reference [97] and (c) Intermediate edge server aggregation [98]

a. FedAvg VS FedSGD

FedSGD : Full-batch size

FedAvg ; Mini-batch size

通过减小每次训练的batch-size来提升计算量使得模型更加准确，减少进行迭代的用户数量

b. Two Stream Model

每一代每个participant都会收到global model，这会给每个participant的训练提供一个参照，通过最小化Maximum Mean Discrepancy（两种模型参数的分布的均值的差值）来使得本地训练的模型达到更快的收敛。

c. Intermediate Edge Server Aggreagation (HighFAVG)

为了减少时延，提出了先通过edge server先对participant的模型进行一次小的aggregation，edge server再将集中化的参数传给中心服务器。

2. Model Compression

参考网址：https://blog.csdn.net/Jinlong_Xu/article/details/79096428

2015年 Han发表的Deep Compression：模型压缩方法的综述性文章

在文章中，通过裁剪、权值共享、量化、编码的方式用于模型压缩，取得了非常好的效果。

1989年 Lecun: Optimal Brain Damage可以将网络中不重要的参数剔除，达到压缩尺寸的作用

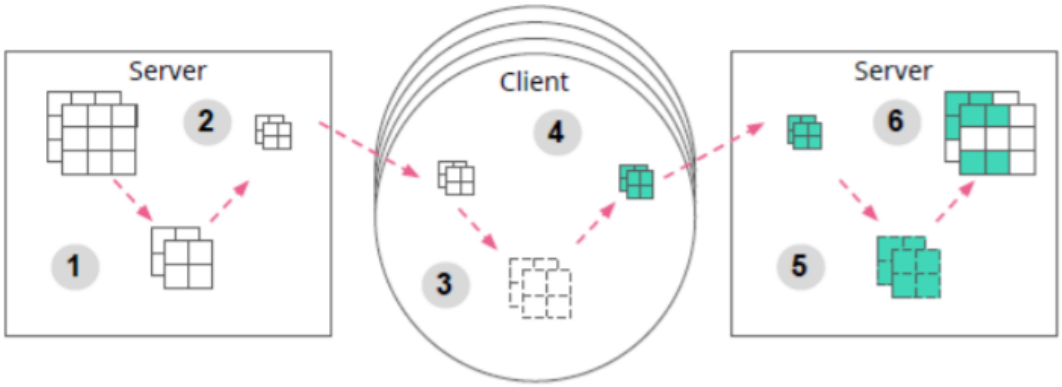

Fig. 7: The compression techniques considered are summarized above by the diagram from authors in [93]. (i) Federated dropout to reduce size of model (ii) Lossy compression of model (iii) Decompression for training (iv) Compression of participant updates (v) Decompression (vi) Global aggregation

a. Federated Dropout

b. 损失函数压缩

c. 模型和损失函数解压缩并进行训练

d. 训练后对updates进行压缩上传

e. 对updates进行解压缩操作

f. 参数集成计算，形成下一代的模型以及损失函数

> 模型压缩主要分为以下三个方向：

1. 更加精细模型的设计
2. 模型裁剪（寻找有效的评判手段，对不重要的connnection或者filter进行裁剪减少模型的冗余）
3. 核的稀疏化（形成稀疏矩阵）
4. 量化
5. Low-rank分解
6. 迁移学习

3. Importance-based Updating

只有相关或者重要的更新才会被上传，减少传输量。

1）eSGD（edge Stochastic Gradient Descent）通过对DNN的观察：大多数的权重值都接近为0且稀疏分布的，因此只需要上传重要的权重信息即可。

数据标注跟随

2）CMFL（Communication-Mitigated Federated Learning）：只有跟global update有较强相关性的更新才会被上传（因为当次的global training暂时没有，所以只能和上次的global Training 进行比较）

Resource Allocation

总结：

TABLE IV: Approaches to resource allocation in FL.

Issue	Ref.	Approach
Participant Selection	[78]	FedCS to select participants based on computation capabilities
	[114]	Hybrid-FL to select participants for IID data collection
	[115]	DRL to determine resource consumption by participants
	[119]	Fair resource allocation
	[121], [123]	Participant selection based on distance threshold to increase SNR in BAA
	[124]	Participant selection to keep signal distortion low
Adaptive Aggregation	[129], [111]	Asynchronous FL where model aggregation occurs once local updates are received by FL server
	[65]	Adaptive global aggregation frequency
Incentive Mechanism	[130]	Stackelberg game and relay network to support model update transfer
	[132]	Stackelberg game to incentivize computation resource usage in FL training
	[133], [62]	Contract theory, reputation mechanisms and blockchain

不同的设备具有不同的数据量、计算能力、energy states和参与意愿

通过优化最大化训练过程的效率，且降低传输时延。

主要通过以下三种方法进行优化：

1. Participant Selection
2. Adaptive Aggregation
3. Incentive Machanism

1. Participant Selection（木桶效应）

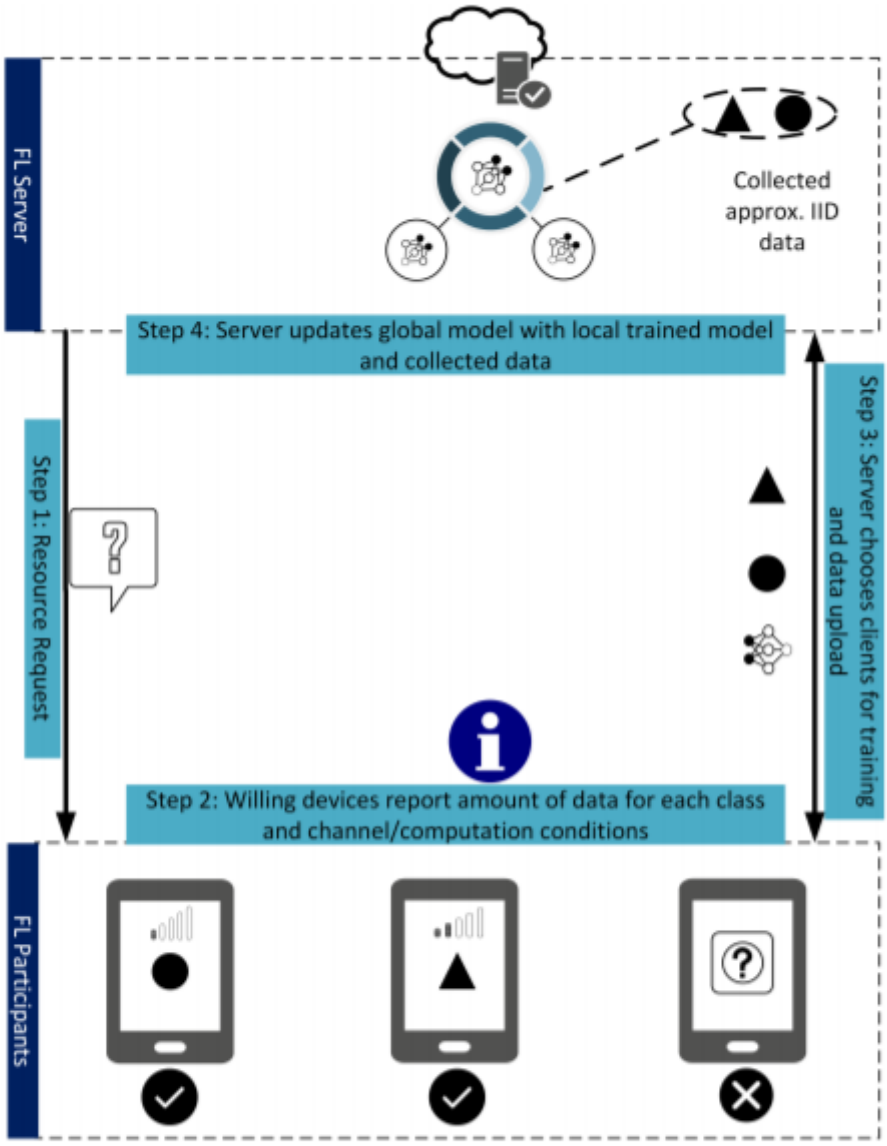

Fig. 8: Participant selection under the FedCS and Hybrid-FL protocol.

a.（FedCS）首先收集信道状况（网络状况）以及计算能力等信息，根据收集到的信息选择在确定期限内能够完成计算任务的participant，从而保证准确率以及效率。

b.（Hybrid-FL）在收集数据阶段，先在用户允许的情况下上传用户的数据，得到他们的数据分布，然后在用户选择的阶段选择与已上传数据独立同分布的participant。仿真显示1%的用户上传data会导致分类的效率得到大幅度的提升。（需要有一个较好的奖励机制）

c.（Mobile Crowd Machine Learning，MCML）网络状况、数据量、CPU使用情况：定义一个关于已经收集到的数据、消耗的能量以及传输时延的函数，然后用Double Deep Q-Network进行解决。

d.（q-Fair FL）模型公平性：如果仅仅选取计算能力较高的设备进行模型训练，那么训练好的模型会在计算能力较强的模型上训练较好但是在计算能力较弱的设备上表现较差。在qFFL算法中，损耗函数将较高的权重分配给了损耗较高的设备，从而使得模型更加公平。

e.（BAA，宽带模拟汇聚）使用了空中计算（over the air computation）的概念，使得在BAA中实现了整个频带的反复利用。它比OFDMA少了Aggregation这一环节。

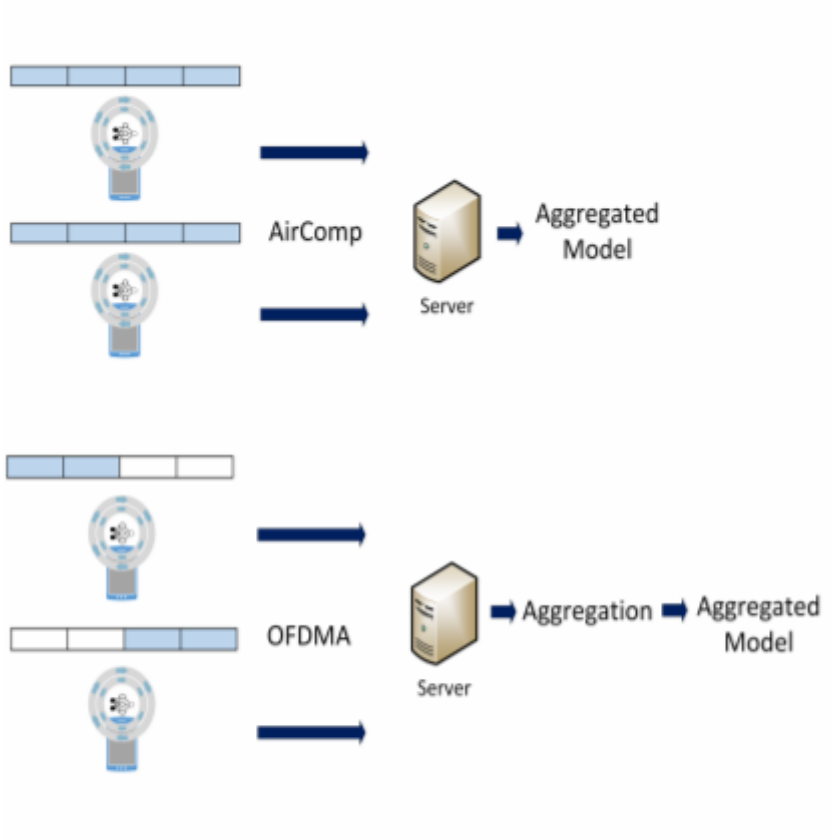

Fig. 9: A comparison [121] between (a) BAA by over-the-air computation which reuses bandwidth (above) and (b) OFDMA (below).

2. Adaptive Aggregation（自适应聚合）

非同步联合学习

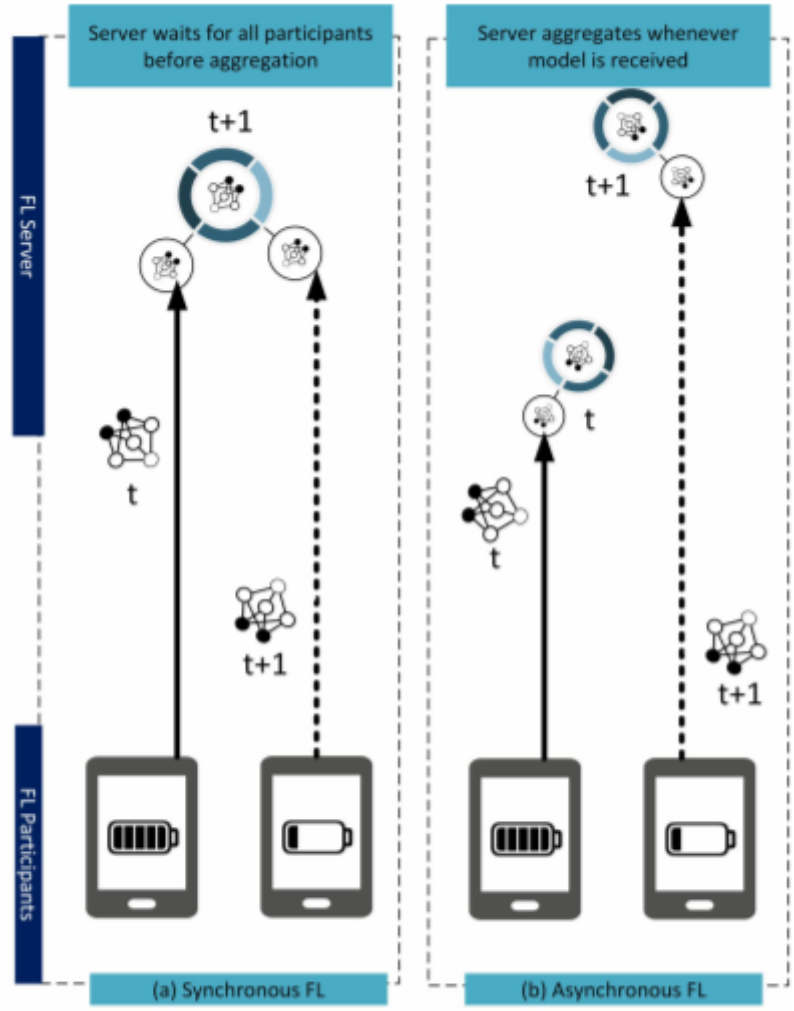

Fig. 10: A comparison between (a) synchronous and (b) asynchronous FL.

3. Incentive Mechanism（奖励机制）

130：将participant看作是厂商，将model_owner看作是消费者。

131：斯塔克尔伯格模型：认为产量的决定依据以下次序：领导性厂商决定一个产量，然后跟随着厂商可以观察到这个产量，然后根据领导性厂商的产量来决定他自己的产量。需要注意的是，领导性厂商在决定自己的产量的时候，充分了解跟随厂商会如何行动——这意味着领导性厂商可以知道跟随厂商的反应函数。因此，领导性厂商自然会预期到自己决定的产量对跟随厂商的影响。正是在考虑到这种影响的情况下，领导性厂商所决定的产量将是一个以跟随厂商的反应函数为约束的利润最大化产量。在斯塔克尔伯格模型中，领导性厂商的决策不再需要自己的反应函数（设备相对较少）。

figure11：资源约束未知的参与者只有选择满足自身约束的捆绑条件，才能发挥最大的效用。换句话讲，才能挣到更多的钱。首先由data owner来宣布需要高性能计算设备任务的酬劳以及需要低性能计算设备任务的酬劳。低等的设备没有能力计算high type的任务，因此会自动选择low type Contract，根据趋利性高性能的计算设备会选择high type的任务，如此以来，可以实现整个奖励机制的优化。

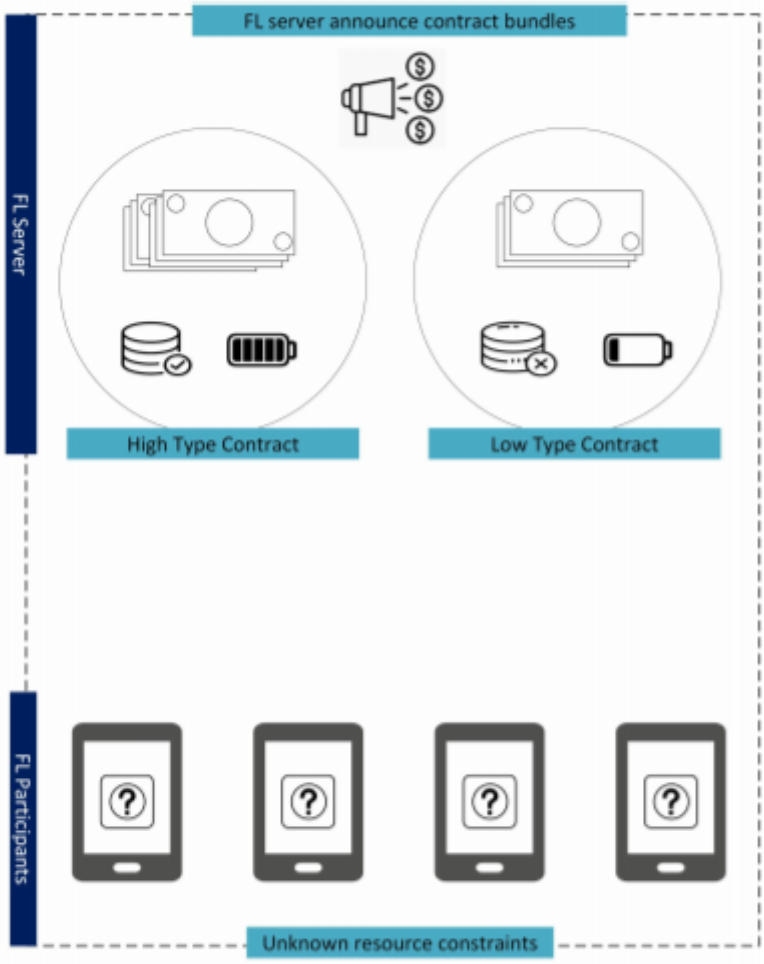

Fig. 11: Participants with unknown resource constraints maximize their utility only if they choose the bundle that best reflects their constraints.

Privacy and Security Issues

Privacy Issues:

仍然可以通过共享模型来推测出敏感信息

问题及解决方案：

1）Information exploiting attacks in machine learning

140：作者开发了一种模型反转算法，该算法在利用基于决策树或面部识别训练模型的信息中非常有效。这种方法的思想是将目标特征向量与每个可能的值进行比较，然后得出作为正确值的加权概率估计。实验结果表明，通过使用这种技术，对手可以从其标签中以很高的精度重建受害者面部的图像

2）Differential privacy-based protection solutions for FL participants

核心思想：给训练好的参数加噪声

3）Collaborative training solutions

核心思想：自主选择需要更新的参数，使得敏感信息无法重建

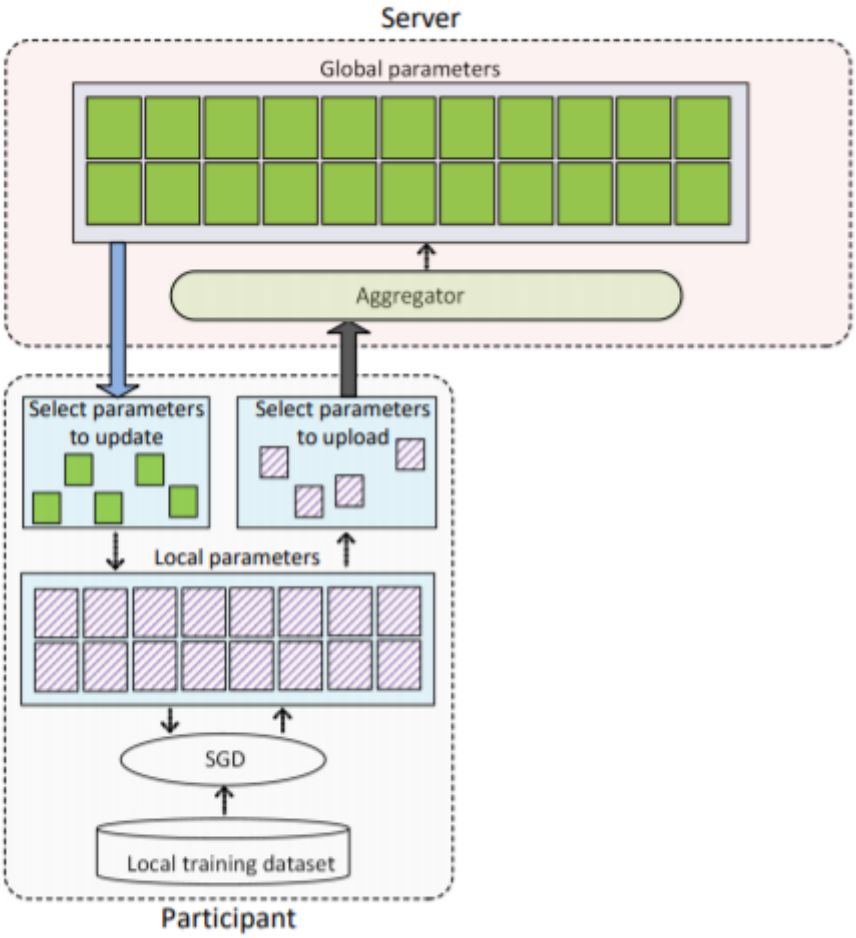

Fig. 12: Selective parameters sharing model.

GAN是一类ML技术，它使用两个神经网络相互竞争以训练数据，即生成器网络和鉴别器网络。生成器网络尝试通过向真实数据中添加一些"噪声"来生成伪造数据。然后，将生成的伪造数据传递到鉴别器网络以进行分类。在训练过程之后，GAN可以生成具有与训练数据集相同的统计数据的新数据。受此想法启发，[149]中的作者开发了一种强大的攻击，该攻击使恶意参与者甚至可以从受害者那里推断出敏感信息，即使受害人只有一部分共享参数，如图13所示。

4）Encryption-based solutions

核心思想：对参数进行加密处理

Security Issues:

恶意的参与者来试图通过自己的模型更新数据对于总体模型进行攻击。

1）Data Poisoning Attacks

通过设计好的图片以及标签的关系对网络进行攻击

FoolsGold: 通过上传的梯度来确定谁是攻击者谁为正常用户

2）Model Poisoning Attacks

1.无需数据，直接通过上传错误权重对网络进行攻击

2.直接对整体网络进行攻击

解决方案：a. 服务器确定是否上传的参数对于模型的表现是否有积极的贡献，通过几轮更新来确定是否为攻击者。 b. 通过鉴别其上传的模型是否与其他人相类似，如差别过大，则可视为攻击者。

3）Free-riding Attacks

对训练进程几乎没有贡献但是却可以从训练好的整个模型中受益

解决方案：

[160]中的作者介绍了一种称为BlockFL的基于区块链的FL架构，其中利用区块链技术交换并验证了参与者的本地学习模型更新。特别地，每个参与者训练并发送训练后的全局模型并将其发送到其在区块链网络中的关联矿工，然后获得与训练后的数据样本数量成正比的奖励，如图14所示。通过这种方式，该框架无法不仅会阻止参与者搭便车，而且还会激励所有参与者为学习过程做出贡献。

Applications of FL For Mobile Edge Computing

1. Cyberattack Detection
2. Edge Caching and Computation Offloading
3. Base Station Association
4. Vehicular Networks

1. Cyberattack Detection

   在该模型中，每个边缘节点作为谁拥有的入侵检测一组数据的参与者。为了提高检测攻击的准确性，培训领域的全球模式之后，每个参与者都将其训练模型发送到服务器FL。服务器将聚集来自参与者的所有参数，并发送更新的全球模式返回到所有参与者如图15所示。

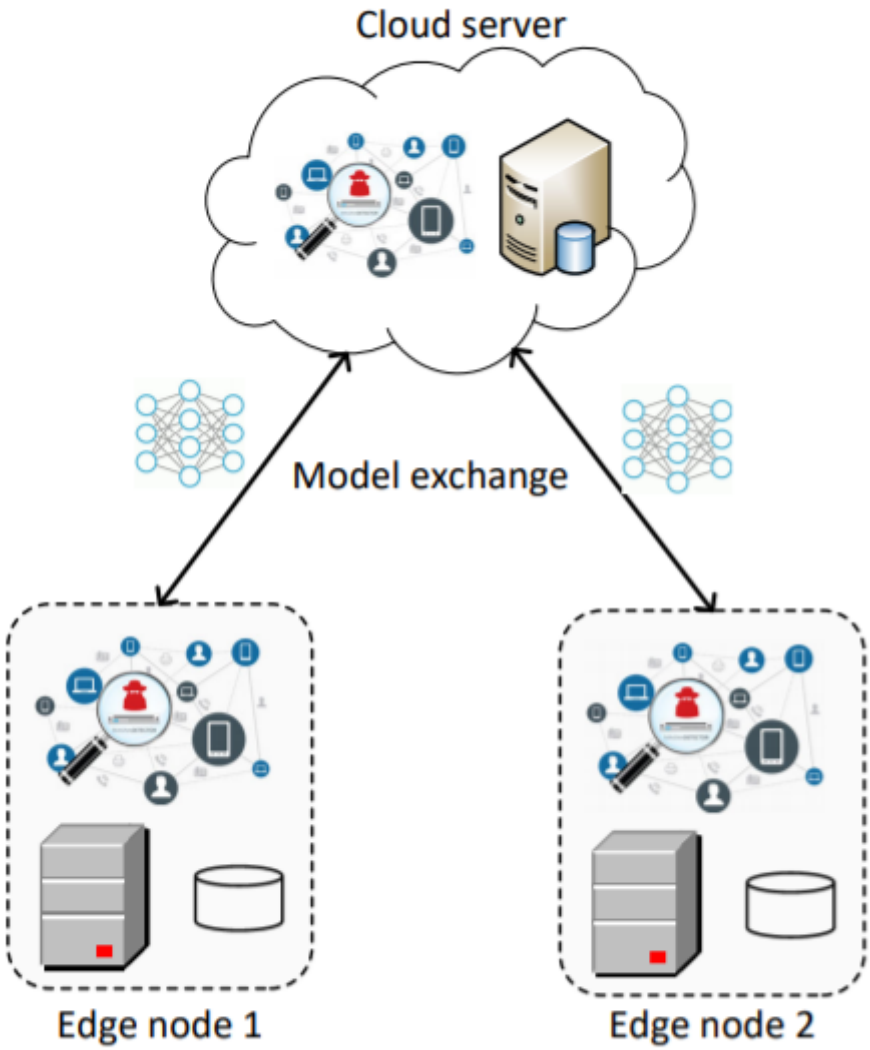

Fig. 15: FL-based attack detection architecture for IoT edge networks.

2. Edge Caching and Computation Offloading

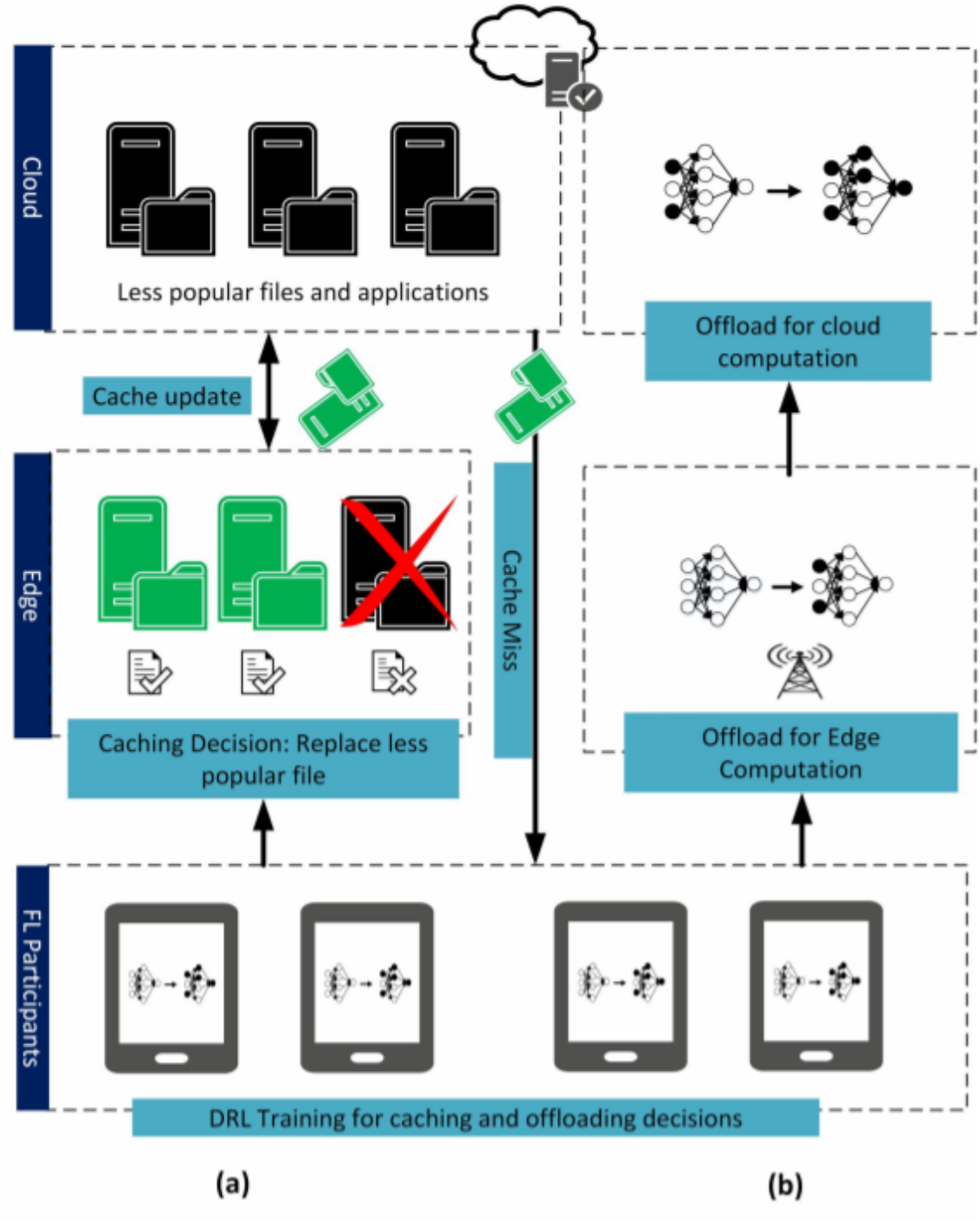

Fig. 16: FL-based (a) caching and (b) computation offloading.

MEC系统由基站覆盖的一组用户设备（UE）组成。对于缓存，可以通过模型的训练来决定缓存还是不缓存下载的文件，以及应该在缓存时替换哪个本地文件。对于计算卸载，UE可以选择通过无线信道将计算任务卸载到边缘节点，也可以选择本地执行。该缓存和卸载决策过程如图16所示。MEC系统的状态包括无线网络状况，UE能耗和任务排队状态，而奖励功能定义为UE的体验质量（QoE）。

3. Base Station Association

使得虚拟现实应用程序用户的在线中断最小化。

为了得出用户位置和方向的预测，基站必须依赖用户的历史信息。但是，存储在每个基站的历史信息仅从每个用户收集部分数据，由此每个基站首先使用其部分数据来训练局部模型。然后，将本地模型进行汇总，以形成能够进行概括的全局模型，即全面预测用户的移动性和方向。从而使得用户的在线中断最小化，提升用户的使用体验。

4. Vehicle Networks（智能交通（路况预测），充电站配置分布）

该方法需要队列状态信息（QSI）和车辆之间的数据交换的足够样本。因此，提出了一种FL方法，其中车辆用户（VUE）使用本地保存的数据训练学习模型，并将仅更新的模型参数上载到路边单元（RSU）。然后，RSU对模型参数求平均值，并将更新的全局模型返回给VUE。在同步方法中，所有VUE在预定时间间隔的末尾上传其模型。