移动边缘异构资源联合学习的客户端选择

一分钟概括

问题

联合学习（FL）: 用于ML的MEC框架(即利用分布式计算训练HPM)。
传统FL协议：要求随机C从S下载可训练的模型，更新自己的数据，更新的模型上传到S
S聚合多个C更新以改善模型。此协议中的C可以不泄露自己的私有数据，但是当某些C的**计算资源有限或在无线信道条件较差**时，导致整个训练过程效率低下。

解决方案

设计了FedCS协议缓解该问题，根据本地资源主动管理C，有效地进行联合学习。

仿真

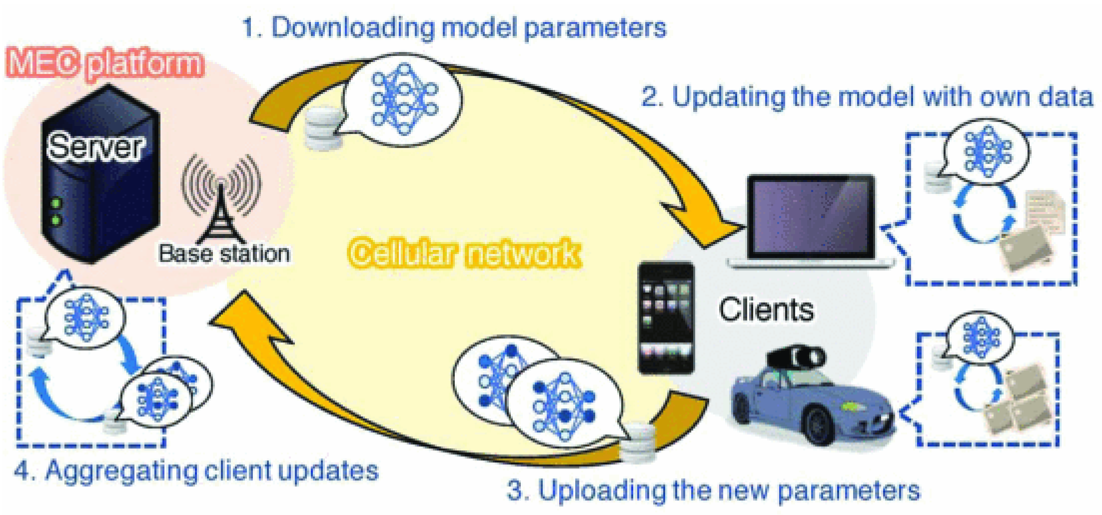
将原始FL和FedCS比较，指标分别是

- 达到要求精度的**时间**
- 最终的**准确精度**

结果表明，与原始FL协议相比，FedCS能够在短得多的时间内完成其训练过程

为什么使用FL

大数据时代，更要尊重用户隐私



FL协议

> Random C
> 1）从某个服务器下载可训练模型的参数
> 2）用自己的数据更新模型
> 3）在询问时将**新模型**参数上传到服务器
> 4）服务器聚合多个客户端更新以进一步改进模型。

假定C安装在可以跑ML模型的机器上的

1: `Initialization`: The server first initializes a global model randomly or by pretraining with public data.
2: `Client Selection`: The server randomly selects $\lceil K \times C \rceil$ clients.
3: `Distribution`: The server distributes the parameters of the global model to the selected clients.
4: `Update and Upload`: Each selected client updates the global model using their data and uploads the updated model parameters to the server.
5: `Aggregation`: The server averages the updated parameters and replaces the global model by the averaged model.
6: All steps but `Initialization` are iterated until the global model achieves a desired performance.

传统FL的缺陷

客户端的计算资源有限→模型更新耗时。

客户端无线信道条件较差→更长的更新时间。

→S生成最后的模型耗时将更长

FedCS优势

FedCS为C设置了一定的Deadline,C用传统FL协议下载，更新和上传ML模型。
MEC选择客户端，以便服务器可以在有限的时间内聚合尽可能多的客户端更新，提高训练效率。
FedCS优势就是：考虑C的CR，选择某些C参与训练，同时限制C的完成时间,

FedCS: Federated Learning with Client Selection

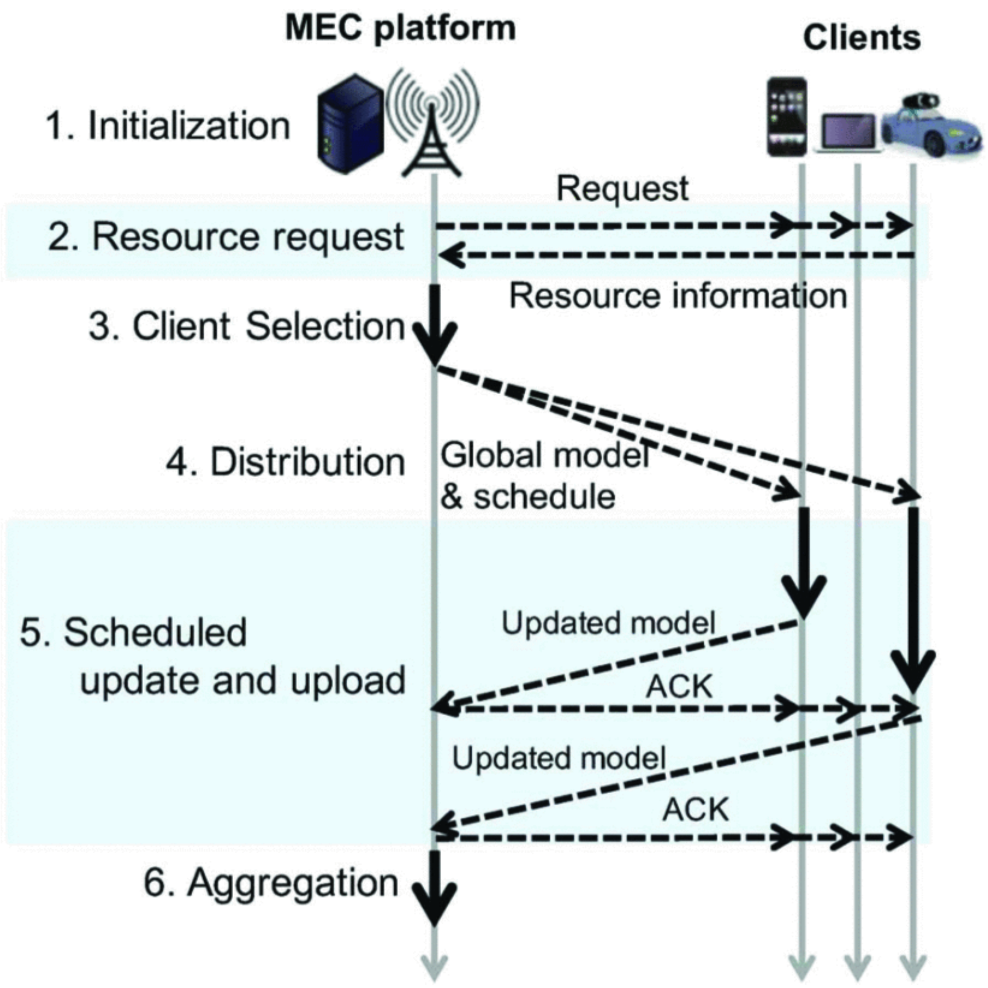
假设前提

- 网络稳定不拥挤
- 忽略丢包，每个客户端信道状态和吞吐量稳定

协议细节

1: `Initialization` in Protocol 1.
2: `Resource Request`: The MEC operator asks $\lceil K \times C \rceil$ random clients to participate in the current training task. Clients who receive the request notify the operator of their resource information.
3: `Client Selection`: Using the information, the MEC operator determines which of the clients go to the subsequent steps to complete the steps within a certain deadline.
4: `Distribution`: The server distributes the parameters of the global model to the selected clients.
5: `Scheduled Update and Upload`: The clients update global models and upload the new parameters using the RBs allocated by the MEC operator.
6: `Aggregation` in Protocol 1.
7: All steps but `Initialization` are iterated for multiple rounds until the global model achieves a desired performance or the final deadline arrives.

```
初始化
While 全局模型足够准确 or Deadline截止：
    MEC随机请求[K×C]个Clients，Clients通知MEC operator自己的CR。
    根据CR,MEC operator决定哪些Clients参与训练过程，并规定训练结束Deadline
    MEC Sever将全局模型参数分发给Clients
    Clients更新模型，并上传新的参数
    MEC Sever整合所有模型
```



客户端选择算法

$$\Theta_i := \begin{cases} 0 & \text{if } i = 0; \\ T_i^{\text{UD}} + T_i^{\text{UL}} & \text{otherwise,} \end{cases} \tag{1}$$

$$T_i^{\text{UD}} = \sum_{j=1}^{i} \max\left\{0, t_{k_j}^{\text{UD}} - \Theta_{j-1}\right\}, \tag{2}$$

$$T_i^{\text{UL}} = \sum_{j=1}^{i} t_{k_j}^{\text{UL}}. \tag{3}$$

客户端选择==S最大化问题：

$$\begin{aligned} &\max_{\mathbb{S}} |\mathbb{S}| \\ &\text{s.t.} T_{\text{round}} \geq T_{\text{cs}} + T_{\mathbb{S}}^{\text{d}} + \Theta_{|\mathbb{S}|} + T_{\text{agg}}. \end{aligned} \tag{4}$$

解决最大化问题（4）并非易事，因为它需要复杂的组合优化，其中S中元素的顺序会影响Θ| S |。 为此，我们针对具有背包约束的最大化问题提出了一种**基于贪婪算法的启发式算法[14]**。 如算法3所示，我们迭代地将消耗最少时间用于模型上载和更新（步骤3、4和9）的客户端添加到S，直到经过的时间t达到截止期限Tround（步骤5、6、7和8）

Require: Index set of randomly selected clients $\mathbb{K}'$
1: **Initialization** $\mathbb{S} \leftarrow \{\}$, $T_{\mathbb{S}=\emptyset}^{\text{d}} \leftarrow 0$, $\Theta \leftarrow 0$
2: **while** $|\mathbb{K}'| > 0$ **do**
3: $\quad x \leftarrow \arg\max_{k \in \mathbb{K}'} \frac{1}{T_{\mathbb{S}\cup k}^{\text{d}} - T_{\mathbb{S}}^{\text{d}} + t_k^{\text{UL}} + \max\{0, t_k^{\text{UD}} - \Theta\}}$
4: $\quad$ remove x from $\mathbb{K}'$
5: $\quad \Theta' \leftarrow \Theta + t_x^{\text{UL}} + \max\{0, t_x^{\text{UD}} - \Theta\}$
6: $\quad t \leftarrow T_{\text{cs}} + T_{\mathbb{S}\cup x}^{\text{d}} + \Theta' + T_{\text{agg}}$
7: $\quad$ **if** $t < T_{\text{round}}$ **then**
8: $\quad\quad \Theta \leftarrow \Theta'$
9: $\quad\quad$ add x to $\mathbb{S}$
10: $\quad$ **end if**
11: **end while**
12: **return** $\mathbb{S}$

T_{round}的选择：算法3中的重要参数是T_{round}。 如果我们将T_{round}设置大，我们预计每轮将涉及更多客户端(即，更大的S集)。 然而，这同时减少了在最终截止日期T_{final}之前可能的更新聚合数量。 我们的实验评估显示了T_{round}的不同选择如何影响训练模型的最终性能

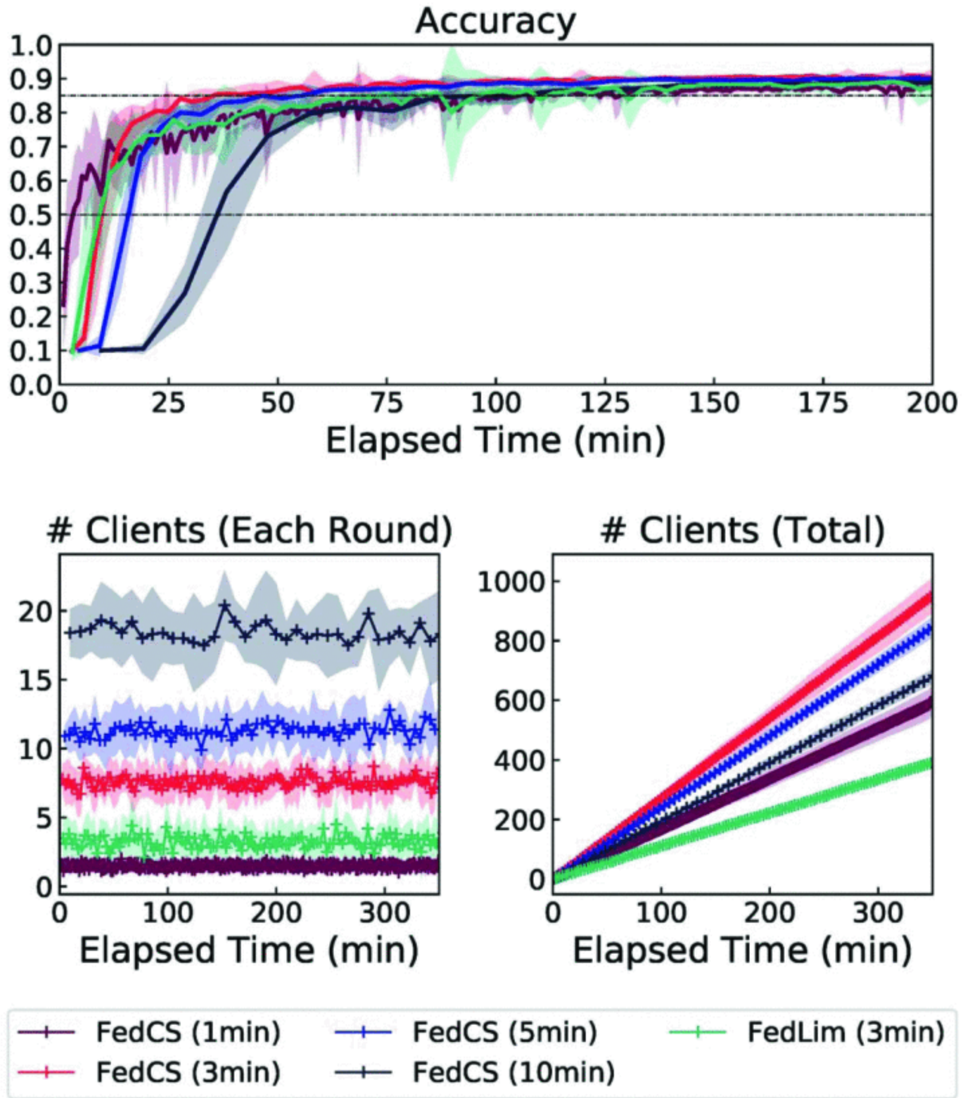
性能评估

一个MEC 1000个客户端，服务半径2KM
t_x^{UL}=D/1.4Mbit/s, D是数据集大小，可变。忽略T_{cs}，T_{agg}，Deadline=360min
ML任务：CIFAR-10，MNIST
对于这两个任务，将训练数据集分配给K = 1000个客户，首先，我们随机确定每个客户拥有的图像数据的数量，范围为100到1000。

每个客户端从整个训练数据集中随机采样指定数量的图像；

Method	CIFAR-10		
	ToA@0.5	ToA@0.75	Accuracy
`FedLim` ($T_{\text{round}} = 3$ min)	38.1	209.2	0.77
FedCS			
$T_{\text{round}} = 3$ min ($r = 0\%$)	**25.8**	**132.7**	**0.79**
$T_{\text{round}} = 3$ min ($r = 10\%$)	**27.9**	**138.1**	**0.78**
$T_{\text{round}} = 3$ min ($r = 20\%$)	**31.1**	**178.3**	**0.78**
Method	**Fashion-MNIST**		
	ToA@0.5	ToA@0.85	Accuracy
`FedLim` ($T_{\text{round}} = 3$ min)	10.4	66.8	0.90
FedCS			
$T_{\text{round}} = 3$ min ($r = 0\%$)	**10.6**	**33.5**	**0.91**
$T_{\text{round}} = 3$ min ($r = 10\%$)	**11.3**	**32.1**	**0.92**
$T_{\text{round}} = 3$ min ($r = 20\%$)	**12.7**	**37.0**	**0.91**

FedCS	CIFAR-10		
	ToA@0.5	ToA@0.75	Accuracy
$T_{\text{round}} = 1$ min ($r = 0\%$)	NaN	NaN	0.50
$T_{\text{round}} = 3$ min ($r = 0\%$)	25.8	132.7	0.79
$T_{\text{round}} = 5$ min ($r = 0\%$)	41.0	166.6	0.79
$T_{\text{round}} = 10$ min ($r = 0\%$)	75.7	281.7	0.76
FedCS	**Fashion-MNIST**		
	ToA@0.5	ToA@0.85	Accuracy
$T_{\text{round}} = 1$ min ($r = 0\%$)	3.0	73.7	0.89
$T_{\text{round}} = 3$ min ($r = 0\%$)	10.6	33.5	0.91
$T_{\text{round}} = 5$ min ($r = 0\%$)	18.1	48.8	0.92
$T_{\text{round}} = 10$ min ($r = 0\%$)	42.0	93.3	0.91


Accuracy
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
0
25
50
75
100
125
150
175
200
Elapsed Time (min)
Clients (Each Round)
20
15
10
5
0
0
100
200
300
Elapsed Time (min)
Clients (Total)
1000
800
600
400
200
0
0
100
200
300
Elapsed Time (min)
FedCS (1min)
FedCS (3min)
FedCS (5min)
FedCS (10min)
FedLim (3min)