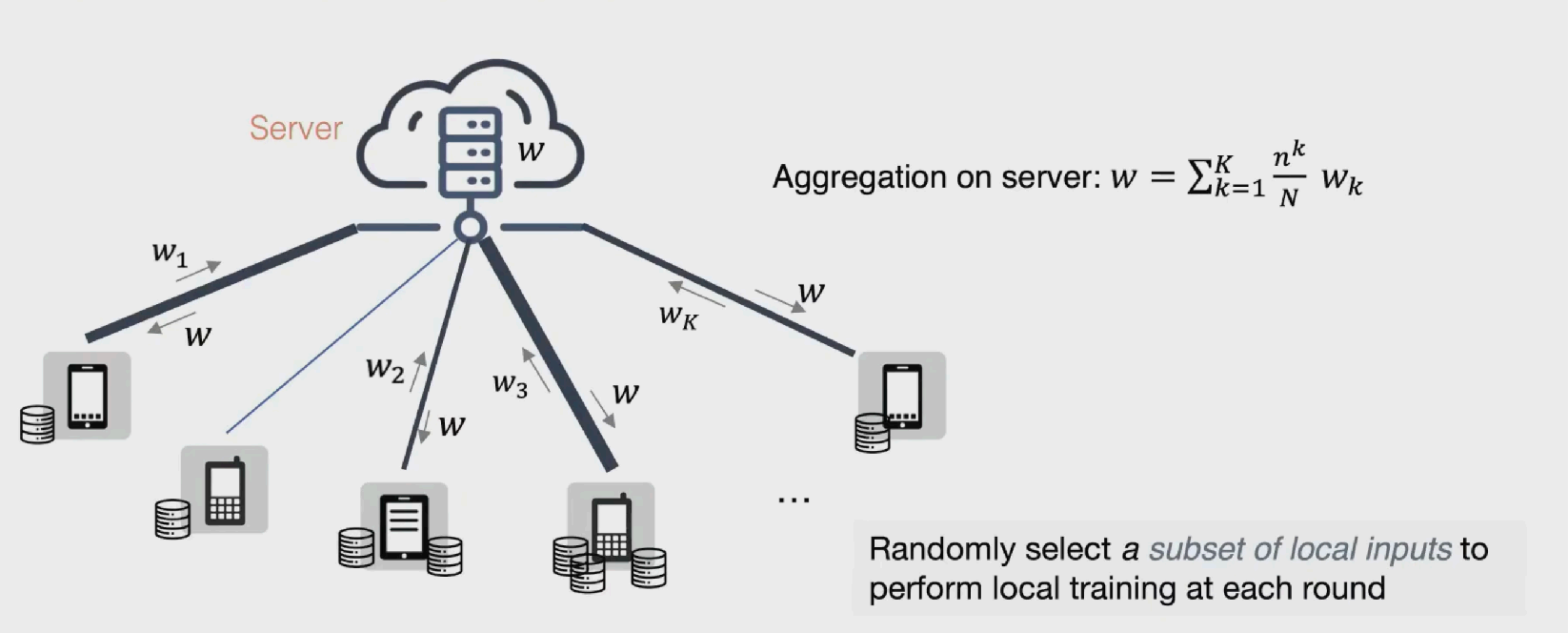
IntelliSys Lab

FedAT:
A High-Performance and Communication-Efficient Federated Learning System with Asynchronous Tiers

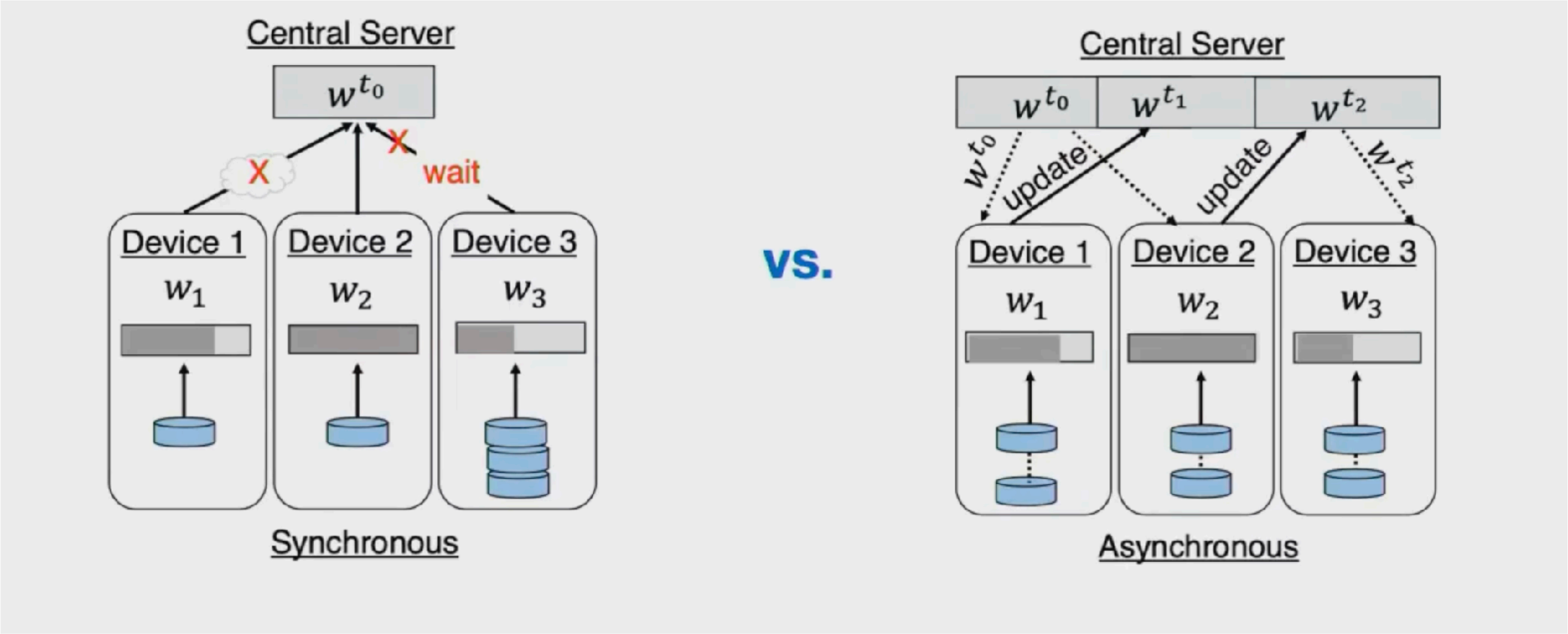
Zheng Chai et al.
George Mason University, Emory University
SC 21

Louisiana State University

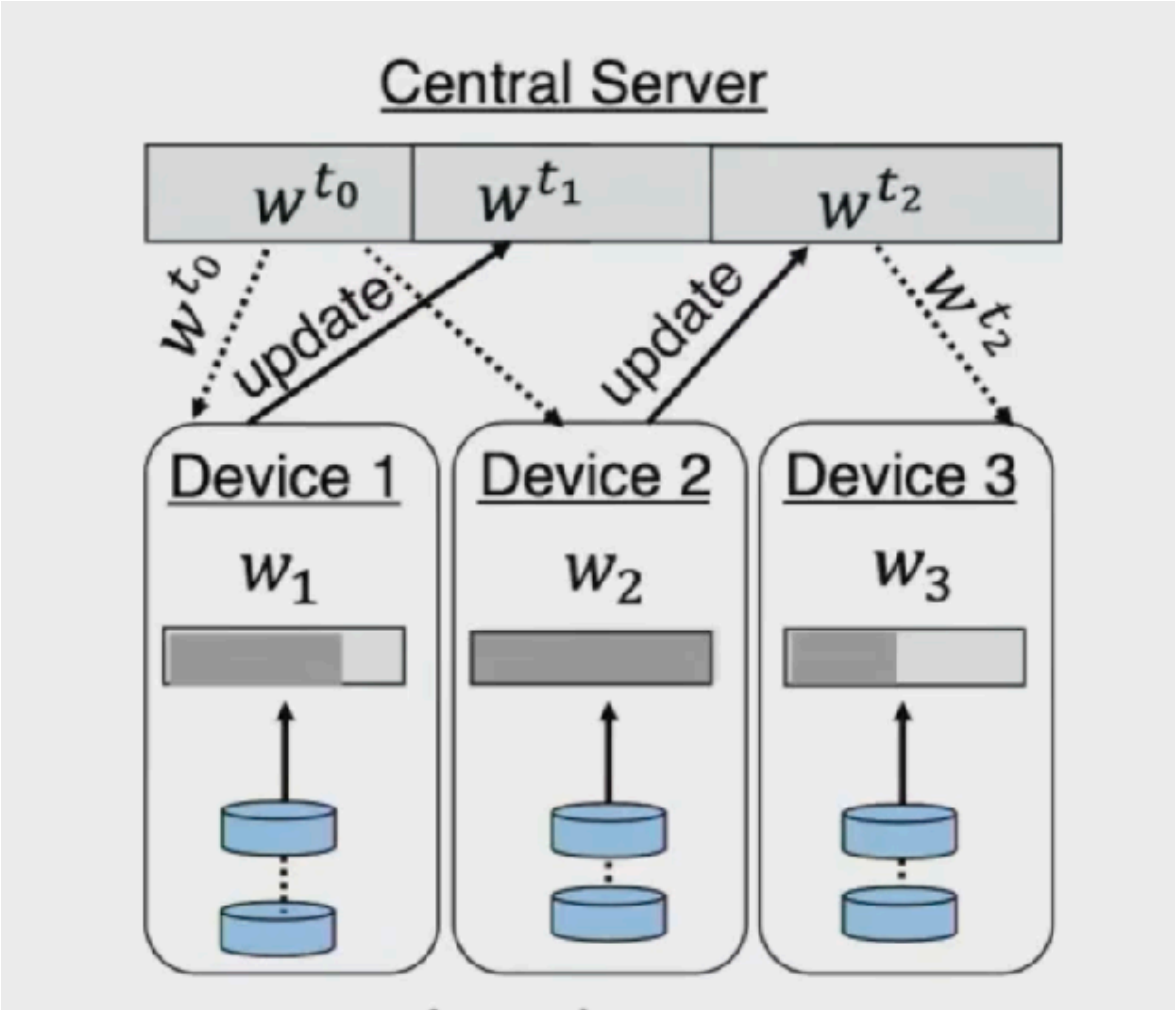
Synchronous Federated Learning



Asynchronous Federated Learning



Asynchronous Federated Learning



Asynchronous Federated Optimization

Cong Xie CX2@ILLINOIS.EDU
Oluwasanmi Koyejo SANMI@ILLINOIS.EDU
Indranil Gupta INDY@ILLINOIS.EDU
Department of Computer Science, University of Illinois Urbana-Champaign

- The server needs to communicate with each local device at each round —communication bottleneck (FedAsync)

- The global model may bias to devices which communicate frequently

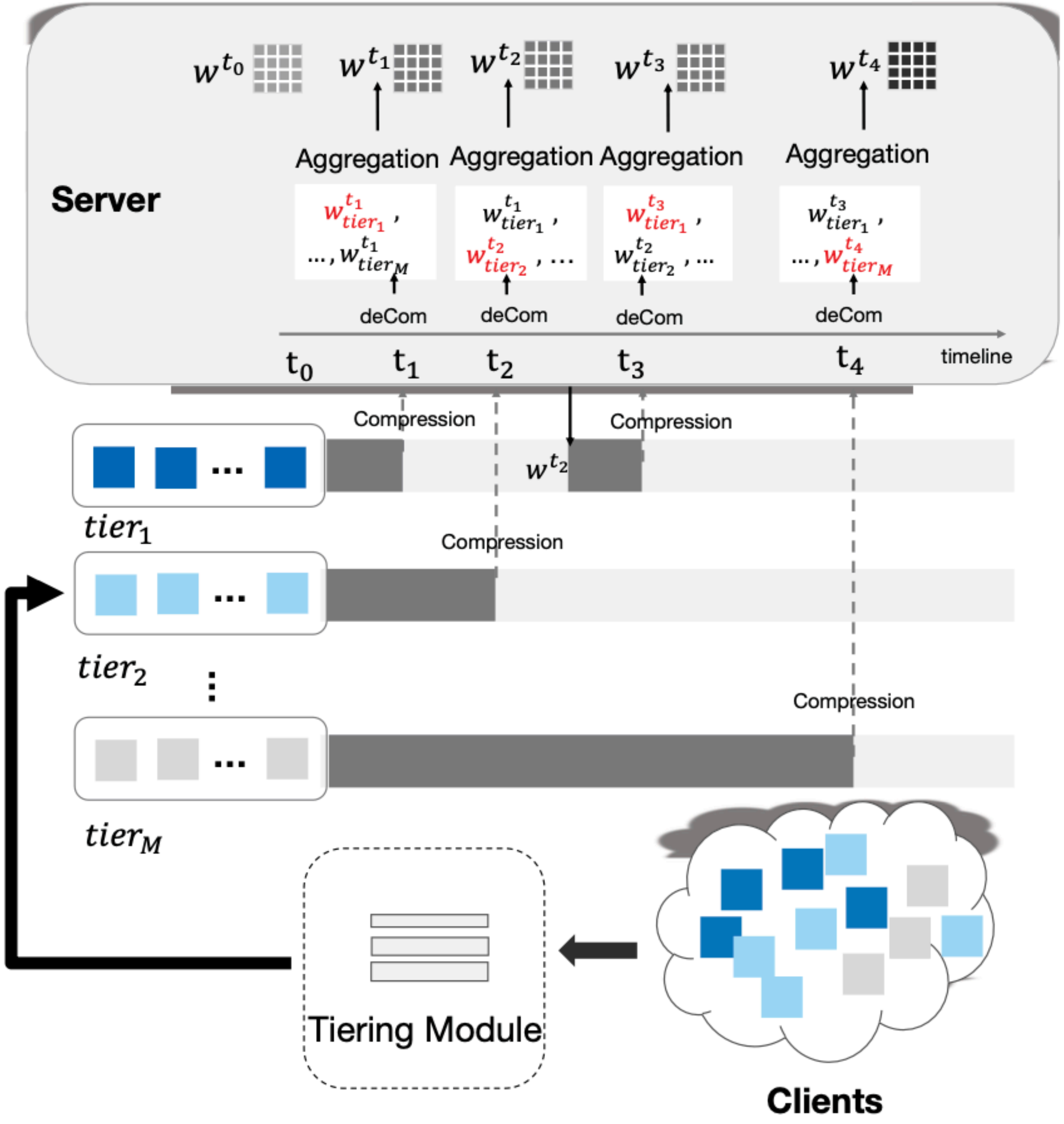
FedAT

A novel communication-efficient FL approach that combines synchronous and asynchronous FL training using a tiering mechanism.

Challenges:

- Stragglers
- Communication bottleneck

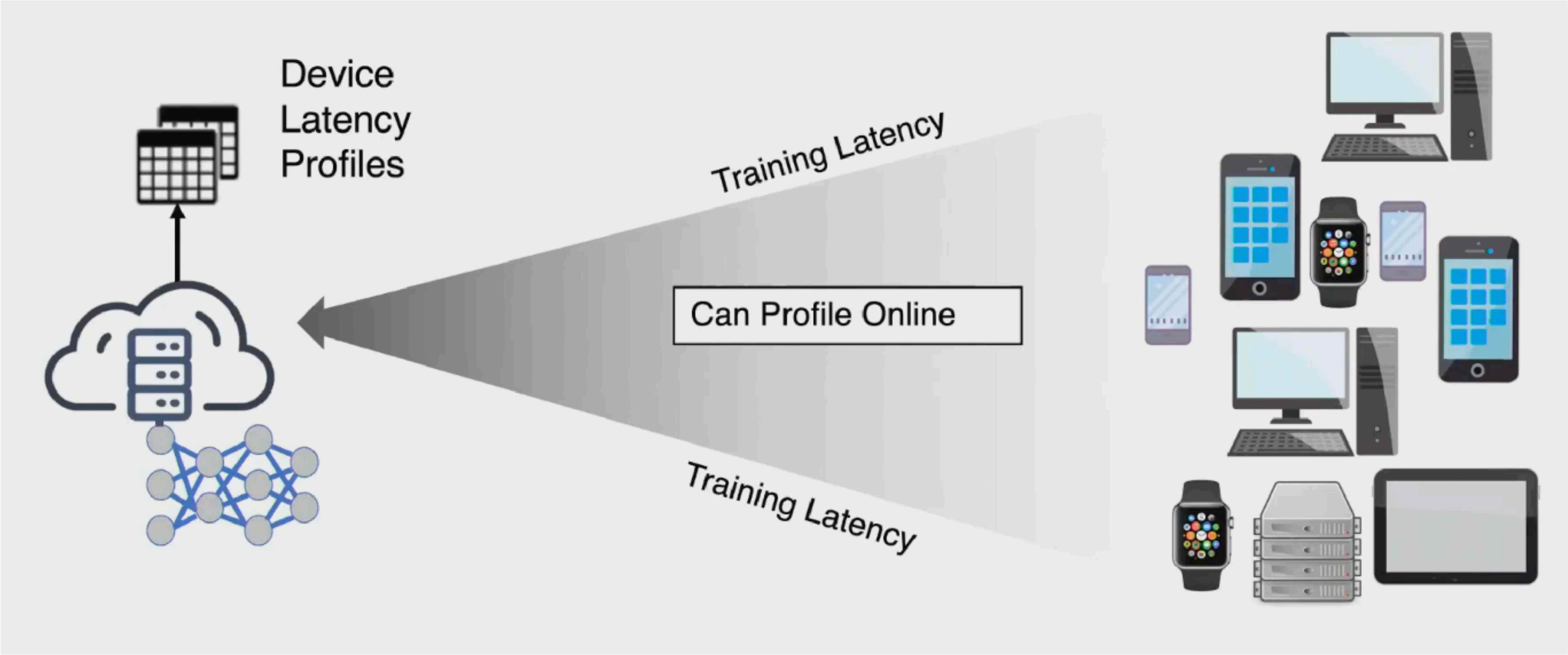
FedAT



w^{t_0}
w^{t_1}
w^{t_2}
w^{t_3}
w^{t_4}
Aggregation
Aggregation
Aggregation
Aggregation
Server
$w^{t_1}_{tier_1}$, …, $w^{t_1}_{tier_M}$
$w^{t_1}_{tier_1}$, $w^{t_2}_{tier_2}$, …
$w^{t_3}_{tier_1}$, $w^{t_2}_{tier_2}$, …
$w^{t_3}_{tier_1}$, …, $w^{t_4}_{tier_M}$
deCom
deCom
deCom
deCom
t_0 t_1 t_2 t_3 t_4
timeline
Compression
Compression
w^{t_2}
$tier_1$
Compression
$tier_2$
Compression
$tier_M$
Tiering Module
Clients

Profiling

- Same as TiFL
- Offline
- Online



Local Training

For training with Non-i.i.d. data, frequent local updates may potentially cause the local models to diverge:

- Varying updating frequency of different tiers
- Underlying data heterogeneity

Local Training

Local objective function:

$$h_k(w_k) = F_k(w_k) + \frac{\lambda}{2}||w_k - w||^2$$

Objective function of tier *m*:

$$f_{tier_m}(w) = \sum_{k=1}^{|\mathcal{S}_t|} \frac{n_k}{N_c} h_k(w_k)$$

$$= \sum_{k=1}^{|\mathcal{S}_t|} \frac{n_k}{N_c} (F_k(w_k) + \frac{\lambda}{2}||w_k - w||^2)$$

Cross-Tier Weighted Aggregation

To achieve unbiased, more balanced training:

- Assign relatively higher weights to slower tiers that update less frequently

- Global model not bias towards the faster tiers

A cross-tier, weighted aggregation rule

- Adjust the relative weights assigned to each tier based on the number of times a tier has updated the global mode

Cross-Tier Weighted Aggregation

Objective function of tier *m*:

$$f(w) = \sum_{m=1}^{M} \frac{T_{tier_{(M+1-m)}}}{T} f_{tier_m}(w)$$

$$T_{tier_1} + T_{tier_2} + \ldots + T_{tier_M} = T$$

$\frac{T_{tier_{(M+1-m)}}}{T}$ is relative weight of $tier$ m and $\sum_{m=1}^{M} \frac{T_{tier_{(M+1-m)}}}{T} = 1$

Algorithm

Algorithm 2: FedAT's Training Process

Input: w_{tier_m}, t, T and T_{tier_m}. w_{tier_m} denotes the weights of Tier m. t represents the global round t. T is the maximum global rounds. T_{tier_m} is the number of updates of tier m

Server: Initialize $w_{tier_1}, w_{tier_2} ... w_{tier_M}$ to w^{t_0}. Initialize $t, T_{tier_1} ... T_{tier_M}$ to 0

for *each tier* $m \in M$ ***in parallel*** **do**

  while $t < T$ **do**

    w^t = **WeightedAverage**($w_{tier_1}, w_{tier_2} ... w_{tier_M}$)

    S_m = (random set of clients from tier m)

    for *each client* $k \in S_m$ ***in parallel*** **do**

      $n_k = |\mathcal{D}_k|$

      $w_k^{t+1} = w_k^t - \eta \nabla h(w^t)$

    $N_c = \sum_{k=1}^{|S_m|} n_k$

    $w_{tier_m} = \sum_{k=1}^{|S_m|} \frac{n_k}{N_c} \cdot w_k^{t+1}$

    $T_{tier_m} = T_{tier_m} + 1$

    $t = t + 1$

function WeightedAverage($w_{tier_1}, w_{tier_2} ... w_{tier_M}$)

if $t == 0$ **then**

  return w^{t_0}

else

  return $\sum_{m=1}^{M} \frac{T_{tier_{(M+1-m)}}}{T} \cdot w_{tier_m}$

Compression

Due to the divergence of Non-i.i.d. data, some model compression methods such as quantization may lead to huge errors and reduce global performance.

Encoded Polyline Algorithm

- Lossy compression algorithm
- Convert a series of numbers to a single string
- Compress both uplink and downlink traffic

Compression

1. Take the initial signed value:
   -179.9832104

2. Take the decimal value and multiply it by 1e5, rounding the result:
   -17998321

3. Convert the decimal value to binary. Note that a negative value must be calculated using its two's complement by inverting the binary value and adding one to the result:
   00000001 00010010 10100001 11110001
   11111110 11101101 01011110 00001110
   11111110 11101101 01011110 00001111

4. Left-shift the binary value one bit:
   11111101 11011010 10111100 00011110

5. If the original decimal value is negative, invert this encoding:
   00000010 00100101 01000011 11100001

Compression

6. Break the binary value out into 5-bit chunks (starting from the right hand side):
   00001 00010 01010 10000 11111 00001
7. Place the 5-bit chunks into reverse order:
   00001 11111 10000 01010 00010 00001
8. OR each value with 0x20 if another bit chunk follows:
   100001 111111 110000 101010 100010 000001
9. Convert each value to decimal:
   33 63 48 42 34 1
10. Add 63 to each value:
    96 126 111 105 97 64
11. Convert each value to its ASCII equivalent:
    `~oia@

Compression

Encoded Polyline Algorithm

Example

Points: (38.5, -120.2), (40.7, -120.95), (43.252, -126.453)

<table>
<tr><th>Latitude</th><th>Longitude</th><th>Latitude in E5</th><th>Longitude in E5</th><th>Change In Latitude</th><th>Change In Longitude</th><th>Encoded Latitude</th><th>Encoded Longitude</th><th>Encoded Point</th></tr>
<tr><td>38.5</td><td>-120.2</td><td>3850000</td><td>-12020000</td><td>+3850000</td><td>-12020000</td><td>_p~iF</td><td>~ps|U</td><td>_p~iF~ps|U</td></tr>
<tr><td>40.7</td><td>-120.95</td><td>4070000</td><td>-12095000</td><td>+220000</td><td>-75000</td><td>_ulL</td><td>nnqC</td><td>_ulLnnqC</td></tr>
<tr><td>43.252</td><td>-126.453</td><td>4325200</td><td>-12645300</td><td>+255200</td><td>-550300</td><td>_mqN</td><td>vxq`@</td><td>_mqNvxq`@</td></tr>
</table>

Encoded polyline: ``_p~iF~ps|U_ulLnnqC_mqNvxq`@``

Compression

Compression process:

1. FedAT flattens the weights of each layer to get a list of decimal values
2. Use polyline encoding to convert every decimal value into a compressed string
3. Transmission
4. Decompress strings to decimal values
5. reshape to the original dimensions

Experimental Setup

Dataset: five datasets including FL benchmarking framework LEAF

- CIFAR-10 ——CNN (three convolutional layers and two fully connected layers)
- Fashion-MNIST ——CNN (three convolutional layers and two fully connected layers)
- Sentiment140 ——logistic regression
- FEMNIST ——CNN
- Reddit ——LSTM

Experimental Setup

FL Methods:

- FedAVG
- TiFL
- FedProx
- FedAsync
- Aso-Fed

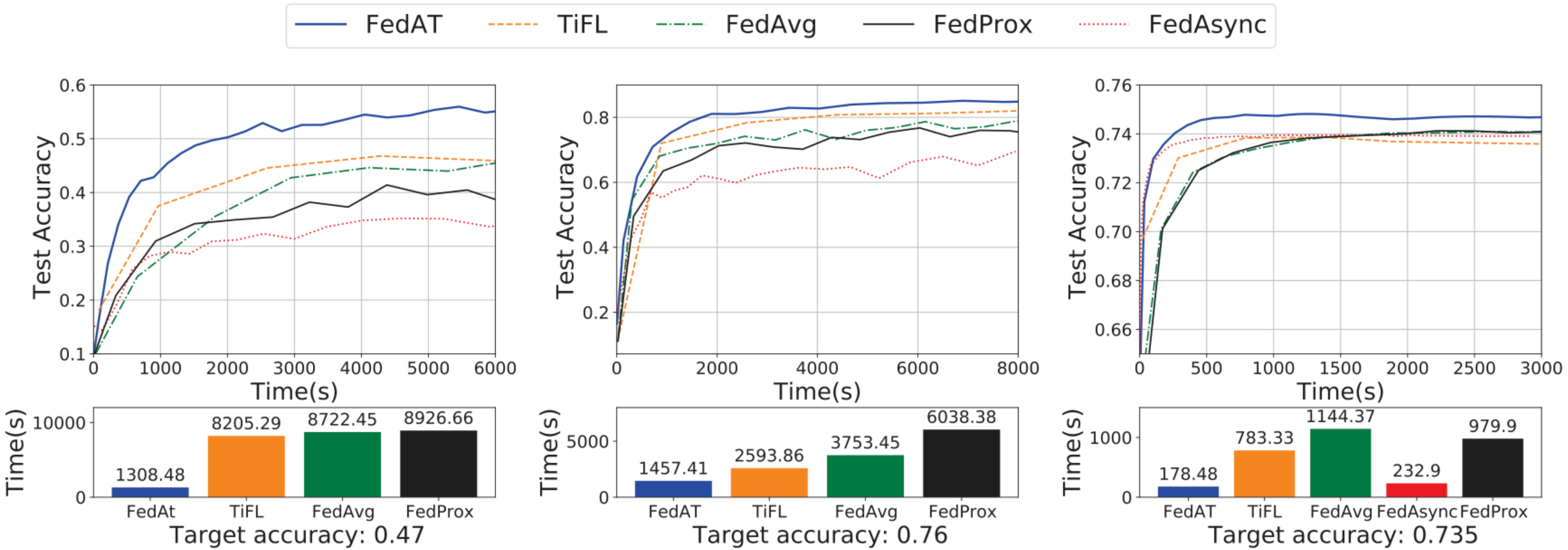
Experimental Setup

- TensorFlow
- An 80% training set and a 20% testing set
- Local constrain parameter λ: 0.4
- Divide all the clients into 5 tiers with delays of 0s, 0 ~ 5s, 6 ~ 10s, 11 ~ 15s, and 20 ~ 30s
- Randomly select 10 “unstable” clients, which would drop out and will not come back again

Prediction Performance

Dataset(#class)		CIFAR-10					Fashion-MNIST	Sentiment140
		#2	#4	#6	#8	i.i.d.	#2	
TiFL	Accuracy	0.527	0.615	0.654	0.655	0.685	0.859	0.739
	Norm. Var.	1.26	2.79	1.33	1.3	2.12	1.29	2.75
FedAvg	Accuracy	0.547	0.628	0.654	0.667	0.686	0.842	0.741
	Norm. Var.	2	5.07	4.33	3.1	4.23	1.86	3.72
FedProx	Accuracy	0.509	0.609	0.624	0.650	0.669	0.831	0.742
	Norm. Var.	1.261	6.75	3.981	2.22	2.992	2.243	3.89
FedAsync	Accuracy	0.480	0.541	0.531	0.561	0.567	0.795	0.740
	Norm. Var.	2	3.93	2.08	1.54	2.69	2	5.69
FedAT	Accuracy	**0.591**	**0.633**	**0.673**	**0.681**	**0.701**	**0.873**	**0.748**
	Abs. Var.	**0.0042**	**0.0014**	**0.0012**	**0.001**	**0.00052**	**0.007**	**$2.67e^{-5}$**
	impr.(a)	7.44%	0.79%	2.82%	2.05%	2.13%	1.6%	0.93%
	impr.(b)	18.78%	14.53%	21.09%	17.62%	19.11%	8.93%	1.2%

Prediction Performance

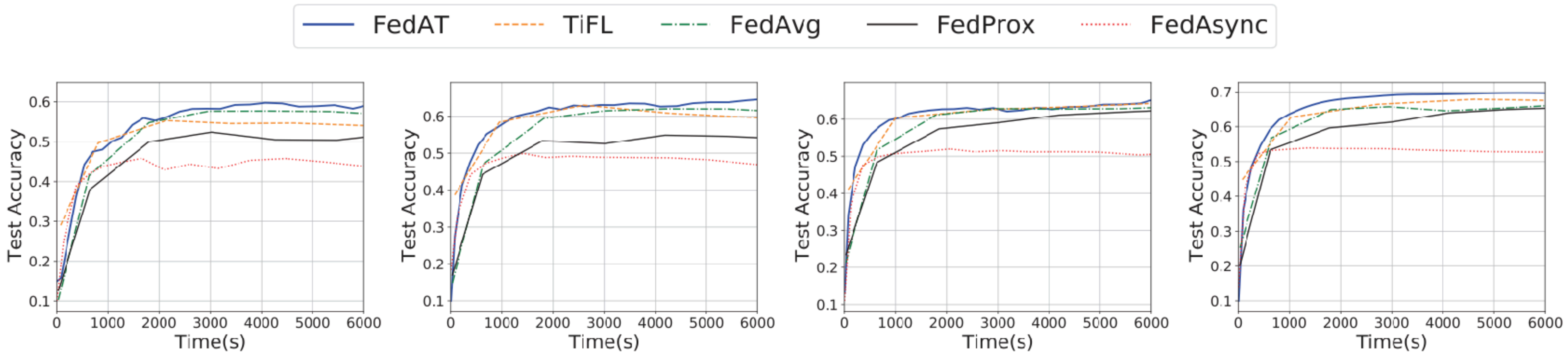


(a) CNN @ CIFAR-10.

(b) CNN @ Fashion-MNIST.

(c) Logistic @ Sentiment140.

Non-i.i.d. Level

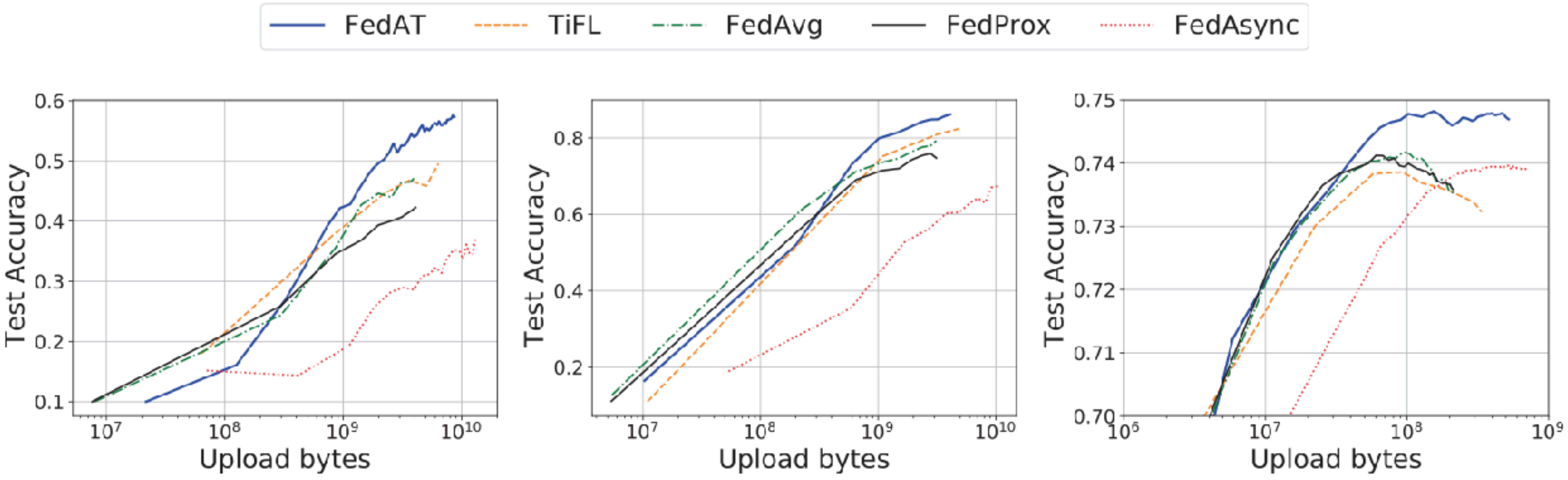


(a) CIFAR-10 Non-i.i.d. (4). (b) CIFAR-10 Non-i.i.d. (6). (c) CIFAR-10 Non-i.i.d. (8). (d) CIFAR-10 i.i.d.

Figure 3: Convergence speed comparison on CIFAR-10 over different level of Non-i.i.d.-ness. The results are average-smoothed for every 40 global rounds.

Communication Cost

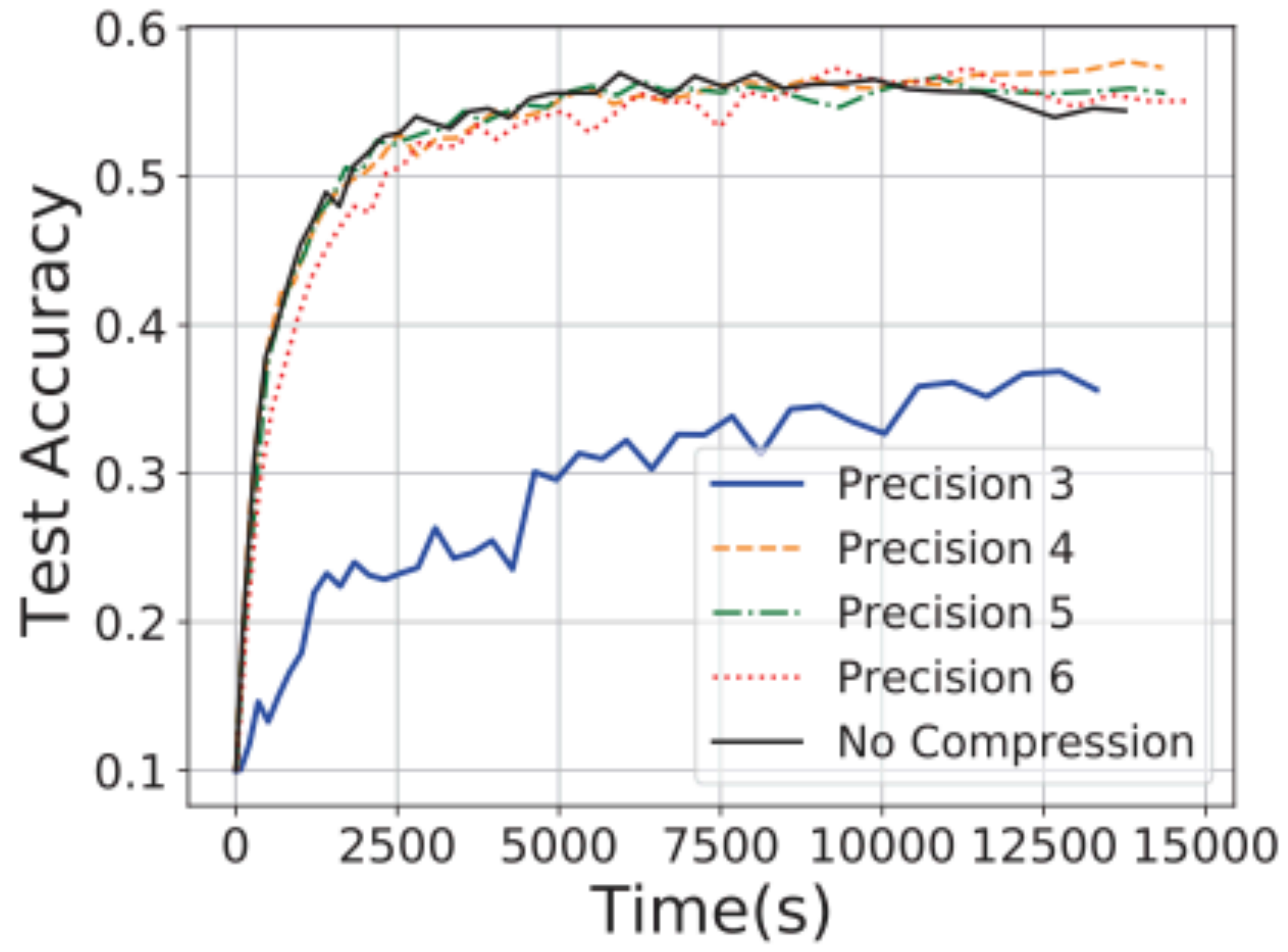
Method	CIFAR-10 (acc. = 0.50)	Fashion-MNIST (acc. = 0.79)	Sentiment140 (acc. = 0.73)
FedAvg	1828.54	1048.25	16.71
TiFL	2140.71	1041.98	17.20
FedProx	–	2169.95	18.42
FedAsync	–	9895.53	82.27
FedAT	**1675.82**	**1041.54**	**16.41**



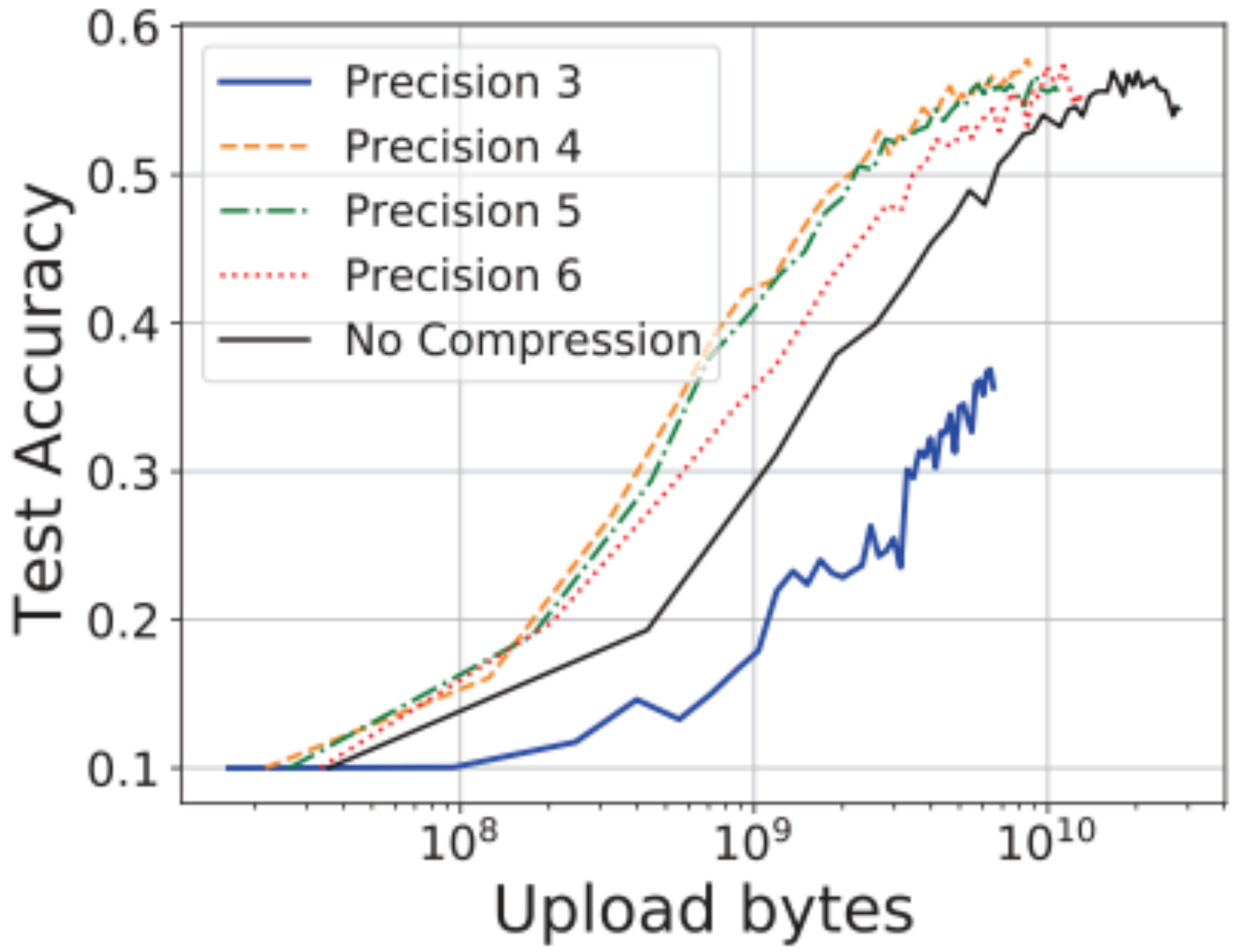
(a) CIFAR-10. (b) Fashion-MNIST. (c) Sentiment140.

Figure 4: Test accuracy as a function of the cumulative amounts of data uploaded from clients to the server for 2-class Non-i.i.d. datasets. The performance curves are average-smoothed for every 40 global rounds. The X-axis is in log-scale.

Communication Cost



(a) Test accuracy.



(b) Uploaded data vs. accuracy (the X-axis is in log-scale).

Figure 5: Impact of FedAT's compression precision on the prediction performance and the communication cost, for the CIFAR-10 Non-i.i.d. 2-class dataset. All results are plotted with the average of every 40 global rounds.

Weighted Aggregation

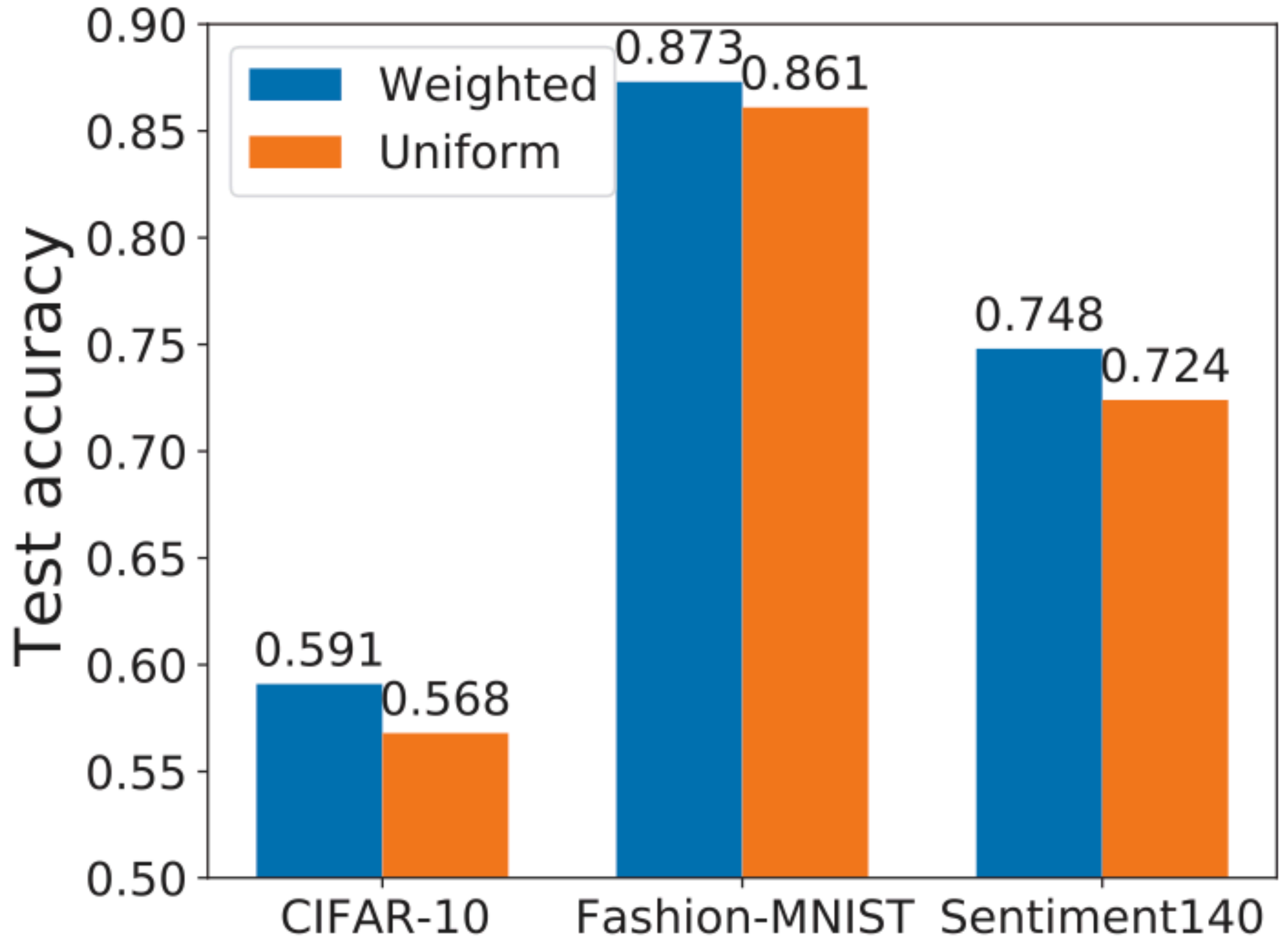
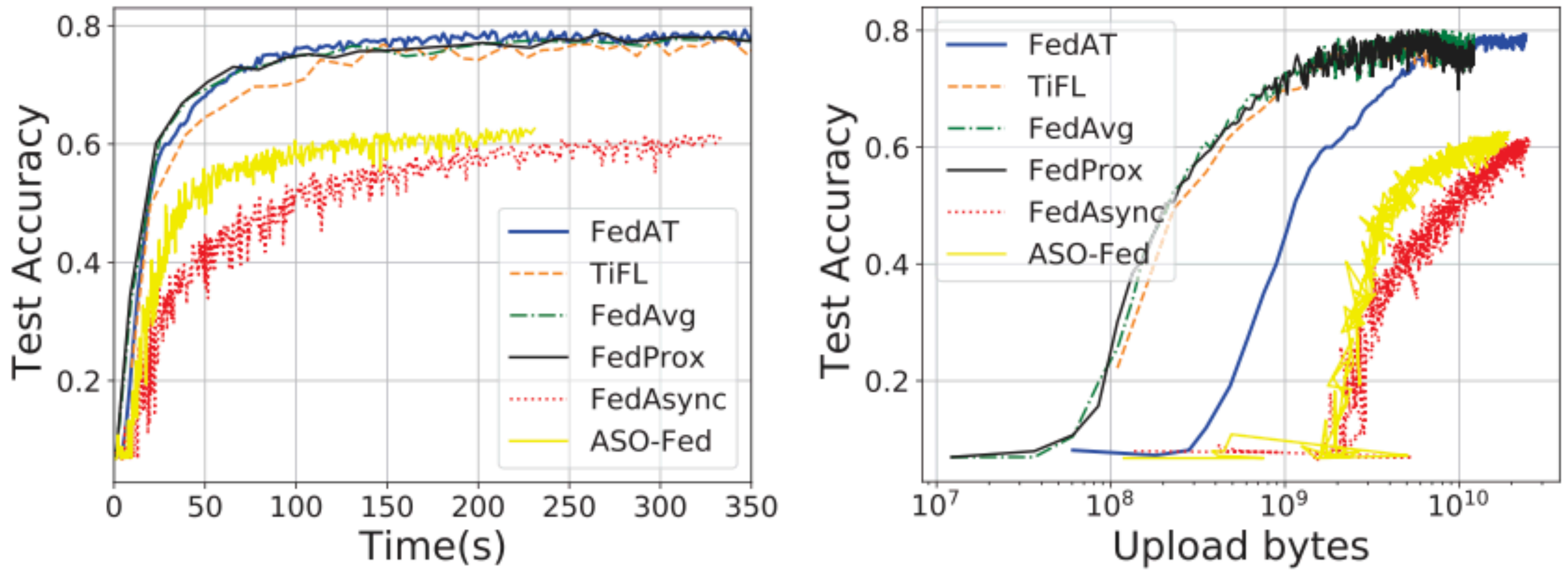


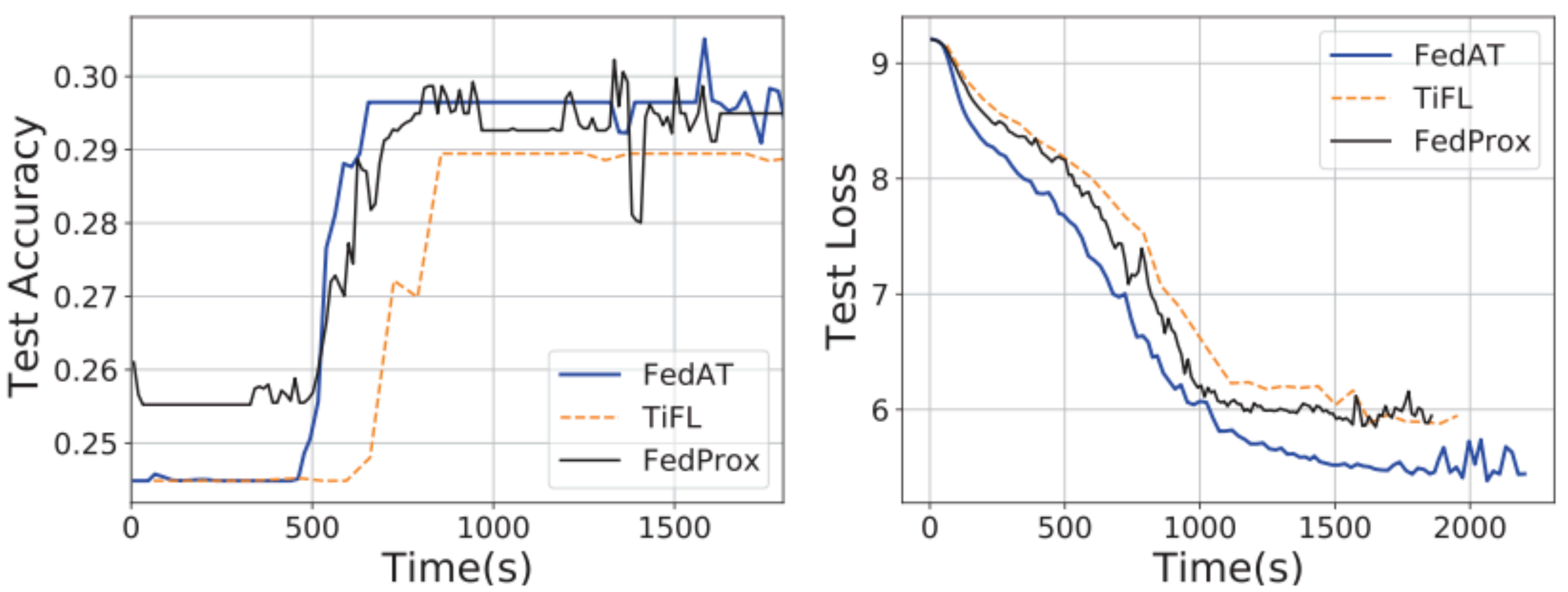
Figure 6: Comparison of FedAT's weighted aggregation heuristic vs. a uniform baseline that assigns uniform weights when aggregating models from different tiers.

Large Scale Training



(a) Accuracy over time. (b) Accuracy over uploaded bytes.

Figure 7: Prediction accuracy of FEMNIST as a function of training time (a) and cumulative amount of data uploaded from clients to the server (b).

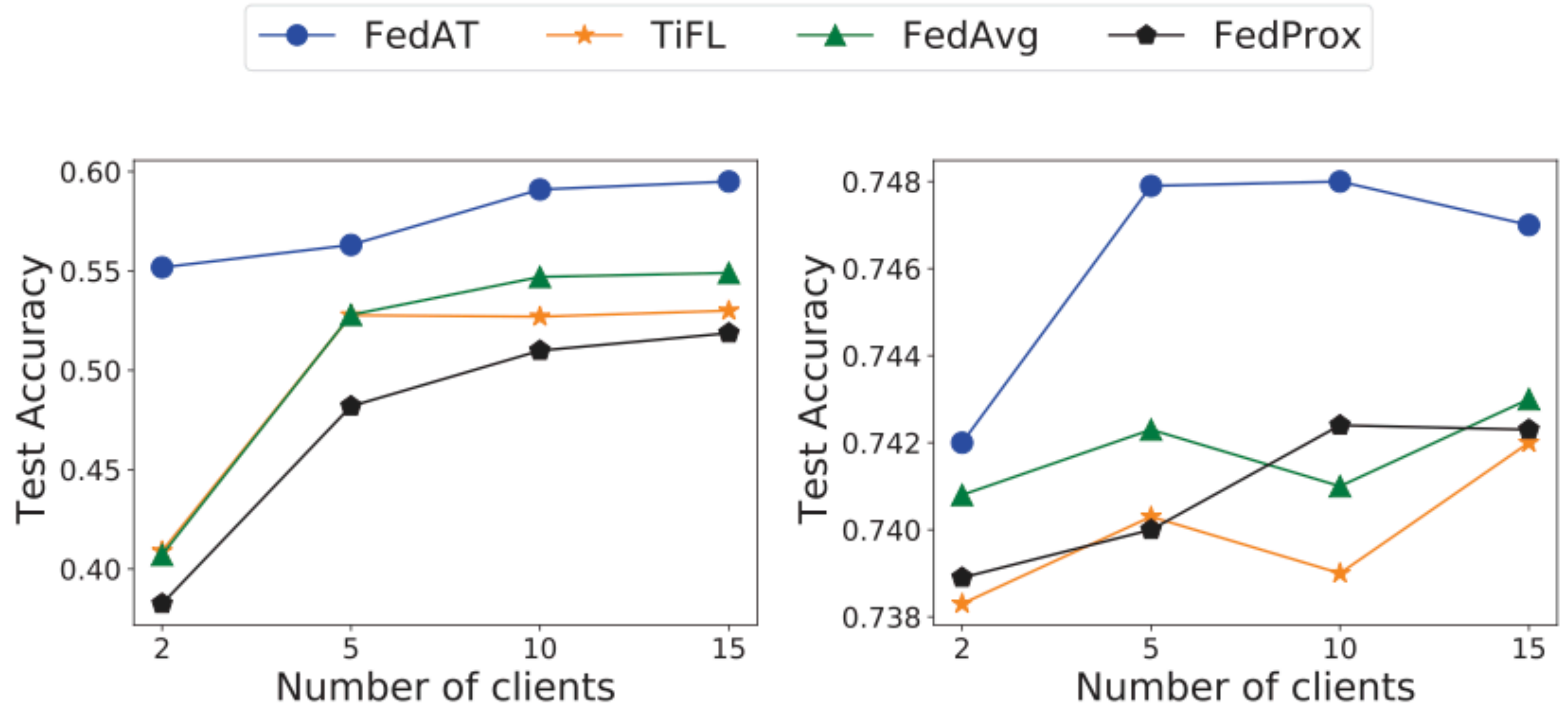


(a) Accuracy over time. (b) Loss over time.

Figure 8: Prediction accuracy (a) and loss (b) of Reddit as a function of training time.

Large-scale experiments on the FEMNIST and Reddit datasets with 500 participating clients deployed on 100 *c5.2xlarge* AWS EC2 VMs

Number of Clients



(a) CIFAR-10 Non-i.i.d.(2).

(b) Sentiment140.

Figure 9: Prediction accuracy of CIFAR-10 (left) and Sentiment140 (right) as a function of the number of participating clients in one iteration. This test compares FedAT against three other FL methods (FedAvg, TiFL, FedProx), which all feature synchronous updating.

Distribution of Clients

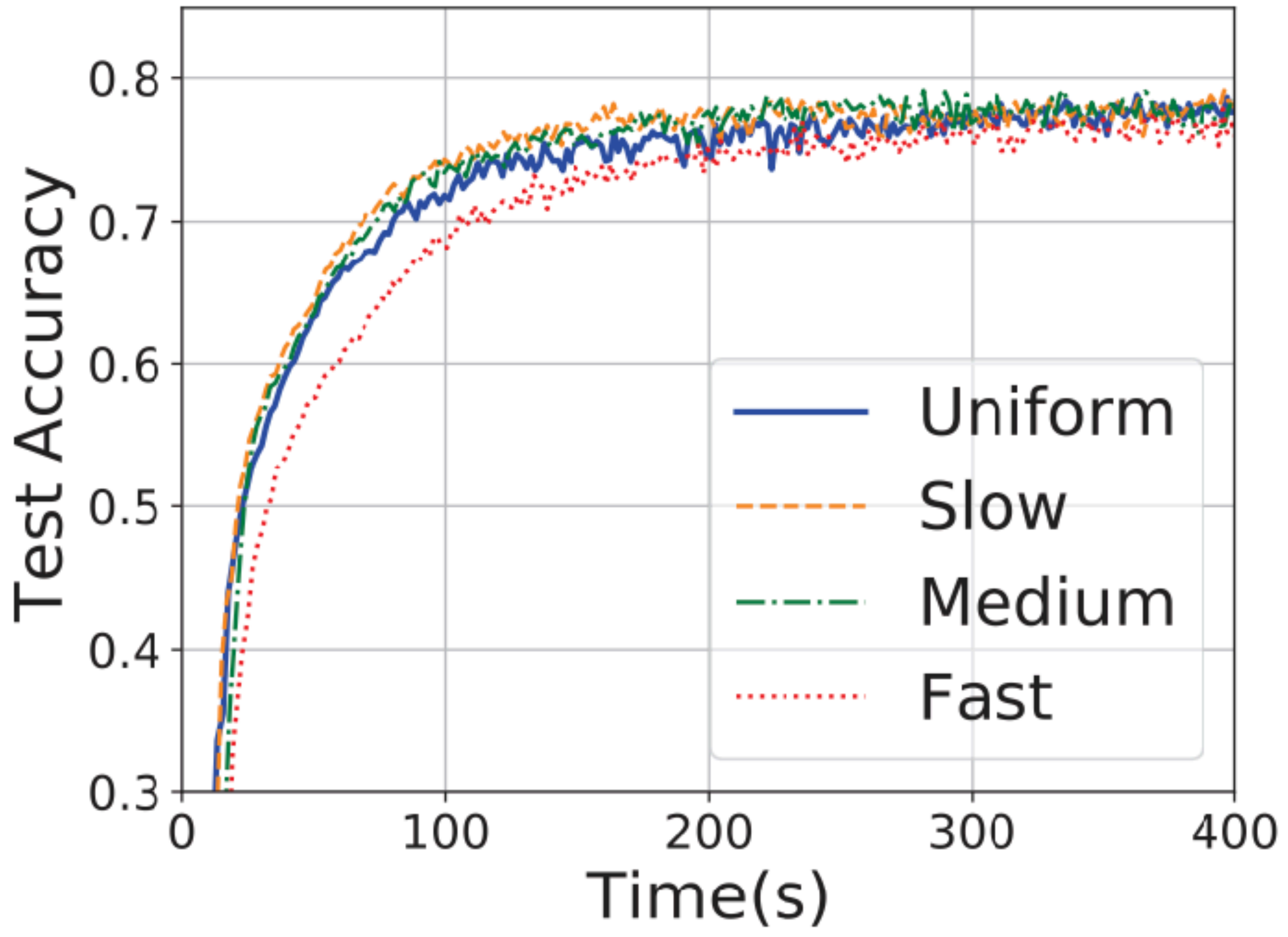


Figure 10: Comparison of prediction accuracy over time on FEMNIST under different configurations of client distribution across tiers.

Uniform: 100/100/100/100/100

Slow: 50/50/100/100/200

Medium: 50/100/200/100/50

Fast: 200/100/100/50/50

Summary

- A new way to compress models
- Experiments are quite detailed and convincing
- The authors don't consider the clients with varying communication capabilities