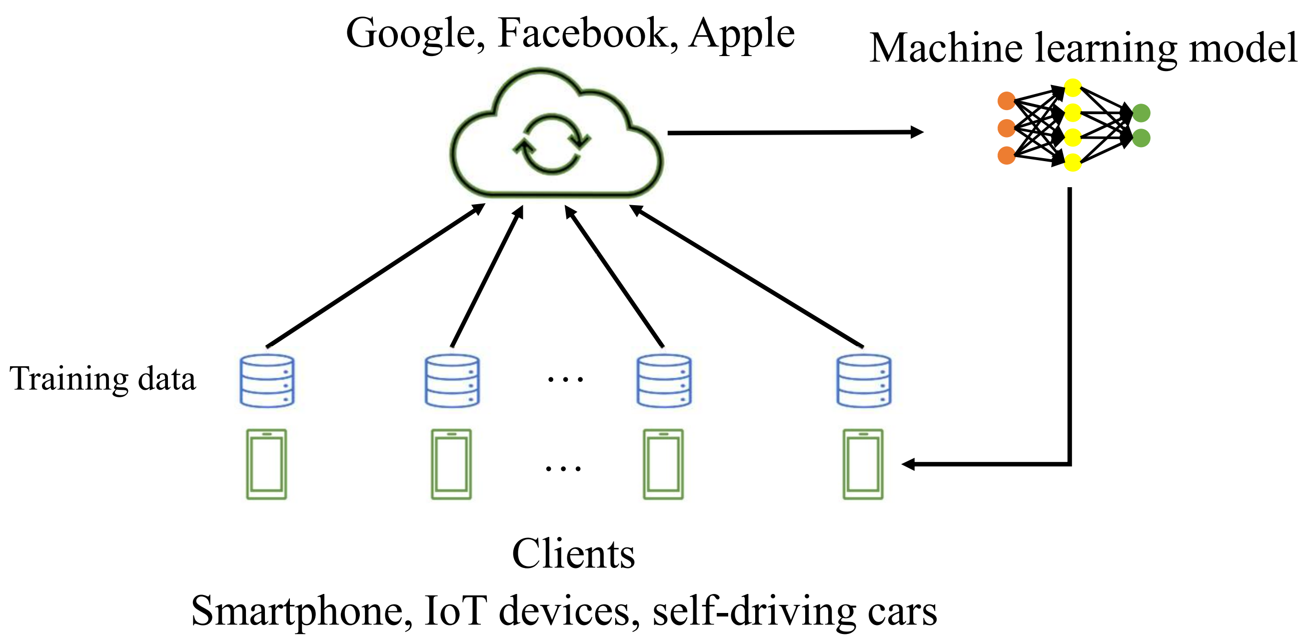
AFLGuard: Byzantine-robust Asynchronous Federated Learning

Minghong Fang, Jia Liu, Neil Zhenqiang Gong, and Elizabeth S. Bentley

Conventional Paradigm: Centralized Learning



Challenges of Centralized Learning

- Data leakage
- High communication cost
  - Intolerable for resource-constrained clients

Federated Learning

- Training data stay locally on clients
- Clients train models locally
- Clients send model updates to server
- Real-world deployment

Gboard

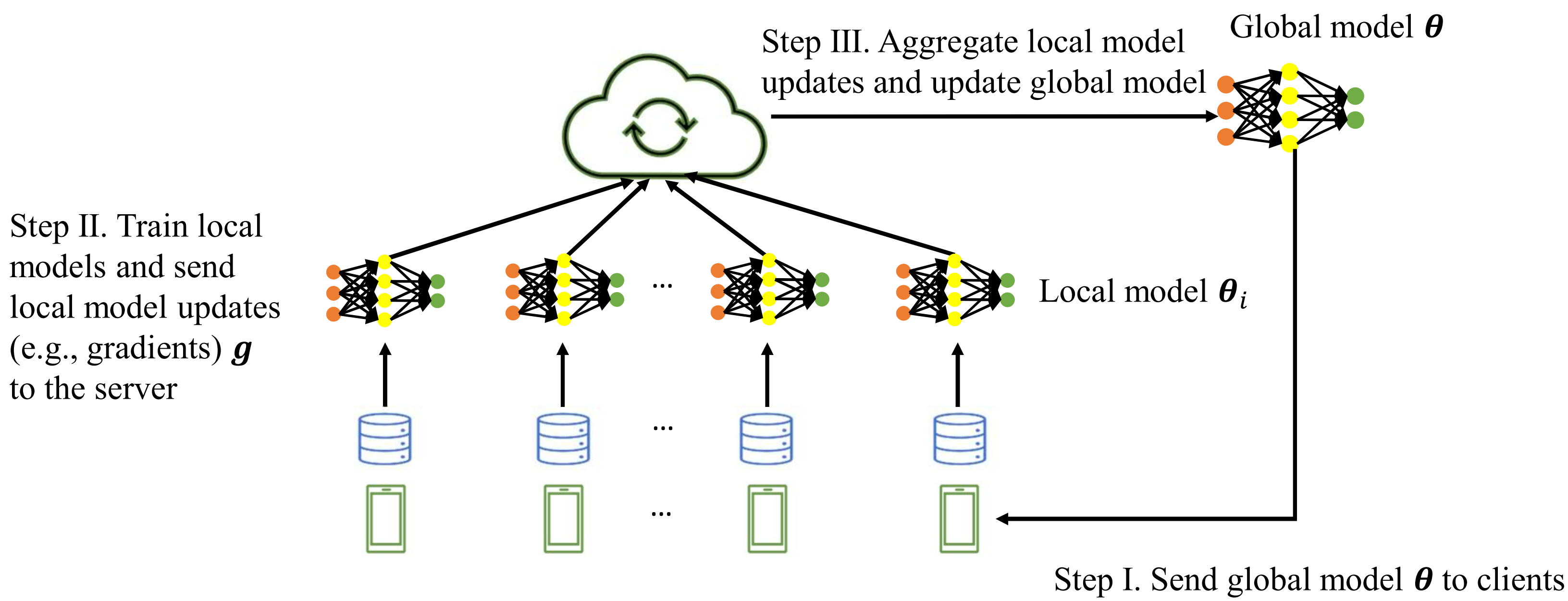
Artificial intelligence / Machine learning

How Apple personalizes Siri without hoovering up your data

The tech giant is using privacy-preserving machine learning to improve its voice assistant while keeping your data on your phone.

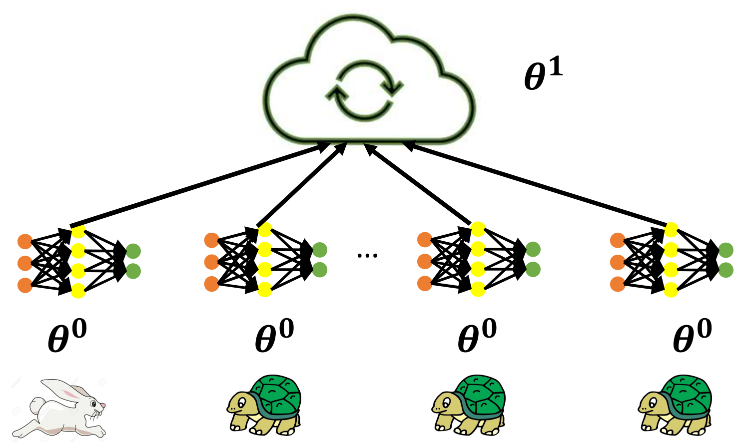
by **Karen Hao** December 11, 2019

Federated Learning Background



Synchronous Federated Learning

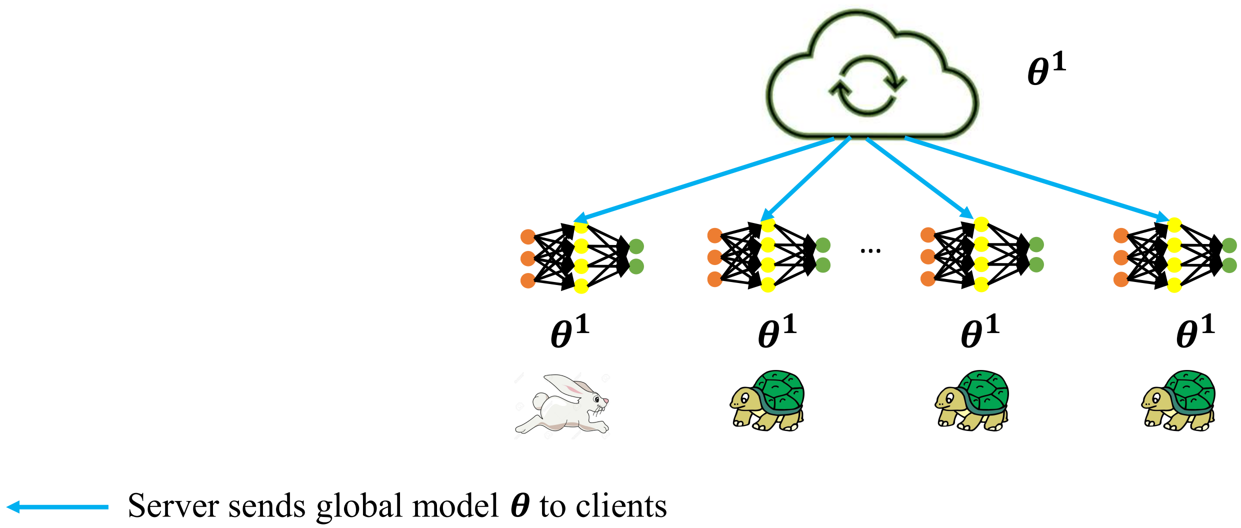
- Clients use the same global model to update local models
- Server has to wait until receiving local model updates from all clients
- Training process is slow, due to straggling clients
  - Heterogenous computing capabilities



→ Clients send local model updates $\boldsymbol{g}$ to the server

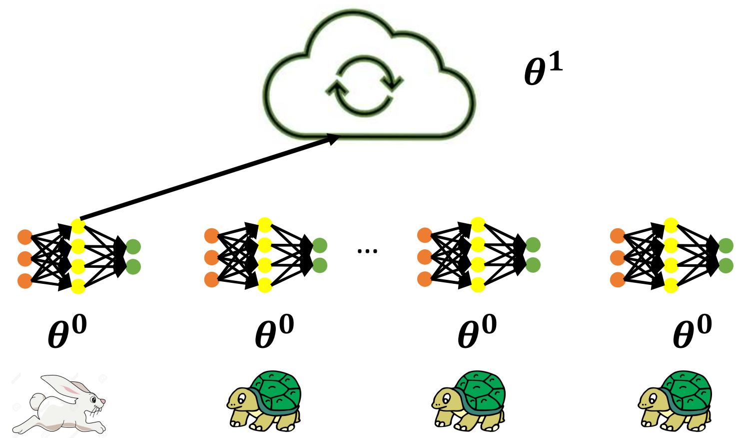
Synchronous Federated Learning

- Clients use the same global model to update local models
- Server has to wait until receiving local model updates from all clients
- Training process is slow, due to straggling clients
  - Heterogenous computing capabilities



Asynchronous Federated Learning

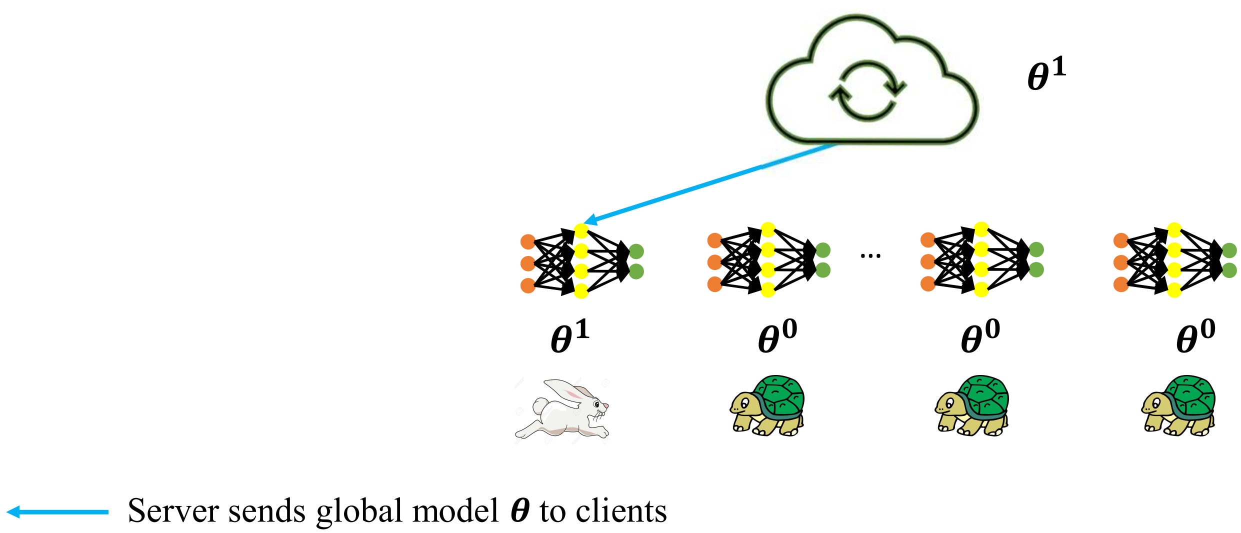
- Clients use different global models to update local models
- Server updates the global model immediately upon receiving local model update from any client



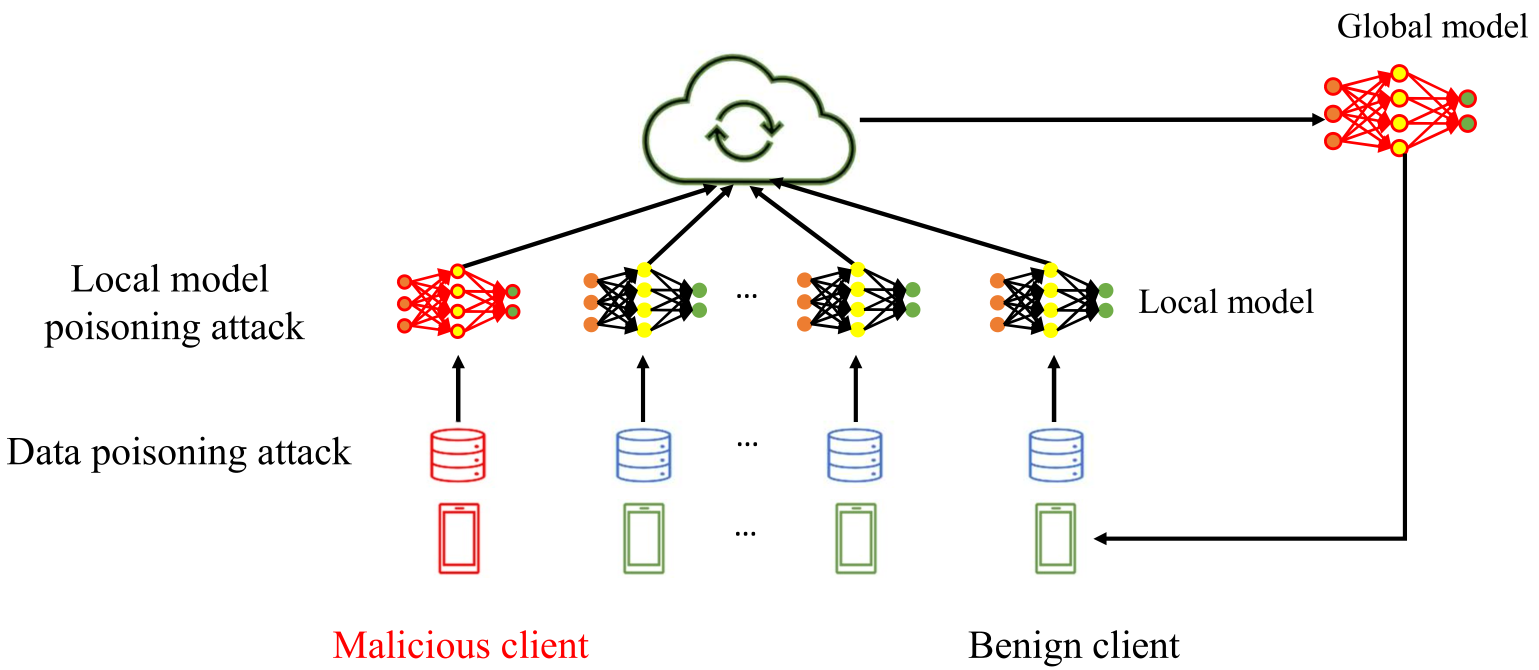
Clients send local model updates $\boldsymbol{g}$ to the server

Asynchronous Federated Learning

- Clients use different global models to update local models
- Server updates the global model immediately upon receiving local model update from any client



Poisoning Attacks to Federated Learning



Challenges

- Only one local model update is received, nothing to compare against
- Difficult to distinguish between malicious local model updates and delayed benign local model updates

Our AFLGuard

- Server collects a small trusted training dataset

- Server maintains a *server model*
  - Like how a client maintains a local model

- Use server model update to filter out malicious information

Our AFLGuard

A client local model update $\boldsymbol{g}_i$ is considered malicious if

- Direction of $\boldsymbol{g}_i$ deviates substantially from that of $\boldsymbol{g}_s$ (server model update) *or*
- Magnitude of $\boldsymbol{g}_i$ deviates substantially from that of $\boldsymbol{g}_s$

$$\|\boldsymbol{g}_i - \boldsymbol{g}_s\| \leq \lambda \|\boldsymbol{g}_s\|$$

Experimental Results

MNIST

100 clients, 20 malicious

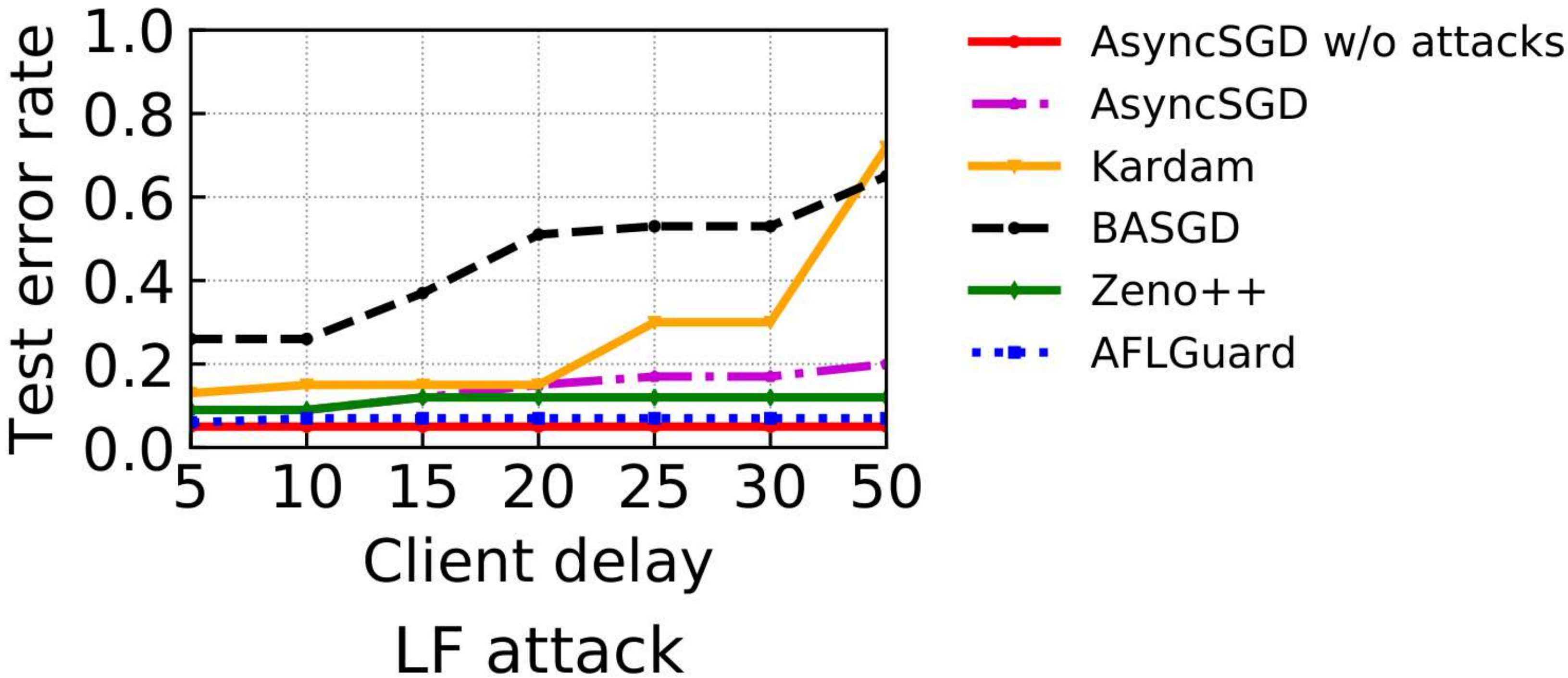
Server's trusted training dataset: 100 examples sampled from MNIST

Maximum client delay and server delay are set to 10

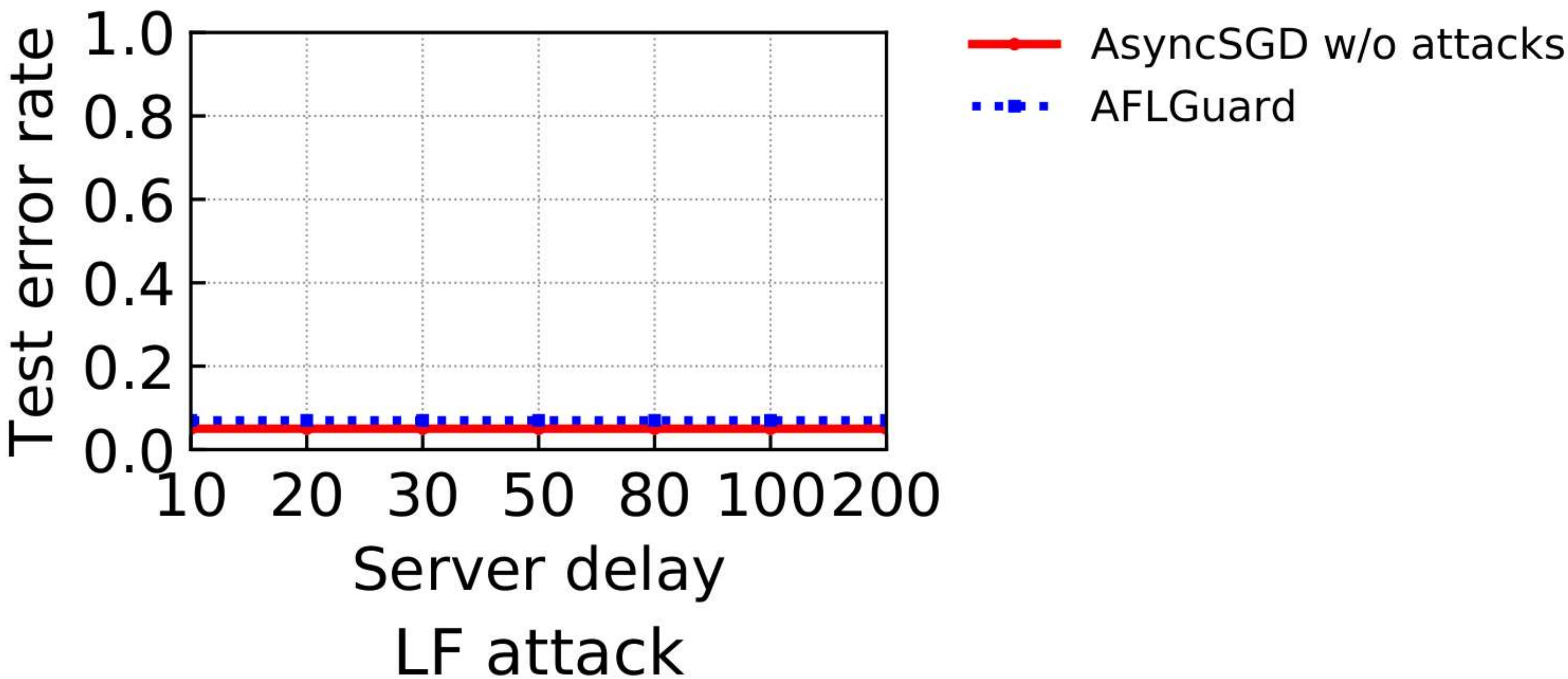
	AsyncSGD	Kardam	BASGD	Zeno++	AFLGuard
No attack	0.05	0.12	0.19	0.08	0.06
LF attack	0.09	0.15	0.26	0.09	0.07
Gauss attack	0.91	0.39	0.27	0.09	0.07
GD attack	0.90	0.90	0.89	0.09	0.07
Adapt attack	0.91	0.91	0.90	0.10	0.07

The testing error rates of the global model.

Client Delay



Server Delay



Conclusion

- We propose a new method called AFLGuard to defend against poisoning attacks in asynchronous federated learning

- We theoretically and empirically show the robustness of AFLGuard

Thank You!