GlueFL: Reconciling Client Sampling and Model Masking for Bandwidth Efficient Federated Learning

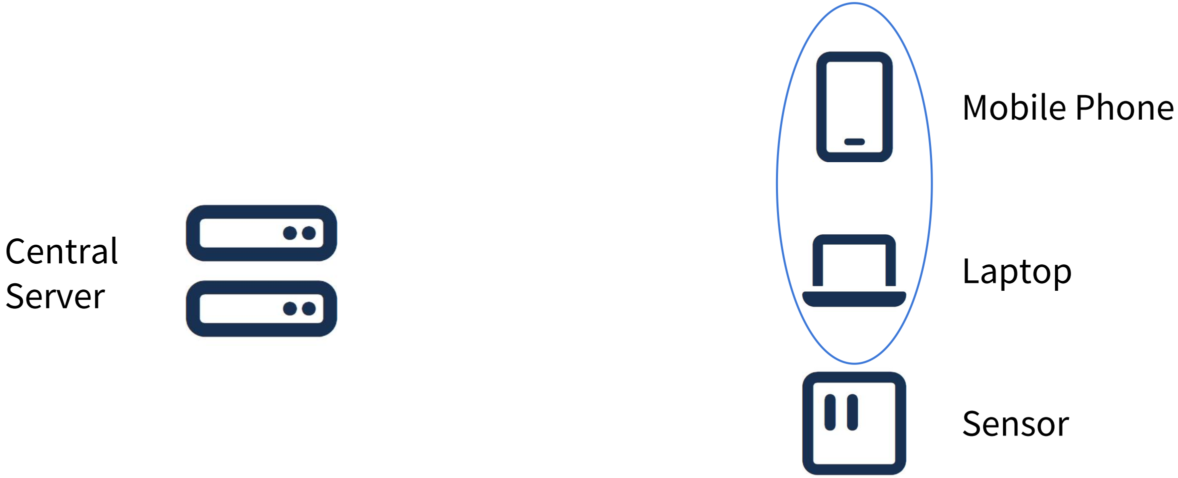
Shiqi He[1], Qifan Yan[1], Feijie Wu[2], Lanjun Wang[3], Mathias Lécuyer[1], Ivan Beschastnikh[1]
[1]University of British Columbia, [2]Purdue University, [3]Tianjin University

Agenda

- Background and Motivation
  - Federated learning (FL)
  - Masking in FL
- GlueFL Framework Design
  - Sticky sampling
  - Mask shifting
- Experiment Results
- Conclusion

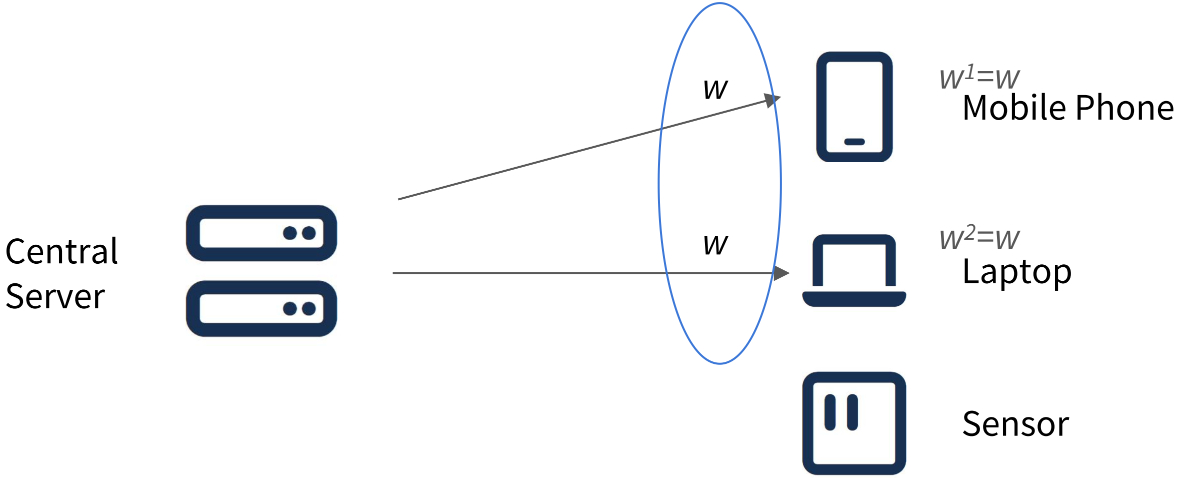
Federated Learning (FL)

- A typical FL training involves four steps in each round
  1. Server **selects** a set of participants from a large number of edge devices (e.g., 10^6)



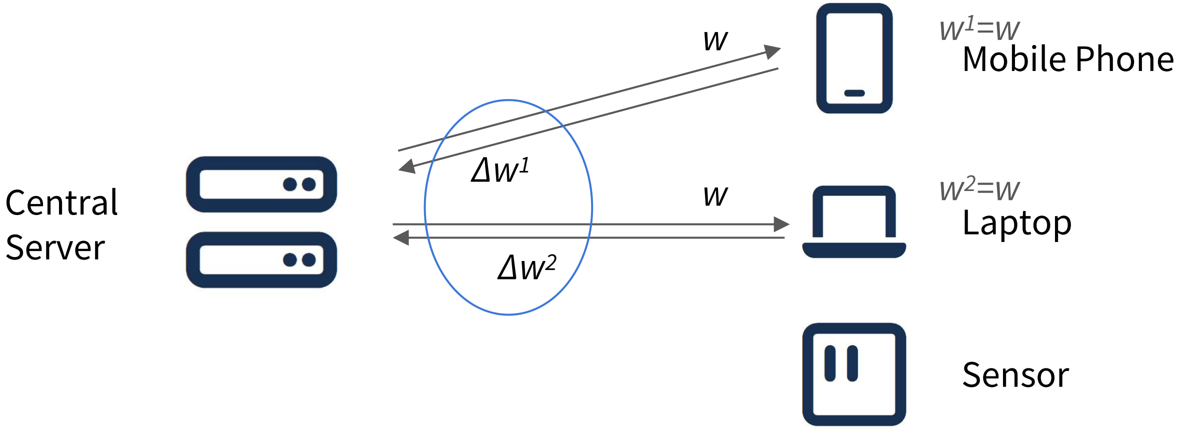
Federated Learning (FL)

- A typical FL training involves four steps in each round
  1. Server **selects** a set of participants from a large number of edge devices (e.g., 10^6)
  2. Server **broadcasts** the global model w to selected clients



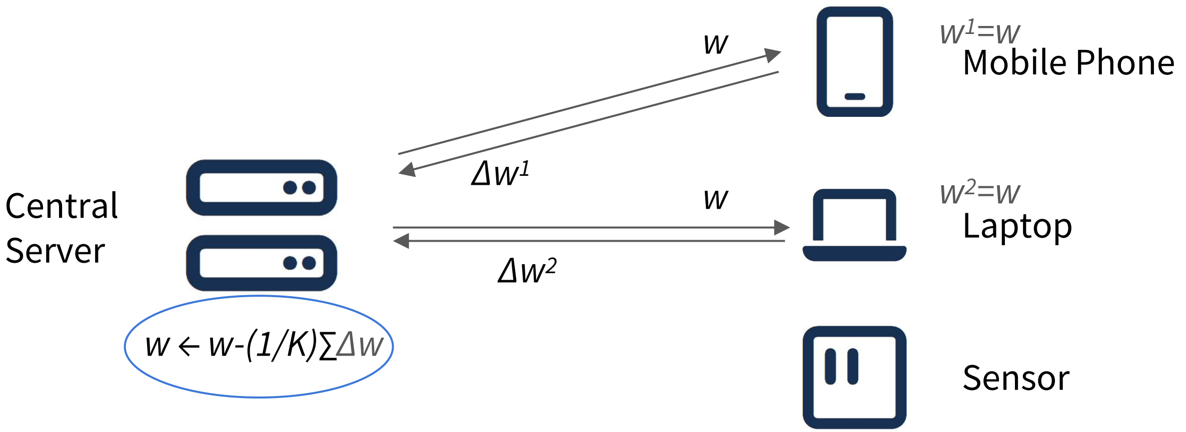
Federated Learning (FL)

- A typical FL training involves four steps in each round
  1. Server **selects** a set of participants from a large number of edge devices (e.g., 10^6)
  2. Server **broadcasts** the global model w to selected clients
  3. Each client **computes** local update **Δw** and send back to server



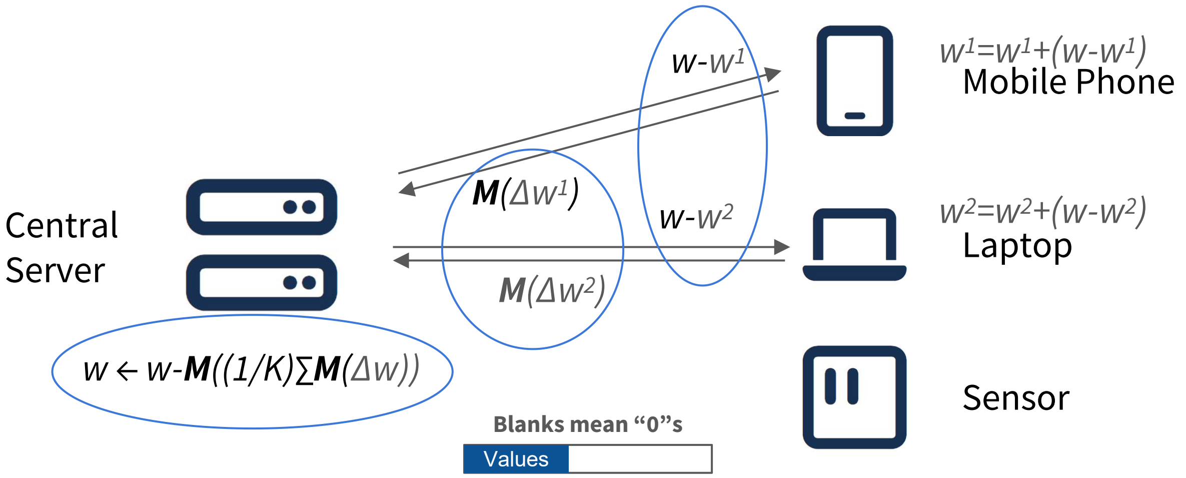
Federated Learning (FL)

- A typical FL training involves four steps in each round
  1. Server **selects** a set of participants from a large number of edge devices (e.g., 10^6)
  2. Server **broadcasts** the global model w to selected clients
  3. Each client **computes** local update **Δw** and send back to server
  4. Server receives updates and **update** the global model



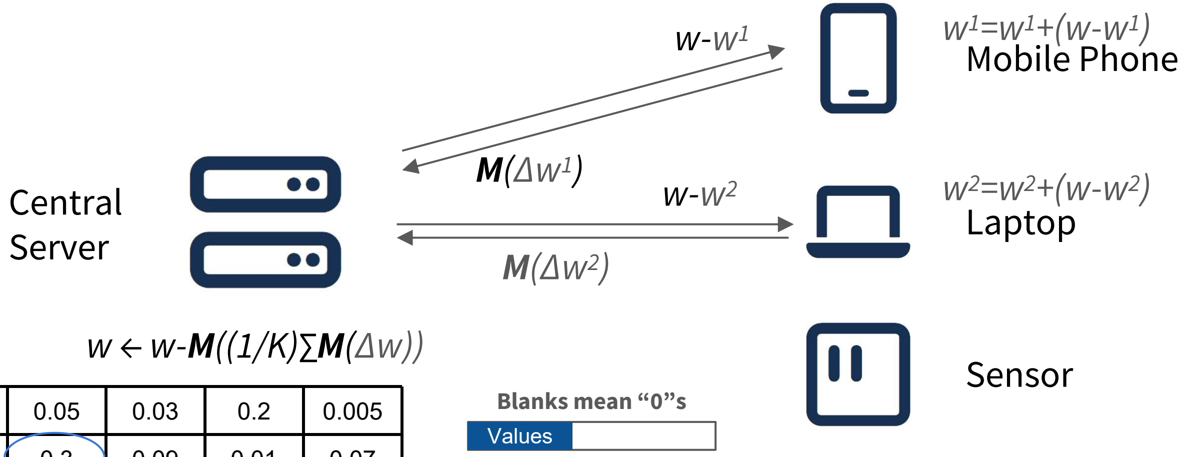
Masking in FL

- To reduce bandwidth usage, apply a mask function ***M*(·)** to both local updates **Δw** and server updates ***(1/K)∑Δw***
  - Masking is commonly used approch in distributed machine learning (ML)
  - Mask function *M*(·) returns the most informative part of the input (e.g., 10% largest values)



Masking in FL

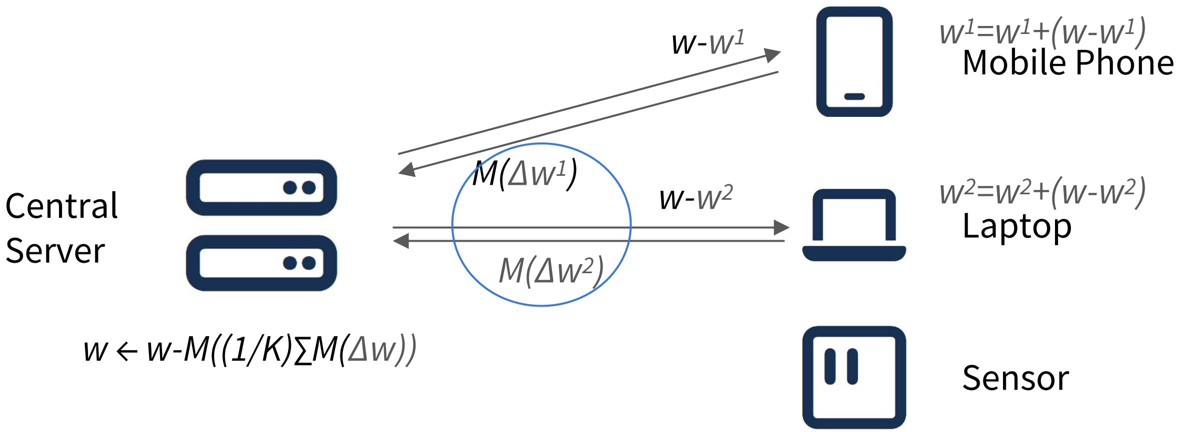
- To reduce bandwidth usage, apply a mask function ***M*(·)** to both local updates ***Δw*** and server updates ***(1/K)∑Δw***
  - Masking is commonly used approch in distributed machine learning (ML)
  - Mask function *M*(·) returns the most informative part of the input (e.g., 10% largest values)



0.01	0.05	0.03	0.2	0.005
0.04	0.3	0.09	0.01	0.07

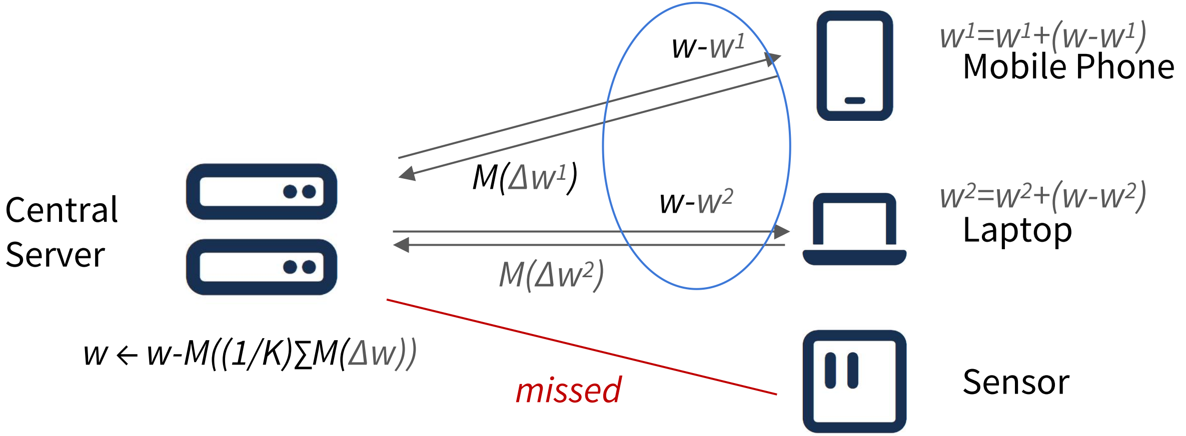
Masking in FL

- Masking saves upstream bandwidth
  - Upstream: $\Delta w^1 \rightarrow M(\Delta w^1)$ (e.g., only need to send largest 10% update)



Masking in FL

- Masking saves upstream bandwidth
  - Upstream: $\Delta w^1 \rightarrow M(\Delta w^1)$ (e.g., only need to send largest 10% update)
- Masking saves downstream bandwidth **?**
  - Downstream: $w \rightarrow w - w^1$ (how much bandwidth can we save?)
  - Due to client sampling, a client will have to download missed updates



Masking in FL

- Downstream: $w \rightarrow w - w^1$ (how much bandwidth can we save?)
- We conduct experiment to evaluate downstream bandwidth usage
  - With **10% client sample ratio and 20% mask ratio**, a sampled client needs to download 70% of the global model

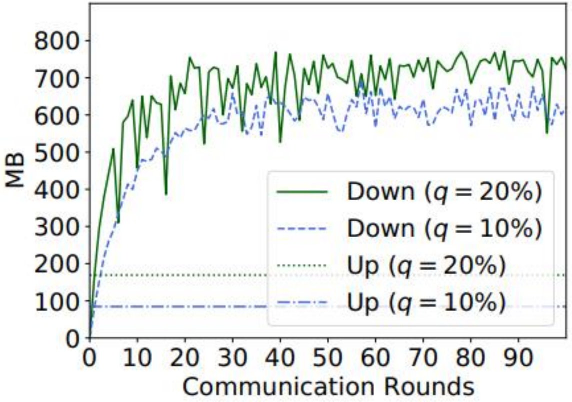


Fig 3. Downstream and upstream bandwidth usage of all clients per round

Masking in FL

- Downstream: $w \rightarrow w - w^1$ (how much bandwidth can we save?)
- We conduct experiment to evaluate downstream bandwidth usage
  - With **10% client sample ratio and 20% mask ratio**, a client has to download the entire model when it is being re-sampled after 20 rounds

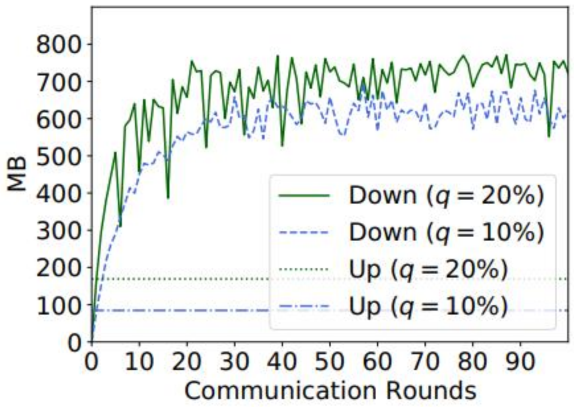


Fig 3. Downstream and upstream bandwidth usage of all clients per round

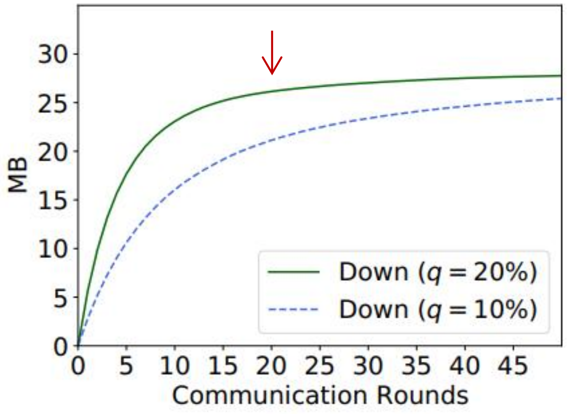
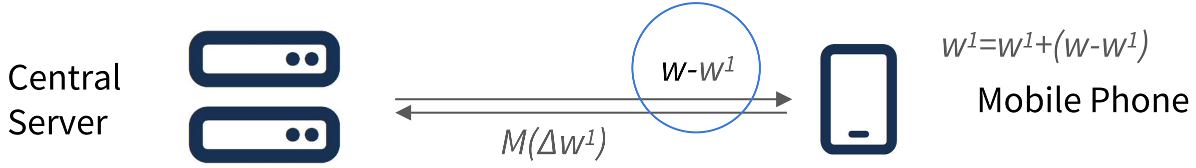


Fig 4. Model size a client must download when being re-sampled after a certain number of rounds

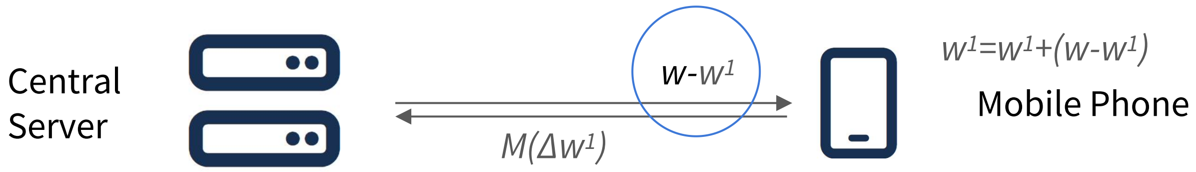
Masking in FL

- Masking **fails** to save much downstream bandwidth
- Downstream bandwidth increases because ***client local model states become stale***
  - A client will skip many rounds by not being sampled in cross-device FL
    - With 10% client sample ratio, a client will be re-sampled after 10 rounds in expectation. It misses the server updates of all these 10 rounds.



Masking in FL

- Masking **fails** to save much downstream bandwidth
- Downstream bandwidth increases because ***client local model states become stale***
  - A client will skip many rounds by not being sampled in cross-device FL
    - With 10% client sample ratio, a client will be re-sampled after 10 rounds in expectation. It misses the server updates of all these 10 rounds.
  - The server updates of two successive rounds have little overlap
    - With 10% mask ratio (server updates 10% global model weights in each round), a client will have to 20% of the model if it skipped 1 rounds.



Agenda

- Background and Motivation
  - Federated learning (FL)
  - Masking in FL
- GlueFL Framework Design
  - Sticky sampling
  - Mask shifting
- Experiment Results
- Conclusion

GlueFL - Sticky Sampling

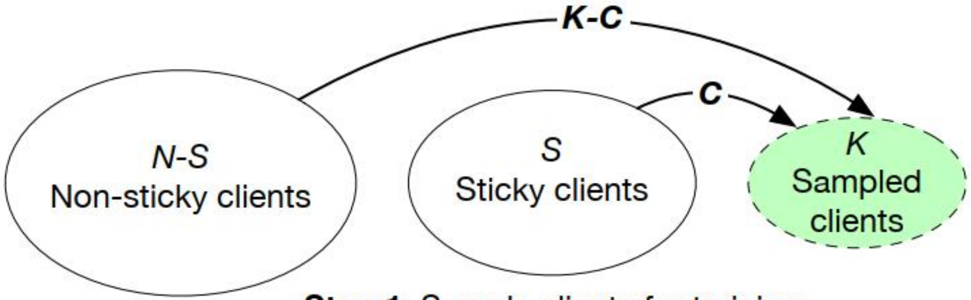
- Sticky sampling ensures that clients with an **up-to-date local state** are **more likely** to be selected
    - Up-to-date local state: clients participated training in the past few rounds
    - So they need to download less updates to synchronize the global model

GlueFL - Sticky Sampling

- Sticky sampling ensures that clients with an **up-to-date local state** are **more likely** to be selected
  - Up-to-date local state: clients participated training in the past few rounds
  - So they need to download less updates to synchronize the global model
- We call these clients **sticky clients (recently used clients)**. We construct a sticky group with sticky clients and the rest clients form a non-sticky group

GlueFL - Sticky Sampling

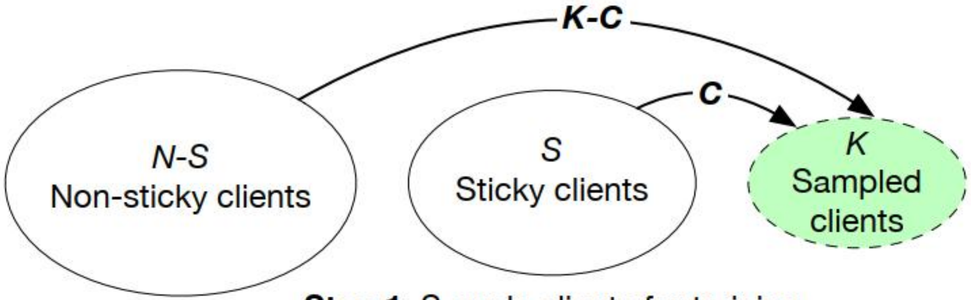
- In each round, the server samples clients from sticky clients and non-sticky clients for training
  - Sticky clients have **higher probabilities** to be sampled than non-sticky clients



Step 1: Sample clients for training

GlueFL - Sticky Sampling

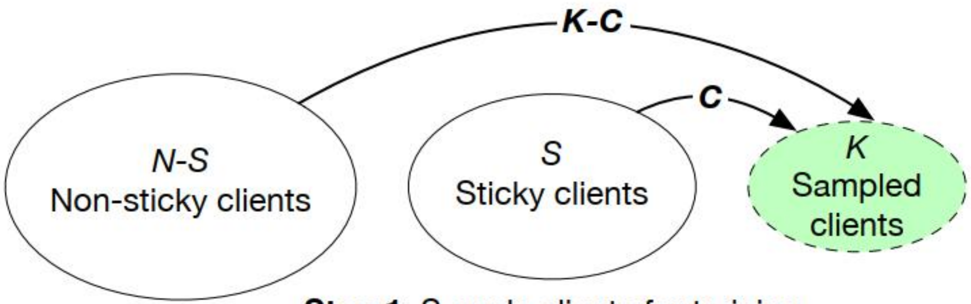
- In each round, the server samples clients from sticky clients and non-sticky clients for training
  - Sticky clients have **higher probabilities** to be sampled than non-sticky clients
  - Example
    - N=3000 clients, K=30 sampled clients (uniform sampling - skip **100 rounds**)



Step 1: Sample clients for training

GlueFL - Sticky Sampling

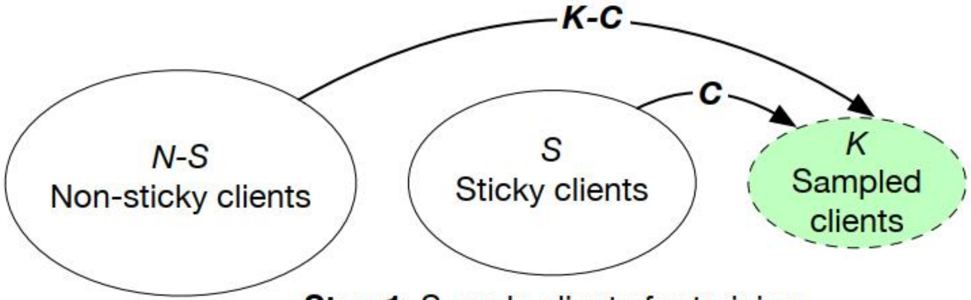
- In each round, the server samples clients from sticky clients and non-sticky clients for training
    - Sticky clients have **higher probabilities** to be sampled than non-sticky clients
    - Example
        - N=3000 clients, K=30 sampled clients (uniform sampling - skip **100 rounds**)
        - S=120 sticky clients, C=24 sticky clients being sampled (sticky clients skip **5 rounds**)



Step 1: Sample clients for training

GlueFL - Sticky Sampling

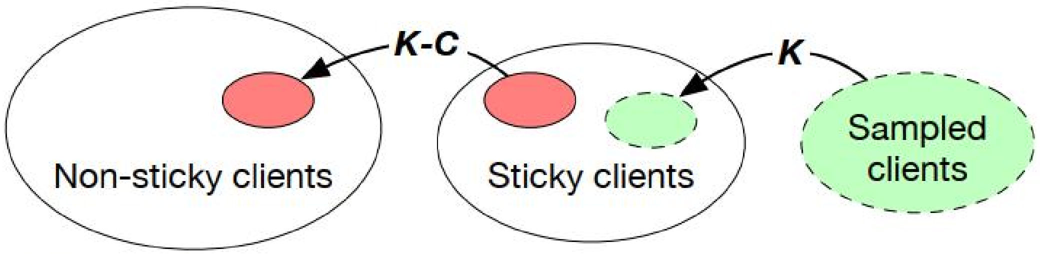
- In each round, the server samples clients from sticky clients and non-sticky clients for training
  - Sticky clients have **higher probabilities** to be sampled than non-sticky clients
  - Example
    - N=3000 clients, K=30 sampled clients (uniform sampling - skip **100 rounds**)
    - S=120 sticky clients, C=24 sticky clients being sampled (sticky clients skip **5 rounds**)
    - Non-sticky clients skips **480 rounds**



Step 1: Sample clients for training

GlueFL - Sticky Sampling

- To update sticky group (formed by sticky clients)
  - Some sticky clients will be randomly selected and marked as non-sticky clients
  - Newly sampled clients will be marked as sticky clients
- Size of the sticky group is **constant**



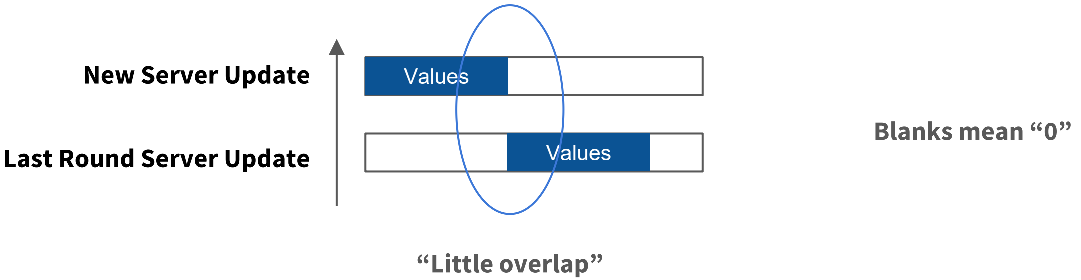
Step 2: Rebalance non-sticky and sticky groups with sampled clients

GlueFL - Mask Shifting

- Sticky sampling is **not** sufficient
  - With sticky masking, a sticky client may skip 5 rounds
  - With *q*=10% mask ratio (server updates 10% model in each round), the client still needs to download **50% global model** in the worst case

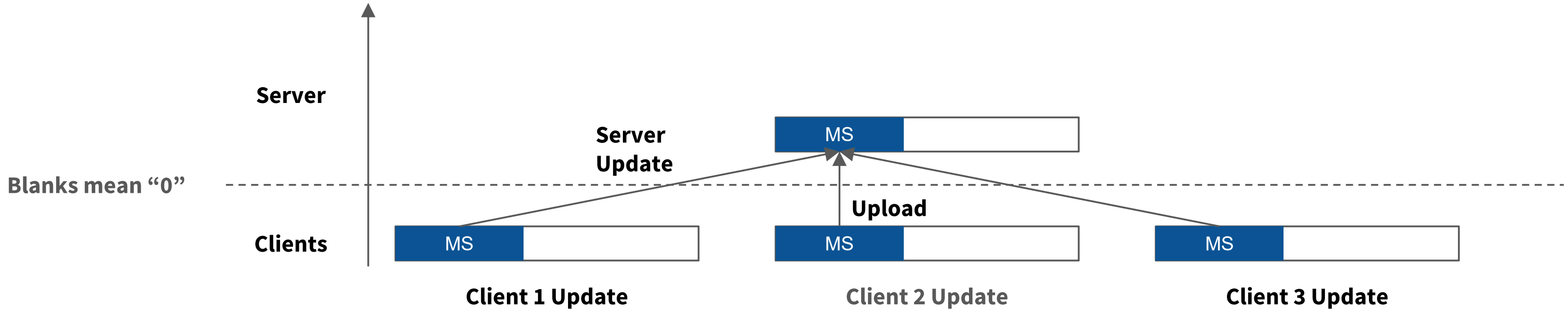
GlueFL - Mask Shifting

- Sticky sampling is **not** sufficient
  - With sticky masking, a sticky client may skip 5 rounds
  - With *q*=10% mask ratio (server updates 10% model in each round), the client still needs to download **50% global model** in the worst case
- In existing masking strategies, the server updates of two successive rounds have **little** overlap. Mask shifting **increases** this overlap



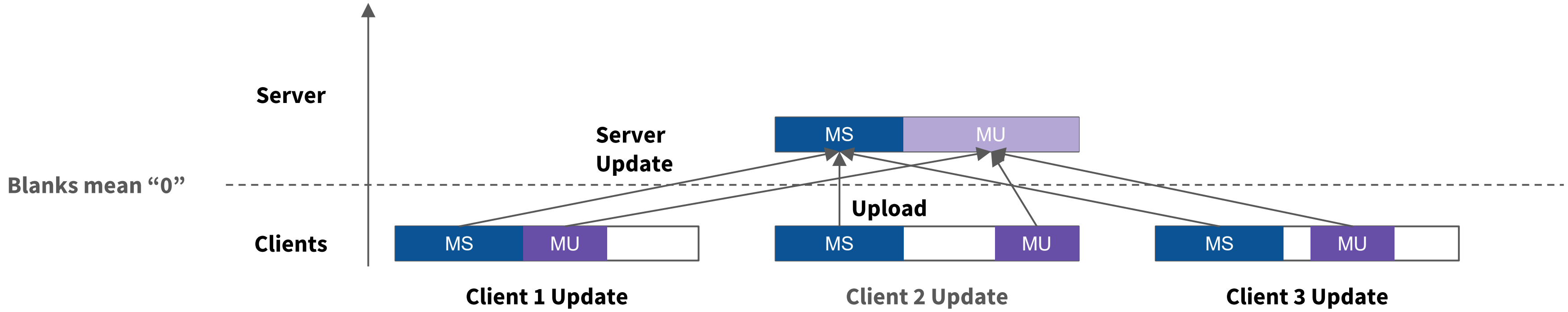
GlueFL - Mask Shifting

- In mask shifting, server maintains a **shared mask (MS)** (e.g., q_{shr}=9% and q=10%)
    1. Each client uploads update values covered by MS (9%)



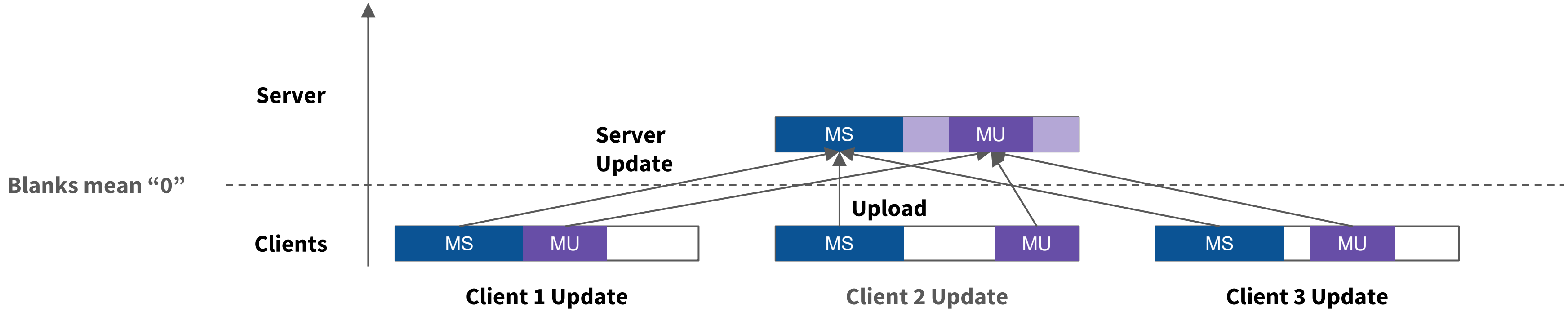
GlueFL - Mask Shifting

- In mask shifting, server maintains a **shared mask (MS)** (e.g., q_{shr}=9% and q=10%)
  1. Each client uploads update values covered by MS (9%)
  2. Each client uploads largest update values not covered by MS. We say these values are covered by **unique masks (MU)** (q-q_{shr}=1%)
     - Now the server has **values** covered by MUs from different clients



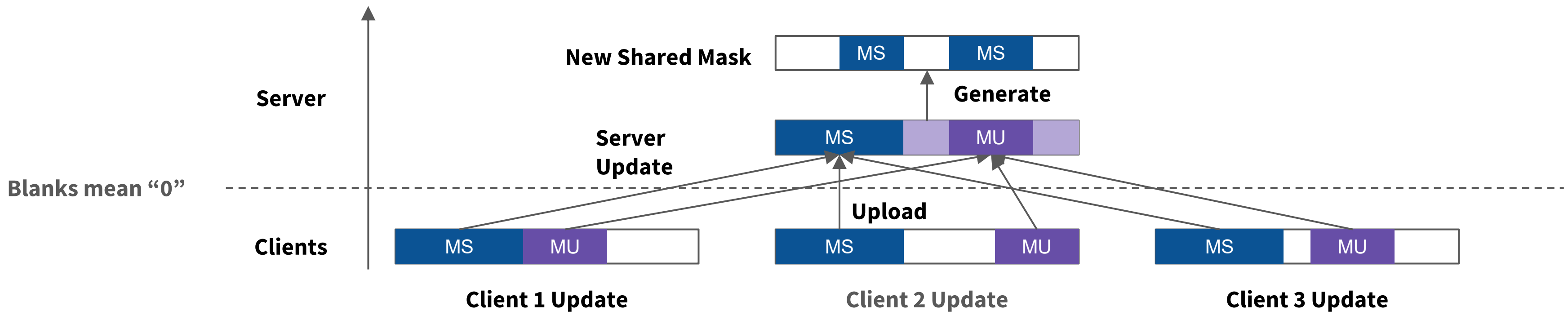
GlueFL - Mask Shifting

- In mask shifting, server maintains a **shared mask (MS)** (e.g., q_{shr}=9% and q=10%)
  1. Each client uploads update values covered by MS (9%)
  2. Each client uploads largest update values not covered by MS. We say these values are covered by **unique masks (MU)** (q-q_{shr}=1%)
  3. The server uses MS and **largest** values (1%) in MUs to update model (9%+1%=10%)



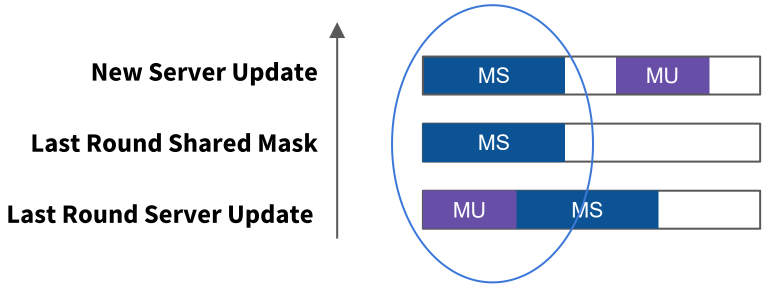
GlueFL - Mask Shifting

- In mask shifting, server maintains a **shared mask (MS)** (e.g., q_{shr}=9% and q=10%)
  1. Each client uploads update values covered by MS (9%)
  2. Each client uploads largest update values not covered by MS. We say these values are covered by **unique masks (MU)** (q-q_{shr}=1%)
  3. The server uses MS and **largest** values (1%) in MUs to update model (9%+1%=10%)
  4. The server generate the new shared mask (9%) by selecting largest values (9%) in last server update



GlueFL - Mask Shifting

- With a 9% shared mask and 1% unique masks, the server updates of two successive rounds have at least 9% overlap
- Example
  - Suppose *S=120* and *C=24*, a sticky client may skip 5 rounds
  - If the *q=10%* (server updates 10% model in each round), the client only needs to download (10% + 1% * 4) = 14% global model after 5 rounds with mask shifting



GlueFL - Other Techniques

- Shared Mask Regeneration (See Full Paper)
  - If the updates are changing dramatically in some rounds, shifting the shared mask using a **small** unique mask will lead to a slow convergence speed
  - Every *I* rounds, GlueFL regenerates the shared mask with a 0% shared mask and *q%* unique masks
- Error-Compensation (See Full Paper)
  - Clients remember local compression error and add them to the next round update
  - GlueFL is compatible with error compensation
  - With sticky sampling, compensation vectors need to be reweighted

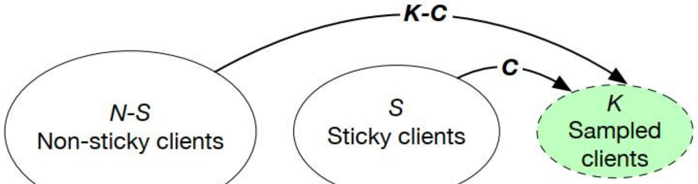
Agenda

- Background and Motivation
    - Federated learning (FL)
    - Masking in FL
- GlueFL Framework Design
    - Sticky sampling
    - Mask shifting
- Experiment Results
- Conclusion

Experiment Results

- Three models on three public datasets
  - FEMNIST - ShuffleNet, MobileNet
  - OpenImage - ShuffleNet, MobileNet
  - Google Speech - ResNet-34
- Three baselines: FedAvg, STC and APF
- User-defined parameters in GlueFL are set the best values (e.g., *S*, *C*)

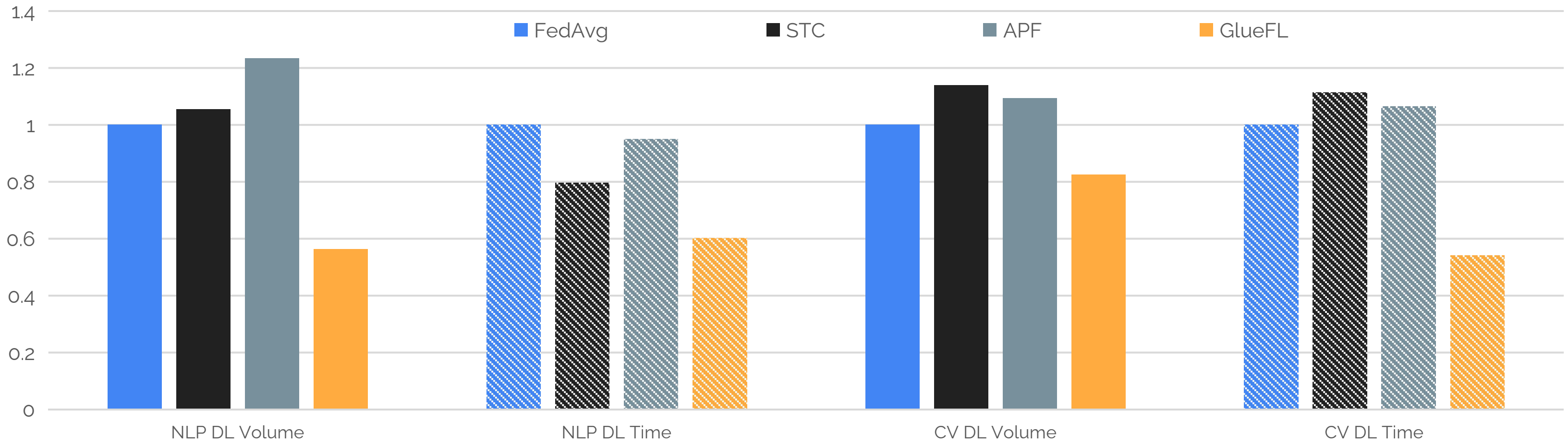
Name	**N (Total number of clients)**	**K (Number of sampled clients)**
FEMNIST	2,800	30
OpenImage	10,625	100
Google Speech	2,066	30



Step 1: Sample clients for training

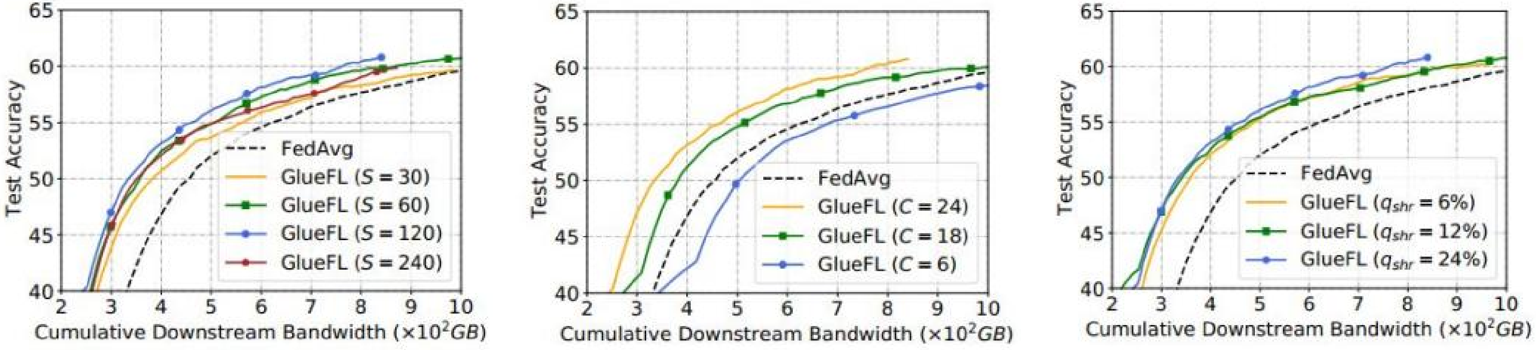
Experiment Results

- *To reach the same target performance, GlueFL needs **significantly less downstream bandwidth (DL Volume) and time (DL Time)** for CV and NLP tasks on average*



Experiment Results

- *With most hyperparameter choices (S - sticky group size, C - # sticky clients, q_{shr} - shared mask size), GlueFL outperforms FedAvg, showing its **robustness***



Conclusion

- Traditional masking strategies fail to save much downstream bandwidth
    - Downstream bandwidth increases because client local model states become stale
- We present an FL framework called **GlueFL** that combines masking with client sampling to reduce downstream bandwidth
    - Sticky sampling - prioritize the most recently used clients
    - Mask shifting - ensure consecutive central model updates share a large number of changed parameters
- We evaluate GlueFL on three public datasets. On average, GlueFL spends 29% less training time with a 27% less downstream bandwidth overhead as compared to FedAvg, STC and APF

Step 1: Sample clients for training

Step 2: Rebalance non-sticky and sticky groups with sampled clients