time to the limited aggregation interval T_k by adaptively adjusting the workload concerning its real-time availability per communication round. The workload is defined as the product of the partial training ratio α and the local epoch number E. `TimelyFL` framework is composed of three main parts, `TimelyFL` server, local time update, and workload scheduling.

Algorithm 1: TimelyFL.

```
   Input: $k$: the aggregation participation target, $n$:
          the number of training concurrency
1  for $r \in \{0, \cdots, R-1\}$ communication rounds do
2      Global server do:
3          Sample $n$ clients uniformly at random to
           define $\mathcal{S}$, and send $W_s^r$ to clients in $\mathcal{S}$ ;
4      Clients $c \in \mathcal{S}$ in parallel do:
5          $\tilde{t}_{total}, \tilde{t}_{cmp}, \tilde{t}_{com}$ = LocalTimeUpdate$(M)$ ;
6      Global server do:
7          $T_k^r \leftarrow$ the $k$th smallest number in $\langle \tilde{t}_{total} \rangle$ ;
8          $\langle E^r \rangle, \langle \alpha^r \rangle, \langle t_{rpt}^r \rangle$ =
           WorkloadScheduling$({T_k}^r, \langle \tilde{t}_{cmp} \rangle, \langle \tilde{t}_{com} \rangle)$ ;
9      Clients $c \in \mathcal{S}$ in parallel do:
10         $W_c^r \leftarrow$ adaptive model training ;
11     Global server do:
12         $W_s^{r+1} \leftarrow$ aggregate $\langle W_c^r \rangle$ ;
13 end
   Output: $W_s^R$
```

`TimelyFL` Server. `TimelyFL` server is in charge of adjusting the aggregation interval T_k, local training epoch E, and partial training ratio α for each device during the FL training, as summarized in Algorithm 1. The aggregation interval T_k in each round equals the kth smallest value among $\langle \tilde{t}_{total} \rangle$, as the estimated unit total time for all clients. At each communication round, `TimelyFL` server randomly samples n clients to construct the collection $\mathcal{S}$ and distributes the global model to the clients inside $\mathcal{S}$, which means n clients would start the local training in this round, same as the definition of training concurrency in the `FedBuff`. Each selected client would perform one data batch full model training to estimate its time consumption and report it to the server. Then, aggregation interval time T_k and training hyperparameters for client c (i.e., local training epoch number E and partial training ratio α) would be adjusted based on all selected clients' status during the FL training process. The server would also return a local computation budget time $t_{rpt,c}$, as the wall-clock time when the client must report its training status.

Local Time Update. To efficiently accommodate the capabilities, each participant needs to update its time consumption to the server as summarized in Algorithm 2. Specifically, each client would collect the real computation time t_{cmp} from one data batch full model training. The unit computation time $\tilde{t}_{cmp}$ is estimated by t_{cmp} and progress β, where β is defined as the ratio of trained batch number to the total data batch number. The local communication time equals the model's file size M over the device's real-time network bandwidth Bw, as the same setting in the previous FL system work [14].

Algorithm 2: Local Time Update.

```
   Input: $M$: the file size of the received global model,
          $Bw$: the real-time network bandwidth
1  $t_{cmp}, \beta \leftarrow$ one data batch training ;
2  $\tilde{t}_{com} = M/Bw$ ;
3  $\tilde{t}_{cmp} = t_{cmp}/\beta$ ;
4  $\tilde{t}_{total} = \tilde{t}_{cmp} + \tilde{t}_{com}$ ;
   Output: $\tilde{t}_{total}, \tilde{t}_{com}, \tilde{t}_{cmp}$
```

Workload Scheduling. `TimelyFL` would adjust the local epoch number E and partial training ratio α for each client in every communication round based on the estimated $\tilde{t}_{com,c}, \tilde{t}_{cmp,c}$ and aggregation interval T_k, as the relationship shown in 1. If one's unit total time is smaller than T_k, then the server would try to maximize its local training utility and minimize the idle time, as to assign more than one local epoch training for the next round. Otherwise, the server would assign less amount of workload to them by decreasing the model training ratio α, which guarantees they can finish at least one local epoch training within the report time $t_{rpt,c}$ and catch up the global aggregation timely. We summarized the scheduler as Algorithm 3.

Algorithm 3: Workload Scheduling.

```
   Input: $T_k$: aggregation interval time, $\langle \tilde{t}_{cmp} \rangle$: unit
          computation time, $\langle \tilde{t}_{com} \rangle$: unit
          communication time
1  for each client $c \in \mathcal{S}$ in parallel do
2      $E_c = \max(\lfloor (T_k - \tilde{t}_{com,c})/\tilde{t}_{cmp,c} \rfloor, 1)$ ;
3      $\alpha_c = \min(T_k/(\tilde{t}_{com,c} + \tilde{t}_{cmp,c}), 1)$ ;
4      $t_{rpt,c} = T_k - \tilde{t}_{com,c} \times \alpha_c$ ;
5  end
   Output: $\langle E \rangle, \langle \alpha \rangle, \langle t_{rpt} \rangle$
```

4. Experiment

4.1. Experimental Settings

Datasets, Models, and Tasks. To demonstrate `TimelyFL`'s effectiveness across tasks, we evaluate `TimelyFL` on three benchmark datasets from various categories of FL applications: