FedFrame: a secure FL framework

Presenter：Dai Yuqi

Date：2023/12/15

CONTENT

01. What is FedFrame?

02. Why FedFrame?

03. Workflow&Puzzles

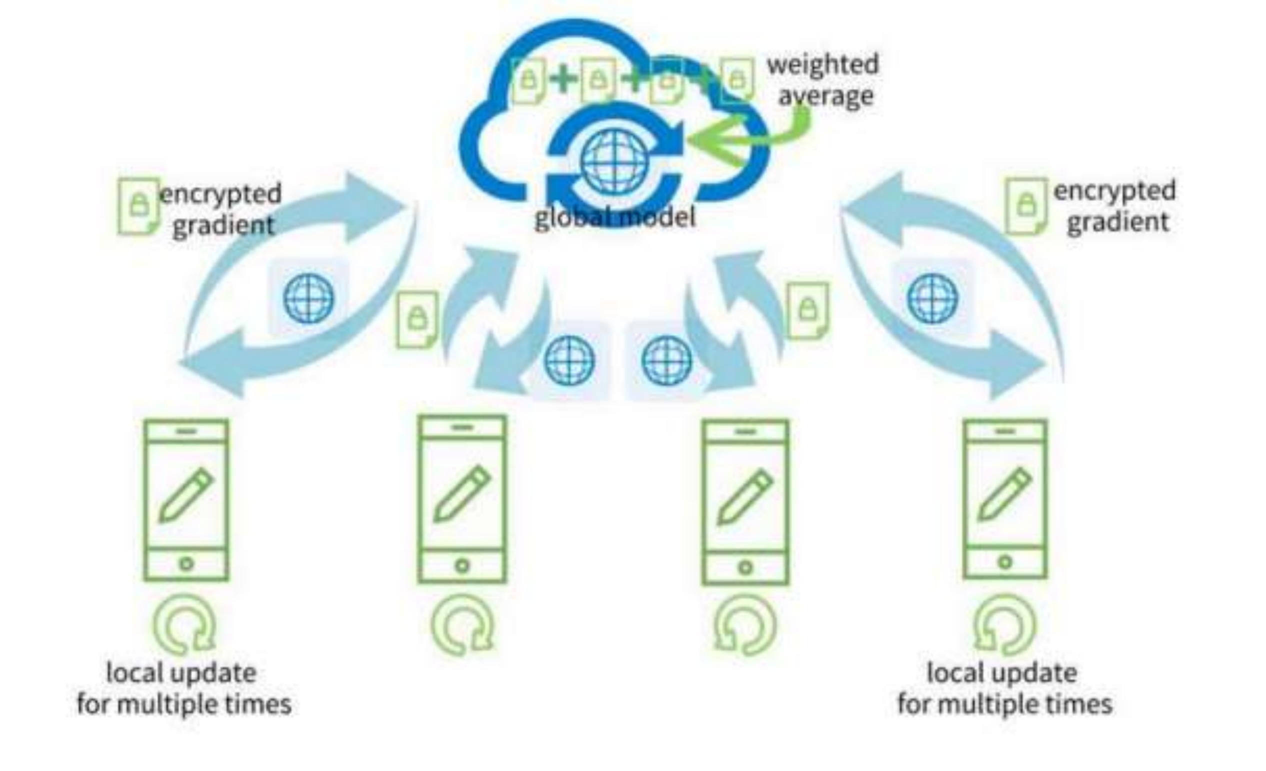
01. What is FedFrame?

- ✓ What is federated learning?

Federated Learning: Collaborative Machine Learning without Centralized Training Data

Aiming: Prevent data leakage from centralized training.

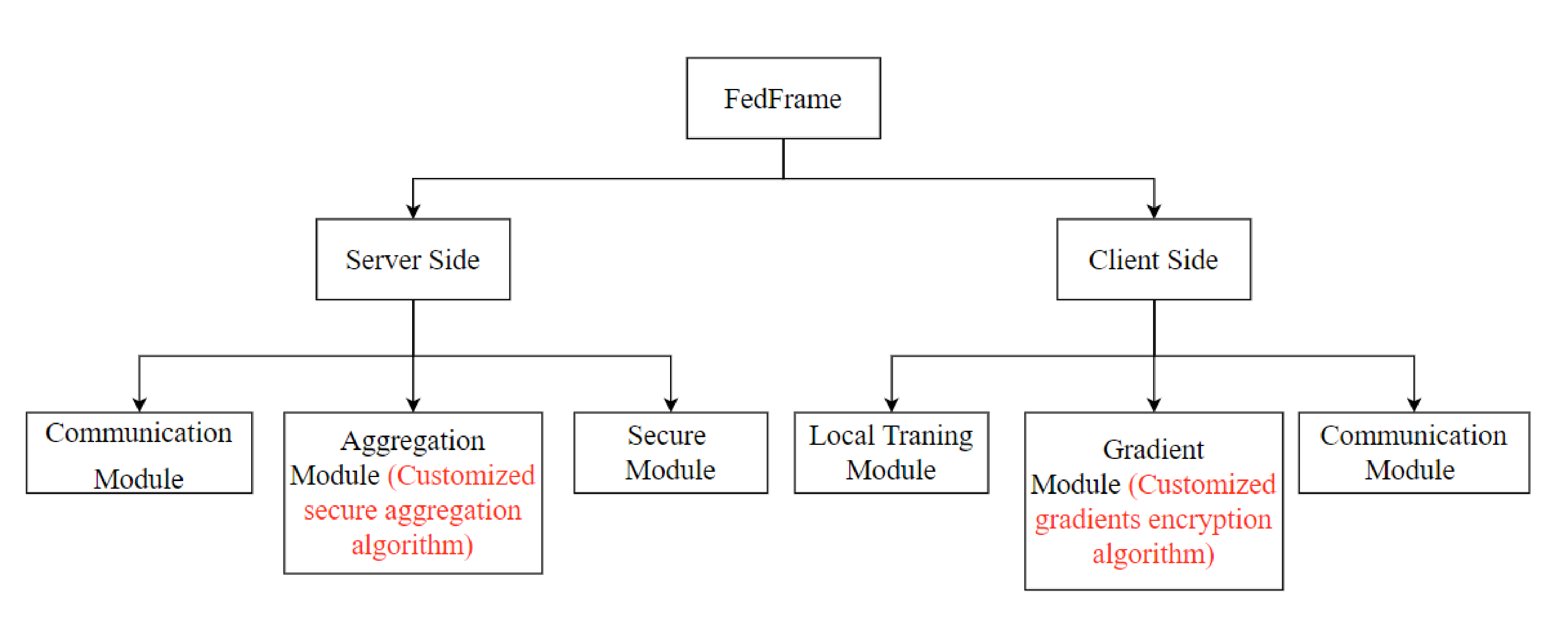
Vulnerabilities: Adversaries can deduce the private training data by global gradients, or hamper global model accuracy by sending malicious local gradients.



Img 2. Diagram of Secure FL

01. What is FedFrame?

FedFrame is a scalable secure federated learning framework which is originally designed to resist model poisoning attack. FedFrame allows users to combine any related secure functions with it to implement a more customized secure FL framework.


FedFrame
Server Side
Client Side
Communication Module
Aggregation Module (Customized secure aggregation algorithm)
Secure Module
Local Traning Module
Gradient Module (Customized gradients encryption algorithm)
Communication Module

Img 1. Diagram of FedFrame

02. Why FedFrame?

✓ Why we need FedFrame?

In FedFrame, I propose and implement a secure aggregation algorithm to distinguish modified maliciou data.

The core principle of this algorithm is that the global gradient change rate is always maintained within a reasonable range. If the modified malicious data couldn’t change the global gradient greatly, the effect of it could be ignored.

Let’s say global gradient is a (i rows, j columns) matrix.

- global gradient last time: $gg_{i\times j}^{n-1}$
- global gradient second last time: $gg_{i\times j}^{n-2}$
- Local gradient from benign client 0: $lg_{i\times j}^{0}$
- Local gradient from malicious client 1: $lg_{i\times j}^{1}$

→

Direction of global gradients changing:

$$d_{(i,j)}^{n-2} = \begin{cases} if\ gg_{(i,j)}^{n-1} - gg_{(i,j)}^{n-2} > 0, & is\ 1 \\ if\ gg_{(i,j)}^{n-1} - gg_{(i,j)}^{n-2} = 0, & is\ 0 \\ if\ gg_{(i,j)}^{n-1} - gg_{(i,j)}^{n-2} < 0, & is\ -1 \end{cases}$$

Direction of local gradient:

$$d_{lg}{}_{(i,j)}^{(0\ or\ 1)} = \begin{cases} if\, d_{lg}{}_{(i,j)}^{(0\ or\ 1)} - gg_{i\times j}^{n-1} > 0, & is\ 1 \\ if\, d_{lg}{}_{(i,j)}^{(0\ or\ 1)} - gg_{i\times j}^{n-1} = 0, & is\ 0 \\ if\, d_{lg}{}_{(i,j)}^{(0\ or\ 1)} - gg_{i\times j}^{n-1} < 0, & is\ -1 \end{cases}$$

02. Why FedFrame?

✓ Why we need FedFrame?

Direction of global gradients changing:

$$d^{n-2}_{(i,j)} = \begin{cases} if\ gg^{n-1}_{(i,j)} - gg^{n-2}_{(i,j)} > 0, & is\ 1 \\ if\ gg^{n-1}_{(i,j)} - gg^{n-2}_{(i,j)} = 0, & is\ 0 \\ if\ gg^{n-1}_{(i,j)} - gg^{n-2}_{(i,j)} < 0, & is\ -1 \end{cases}$$

Direction of local gradient:

$$d_{lg}{}^{(0\ or\ 1)}_{(i,j)} = \begin{cases} if\, d_{lg}{}^{(0\ or\ 1)}_{(i,j)} - gg^{n-1}_{i\times j} > 0, & is\ 1 \\ if\, d_{lg}{}^{(0\ or\ 1)}_{(i,j)} - gg^{n-1}_{i\times j} = 0, & is\ 0 \\ if\, d_{lg}{}^{(0\ or\ 1)}_{(i,j)} - gg^{n-1}_{i\times j} < 0, & is\ -1 \end{cases}$$

✓ Direction difference between local and global gradient:

$$\Delta\, d_{lg}{}^{(0\ or\ 1)}_{(i,j)} = \begin{cases} if\ d_{lg}{}^{(0\ or\ 1)}_{(i,j)} = d^{n-2}_{(i,j)}, & is\ 1 \\ if\ d_{lg}{}^{(0\ or\ 1)}_{(i,j)}\ !=d^{n-2}_{(i,j)}, & is\ -1 \end{cases}$$

✓ Global gradient change rate:

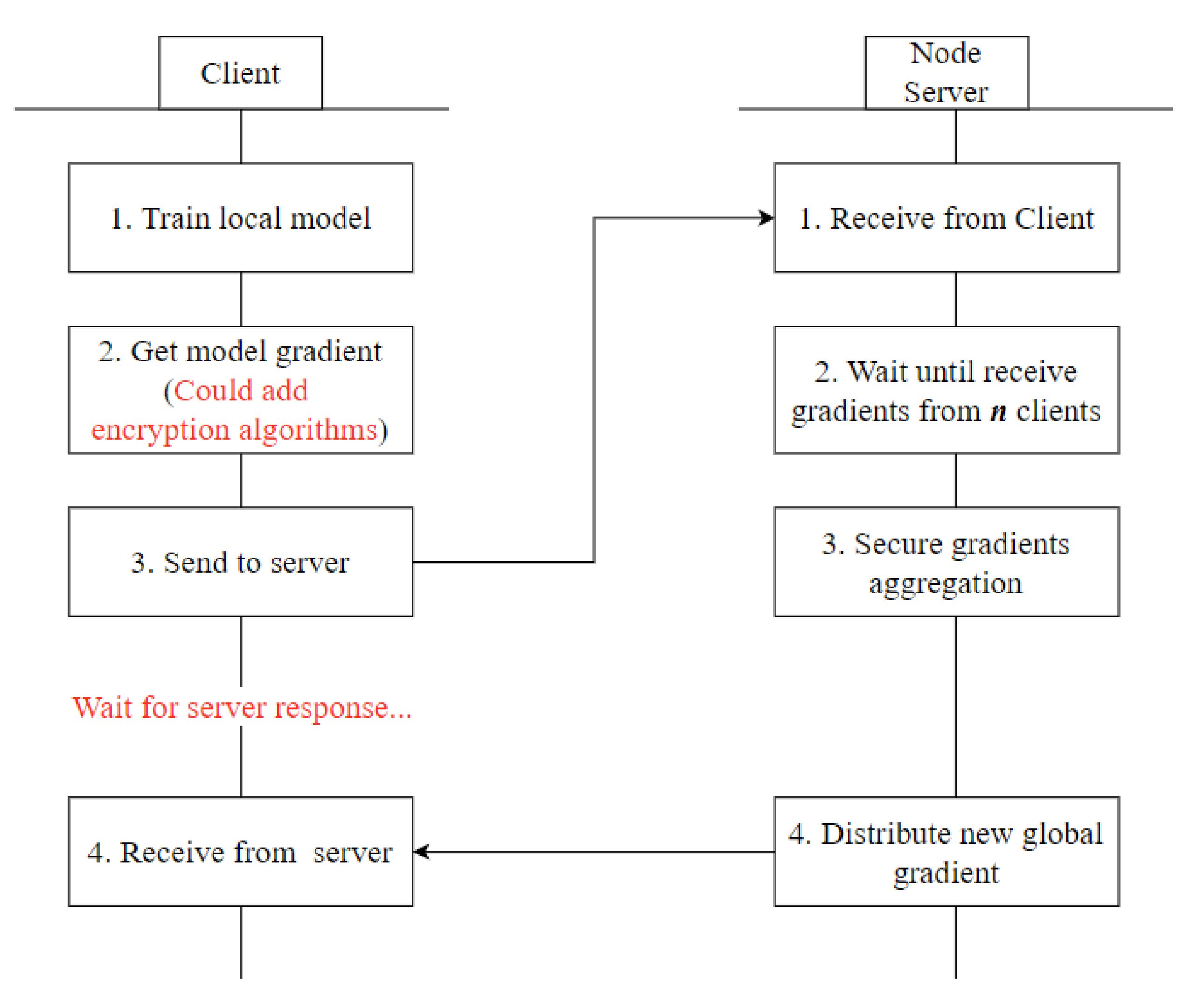
$$\Delta rate_{(i,j)} = (lg^{(0\ or\ 1)}_{(i,j)})^2 \times \Delta\, d_{lg}{}^{(0\ or\ 1)}_{(i,j)}$$

$$\text{Global gradient change rate} = \sum_{i=0}^{i} \sum_{j=0}^{j} \Delta rate_{(i,j)}$$

03. Workflow&Puzzles

✓ Workflow of FedFrame

Situation One: Node server collects the gradients from edge clients, calculate global gradients then distribute global gradients to clients.

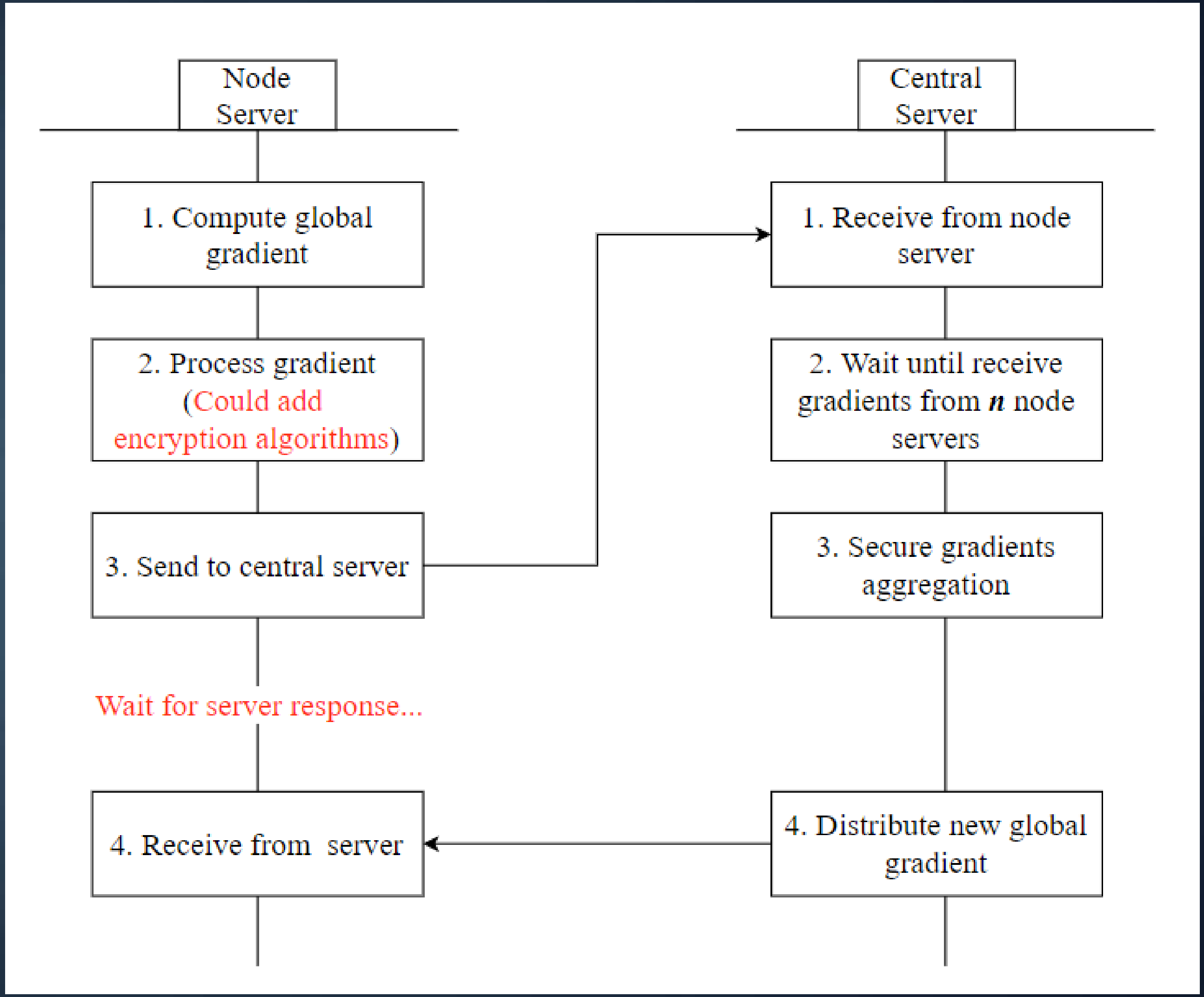

Client
Node Server
1. Train local model
2. Get model gradient (Could add encryption algorithms)
3. Send to server
Wait for server response...
4. Receive from server
1. Receive from Client
2. Wait until receive gradients from *n* clients
3. Secure gradients aggregation
4. Distribute new global gradient

Img 4. Workflow of FedFrame situation one

03. Workflow&Puzzles

✓ Workflow of FedFrame

Situation Two: Node servers send partial global gradient to central server. Central server compute a global gradient and distribute it to the n node servers which participates the gradients aggregation process.



Node Server
1. Compute global gradient
2. Process gradient (Could add encryption algorithms)
3. Send to central server
Wait for server response...
4. Receive from server
Central Server
1. Receive from node server
2. Wait until receive gradients from *n* node servers
3. Secure gradients aggregation
4. Distribute new global gradient

Img 5. Workflow of FedFrame situation two

03. Workflow&Puzzles

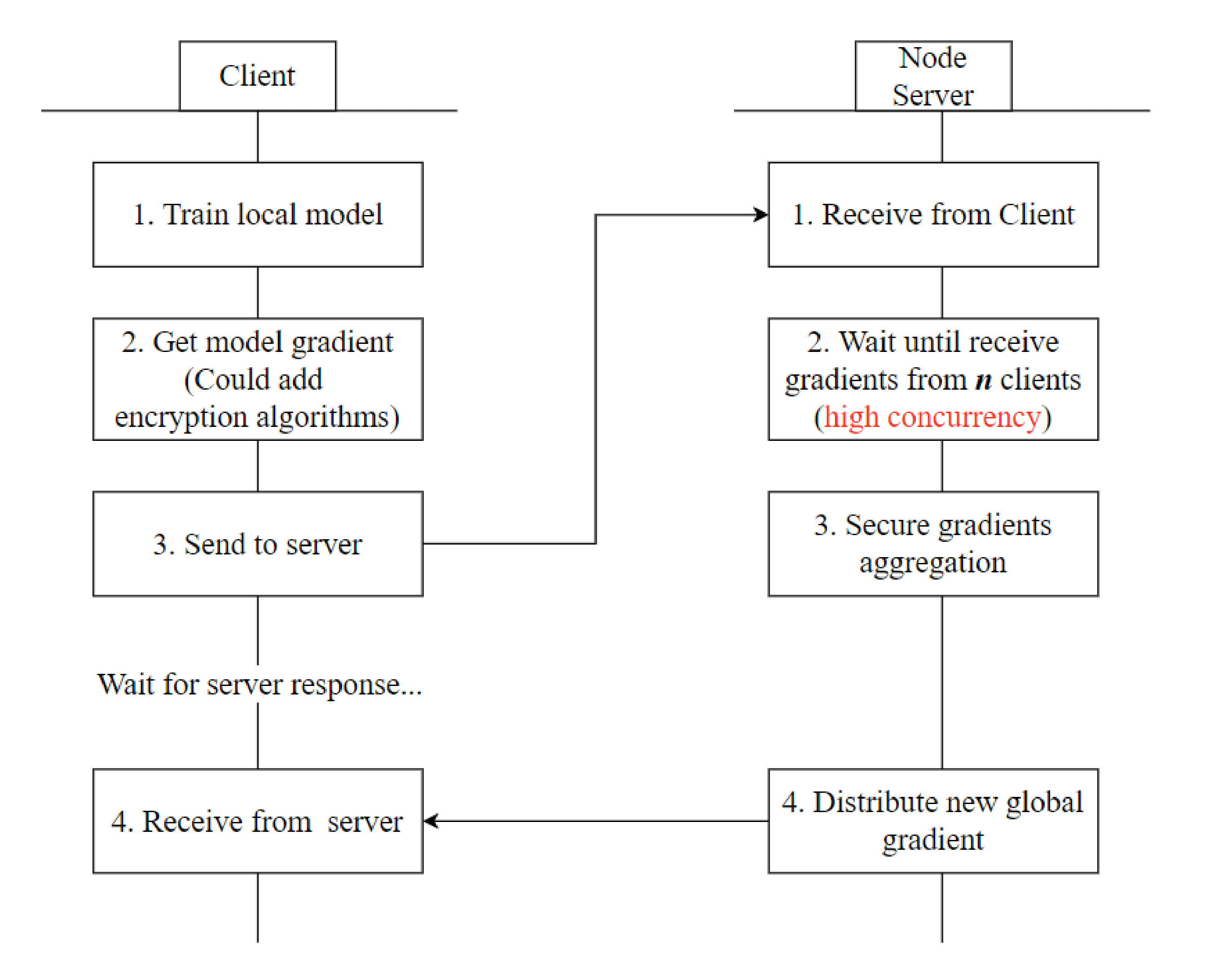
Puzzles

Puzzle One: High concurrency

In a large training system, there are a lot of clients sending local gradients all the time. How to handle all of the requests?

Solution 1:
Cache simulation.

Solution 2:
Multithreading.


Client
Node Server
1. Train local model
2. Get model gradient (Could add encryption algorithms)
3. Send to server
Wait for server response...
4. Receive from server
1. Receive from Client
2. Wait until receive gradients from *n* clients (high concurrency)
3. Secure gradients aggregation
4. Distribute new global gradient

Img 6. high concurrency in situation one

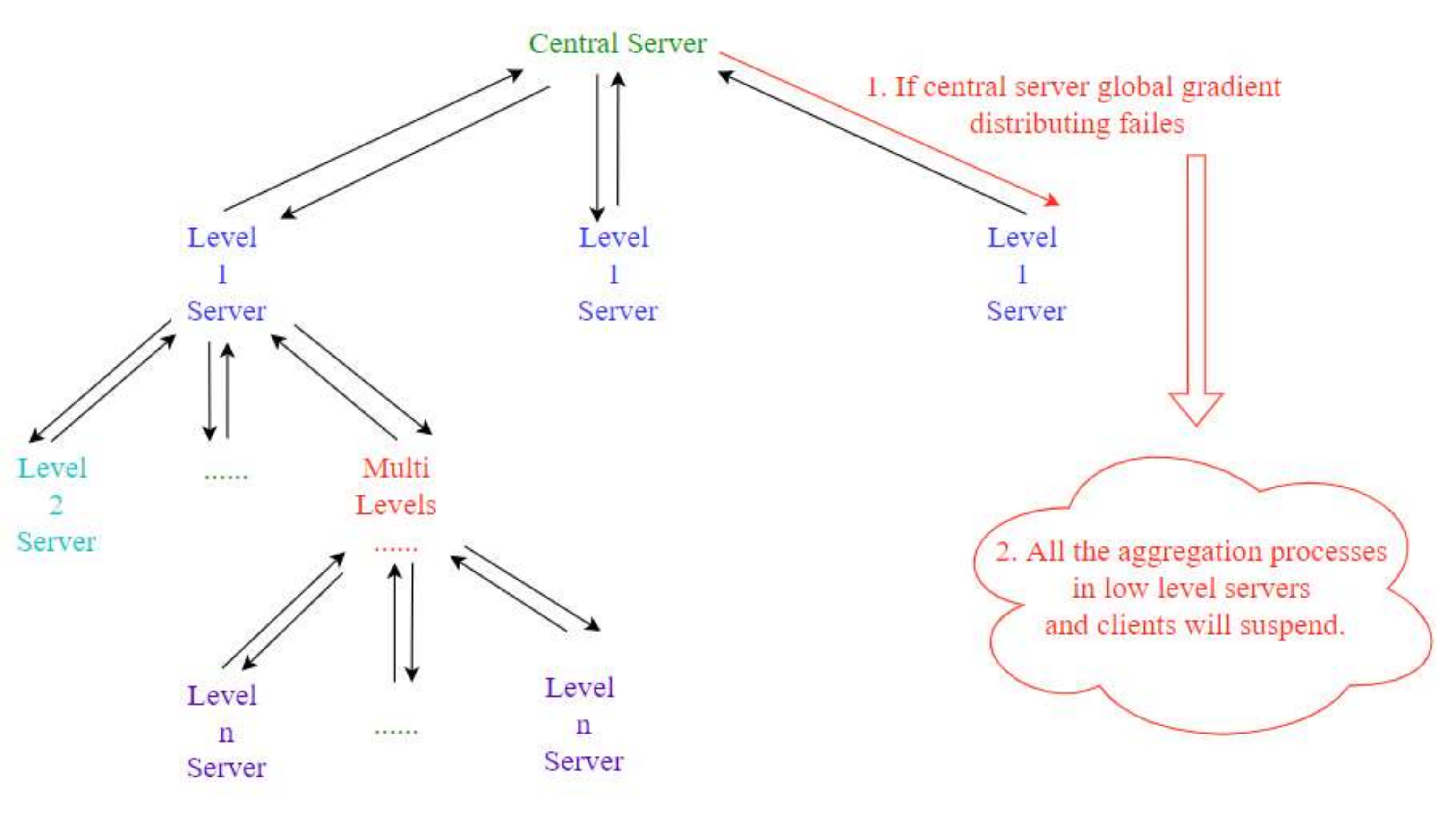
03. Workflow&Puzzles

✓ Puzzles

Puzzle Two: Complex FedFrame network

If central server fails to send global gradient to one node server, the low level servers and clients will suspend.

If clients fails to send local gradient to node server, the high level servers will suspend.


Central Server
1. If central server global gradient distributing failes
Level 1 Server
Level 1 Server
Level 1 Server
Level 2 Server
......
Multi Levels ......
2. All the aggregation processes in low level servers and clients will suspend.
Level n Server
......
Level n Server

Img 7. Diagram of an issue in complex FedFrame network

Thank you