# 🧠 FedTS Enhancements: Federated Learning Simulation Report

---

## 🎯 Goal

The goal of this script is to **simulate a dynamic and robust Federated Learning (FL)** environment that supports:

- Comparison between **Thompson Sampling (Fed-TS)** and **Random client selection**
- Integration of **malicious clients** (via label-flipping attacks)
- Use of complex datasets like **CIFAR-10**
- **Tracking and visualizing** model performance over rounds
- Enhancing trust-based **client participation** in FL systems
- Optional **model checkpoint saving** for reproducibility

---

## 🧩 Problem Statement

Traditional Federated Learning assumes that all participating clients behave honestly and contribute fairly. However, in real-world scenarios:

- Some clients may behave **maliciously** (e.g., label-flipping attacks).
- Client reliability may vary across training rounds.
- Static client participation doesn't reflect practical deployments.

Hence, we simulate:

- **Dynamic client participation** with malicious clients
- **Trust-aware selection mechanism** using **Fed-TS**

- Compare it against naive **random client selection**

---

## ✅ What This Script Does

1. **Dataset Preparation**
    - Loads **CIFAR-10** (or MNIST if toggled) with `torchvision`.
    - Partitions dataset among `CLIENTS_TOTAL` clients non-IID style.
    - Adds **label-flipping attackers** among dynamically inserted clients.
2. **Client Simulation**
    - `CLIENTS_ORIGINAL` are the fixed set.
    - After `INSERT_NEW_AT` rounds, dynamic clients are added using **Thompson Sampling** based on trust scores.
3. **Model Definitions**
    - Implements both **SimpleNN** and **SimpleCNN**.
    - Chooses CNN for CIFAR-10 (due to image complexity).
4. **Training Procedure**
    - Each round, clients train locally and send updated models.
    - Global model is updated by **averaging** local weights.
    - Two separate global models:
        - One for **Fed-TS selection**
        - One for **Random selection** (baseline)
5. **Drift Detection**

   - Measures the **weight update distance** (`get_path_drift`) of each client.
   - Uses **KMeans clustering** to define a threshold for suspicious updates.
   - Updates **Beta distributions** (success/failure counts) to influence future client selection.

6. **Visualization**
   - Plots accuracy over time for both Fed-TS and Random.
   - Visualizes **per-client accuracy** trend for deeper insight.

---

## 🚀 Technologies Used

| Component | Tool/Library |
| --- | --- |
| Deep Learning | PyTorch (`torch`) |
| Dataset | CIFAR-10 / MNIST |
| Federated Partitioning | Custom + `Subset` |
| Visualization | `matplotlib` |
| Sampling Algorithm | **Thompson Sampling** |
| Clustering | `sklearn.cluster.KMeans` |
| Client Behavior Attack | Label-Flipping Strategy |
| Persistence (Optional) | `torch.save()` |

---

## ⚙️ Key Configuration

| Parameter | Value |
| --- | --- |
| `ROUND_TOTAL` | 20 |
| `CLIENTS_ORIGINAL` | 5 |

| | |
|---|---|
| `CLIENTS_DYNAMIC` | 5 |
| `USE_CIFAR` | `True` |
| `ATTACK_LABEL_FLIPPING` | `True` |
| `SAVE_MODEL` | `True` *(optional)* |

---

## 🤖 Why Thompson Sampling?

**Problem**: Malicious or unproductive clients degrade model performance if selected blindly.

**Solution**: Use **Thompson Sampling**, a **Bayesian multi-armed bandit** approach that:

- Scores each client based on success/failure (drift)
- Samples from Beta distribution to **balance exploration vs exploitation**
- Reduces participation of unreliable clients

---

## 📊 What Did We Learn?

- **Fed-TS significantly outperforms Random** selection in the presence of attackers.
- Thompson Sampling enables **adaptive trust-based selection**.
- The system is **scalable and dataset-agnostic**, supporting CIFAR-10 and MNIST.
- Visualizations clearly demonstrate the **robustness** of trust-aware client filtering.