Experiment No 4

AIM

To design and deploy a Fog Computing architecture using the **iFogSim** simulator.

THEORY

1. Fog Computing

Fog Computing extends Cloud Computing by placing **processing and storage resources closer to the data source** (e.g., sensors, cameras, IoT devices).

- **Goal**: Reduce **latency**, **network congestion**, and **cloud dependency**.
- **Use Cases**: Smart surveillance, healthcare monitoring, industrial IoT, real-time traffic control.
- **Advantages**:
  - Faster response for delay-sensitive tasks.
  - Efficient bandwidth usage.
  - Balanced load between edge and cloud.

---

2. iFogSim

iFogSim is a simulation toolkit built on top of **CloudSim** for modeling Fog and Edge environments. It enables researchers to:

- Design layered IoT–Fog–Cloud architectures.
- Model applications as **dataflow graphs**.
- Simulate **latency**, **network usage**, and **energy consumption**.
- Evaluate **module placement strategies** (Cloud vs Edge vs Hybrid).

---

3. Case Study: Smart Surveillance

In a traditional setup, video streams from smart cameras are processed in the **cloud**, causing latency and bandwidth overhead. Fog-based surveillance solves this by **processing motion detection and tracking at edge devices**, reducing delay and enabling real-time responses (like camera rotation).

---

4. Structure of the Simulation

Physical Devices Created

- **Cloud** → High-powered central server (most distant, highest latency).
- **Proxy Server** → Acts as an intermediate between cloud and edge.
- **Routers** → Local gateways managing edge devices.
- **Smart Cameras** → Edge IoT devices with sensors and actuators.

Each device has specific **CPU (MIPS), RAM, bandwidth, storage, and power usage**.

---

5. Main Components

1. **Main Method**
   - Initializes CloudSim/iFogSim.
   - Creates a **broker** (simulation user).
   - Builds the surveillance application.
   - Creates Fog/Edge/Cloud devices.
   - Maps modules to devices.
   - Starts simulation.
2. **createApplication(...)**
   Defines application modules:
   - **motion_detector** – detects movement.
   - **object_detector** – identifies objects when motion occurs.
   - **object_tracker** – tracks detected objects.
   - **user_interface** – displays data to the user.
   - **CAMERA** – sensor sending video feed.
   - **PTZ_CONTROL** – actuator controlling camera orientation.

Application Loops:

- `motion_detector → object_detector → object_tracker`
- `object_tracker → PTZ_CONTROL`

3. **createFogDevices(...)**
   Builds the hierarchy of devices: Cloud → Proxy → Routers → Cameras.
4. **addArea(...) & addCamera(...)**
   - Adds routers for each area.
   - Adds cameras with sensors and actuators.
5. **createFogDevice(...)**
   - Generic method to configure CPU, RAM, bandwidth, storage, and power consumption.
6. **ModuleMapping & ModulePlacement**
   - **Motion detection** always runs on cameras (edge).
   - Other modules may run on Fog or Cloud, depending on the simulation setup.
   - If `CLOUD = true` → heavy tasks executed in Cloud.

---

6. Simulation Output

The simulator provides:

- **Latency** measurements.
- **Tuple transfers** (data packets exchanged).
- **Device utilization**.
- **Power consumption** across devices.

CONCLUSION

This experiment successfully demonstrates the **design and deployment of a Fog Computing architecture using iFogSim**.

Key Results:

- **Reduced latency** by processing tasks at edge devices.
- **Lower bandwidth usage** compared to cloud-only solutions.
- **Faster response times** for camera control.
- **Balanced energy consumption** between edge and cloud layers.

By modeling IoT–Fog–Cloud systems, iFogSim enables researchers to **test real-world applications** like smart surveillance in a cost-effective and scalable way.

Future Scope: Integrating **AI/ML techniques** with Fog deployment can enhance predictive analysis and autonomous decision-making in IoT applications.