Aim: To configure OSPF for interconnecting multiple routers and perform troubleshooting to verify and ensure successful routing operation in Cisco Packet Tracer.

- **OSPF (Open Shortest Path First)** is a link-state routing protocol that uses Dijkstra's algorithm to calculate the shortest path.
- OSPF is a classless protocol, supporting Variable Length Subnet Masking (VLSM) and CIDR.
- OSPF organizes routers into **areas**, where **Area 0 (Backbone Area)** is the central area.

Key Features of OSPF:

1. Faster convergence compared to RIP.
2. Supports authentication for secure communication.
3. Multicasts updates to 224.0.0.5 (all OSPF routers).
4. Uses cost as a metric.

Required Components

1. Cisco Packet Tracer software.
2. Devices:
    - 2 or more Cisco Routers.
    - End devices (PCs) for testing connectivity.
    - Switches (if required).
3. Serial or Ethernet cables.

Steps for OSPF Configuration

1. **Network Topology**
    - Create a network with at least two routers.
    - Connect routers using serial or Ethernet links.
    - Assign IP addresses to all interfaces.

    Example IP Plan:

    - Router1:
        - G0/0: `192.168.1.1/24`
        - S0/0/0: `10.0.0.1/30`
    - Router2:
        - G0/0: `192.168.2.1/24`
        - S0/0/0: `10.0.0.2/30`
2. **Basic Router Configuration**
    - Configure router interfaces with IP addresses:

```
bash
Copy code
Router> enable
Router# configure terminal
Router(config)# interface <interface-id>
Router(config-if)# ip address <IP-address> <subnet-mask>
Router(config-if)# no shutdown
Router(config-if)# exit
```

- Repeat for all connected interfaces.

3. **Enable OSPF**
   - Start OSPF configuration in global configuration mode:

```
bash
Copy code
Router(config)# router ospf <process-id>
```

     - The `process-id` is a locally significant identifier (1-65535). It does not need to match across routers.

4. **Add Networks to OSPF**
   - Specify the networks and their associated areas:

```
bash
Copy code
Router(config-router)# network <network-address> <wildcard-
mask> area <area-id>
```

     Example:

```
bash
Copy code
Router(config-router)# network 192.168.1.0 0.0.0.255 area 0
Router(config-router)# network 10.0.0.0 0.0.0.3 area 0
```

     - **Wildcard Mask**: In OSPF, the wildcard mask is the inverse of the subnet mask.

5. **Verify Configuration**
   - Exit configuration mode:

```
bash
Copy code
Router(config-router)# exit
Router(config)# exit
```

6. **Repeat for All Routers**
   - Perform steps 2-5 on all routers in the topology.

Verification

1. **Check OSPF Neighbors**
   - Verify OSPF neighbor relationships:

```
bash
Copy code
Router# show ip ospf neighbor
```

2. **Check OSPF Routing Table**
   - Confirm that OSPF routes are present in the routing table:

```
bash
Copy code
Router# show ip route
```

3. **Check OSPF Process**
   - View the OSPF process and interfaces participating in OSPF:

```
bash
Copy code
Router# show ip ospf
```

4. **Ping Test**
   - Test connectivity between end devices using the `ping` command.

Troubleshooting Steps

1. **No Neighbor Relationships**
   - **Cause**: Incorrect network or wildcard mask, mismatched area IDs, or interface shutdown.
   - **Solution**:
     - Check interface status:

```
bash
Copy code
Router# show ip interface brief
```

     - Verify OSPF neighbors:

```
bash
Copy code
Router# show ip ospf neighbor
```

     - Reconfigure OSPF with correct networks and area IDs.
2. **No OSPF Routes in Routing Table**
   - **Cause**: Missing OSPF configuration or mismatched wildcard masks.
   - **Solution**:
     - Verify OSPF configuration:

```
bash
Copy code
Router# show running-config
```

     - Reconfigure OSPF networks if needed.
3. **Unstable OSPF Network**

- **Cause**: Flapping links or duplicate router IDs.
- **Solution**:
  - Assign a unique router ID manually:

    ```
    bash
    Copy code
    Router(config)# router ospf <process-id>
    Router(config-router)# router-id <router-id>
    ```

  - Example: `router-id 1.1.1.1`.

4. **Interface Issues**
   - **Cause**: Interfaces are down or incorrectly configured.
   - **Solution**:
     - Check interface IP addresses:

       ```
       bash
       Copy code
       Router# show ip interface brief
       ```

     - Verify physical connections and use `no shutdown` to enable interfaces.