QUESTION 4: MULTI HOPNWAN FAULT TOLERANCE

QUESTION 1: TOPOLOGY EXTENSION AND ADDRESSING

Branch-C (Node B) → (Net1: 10.0.10.0/30) → DC-A (Router) → (Net2: 10.0.20.0/30) →DR-B (Server)

→ (Backup Net3: 10.0.30.0/30) → DR-B (direct backup link)

IP addressing example:

- **Branch-C ↔ DC-A (Net1): 10.0.10.0/30 → Branch .1, DC .2**
- **DC-A ↔ DR-B primary (Net2): 10.0.20.0/30 → DC .1, DR .2**
- **DC-A ↔ DR-B backup (Net3): 10.0.30.0/30 → DC .1, DR .2**

Make sure subnets are non-overlapping.

Q2 — Ipv4StaticRouting commands for normal & backup operation

On Branch-C (client):

- Route to DR via DC:

```
Ptr <Ipv4StaticRouting> srBranch = Ipv4RoutingHelper::GetStaticRouting(branch->GetObject  <Ipv4>());

srBranch→AddNetworkRouteTo(Ipv4Address("DR-LAN"),
```

```
Ipv4Mask("255.255.255.0"), Ipv4Address("10.0.10.2"), 1);
```

On DC-A (router):

- Normal route: Branch→DR via Net2 (primary)

  Ptr <Ipv4StaticRouting> srDC = Ipv4RoutingHelper::GetStaticRouting(dc->GetObject <Ipv4>());

  // Route to DR-LAN via primary interface to DR (10.0.20.2)

  srDC→ AddNetworkRouteTo (Ipv4Address("DR-LAN"), Ipv4Mask("255.255.255.0"), Ipv4Address("10.0.20.2"), interface primary/1);

- Backup route: DC→DR via Net3 (if primary fails)

srDC→AddNetworkRouteTo(Ipv4Address("DR-LAN"), Ipv4Mask("255.255.255.0"), Ipv4Address("10.0.30.2"), /*backup interface/2);

On DR-B (server):

- Route to Branch via DC (primary)

  Ptr<Ipv4StaticRouting> srDR = Ipv4RoutingHelper::GetStaticRouting(dr->GetObject<Ipv4>());

srDR→AddNetworkRouteTo(Ipv4Address("Branch-LAN"),

Ipv4Mask("255.255.255.0"), Ipv4Address("10.0.20.1"), /if to DC/1);

- Backup: route to Branch via DC through backup interface if needed (DR already connected to two DC links so reverse entries can be symmetric).

Administrative distance (real routers):

- Configure primary route with lower administrative distance; make backup with higher AD so it's used only if primary unreachable. In ns-3 static routing you emulate this by ordering or by programmatically deleting the primary route on failure.

Q3 — Simulate link failure at t = 5 s

Code snippet (disable DC→DR primary link):

```
Suppose devDC_DR_primary is the NetDevice on DC towards DR (primary)
Simulator::Schedule(Seconds(5.0), MakeCallback([](Ptr<NetDevice> dev1,
Ptr<NetDevice> dev2){

   Ptr<PointToPointNetDevice> d1 = DynamicCast<PointToPointNetDevice>(dev1);

 Ptr<PointToPointNetDevice> d2 = DynamicCast<PointToPointNetDevice>(dev2);

   if (d1) d1->SetLinkDown();

   if (d2) d2->SetLinkDown();

}), devDC_DR_primary, devDR_DC_primary_otherSide);
```

Immediate effect on routing tables:

- With static routing only: traffic to DR still has the static primary route entry; unless you remove it or the node detects link down and removes the associated route, packets will be sent to the now-down interface and be dropped. If you used SetLinkDown() and the device reports link state to Ipv4, ns-3 may mark interface down and **Ipv4StaticRouting** will stop using that interface if appropriate. In many cases you will need to programmatically update static routes: delete primary and rely on backup route.

Q4 — Convergence analysis (static vs dynamic / OSPF)

Static routing: failover requires external control (scripted route changes) — downtime equals detection time + update time (your script). No automatic reconvergence.

Dynamic routing (OSPF):

- Implement OSPF by using either:
  - An external OSPF ns-3 module (community code), or
  - DCE + an OSPF daemon (Quagga/FRR) to run inside the nodes for realistic behavior. OSPF will detect the link down and recompute SPF, then update forwarding tables automatically. Convergence time depends on Hello/LSA timers and SPF compute time.

Which NS-3 helper for OSPF?

- ns-3 core does not provide a standard OspfHelper; you can either use an external ospf module (third-party) or run Quagga/FRR via DCE for realistic OSPF behavior. See community modules and DCE

Compare convergence :

- Measure time between link failure event and successful packet delivery to DR using FlowMonitor or custom traces.
- Expect dynamic OSPF with tuned timers to converge in tens to hundreds of ms (depending on timers), static scripted change converges immediately only if your script changes routes exactly at failure time, otherwise until detection.

Q5 — Business continuity verification plan (FlowMonitor + traces)

Use FlowMonitor to record:

- **Before failure:** record route (via per-hop TTL analysis or NetDevice Tx traces to confirm primary path).
- **At failure:** detect drops or path change events; if static routes, packets drop; if dynamic OSPF, observe re-established flow via alternative path.
- **Metrics to collect:** per-flow rxPackets, txPackets, delaySum, packetLoss, hop count (inferred).
- **Procedure:** run two experiments (static-only vs dynamic (DCE/OSPF)), trigger link failure at t=5s, and plot:
  - Flow throughput vs time
  - Packet loss vs time
  - Average latency before vs after failover