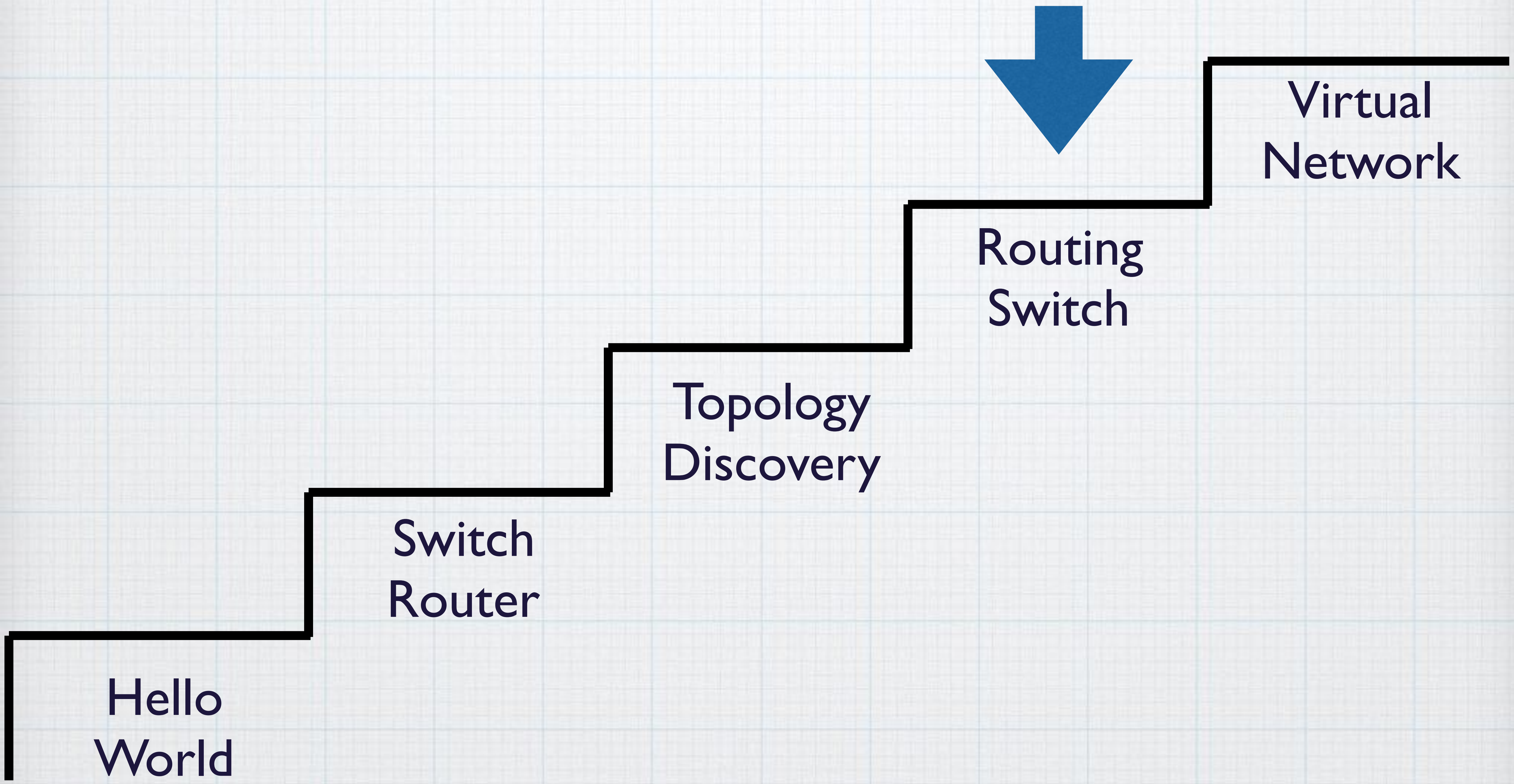
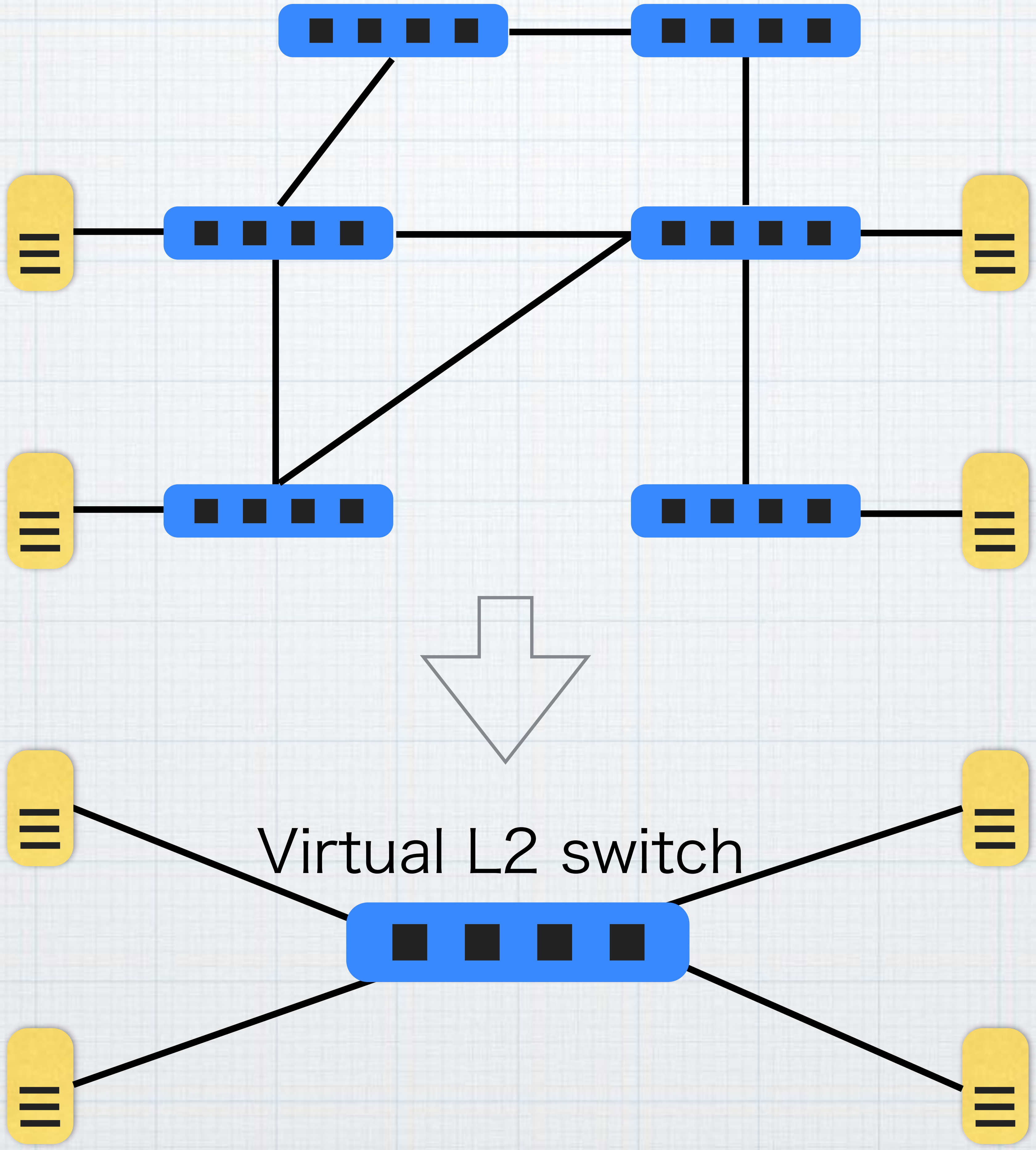
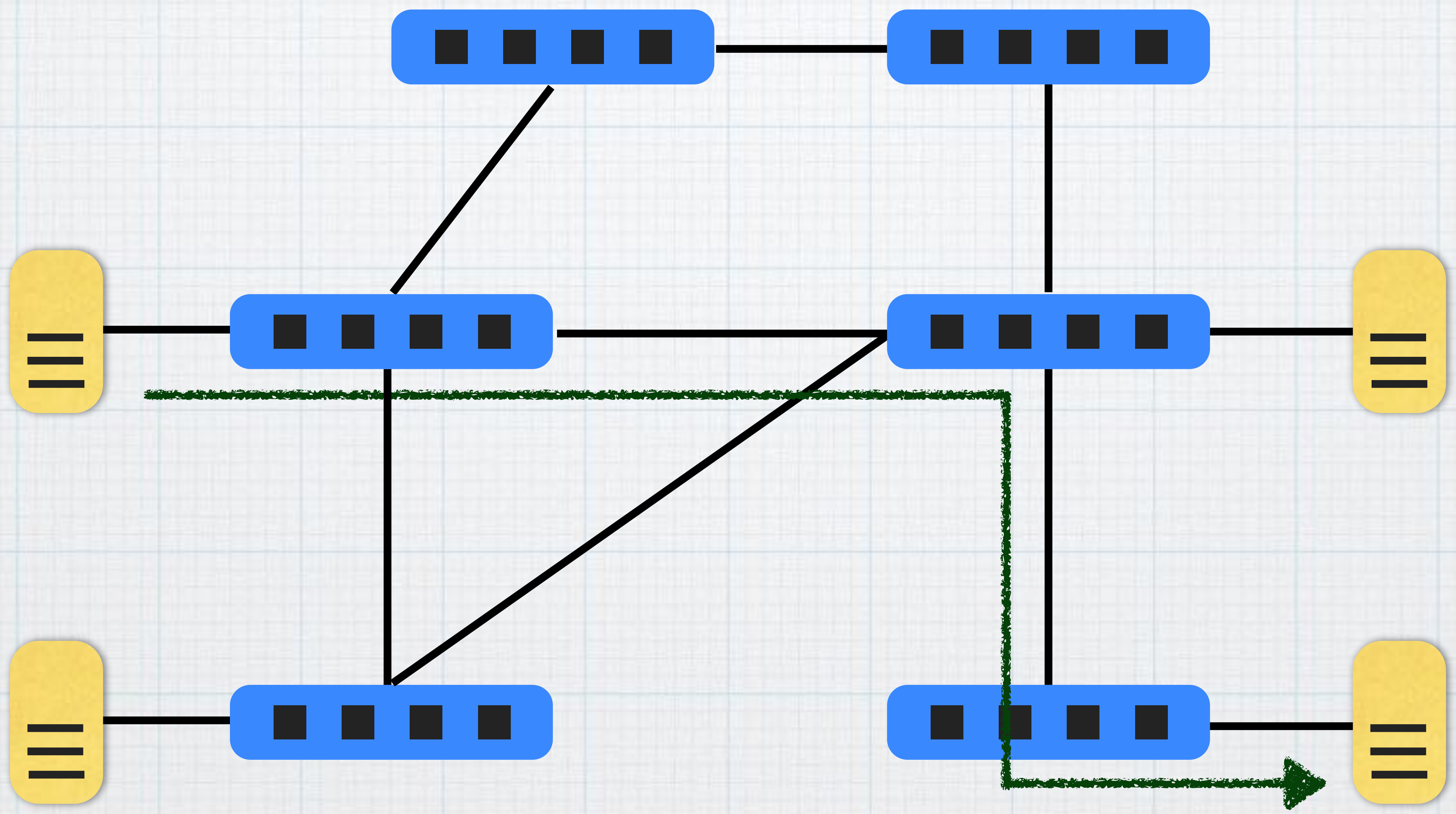
Control Multiple Switches

高宮安仁 @yasuhito


Hello World
Switch Router
Topology Discovery
Routing Switch
Virtual Network


Virtual L2 switch

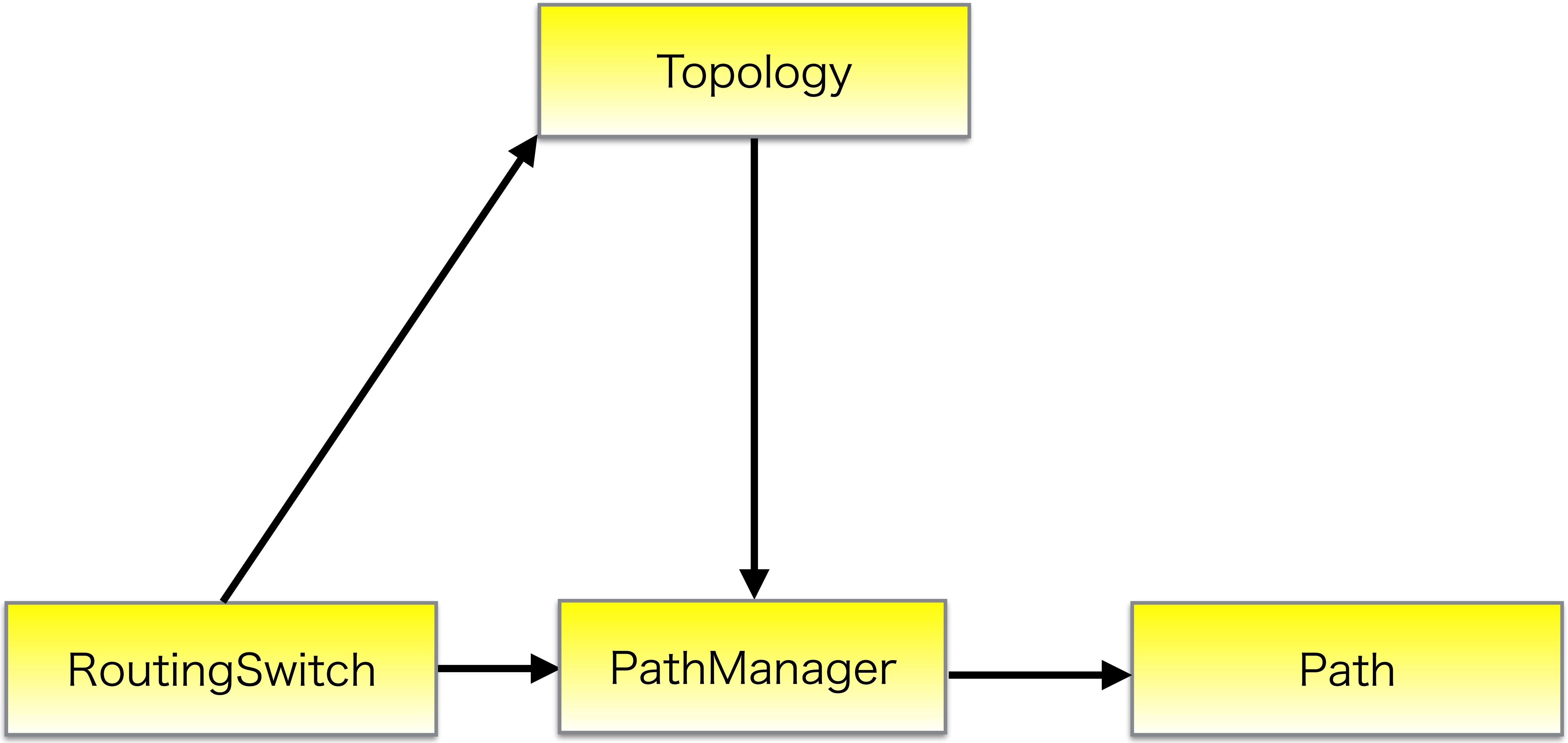
Fundamentals of Routing



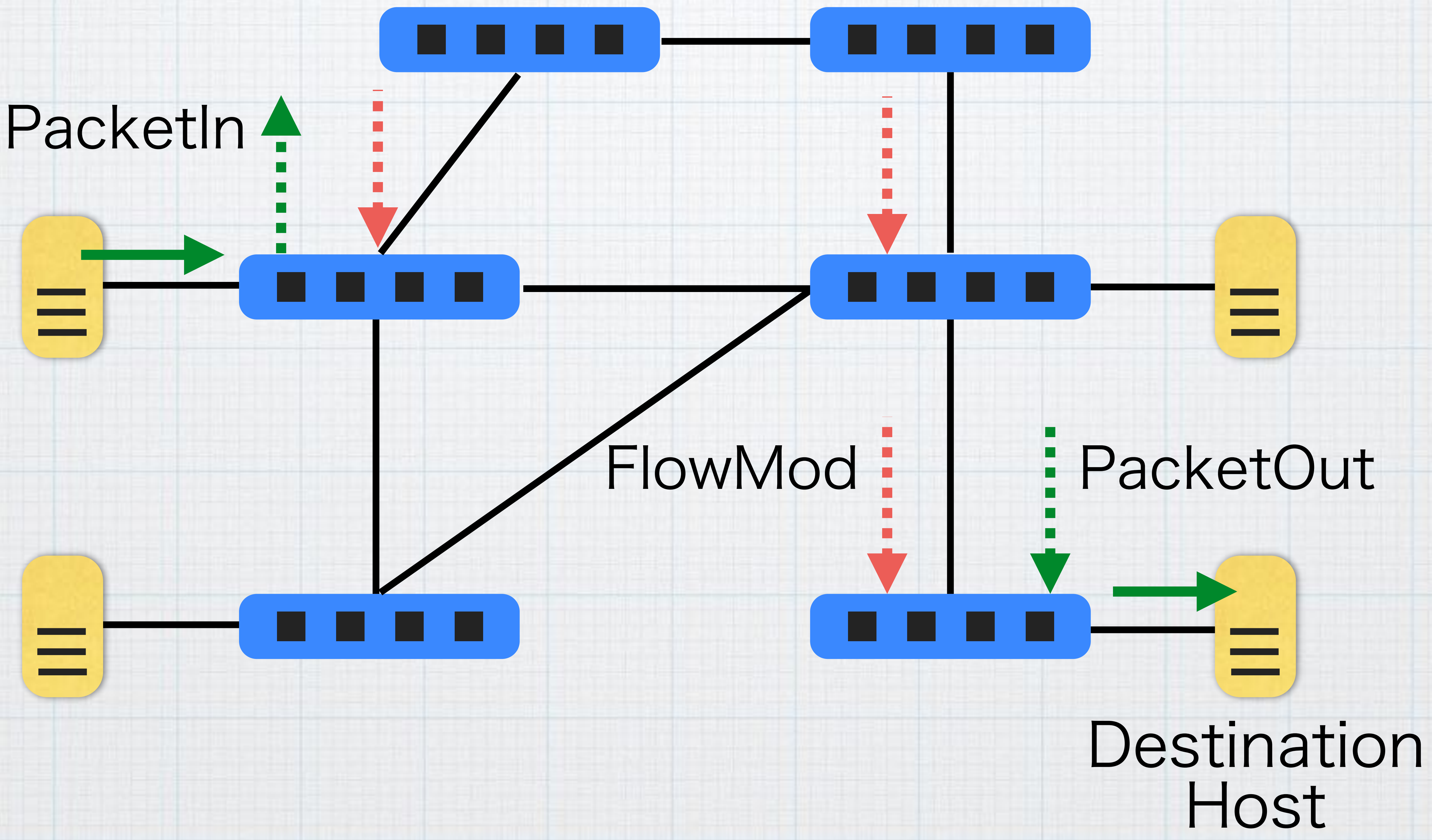
How to Code a Controller

- Connecting to switches
- Discovering topologies
- Searching shortest paths
- Managing flow entries

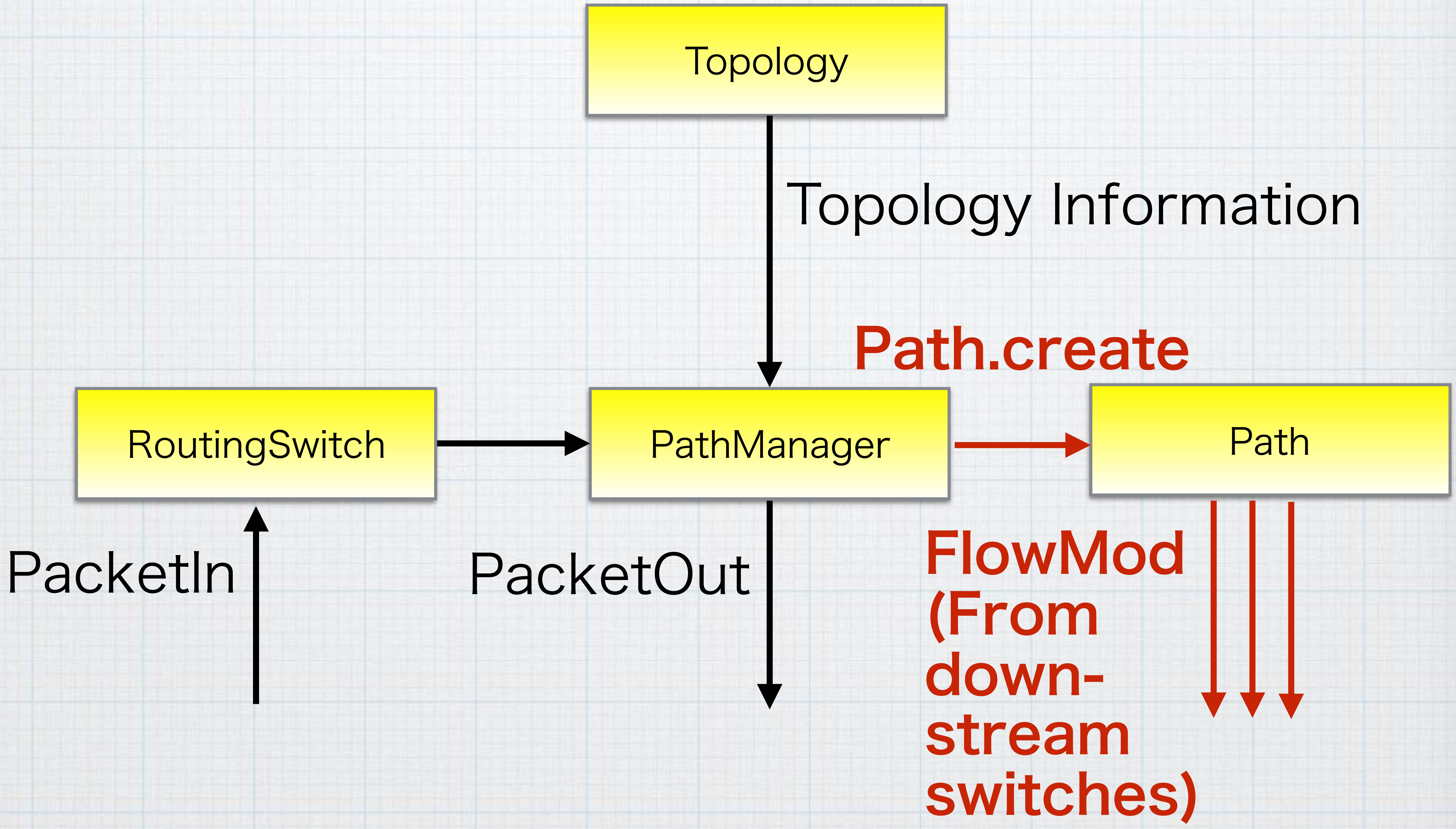
Using Small Classes



When a packet is sent…



Create shortest paths



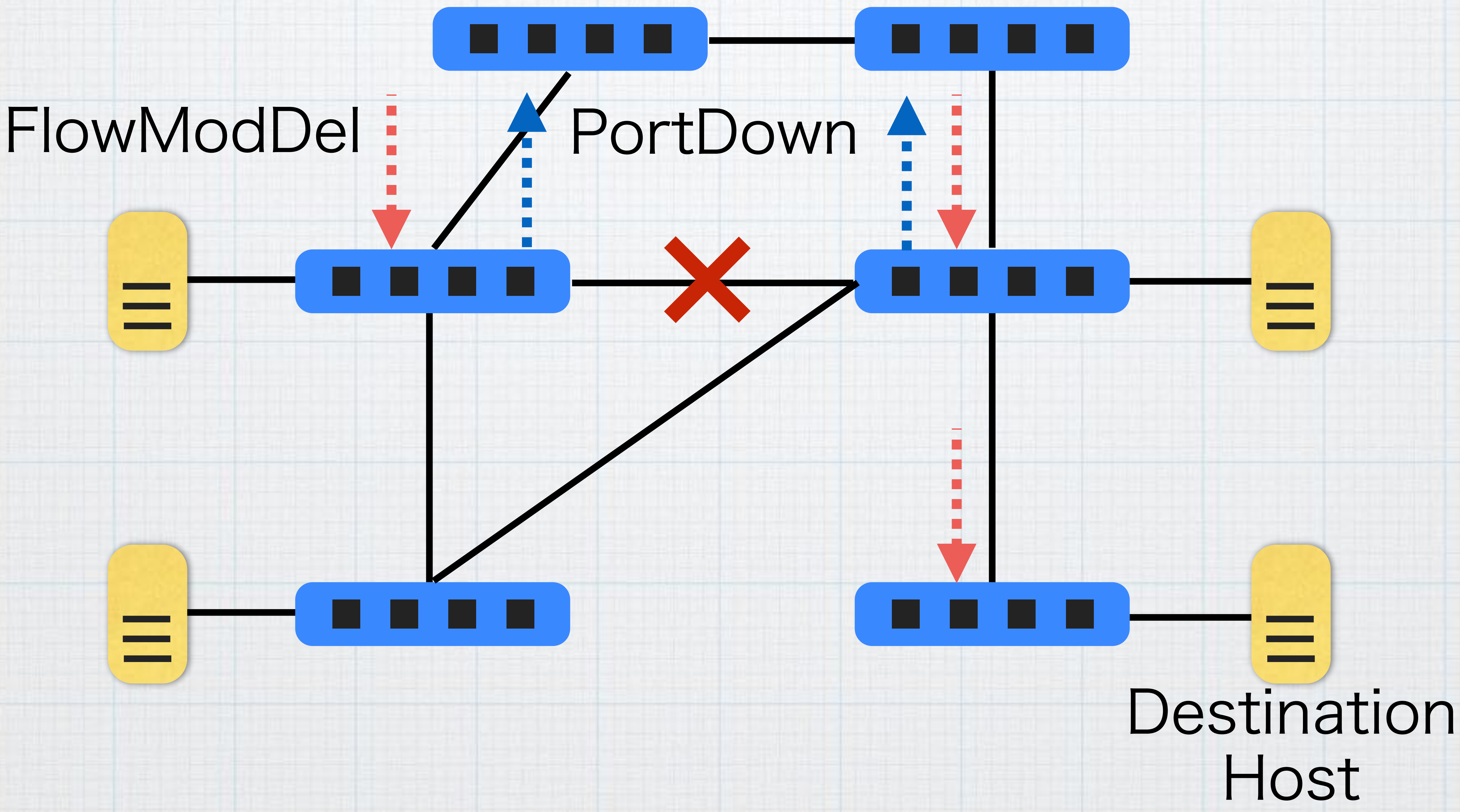
Encapsulate Procedures

- `Path.create(path, packet_in)`
  Invoke **`FlowModAdd`** along with **`path`** from downstream switches

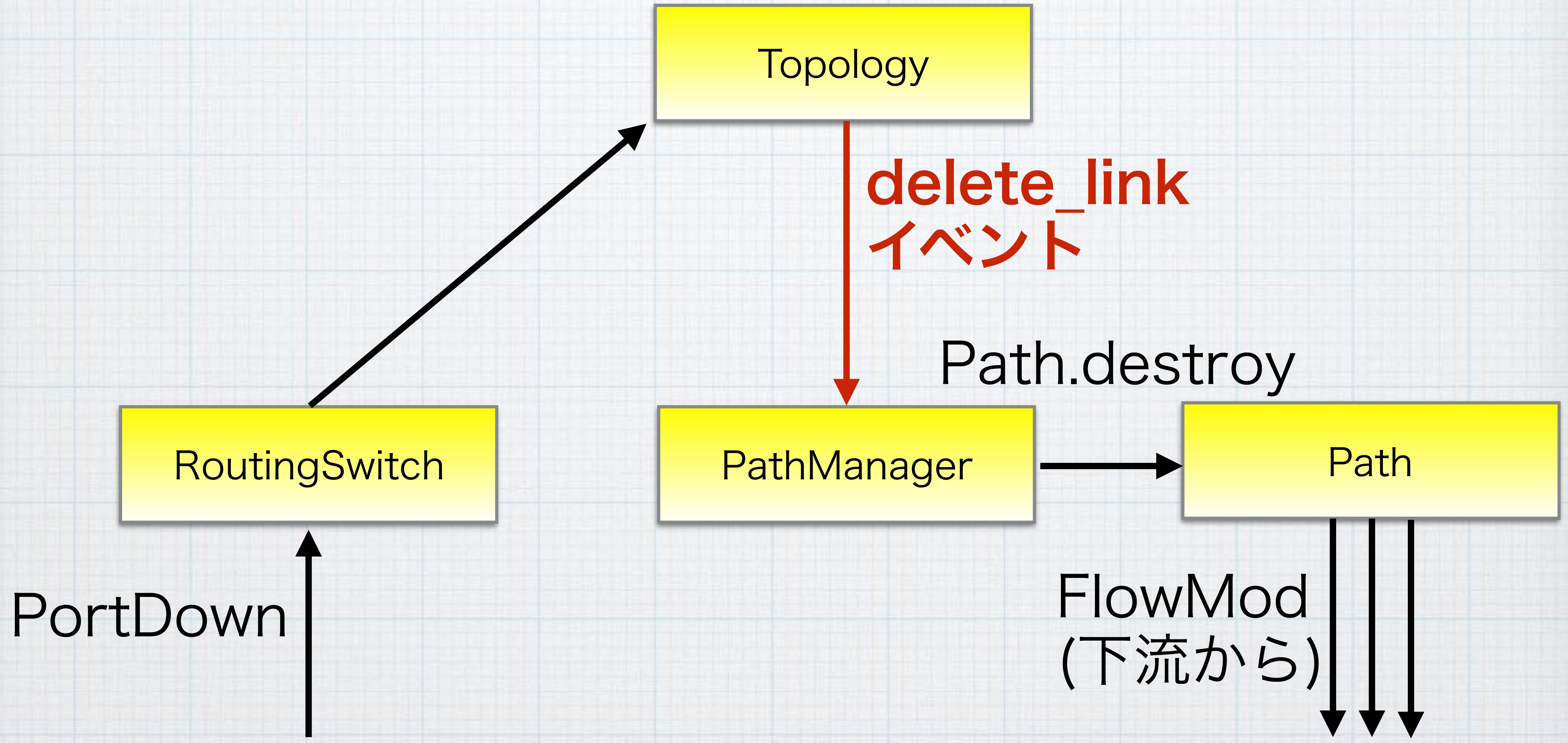
- `Path.destroy(path)`
  Invoke **`FlowModDel`** along with **`path`** from upstream switches

-
  ```
  Path.select do |each|
    each.link?(port_a, port_b)
  end.each(&:destroy)
  ```
  Destroy all links between ports a and b

When a link is tore down…



Delete all invalid paths



Observer Pattern

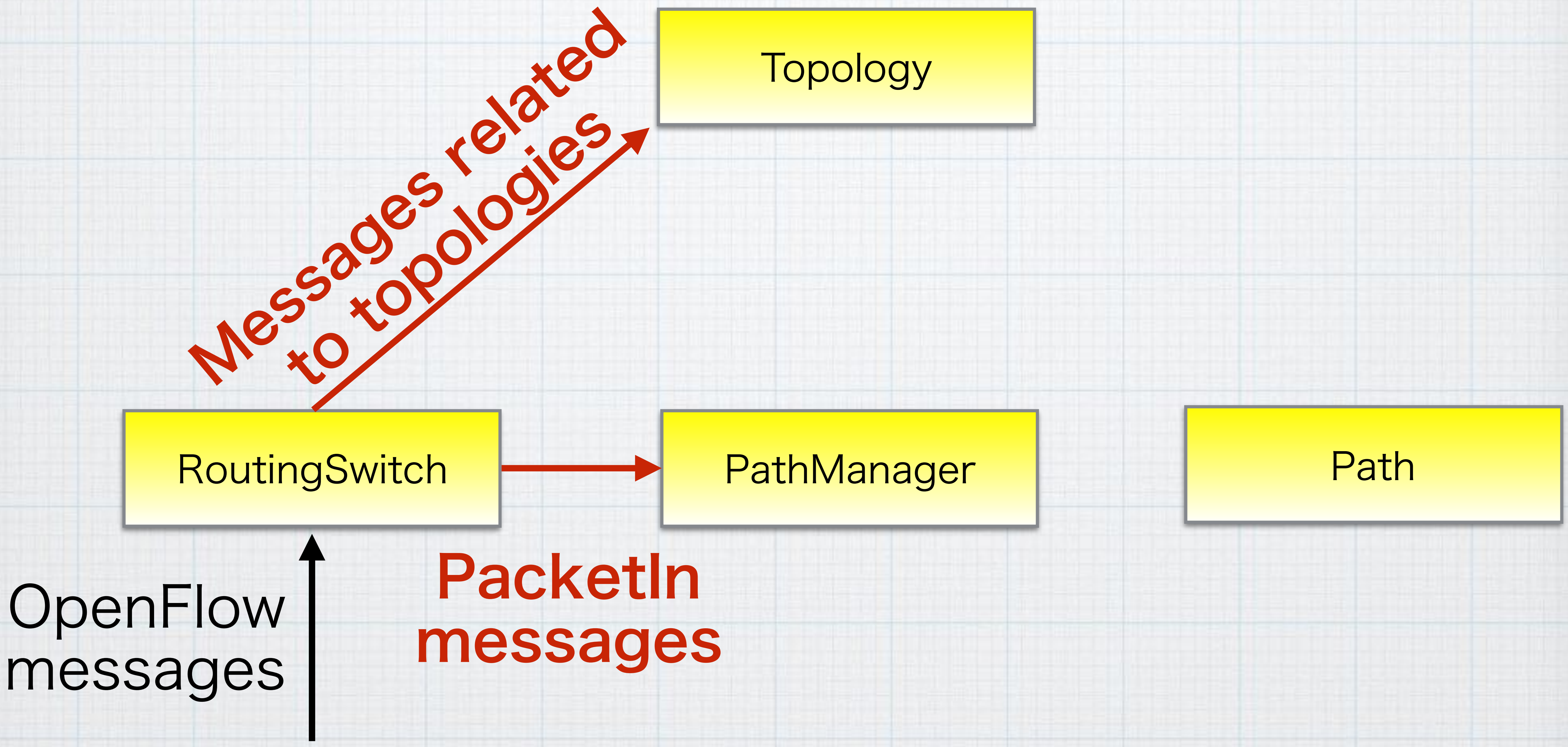
Notify PathManager of events on topologies

```
class RoutingSwitch < Trema::Controller
  # ...
  def start_topology
    TopologyController.new { |topo| topo.add_observer @path_manager }.start
  end
end
```

Delete paths through event handlers

```
class PathManager < Trema::Controller
  # ...
  def delete_link(port_a, port_b, _topology)
    @graph.delete_link port_a, port_b
    Path.select { |each| each.link?(port_a, port_b) }.each(&:destroy)
  end
```

Forwarding Messages



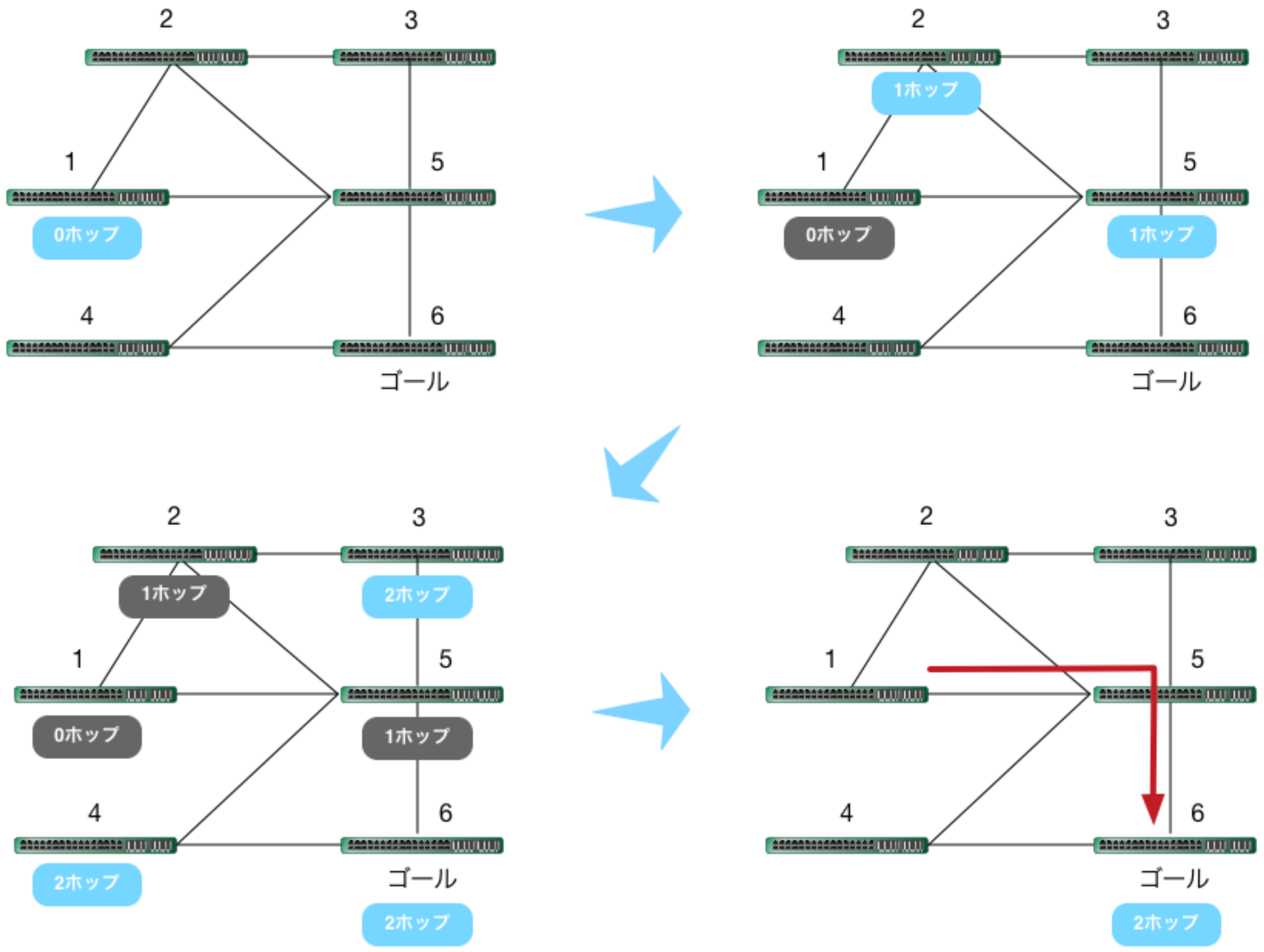
Delegate Handlers

Message related to topologies → Topology

```
class RoutingSwitch < Trema::Controller
  delegate :switch_ready, to: :@topology
  delegate :features_reply, to: :@topology
  delegate :switch_disconnected, to: :@topology
  delegate :port_modify, to: :@topology
```

PacketIn → Topology and PathManager

```
def packet_in(dpid, packet_in)
  @topology.packet_in(dpid, packet_in)
  @path_manager.packet_in(dpid, packet_in) unless packet_in.lldp?
end
```


2
3
1
5
4
6
0ホップ
ゴール
2
3
1ホップ
1
5
0ホップ
1ホップ
4
6
ゴール
2
3
1ホップ
2ホップ
1
5
0ホップ
1ホップ
4
6
2ホップ
ゴール
2ホップ
2
3
1
5
4
6
ゴール
2ホップ

Conclusion

- Create small classes (components)
  - Reuse such small components
  - Make source codes clear
- Fundamental techniques of object-oriented programming
  - Such as encapsulation and delegation