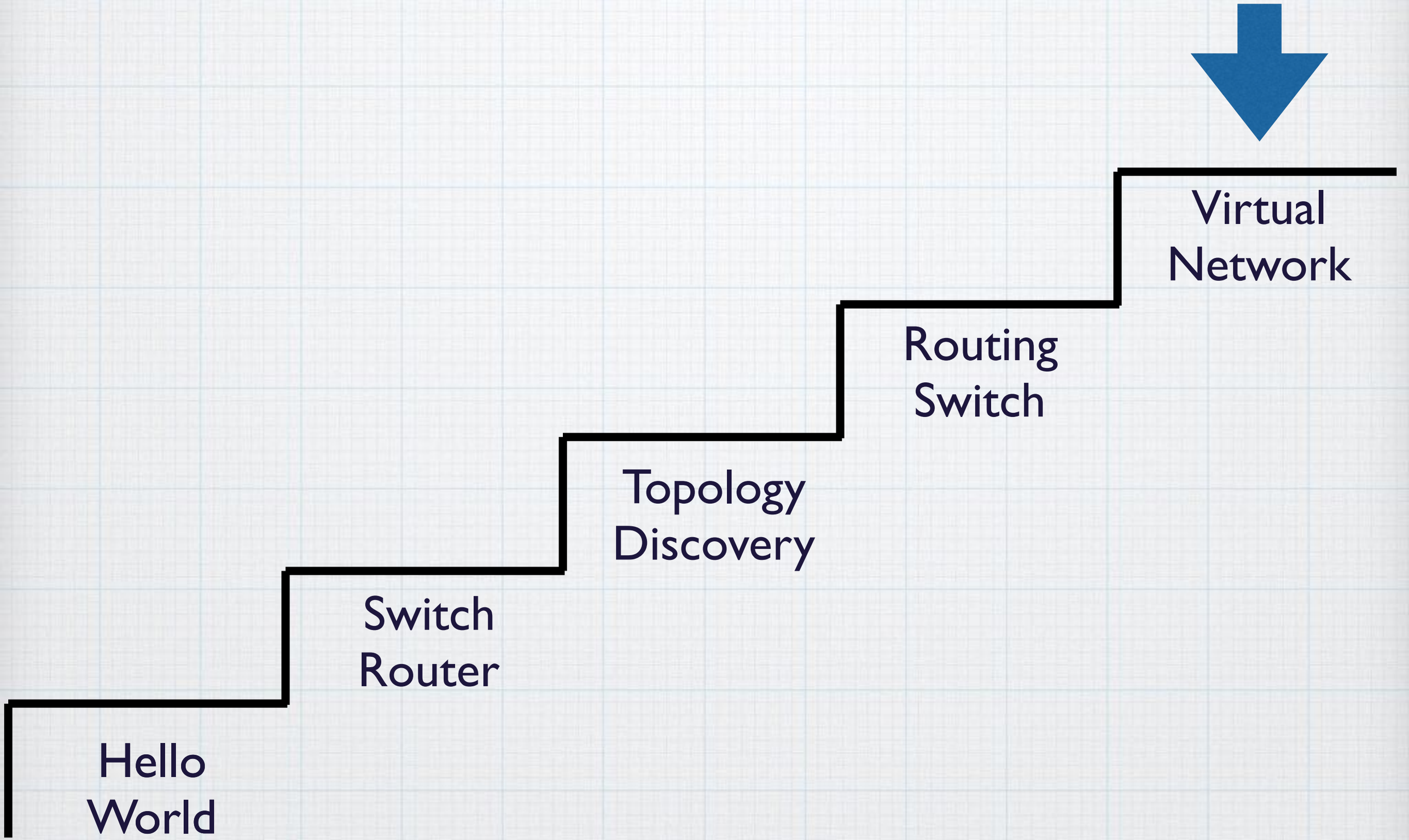
Virtual Network (Slicing)

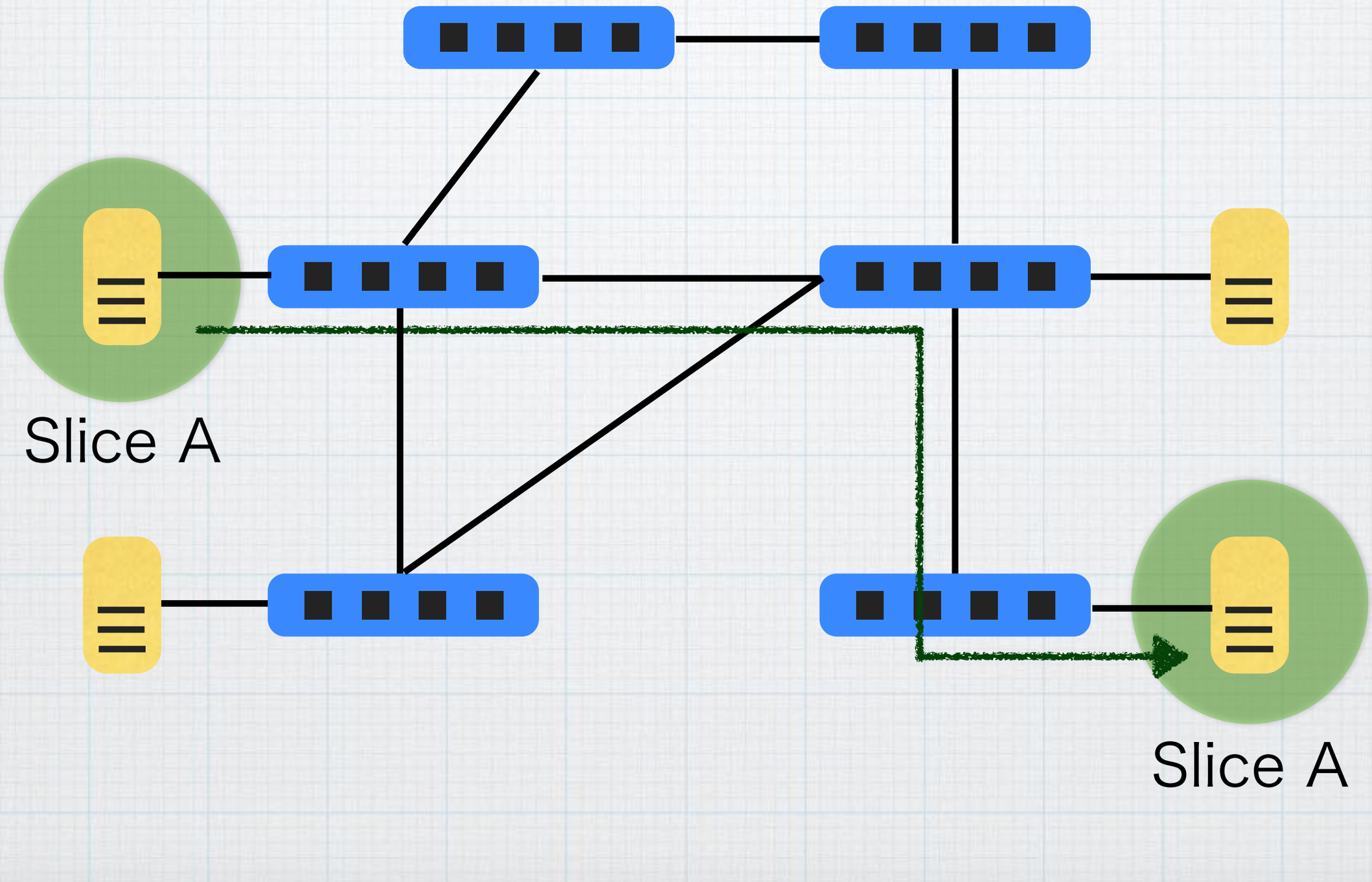
高宮 安仁 @yasuhito


Virtual
Network
Routing
Switch
Topology
Discovery
Switch
Router
Hello
World

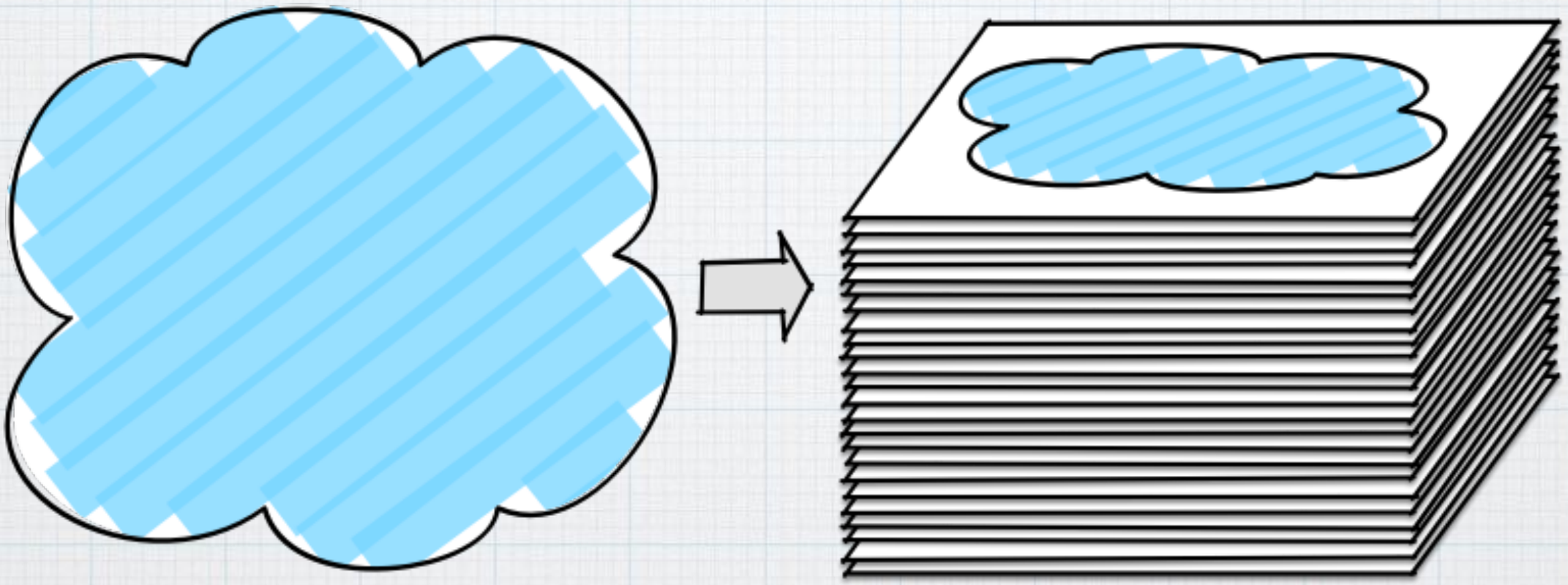
Sliceable Switch

Functions

Network Slice



Schematic of Slices



Sliceable Switch

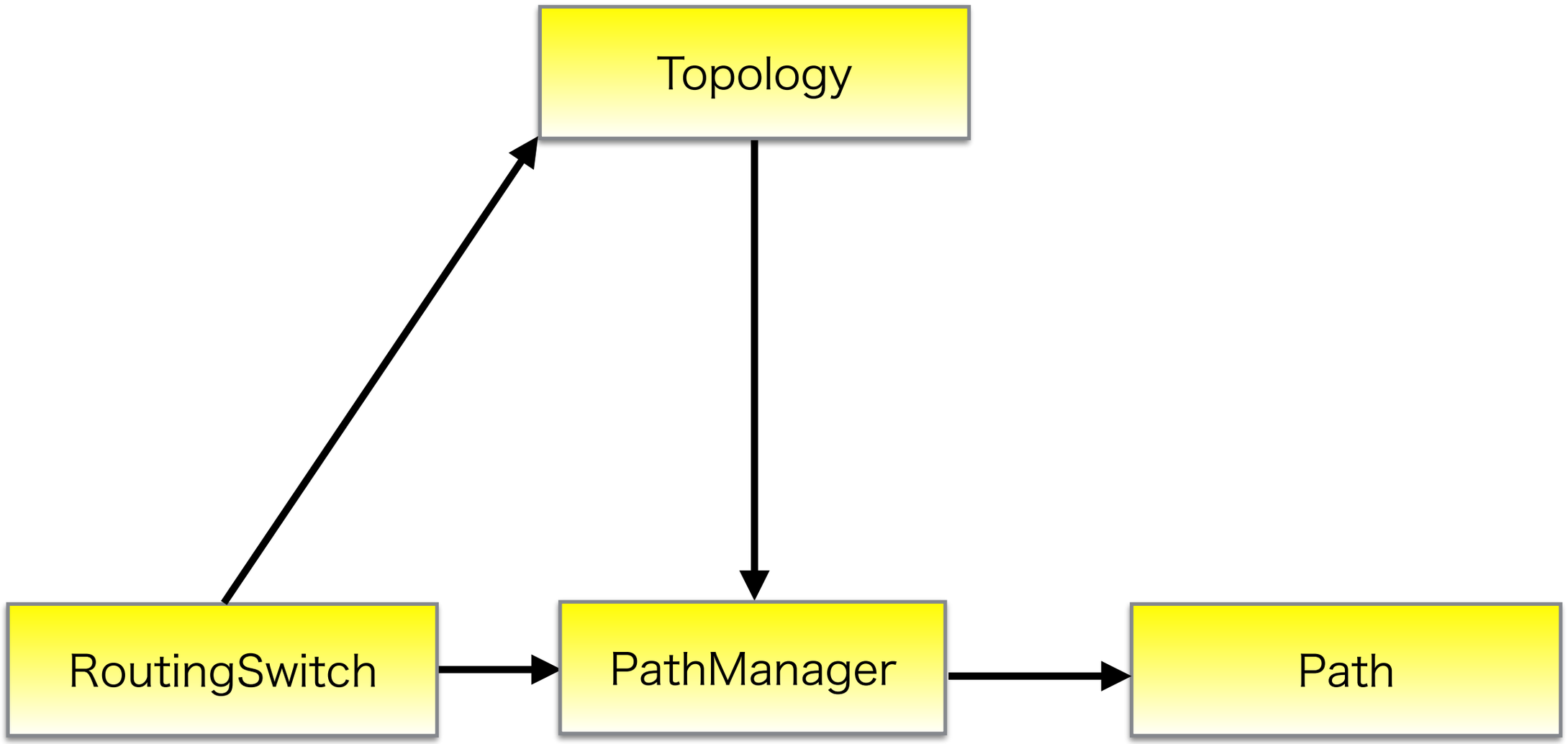
Routing switch with slice management functions

- Extend a routing switch to enable it to manage network slices

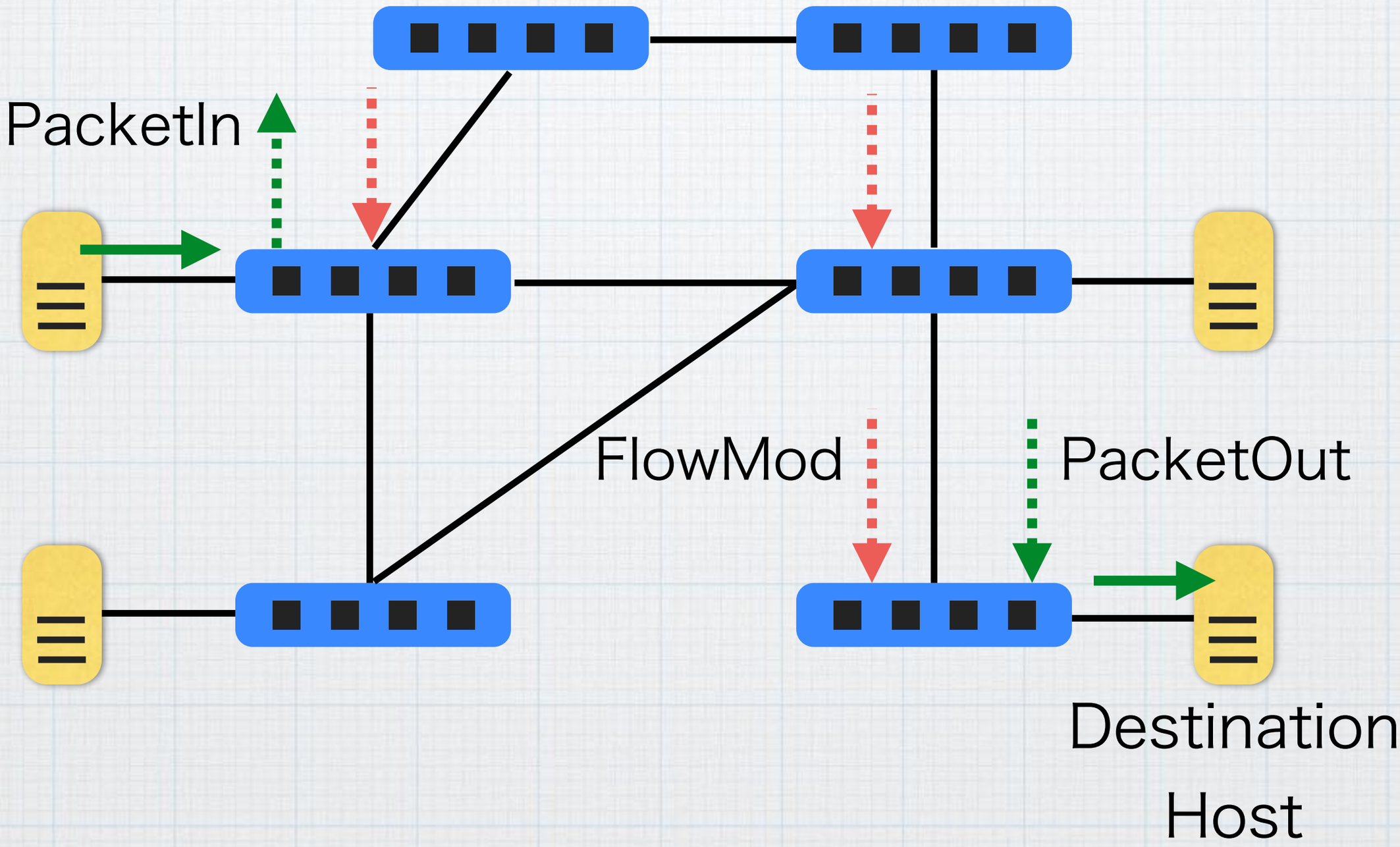
Sliceable Switch

A review of functions
of a routing switch

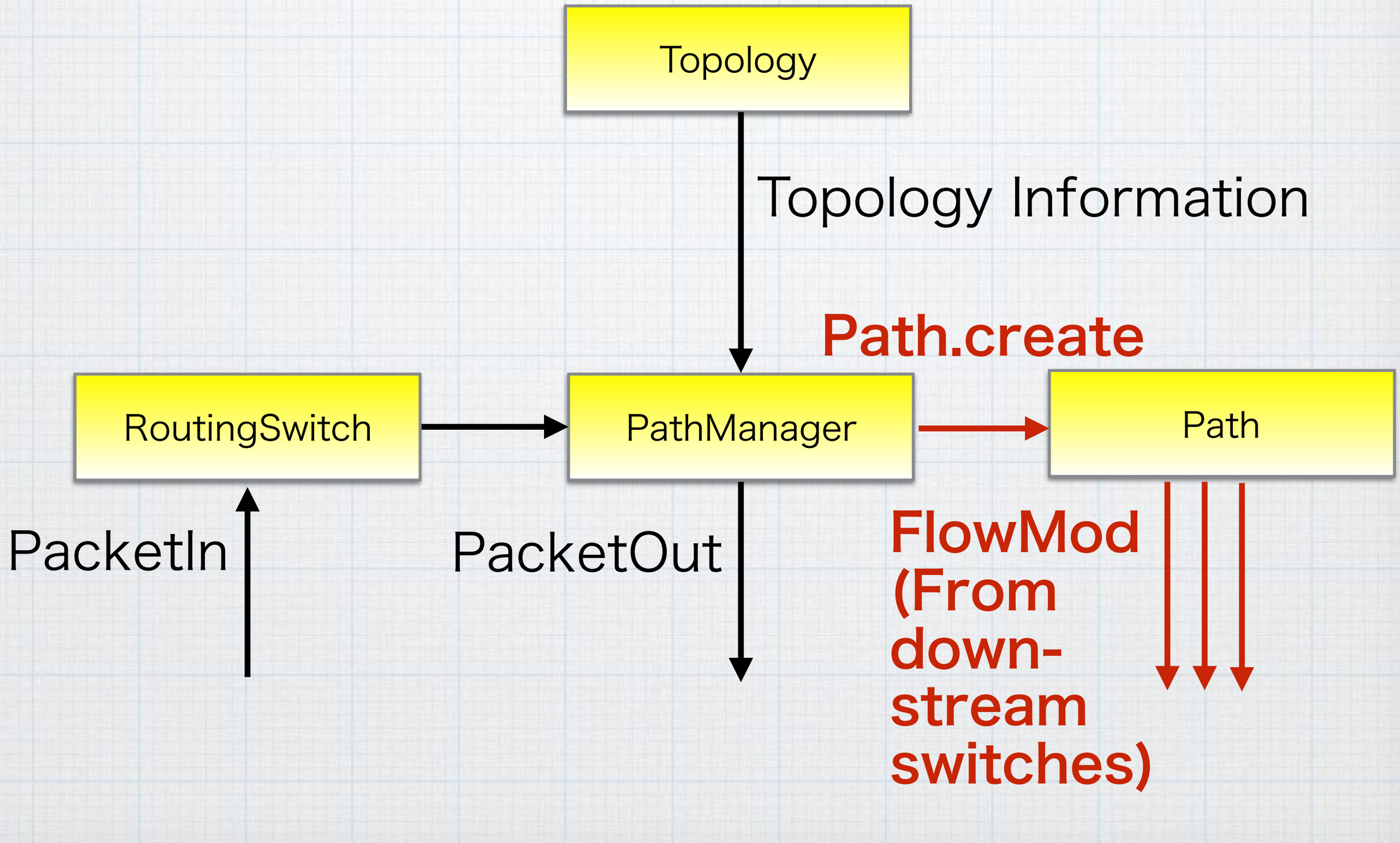
Building Blocks (Classes) of a Routing Switch



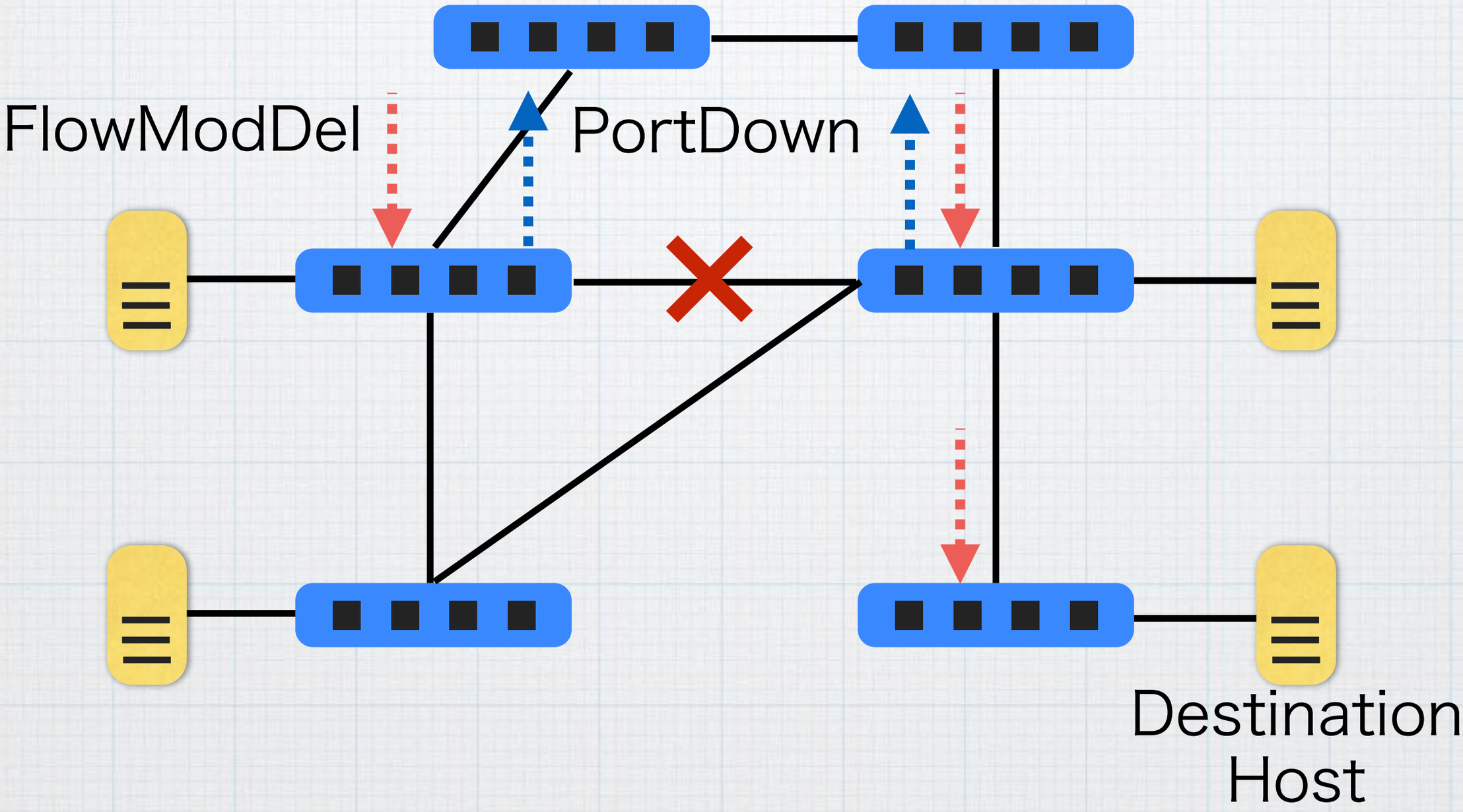
When a packet is sent…



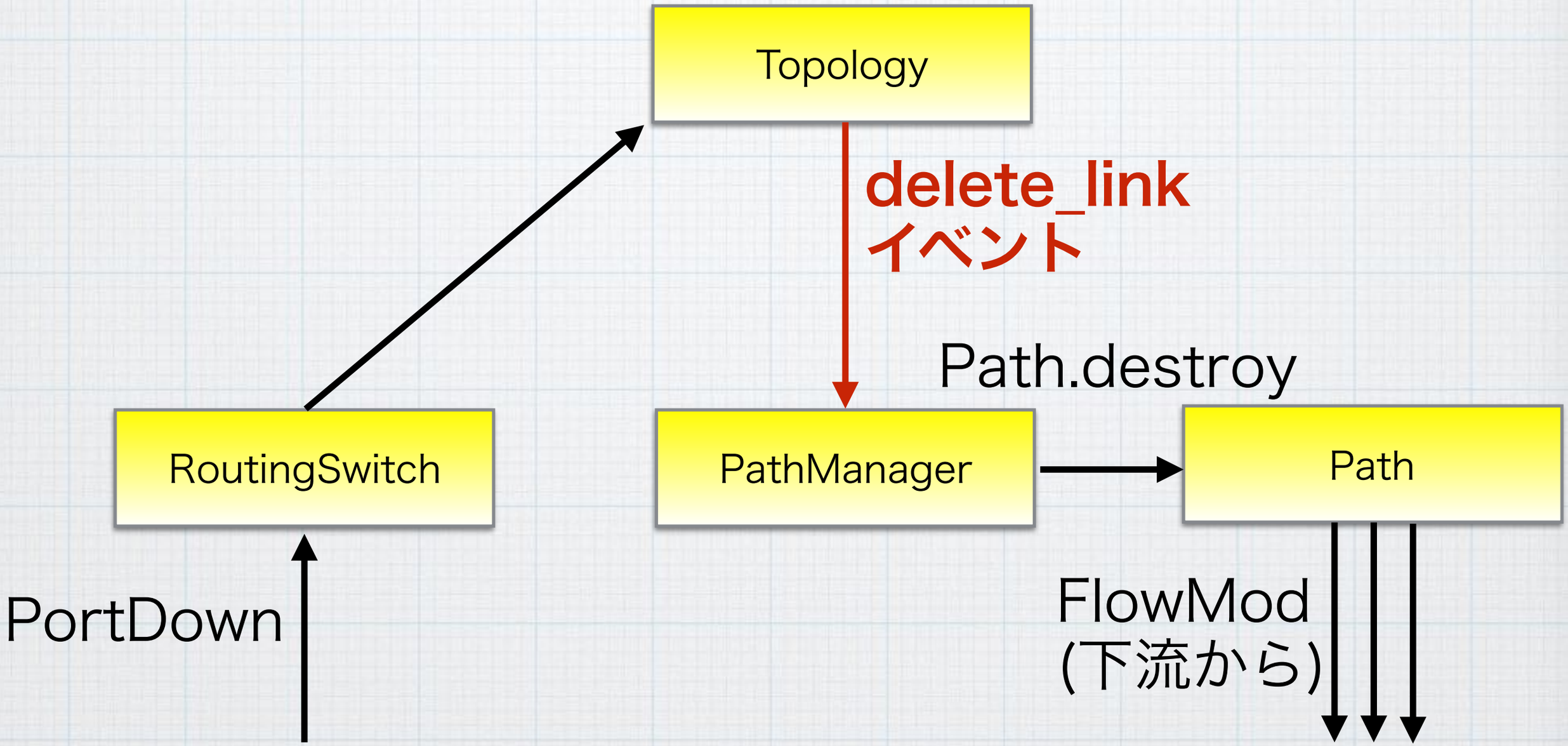
Create shortest paths



When a link is tore down…



Delete all invalid paths



Forwarding Messages

Messages related to topologies

Topology

RoutingSwitch

PathManager

Path

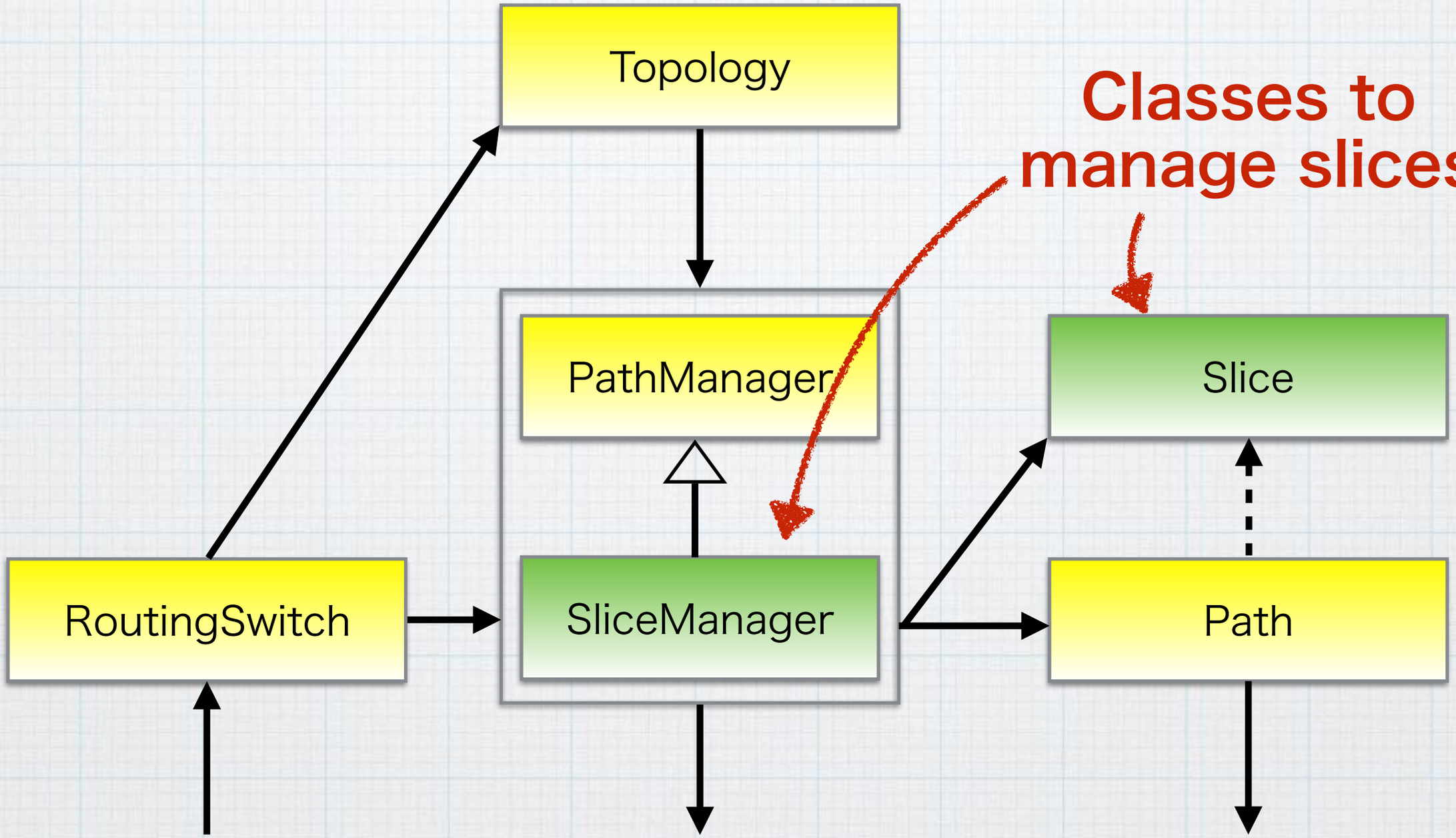
OpenFlow messages

PacketIn messages

Sliceable Switch

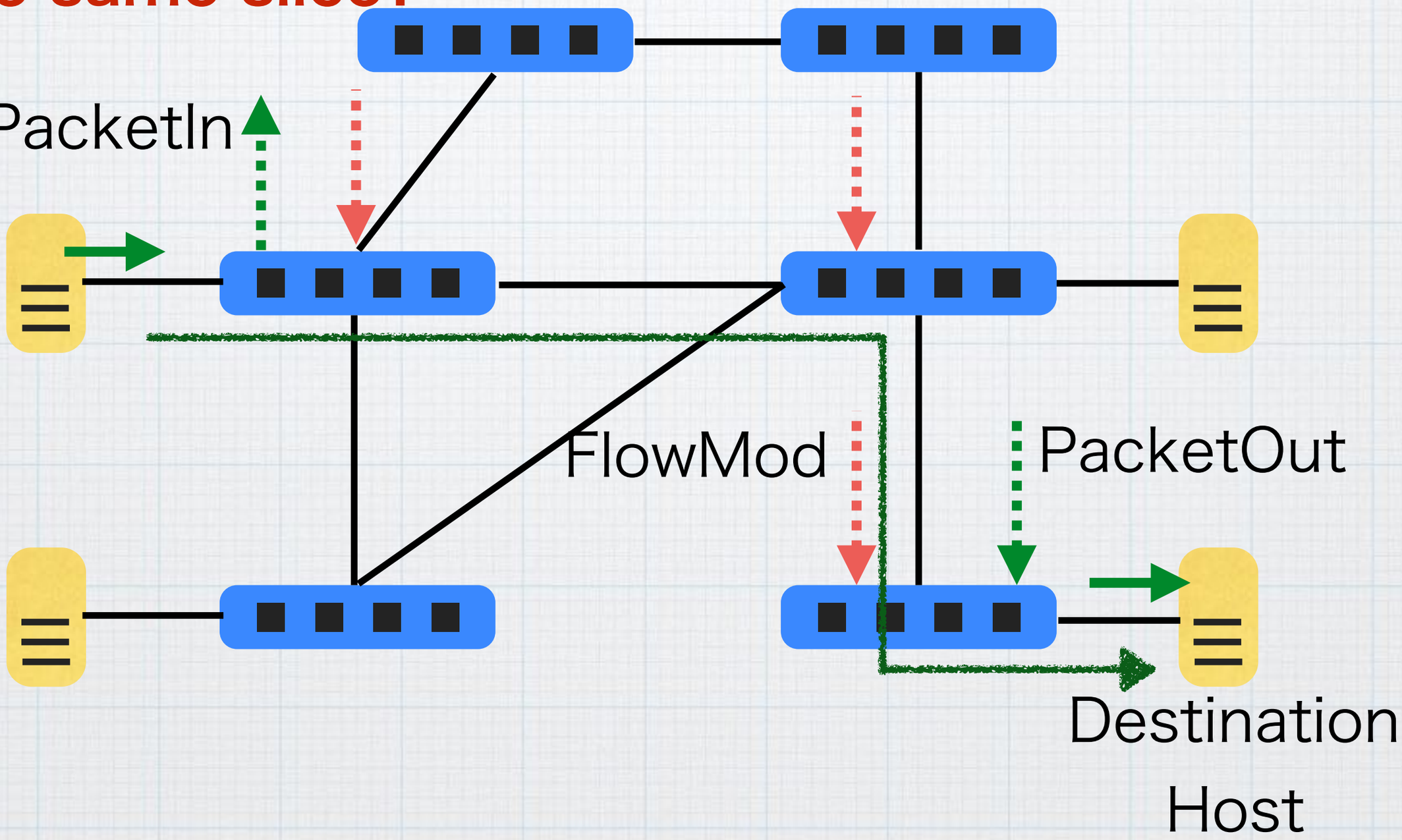
Class Structure

Add Classes to Manage Slices

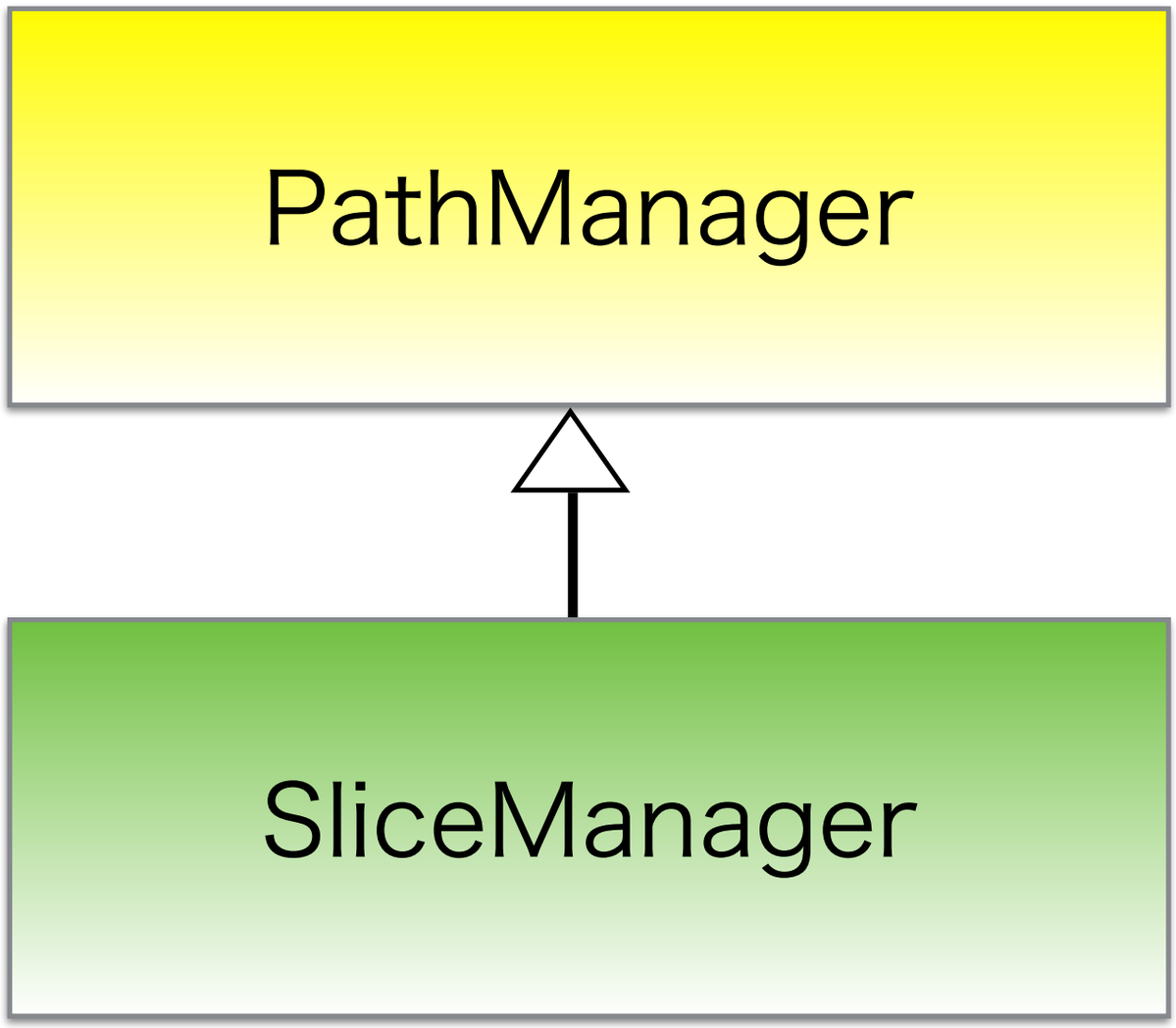


Shortest Path

On the same slice?



Inheritance and Overriding



- **Topology handler**
  Create a graph for the topology
- **PacketIn handler**
  Create the shortest path

- **Topology hander(Inheritance)**
- **packet_in(Overriding)**
  Check whether target entities are on the same slice before creating the shortest path

On the same slice?

```
def packet_in(_dpid, packet_in)
  slice = Slice.find do |each|
    each.member?(packet_in.slice_source) &&
    each.member?(packet_in.slice_destination(@graph))
  end
  ports = if slice
            path = maybe_create_shortest_path_in_slice(slice.name, packet_in)
            path ? [path.out_port] : []
          else
            external_ports(packet_in)
          end
  packet_out(packet_in.raw_data, ports)
end
```

Look up a slice where both src and dst belong

- If such slice exists, forward the packet along the shortest path.
- Otherwise, forward it to all external ports

API of Slice Class

```ruby
def self.find(&block)
  all.find(&block)
end
```

Find slices that meet the condition given by `block`

```ruby
def self.create(name)
  if find_by(name: name)
    fail SliceAlreadyExistsError, "Slice #{name} already exists"
  end
  new(name).tap { |slice| all << slice }
end
```

Create a slice

Topology

PathManager

SliceManager

RoutingSwitch

Slice

Path

OpenFlow Message

PacketOut

FlowMod

Encapsulation

- Add/Delete slices
- Add/Delete hosts

Sliceable Switch

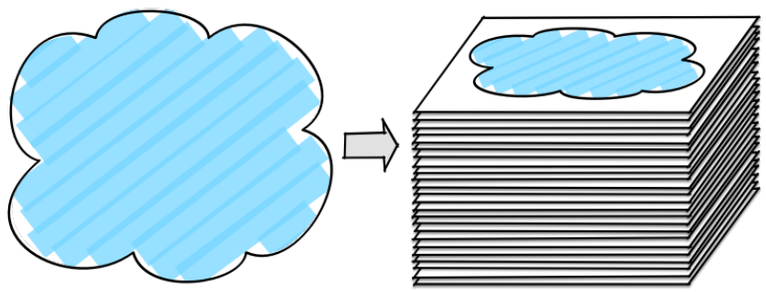
Use it as a Component of IaaS

For instance,
in the case of Mini IaaS

Web Interface

REST API

Sliceable Switch



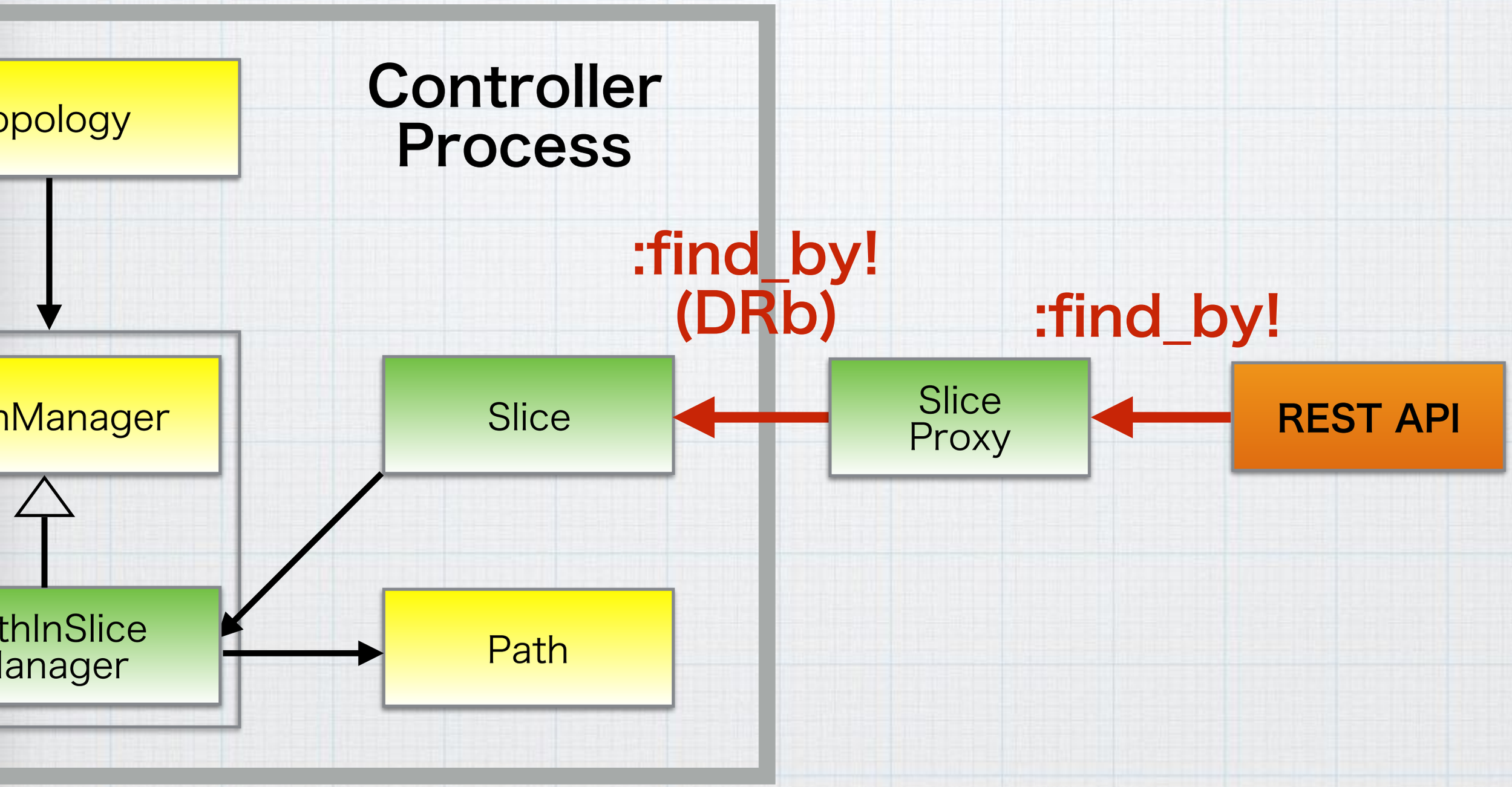
REST API

Docker

REST API

Description	Method	URI
Create a slice	POST	/slices
Delete a slice	DELETE	/slices
List slices	GET	/slices
Shows a slice	GET	/slices/:slice_id
Add a port to a slice	POST	/slices/:slice_id/ports
Delete a port from a slice	DELETE	/slices/:slice_id/ports
List ports	GET	/slices/:slice_id/ports
Shows a port	GET	/slices/:slice_id/ports/:port_id
Adds a host to a slice	POST	/slices/:slice_id/ports/:port_id/mac_addresses
Deletes a host from a slice	DELETE	/slices/:slice_id/ports/:port_id/mac_addresses
List MAC addresses	GET	/slices/:slice_id/ports/:port_id/mac_addresses
Shows a MAC address	GET	/slices/:slice_id/ports/:port_id/mac_addresses/:mac_address

REST API


Controller
Process
:find_by!
(DRb)
:find_by!
REST API
Slice
Slice
Proxy
Path
pology
nManager
thInSlice
lanager

Call Method of Remote Classes

Implementing REST API

```
get 'slices/:slice_id' do
  rest_api { Slice.find_by!(name: params[:slice_id]) }
end
```

Call methods of Slice class

Slice Class Proxy

```
class Slice
  def self.find_by!(query)
    # ...
    remote_klass =
      Trema.trema_process('RoutingSwitch', socket_dir).controller.slice
    remote_klass.find_by!(query)
  end
end
```

Sliceable Switch

Conclusion and Assignment

Conclusion

How to realize virtual networks with OpenFlow

- Extend the routing switch appropriately
  - Add classes to manage slices
  - Use inheritance
- REST API
  - Use the sliceable switch as a component of IaaS

Assignment: Extending a slice function

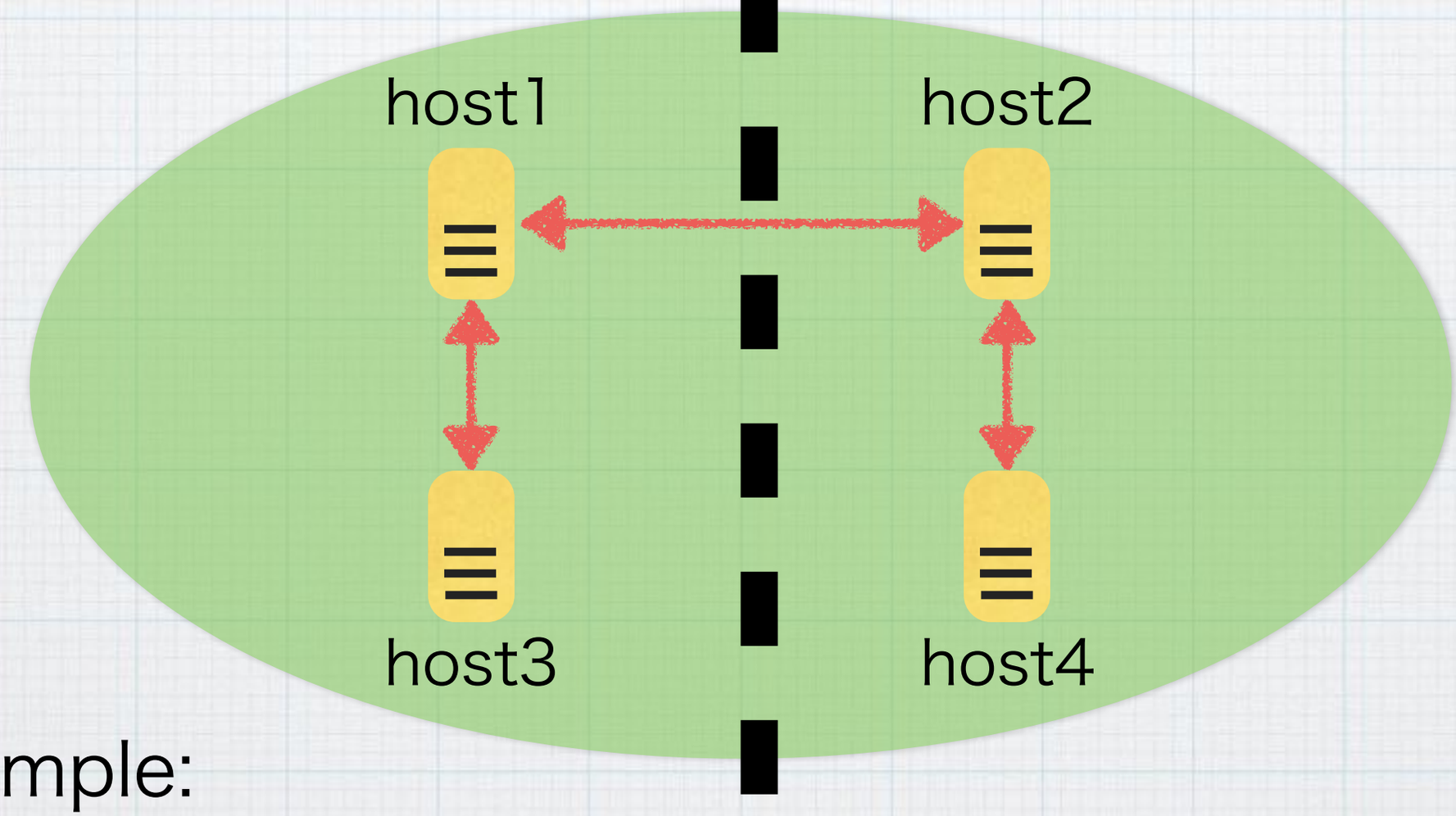
Add functions to divide a slice and to integrate slices.

Add a function to display slices on a web browser.

Add REST API for the functions of slice division and slice integration.

Run your virtual network on the physical OpenFlow switch

Command for slice division



Example:

$ slice split slice_a

—into slice_b:host1,host3 slice_c:host2,host4