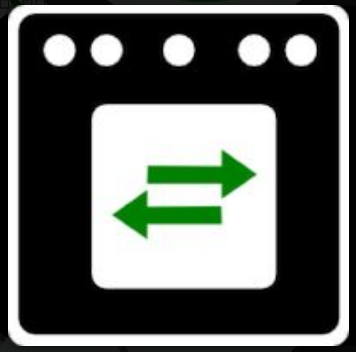
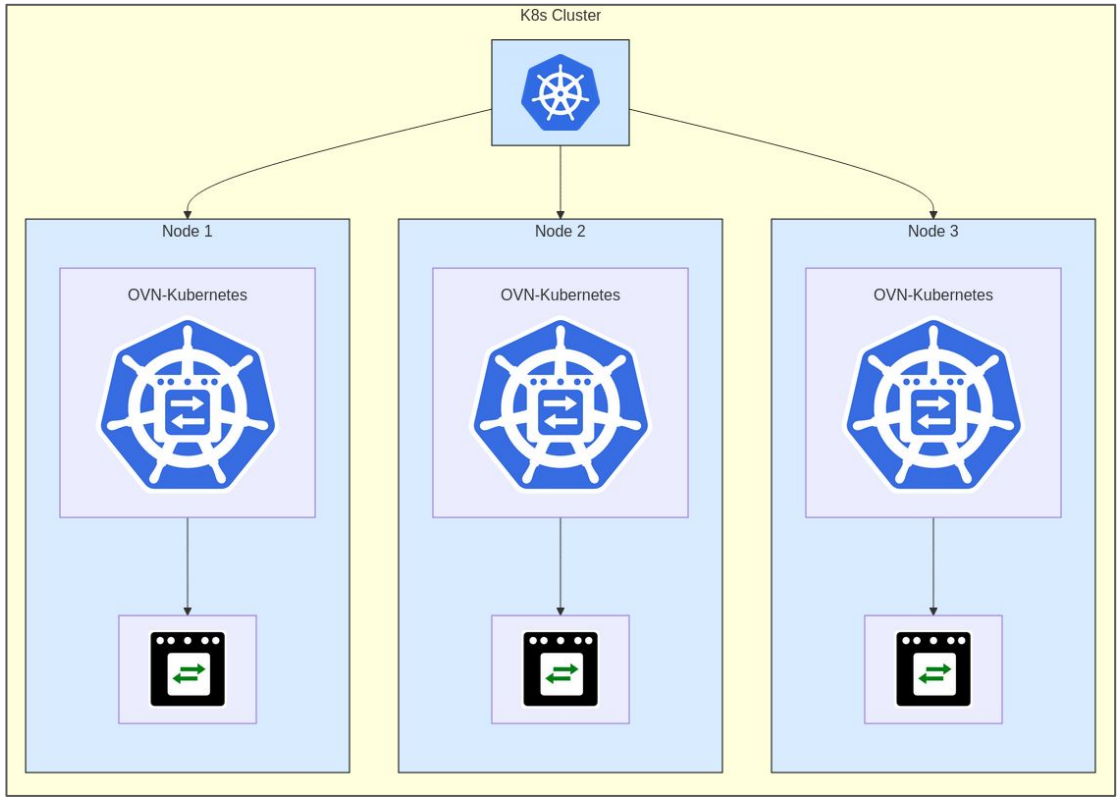


You Want to Upgrade WHAT?

A Field Guide to Risky Network Changes

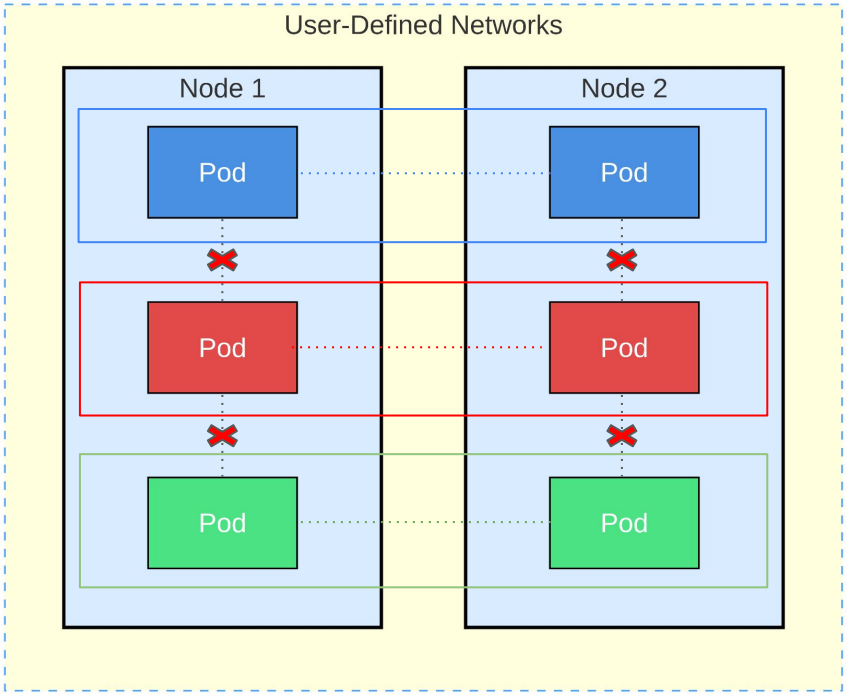
Nadia Pinaeva, NVIDIA
Dumitru Ceară, Red Hat
Patryk Diak, Red Hat

OVN-Kubernetes



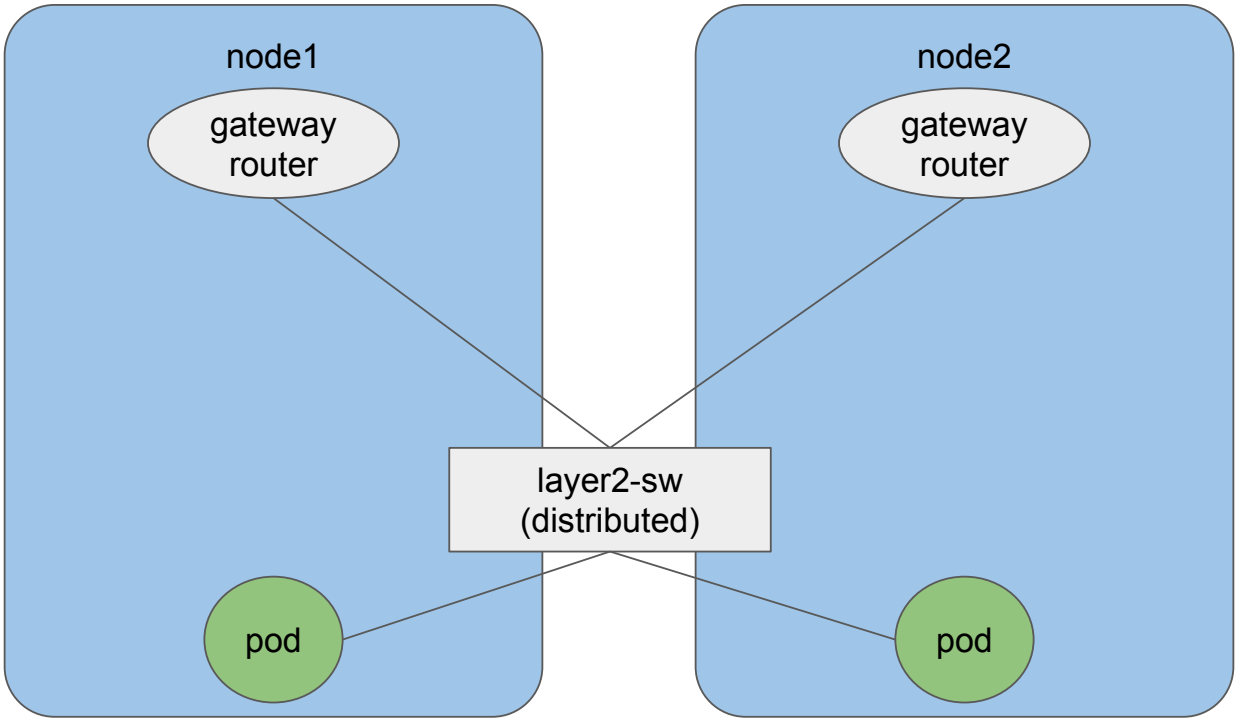
K8s Objects	OVN Logical Entities
Node	Switch, Router, Routes
Pod	Logical Switch Ports
Service	Load Balancer
Network Policy	ACL
EgressIP	NAT, Router Policies

User defined networks

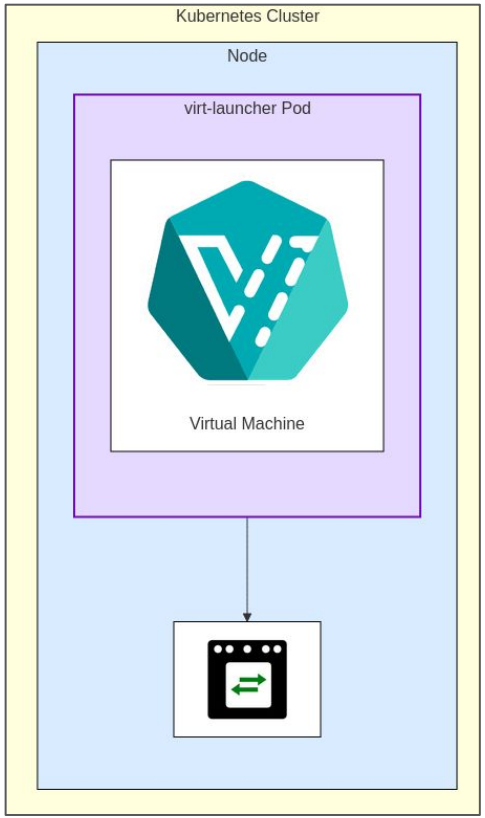


- Network Isolation
- Overlapping IPs
- Cross-Node & External Connectivity
- Available Topologies:
  - **Layer3**: Routed network with per-node subnets
  - **Localnet**: Bridge to physical underlay network
  - **Layer2**: Flat network with single broadcast domain

Layer2 User Defined Networks - legacy architecture



Virtual machines live migration



Live Migration Requirements:

1. **VM state remains unchanged** - no network reconfiguration inside VM
2. **IP and MAC addresses preserved** - VM keeps same network identity
3. **Active TCP connections maintained** - minimal disruption
4. **Gateway must remain reachable** - default route stays valid

What could go wrong?....

The challenge: Gateway MAC

IPv4 Issue:

- Gateway MAC is **node-specific**
- After migration to node2, VM **still has node1's MAC** in ARP cache

Workaround:

- Send a Gratuitous ARP after migration

The challenge: Router advertisements

IPv6 Issue:

- Multiple gateway routers send Router Advertisements (RA)
- VM learns multipath default gateway with multiple link-local addresses
- After migration + node shutdown → **one path is broken**

Workaround:

- Block external gateway RAs with iptables in every Layer2 pod
- Send unsolicited RAs to reconcile gateways

The challenge: list goes on...

Maintenance burden

- Doubled complexity for every new feature
- Harder to maintain and test
- Inconsistent behavior between topologies

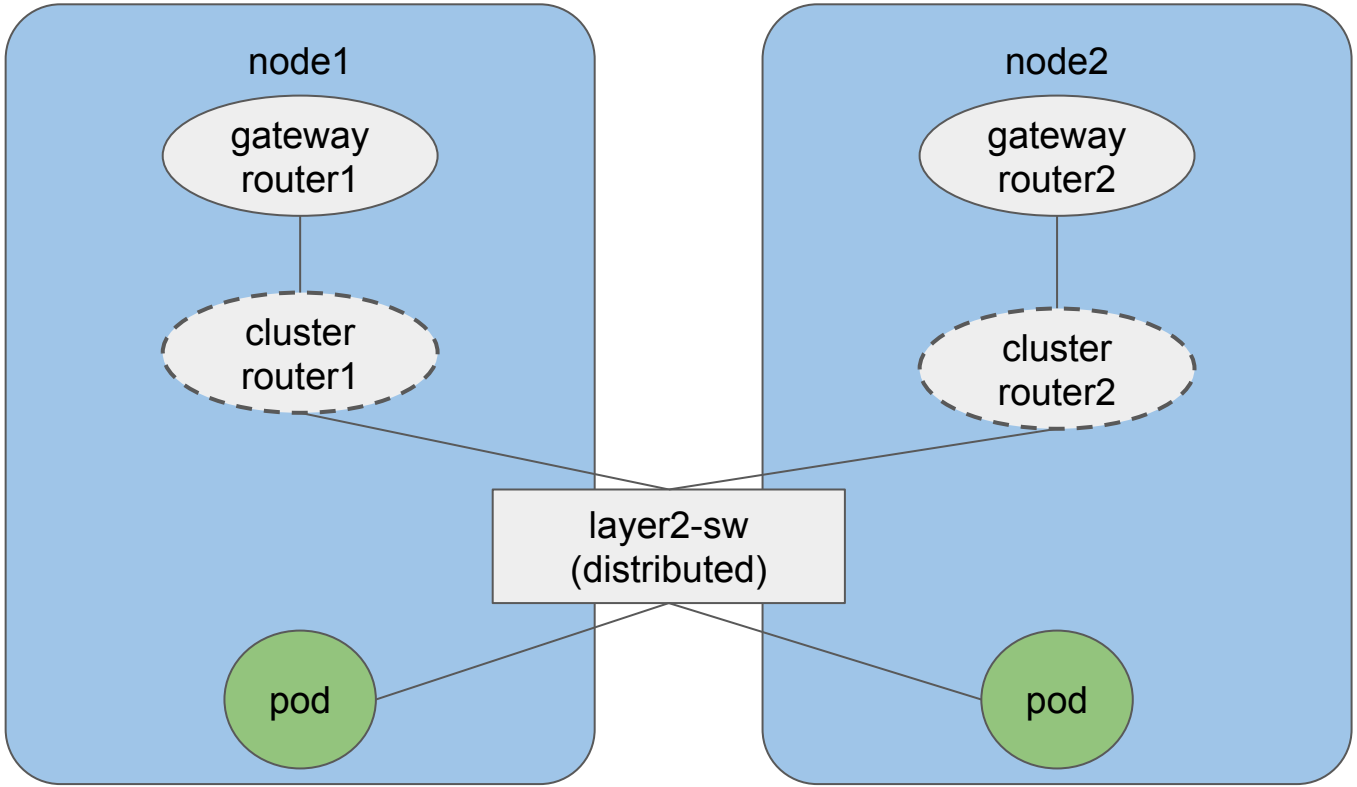
Connecting UDNs

- Gateway routers are node-specific giving us no common routing point between networks

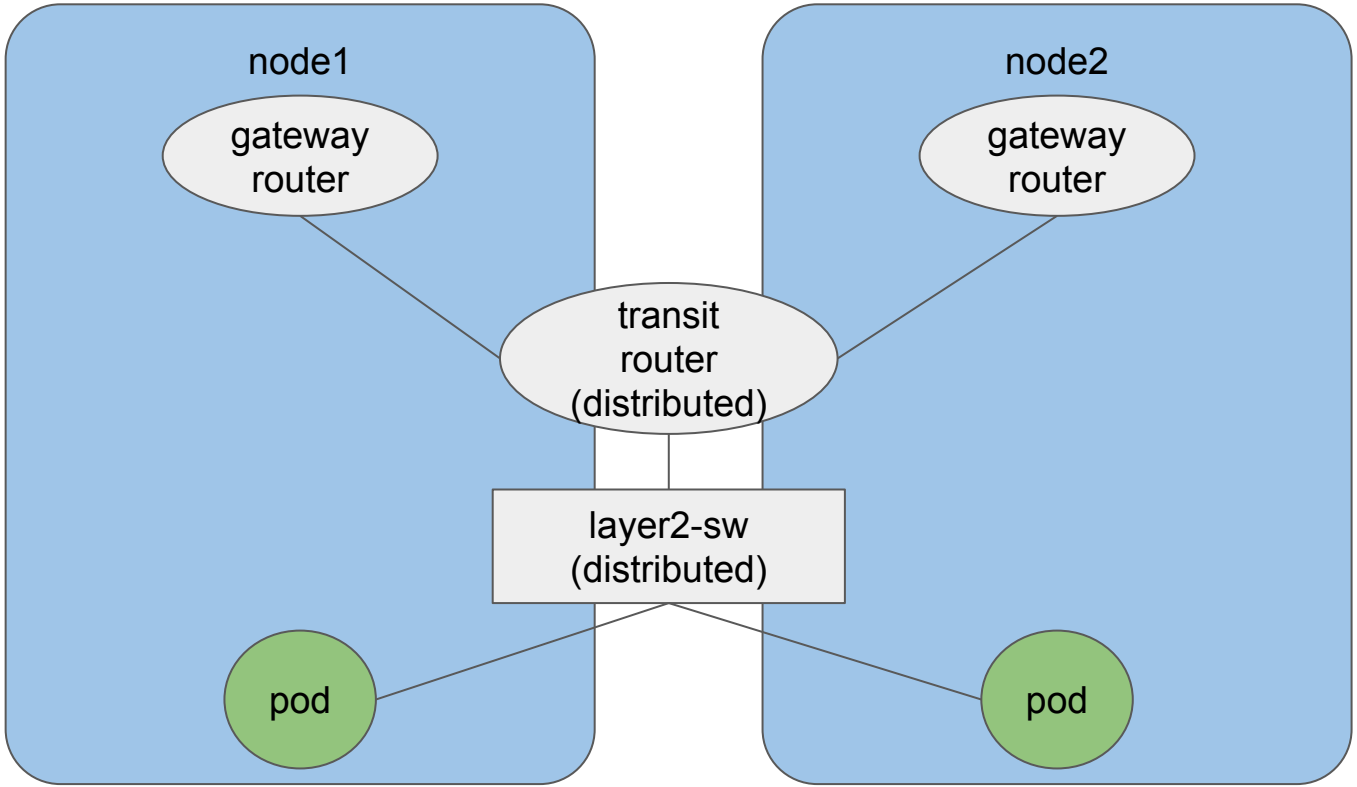
EgressIP

- A workaround configures the LRP for a pod on one of the EgressIP nodes to use the external gateway IP as one of the next hops to achieve proper load balancing...

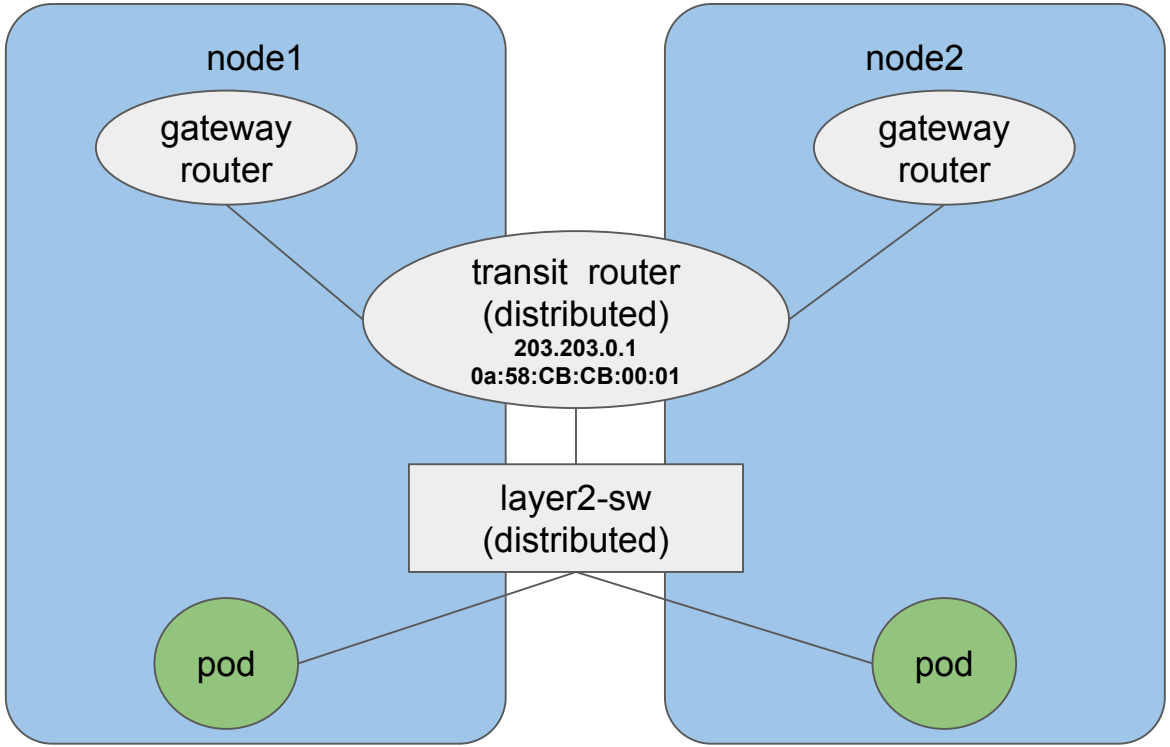
layer2 topology without transit router = :/



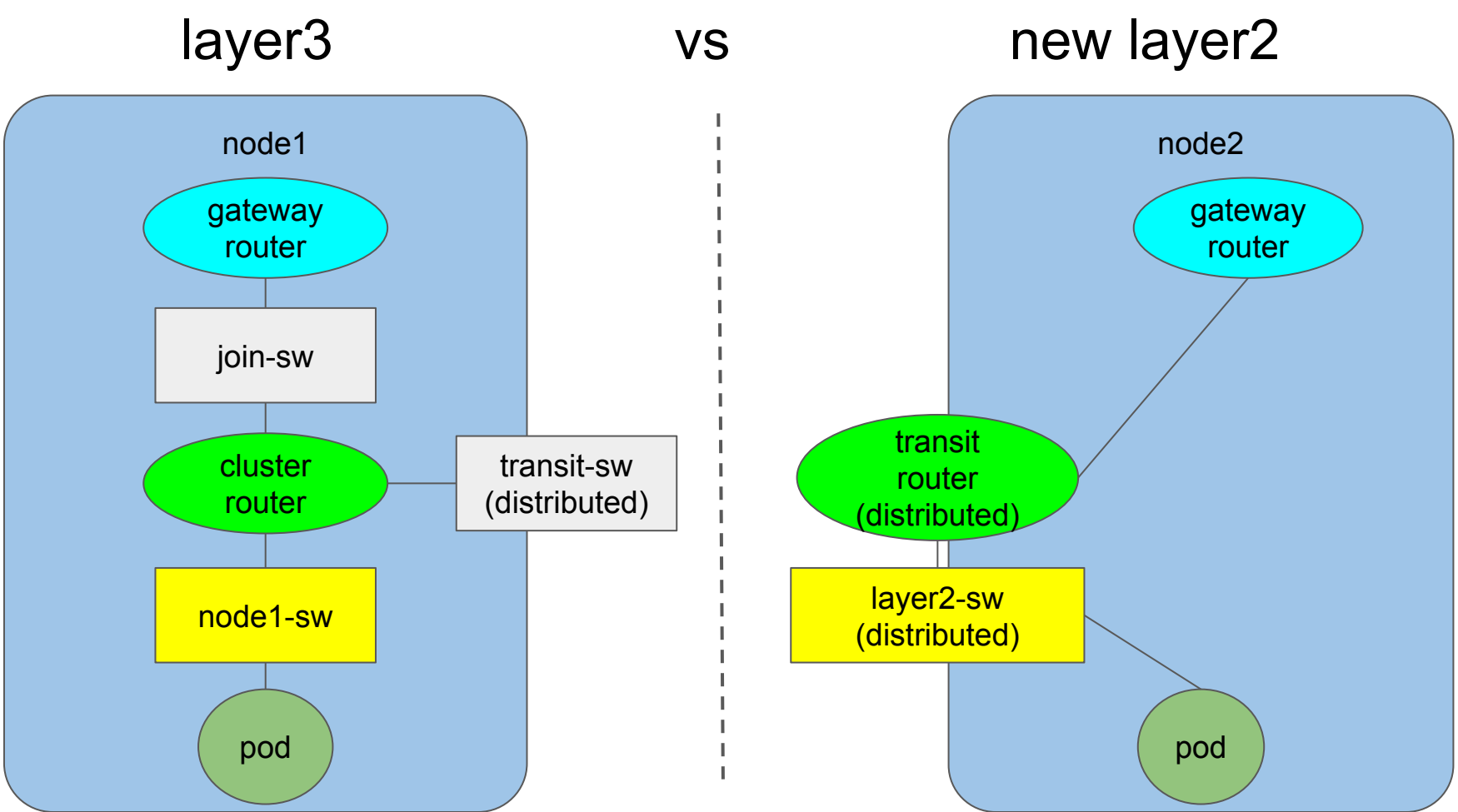
New layer2 topology + transit router = <3



VMs are now happy



Transit router is a default gateway for the network. It has the same IP and MAC on every node.


layer3
vs
new layer2
node1
gateway router
join-sw
cluster router
transit-sw (distributed)
node1-sw
pod
node2
gateway router
transit router (distributed)
layer2-sw (distributed)
pod

We need to upgrade this??

To avoid supporting old topology forever, we want to upgrade all existing networks to the new topology... **without any disruption.**

Live rewiring doesn't let you do that, but we have a k8s upgrade process:

- Rolling upgrade: nodes are upgraded one after the other
- Running workloads are migrated from the to-be-upgraded node
  - K8s node upgrade often includes reboot and restarts of different services
- This is our chance to rewire everything before workloads come back!

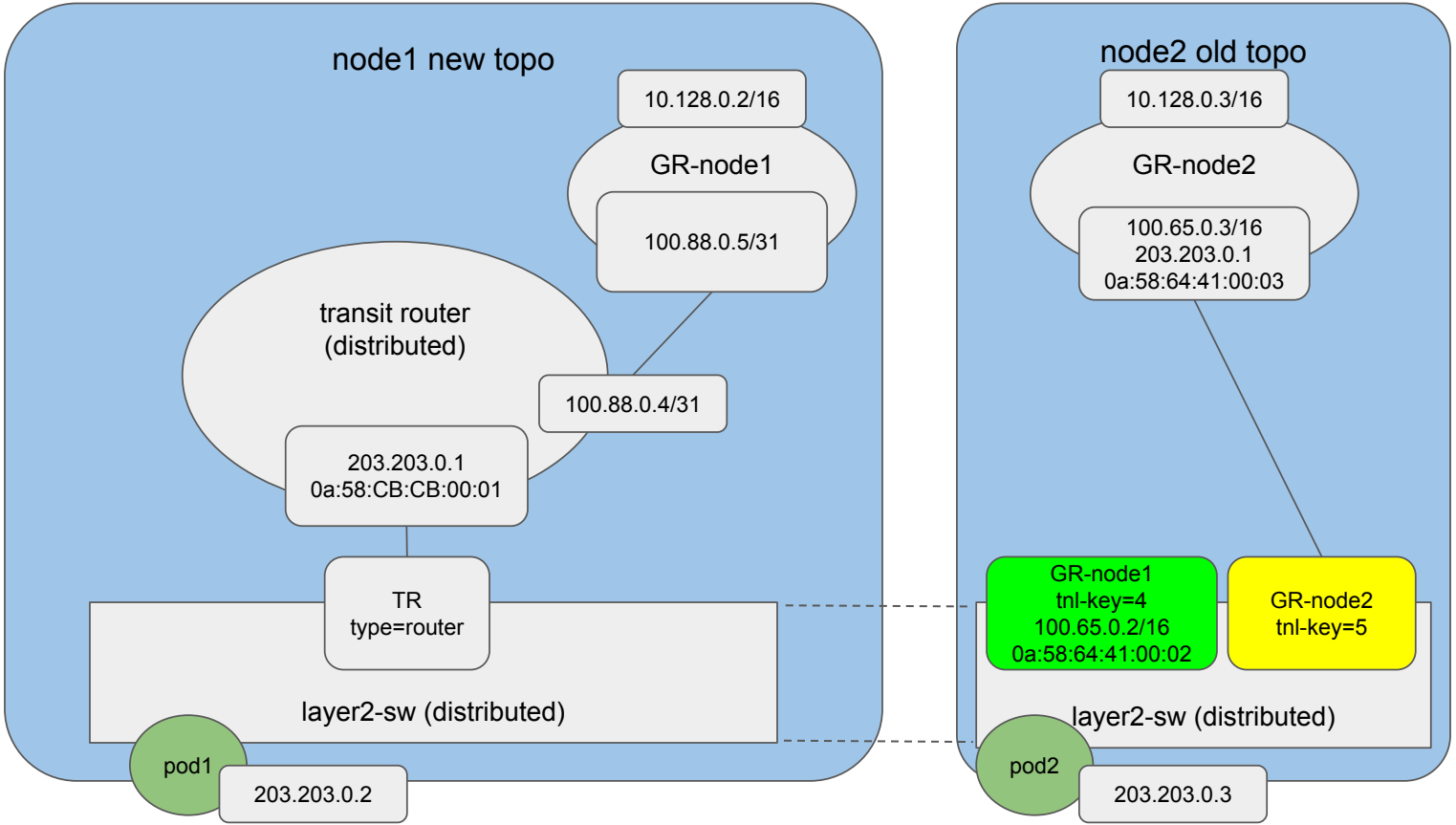
Upgrade topology: let the hacking begin

NOTE to viewers: *This is a work of fiction. Any similarity to actual persons, living or dead, or actual events, is purely coincidental.*

Well, not exactly true, this is based off actual slack conversations and long joint debugging sessions between **A LOT** of participants from OVN and OVN-Kubernetes communities.

Thanks to everyone who participated!

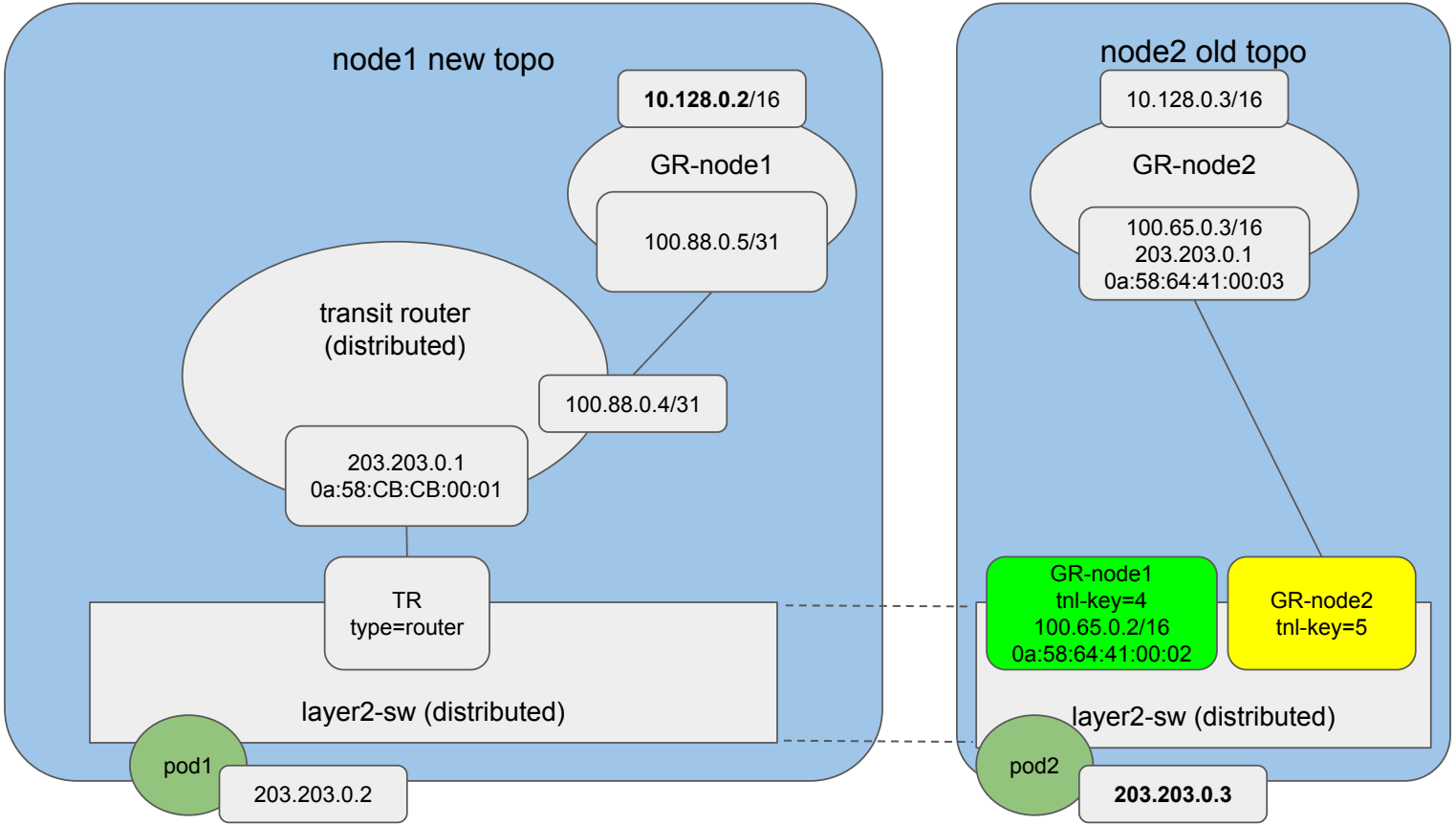
Layer2 - cluster partially upgraded



But does traffic actually flow correctly across the OVN interconnect?

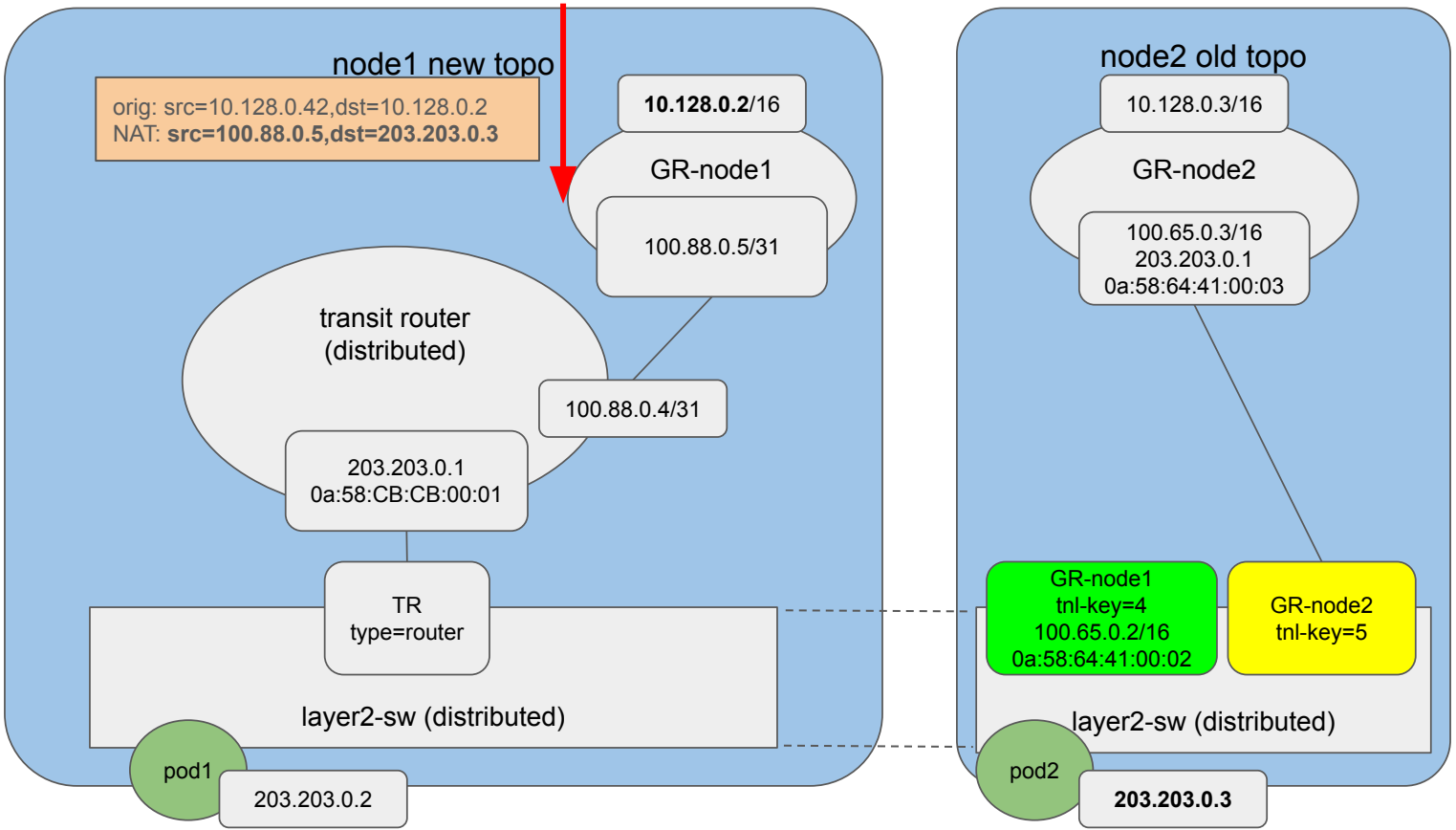
Traffic: external -> node1 -> pod2 (original direction)

External (10.128.0.42) -> GR-node1 (**10.128.0.2**, load balancer -> pod2 (**203.203.0.3**)



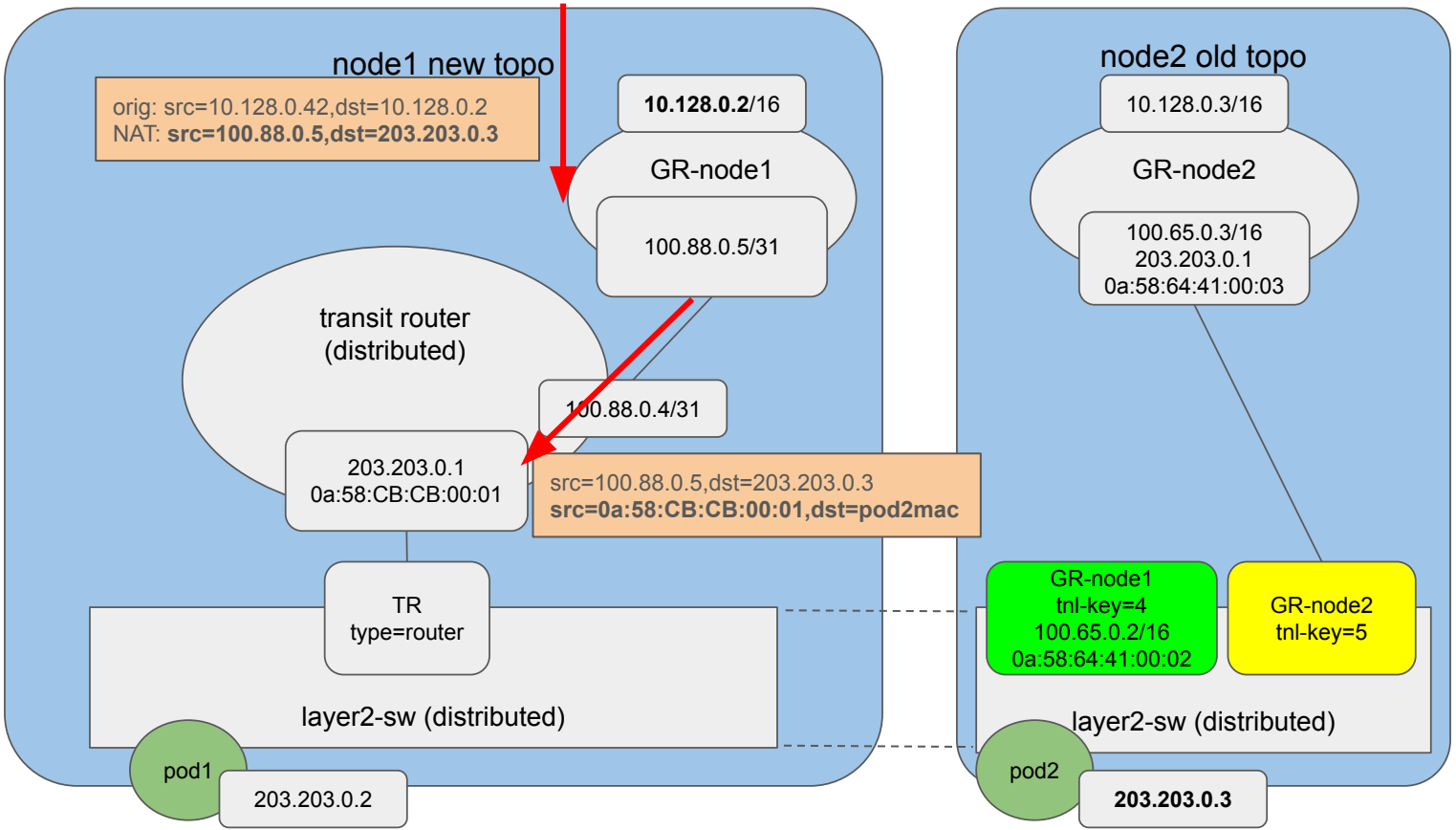
Traffic: external -> node1 -> pod2 (original direction)

External (10.128.0.42) -> GR-node1 (**10.128.0.2**, load balancer -> pod2 (**203.203.0.3**)



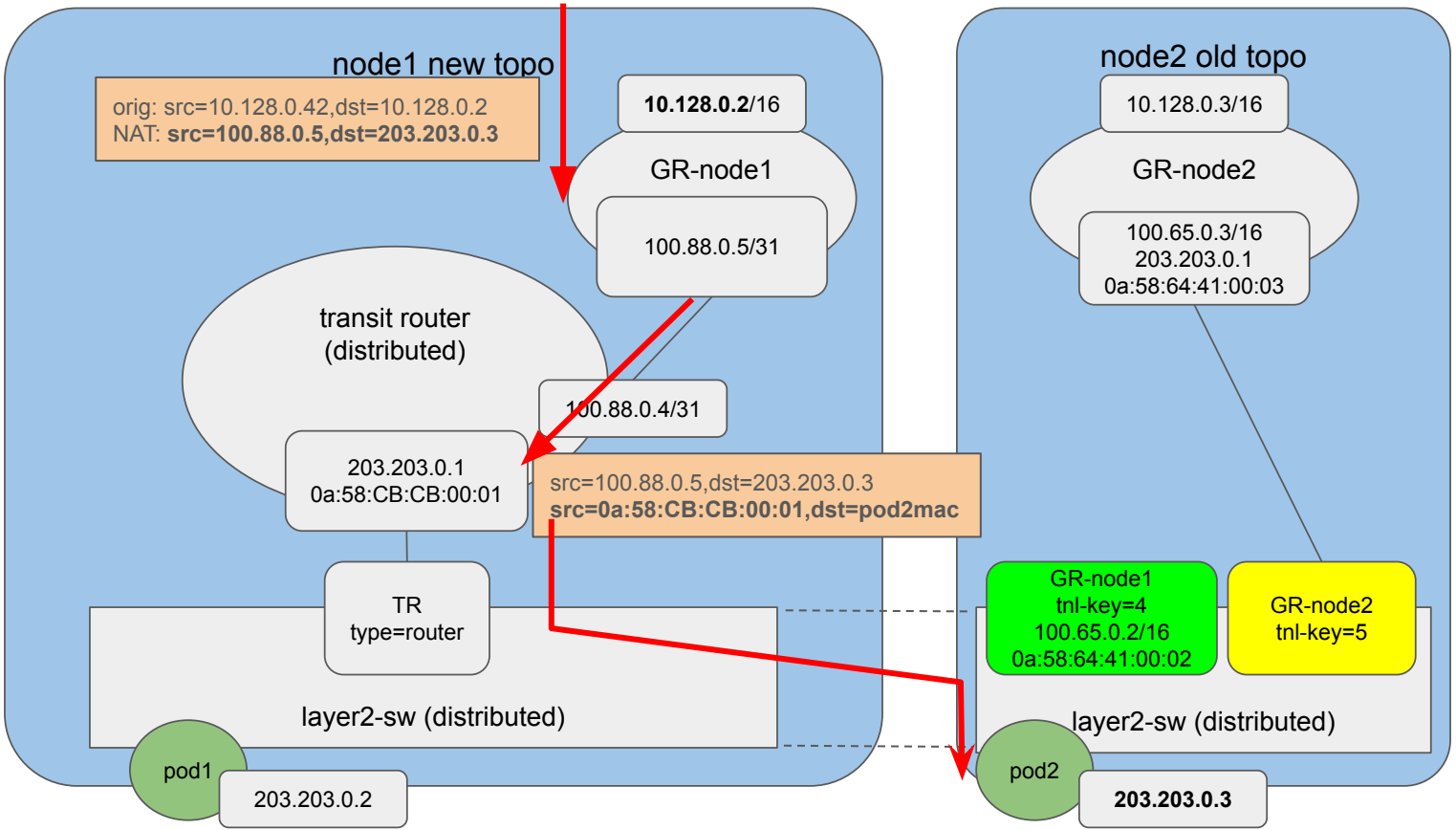
Traffic: external -> node1 -> pod2 (original direction)

External (10.128.0.42) -> GR-node1 (**10.128.0.2**, load balancer -> pod2 (**203.203.0.3**)



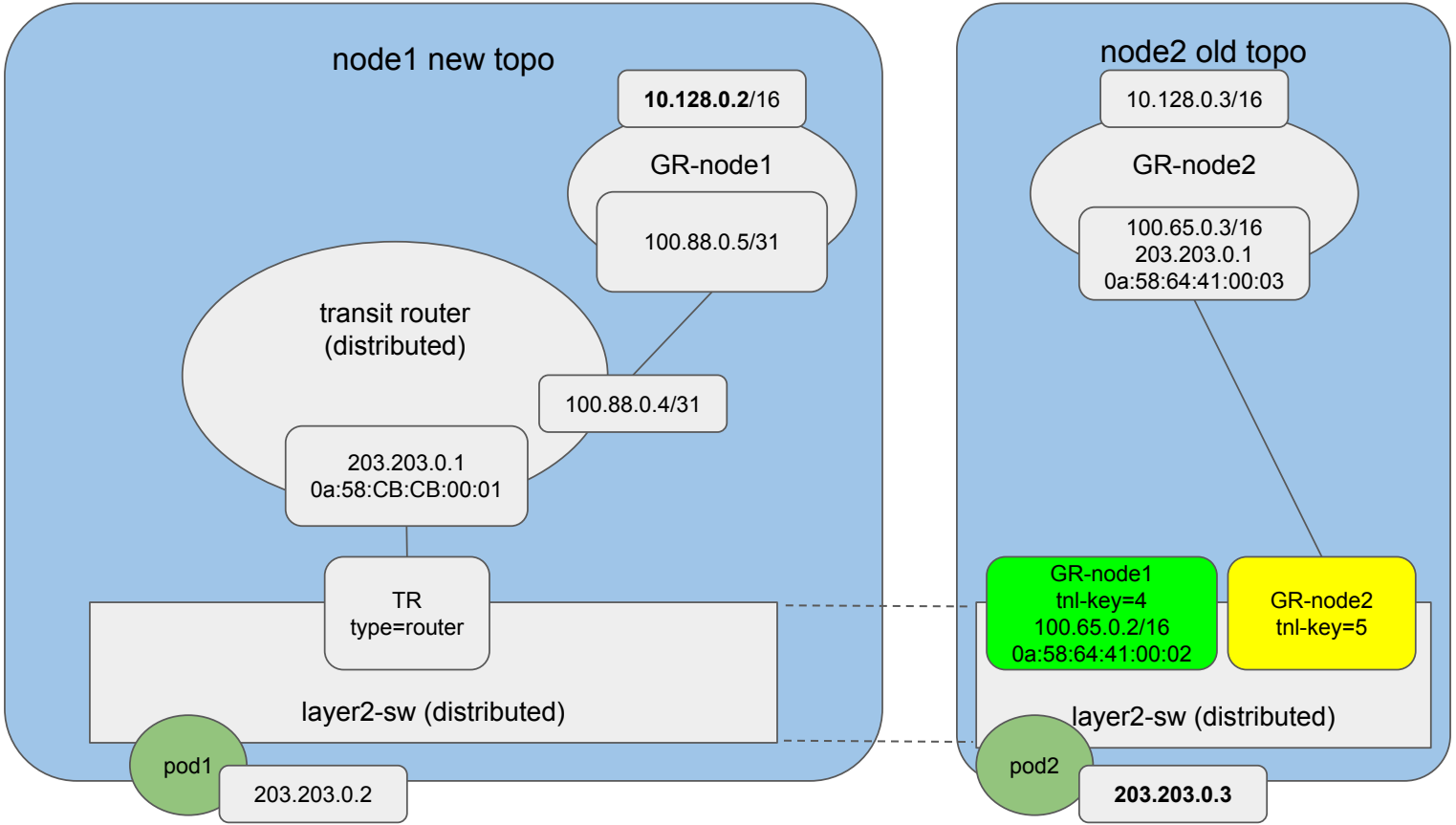
Traffic: external -> node1 -> pod2 (original direction)

External (10.128.0.42) -> GR-node1 (**10.128.0.2**, load balancer -> pod2 (**203.203.0.3**)



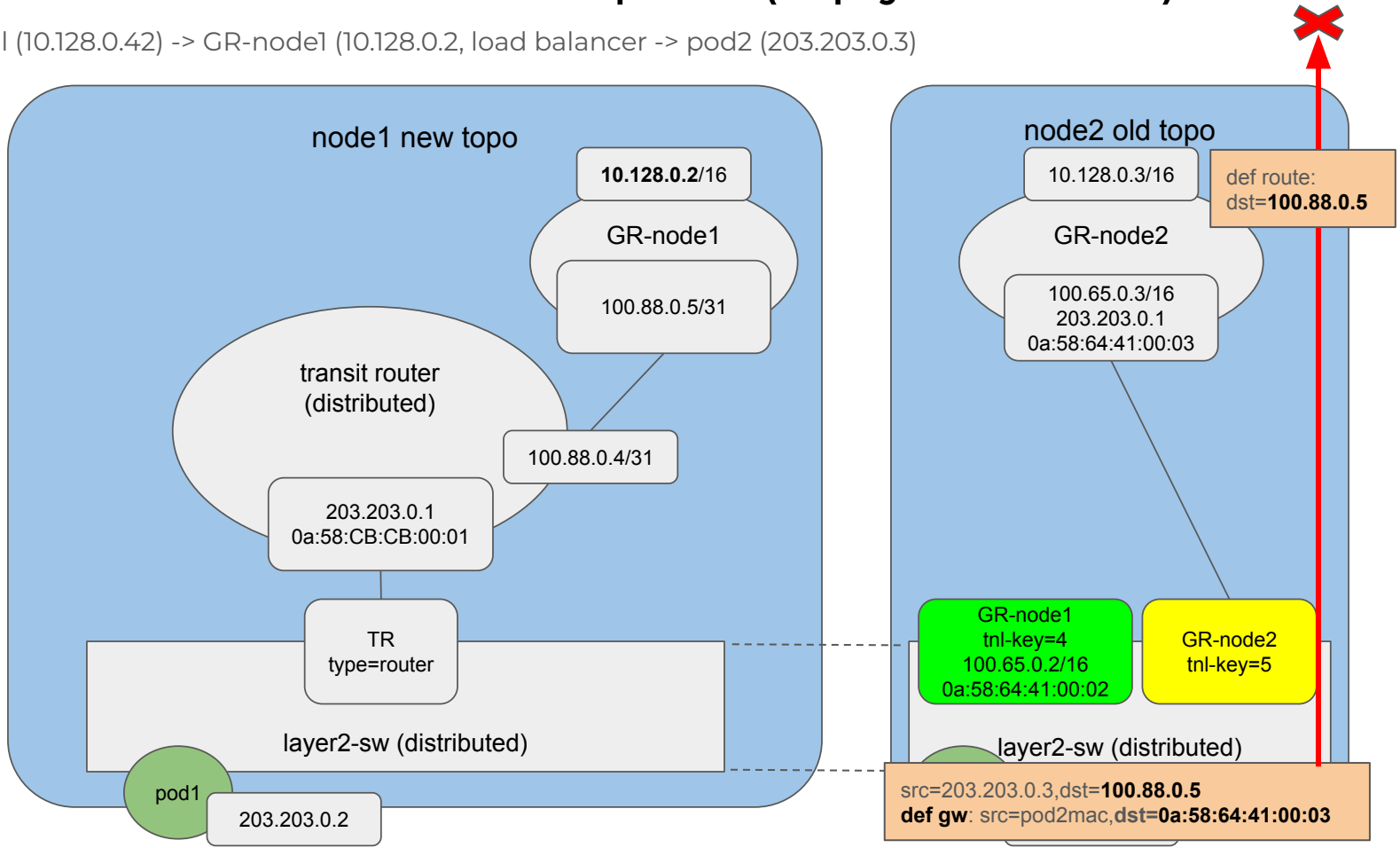
Traffic: external -> node1 -> pod2 (reply direction)

External (10.128.0.42) -> GR-node1 (10.128.0.2, load balancer -> pod2 (203.203.0.3)



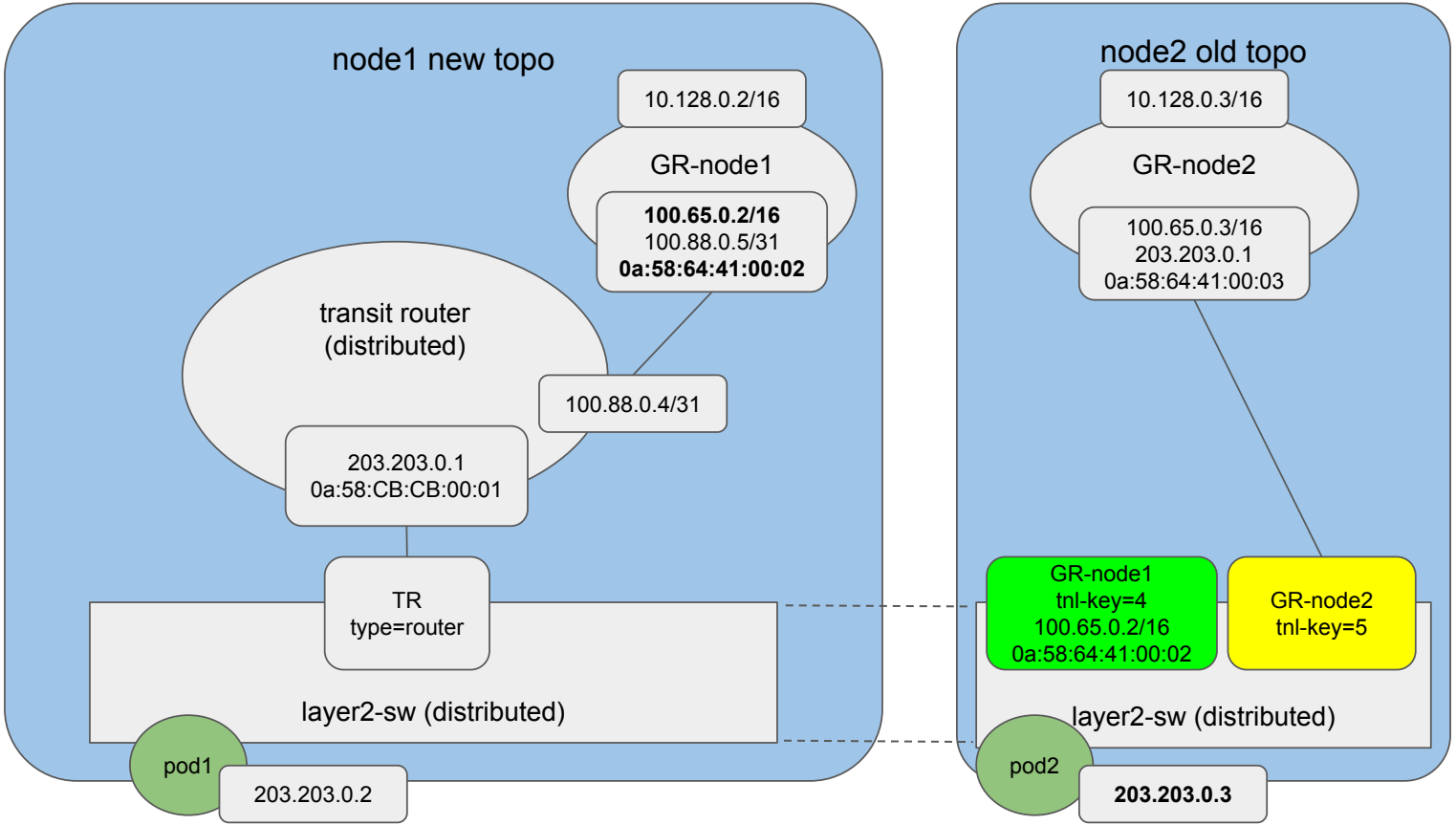
Traffic: external -> node1 -> pod2 (reply direction)

External (10.128.0.42) -> GR-node1 (10.128.0.2, load balancer -> pod2 (203.203.0.3)



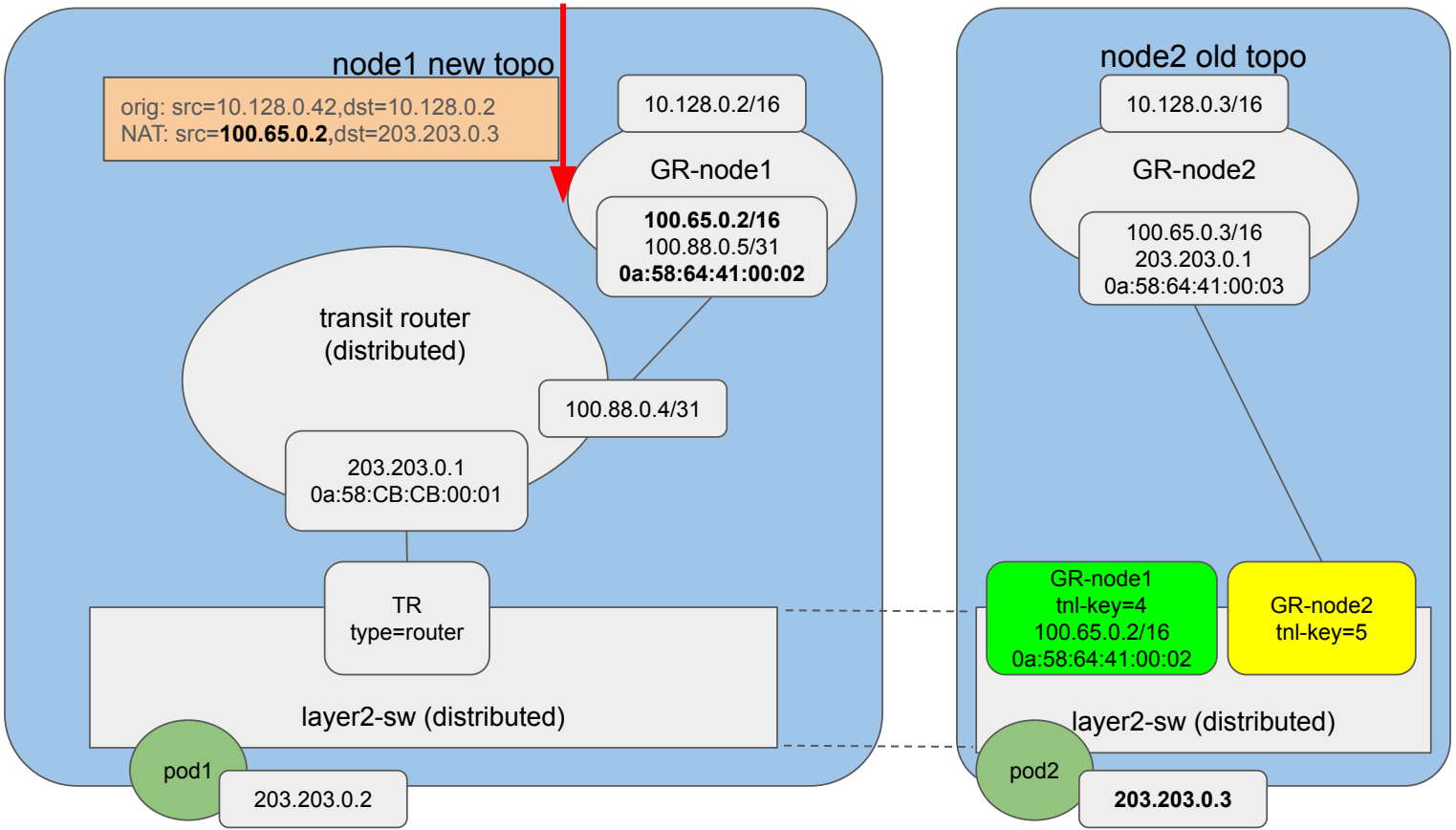
Traffic: external -> node1 -> pod2 (original direction), **take 2**

External (10.128.0.42) -> GR-node1 (10.128.0.2, load balancer -> pod2 (203.203.0.3)



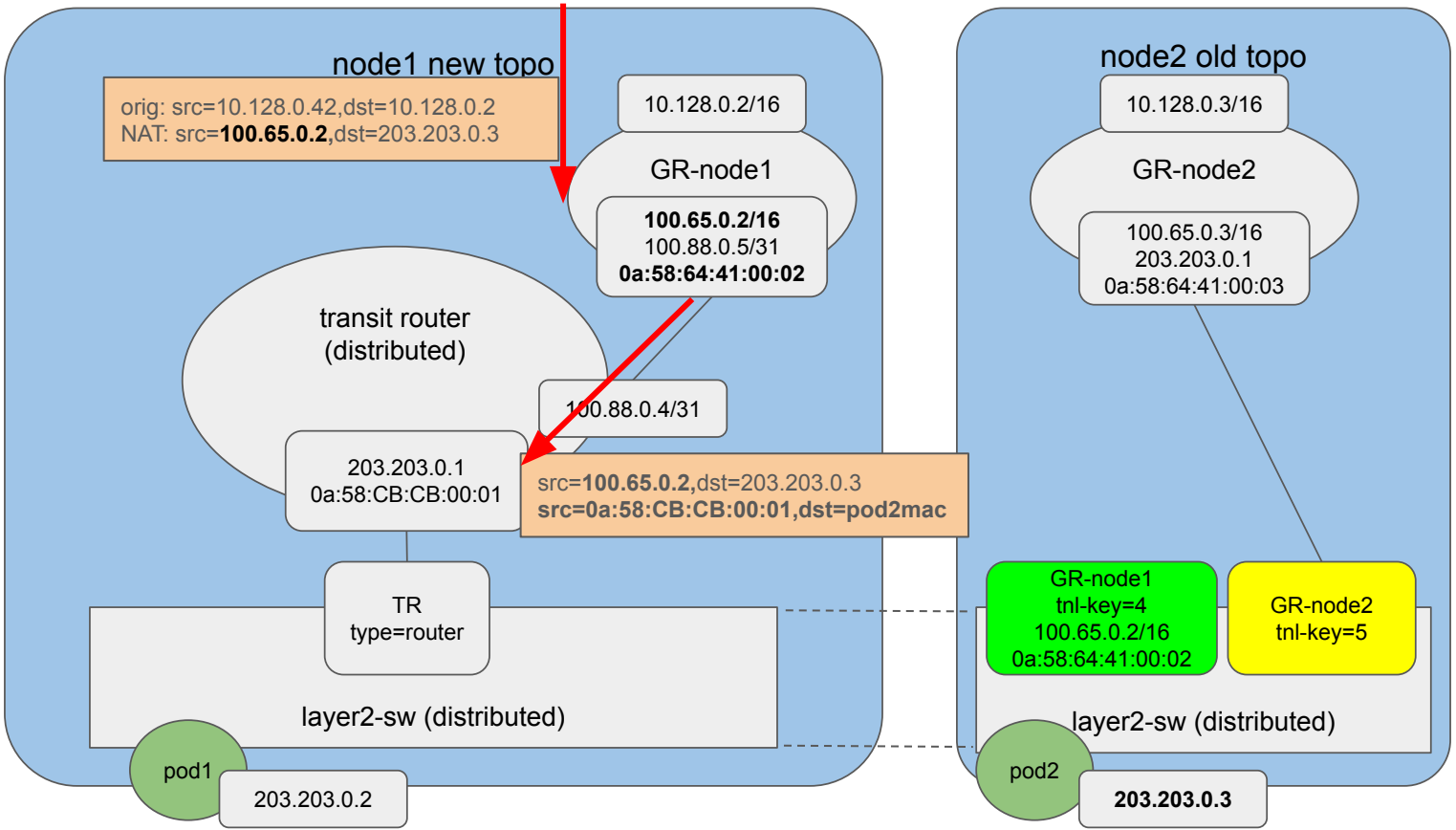
Traffic: external -> node1 -> pod2 (original direction), **take 2**

External (10.128.0.42) -> GR-node1 (10.128.0.2, load balancer -> pod2 (203.203.0.3)



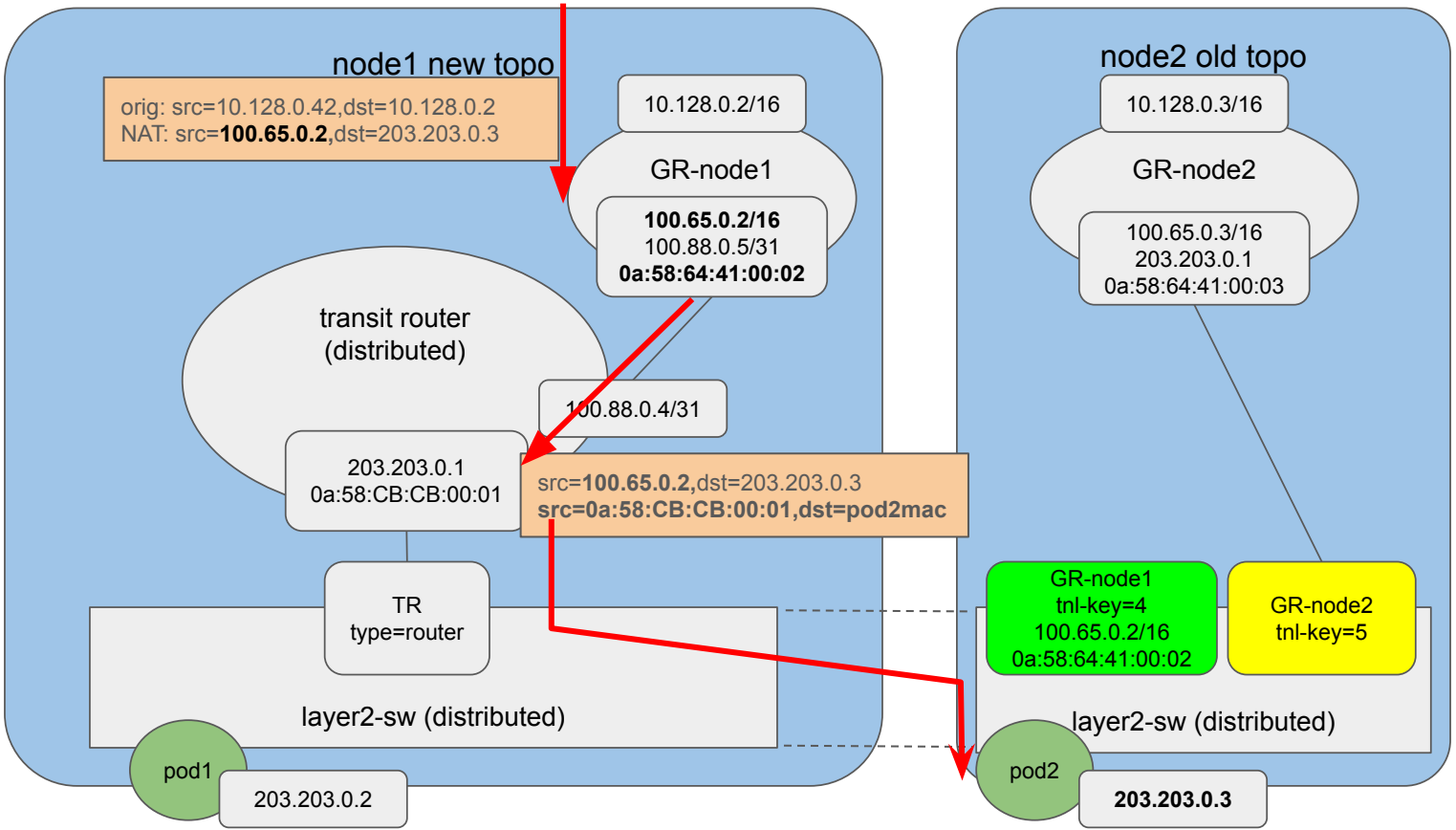
Traffic: external -> node1 -> pod2 (original direction), **take 2**

External (10.128.0.42) -> GR-node1 (10.128.0.2, load balancer -> pod2 (203.203.0.3)



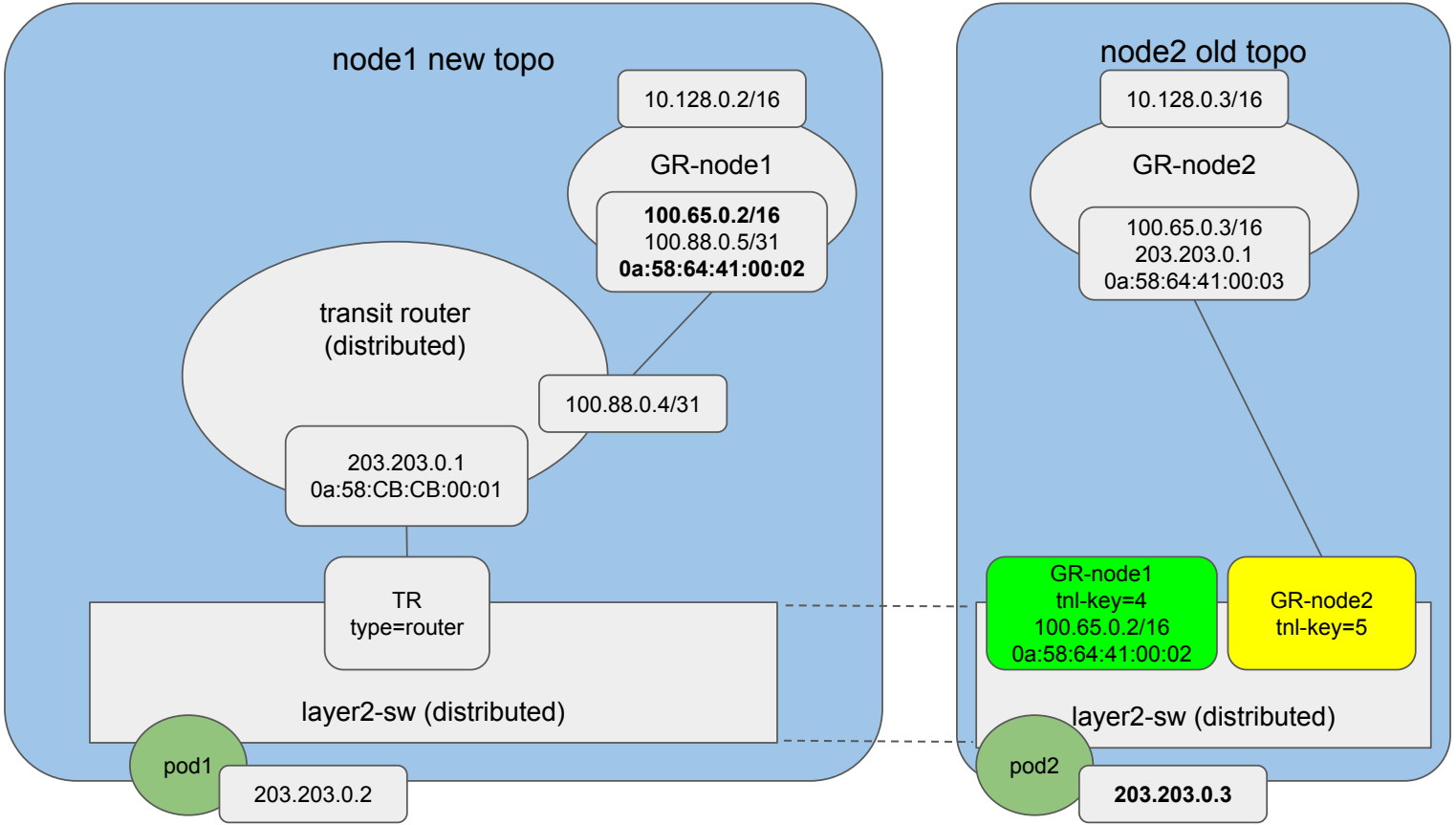
Traffic: external -> node1 -> pod2 (original direction), **take 2**

External (10.128.0.42) -> GR-node1 (10.128.0.2, load balancer -> pod2 (203.203.0.3)



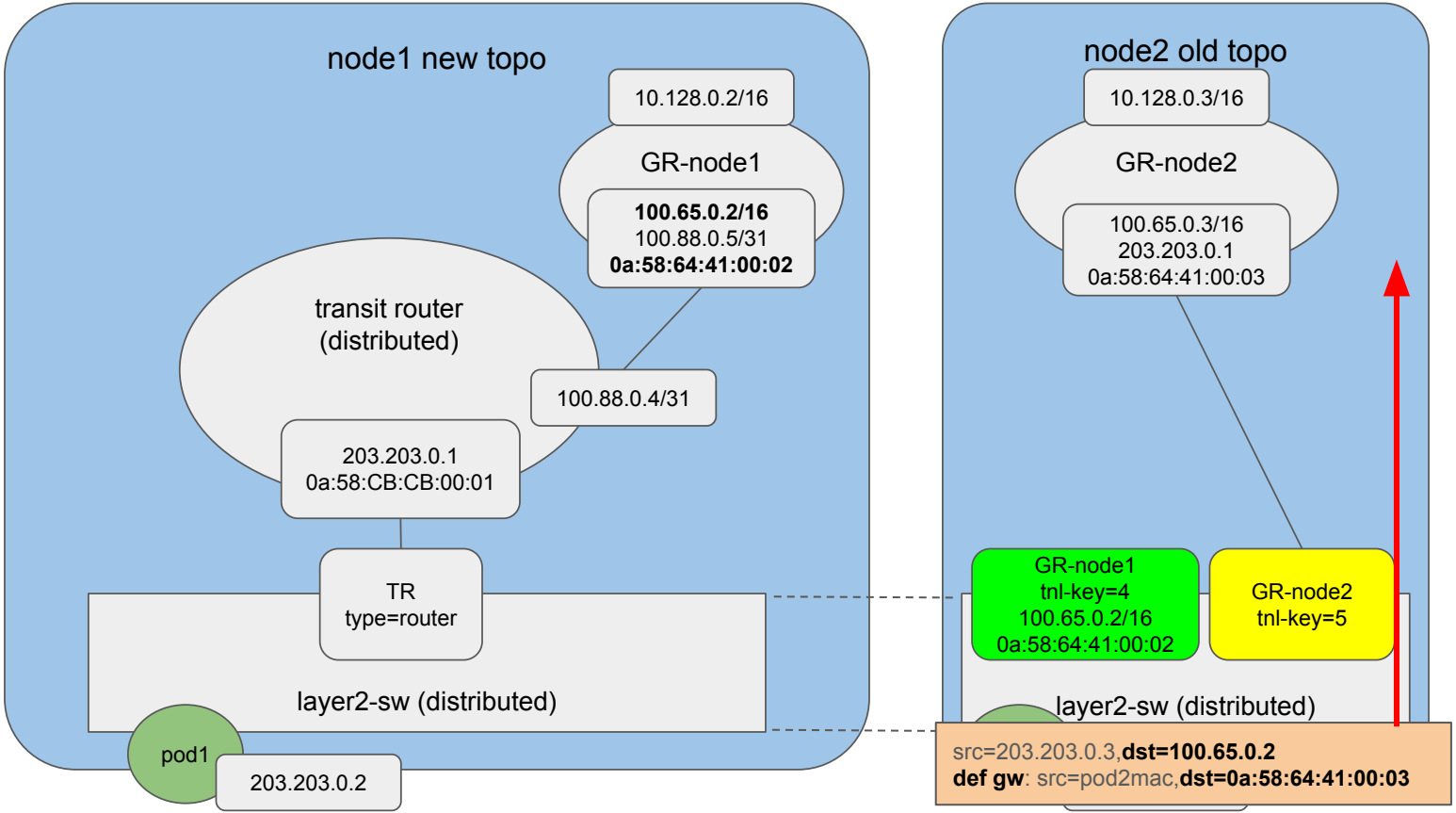
Traffic: external -> node1 -> pod2 (reply direction), **take 2**

External (10.128.0.42) -> GR-node1 (10.128.0.2, load balancer -> pod2 (203.203.0.3)



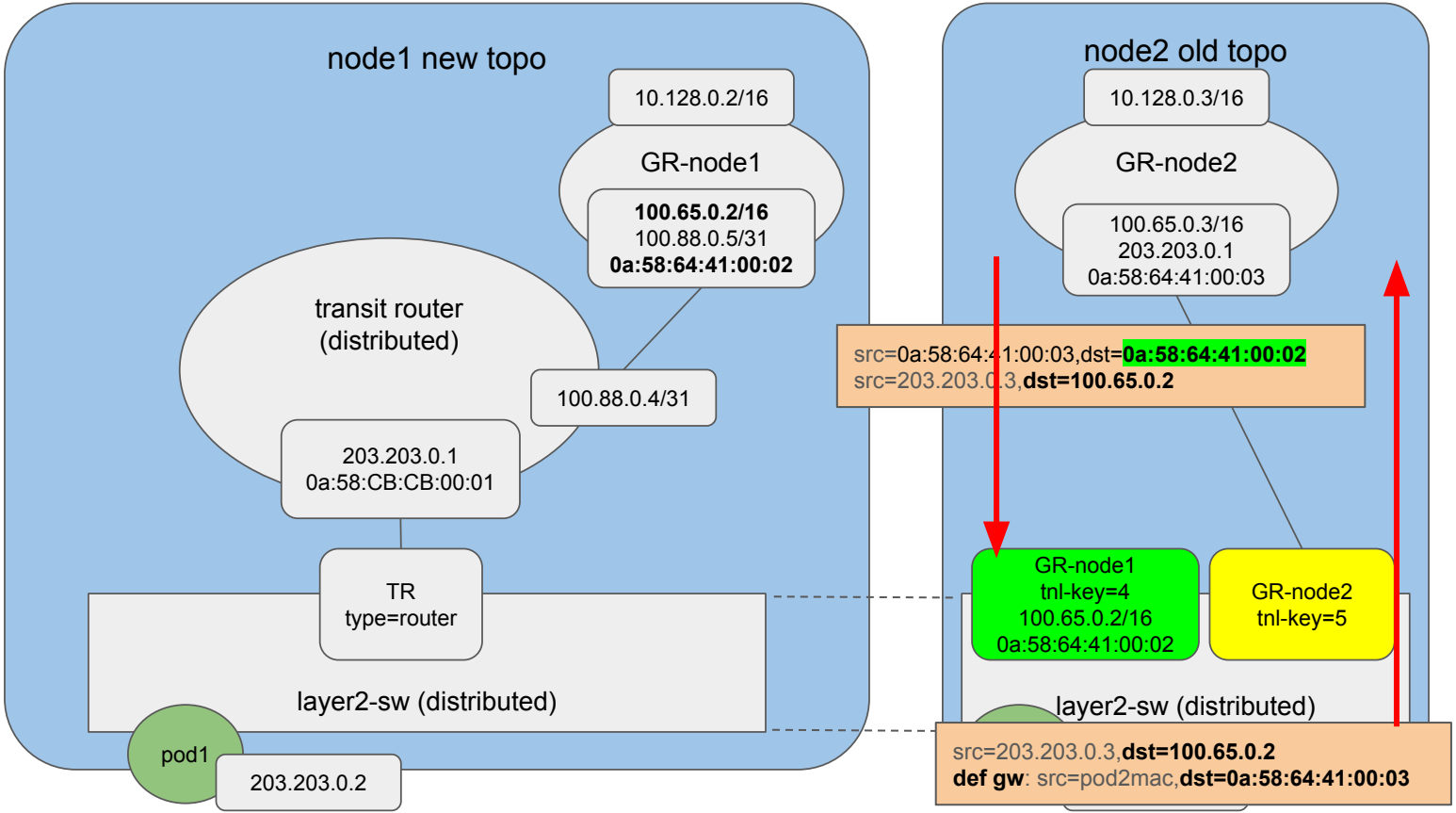
Traffic: external -> node1 -> pod2 (reply direction), **take 2**

External (10.128.0.42) -> GR-node1 (10.128.0.2, load balancer -> pod2 (203.203.0.3)



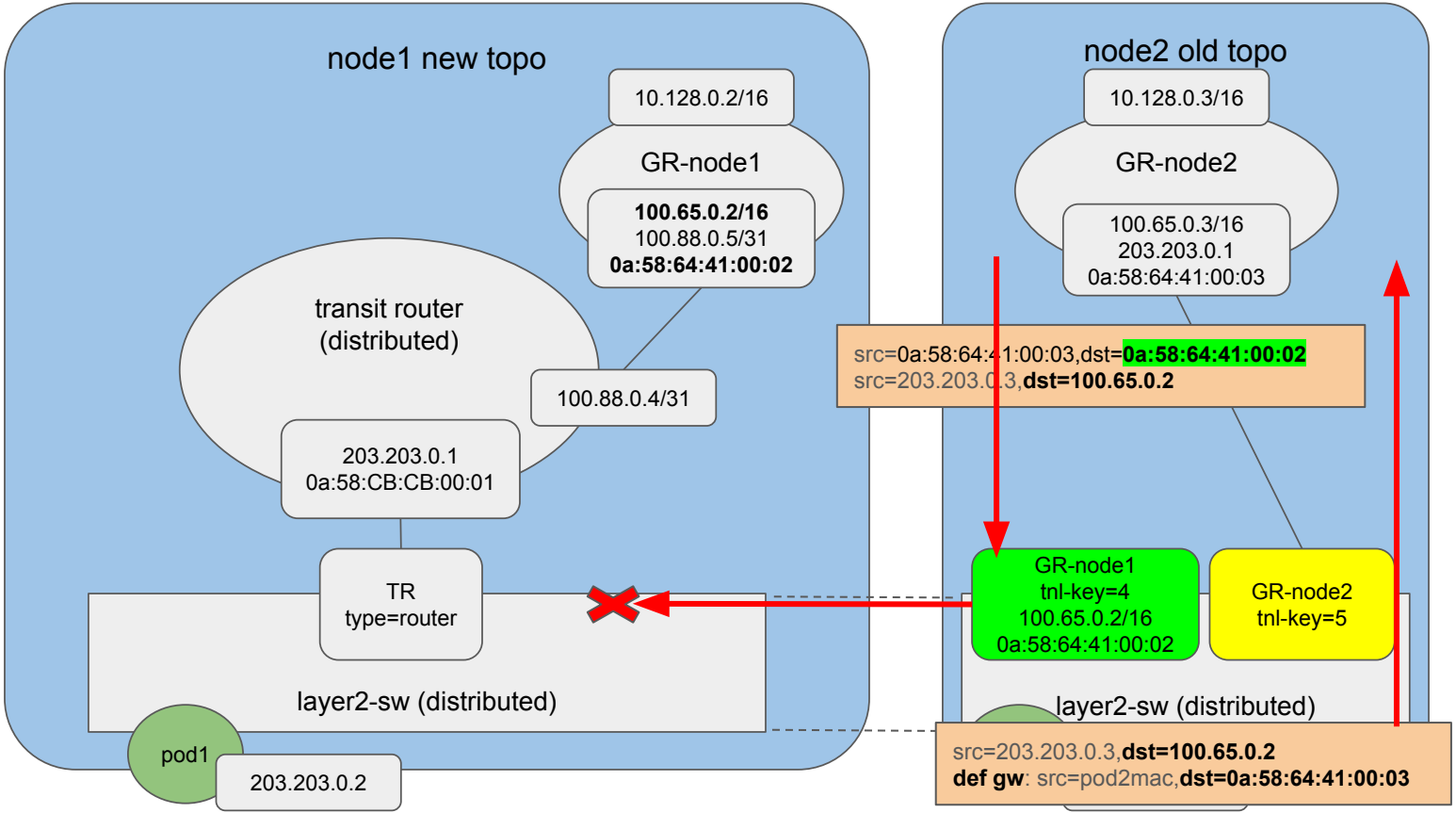
Traffic: external -> node1 -> pod2 (reply direction), **take 2**

External (10.128.0.42) -> GR-node1 (10.128.0.2, load balancer -> pod2 (203.203.0.3)



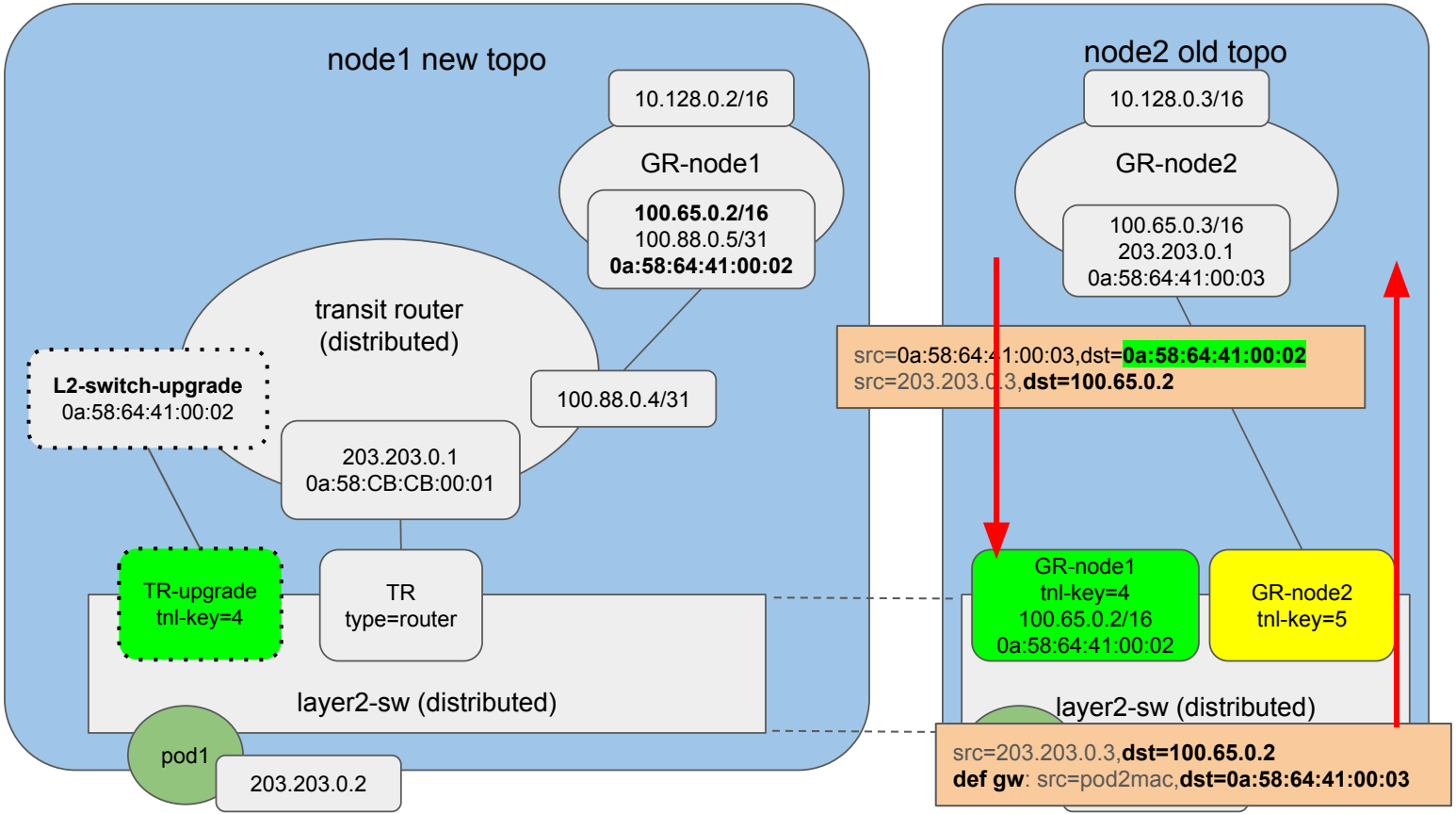
Traffic: external -> node1 -> pod2 (reply direction), **take 2**

External (10.128.0.42) -> GR-node1 (10.128.0.2, load balancer -> pod2 (203.203.0.3)



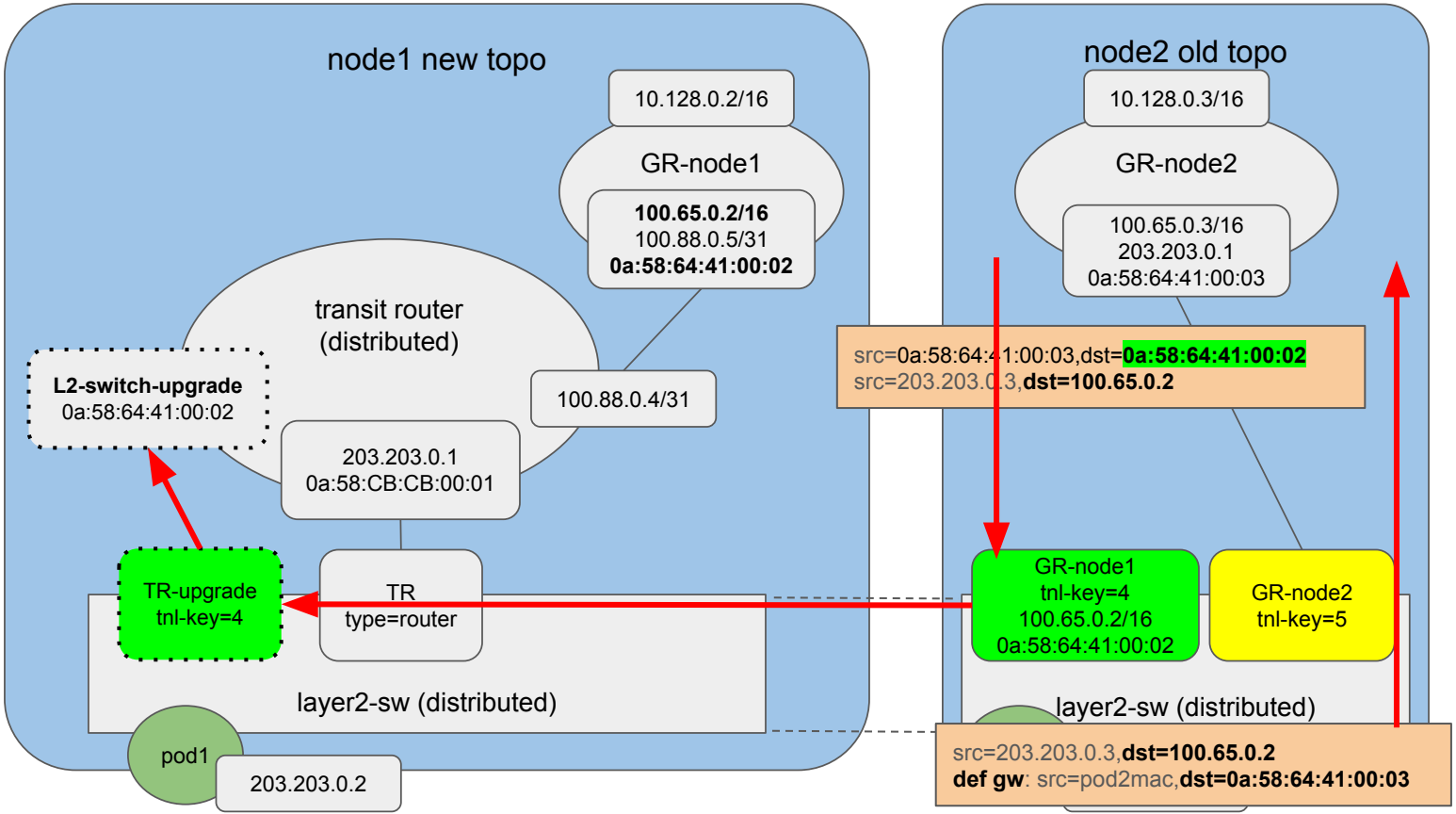
Traffic: external -> node1 -> pod2 (reply direction), **take 3**

External (10.128.0.42) -> GR-node1 (10.128.0.2, load balancer -> pod2 (203.203.0.3)



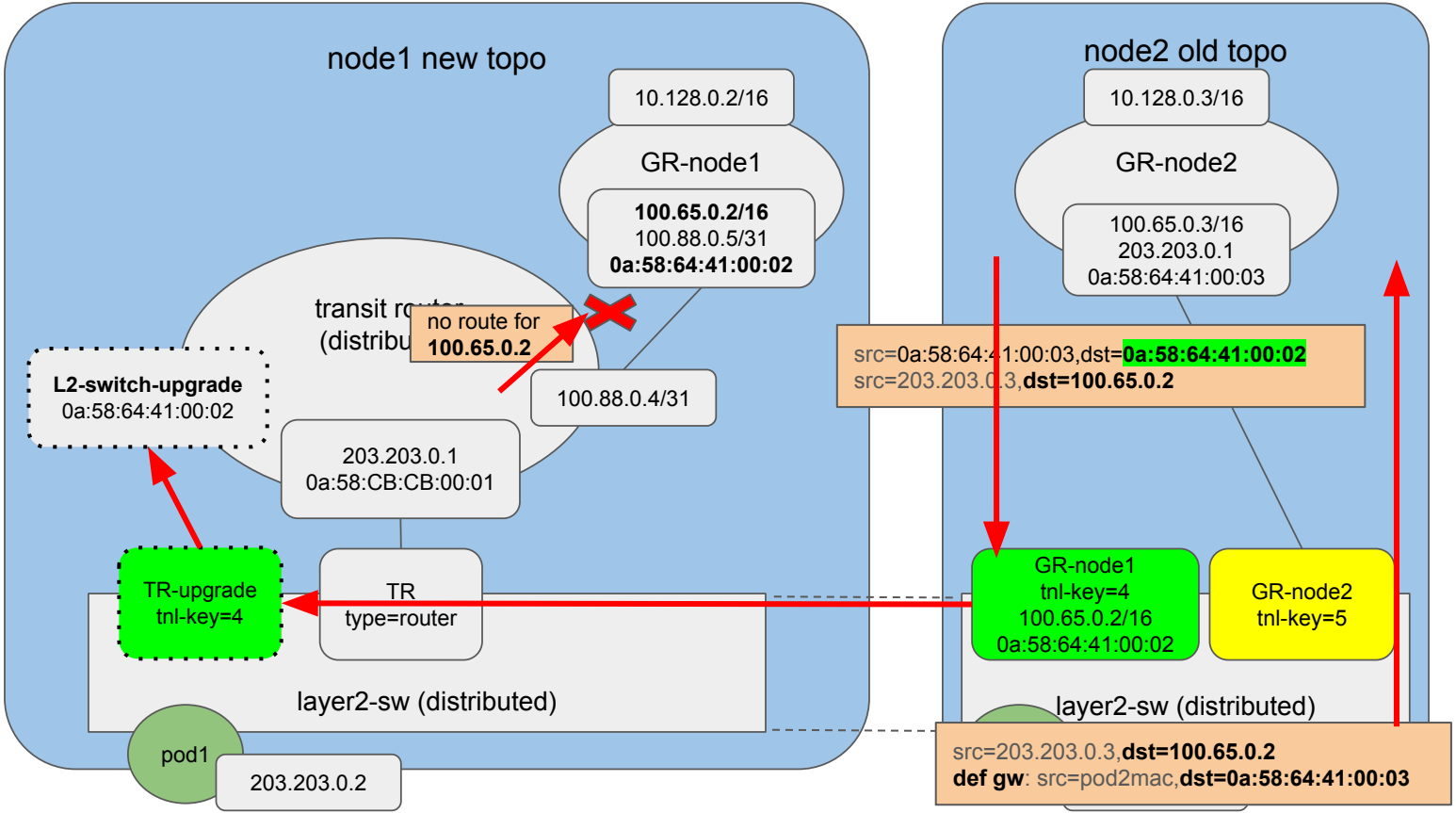
Traffic: external -> node1 -> pod2 (reply direction), **take 3**

External (10.128.0.42) -> GR-node1 (10.128.0.2, load balancer -> pod2 (203.203.0.3)



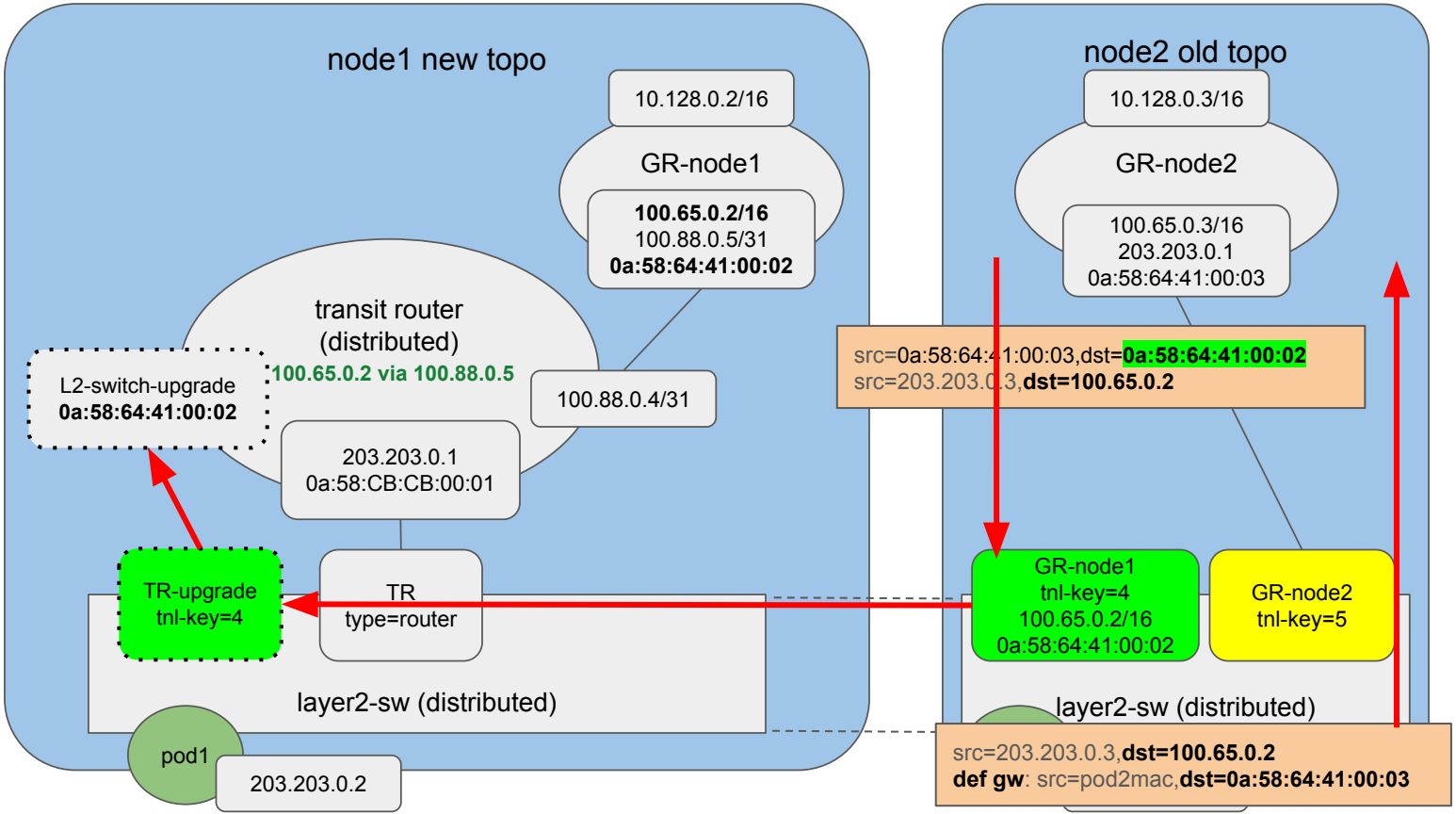
Traffic: external -> node1 -> pod2 (reply direction), **take 3**

External (10.128.0.42) -> GR-node1 (10.128.0.2, load balancer -> pod2 (203.203.0.3)



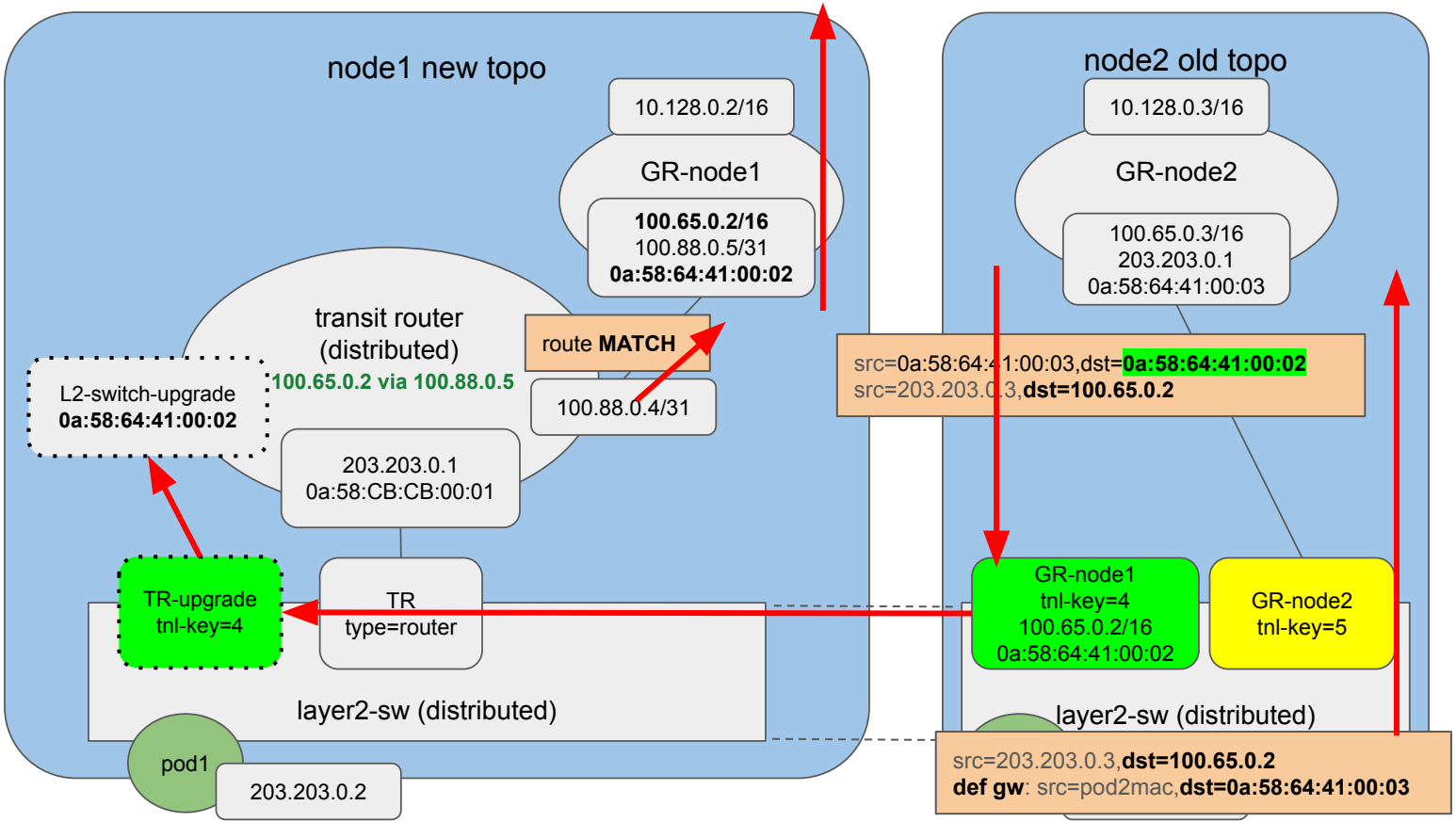
Traffic: external -> node1 -> pod2 (reply direction), **take 4**

External (10.128.0.42) -> GR-node1 (10.128.0.2, load balancer -> pod2 (203.203.0.3)



Traffic: external -> node1 -> pod2 (reply direction), **take 4**

External (10.128.0.42) -> GR-node1 (10.128.0.2, load balancer -> pod2 (203.203.0.3)



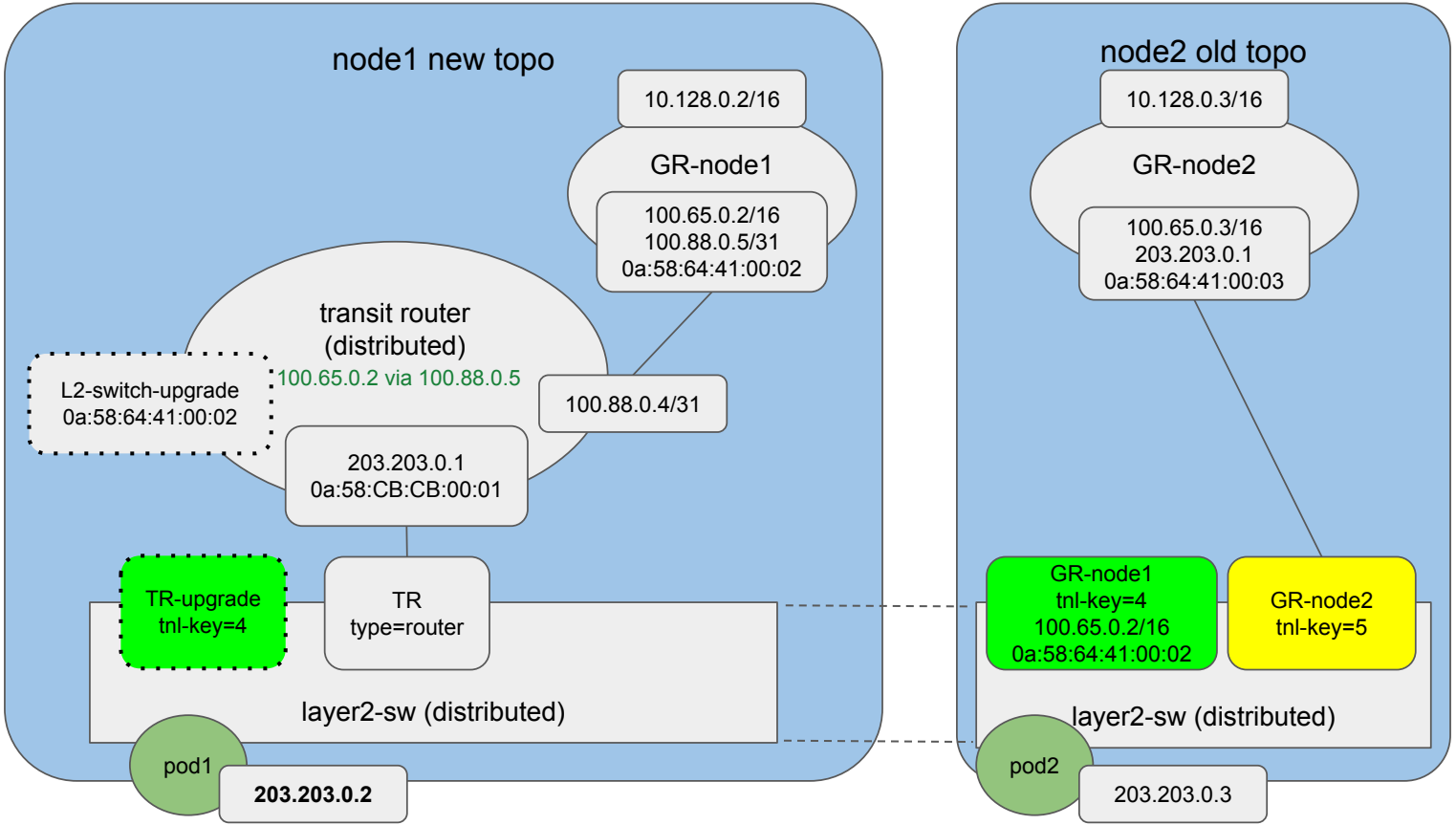
Traffic entering the cluster via the “old” node towards the “new” pod will work right out of the box, right?

Traffic entering the cluster via the “old” node towards the “new” pod will work right out of the box, right?

RIGHT???

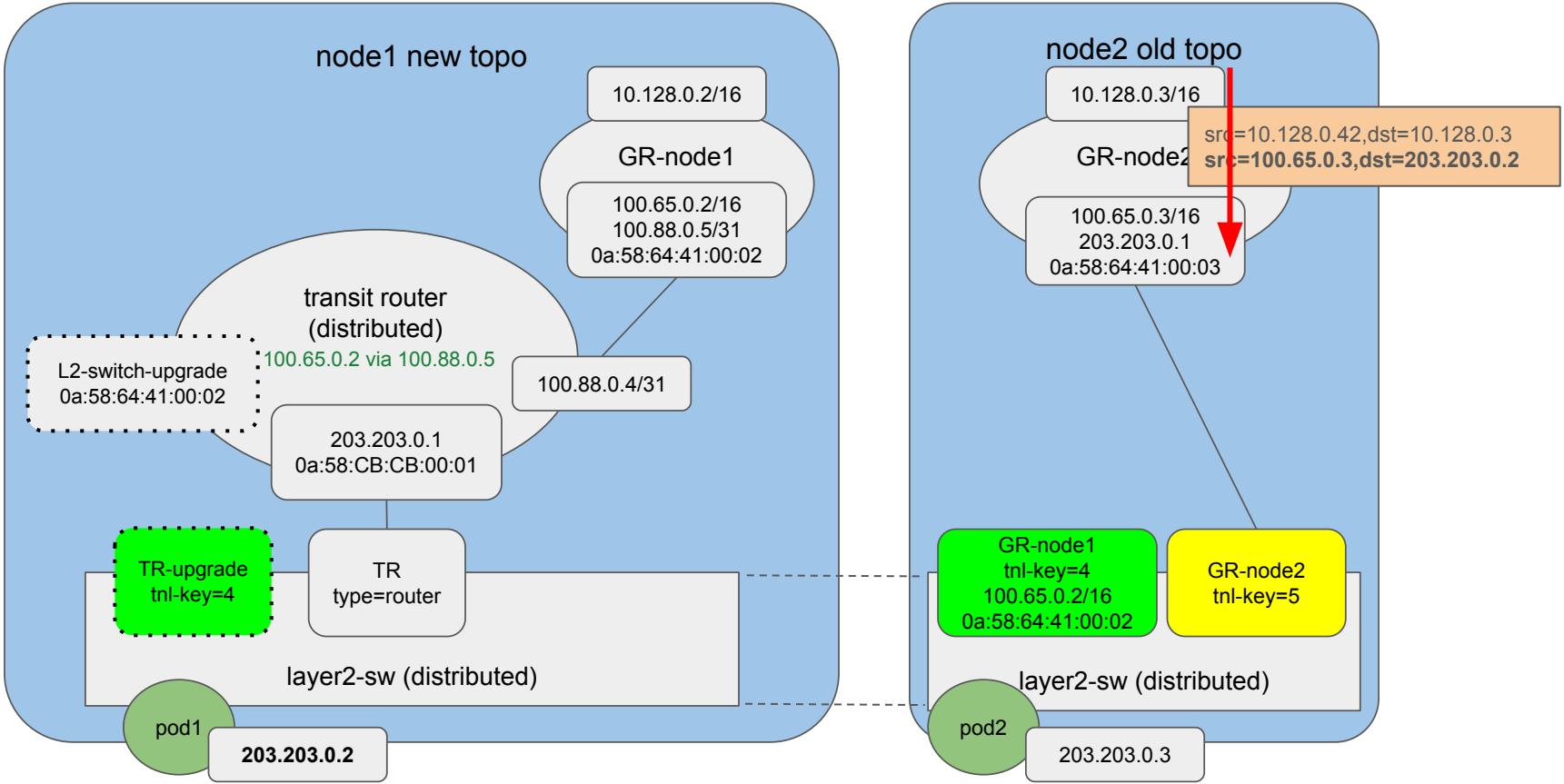
Traffic: external -> node2 -> pod1 (original direction)

external (10.128.0.42) -> GR-node2 (10.128.0.3, load balancer -> pod2 (203.203.0.2)



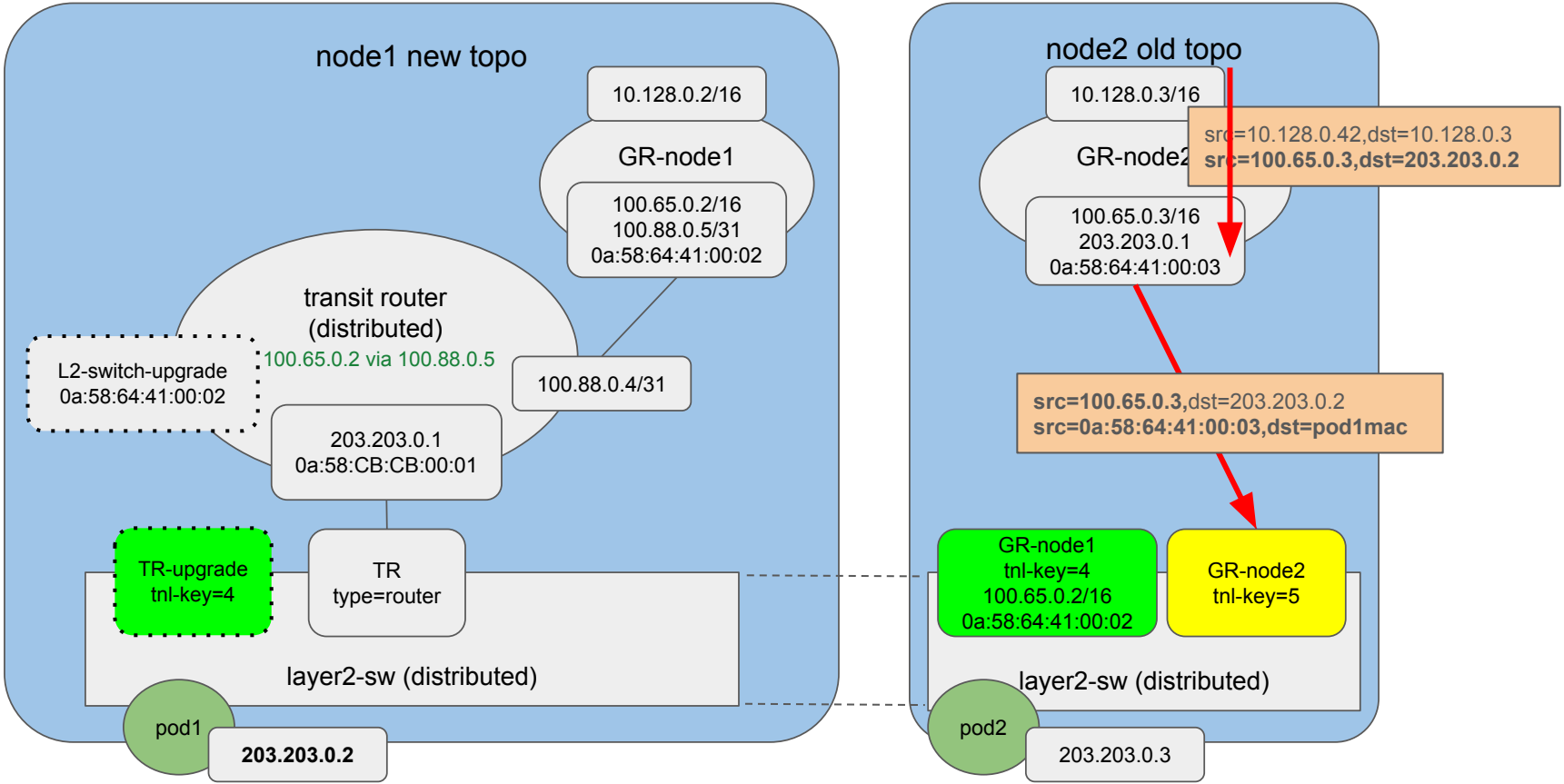
Traffic: external -> node2 -> pod1 (original direction)

external (10.128.0.42) -> GR-node2 (10.128.0.3, load balancer -> pod2 (203.203.0.2)



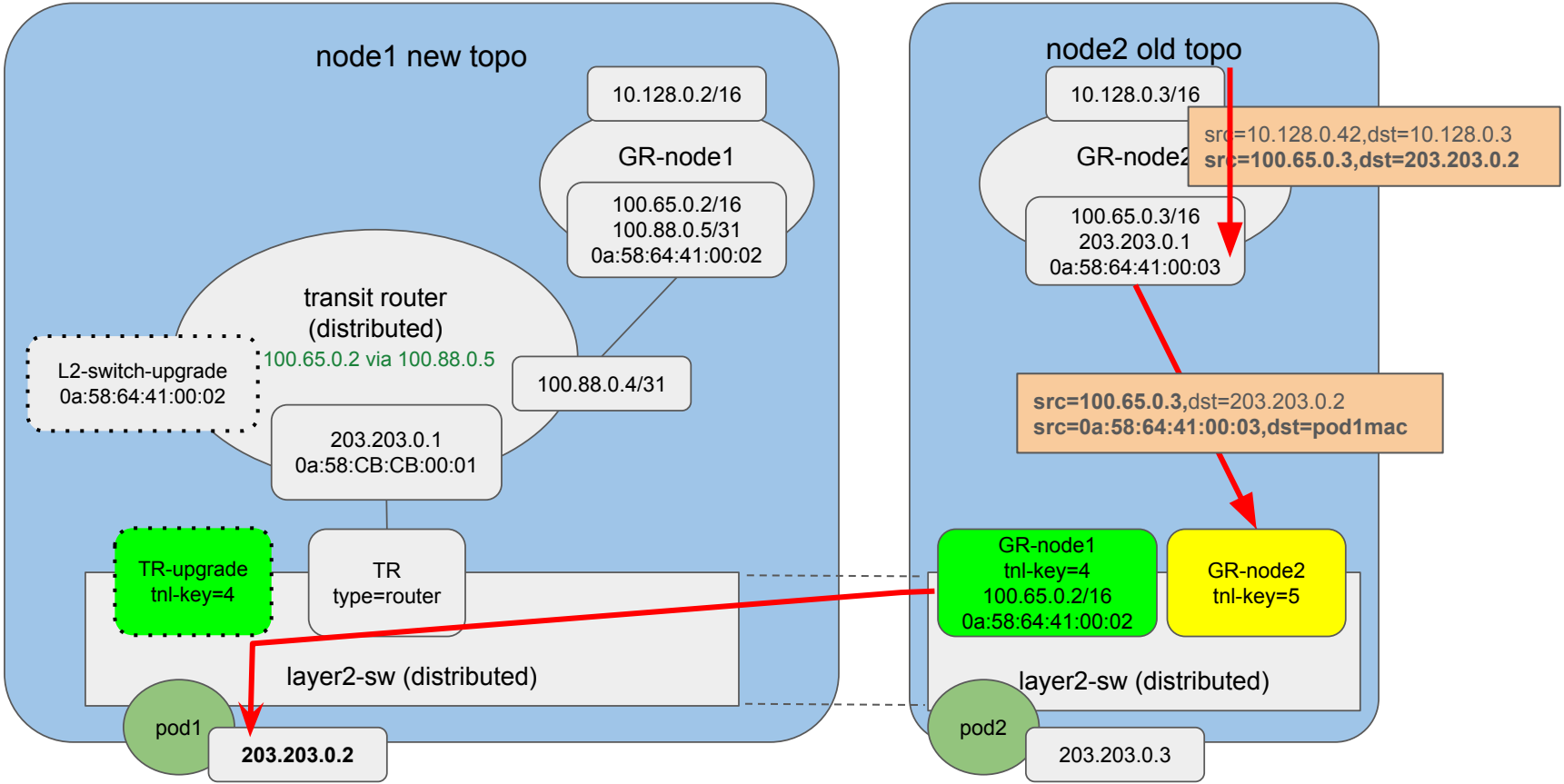
Traffic: external -> node2 -> pod1 (original direction)

external (10.128.0.42) -> GR-node2 (10.128.0.3, load balancer -> pod2 (203.203.0.2)



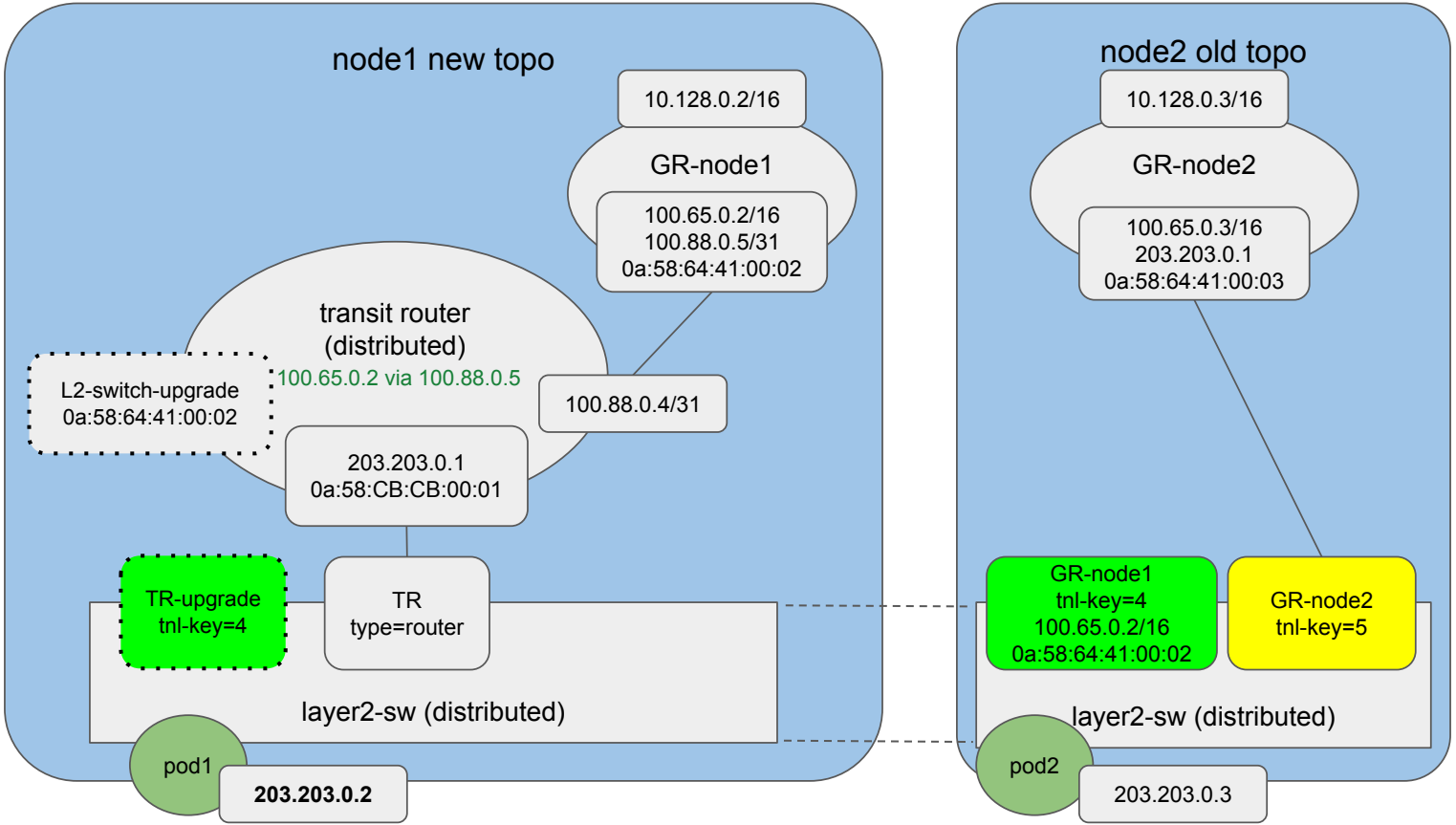
Traffic: external -> node2 -> pod1 (original direction)

external (10.128.0.42) -> GR-node2 (10.128.0.3, load balancer -> pod2 (203.203.0.2)



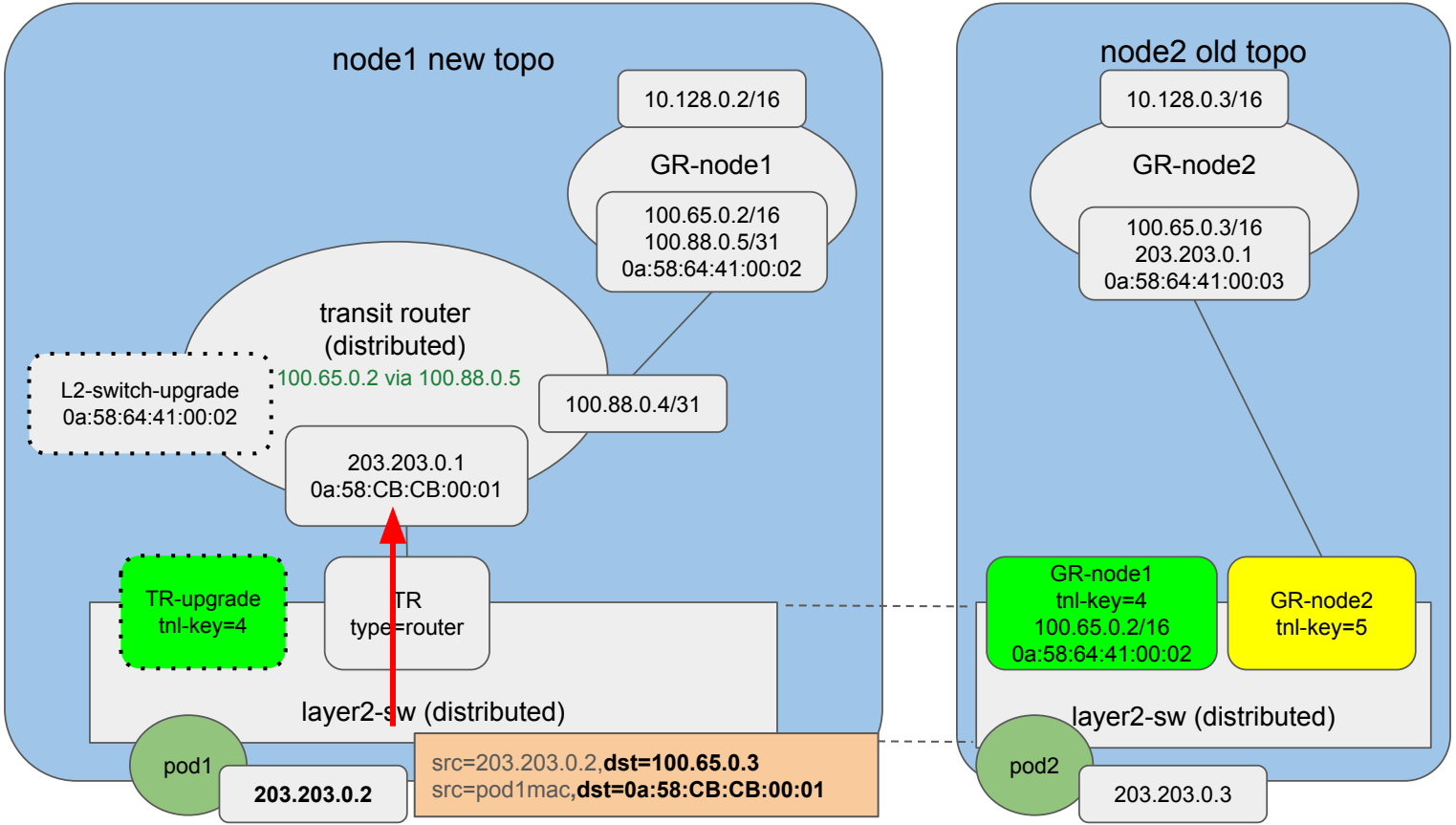
Traffic: external -> node2 -> pod1 (reply direction)

external (10.128.0.42) -> GR-node2 (10.128.0.3, load balancer -> pod2 (203.203.0.2)



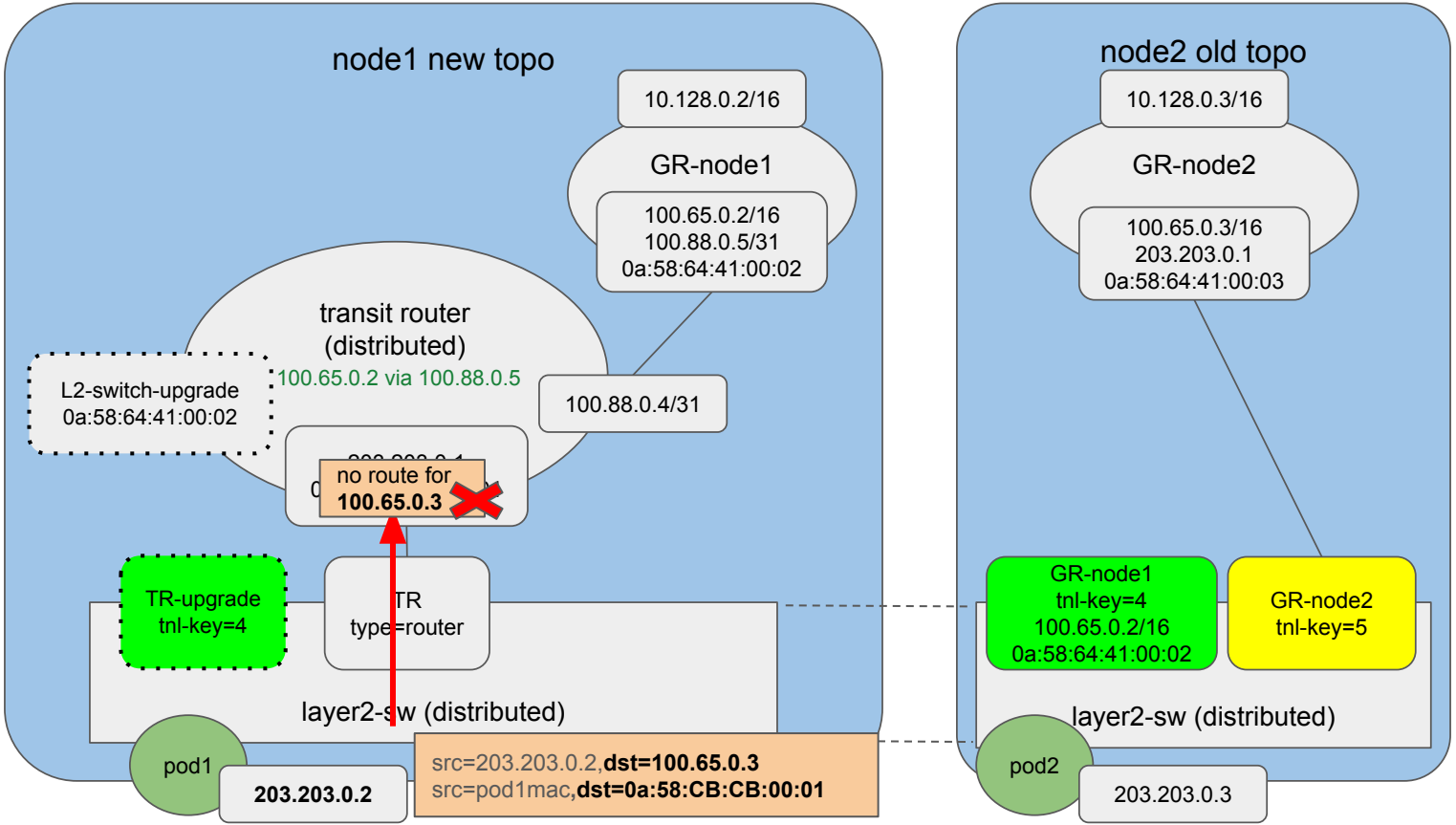
Traffic: external -> node2 -> pod1 (reply direction)

external (10.128.0.42) -> GR-node2 (10.128.0.3, load balancer -> pod2 (203.203.0.2)



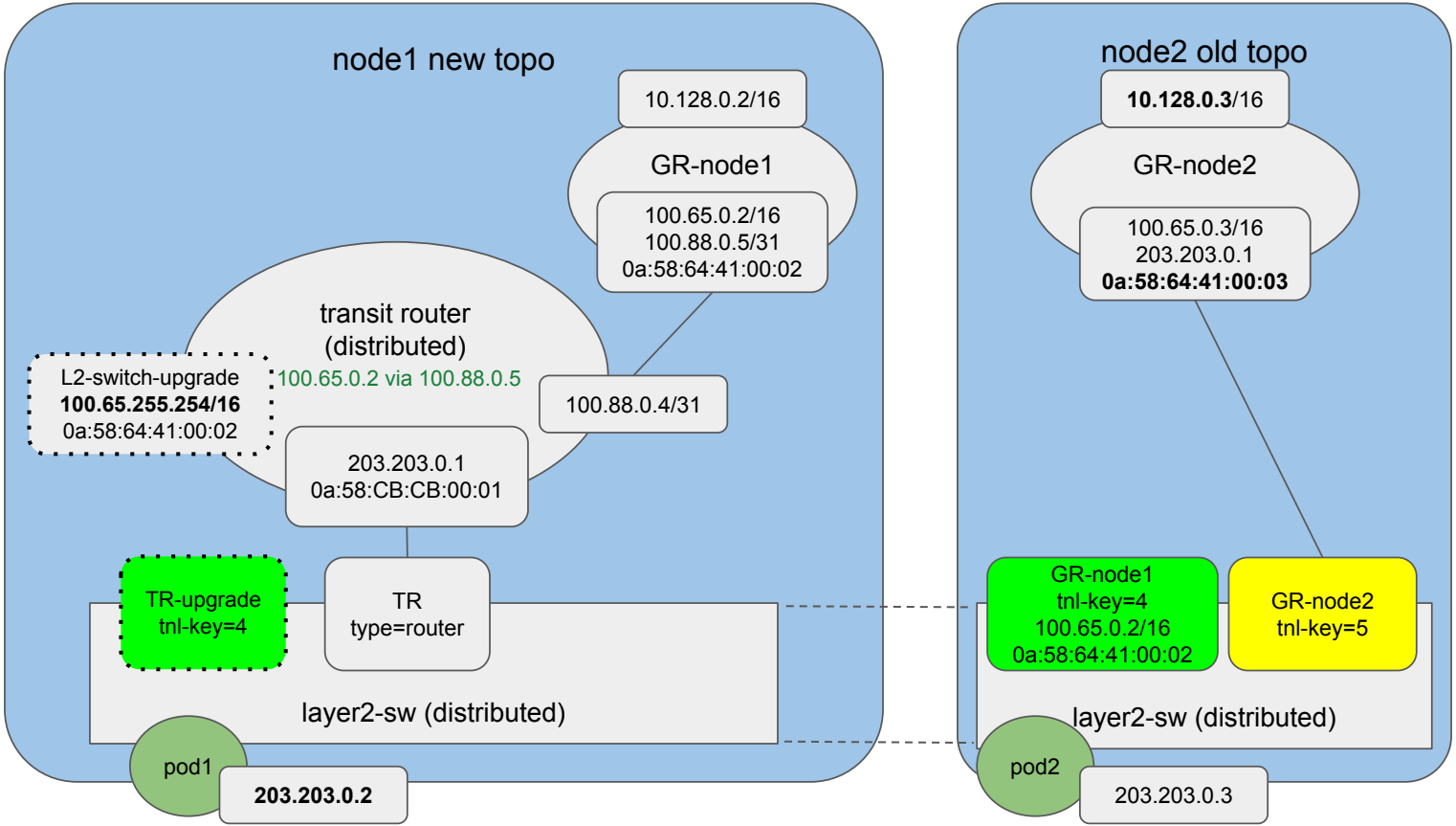
Traffic: external -> node2 -> pod1 (reply direction)

external (10.128.0.42) -> GR-node2 (10.128.0.3, load balancer -> pod2 (203.203.0.2)



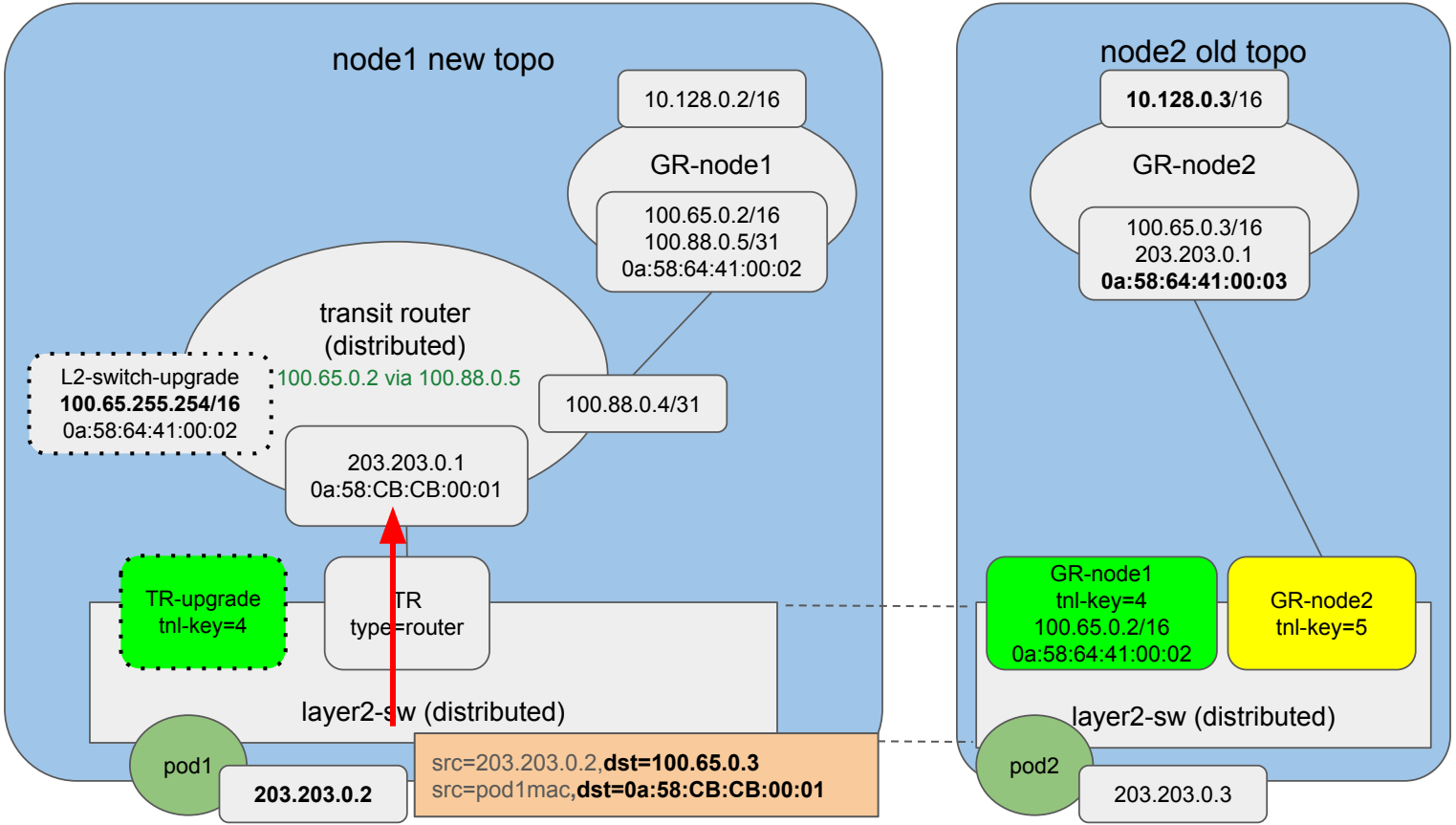
Traffic: external -> node2 -> pod1 (reply direction), take 2

external (10.128.0.42) -> GR-node2 (10.128.0.3, load balancer -> pod2 (203.203.0.2)



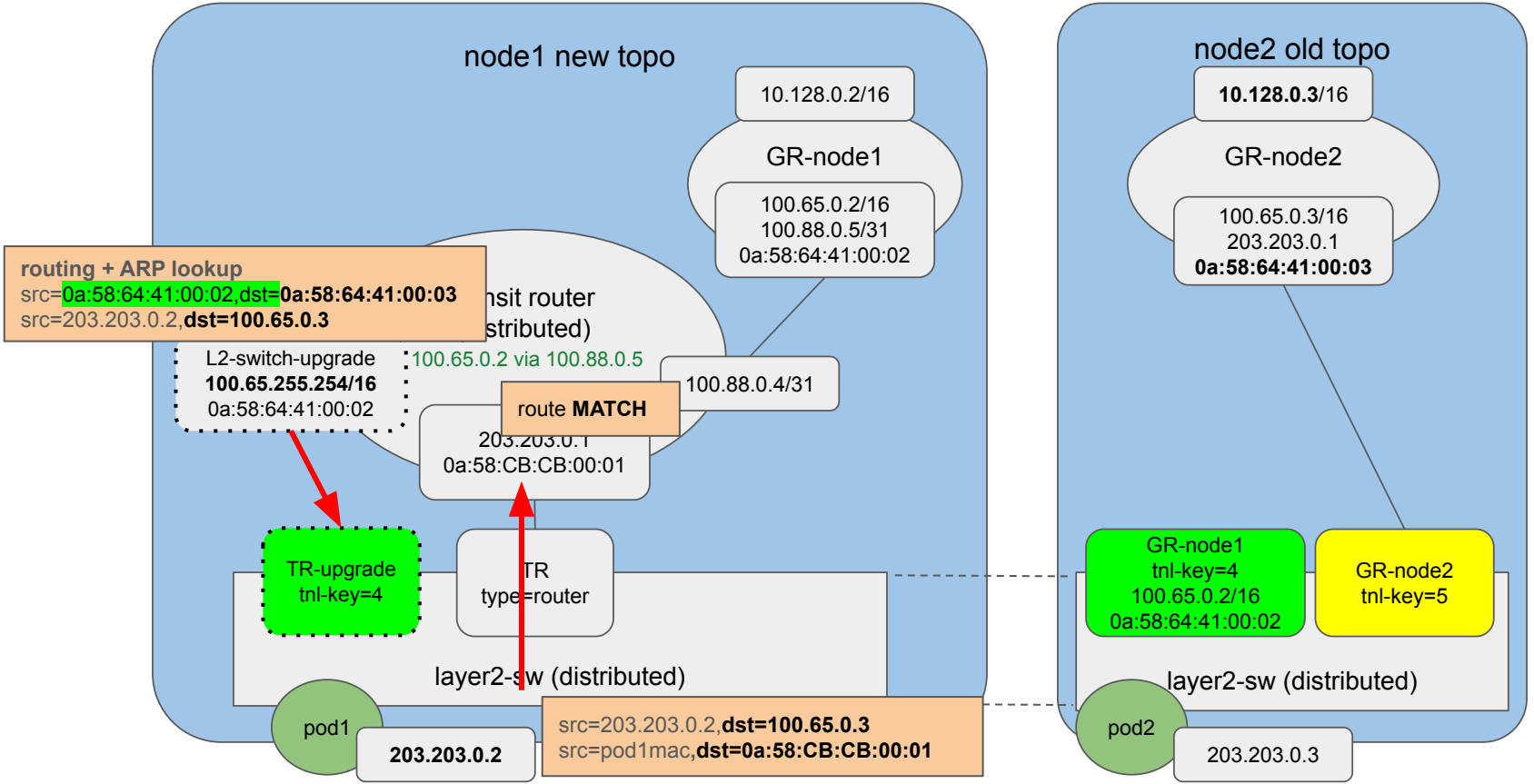
Traffic: external -> node2 -> pod1 (reply direction), take 2

external (10.128.0.42) -> GR-node2 (10.128.0.3, load balancer -> pod2 (203.203.0.2)



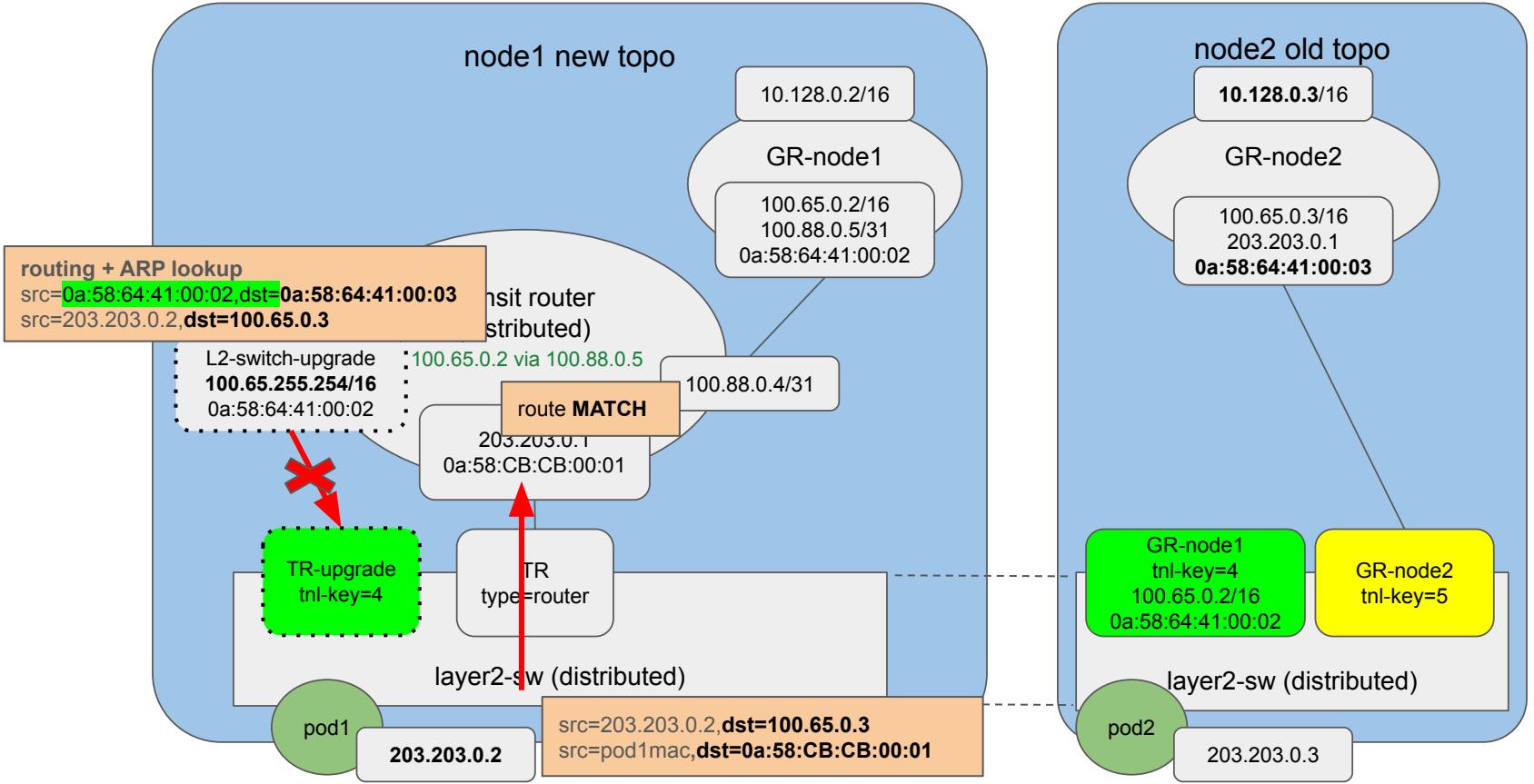
Traffic: external -> node2 -> pod1 (reply direction), take 2

external (10.128.0.42) -> GR-node2 (10.128.0.3, load balancer -> pod2 (203.203.0.2)



Traffic: external -> node2 -> pod1 (reply direction), take 2

external (10.128.0.42) -> GR-node2 (10.128.0.3, load balancer -> pod2 (203.203.0.2)



Nadia/Patryk/Quique/Tim: *We had a connected route for the destination IP? How come that didn't match the packet and forward it to node-2??*

Nadia/Patryk/Quique/Tim: *We had a connected route for the destination IP? How come that didn't match the packet and forward it to node-2??*

Dumitru: "*man ovn-nb.5*"; *your POD subnet is* ***203.203.0.0/24*** *so there's also this route:*

```
policy=src-ip 203.203.0.0/24 via 10.88.0.5 (to GR)
```

It's preferred (as documented) over the one you added:

```
(policy=dst-ip) 100.65.255.254/16 connected (to L2 TS)
```

Nadia/Patryk/Quique/Tim: *We had a connected route for the destination IP? How come that didn't match the packet and forward it to node-2??*

Dumitru: "*man ovn-nb.5*"; *your POD subnet is* ***203.203.0.0/24*** *so there's also this route:*

`policy=src-ip 203.203.0.0/24 via 10.88.0.5 (to GR)`

It's preferred (as documented) over the one you added:

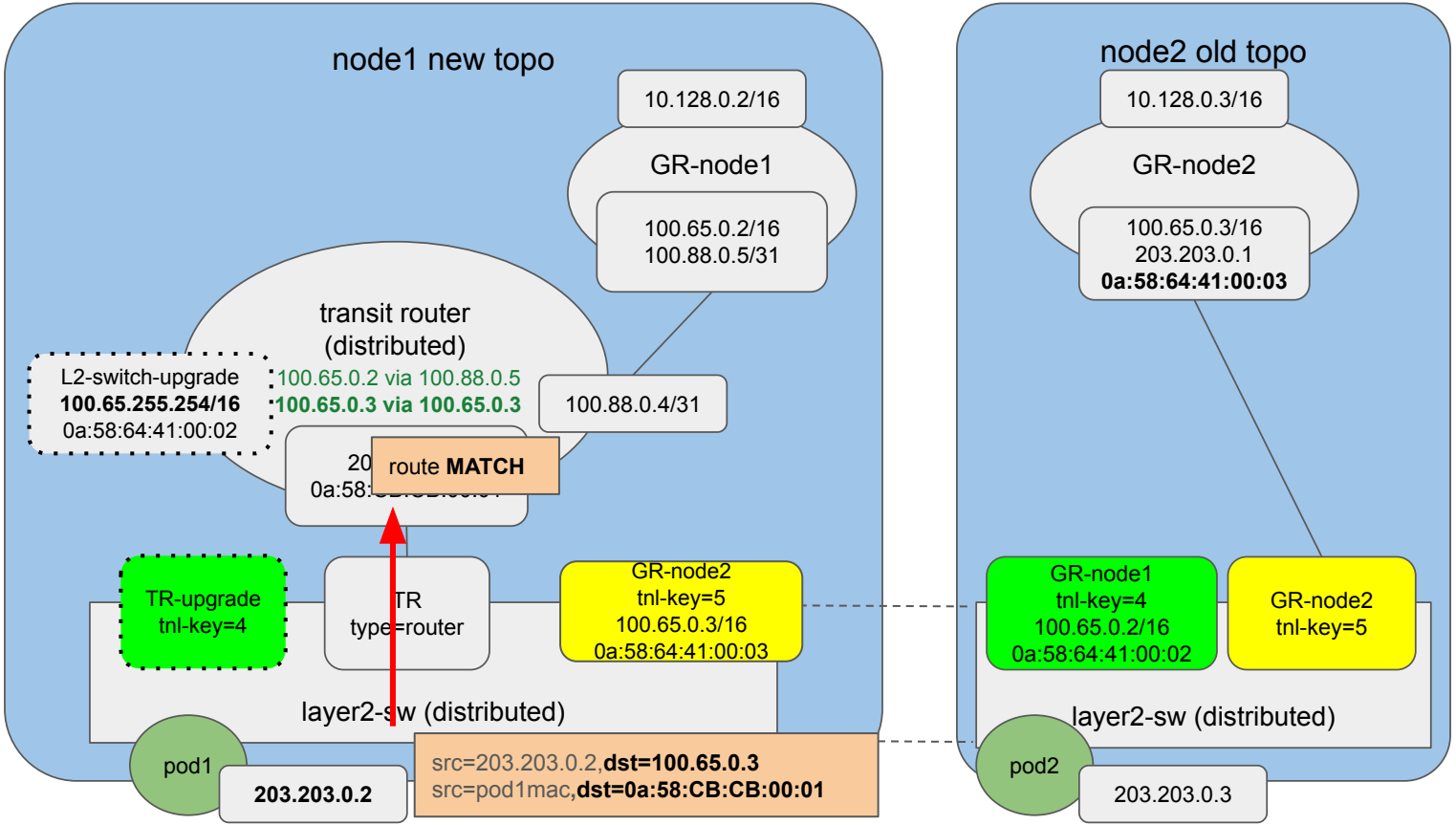
`(policy=dst-ip) 100.65.255.254/16 connected (to L2 TS)`

Nadia/Patryk/Quique/Tim: *OK... Let's add a "funny" route:*

`(policy=dst-ip) 100.65.0.3/32 via 100.65.0.3 (to L2 TS)`

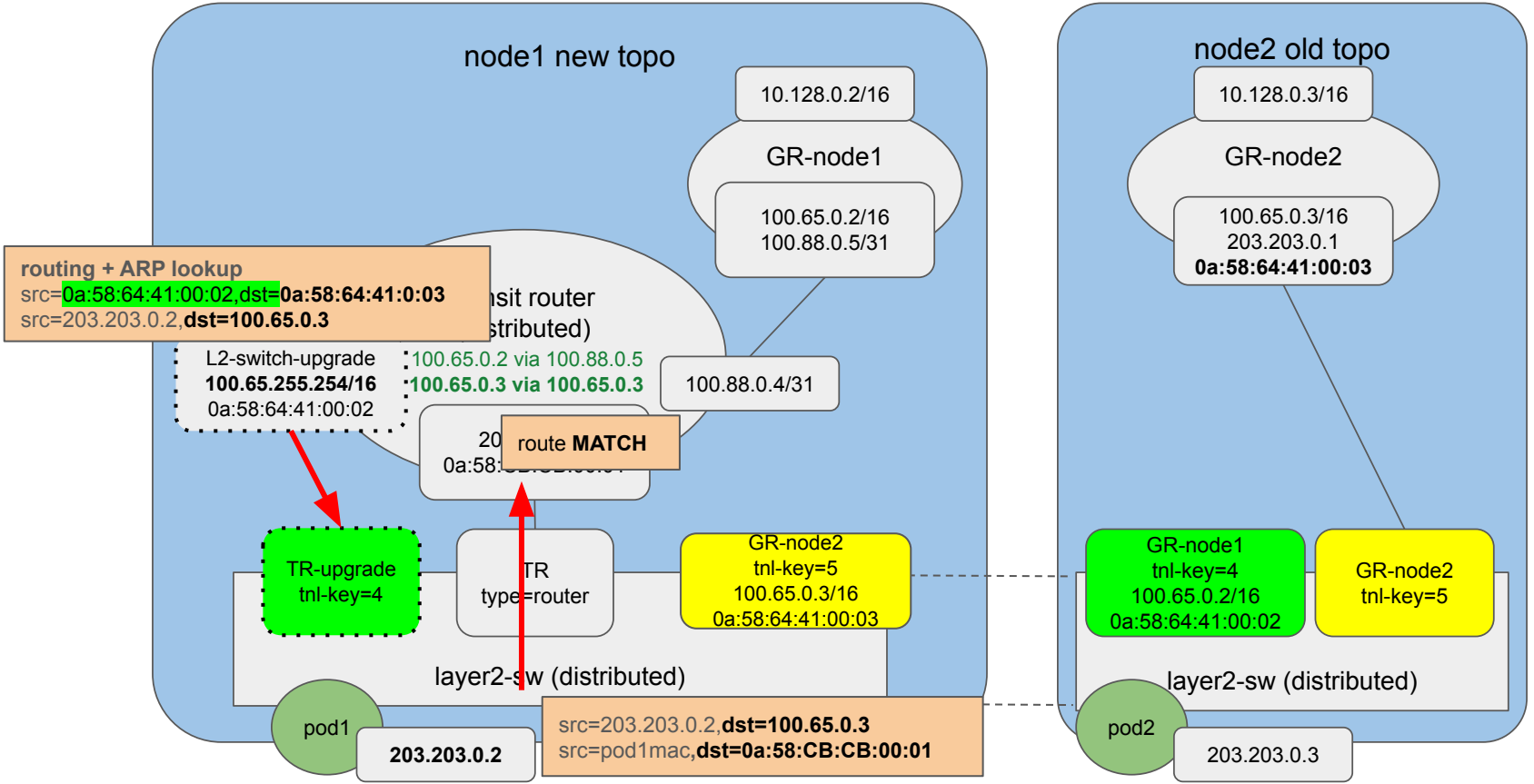
Traffic: external -> node2 -> pod1 (reply direction), take 3

external (10.128.0.42) -> GR-node2 (10.128.0.3, load balancer -> pod2 (203.203.0.2)



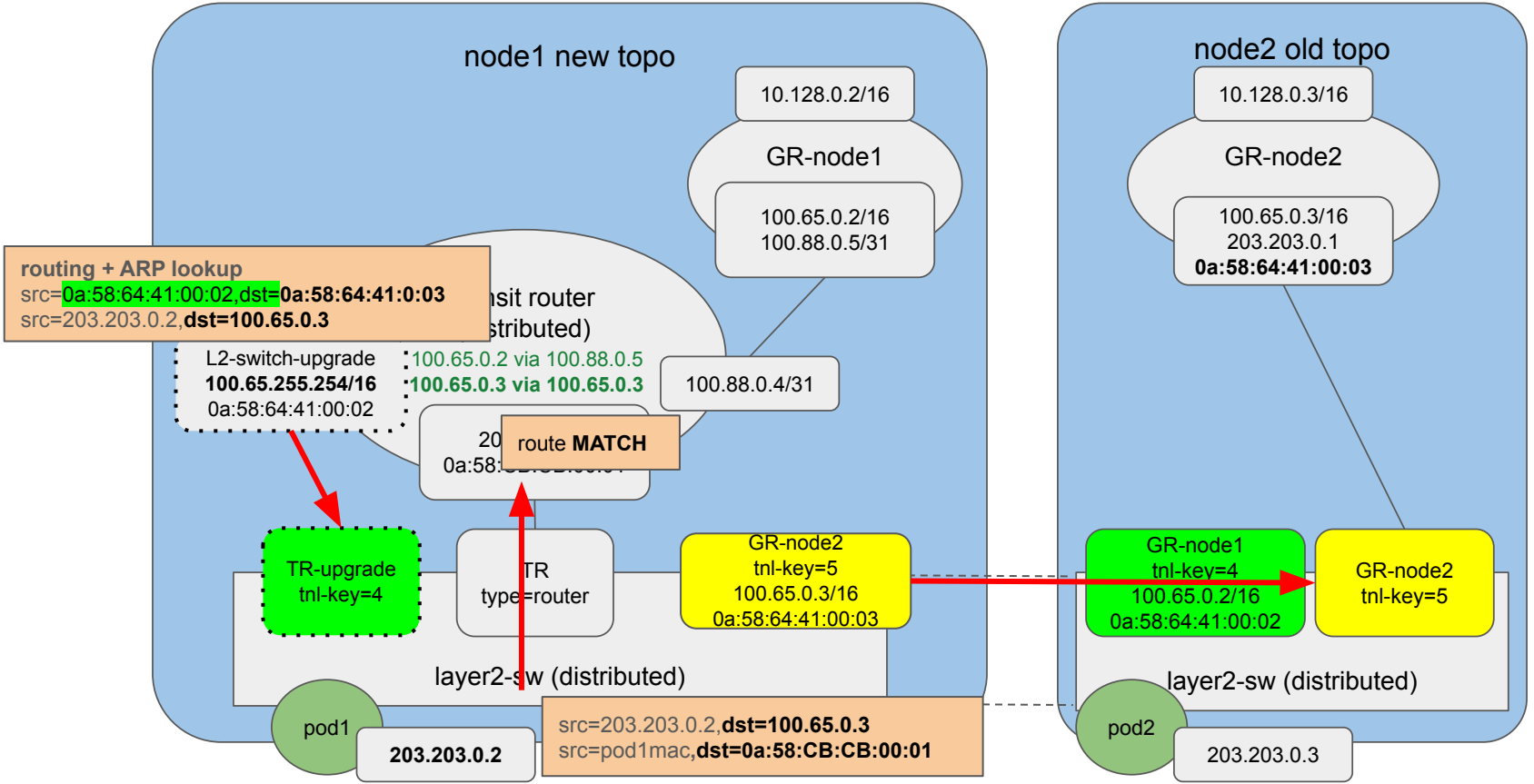
Traffic: external -> node2 -> pod1 (reply direction), take 3

external (10.128.0.42) -> GR-node2 (10.128.0.3, load balancer -> pod2 (203.203.0.2)



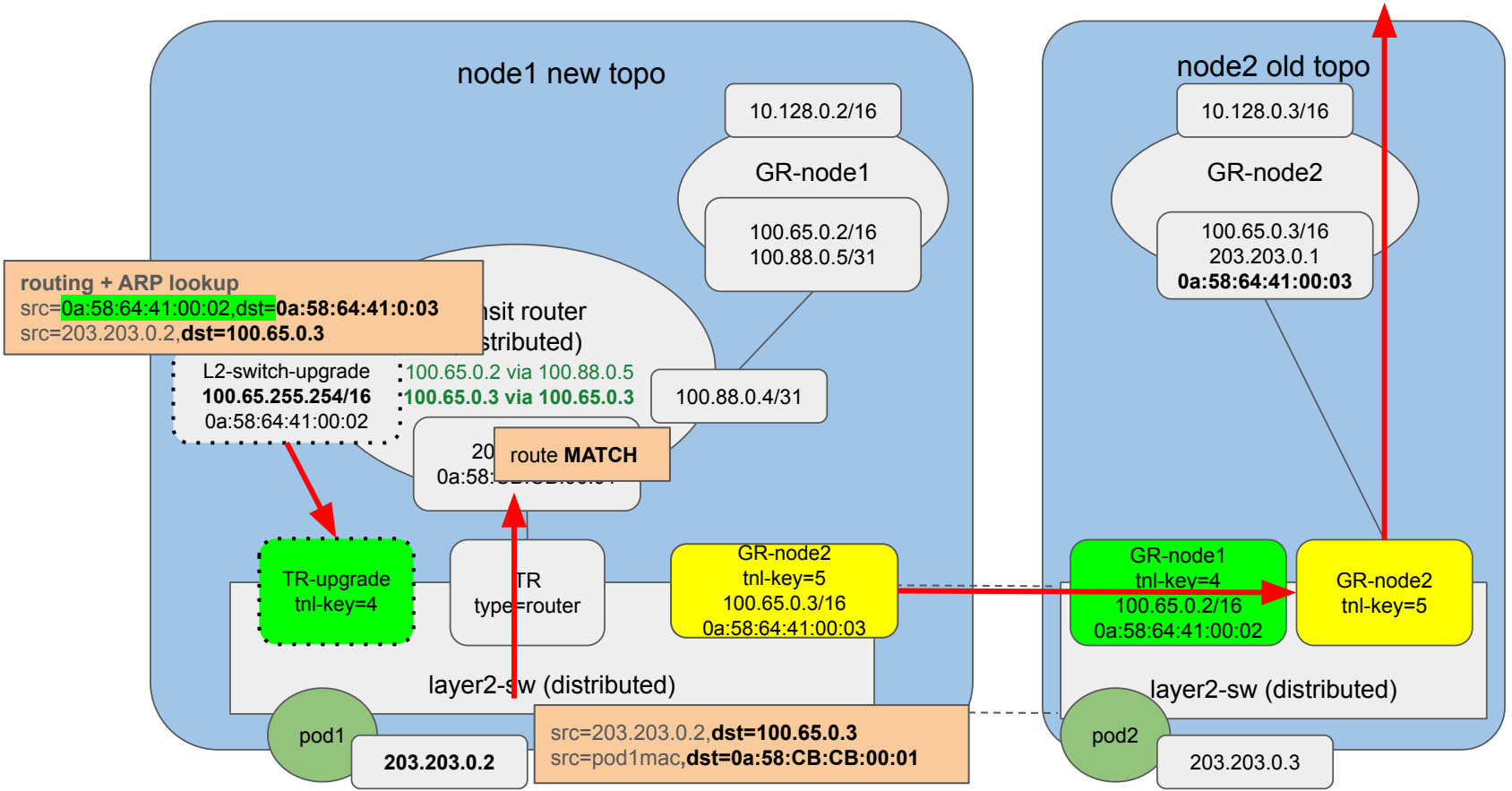
Traffic: external -> node2 -> pod1 (reply direction), take 3

external (10.128.0.42) -> GR-node2 (10.128.0.3, load balancer -> pod2 (203.203.0.2)



Traffic: external -> node2 -> pod1 (reply direction), take 3

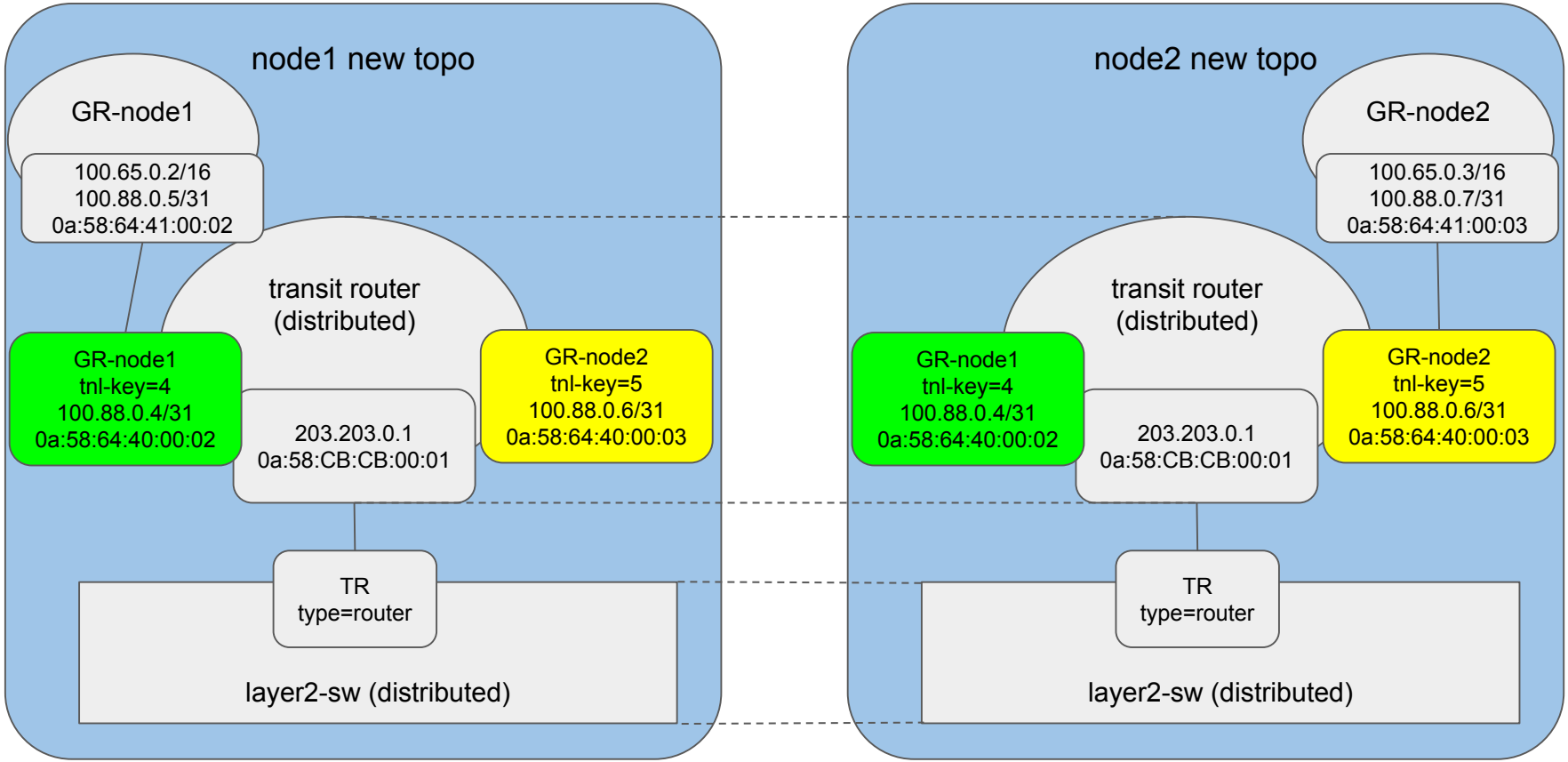
external (10.128.0.42) -> GR-node2 (10.128.0.3, load balancer -> pod2 (203.203.0.2)



Conclusions

- Non-disruptive topology upgrades are **HARD**, we had to:
  - maintain an additional “fake” link between a router and a switch
  - duplicate a MAC address between router ports
  - use dummy IPs and (seemingly) “funny” routes to fix routing
  - hand-craft Route Advertisements to update default routes on migrated VMs
- Designing a ***perfect*** network from day-1 is **HARD**
  - luckily networking gives us enough *footguns* to work around the problems
- OVN was flexible enough to implement all the workarounds needed for the topology upgrade
  - we were lucky this time

Now you have a new topology!



Special thanks to Quique (@qinqon) for

- starting off this ambitious project
- writing an enhancement proposal
- making a PoC
- helping with VM-related hacks

Thank you!