Resilience in SDN and NFV: How to divide state from stateless

DRCN 2016 Paris Tutorial

Hagen Woesner,
BISDN GmbH, Berlin, Germany
hagen@bisdn.de

March, 15th, 2016

Disclaimer: This is a tutorial

Do not expect original research here.

Just stuff copied from the web, plus some structuring thoughts.
(plus a few own drawings...)

Outline: Resilience in SDN and NFV

- Resilience in SDN
  - What is SDN, anyway?
  - Data plane resilience
    - Link failures
    - Path failures
    - Node failures
  - Control plane resilience
    - Control link failures
    - Controller node failures

Outline: Resilience in SDN and NFV (cont.)

- Resilience in NFV
  - What is NFV, then?
  - NFVI failures
  - Service Function failures
  - „Microservices“
- Conclusions
  - „Treat your network and servers as cattle, not pets!“
  - Automate as much as you can (using chef, puppet, salt, ansible)
  - Reboot, re-install often (check if you are resilient)
  - Future hardware comes in white boxes (OPC, TIP)

Terminology: SDN

SDN: Programming *Network Functions*

$$y = f(x)$$

$$output = action(match)$$

As f is most of the time non-steady, use discrete value tables

Match	Action	Output	Stats
IP dst==8.8.8.8	Set L2_dst=03:04:05:06:07:01; Set L2_src=01:02:03:04:05:06; decr. TTL	Port 3	#packets, #bytes
IP dst==4.4.4.4	Set L2_dst=04:05:06:07:08:09; Set L2_src=01:02:03:04:05:06; decr. TTL	Port 2	#packets, #bytes

SDN: Software Defined Networks

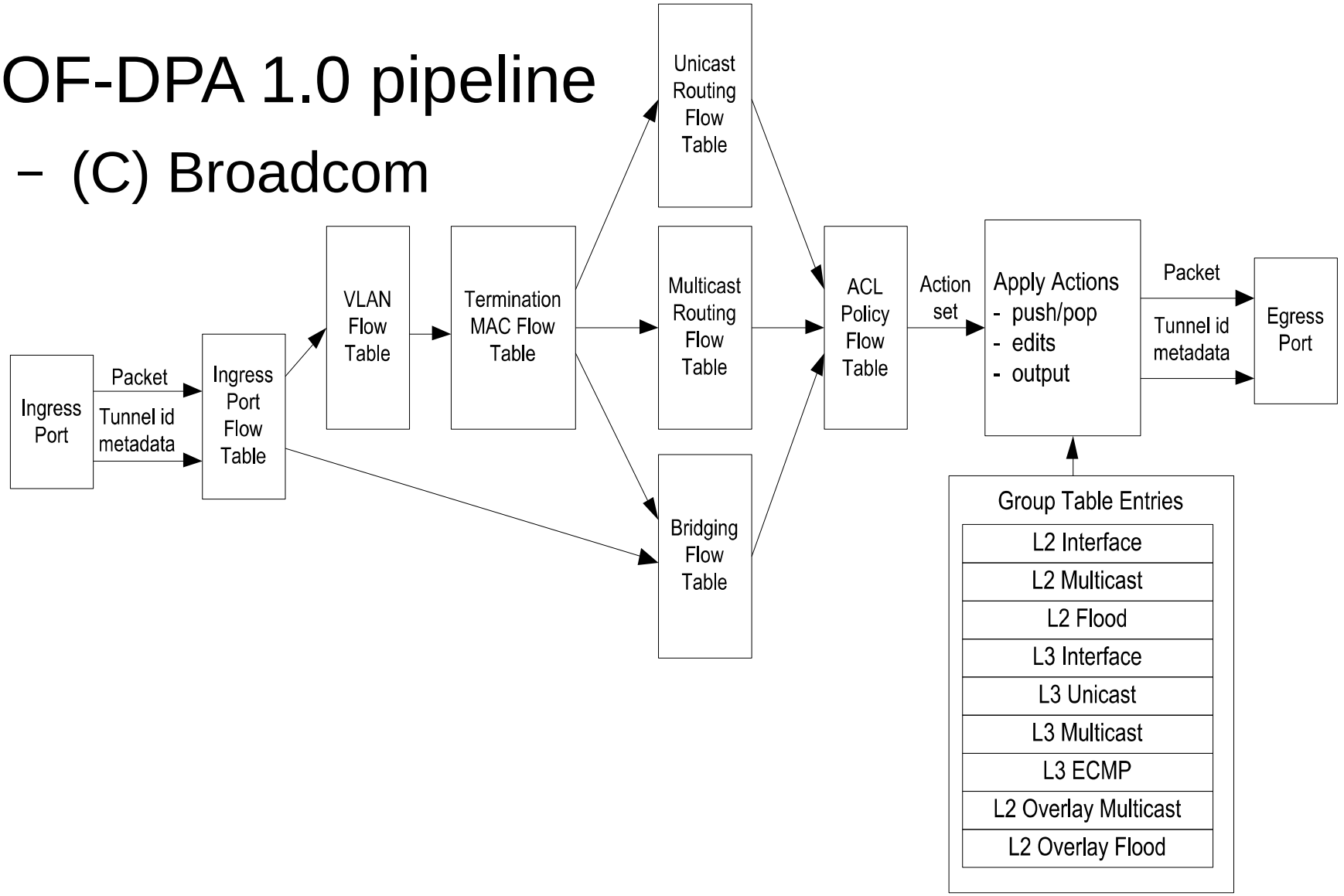
- Make f programmable to implement
  - Encryption (e.g., IPSec)
  - Compression / network coding
  - Transcoding
  - Stateful firewalls

- For most applications, simply have to transfer a packet from A to B
  - f is simple and ***stateless***

  $\Rightarrow$ OpenFlow can be used to write the forwarding table

SDN is not that simple, though...

- OF-DPA 1.0 pipeline
  - (C) Broadcom

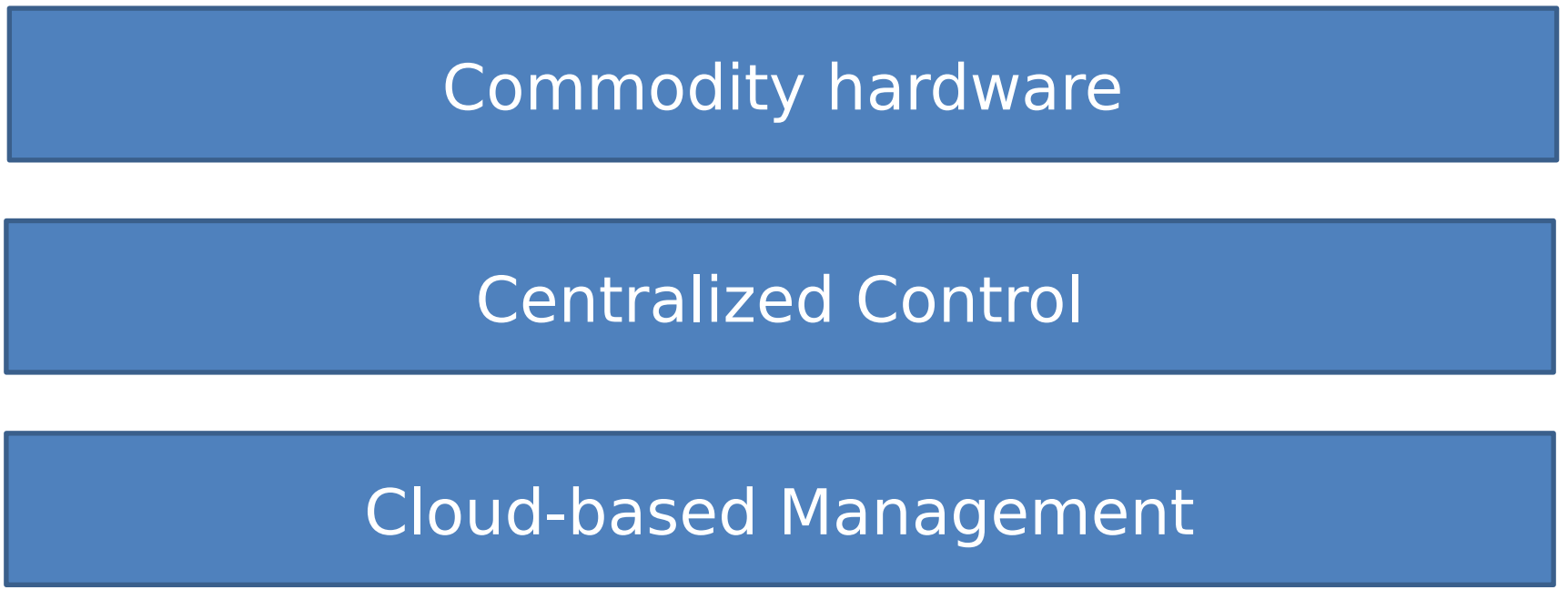


Programming the network functions

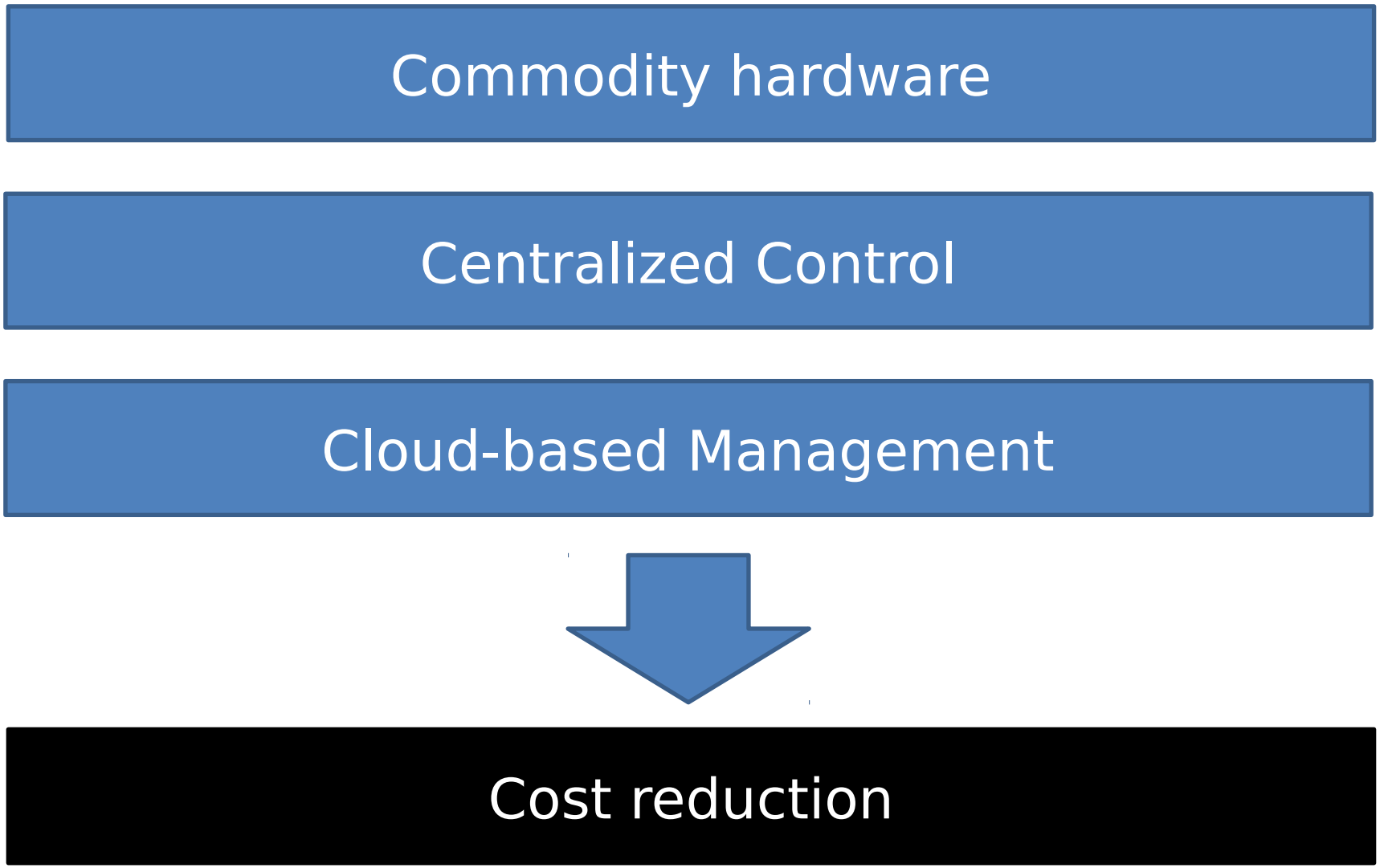
- More programmability with real programming languages like P4
  - P. Bosshart, D. Daly, G. Gibb, M. Izzard, N. McKeown, J. Rexford, C. Schlesinger, D. Talayco, A. Vahdat, G. Varghese, D. Walker: "*P4: Programming Protocol-Independent Packet Processors*", ACM Sigcomm 2014
- But if f is really complicated?
  - Beyond action lists or methods?
- Download and execute code on x86 platforms

Network function virtualization

- NFV: Separate functionality from ASIC
  - Split out code blocks that implement f
  - Place f into programmable hardware
  - X86 or ARM platforms



So, how does SDN/NFV add value?



Definitions (cont.)

- Resilience:

  "the ability [of a system] to cope with change"

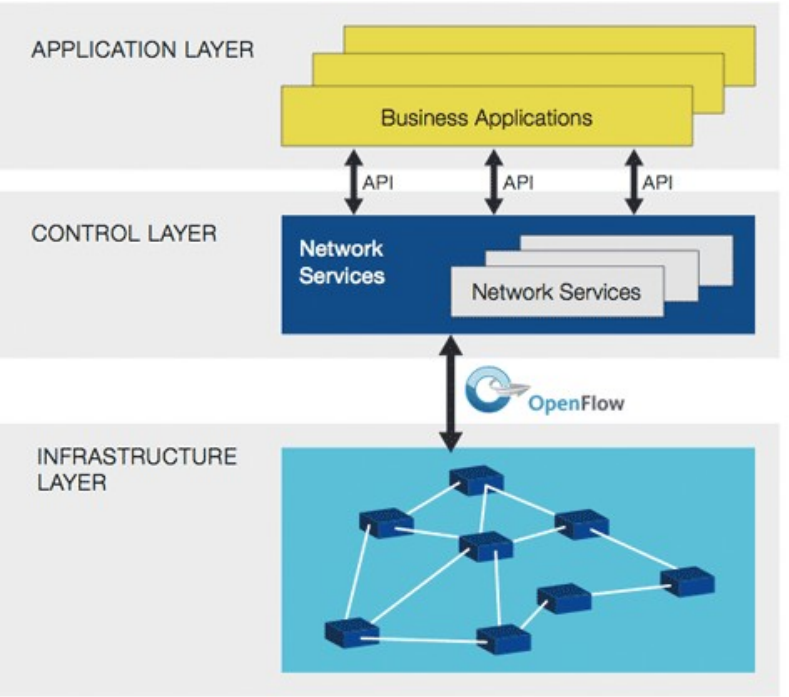
  https://en.wikipedia.org/wiki/Resilience

  - Here: the ability to restore normal service behavior after certain failures in the system

- Thesis: Restoration of stateless services is easy
  - Try and make all services state-less
    - By moving state ***out***

SDN: Data plane resilience: Link Failures

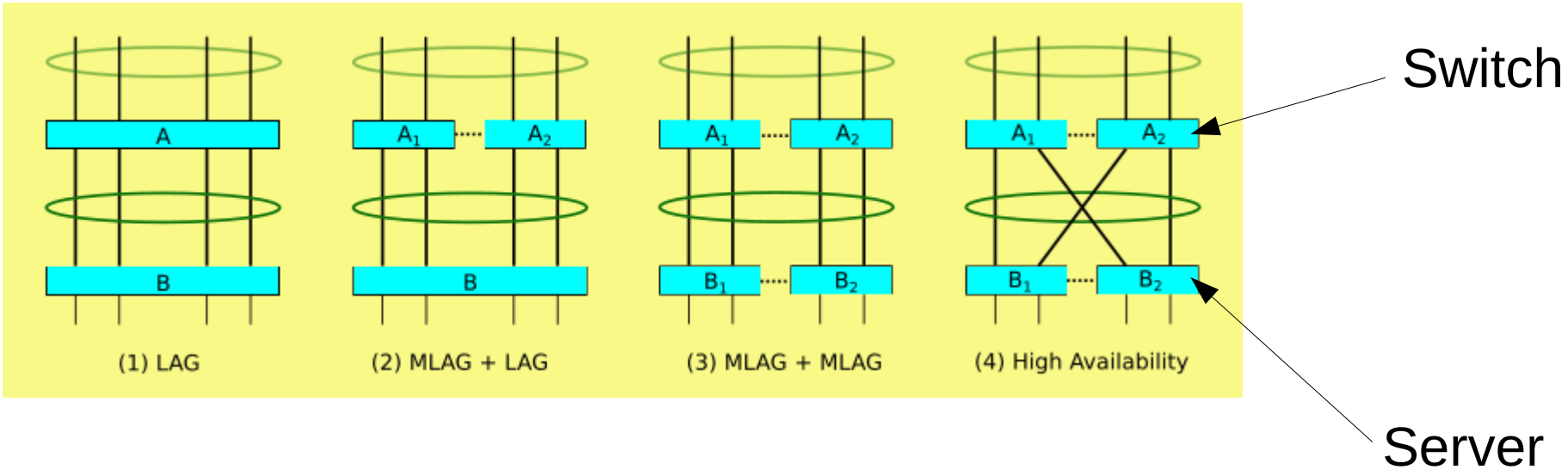
- Deal with the problems bottom-up
- First: Mitigate link or port failures
- Two ways:
  - Re-route
  - Provide back-up links



https://www.opennetworking.org/sdn-resources/sdn-definition

Multi-Chassis Link Aggregation

- Compute servers are typically connected to 2 ToR switches

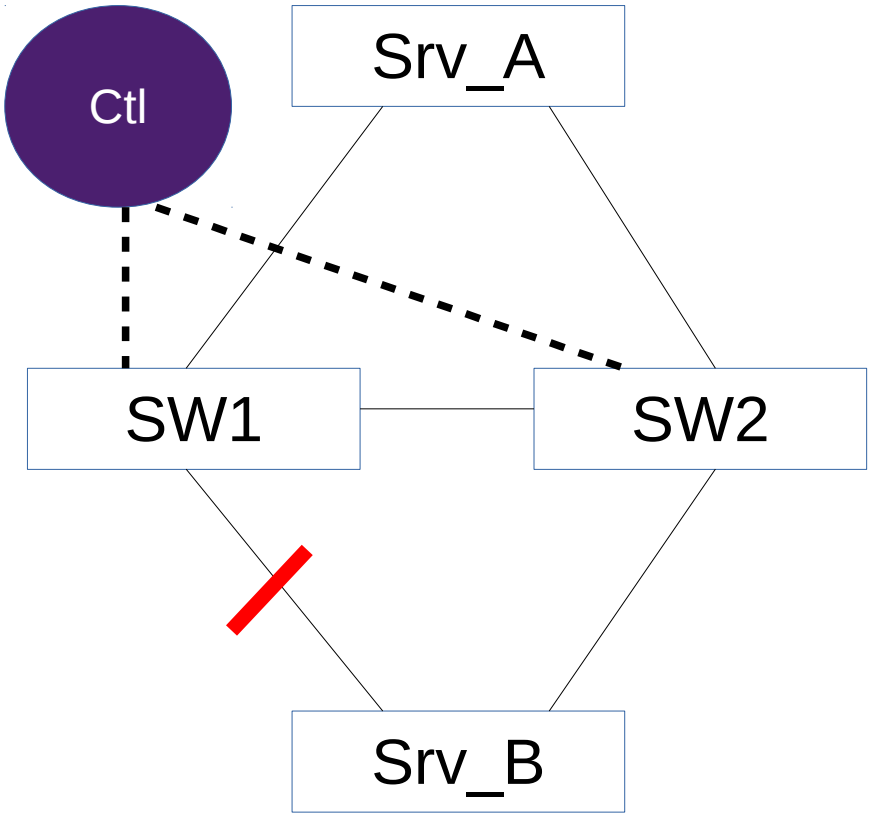


- What if SDN ends at the switch ports?

https://upload.wikimedia.org/wikipedia/commons/2/2f/Using_Multi-Chassis_LAG_%28MC-LAG%29_for_High_Availability.png

Example: Link failure

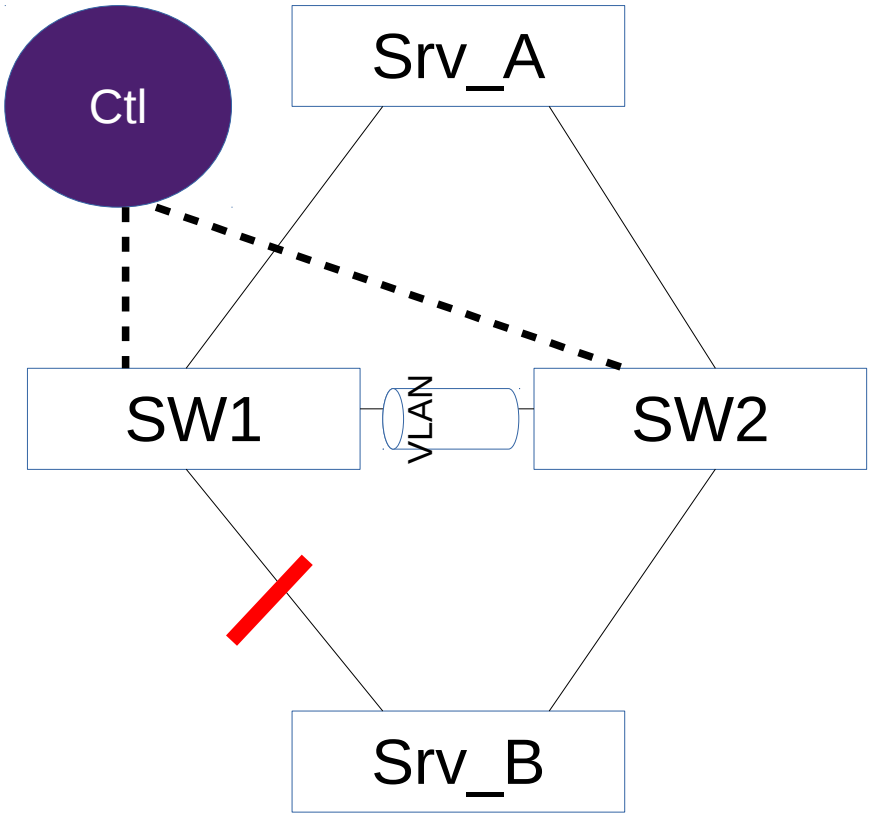
- Ctl implements L2 learning switch
- Srv_A uses link bonding
- Link between SW1 and Srv_B fails
- Controller re-routes traffic
- … SW2 stops forwarding
  - Why?



Example: Link failure

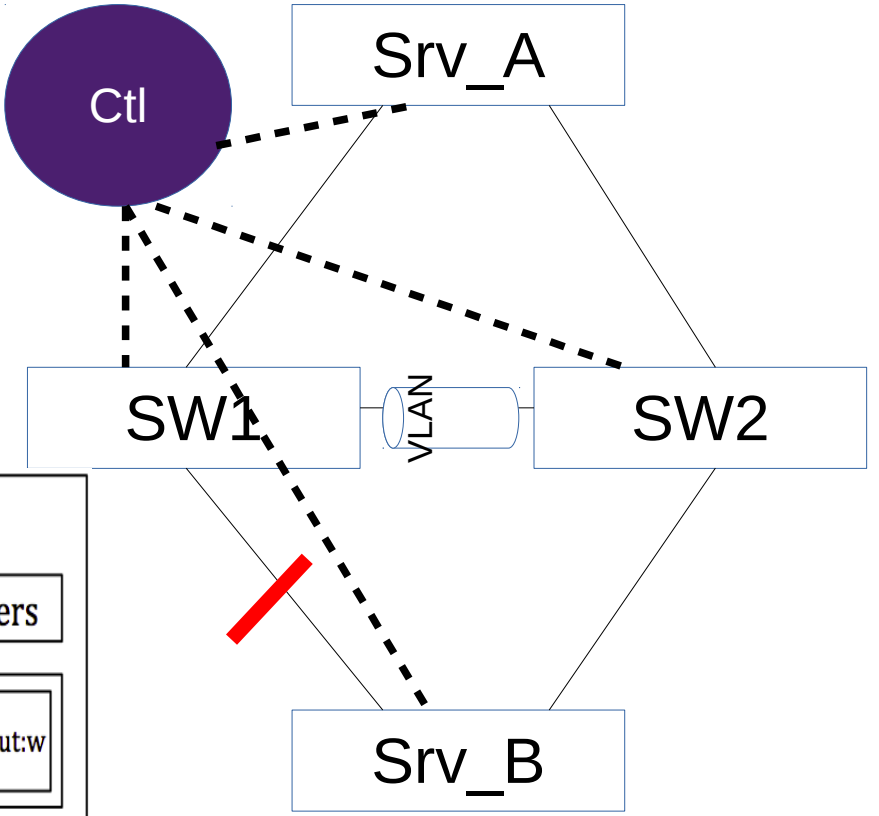
- In-built src learning table has same MAC from 2 ports
  - And 'thinks' it is a loop
- One (not very clever) solution:

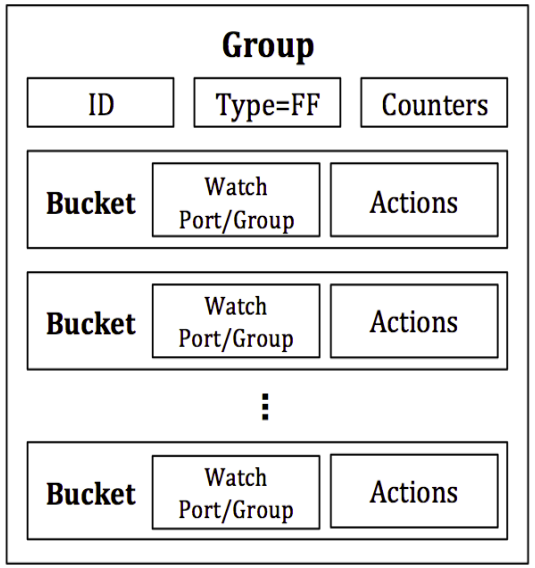
  Create fail-over VLAN/VXLAN and fail-over rules that push and pop

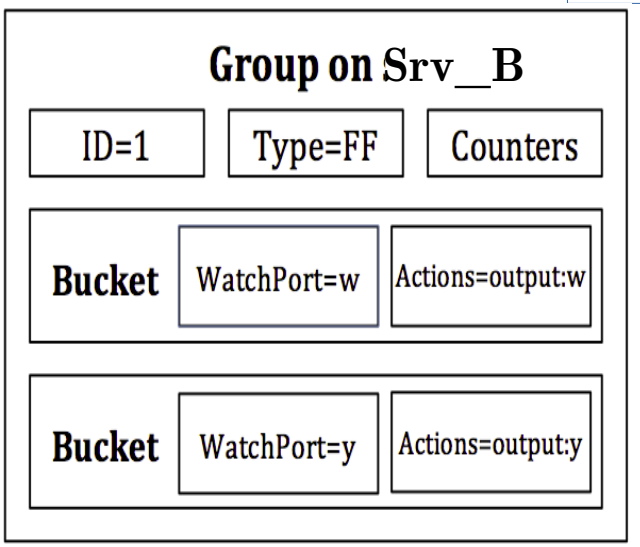


Example: Link failure

- When Ctl spans the servers as well, fail-over groups can be used
- Create fail-over group in Srv-B







https://floodlight.atlassian.net/wiki/display/floodlightcontroller/How+to+Work+with+Fast-Failover+OpenFlow+Groups#HowtoWorkwithFast-FailoverOpenFlowGroups-OpenFlowGroups

SDN: Data plane resilience: Path failures

- Again, depending on the size of the domain, re-routing may be easy
- When path crosses domain borders, BFD or Eth OAM is used
- Implementation in controller is possible, but has low performance

  https://events.nordu.net/display/ndn2012web/Ethernet+OAM+integration+in+OpenFlow

- Better use in-built hardware features of commercial ASICs

Use of hardware features is difficult

- Complete knowledge of underlying pipeline is required!
- c.f. TTP discussion
  https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/OpenFlow%20Table%20Type%20Patterns%20v1.0.pdf

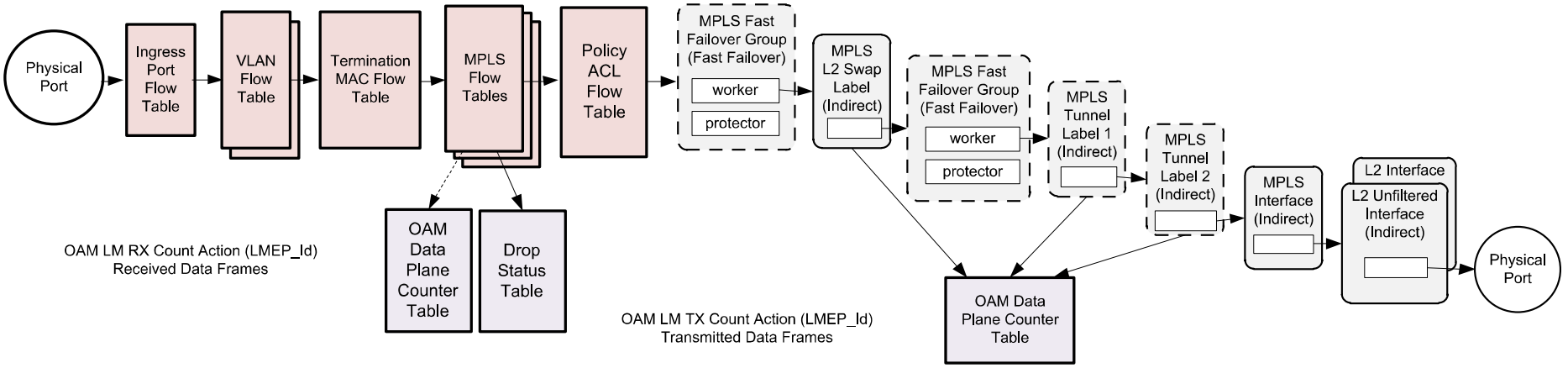


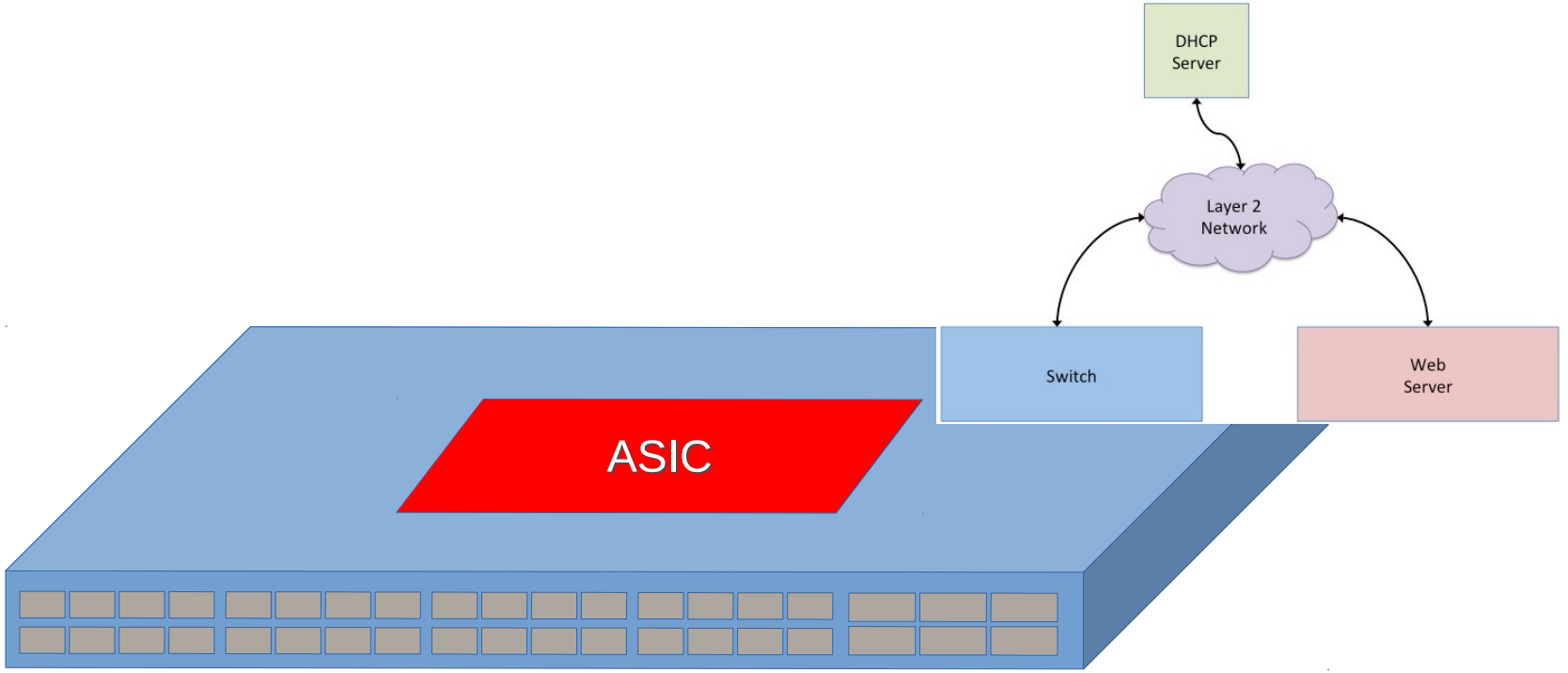
Figure 35. MPLS-TP - G.8113.1 OAM LSR Data Frame

SDN: Data plane resilience: Node failures

- What to do when the switch is down?
  - Replace it!
  - In the meantime, re-route around the failed node (delete all entries in the neighbor nodes)
- After replacement, how can the switch's state be re-established?
  - Address configuration
  - Operating system
  - Feature set
  - Controller address
  - Tunnels, fail-over groups, flowmods?

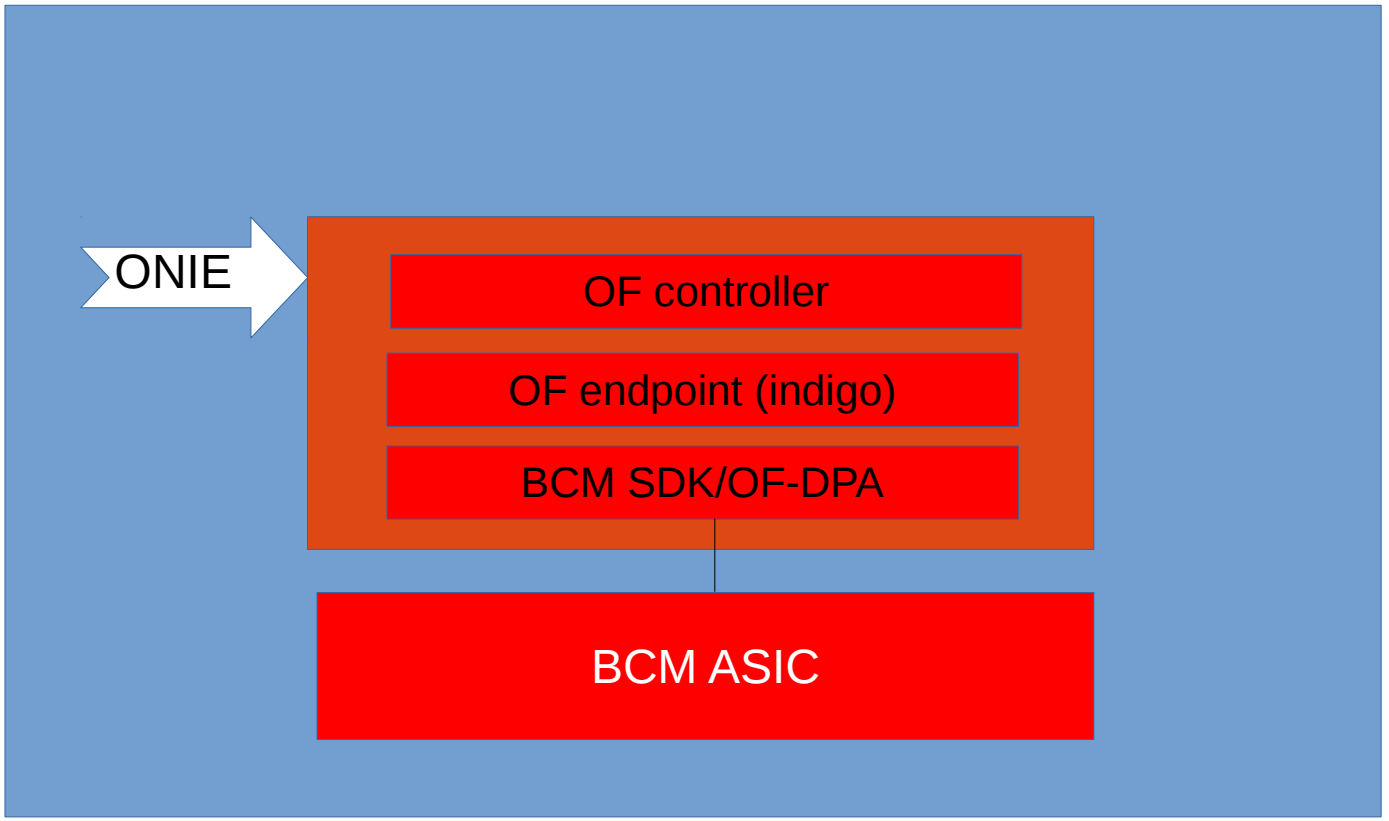
ONIE - Open Network Install Environment

- Combines a boot loader with a modern Linux kernel and BusyBox
- Provides an environment for installing any network OS
- Switch uses mgt network interface to dhcp
- Receives address of image server to fetch OS from

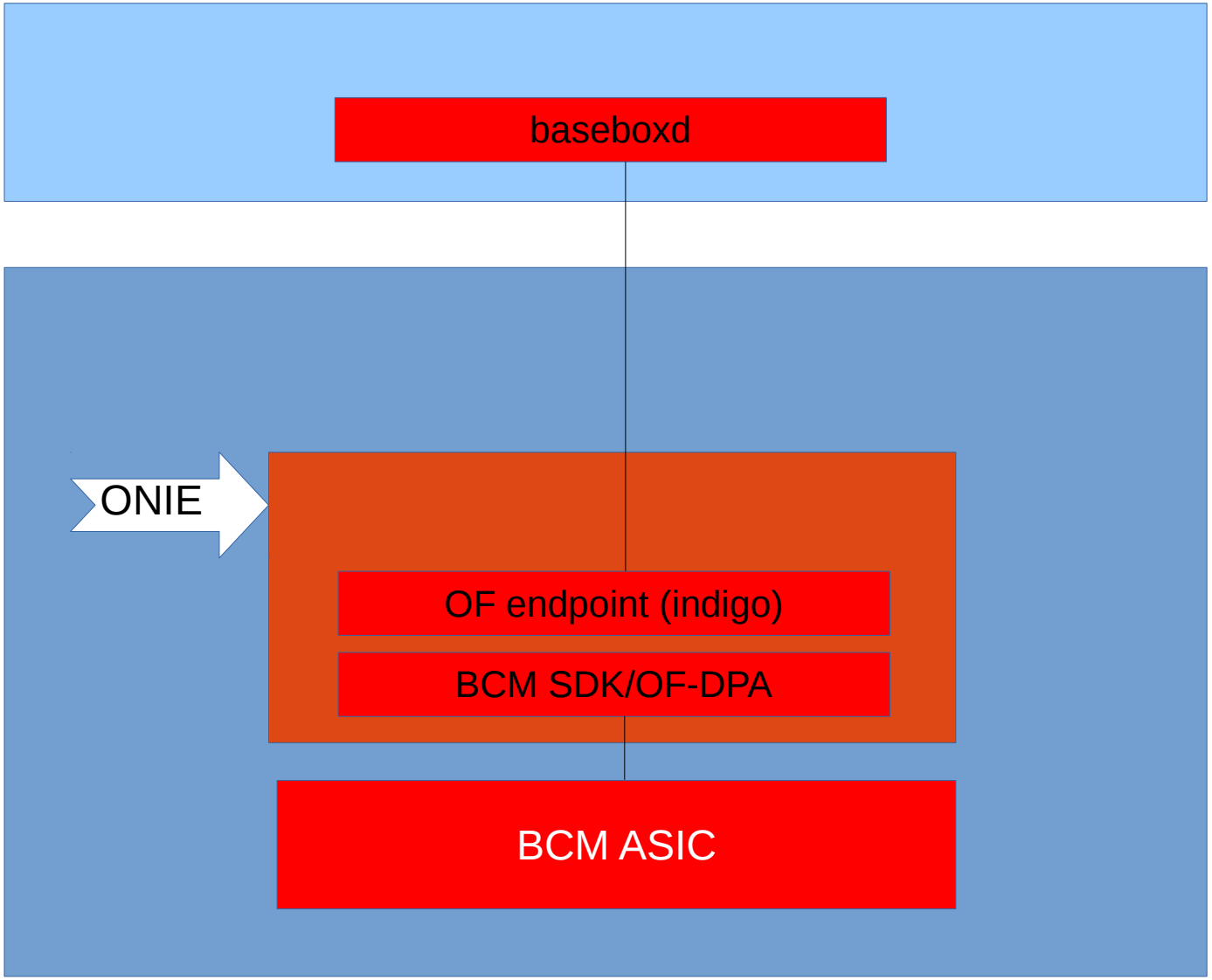


https://github.com/opencomputeproject/onie/wiki/Quick-Start-Guide

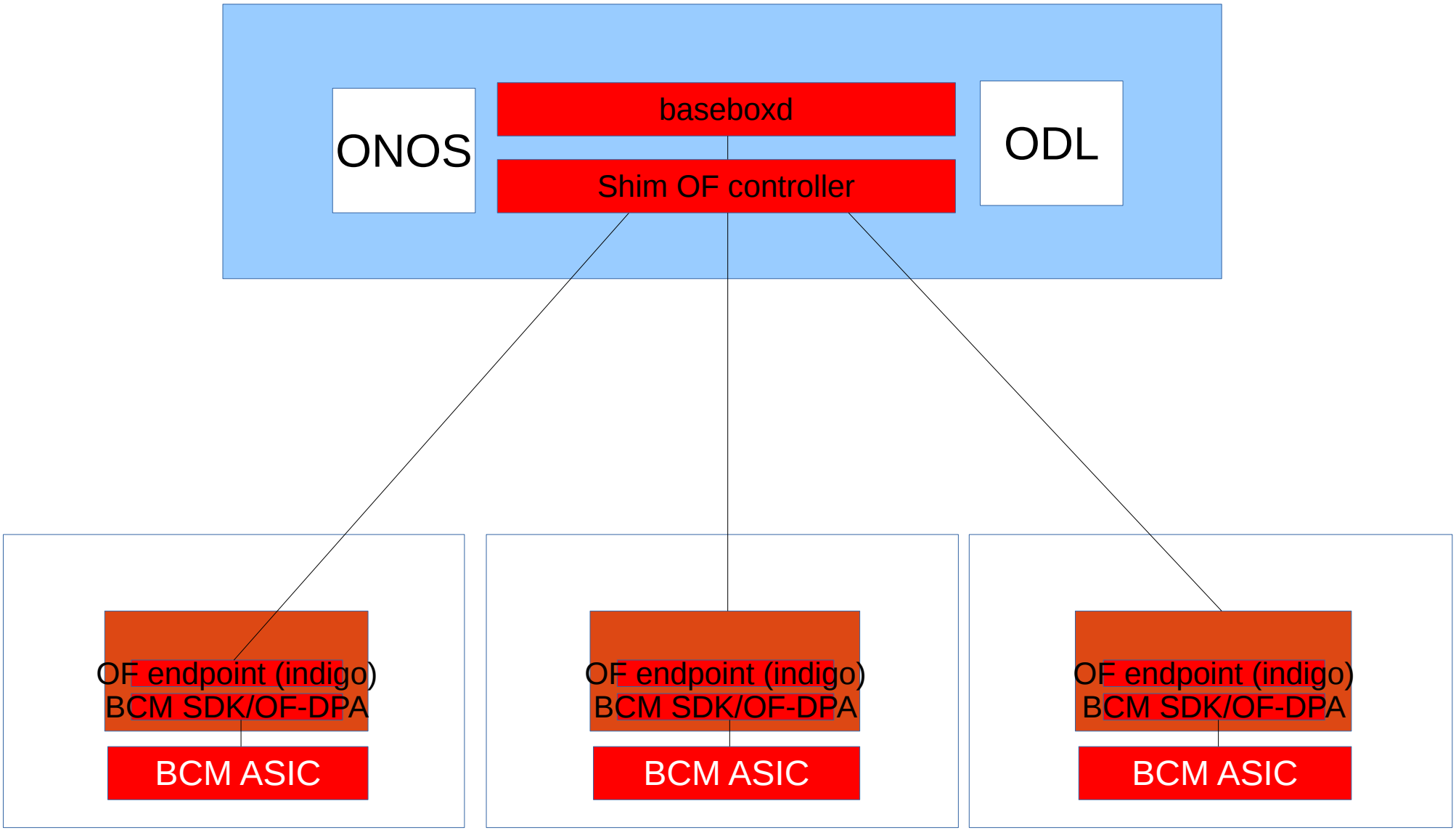
Install the OS image via ONIE



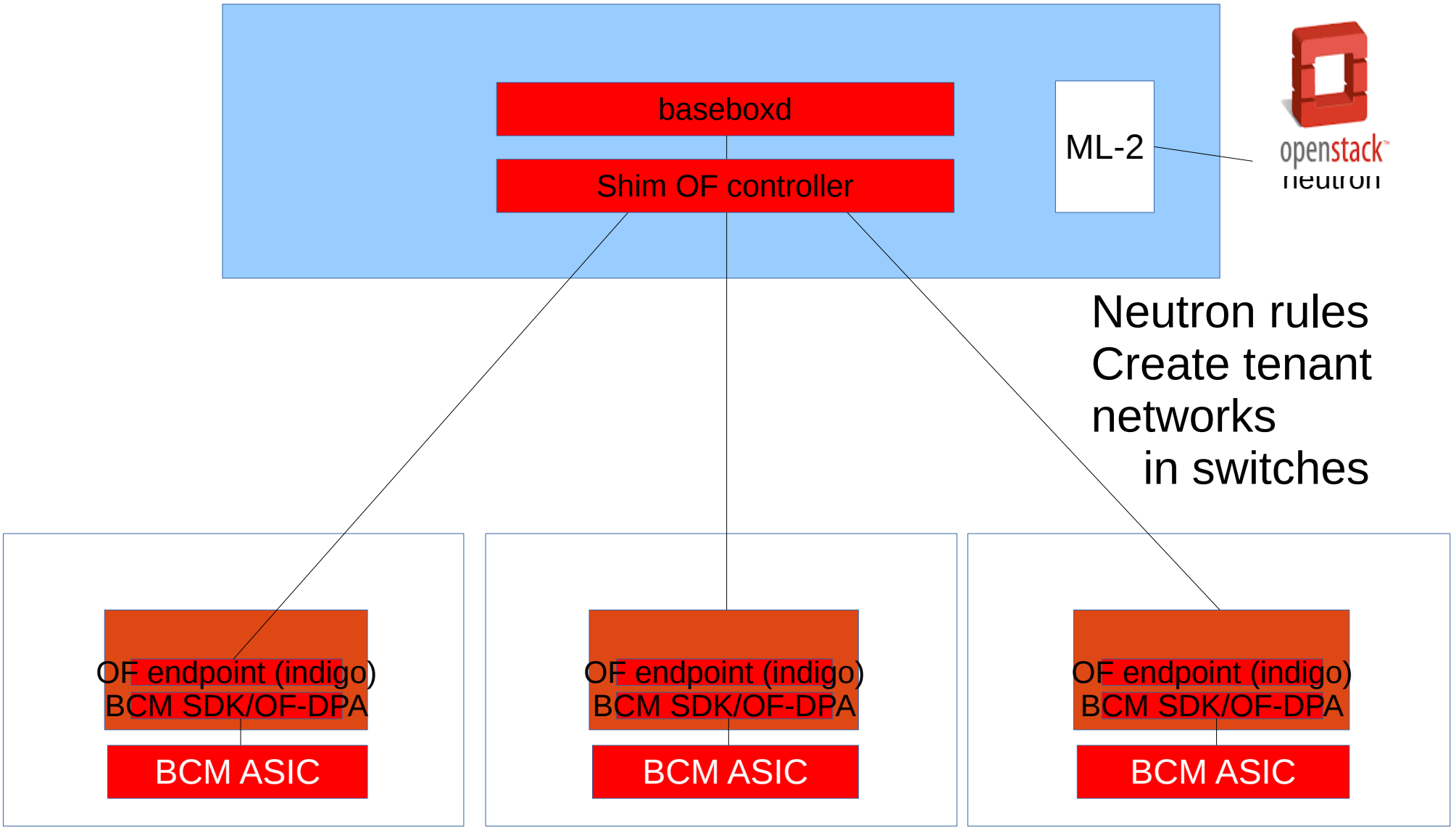
Retrieve controller address via dhcp/salt/puppet/...



Extend control over a domain of switches



And finally, integration into OpenStack

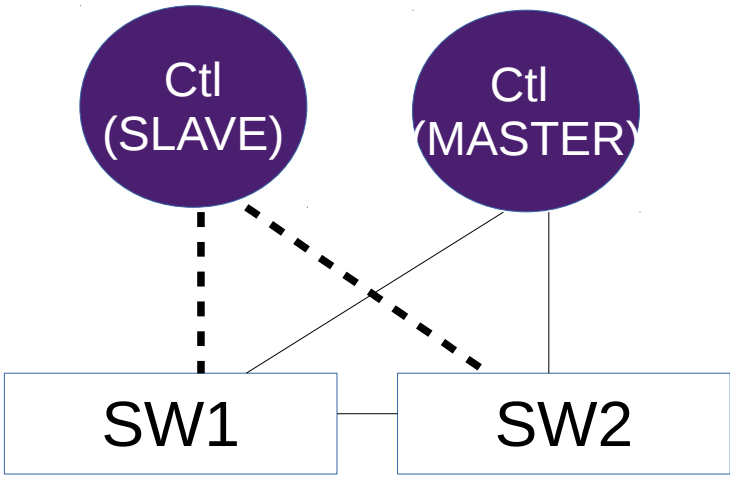


SDN: Data plane resilience: Node failures

- What to do when the switch is back up again?
  - (In the meantime, controller re-routed around the failed switch)
- Controller needs to re-populate the switch's fwd table
- This may involve re-calculating previously embedded paths
  - And re-populate the neighbor switches, as well
- It may be best to keep all state as high as possible
  - Re-calculate all routes from top to bottom.

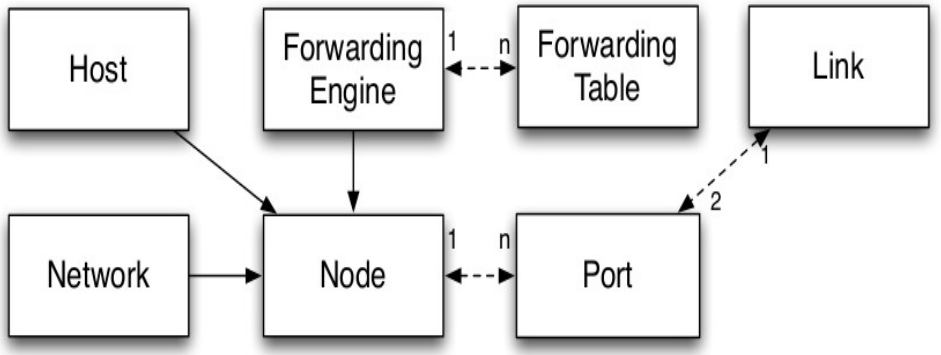
SDN: Control plane resilience: controller link

- Assume a setting of switches and controllers
- Switches are connected to multiple controllers (virtualization left aside here)
- The default role of a controller is OFPCR_ROLE_EQUAL
  - controller has full access to the switch
  - receives all asynchronous messages (such as packet-in, flow-removed)
  - Can send commands to the switch
- In role OFPCR_ROLE_SLAVE
  - the controllerhas read-only access to the switch. By default, the controller does not receive switch asynchronous messages, apart from Port-status messages.
- Role OFPCR_ROLE_MASTER similar to EQUAL



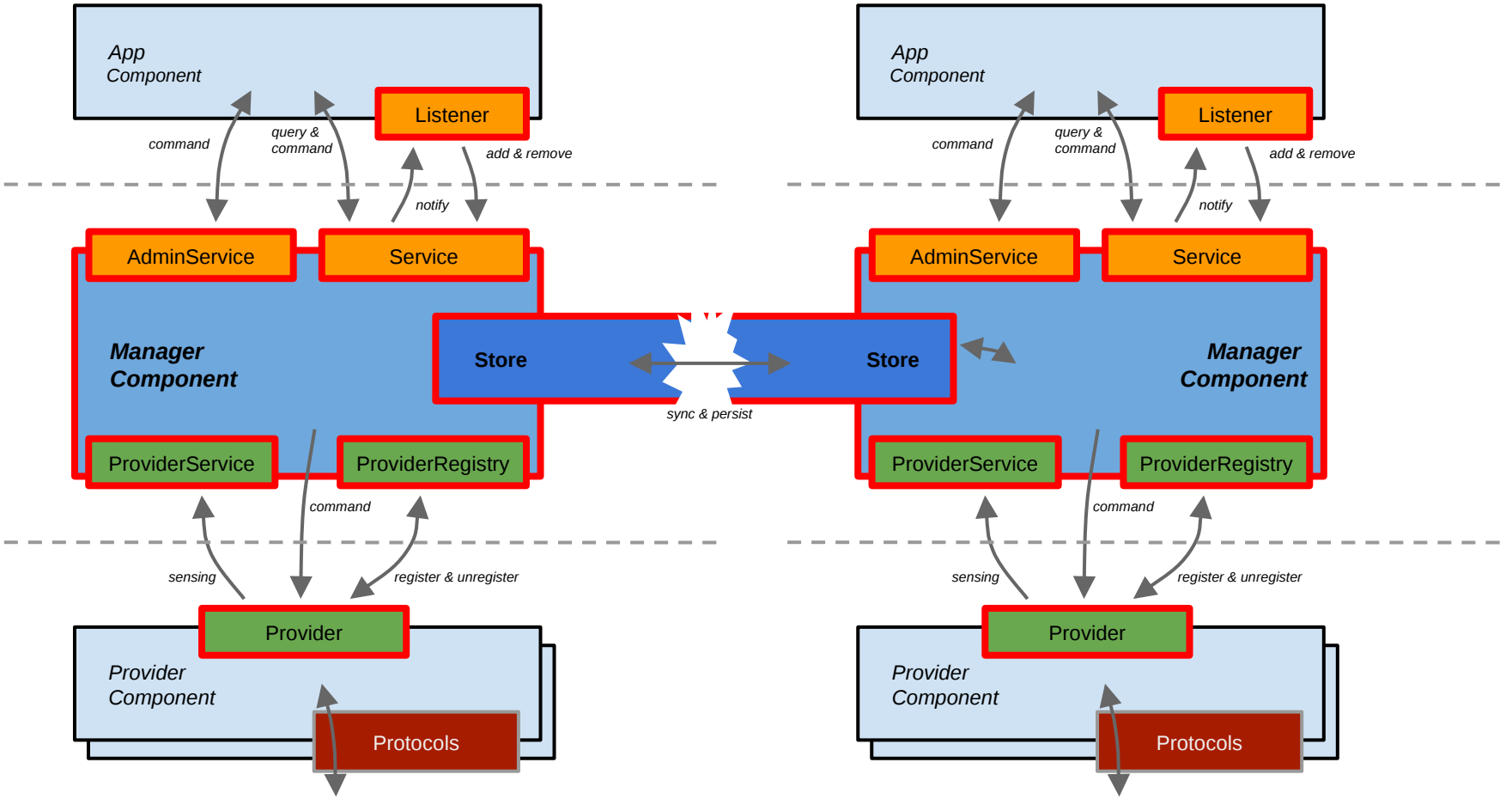
SDN: Control plane resilience: controller node

- Originally, OpenFlow controllers like NOX were single points of failure
  - With ONIX (Martin Casado, later Nicira, later VmWare) controllers were split into a HA database and stateless frontends.
  - The concept of NIB (network information base) was born:



http://yuba.stanford.edu/~casado/onix-osdi.pdf

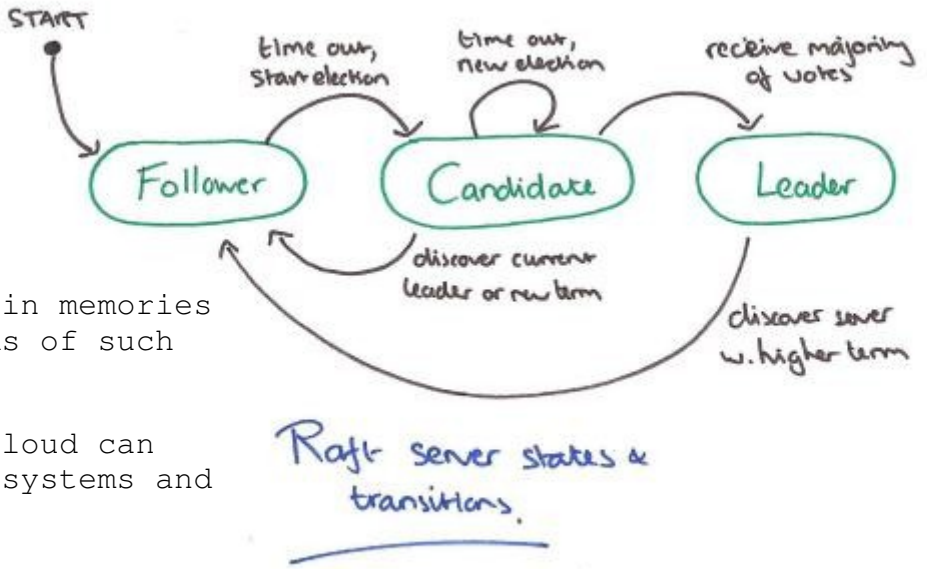
ONOS, starting form Cassandra (now using RAMCloud)



https://docs.google.com/presentation/d/1pPOXPy4_KagPcrwgQOX332uZn7KXKWfMEe7vNfhFDqo/edit?usp=sharing

RAMCloud

- Highly performant distributed key/value store based on raft consensus algorithm
- Hierarchical consensus

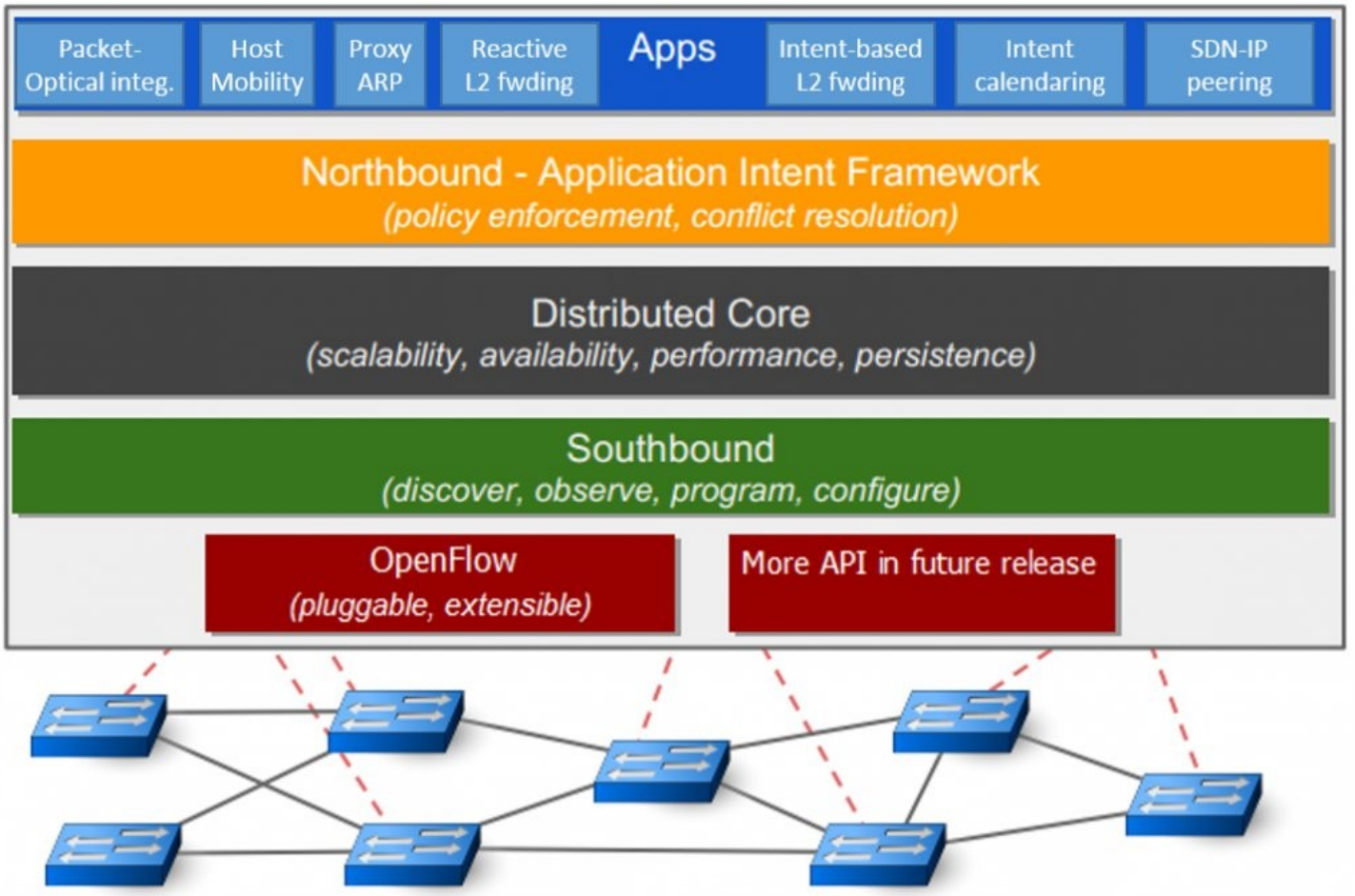


RAMCloud stores all of its information in the main memories of commodity servers, using hundreds or thousands of such servers to create a large-scale storage system.

Because all data is in DRAM at all times, a RAMCloud can provide 100-1000x lower latency than disk-based systems and 100-1000x greater throughput.

Although the individual memories are volatile, a RAMCloud can use replication and backup techniques to provide data durability and availability equivalent to disk-based systems.

Open Network Operating System (ONOS)



http://sdnhub.org/tutorials/onos/

Update on controllers and data bases

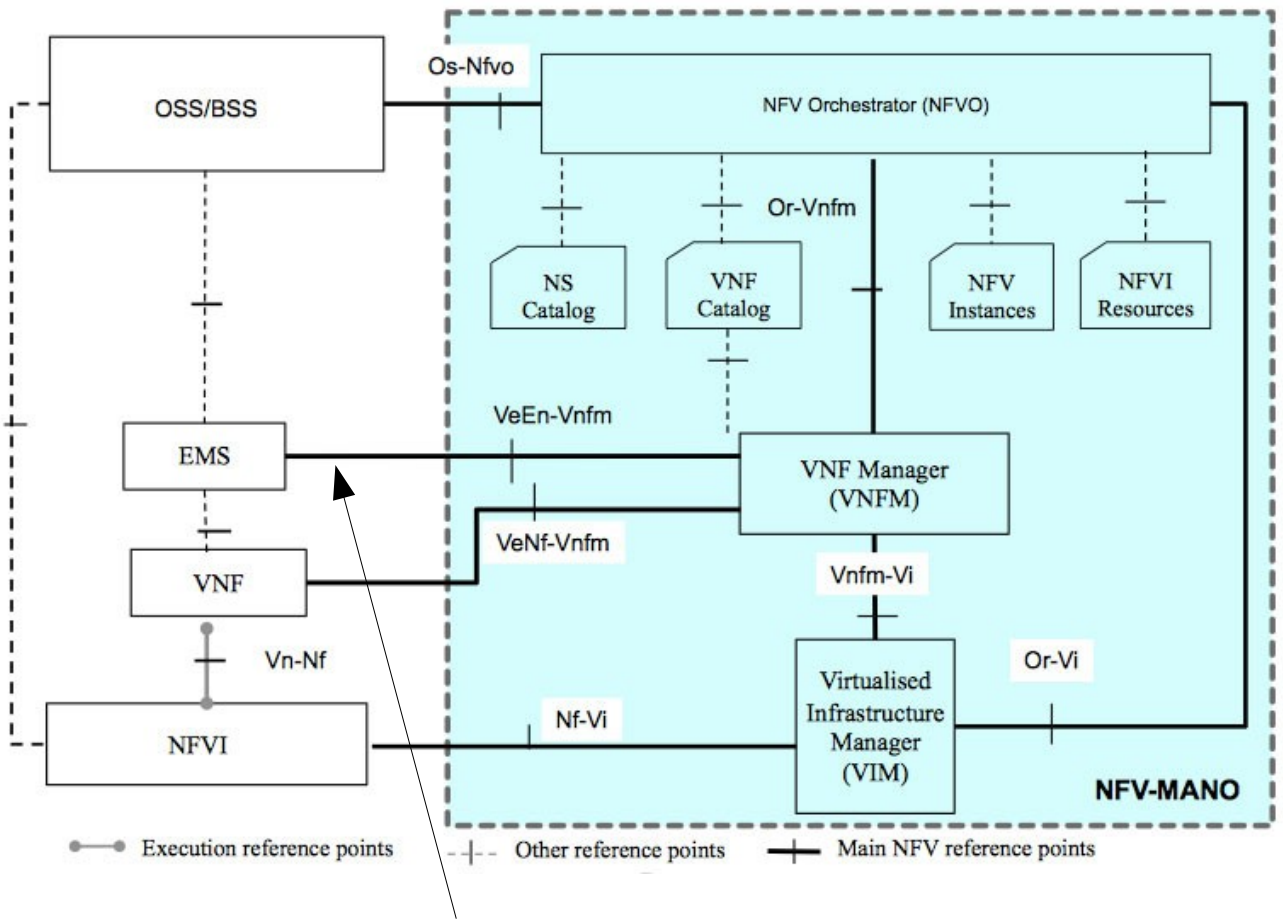
http://ravel-net.org/docs/SOSR16slide2.pdf

- Back to SQL for structured queries
  - Performance??? Looks OK (few ms for rule update)
    - Depends on the application

Resilience in NFV

NFVI failure Covered by OpenStack, VmWare

- But my VNF is gone!
- Use cloud-init to initially configure VNF
- But dynamic state?
- How is this EMS thing done?



This one is new compared to original NFV arch.
Is used for scaling in and out.

Resilience in NFV: Is my VNF up?

- NFV == OpenStack == KVM (at least until recently)
- VNF is a set of processes inside a VM
  - VNF may be down, though the VM may still be running
  - NFVI has no means to detect this!
- Can the „EMS“ find out? (original: element management system)
  - Constant monitoring required
  - EMS turns into „service control plane“

Resilience in NFV: Moving state out

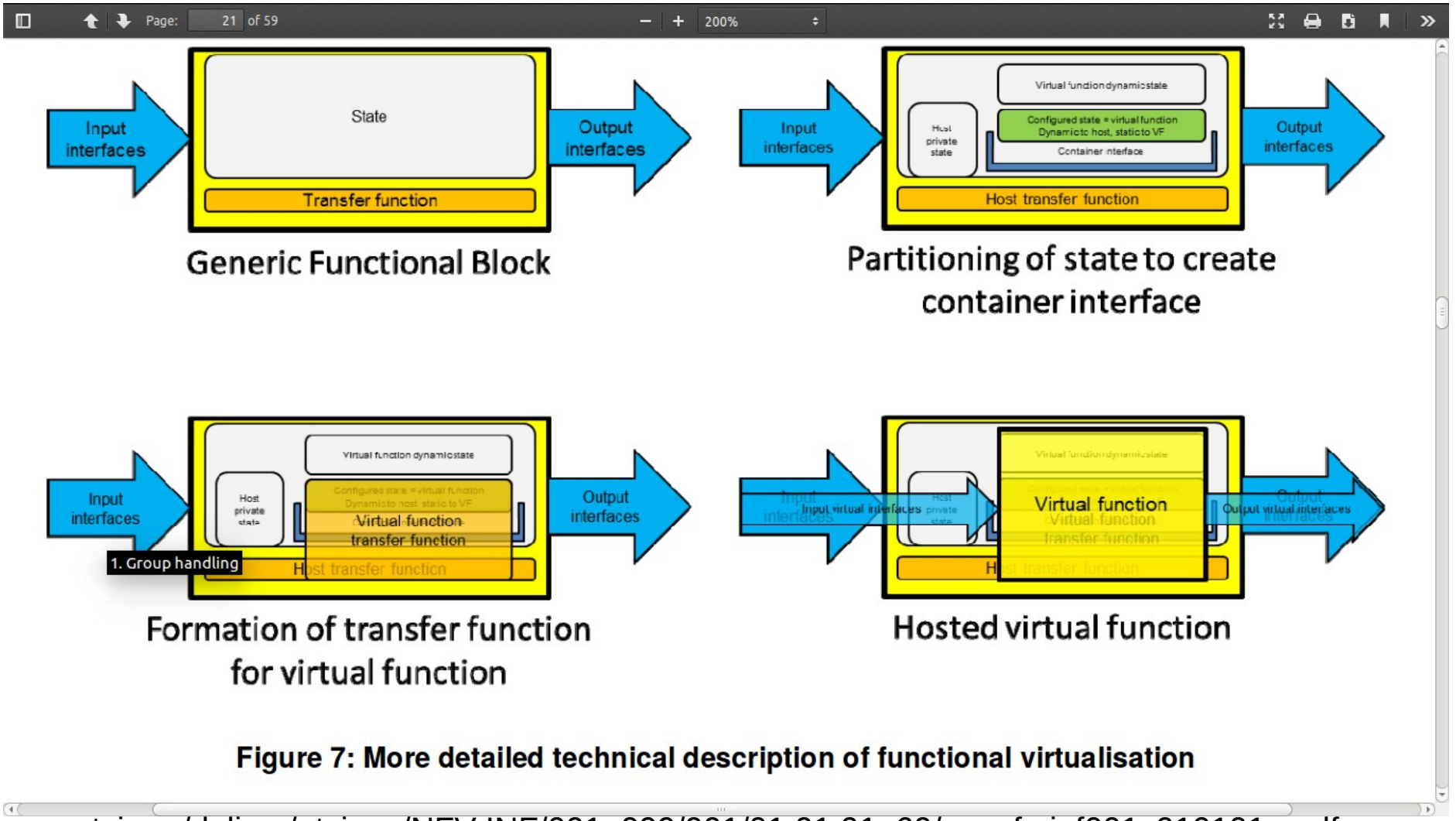


Figure 7: More detailed technical description of functional virtualisation

http://www.etsi.org/deliver/etsi_gs/NFV-INF/001_099/001/01.01.01_60/gs_nfv-inf001v010101p.pdf

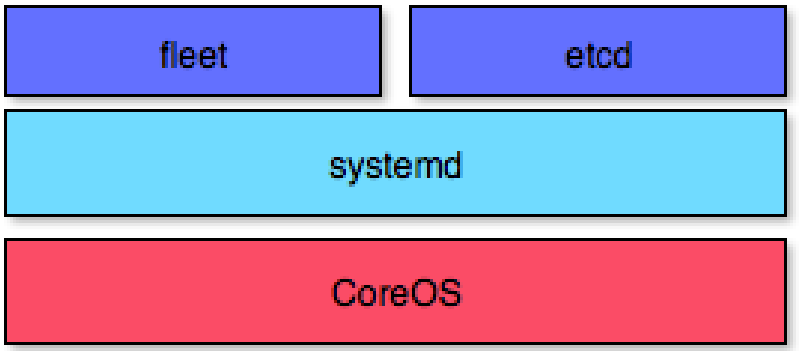
Reinventing services: Micro-services

- Containers (docker, lxc) ideally run a single process
- If process is done, container is down
- How is logging, monitoring etc. done?
  - There are tools for docker.io that allow to collect logging to stdout
  - How would this be split up into services?
- Use service-specific logging, i.e. integrate the control plane (“EMS”)

Reinventing Linux: CoreOS, systemd, etcd

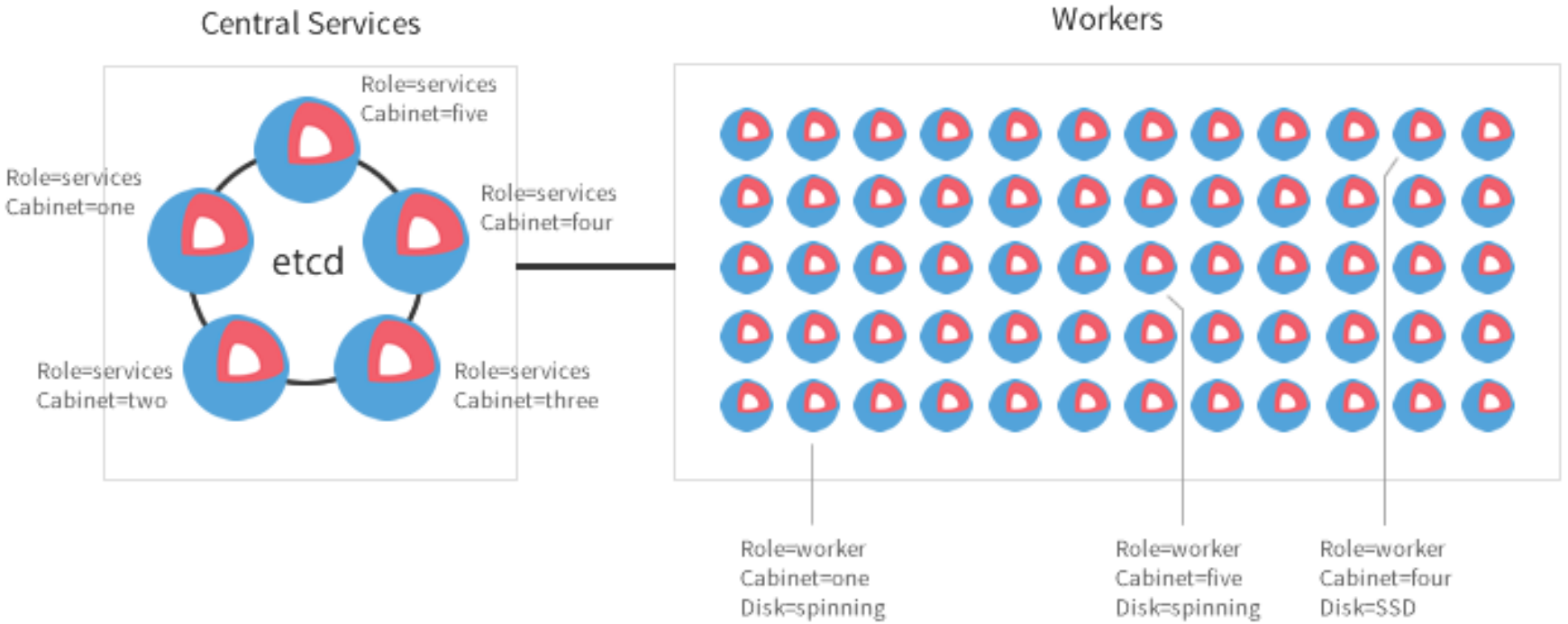
- Systemd is a replacement for SysV init() process
  - Accepted by major Linux distros (Fedora, Ubuntu)
  - Configuration of all services via /etc/systemd/… directory
  - Etcd (“/etc distributed”)
    - Distributed KV store
    - Raft consensus
  - Fleet allows remote steering of VNFs via etcd and systemd

https://github.com/coreos/etcd



CoreOS worker set running etcd

- Workers can log and receive data and instructions via /etc
  - Example: central services manage pool of datapath ids for switches, workers fetch a unique id before connecting to controller.



https://deis.com/blog/2016/coreos-overview-p2

Conclusions

- Resilience is preparation for loss of state
- General approach is to move out state to HA data bases
- SDN controllers and NFV workers use the same approach
- Re-designing NFVs to fit the model is essential, otherwise NFV resilience requirements cannot be met
- Move out as much state as possible from switches and NFVs
  - Replace rather than repair
  - In other words: treat switches and servers as cattle, not pets.