Softening the Network: Virtualization's Final Frontier

Steve Riley
Technical Director, Office of the CTO
Riverbed Technology
steve.riley@riverbed.com
http://blog.riverbed.com

Abstractions We've Seen

virtual memory

virtual disk volumes

virtual machines

→ the *illusion* of a thing

abstraction

no re-programming

sometimes is disruptive


VM-1
VM-3
VM-2
VM-4
PM-1
VM-1
VM-3
VM-2
PM-2

meets needs

provisioning

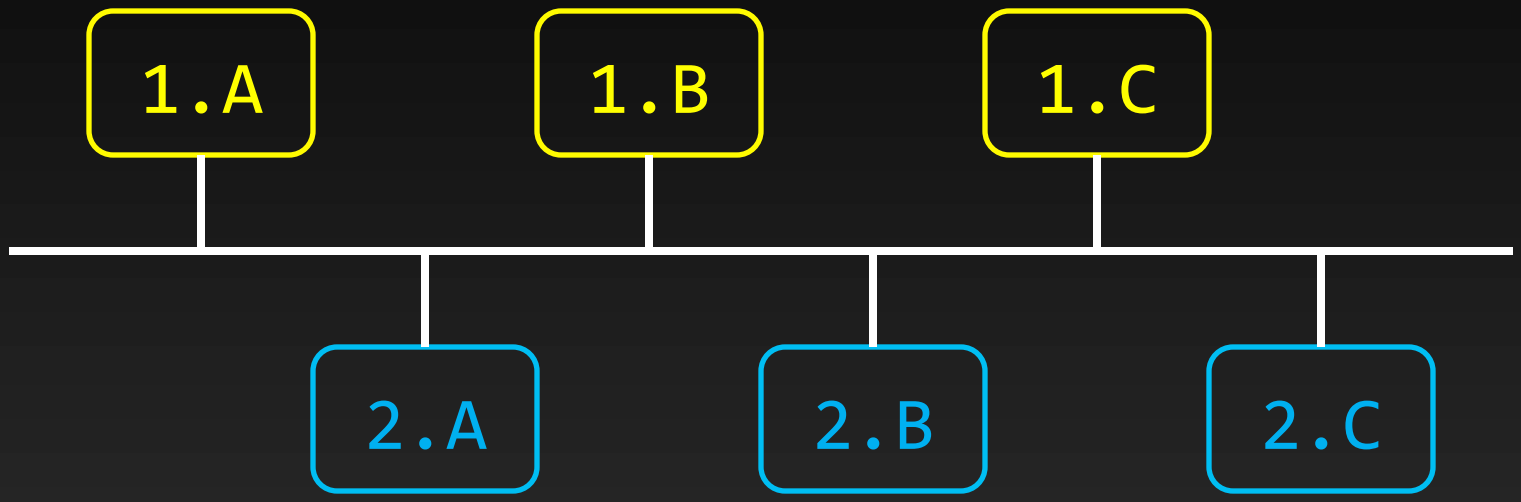
moving

snapshotting

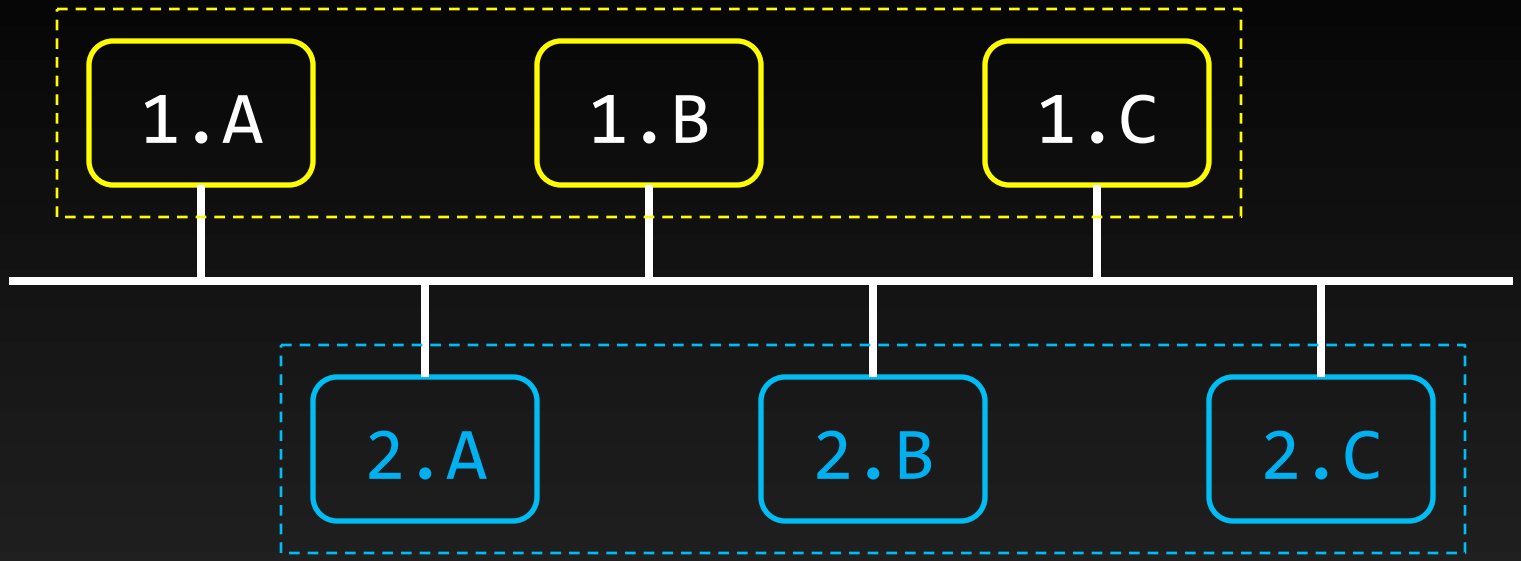
roll back

New?

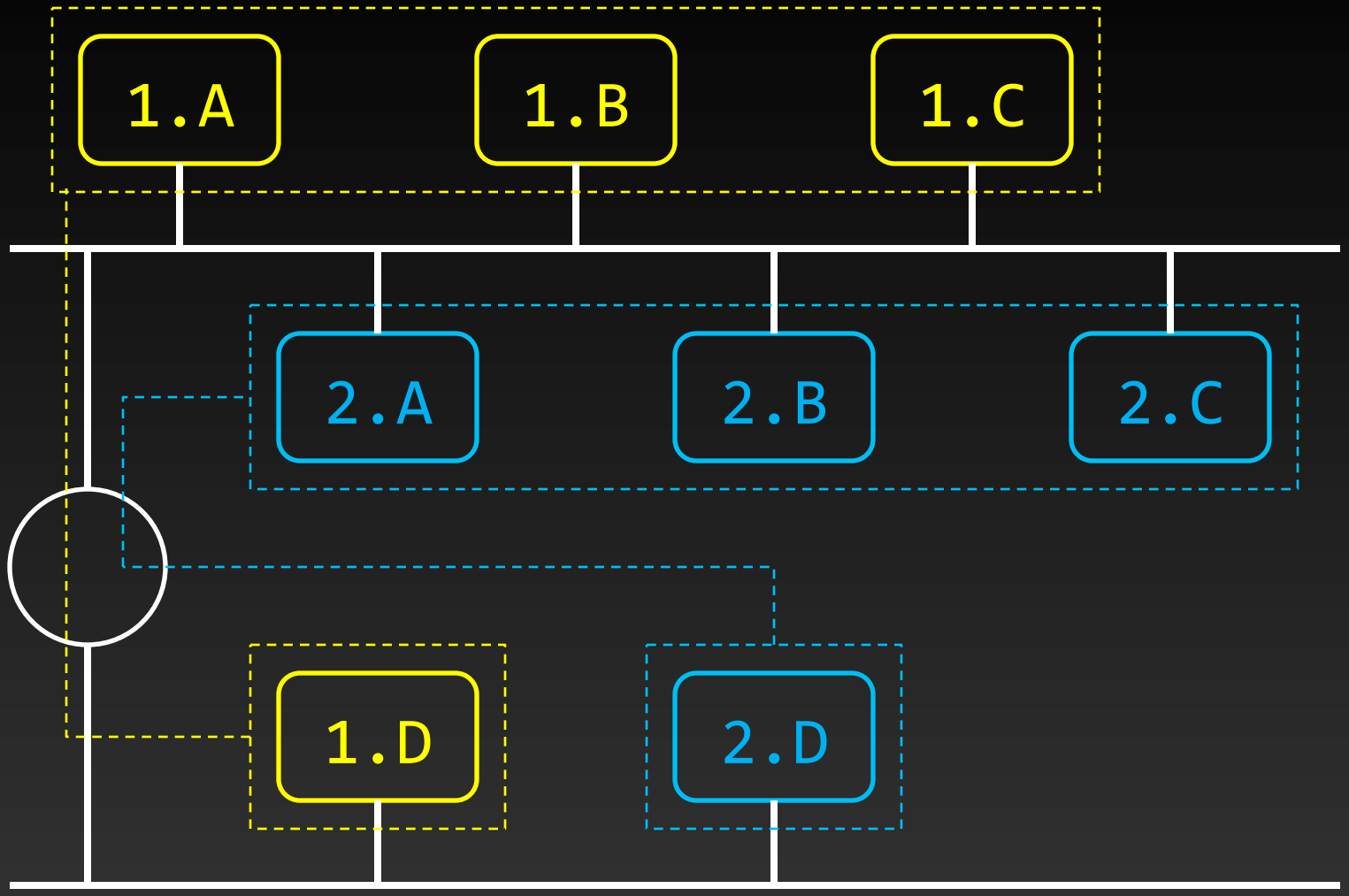
crude



less crude



less crude



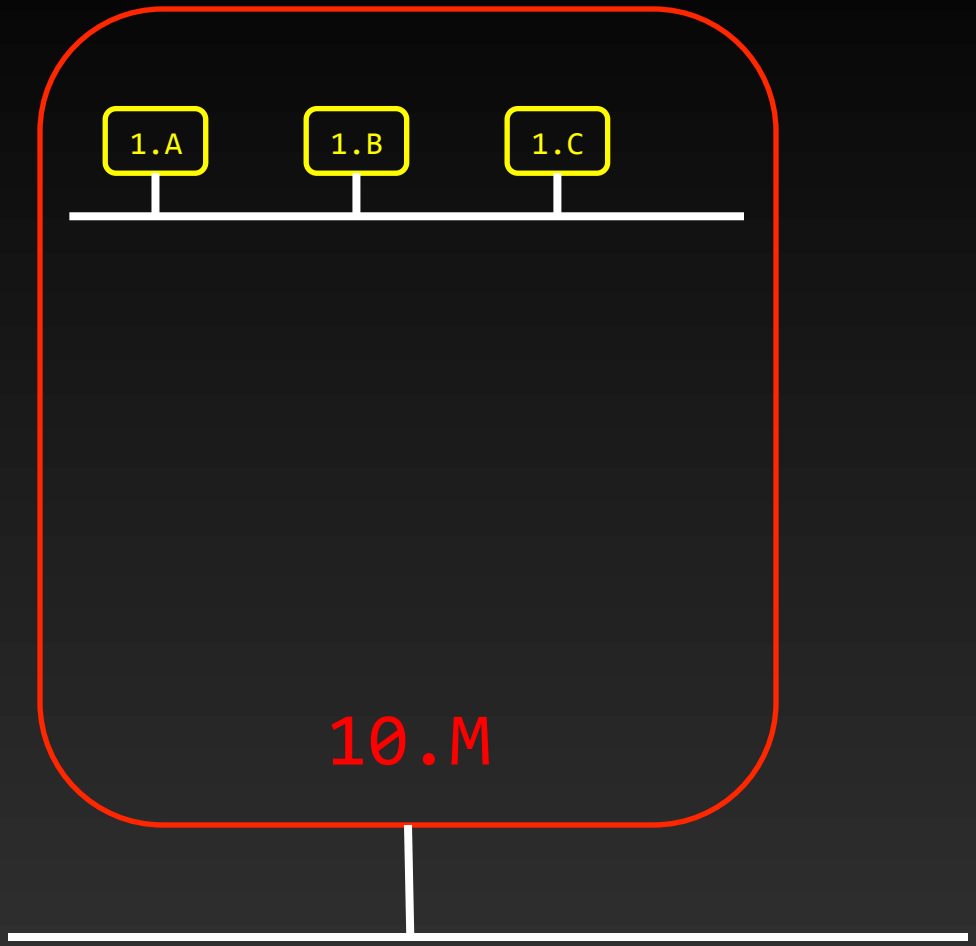
limitations

$$n < \infty$$

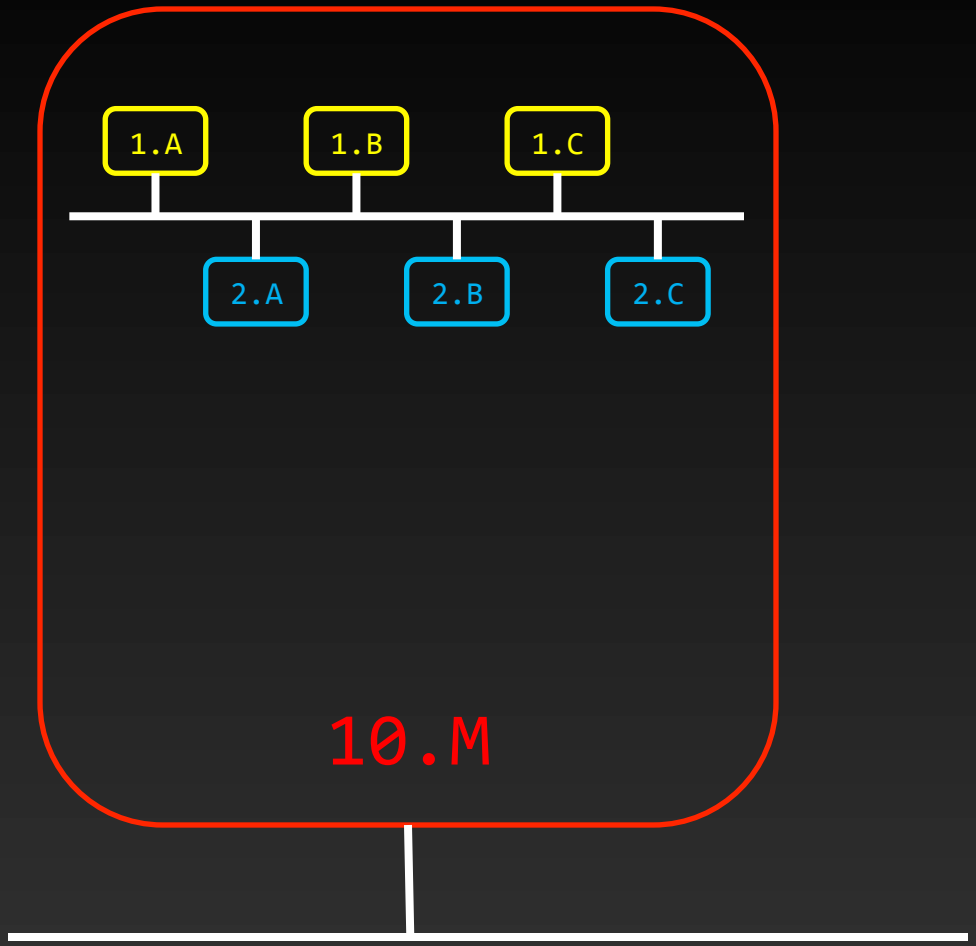
topology

static

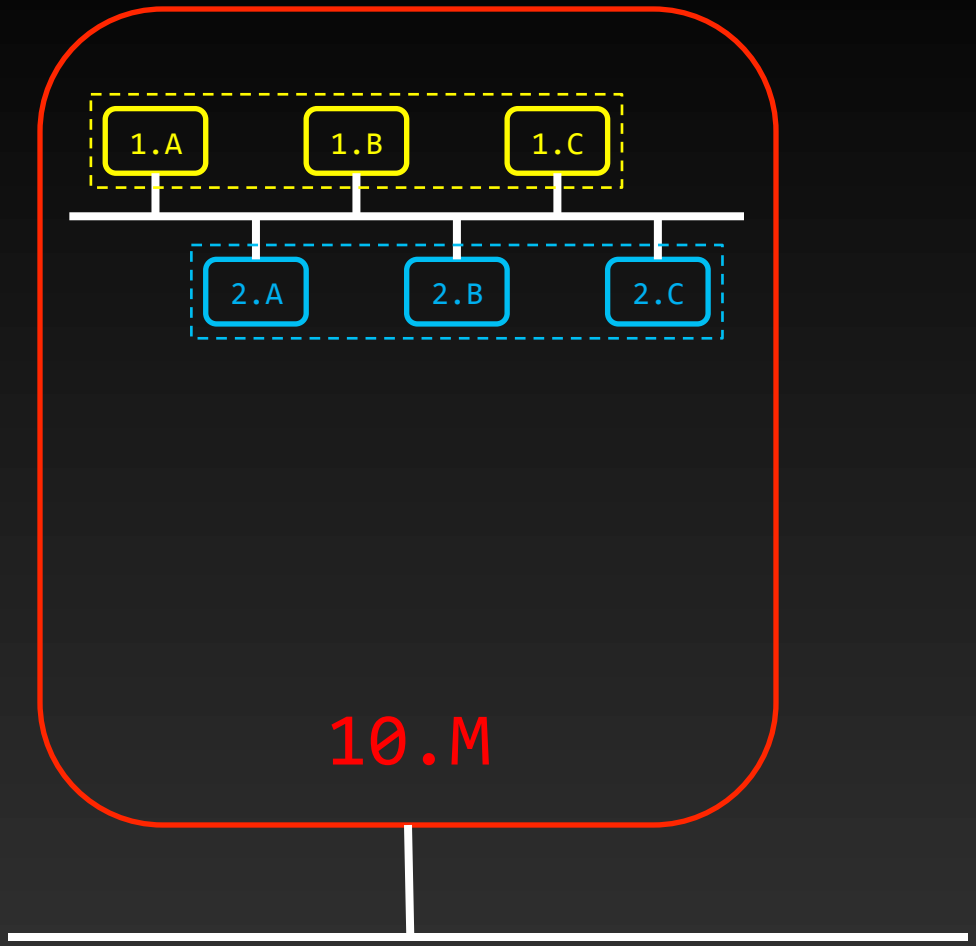
interesting



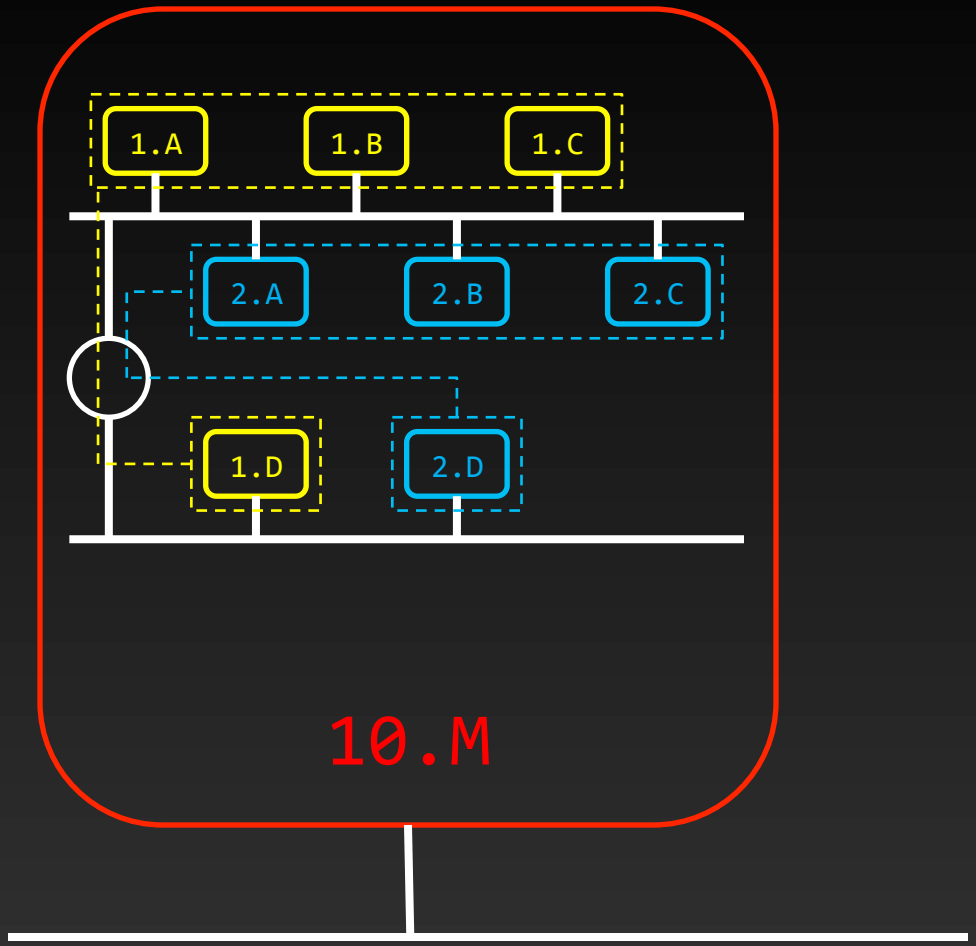
interesting +



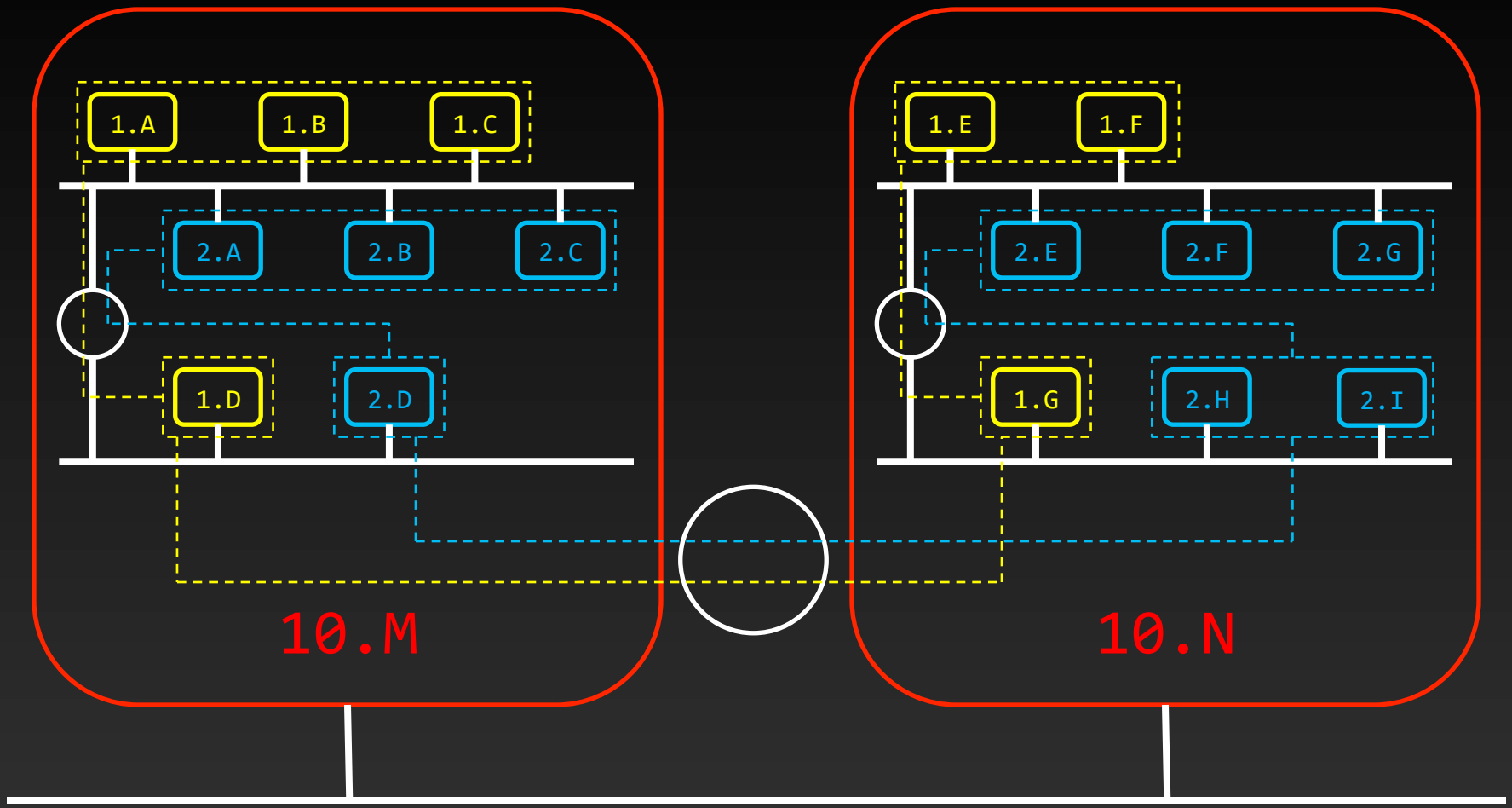
interesting +?



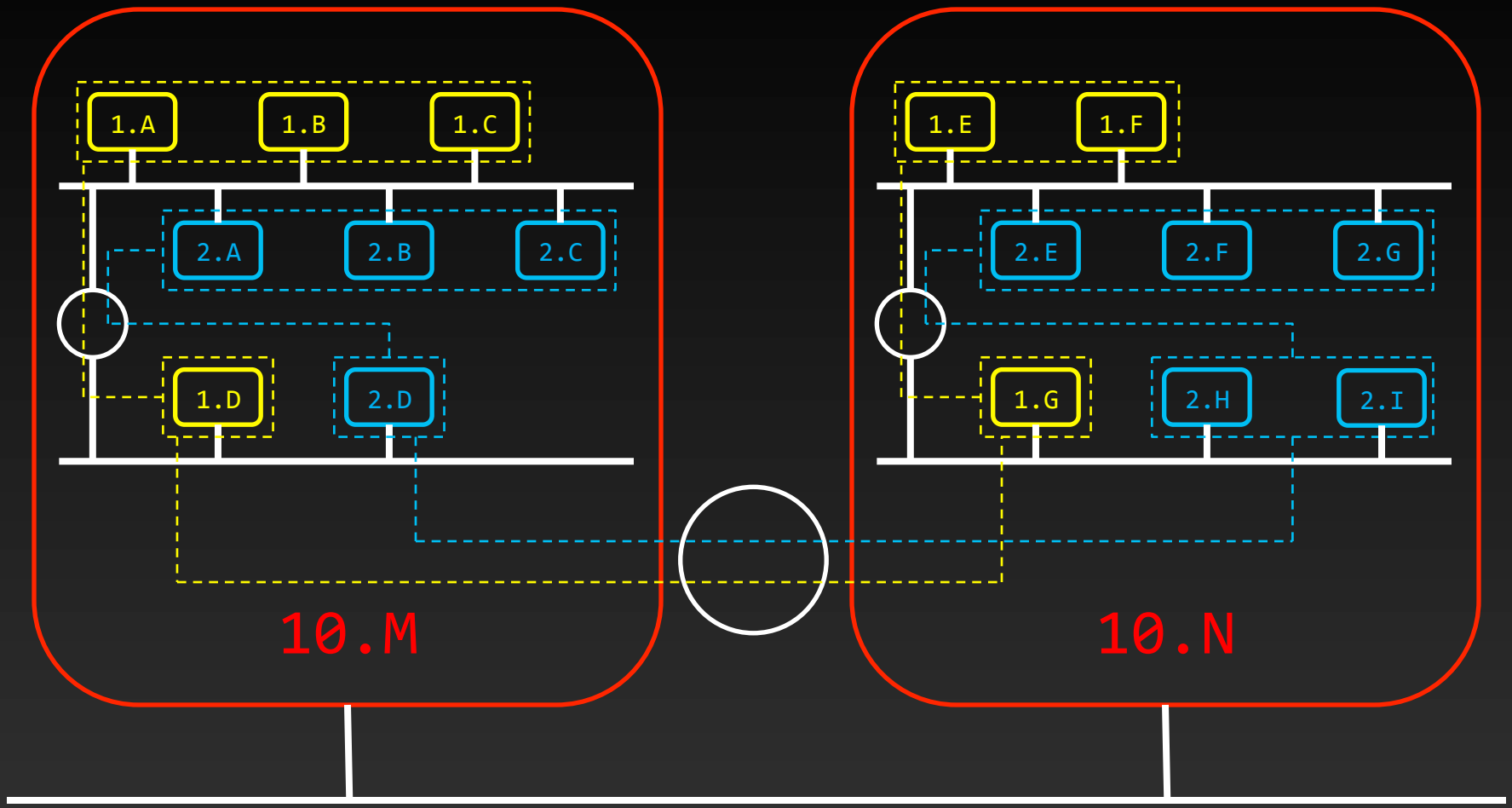
interesting +??



interesting +!!!



insane ;)

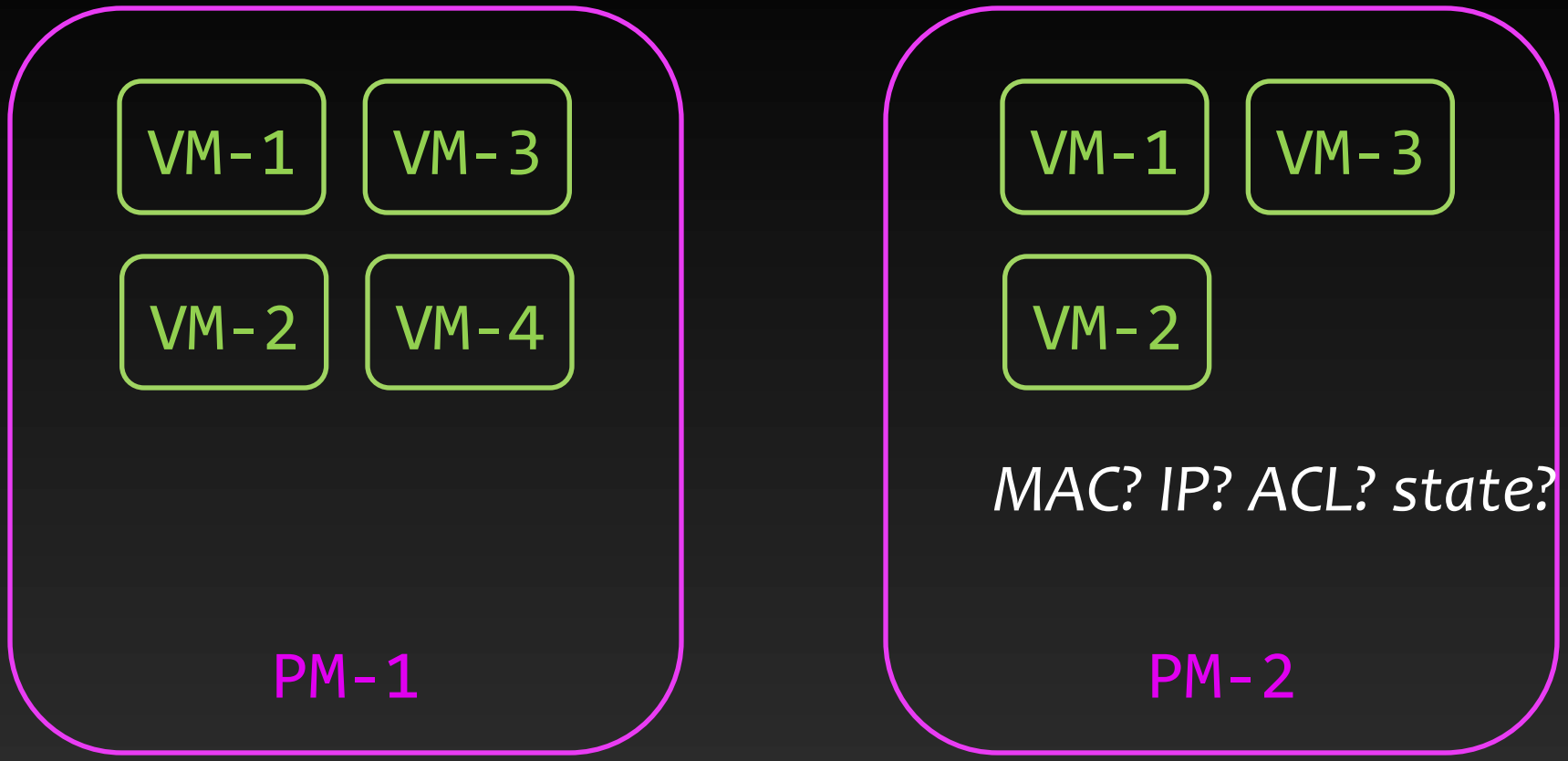


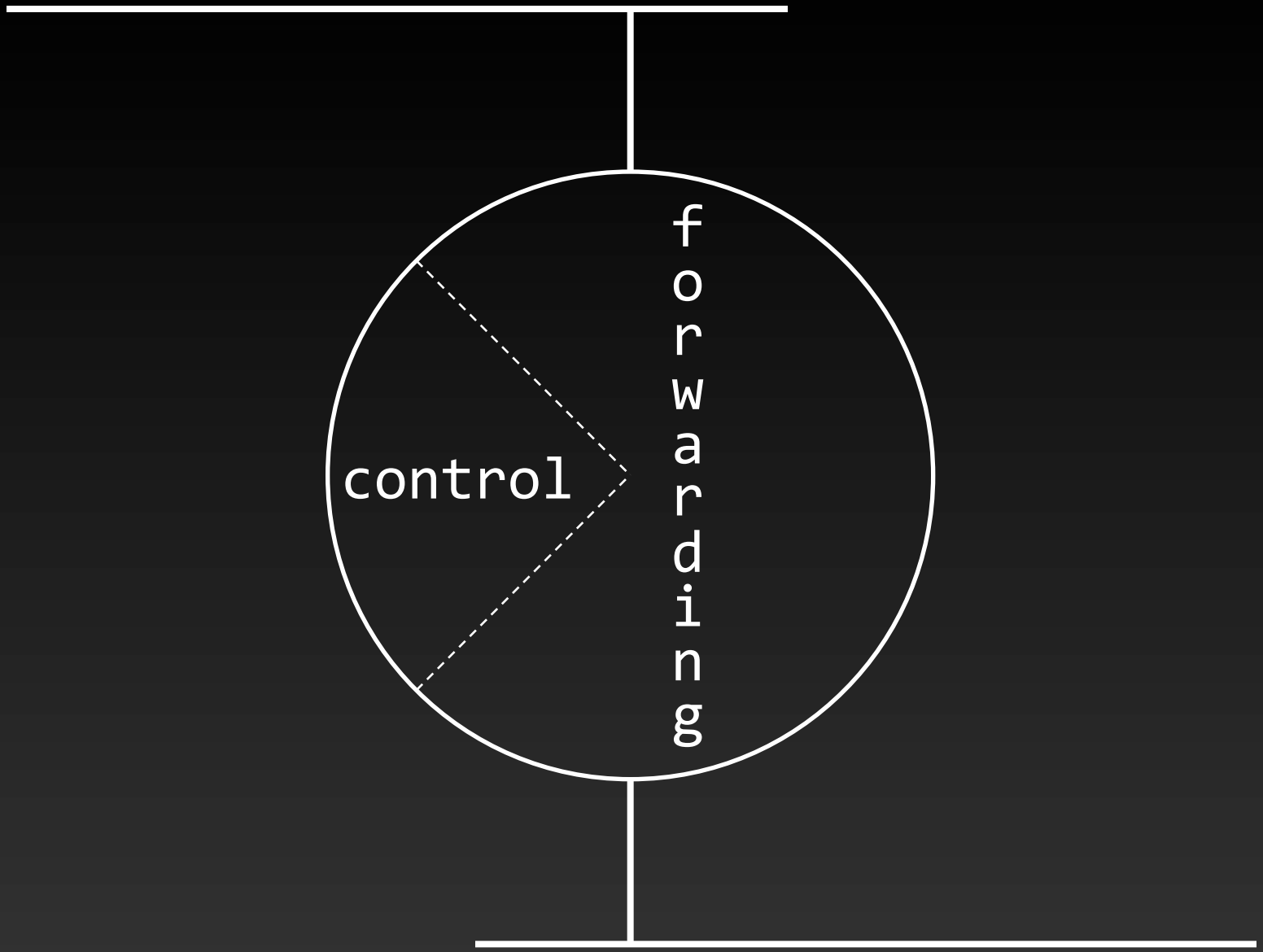
limitations

as before

+ not cloudable

operational abstractions aren’t useful


VM-1
VM-3
VM-2
VM-4
PM-1
VM-1
VM-3
VM-2
MAC? IP? ACL? state?
PM-2


control
forwarding

limitations

topology mandates/constraints

no overlapped addresses

slooooooow to change

+ not cloudable

requirements

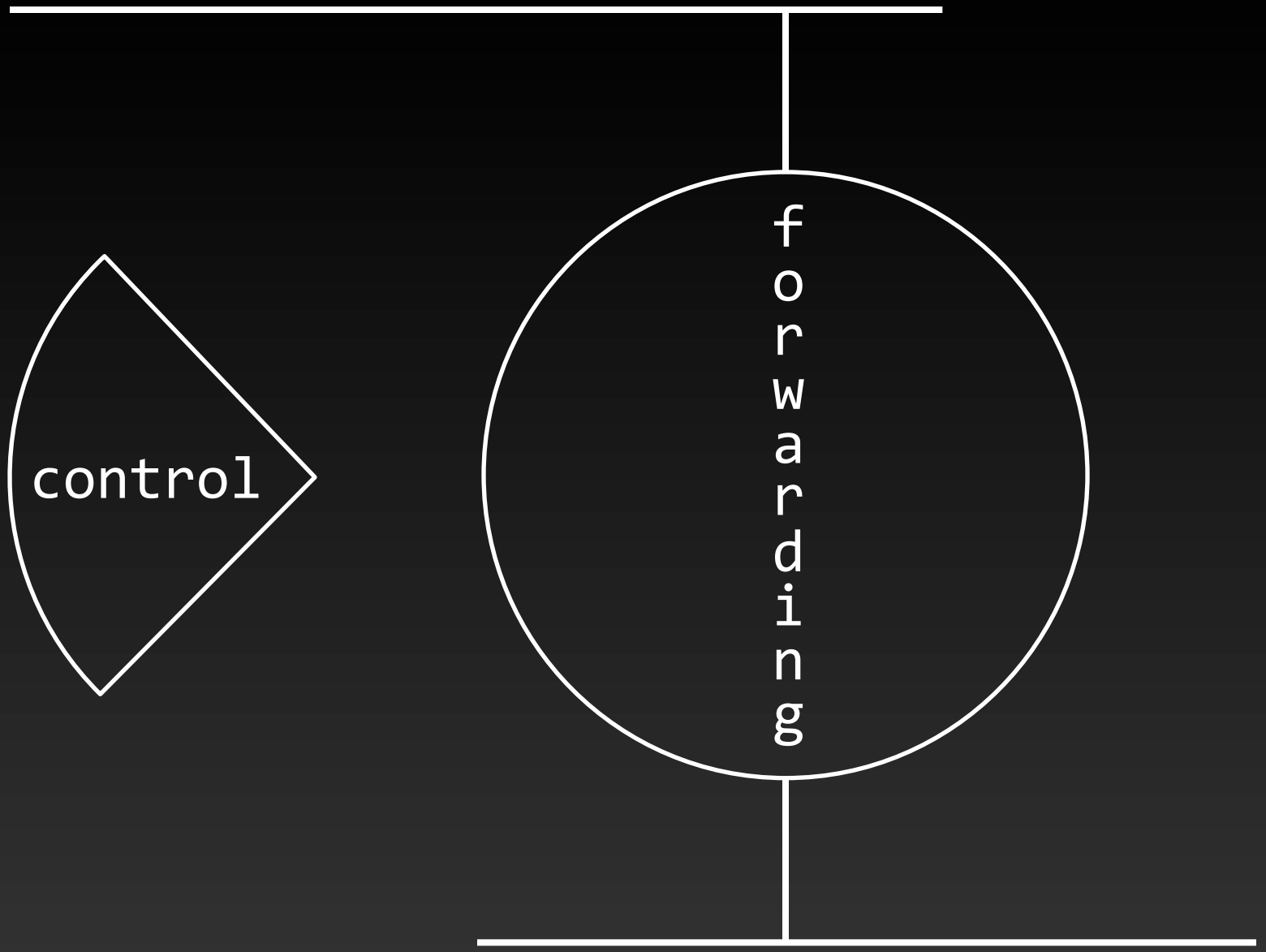
decouple V from P

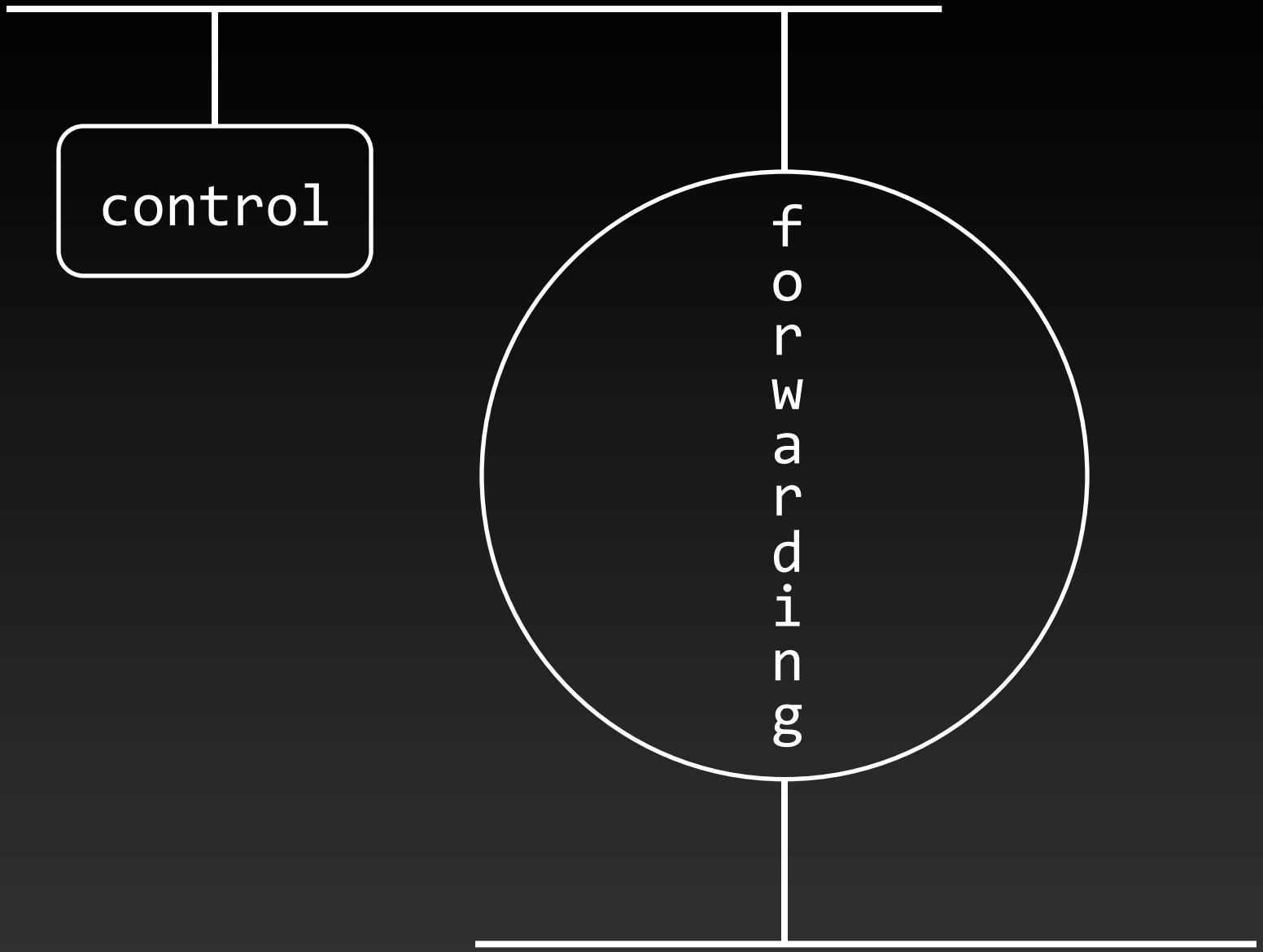
V looks like P

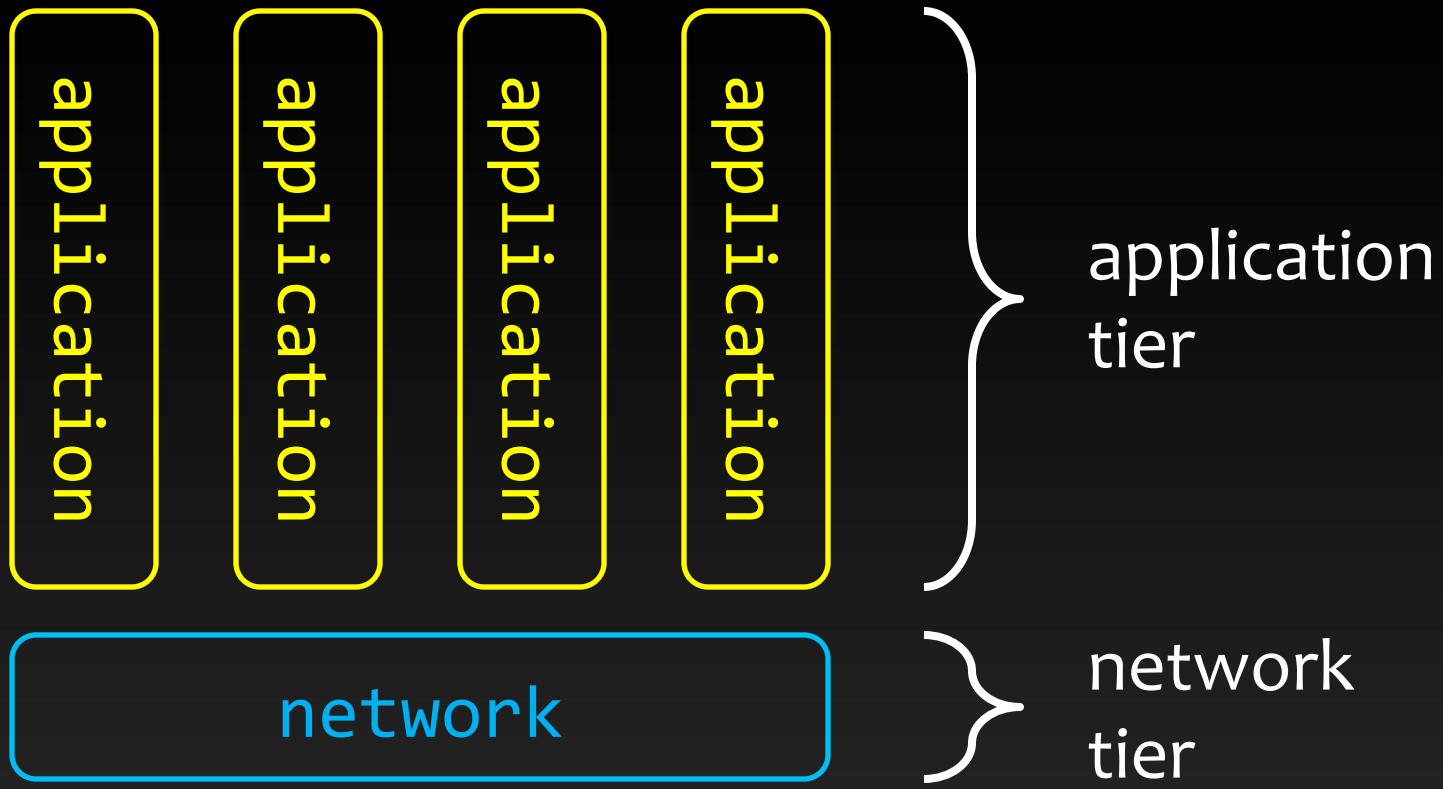
V allows units of operation

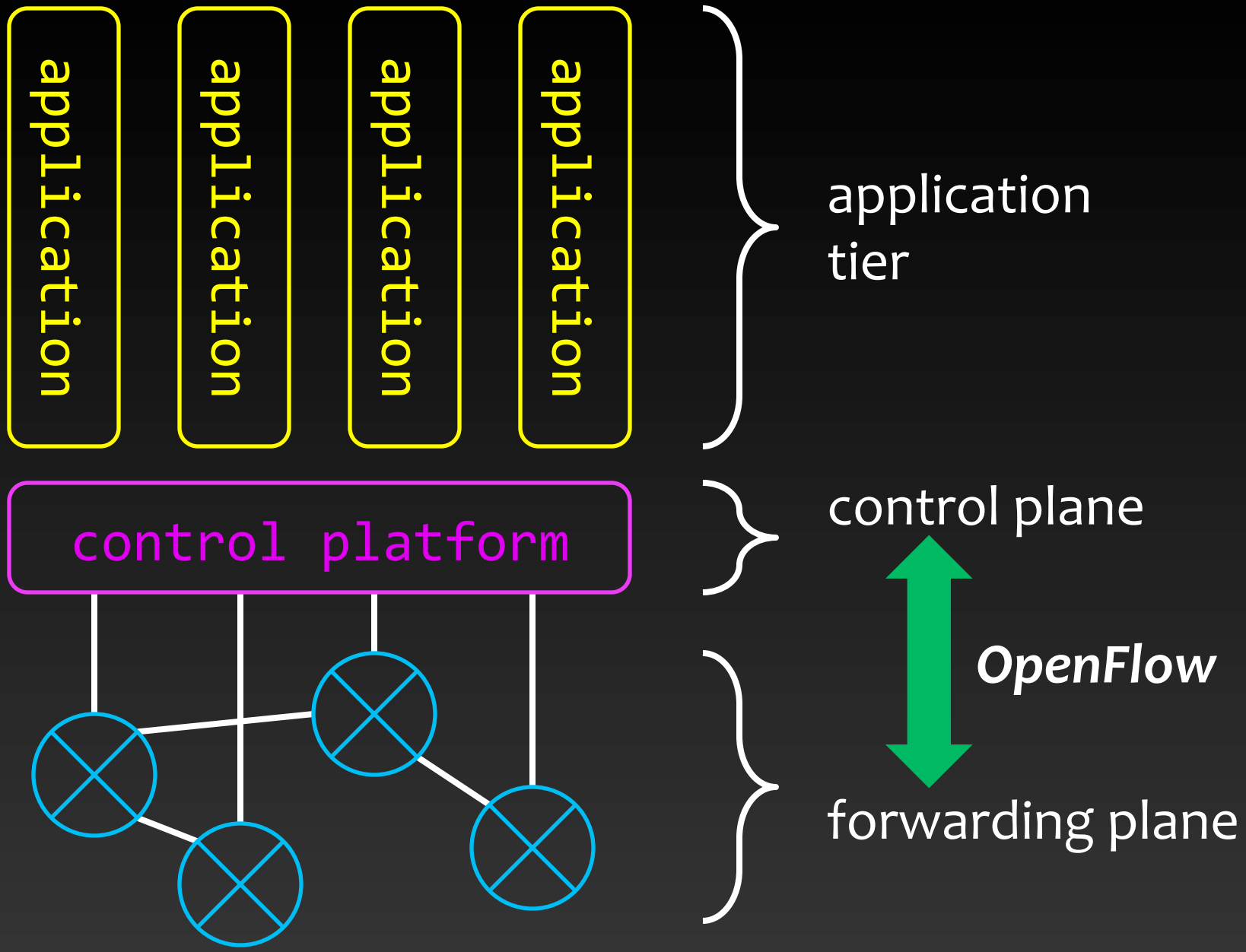
Software Defined Networking (*)

* One popular, but not necessarily universal, definition


control
forwarding


control
forwarding


application
application
application
application
application tier
network
network tier


application
application
application
application
application tier
control platform
control plane
OpenFlow
forwarding plane

forwarding plane

relatively “dumb”

does what it’s told

R.I.P.,

RIP

OSPF

IS-IS

&c.

control plane

centralized

end-to-end view
(not hop-by-hop)

programmable

naturally multitenant

maintains state


that’s it?

virtual server

$\neq$

virtual network

	Datapath	Consistency
Virtual server	CPU memory device I/O nanosecond operation	self-contained
Virtual network	address contexts	all-port knowledge N instances of N states consistency on all paths timely distribution

	Datapath	Consistency
Virtual server	CPU memory device I/O nanosecond operation **= complexity at speed**	self-contained
Virtual network	address contexts	all-port knowledge N instances of N states consistency on all paths timely distribution **= complexity at scale**

The Virtual Network

decoupled from h/w

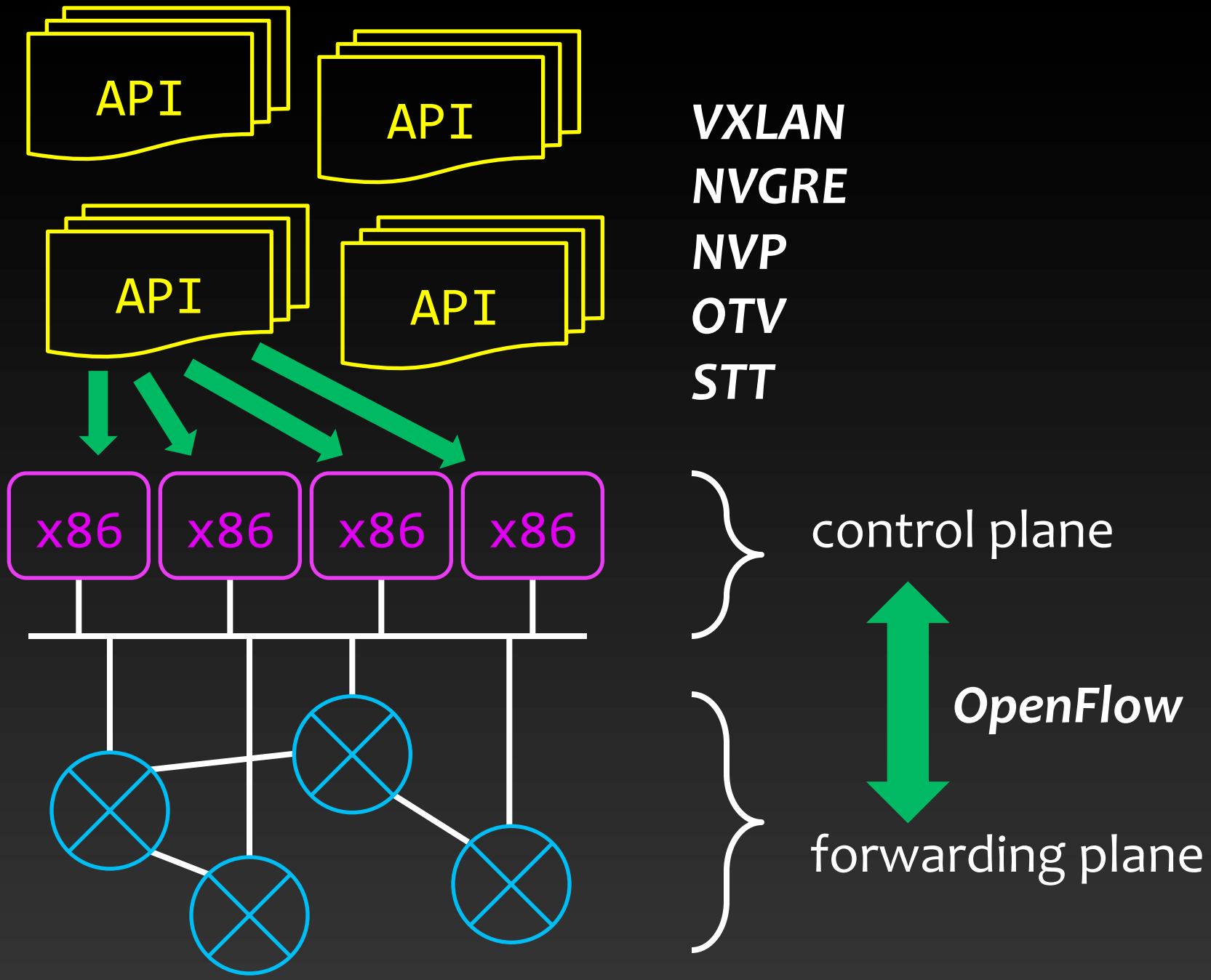
independent from others

delegated control

ephemeral

SDN (*) is a useful tool

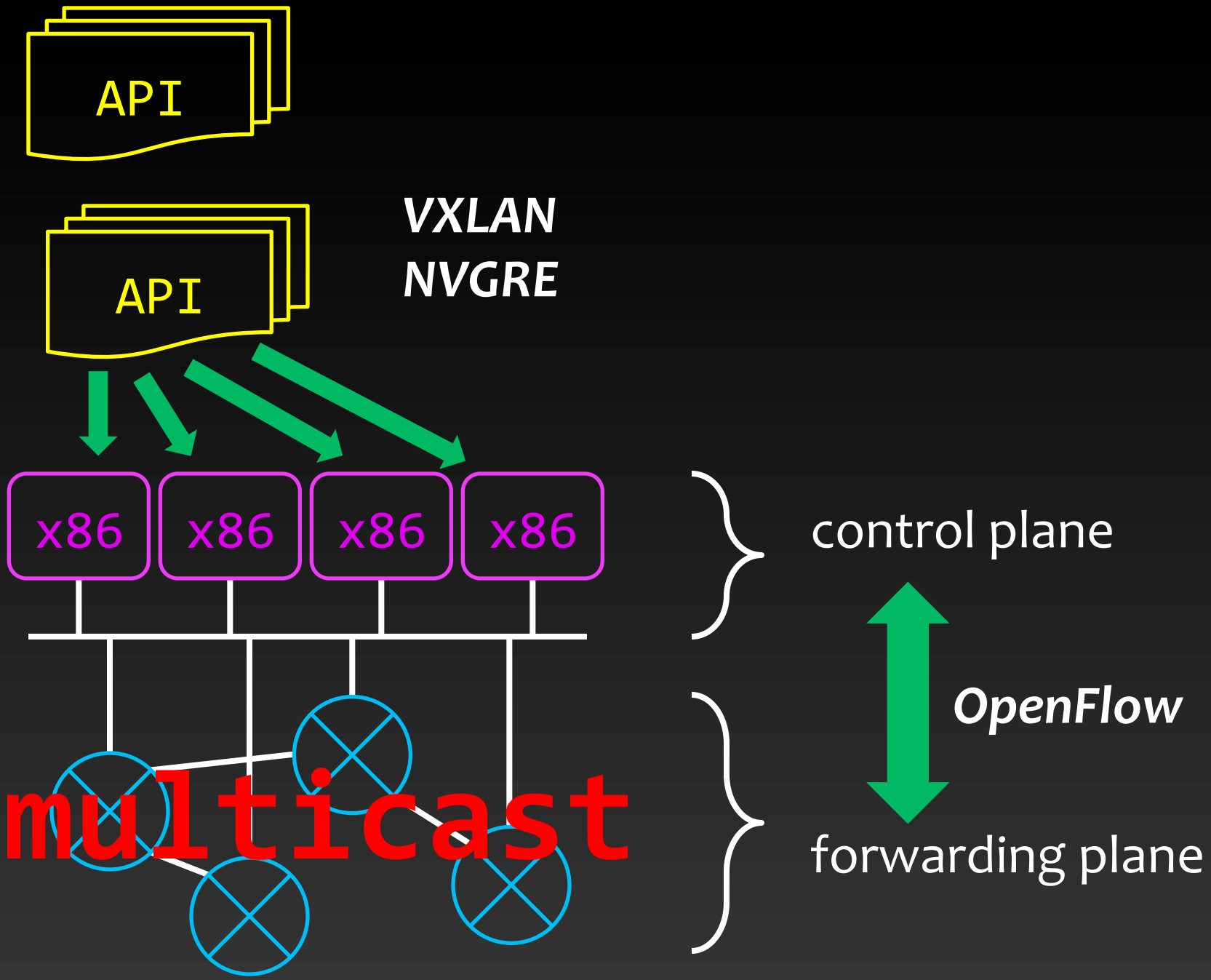
* As defined previously


API
API
API
API
x86
x86
x86
x86
VXLAN
NVGRE
NVP
OTV
STT
control plane
OpenFlow
forwarding plane

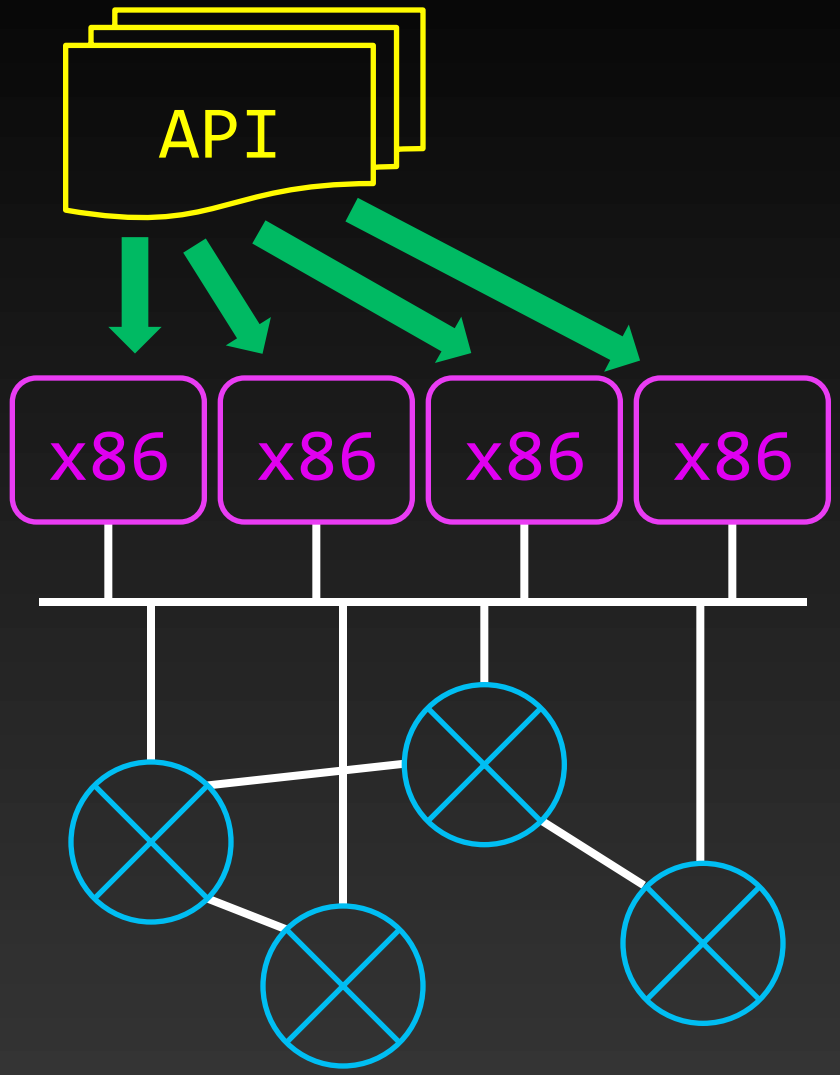
(

is SDN (*) a requirement?

* As defined previously


API
API
VXLAN
NVGRE
x86
x86
x86
x86
multicast
control plane
OpenFlow
forwarding plane

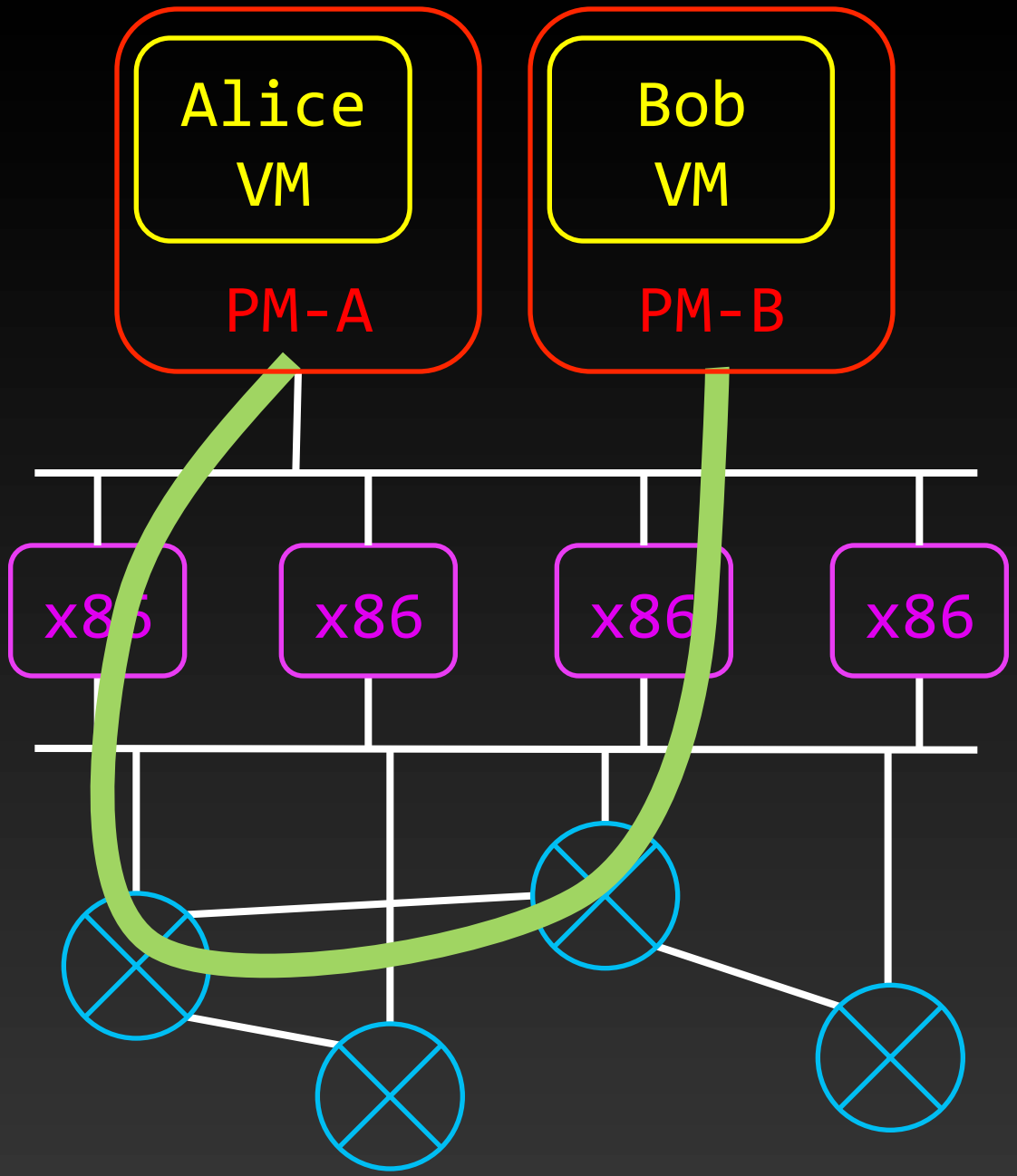
How does Alice talk to Bob?



vAPI :
"I need a virtual L2-L3 network with these properties…"

vSWITCHes in x86 boxes determine optimal path

OPENFLOW:
"Hardware, plumb the following…"


Alice
VM
Bob
VM
PM-A
PM-B
x86
x86
x86
x86

E  L

one flow, two VMs, separate hypervisors

	Throughput	Recv CPU	Send CPU
Linux bridge	9.3 Gbps	85%	75%
OVS bridge	9.4 Gbps	82%	70%
OVS-STT	9.5 Gbps	70%	70%
OVS-GRE	2.3 Gbps	75%	97%

aggregate, four VMs, two hypervisors

	Throughput	CPU
OVS bridge	18.4 Gbps	150%
OVS-STT	18.5 Gbps	120%
OVS-GRE	2.3 Gbps	150%

http://networkheresy.com/2012/06/08/the-overhead-of-software-tunneling/

possibilities

security application

QoS application

WAN op application (*)

* *hard:* distributed cache and symbol vocabulary

on-demand VPN/C

infrastructure → code

network → code

decoupled
and
delegated

physical L2-L3

logical L2-L3
L4-L7 services

x86

x86, really?

complex → much CPU

FW/LB use CPU at flow start

optimized stacks ↑ performance

↑ upgrade certainty

distinct

security

forwarding

shaping

priority

...

outcomes

working multitenancy

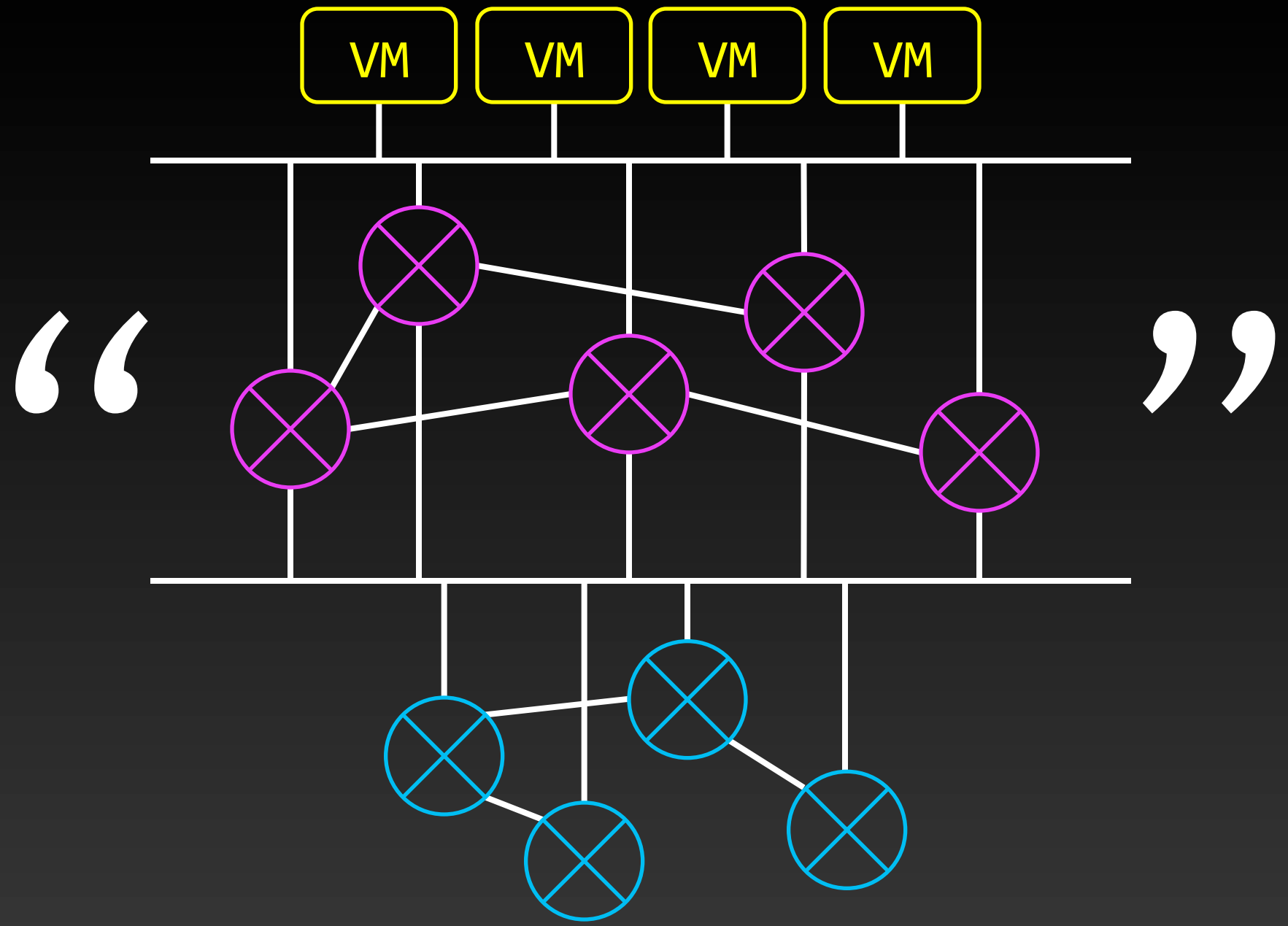
isolated addressing

programmable

independent

and

ephemeral


VM
VM
VM
VM

virtual IP
virtual MAC

route my packets/frames
without collisions

move v-net
without changes

tear down when finished

separately alter
physical and virtual
topologies

consider:

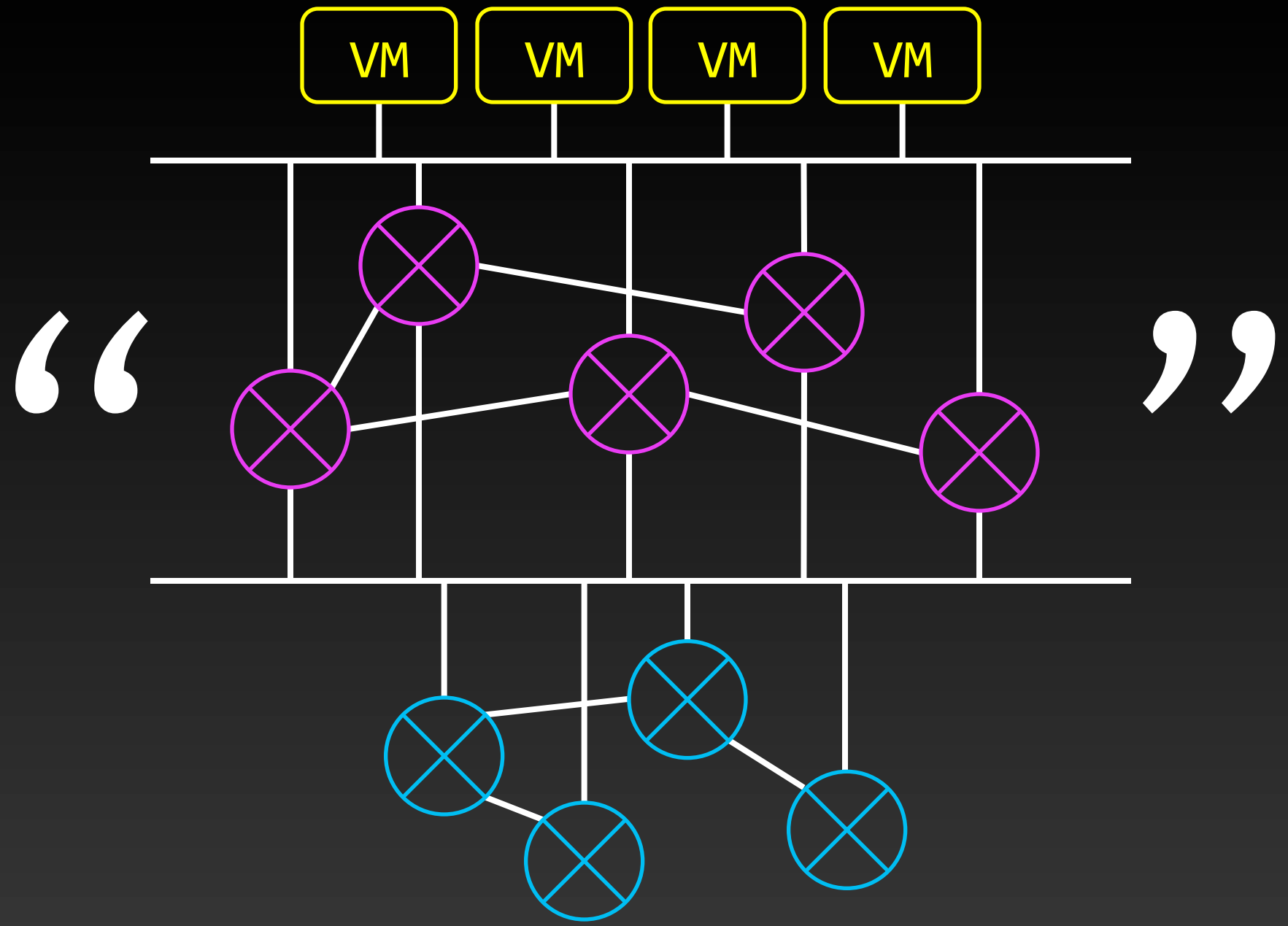
on-demand HA/DR

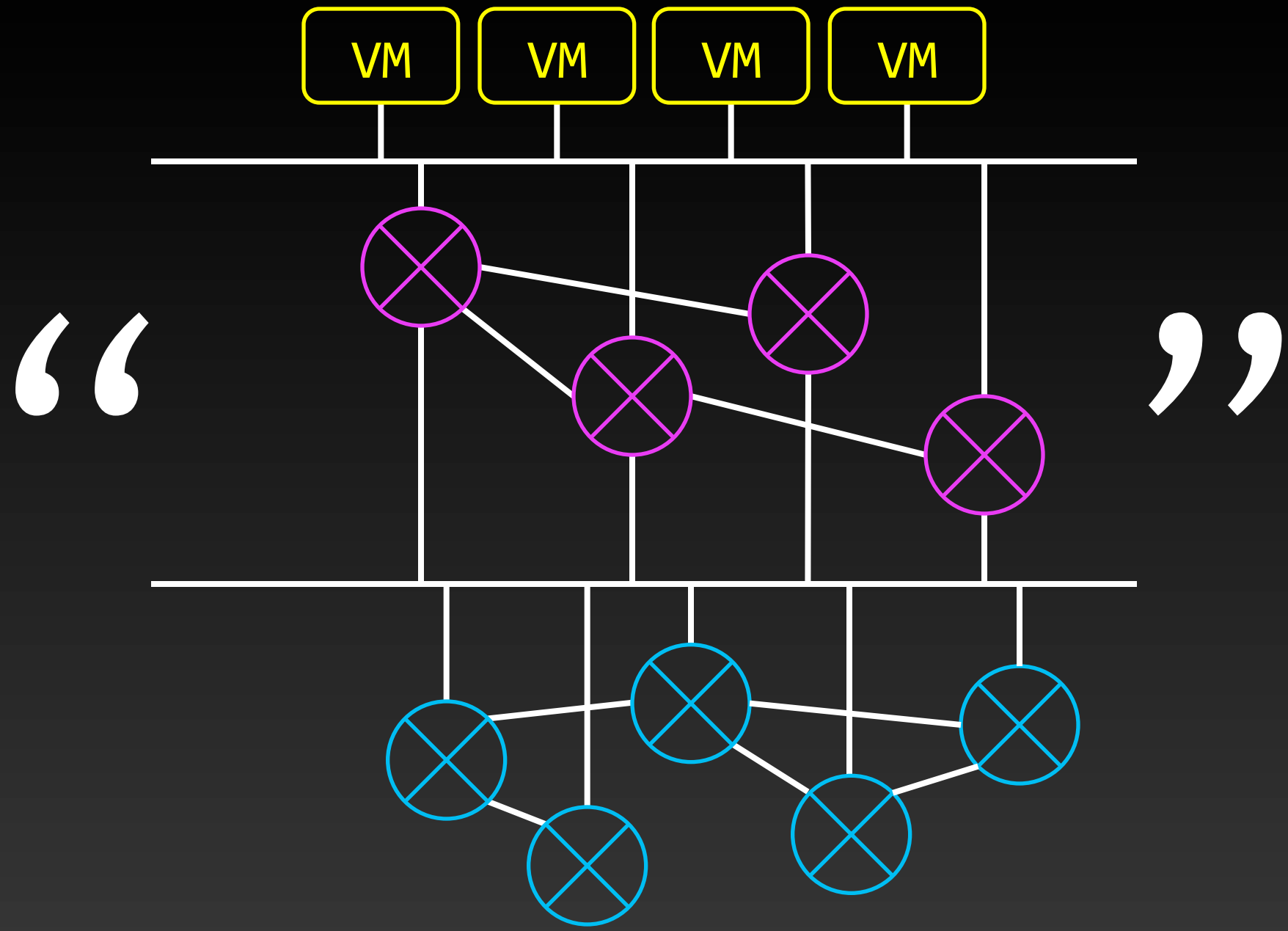
consider:

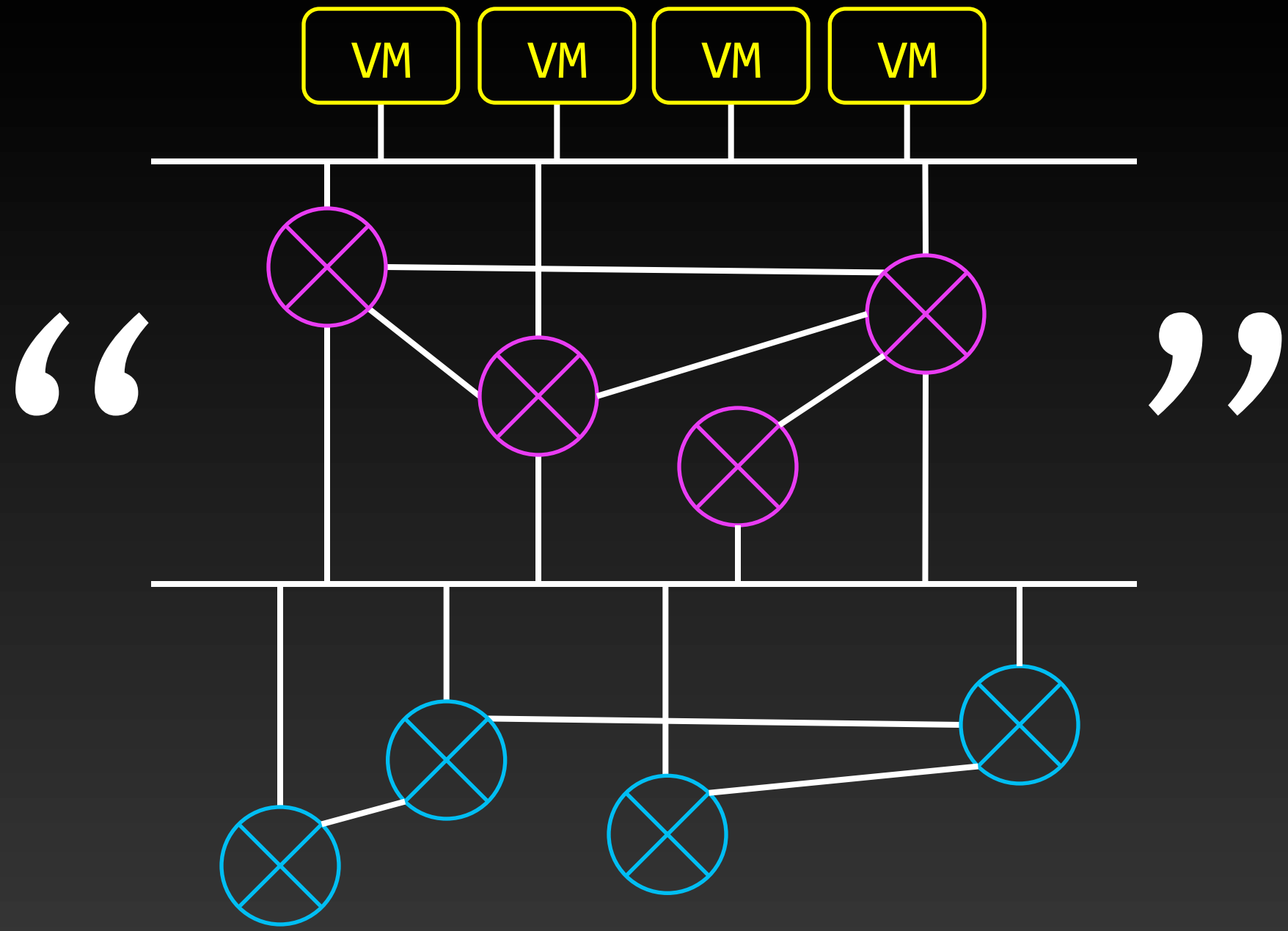
on-demand HDR

SDN (*) manages state

* As defined previously


VM
VM
VM
VM


VM
VM
VM
VM


VM
VM
VM
VM
“
”

abstractional consistency

(mature orchestration?)

servers

=

disposable horsepower

networks

=

disposable pathways

	Datapath	Consistency
Virtual server	CPU memory device I/O nanosecond operation **= complexity at speed**	self-contained
Virtual network	address contexts	all-port knowledge N instances of N states consistency on all paths timely distribution **= complexity at scale**

	Datapath	Consistency
Virtual server	CPU memory device I/O nanosecond operation **= complexity at speed**	self-contained
Virtual network	address contexts	all-port knowledge N instances of N states consistency on all paths timely distribution **= complexity at scale**

easy

familiar

↓

point solution ideas

Tagging	Segmentation, not isolation Same address in “both” worlds Hardware has to understand No mobility

Tagging	Segmentation, not isolation Same address in “both” worlds Hardware has to understand No mobility
Address mapping	Like NAT: update address in place Multiplex large space into small: how? Virtual-to-virtual: physical “punch”

Tagging	Segmentation, not isolation Same address in “both” worlds Hardware has to understand No mobility
Address mapping	Like NAT: update address in place Multiplex large space into small: how? Virtual-to-virtual: physical “punch”
Encapsulation	Or tunnels, or overlays (sigh) Worlds can be totally distinct Different forwarding for V and P Strong isolation: no V on P w/o bridge

	Datapath	Consistency
Virtual server	CPU memory device I/O nanosecond operation **= complexity at speed**	self-contained
Virtual network	address contexts	all-port knowledge N instances of N states consistency on all paths timely distribution **= complexity at scale**

programmability
and cloudability

hard

scary

↓

innovative advancements

Resources

networkheresy.com

packetpushers.net

blog.ioshints.con

sdncentral.com

Thanks for coming!

Steve Riley
Technical Director, Office of the CTO
Riverbed Technology
steve.riley@riverbed.com
http://blog.riverbed.com