PATTERNS IN NETWORK ARCHITECTURE:

CLOUD COMPUTING

OUTLINE

1 Discussion of Alloy homework (net4.als)

2 Discussion of “VL2: A scalable and flexible data center network”

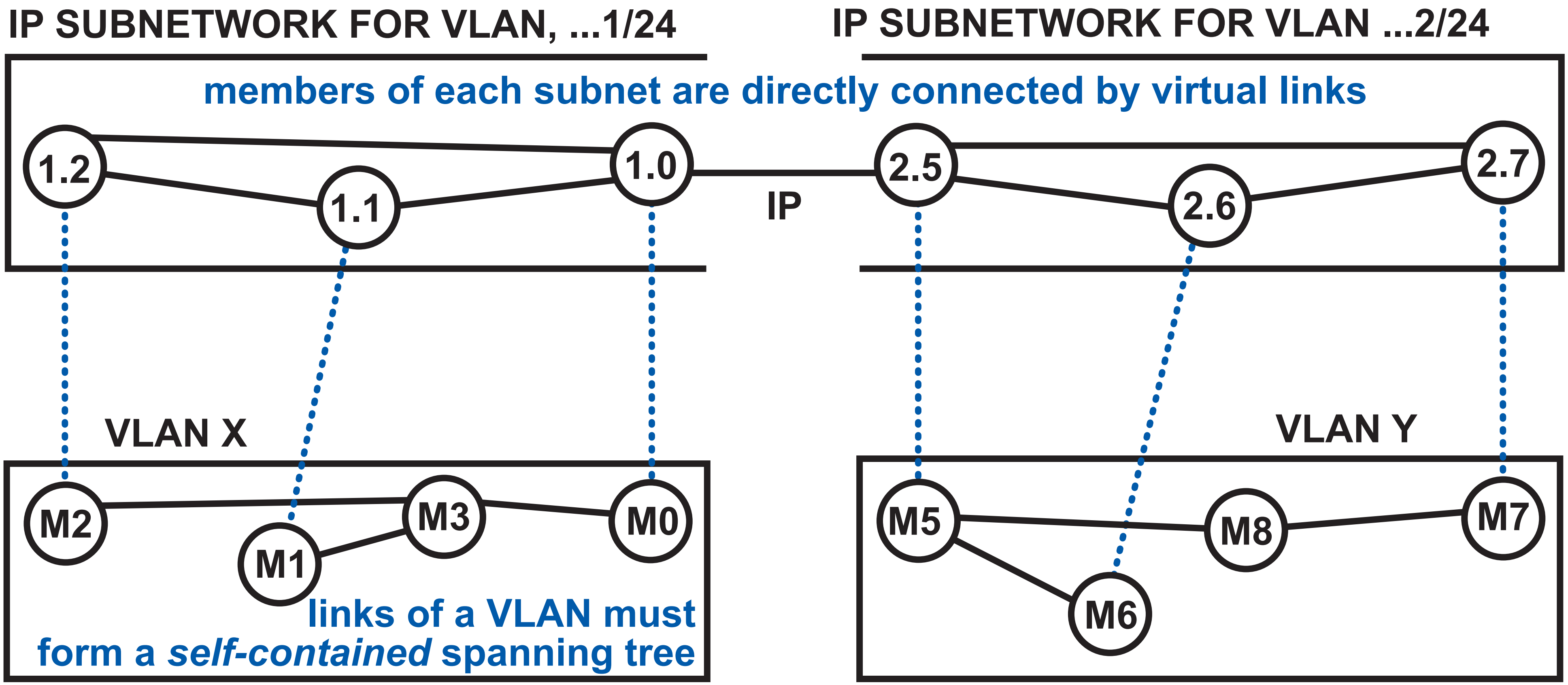
3 Models of VL2 and SEATTLE

4 Discussion of “Stratos: A network-aware orchestration layer for virtual middleboxes in clouds”

5 Model of a cloud design and comparisons to literature

6 The Nicira paper

VLAN TECHNOLOGY (ACCORDING TO VL2 PAPER)



IP SUBNETWORK FOR VLAN, ...1/24

IP SUBNETWORK FOR VLAN ...2/24

members of each subnet are directly connected by virtual links

VLAN X

links of a VLAN must form a *self-contained* spanning tree

VLAN Y

although each LAN has
2 access routers, inter-LAN
bandwidth is severely limited

LAN A

all members of VLAN X must be connected to this LAN

LAN B

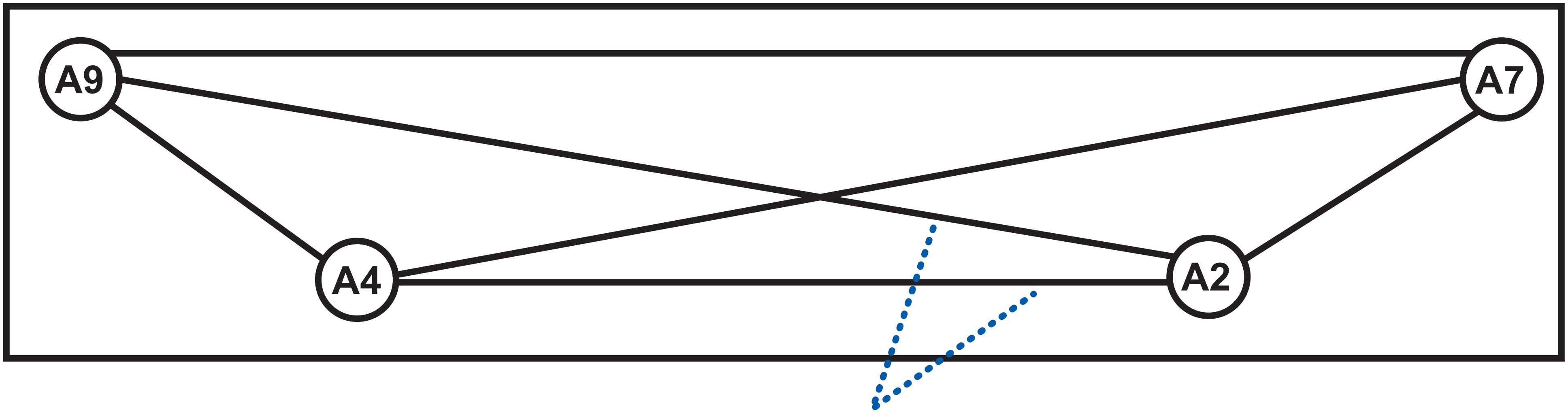
all members of VLAN Y must be connected to this LAN

VL2 ARCHITECTURE

connections to the public
Internet are not shown

VL2:

names are a random subset of the AA space



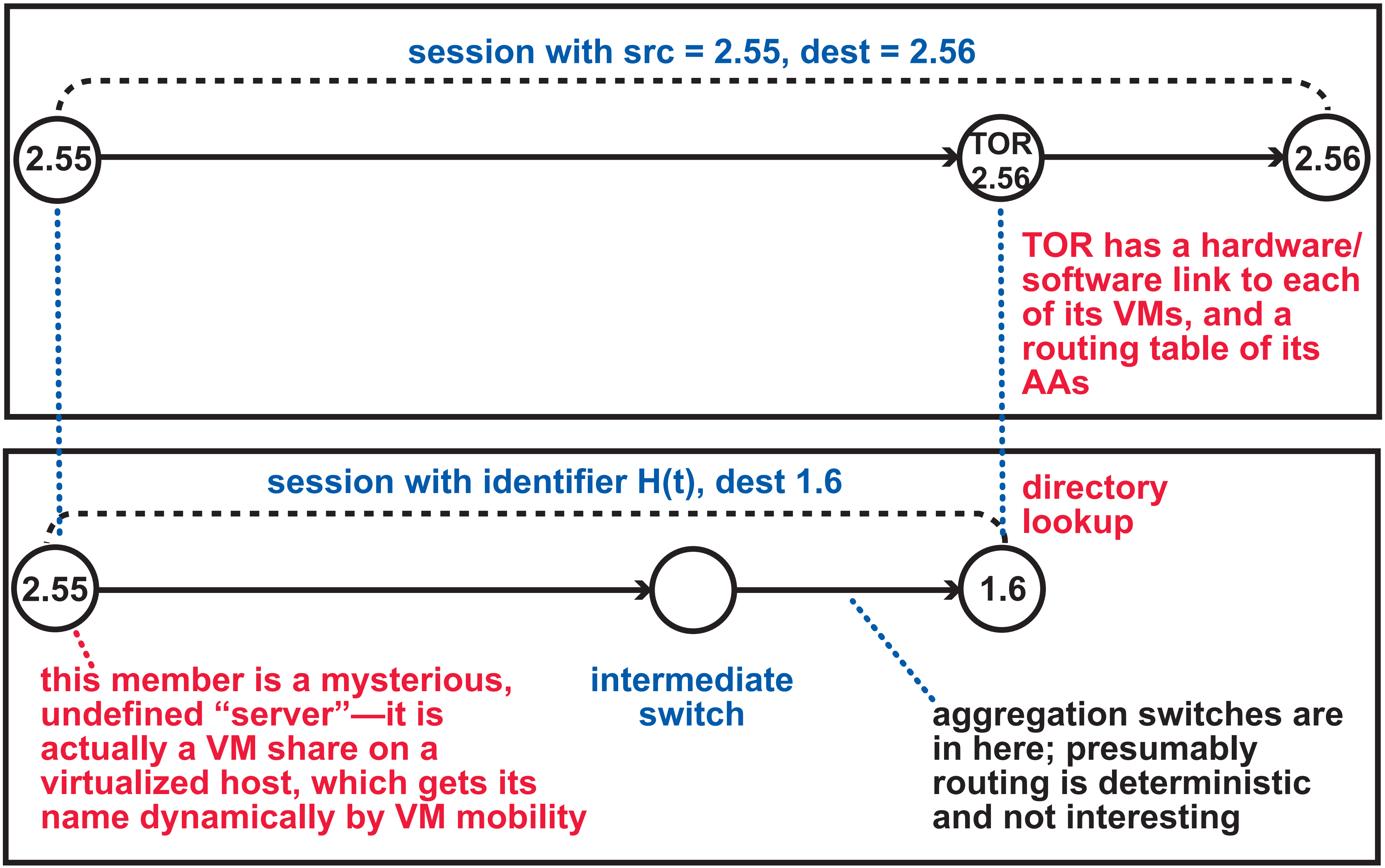
members are fully connected
by dynamic links

a member is a virtual machine,
meaning the data and
processing state—what
might be called the VM “image”

a VM can migrate from one
location to another,
without changing anything
in the VL2

VL2 ARCHITECTURE

VL2



session with src = 2.55, dest = 2.56

TOR has a hardware/ software link to each of its VMs, and a routing table of its AAs

session with identifier H(t), dest 1.6

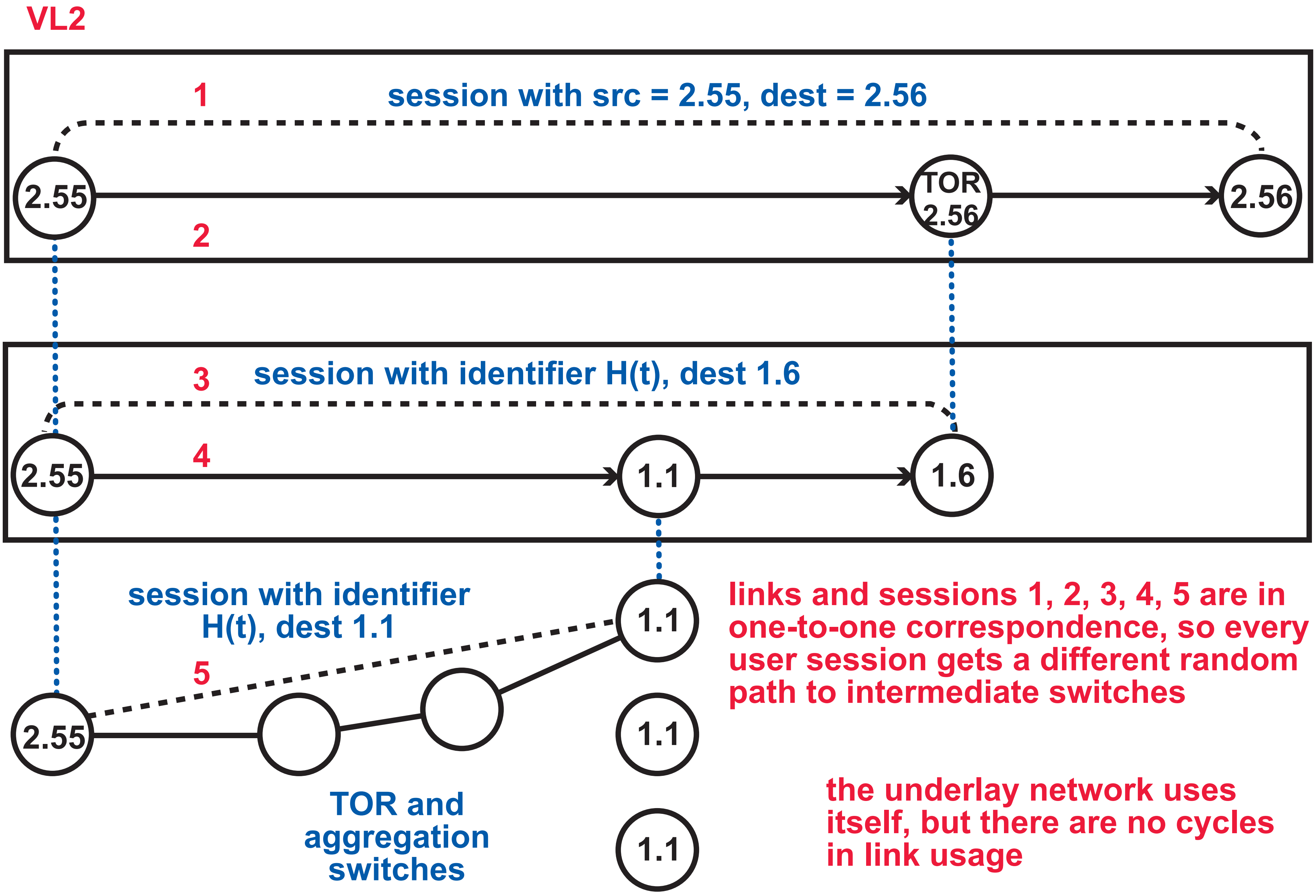
directory lookup

this member is a mysterious, undefined “server”—it is actually a VM share on a virtualized host, which gets its name dynamically by VM mobility

intermediate switch

aggregation switches are in here; presumably routing is deterministic and not interesting

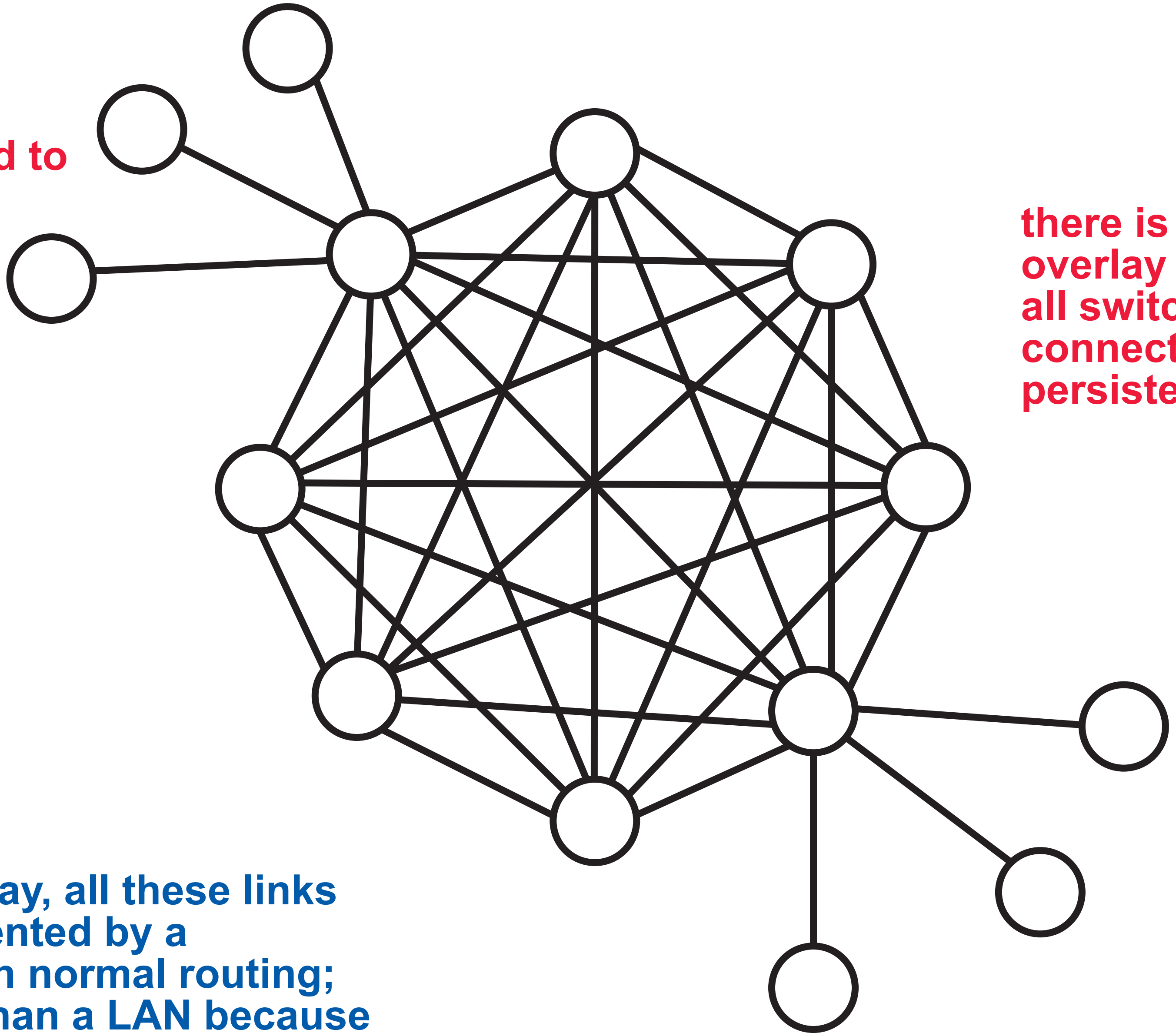
VL2 ARCHITECTURE



links and sessions 1, 2, 3, 4, 5 are in one-to-one correspondence, so every user session gets a different random path to intermediate switches

the underlay network uses itself, but there are no cycles in link usage

SEATTLE ARCHITECTURE



Ethernet
hosts are
connected to
switches

there is an
overlay in which
all switch pairs are
connected by direct,
persistent links

in an underlay, all these links
are implemented by a
network with normal routing;
it is better than a LAN because
links need not be confined to
a spanning tree

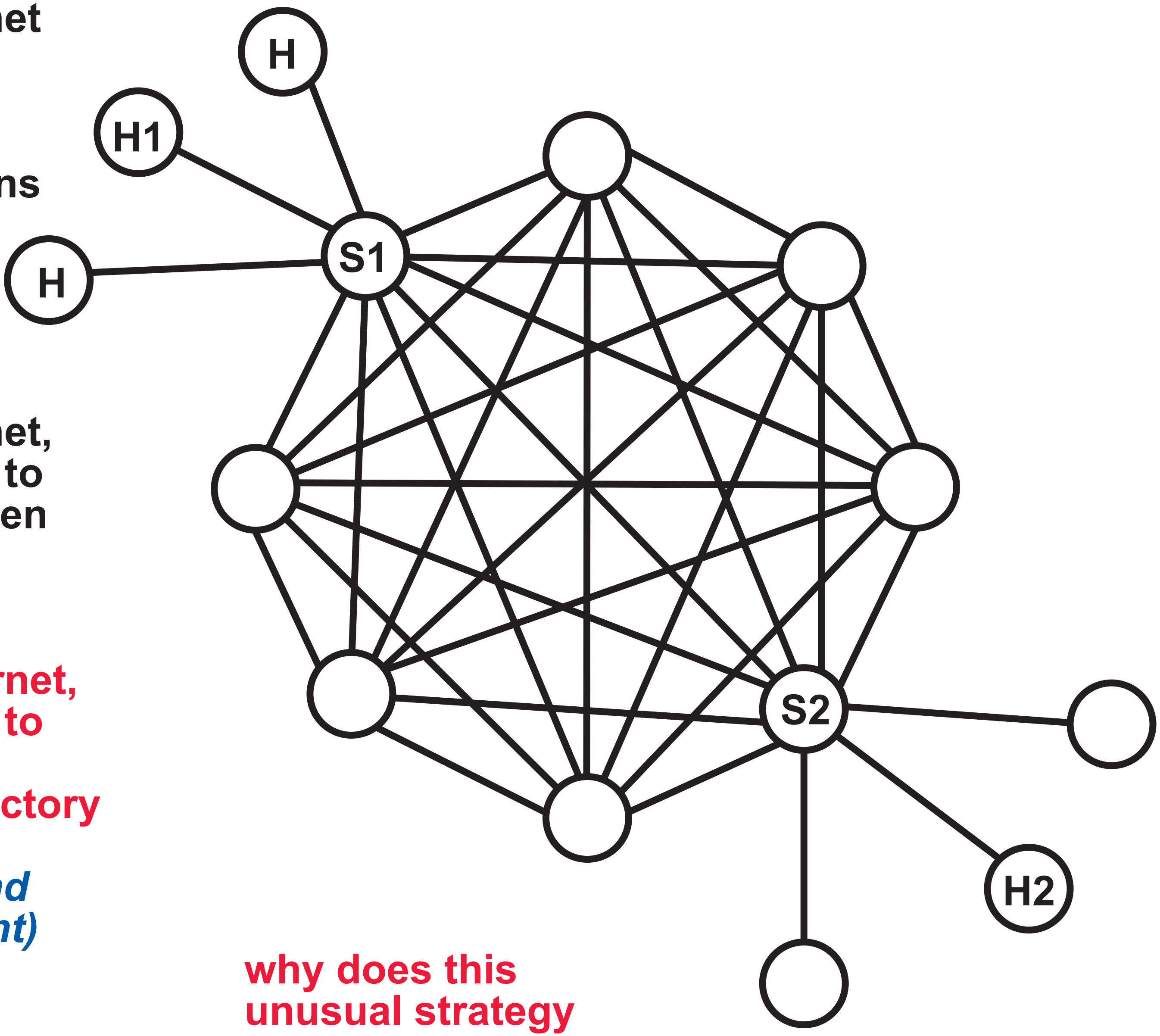
SEATTLE ARCHITECTURE

as in a normal Ethernet each switch has a sparse routing table, containing only entries for destinations it is currently communicating with

as in a normal Ethernet, a switch gets a route to a new destination when it needs one

unlike a normal Ethernet, a switch gets a route to a new destination by looking it up in a directory

(it cannot flood, and this is more efficient)



why does this unusual strategy work for this architecture?

SEATTLE ARCHITECTURE

unlike a normal Ethernet, a switch gets a route to a new destination by looking it up in a directory

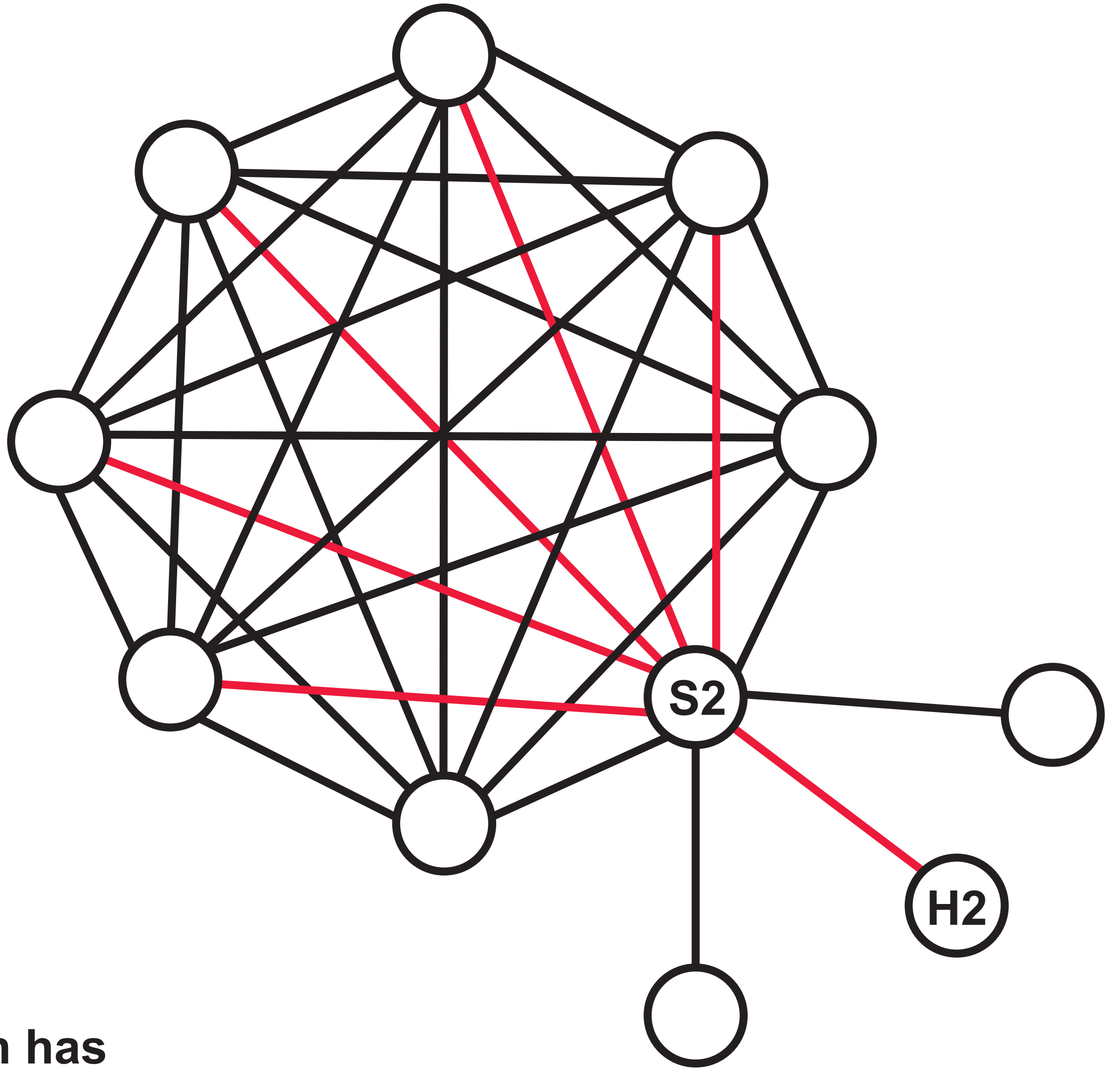
why does this unusual strategy work for this architecture?

1 normally, a network has many different routes to a destination, used by different sources

2 in this case each outlink from a switch is identified with the MAC address of the switch at the other end

with this scheme, each switch has the same route to a particular switch, and also the same route to a host on it

for all switches, the route to H2 is S2



MIDDLEBOXES IN CLOUD COMPUTING

LOGICAL PROBLEMS OF SERVICE CHAINING

- routing loops
- large number of switch-level forwarding rules
- session affinity
- middleboxes that modify the 5-tuple used to identify packets
- middleboxes that classify packets

PROBLEMS OF DEPLOYMENT AND DYNAMIC RESOURCE ALLOCATION

- how is service chaining deployed in a cloud data center?
- what happens when load must be redistributed?
- what happens when a virtual machine migrates?

A CLOUD DESIGN

DESIGN GOALS

- accommodate clouds of the largest size

  10 data centers
  100 K hosts per data center,
  100 M virtual machines

- put in *all* the capabilities desirable in large-scale, multi-tenant clouds

NEW SOURCES AND COMPARISONS

- SIMPLE
- Stratos

SOME SOURCES

- CloudNaaS
  [Benson, Akella, Shaikh 11]

  tenant-specific address spaces,
  policy links

- VL2 *[Greenberg et al. 09]*

  identifier/locator split,
  IP routing in cloud layer

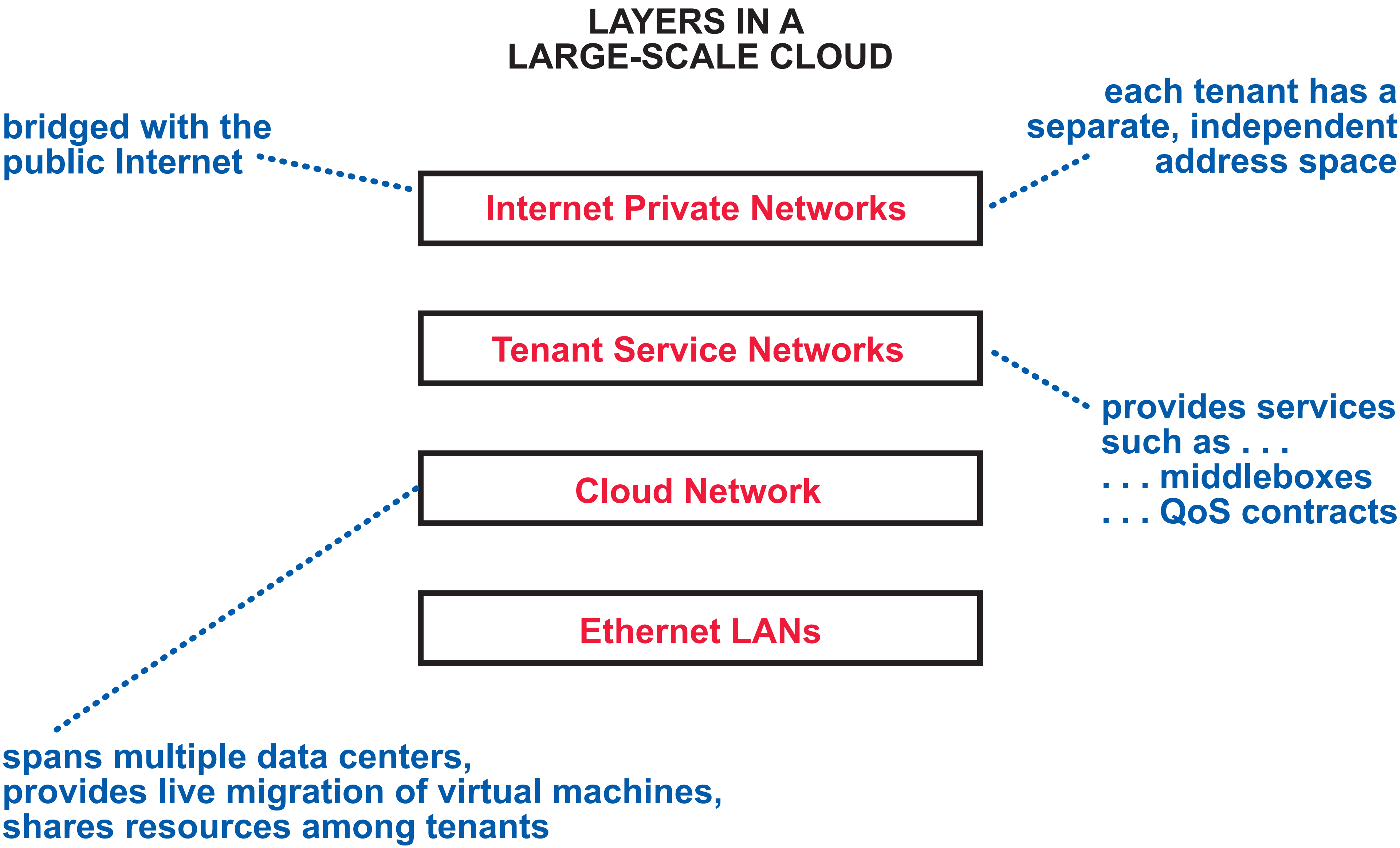
- WL2 *[Chen, Liu, Liu, Loo, Ding 14]*

  multiple data centers,
  VM migration

- OpenStack

  tenant-specific links

NETWORKS CONTRIBUTING TO THE CLOUD



LAYERS IN A
LARGE-SCALE CLOUD

Internet Private Networks

Tenant Service Networks

Cloud Network

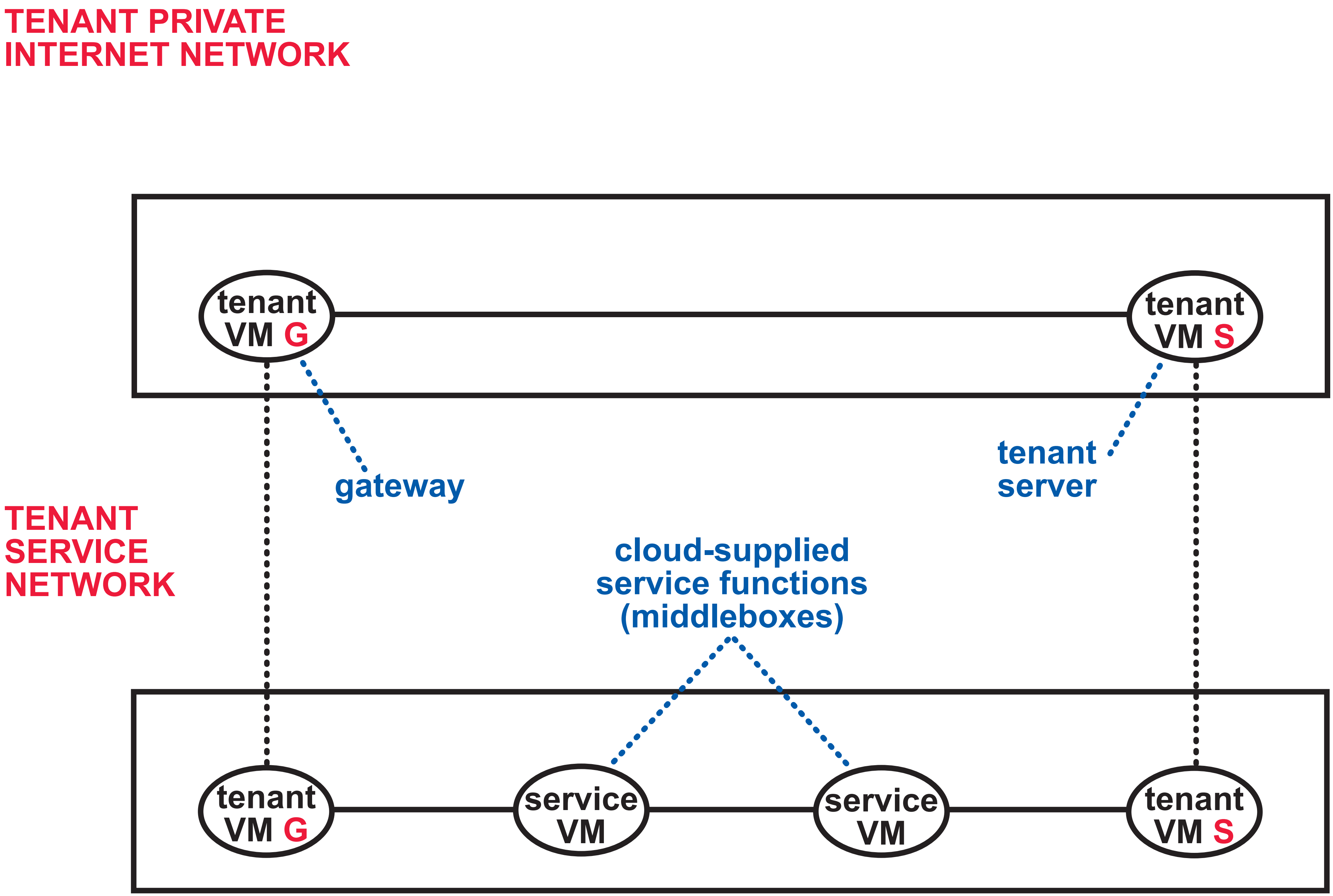
Ethernet LANs

bridged with the
public Internet

each tenant has a
separate, independent
address space

provides services
such as . . .
. . . middleboxes
. . . QoS contracts

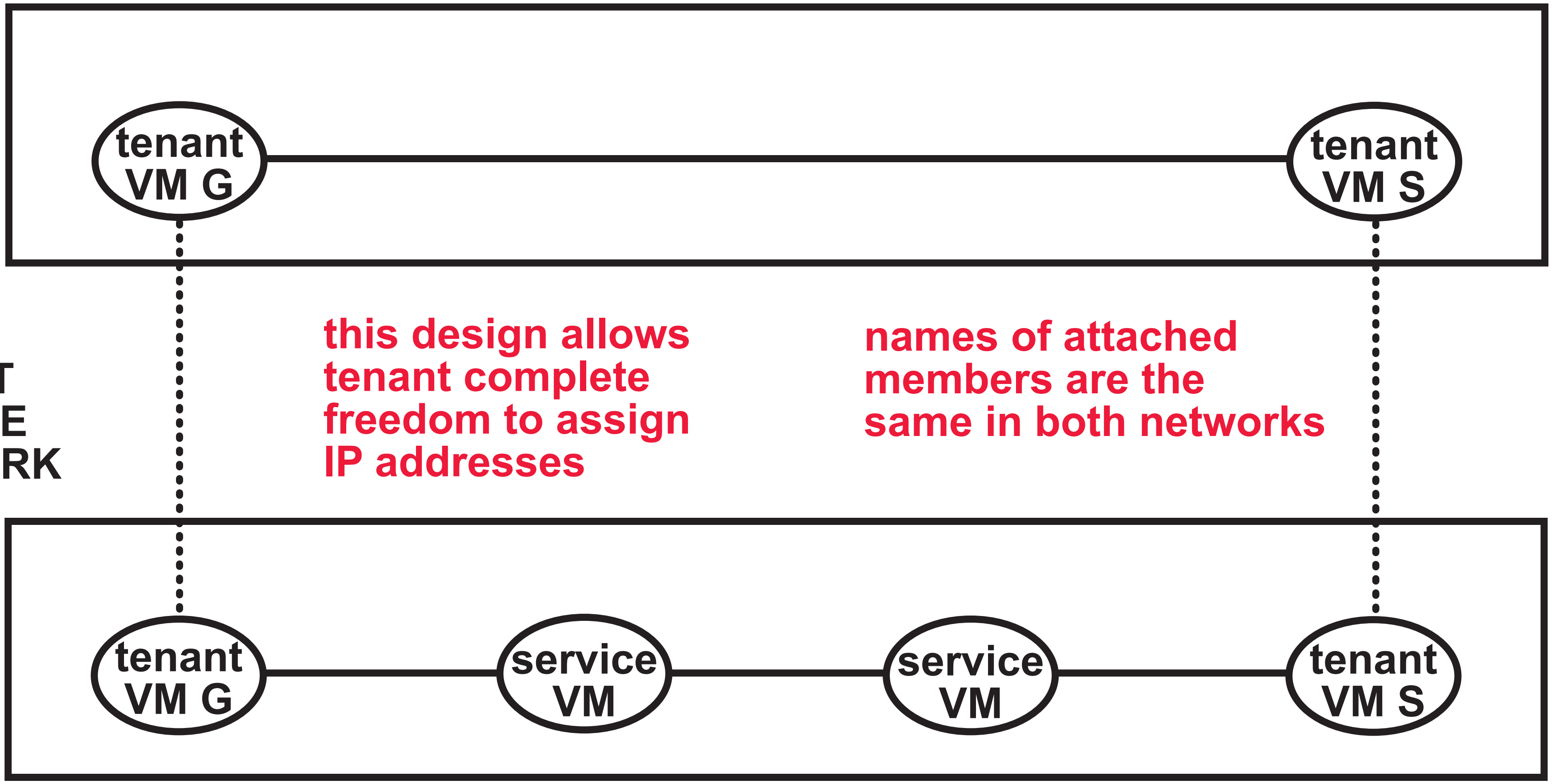
spans multiple data centers,
provides live migration of virtual machines,
shares resources among tenants


TENANT PRIVATE INTERNET NETWORK
tenant VM G
tenant VM S
gateway
tenant server
TENANT SERVICE NETWORK
cloud-supplied service functions (middleboxes)
tenant VM G
service VM
service VM
tenant VM S

TENANT PRIVATE INTERNET NETWORK

for each tenant, VL2 lumps the two networks together

VL2 paper does not say how tenant is provided with expected IP addresses



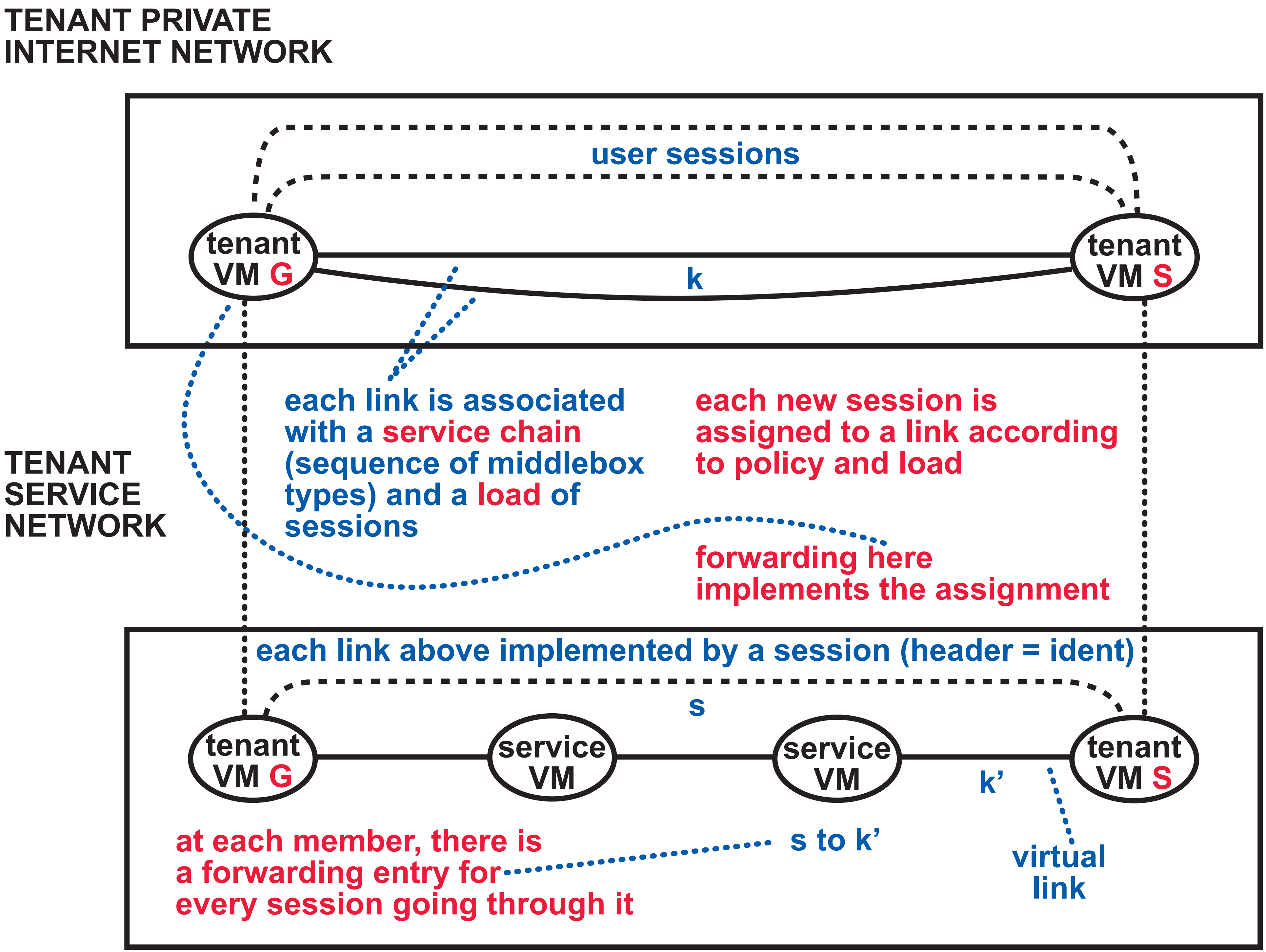
TENANT SERVICE NETWORK

this design allows tenant complete freedom to assign IP addresses

names of attached members are the same in both networks

Stratos lumps both networks, for all tenants, together

Stratos paper does not say how IP addresses are shared by tenants



each link is associated with a service chain (sequence of middlebox types) and a load of sessions

each new session is assigned to a link according to policy and load

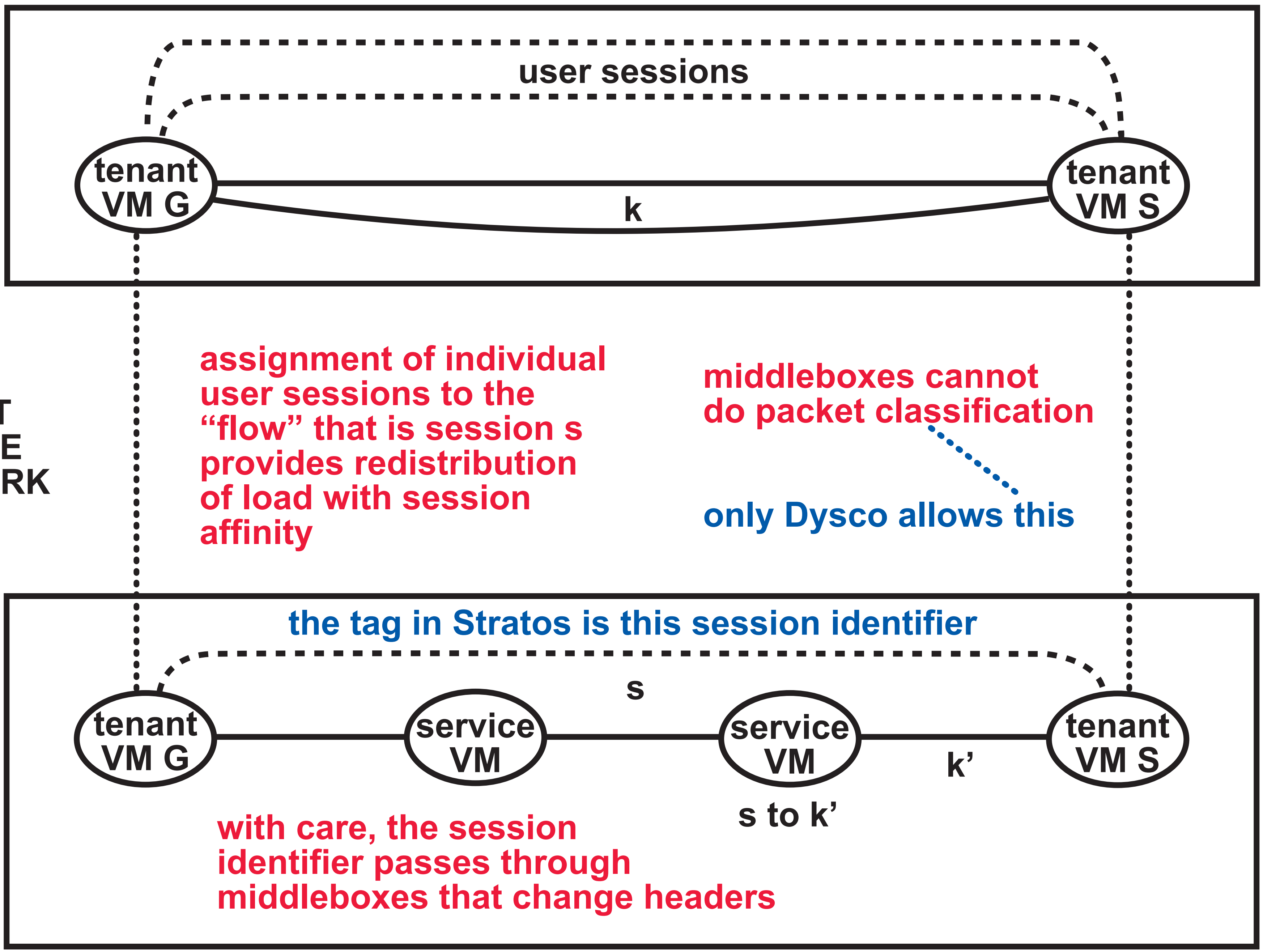
forwarding here implements the assignment

each link above implemented by a session (header = ident)

at each member, there is a forwarding entry for every session going through it

TENANT PRIVATE
INTERNET NETWORK

VL2 paper does not have
service chaining



user sessions

tenant
VM G

k

tenant
VM S

TENANT
SERVICE
NETWORK

assignment of individual
user sessions to the
“flow” that is session s
provides redistribution
of load with session
affinity

middleboxes cannot
do packet classification

only Dysco allows this

the tag in Stratos is this session identifier

s

tenant
VM G

service
VM

service
VM

k’

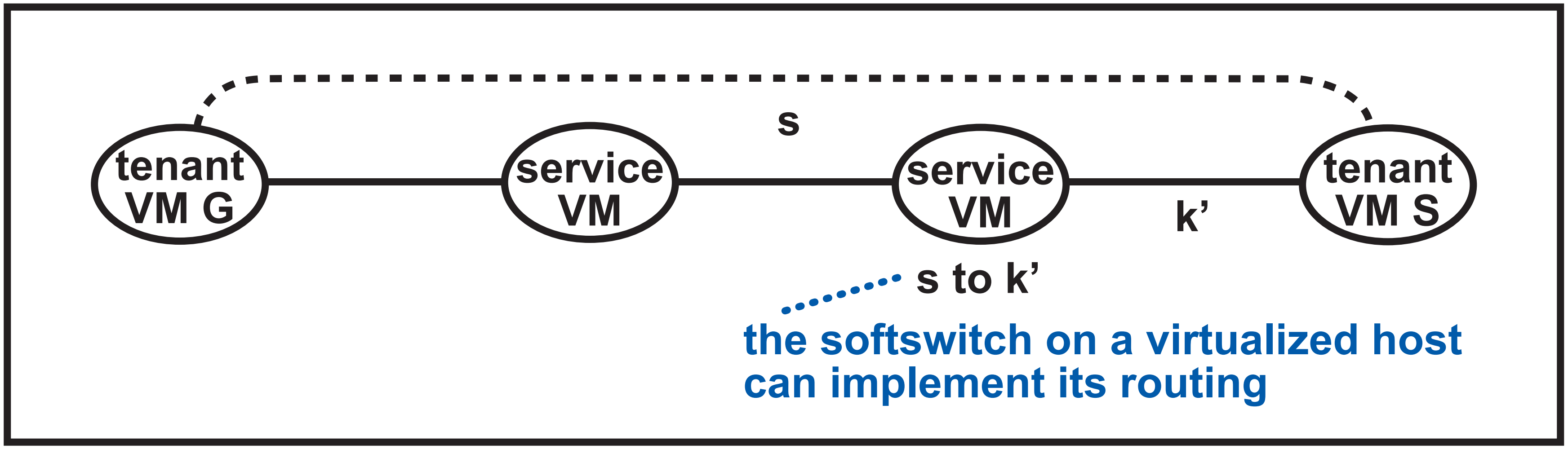
tenant
VM S

s to k’

with care, the session
identifier passes through
middleboxes that change headers

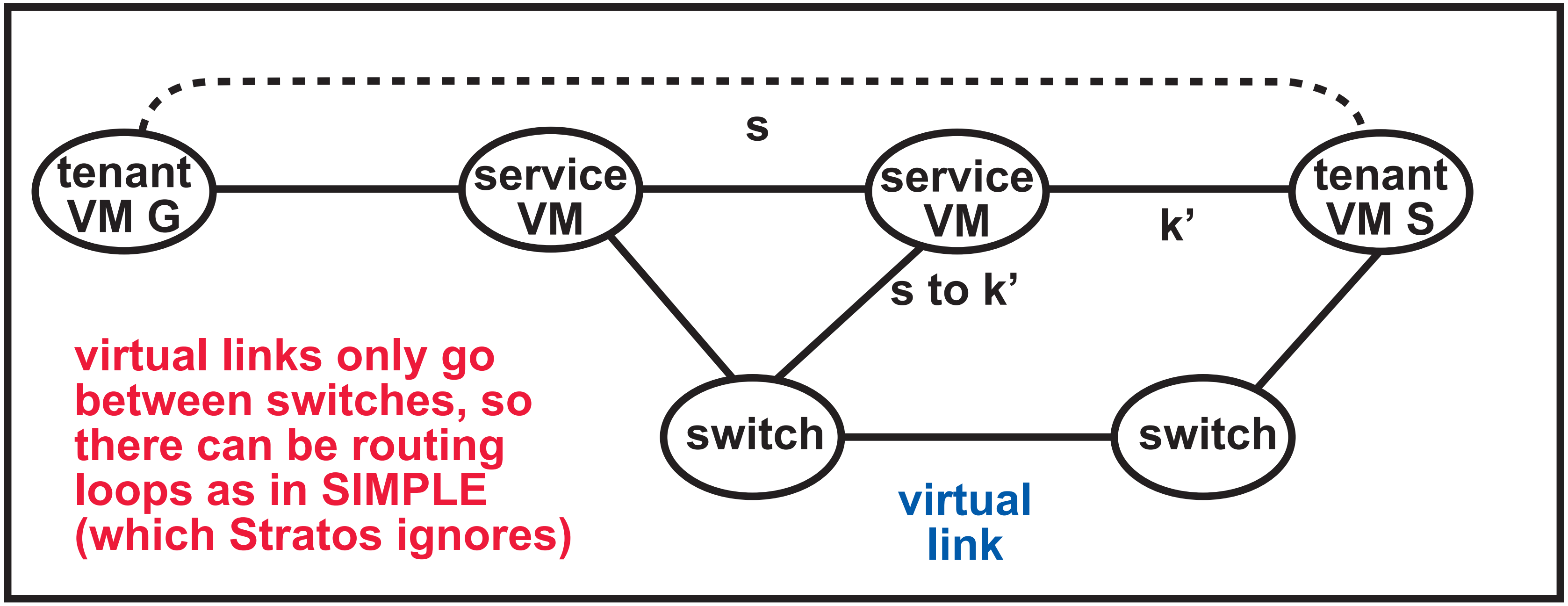
TENANT SERVICE NETWORK

in the cloud design, virtual links go between middleboxes, so there are no routing loops



the softswitch on a virtualized host can implement its routing

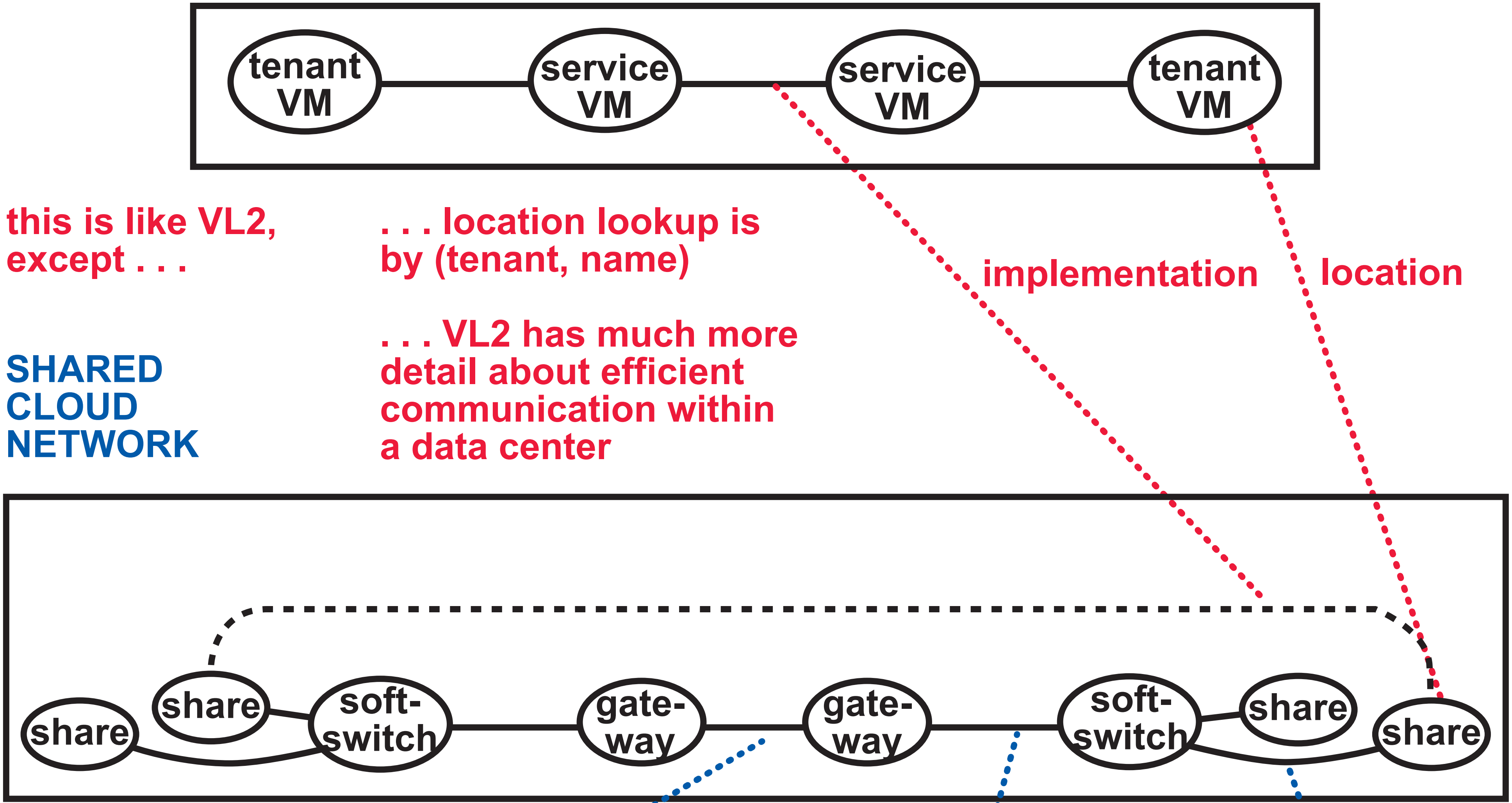
Stratos paper does not say what the switches are (soft-switches? TOR switches?)



virtual links only go between switches, so there can be routing loops as in SIMPLE (which Stratos ignores)

TENANT
SERVICE
NETWORK

Stratos has an underlay implementing virtual links between switches but it does not extend to middleboxes and does not provide for migration of VMs



this is like VL2, except . . .

. . . location lookup is by (tenant, name)

. . . VL2 has much more detail about efficient communication within a data center

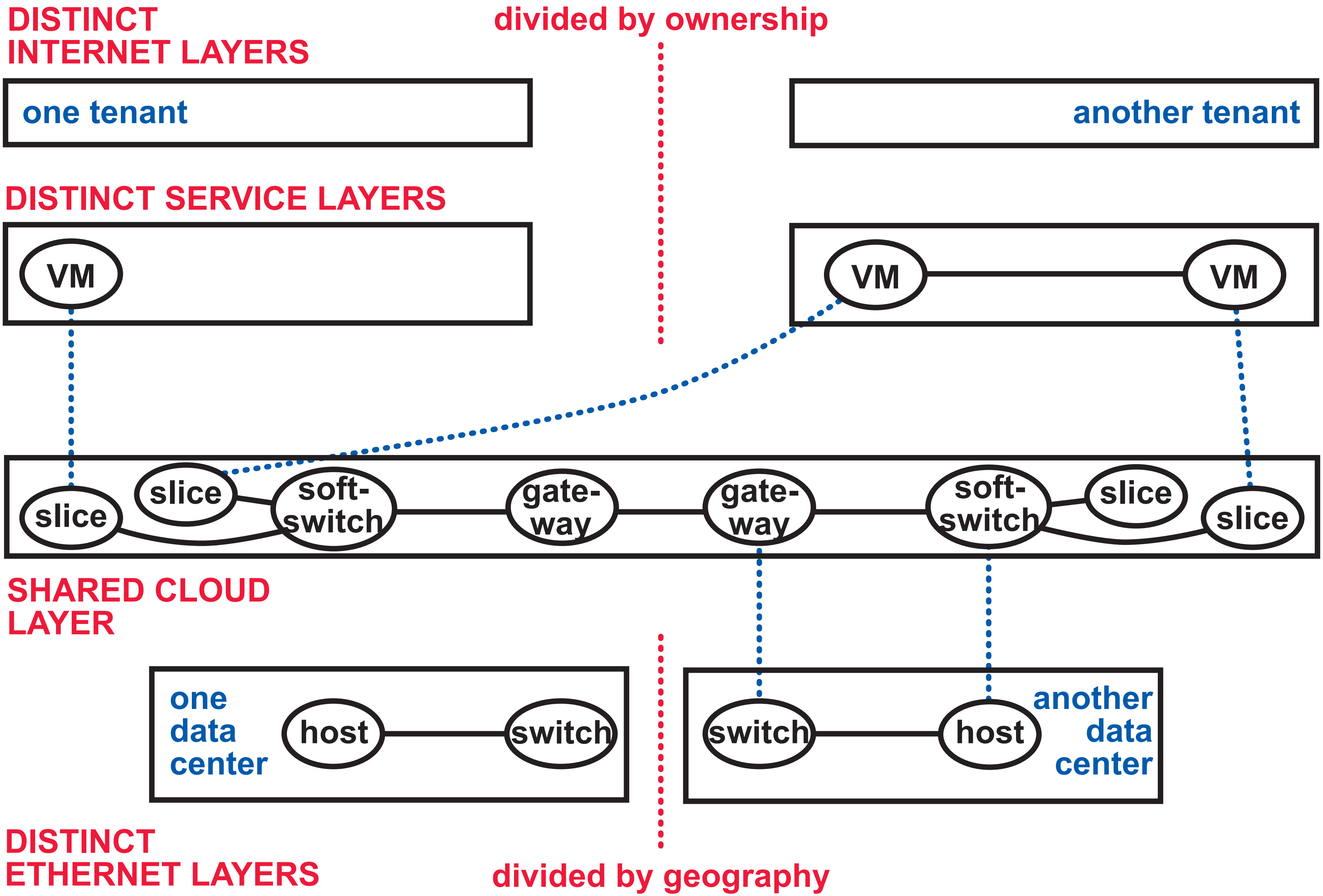
SHARED
CLOUD
NETWORK

trunk between data centers

link inside a data center (TORs and other switches can be here, too)

implemented by hypervisor of shared machine

VM MIGRATION: CLOUD LAYER HIERARCHY



DISTINCT INTERNET LAYERS

divided by ownership

one tenant

another tenant

DISTINCT SERVICE LAYERS

VM

VM

VM

slice

slice

soft-switch

gate-way

gate-way

soft-switch

slice

slice

SHARED CLOUD LAYER

one data center

host

switch

switch

host

another data center

DISTINCT ETHERNET LAYERS

divided by geography

VM MIGRATION: MOBILITY

this shows a link in the service layer, and the session in the cloud layer that implements it

BEFORE

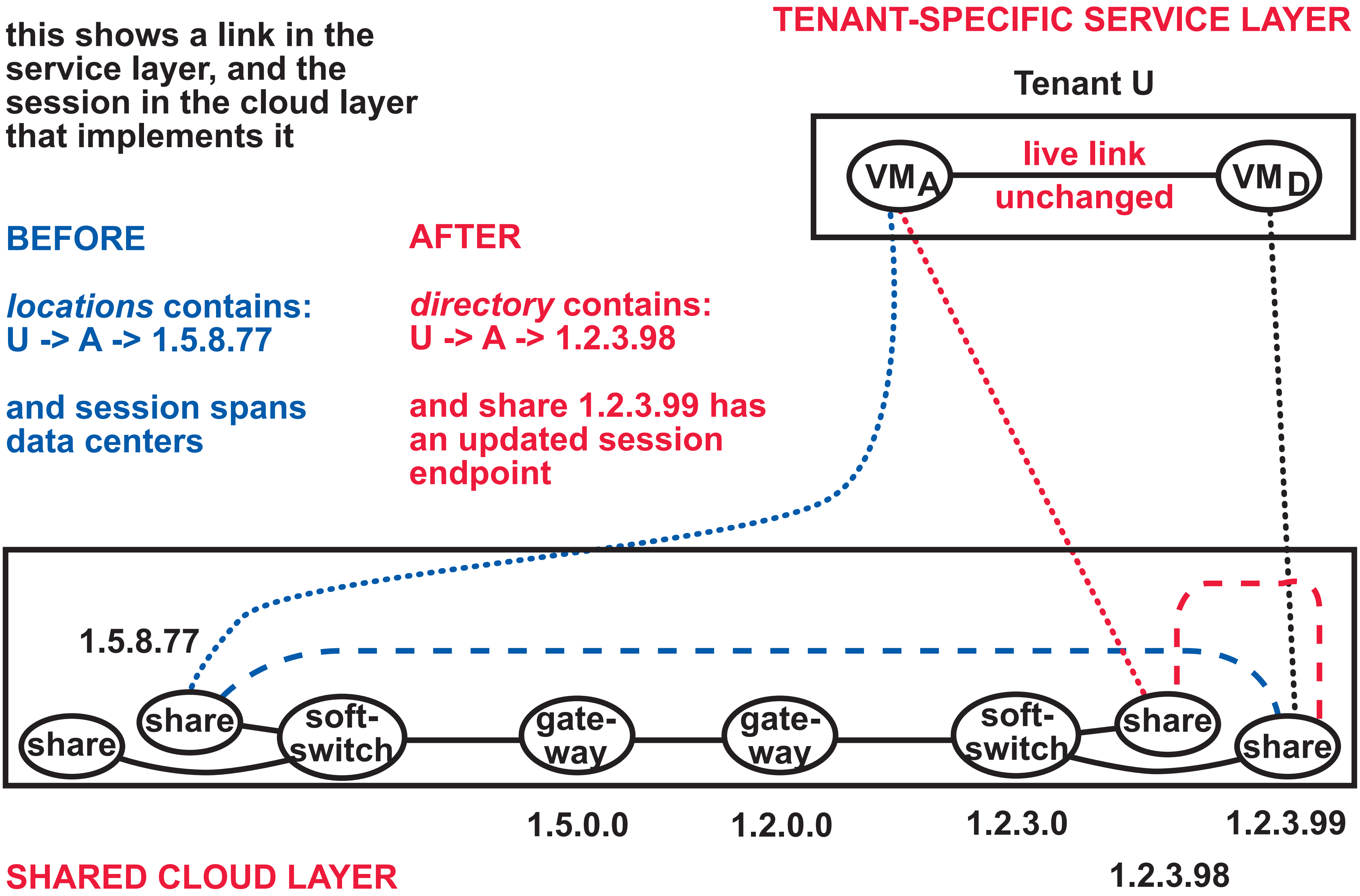
locations contains:
U -> A -> 1.5.8.77

and session spans data centers

AFTER

directory contains:
U -> A -> 1.2.3.98

and share 1.2.3.99 has an updated session endpoint

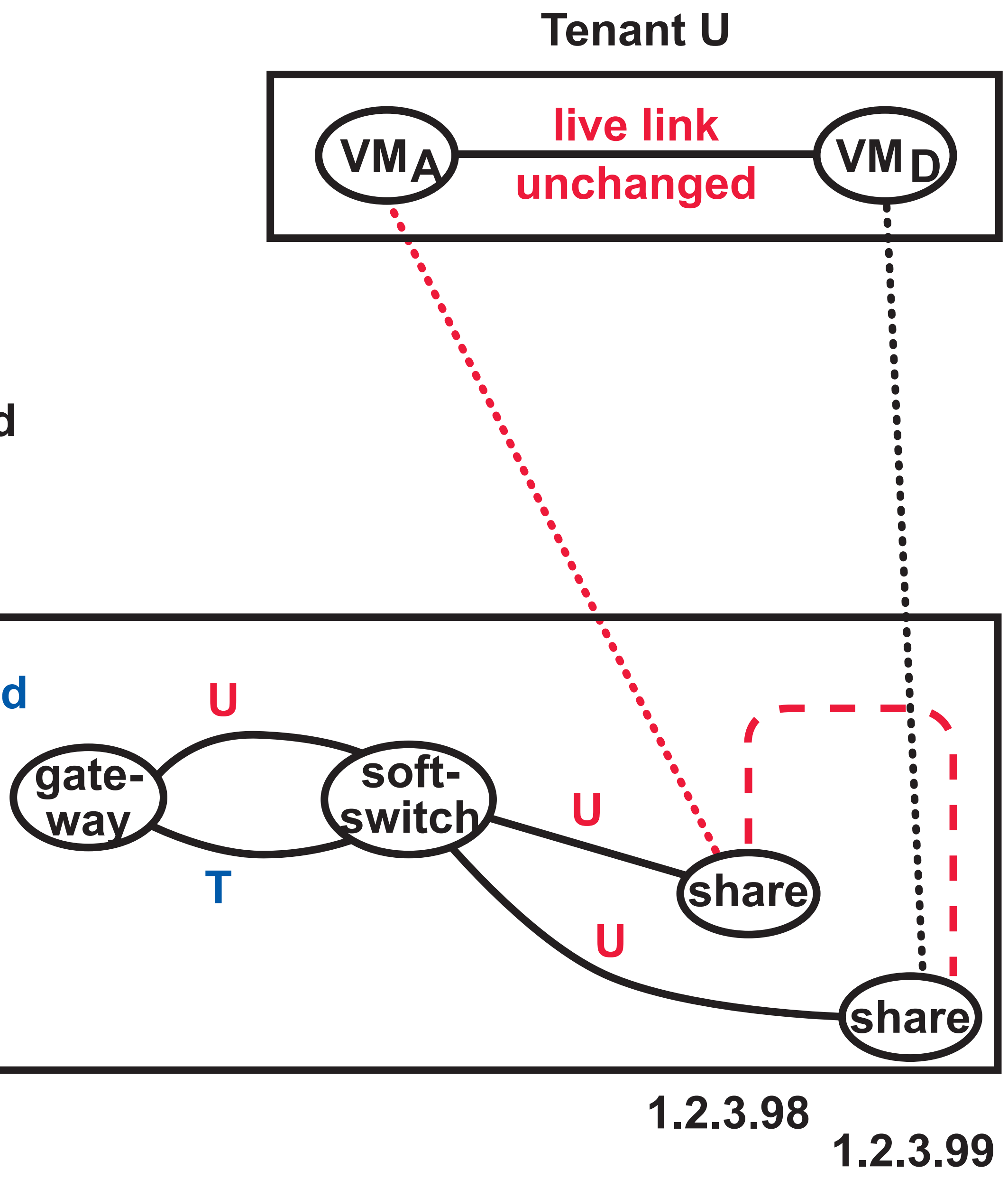


VM MIGRATION: A THREAT TO TENANT ISOLATION

we want to verify that a tenant’s VM can never receive messages from another tenant’s VM

enforcement is by means of tenant-specific links in the cloud layer (implemented on shared links in the Ethernet layers)

if it is proved that forwarding is limited to chains of links of the same tenant, tenant isolation should be guaranteed by this layer



before share 1.2.3.98 was allocated to Tenant U, it belonged to Tenant T, which is still sending messages to it

when the share changes tenants, its link must also change ownership, atomically

MIDDLEBOX POLICIES: UPDATES ARE CONSISTENT BY CONSTRUCTION

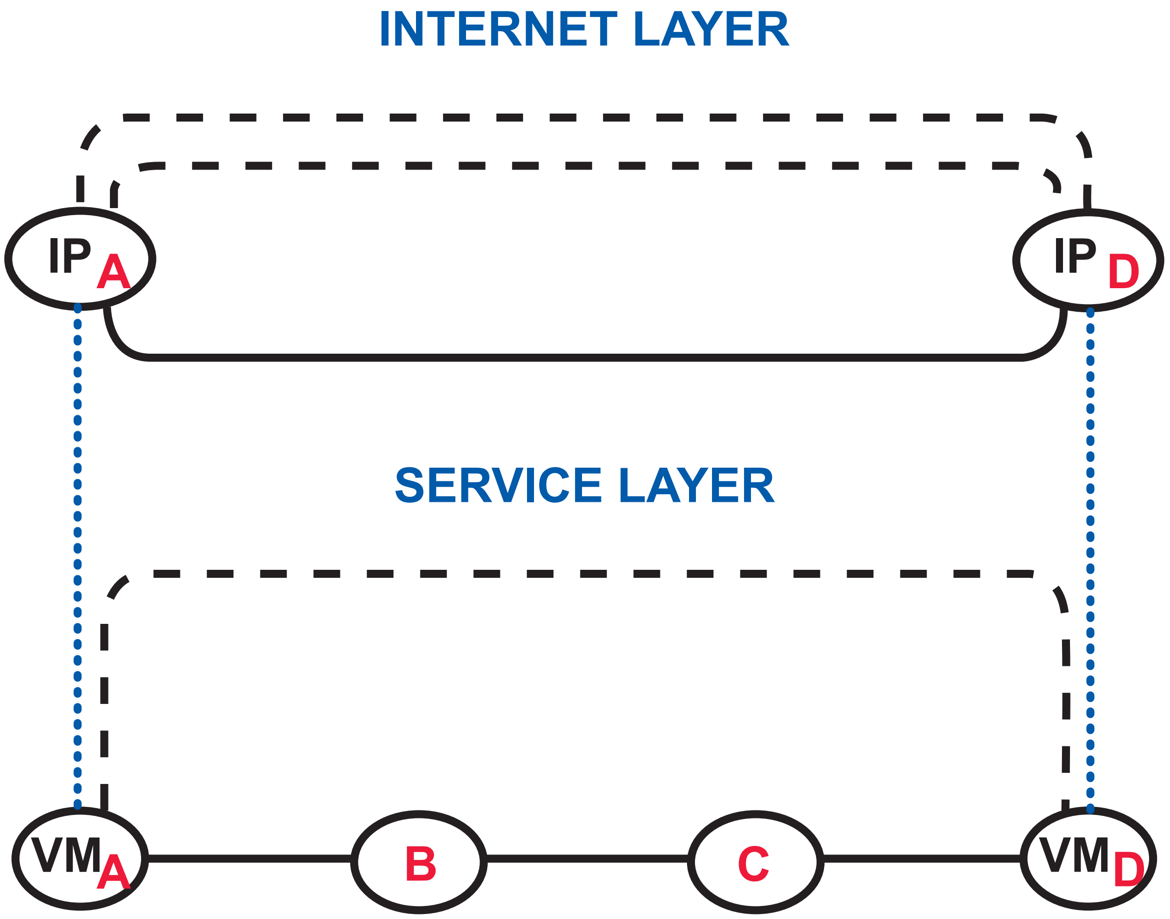
LAYERED CONTROL PROGRAMS

detect need for more capacity

create new policy link

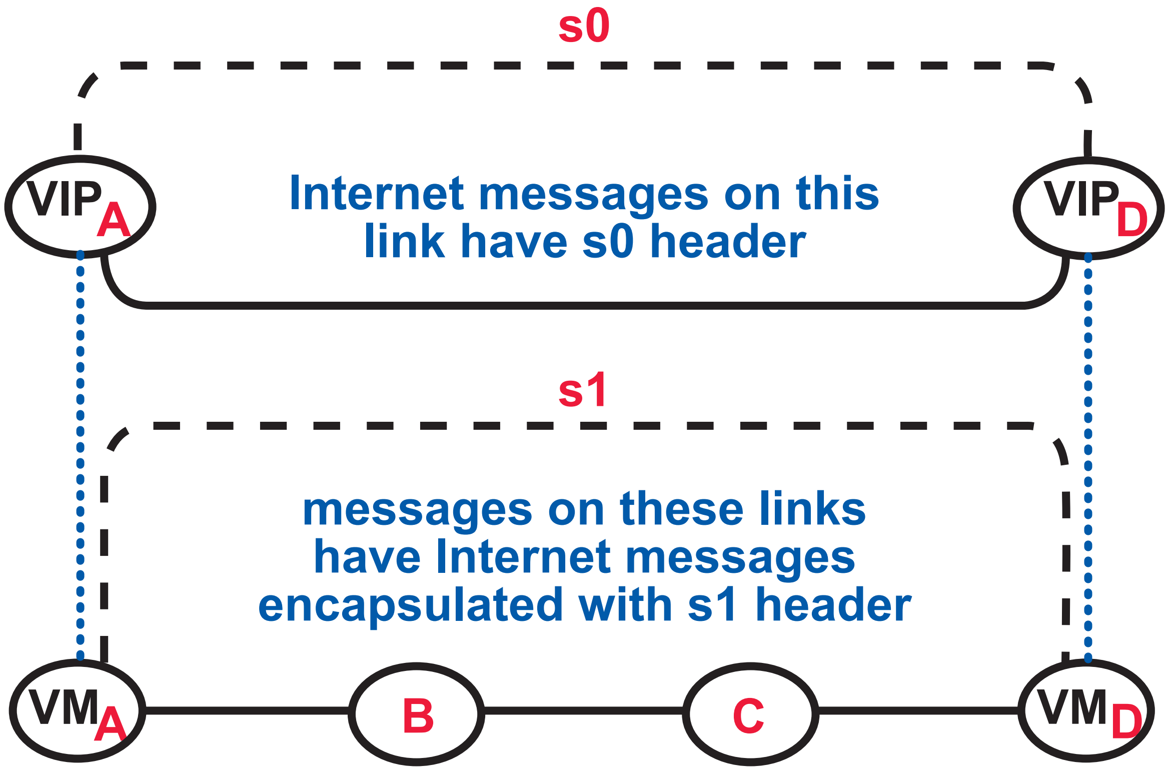
create new service session

allocate middleboxes, create links and forwarding for session



application sessions are not allocated to new policy link until this call returns

HEADER OPTIMIZATION



Internet messages on this link have s0 header

messages on these links have Internet messages encapsulated with s1 header

and so on for levels below

soundness of optimizations is easy to reason about in Alloy

if you optimize, you know what generality you are losing

HOWEVER, . . .

- if names or link/session identifiers coincide in two layered networks, they can be omitted from one of the headers

. . . and,
if sessions are set up by control plane (rather than by exchange of messages). . .

- if there is no more than one session between two endpoints, header can omit identifier
- if there is no more than one hop (link) in a session path, header can omit names

SUMMARY: REASONING WITH THE FORMAL MODEL

LOGICAL EFFECTIVENESS OR REACHABILITY

- legitimate destinations are reachable from legitimate sources

verified separately for each layer

- the mobility mechanism always succeeds in the cloud layer

even without central control, both endpoints moving simultaneously

SECURITY

- only allowed and authenticated messages are delivered

verified separately for each layer

- middlebox policies are enforced by the service layer
- one tenant's VM cannot receive messages from another tenant's VM in the cloud layer

UPDATE CONSISTENCY

- for propagation of top-down changes due to tenant configuration, policies, or load

consistency by construction, using informal hierarchical reasoning

- for propagation of bottom-up changes due to mobility, resource failure, or resource reconfiguration

verification and informal reasoning, both hierarchical

HEADER OPTIMIZATION (WHEN POSSIBLE TO OMIT FIELDS)

verified separately for each layer

BANDWIDTH TRACEABILITY (SUPPORT FOR QoS CONTRACTS)

load from each tenant is formally defined and traceable

NETWORK VIRTUALIZATION IN

MULTI-TENANT DATACENTERS

by Teemu Koponen and 24 others, mostly from VMware

NSDI `14

This is believed to be the ultimate cloud design, but no one understands the paper. Good time to try again.