Virtualized Congestion Control

Bryce Cronkite-Ratcliff, Aran Bergman, Shay Vargaftik,
Madhusudhan Ravi, Nick McKeown, Ittai Abraham, Isaac Keslassy

Stanford PLATFORMLAB

TECHNION Israel Institute of Technology

vmware RESEARCH

Data Center TCP (DCTCP)

Mohammad Alizadeh[‡†], Alb
Parveen Patel[†], Balaji Pr

[†]Micros
{albert, dmaltz, padhye
{ali

ABSTRACT

Cloud data centers host diverse applications, mixing v
require small predictable latency with others requir
tained throughput. In this environment, today's state-
protocol falls short. We present measurements of
production cluster and reveal impairments that lead to
tion latencies, rooted in TCP's demands on the limite
available in data center switches. For example, band
"background" flows build up queues at the switches.
pact the performance of latency sensitive "foreground

To address these problems, we propose DCTCP, a
tocol for data center networks. DCTCP leverages Ex
tion Notification (ECN) in the network to provide
back to the end hosts. We evaluate DCTCP at 1 and 1
using commodity, shallow buffered switches. We fin
livers the same or better throughput than TCP, whi
less buffer space. Unlike TCP, DCTCP also provid
tolerance and low latency for short flows. In handli
derived from operational measurements, we found D
the applications to handle 10X the current backgroun
out impacting foreground traffic. Further, a 10X inc
ground traffic does not cause any timeouts, thus large
incast problems.

Categories and Subject Descriptors: C.2.2 [Computer-
Networks]: Network Protocols
General Terms: Measurement, Performance
Keywords: Data center network, ECN, TCP

1. INTRODUCTION

In recent years, data centers have transformed co
large scale consolidation of enterprise IT into data
and with the emergence of cloud computing service
Amazon, Microsoft and Google. A consistent them
ter design has been to build highly available, high
computing and storage infrastructure using low cos
components [16]. A corresponding trend has also en
center networks. In particular, low-cost switches are
top of the rack, providing up to 48 ports at 1Gbps, a
under $2000 — roughly the price of one data cente

SIGCOMM'10, August 30–September 3, 2010, New Delhi,

pFabric: Minimal Near-Optimal Datacenter Transport

Mohammad Alizadeh[†‡]
Nick McKeown[†],

[†]Stanford University
{alizade, shyang, msharif, skatti, ni

ABSTRACT

In this paper we present pFabric, a minimalistic datace
port design that provides near theoretically optimal flo
tion times even at the 99th percentile for short flows,
minimizing average flow completion time for long flo
over, pFabric delivers this performance with a very sim
that is based on a key conceptual insight: datacenter trans
decouple flow scheduling from rate control. For flow s
packets carry a single priority number set independentl
flow; switches have very small buffers and implement a
ple priority-based scheduling/dropping mechanism. Rat
also correspondingly simpler; flows start at line rate a
back only under high and persistent packet loss. We pr
oretical intuition and show via extensive simulations tha
bination of these two simple mechanisms is sufficient
near-optimal performance.

Categories and Subject Descriptors: C.2.1 [Computer-Com
Networks]: Network Architecture and Design
General Terms: Design, Performance
Keywords: Datacenter network, Packet transport, Flow schedu

1. INTRODUCTION

Datacenter workloads impose unique and stringent re
on the transport fabric. Interactive soft real-time work
as the ones seen in search, social networking, and retail
large number of small requests and responses across th
ter that are stitched together to perform a user-request
tation (e.g., delivering search results). These applicatio
low latency for each of the short request/response flows,
perceived performance is dictated by how quickly respo
(or a large fraction of) the requests are collected and deli
to the user. However in currently deployed TCP-bas
the latency for these short flows is poor — flow comple
(FCT) can be as high as tens of milliseconds while in th
flows could complete in 10-20 microseconds. The rea
these flows often get queued up behind bursts of packets
flows of co-existing workloads (such as backup, replic
mining, etc) which significantly increases their completi

SIGCOMM'13, August 12–16, 2013, Hong Kong, China.

TIMELY: RTT-based Congestion Control for the Datacenter

Radhika Mittal*(UC Berkeley), Vinh The Lam, Nandita Dukkipati, Emily Blem, Hassan Wassel,
Monia Ghobadi*(Microsoft), Amin Vahdat, Yaogong Wang, David Wetherall, David Zats

Google, Inc.

ABSTRACT

Datacenter transports aim to deliver low latency messaging together with high throughput. We show that simple packet delay, measured as round-trip times at hosts, is an effective congestion signal without the need for switch feedback. First, we show that advances in NIC hardware have made RTT measurement possible with microsecond accuracy, and that these RTTs are sufficient to estimate switch queueing. Then we describe how TIMELY can adjust transmission rates using RTT gradients to keep packet latency low while delivering high bandwidth. We implement our design in host software running over NICs with OS-bypass capabilities. We show using experiments with up to hundreds of machines on a Clos network topology that it provides excellent performance: turning on TIMELY for OS-bypass messaging over a fabric with PFC lowers 99 percentile tail latency by 9X while maintaining near line-rate throughput. Our system also outperforms DCTCP running in an optimized kernel, reducing tail latency by 13X. To the best of our knowledge, TIMELY is the first delay-based congestion control protocol for use in the datacenter, and it achieves its results despite having an order of magnitude fewer RTT signals (due to NIC offload) than earlier delay-based schemes such as Vegas.

CCS Concepts

•**Networks** → **Transport protocols;**

Keywords

datacenter transport; delay-based congestion control; OS-bypass; RDMA

*Work done while at Google

SIGCOMM '15 August 17-21, 2015, London, United Kingdom

ACM ISBN 978-1-4503-3542-3/15/08.
DOI: http://dx.doi.org/10.1145/2785956.2787510

1. INTRODUCTION

Datacenter networks run tightly-coupled computing tasks that must be responsive to users, e.g., thousands of back-end computers may exchange information to serve a user request, and all of the transfers must complete quickly enough to let the complete response to be satisfied within 100 ms [24]. To meet these requirements, datacenter transports must simultaneously deliver high bandwidth (≫Gbps) and utilization at low latency (≪msec), even though these aspects of performance are at odds. Consistently low latency matters because even a small fraction of late operations can cause a ripple effect that degrades application performance [21]. As a result, datacenter transports must strictly bound latency and packet loss.

Since traditional loss-based transports do not meet these strict requirements, new datacenter transports [10, 18, 30, 35, 37, 47], take advantage of network support to signal the onset of congestion (e.g., DCTCP [35] and its successors use ECN), introduce flow abstractions to minimize completion latency, cede scheduling to a central controller, and more. However, in this work we take a step back in search of a simpler, immediately deployable design.

The crux of our search is the congestion signal. An ideal signal would have several properties. It would be fine-grained and timely to quickly inform senders about the extent of congestion. It would be discriminative enough to work in complex environments with multiple traffic classes. And, it would be easy to deploy.

Surprisingly, we find that a well-known signal, properly adapted, can meet all of our goals: delay in the form of RTT measurements. RTT is a fine-grained measure of congestion that comes with every acknowledgment. It effectively supports multiple traffic classes by providing an inflated measure for lower-priority transfers that wait behind higher-priority ones. Further, it requires no support from network switches.

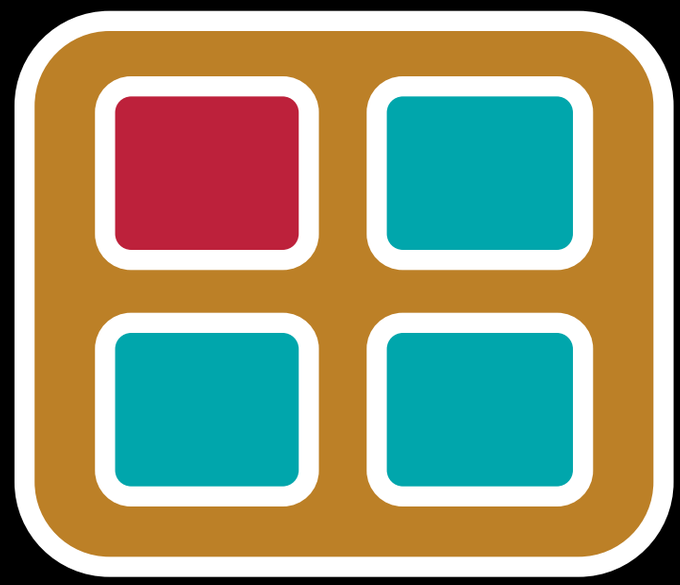
Delay has been explored in the wide-area Internet since at least TCP Vegas [16], and some modern TCP variants use delay estimates [44, 46]. But this use of delay has not been without problems. Delay-based schemes tend to compete poorly with more aggressive, loss-based schemes, and delay

Multitenant Datacenters

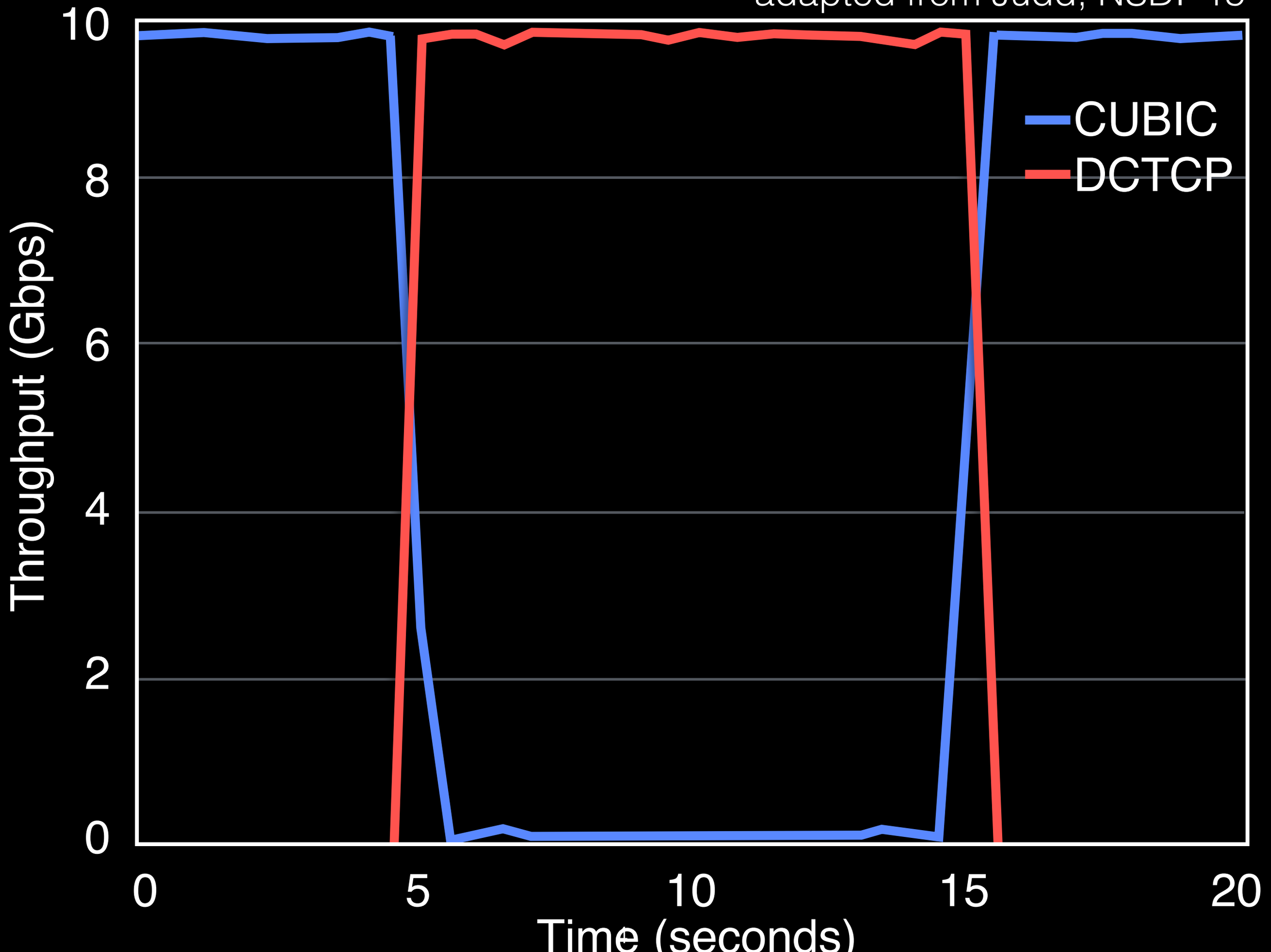
Tenants choose Congestion Control

Enterprise Datacenters

Can’t upgrade legacy applications



adapted from Judd, NSDI '15



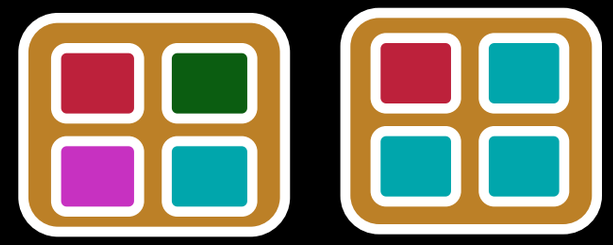
New Congestion Control Algorithms

pFabric

Timely

DCTCP

Multitenant & Enterprise Datacenters



Tahoe

NewReno

CANIT

Illinois

Reno

Vegas

Cubic

MP-TCP

TFRC

Agile-SD

Zeta

Westwood+

YeAH

FIT

Veno

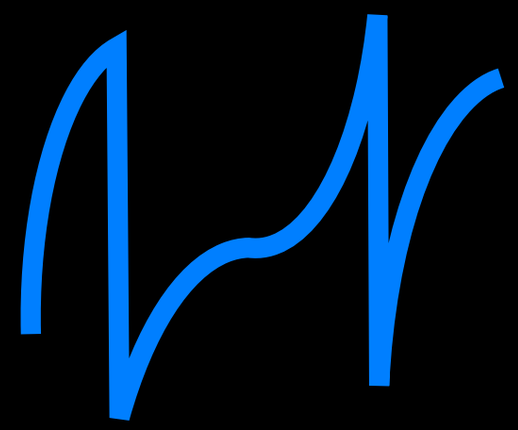
XCP

Compound

H-TCP

LP

HSTCP



Scalable

Hybla

FAST

Indy

BIC

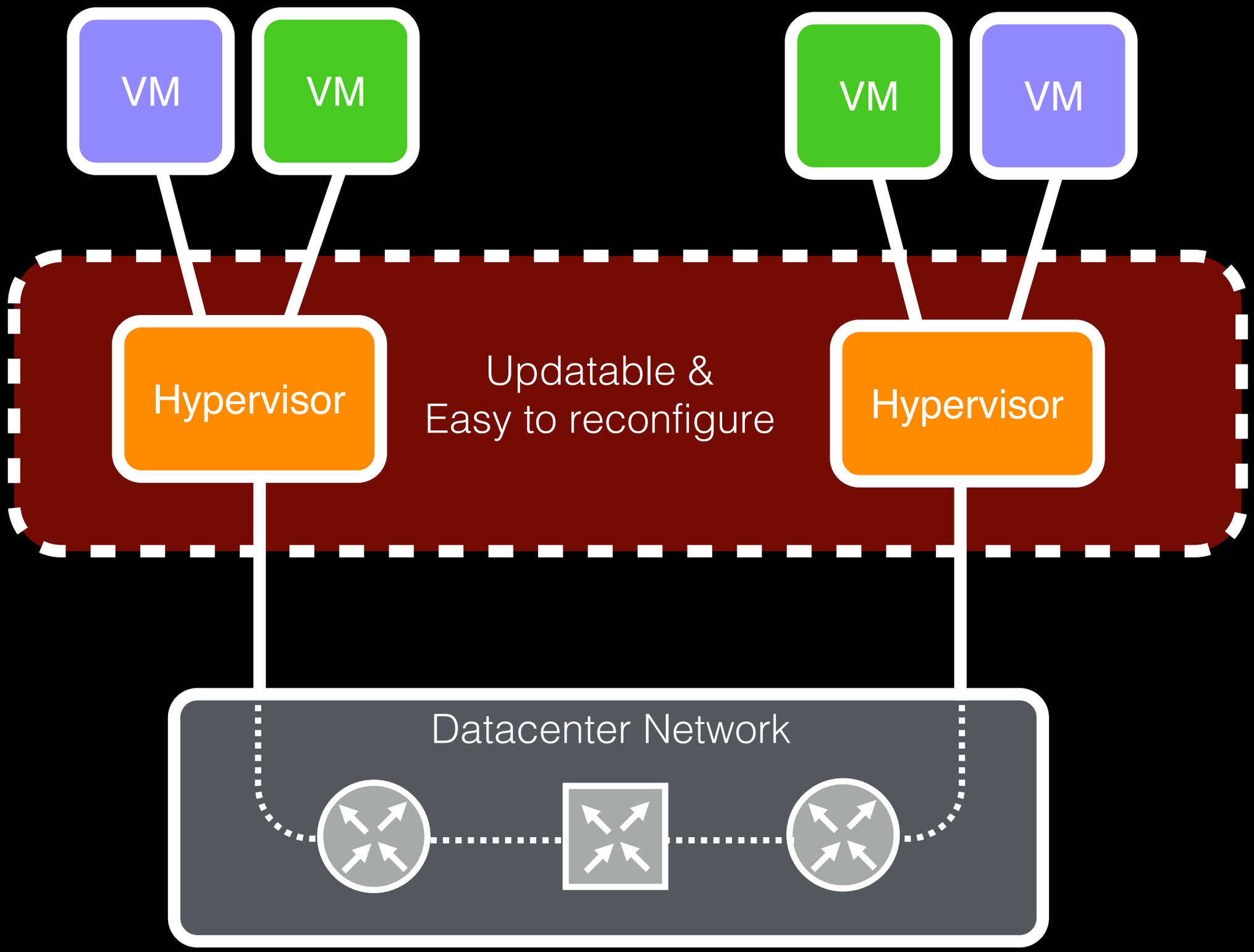
Jersey

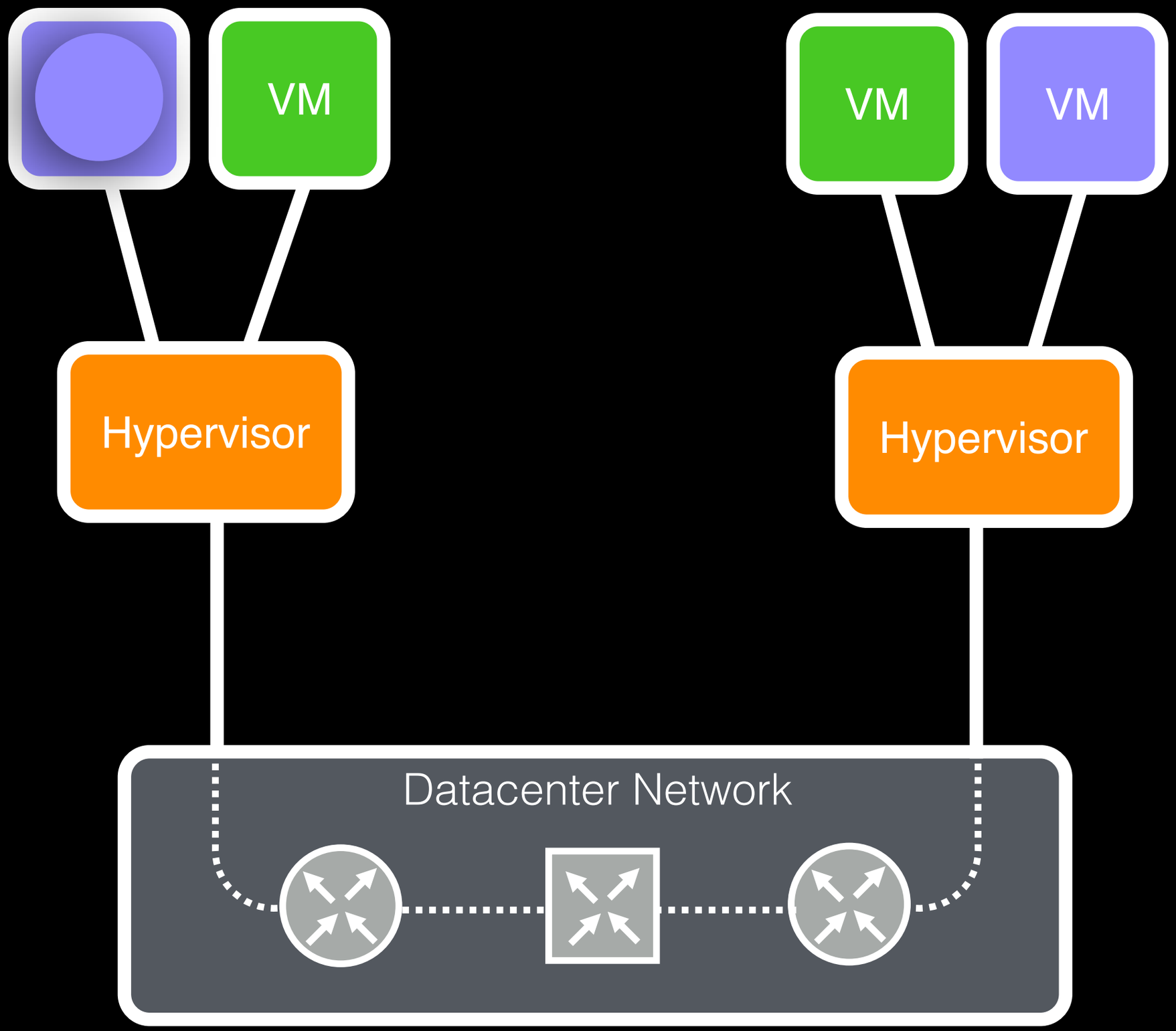
Real

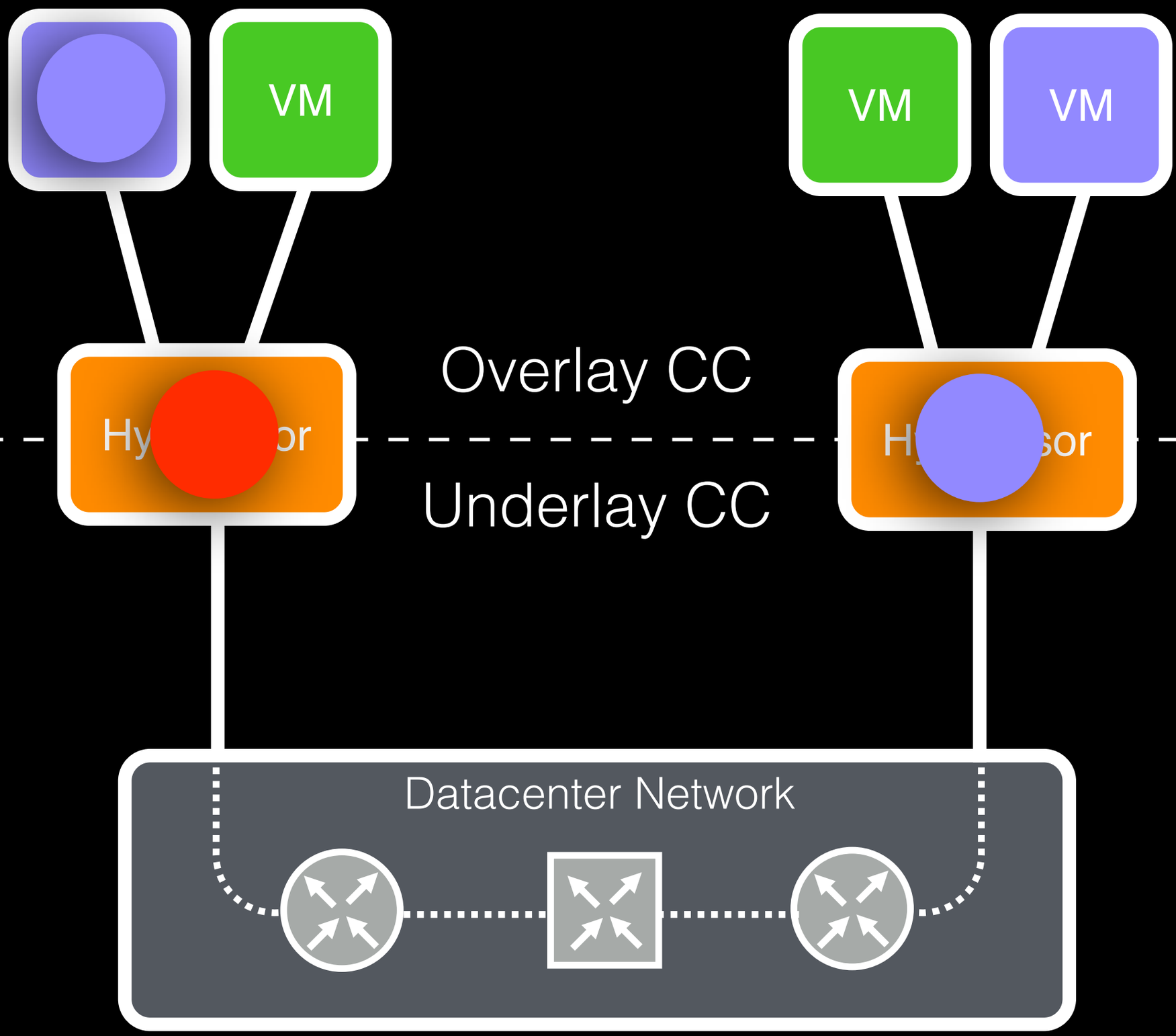
pFabric

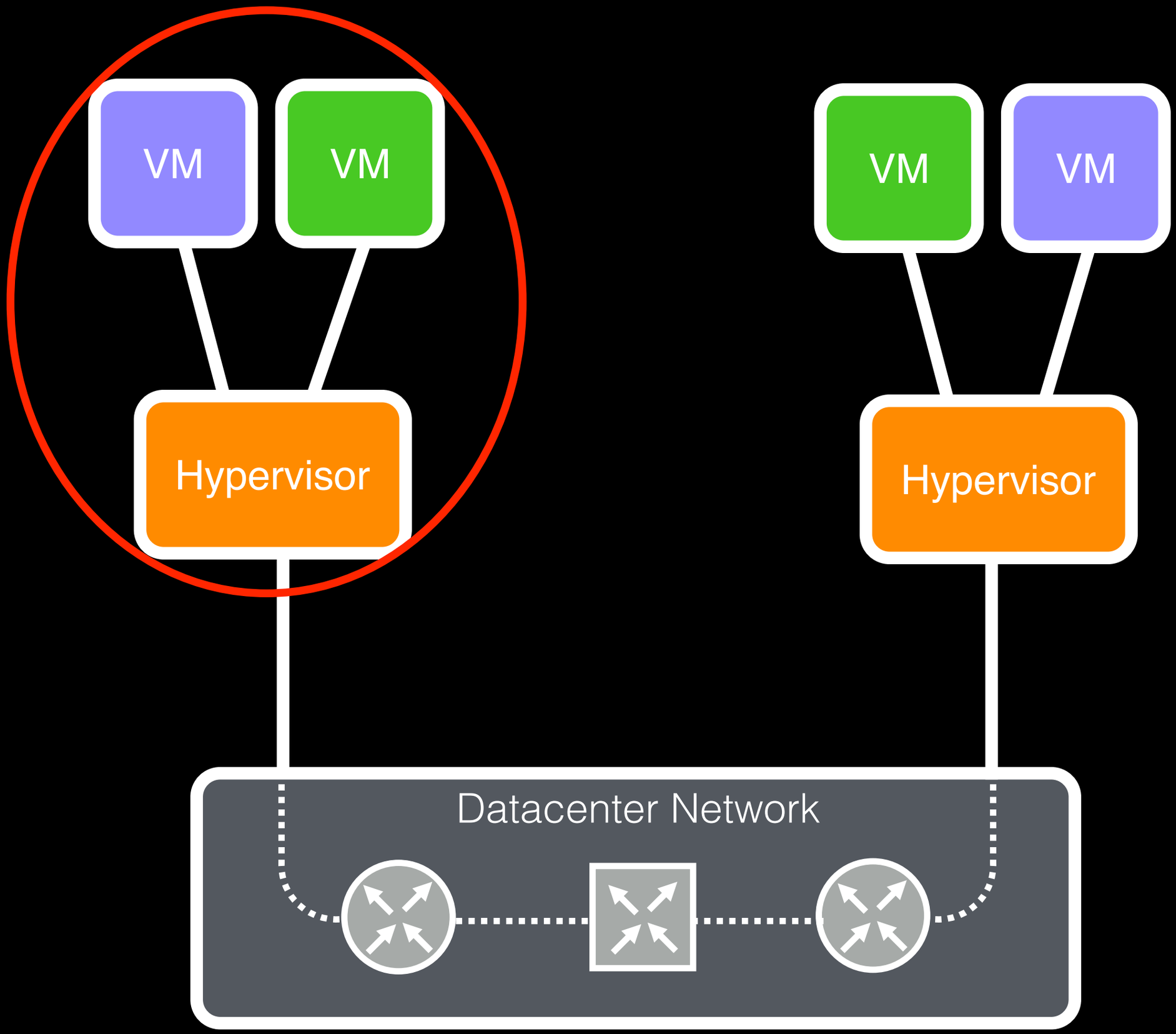
Timely

DCTCP

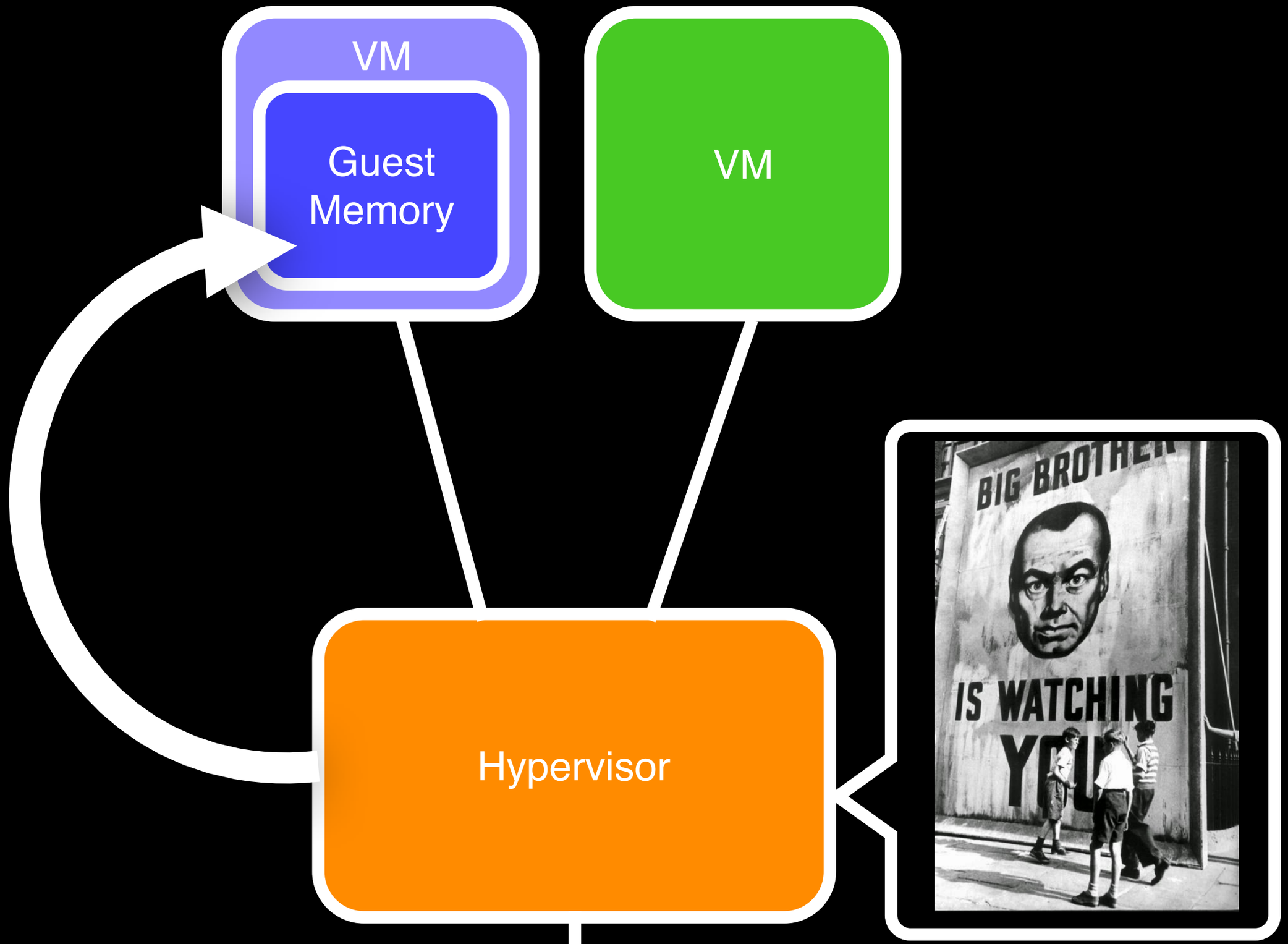

VM
VM
VM
VM
Hypervisor
Updatable &
Easy to reconfigure
Hypervisor
Datacenter Network


VM
VM
VM
Hypervisor
Hypervisor
Datacenter Network

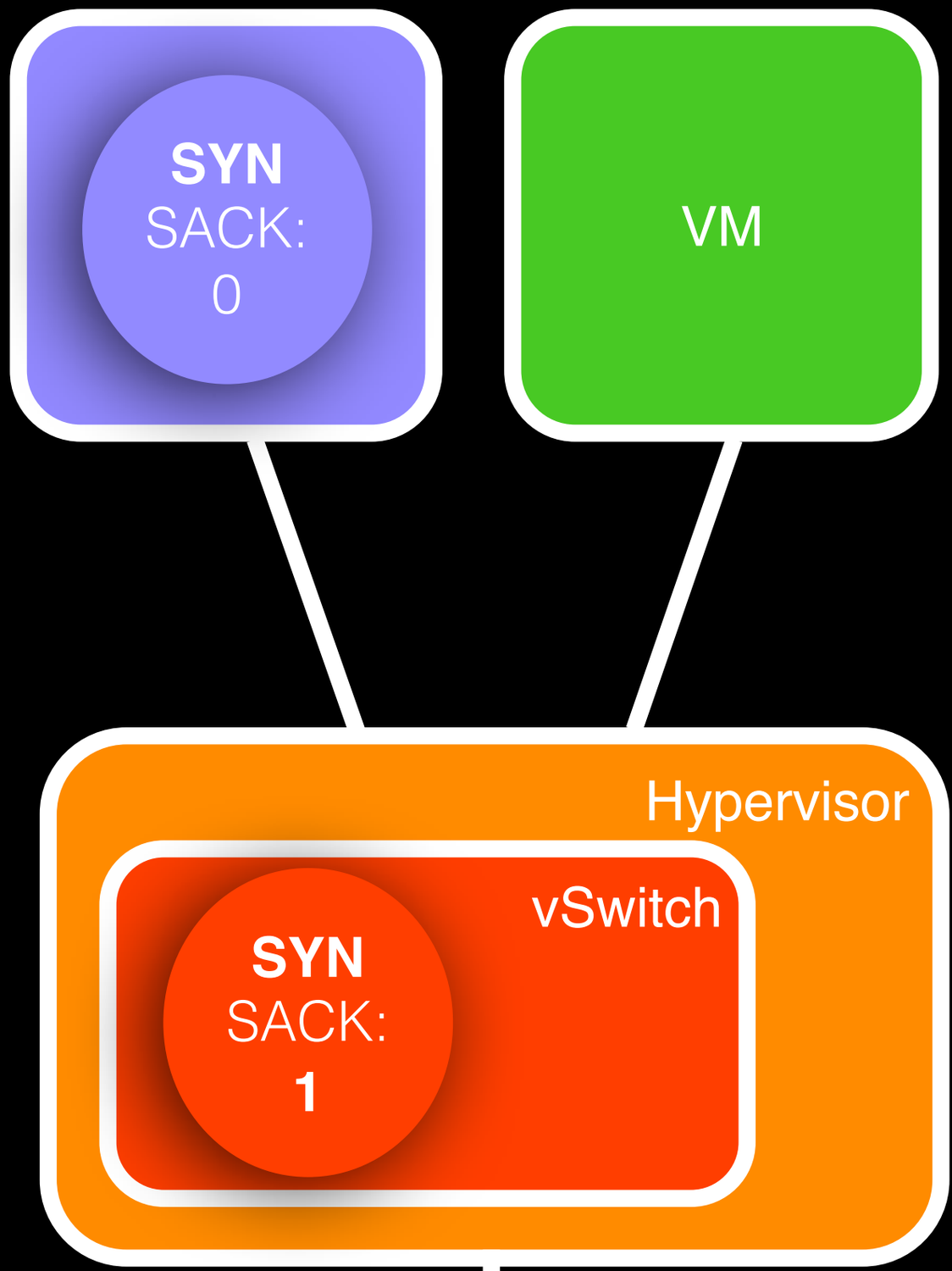

VM
VM
VM
Overlay CC
Underlay CC
Datacenter Network


VM
VM
Hypervisor
VM
VM
Hypervisor
Datacenter Network

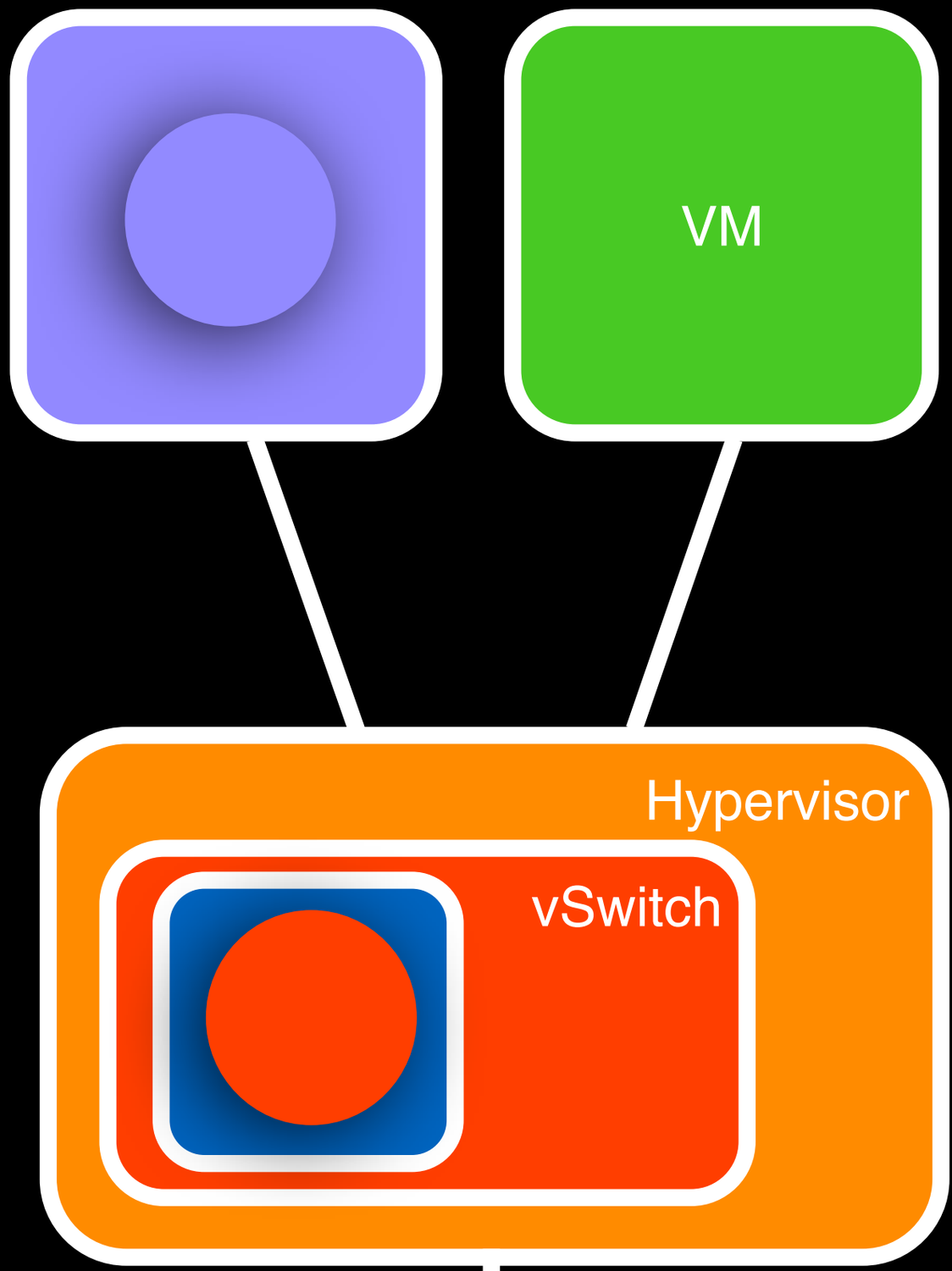
Guest Introspection

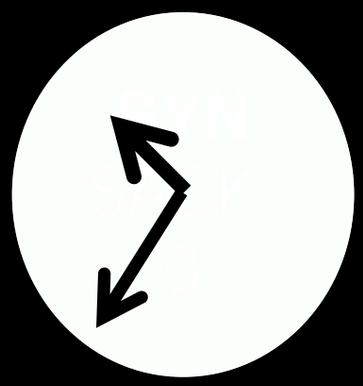


TCP Header Modification

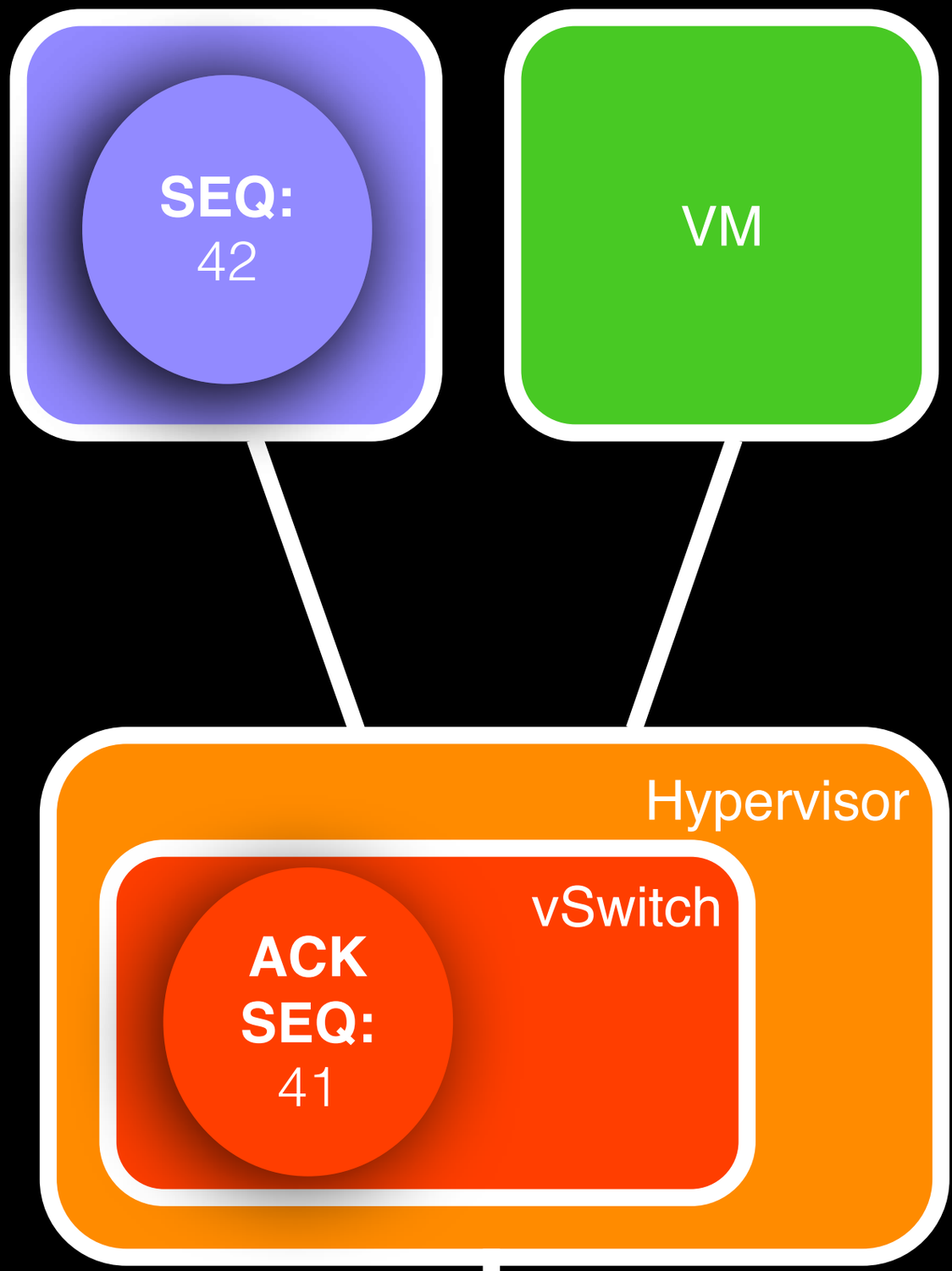


Buffering

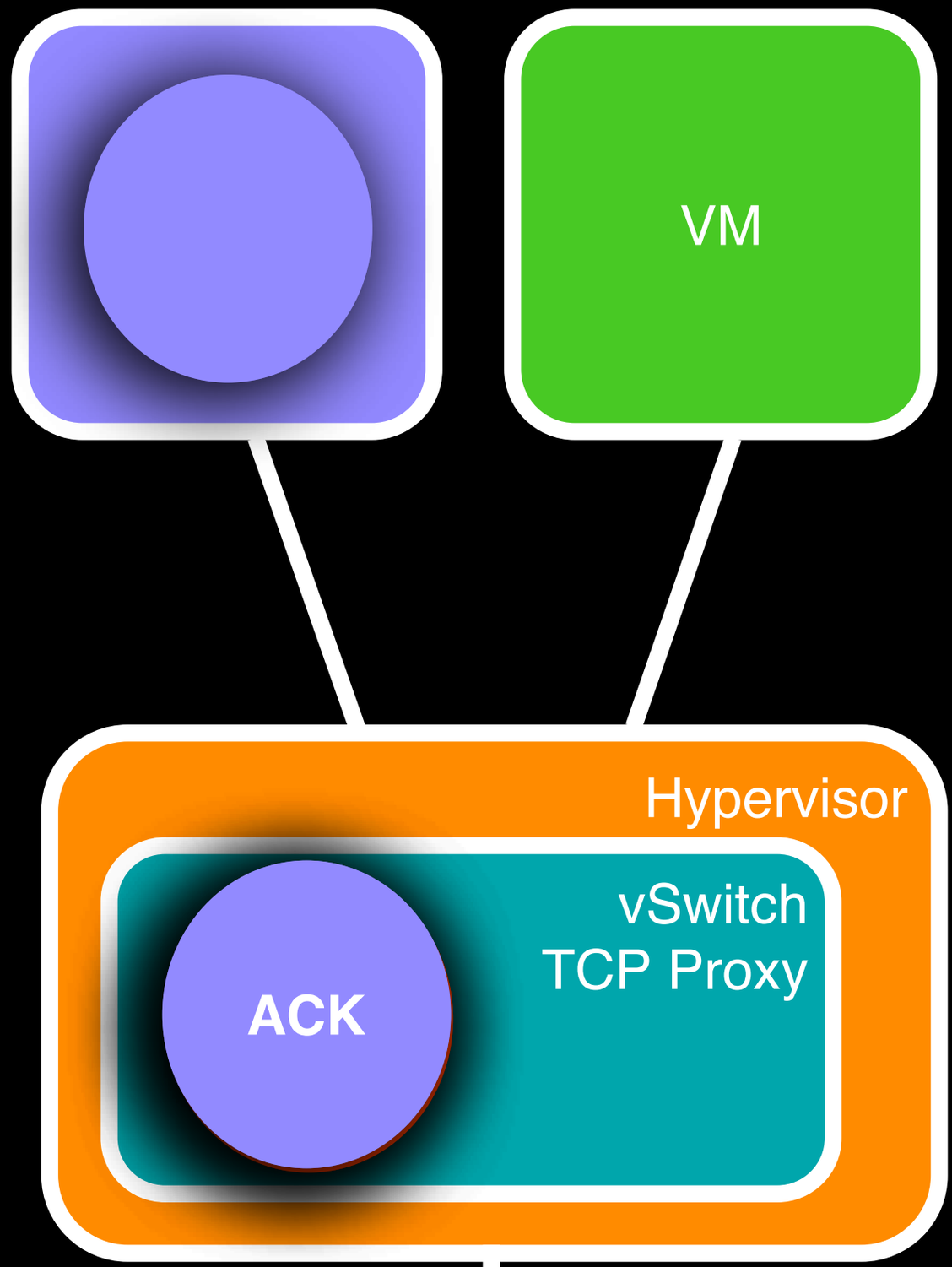




Fake ACKs



TCP Proxy



vCC Flow Modification Techniques

- Guest Introspection
- Buffering
- TCP Header Modification
- Fake ACK generation
- TCP Proxy

A Binary Feedback Sch
Avoidance in Computer

K. K. RAMAKRISHNAN and RAJ JAIN
Digital Equipment Corporation

We propose a scheme for *congestion avoidance* in
network layer. The scheme uses a minimal amoun
adjust the amount of traffic allowed into the netw
and set a *congestion-indication* bit on packets fl
indication is communicated back to the users th
scheme is distributed, adapts to the dynamic state
point, is quite simple to implement, and has low ov
provided to multiple sources. This paper presents
choice of the various decision mechanisms. We al
transient changes in the network and pathological

Categories and Subject Descriptors: C.2.1 **[Com**
Architecture and Design—*network communications*
forward networks; C.2.3 **[Computer Communica**
monitoring; C.4 **[Computer System Organizatio**

General Terms: Algorithms, Design, Performance

Additional Key Words and Phrases: Computer c
avoidance, congestion control, congestion indicatio
performance, network power

1. INTRODUCTION

Congestion in computer networks is a si
networks and increased link speeds. Fl
problems that have been addressed by s
the increasing range of speeds of link
distributed computing, effective control
important. The lack of control may resu
missions, may ultimately lead to *congesti*

The control mechanisms adopted to c
may be categorized into two distinct typ

An earlier version of this paper was presented at t
nications Architectures and Protocols," Stanford, C
Authors' address: Distributed Systems Architectur
tion, Littleton, MA 01460-1289.

TCP and Explicit Congestion Notification

Sally Floyd*

Lawrence Berkeley Laboratory
One Cyclotron Road, Berkeley, CA 94704
floyd@ee.lbl.gov

Abstract

This paper discusses the use of Explicit Congestion Notification (ECN) mechanisms in the TCP/IP protocol. The first part proposes new guidelines for TCP's response to ECN mechanisms (e.g., Source Quench packets, ECN fields in packet headers). Next, using simulations, we explore the benefits and drawbacks of ECN in TCP/IP networks. Our simulations use RED gateways modified to set an ECN bit in the IP packet header as an indication of congestion, with Reno-style TCP modified to respond to ECN as well as to packet drops as indications of congestion. The simulations show that one advantage of ECN mechanisms is in avoiding unnecessary packet drops, and therefore avoiding unnecessary delay for packets from low-bandwidth delay-sensitive TCP connections. A second advantage of ECN mechanisms is in networks (generally LANs) where the effectiveness of TCP retransmit timers is limited by the coarse granularity of the TCP clock. The paper also discusses some implementation issues concerning specific ECN mechanisms in TCP/IP networks.

1 Introduction

This paper proposes guidelines for TCP's response to ECN (Explicit Congestion Notification) mechanisms, and explores the effect upon performance of ECN mechanisms in TCP/IP networks. The paper discusses some implementation issues concerning ECN mechanisms, but does not make specific recommendations concerning the use of ECN mechanisms in TCP/IP networks.

In current TCP/IP networks, TCP relies on packet drops as the indication of congestion. The TCP source detects dropped packets either from the receipt of three duplicate acknowledgements (ACKs) or after the timeout of a retransmit timer, and responds to a dropped packet by reducing the congestion window [J88]. TCP implementations also respond to ICMP Source Quench messages, but Source Quench messages are rarely used, in part because they can consume network bandwidth in times of congestion.

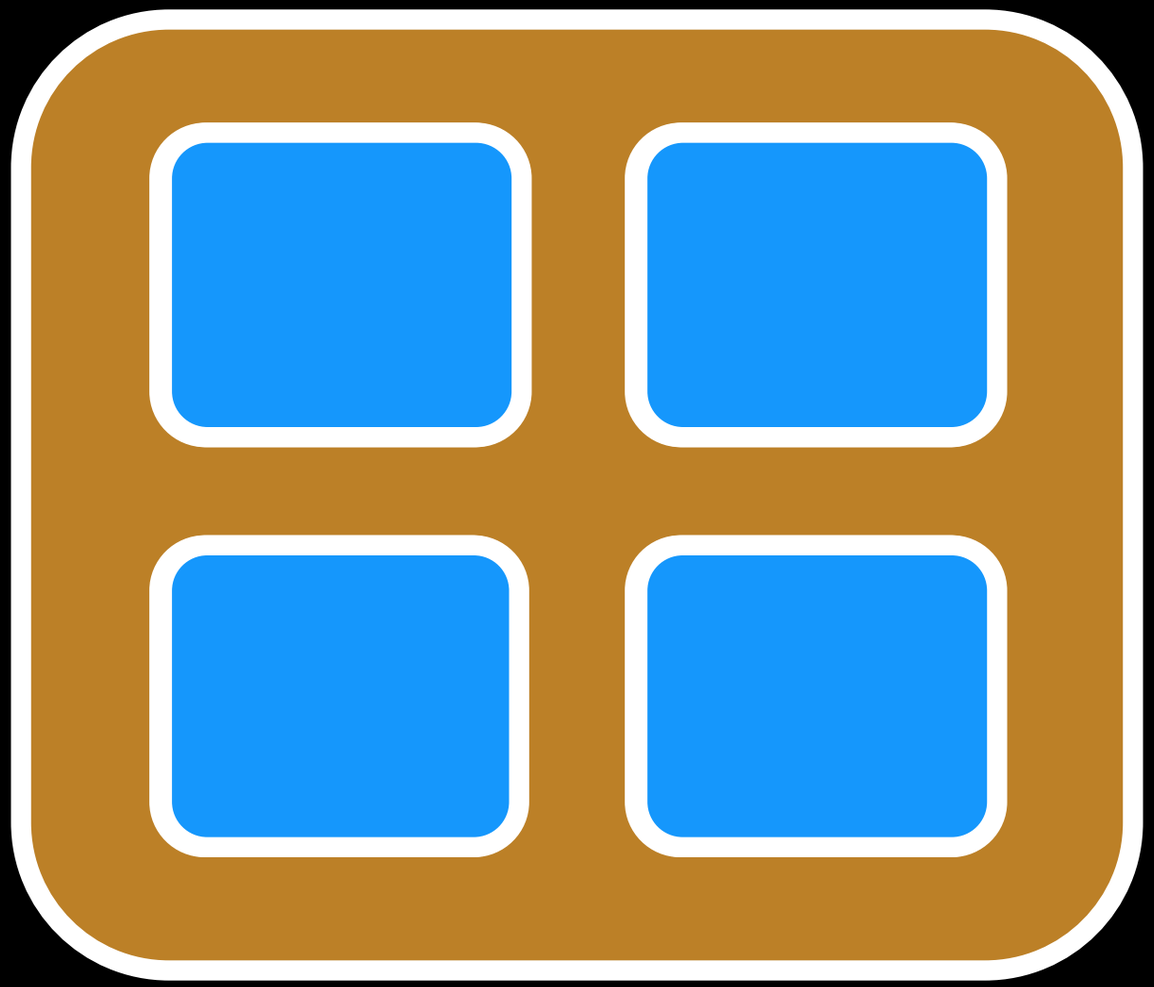
The reliance on packet drops as the indication of congestion is perfectly appropriate for a network with routers whose main function is to route packets to the appropriate output port. Most current routers in TCP/IP networks have no provision for the detection of incipient congestion. When a queue overflows, packets are dropped. When the TCP source detects this packet drop, the TCP source infers the presence of congestion in the network.

Future routers are likely to have more developed mechanisms for the detection of incipient congestion. With the DECbit scheme, for example, routers detect incipient congestion by computing the average queue size, and set the ECN bit in packet headers when the average queue size exceeds a certain threshold [RJ90]. Recently-proposed Random Early Detection (RED) gateways have a similar ability to detect incipient congestion [FJ93]. Gateways with mechanisms for detecting incipient congestion before the queue overflows are not limited to packet drops as the method of informing sources of congestion.

For networks with mechanisms for the detection of incipient congestion, the use of ECN mechanisms for the notification of congestion to the end nodes prevents unnecessary packet drops. For bulk-data connections, the user is concerned only with the arrival time of the last packet of data, and delays of individual packets are of no concern. For some interactive traffic, however, such as telnet traffic, the user is sensitive to the delay of individual packets. For such low-bandwidth delay-sensitive TCP traffic, unnecessary packet drops and packet retransmissions can result in noticeable and unnecessary delays for the user. For some connections, these delays can be exacerbated by a coarse-granularity TCP timer that delays the source's retransmission of the packet.

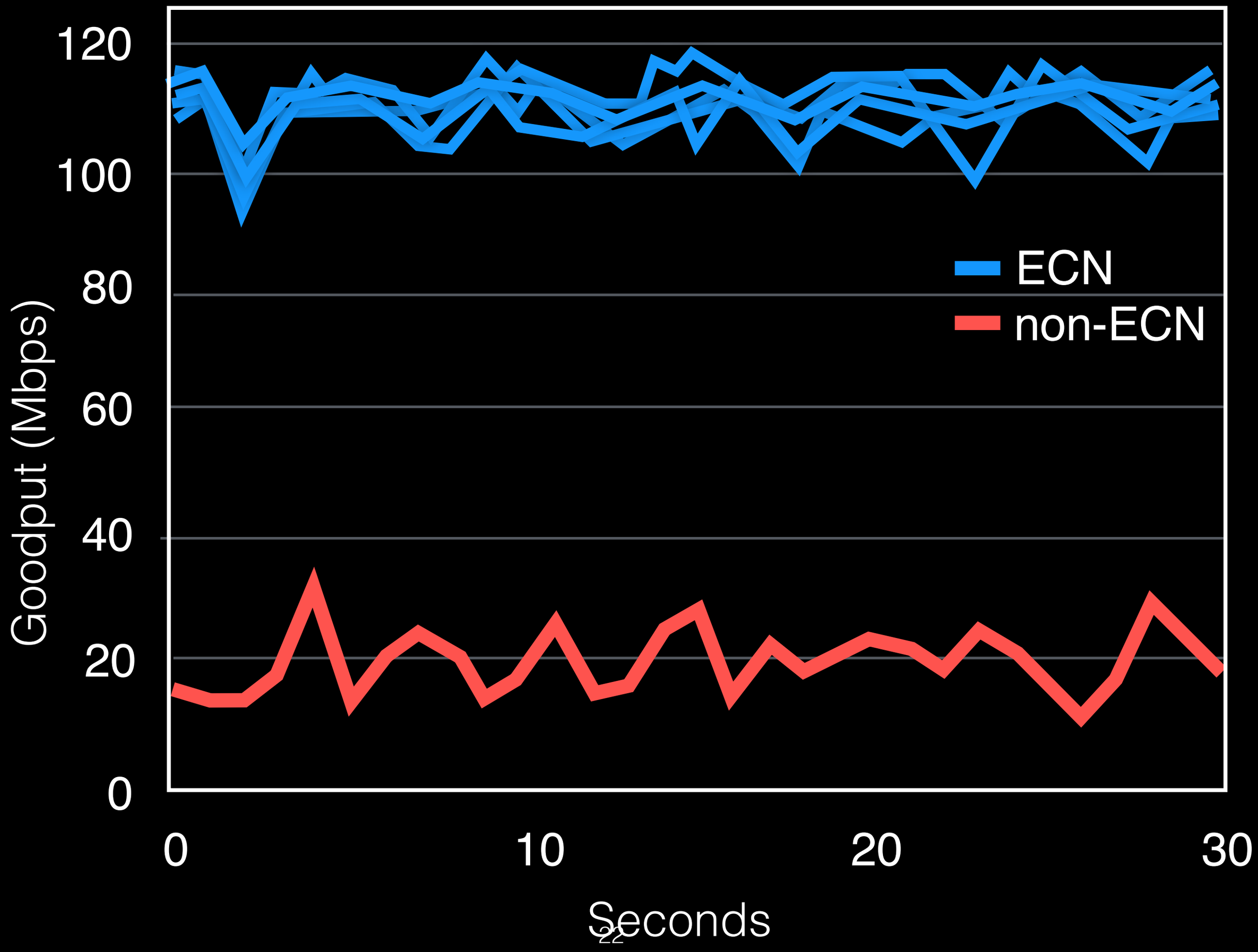
A second benefit of ECN mechanisms is that with ECN, sources can be informed of congestion quickly and unambiguously, without the source having to wait for either a retransmit timer or three duplicate ACKs to infer a dropped packet. For bulk-data TCP connections,

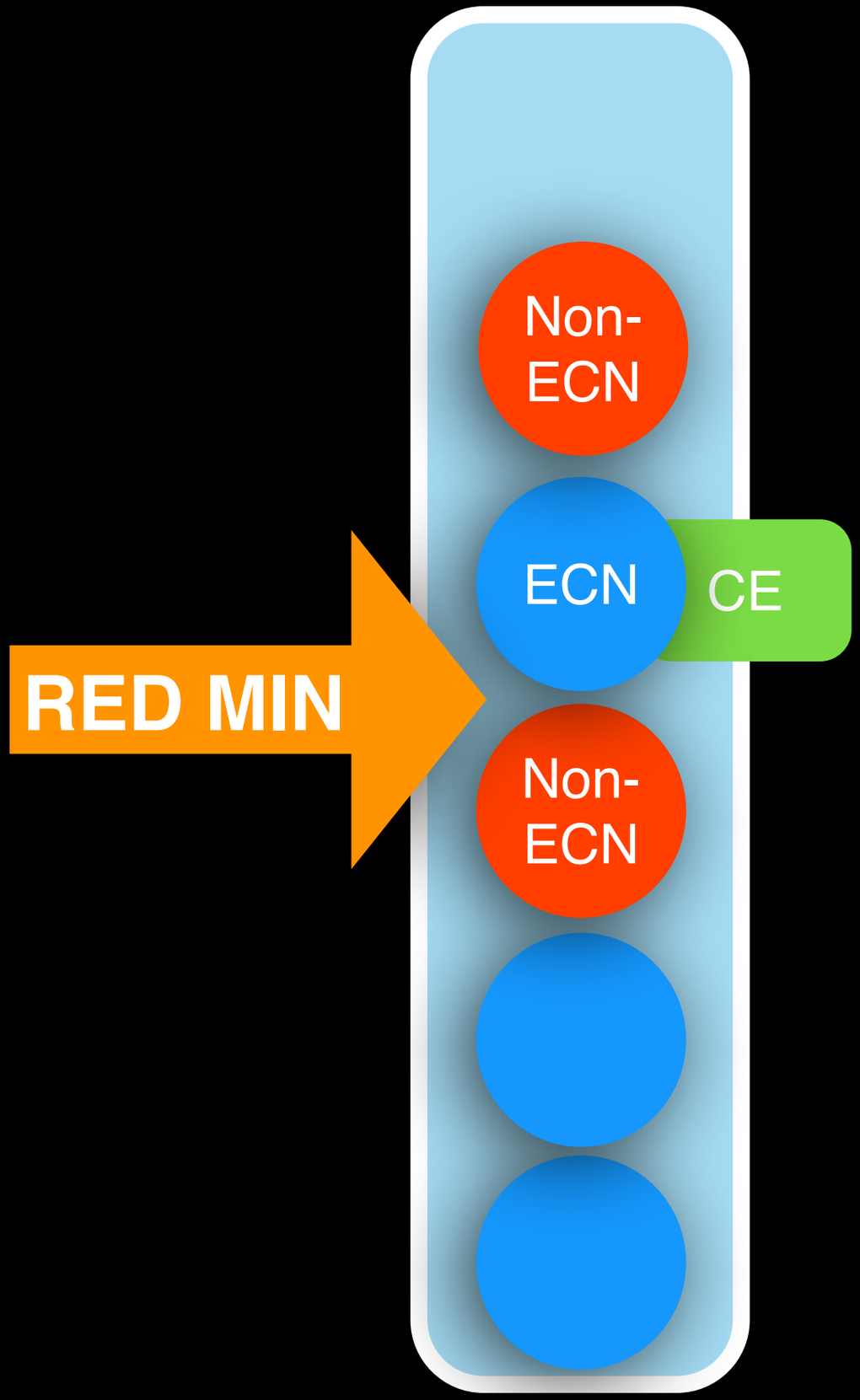
*This work was supported by the Director, Office of Energy Research, Scientific Computing Staff, of the U.S. Department of Energy under Contract No. DE-AC03-76SF00098.




DO NOT
PRESS THE
BIG RED
BUTTON


CARRIER
TRANSDATA
LIMITED
DATA
307
1
2
3
4
5
6
7
8
9
*
0
#


120
100
80
60
40
20
0
Goodput (Mbps)
ECN
non-ECN
0
10
20
30
Seconds



3 Dropped

2 Marked

1 Queued


All Packets Dropped
ECN Marked & non-ECN Dropped
All Packets Queued

Sender

Bottleneck Link

ECN

ECN

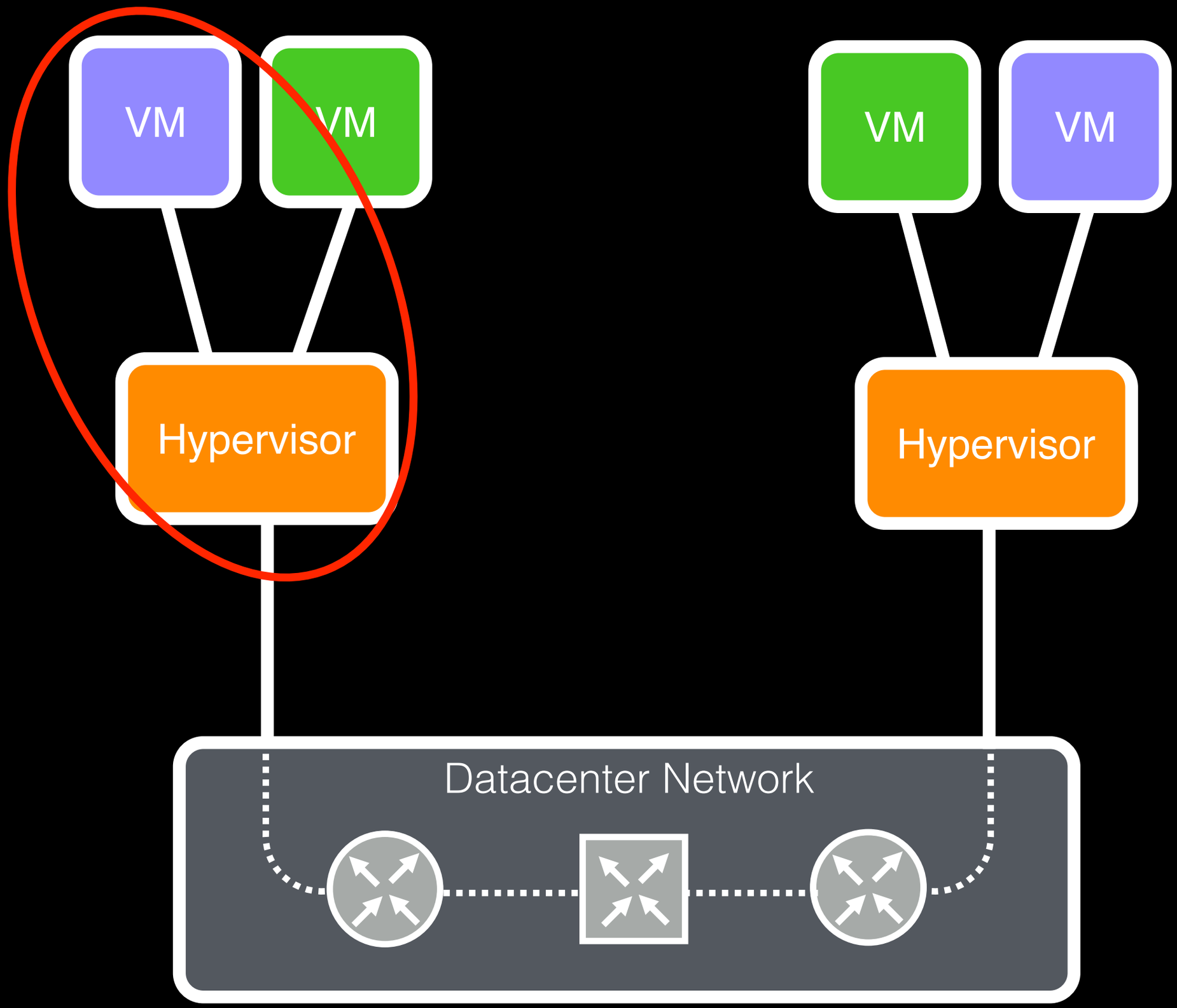
ECN

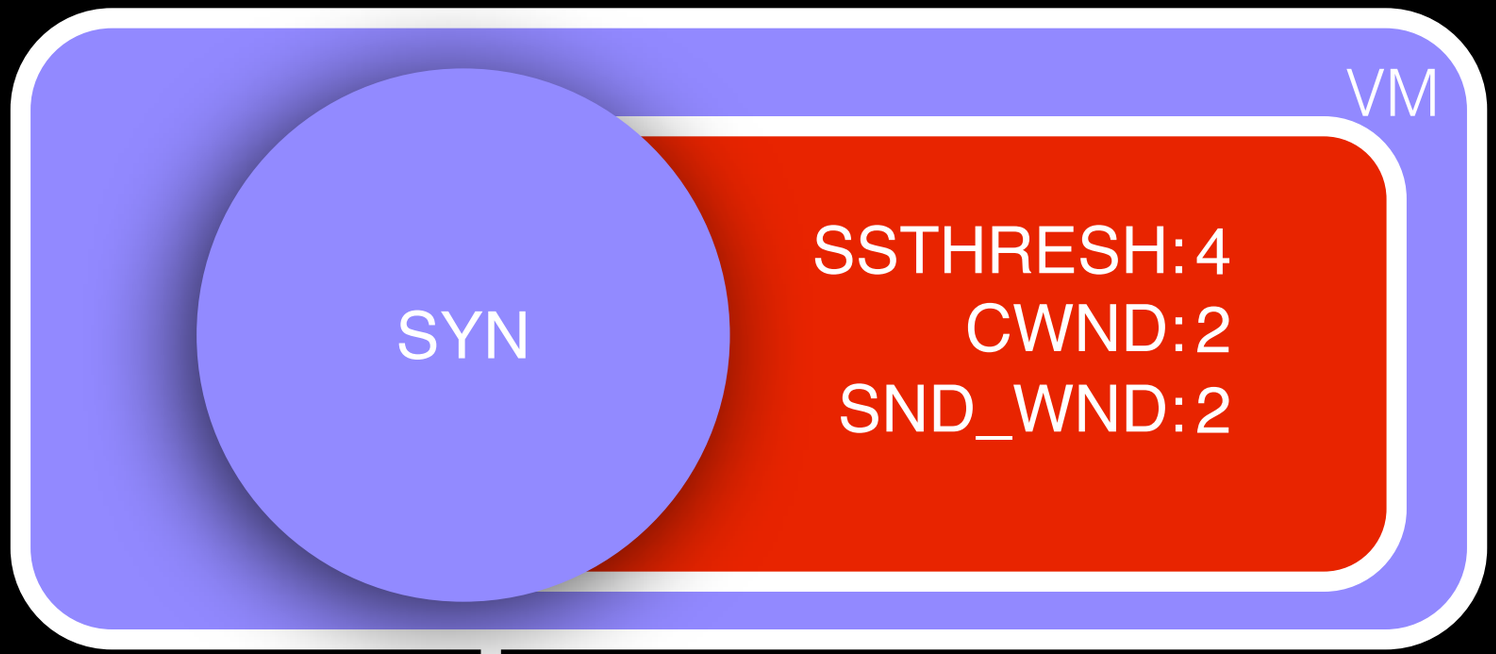
translated to ECN

Receiver

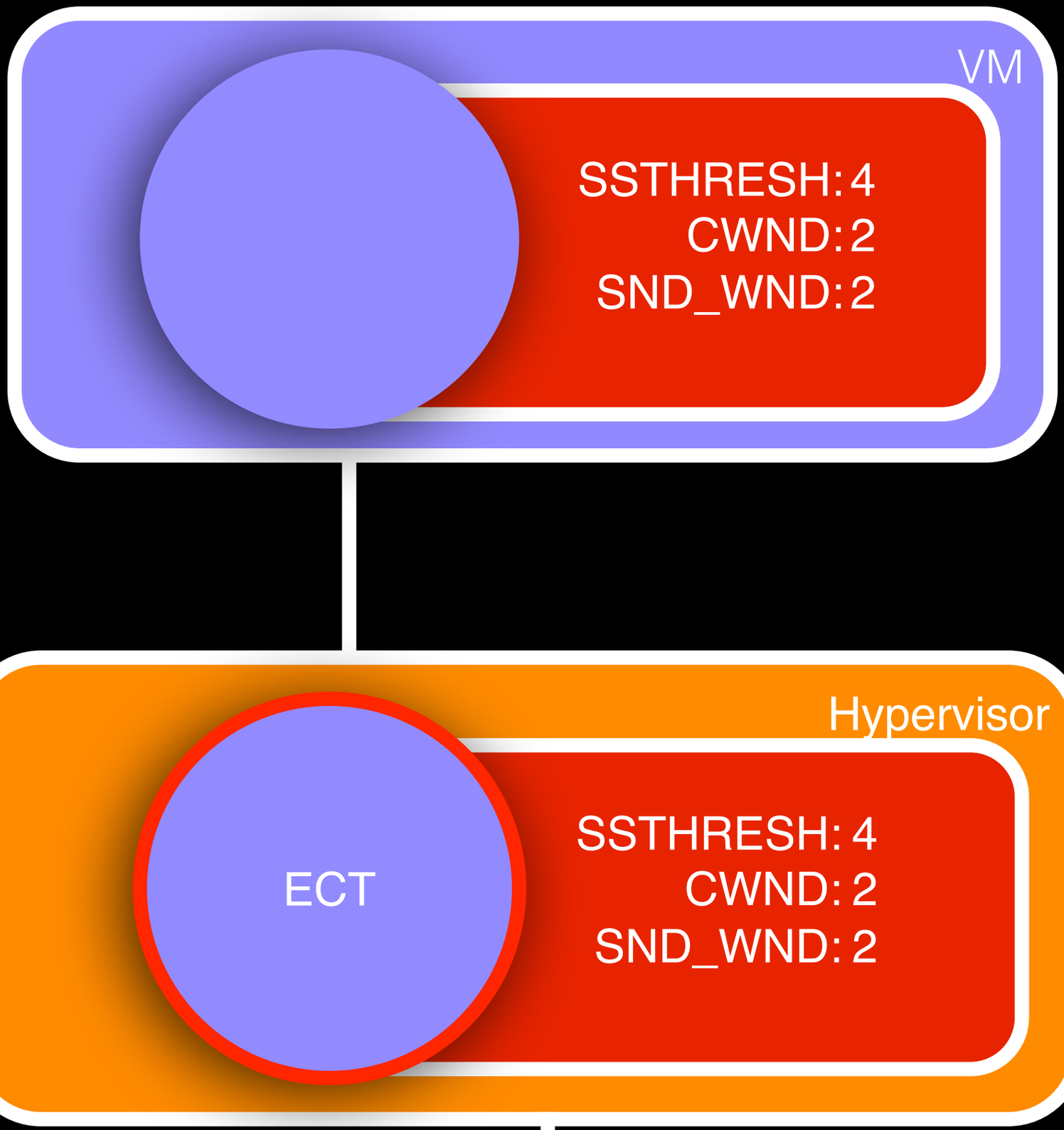
vCC

Virtualized Congestion Control


VM
VM
Hypervisor
VM
VM
Hypervisor
Datacenter Network



SSTHRESH: Slow Start Threshold
CWND: Congestion Window
SND_WND: Send Window


VM
SSTHRESH: 4
CWND: 2
SND_WND: 2
Hypervisor
ECT
SSTHRESH: 4
CWND: 2
SND_WND: 2

VM

SSTHRESH: 4

CWND: ~~3~~ 4

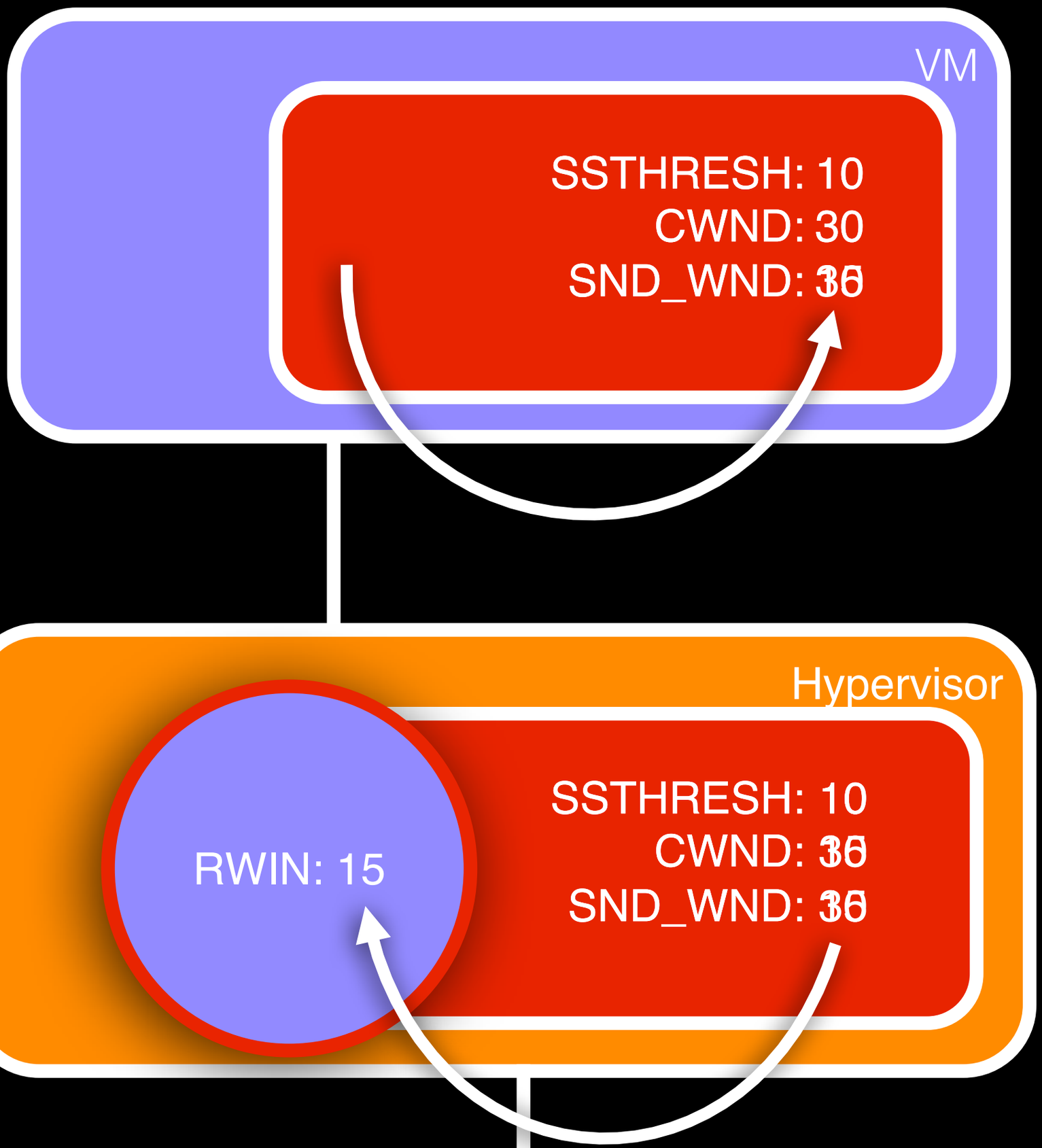
SND_WND: ~~3~~ 4

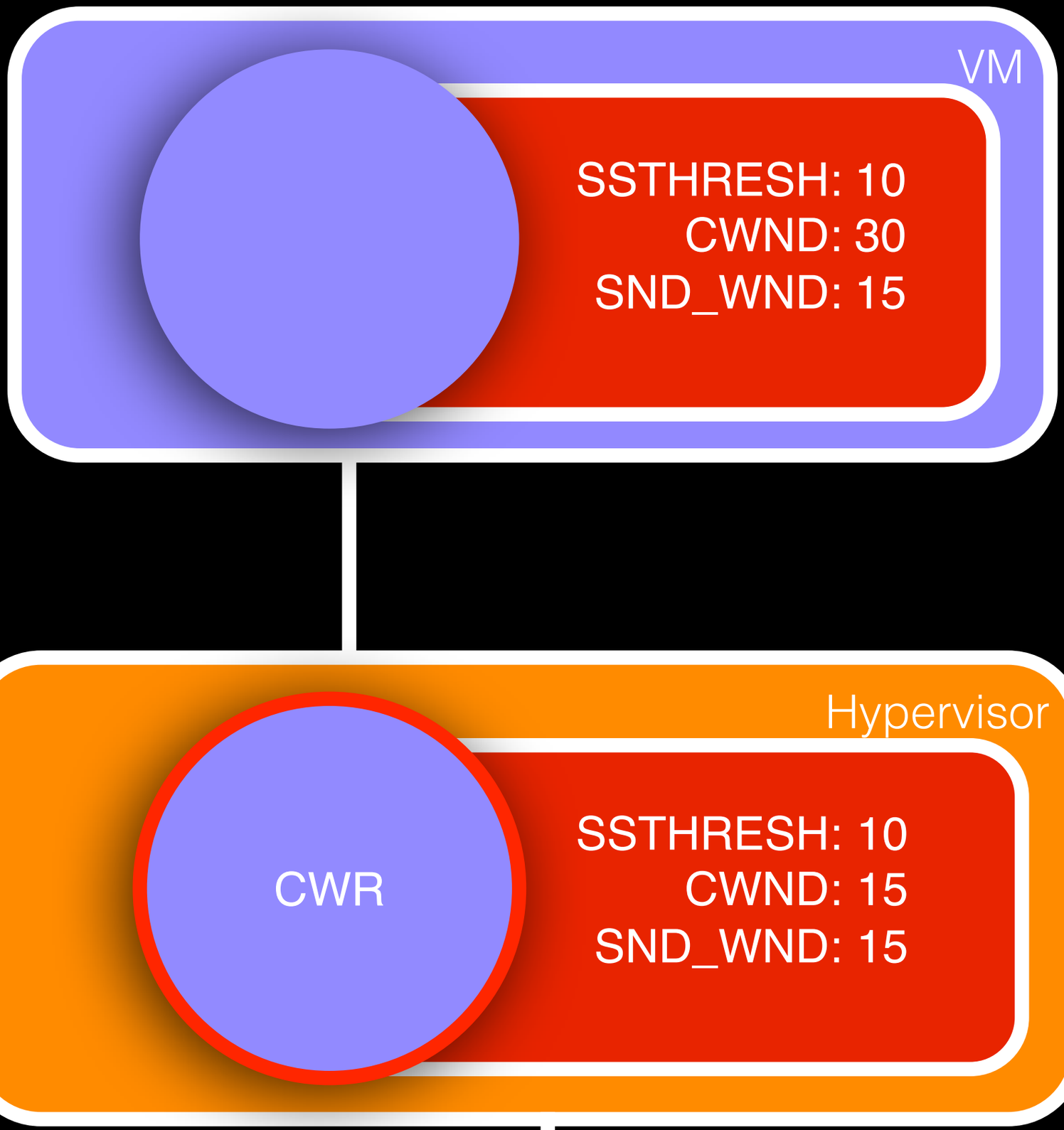
Hypervisor

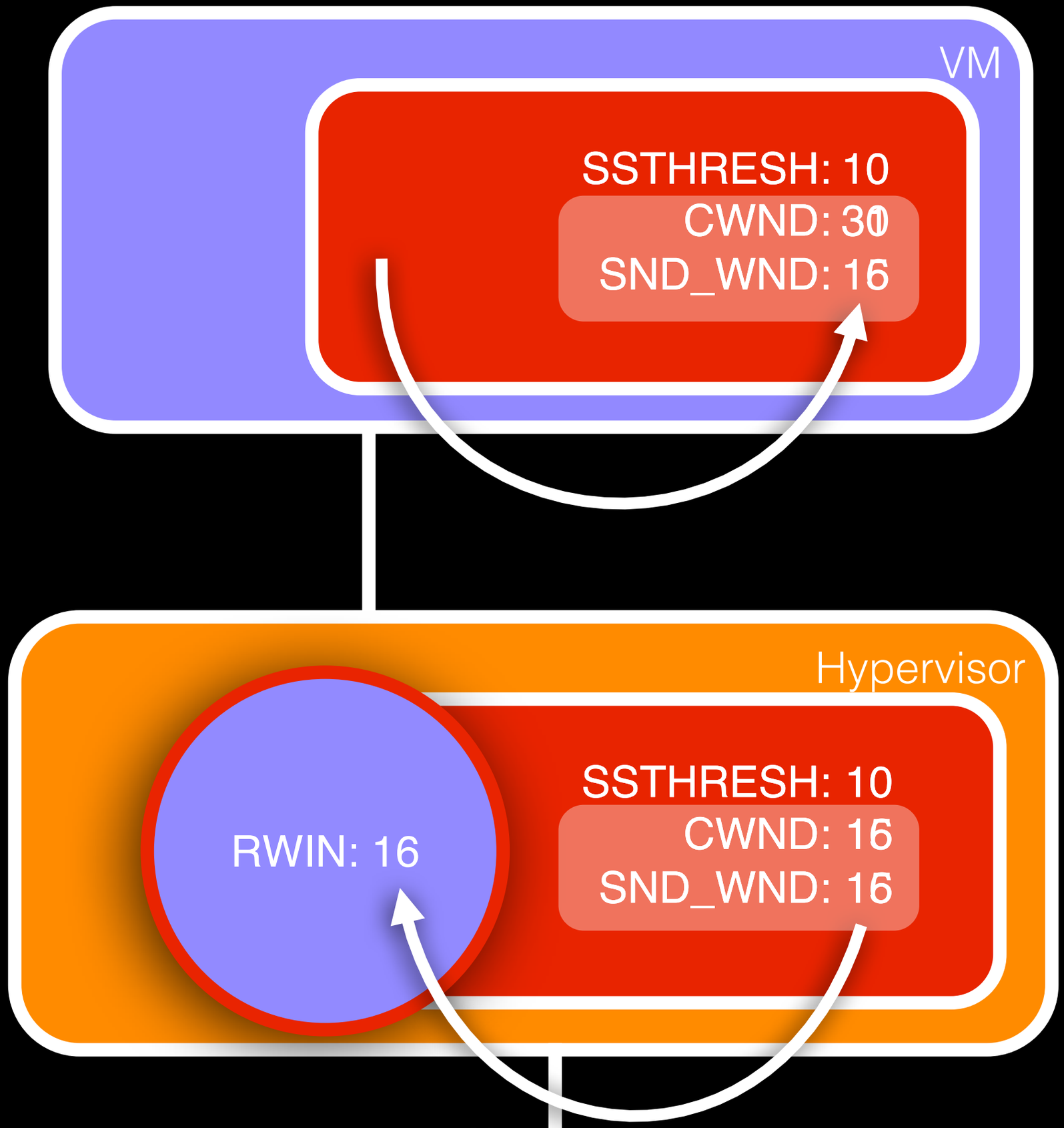
SSTHRESH: 4

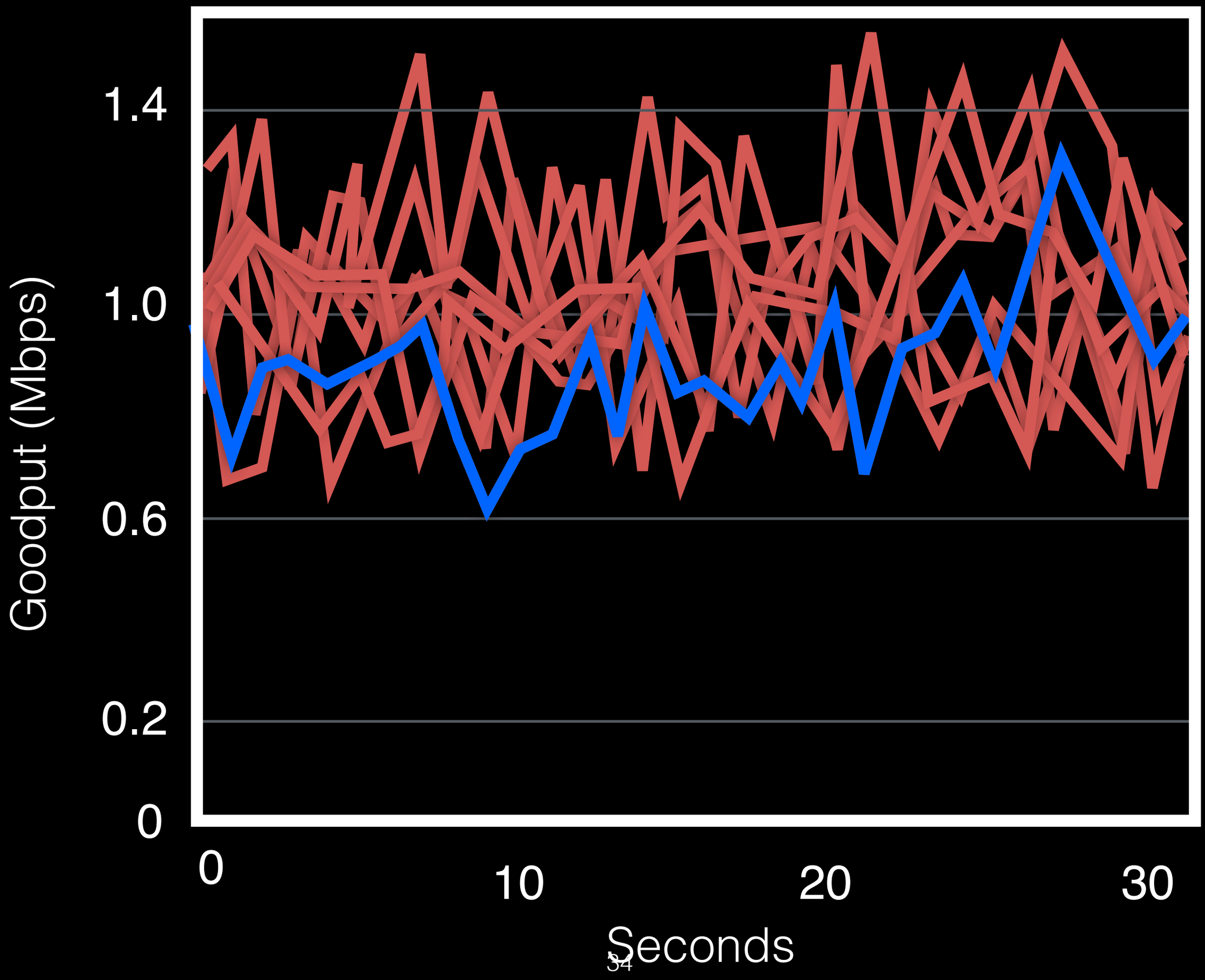
CWND: ~~3~~ 4

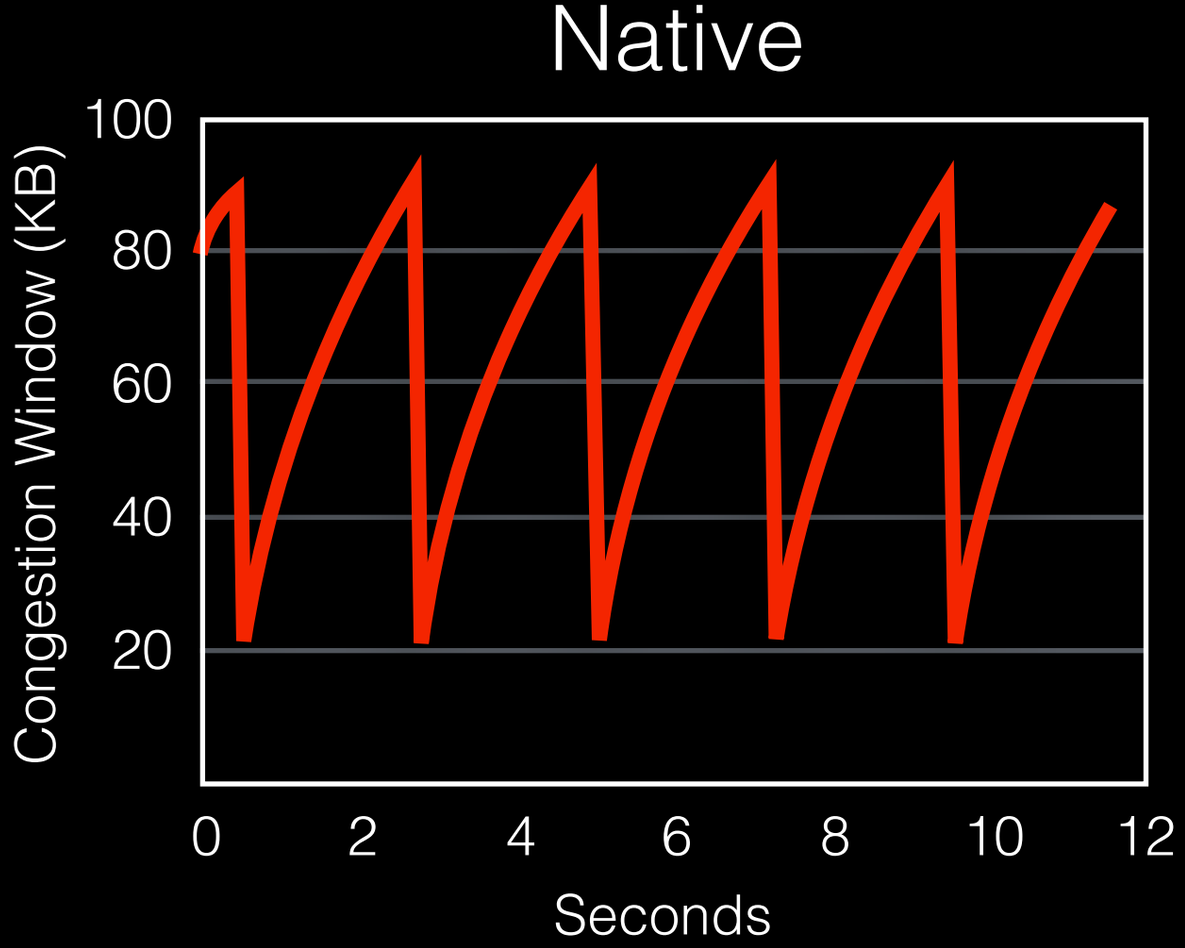
SND_WND: ~~3~~ 4

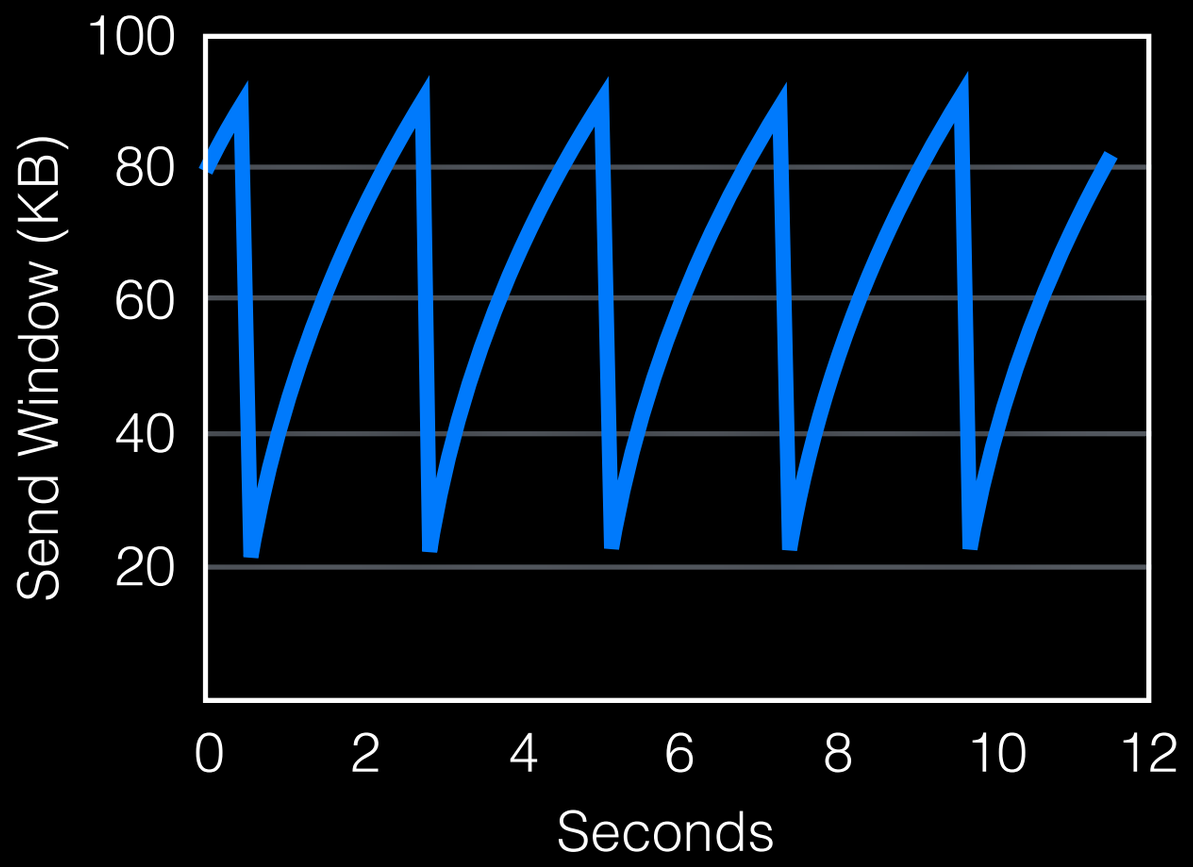

VM
SSTHRESH: 10
CWND: 30
SND_WND: 30
Hypervisor
RWIN: 15
SSTHRESH: 10
CWND: 30
SND_WND: 30


VM
SSTHRESH: 10
CWND: 30
SND_WND: 15
Hypervisor
CWR
SSTHRESH: 10
CWND: 15
SND_WND: 15

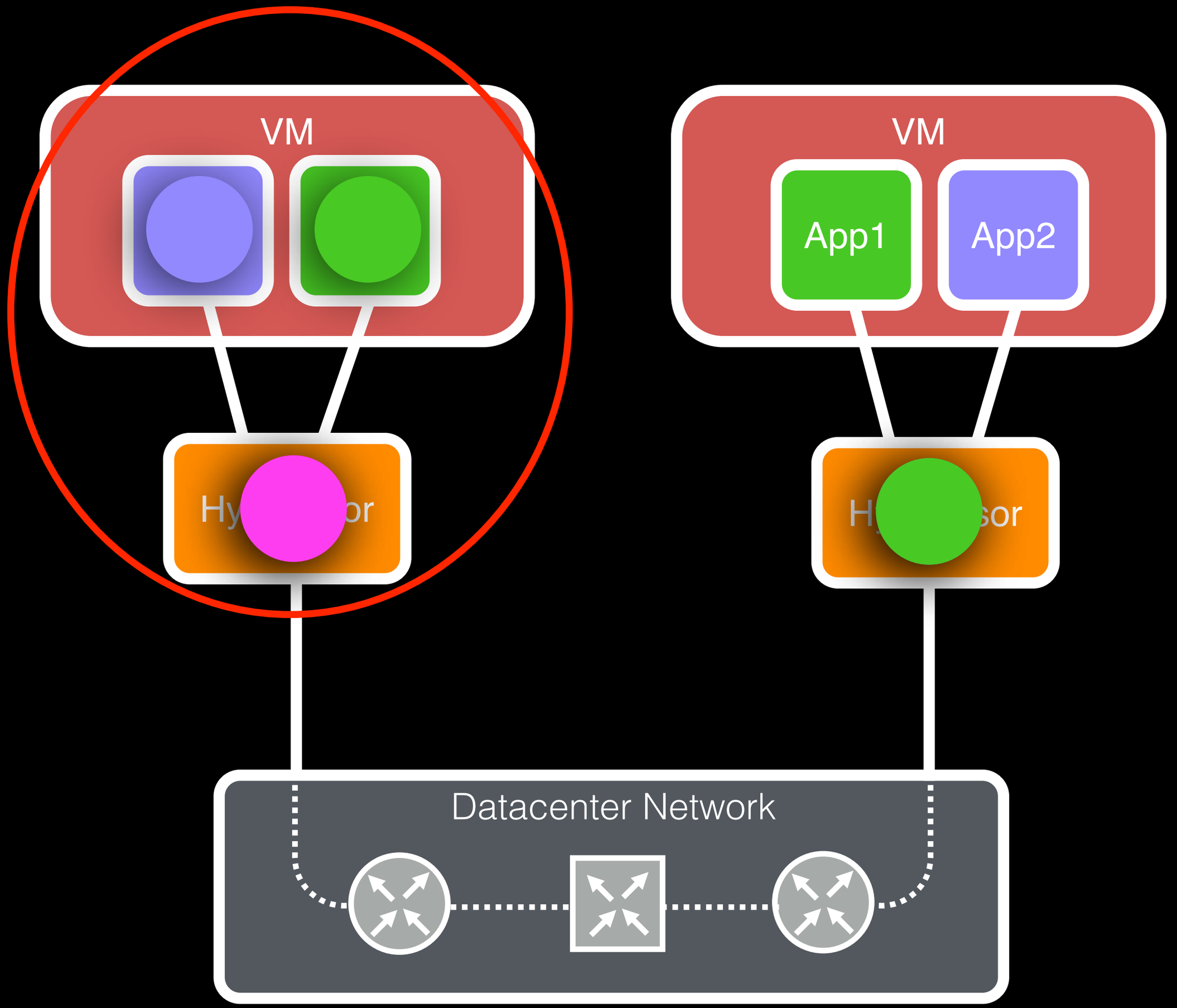

VM
SSTHRESH: 10
CWND: 30
SND_WND: 16
Hypervisor
RWIN: 16
SSTHRESH: 10
CWND: 16
SND_WND: 16

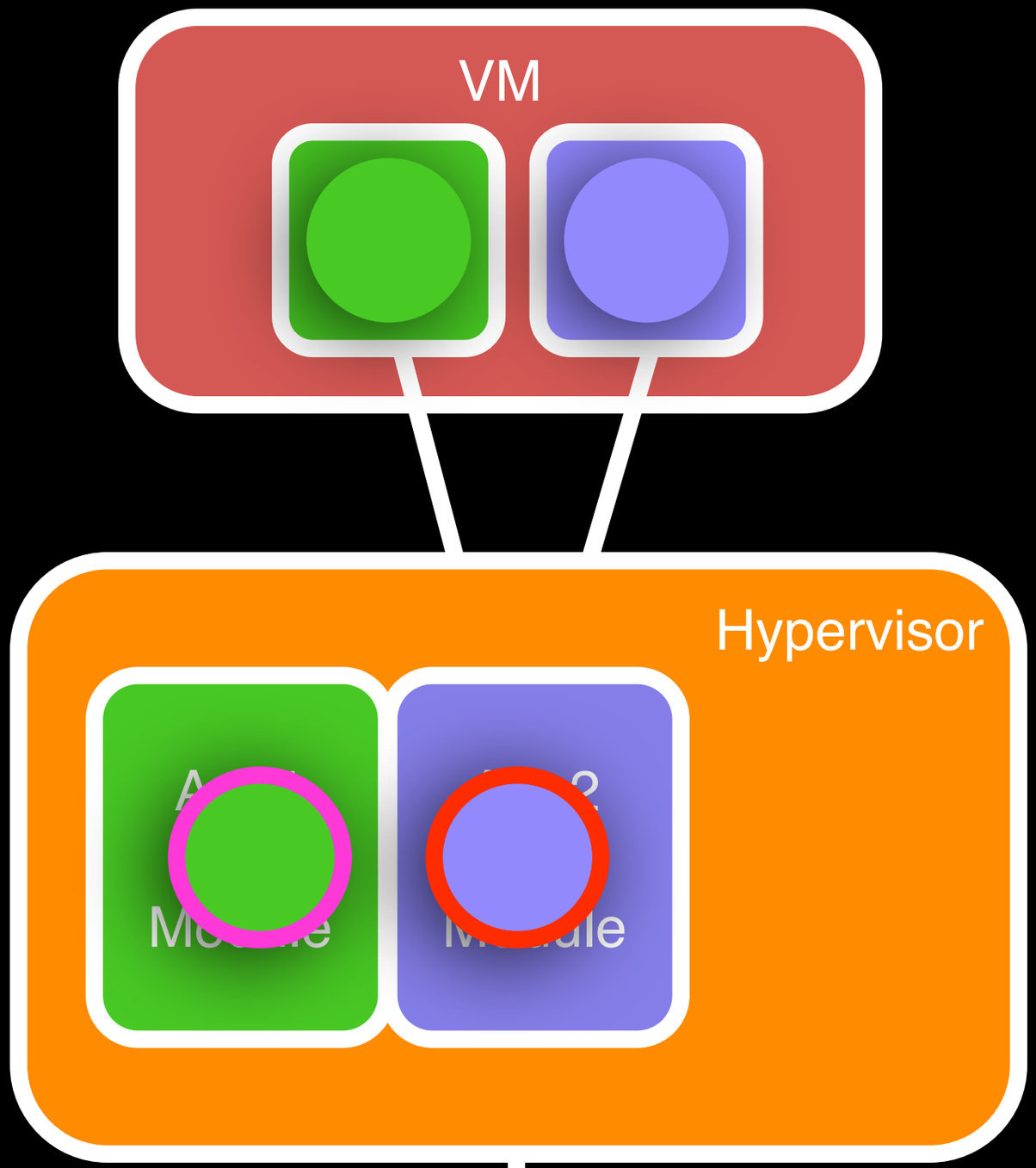

Goodput (Mbps)
1.4
1.0
0.6
0.2
0
0
10
20
30
Seconds

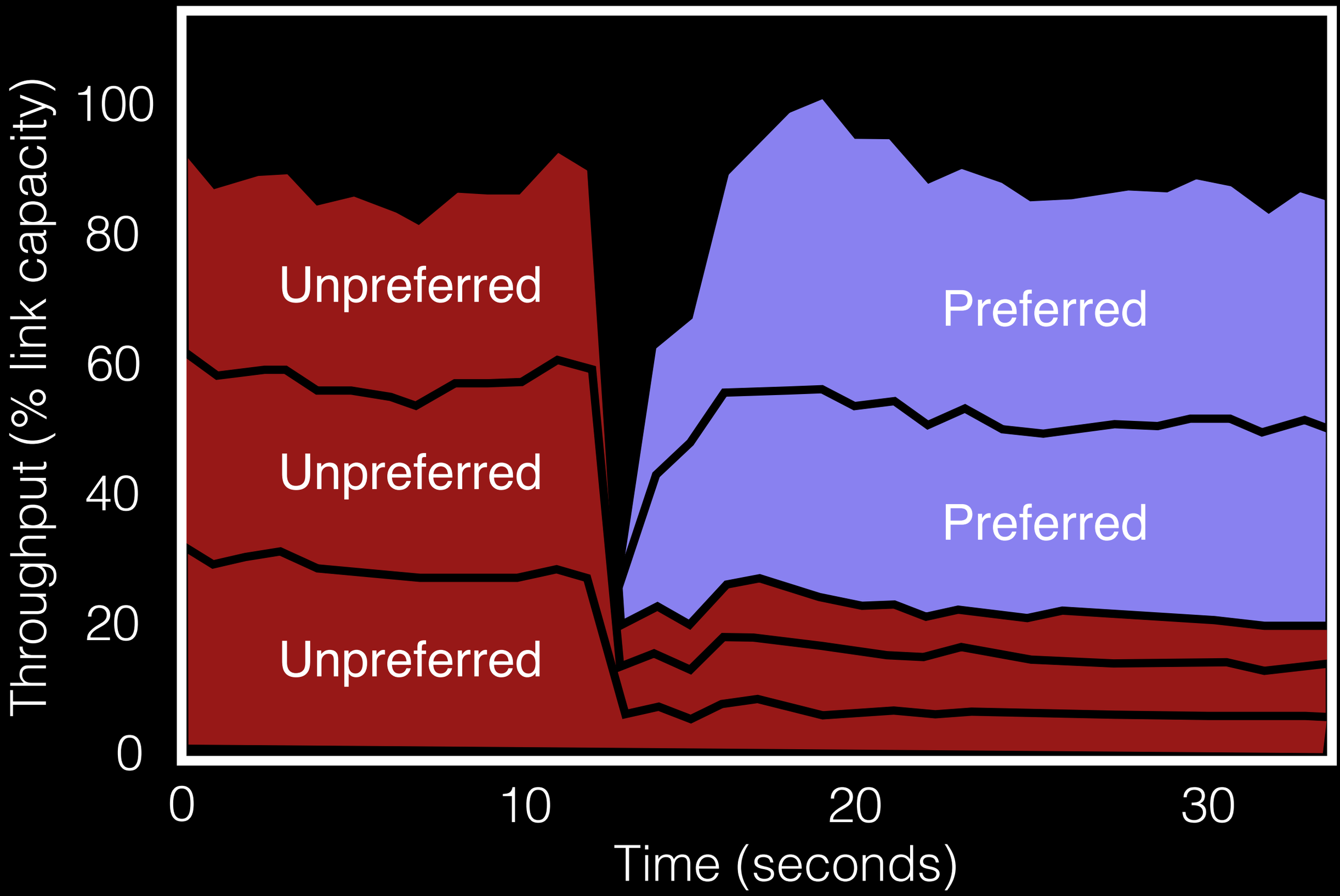

Native
Congestion Window (KB)
100
80
60
40
20
0
2
4
6
8
10
12
Seconds


vCC
Send Window (KB)
100
80
60
40
20
0
2
4
6
8
10
12
Seconds


Squirtle
CP ???
AR


VM
VM
App1
App2
Datacenter Network


VM
Hypervisor


Throughput (% link capacity)
100
80
60
40
20
0
Unpreferred
Unpreferred
Unpreferred
Preferred
Preferred
0
10
20
30
Time (seconds)

Tahoe

NewReno

CANIT

Illinois

Vegas

Cubic

Reno

MP-TCP

TFRC

Scalable

Agile-SD

Hybla

Zeta

FAST

Westwood+

Indy

BIC

YeAH

Jersey

FIT

Veno

XCP

Real

pFabric

Compound

Timely

H-TCP

LP

HSTCP

DCTCP

ECN Experiments in this Presentation:

Linux 3.19 mininet testbed
vCC implemented as a kernel patch

Public vCC Code:

Linux patch and mininet test suite available on Github
Public Code and Extended Paper:

http://webee.technion.ac.il/~isaac/vcc/

Our Paper:

ECN unfairness, virtual-ECN vCC, ESX app-level throttling vCC, technique survey, future implementation analysis

Thank You!







Stanford PLATFORMLAB

TECHNION Israel Institute of Technology

vmware® RESEARCH

vCC

Virtualized Congestion Control

Translate between Congestion Control Algorithms
in the Hypervisor

in order to

- Achieve uniform congestion control in multitenant and enterprise datacenters
- Programatically assign congestion control to fine-grained flow signatures
- Simplify rollout of new congestion control algorithms