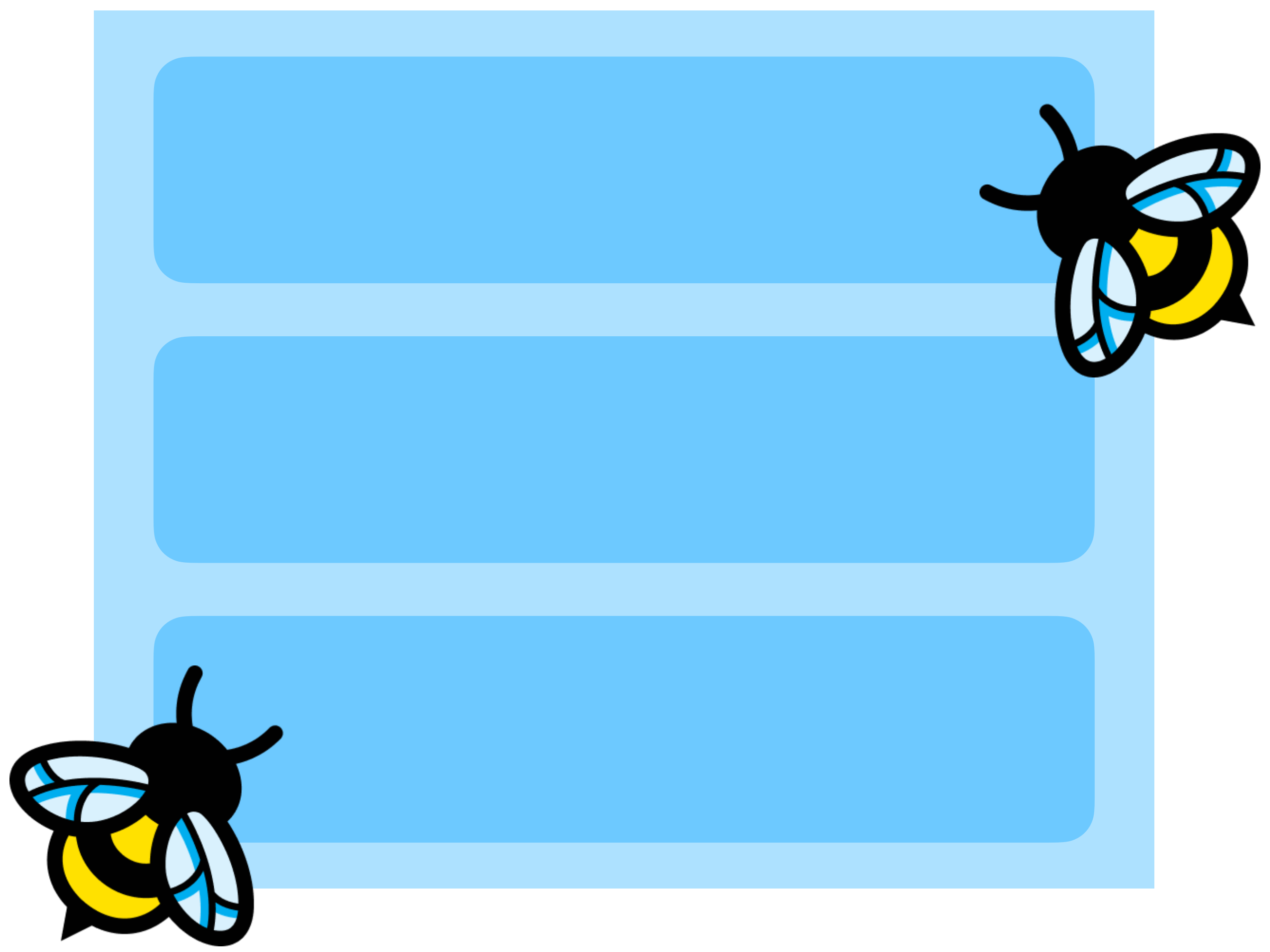
HEELS: A Host-Enabled eBPF-Based Load Balancing Scheme



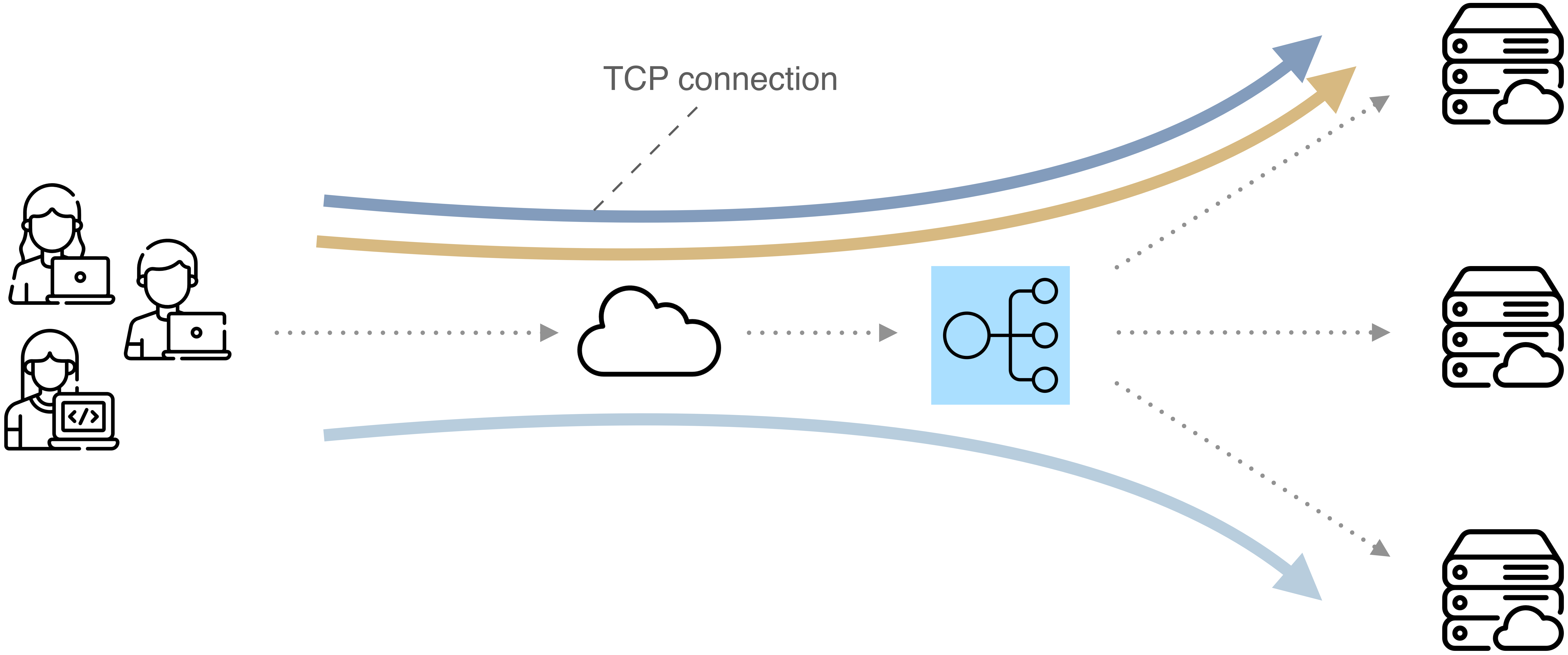
Rui Yang*   Marios Kogias†

eBPF workshop, SIGCOMM

September 10 2023

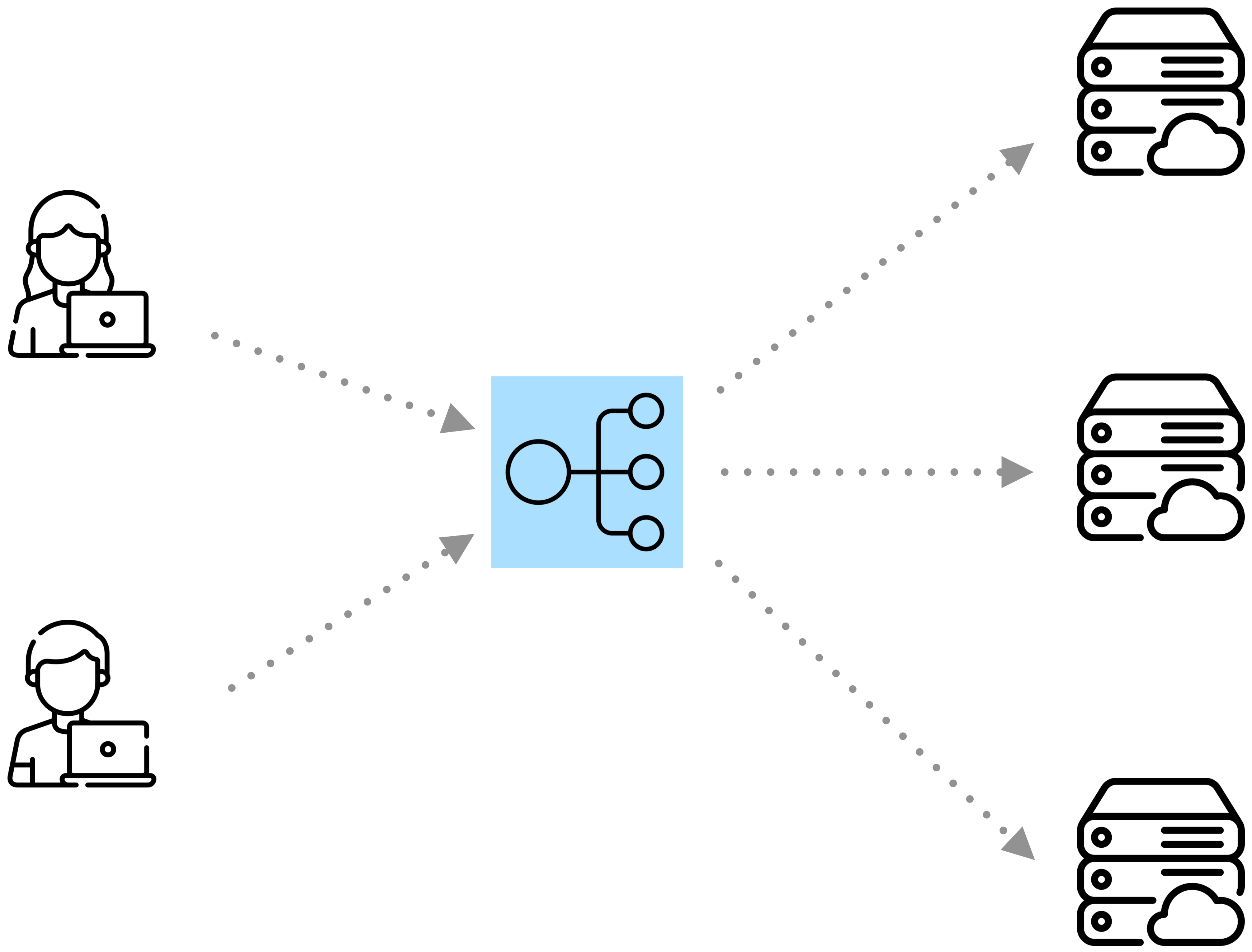
* EPFL   † Imperial College London

Layer 4 load balancer



L4 load balancer: Centralized Design

Maglev [NSDI ’16], SilkRoad [Sigcomm ’17], Katran [Meta]

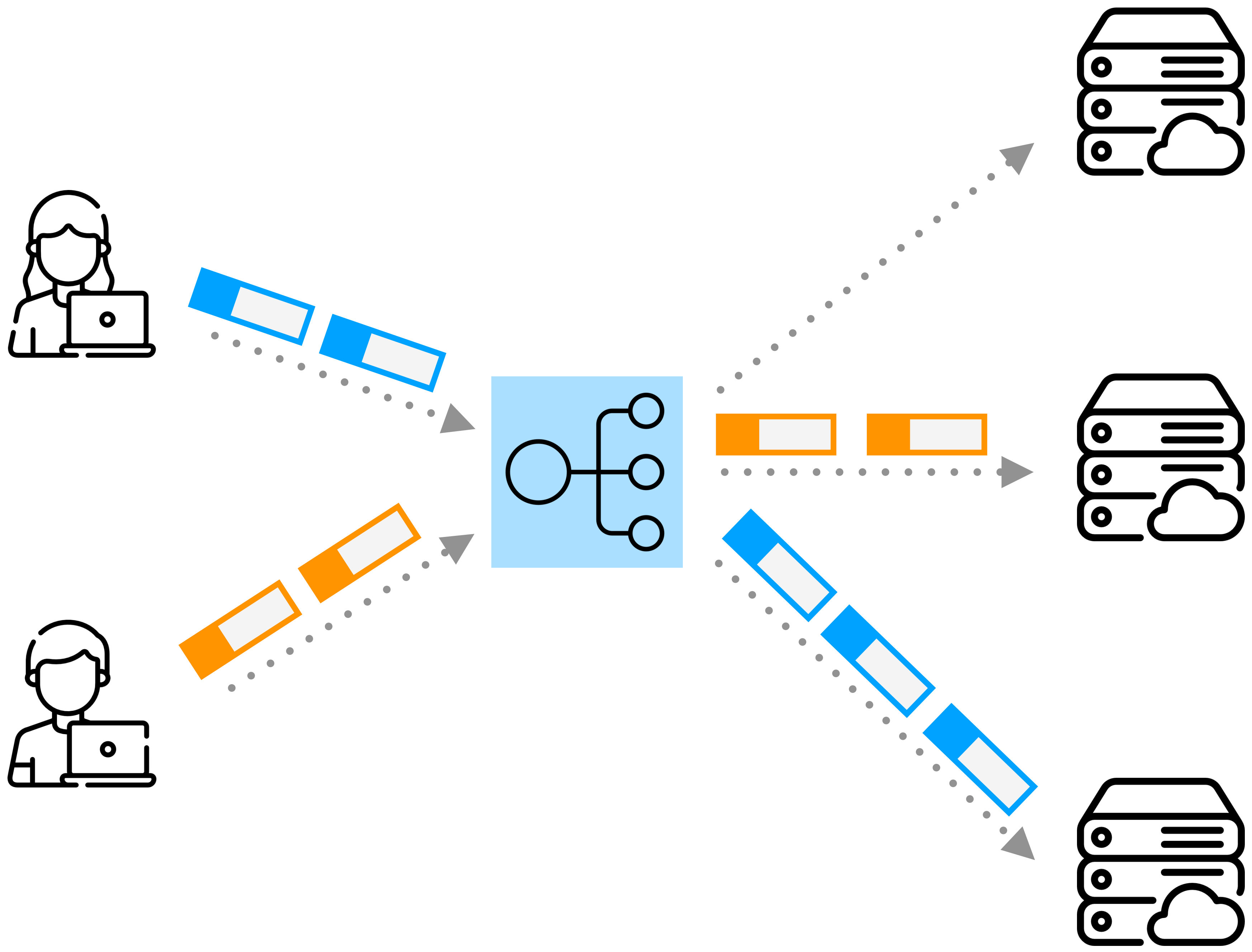


L4 load balancer: Centralized Design

Maglev [NSDI '16], SilkRoad [Sigcomm '17], Katran (Meta)

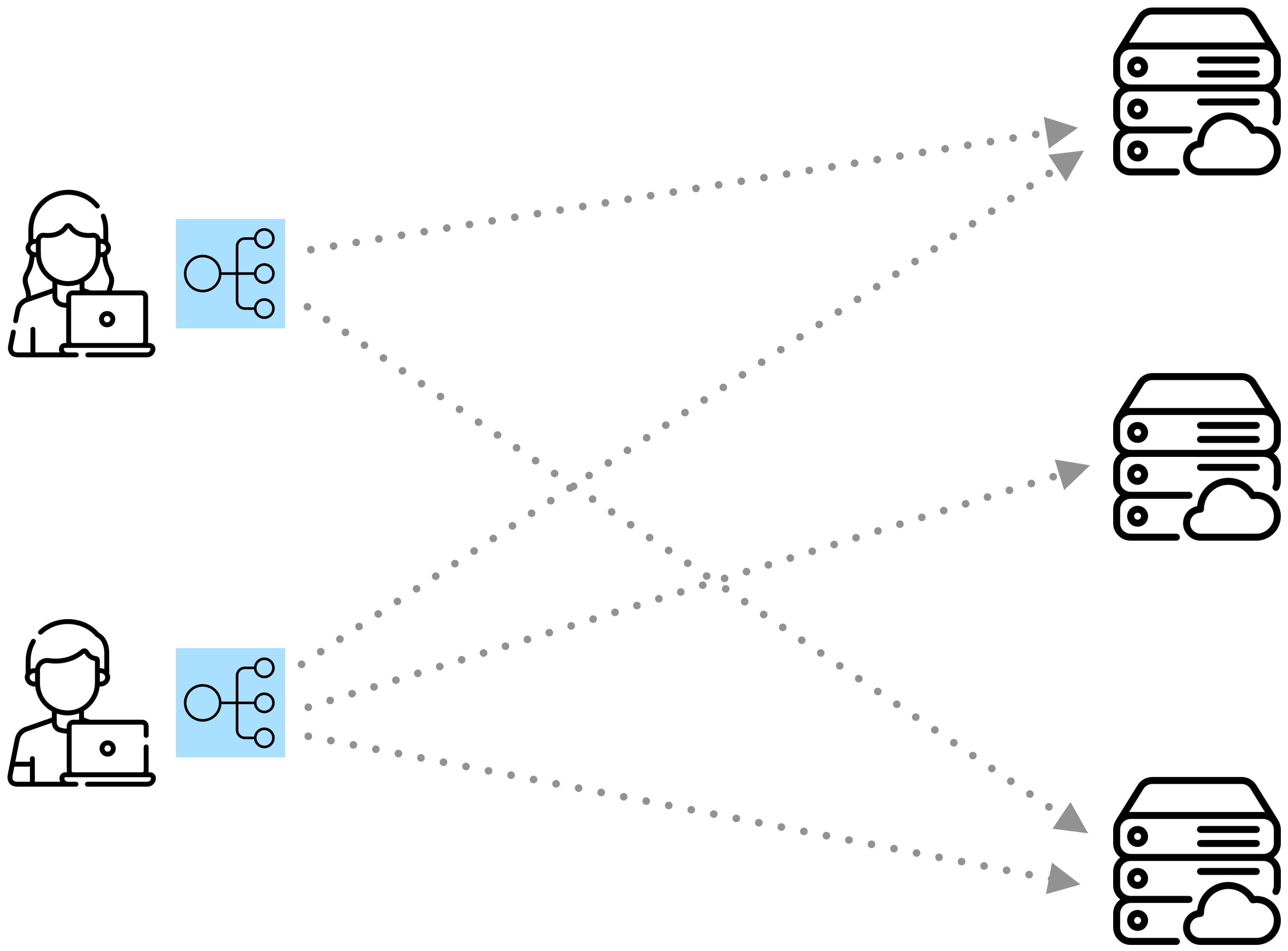
Efficiency ✓
load balancer has a global view

Scalability ✗
easily result in IO bottleneck



L4 load balancer: Decentralized Design

IPVS Kube-proxy (Kubernetes)

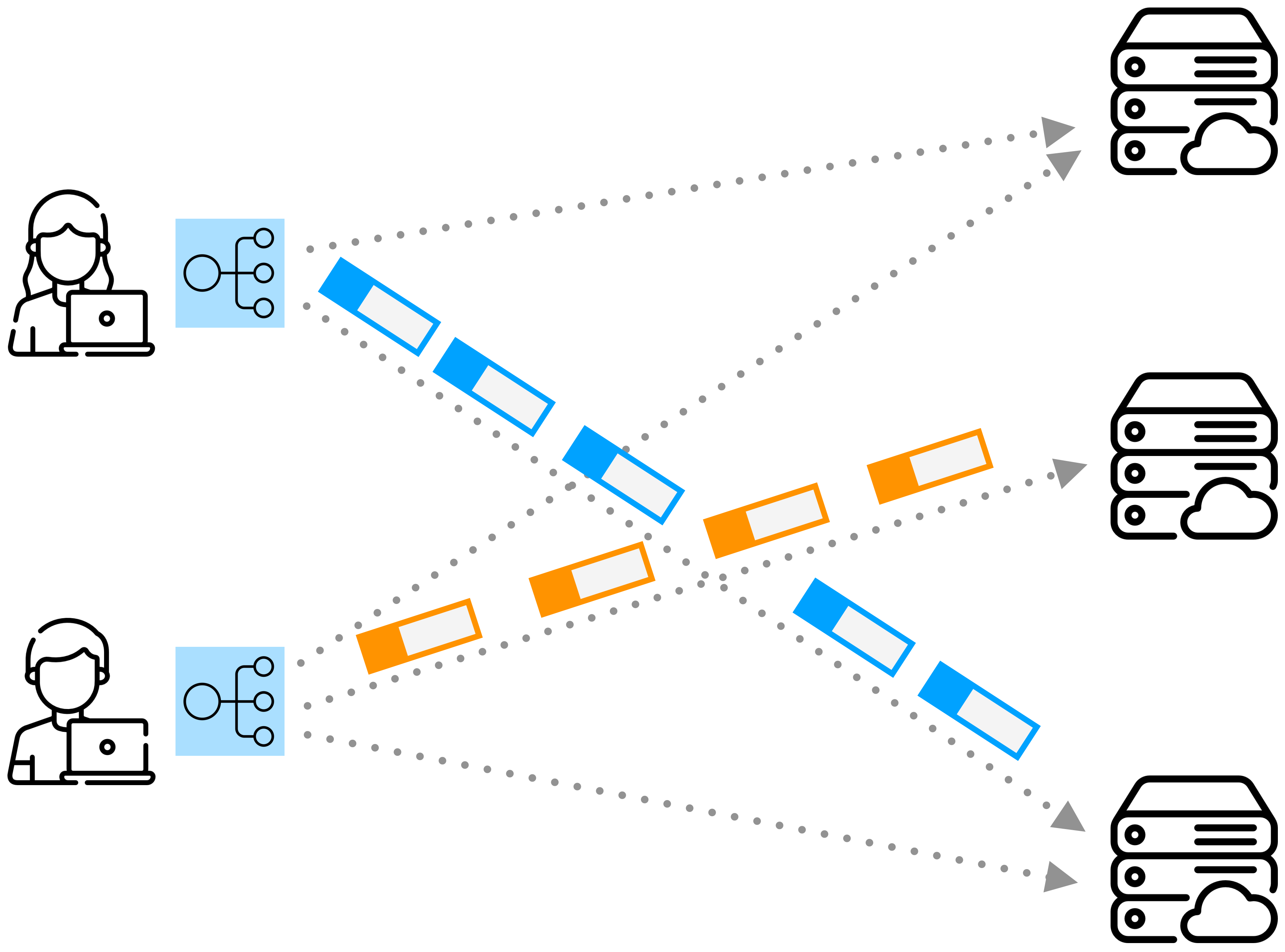


L4 load balancer: Decentralized Design

IPVS Kube-proxy (Kubernetes)

Efficiency ✗
load imbalance

Scalability ✓
Every node acts as a load balancer



L4 load balancers: best of both worlds

Efficiency ✓

Scalability ✓

CRAB

2020

Bypassing the Load Balancer Without Regrets

Marios Kogias
EPFL

Rishabh Iyer
EPFL

Edouard Bugnion
EPFL

ABSTRACT

Load balancers are a ubiquitous component of cloud deployments and the cornerstone of workload elasticity. Load balancers can significantly affect the end-to-end application latency with their load balancing decisions, and constitute a significant portion of cloud tenant expenses.

We propose CRAB, an alternative L4 load balancing scheme that eliminates latency overheads and scalability bottlenecks while simultaneously enabling the deployment of complex, stateful load balancing policies. A CRAB load balancer only participates in the TCP connection establishment phase and stays off the connection's datapath. Thus, load balancer provisioning depends on the rate of new connections rather than the actual connection bandwidth. CRAB depends on a new TCP option that enables connection redirection. We provide different implementations for a CRAB load balancer on different technologies, *e.g.*, P4, DPDK, and eBPF, showing that a CRAB load balancer does not require many resources to perform well. We introduce the connection redirection option to the Linux kernel with minor modifications, so that it that can be shipped with the VM images offered by the cloud providers. We show how the same functionality can be achieved with a vanilla Linux kernel using a Netfilter module, while we discuss how CRAB can work while clients and servers remain completely agnostic, based on functionality added on the host.

Our evaluation shows that CRAB pushes the IO bottleneck from the load balancer to the servers in cases where vanilla L4 load balancing does not scale and provides end-to-end latencies that are close to direct communication while retaining all the scheduling benefits of stateful L4 load balancing.

ACM Reference Format:

Marios Kogias, Rishabh Iyer, and Edouard Bugnion. 2020. Bypassing the Load Balancer Without Regrets. In *ACM Symposium on Cloud Computing (SoCC '20), October 19–21, 2020, Virtual Event, USA.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3419111.3421304

1 INTRODUCTION

Load balancing is ubiquitous: nearly all applications today running in datacenters, public clouds, at the edge, or as core internet services rely on some form of load-balancing for both availability and scalability. Load balancing can have different forms, *e.g.*, L4, L7, DNS-based *etc.* and can be implemented in hardware or in software. There has been considerable research on load balancing [3, 9, 16, 24, 35, 42, 43, 47–49] both from academia and industry due to not only the demands for mass deployments, high throughput, and low latency variability, but also the demands to lower provider resources specifically dedicated to it. For instance, Google reports that software-based load balancing can take up to 3-4% of a datacenter's resources [16].

This paper focuses on internal load balancers, which are deployed between clients and servers within the same datacenter or public cloud. Internal load balancers can have a significant impact on the end-to-end latency both due to their load balancing decisions and the intermediate hop, while also constituting a major part of the infrastructure costs for cloud tenants. A common pattern includes the deployment of an internal cloud service, placed behind an internal load balancer, that spawns new service instances according to load requirements and registers them with the load balancer, leading to seamless scalability and elasticity.

Figure 1 illustrates a sample cloud-based, two-tier application. Users using their browsers hit the public IP of the external load balancer and their requests end up being served by the two web servers. Those servers act as internal clients for the backend-servers that are behind the internal load balancer and communicate with a managed database service. This design pattern allows the web tier and the back-end tier to scale independently and remain agnostic to each other due to the use of the two load balancers. Similar examples of such design patterns for services (or microservices) include ML inference to create recommendations, a user authentication microservice [23], generic application servers, and any workload orchestrated in containers such as Kubernetes[39].

Internal load balancers must be able to handle low-latency, high-throughput RPCs, typically implemented on protocols such as gRPC [26], Thrift [55], HTTP, or even custom proto-

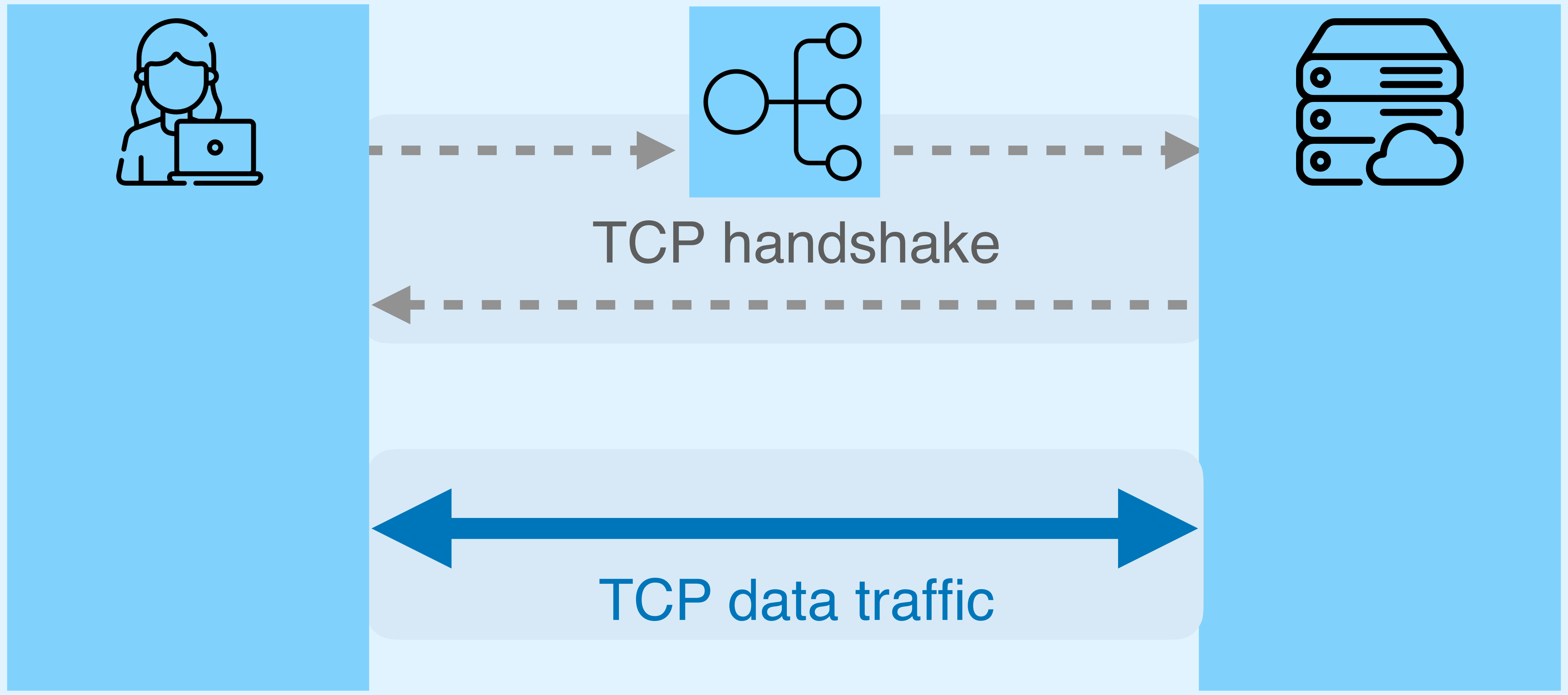
L4 load balancers: best of both worlds

Efficiency ✓

Scalability ✓

CRAB

2020



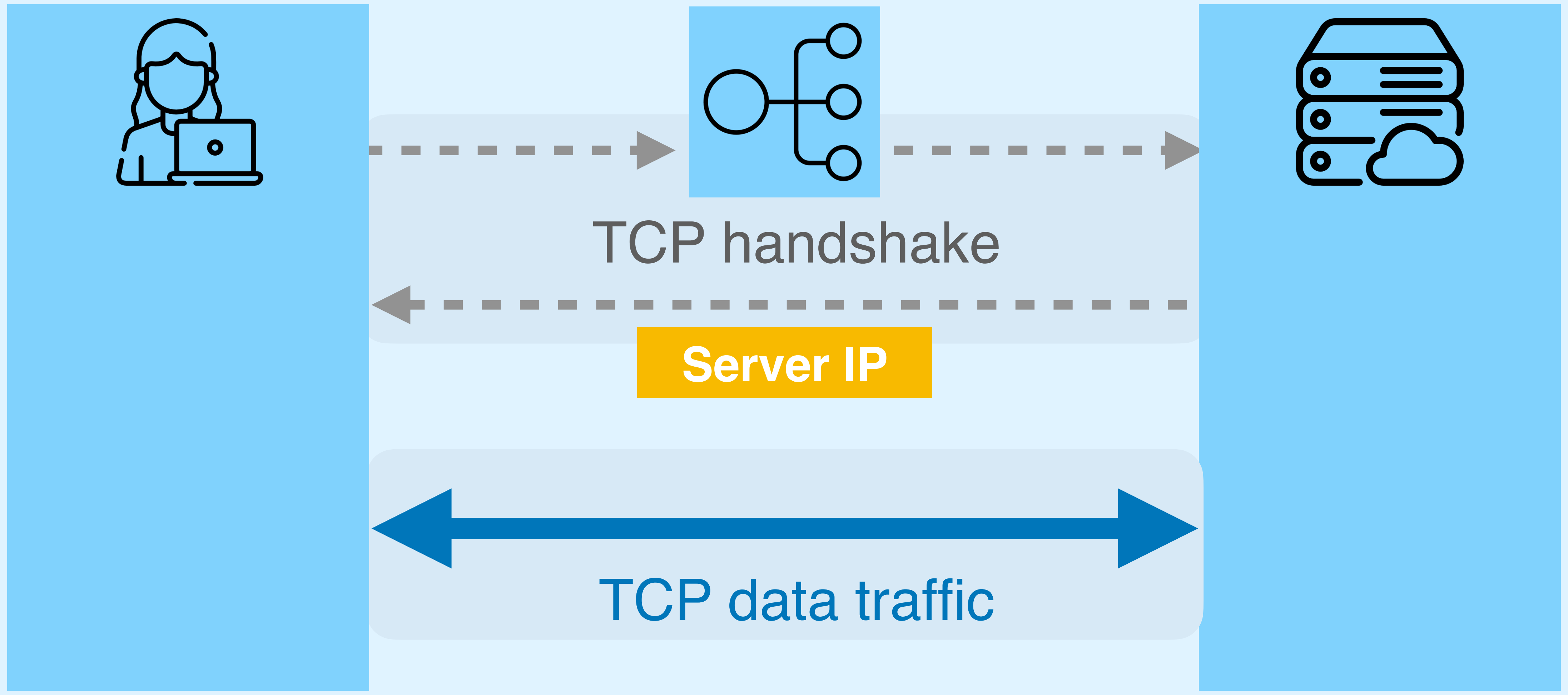
L4 load balancers: best of both worlds

Efficiency ✓

Scalability ✓

CRAB

2020



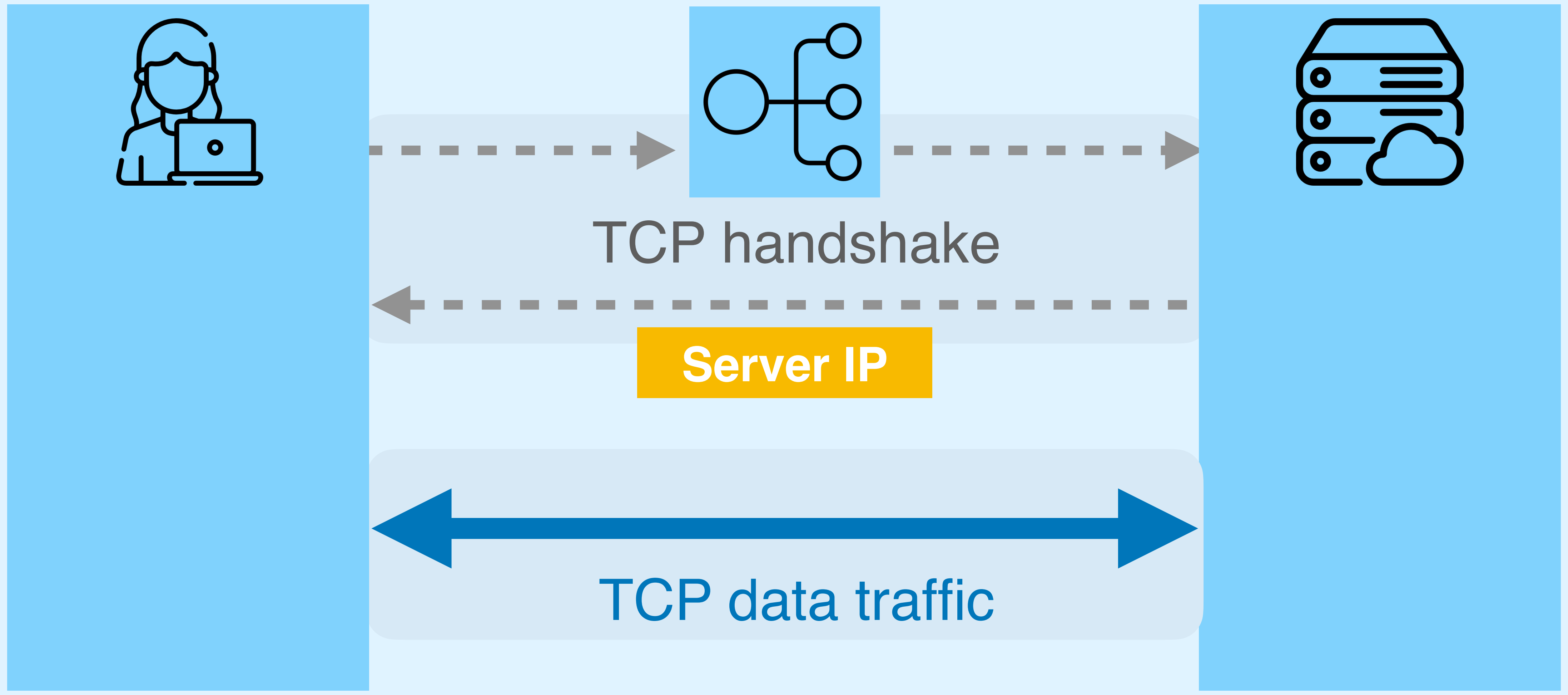
L4 load balancers: best of both worlds

Efficiency ✓

Scalability ✓

CRAB

2020

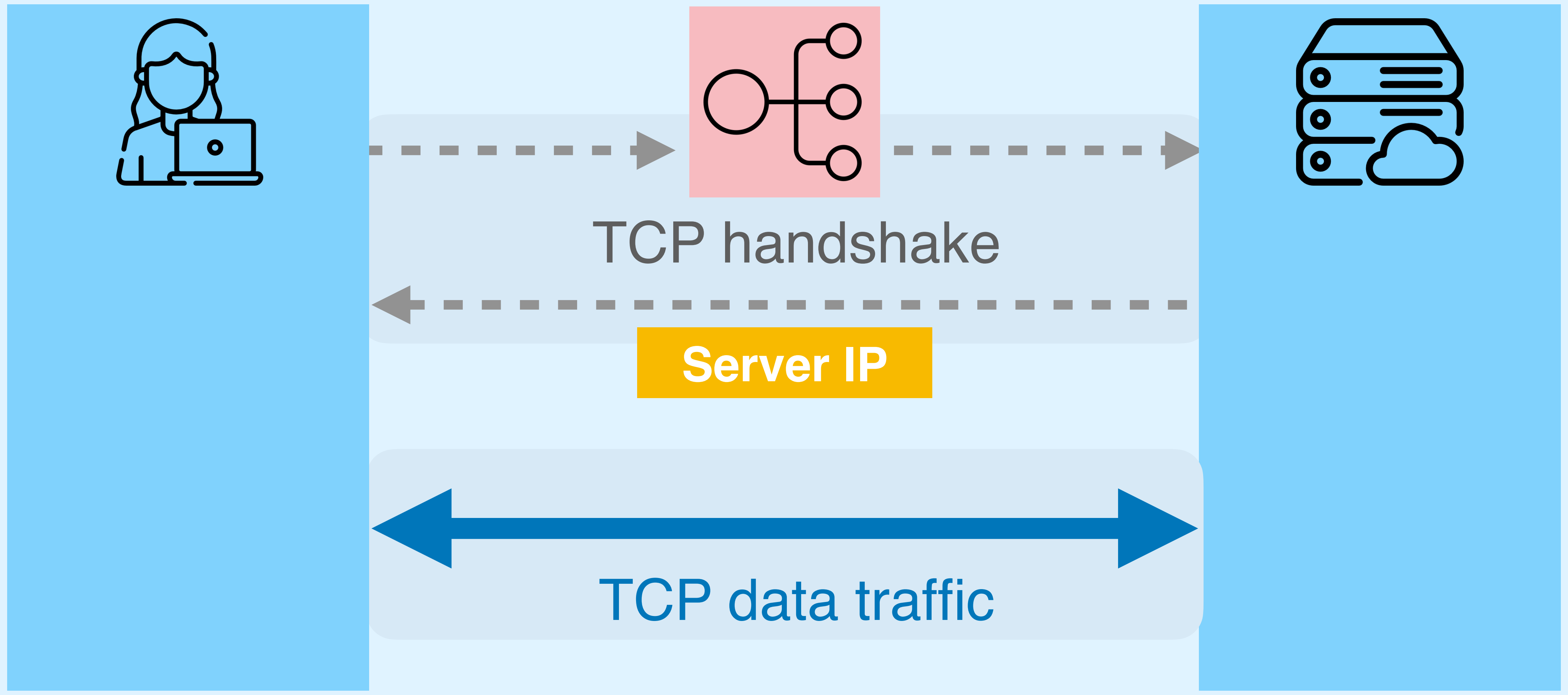


CRAB is designed for the ***internal*** cloud workloads

Poor deployability

CRAB

2020



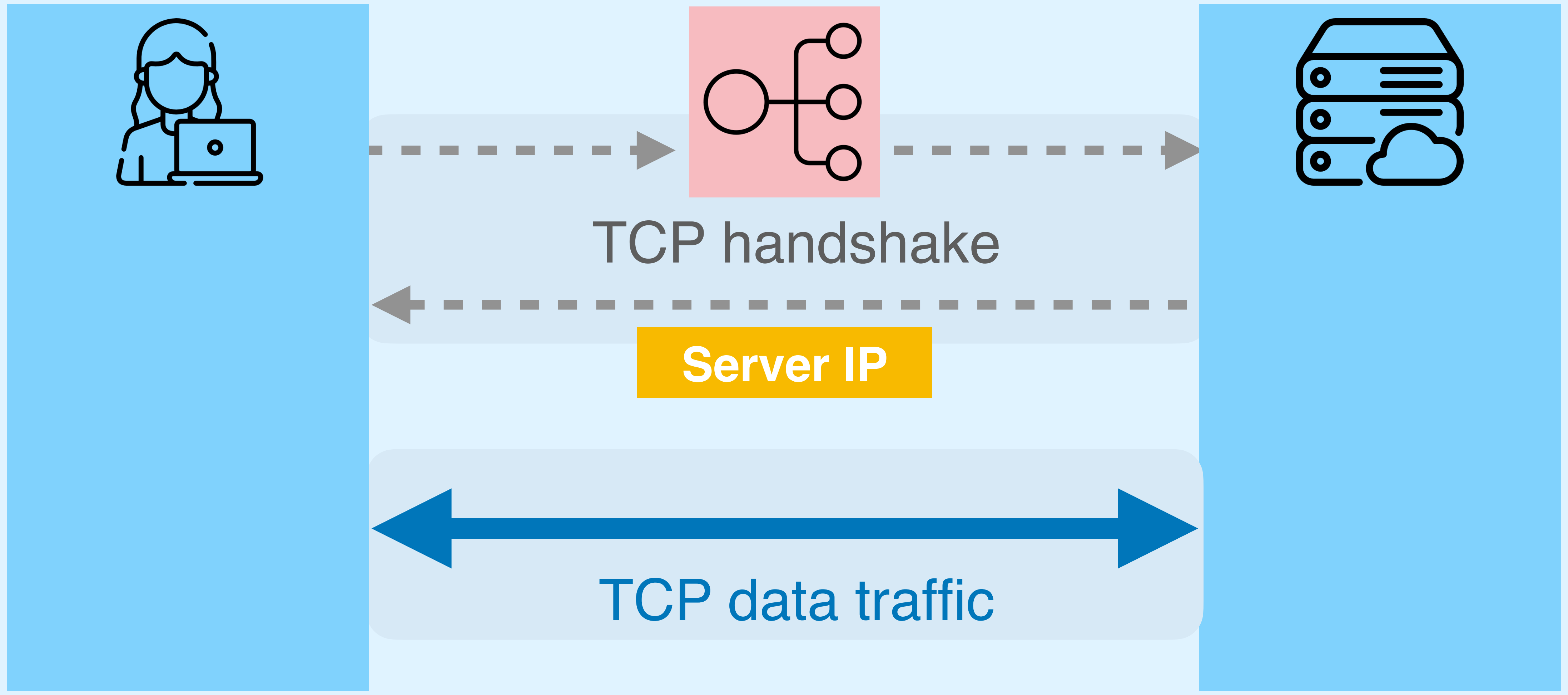
Poor deployability

Requires a customized load balancer

incompatible with real-world ones

CRAB

2020



CRAB

2020

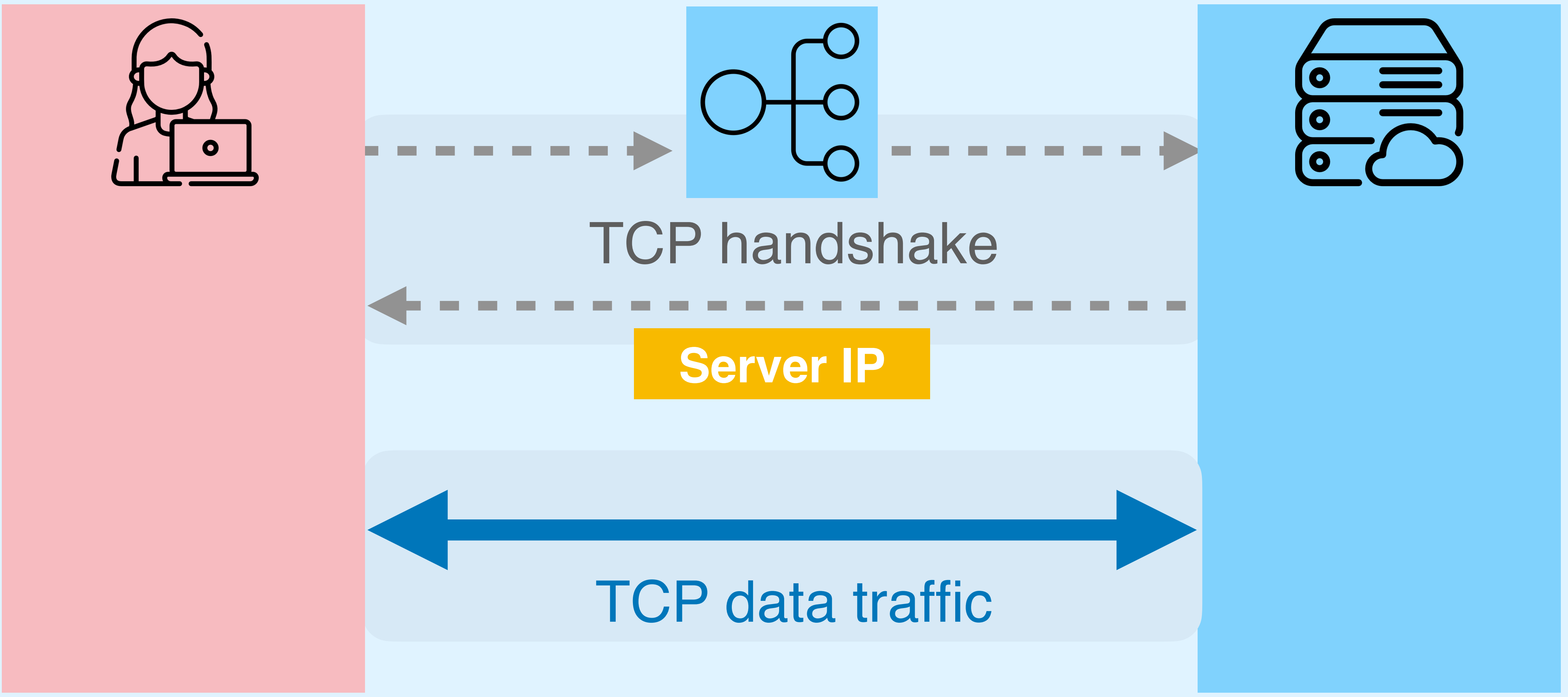
Poor deployability

Requires a customized load balancer

incompatible with real-world ones

Requires kernel changes at client side

through direct kernel patching or module loading



Comparison of existing L4 load balancers

	Centralized Designs	Decentralized Designs	CRAB 2020
Efficiency	✓	✗	✓
Scalability	✗	✓	✓
Deployability	✓	✓	✗

Comparison of existing L4 load balancers

	Centralized Designs	Decentralized Designs	CRAB 2020	HEELS
Efficiency	✓	✗	✓	✓
Scalability	✗	✓	✓	✓
Deployability	✓	✓	✗	✓

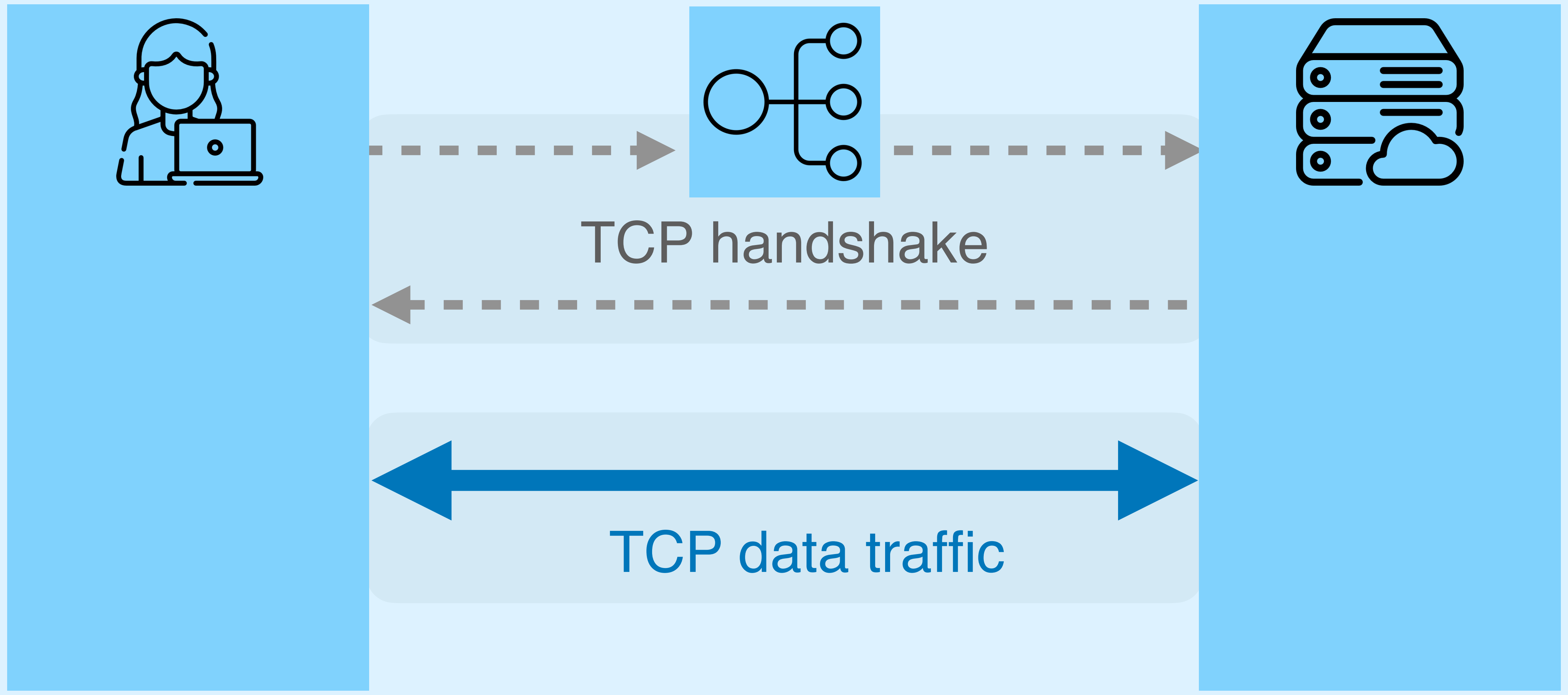
HEELS bypasses the centralized load balancer

Efficiency ✓

Scalability ✓

HEELS: A Host-Enabled eBPF-Based Load Balancing Scheme

2023



HEELS is also designed for the ***internal*** cloud workloads

16

HEELS is readily deployable on the public cloud

Deployability ✓

Compatible with a wide range of LBs

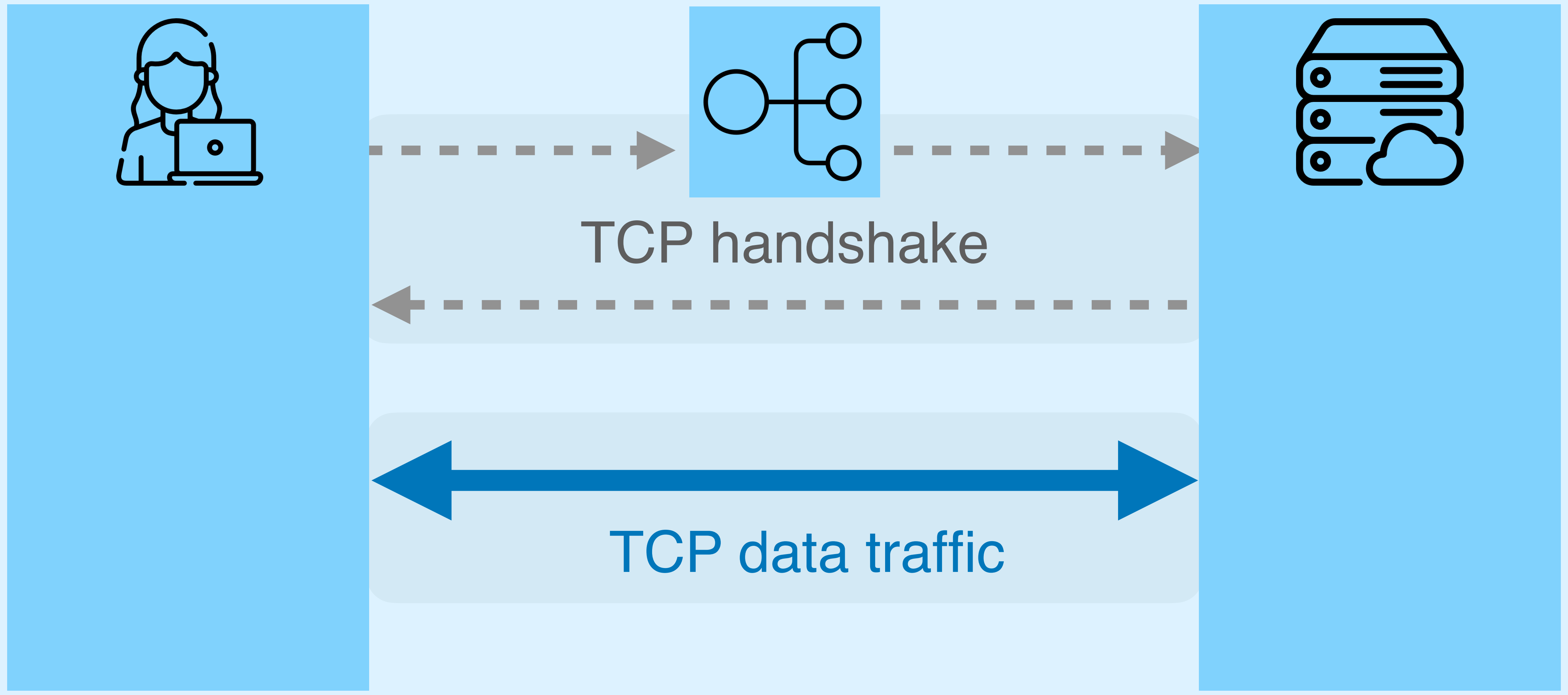
Both open-source and proprietary ones

Requiring no kernel modifications

Leveraging different eBPF hooks

HEELS: A Host-Enabled eBPF-Based Load Balancing Scheme

2023



17

HEELS is readily deployable on the public cloud

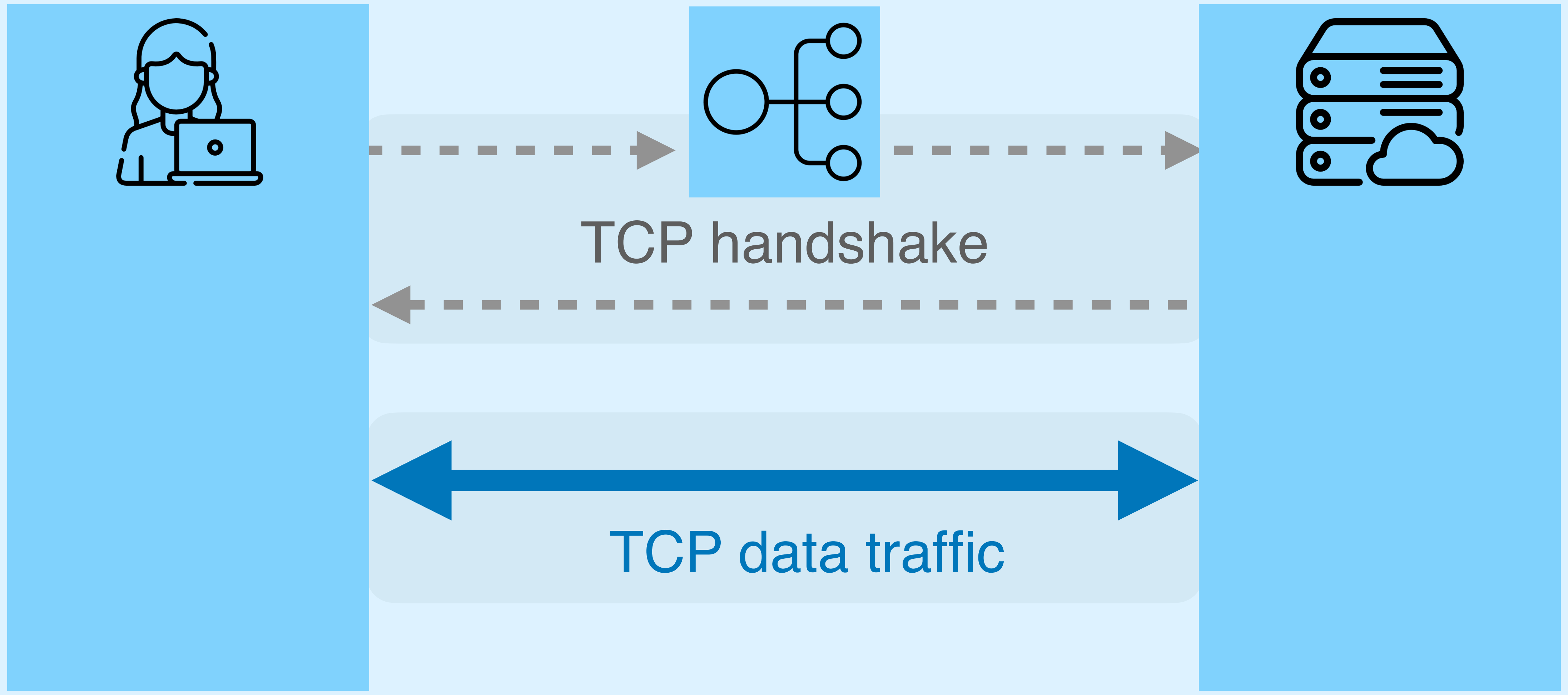
Deployability ✓

Compatible with a wide range of LBs
Both open-source and proprietary ones

Requiring no kernel modifications
Leveraging different eBPF hooks

HEELS: A Host-Enabled eBPF-Based Load Balancing Scheme

2023

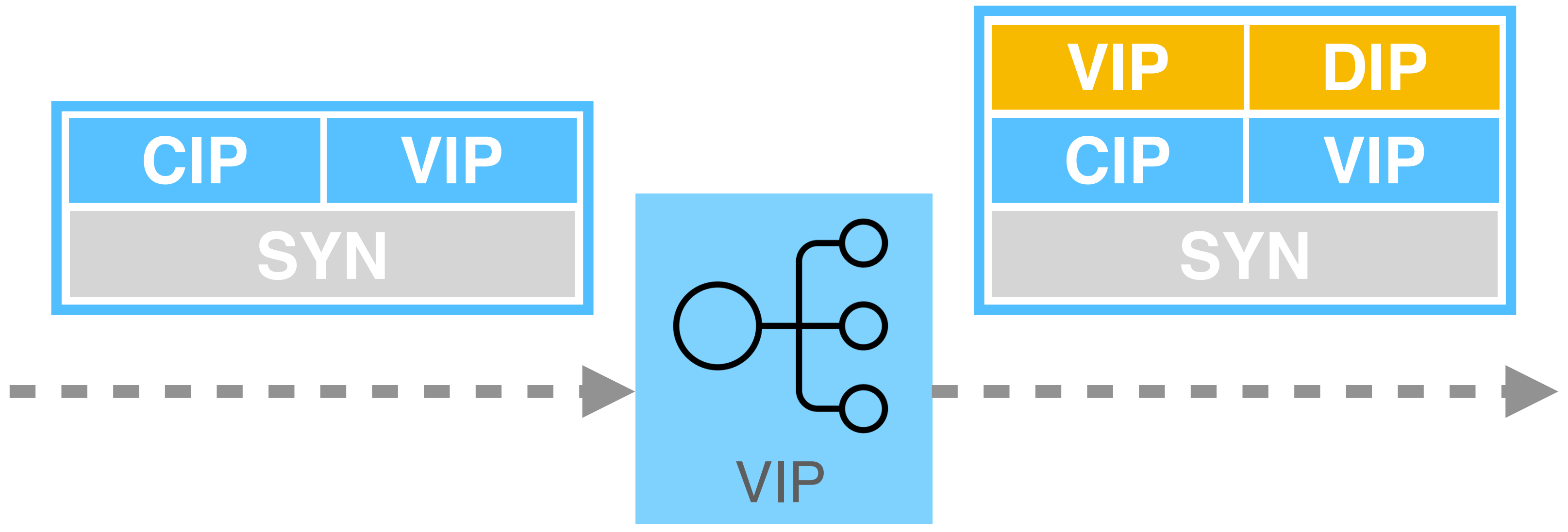


18

Different mechanisms of L4 load balancers

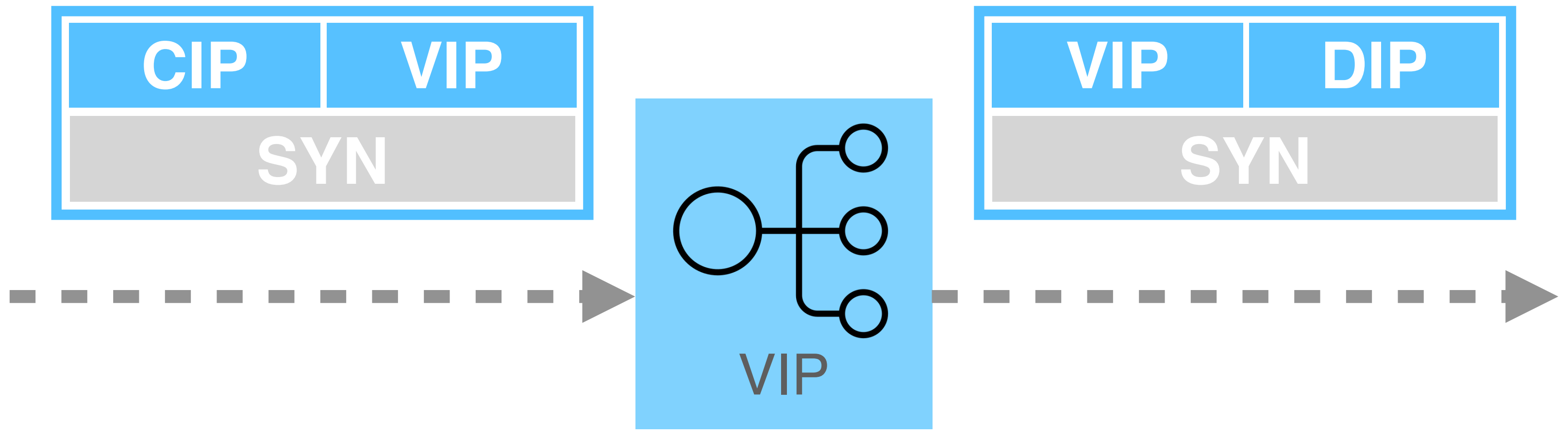
Packet-encapsulation LB

Katran from Meta



Packet-rewriting LB

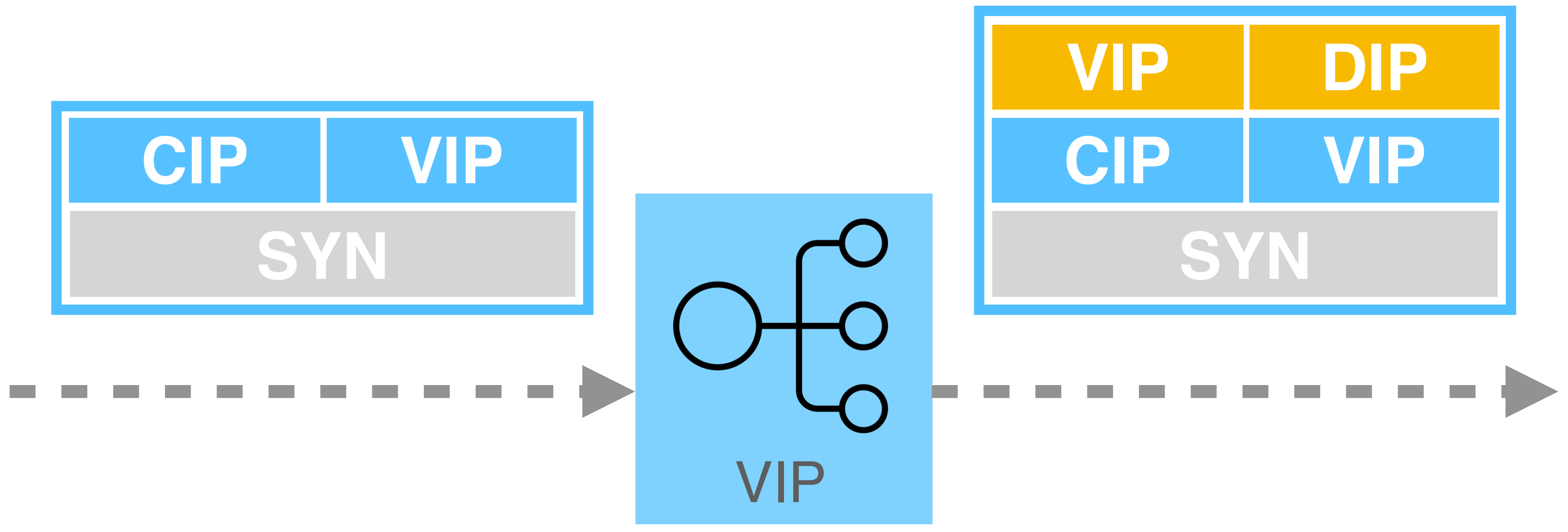
AWS Network Load Balancer



Different mechanisms of L4 load balancers

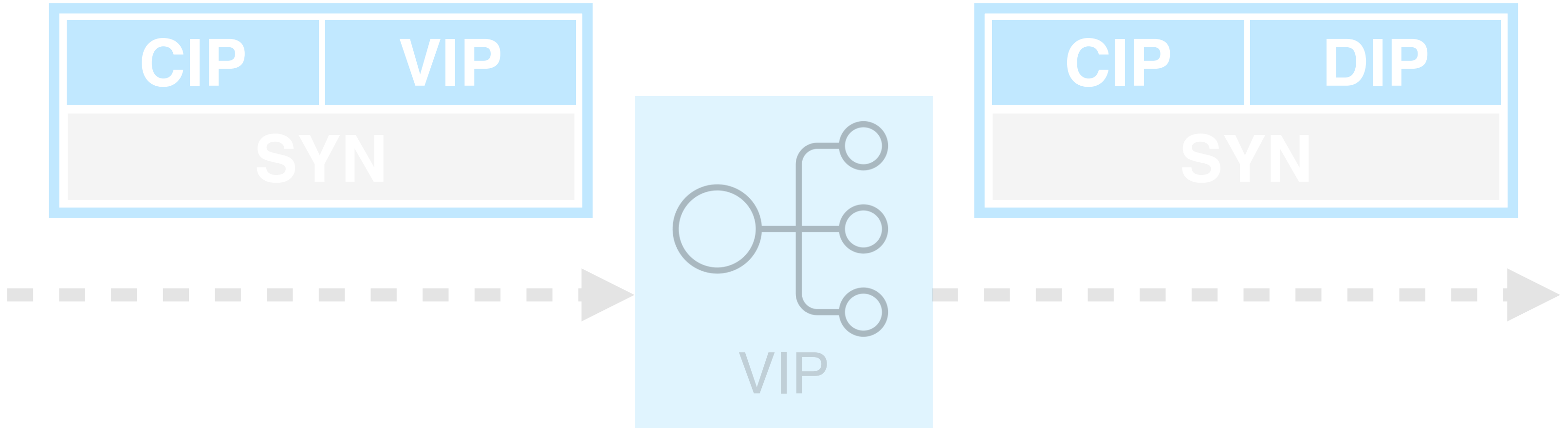
Packet-encapsulation LB

Katran from Meta



Packet-rewriting LB

AWS Network Load Balancer

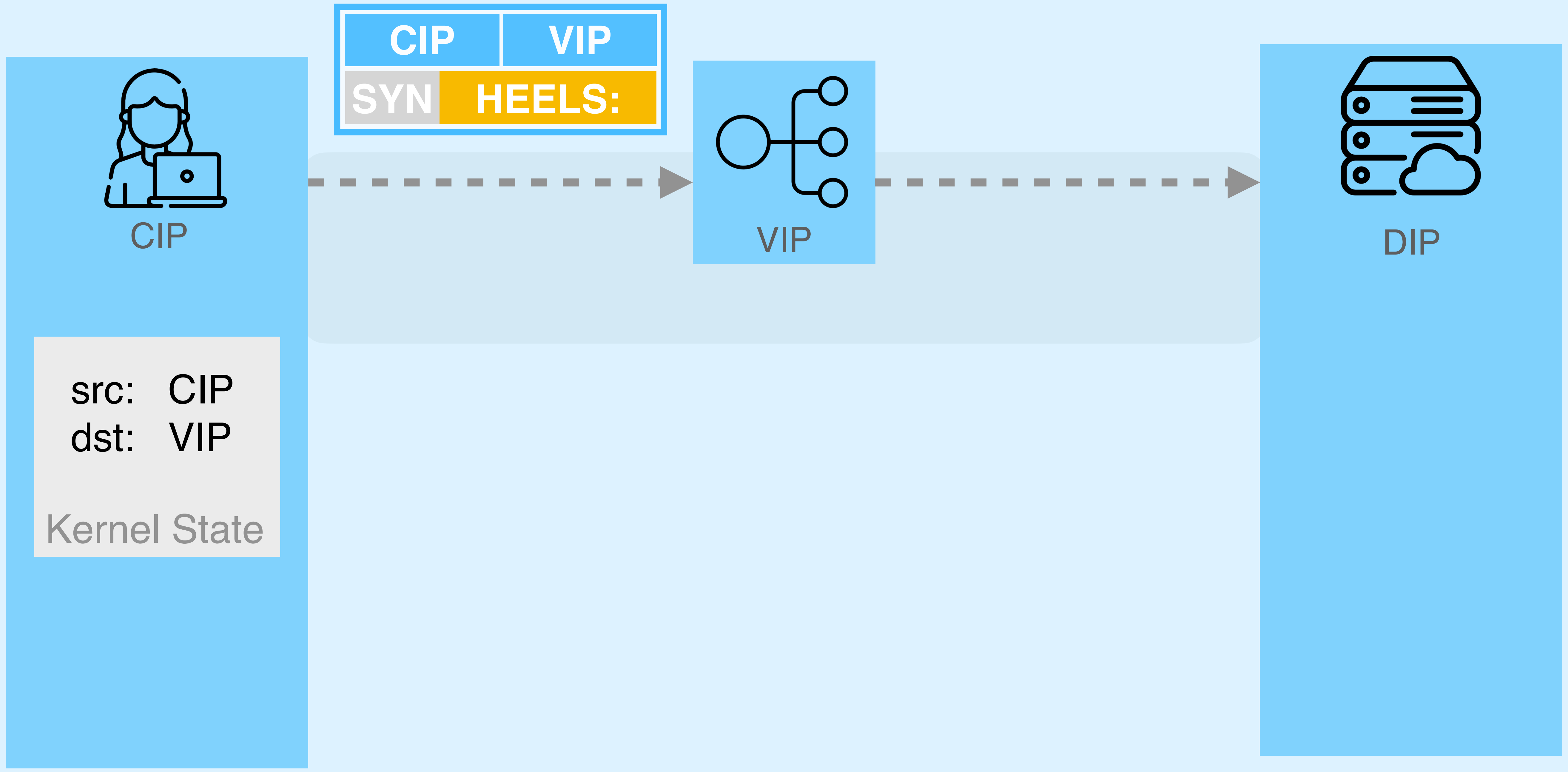


HEELS is compatible with a wide range of LBs

CIP	Client IP
VIP	Load Balancer IP
DIP	Server IP

HEELS: A Host-Enabled eBPF-Based Load Balancing Scheme

2023

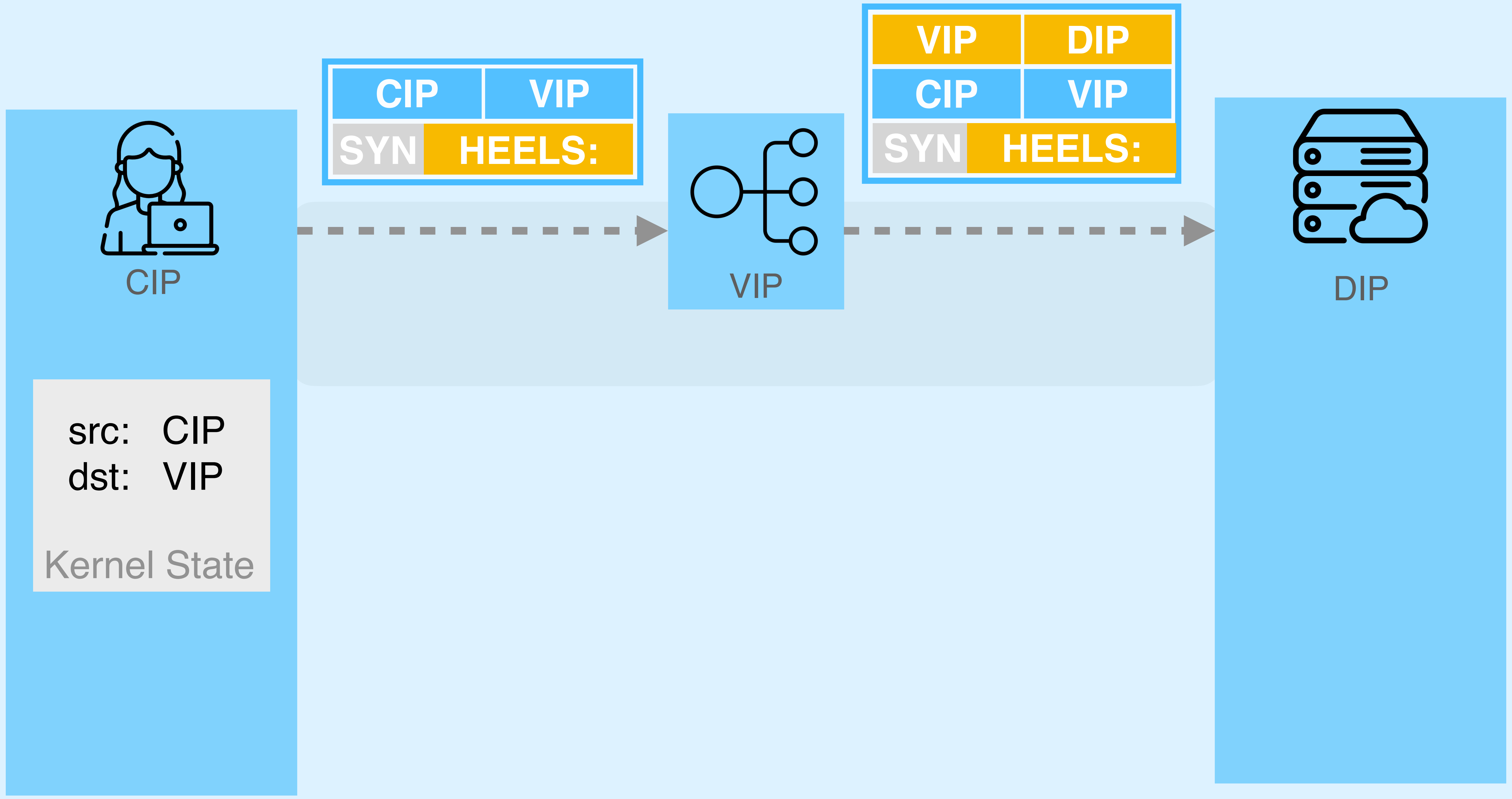


HEELS relies on a customized TCP option

HEELS is compatible with a wide range of LBs

HEELS: A Host-Enabled eBPF-Based Load Balancing Scheme

2023

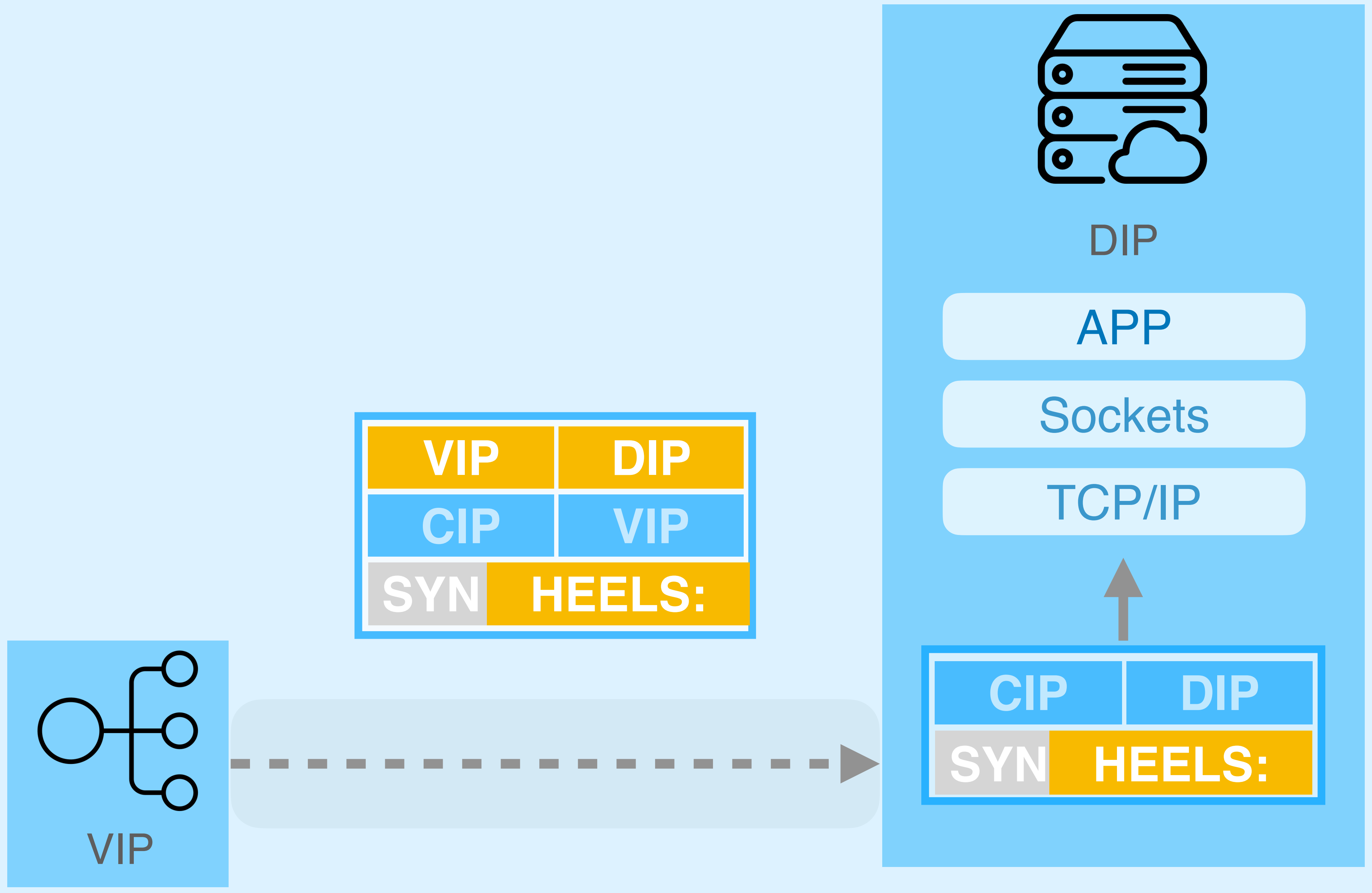


HEELS requires no modifications to the load balancer itself

HEELS is compatible with a wide range of LBs

HEELS: A Host-Enabled eBPF-Based Load Balancing Scheme

2023

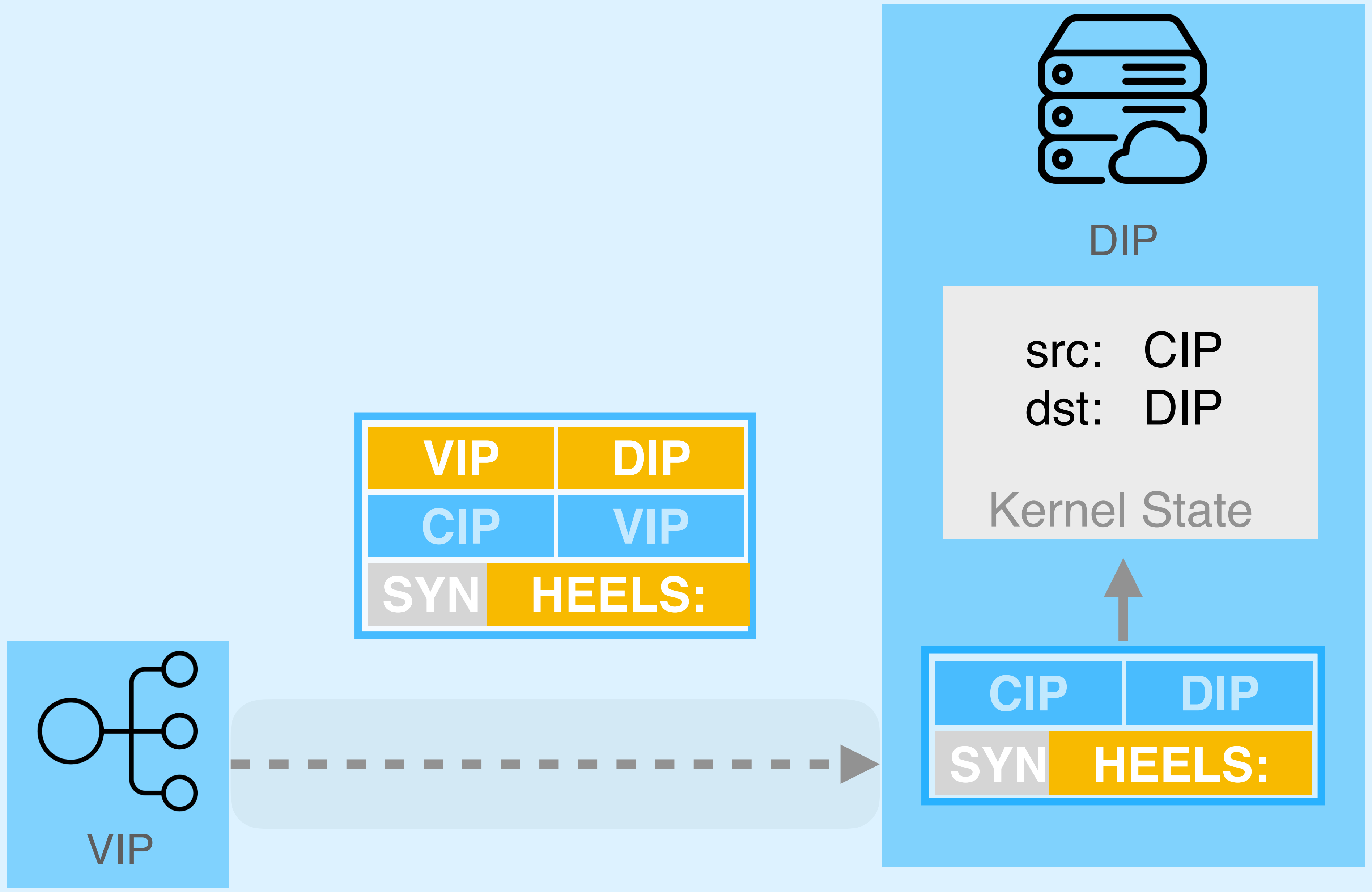


The server modifies the incoming SYN packet ***before*** TCP/IP stack

HEELS is compatible with a wide range of LBs

HEELS: A Host-Enabled eBPF-Based Load Balancing Scheme

2023

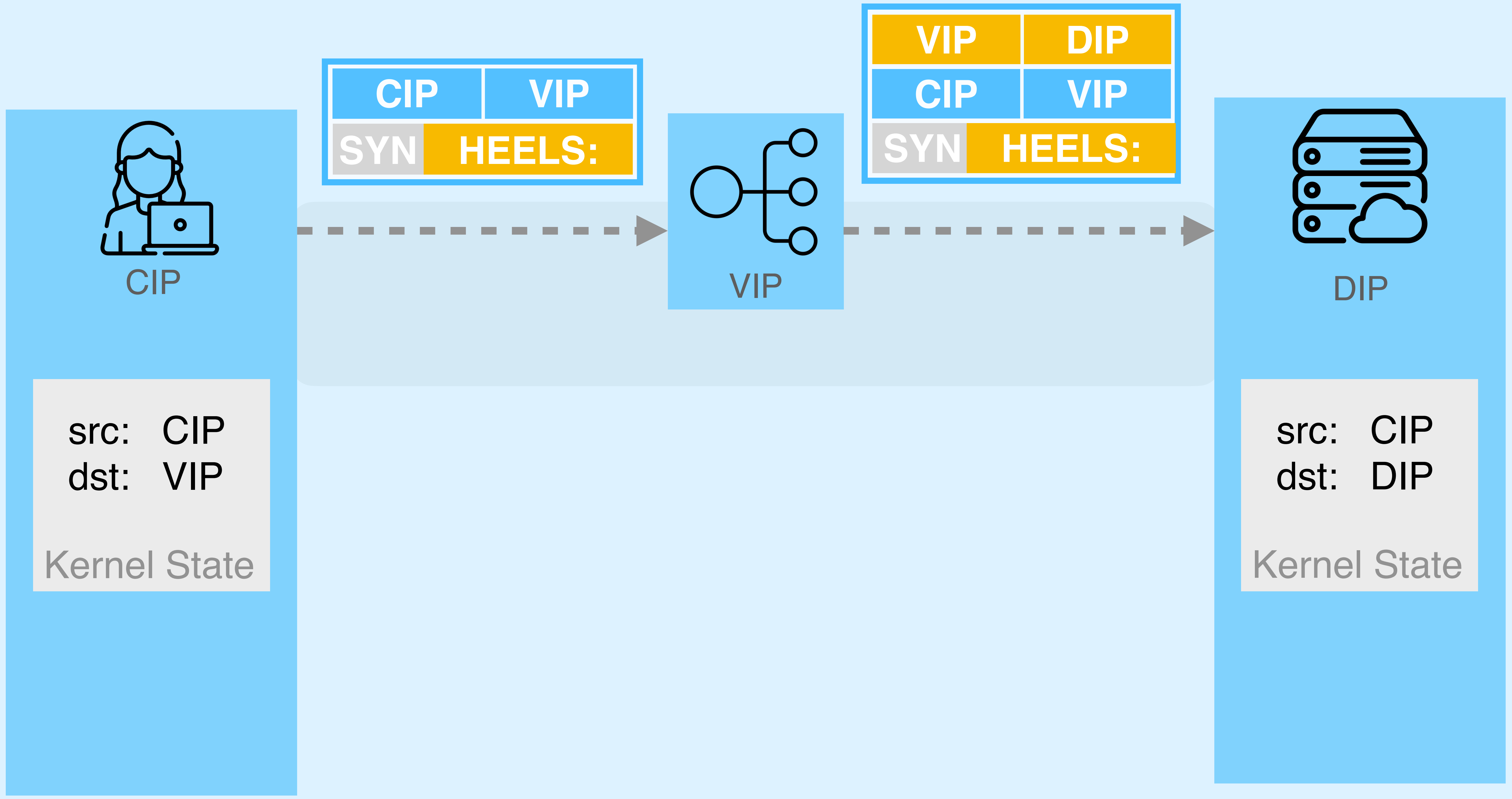


The server modifies the incoming SYN packet ***before*** TCP/IP stack

HEELS is compatible with a wide range of LBs

HEELS: A Host-Enabled eBPF-Based Load Balancing Scheme

2023

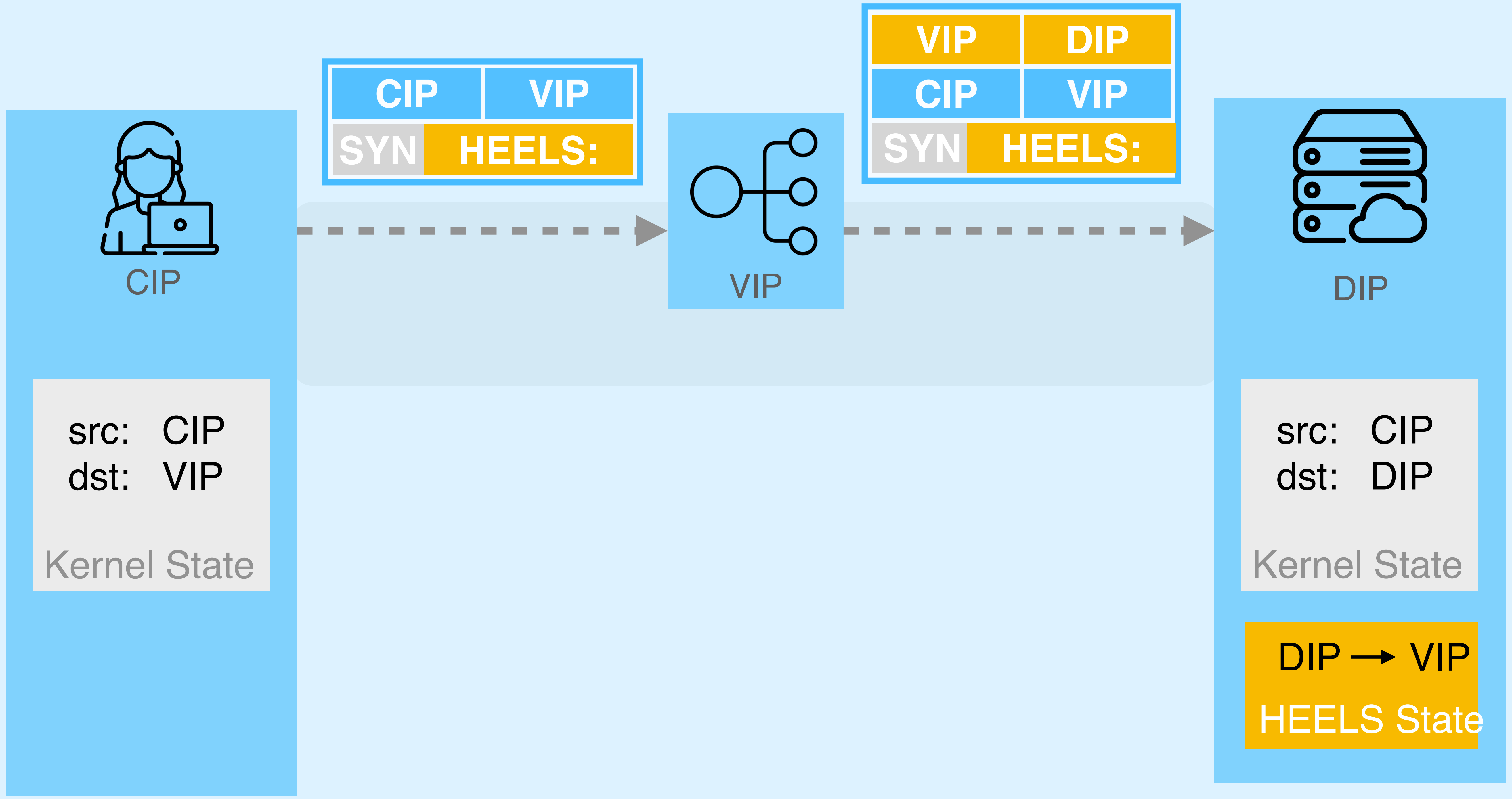


HEELS maintain its own state for TCP connections

HEELS is compatible with a wide range of LBs

HEELS: A Host-Enabled eBPF-Based Load Balancing Scheme

2023

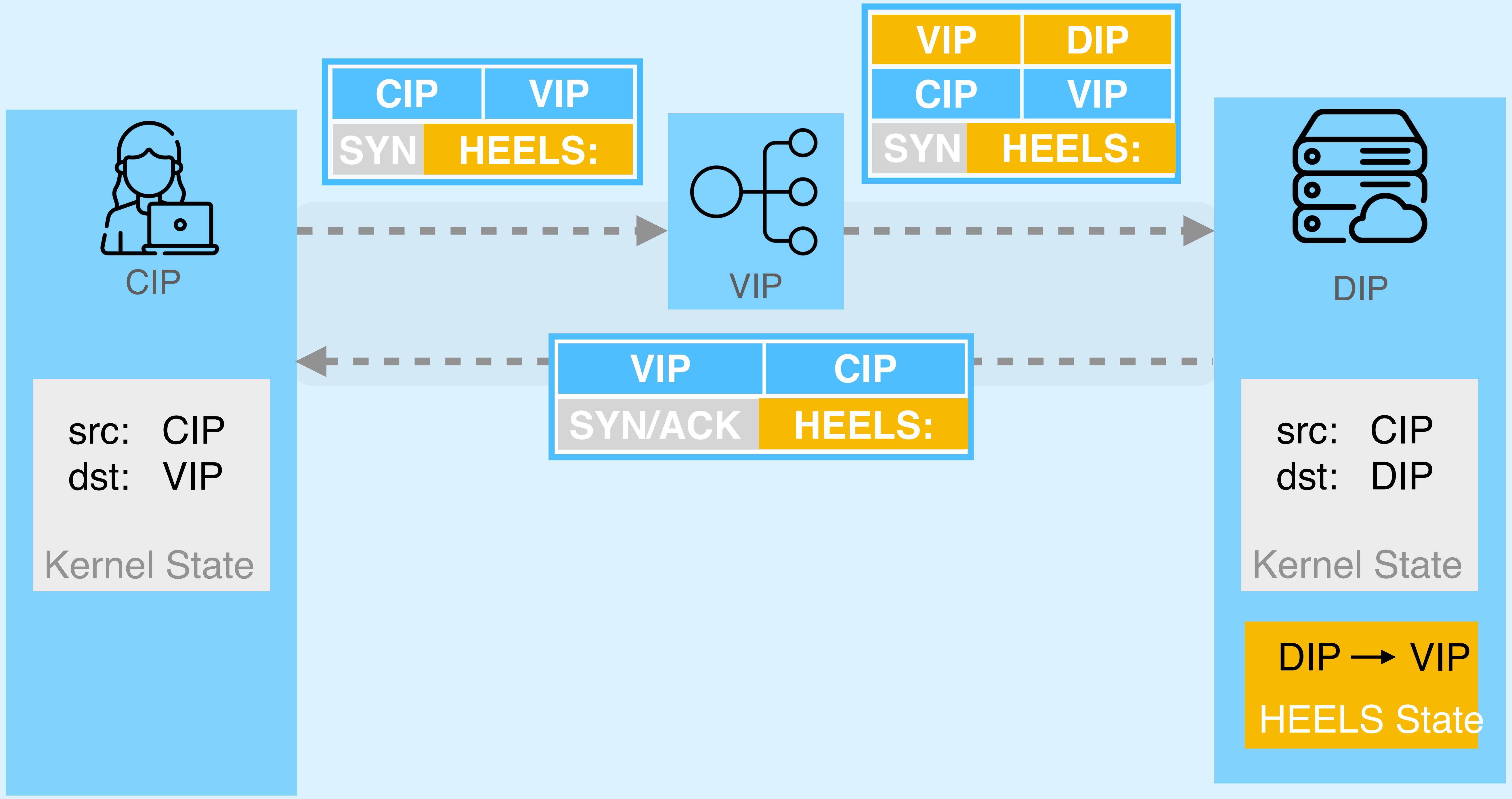


HEELS maintain its own state for TCP connections

HEELS is compatible with a wide range of LBs

HEELS: A Host-Enabled eBPF-Based Load Balancing Scheme

2023

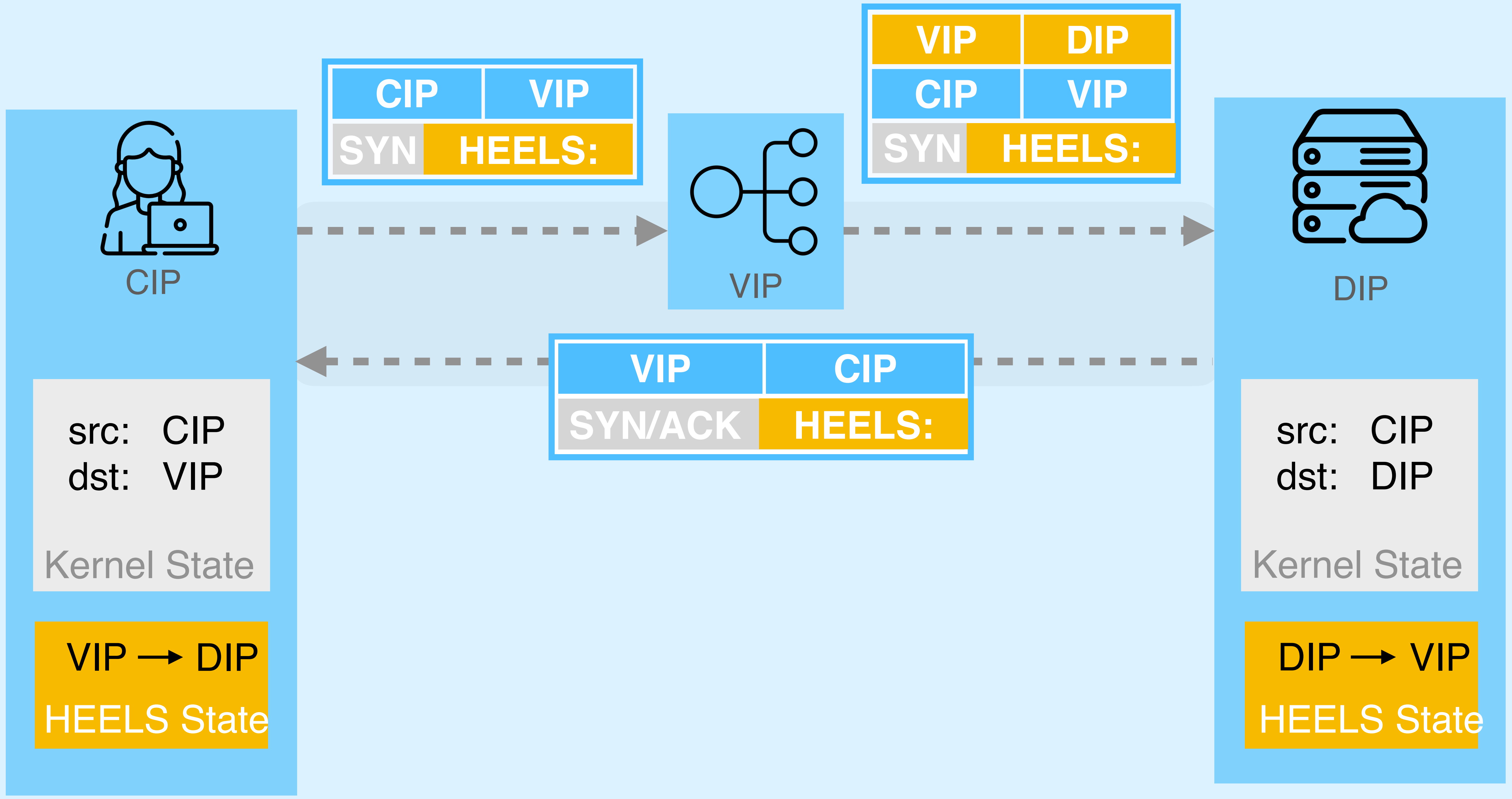


HEELS rewrites ***every*** outgoing packet to match the kernel state of the other end

HEELS is compatible with a wide range of LBs

HEELS: A Host-Enabled eBPF-Based Load Balancing Scheme

2023

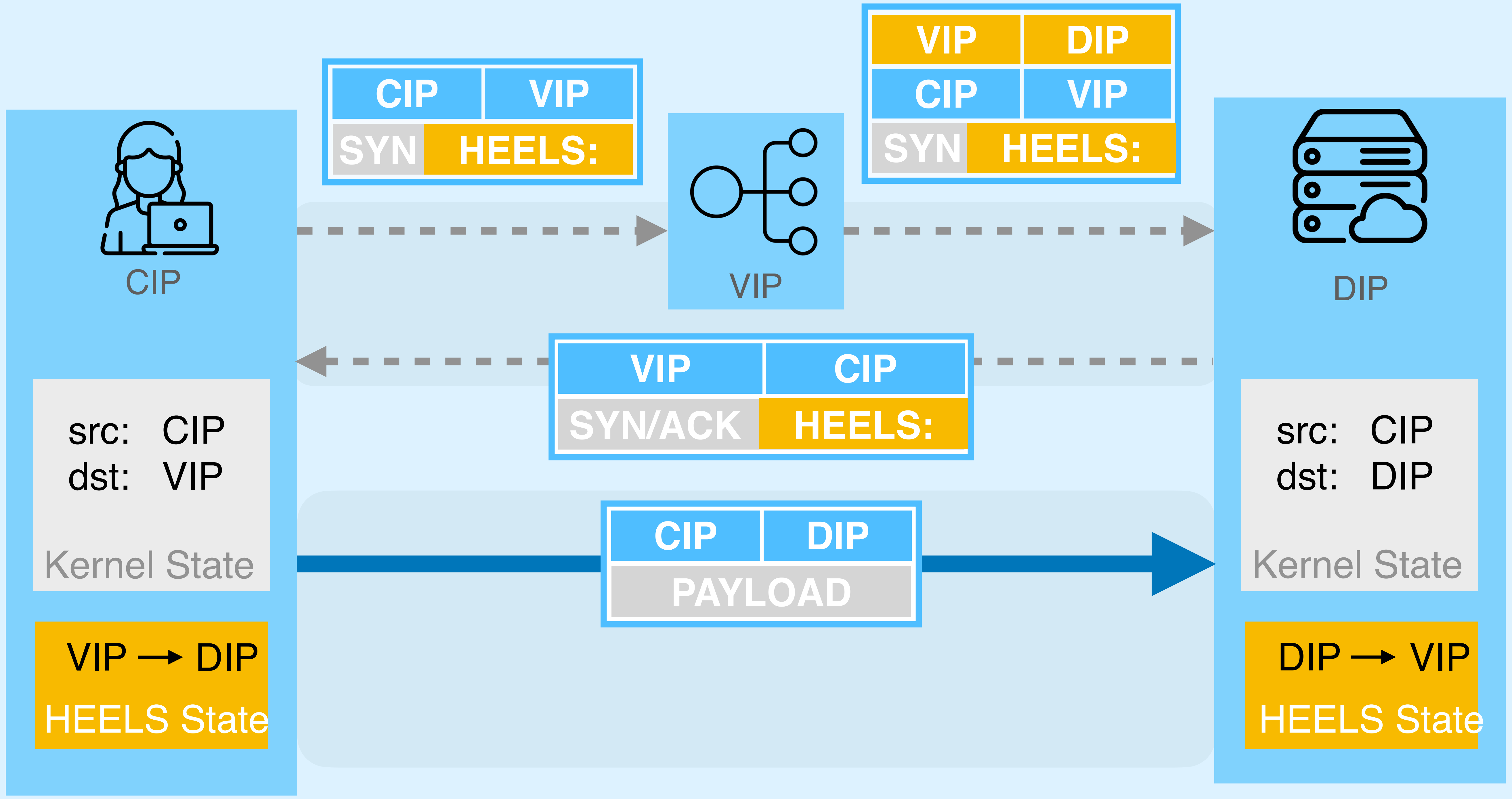


HEELS rewrites ***every*** outgoing packet to match the kernel state of the other end

HEELS is compatible with a wide range of LBs

HEELS: A Host-Enabled eBPF-Based Load Balancing Scheme

2023

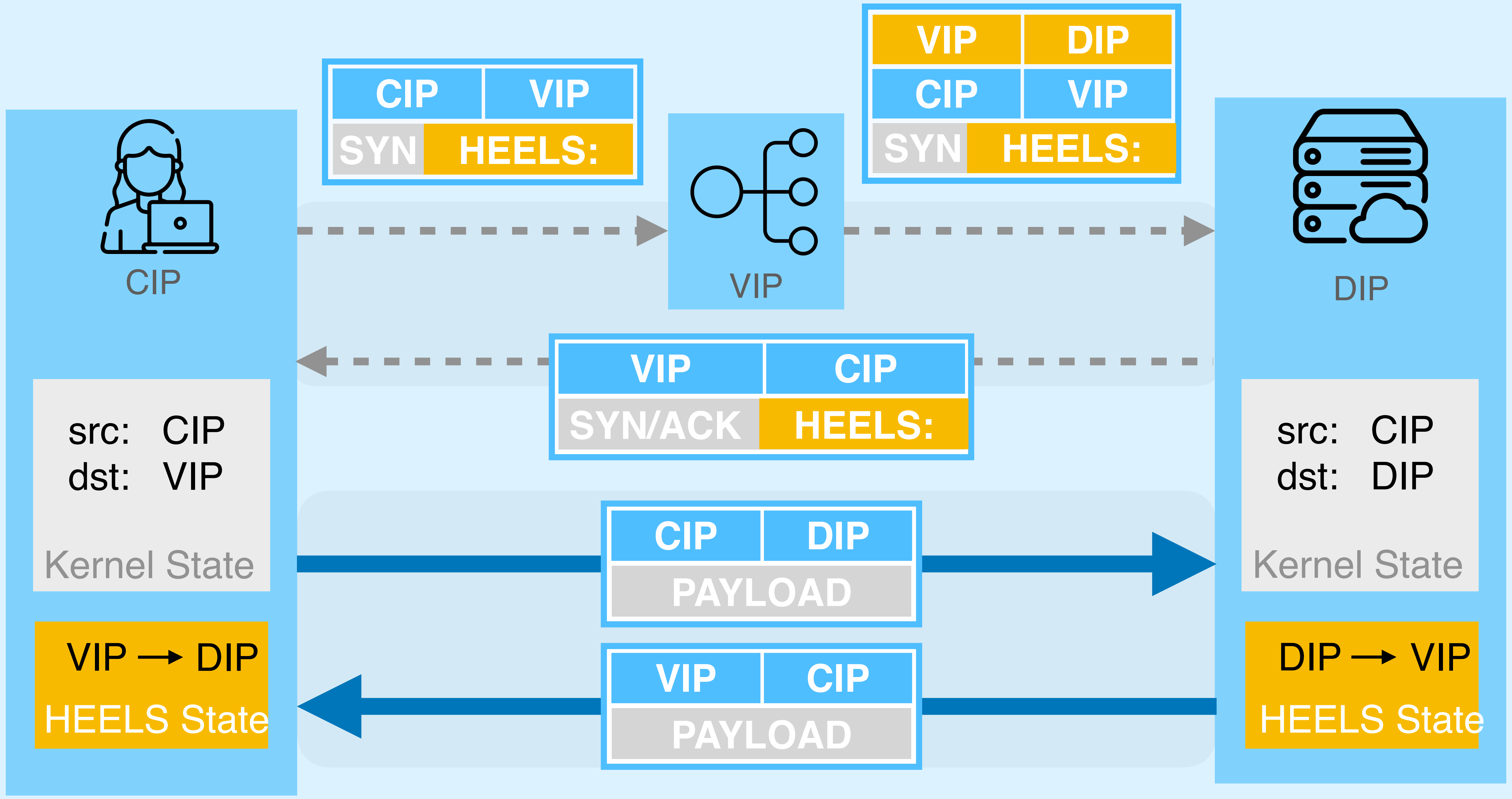


HEELS rewrites ***every*** outgoing packet to match the kernel state of the other end

HEELS is compatible with a wide range of LBs

HEELS: A Host-Enabled eBPF-Based Load Balancing Scheme

2023

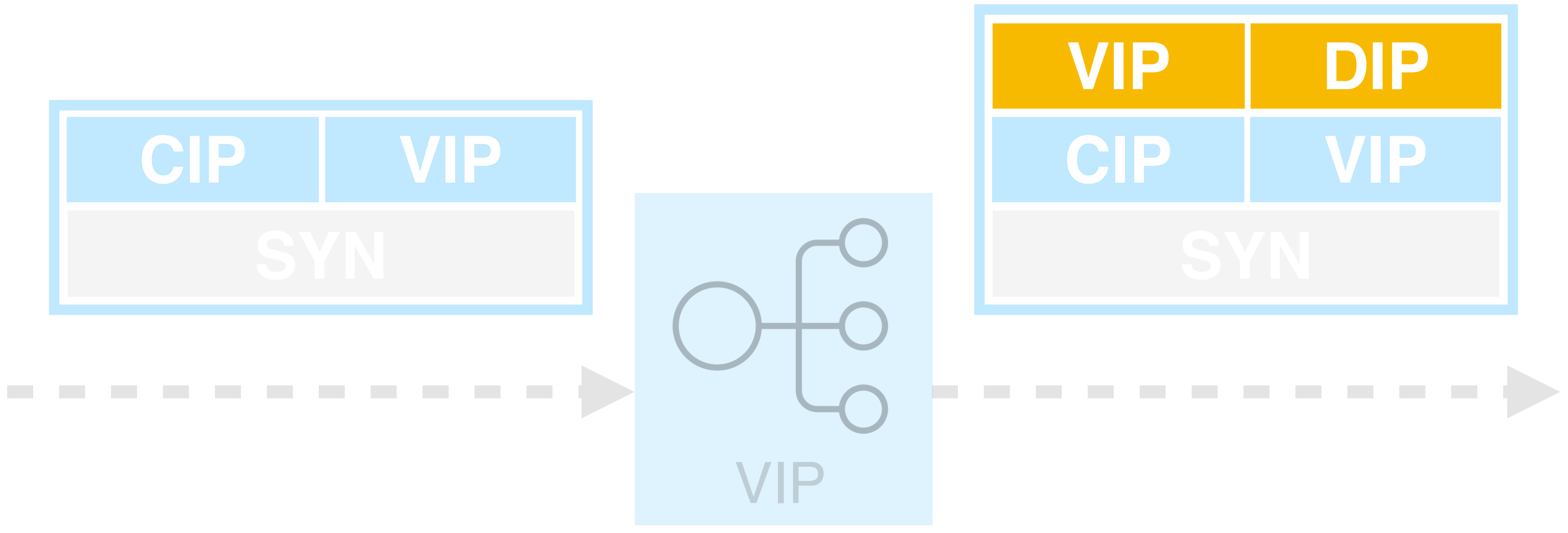


Direct communication after the handshake

Different mechanisms of L4 load balancers

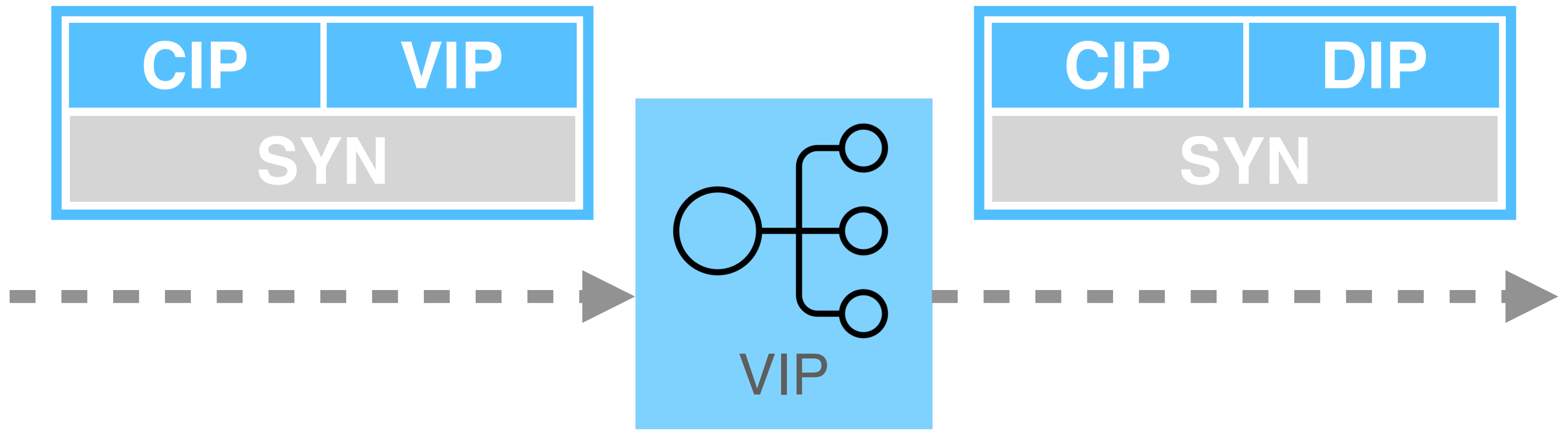
Packet-encapsulation LB

Katran from Meta



Packet-rewriting LB

AWS Network Load Balancer



HEELS is readily deployable on the public cloud

Deployability ✓

Compatible with a wide range of LBs

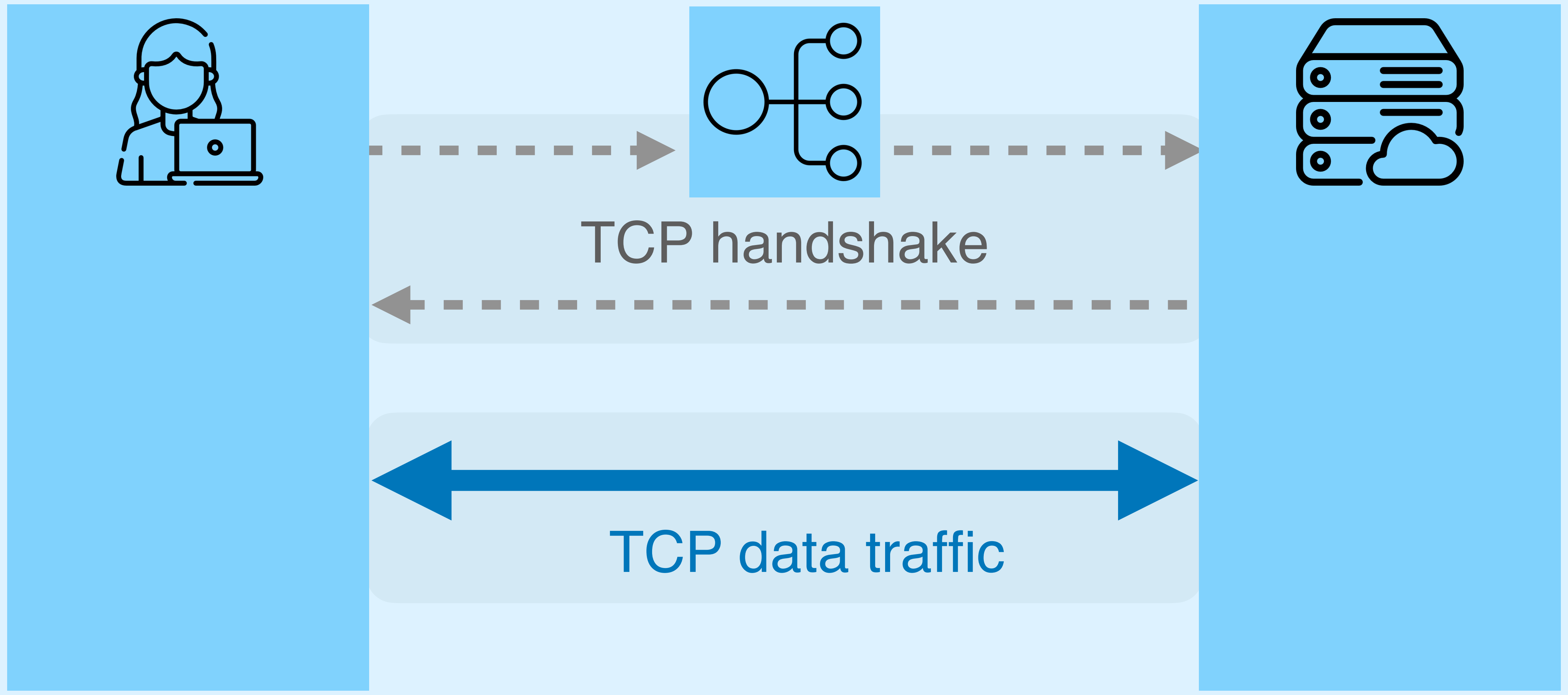
Both open-source and proprietary ones

Requiring no kernel modifications

Leveraging different eBPF hooks

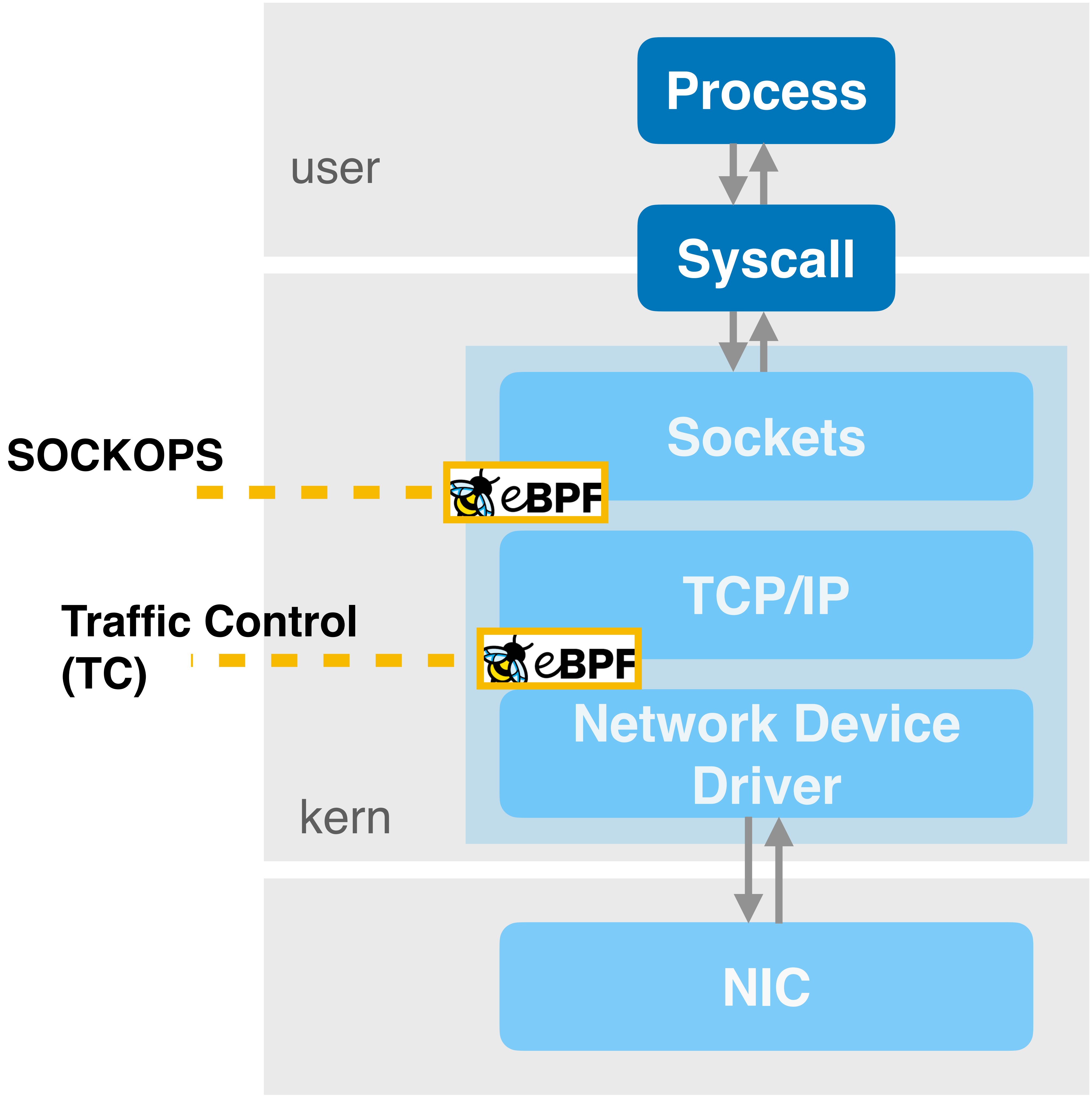
HEELS: A Host-Enabled eBPF-Based Load Balancing Scheme

2023



32

HEELS *implements* its design using a set of eBPF programs

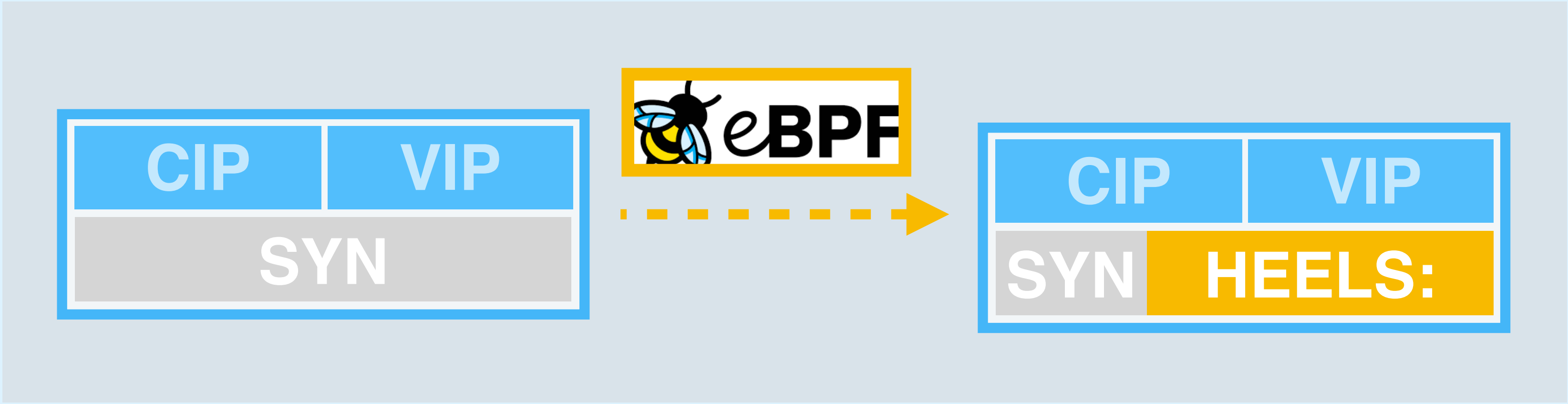


HEELS

2023

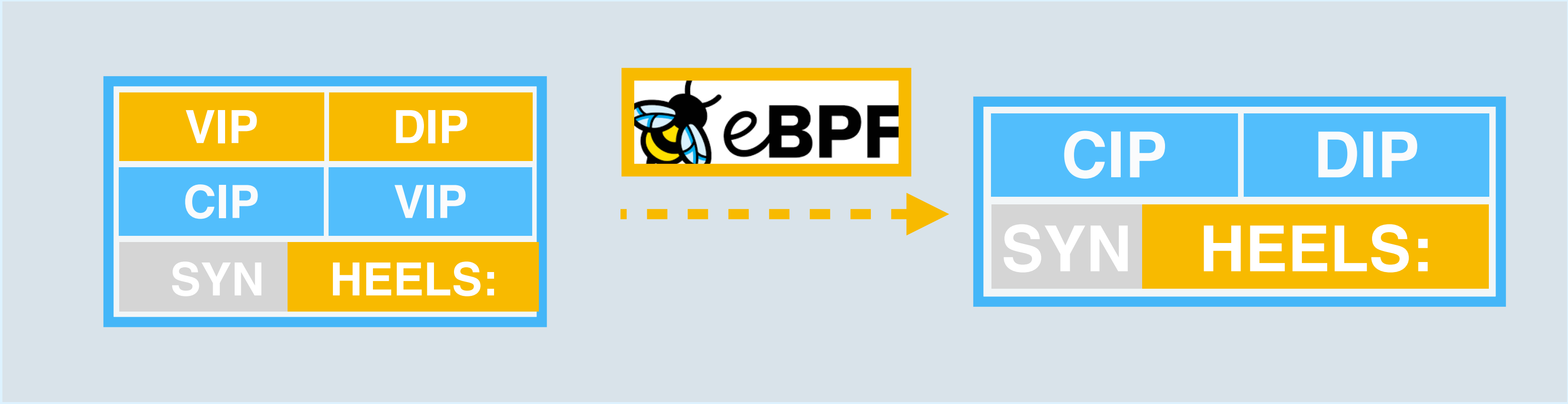
SOCKOPS

Adding and extracting TCP options



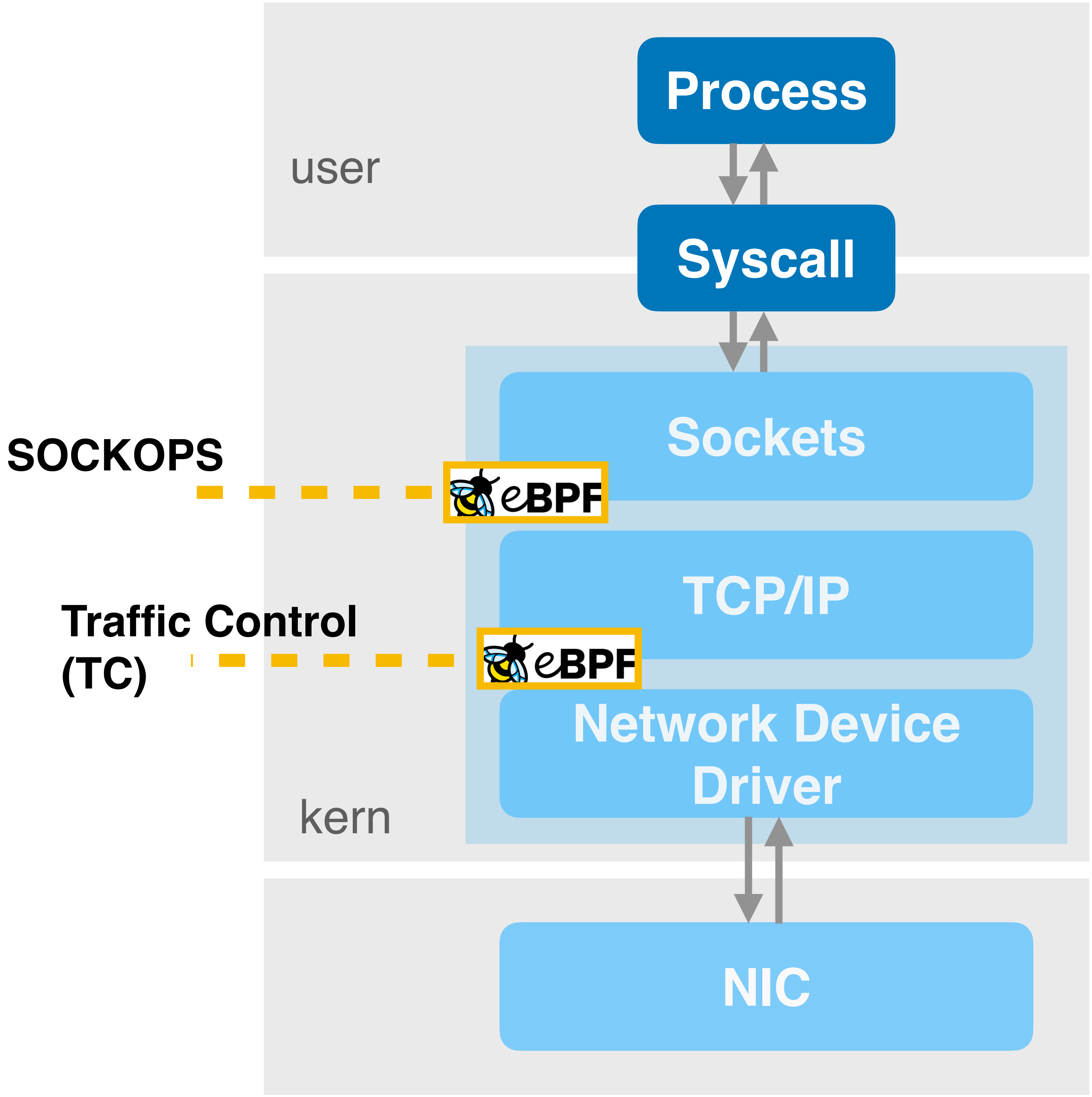
Traffic Control (TC)

Rewriting ingress SYN packet at the server



eBPF programs for handshake phrase

HEELS *implements* its design using a set of eBPF programs

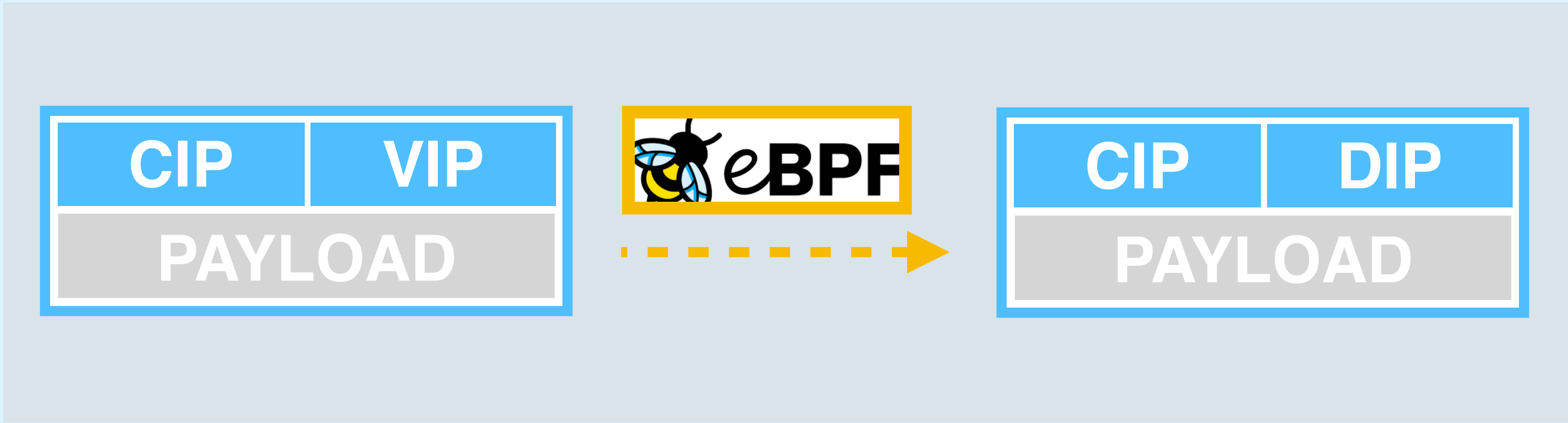


HEELS

2023

Traffic Control (TC)

Rewriting egress packets at end hosts



eBPF programs for data transmission phrase

HEELS *implements* its design using a set of eBPF programs

SK_STORAGE

Storing HEELS state at end hosts

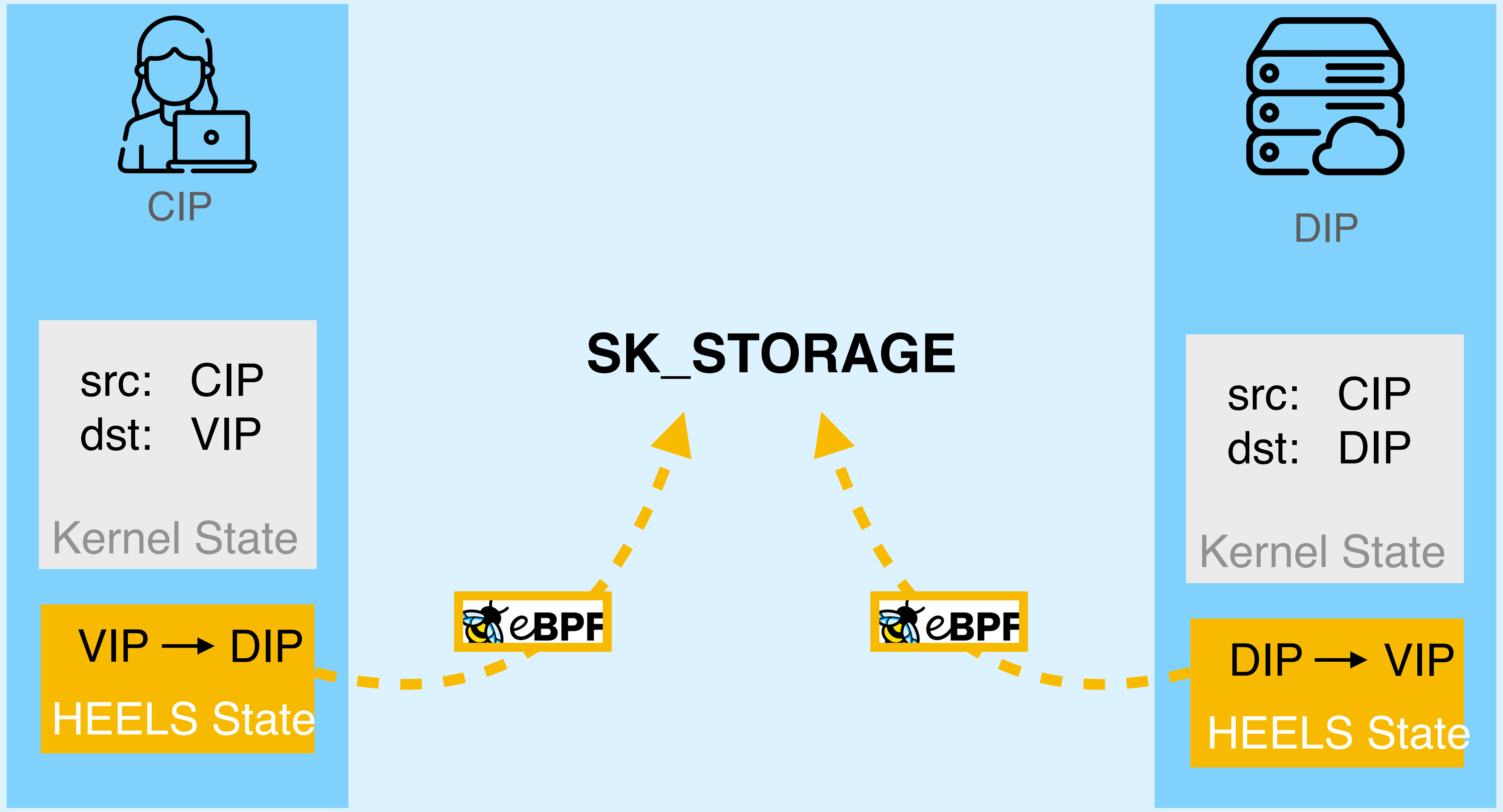
Requires no changes to kernel state

Per-connection eBPF data structure

Same lifetime as the TCP connection

HEELS

2023



Created at TCP handshake phrase and accessed throughout the connection

We evaluate **HEELS** on both local testbed and public cloud

Implementation

~1.2k lines of eBPF code

Supports both Katran and AWS Network Load Balancer (NLB)

Questions

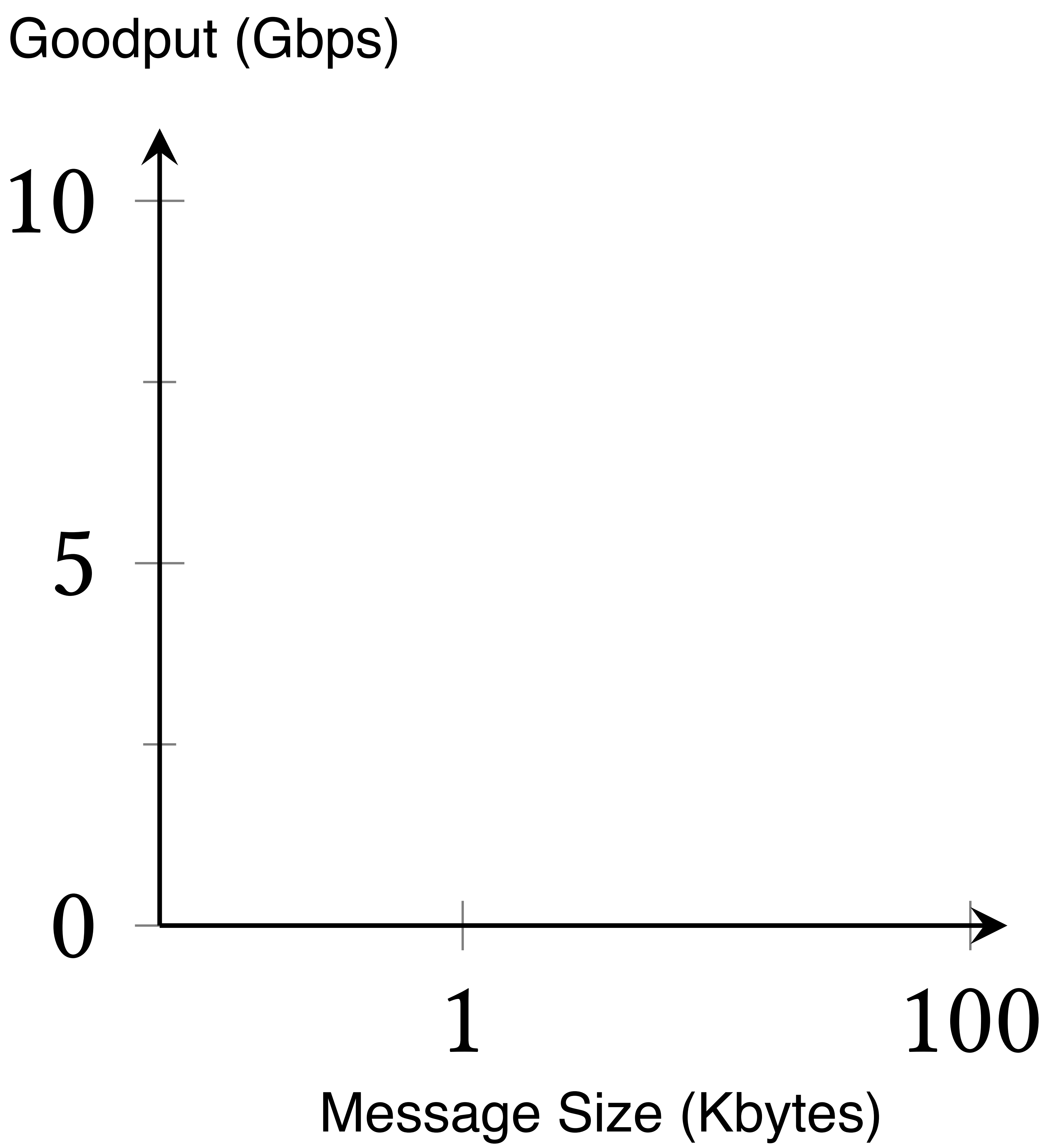
Q1: Does HEELS bring significant overhead?

Deploy with Katran on local testbed

Q2: What benefits does HEELS bring on the cloud?

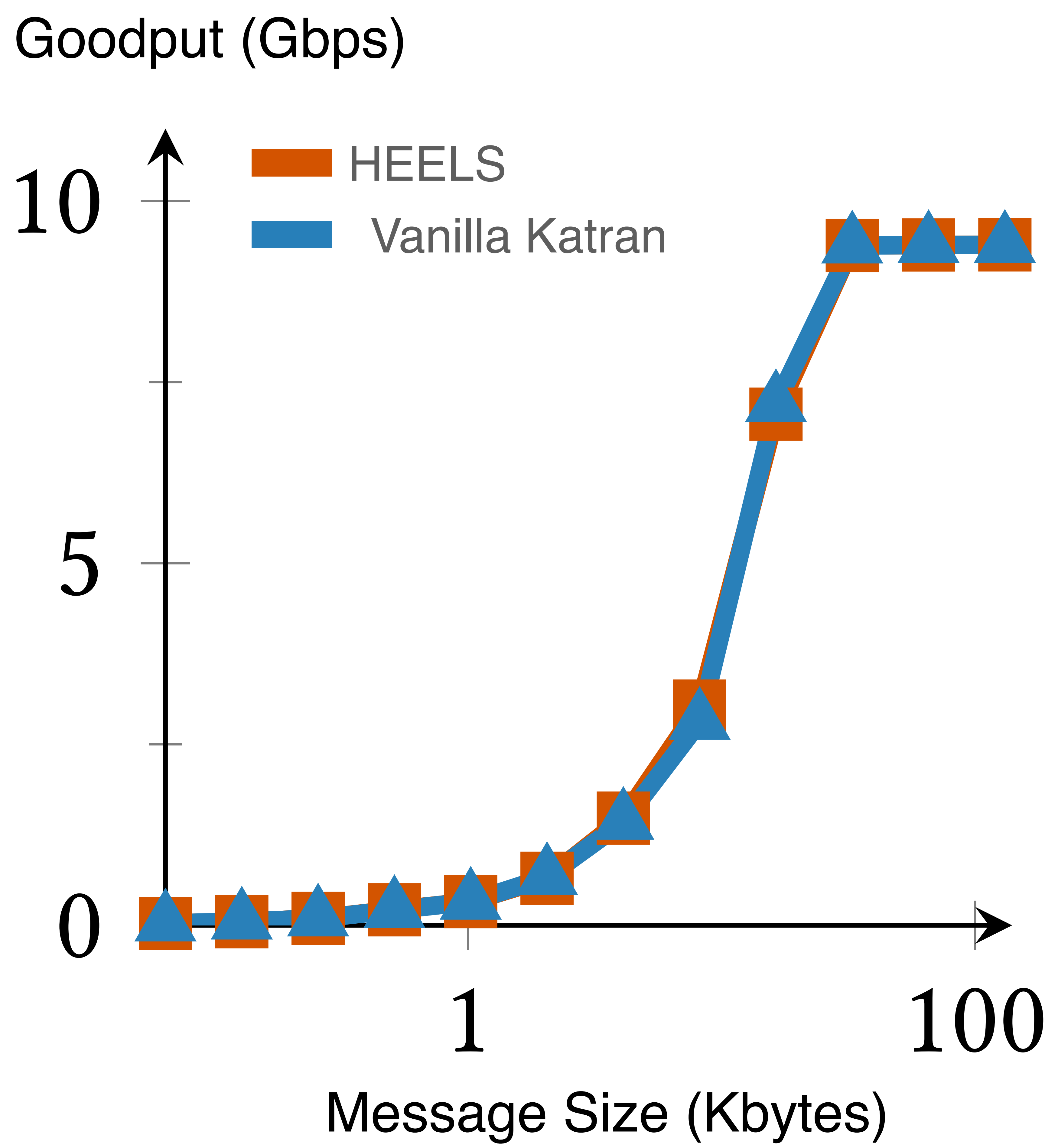
Deploy with AWS NLB on the cloud

HEELS introduces minimal performance overhead



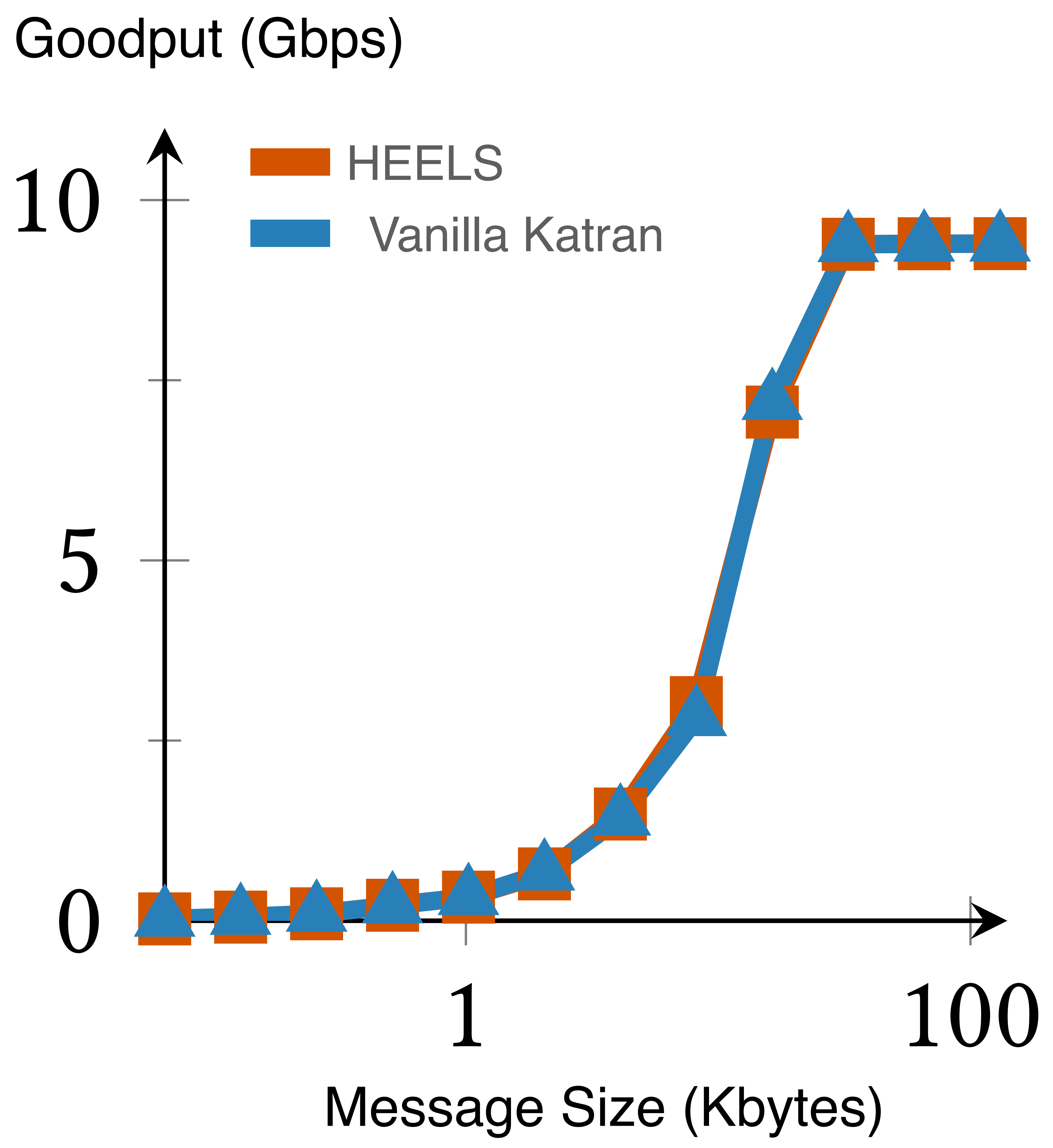
All cores enabled

HEELS introduces minimal performance overhead

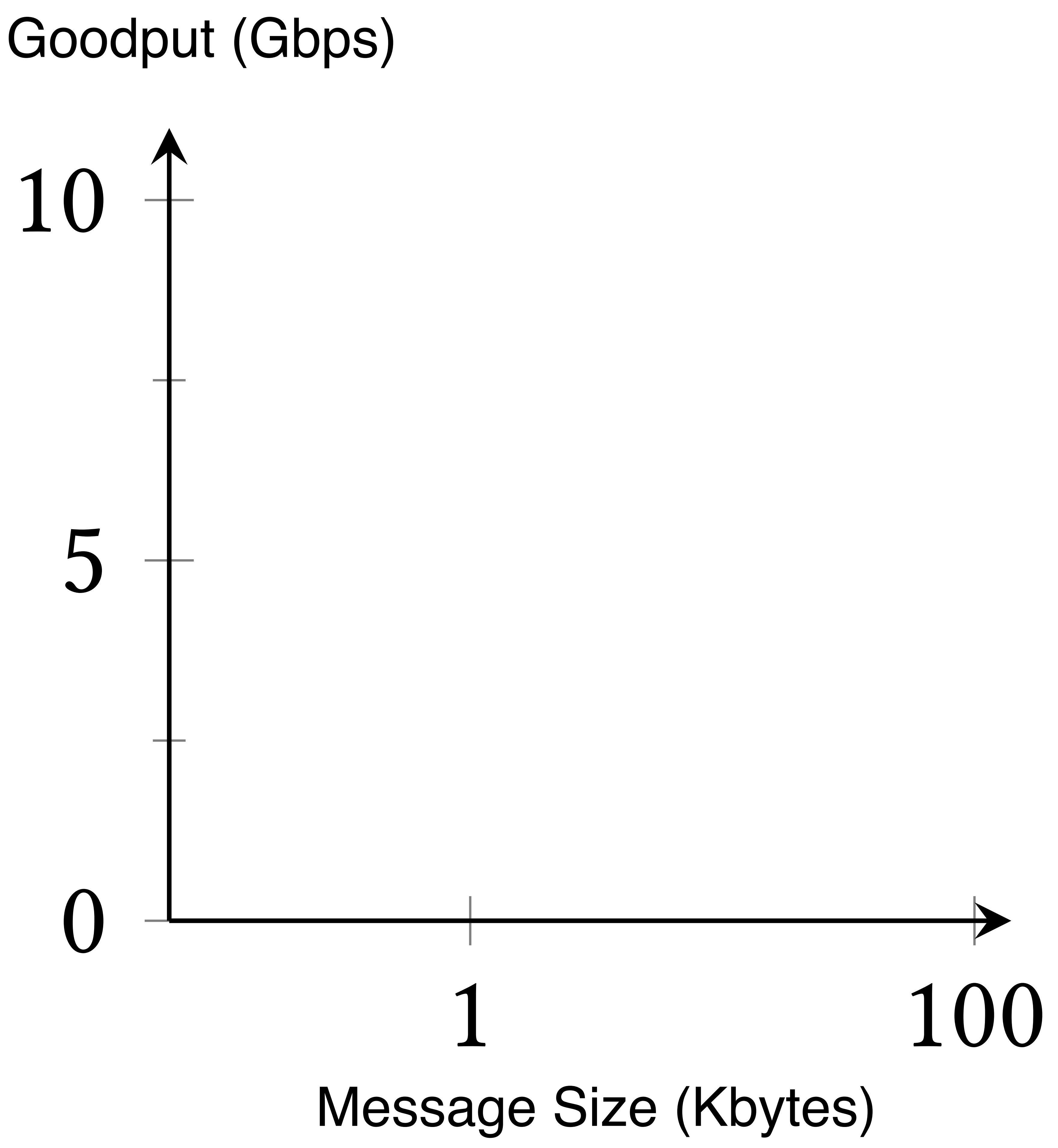


All cores enabled

HEELS introduces minimal performance overhead

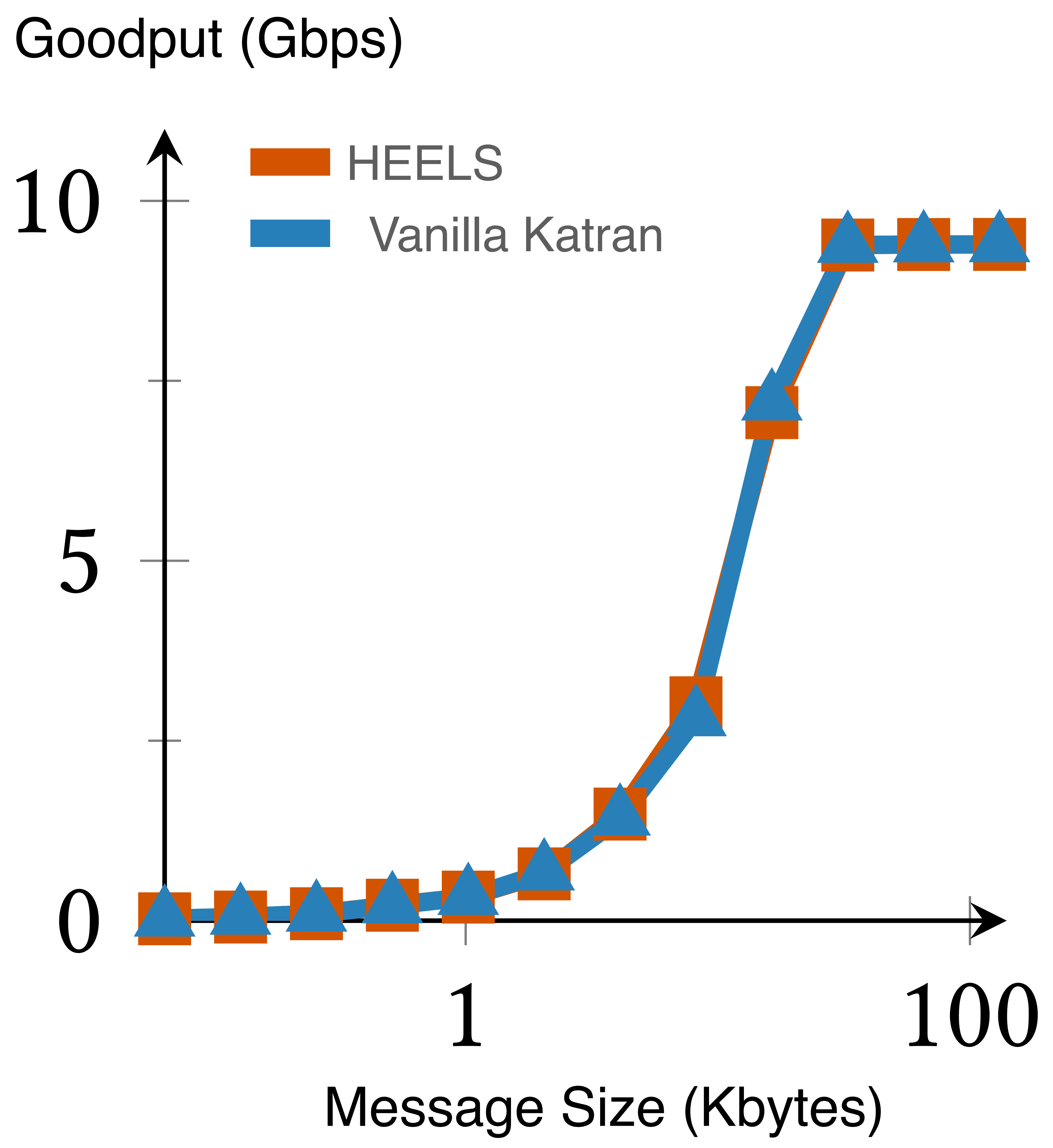


All cores enabled

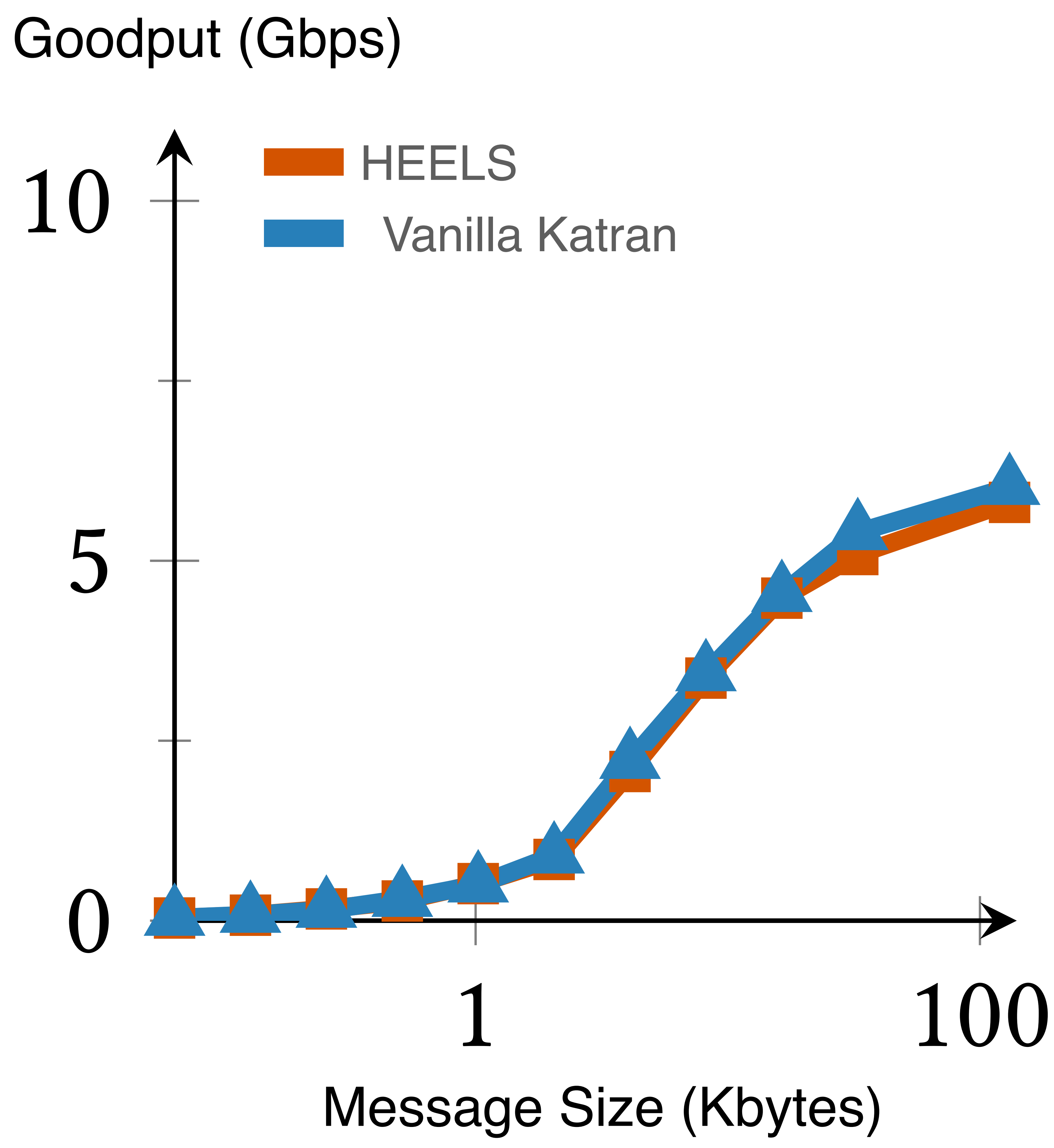


Single core enabled

HEELS introduces minimal performance overhead

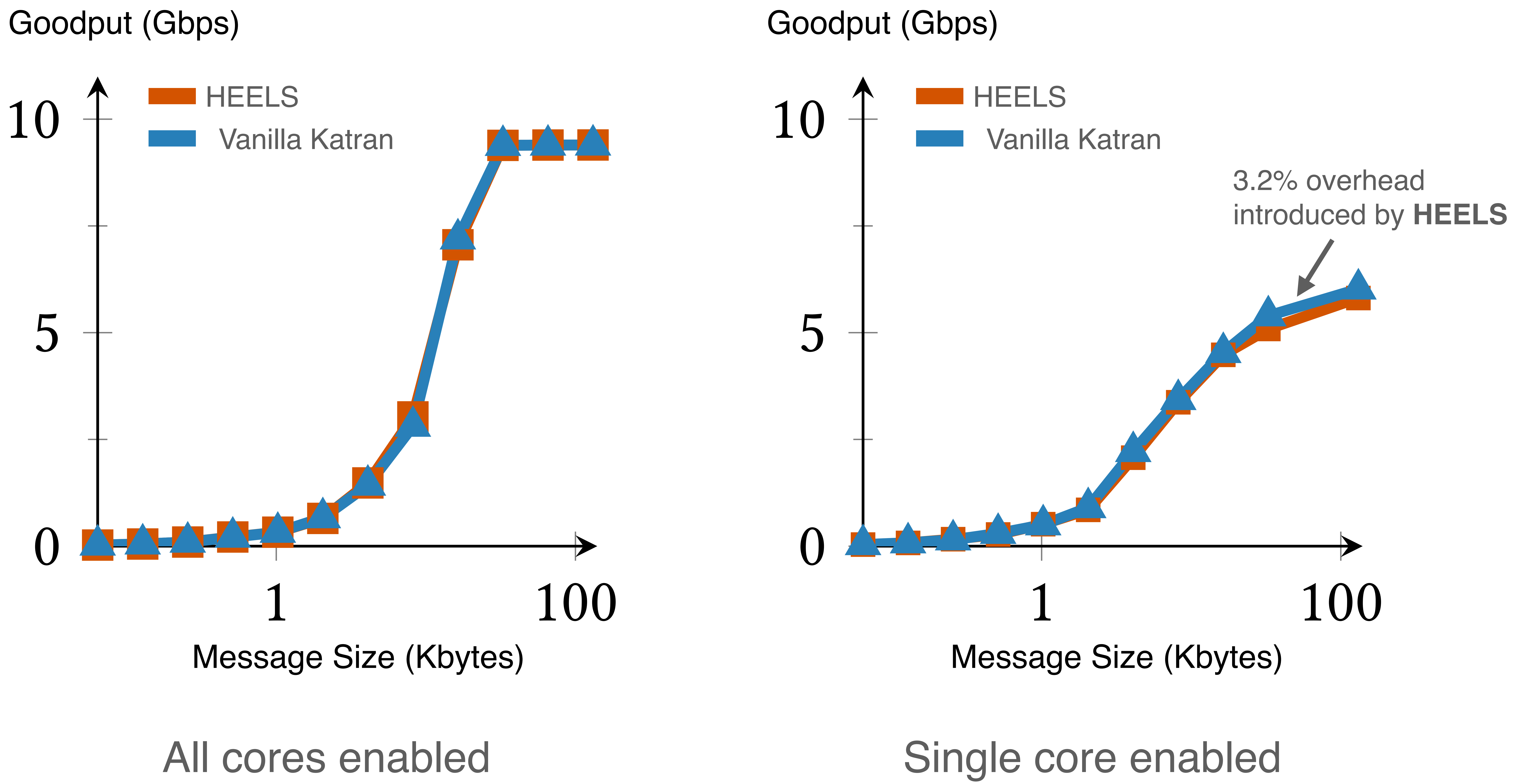


All cores enabled

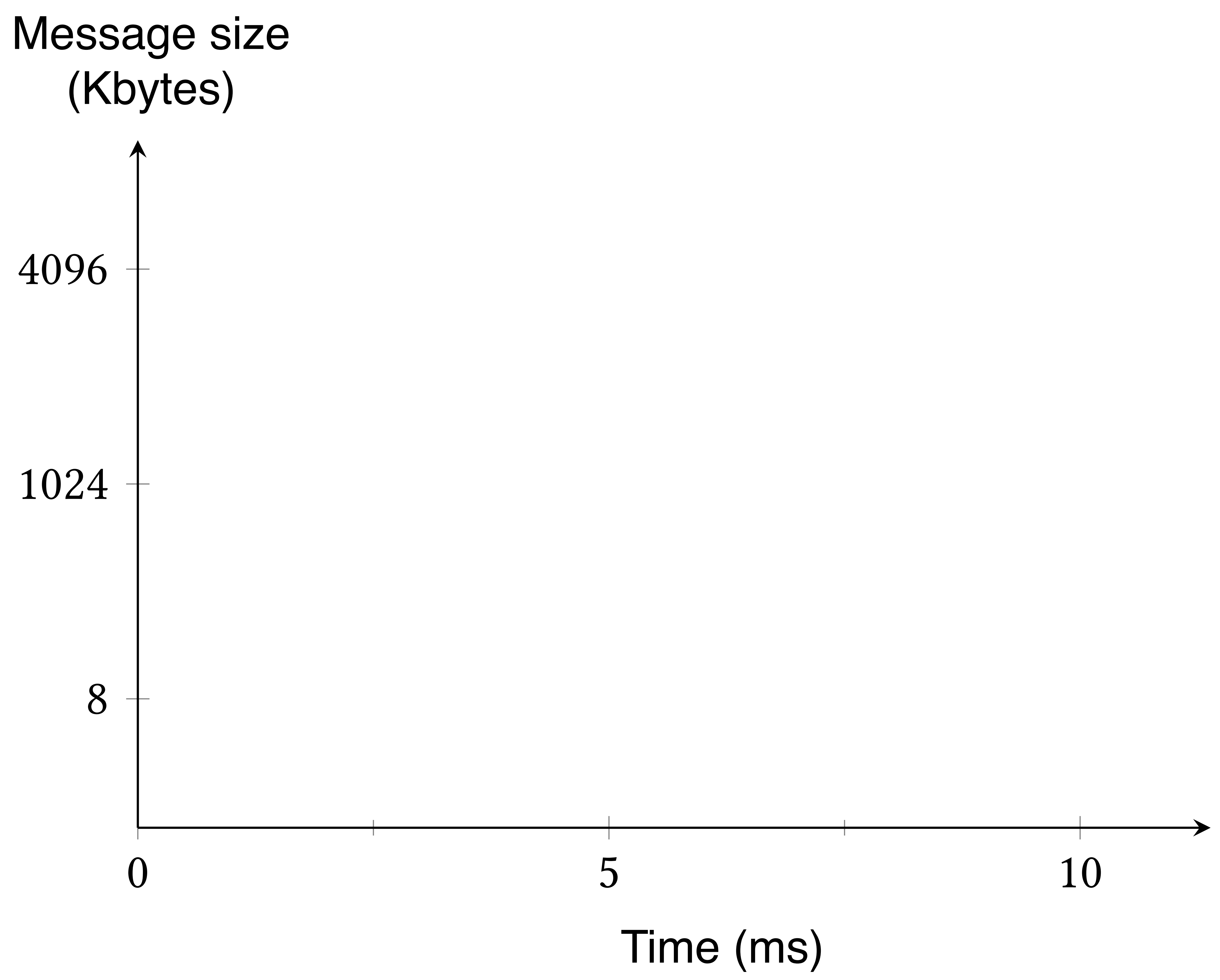


Single core enabled

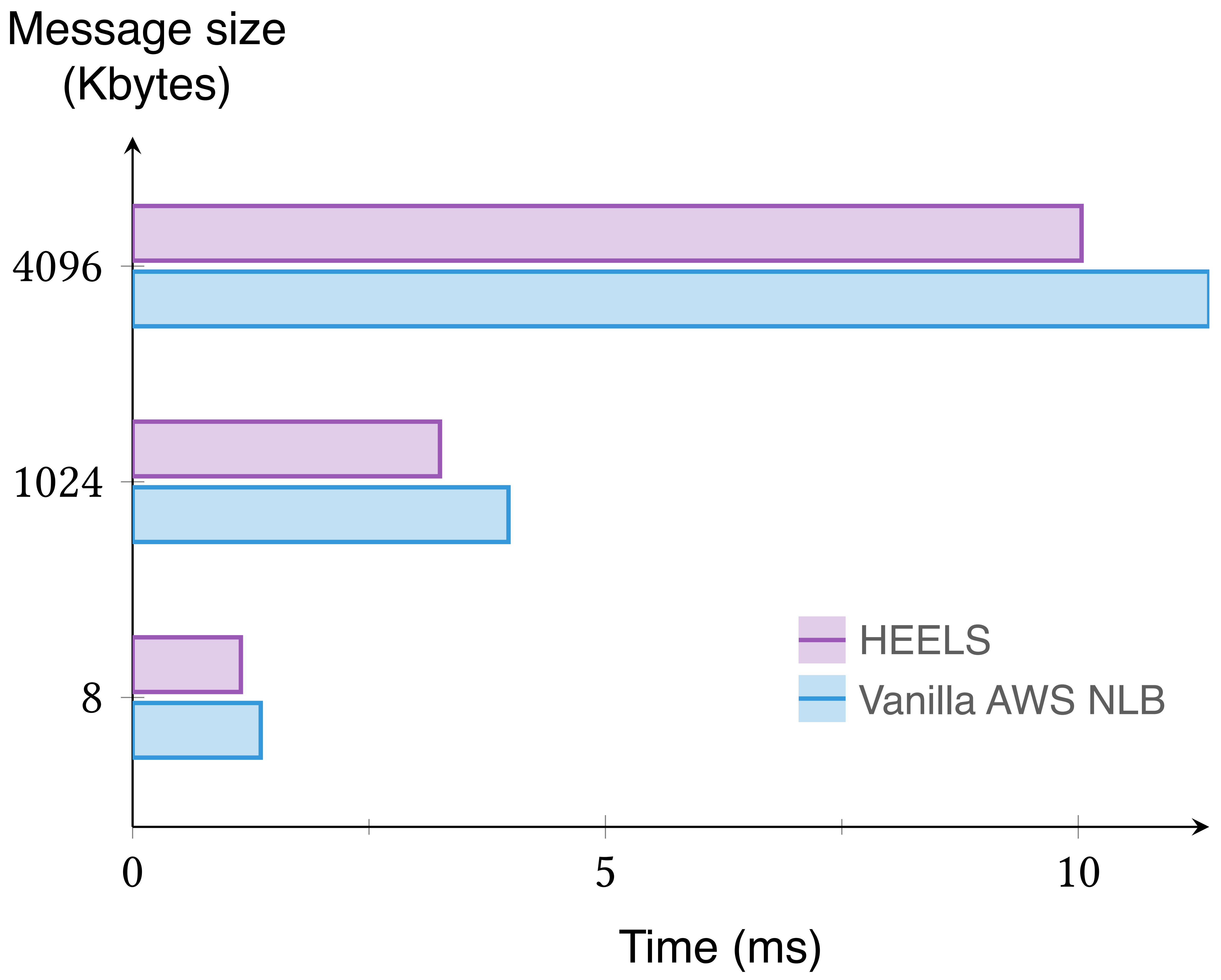
HEELS introduces minimal performance overhead



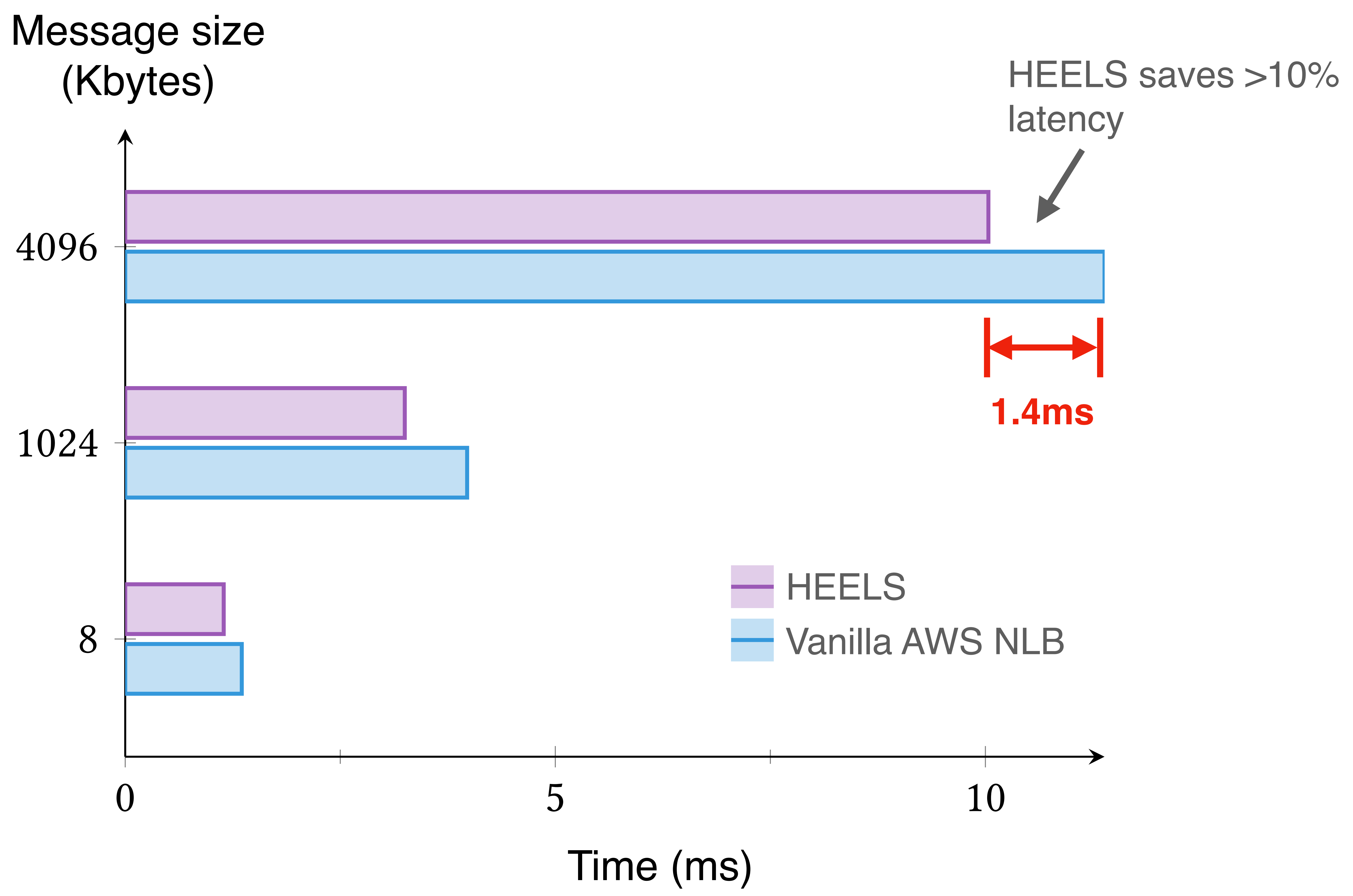
HEELS improves the latency introduced by centralized LBs



HEELS improves the latency introduced by centralized LBs



HEELS improves the latency introduced by centralized LBs



HEELS offers significant cost benefits for cloud users

AWS NLB pricing

Cost for using AWS NLB
a flat rate of $0.027/hr

Cost for data traversing AWS NLB
a $0.006/hr rate for every GB processed.

Message size (Kbytes)	Price per hour ($/hr)	
	Vanilla AWS NLB	HEEL
8		
1024		
4096		

HEELS offers significant cost benefits for cloud users

AWS NLB pricing

Cost for using AWS NLB
a flat rate of $0.027/hr

Cost for data traversing AWS NLB
a $0.006/hr rate for every GB processed.

Message size (Kbytes)	Price per hour ($/hr)	
	Vanilla AWS NLB	HEEL
8	0.028	0.027
1024	0.135	0.027
4096	0.459	0.027

HEELS offers significant cost benefits for cloud users

AWS NLB pricing

Cost for using AWS NLB

a flat rate of $0.027/hr

Cost for data traversing AWS NLB

a $0.006/hr rate for every GB processed.

Message size (Kbytes)	Price per hour ($/hr)	
	Vanilla AWS NLB	HEEL
8	0.028	0.027
1024	0.135	0.027
4096	0.459	0.027

constant costs

HEELS offers significant cost benefits for cloud users

AWS NLB pricing

Cost for using AWS NLB
a flat rate of $0.027/hr

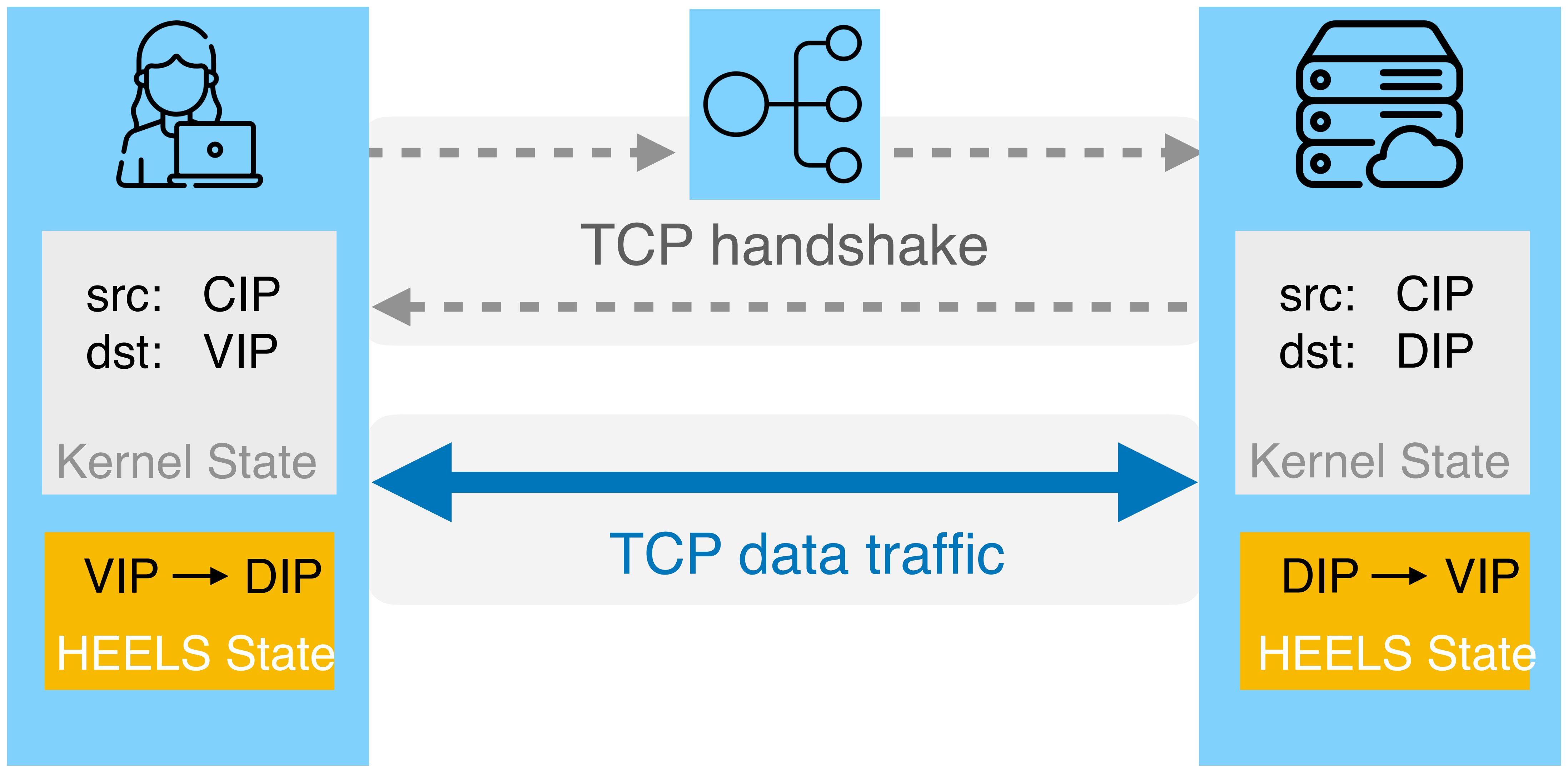
Cost for data traversing AWS NLB
a $0.006/hr rate for every GB processed.

Message size (Kbytes)	Price per hour ($/hr)	
	Vanilla AWS NLB	HEEL
8	0.028	0.027
1024	0.135	0.027
4096	0.459	0.027

Increasing costs as the message size grows

constant costs

HEELS: A Host-Enabled eBPF-Based Load Balancing Scheme



A new eBPF-based load balancing scheme

Readily deployable on the cloud

Bringing both performance and cost benefits to users