OS Kernel Support for a Low-Overhead Container Overlay Network

Danyang Zhuo, Kaiyuan Zhang, Yibo Zhu, Hongqiang Harry Liu
Matthew Rockett, Arvind Krishnamurthy, Thomas Anderson







Containers are ubiquitous

Cache



Big data





Web server





Deep learning





Database

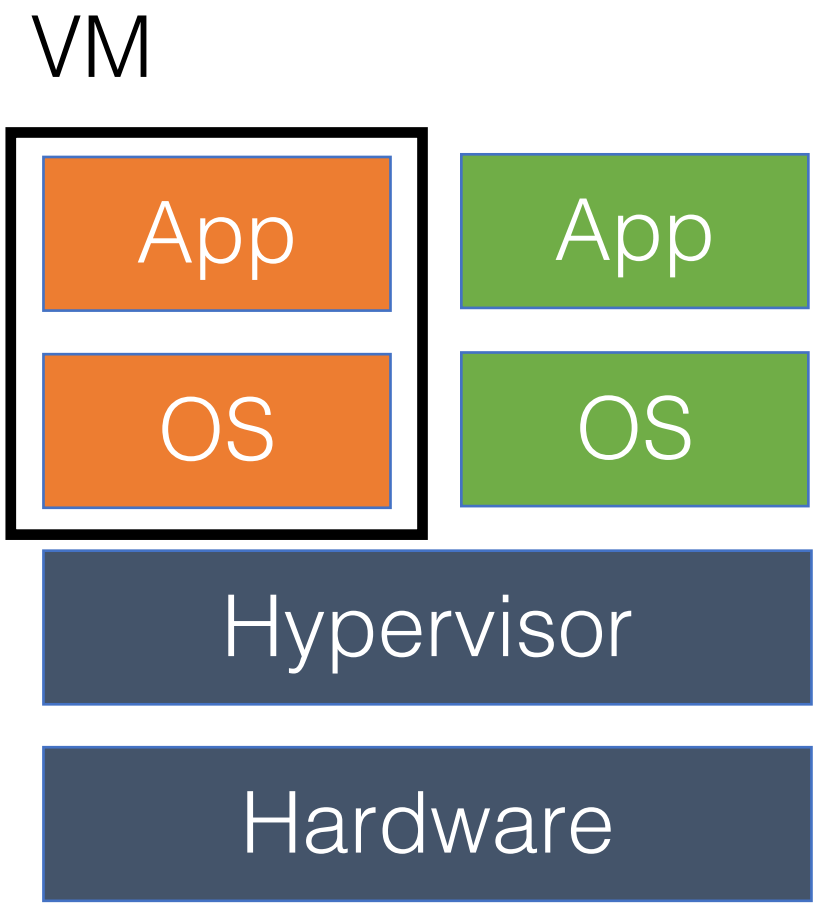




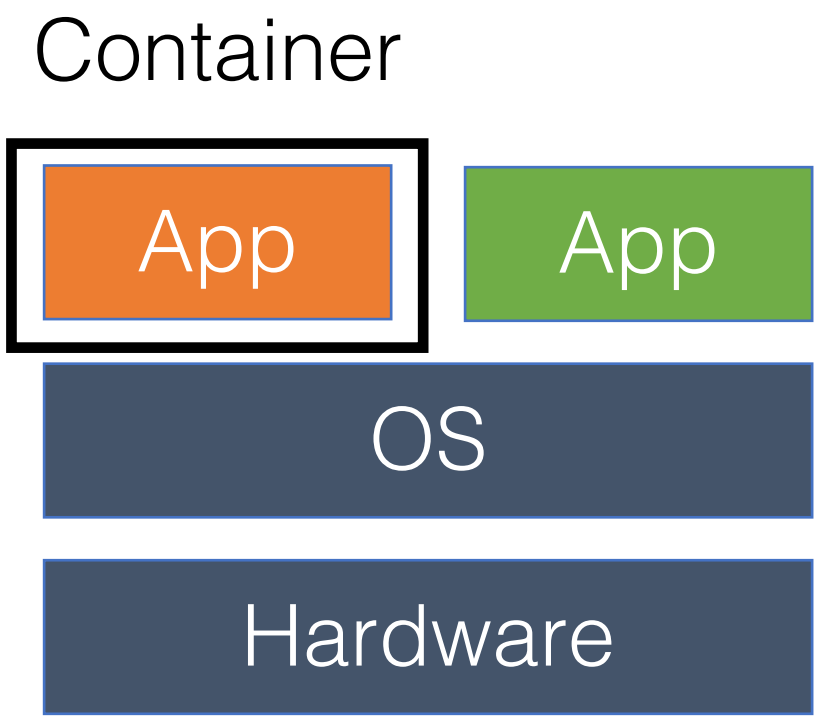
Microservice






VM
App
App
OS
OS
Hypervisor
Hardware


docker
kubernetes


Container
App
App
OS
Hardware

How do containers communicate?

- Host mode
  - Use the host network interface to communicate

Two containers cannot bind to the same port.

- Macvlan mode (or SR-IOV)
  - Make container's IP address routable on the host network

Complicates host network routing

- Overlay mode
  - Container network virtualization

High overheads

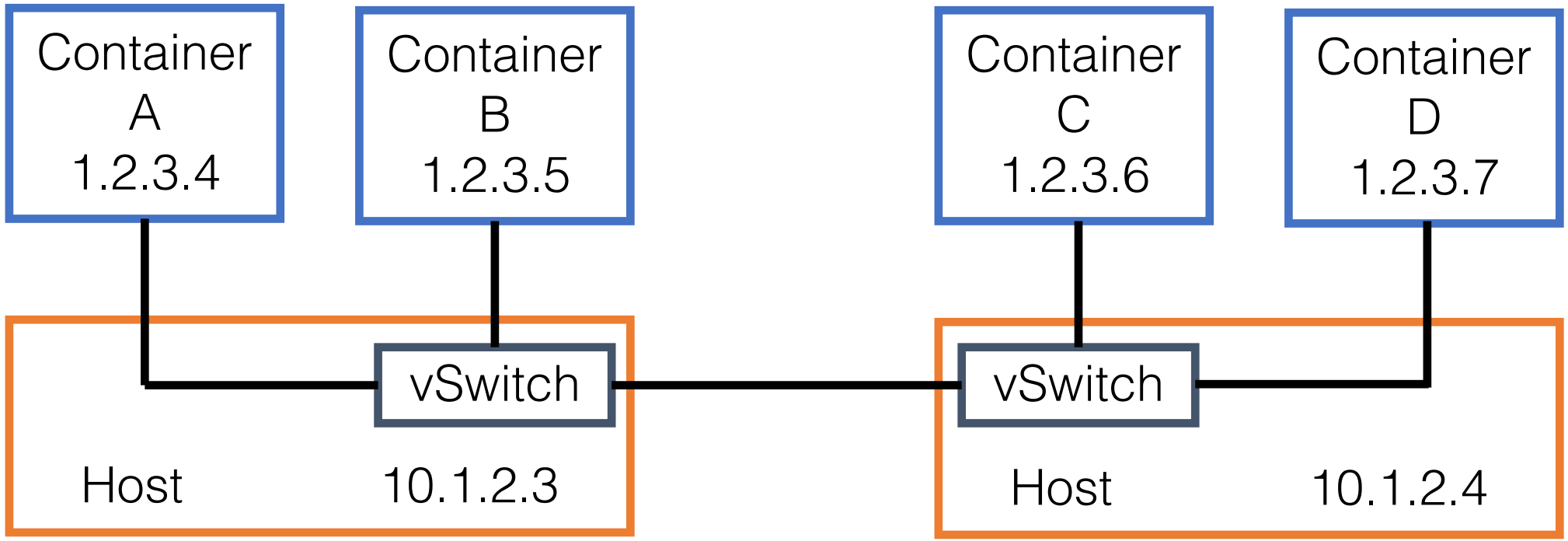
Are the network virtualization overheads fundamental?



In this talk…

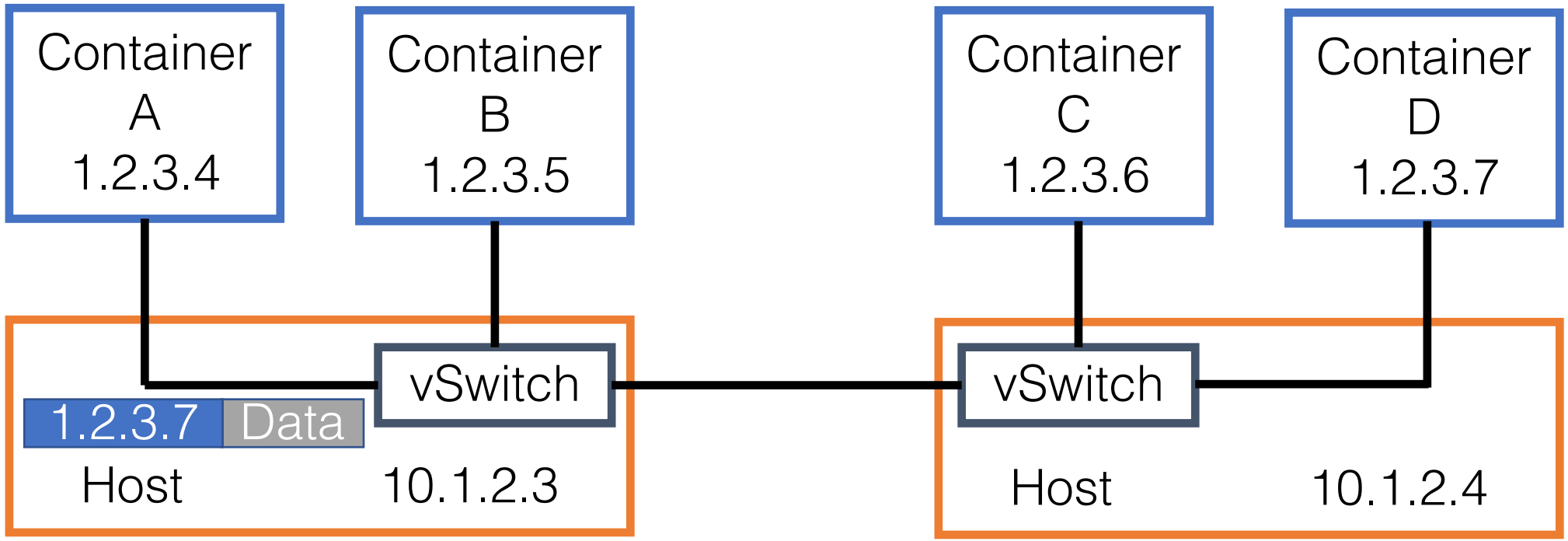
- Existing approach: Packet-based network virtualization results in high overheads.
- Slim: connection-based network virtualization that is compatible with existing Linux applications.
- Saving up to 56% CPU cycles on popular cloud applications (e.g., Memcached, Nginx, PostgreSQL, Apache Kafka).

Container network virtualization

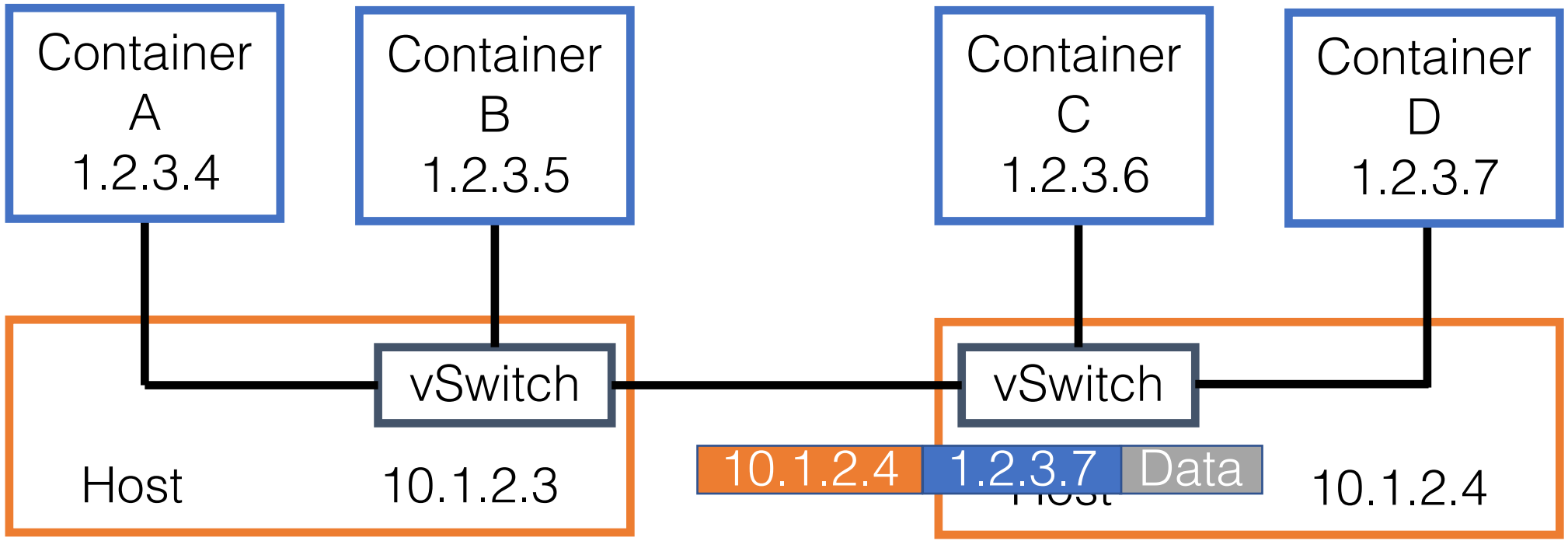


Give a set of containers an illusion of owning a dedicated network.

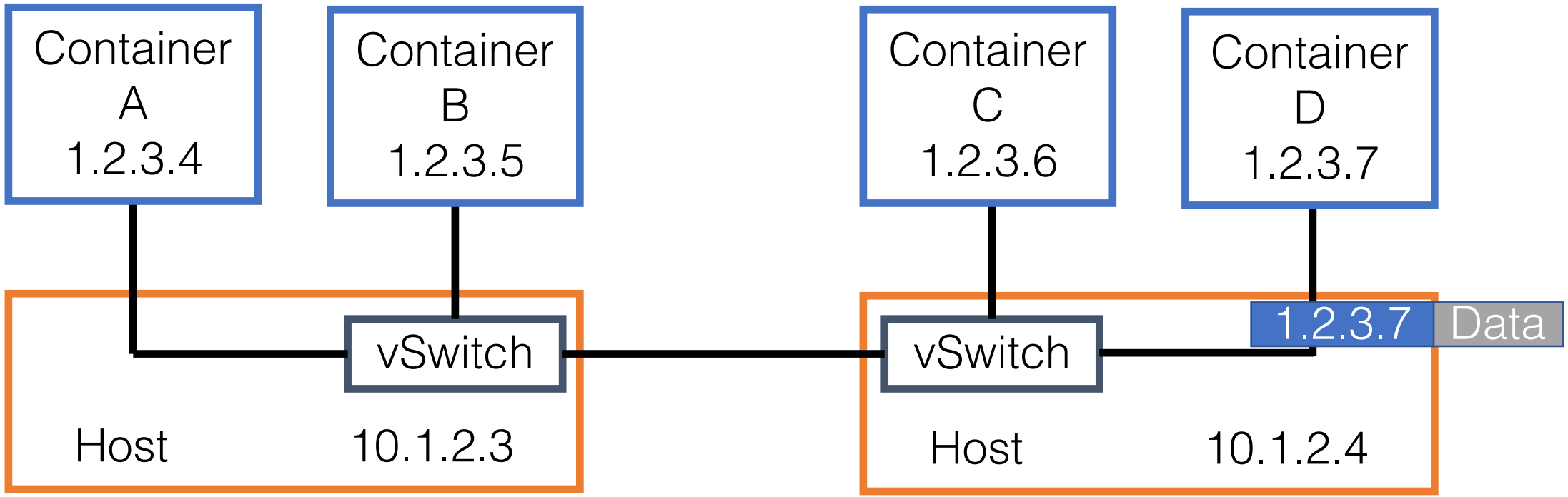
Container network virtualization



Container network virtualization

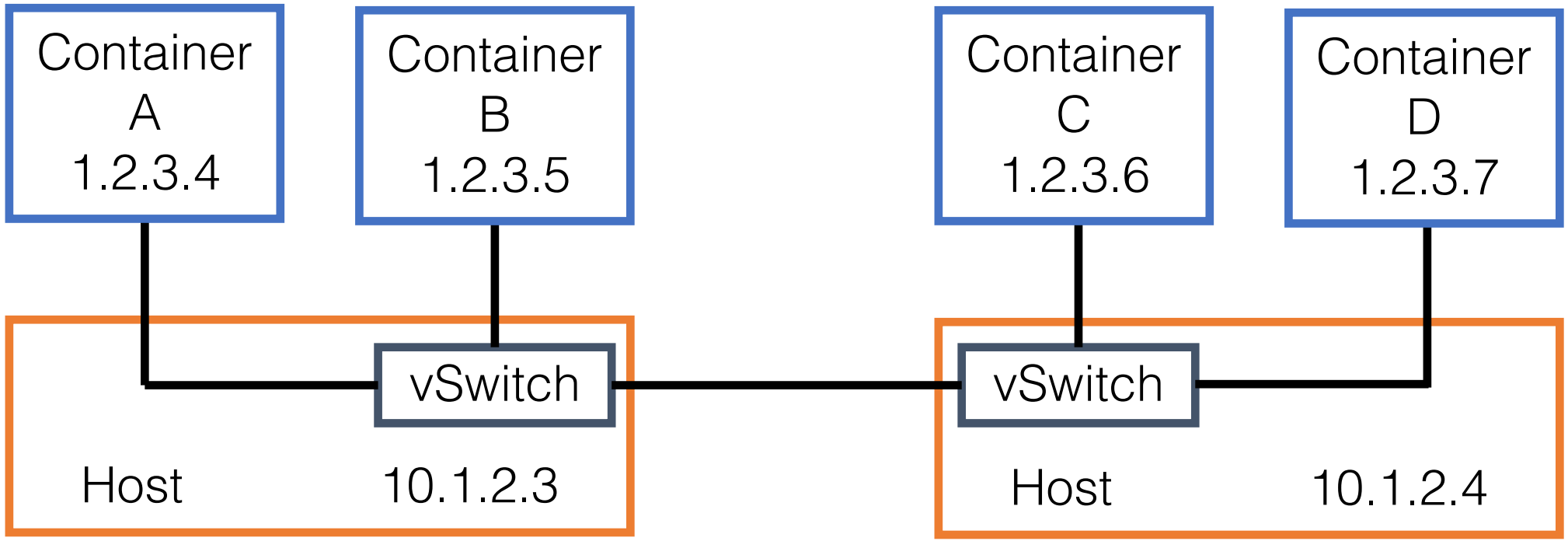


Container network virtualization



Packet-based network virtualization

Why packet-based virtualization?



Performance overheads

Setup	Throughput	Latency (RTT)
Intra, Host	48.4 Gbps	5.9 us
Intra, Container	37.4 Gbps (23%)	7.9 us (34%)
Inter, Host	26.8 Gbps	11.3 us
Inter, Container	14.0 Gbps (48%)	20.9 us (85%)

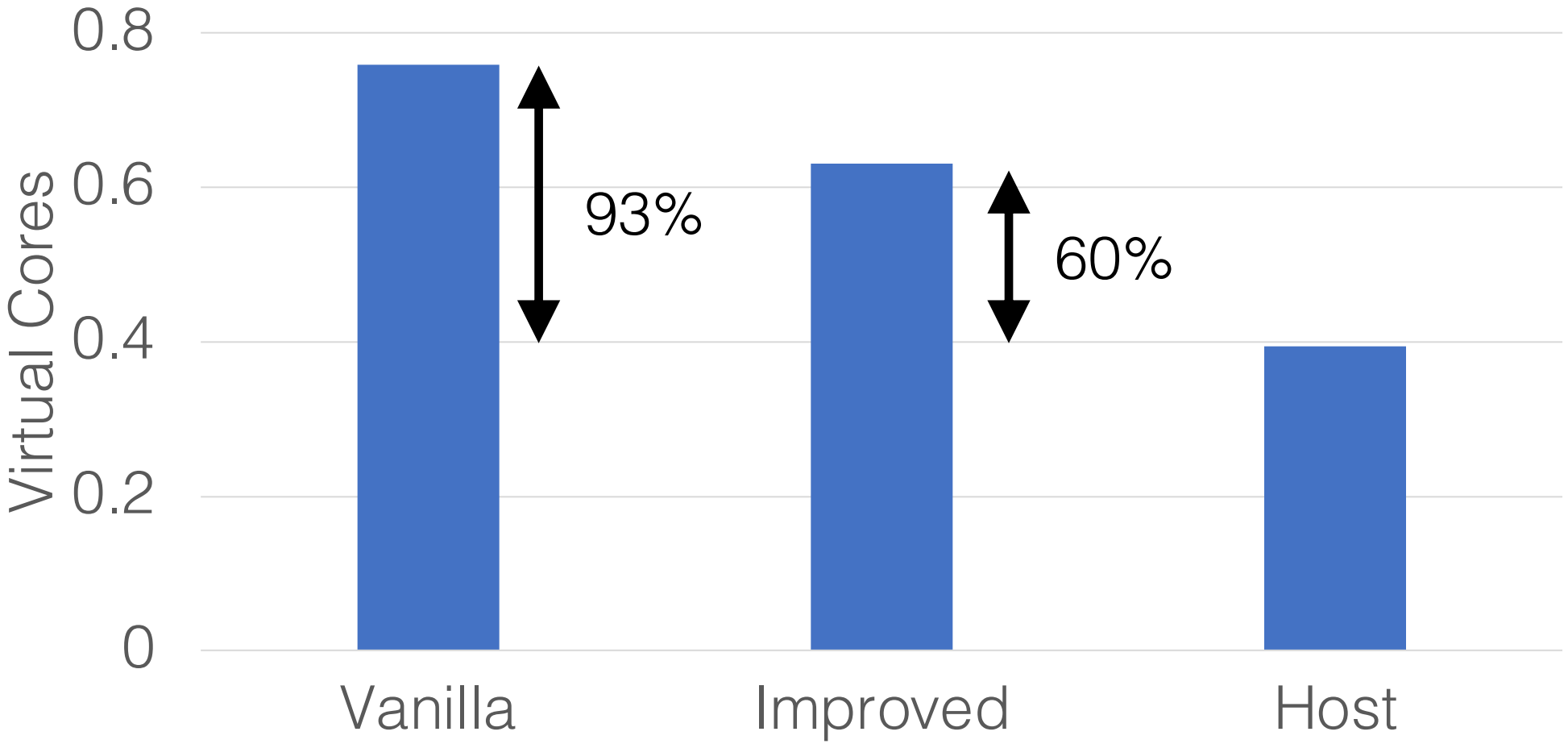
Intel Xeon E5-2680 (2.5 GHz), Linux 4.4, Intel XL710 NIC (40G).

Performance overheads

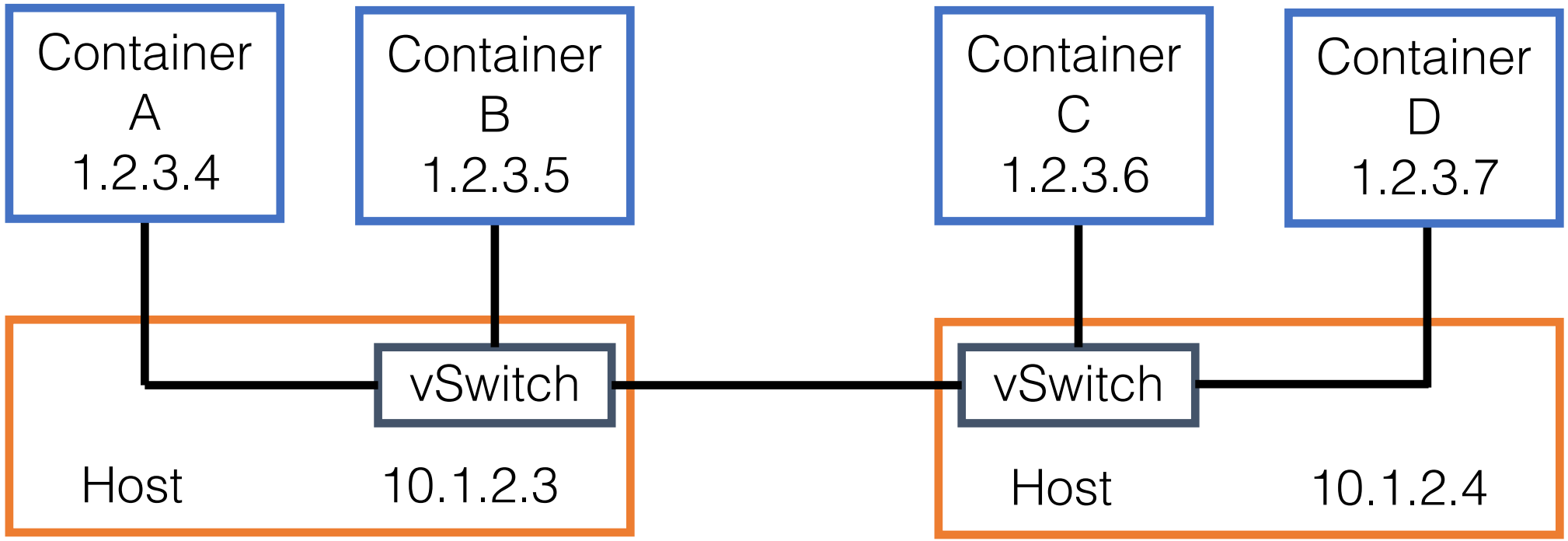
Setup	Throughput	Latency (RTT)
Vanilla	14.0 Gbps (48%)	20.9 us (85%)
Improved	24.5 Gbps (9%)	21.2 us (88%)
Host	26.8 Gbps	11.3 us



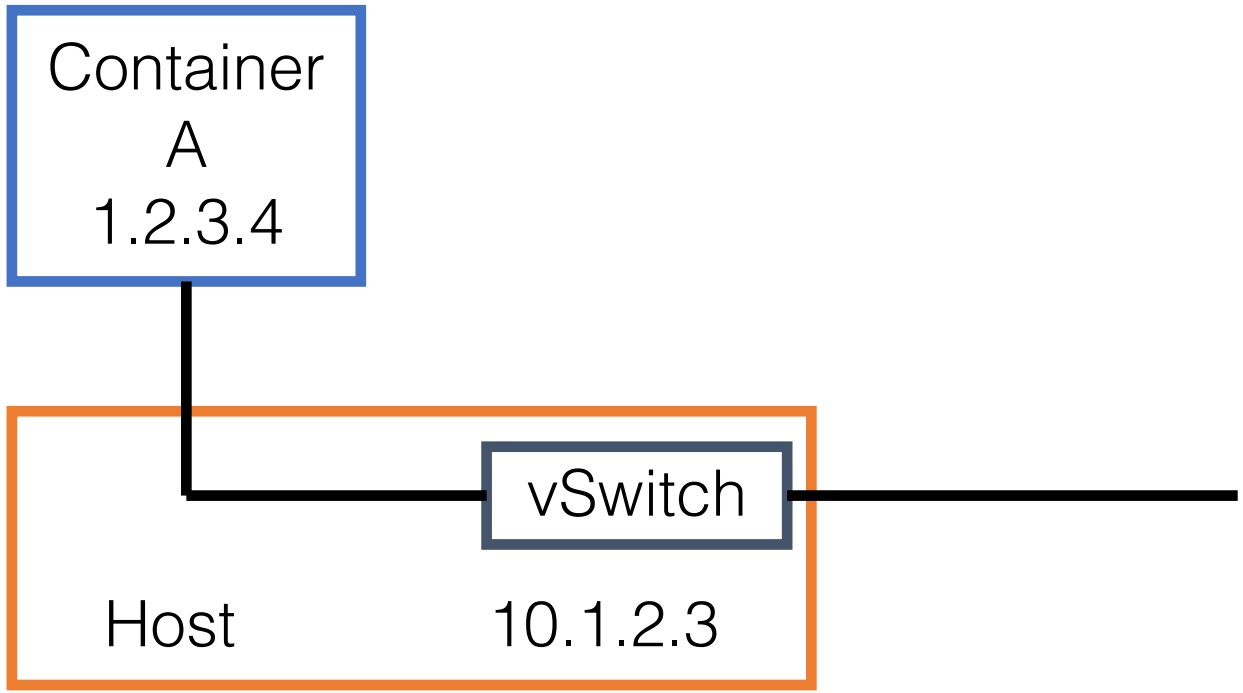
CPU overheads, 10 Gbps



Packet-based virtualization



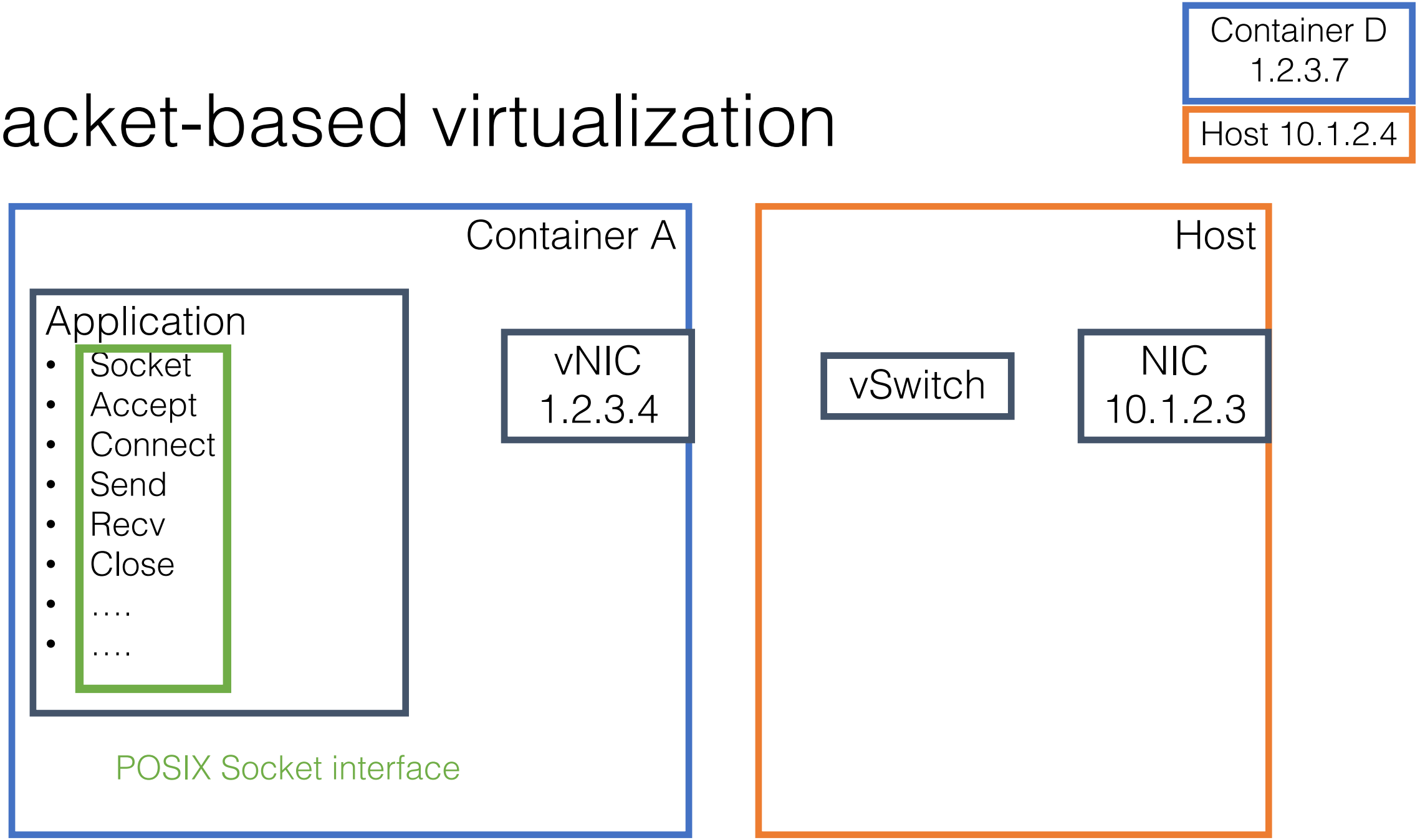
Packet-based virtualization



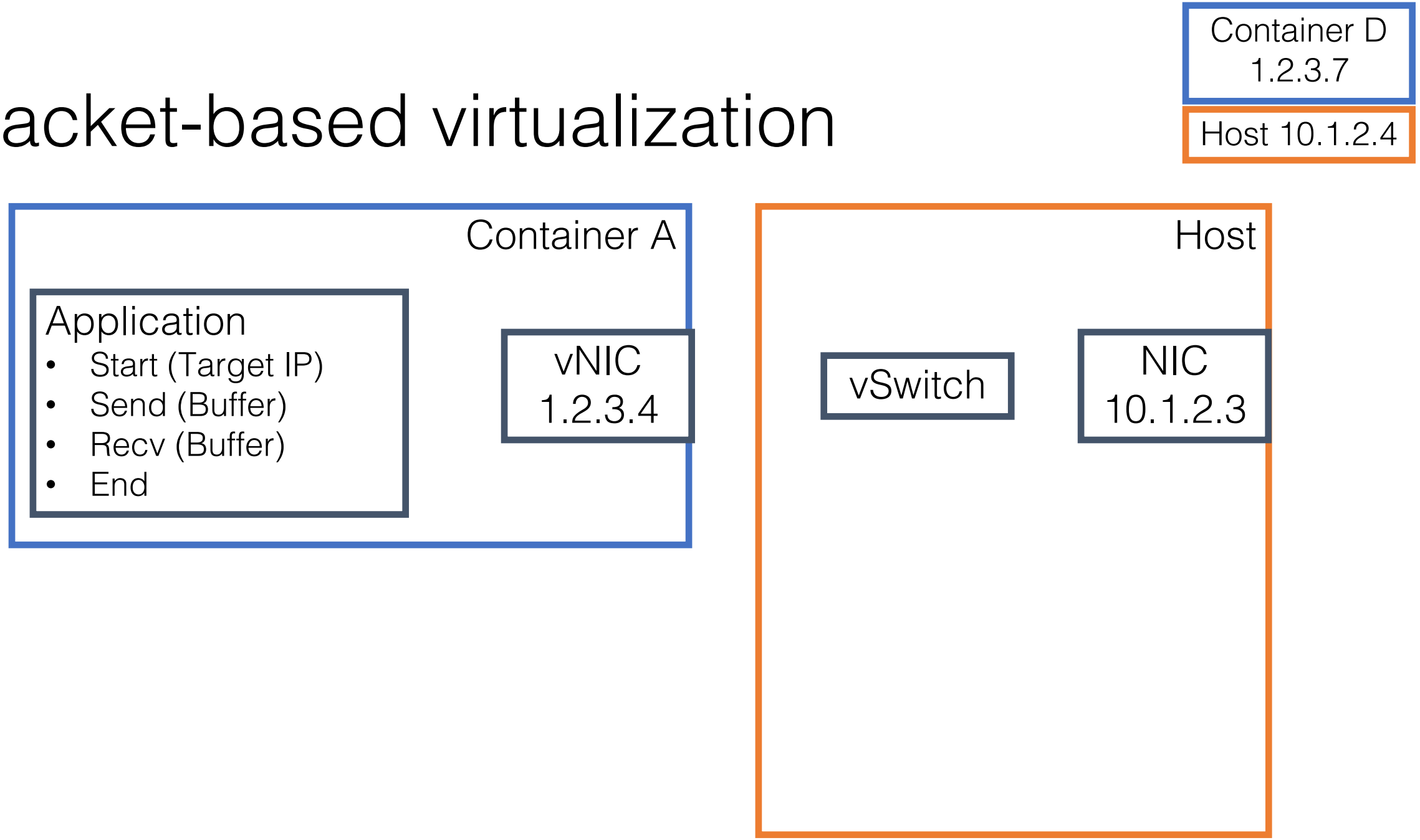
Container D
1.2.3.7

Host 10.1.2.4

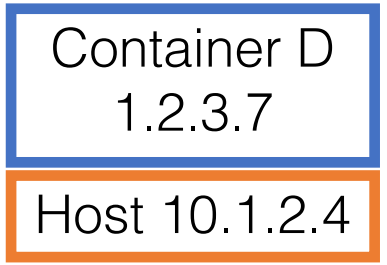
Packet-based virtualization

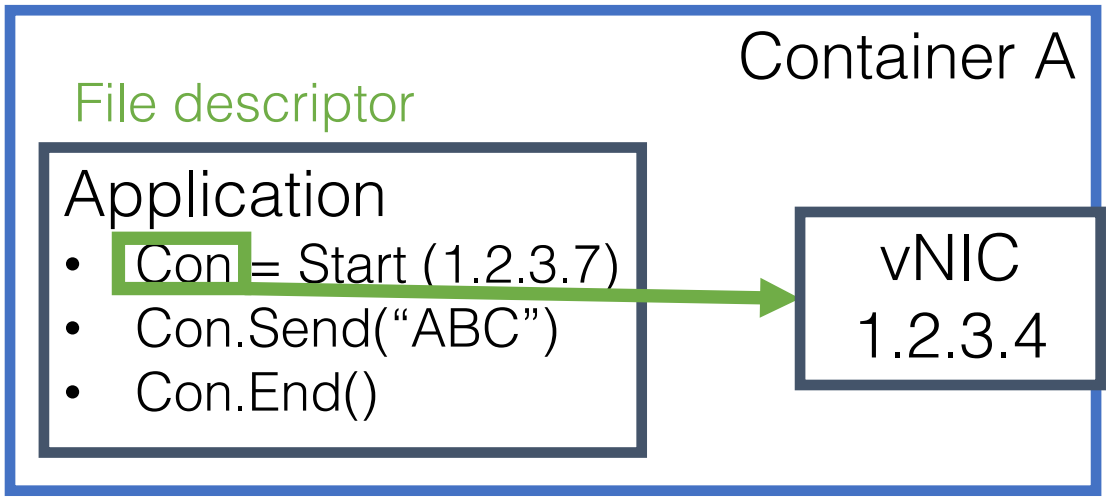


Packet-based virtualization

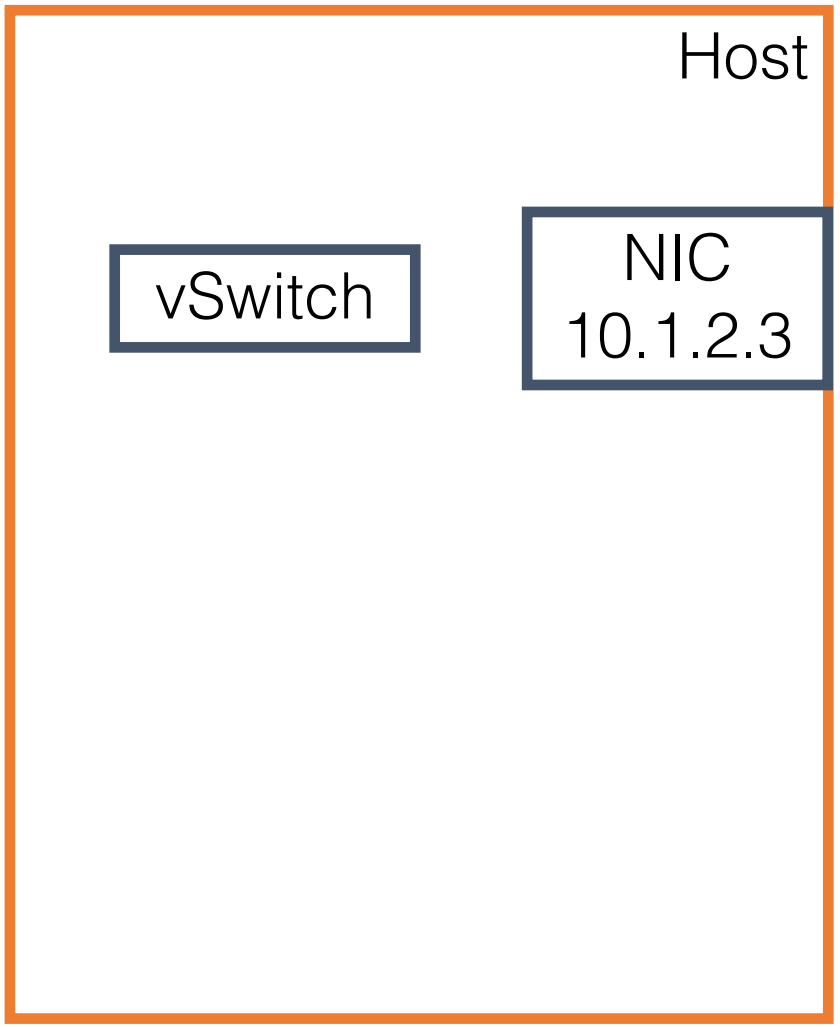


Packet-based virtualization

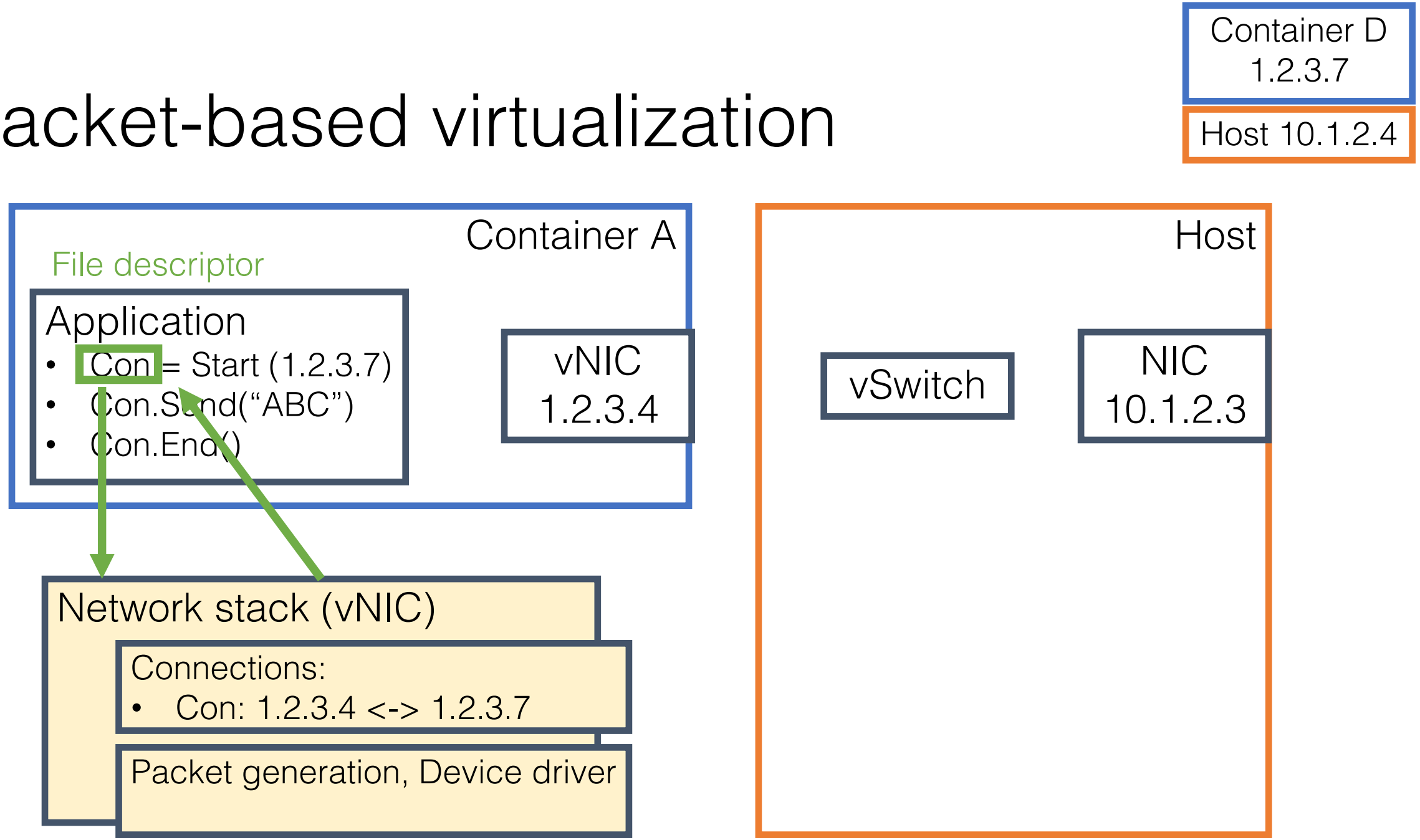




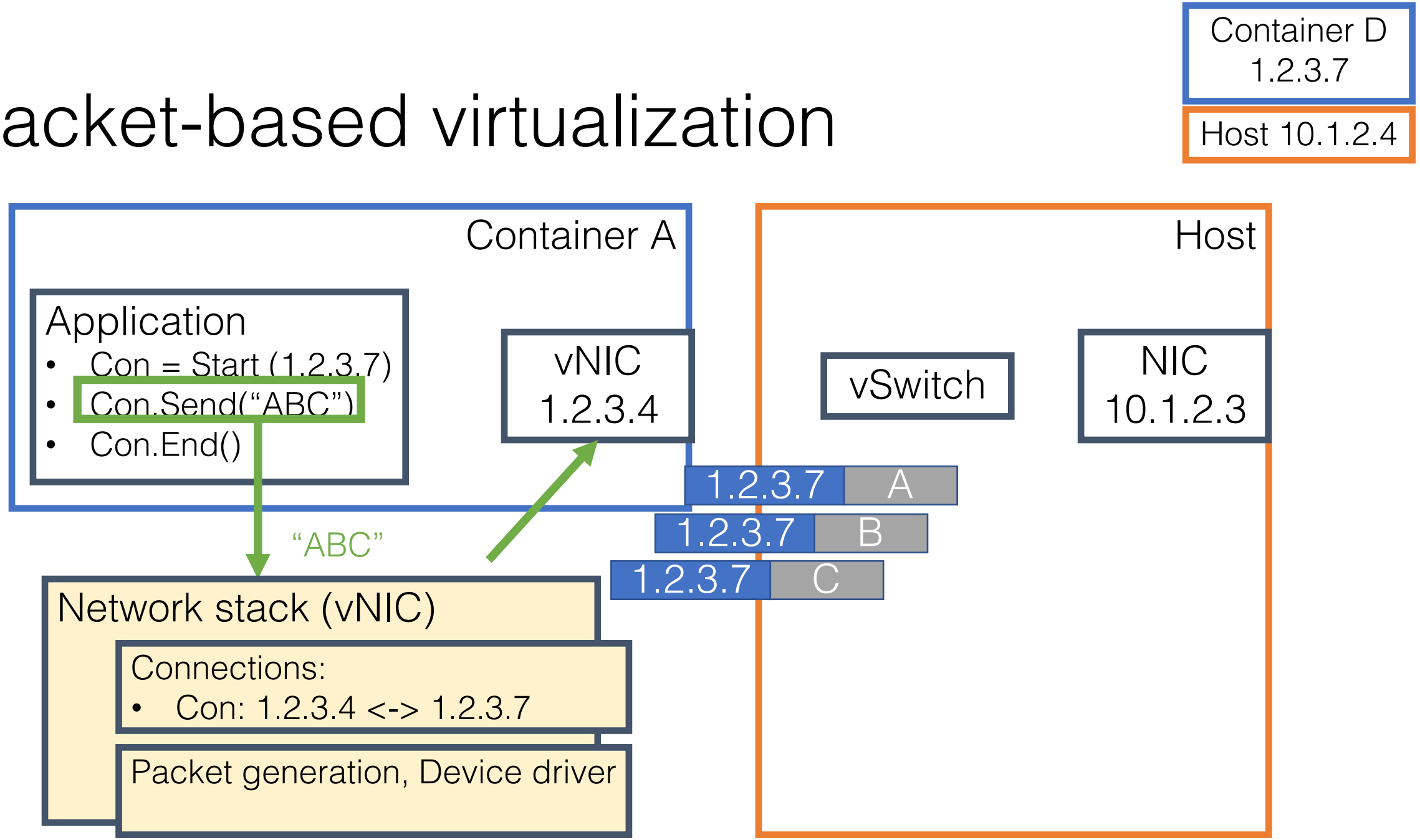
A capability to send/receive packets to/from 1.2.3.7 through vNIC.



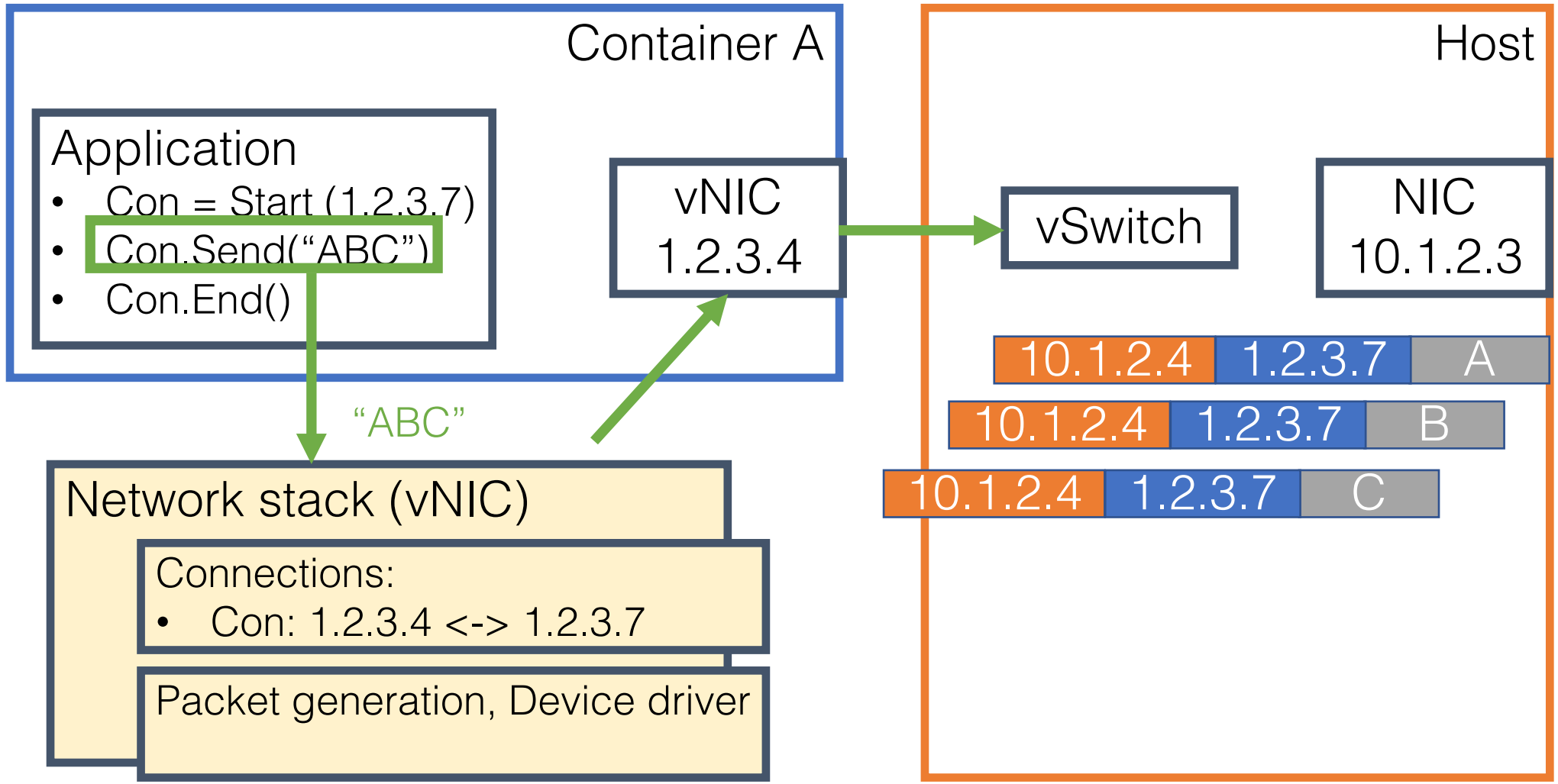
Packet-based virtualization



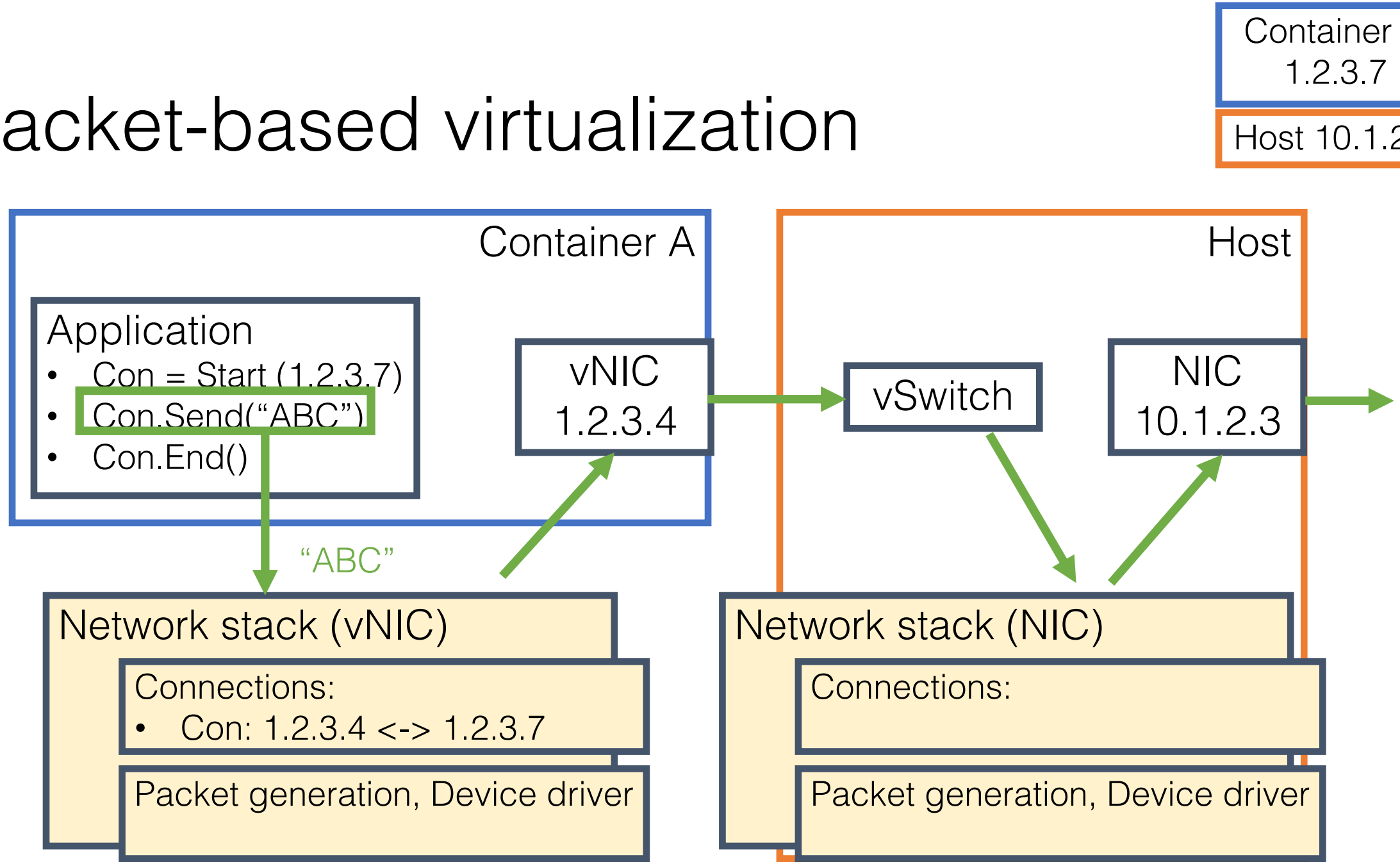
Packet-based virtualization



Packet-based virtualization



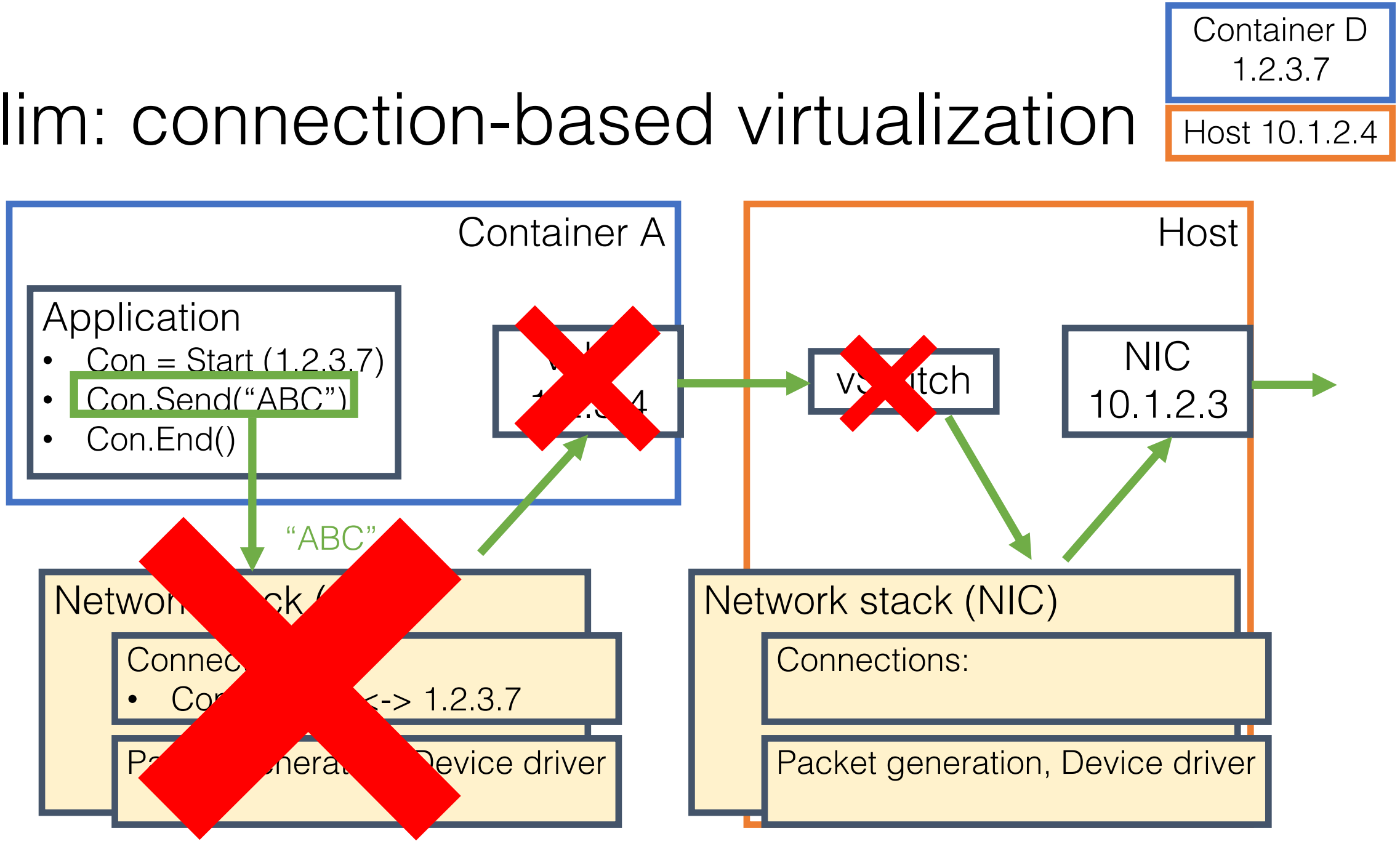
Packet-based virtualization



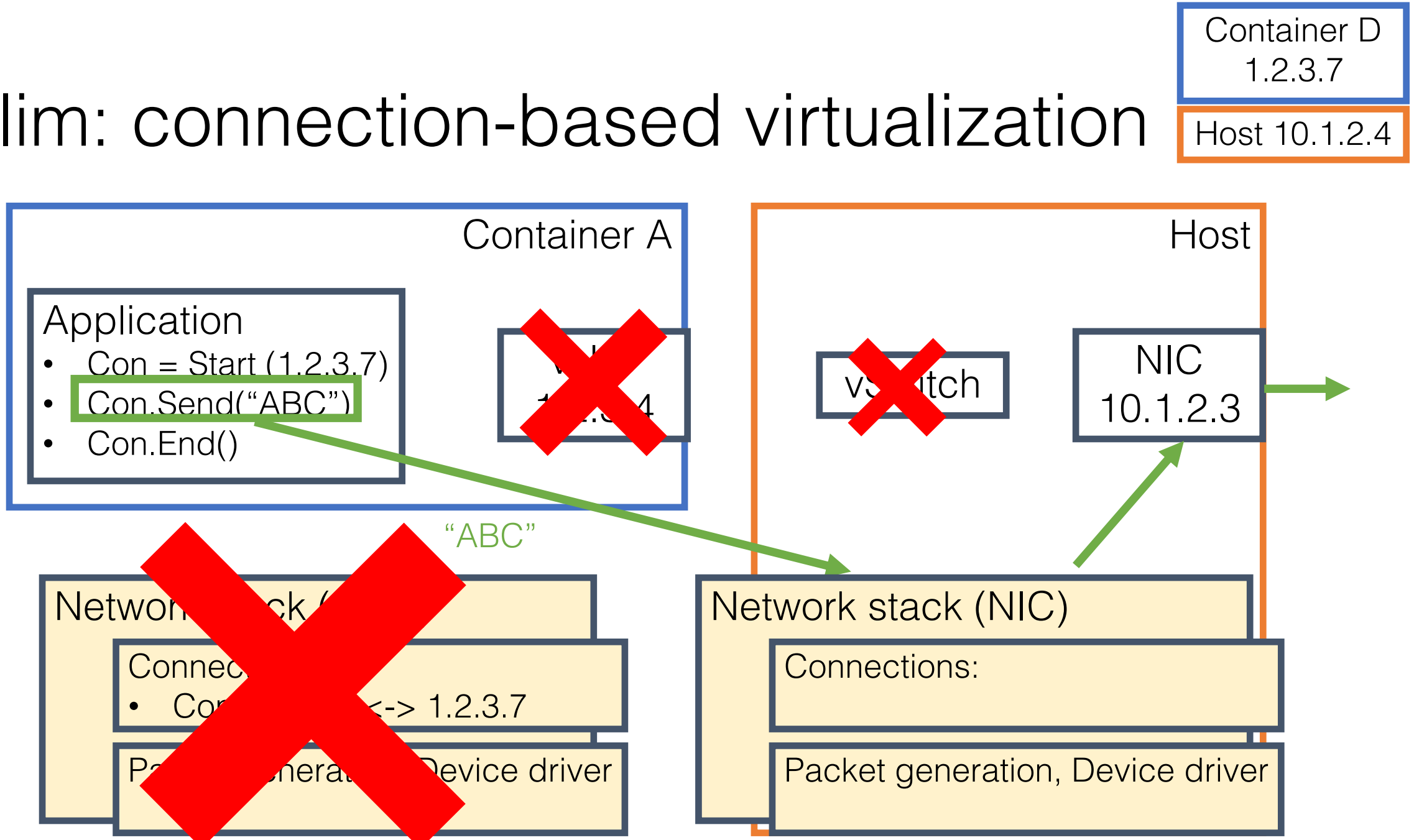
In this talk…

- Existing approach: Packet-based network virtualization results in high overheads.
- Slim: connection-based network virtualization that is compatible with existing Linux applications.
- Saving up to 56% CPU cycles on popular cloud applications (e.g., Memcached, Nginx, PostgreSQL, Apache Kafka).

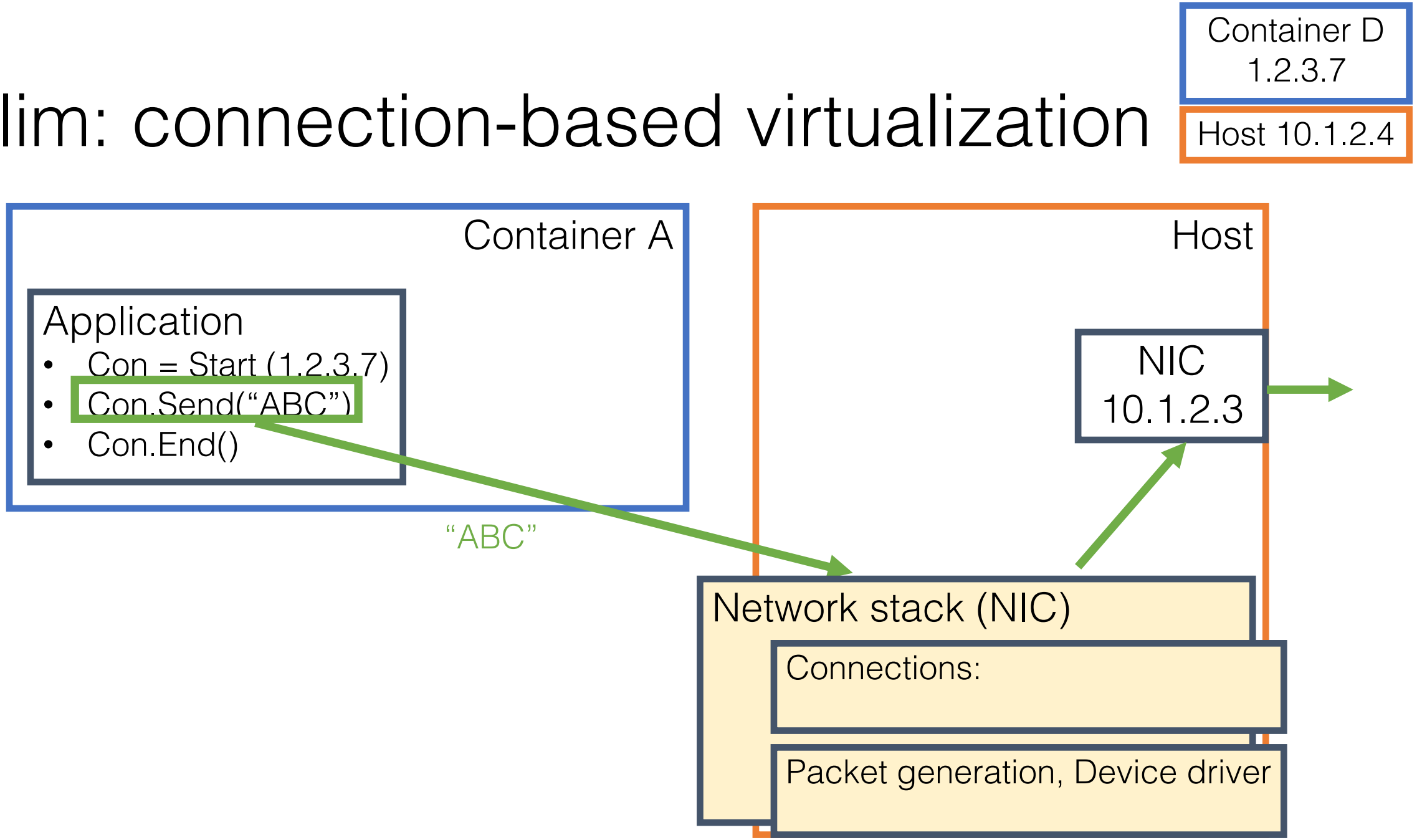
Slim: connection-based virtualization



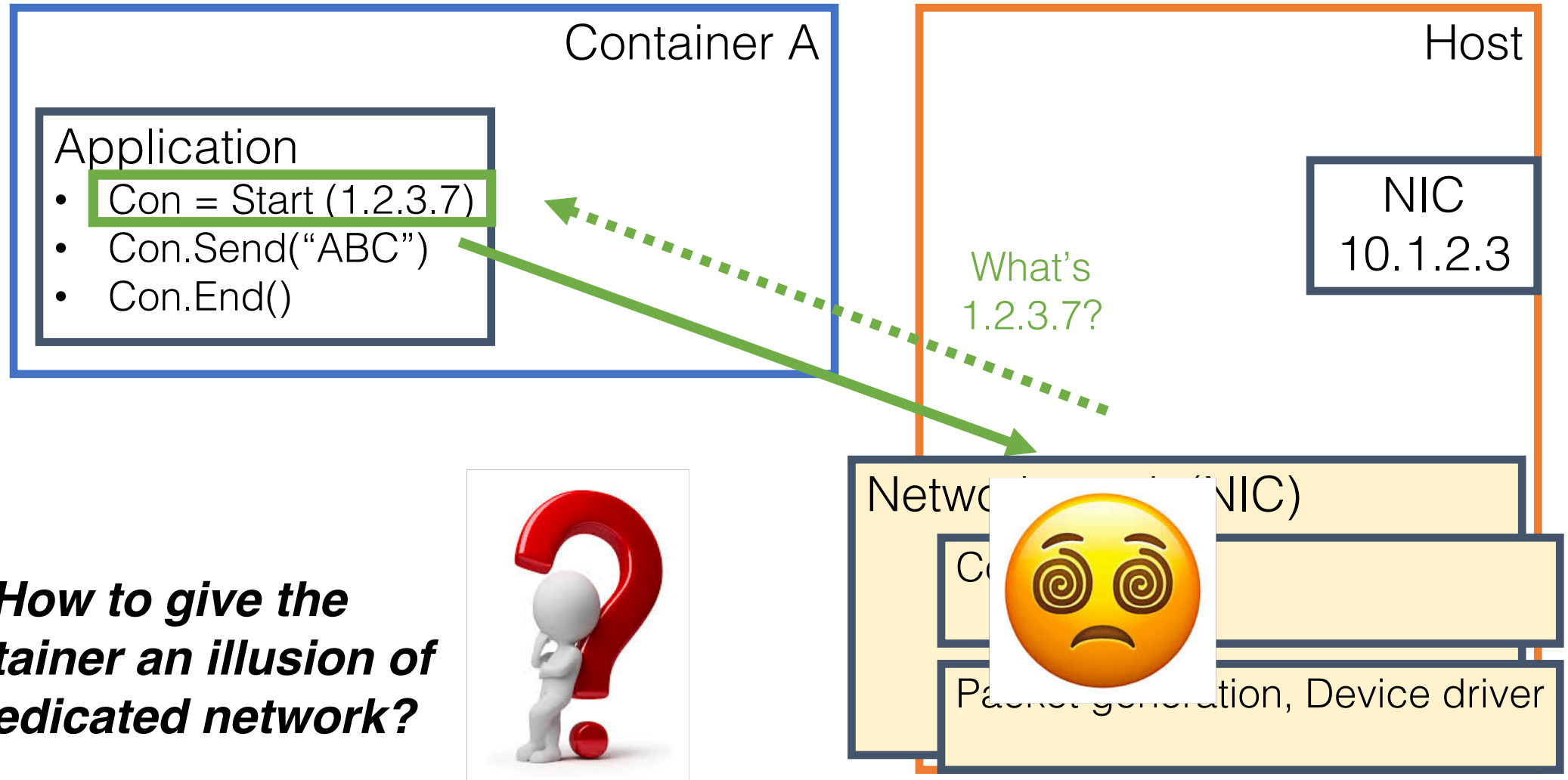
Slim: connection-based virtualization



Slim: connection-based virtualization



Challenge #1: Network virtualization



How to give the container an illusion of a dedicated network?

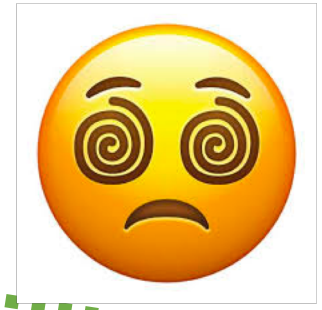
Challenge #2: Compatibility

Container D
1.2.3.7

Host 10.1.2.4

Container A

Application
- Con = Start (1.2.3.7)
- Con.Send("ABC")
- Con.End()



Where's my NIC?

How to work with unmodified applications?



Host

NIC
10.1.2.3

Network stack (NIC)

Connections:

Packet generation, Device driver

Challenge #3: Network Policies

Container D
1.2.3.7

Host 10.1.2.4

Container A

Application
- Con = Start (1.2.3.7)
- Con.Send("ABC")
- Con.End()

"ABC"

Host

NIC
10.1.2.3

Network stack (NIC)

Connections:

Packet generation, Device driver

How do we enforce network policies?



Challenge #4: Security

Container D
1.2.3.7

Host 10.1.2.4

Container A

Application
- Con = Start (1.2.3.7)
- Con.Send("ABC")
- Con.End()

Host



NIC
10.1.2.3

10.1.2.3?

Tell me my IP address



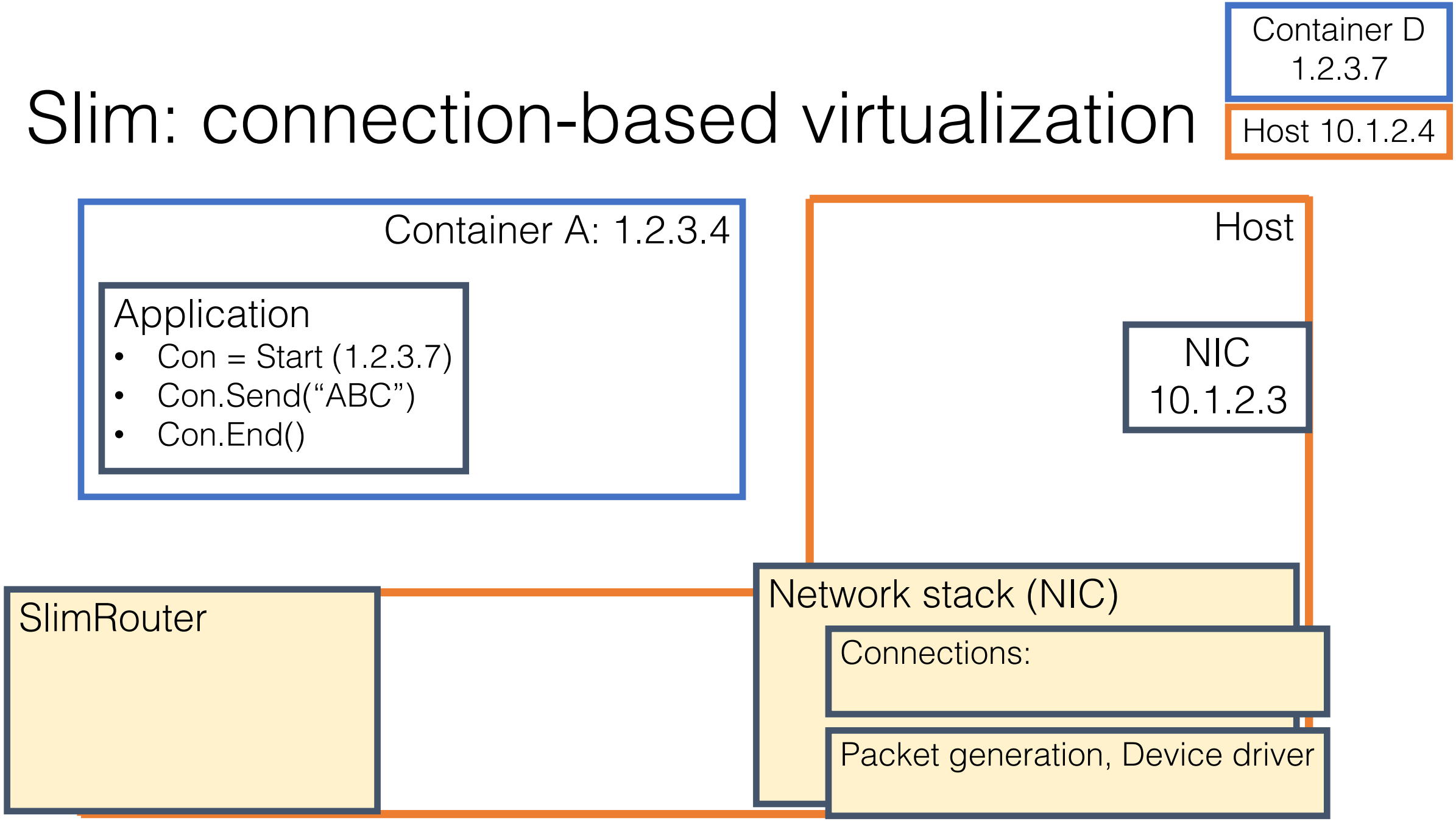
Network stack (NIC)

Connections:

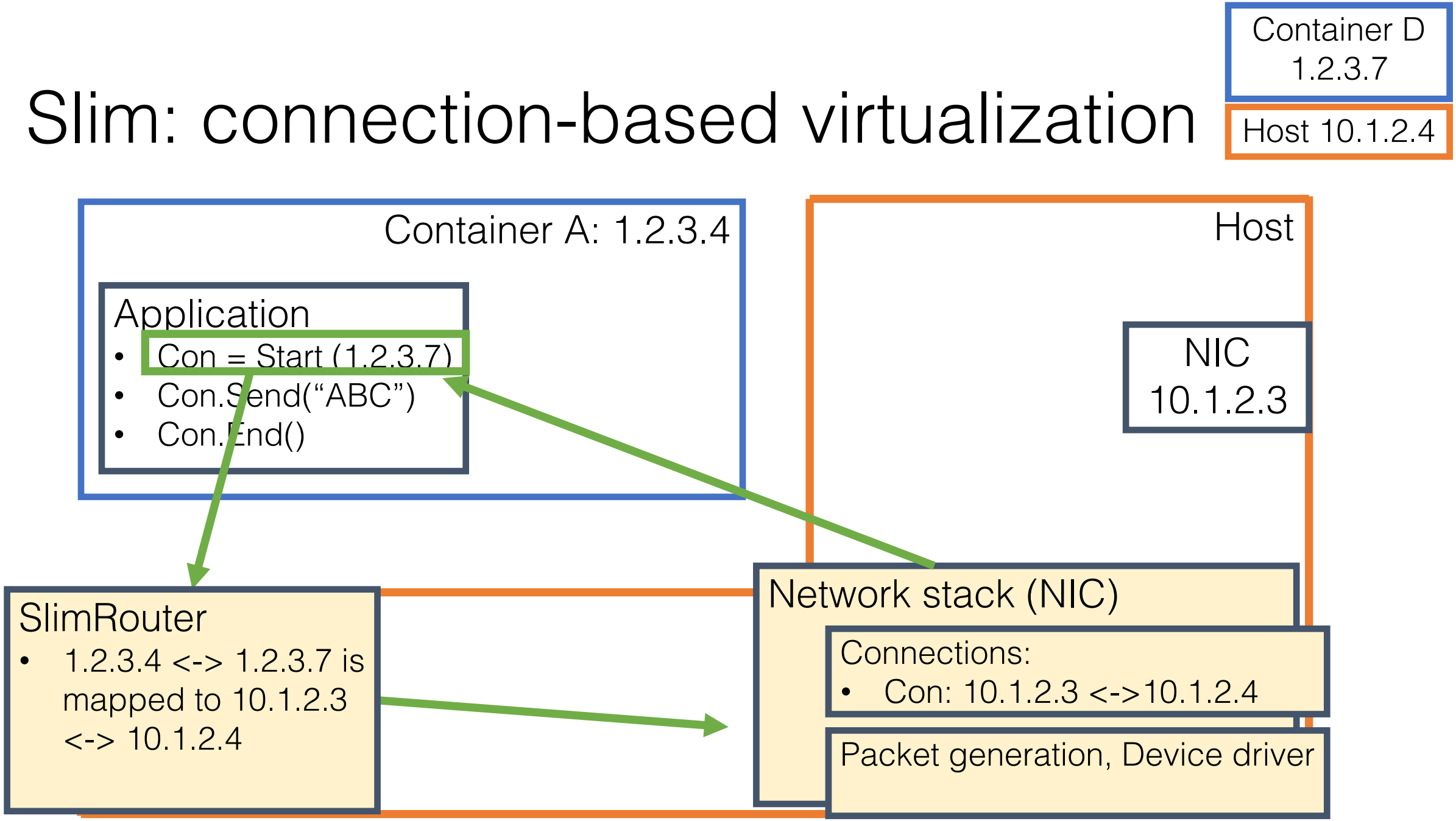
Packet generation, Device driver

How do we enforce security?

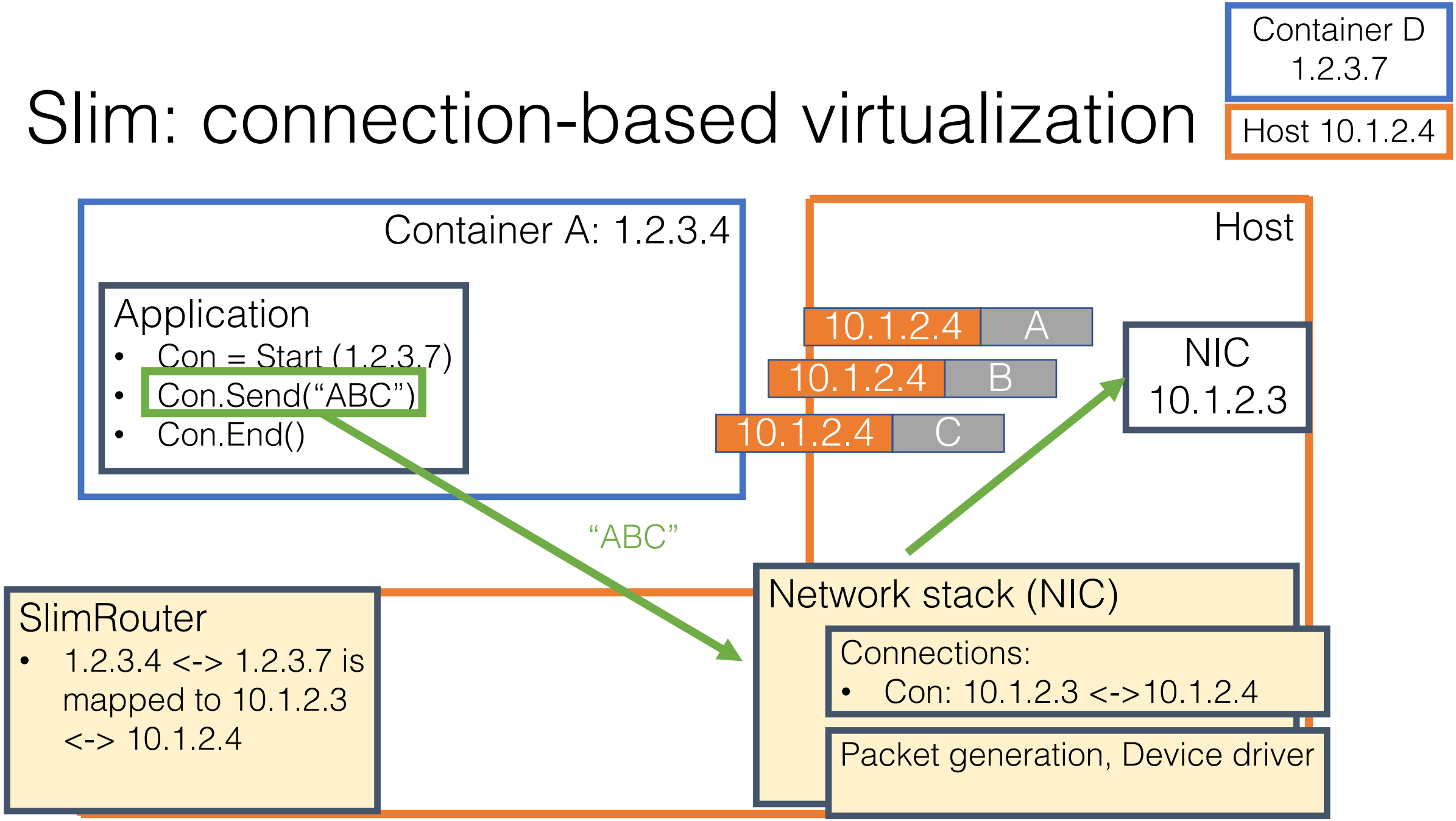
Slim: connection-based virtualization



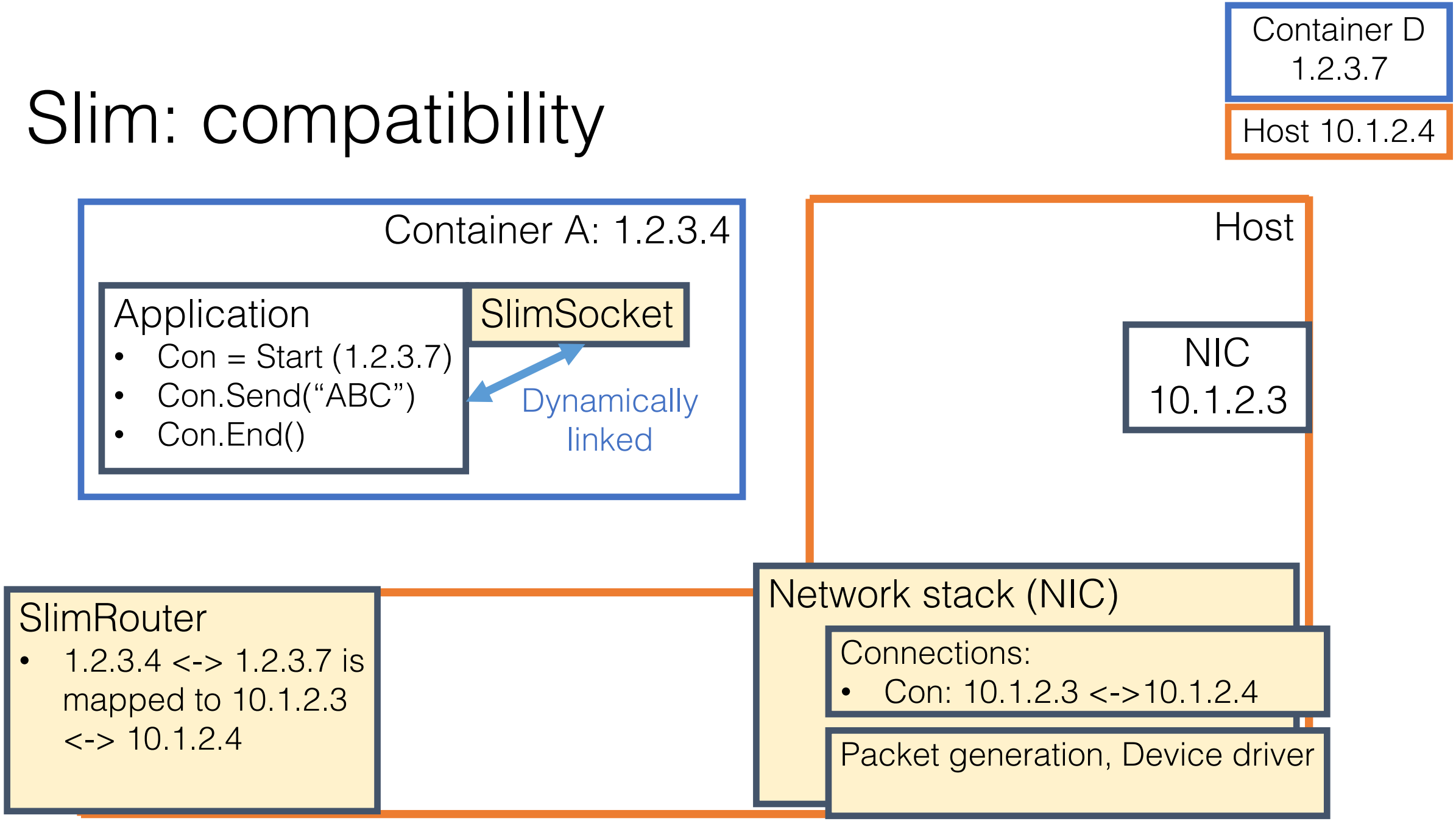
Slim: connection-based virtualization

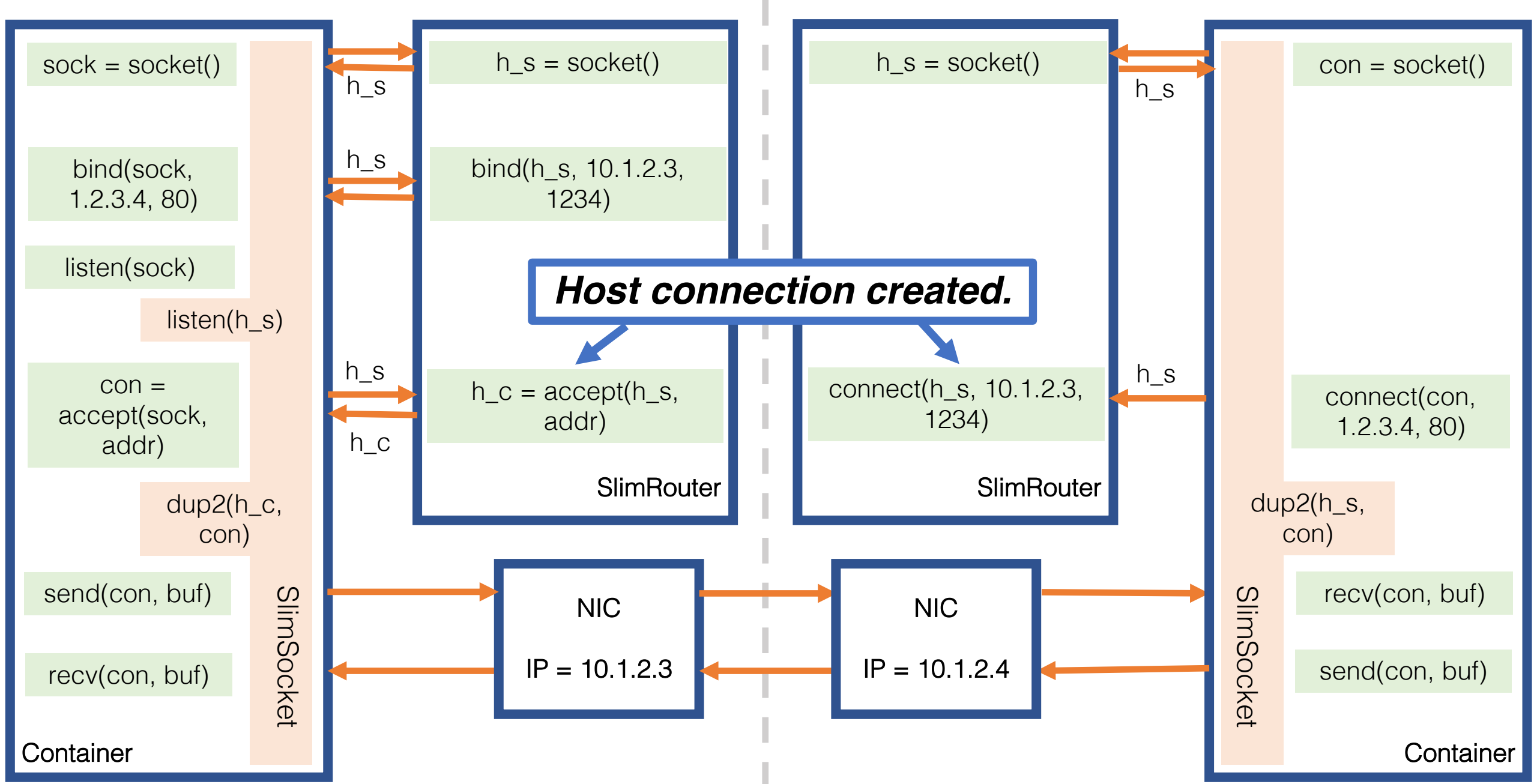


Slim: connection-based virtualization



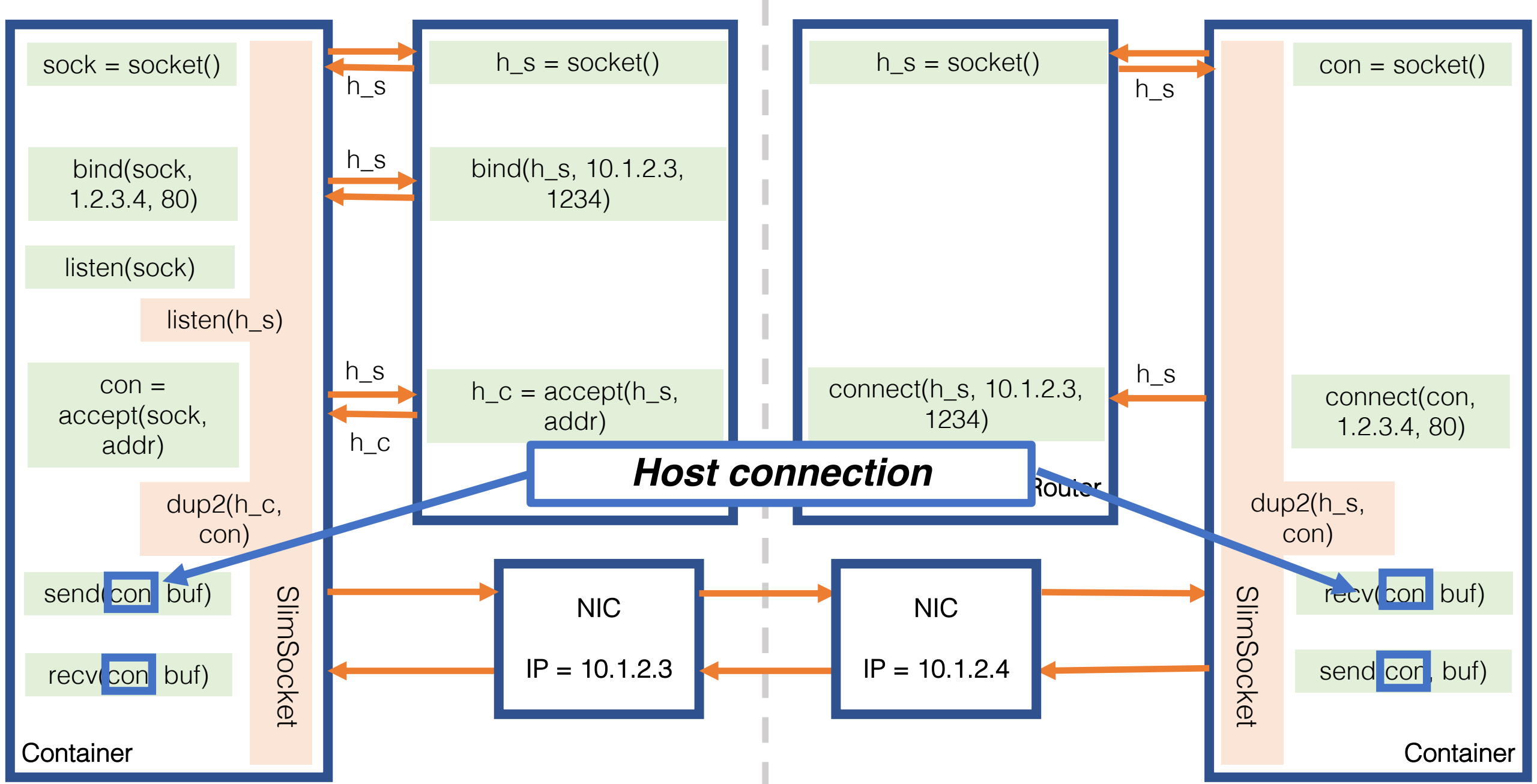
Slim: compatibility





(a) Web server

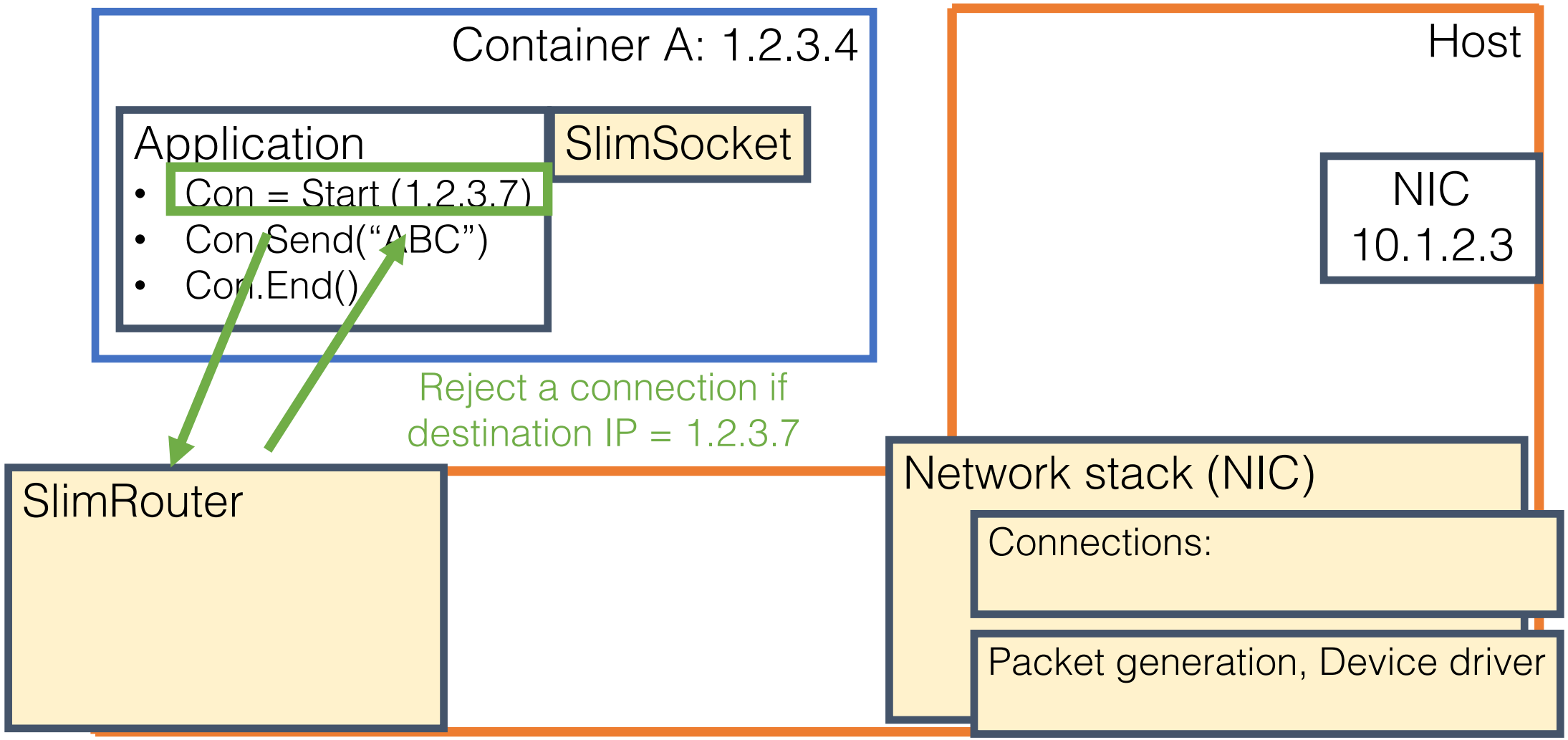
(b) Web client



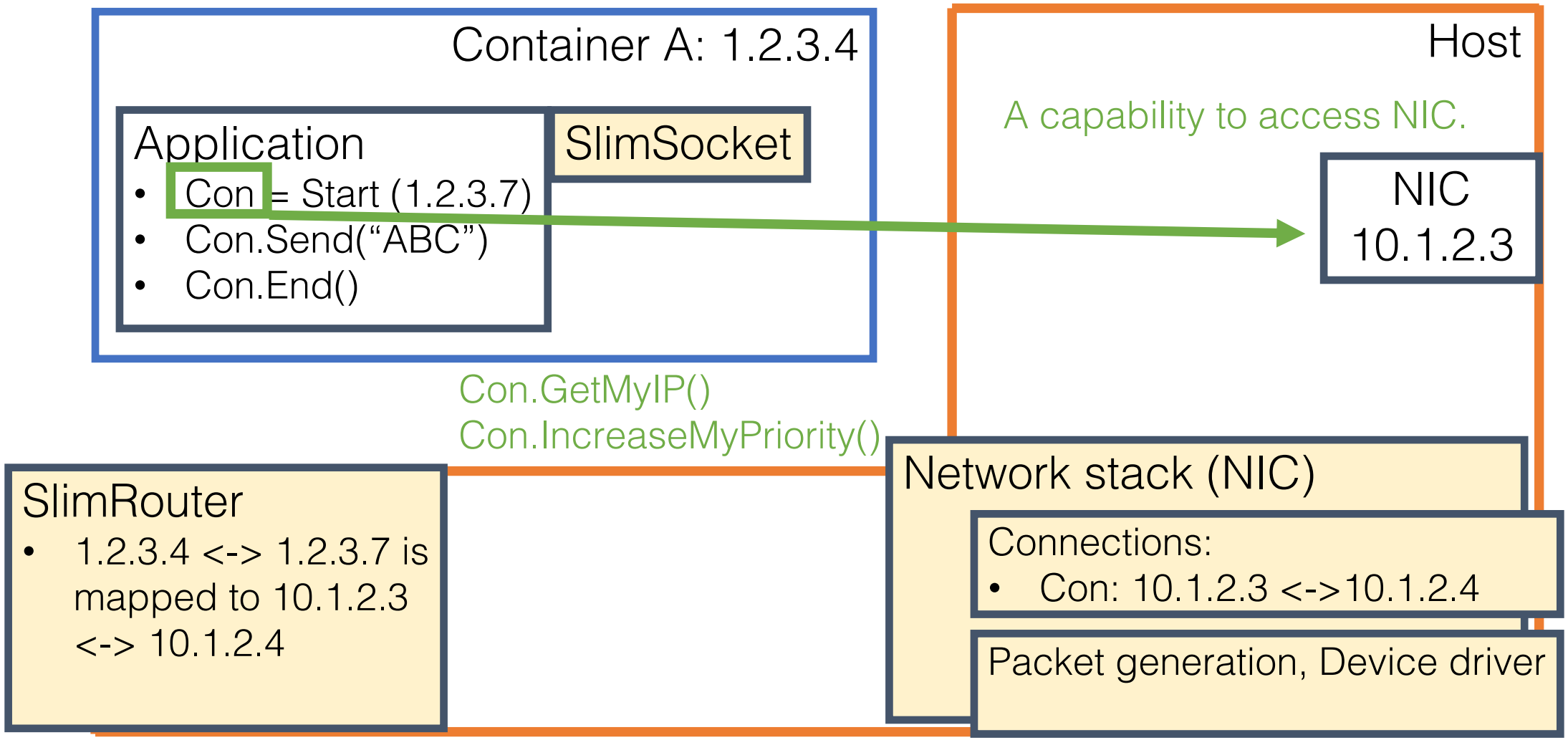
(a) Web server

(b) Web client

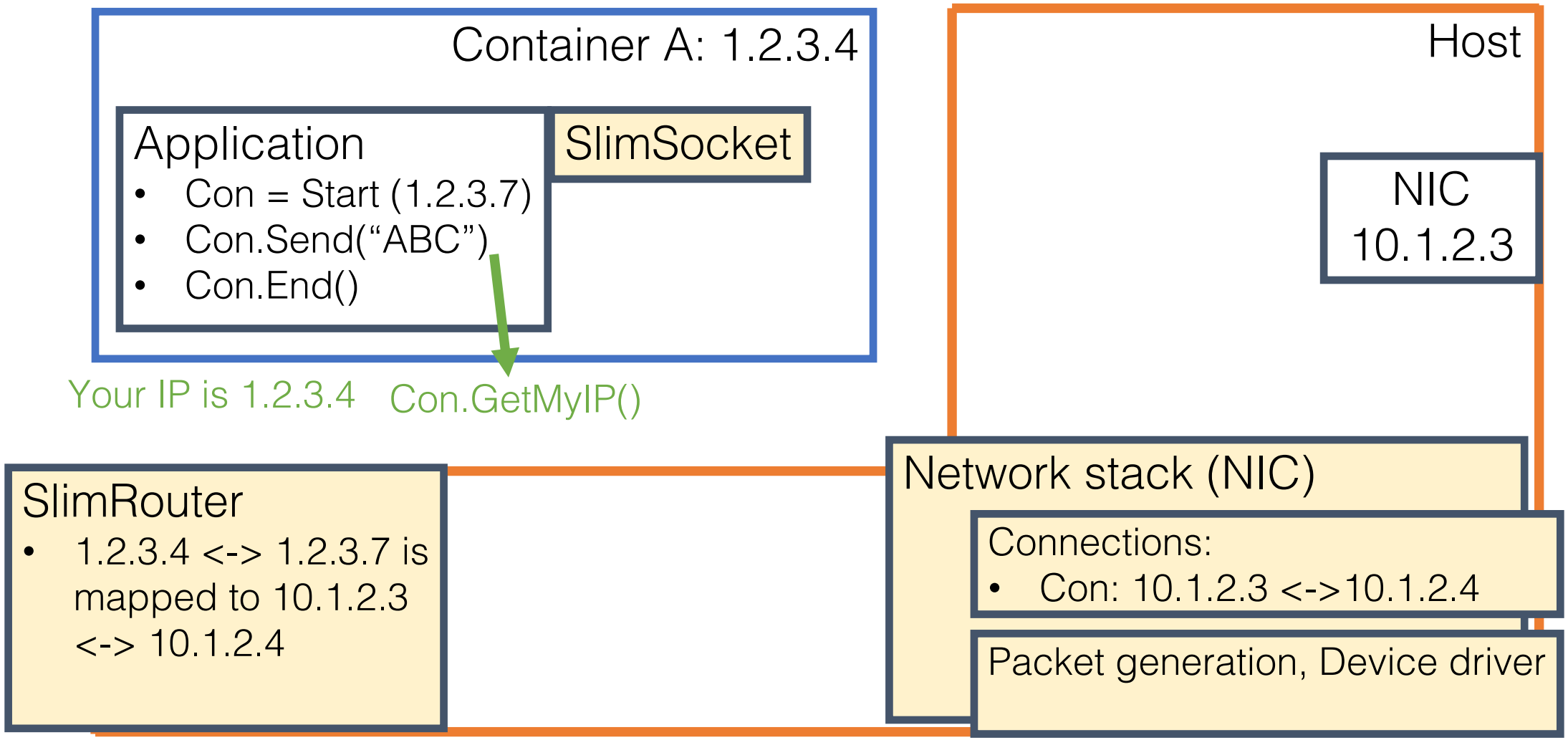
Slim: Support network policies



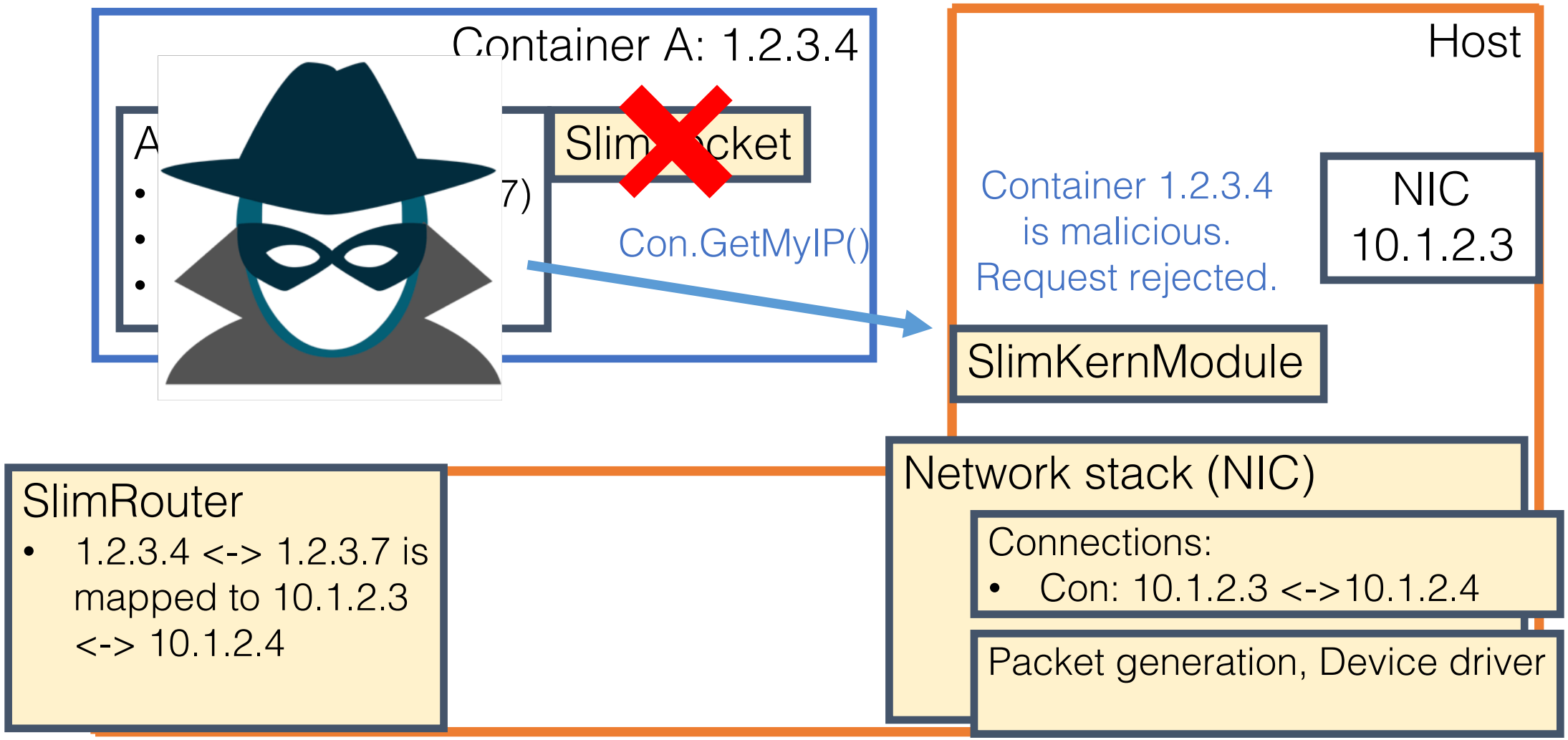
Slim: Security enforcement



Slim: Security enforcement



Slim: Security enforcement



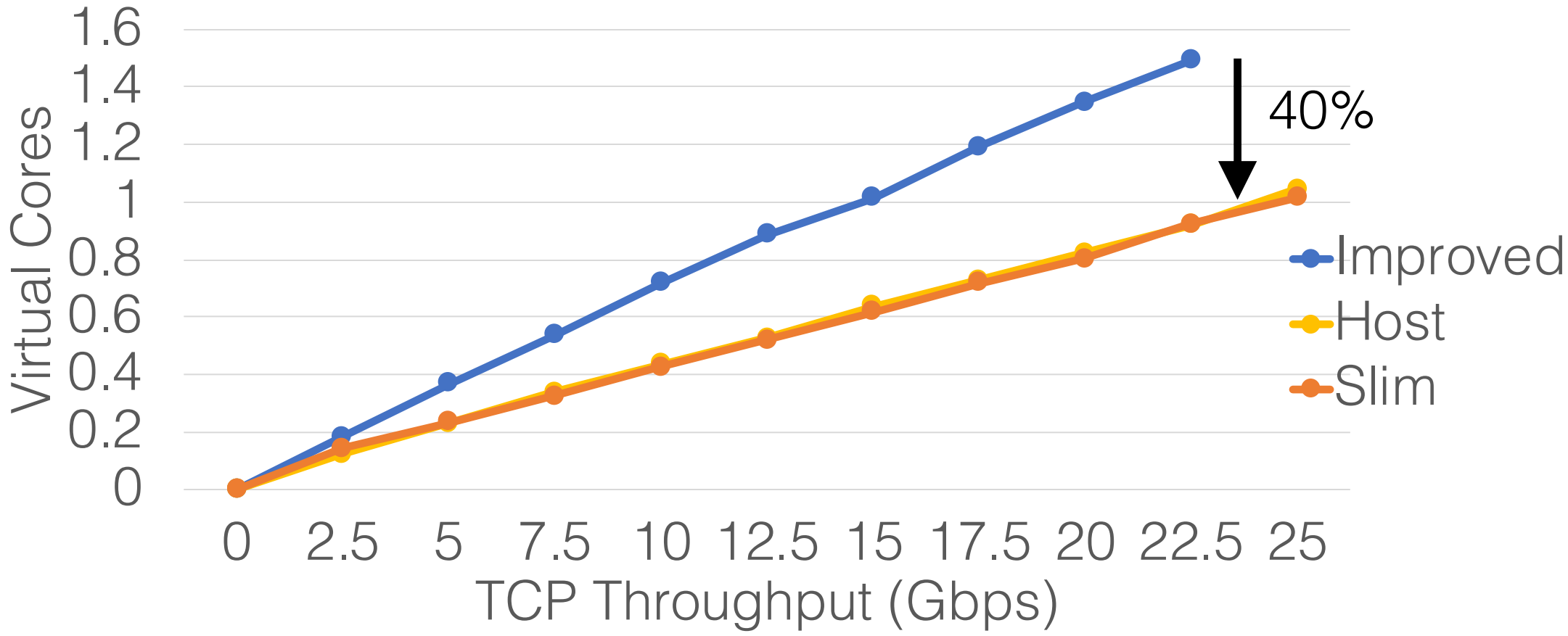
In this talk…

- Existing approach: Packet-based network virtualization results in high overheads.
- Slim: connection-based network virtualization that is compatible with existing Linux applications.
- Saving up to 56% CPU cycles on popular cloud applications (e.g., Memcached, Nginx, PostgreSQL, Apache Kafka).

Microbenchmark: performance

Setup	Throughput	Latency (RTT)
Vanilla	14.0 Gbps (48%)	20.9 us (85%)
Improved	24.5 Gbps (9%)	21.2 us (88%)
Slim	26.8 Gbps	11.4 us
Host	26.8 Gbps	11.3 us

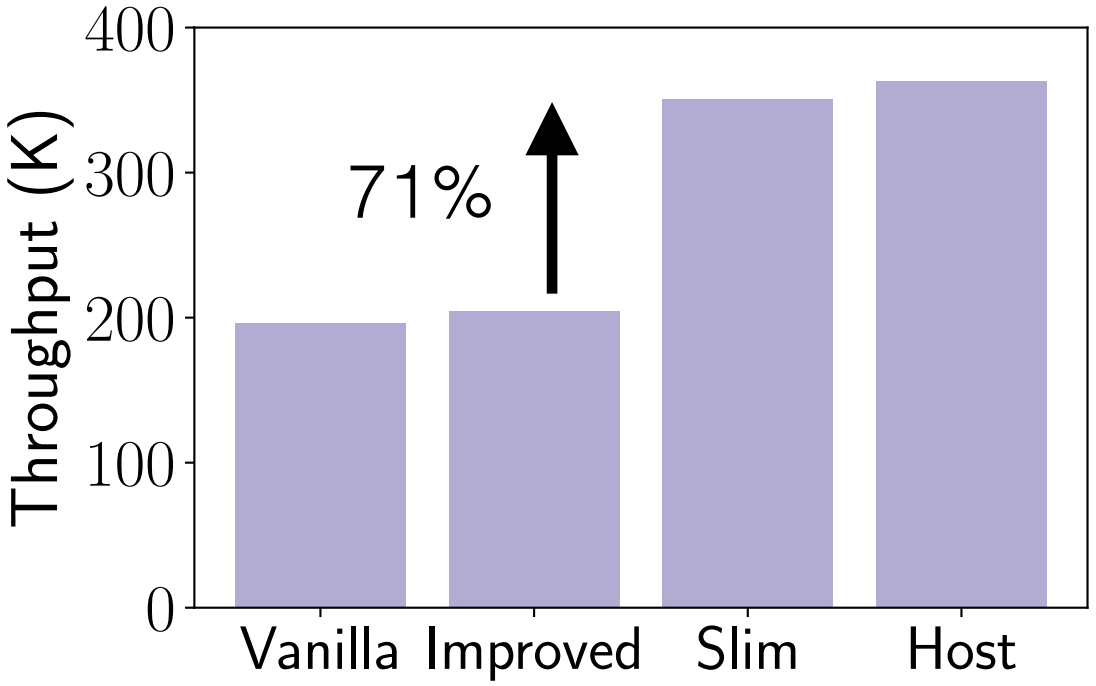
Microbenchmark: CPU

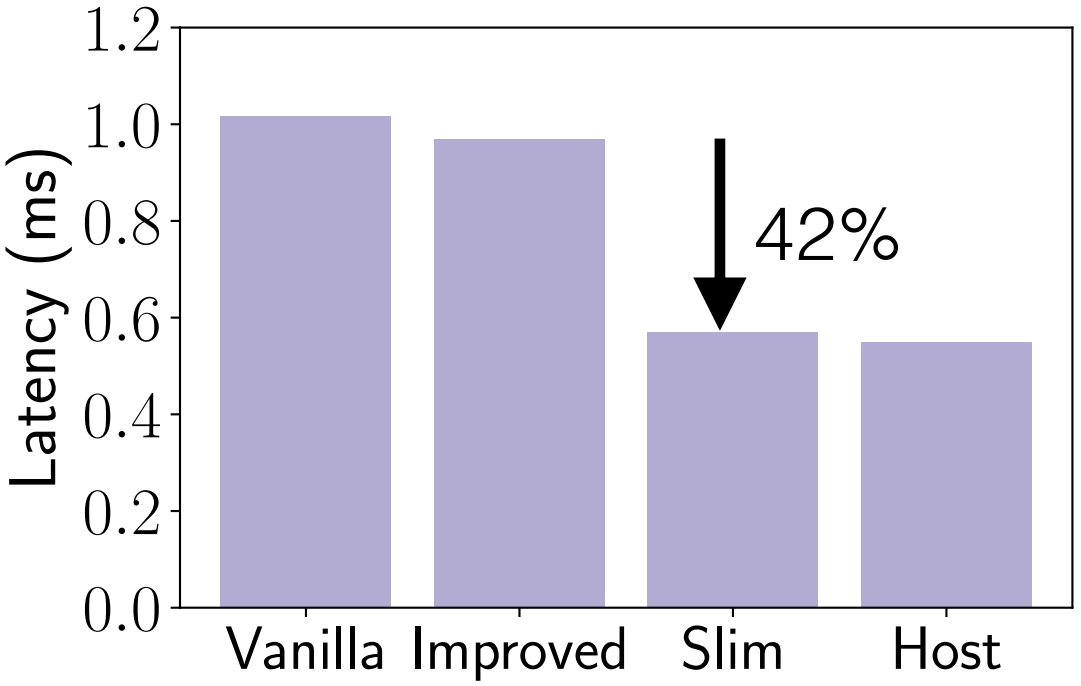


Evaluation: applications

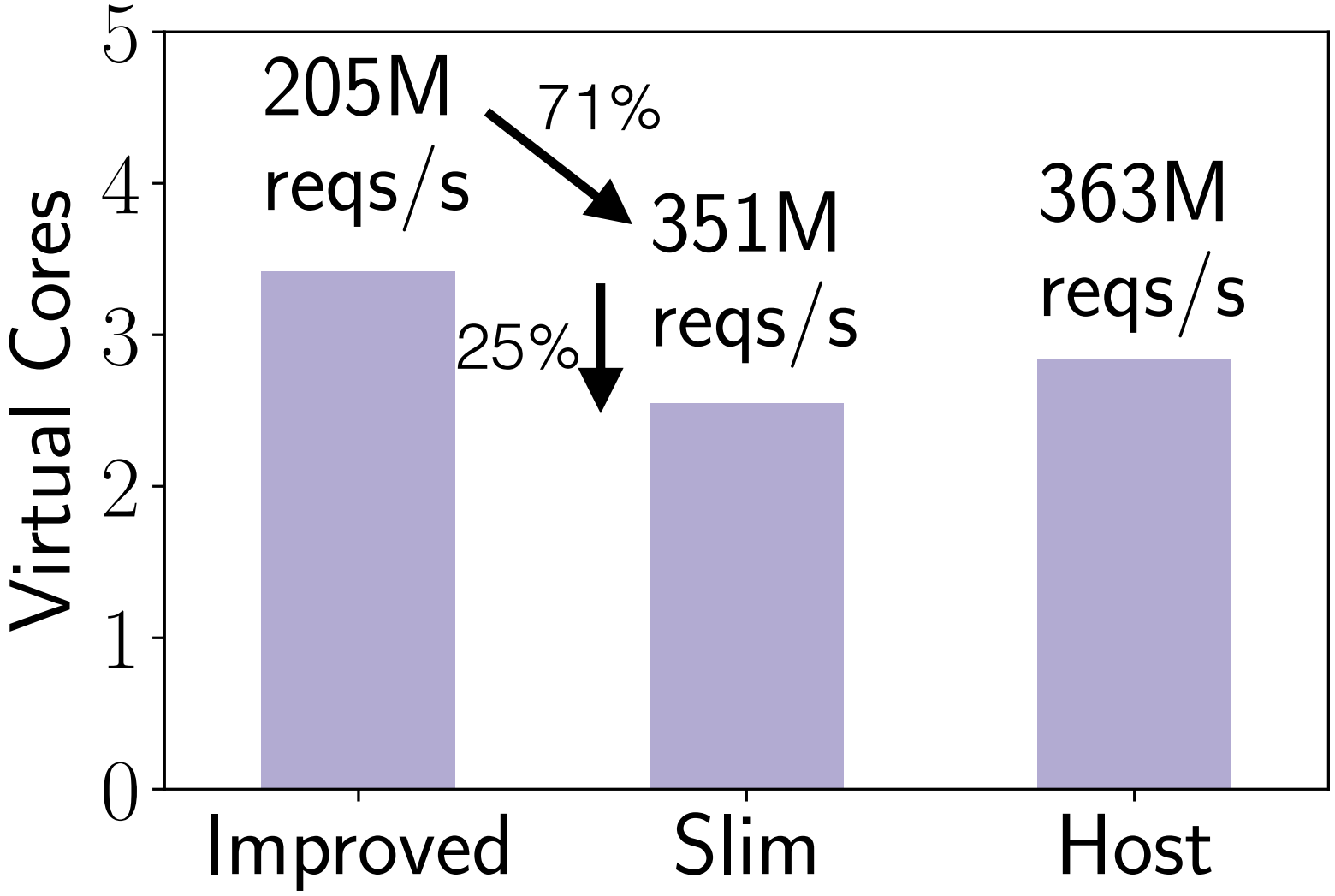
- In-memory key-value store
  - Memcached 1.5.6
- Web server
  - Nginx 1.10.3
- Database
  - PostgreSQL 9.5
- Stream-processing framework
  - Kafka 2.0.0

Evaluation: Memcached performance





Evaluation: Memcached CPU



CPU reduction:
56%

Evaluations: CPU

Application	CPU utilization reduction
Memcached	56%
Nginx	24%
PostgreSQL	22%
Apache Kafka	10%

Limitation

- Connection setup time is longer
  - 270 us -> 556 us
- Limited support for packet-based network policies
- Cannot support unmodified low-level network tools
- Cannot speed up datagram sockets (i.e., UDP)

Summary

- Packet-based network virtualization results in high overheads in terms of throughput, latency, and CPU utilization.

- Slim integrates efficient connection-based network virtualization support natively into the OS network stack.
  - Saving up to 56% CPU cycles on popular cloud applications.

- Open sourced at https://github.com/danyangz/slim.