Optimizing Intra-Node Networking with Live-Migration-Based Colocation

Doctoral Pre-defense

Jihong Min (Ph.D. Candidate, Dept. of CSE)

Computer Systems and Platforms Lab

Department of Computer Science and Engineering

Seoul National University

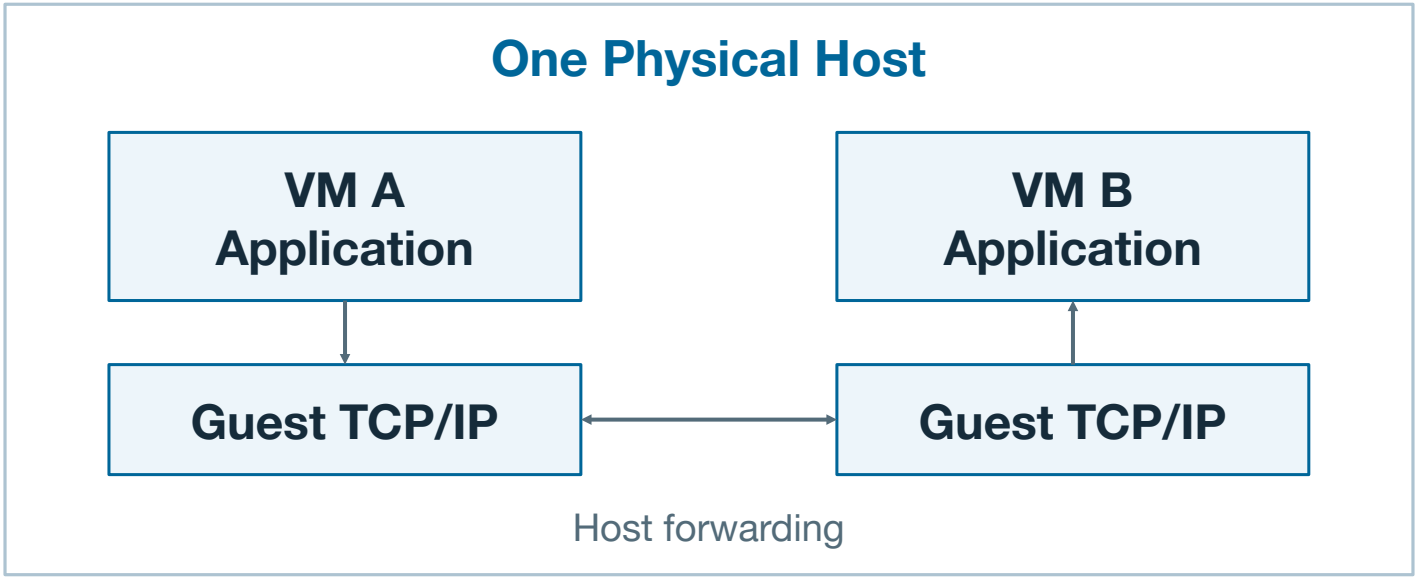
Intra-Node TCP vs Shared-Memory IPC

Linux TCP Socket		Lockless Shared-Memory Queue
Round-Trip Latency		
11 µs	**44× lower**	**0.25 µs**
Single-Core Message Throughput		
0.9 M op/s	**30× higher**	**27 M op/s**

8-byte messages. SocksDirect, SIGCOMM 2019, Table 2: measured IPC reference.

Colocated Apps Still Use the Network Stack

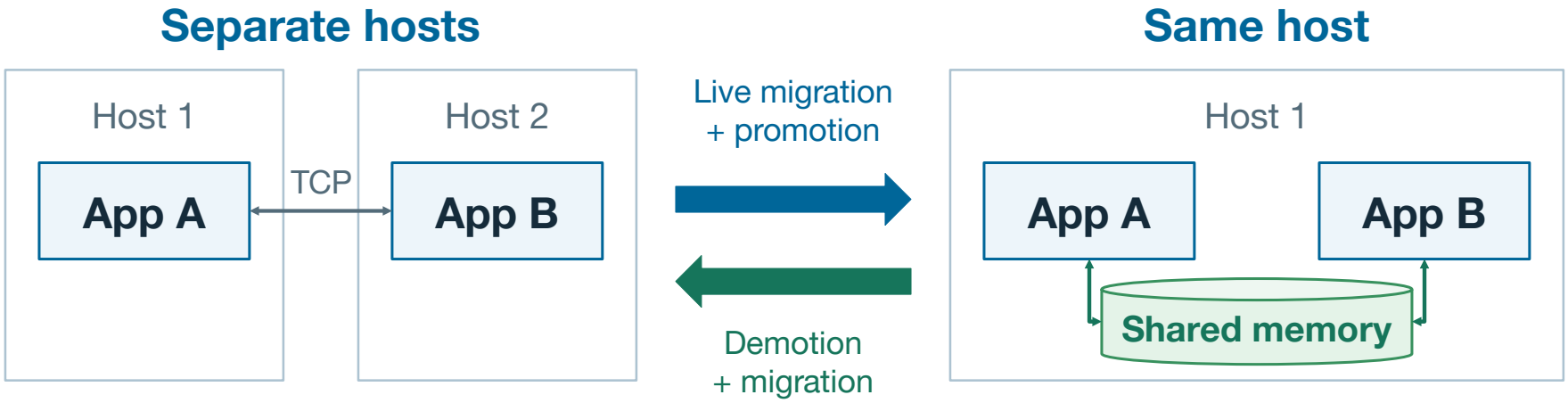
VMs share a host through placement or live migration.



Their sockets still use the network stack.

Goal of this Thesis

Transparently route network traffic of **colocated** applications
through **shared memory** and **retain connectivity** across live migration.



This idea applies to VMs, containers, and host processes.

Android Automotive Example

Example use case: Android Automotive - gRPC [1] across isolated display VMs

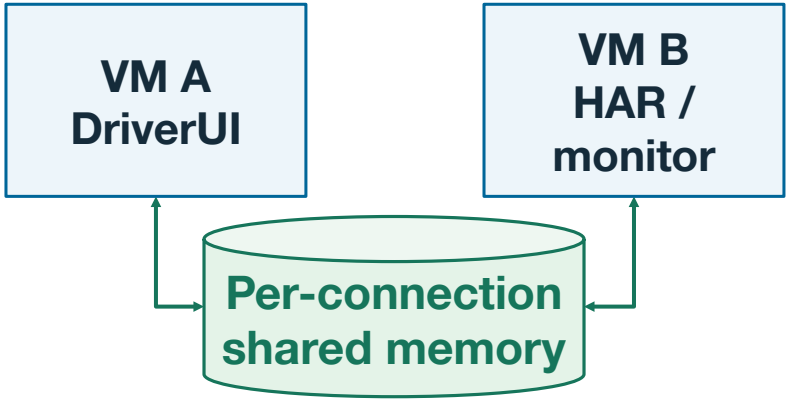
Network transport

TCP/IP retained



Direct shared memory

No TCP/IP encapsulation



[1] gRPC is an RPC protocol based on HTTP/2.

Traffic Within Cloud Infrastructure

Within a data center [1]

71.5%

of forecast data-center traffic stays within one data center

2021 forecast; Cisco GCI (2018)
Traffic within a rack excluded.

Across customer tenants [2]

10–40%

of measured internal customer-VM traffic crosses tenant boundaries

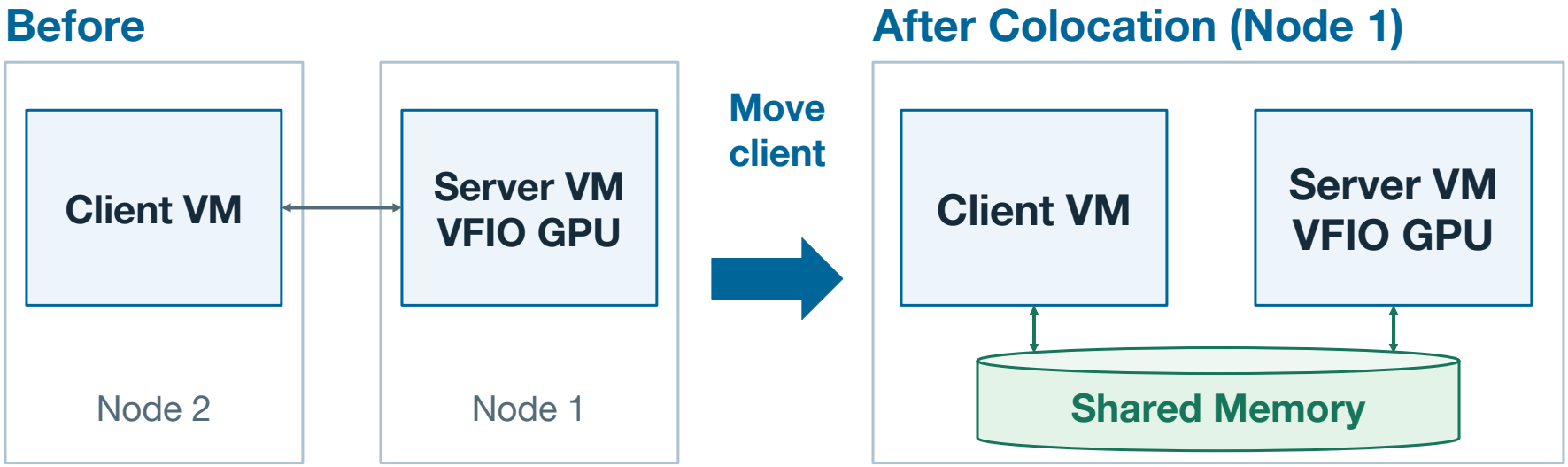
8 datacenters; measured Aug. 2012
Customer-VM traffic only.

A customer cannot merge another tenant's workload into its own guest.
Live migration can colocate their separate VMs / containers.

[1] Cisco Global Cloud Index 2016–2021, Figure 5.
[2] Ballani et al., Chatty Tenants and the Cloud Network Sharing Problem, NSDI 2013, §2.1.

Intra-Cloud Colocation Example

Example use case: NVIDIA Triton Inference Server



HTTP / gRPC across hosts

Same HTTP / gRPC connection
Local transport without TCP/IP

The GPU service stays in place. The connection must survive.

Requirements for Transparent Communication

Replace ongoing TCP communication with shared memory, transparently.

Transparency	Keep application binaries, socket behavior and connections.
Minimal changes	Use standard OS and hypervisor interfaces.
Maintainability	Avoid platform-internal patches and custom TCP stacks.

Related Work: Five Transport Families

Payload hot paths of transparent socket approaches

Approach	TCP	Kernel I/O	SHM Scope	Transport Hot Swapping
Kernel TCP forwarding	Yes	Yes	Packet rings	Supported
Userspace TCP	Yes	Varies	App–stack IPC	Varies
Kernel IPC	No	Yes	VM / local	VM paths
Userspace SHM	No	Varies	Same kernel	Same kernel
RDMA	No	Varies	Service IPC	Varies
USERNET	**No**	**No**	**All 3**	**All 3**

Varies by implementation. All 3: VM / container / process. Hot swapping preserves connections.

USERNET combines userspace byte queues with transport hot swapping in all 3 domains.

Contributions and Publications

1 **OverlaySys** — GECON 2026

Transparent syscall and signal interception.

2 **USERNET** — Under review

Direct shared-memory sockets with userspace waiting and readiness.

2A **Transport Hot Swapping**

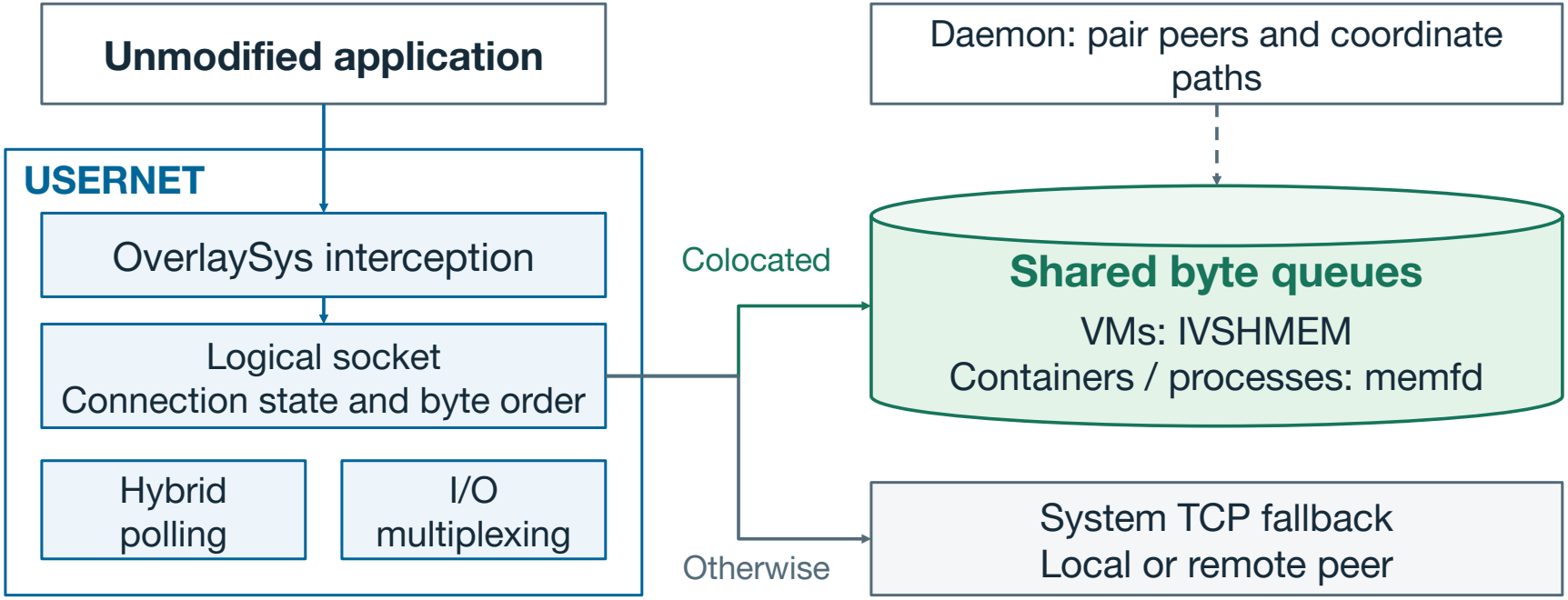
Connection-preserving migration across VM, container, and process pairs.

2B **Broad Comparative Evaluation**

About 30 systems: performance, colocation and live migration.

Architecture Overview

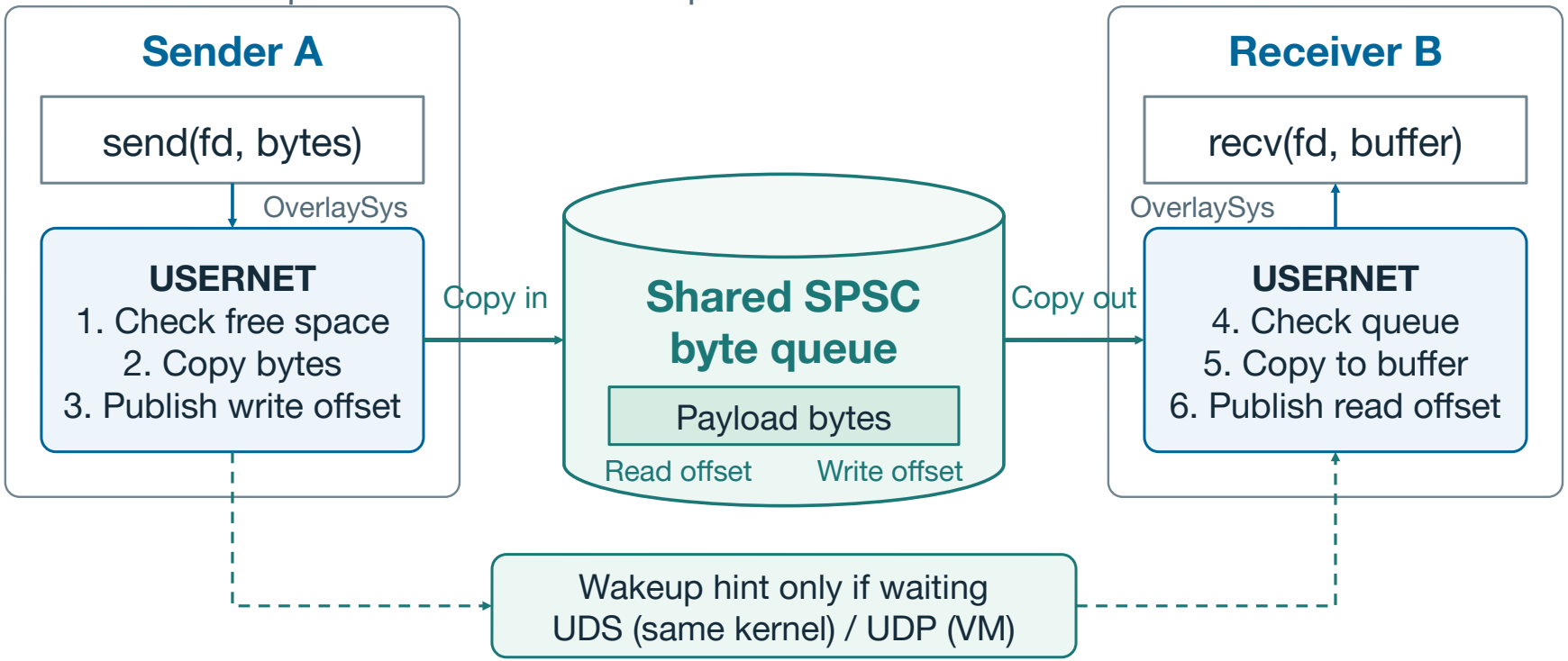
One logical socket layer for VMs, containers and host processes.



Wakeup hints: UDP between VMs; UDS within a kernel.

Send/Recv Walk-Through

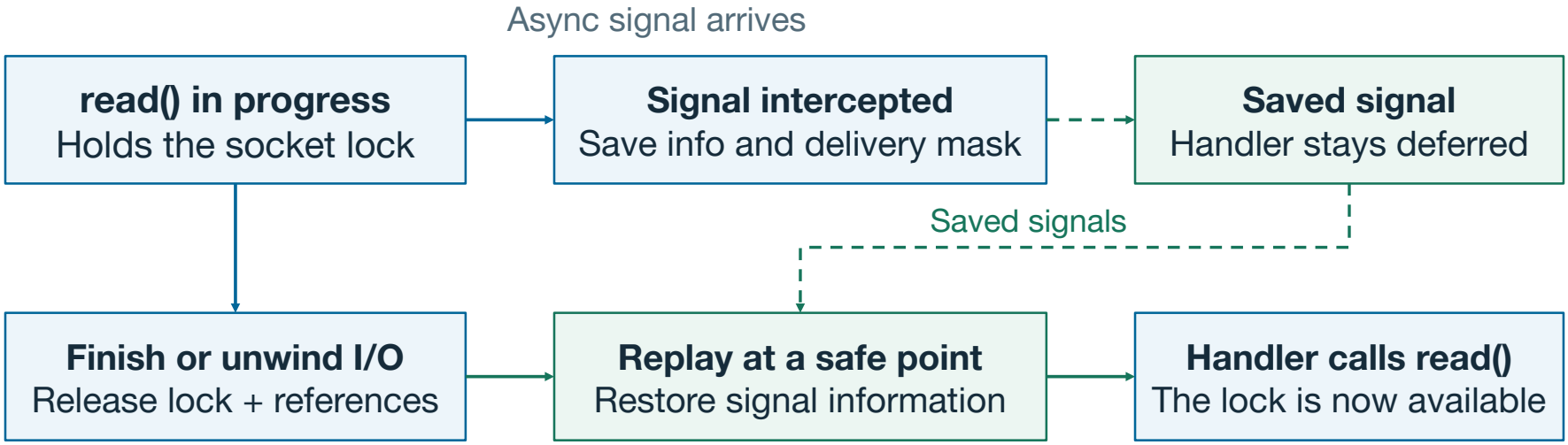
Established, colocated connection.
Per-connection queue bounds / cursor updates are validated.



USERNET Interception with OverlaySys

Challenge: a signal handler may request the lock held by interrupted I/O.

OverlaySys intercepts signals. USERNET defers handlers until resources are released.



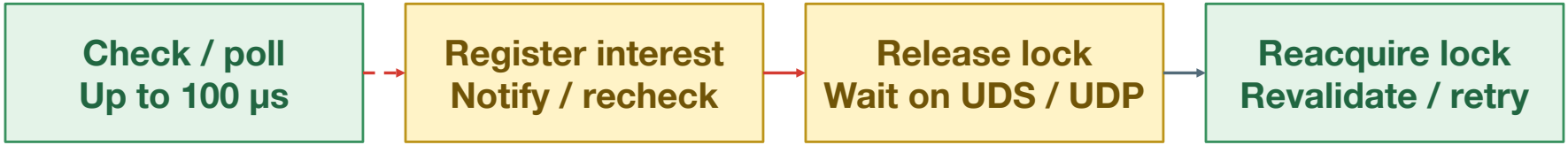
The handler can re-enter USERNET without waiting on its interrupted operation.

Hybrid Polling

Challenge: keep short waits fast without wasting a CPU on long waits.

Linux block I/O sleeps then polls. USERNET polls first, then waits.

If data or space is ready, return immediately.



Wakeup Hints

UDS: same kernel
UDP: inter-VM

If a UDP Hint Is Lost

Recheck after at most 1 s of waiting.
Keep the original application deadline.

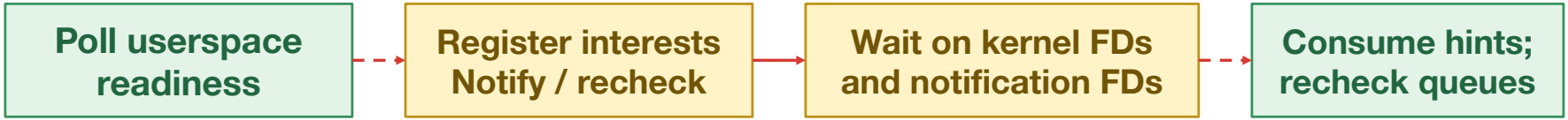
Short waits avoid blocking. Long waits avoid continuous polling.

I/O Multiplexing

Challenge: combine shared-memory readiness with other file descriptors.

select / poll / epoll_wait

Signal-mask variants: pselect / ppoll / epoll_pwait / epoll_pwait2



Return merged application events; block only if none are ready.

One readiness view

- Merge shared-queue readiness with native FD events.
- Consume wakeup hints; return application events only.
- Keep deadlines and signal masks.

Default timing budgets

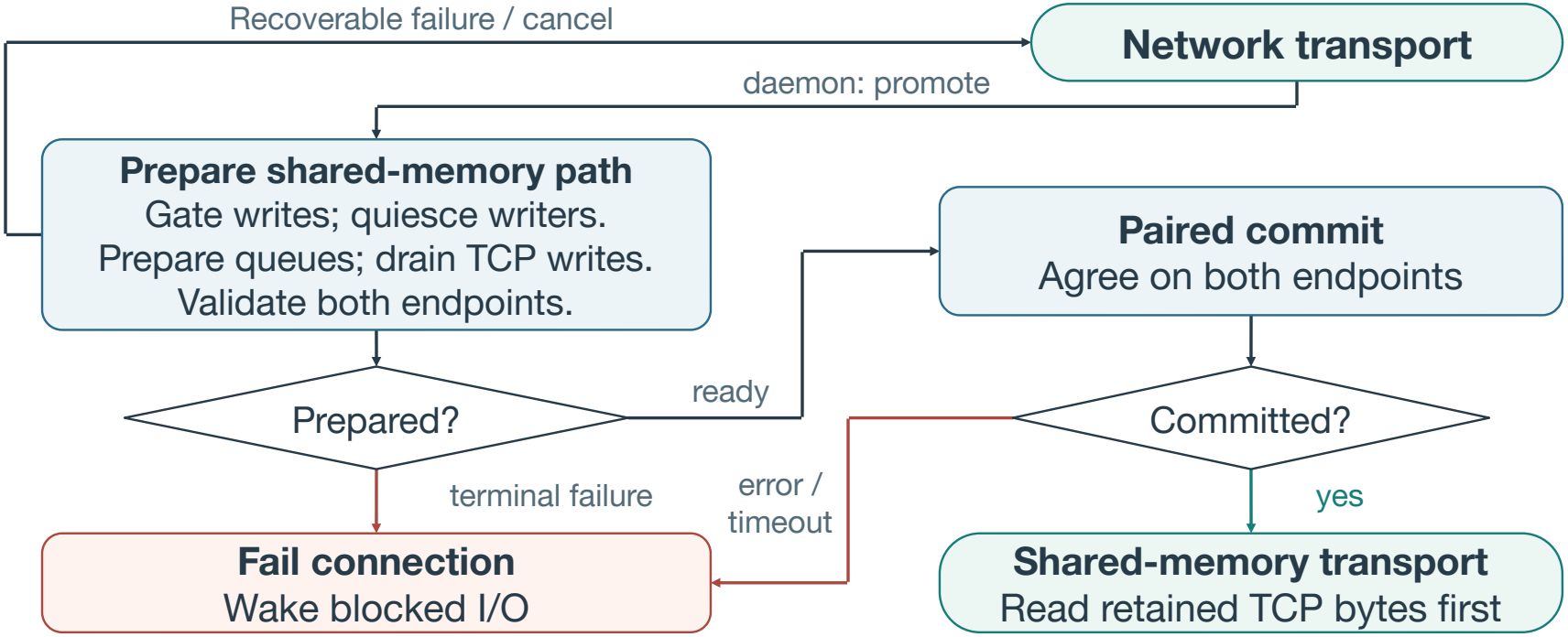
100 µs userspace scan budget

1 ms kernel recheck interval

A global check deadline preserves fairness for other FD types across calls.

Transport Hot Swapping - Promotion

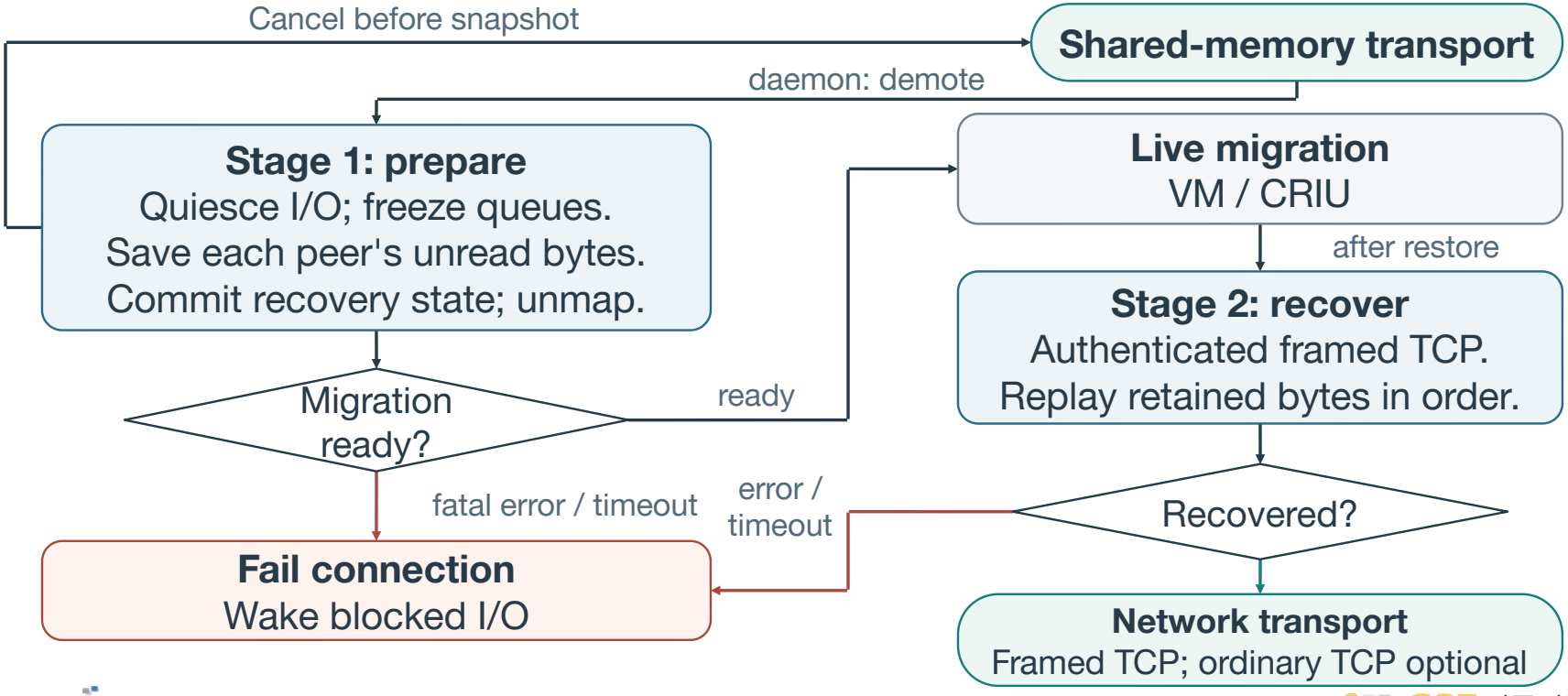
Challenge: switch an active TCP stream without losing byte order.



USERNET supports synchronous and asynchronous promotion.

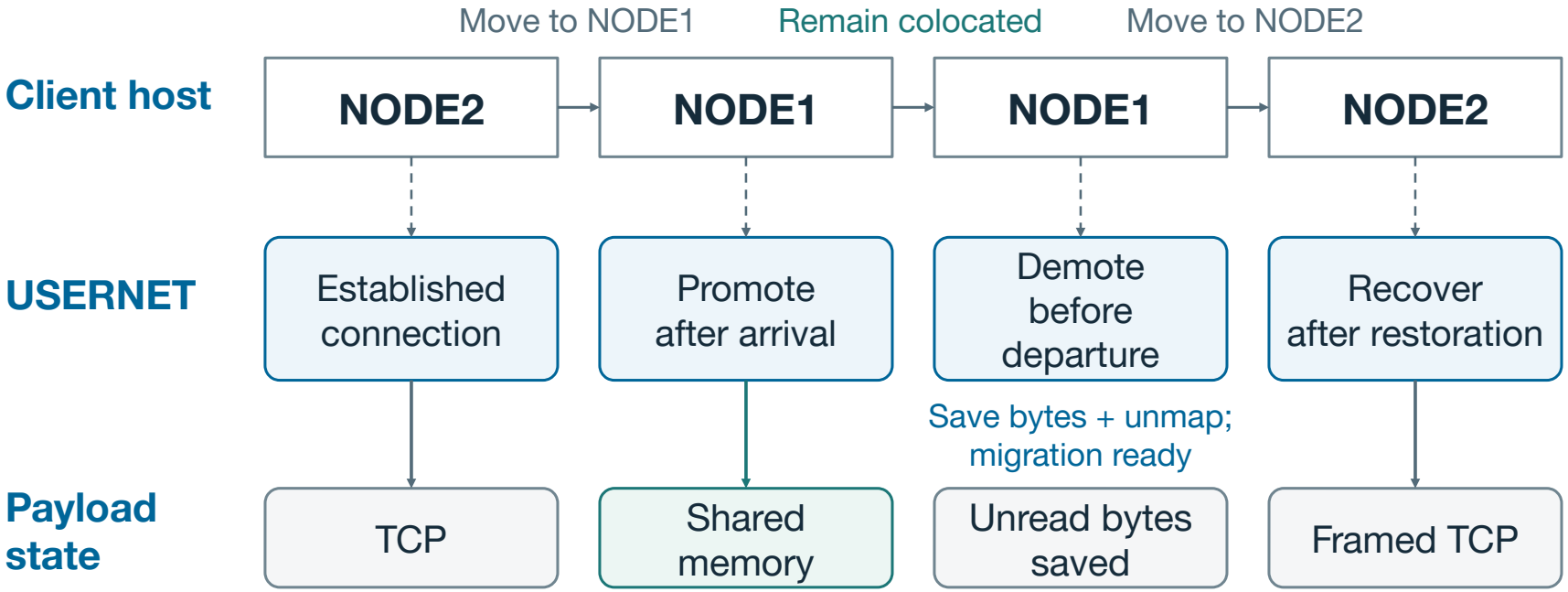
Transport Hot Swapping - Demotion

Challenge: retain unread bytes when shared memory cannot move.



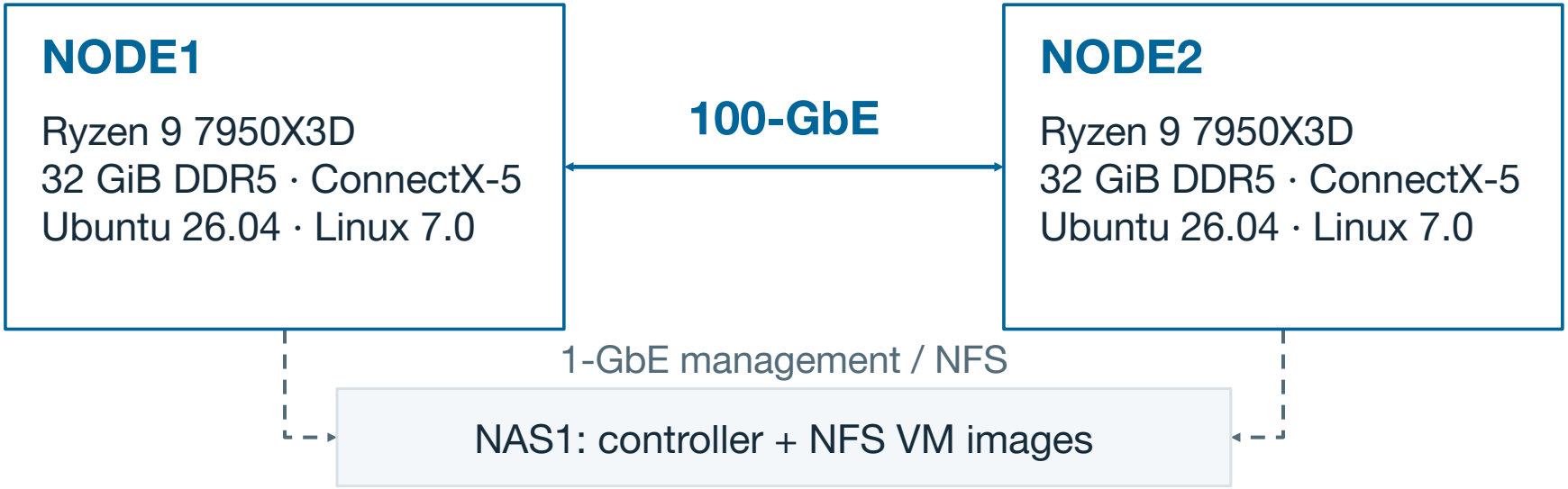
Migration Timeline Example

One client over time; the server remains on NODE1.



Application FDs stay unchanged; no application reconnection.

Evaluation Environment



6 physical cores per endpoint · 8 GiB per VM / container

Static tests: Both endpoints on NODE1

Migration tests: NODE1 <-> NODE2; client moves NODE2 → NODE1 → NODE2

Selecting Comparable Systems

About 30 systems on a common Ubuntu 26.04 platform

Screening

Netperf TCP_STREAM / TCP_RR
1 thread · 1 connection · 10 s
VM / container / process

Allowed Modifications

Platform ports / correctness fixes.
Preserve mechanisms and workloads.
Netperf / NetPIPE: socket fixes only.

Selection

1. Winners per domain and metric

Best STREAM and RR from all results, verified pre-copy and post-copy pools.

2. Require connection continuity

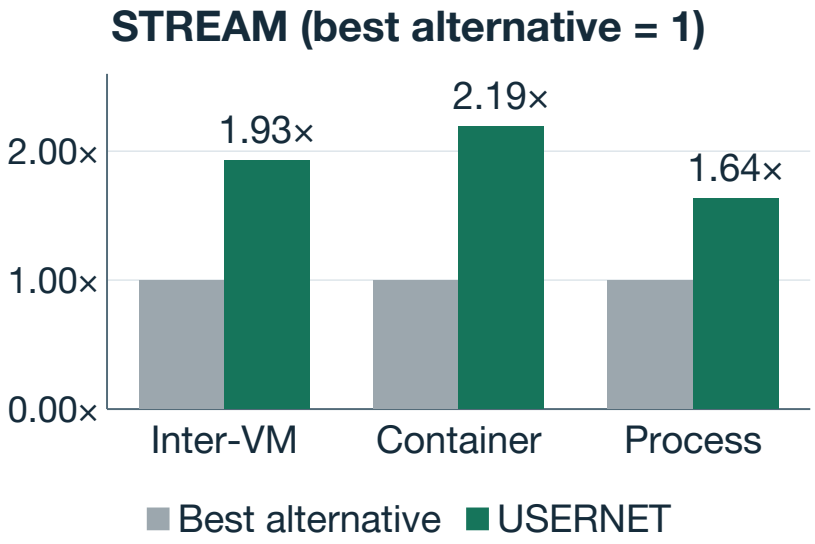
Migration pools require promotion and demotion without reconnection.

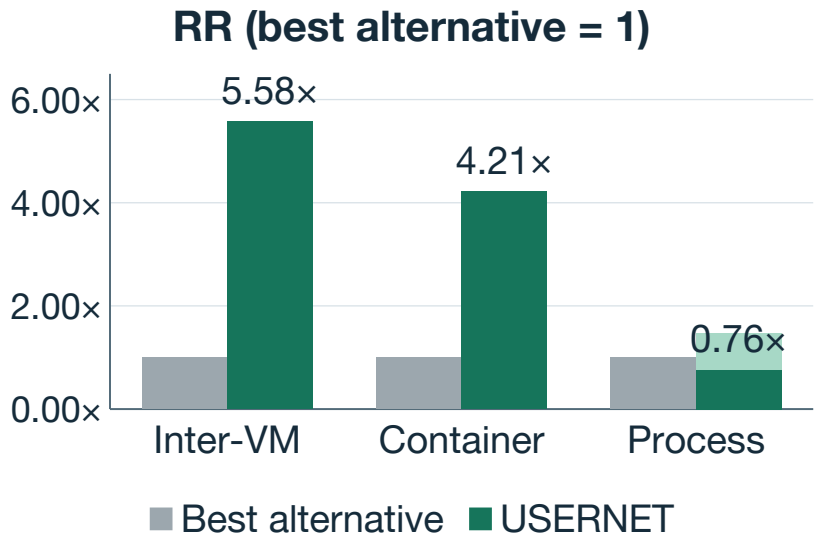
3. Deduplicate and add similar designs

Keep each winner once; add shared-memory and hybrid-waiting mechanism peers.

VM/Container: ~2× STREAM and 4–6× RR

Microbenchmark: Netperf TCP_STREAM / TCP_RR



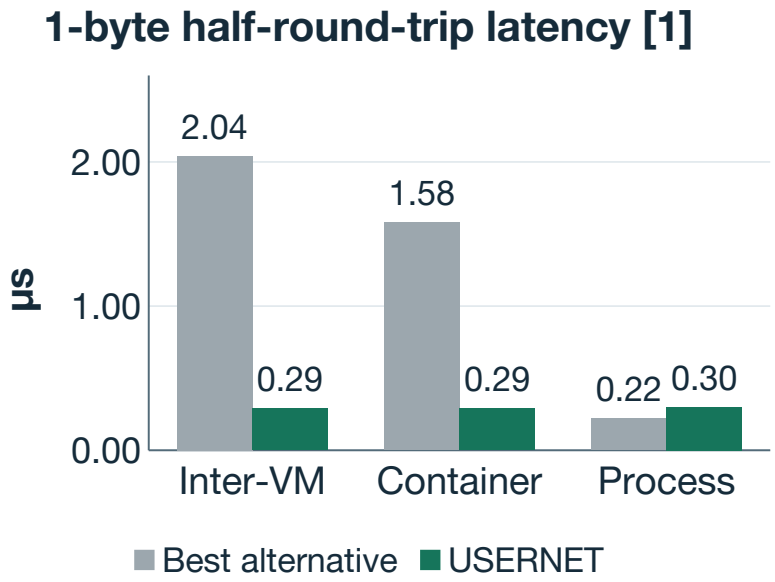


Inter-VM: 127.5 Gbit/s and 1.68 million transactions/s.

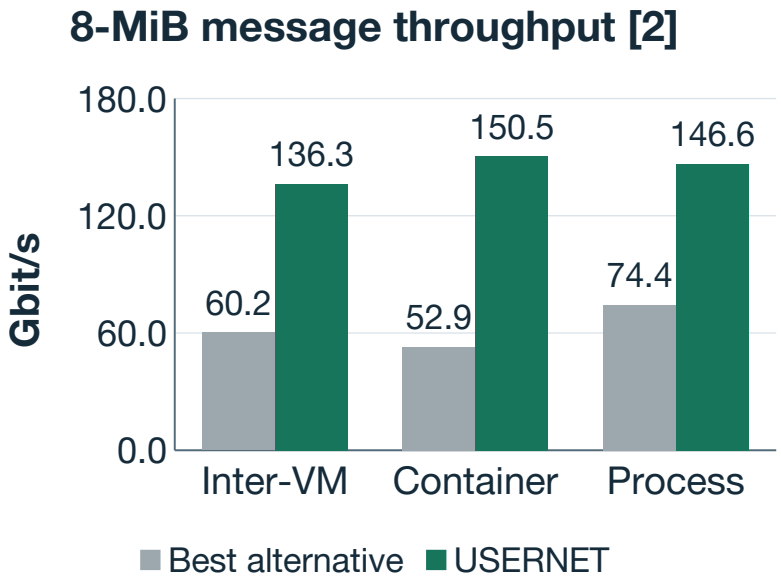
Process RR: USERNET is 0.76× Junction; its uProcs share one process/core.
USERNET used separate L3 groups. A separate same-L3 run reaches 1.46× Junction (pale bar: 3,296.54 vs. 2,254.60 ktrans/s).

Sub-Microsecond Latency and Over 130 Gbit/s

Microbenchmark: NetPIPE latency and throughput at two message sizes



Best: VMA / VMA / Junction



Best: OVS-DPDK VF / OVS-DPDK VF / SocksDirect (rework copy)

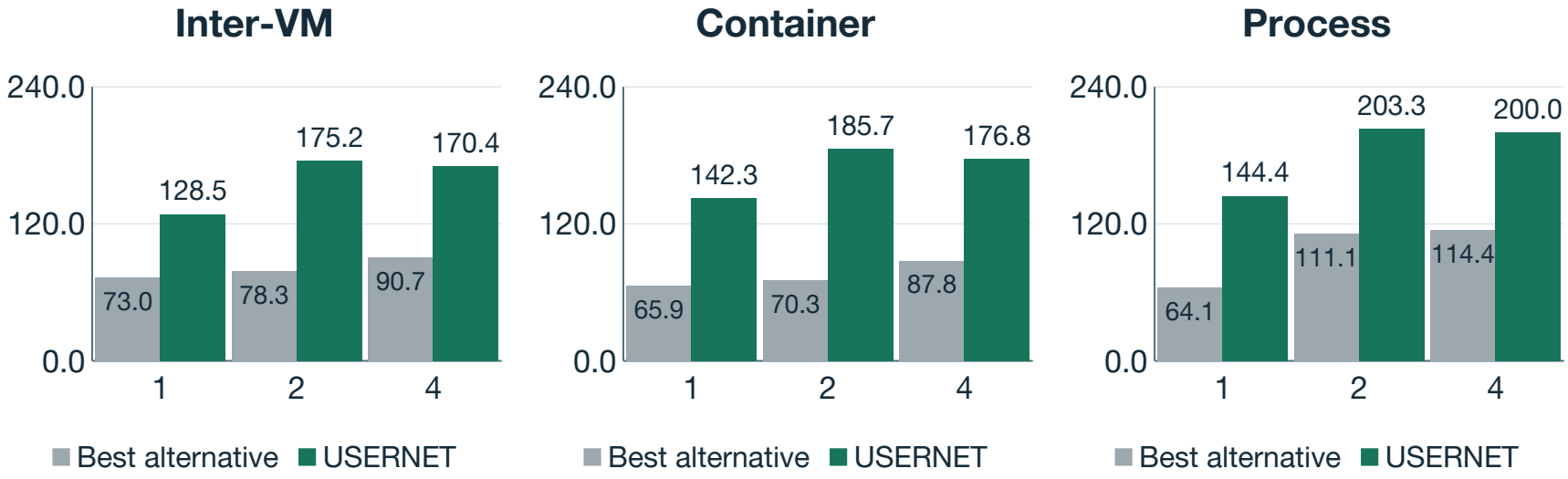
[1] NetPIPE's default starting message size.

[2] Largest power-of-two message size in NetPIPE's default sweep.

Bulk Throughput Rises, Then Reaches a Plateau

Microbenchmark: neper · 4 connections per worker · 1, 2 and 4 workers

128-KiB streaming blocks (tcp_stream -B 131072 ...) · Gbit/s



Best: VMA (1);
OVS-DPDK VF (2, 4)

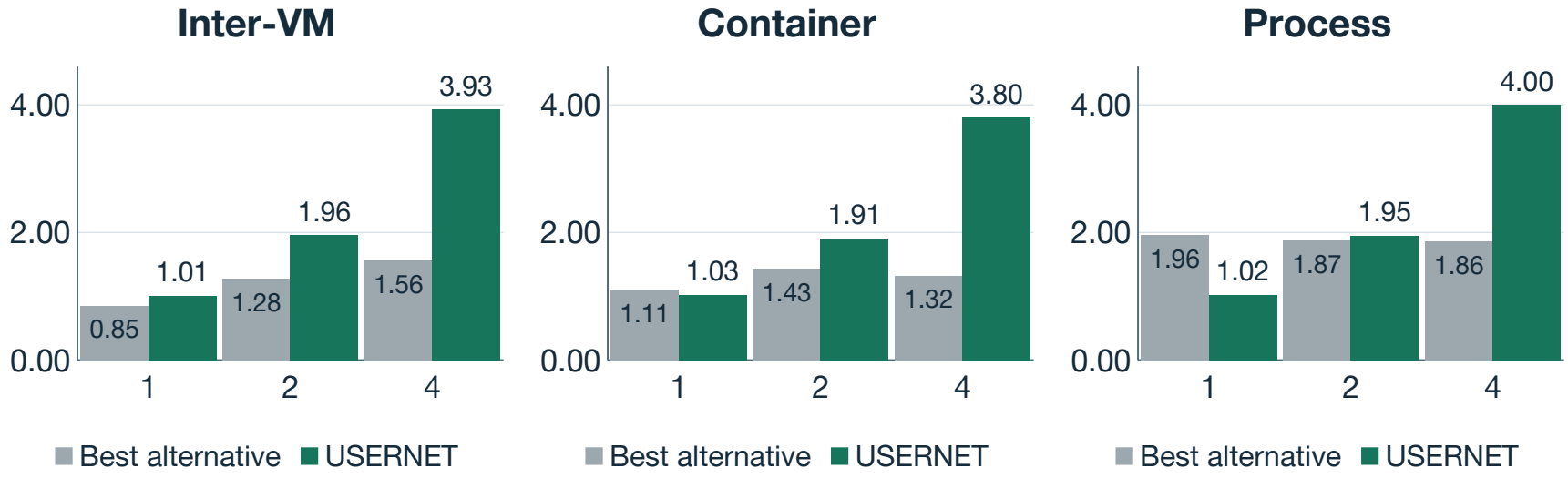
Best: VMA (1); ipvlan (2, 4)

Best: SocksDirect rework copy (1);
paper version (2, 4)

Request Rate Reaches 4 Million Transactions/s

Microbenchmark: neper · 4 connections per worker · 1, 2 and 4 workers

One-byte requests and responses [1] · Million transactions/s



Best: VMA at all worker counts

Best: VMA (1, 2); TAS (4)

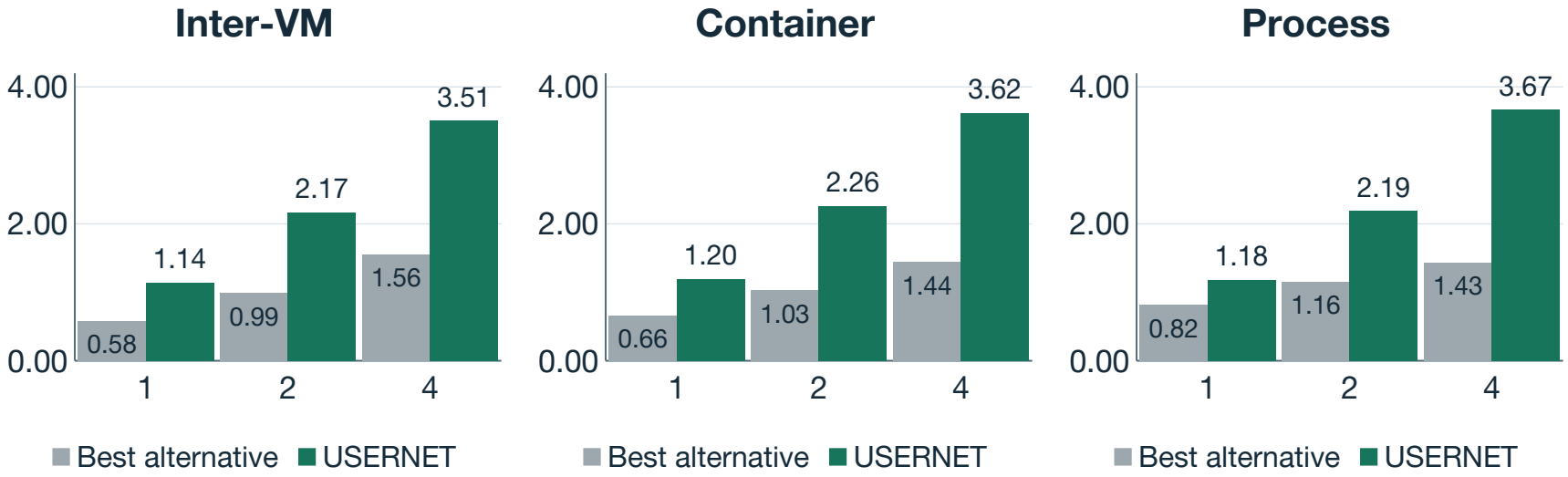
Best: Junction at all worker counts

[1] neper's default configuration.

Memcached Reaches 3.5–3.7 Million Operations/s

Application: memcached / memaslap · 4 connections per worker · 1, 2, 4 workers

1-KiB values [1] · Million operations/s



Best: VMA at all worker counts

Best: VMA at all worker counts

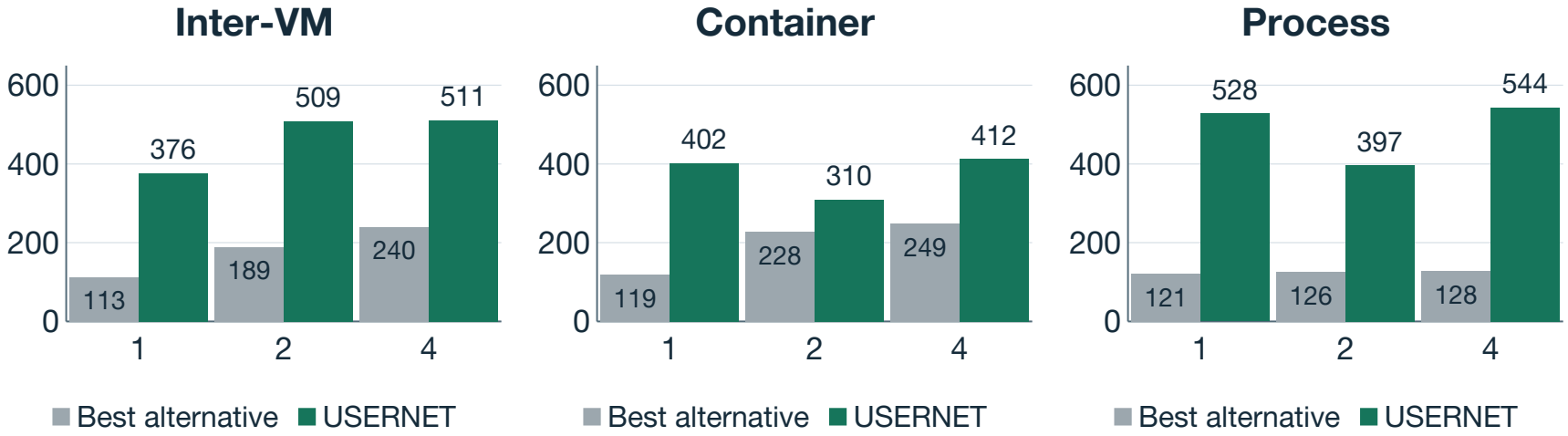
Best: TAS at all worker counts

[1] memaslap's default configuration.

nginx Serves More Requests with USERNET

Application: nginx / wrk · 4 connections per worker · 1, 2, 4 workers

HTTP keep-alive [1] · 3-byte response · Thousand requests/s



Best: OVS-DPDK vhost-user at all worker counts

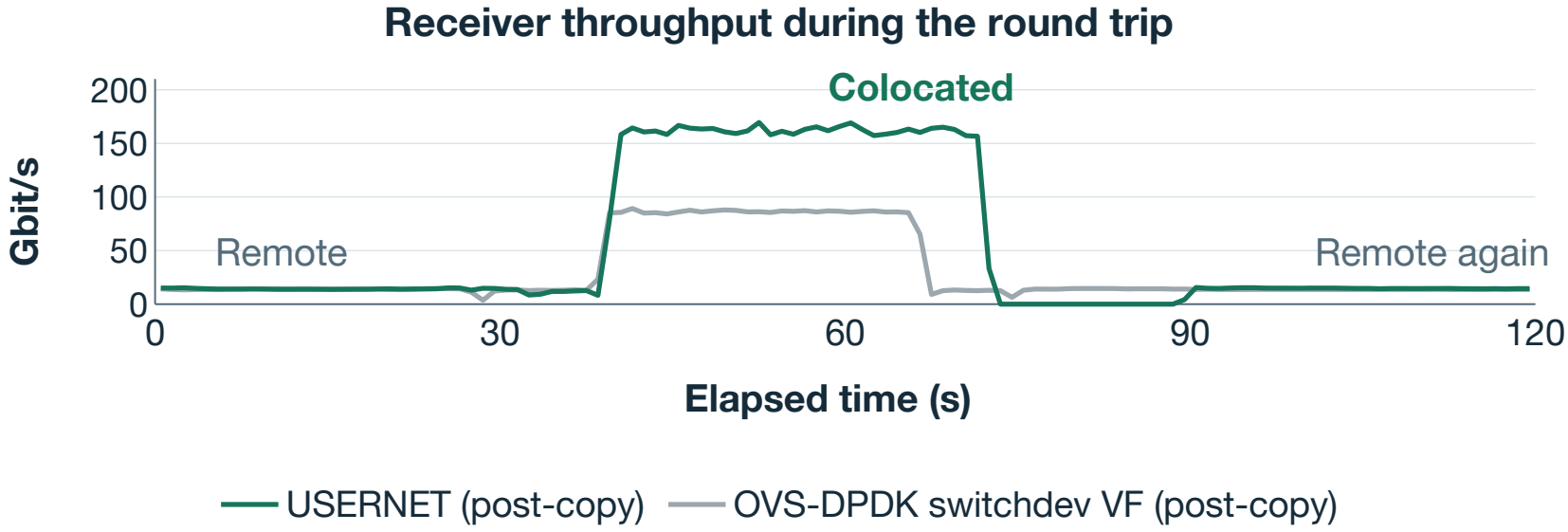
Best: OpenOnload (1); OVS bridge (2); ipvlan (4)

Best: OpenOnload at all worker counts

[1] wrk's default configuration (persistent HTTP connections), with custom nginx settings: worker_processes 4; worker_connections 4096; keepalive_timeout 1d; keepalive_requests 100000000; sendfile off.

Connections Survive VM Post-Copy Migration

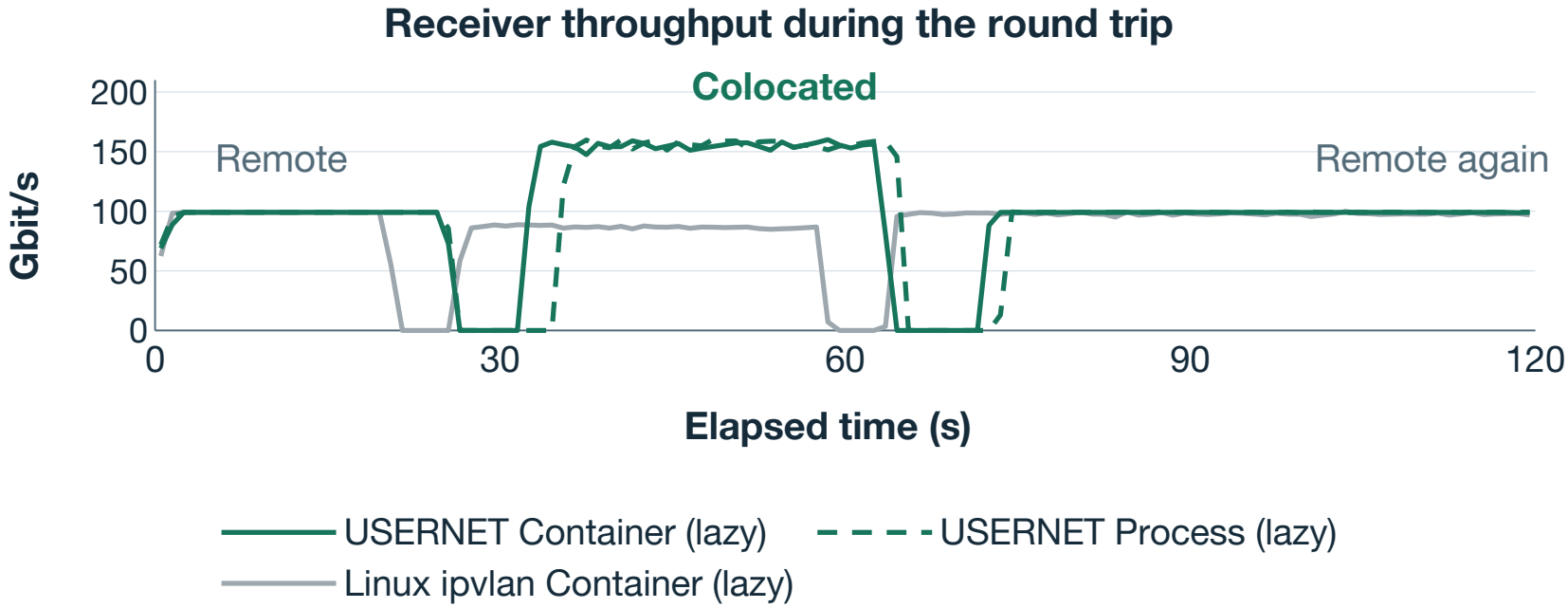
6 established iperf3 connections · NODE2 → NODE1 → NODE2



Both VM curves use OVS-DPDK switchdev VF, the intra-node Netperf TCP_STREAM winner.
Selected with 1 thread and 1 connection, this remote path scaled poorly with inter-node iperf3 -P 6.
The container/process paths on slide 28 did not show this limitation.

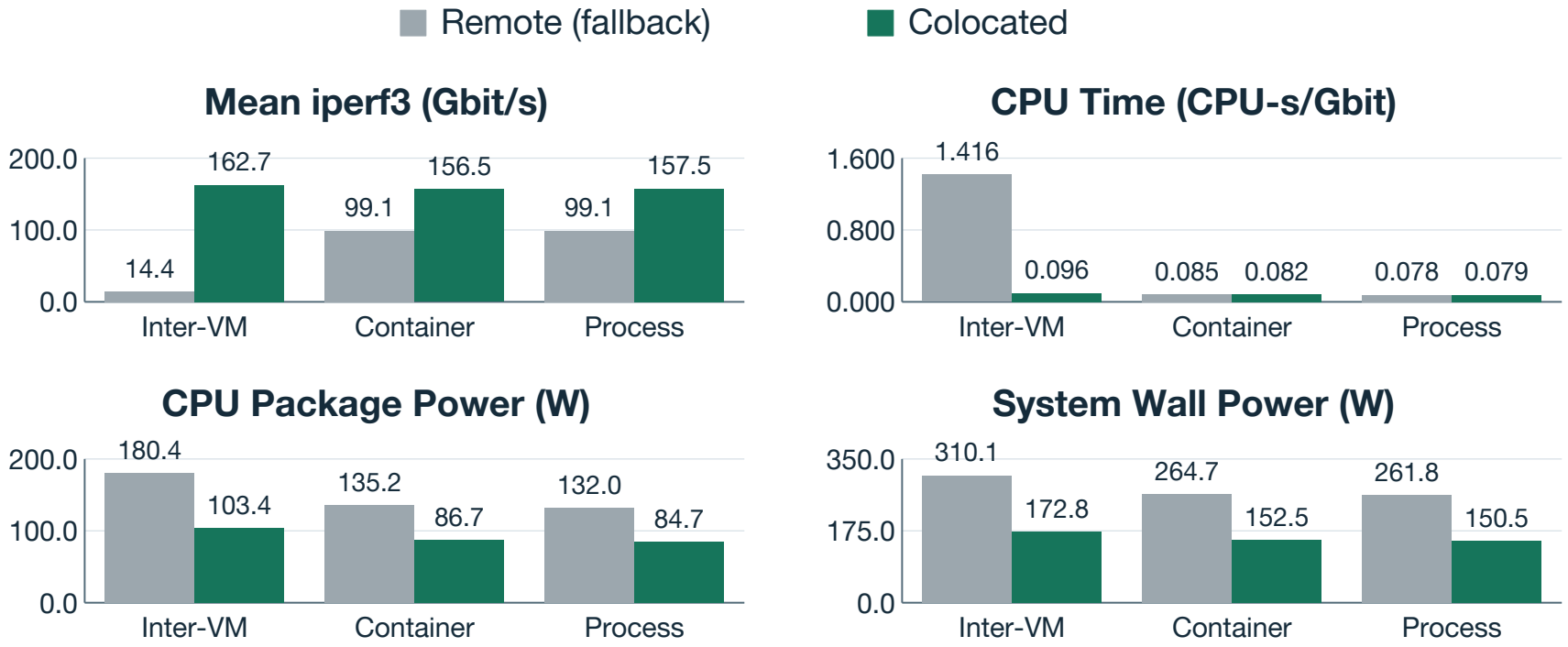
Connections Survive Container/Process Migration

6 established iperf3 connections · NODE2 → NODE1 → NODE2



Resource Usage Comparison

Remote: NODE1 + NODE2 · Colocated: NODE1



Inter-VM: OVS-DPDK switchdev VF · Container: Linux ipvlan · Process: native host TCP

Connection Lifetime Must Amortize Setup

21.5 ms

Inter-VM connect()
One-time initial shared-memory setup

OVS-DPDK switchdev VF: 43 µs

Reuse connections with HTTP keep-alive.

Illustrative RR Break-Even

~740 round trips

21.42 ms extra setup ÷ 28.96 µs/RR
Baseline: OVS-DPDK switchdev VF

Model from separate connect / Netperf tests.
Assumes serial 1-byte requests at measured rates.
Excludes close and migration costs.

Short-lived connections may not justify the setup and migration costs.

Limitations

Connection lifetime & placement

Short connections may not amortize setup / migration costs.
HTTP keep-alive reuses connections to amortize setup.
Placement and benefit prediction are outside scope.

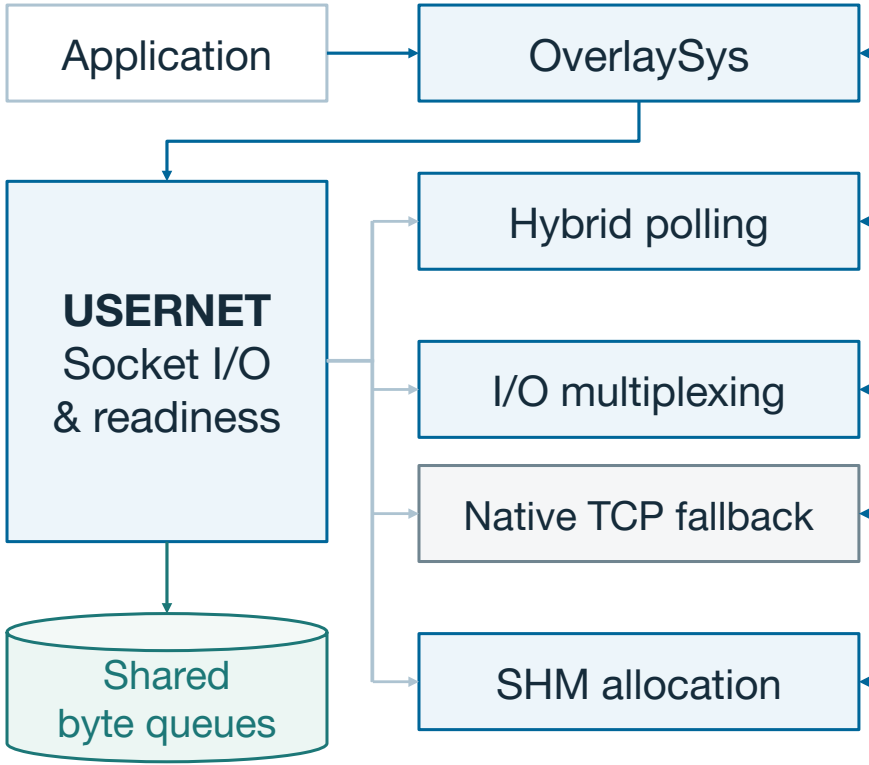
Supported scope

io_uring support is outside scope.
Cross-domain shared-memory communication and nested virtualization remain future work.

Daemon trust & cryptography

Daemon admission and allocation remain trusted.
No additional payload encryption or side-channel isolation mechanisms.

Future Work



Dynamic-Library Interception

Support libraries loaded at runtime (e.g., dlopen).

Adaptive Polling Budgets

Adapt to workload and CPU load to balance latency and CPU use.

Userspace I/O Multiplexing

Explore hybrid processing strategies.

Inter-Node Acceleration

Combine with inter-node acceleration beyond the system TCP fallback.

Shared-Memory Resource Reuse

Reuse or pool allocations to reduce connection setup cost.

Summary

~2× throughput 4–6× request rate

Netperf TCP_STREAM / TCP_RR · VM and container · Best alternative per metric

Direct shared-memory I/O with transport hot swapping.

Transparency	✓	Supported applications run unchanged.
Minimal system changes	✓	Guest module only for inter-VM mapping.
Maintainability	✓	Standard interfaces; reuse native TCP.

Process TCP_RR: 0.76× Junction in the original placement; 1.46× in a separate test with USERNET client/server on distinct cores in the same L3 cache group.

Thank You

Questions?