Thesis Plan:
A Low-Latency Evaluation of Userspace TCP/IP Stacks for high frequency workloads

Josh Brice

November 2025

1 Introduction

High-frequency trading and ultra-low-latency (ULL) financial workloads impose extremely tight constraints on networking systems. These environments require sub-5 μs round-trip times, predictable tail latency, and minimal CPU or cache disturbance. Small fluctuations in delay can have material financial impact, making latency variance (jitter) and tail behaviour just as important as average performance.

This thesis aims to rigorously benchmark and compare three major classes of networking approaches used in low-latency systems:

- Linux **kernel TCP/IP** stack
- **Userspace TCP stacks** built on DPDK
- **Hardware-bypass NIC APIs**, including Mellanox Verbs and Solarflare `ef_vi`

Unlike most existing evaluations that focus primarily on throughput, this work emphasises microsecond-scale behaviour relevant to high frequency: small-message performance, jitter stability, tail latency (p99–p99.99), cache effects, and queue dynamics. The goal is not only to measure performance, but to extract architectural principles that explain the causes of tail spikes and define what constitutes an "ideal" low-latency TCP/IP stack.

2 Phase 1: Measurement and Characterisation

2.1 Systems Under Test

w The first phase evaluates three categories of networking paths:

a) **Linux Kernel TCP** Using standard socket APIs, NAPI interrupt/polling behaviour, `sk_buff` allocation, and in-kernel congestion control.

b) **DPDK-Based Userspace TCP Stacks** A kernel-bypass packet I/O framework using poll-mode drivers, hugepages, and userspace NIC queues. TCP/IP functionality is provided by an existing userspace stack (e.g, mTCP, F-Stack)

c) **Kernel-Bypass NIC APIs (Verbs, `ef_vi`)** Vendor-specific APIs exposing NIC queues directly to userspace. These bypass the kernel entirely, avoid copies, allow predictable DMA behaviour, and require implementing or integrating a custom TCP transport

2.2 Workloads

To reflect high frequency-style behaviour, workloads consist of:

- 64–200 byte request/response messages (order+acknowledgement)
- Low packets-per-second but strict tail sensitivity
- Occasional one-way UDP microbenchmarks for comparison
- NUMA-pinned cores and isolated RX/TX queue paths

2.3 Platforms

Experiments will be run across:

- **AWS** using SR-IOV ENA virtual NICs
- **Bare metal** with Mellanox ConnectX hardware

2.4 Metrics

Key measurements include:

- Latency: p50, p99, p99.9, p99.99
- Jitter distribution
- Queue depth behaviour (NIC queues, software rings)
- CPU profile and instruction-level pipeline stalls
- Cache behaviour, zero-copy feasibility, and memory-copy costs
- Polling stability and batching effects across stacks

Phase 1 ultimately produces a comparative performance dataset and identifies the microarchitectural events responsible for latency outliers.

3 Phase 2: Extraction of Architectural Principles

From the empirical data, the thesis will isolate design principles that minimise jitter and tail variance, including:

- How batching thresholds influence tail spikes
- How copy paths differ between the three evaluatees
- When zero-copy strategies improve performance vs. when they harm predictability
- How NIC-to-L3-cache placement impacts load-to-use latency
- How cloud virtualisation environments distort queueing and timing
- Polling strategies that remain stable under variable workloads
- NUMA placement, CPU pinning, and queue steering requirements

The outcome of Phase 2 is a clear understanding of why certain approaches behave poorly in the tail and which techniques consistently produce predictable behaviour.

4 Phase 3: Proposed Client-Side Low-Latency Architecture

Here, client side refers to a trading firm's order gateway / strategy server that is running inside a colocation facility, sending TCP order flow to the exchange's matching engine.

The final phase synthesises the findings into a prescriptive *client-side* networking architecture for microsecond-scale high frequency order flow over TCP. Rather than redesigning TCP itself, the focus is on how a trading host should compose kernel-bypass I/O, a userspace TCP implementation, and CPU/NIC configuration to minimise tail latency and jitter.

This design includes:

- Memory layout and buffer reuse strategy on the client host
- Queueing model and NIC interaction path (kernel TCP, DPDK, or Verbs/`ef_vi`)
- Polling strategy that avoids pathological jitter amplification

- Zero-copy or one-copy fast paths, constrained by DMA and cache behaviour
- NUMA-aware core placement and RX/TX queue mapping for the order gateway
- Client-side TCP configuration (e.g., window sizing, congestion control) tuned for low-PPS, latency-sensitive flows

Optional deliverable: a minimal prototype of the client-side "fast path"-for example, a userspace TCP send/receive loop built on top of a DPDK or Verbs backend-used to validate that the proposed architecture delivers the expected latency and tail-latency improvements in practice.

5 Conclusion

This thesis seeks to provide an evaluation of userspace TCP/IP stacks across kernel-bypass technologies. TBF