Network Design Considerations for Trading Systems

Andy Myers
Jane Street
New York, NY, USA
amyers@janestreet.com

Brian Nigito
Jane Street
New York, NY, USA
bnigito@janestreet.com

Nate Foster
Cornell and Jane Street
Ithaca, NY, USA
jnfoster@cs.cornell.edu

ABSTRACT

The quest to build scalable data center networks has driven much of the innovation in the networking community in recent decades. But the one-size-fits-all service model offered by these networks does not meet the needs of every application. In response, cloud providers have started to design custom clusters to support the specialized workloads used in finance, AI, high-performance computing, and other areas.

This paper pulls back the curtain on one such area: the low-latency networks used for algorithmic trading systems. We present requirements and architectures, and discuss implementation approaches that make different trade-offs in terms of performance, hardware requirements, and ease of management. We identify concrete steps for making it easier to design low-latency networks in the future.

CCS CONCEPTS

• **Networks** → **Network design principles**; **Bridges and switches**.

KEYWORDS

Network architecture, low-latency, trading systems.

ACM Reference Format:
Andy Myers, Brian Nigito, and Nate Foster. 2024. Network Design Considerations for Trading Systems. In *The 23rd ACM Workshop on Hot Topics in Networks (HOTNETS '24), November 18–19, 2024, Irvine, CA, USA.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3696348.3696890

1 INTRODUCTION

Over the past few decades, the networking community has developed powerful techniques for implementing scalable data center networks. Building on ideas like Clos topologies, commodity switches, and software-defined control planes, modern data center networks are able to deliver increasing amounts of bandwidth to hosts, while keeping cost, power utilization, and operational complexity manageable [23, 25].

But while data center networks are an undeniable success, their one-size-fits-all service model does not meet the needs of every application. For instance, the networks used in the finance industry require low latency and features that go beyond what data center networks provide. Financial exchanges need fair networks that deliver market data to all participants simultaneously. Trading firms need low-latency networks to implement competitive trading strategies, as well as large numbers of multicast groups and precise timestamps. But rather than evolving to provide these features, equipment vendors have been shifting toward larger, more programmable pipelines that sacrifice latency and determinism to obtain bandwidth and flexibility, driven by large equipment purchases by hyperscalers. Hence, there is a growing gap between the requirements of the finance industry and the capabilities of commodity network hardware.

To better understand the unique challenges that arise in financial networks, consider typical algorithmic trading systems. At a high level, they use custom, proprietary strategies to process the market data feeds coming from one or more exchanges and submit orders to those exchanges. There are many ways to implement a trading system, but the most important requirement is to be fast—the likelihood that an order will be profitable rapidly decays as the market data it was based on becomes stale. Some firms build trading systems that operate at the physical limits for communication—e.g., deploying algorithms on specialized hardware directly connected to exchanges. These systems are limited mostly by the speed of light, and can execute trades in 10s to 100s of nanoseconds. Other firms build systems that operate at slower timescales but are nevertheless profitable, being based on a deeper analysis of current market conditions. But even for these slower trading systems, latency matters.

The unique requirements of trading systems illustrates some fundamental challenges for network designers. Most firms use multiple machines to perform trading, with each machine implementing a potentially different strategy. Some strategies only analyze a subset of the feed—e.g., orders involving a certain symbol—while others consume market data

ACM ISBN 979-8-4007-1272-2/24/11
https://doi.org/10.1145/3696348.3696890

from several feeds. Hence, it is helpful to have a network that can split and merge feeds from several exchanges. To minimize processing time, some trading firms filter and normalize the raw feeds coming from each exchange so each machine can execute its strategy directly on the relevant market data. Normalizing the market data also avoids having to perform certain common processing steps redundantly on multiple machines. To determine whether market data from a remote exchange is up to date, strategies often rely on precise timestamps. Timestamps are also used for conducting simulations after the trading day has ended, and for analyzing the performance of new strategies being developed.

This paper pulls back the curtain on architectures and requirements for trading systems. We present three possible network designs: one using commodity switches, another using cloud-hosted infrastructure, and a third using layer-1 switches. Our goal is to draw attention to some of the unique challenges that arise in this domain and help identify directions for future work. We believe these insights will be of broad interest as cloud networks are gradually shifting away from one-size-fits-all service models, which have driven innovation in architectures, hardware, and protocols over the past few decades, to new models that better accommodate the specific requirements of emerging domains.

Our focus in this paper is on the low-latency networks that support trading systems, rather than financial exchanges, for a few reasons. First, exchanges are primarily concerned with fairness rather than latency—i.e., a "slow" exchange that operates at the higher latency guarantees offered by public clouds will generally work fine, provided it delivers market data to and accepts orders from all participants in the same manner. Second, the design space for networks to support trading is larger—trading firms are willing to experiment with non-standard approaches and bleeding-edge hardware if it gives them an advantage. Finally, network support for financial exchanges has already been well-studied in prior work [14, 15].

The rest of this paper is structured as follows. We present an in-depth overview to the design and implementation of trading systems (§ 2) and study trends in workloads and commodity hardware (§ 3). We explore three designs for trading networks, as summarized above (§ 4). Finally, we analyze pain points, suggest opportunities for the research community, discuss related work, and conclude (§ 5).

2 BACKGROUND ON TRADING SYSTEMS

There are a wide range of techniques used to execute trades in the finance industry, but in this paper we will focus on highly-automated, algorithmic trading. The algorithms that execute trades, which we will call "strategies," choose prices to buy and sell based on market data from the exchanges as well as other inputs. It is common to use strategies that analyze combined market data received from many exchanges and send orders to many exchanges. Being able to perform these operations quickly is critical to implementing effective strategies. Opportunities in the market are fleeting, in part because many firms often compete to make the same trades. Repricing orders as quickly as possible is also critical because exchanges will continue matching with an old order's price until it is updated, making trades that are no longer desired.

Competitive latencies vary from strategy to strategy. Simpler strategies tend to be more latency sensitive—at nanosecond timescales, with only a small number of possible operations, there are few ways other than latency to compete. As Deutsche Börse group has observed, some strategies respond to events within 10's to 100's of nanoseconds [10]. Other strategies tend to involve more complex decisions on more inputs, which can move the competitive frontier from nanoseconds to microseconds and beyond.

Trading firms buy and sell various financial instruments (e.g. stocks, ETFs, futures, and options) on hundreds of exchanges worldwide. In the United States, there are currently 16 equities exchanges [8], 18 options exchanges [22], as well as a number of other exchanges for futures and other asset classes. Exchanges receive orders from participants, match up compatible buy and sell orders ("trades"), and disseminate a real-time feed of orders and trades ("market data").

To minimize speed-of-light delays, trading firms co-locate their servers in the same data centers as the exchanges' systems. Trading on all U.S. equities markets requires placing servers in three different co-location facilities ("colos") that are tens of miles apart (see Figure 1(a)). Space in these colos is often over-subscribed (see Figure 1(b)) so it is usually important to minimize a firm's total hardware footprint.

Strategies often analyze market data from different exchanges, many of which are in remote colos. To transport data between colos, trading firms operate private WANs and carefully manage circuits to minimize latency. Some firms employ microwave or laser links to reduce latency further [20]. Microwave links are used even though they are both less reliable (e.g., rain can cause packet loss) and offer less bandwidth than corresponding fiber links. Feeds are not particularly bandwidth intensive, but they can and do burst to full line rate. Bursts across different feeds are often correlated because the underlying market conditions are related—e.g., the announcement of a new government regulation might cause the value of symbols in a sector to shift, in both equities and options markets.

Within a co-location facility (see Figure 1(b)), there are areas ("cages") dedicated to the exchange as well as the various trading firms. Each trading firm has one or more private, dedicated connections ("cross-connects") to the exchange,

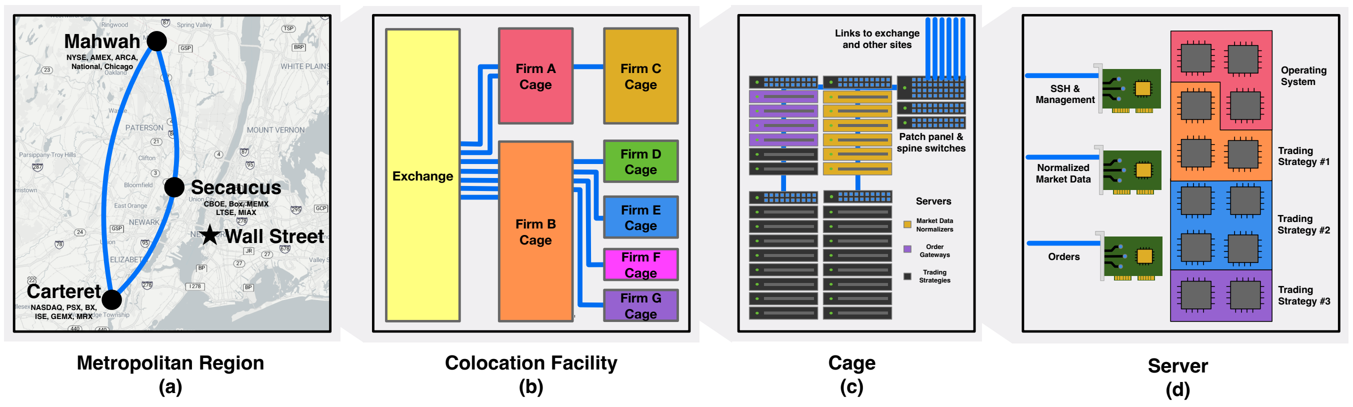


Figure 1: Architecture of a typical trading system. (a) Exchanges are located in "co-location facilities" around the world. Even exchanges in the same region are often miles apart. (b) Co-location facilities allow trading firms to rent one or more "cages" and connect directly to the exchange. (c) Within a cage, a trading firm has racks of servers and switches. Availability of space and power impose practical restrictions. (d) Separate server cores are used for the operating system and for strategies and other functions and separate NICs are used for remote management, market data, and orders.

usually via 10 Gbps Ethernet. Usually the exchange guarantees that all connections will be latency-equalized so that the location of a firm's cage in the data center does not affect the latency of its connection to the exchange. This connection is managed via standard routing protocols (e.g., BGP).

The cross-connects contain both the market data feed for the exchange as well as the orders that the trading firm sends to the exchange. Both use highly-optimized, stateful protocols to minimize delays caused by queueing or serialization. Market data is sent via UDP over IP multicast, often with multiple individual update messages packed into each packet for efficiency—e.g. PITCH [9]. Often exchanges will partition this feed across multiple multicast groups. Each exchange chooses its own binary formats and multicast partitioning scheme. Some exchanges partition based on the name of the instrument (e.g. alphabetical by stock ticker's first letter), while others partition based on the type of instrument (e.g. equities on one group, ETF's on another).

Orders are sent via long-lived (e.g., 6+ hours) TCP connections initiated from the trading firm's servers to endpoints on servers inside the exchange's infrastructure. Again, each exchange can choose its own format and protocol for this (e.g. BOE [7]). The protocols allow for operations such as entering a new order, canceling an existing order, or modifying the price or size of an order. The exchange sends back acknowledgements for state changes, rejects for invalid requests (e.g. sending an order with an invalid ticker), and fills to indicate that an open order has traded with another participant. These protocols often exhibit races—e.g. if a firm's request to cancel an order is sent at the same time as a notification that the order has been filled.

Within a cage (see Figure 1(c)), trading firms divide their trading applications into multiple components, partitioning both by the instruments being traded and by function. A potential decomposition would be to have three types of functions: market data normalizers, strategies, and order entry gateways. The normalizer's purpose is to convert from each exchange's format to an internal standard format, and also to re-partition the data, again according to some standard. To scale to a large number of recipients, normalizers send the data via IP multicast. Strategies subscribes to normalizers and implement the custom algorithms that decide which orders to send. Each strategy has a TCP connection to one or more gateways. The purpose of the gateway is to translate from internal order entry formats back to the protocols that the exchanges use. Because the strategy algorithms can be computationally expensive, there are many more servers dedicated to strategies than to either normalizers or gateways. The servers (see Figure 1(d)) have dedicated cores for each type of system, and separate NICs for management, market data feeds, and orders.

For both monitoring and research, trading firms want to record their network traffic with precise timestamps. Timestamps are used to calculate a strategy's latency by subtracting the time at which the strategy sends an order from the time at which the strategy's most recent input event arrived. For research, precise timestamps are necessary for understanding the ordering of market data events. Some trading firms desire precision below 100 picoseconds [27].

3 WORKLOAD & HARDWARE TRENDS

Due to the competitive nature of the finance industry, specific details about the inner-workings of trading systems, such as

Feed	min	avg	median	max
Exchange A	73	92	89	1514
Exchange B	64	113	76	1067
Exchange C	81	151	101	1442

Table 1: Frame lengths from market data feeds

architectures, hardware, and latency, are generally hard to obtain. Nevertheless, this section presents some broad trends that are relevant to the design of trading systems. We begin by covering some aspects of market data workloads and discussing how these workloads have affected trading system design. We then discuss the evolution of switches with respect to their latency and support for multicast—aspects that have a direct bearing on trading system performance.

Growth of Market Data. Trading systems process large volumes of data—millions of transactions per second from each exchange. Figure 2(a) depicts the growth in market data events for US equities and options markets over the last 5 years. We focus on these markets in particular because they have exhibited dramatic growth in data volumes, but we have observed similar growth elsewhere. Note that the arrival rates are variable, even at the granularity of individual days, but also high in absolute terms. The plot shows tens of billions of events per day, which works out to an average rate of more than 500k events per second. We have observed burst rates over smaller timescales that are at least an order of magnitude larger. Also, the number of events has been growing over time. A trading system must be designed for current workloads and also have capacity to grow.

The bandwidth required for market data is surprisingly small. As discussed earlier, market data is encoded using efficient binary formats. Table 1 shows typical packet sizes for a sample of frame lengths from the middle of the trading day. Though each feed is slightly different, all encode their data efficiently. Note that lengths in the table are inclusive of Ethernet, IP, and UDP headers. Across all feeds, 40 bytes of network headers (plus another 8-16 bytes of protocol-specific headers) represent 25%-40% of the data sent.

Figure 2(b) zooms in to show options market data events for a single stock on a single day. The market data events have been filtered to just those that affect the best bid and offer prices or sizes on any of the 18 options exchanges. Each point represents a count of events in a 1 second long window. Options on this stock trade from 9:30am to 4:00pm, with little to no activity outside of this range. The median second has over 300k events, and the busiest second contains 1.5M events. During that busy second, to be able to process a single second's events as quickly as they arrive, a trading system would need to be able to process each event in around 650 nanoseconds—a feasible performance target, though a challenge for even well-tuned software implementations.

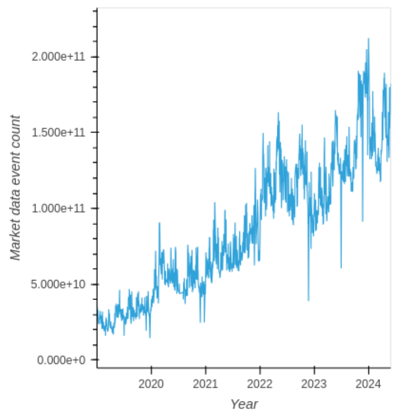
Figure 2(c) again shows market data events for just the busiest second from the previous figure, aggregating events into 100 microsecond windows. The median 100 microsecond interval contains 129 events, and the busiest interval contains 1066 events. For a trading system to keep up with this peak workload requires processing at 100 nanoseconds per event—i.e., a software system would have little time to perform any operations beyond copying data into memory.

Implications for trading systems. To handle the bursty workloads associated with market data, trading system are partitioned so that no single component needs to process all events. A variety of partitioning schemes are possible—e.g. dividing the workload by asset class, exchange, stock, or combinations thereof. A key design choice is where to filter out the market data that will not be used by a partition. One option is to filter in the same process as the trading system itself. Intuitively, if the combined time spent discarding data and the time spent processing data is larger than the arrival rate, then filtering should happen outside the trading system—either on another core on the same server or on a middlebox. When several systems employ the same partitioning scheme, middleboxes can be more efficient in terms of the number of cores used. IP multicast is often used as the transport for the partitioned data as it scales well with the number of receivers.

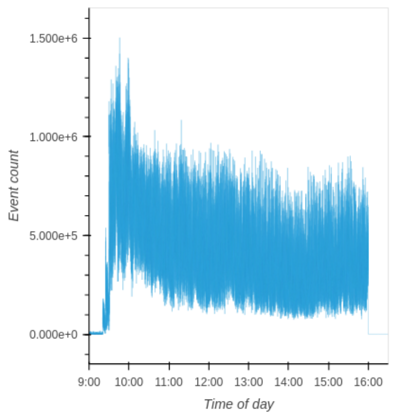
Partitioning allows a strategy to scale with changes in market data load. The number of partitions can be scaled up as the volume of market data increases—e.g. because of growth in events, the opening of a new exchange [21], or as new functionality is incorporated into a strategy. The number of partitions may also decrease when the strategy is optimized or simplified. However, the recent trend has been toward larger numbers of partitions—e.g., for one representative strategy, the number of partitions roughly doubled from around 600 to over 1300 over the past two years.

Next we will discuss how recent trends in switch hardware are at odds with the desire to scale via partitioning.

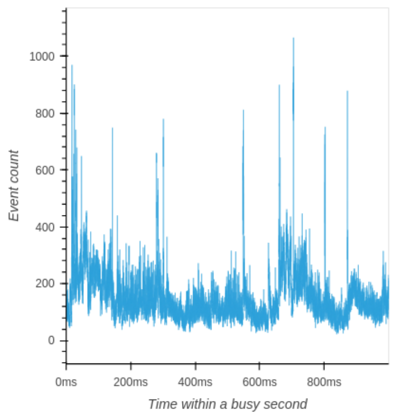
Latency Trends. Scaling to process high volumes of data while achieving low-latency imposes tight constraints on network hardware. However, while commodity switches have roughly doubled the amount of bandwidth with each successive generation, they have not made significant improvements in latency. In fact, latency has been slowly increasing, which is not surprising: as vendors have created more sophisticated and flexible hardware pipelines to provide the features used in data centers, minimum latency has gone up, even for simple pipelines in cut-through mode. While some vendors do offer specialized low-latency switches, they often have reduced table sizes or lack other features, which



(a) U.S. options and equities event count by day



(b) Options events for a single stock on a single day



(c) Options events in busiest second of the day (single stock)

makes them unsuitable for trading systems. Compared to the models of switches that were deployed a decade ago, switches today have around 20% higher latency, at roughly 500 nanoseconds. Meanwhile, latency for a hop through a software host has been decreasing over time and is now below 1 microsecond [11]—i.e., for an empty "ping pong" application. Hence, network latency is a large and increasing share of total system latency.

Multicast Trends. Support for multicast has also not improved in recent network devices. This fact is not surprising—implementing multicast in hardware is fundamentally hard. Switches must maintain a hop-by-hop flow table (the "mroute table") using dedicated memory on the switch ASIC to avoid creating loops. Internal, ASIC-specific tables are often not well-documented, and can sometimes overflow, even when using the configuration APIs endorsed by vendors. When a table overflows, switches generally fall back to software forwarding, which cripples performance and induces heavy packet loss. Experience with several generations of switches from multiple vendors has shown that while bandwidth is steadily increasing, the number of multicast groups is flat or increasing marginally. As Figure 2(a) shows, market data has increased 500% over the last 5 years. However, the latest generation of switches supports only 80% more mulitcast groups than earlier generations. Finally, alternatives such as end-system multicast [16] are not applicable to trading systems as the cost of receiving and re-forwarding data would be prohibitively high.

4 DESIGNS FOR TRADING NETWORKS

This section explores the design space for low-latency trading networks. To keep the discussion grounded, we focus on three concrete designs that can be realized on current hardware. The first uses the traditional building blocks for data centers: spine-leaf topologies and commodity switches. This design is flexible and can support a large number of hosts, but has shortcomings in terms of latency. The second uses public cloud infrastructure, enhanced to equalize latency across all tenants. The cloud has obvious advantages, but current proposals lack the ability to aggregate information across multiple servers and remote exchanges. The third uses Layer-1 switches, which trade off the ability to implement certain forms of packet processing for better latency.

In terms of scale, our aim will be to support a network of roughly 1,000 servers running normalizers, gateways and strategies. Each function is implemented using a subset of the servers: a few dozen each for normalizers and gateways and the rest for strategies. We will assume that the average latency of each function is less than 2 microseconds.[1]

4.1 Design 1: Traditional Switches

Consider a standard leaf-and-spine topology, where each rack of servers has a top-of-rack (ToR) switch and there is another layer of switches to connect the ToRs. We will dedicate one ToR to connect to the exchanges, so every host on the network is equidistant from the exchange. This ToR also provides a convenient place to enforce policy such as firewall rules. To calculate routes, we will use a standard Layer-3 protocol. As is well known, this design can be scaled out to essentially arbitrary sizes, while providing ample bandwidth to each host [2, 25]. The major issue is that the latency incurred by the network is high. A reasonable design would group servers with common functions by rack, in which case, a round trip (exchange, normalizer, strategy, gateway, and back to the exchange) would involve 12 switch hops and 3 software hops. Assuming each switch hop incurs 500 nanoseconds of latency, half of the overall time through the

[1]Of course, tail latency matters too, but we'll focus on average latency.

system is spent in the network! We could try to reduce switch hops by placing servers in more optimal ways, but in our system, the distribution of normalizers, trading strategies, and order gateways is not uniform, so we could only optimize placement for a few strategies and the majority would not benefit. And, as we have remarked, the number of multicast groups that the switches support would be lower than we would want for partitioning strategies.

4.2 Design 2: The Cloud

Another approach is to rely on cloud networks that have been carefully engineered to equalize latency, thereby ensuring fairness for all tenants. As discussed in recent papers [14, 15], these networks assume a model in which (i) the cloud provider manages the network, (ii) the connections to/from the exchange support multicast and can be effectively equalized with respect to latency, (iii) virtual latency equalization (for machines owned by the cloud provider) and physical latency equalization (for co-located machines owned by a trading firm) are approximately equivalent.

However, while cloud-based designs are a reasonable approach for smaller, isolated trading systems, it is still unclear how to use them at scale. Large-scale trading systems require support for broad internal communication, both to disseminate market data to strategies, and to satisfy regulations associated with equities and options. For instance, the US Security and Exchange Commission (SEC) imposes rules that prohibit advertising prices that "lock" (i.e., where a bid on one exchange equals the asking price on another exchange) or "cross" (i.e., where a bid on one exchange is higher than the asking price on another exchange), as well as "trading through" (i.e., trading at prices worse than those advertised at other markets). Firms also track metrics akin to a firm-wide net position, for regulatory reasons and to assess risk.

So while the cloud offers impressive flexibility and scalability, current proposals for relocating exchanges and trading systems to clouds have significant challenges at scale. On the one hand, if both exchanges and the machines used for trading are hosted in the cloud and connected via fair networks that offer latency equalization, then latency for communication beyond the cloud will be excessive. On the other hand, if the cloud only hosts the exchange and trading firms colocate their own infrastructure, then each firm will still need to build a separate network to support trading—i.e., using one of the designs considered in this section. Overall, the need for broad internal communication, regulatory requirements, and performance considerations make traditional networking solutions more suitable for trading networks. Of course, as technology evolves, hybrid solutions that better address these challenges may emerge.

4.3 Design 3: Layer-1 Switches

If designs based on commodity switches are too slow, it's natural to consider Layer-1 switches (L1Ses), which operate at the opposite end of the functionality vs. latency spectrum. Devices such as the Arista 7130 [3] are essentially circuit switches and cannot implement packet classification and filtering, nor can they split traffic across multiple paths.[2] Despite these limitations, L1Ses have several attractive features relevant to trading systems: (i) they can forward traffic from any input port to any set of output ports with only 5-6 nanoseconds of latency; (ii) they have built-in accurate timestamping; and (iii) they can merge traffic from multiple input interfaces onto a single output interface, at the expense of an additional 50 nanoseconds of latency.

To use L1Ses in a trading system, one would essentially construct four different networks between each of: exchanges and normalizers, normalizers and strategies, strategies and gateways, and gateways and exchanges. Such a network can be configured to deliver traffic in nanoseconds to essentially arbitrary sets of hosts, at two orders of magnitude lower latency than commodity switches. The main issue that arises with L1S-based networks is interface proliferation: it is feasible to deliver each normalizer's output to every rack or even every strategy server. If a strategy wants to process the digested feeds coming from the normalizers, then it either needs a separate NIC for each feed, or the feeds must be merged into a feed that can be consumed by a single NIC.

Either approach has limitations: adding extra NICs to servers does not scale, while merging feeds creates bottlenecks. Recall that market data is bursty, so merged feeds can easily exceed the available bandwidth, leading to latency from queuing or packet loss. An analogous problem arises with commodity switches—e.g., suppose a trading strategy subscribes to too many multicast groups, resulting in congestion. The difference is that with L1Ses, the feeds are coarser-grained, and cannot be as easily reconfigured. A practical workaround for NIC proliferation is to restrict the total number of normalizers each trading strategy can subscribe to. But the effect of limiting subscriptions would be that the normalizers cannot be partitioned as widely, leading to increased latency and reduced performance.

5 OPEN PROBLEMS AND RELATED WORK

As we have seen, none of the current approaches to building networks fully satisfy the requirements of trading firms. Prior work has investigated some aspects of this problem [1, 5, 6, 13, 19], but we argue that solving it requires a more comprehensive effort, rethinking the design of the entire network

[2]Some commercial L1Ses include an FPGA that can implement additional features such as packet classification and filtering, at the cost of increased latency.

from the bottom up. Taking inspiration from the push toward deep programmability [12] motivated by the requirements of the cloud—e.g., SDN controllers, programmable switches and NICs, custom transport protocols, etc.—we present some possible directions for future work on low-latency networks.

Hardware. The attractive latency characteristics of L1Ses are a promising foundation for trading systems, even though their features are limited compared to commodity switches. Are there capabilities we could add to L1Ses to increase their functionality without slowing them down too much? Any feature that results in queueing is unlikely to be helpful, but it is possible to add support for filtering and splitting feeds, and load balancing across multiple forwarding paths. Already several commercial L1Ses take advantage of accelerators based on reconfigurable hardware [26]. These devices appear to offer the best of both worlds—100-nanosecond latency and standard IP forwarding and multicast—although they tend to have small forwarding tables. Such switches could also be configured to combine data arriving on multiple interfaces, thereby reducing the proliferation of interfaces. Applied naively, merging would lead to queueing or packet loss. But when combined with other ideas, such as header compression or data filtering, it should be possible to safely merge feeds while avoiding these issues.

Protocols. Most trading systems rely on standard transport protocols, as they have ubiquitous support in hardware and software, and are well-understood by network operators. However, at 10Gbps, processing the Ethernet, IP, and TCP headers costs 40 nanoseconds, even though strategies routinely ignore most if not all of the data in these headers. Similarly, orders are usually just a few bytes long—26 bytes for a new order and 14 bytes for an order cancellation on PITCH [9]—so the overhead of standard protocol headers is excessive. It seems fruitful to consider designing custom transport protocols for use in trading systems. One could also imagine designing custom transport protocols with the constraints of L1Ses in mind—e.g., exposing information that can be used for filtering or load balancing.

Routing. Routing is one of the oldest areas in networking, but the heavy use of multicast based on application-level data in trading networks brings new dimensions to consider. How can we design routing schemes that deliver relevant market data to strategies? By co-designing the algorithm used to transform raw market data to normalized feeds as well as the mapping from feeds to multicast groups, can we achieve a more efficient design? In situations where the optimal solutions are not feasible how can we approximate? How can we incorporate the constraints induced by the placement of taps and capture appliances? There are potential synergies with some work in pub-sub systems [18], and in content-centric and named-data networking [17], although the latency constraints in trading networks suggest that the underlying mechanisms would look different. Wide-area routing based on speed-of-light networking is also relevant [4].

Cluster Management. Cloud computing has shown that it is beneficial to automate provisioning, placement, and scaling of network services [24]. Can we develop the same kinds of systems for trading networks? Such a system would need to optimize latency above other criteria, but would also need to take into account bandwidth as well as constraints induced by the application level (e.g., connecting a strategy to a particular normalized feed). A related problem is how to migrate a given job from one server to another. The jobs in trading networks run on bare metal servers, so there are likely to be subtle differences compared to to prior work on virtual machines and containers.

Outlook. It is not surprising that the technologies developed for cloud computing do not meet the needs of trading systems. We argue it is time to explore a new research agenda focused on the constraints in low-latency networks, cutting across layers to redesign hardware, protocols, routing schemes, cluster management, and more.

ACKNOWLEDGMENTS

The authors wish to thank the HotNets reviewers, Jasper Chapman-Black, Al Esposito, Oliver Gugenheim, Andrew Redmon, and Yaron Minsky for helpful feedback. We are also grateful to Dave Maltz and Jennifer Rexford for early discussions on designing networks for trading systems.

REFERENCES

[1] Andrew Addison, Charles Andrews, Newas Azad, Daniel Bardsley, John Bauman, Jeffrey Diaz, Tatiana Didik, Komoliddin Fazliddin, Maria Gromoa, Ari Krish, Ryan Prins, Larry Ryan, and Nicole Villette. 2019. Low-Latency Trading in the Cloud Environment. In *IEEE International Conference on Computational Science and Engineering (CSE) and IEEE International Conference on Embedded and Ubiquitous Computing (EUC)*. 272–282. https://doi.org/10.1109/CSE/EUC.2019.00060

[2] Mohammad Al-Fares, Alexander Loukissas, and Amin Vahdat. 2008. A Scalable, Commodity Data Center Network Architecture. In *ACM SIGCOMM Conference on Data Communication (SIGCOMM)*. 63–74. https://doi.org/10.1145/1402958.1402967

[3] Arista Networks. 2021. The Arista 7130 Series. https://www.arista.com/assets/data/pdf/7130-product-overview.pdf Retrieved June 24, 2024.

[4] Debopam Bhattacherjee, Waqar Aqeel, Sangeetha Abdu Jyothi, Ilker Nadi Bozkurt, William Sentosa, Muhammad Tirmazi, Anthony Aguirre, Balakrishnan Chandrasekaran, P. Brighten Godfrey, Gregory Laughlin, Bruce Maggs, and Ankit Singla. 2022. cISP: A Speed-of-Light Internet Service Provider. In *USENIX Symposium on Networked Systems Design and Implementation (NSDI)*. 1115–1133. https://www.usenix.org/conference/nsdi22/presentation/bhattacherjee

[5] Debopam Bhattacherjee, Waqar Aqeel, Gregory Laughlin, Bruce M. Maggs, and Ankit Singla. 2020. A Bird's Eye View of the World's

Fastest Networks. In *ACM Internet Measurement Conference (IMC)*. 521–527. https://doi.org/10.1145/3419394.3423620

[6] Andrew Brook. 2015. Low-latency Distributed Applications in Finance. *Communications of the ACM (CACM)* 58, 7 (June 2015), 42–50. https://doi.org/10.1145/2747303

[7] Cboe Global Markets. 2024. US Equities Exchange Binary Order Specification. https://cdn.cboe.com/resources/membership/Cboe_US_Equities_BOE_Specification.pdf

[8] Cboe Global Markets. 2024. US Equities Market Share. https://www.cboe.com/us/equities/market_share/

[9] Cboe Global Markets. 2024. US Equities/Options Multicast Depth of Book (PITCH) Specification. https://cdn.cboe.com/resources/membership/US_EQUITIES_OPTIONS_MULTICAST_PITCH_SPECIFICATION.pdf

[10] Deutsche Börse Group. 2023. Insights into Trading System Dynamics. https://www.eurex.com/resource/blob/48918/e8d4df56f75c9a96fb0f6fff6b18a14f/data/presentation_insights-into-trading-system-dynamics_en.pdf Retrieved June 24, 2024.

[11] Advanced Micro Devices. 2023. Onload User Guide. https://docs.amd.com/r/en-US/ug1586-onload-user Version UG1586.

[12] Nate Foster, Nick McKeown, Jennifer Rexford, Guru Parulkar, Larry Peterson, and Oguz Sunay. 2020. Using Deep Programmability to Put Network Owners in Control. *ACM SIGCOMM Compututer Communications Review (CCR)* 50, 4 (Oct. 2020), 82–88. https://doi.org/10.1145/3431832.3431842

[13] Yilong Geng, Shiyu Liu, Zi Yin, Ashish Naik, Balaji Prabhakar, Mendel Rosunblum, and Amin Vahdat. 2018. Exploiting a Natural Network Effect for Scalable, Fine-grained Clock Synchronization. In *SENIX Conference on Networked Systems Design and Implementation (NSDI)*. 81–94. https://www.usenix.org/conference/nsdi18/presentation/geng

[14] Eashan Gupta, Prateesh Goyal, Ilias Marinos, Chenxingyu Zhao, Radhika Mittal, and Ranveer Chandra. 2023. DBO: Fairness for Cloud-Hosted Financial Exchanges. In *ACM SIGCOMM Conference on Data Communication (SIGCOMM)*. 550–563. https://doi.org/10.1145/3603269.3604871

[15] Muhammad Haseeb, Jinkun Geng, Ulysses Butler, Xiyu Hao, Daniel Duclos-Cavalcanti, Anirudh Sivaraman, and Srinivas Narayana. 2024. Design and Implementation of a Scalable Financial Exchange in the Public Cloud. https://arxiv.org/abs/2402.09527

[16] Yang hua Chu, S.G. Rao, S. Seshan, and Hui Zhang. 2002. A Case for End-system Multicast. *IEEE Journal on Selected Areas in Communications (JSAC)* 20, 8 (2002), 1456–1471. https://doi.org/10.1109/JSAC.2002.803066

[17] Van Jacobson, Diana K. Smetters, James D. Thornton, Michael F. Plass, Nicholas H. Briggs, and Rebecca L. Braynard. 2009. Networking Named Content. In *International Conference on Emerging Networking Experiments and Technologies (CoNEXT)*. Association for Computing Machinery, New York, NY, USA, 1–12. https://doi.org/10.1145/1658939.1658941

[18] Theo Jepsen, Ali Fattaholmanan, Masoud Moshref, Nate Foster, Antonio Carzaniga, and Robert Soulé. 2022. Forwarding and Routing With Packet Subscriptions. *IEEE/ACM Transactions on Networking (ToN)* 30, 6 (May 2022), 2464–2479. https://doi.org/10.1109/TNET.2022.3172066

[19] John W. Lockwood, Adwait Gupte, Nishit Mehta, Michaela Blott, Tom English, and Kees Vissers. 2012. A Low-Latency Library in FPGA Hardware for High-Frequency Trading (HFT). In *IEEE Annual Symposium on High-Performance Interconnects (HOTI)*. 9–16. https://doi.org/10.1109/HOTI.2012.15

[20] McKay Brothers. 2024. Low Latency Microwave. https://www.mckay-brothers.com/

[21] Miami International Holdings. 2024. Miami International Holdings Announces Successful Launch of MIAX Sapphire Options Exchange. https://www.miaxglobal.com/news/miami-international-holdings-announces-successful-launch-miax-sapphire

[22] Options Clearing Corporation. 2023. *Participant Exchanges & Futures Markets*. https://www.theocc.com/clearance-and-settlement/participant-exchanges

[23] Leon Poutievski, Omid Mashayekhi, Joon Ong, Arjun Singh, Mukarram Tariq, Rui Wang, Jianan Zhang, Virginia Beauregard, Patrick Conner, Steve Gribble, Rishi Kapoor, Stephen Kratzer, Nanfang Li, Hong Liu, Karthik Nagaraj, Jason Ornstein, Samir Sawhney, Ryohei Urata, Lorenzo Vicisano, Kevin Yasumura, Shidong Zhang, Junlan Zhou, and Amin Vahdat. 2022. Jupiter Evolving: Transforming Google's Datacenter Network via Optical Circuit Switches and Software-Defined Networking. In *ACM SIGCOMM Conference on Data Communication (SIGCOMM)*. Association for Computing Machinery, New York, NY, USA, 66–85. https://doi.org/10.1145/3544216.3544265

[24] Kexin Rong, Mihai Budiu, Athinagoras Skiadopoulos, Lalith Suresh, and Amy Tai. 2023. Scaling a Declarative Cluster Manager Architecture with Query Optimization Techniques. *Proceedings of the VLDB Endowment (VLDB)* 16, 10 (June 2023), 2618–2631. https://doi.org/10.14778/3603581.3603599

[25] Arjun Singh, Joon Ong, Amit Agarwal, Glen Anderson, Ashby Armistead, Roy Bannon, Seb Boving, Gaurav Desai, Bob Felderman, Paulie Germano, Anand Kanagala, Jeff Provost, Jason Simmons, Eiichi Tanda, Jim Wanderer, Urs Hölzle, Stephen Stuart, and Amin Vahdat. 2015. Jupiter Rising: A Decade of Clos Topologies and Centralized Control in Google's Datacenter Network. In *ACM SIGCOMM Conference on Data Communication (SIGCOMM)*. 183–197. https://doi.org/10.1145/2785956.2787508

[26] Greg Stitt, Wesley Piard, and Christopher Crary. 2024. Low-Latency, Line-Rate Variable-Length Field Parsing for 100+ Gb/s Ethernet. In *ACM/SIGDA International Symposium on Field Programmable Gate Arrays (FPGA)*. 12–21. https://doi.org/10.1145/3626202.3637559

[27] Zuotian Tatum. 2020. From Instant to Eternity - Challenges in Data Engineering. https://www.youtube.com/watch?v=pe6Gh_fi3Wc YouTube video.