Rethinking Cloud-hosted Financial Exchanges for Response Time Fairness

Prateesh Goyal[1], Ilias Marinos[1], Eashan Gupta[1,2], Chaitanya Bandi[1]
Alan Ross[3], Ranveer Chandra[1]
[1]Microsoft Research, [2]UIUC, [3]Microsoft

Abstract

In this paper, we consider the problem of supporting modern financial exchange services on the cloud premises. Important exchange services rely on predictable, equal latency from the servers to the participants for fair competition. Existing cloud networks, however, are unable to offer such property, as they were not originally designed for this purpose. We attempt to tackle the problem of unfairness that stems from the lack of determinism in cloud networks. We argue that predictable or bounded latency is not necessary to achieve fairness. Inspired by the use of logical clocks in distributed systems, we propose a new approach that instead corrects for differences in latency to the participants for fairness. We evaluate our approach in simulation and show that it is feasible to achieve fairness under highly variable network latency. Our approach is deployable in contemporary cloud environments; it avoids limitations of state-of-the-art and outperforms it.

CCS Concepts

• **Networks** → **Network protocol design**; **Network architectures**.

Keywords

Financial exchange, logical clock, cloud, fairness

ACM Reference Format:
Prateesh Goyal[1], Ilias Marinos[1], Eashan Gupta[1,2], Chaitanya Bandi[1], Alan Ross[3], Ranveer Chandra[1], [1]Microsoft Research, [2]UIUC, [3]Microsoft. 2022. Rethinking Cloud-hosted Financial Exchanges for Response Time Fairness. In *The 21st ACM Workshop on Hot Topics in Networks (HotNets '22), November 14–15, 2022, Austin, TX, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3563766.3564098

HotNets '22, November 14–15, 2022, Austin, TX, USA

ACM ISBN 978-1-4503-9899-2/22/11…$15.00
https://doi.org/10.1145/3563766.3564098

1 Introduction

Cloud providers are continuously improving their datacenters to provide better computing, networking and storage resources to end customers. These innovations have helped many industries to forego the cumbersome task of building and maintaining their own on-premise data centers and move to the cloud. Despite all the investment and innovation from cloud providers, cloud environments today are still not well suited for many industries.

Major financial exchanges such as NASDAQ, CME, and NYSE run their Central Exchange Server (CES) in on-premise data centers. At a high level, the CES generates market data and distributes it to various market participants (MPs) in real time. Certain MPs (commonly known as high-frequency traders), rapidly *react* to new market data issuing a high volume of transactions: their profit is highly dependent on winning the 'race for speed', aiming to submit their trade orders before other competitors. High frequency traders are very aggressive in reducing latency: they often use smartNICs to shave off just μs of latency on their end to gain competitive edge. To accommodate fair competition based on speed of trading, modern financial exchanges offer simultaneous delivery of market data to the interested MPs, as well as ordered processing of the trade transactions based on their submission time (measured at the MP). Such fairness is only provided to a fraction of the MPs, and comes at a premium cost: financial exchanges offer colocation services for MPs' servers at the same datacenter as the exchange's CES, where they are able to provably guarantee equal bidirectional latency from the CES to all colocated MPs. Exchanges go to a great extent to ensure fairness for their colocated MP customers; it is not uncommon, for example, to use layer-1 fan-out switches for market data stream replication and equal-length cables to all colocated MPs. For the rest of the MPs – who either do not profit from such trading strategies or cannot afford the colocation services – fairness of such kind is not available. Such MPs typically receive the market data stream and submit orders over variable-latency private or shared WAN links, or through intermediate brokers.

Moving the CES to the cloud presents a huge business opportunity for major cloud providers such as Amazon, Google, and Microsoft [11, 12]. Financial exchanges also have a strong incentive to move their business to the cloud: they could rapidly increase market access to more participants, and also benefit from modern cloud's elastic resource scaling. To achieve

smooth migration to the cloud, however, all of modern exchanges' services need to be accommodated, including fairness on speed trading which presents unique challenges. In particular, ensuring fairness by providing deterministic equal latency to the MPs similar to the on-premise datacenters, would be quite challenging in the cloud. Cloud datacenters have originally been designed for a heterogeneous, multi-tenant environment, aiming to accommodate diverse workloads. Even if the MPs are located within the same cloud region as the CES, it is hard to guarantee that the latency between CES and various MPs will be the same. Copper and fiber optics cables are not necessarily of equal length, network traffic is not evenly balanced among the different paths, multiple vendors' network elements have different performance characteristics, network oversubscription is still common in datacenters, and network quality of service mechanisms for concurrent workloads are only best effort.

Over the recent years, this problem has received some attention from both the financial and computer science research communities. CloudEx [3] proposes using high-precision clock synchronization to ensure that the market data is released to MPs simultaneously. In the event of latency spikes beyond a certain threshold, however, CloudEx incurs unfairness. Unfortunately, production datacenters do not guarantee bounded latency. Latency spikes – up to a few orders of magnitude increase than average – are quite frequent [10]. Libra [9] orders incoming trade requests based on their contents while Budish et al. [1] propose aggregating real-time market data and delivering it in batches to the MPs along with aggregating incoming trades corresponding to a batch. These approaches, however, impose significant restrictions as they require changes to how trades are processed at the CES [3].

In this paper, we set out to tackle the problem of providing fairness for financial exchange systems that operate in modern cloud datacenters. We observe that provably equal latency between CES and MPs is a useful property that helps ensuring equal opportunity for market participants, but it is both hard to achieve and not strictly essential for the purpose. Instead, for our solution we adopt a radically different approach: we choose to relax any assumptions for tight clock synchronization among cloud nodes or predictable bounded latency in datacenter networks. Further, our approach is general since it does not require any changes to the core central exchange algorithms. Our key premise is that, simultaneous delivery of market data is only essential for *reactive* trades that are generated directly and quickly in response to specific real-time market data points [9]. For all other trades, minor differences in delivery times are not critical. For such reactive trades, we do not need to ensure simultaneous delivery of market data. We can alternatively achieve fairness by enforcing *ordering* on incoming trade requests based on the duration it took for each participant to react, i.e, time taken to submit a trade since the reception of some particular market data (response time).

The challenge in ordering trades this way is that it is hard to measure response times since the cloud-provider does not know how the MP generated a specific trade (i.e., which data triggered the trade). Inspired by the use of logical clocks for ordering events in distributed systems [6, 7], we introduce the notion of "Delivery Clocks" to overcome this issue. Corresponding to each MP, we maintain a delivery clock that tracks the progress of market data delivery to the MP. We show that *ordering trades from the MP based on its delivery clock (DBO)* coupled with *controlling how these delivery clocks advance* can help account for delay variations in delivery of market data to MPs and enable the CES to achieve response time fairness.

To this end, in §3, we first show how DBO on its own can help improve fairness. Next, we establish some fundamental requirements on pacing of market data delivery for achieving perfect fairness. We show that if these requirements are met, DBO can indeed provide perfect fairness. In §5, we use DBO and the constraints on pacing as our guiding principles to propose a few simple schemes and evaluate their performance.

Contrary to the current *modus operandi* of financial exchanges, where fairness has limited scalability and comes at a premium, our solution does not rely on equal latency between the CES and MPs and hence scales to arbitrarily sized datacenters or even across datacenters and regions. This property could be the building block for democratizing fairness in financial exchanges, with all MPs getting equal opportunities when trading.

2 Background

2.1 High-level Architecture

Fig. 1 shows the main components of our system. This architecture is roughly in line with earlier works [3, 9].

Outgoing market data: The CES generates a stream of real-time market data. There are multiple MPs, all located within the cloud. For each MP there is an associated Release Buffer (RB) that is controlled by the cloud-provider. Each RB receives market data directly from the CES; then it decides when the market data should be released to the corresponding MP.

Incoming trade orders: Each MP generates trades based on the market data stream and *submits it* to the corresponding RB. The RB tags the incoming trade with any additional information necessary for fair ordering and forwards it to the CES. At the CES, the ordering buffer (OB) orders incoming trades based on the tagged information (§4.1) and forwards it to the matching engine (ME).The ME, matches buy orders against selling orders, and executes matched trade orders.

Notation: We refer to the x^{th} market data point as x. (i,k) refers to the k^{th} trade from MP_i. Table 1 lists the notations used in this paper.

Assumptions:

Proximity of RB to MP: Each RB is located sufficiently close to its MP (e.g., colocated in the same VM as the MP, §4.2) and the latency between a RB and MP pair does not impact fairness.

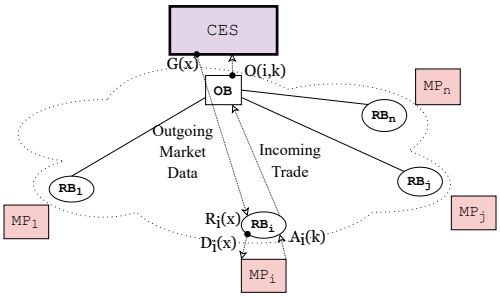


Figure 1: Basic components. RBs and OB are controlled by the cloud provider.

Notation	Definition
$G(x)$	Real Time at which x was generated at the CES.
$R_i(x)$	Real Time at which x was received at RB_i.
$D_i(x)$	Real Time at which x was delivered by RB_i to MP_i.
$A_i(k)$	Real Time at which (i,k) was submitted by MP_i to RB_i.
$f_i(k)$	Market data point used to generate (i,k). This function is not known to anyone besides MP_i.
$rt_i(k)$	Response time of (i,k). $rt_i(k)=A_i(k)-D_i(f_i(k))$.
$O(i,k)$	The order in which OB forwards trades to the ME. If $O(i,k)<O(j,l)$ then (i,k) is ordered before (j,l).

Table 1: Notation.

Communication model: Messages between a RB and the CES are delivered in-order using a reliable transport (such as TCP). Connection disruptions can be handled using timeouts (§5).We focus on trades generated directly in response to the real-time market data. [1]

No failures: We do not consider scenarios where the MPs or the CES fail. Existing solutions for replicating state machines can potentially be used to handle such failures [6]. In the event of a failure, our approach will likely incur unfairness.

2.2 Related Work

CloudEx [3] leverages clock synchronization for fairness. Market data x is delivered simultaneously to all MPs at a pre-specified threshold (Th) from generation time (i.e., $D_i(x) = \max(R_i(x), G(x)+Th)$. Incoming trades are simply processed in the order of submission time ($O(i,k)=A_i(k)$). An MP experiences unfairness if the latency between the CES and the MP goes beyond Th and market data is delivered to it later than intended. Our approach does not require synchronized clocks. This is because, our approach only requires measuring time intervals locally at each RB. For such purposes, each RB can use its own local clock as long as the clock drift rate is small ($<0.02\%$ in practice under a wide range of scenarios [8]).

3 Requirements for Achieving Fairness

In this section, we will establish fundamental requirements for achieving fairness. We consider different variants of fairness. At a high level, each variant argues that trades generated by

[1] MPs can also have access to external streams of data such as financial news streams. Compared to real-time market data, such external streams are not latency-critical and important for fairness.

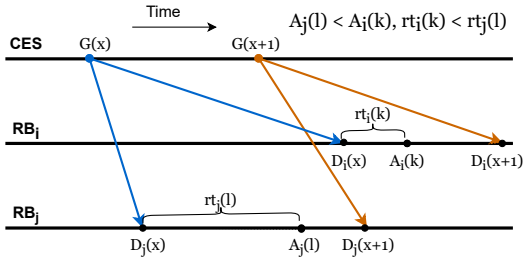


Figure 2: DBO can help correct for late delivery of data. Delivery of market data to MP_i is lagging behind MP_j. There are two trades (i,k) and (j,l) generated in response to the same market data x. (j,l) was submitted before (i,k) but response time of (i,k) is less than (j,l). With DBO, $O(i,k)(=\langle x, rt_i(k)\rangle) < O(j,l)(=\langle x, rt_j(l)\rangle)$ and trade (i,k) is correctly ordered ahead of (j,l). Ordering based on the submission time leads to incorrect ordering.

different market participants based on the same market data information should be ordered based on the response time of the market participants (i.e., faster MP's trades should be ordered ahead of other MPs). Note that, in this section our goal is not to provide exact schemes for the delivery and ordering processes; rather we only aim to establish the minimum constraints on them to achieve perfect fairness. We will use these constraints as guiding principles to propose concrete schemes in the next section and show that these schemes can provide fairness with a high probability. We begin by introducing delivery-time-based ordering (DBO) and showing how it can be used to improve fairness.

Delivery Clock: Competing MPs make trade decisions directly in response to the same market data stream. Events corresponding to the delivery of the same market data point to different MPs, thus, relate to the same conceptual event. Each RB maintains a separate delivery clock to track these conceptual events. Delivery clock is represented by a lexicographical tuple and it increases monotonically with time. Formally, delivery clock at RB_i at time t is given by,

$$DC_i(t) = \langle x_l(t), t - D_i(x_l(t))\rangle. \quad (1)$$

where $x_l(t)$ is the latest data point that was delivered to MP_i (i.e., $D_i(x_l(t)) \le t < D_i(x_l(t)+1)$). Interval, $t-D_i(x_l(t))$, corresponds to the time that has elapsed since the latest delivery.[2] This tuple tracks the progress of market data delivery to the corresponding MP.

DEFINITION 1. *DBO satisfies the following condition,*

$$O(i,k) = DC_i(A_i(k)). \quad (2)$$

With DBO, trades are ordered based on the RB *delivery clock time at trade submission*.[3] In other words, DBO is ordering trades from MPs relative to when they received the market data. Intuitively, DBO can be thought of as a post hoc way of correcting for time differences in delivery of market data to MPs. For example, if market data delivery to a particular MP lags behind other MPs (e.g., due to a latency spike), then its

[2] $t - D_i(x_l(t))$ can be computed based on the local clock of RB_i.

[3] For DBO, each RB can tag this delivery clock time in trades before forwarding them to the CES.

delivery clock also lags behind. Compared to ordering trades based on the submission time, with DBO, trades from this MP receive a boost in ordering that can correct for the late delivery (example in Fig. 2).

DBO alone is capable of partially correcting differences in market data delivery across MPs. Perfect correction, would require measuring the response time of a trade. The challenge, however, is that a trade could have been generated in response to any of the market data points delivered to the MP (and not just the latest data point x_l). The RB/OB cannot trust the MP to truthfully offer this information. We show that we can alleviate this issue by enforcing certain restrictions on how the delivery clocks advance across RBs (i.e., restrictions on the pace of market data delivery to the MPs).

We will now derive the minimum requirements on the delivery processes for achieving fairness for arbitrary trade orders for *any* ordering process.[4] Further, if these delivery requirements are met, DBO ensures perfect ordering for fairness.

3.1 Strong Fairness

DEFINITION 2. *An ordering process O is strongly fair if it satisfies the following conditions,*

$C1$: *If* $A_i(k) < A_i(l)$, *then,* $O(i,k) < O(i,l)$.

$C2$: *If* $f_i(k) = f_j(l) \wedge rt_i(k) < rt_j(l)$, *then,* $O(i,k) < O(j,l)$.

Condition $C1$ states that a MP is always better-off submitting the trade order as early as possible. $C2$ states that trade orders generated based on the same market data point should be ordered based on the response time of the MPs.

THEOREM 1. *The necessary and sufficient conditions on the delivery processes for strongly fair ordering are given by,*

$$D_i(x+1) - D_i(x) = D_j(x+1) - D_j(x), \qquad \forall i,j,x.$$

The theorem states that for strong fairness the inter-delivery times should be the same across all MPs. In other words, the delivery clocks at all RBs (at any given delivery clock time) must advance at the same rate. Please see [5] for details of the proof. We also show that if the above conditions are met then DBO satisfies the fairness properties.

On impossibility of strong fairness: It is possible to show that if communication latency is not bounded then RBs cannot achieve the same inter-delivery times always. In the interest of space we skip this proof. The high-level idea is that if two RBs can achieve the same inter-delivery times then they can also agree to execute some task (not known at the start) simultaneously. From earlier work on the folklore two-generals-problem [4], it is well known that such simultaneous execution is impossible if the communication latency is not bounded. While it might not be possible to achieve the same inter-delivery times all the time, we can achieve it with high probability at the cost of some additional latency at the RB [3].

Next, we consider two weaker variants of fairness. We give the necessary and sufficient conditions for the delivery processes in each case. The proofs in each case are similar to that of Theorem 1. Again, if the conditions are met then DBO satisfies the fairness properties. Depending on the needs of the application one can also consider alternate definitions of fairness, derive the desired properties and construct schemes that try to ensure these properties for fairness.

[4]We assume f_i is not known to the ordering process.

3.2 Limited Fairness

DEFINITION 3. *An ordering process O ensures limited fairness if it satisfies* $C1$ *and the following condition,*

$C3$: *If* $f_i(k) = f_j(l) \wedge rt_i(k) < rt_j(l) \wedge rt_i(k) < \delta$, *then,* $O(i,k) < O(j,l)$.

where δ is a positive constant.

Intuitively, $C3$ states that if the response time of a MP is bounded, then its trades will be ordered ahead of corresponding trades from other MPs as long as it is faster than other MPs. This definition is most relevant in the high frequency trading world where the response time is in the order of a few microseconds.

COROLLARY 1. *The necessary and sufficient conditions on the delivery processes for limited fairness are given by,*

If $D_i(x+1) - D_i(x) < \delta$, *then,*

$$D_j(x+1) - D_j(x) = D_i(x+1) - D_i(x), \qquad \forall j.$$

Compared to strong fairness, the above condition offers some leeway for how the delivery clocks can advance. In §5, we will consider a simple scheme for data delivery that meets the above condition at all times regardless of the variations in latency.

3.3 Approximate Fairness

DEFINITION 4. *An ordering process O is approximately fair if it satisfies* $C1$ *and the following condition,*

$C4$: *If* $f_i(k) = f_j(l) \wedge rt_i(k) \cdot (1+\epsilon) < rt_j(l)$, *then,* $O(i,k) < O(j,l)$,

where ϵ is a positive constant.

Intuitively, $C4$ states that as long as a MP is faster than other MPs by a certain margin, it's trades will be ordered ahead.

COROLLARY 2. *The necessary and sufficient conditions on the delivery processes for approximately fair ordering are given by,*

$$D_i(x+1) - D_i(x) \leq (D_j(x+1) - D_j(x)) \cdot (1+\epsilon), \quad \forall i,j,x.$$

The above condition also offers some leeway for inter-delivery times to differ. This leeway can be useful for masking fluctuations in latency (§5).

4 Discussion

4.1 Enforcing DBO at the Ordering Buffer

The latency from the MPs to the CES could also be variable. Such variations do not affect fairness as long as the OB only forwards a trade (i,k) to the ME once it has (received and) forwarded all other trades (j,l) that should be ordered ahead of (i,k) (i.e., $O(j,l) < O(i,k)$). This requirement means that the OB might need to delay received trades for a certain duration (buffering) before forwarding them to the ME. There

are multiple ways to achieve this requirement; we describe a simple one here. Each RB sends an acknowledgement (ACK) for every market data point it delivers to the MP. We assume that ACKs and trades from each RB are delivered to the OB in-order. An ACK from RB_j for data x thus tells the OB that it has received all trades (j,l) from MP_j s.t., $O(j,l) \leq \langle x,0 \rangle$. The OB uses a priority-queue to buffer trades. The OB uses the ACK information to forward trades respecting the above requirement. In the event of RB failures, the OB could stall indefinitely waiting for ACKs from a failed RB; we can protect against such scenarios by introducing a timeout threshold.

4.2 DBO Cloud Architecture

In a typical on-premise deployment, the CES servers and physical network are part of the trusted infrastructure of the exchange: the exchange operators have exclusive access to the physical machines, and network cables. On the other end, the MPs own the physical servers that connect to the exchange network. Migrating such components to the public cloud seems intuitive: CES servers and MPs could correspond to virtual machines owned by the different parties.

Compared to on-premise deployments one important differentiation is that our solution requires the use of Release Buffer components for correctness, hence those need to be part of the trusted infrastructure (i.e., the MPs should have no control over the RBs). As we have discussed, the RBs should be close enough to the MPs, so that the latency between them does not impact fairness. The Release Buffer components need to *pace* the delivery of data to MPs. The cloud operator could provide the facilities required for the RB functionality. We believe that such functionality, could potentially be embedded in many places such as the hypervisor of cloud nodes that host MPs, or better the programmable NIC of such cloud hosts (most operators already have their own programmable smartNICs [2] deployed). There are several challenges that need to be considered such as performance isolation, but the cloud operator already has fine-grained control over the cloud hardware/software stack and can address this problem easily.

5 Achieving Fairness

In §3, we derived the minimum constraints on the delivery processes for achieving different variants of fairness. In each case, we also showed that if these constraints are met, then, DBO achieves fairness. Which property/properties for the delivery process should the cloud-provider aim for and what is the best way to achieve them (with low latency and high probability) depends on many things, including the requirements for fairness of the financial exchange and the nature of latency variations in the cloud-provider. In this paper, we do not attempt to provide any verdict on this question. Instead, in this section we only aim to show that DBO coupled with controlling how delivery clocks advance can indeed provide fairness with high probability in scenarios where the network latency is highly variable. To this end, we propose two simple schemes (with very different trade-offs for fairness) for delivering market data

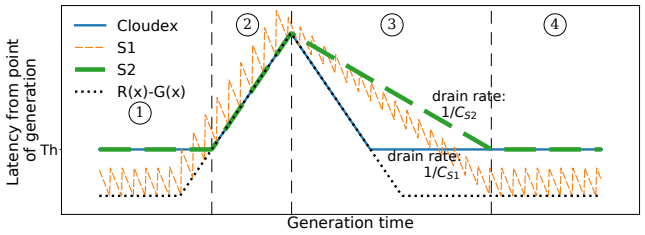


Figure 3: Visualizing delivery times: x-axis shows the generation time of the market data. y-axis plots the delivery times relative to the generation time for market data points (i.e., $D_i(x)-G(x)$) for different schemes. We also include the latency from the CES to the MP for reference (dotted black line). The dashed vertical lines demarcate various regions of interest for scheme S2.

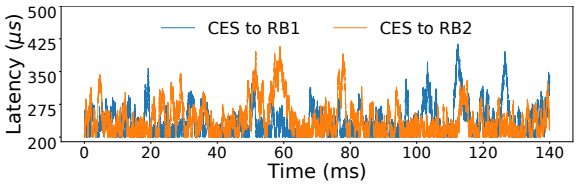


Figure 4: Variations in latency from the CES to the RBs.

that both use DBO for ordering trades. We also compare these schemes against CloudEx.

To help understand these two schemes, we consider a simple scenario where the latency from the CES to a single MP is mostly constant except a transient spike. Fig. 3 depicts the delivery times of each of the two schemes for this particular MP.

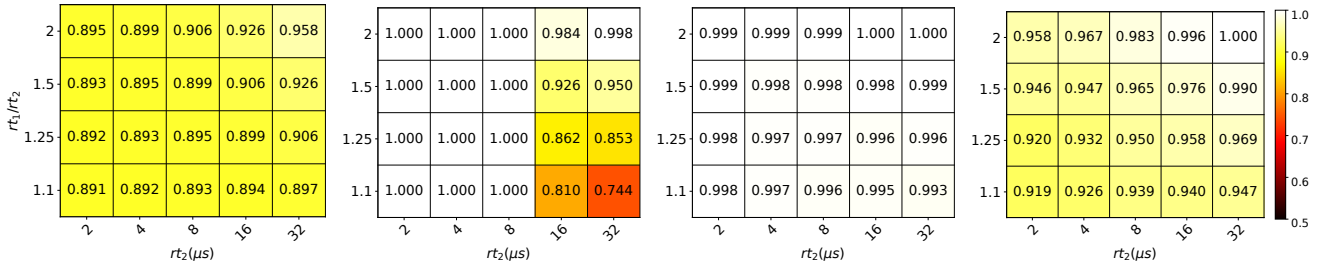
Scheme 1 (S1): The inter-delivery times respect the constraints in Corollary 1 and S1 provides limited fairness regardless of the variations in network latency. In S1, the CES splits the market data points into batches. At the RB, all the market data points corresponding to the same batch are delivered simultaneously (i.e., the inter-delivery time for market data points within a batch is zero). The time interval between delivery of two batches is greater than or equal to δ. Note that, as per Corollary 1, the inter-batch time at a MP can differ from other MPs. As a result, S1 can handle arbitrary latency spikes without violating the inter-delivery constraints. Formally, delivery time of a batch b, $(D_i(b))$, is given by,

$$D_i(b) = \max(R_i(b), D_i(b-1)+\delta),$$

where $R_i(b)$ is the time at which the last market data point in b is received by RB_i. The CES chooses the batch boundaries based on the generation time of the market data. Market data x corresponds to batch number $(b(x))$ given by $b(x) = \lfloor \frac{G(x)}{C_{S1} \cdot \delta} \rfloor$ or equivalently $C_{S1} \cdot \delta \cdot b(x) \leq G(x) < C_{S1} \cdot \delta \cdot (b(x)+1)$, where $C_{S1} (> 1)$ is a constant. With S1 batches can be arbitrarily small and multiple batches can be outstanding within a round trip.

Scheme 2 (S2): For this scheme, we assume clock synchronization between RBs. The mechanism for data delivery is an extension over CloudEx based on the constraints in §3. Formally,

$$D_i(x) = \max\left(R_i(x), G(x)+Th, D_i(x-1)+\frac{G(x)-G(x-1)}{C_{S2}}\right)$$



(a) CloudEx **(b) S1 (Batch delivery + DBO)** **(c) S2 (CloudEx$^+$ + DBO)** **(d) S3 (CloudEx + DBO)**

Figure 5: Fraction of trades that were ordered fairly for different values of response time. Closer to 1 (white) is better.

where $C_{S2}(>1)$ is a constant. Intuitively, if the network latencies for all the MPs are below Th consistently (region ① and ④ in Fig. 3), then, the inter-delivery time at each MP is the same (equal to the inter-generation time, i.e., $D_i(x) - D_i(x-1) = G(x) - G(x-1), \forall i$). The delivery times thus respect the constraints in Theorem 1 and S2 provides strong fairness at such times. Compared to CloudEx, the main differentiation of the scheme above is how data delivery is handled after a latency spike between the CES and a particular MP (third term in the equation). In such cases (region ③), the inter-delivery time differs from the inter-generation time by a relative factor (C_{S2}) and $D_i(x) - D_i(x-1) = \frac{G(x)-G(x-1)}{C_{S2}}$. At such times, if the network latency to other MPs remains consistently low, then the inter-delivery time gaps respect the constraints listed in Corollary 2 (with $\epsilon = C_{S2} - 1$) and $S2$ provides approximate fairness.The inter-delivery time gaps at MPs, however, can differ significantly when latency spikes happen (region ②), hence leading to reduced fairness by S2. Despite this, since S2 uses DBO, it can still correct for late delivery of market data and provide better fairness than CloudEx in such cases.

Note that, it is equally possible to achieve the above inter-delivery properties without clock-sync. We chose S2 specifically to show incremental benefits over CloudEx from incorporating DBO and pacing of market data.

5.1 Simulation Results

To evaluate the proposed schemes, we run a simulation experiment using two dummy MPs with *highly* variable latency between the CES and each RB (generated by a random-walk-like process); see Fig. 4. In CloudEx, latency on the reverse path can cause additional unfairness (which can be alleviated using the buffering process we describe earlier). For simplicity, the latency between each RB and the CES is assumed to be fixed (100 μs). The CES generates market data continuously every $2\mu s$. We assume that both MPs submit a trade in response to every market data point. We do several runs of the experiment with different values of response times for each MP (fixed across data points within a run). For MP2, we vary the response time (rt_2) from 2 to 32 μs. For each value of rt_2, we run the experiment such that response time of MP1 (rt_1) is higher than from rt_2 by some multiplicative factor ($rt_1 = f \cdot rt_2$). We vary f from 1.1 to 2.

	CloudEx	S1	S2	S3
Latency (μs)	403	382	408	406

Table 2: End-to-end latency for different schemes.

To measure fairness, we calculate the fraction of MP2's trades that were executed before the corresponding ones (based on the same market data) from MP1. Fig. 5, shows the results for the three schemes. Table 2 compares the average end-to-end latency[5] (measured across trades) for these schemes.

In CloudEx, Th governs the trade-off between fairness and latency. Here, we set $Th = 300\mu s$: the network latency from CES to the MP is below this threshold most of the time. For S1, we use a small value of $\delta = 14\mu s$. We chose $C_{S1} \cdot \delta = 16$ s.t., the latency is similar/lower to CloudEx. For S2, we use $C_{S2} = 1.095$ (< 1.1) to try to ensure the constraint from Corollary 2 for $\epsilon = 0.095$[6]. We use the same Th as CloudEx to equalize latency.

S1 achieves perfect fairness when $rt_2 < \delta$. When the response time is higher, S1 achieves the worst fairness. It is possible to improve fairness for such trades by additionally trying to ensure that inter-batch delivery times are similar across MPs.

S2 provides close to ideal fairness at the cost of some additional latency. As explained earlier, this is because the combination of DBO and controlling how delivery clocks advance enables S2 to handle latency spikes better than CloudEx. To understand where the wins in S2 are coming from, we consider another scheme. S3 uses DBO and market data delivery is same as CloudEx. DBO on its own helps correct for differences in latency to the MPs and provides better fairness than CloudEx. By controlling the delivery clocks, S2 can further improve fairness.

6 Conclusion

This paper proposes a new approach for achieving response-time fairness in cloud-hosted financial exchanges. We introduce the concept of logical "Delivery Clocks" and show how it can be used to order trades fairly in the presence of highly variable network latency, without requiring clock synchronization. Further, we note that our approach is general and suitable for providing fairness in other settings such as multi-player cloud gaming and advertisement exchanges.

[5]The round trip latency between market data generation and trade handling excluding the response time.

[6]If this constraint is met at all times, then, S2 will achieve ideal fairness when $rt_1 > rt_2 * 1.095$

References

[1] Eric Budish, Peter Cramton, and John Shim. 2015. The high-frequency trading arms race: Frequent batch auctions as a market design response. *The Quarterly Journal of Economics* 130, 4 (2015), 1547–1621.

[2] Daniel Firestone, Andrew Putnam, Sambhrama Mundkur, Derek Chiou, Alireza Dabagh, Mike Andrewartha, Hari Angepat, Vivek Bhanu, Adrian Caulfield, Eric Chung, Harish Kumar Chandrappa, Somesh Chaturmohta, Matt Humphrey, Jack Lavier, Norman Lam, Fengfen Liu, Kalin Ovtcharov, Jitu Padhye, Gautham Popuri, Shachar Raindel, Tejas Sapre, Mark Shaw, Gabriel Silva, Madhan Sivakumar, Nisheeth Srivastava, Anshuman Verma, Qasim Zuhair, Deepak Bansal, Doug Burger, Kushagra Vaid, David A. Maltz, and Albert Greenberg. 2018. Azure Accelerated Networking: SmartNICs in the Public Cloud. In *Proceedings of the 15th USENIX Conference on Networked Systems Design and Implementation* (Renton, WA, USA) *(NSDI'18)*. USENIX Association, USA, 51–64.

[3] Ahmad Ghalayini, Jinkun Geng, Vighnesh Sachidananda, Vinay Sriram, Yilong Geng, Balaji Prabhakar, Mendel Rosenblum, and Anirudh Sivaraman. 2021. CloudEx: a fair-access financial exchange in the cloud. In *HotOS '21: Workshop on Hot Topics in Operating Systems, Ann Arbor, Michigan, USA, June, 1-3, 2021*, Sebastian Angel, Baris Kasikci, and Eddie Kohler (Eds.). ACM, 96–103. https://doi.org/10.1145/3458336.3465278

[4] Piotr J Gmytrasiewicz and Edmund H Durfee. 1992. Decision-theoretic recursive modeling and the coordinated attack problem. In *Artificial Intelligence Planning Systems*. Elsevier, 88–95.

[5] Prateesh Goyal, Ilias Marinos, Eashan Gupta, Chaitanya Bandi, Alan Ross, and Ranveer Chandra. 2022. *Requirements for achieving strong response time fairness*. Technical Report. Microsoft Research. https://github.com/eash3010/strong-fairness-proof

[6] Leslie Lamport. 2001. Paxos made simple. *ACM SIGACT News (Distributed Computing Column) 32, 4 (Whole Number 121, December 2001)* (2001), 51–58.

[7] Leslie Lamport. 2019. Time, clocks, and the ordering of events in a distributed system. In *Concurrency: the Works of Leslie Lamport*, Dahlia Malkhi (Ed.). ACM, 179–196. https://doi.org/10.1145/3335772.3335934

[8] Yuliang Li, Gautam Kumar, Hema Hariharan, Hassan M. G. Wassel, Peter Hochschild, Dave Platt, Simon L. Sabato, Minlan Yu, Nandita Dukkipati, Prashant Chandra, and Amin Vahdat. 2020. Sundial: Fault-tolerant Clock Synchronization for Datacenters. In *14th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2020, Virtual Event, November 4-6, 2020*. USENIX Association, 1171–1186. https://www.usenix.org/conference/osdi20/presentation/li-yuliang

[9] Vasilios Mavroudis and Hayden Melton. 2019. Libra: Fair Order-Matching for Electronic Financial Exchanges. In *Proceedings of the 1st ACM Conference on Advances in Financial Technologies, AFT 2019, Zurich, Switzerland, October 21-23, 2019*. ACM, 156–168. https://doi.org/10.1145/3318041.3355468

[10] Radhika Mittal, Vinh The Lam, Nandita Dukkipati, Emily R. Blem, Hassan M. G. Wassel, Monia Ghobadi, Amin Vahdat, Yaogong Wang, David Wetherall, and David Zats. 2015. TIMELY: RTT-based Congestion Control for the Datacenter. In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication, SIGCOMM 2015, London, United Kingdom, August 17-21, 2015*, Steve Uhlig, Olaf Maennel, Brad Karp, and Jitendra Padhye (Eds.). ACM, 537–550. https://doi.org/10.1145/2785956.2787510

[11] NASDAQ. [n. d.]. Nasdaq and AWS Partner to Transform Capital Markets. https://www.nasdaq.com/press-release/nasdaq-and-aws-partner-to-transform-capital-markets-2021-12-01.

[12] POSTTRADE. [n. d.]. CME and Nasdaq move their markets to the cloud. https://posttrade360.com/news/technology/cme-and-nasdaq-move-their-markets-to-the-cloud/.