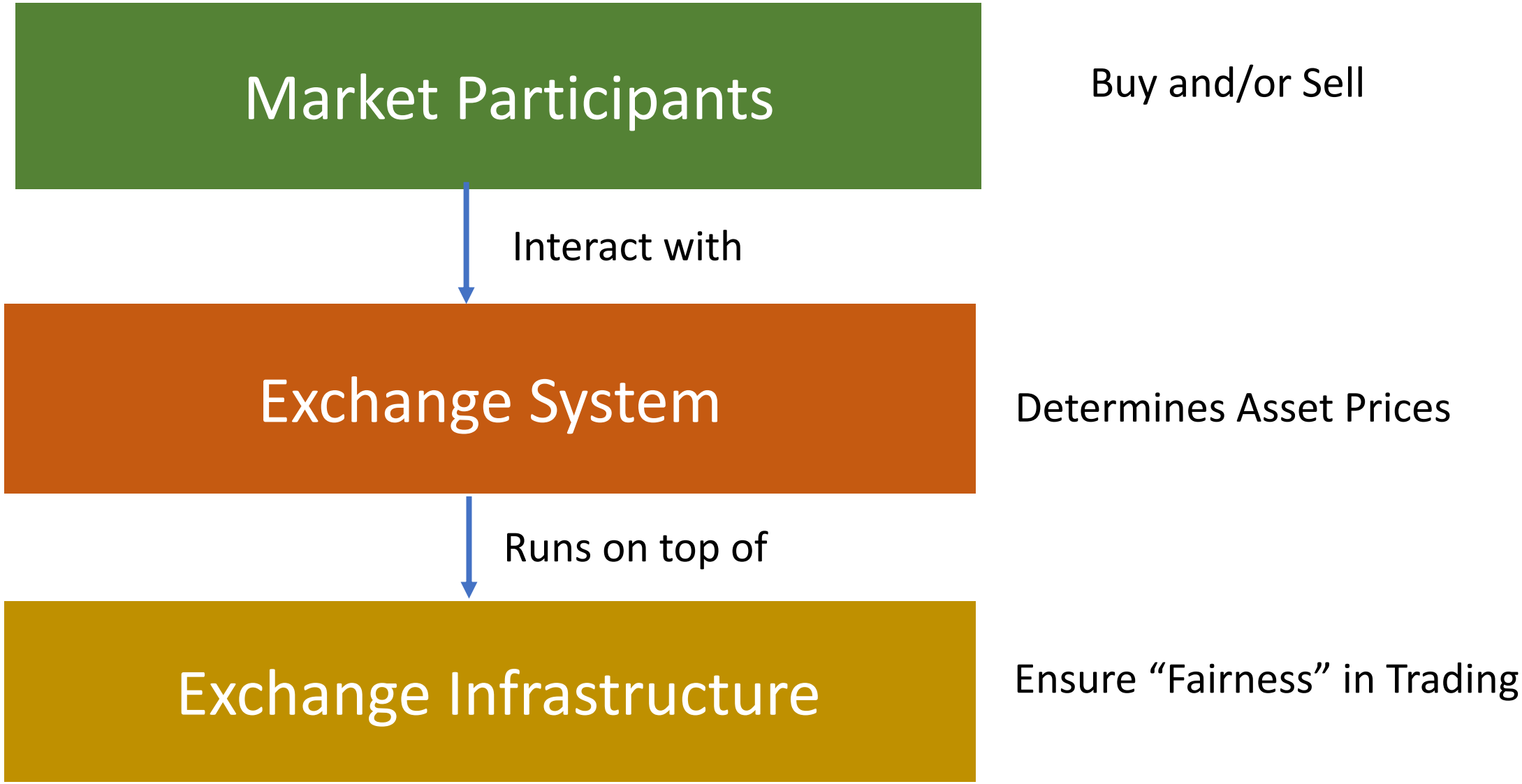
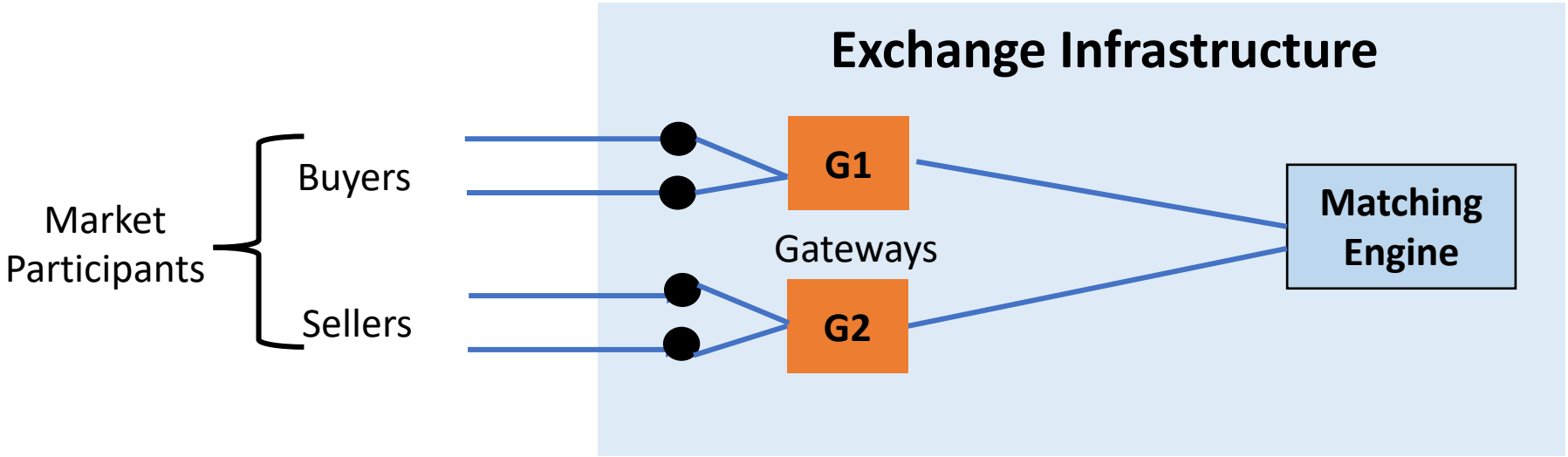
CloudEx: Building a Jitter-free Financial Exchange in the Cloud

Jinkun Geng, Vinay Sriram, Ahmad Ghalayini,
Vig Sachidananda, Balaji Prabhakar and Mendel Rosenblum

Trading Exchanges: Main Components and Goals



Definition of Fairness



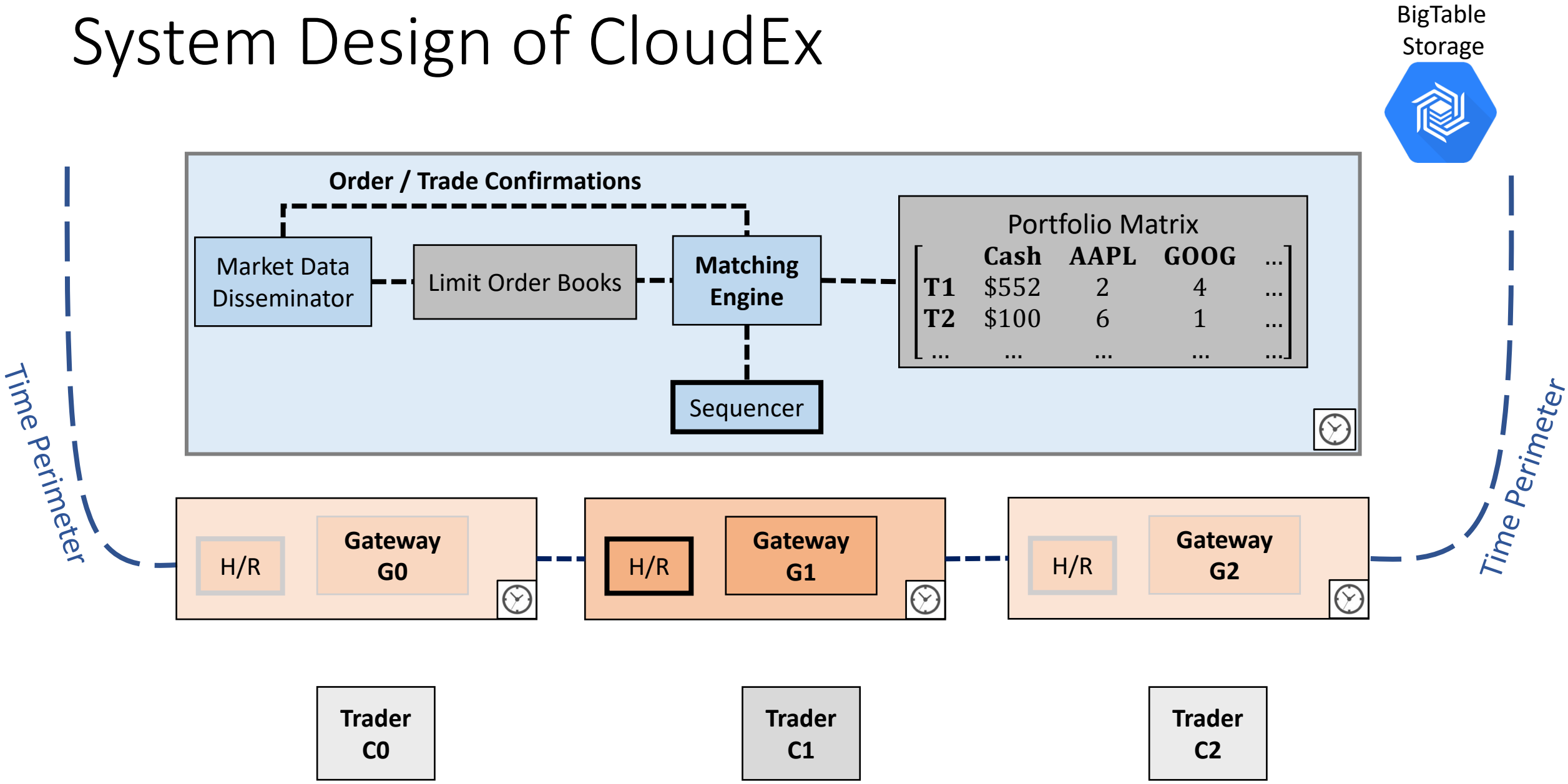
Fairness Requirements

- **Inbound**: Orders are processed in globally FIFO manner, regardless of which gateway (G1 or G2) they arrive at
- **Outbound**: Market data (i.e. trades and limit order book) is simultaneously released to market participants.

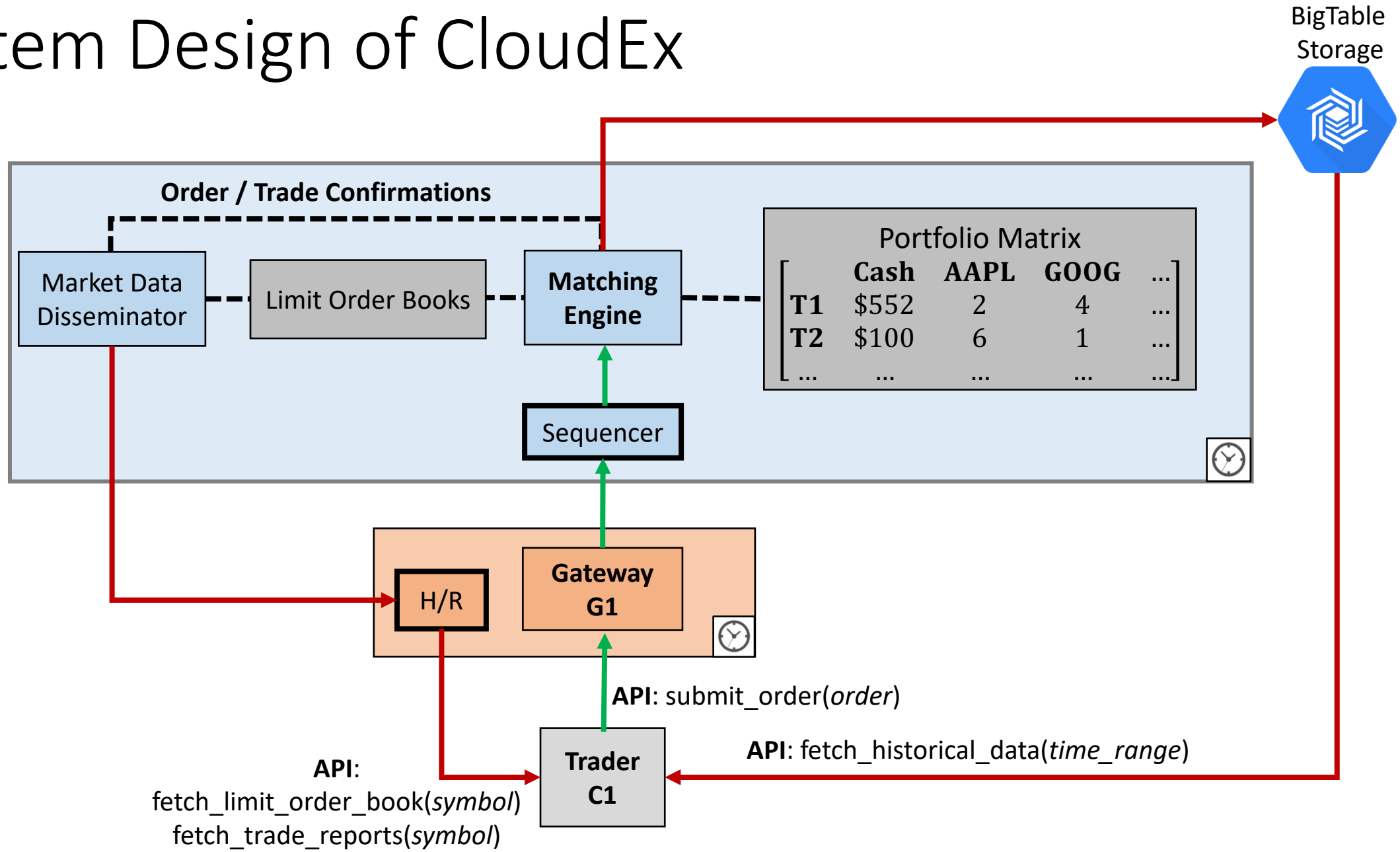
Motivation for CloudEx

- Carefully-engineered networks are expensive to scale and maintain
  - Cloud-based solutions are elastic and typically easier to maintain
- Research tool
  - Fully configurable end-to-end exchange
- Teaching tool
  - Prototype system for learning about (i) infrastructure, (ii) trading algorithms, and (iii) matching algorithms

System Design of CloudEx



System Design of CloudEx



Major Components of our Work

1. Infrastructure Development
   - Networking: Market participant → Gateway →Resequencing Buffer → Matching Engine → H/R buffers → Market Participants
   - The Trading Exchange: Matching Engine, Limited Order Book System, Portfolio Matrix, Big Table for market data persistence and dissemination

2. Matching Engine and Trading Algorithm Design
   - Currently, we're implementing a continuous price-time matching; in future, we can also try batch auctions and other types of mechanisms
   - A toolkit of basic trading algorithms for automatic trading

3. Data Analysis
   - Network traffic: timestamped data at the MP, Gateway, RB, ME and H/R buffers
   - Asset prices: stock price variations as a function of trader strategies and algorithms

Evaluation Setup

- Deploy VMs in Google Cloud
  - 1 gateway VM serves 3 trader VMs, and 1 matching engine VM serves all gateways
- Leverage software-based (i.e., use VM clocks) clock synchronization algorithm[1] for:
  - Resequencing orders
  - Hold-and-releasing market information
- Trading setup:
  - Limit 1 outstanding order per trader, with every trader submitting one new random order as soon as they receive their outstanding order's confirmation
  - 8 symbols available for trading
- Conduct two experiments with different number of traders:
  - 48 traders (16 gateways, 1 matching engine)
  - 96 traders (32 gateways, 1 matching engine)

1. Geng, Yilong, et al. "Exploiting a natural network effect for scalable, fine-grained clock synchronization." 15th USENIX Symposium on Networked Systems Design and Implementation (NSDI 18). 2018.

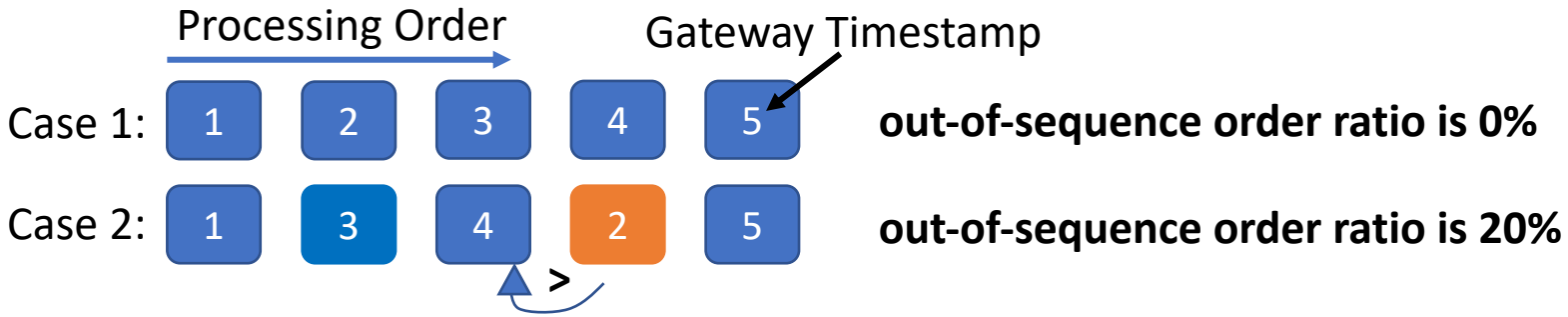
Evaluation Parameters and Metrics

Parameters:

- **resequencing delay parameter:** the *minimum* duration we force all orders to experience (1) from when they get timestamped at the gateway (2) to when they get processed at matching engine.
  - note that if the order's **buffer queueing delay** is large enough**,** no extra waiting is incurred

Metrics:

- **(fairness metric) out-of-sequence order ratio:** the percentage of processed orders whose gateway timestamp is smaller than the previous processed order

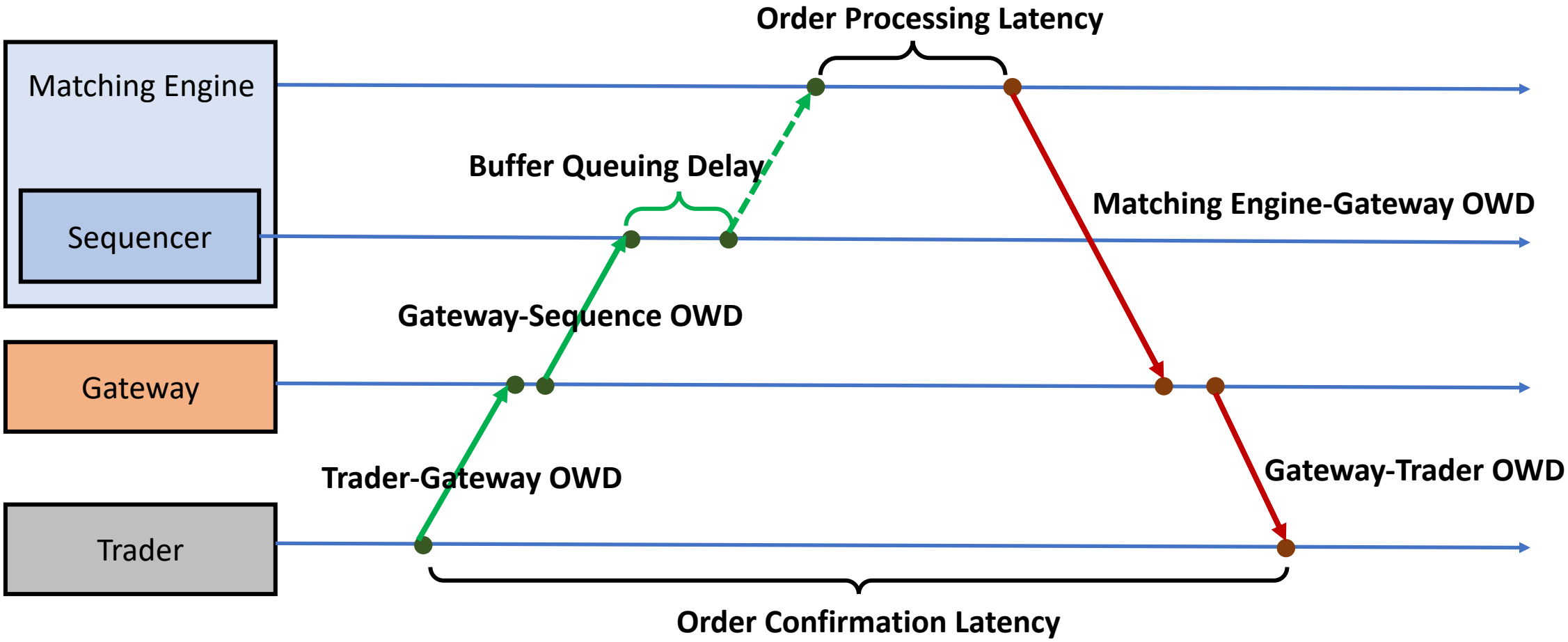


- **(performance metric) number of orders per second:** the average number of orders that the matching engine can process in a second

Resequencing Delay Parameter

- We implement the resequencing buffer as a priority queue which dequeues buffered orders with the *smallest gateway timestamp first*
  - if we disable priority dequeuing: **out-of-sequence order ratio > 20%**
- For each experiment, we sweep the resequencing delay parameter across the following values (in milliseconds):
  - 0, 0.1, 0.2, 0.3, 0.4, 0.5, 1, 2, 3

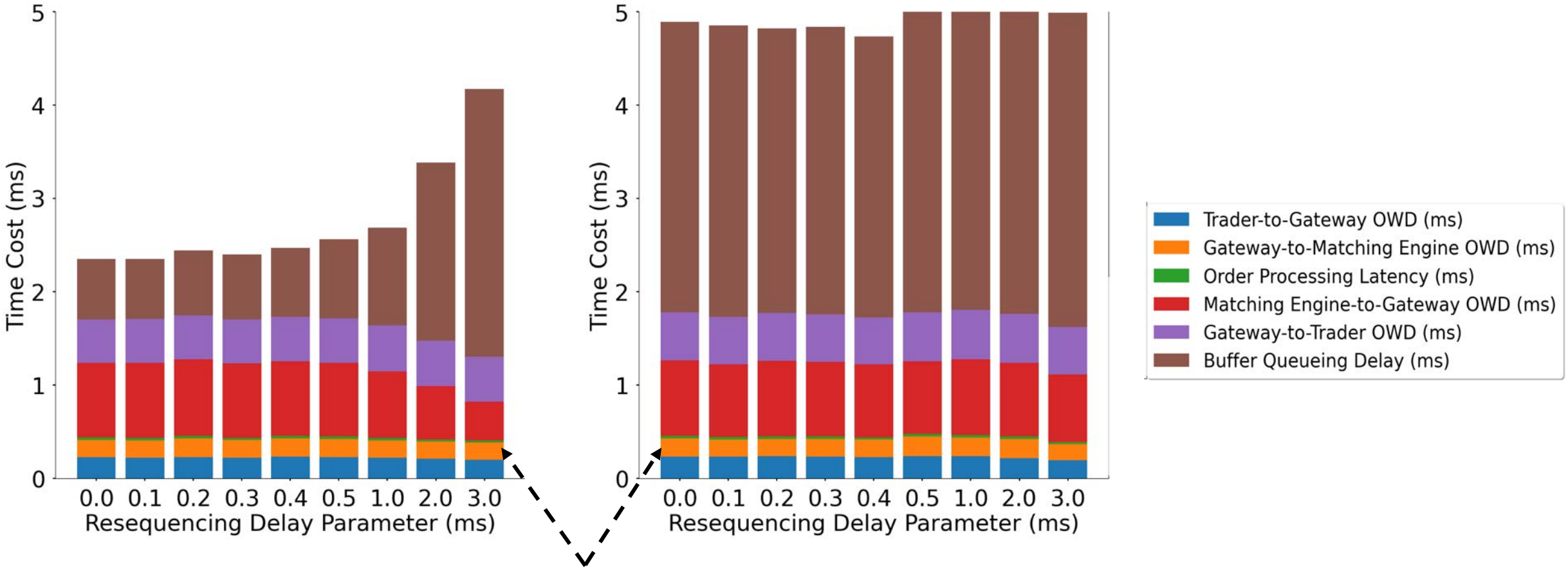
Measurement Setup



Latency Comparison

48-Trader Experiment

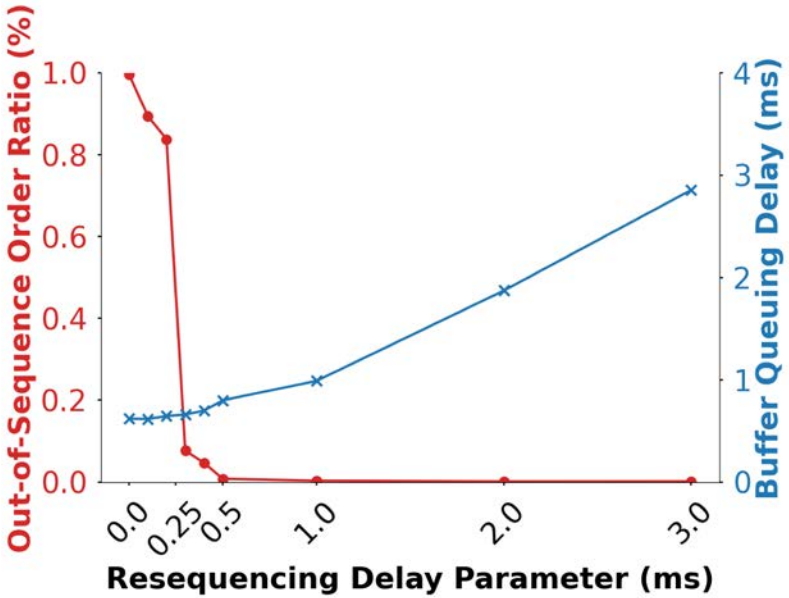
96-Trader Experiment

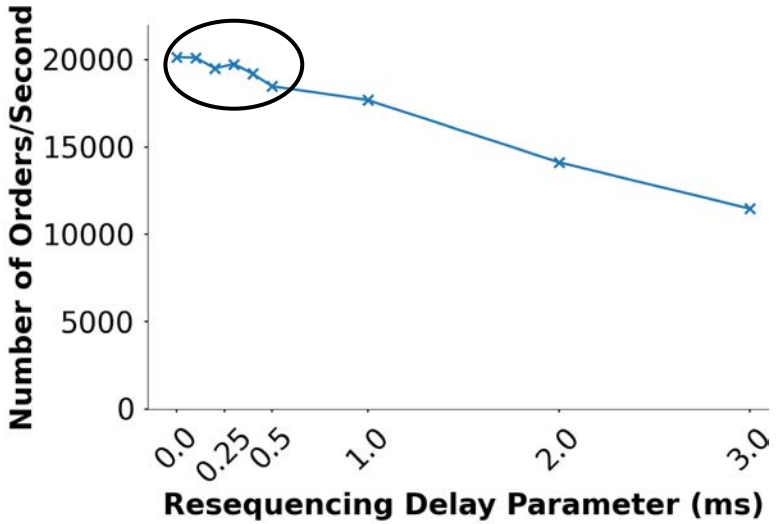


Gateway to Matching Engine OWD (ms)
1p: 0.125, mean: 0.192, 99p: 0.330

→ important to consider for an effective resequencing delay parameter value

48-Trader Experiment

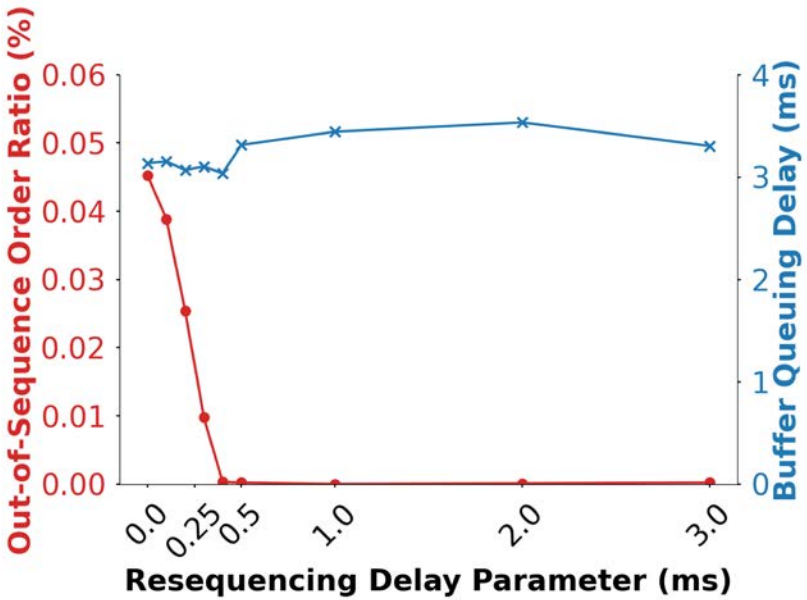


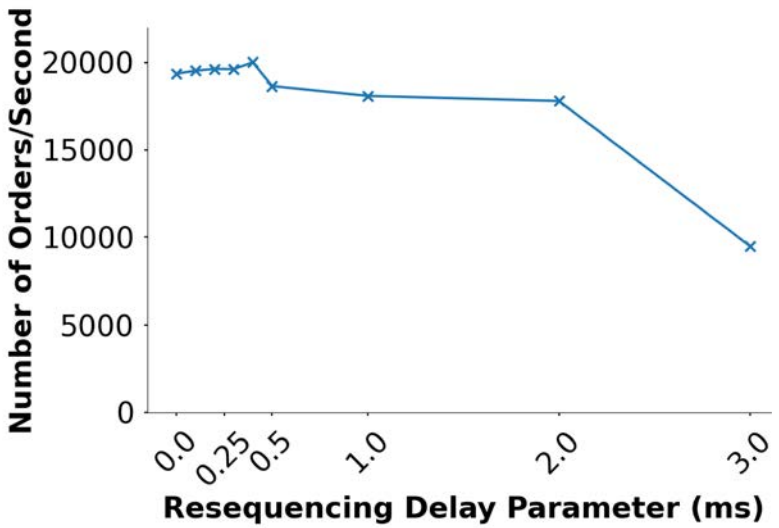


- Larger *resequencing delay parameter* leads to lower *out-of-sequence-order ratio*
  - In this case: 0.5ms offers the best balance between buffer queueing delay (0.799ms) and out-of-sequence order ratio (0.008%)

- *Resequencing delay parameter* **does not** degrade matching engine throughput (i.e. *number of orders per second*) under a certain threshold

96-Trader Experiment





- When matching engine is slow, more traders lead to longer buffer queuing delays
  - the *resequencing delay parameter* becomes *less* significant for decreasing the out-of-sequence order ratio

Conclusion

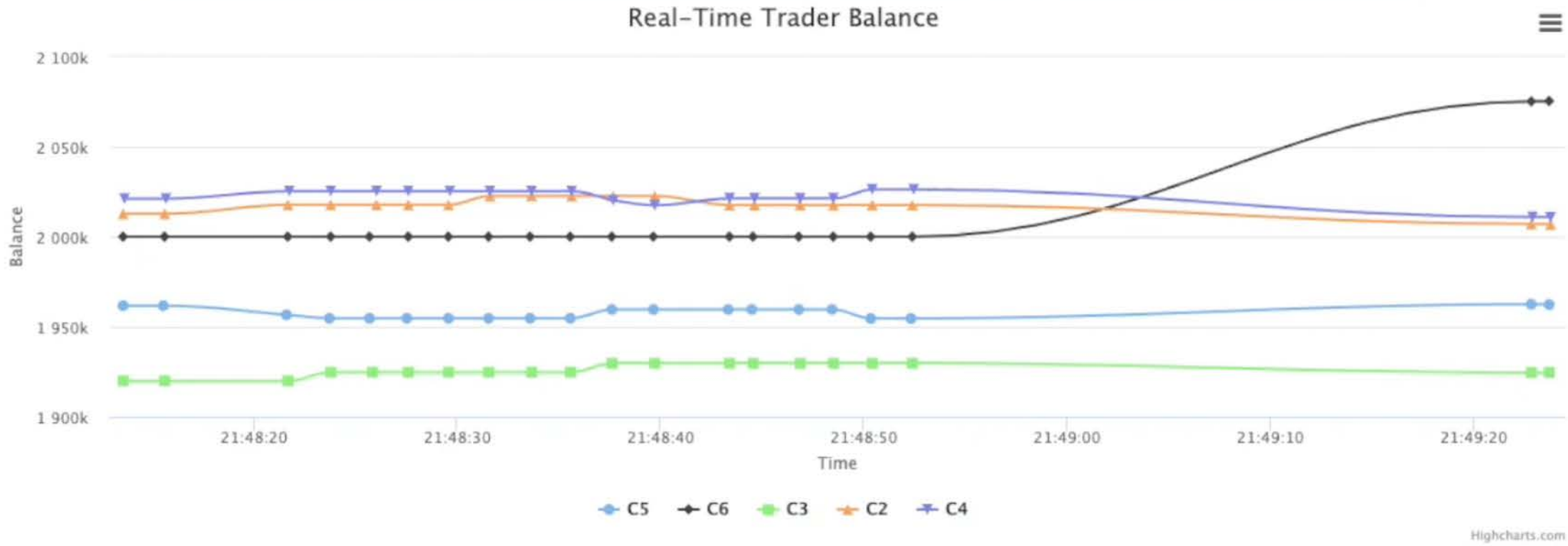
- Cloud environments can be challenging for building fair exchanges
  - Clock-based resequencing can effectively enable fairness
  - With NIC timestamps, the resequencing delay parameter can be reduced by an order of magnitude
- Future work:
  - Infrastructure
    - Sharding matching engine to enhance order processing speed and throughput
    - Tolerate exchange failures while guaranteeing continuous service availability
  - New algorithms and scenarios
    - Design new algorithmic trading strategies and matching engine policies
    - Consider two (or more) categories of market participants: close/distant from gateways
  - Analysis
    - Build a framework for the automatic analysis of traffic and for raising alerts
    - Build methods/tools for finding correlations in asset prices and network traffic at: (i) different timescales, and (ii) across assets and nodes (market participants/gateways)

Cloud Exchange

Demo

Trader | My Orders | My Trades | Portfolio | Limit Order Book | Share Prices

CASH



Client ID	AA	AB	AC	AD	AE	AF	AG	AH	CASH
C6	1998836	2000000	2000000	2000000	2000000	2000000	2000000	2000000	2070299