The Financial Exchange as a Platform

Balaji Prabhakar
Departments of Electrical Engineering and Computer Science
Stanford University
A quest to make networks

*autonomous*: network should sense and monitor itself; program and control itself

*interactive*: network should be simple and fun to use, especially for 3rd party users

**SPN** = Engineering + Design

**Autonomy**
- Systems
- Programs
- ML/AI Algorithms

**Interactivity**
- Visual
- Transparent
- “Friendly and chatty”
**Self-Programming Network (SPN)**

- **Workload**
- **Sense**
- **Control**
- **New Functionality**
  - Timestamping As A Service
  - Fine-grained Network Telemetry
  - App-Network Perf Monitoring

**Plain Old Data Center**

**Sense, Infer, Learn and Control (SILC)**

**Operator Policies**

**Interactive Dashboard & Query Engine**
Self-Programming Network (SPN)

- **SPN App**
- **Workload**
- **Plain Old Data Center**
- **SILC**
- **Sense, Infer, Learn and Control**
  - **New Functionality**
    - Timestamping As A Service
    - Fine-grained Network Telemetry
    - App-Network Perf Monitoring

- **Interactive Dashboard & Query Engine**
  - Intuitive DB and QE
    - Simple + visual + chatty
    - App+network perf views
  - NIC-centric Architecture
    - Sensing and control at NICs
    - Smart NICs: big industry trend
  - Data and ML Intensive
    - Use data and NNs to accelerate learning and for real-time processing
Huygens: A Scalable and Accurate Software Clock Synchronization System
With NIC timestamps: nanosecond-level accurate
With CPU/VM/container timestamps: microsecond-level accurate

Computing

Shiyu Liu
- Distributed Databases: Consistency
- Clockchain: Distributed Ledgers
- CRaft: Consensus Protocols
- Distributed Tracing

Feiran Wang

Vig and Vin Sachidananda
- ChatBots

Networking

SIMON: Accurate, Edge-based Network Telemetry

Yilong Geng, Zi Yin and Shiyu Liu

Smart NIC-based Control

Yilong Geng, Zi Yin and Shiyu Liu

Shiyu Liu and Sean Choi
Demo of Clock Sync System
Technological Bifurcations 50-60 Years Ago

**Networking**
- Circuit-switched → Telephony, no internal buffering, high QoS
- Packet-switched → Internet, packet buffers, random packet delays
  **Key problem:** Loss of “jitter-free networks”

**Computing**
- Centralized → Eniac, supercomputing/high-performance, real-time
- Distributed → Cluster computing/batch processing, high throughput
  **Key problem:** Need to coordinate cluster to achieve consensus and provide consistency guarantees

Packet-switching and distributed computation have been an unqualified success
- **Packet-switched Networks:** Internet, Data Centers
- **Distributed Computing:** Cloud Computing, Distributed Databases/Ledgers, Online e-Commerce, ...
In this talk ...

Look at Financial Trading Systems (esp an Exchange) as a Platform

• That is, there exist specific primitives (trust, fairness, security) the Platform ought to offer
• In electronic markets, these are provided by networking and computing technologies
• Protocols are developed to enable “trading” (just as network protocols enable “communication”)

Consider problems in financial trading caused by network “jitter”, and ask

• How can nanosecond-level clock sync at scale remedy these problems?
• How to obtain accurate clock sync at scale in commodity networks without special hardware or infrastructure investment?

Focus on problems relevant to the financial industry

• Current exchange networks in carefully-engineered data centers
• Future exchange networks in ad hoc, heterogeneous cloud environments
  ➔ Including some demos
Financial Trading Platforms

- **Financial Trading Systems**
  - Trading, Clearance, Settlement, Security, ...
  - Systems for enabling machines to trust and trade

- **Computing**
  - Systems for Computing, DBs, Storage, ML/AI, ...

- **Networking**
  - Protocols, systems to enable machines to talk to each other
Requirements and Challenges in Financial Trading

**Fairness**
1. Ensure the in-order execution of transactions *regardless* of the gateway through which the transaction arrives at the exchange
2. Ensure market data is disseminated to market participants *simultaneously*

**Front running**
3. When a market participant places orders at different exchanges so as to get the best price, ensure they are not beaten by others with faster pipes

**Challenges**
- Provide the above requirements at scale *without* expensive infrastructure investment
- Generalize to exchange networks in ad hoc, heterogeneous cloud environments
#1: In-order Execution of Transactions

In on-prem exchanges, links from Gateways to Matching Engine carefully engineered to ensure equal transit times.

This is not possible in the Cloud; hence more challenging.
Solution: Create “Time Perimeter” Using Accurate Clock Sync

- Create a “Time Perimeter” by synchronizing the Gateway clocks with the Matching Engine
- Timestamp transactions at Gateways to establish precise order of arrival
- Reorder transactions in a “reordering buffer” before execution
Whittling a New York Minute To 100 Billionths of a Second

By JOHN MARKOFF

SAN FRANCISCO — Computer scientists at Stanford University and Google have created technology that can track time down to 100 billionths of a second. It could be just what Wall Street is looking for.

System engineers at Nasdaq, the New York-based stock exchange, recently began testing an algorithm and software that they hope can synchronize a giant network of computers with that nanosecond precision. They say they have built a prototype, and are in the process of deploying a bigger version.

For an exchange like Nasdaq, such refinement is essential to accurately order the millions of stock trades that are placed on their computer systems every second.

Ultimately, this is about money. With stock trading now dominated by computers that make buying and selling decisions and execute them with blazing speed, keeping that order also means protecting profits. So-called high-frequency trading firms place trades in a fraction of a second, sometimes in a bet that they can move faster than bigger competitors.

The pressure to manage these high-speed trades grows when the stock market becomes more volatile, as it has been in recent months, in part to prevent the fastest traders from taking unfair advantage of slower firms. High-frequency traders typically ac-

Continued on Page A17
#2: Deliver Market Data Simultaneously

Market data is currently delivered simultaneously to participants by multicasting.

*Multicasting is not easy in the Cloud*
Solution: Time Perimeters + Hold-and-Release Buffers

- Create Time Perimeters at the Gateways ...
- ... or adjacent to the Market Participants
- Timestamp order books at the Matching Engine and ...
- ... release them simultaneously at (geographically) different locations (Gateways or MP nodes)
Demo of Cloud Exchange
#3: Multi-venue Trades

A trader places an order in Singapore and sends it on to HK and Tokyo.

A Market Maker sees the order in Singapore and anticipates it in Tokyo because they've a faster SG → Tokyo link.

Results in a sub-optimal deal for the trader!
Solution: Time Perimeter + Hold-and-Release Buffers

Create Time Perimeter around SG, HK and Tokyo

1. Hold SG trade in buffer
2. Send it to HK and Tokyo
3. Release it “simultaneously” in SG, HK and Tokyo at future time

Note: Clock sync accuracy only needs to be better than speed of light distance between venues.

Speed of light SG → Tokyo: 18ms
Cloud Exchange in CS 349F
To be offered in Spring 2020
Order Types

• Market Orders
  – Market Buy – Fill as much of the order as possible using asks of any price.
  – Market Sell – Full as much of the order as possible using bids of any price.

• Limit Orders
  – Limit Buy – Fill as much of the order as possible using asks $\leq$ limit price.
  – Limit Sell – Fill as much of the order as possible using bids $\geq$ limit price.

• When filling an order, we select the opposing order in the limit order book with the best price, breaking ties using the timestamp (FIFO).
### Trade Messages

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Action</th>
<th>Shares</th>
<th>Type</th>
<th>TTL</th>
<th>Bid/Ask Price</th>
<th>Timestamp</th>
<th>Client ID</th>
<th>Order ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>APPL</td>
<td>Buy</td>
<td>100</td>
<td>Limit</td>
<td>3min</td>
<td>$210</td>
<td>1569290045000</td>
<td>C1</td>
<td>SAD651GH</td>
</tr>
<tr>
<td>GOOG</td>
<td>Buy</td>
<td>50</td>
<td>Market</td>
<td>15ms</td>
<td>$1200</td>
<td>1569290048121</td>
<td>A2</td>
<td>3AS5SDF2</td>
</tr>
<tr>
<td>MSFT</td>
<td>Sell</td>
<td>325</td>
<td>Limit</td>
<td>1day</td>
<td>$140</td>
<td>1569290053007</td>
<td>B2</td>
<td>A22S134H</td>
</tr>
</tbody>
</table>

**Examples**

- **APPL**: Buy, 100 shares at $210 limit price with a 3-minute TTL until 1569290045000, client ID C1, and order ID SAD651GH.
- **GOOG**: Buy, 50 shares at $1200 market price with a 15ms TTL until 1569290048121, client ID A2, and order ID 3AS5SDF2.
- **MSFT**: Sell, 325 shares at $140 limit price with a 1-day TTL until 1569290053007, client ID B2, and order ID A22S134H.