Ch 2. The Critical Components of a Trading System

Understanding the trading system

The key points to consider when designing a trading system

1. Asset Class

- **Characteristics**: Different asset classes (e.g., US equities vs. foreign exchange) have unique characteristics that affect data structure and system architecture.
- **US Equities**: Typically traded on the NYSE and NASDAQ, with around 7,000 listed firms.
- **Foreign Exchange (FX)**: Consists of six main, six minor, and six exotic currency pairs, with potentially up to 100 pairs available across hundreds of exchanges.

2. Trading Strategy Type

- **High Frequency Trading (HFT)**: Requires extremely low latency for order transmission, measured in microseconds (US stocks) or even nanoseconds (CME). C++ or Java are preferred for speed over Python.
- **Long-Term Positioning**: Speed is less critical, focusing more on the ability to execute trades over extended periods (days or weeks).

3. Number of Users and Trading Techniques

- **Scalability**: As the number of traders and trading techniques grows, the volume of orders increases, necessitating a system that can handle the load efficiently.
- **Order Validation**: Ensuring the validity of orders before submission involves compliance checks, position limits, and other regulatory requirements, which can increase processing time.

Key Functions of a Trading System

- **Data Collection**: Capturing real-time market data, including price updates.
- **Order Management**: Sending orders to the exchange and handling responses (canceled, rejected, filled, or partially filled).
- **Performance Metrics**: Calculating and tracking portfolio performance (profit/loss, risk metrics, etc.).

Considerations for Trading System Design

1. Latency and Speed

- **HFT**: Microsecond to nanosecond latency requirements necessitate efficient coding practices and possibly hardware acceleration.
- **Long-Term Trading**: Latency is less of a concern, allowing for more flexibility in programming languages and system design.

2. Scalability

- **User and Order Volume**: The system must scale to accommodate increasing numbers of users and orders.
- **Compliance and Verification**: Ensuring orders comply with regulations and internal policies, which becomes more complex with higher volumes.

3. Regulatory Compliance

- **Order Compliance**: Validating orders against regulatory rules and internal policies to ensure compliance before execution.

Example System Architecture

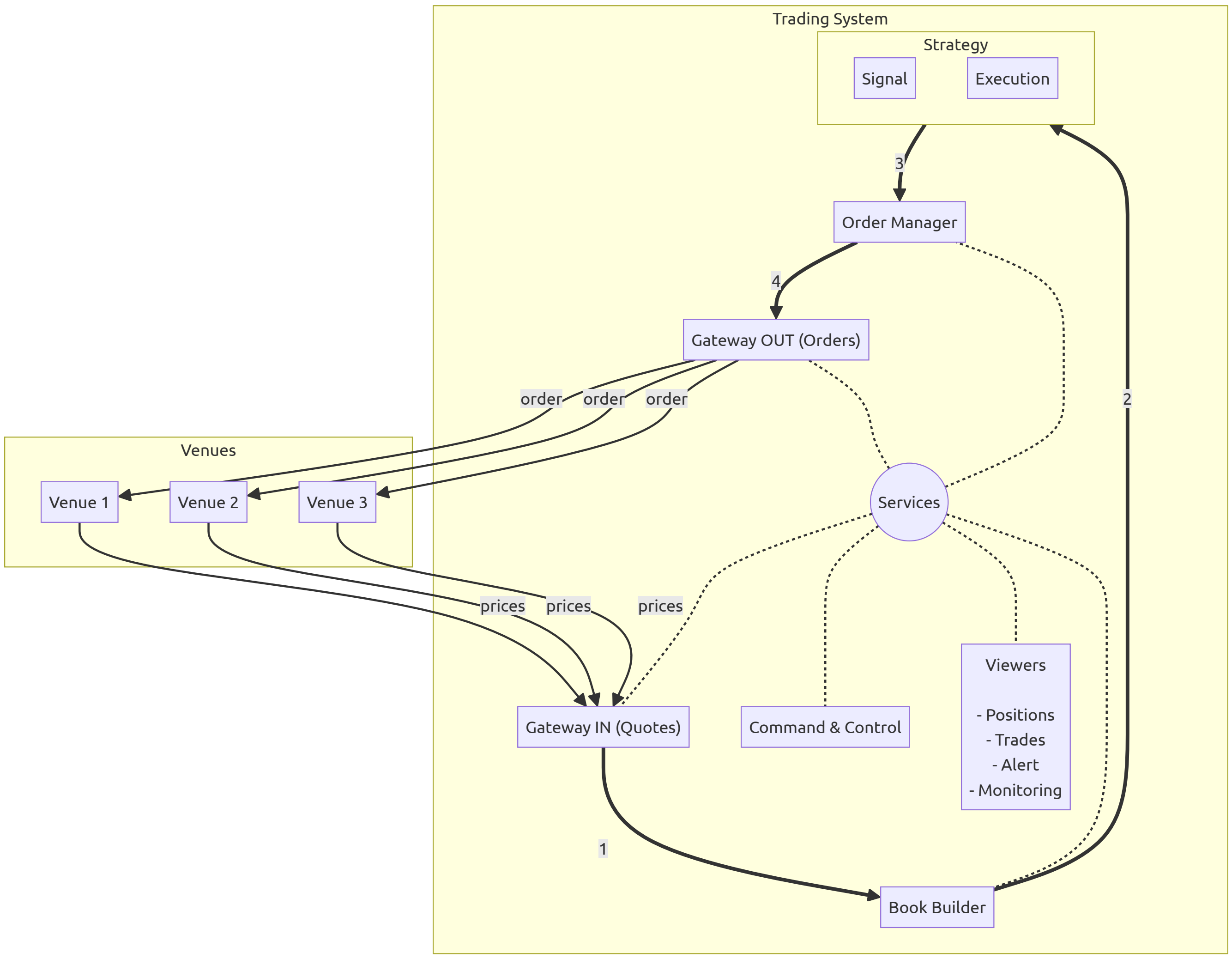
US Equities Trading System

- **Data Source**: Market data from NYSE and NASDAQ.
- **Order Routing**: Efficient order management to handle thousands of firms.
- **Latency Optimization**: Use of low-latency programming languages and possibly hardware acceleration.

FX Trading System

- **Data Source**: Multiple exchanges with up to 100 currency pairs.
- **Order Management**: Handling orders across a diverse set of exchanges.
- **Latency and Scalability**: Focus on latency for high-frequency strategies, and scalable architecture for handling multiple exchanges and pairs.

Trading system architecture



Venues

- **Venue 1, Venue 2, Venue 3**: Platforms that match buy and sell orders for securities and derivatives. These can be trading exchanges, ECNs, aggregators, or banks.

Trading System

- **Gateway IN (Quotes)**: Collects price updates from venues.

- **Book Builder**: Constructs the limit order book using data from Gateway IN.
- **Strategy**: Comprised of Signal and Execution components.
    - **Signal**: Generates trade signals based on the market data.
    - **Execution**: Implements the trade signals.
- **Order Manager**: Manages all orders generated by strategies and tracks their lifecycle.
- **Gateway OUT (Orders)**: Sends orders to the venues.
- **Command & Control**: Starts and stops system components.
- **Viewers**: Displays positions, trades, alerts, and monitoring information for system health and trading strategies.
- **Services**: Supports various non-critical functions within the trading system.

Workflow

1. **Price Collection**:
    - Venues (V1, V2, V3) provide price updates to Gateway IN.
2. **Book Building**:
    - Gateway IN sends price updates to the Book Builder, which constructs the limit order book.
3. **Strategy Execution**:
    - Book Builder sends the order book to the Strategy component.
    - The Signal component generates trading signals.
    - The Execution component carries out the trading signals by sending orders to the Order Manager.
4. **Order Management**:
    - The Order Manager collects orders from the Strategy and manages their lifecycle.
    - Orders are sent to the venues through Gateway OUT.
5. **Additional Services**:
    - **Services** provide additional functionalities such as monitoring, data storage, and reporting.
    - **Command & Control** can start or stop system components.

- **Viewers** display critical information for monitoring and alerting purposes.

6. **Order Transmission**:

- Gateway OUT sends orders to Venues (V1, V2, V3).

Gateways Connecting to Trading Exchanges

Gateways are vital components of a trading system, acting as the bridge between the trading exchanges and the trading system. They handle the communication and data transfer, collecting price updates and transmitting orders. These operations are resource-intensive and require efficient handling to ensure minimal latency and high throughput.

Key Functions of Gateways

1. **Data Collection**:

- Gateways collect price updates from various trading venues such as exchanges, ECNs, and dark pools.
- They establish network connections with these venues, authenticate, and subscribe to specific instruments to receive real-time data.

2. **Order Processing**:

- Gateways also handle sending and receiving orders. They transmit orders to the trading venues and handle acknowledgments.
- If an order matches, the venue sends a confirmation message back to the trading system. If the order is not received, the gateway must handle timeouts and errors.

Gateway Workflow

1. **Establish Connection**:

- The gateway connects to the venue's network using various protocols.
- Possible network media include wires, wireless networks, the internet, microwaves, and fibers, each with different speed, data loss, and bandwidth characteristics.

2. **Data Transmission**:

    - Once connected, the gateway subscribes to price updates and receives a continuous stream of data.
    - This data is then forwarded to the trading system for processing and decision-making.
3. **Order Transmission**:

    - The trading system generates orders based on the processed data and sends them to the gateway.
    - The gateway transmits these orders to the venue, which acknowledges receipt and processes the order.
    - The gateway then receives confirmation or error messages from the venue and updates the trading system accordingly.

Data Flow and Communication

- **Bidirectional Communication**: The arrows for price updates and orders are bidirectional, indicating that data is both sent to and received from the venues.
- **Error Handling**: If an order is not acknowledged by the venue, the trading system must declare a timeout, prompting trader intervention.

Communication Handled by Gateways

Gateways in trading systems must handle various protocols to transform them into trading system data structures. This requires a deep understanding of network protocols and APIs provided by trading venues.

Network Protocols

Network protocols define the rules of communication between devices. In trading finance, the common protocols are:

- **User Datagram Protocol (UDP)**: A connectionless protocol that does not guarantee message delivery.
- **Transmission Control Protocol (TCP)**: A connection-oriented protocol that ensures message delivery in the order sent.

Communication API

The communication API, provided by the trading venues, defines how to:

- Send an order
- Receive price updates

Fundamentals of Networking

Networks enable computers to connect and share data using a physical layer. The choice of medium affects speed, reliability, and security. In trading finance, common media include:

- **Wire**: Electrical currents with narrow bandwidth.
- **Fiber**: High bandwidth.
- **Microwave**: Easy setup, high bandwidth, but susceptible to storms.

OSI Model and IP

The OSI model outlines the layers of network communication, starting with the physical layer. In trading finance, IP (Internet Protocol) is part of the network layer, establishing rules for packet routing.

Transport Layer Protocols

- **TCP (Transmission Control Protocol)**: Ensures reliable communication between two machines, maintaining the order of messages.
- **UDP (User Datagram Protocol)**: Does not guarantee message delivery or order, making it faster but less reliable.

Protocols in Trading Finance

Trading exchanges use TCP or UDP for communication. Each has its benefits and trade-offs, with TCP providing reliability and UDP offering speed.

Order Book Management

Order book management in a trading system involves copying the limit order book from various venues into the system. The book builder is responsible for gathering, categorizing, and integrating the different books received from the venues.

Key Components of Order Book Management

- **Gateways**: They collect price updates from the trading venues and transform these updates into a format suitable for the trading system.
- **Book Builder**: It integrates the different order books received from the gateways, sorting and organizing the price changes.

Process Flow

1. **Price Collection**: Gateways receive price updates from venues and pass these updates to the book builder.
2. **Integration**: The book builder integrates the order books from various venues, considering the bids and offers from each.

Structure of an Order Book

An order book for a financial product contains two main parts:

- **Bids**: Orders to buy, listing the volume and price.
- **Offers (Asks)**: Orders to sell, also listing the volume and price.

The offer (or ask) price will almost always be greater than the bid price.

- it would be far too simple to profit if you could purchase for less than you could sell.

Each venue sends its own order book, and the book builder's role is to merge these books into a consolidated view.

Example of an Order Book

- **Venue 1**:
  - **Bid**: 1000 shares at $1.20
  - **Ask**: 1000 shares at $1.21

- **Venue 2**:
    - **Bid**: 500 shares at $1.19
    - **Ask**: 500 shares at $1.22
- **Venue 3**:
    - **Bid**: 2000 shares at $1.18
    - **Ask**: 2000 shares at $1.23

Consolidation by Book Builder

The book builder takes the books from all venues, organizes, and sorts them. It ensures the bid prices are always lower than the ask prices to prevent arbitrage opportunities. The consolidated book might look something like this:

- **Bids**:
    - 2000 shares at $1.18 (Venue 3)
    - 500 shares at $1.19 (Venue 2)
    - 1000 shares at $1.20 (Venue 1)
- **Asks**:
    - 1000 shares at $1.21 (Venue 1)
    - 500 shares at $1.22 (Venue 2)
    - 2000 shares at $1.23 (Venue 3)

Order Book Considerations

Overview

- **Function**: A limit order book collects and arranges price updates (orders) to facilitate trading strategies.
- **Use by Exchanges**: Tracks bids and offers to provide asset price indications and best prices.
- **Communication**: Network required to convey updates to/from the exchange, which is on another platform/server.

Methods for Sending Order Book Updates

1. **Full Book Transfer**:
    - **Description**: Sends the entire order book each time.
    - **Drawbacks**: Time-consuming, especially for large exchanges like NYSE or NASDAQ, potentially saturating the network.
2. **Snapshot with Incremental Updates**:
    - **Description**: Sends a full snapshot initially, followed by incremental updates.
    - **Benefits**: More efficient for large volumes of orders, reduces network load.

Order Book Operations

- **Insertion**:
    - Adds a new order to the book.
    - Requires a data structure with O(1) or O(log n) complexity for efficiency.
- **Amendment/Modification**:
    - Uses the order ID to find and modify an existing order.
    - Complexity should be similar to insertion.
- **Cancelation**:
    - Uses the order ID to remove an order from the book.

Data Structure Considerations

- **Performance Impact**:
    - Crucial for HFT systems to select the appropriate data structure for efficient operations.
    - Order-based books require optimized execution.
- **Key Requirements**:
    - **Constant Lookup & Fast Quantity Update**:

    - Handle millions of orders per second.
    - Must support constant-time lookups for order IDs.
  - **Iteration in Order of Prices**:
    - Efficiently find orders to match desired volumes starting from the best price.
  - **Retrieve Best Bid and Ask in Constant Time**:
    - Essential for rapid trading decisions.

Recommended Data Structures

- **Associative Arrays**:
  - Use structures like `std::unordered_map` or `std::vector` in C++ for order identifiers.
  - Order metadata should reference the order book and price-level directly for quick updates.
- **Price Levels Vector**:
  - Organize each book's price levels in vectors for quick average price lookup.
  - Linear search from the vector's end is typically faster due to cache efficiency.
- **Order Metadata**:
  - Maintain references to the order book and price-level.
  - Store pointers to the next and previous orders to track time priority.

Performance Considerations

- **Efficiency**:
  - Vector-based approach yields fast average lookups, insertions, updates, and deletions.
  - Best-case: O(1) behavior for primary operations.
  - Worst-case: O(N) with low probability, due to cache, TLB, and compiler constraints.
- **Practicality**:
  - Generally provides optimal performance for Best Bid and Offer (BBO) updates.

- Emphasizes the need for deep understanding of computer operating systems and programming for HFT systems.

Strategy Making Decisions on When to Trade

Trading Strategy Components

1. **Signal Component**:
    - **Purpose**: Generates trading signals based on the algorithm representing the trading concept.
    - **Limitations**: Signals do not guarantee liquidity due to potential order rejections, especially in HFT where speed is critical.
2. **Execution Component**:
    - **Purpose**: Handles market responses and decides actions based on those responses.
    - **Example**: If an order is rejected, the strategy attempts to find equivalent liquidity at a different price.

Order Management System (OMS)

- **Function**: Manages the lifecycle of orders submitted by the trading strategy.
- **Order Lifecycle**: Includes creation, execution, amendment, cancellation, and rejection.

Key Features of OMS

- **Order Collection**: Gathers and processes orders from the trading strategy.
- **Error Handling**:
    - **Invalid/Malformed Orders**: OMS can reject orders for reasons such as:
        - Excessive quantity
        - Incorrect direction
        - Erroneous prices
        - Excessive outstanding positions
        - Unsupported order types by the exchange

- **Pre-Exchange Rejection**: Detects and rejects problematic orders within the system, allowing the trading strategy to react faster than if the exchange handled the rejection.

Critical Components of a Trading System

Key Components

1. **Gateways**:

   - Interface between the trading system and external markets/exchanges.
   - Handles communication, order transmission, and market data reception.

2. **Book Builder**:

   - Constructs and maintains the order book from market data.
   - Provides real-time views of market depth and price levels.

3. **Strategies**:

   - Implements trading algorithms to generate signals and execute trades.
   - Divided into:
     - **Signal Component**: Generates trade signals.
     - **Execution Component**: Handles market responses and actions.

4. **Order Management System (OMS)**:

   - Manages the lifecycle of orders (creation, execution, amendment, cancellation, rejection).
   - Ensures quick reaction to invalid or malformed orders before they reach the exchange.

Performance Measurement

- **Tick-to-Trade (Tick-to-Order) Period**:

  - Measures the time from receiving a price update to submitting an order.
  - Aggregates processing times of all critical components to evaluate system speed.

Non-Critical Components

Functions

- **Settings Adjustment**: Modify parameters in real-time.
- **Data Collection and Reporting**: Gather and report trading data.

Example Component

- **Command and Control**:
    - Interface between traders and the trading system.
    - Can be command-line or graphical, facilitating order routing and parameter adjustments.

Additional Services

1. **Position Server**:
    - Tracks all trades and updates positions for financial assets.
    - Ensures compliance with position limits before order execution.
2. **Logging System**:
    - Collects logs from all components.
    - Aids in debugging, issue tracking, and reporting.
3. **Viewers**:
    - **Read-Only User Interface**: Displays trading data (positions, orders, trades, task monitoring).
    - **Control Viewers (Interactive)**: Allows parameter modification and component control.
4. **News Server**:
    - Aggregates news from sources like Bloomberg, Reuters, Ravenpack.
    - Provides real-time or on-demand news to the trading system.