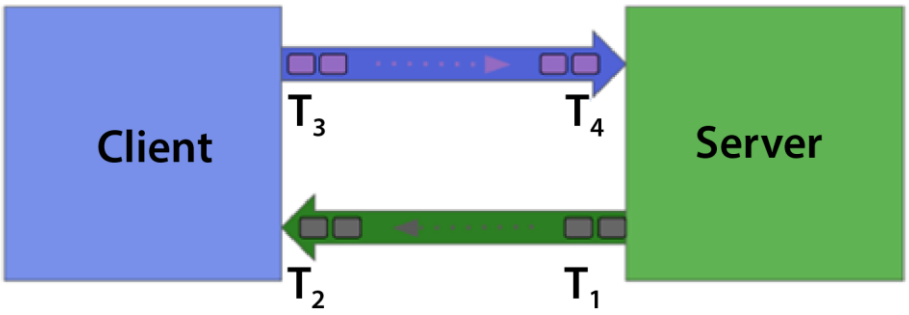
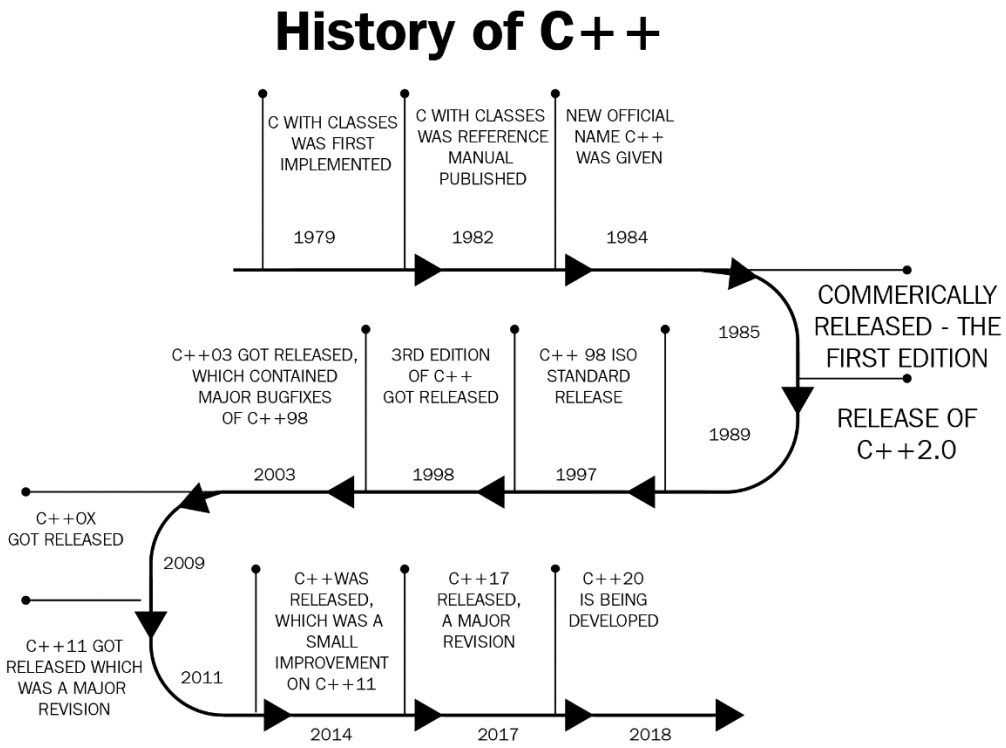
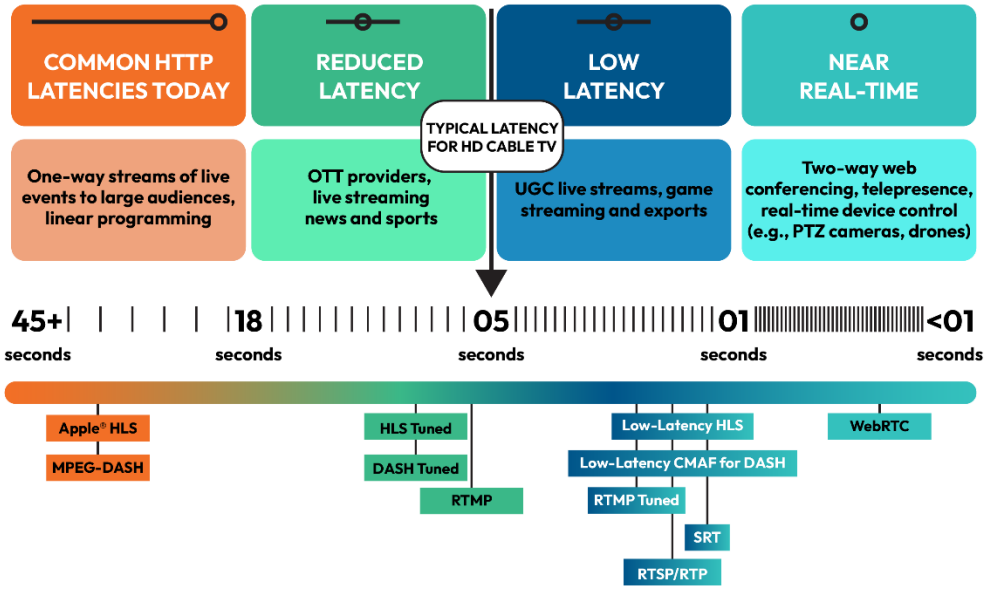
Chapter 1: Introducing Low Latency Application Development in C++

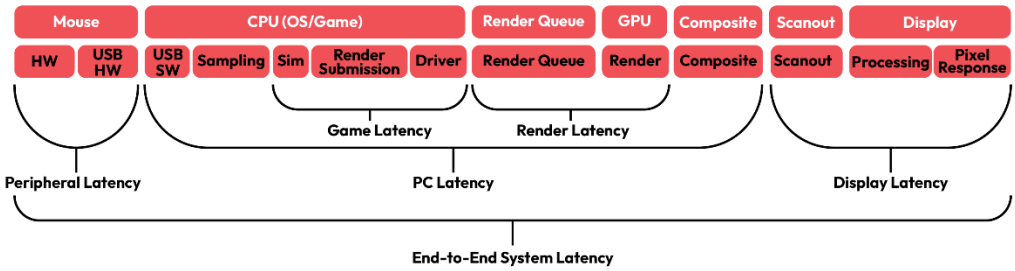


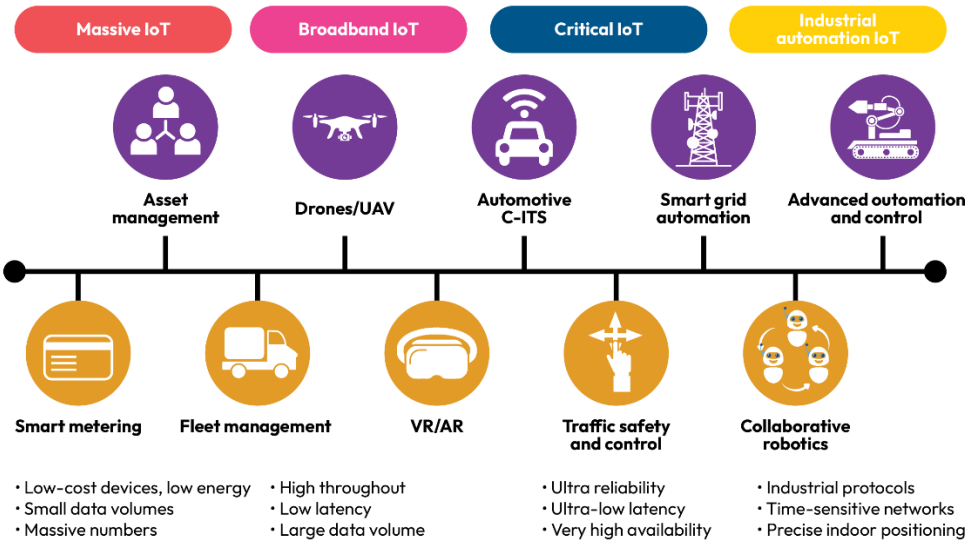
T_1: Time server puts first byte of packet on the wire.
T_2: Time client reads first byte of packet from the wire.
T_3: Time client puts a packet containing a response on the wire.
T_4: Time server reads first byte of packet containing the response from the wire.

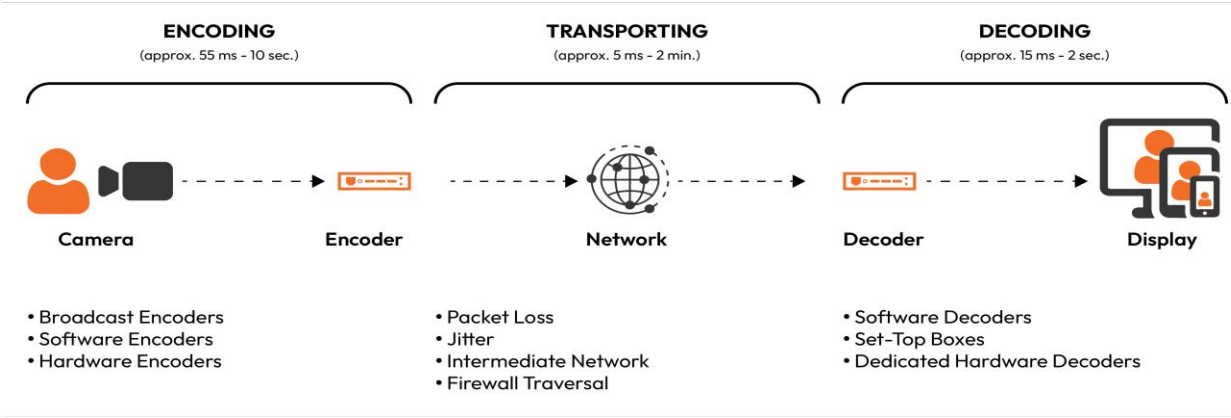


Chapter 2: Designing Some Common Low Latency Applications in C++

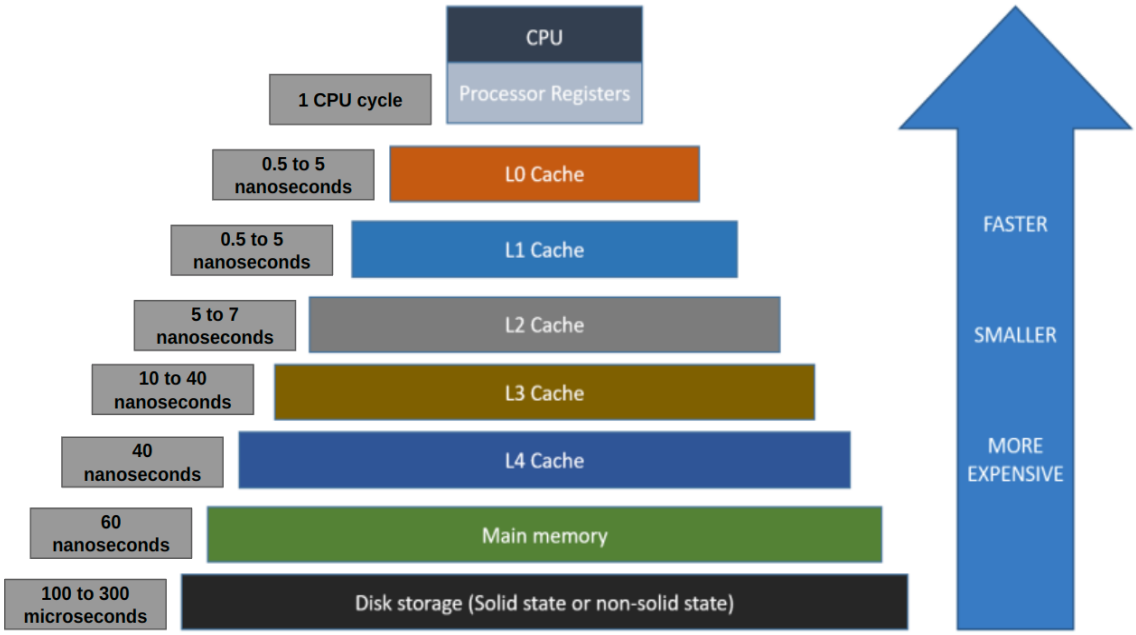





Massive IoT
Broadband IoT
Critical IoT
Industrial automation IoT
Asset management
Drones/UAV
Automotive C-ITS
Smart grid automation
Advanced automation and control
Smart metering
Fleet management
VR/AR
Traffic safety and control
Collaborative robotics
• Low-cost devices, low energy
• Small data volumes
• Massive numbers
• High throughout
• Low latency
• Large data volume
• Ultra reliability
• Ultra-low latency
• Very high availability
• Industrial protocols
• Time-sensitive networks
• Precise indoor positioning


ENCODING
(approx. 55 ms - 10 sec.)
TRANSPORTING
(approx. 5 ms - 2 min.)
DECODING
(approx. 15 ms - 2 sec.)
Camera
Encoder
Network
Decoder
Display
• Broadcast Encoders
• Software Encoders
• Hardware Encoders
• Packet Loss
• Jitter
• Intermediate Network
• Firewall Traversal
• Software Decoders
• Set-Top Boxes
• Dedicated Hardware Decoders

Chapter 3: Exploring C++ Concepts from A Low-Latency Application's Perspective



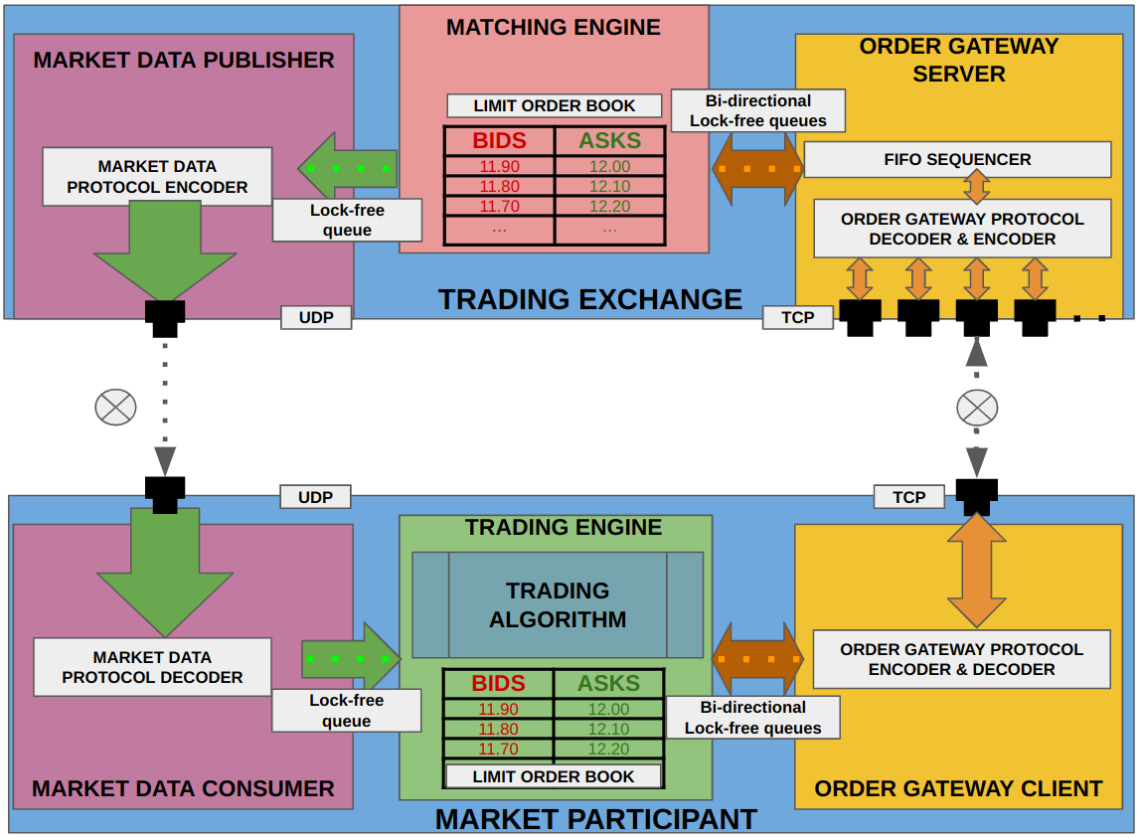
Chapter 4: Building the C++ Building Blocks for Low Latency Applications

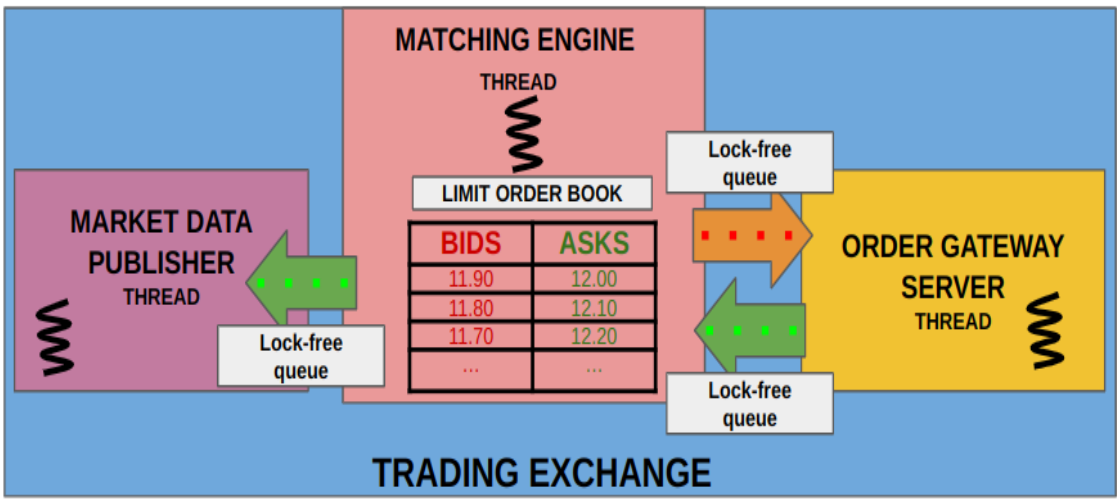
is_free_or_in_use_ vector Memory : 0x00AA00CC	
0	is_free_ = true;
1	is_free_ = false;
2	is_free_ = false;
3	is_free_ = false;
4	is_free_ = true;
…	
…	
N	is_free_ = false;

T object vector Memory: 0xFF00EEDD	
0	T object_;
1	T object_;
2	T object_;
3	T object_;
4	T object_;
…	
…	
N	T object_;

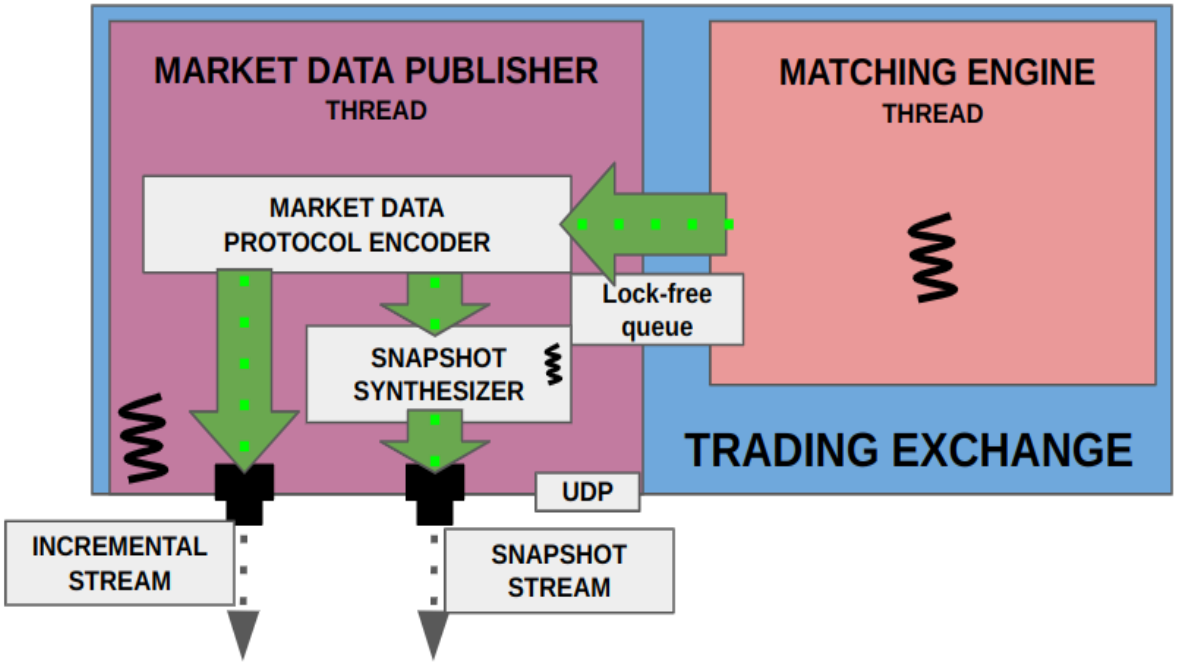
{is_free_or_in_use_, T object} vector Memory : 0x00AA00CC	
0	{is_free_ = true, T object_};
1	{is_free_ = false, T object_};
2	{is_free_ = false, T object_};
3	{is_free_ = false, T object_};
4	{is_free_ = true, T object_};
…	
…	
N	{is_free_ = false, T object_};

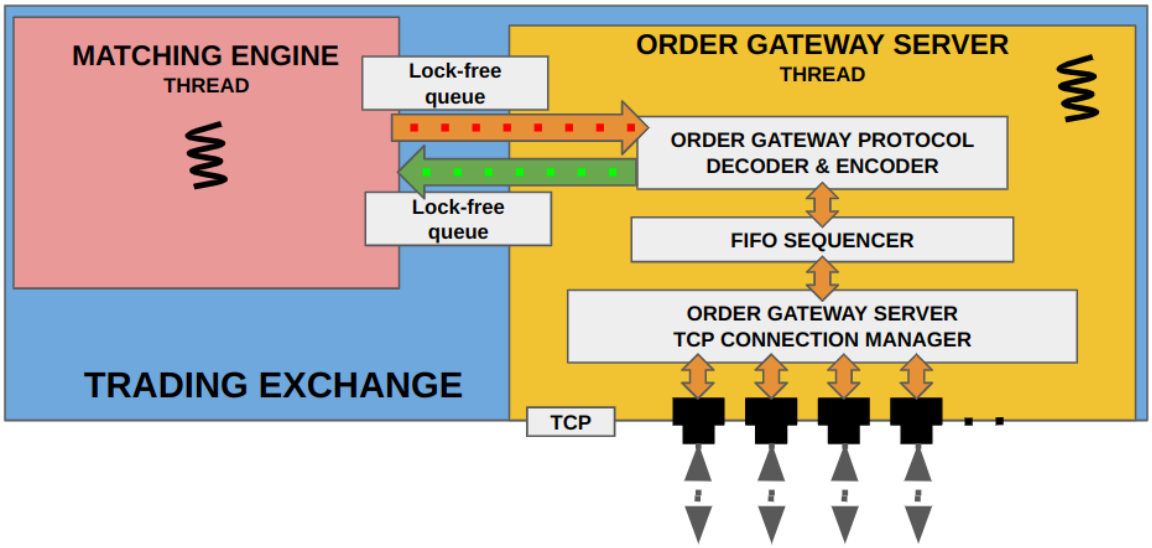
Chapter 5: Designing Our Trading Ecosystem

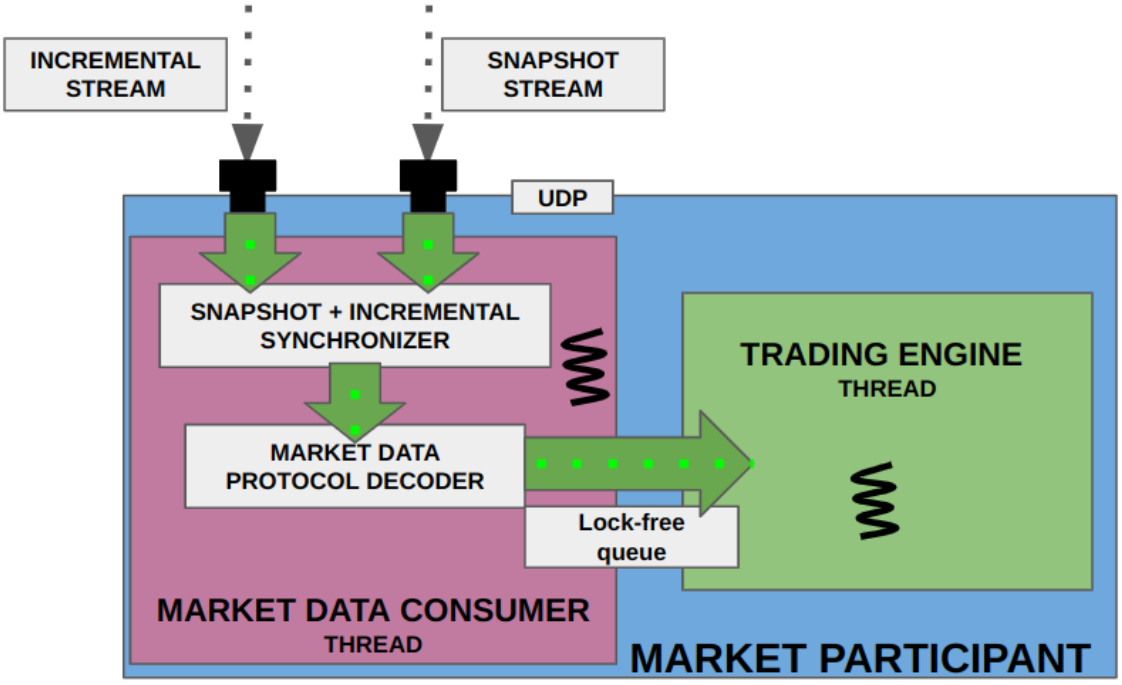


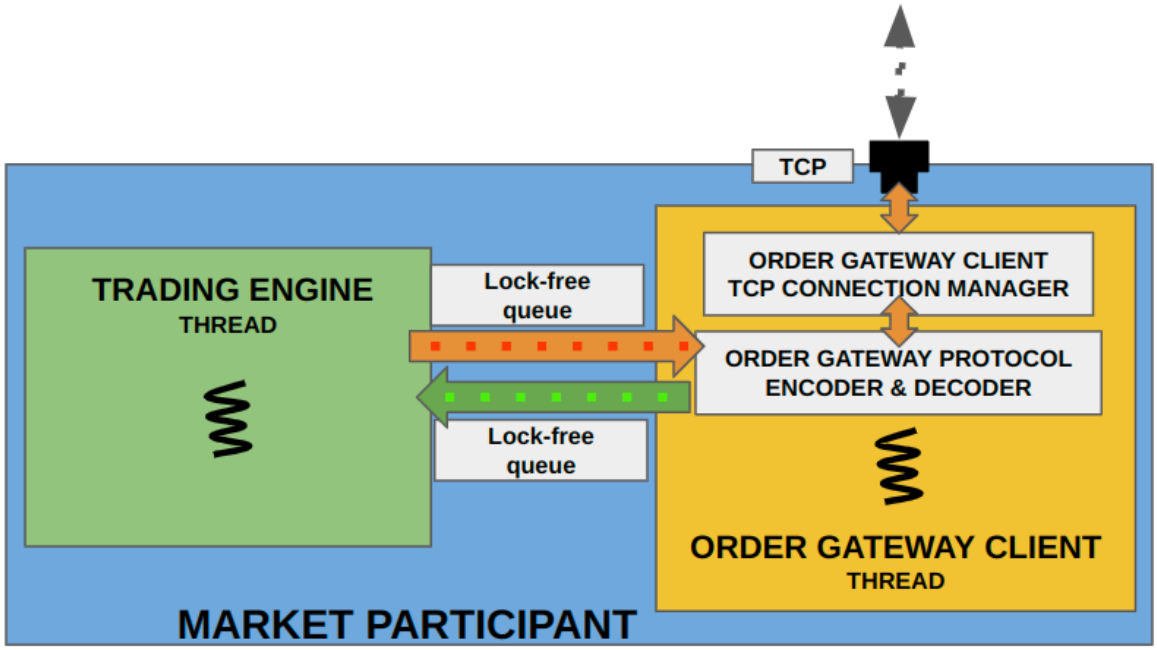



Market state changes
Instrument updates
Market statistics
MARKET UPDATES
Trade messages
Order updates
Order Add
Order Delete
Order Modify


MARKET DATA PUBLISHER
THREAD
MATCHING ENGINE
THREAD
MARKET DATA PROTOCOL ENCODER
Lock-free queue
SNAPSHOT SYNTHESIZER
TRADING EXCHANGE
UDP
INCREMENTAL STREAM
SNAPSHOT STREAM

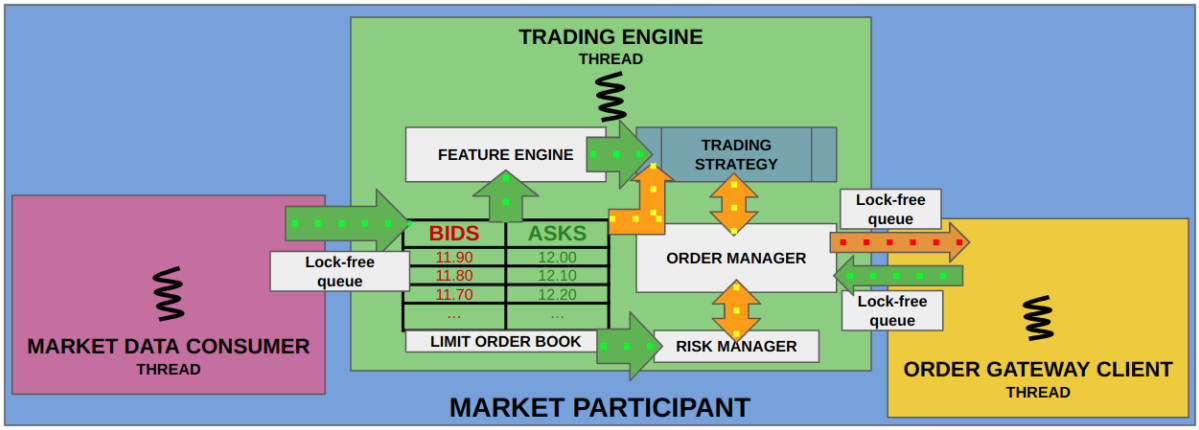

MATCHING ENGINE
THREAD
Lock-free
queue
ORDER GATEWAY SERVER
THREAD
ORDER GATEWAY PROTOCOL
DECODER & ENCODER
Lock-free
queue
FIFO SEQUENCER
ORDER GATEWAY SERVER
TCP CONNECTION MANAGER
TRADING EXCHANGE
TCP


INCREMENTAL
STREAM
SNAPSHOT
STREAM
UDP
SNAPSHOT + INCREMENTAL
SYNCHRONIZER
TRADING ENGINE
THREAD
MARKET DATA
PROTOCOL DECODER
Lock-free
queue
MARKET DATA CONSUMER
THREAD
MARKET PARTICIPANT

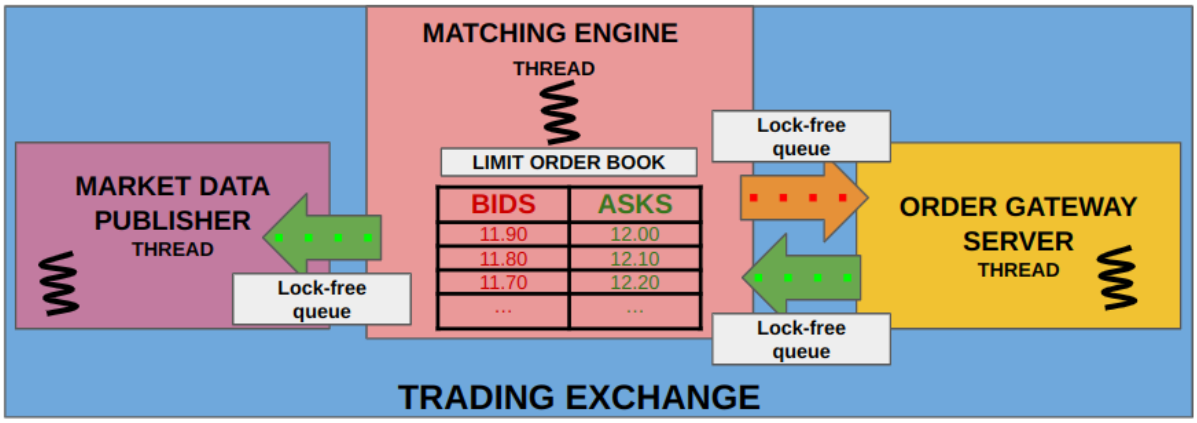


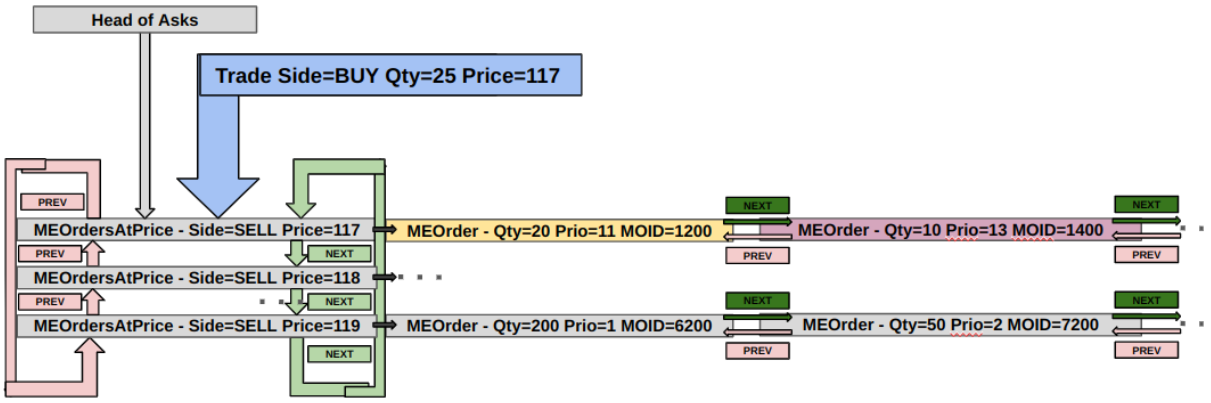
RISK MANAGEMENT

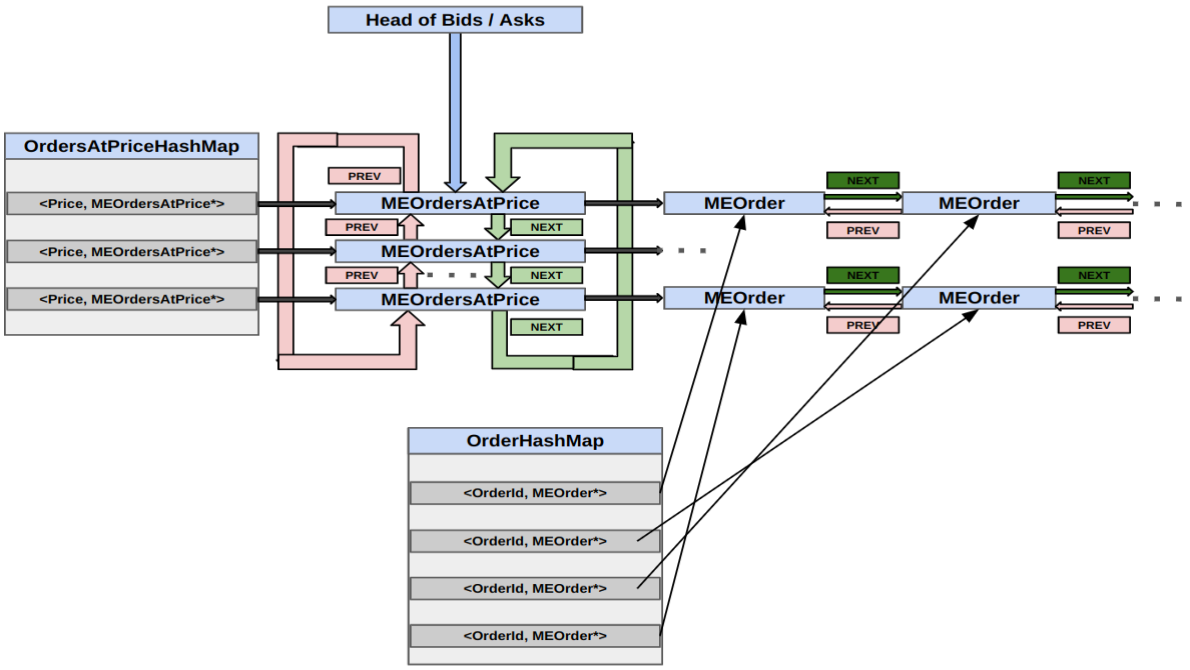
- Maximum order quantity.
- Maximum position for a desk / firm.
- Worst case position for a desk / firm.
- Maximum realized and unrealized losses.
- Maximum traded volumes for an algorithm.
- Rate-based risk measures.
  - Number of orders sent.
  - Number of trades executed.
  - Realized losses in a time period.


TRADING ENGINE
THREAD
FEATURE ENGINE
TRADING STRATEGY
BIDS
ASKS
11.90
11.80
11.70
...
12.00
12.10
12.20
...
LIMIT ORDER BOOK
ORDER MANAGER
RISK MANAGER
Lock-free queue
Lock-free queue
Lock-free queue
MARKET DATA CONSUMER
THREAD
ORDER GATEWAY CLIENT
THREAD
MARKET PARTICIPANT

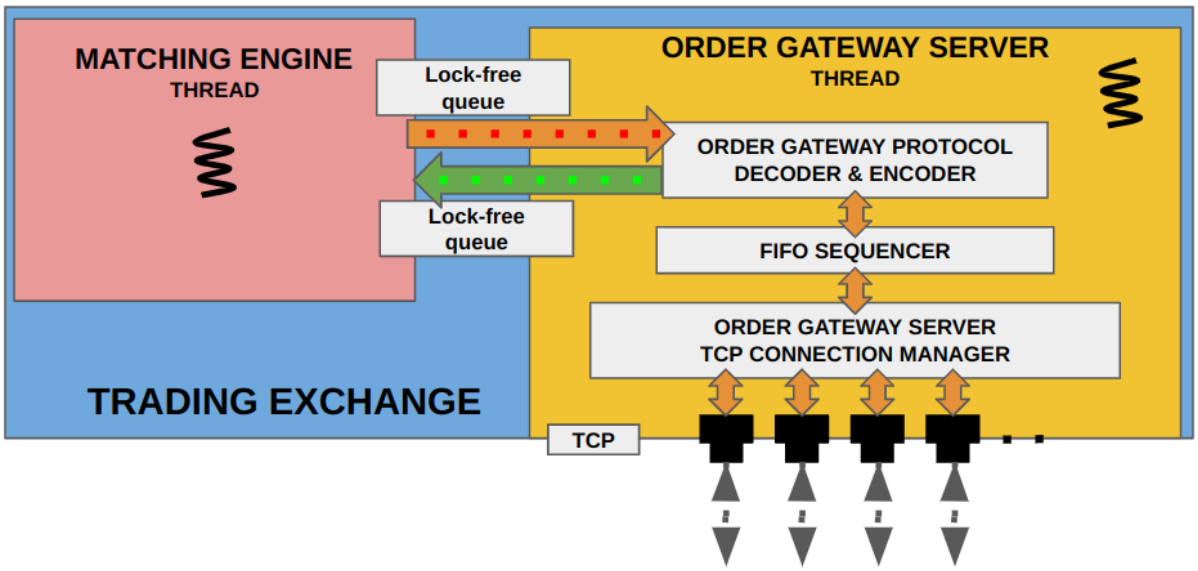
Chapter 6: Building the C++ Matching Engine

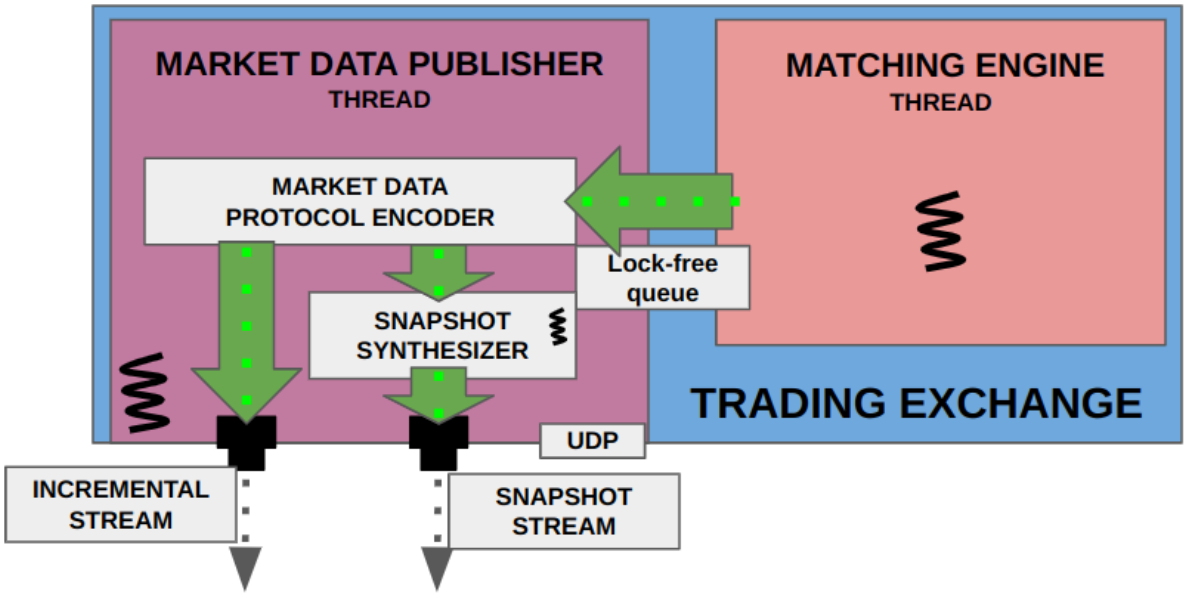


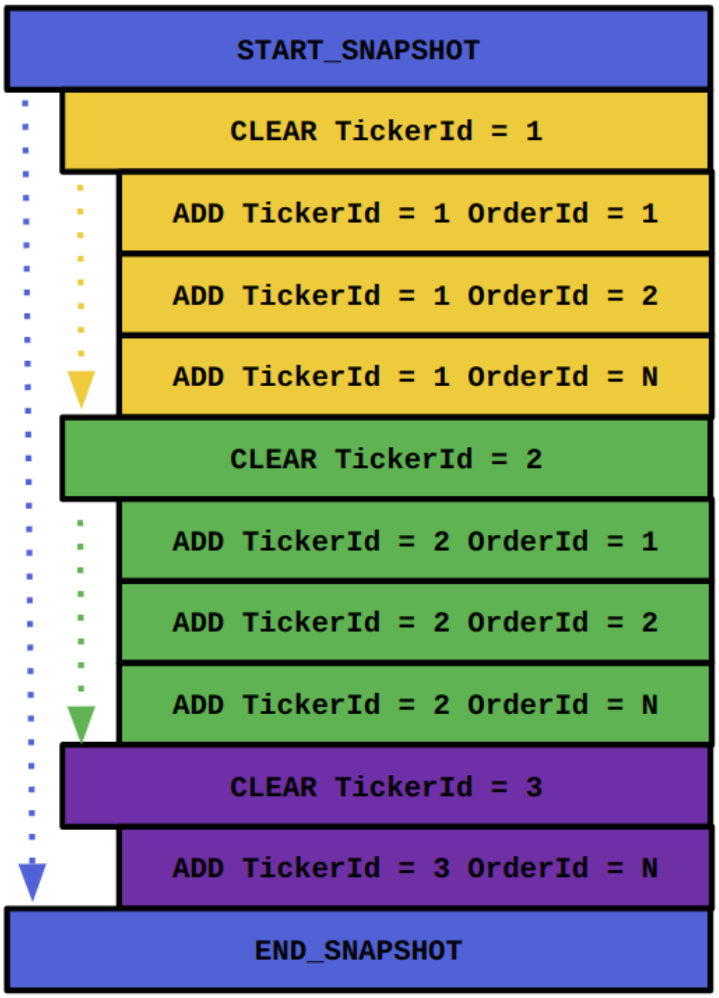


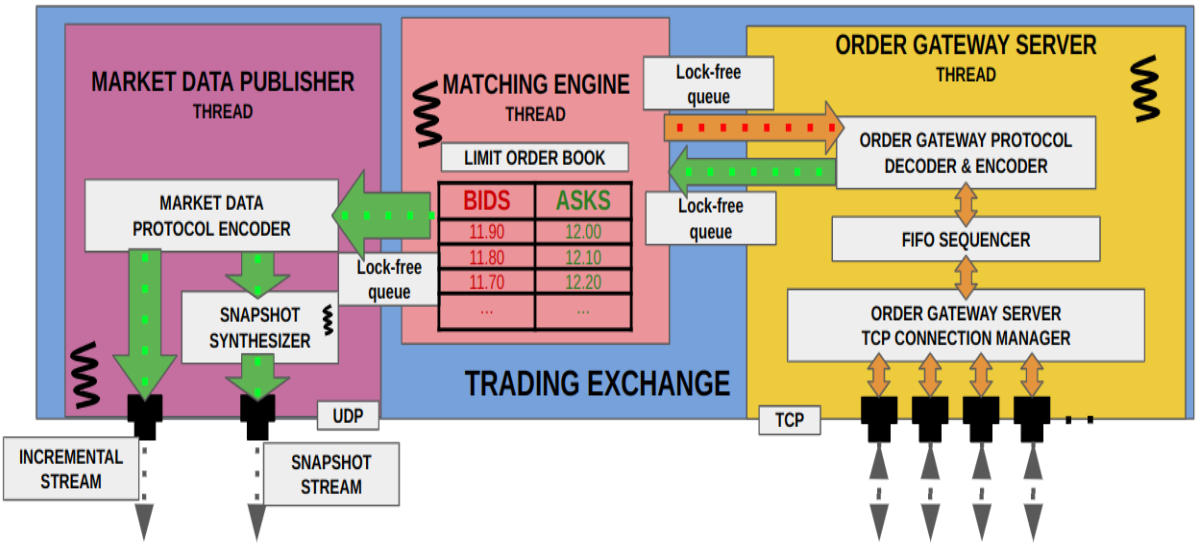

Head of Bids / Asks
OrdersAtPriceHashMap
<Price, MEOrdersAtPrice*>
<Price, MEOrdersAtPrice*>
<Price, MEOrdersAtPrice*>
PREV
MEOrdersAtPrice
NEXT
PREV
MEOrdersAtPrice
NEXT
PREV
MEOrdersAtPrice
NEXT
MEOrder
NEXT
PREV
MEOrder
NEXT
PREV
MEOrder
NEXT
PREV
MEOrder
NEXT
PREV
OrderHashMap
<OrderId, MEOrder*>
<OrderId, MEOrder*>
<OrderId, MEOrder*>
<OrderId, MEOrder*>

Chapter 7: Communicating with Market Participants

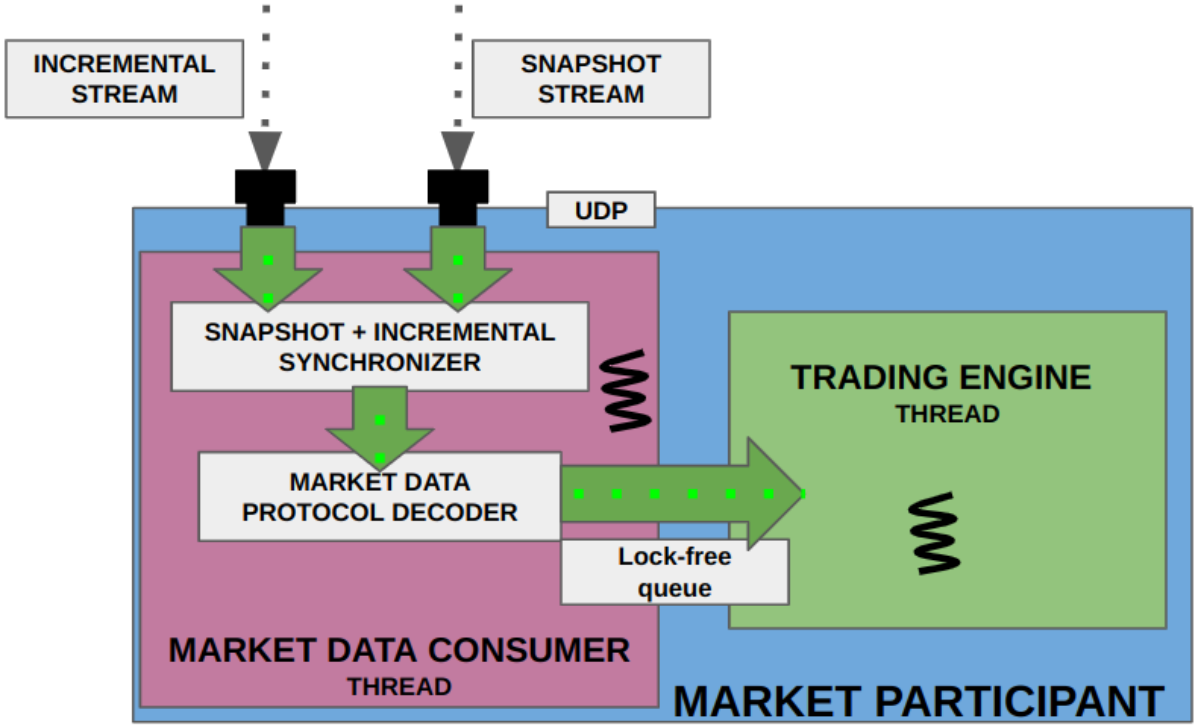


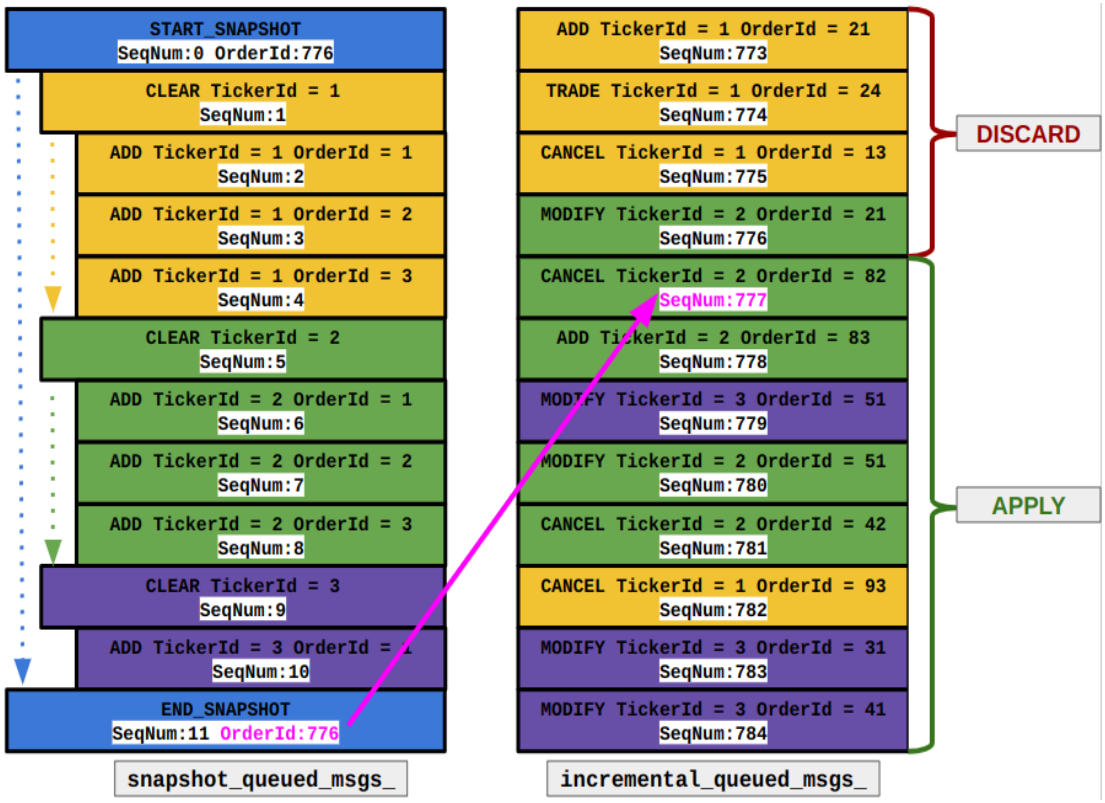


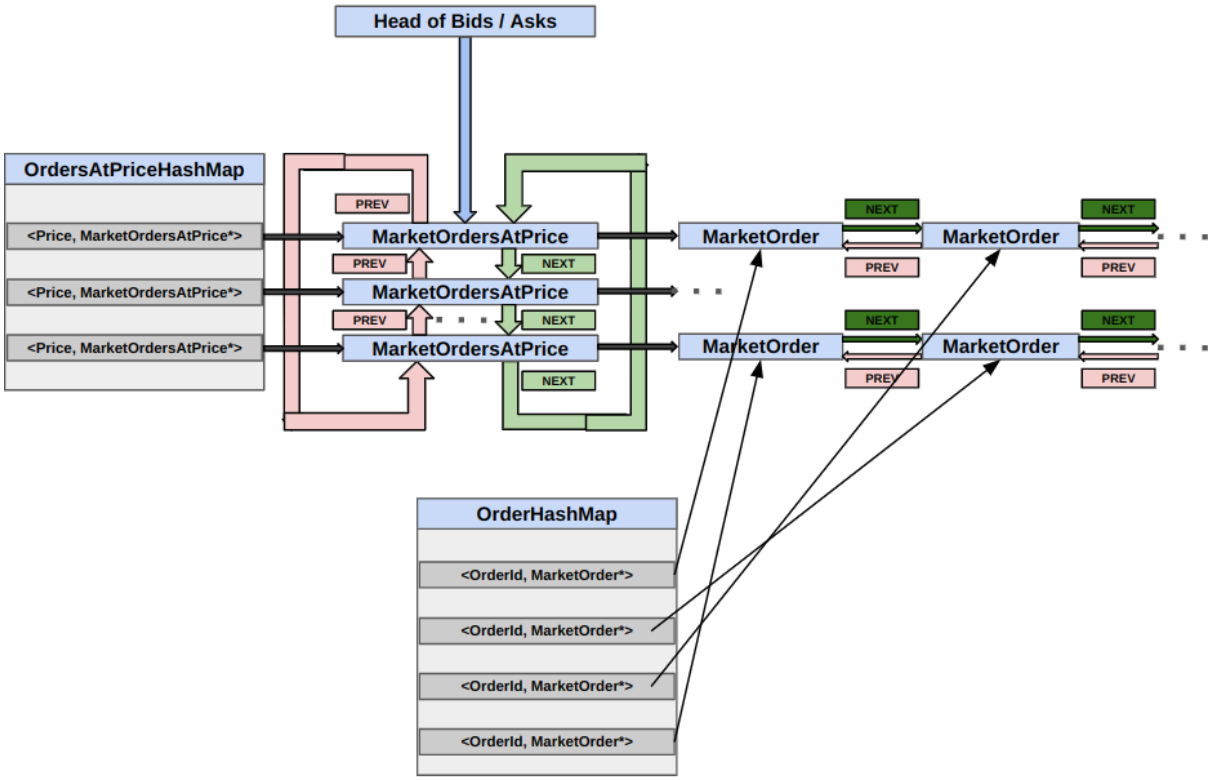

START_SNAPSHOT
CLEAR TickerId = 1
ADD TickerId = 1 OrderId = 1
ADD TickerId = 1 OrderId = 2
ADD TickerId = 1 OrderId = N
CLEAR TickerId = 2
ADD TickerId = 2 OrderId = 1
ADD TickerId = 2 OrderId = 2
ADD TickerId = 2 OrderId = N
CLEAR TickerId = 3
ADD TickerId = 3 OrderId = N
END_SNAPSHOT

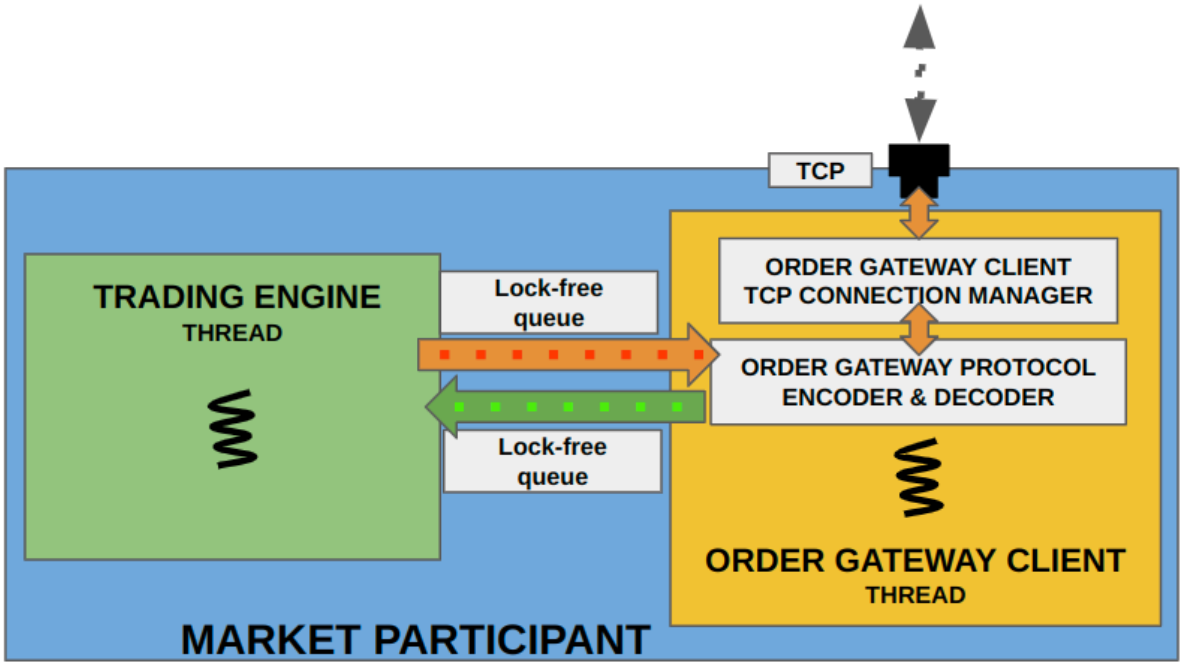

MARKET DATA PUBLISHER
THREAD
MARKET DATA
PROTOCOL ENCODER
SNAPSHOT
SYNTHESIZER
UDP
INCREMENTAL
STREAM
SNAPSHOT
STREAM
MATCHING ENGINE
THREAD
LIMIT ORDER BOOK
BIDS
ASKS
11.90
12.00
11.80
12.10
11.70
12.20
Lock-free
queue
Lock-free
queue
Lock-free
queue
TRADING EXCHANGE
ORDER GATEWAY SERVER
THREAD
ORDER GATEWAY PROTOCOL
DECODER & ENCODER
FIFO SEQUENCER
ORDER GATEWAY SERVER
TCP CONNECTION MANAGER
TCP

Chapter 8: Processing Market Data and Sending Orders to the Exchange in C++

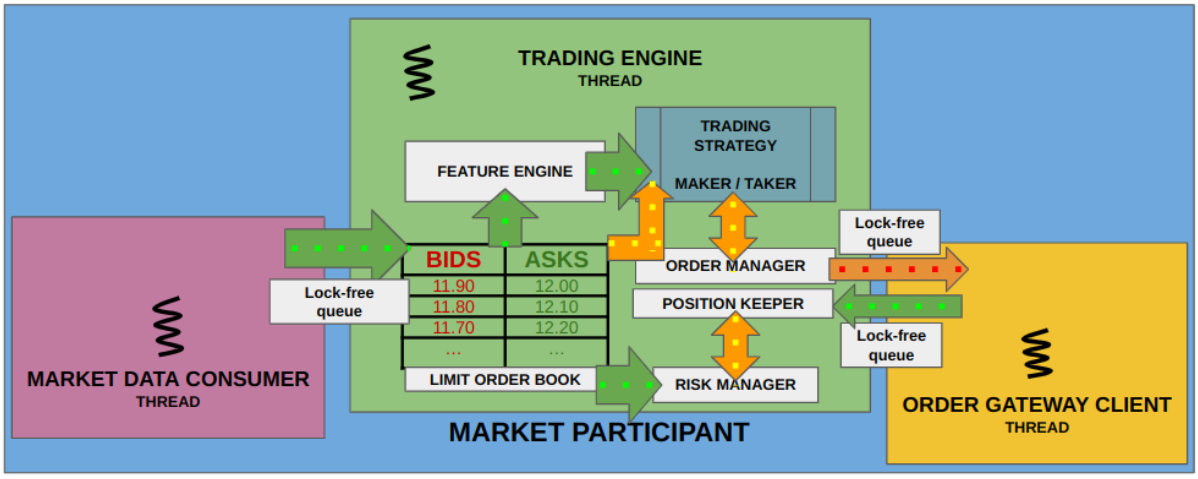





Head of Bids / Asks
OrdersAtPriceHashMap
<Price, MarketOrdersAtPrice*>
<Price, MarketOrdersAtPrice*>
<Price, MarketOrdersAtPrice*>
PREV
NEXT
MarketOrdersAtPrice
MarketOrdersAtPrice
MarketOrdersAtPrice
MarketOrder
MarketOrder
MarketOrder
MarketOrder
OrderHashMap
<OrderId, MarketOrder*>
<OrderId, MarketOrder*>
<OrderId, MarketOrder*>
<OrderId, MarketOrder*>


TCP
TRADING ENGINE
THREAD
Lock-free queue
Lock-free queue
ORDER GATEWAY CLIENT
TCP CONNECTION MANAGER
ORDER GATEWAY PROTOCOL
ENCODER & DECODER
ORDER GATEWAY CLIENT
THREAD
MARKET PARTICIPANT

Chapter 9: Building the C++ Trading Algorithm's Building Blocks



Chapter 10: Building the C++ Market Making and Liquidity Taking Algorithms

Our MM strategy bid	Market bid order count	Market bid quantity	Market bid price	Market ask price	Market ask quantity	Market ask order count	Our MM strategy ask
	12	2500	10.21	10.22	4500	8	
	25	4700	10.20	10.23	5500	18	

Our MM strategy bid	Market bid order count	Market bid quantity	Market bid price	Market ask price	Market ask quantity	Market ask order count	Our MM strategy ask
100	13	2600	10.21	10.22	4600	9	100
	25	4700	10.20	10.23	5500	18	

Our MM strategy bid	Market bid order count	Market bid quantity	Market bid price	Market ask price	Market ask quantity	Market ask order count	Our MM strategy ask
100	3	600	10.21	10.22	6500	11	100
	26	4800	10.20	10.23	5500	18	

Our MM strategy bid	Market bid order count	Market bid quantity	Market bid price	Market ask price	Market ask quantity	Market ask order count	Our MM strategy ask
	2	500	10.21	10.22	6500	11	100
100	26	4800	10.20	10.23	5500	18	

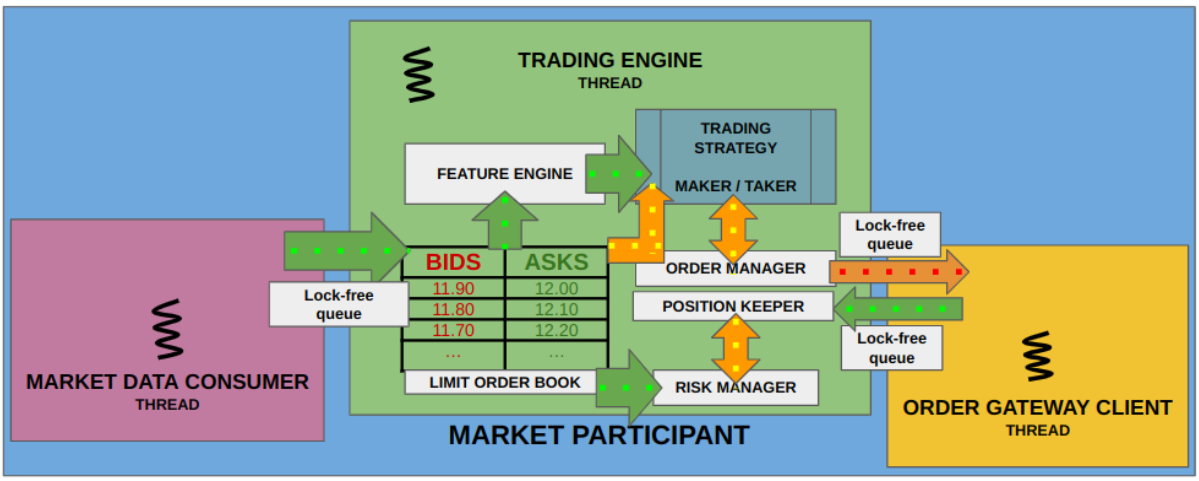
Market bid order count	Market bid quantity	Market bid price	Market ask price	Market ask quantity	Market ask order count
12	2500	10.21	10.22	4500	8
25	4700	10.20	10.23	5500	18

Market bid order count	Market bid quantity	Market bid price	Market ask price	Market ask quantity	Market ask order count
12	2500	10.21	10.22	4500	8
25	4700	10.20	10.23	5500	18

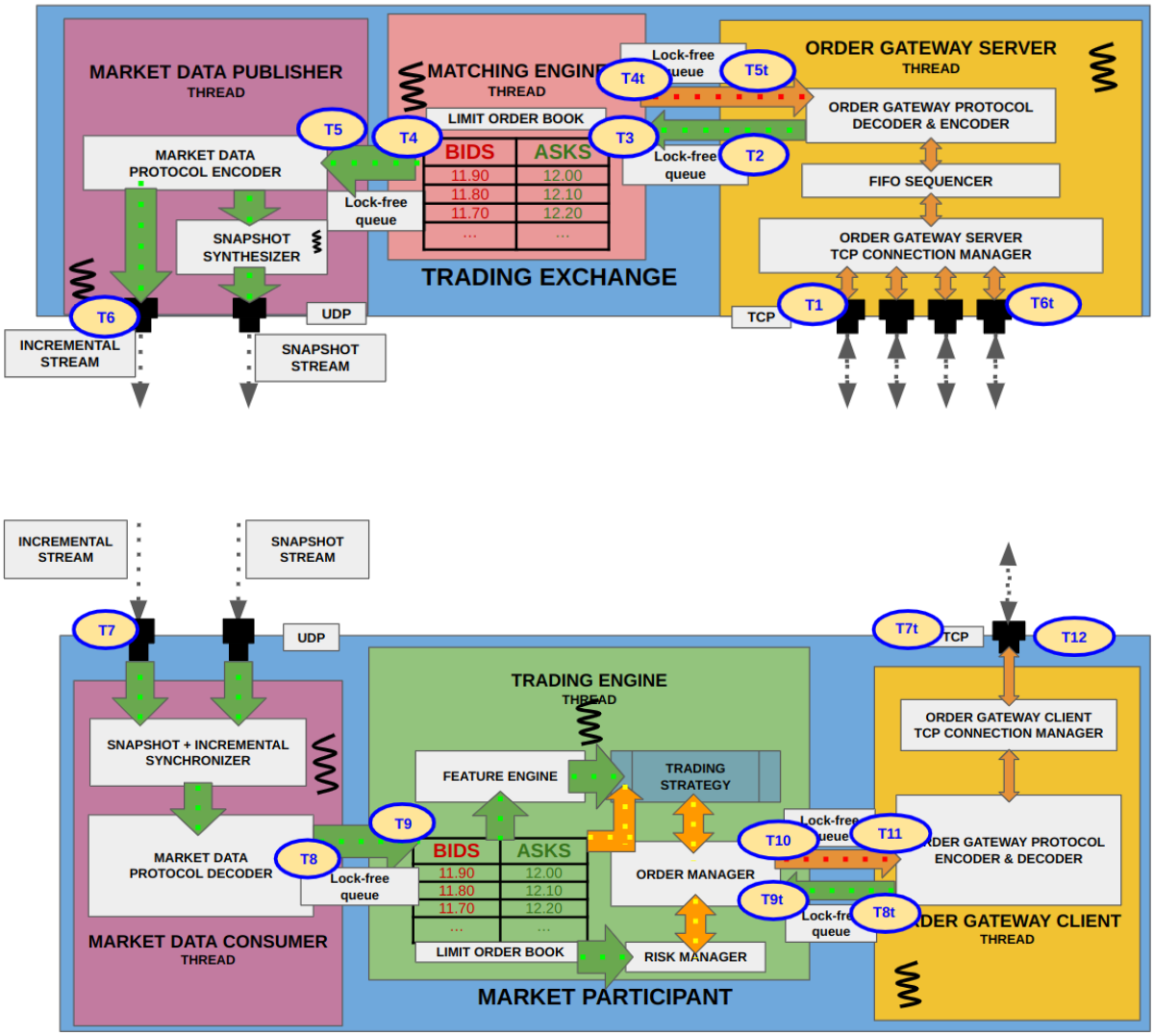
Sell 2200@10.21

Market bid order count	Market bid quantity	Market bid price	Market ask price	Market ask quantity	Market ask order count
2	300	10.21	10.22	4500	8
25	4700	10.20	10.23	5500	18

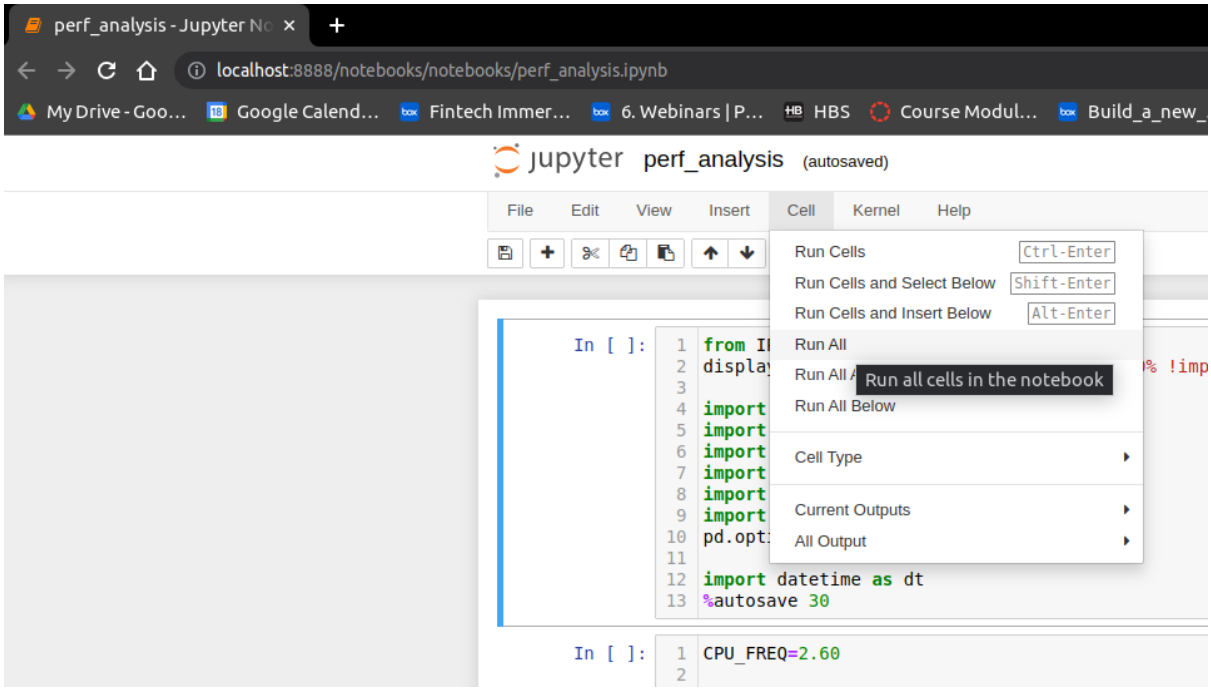
LT Sell 100@10.21

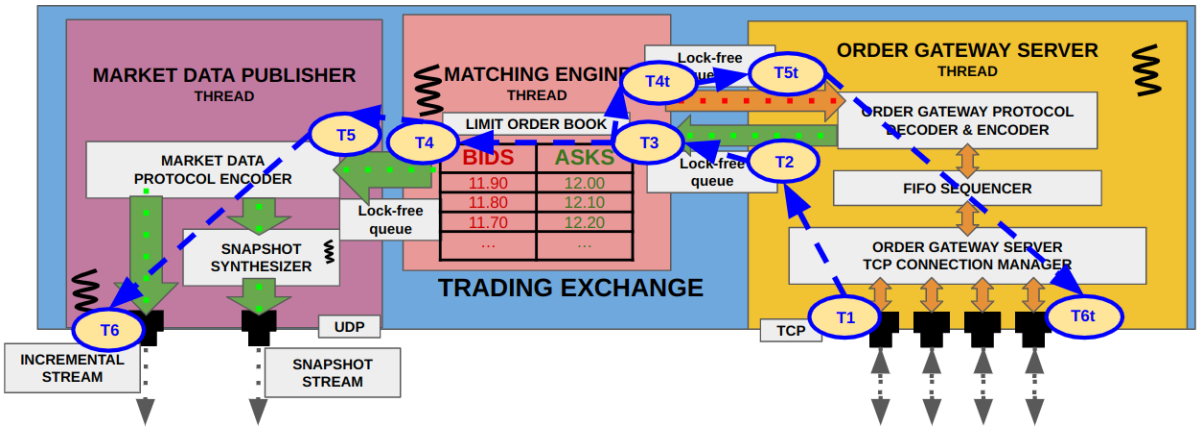


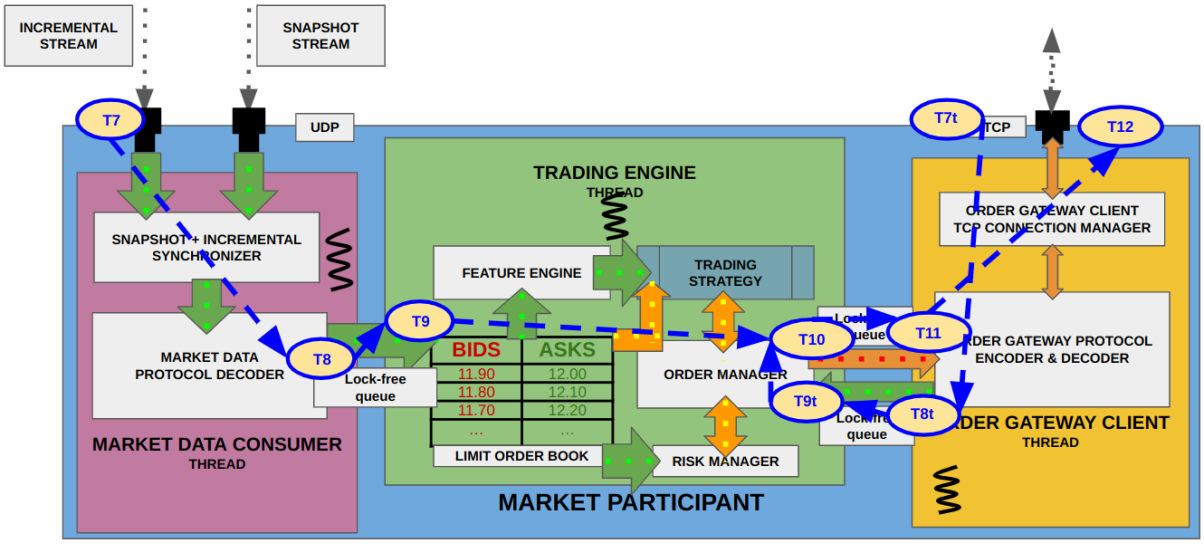
Chapter 11: Adding Instrumentation and Measuring Performance



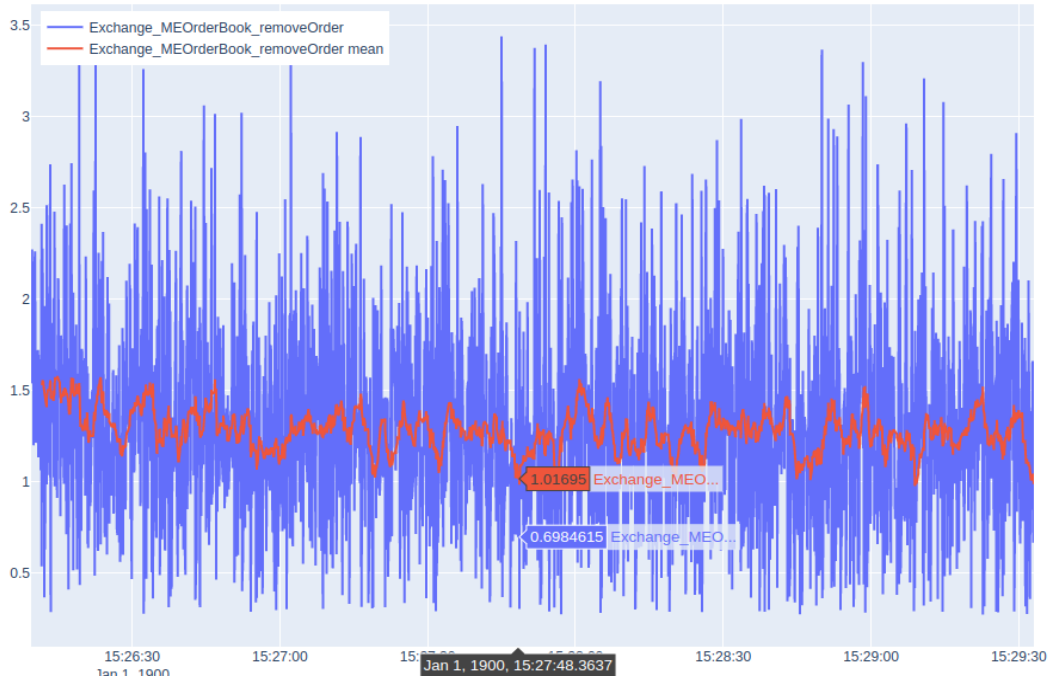
Chapter 12: Analyzing and Optimizing the Performance of Our C++ System



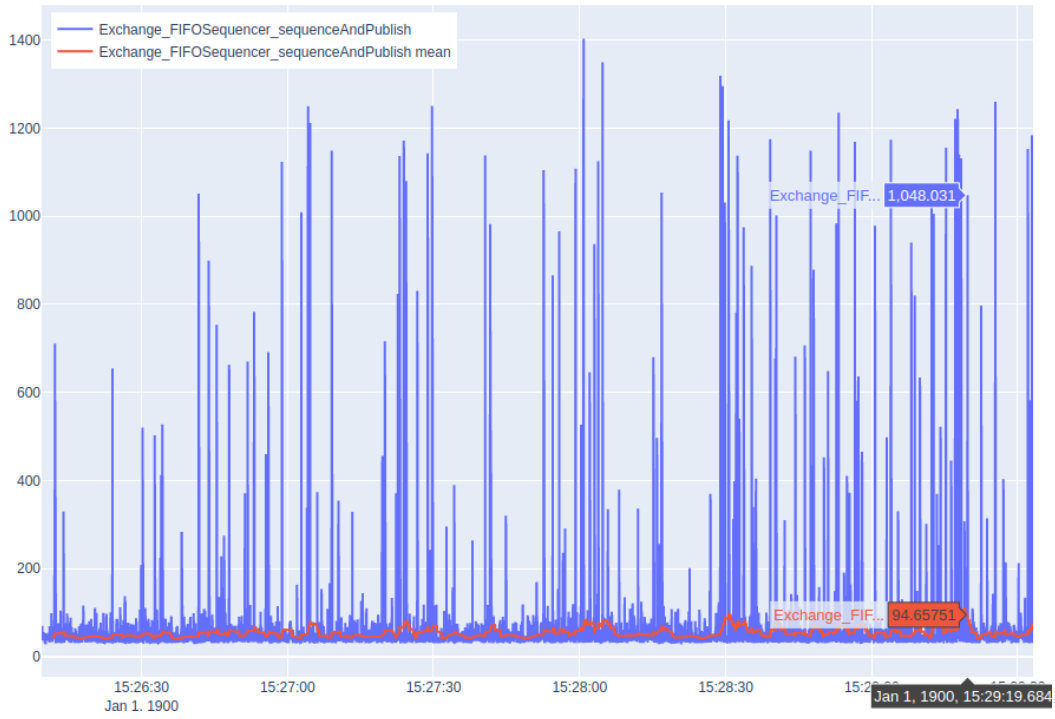




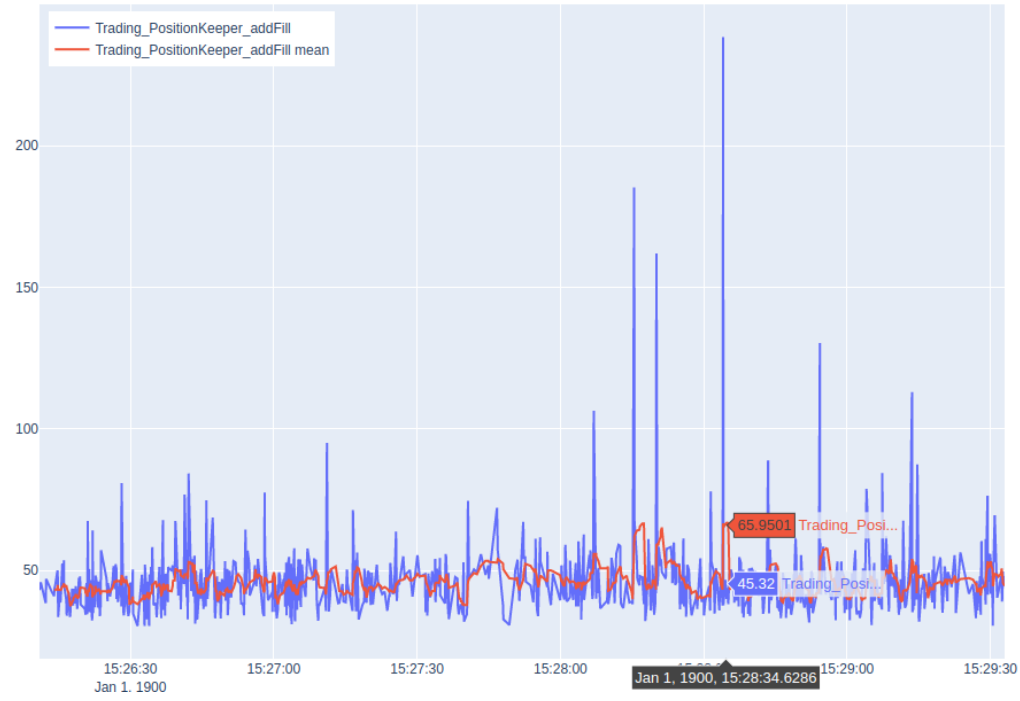
performance Exchange_MEOrderBook_removeOrder microseconds



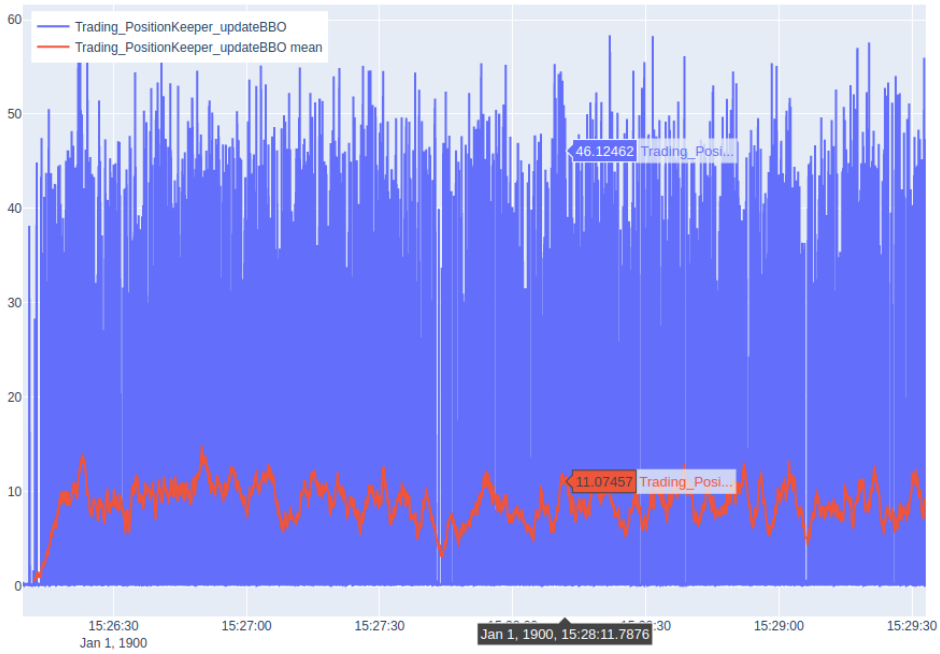
performance Exchange_FIFOSequencer_sequenceAndPublish microseconds



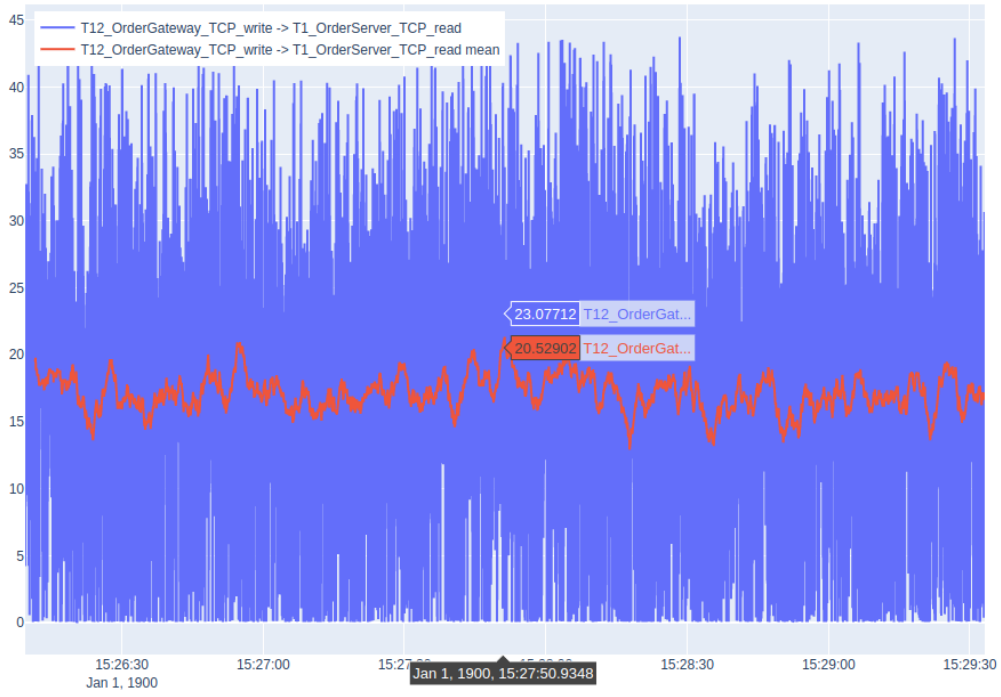
performance Trading_PositionKeeper_addFill microseconds



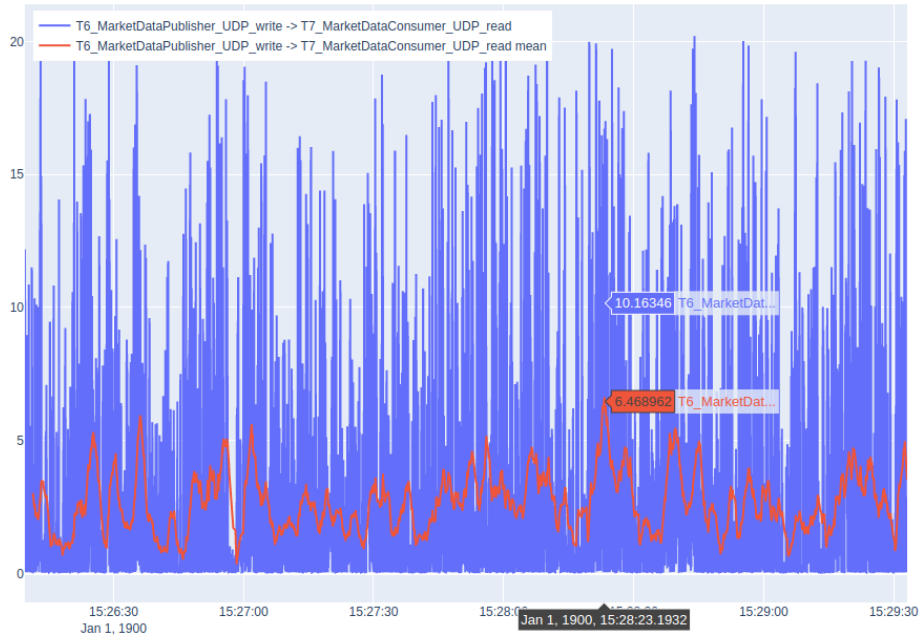
performance Trading_PositionKeeper_updateBBO microseconds



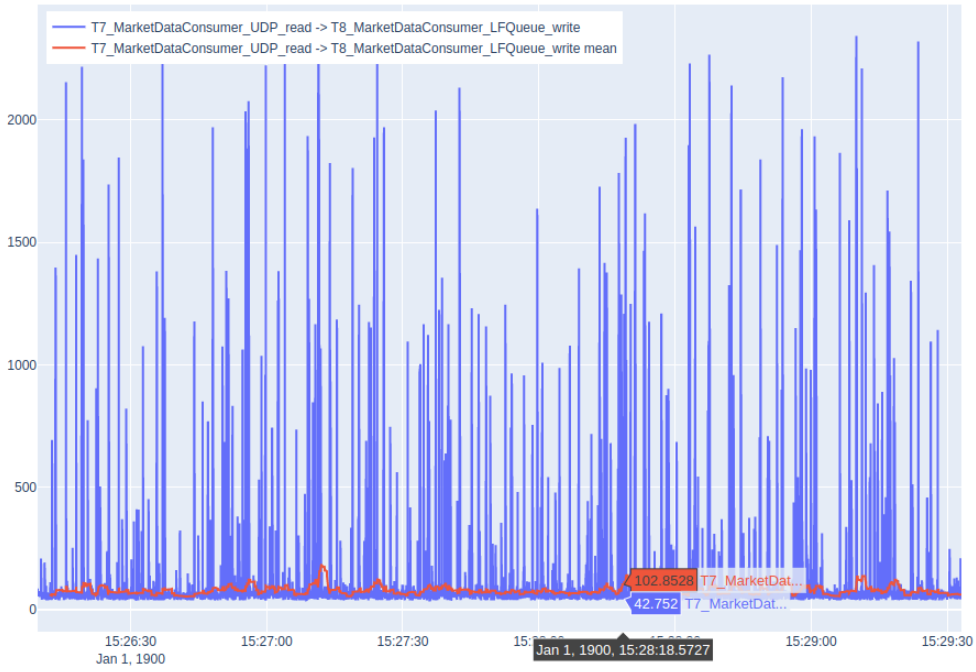
performance T12_OrderGateway_TCP_write -> T1_OrderServer_TCP_read milliseconds



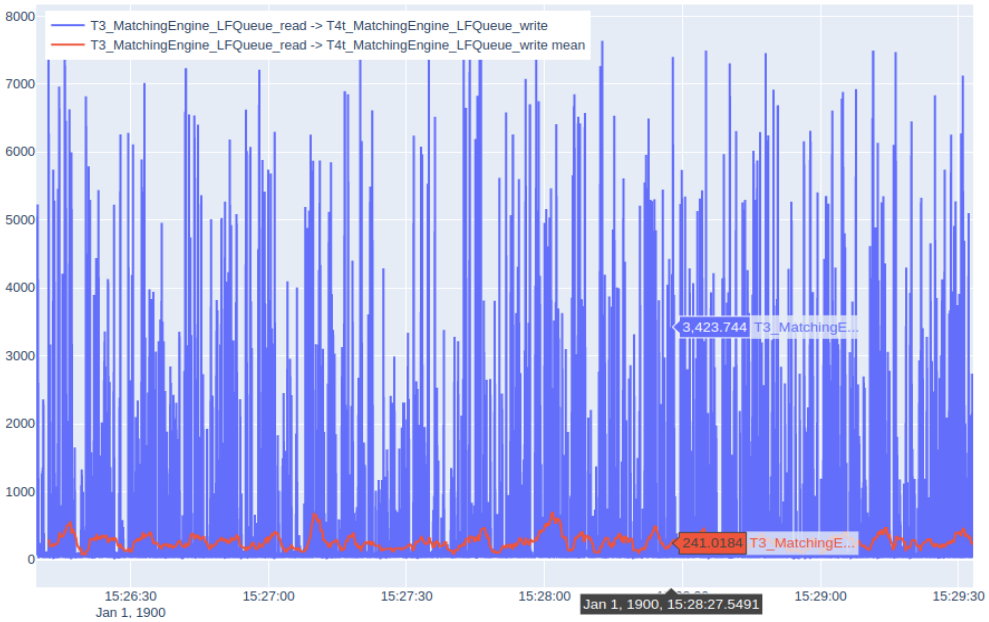
performance T6_MarketDataPublisher_UDP_write -> T7_MarketDataConsumer_UDP_read milliseconds

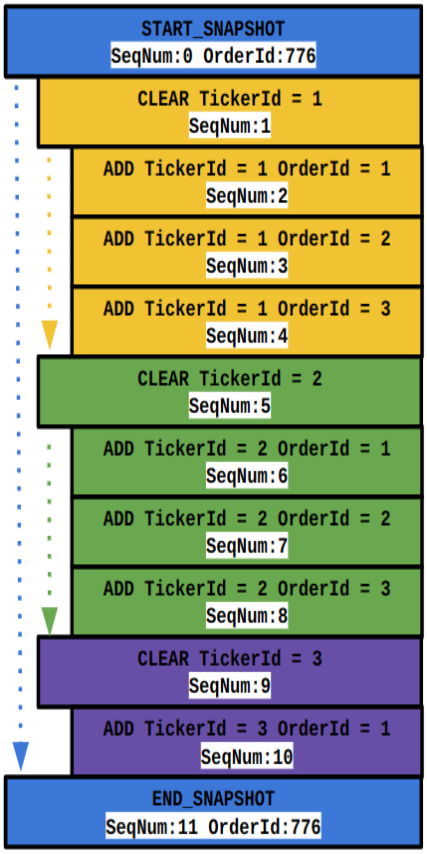


performance T7_MarketDataConsumer_UDP_read -> T8_MarketDataConsumer_LFQueue_write microseconds

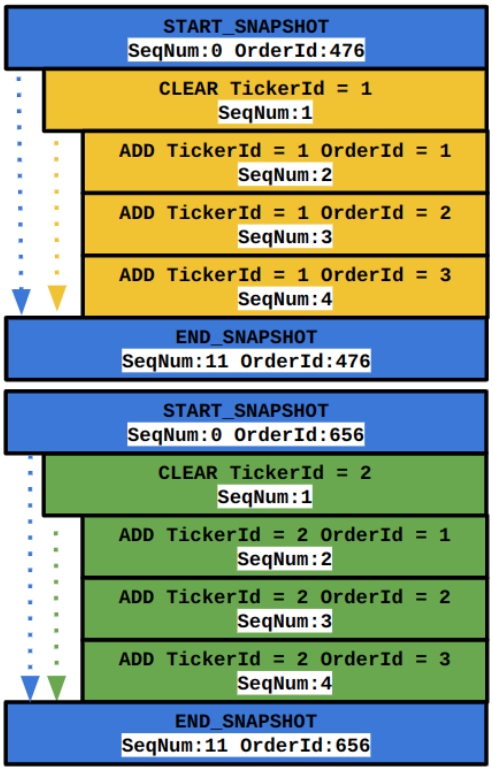


performance T3_MatchingEngine_LFQueue_read -> T4t_MatchingEngine_LFQueue_write microseconds





T = 0 secs, 60 secs, 120 secs, 180 secs, ...



T = 0 secs, 60 secs, 120 secs, 180 secs, ...

T = 10 secs, 70 secs, 130 secs, 190 secs, ...