

We examine Solace Systems' high-performance messaging middleware appliance to see how it supports low latency access to execution venues, high-throughput internal distribution, and external customer connectivity.

Middleware: Rise of the Appliances

Foreign exchange trading has become a major source of income for many market participants over the past decade. In 2010, the Bank for International Settlements estimated the average daily turnover of the foreign exchange (FX) market to be four trillion dollars, which dwarfs equity markets by comparison. Since FX trading is mostly a byproduct of global trade, it is not directly dependent on investment cycles like the equity and fixed income markets.

While broker-dealers serve as the primary financial intermediaries for listed equities and derivatives, banks dominate the FX market. Corporate and institutional customers provide banks with a steady source of FX trading, especially in Asia, which has many currencies and large export markets.

The FX trading environment has become increasingly complicated, with liquidity available from banks internal sources, electronic communication networks (ECNs) such as Currenex, and other financial institutions' electronic trading platforms. In turn, banks have had to adapt their trading systems to take advantage of liquidity provided by many different sources.

Foreign exchange has historically provided strong returns for banks in Asia. Competition has increased, however, as more participants have entered the market and trading technology has become more sophisticated. It has become increasingly important for banks to have a high-performance messaging middleware platform to remain competitive.

Hence, connectivity to external FX trading venues, aggregation of price feeds, and high-speed calculation and distribution of rates have become critical requirements for electronic trading systems.

A new breed of customers has also entered the market: hedge funds and high-frequency trading firms have embraced FX trading to go beyond saturated equity markets and implement multi-asset trading strategies. While providing a ready source of trading demand, these customers have made high-speed FX rate distribution and high-throughput order routing a major concern.

Latency has become an important consideration for servicing customers focused on automated trading. When delivery of FX rates to customers is delayed, a bank's prices may become uncompetitive. Worse still, a bank may irk customers by rejecting their orders because the delivery of tradable rates or order processing is too slow.

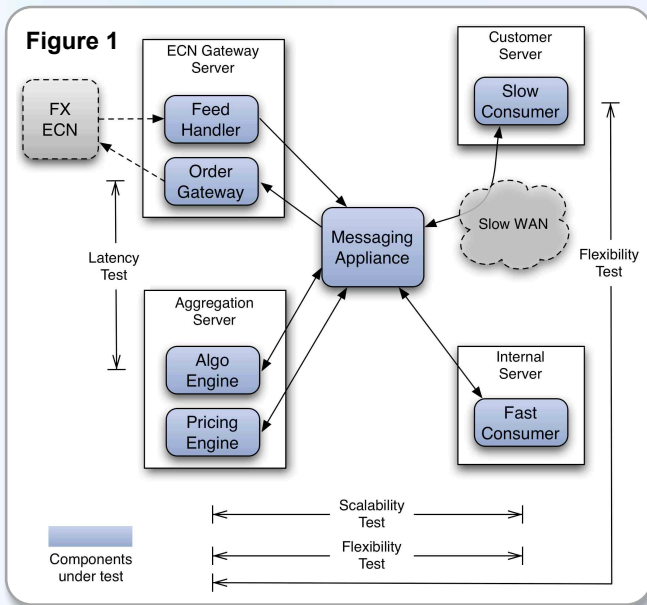
In this environment, messaging middleware has, more than ever, become a strategic technology platform. The capabilities provided by middleware technology can enable or limit the business application strategy.

The diversity challenge

The ability to accommodate a wide range of customer needs is a critical requirement for middleware platforms that support FX trading. Within the bank, trading systems require low inter-process latency so that liquidity from different sources can be quickly aggregated into a single view and traded before other market participants take up the liquidity. Conversely, other divisions – such as wealth management and commercial banking groups – will be more concerned with ease of access and integration.

The needs of external customers are also diverse. Small corporate customers may trade using a web-based user interface provided by the bank, while large corporate customers require application programming interfaces (APIs) to support integration with their treasury systems. Some corporate systems and networks may not be able to cope with high-speed FX rate distribution. At the same time other high frequency trading customers will demand this capability.

Streaming tradable FX rates to large numbers of customers is a major technological challenge. Computing FX rates for many combinations of cross-rates, tenors, and strike prices can create an explosion of data that overwhelms both internal and external systems. Furthermore, generating margined price streams for specific customer groups increases the amount of real-time rate data that must be managed. Advanced trading platforms may send different FX rates to clients



over hundreds of thousands of unique message distribution streams. In contrast, equity trading typically involves far fewer price streams and many more consumers for each stream. Only a limited number of stocks are traded, and the same price update can be distributed to all customers monitoring a stock's price.

Architecting and implementing an FX distribution platform that can address all of these requirements is a major undertaking. For equity trading, a common model has been to use multiple messaging products: one for performance-sensitive direct market access, another for general trading system connectivity, and possibly a third for market data distribution. Such multi-platform designs are costly to implement and maintain. Ideally, a single middleware distribution platform would be used to minimize cost and design complexity while addressing all of the key technical requirements. Fortunately, advances in hardware-based message distribution are beginning to make this vision possible.

Next-generation middleware

For the past twenty years, the prevalent design pattern for trading systems has been for applications to communicate with one another using messaging software that employs reliable broadcast protocols. Reliable broadcast data transmission facilities were originally developed on top of the User Datagram Protocol (UDP) to work around the limitations of connection-oriented point-to-point data distribution. Software-based TCP point-to-point delivery was unable to scale efficiently and led to delivery fairness concerns; the last recipient of a message could potentially receive it significantly later than the first recipient.

UDP broadcast was more efficient for distribution from one sender to many receivers, as is often the case with market data. However, UDP's major drawback was that it put extra load on subscribing applications' servers, requiring them to filter out unwanted messages. Multicast UDP distribution improved on this by allowing finer segmentation of data stream groups, thus reducing this drawback, but not eliminating it.

This concern is very relevant for FX, where prices may be generated individually for each customer, and each application may access a distinct subset of the instruments available. Since a limited number of customers receive the same information, a large number of multicast groups must be managed. Multicast distribution is also inflexible; data streams have to be assigned to specific multicast groups in advance, and it is not a simple task to partition large tens or even hundreds of thousands unique data streams into multicast groups.

Since reliable broadcast protocols were introduced, networking hardware has advanced, eliminating the network performance and efficiency concerns that led to the development of UDP-based distribution protocols. Switched networks provide dedicated bandwidth to each server endpoint, ensuring that traffic does not impact other servers on the same network segment. Likewise, custom-designed hardware appliances have overcome many of the limitations of software-based middleware. These devices are more

like network routers than servers, using field programmable gate array (FPGA) and network processor (NP) technologies. Implementing filtering and distribution functions in silicon rather than software can avoid the limitations and bottlenecks associated with centralized distribution server architectures. Hence, improved network and middleware appliance technology has made point-to-point TCP-based distribution efficient and a practical alternative to UDP multicast distribution.

Multi-purpose middleware appliances also provide a number of advantages over server-based software implementations. The deterministic nature of hardware processing can reduce the number and severity of latency spikes, and battery-backed or nonvolatile memory can be used to store messages for guaranteed delivery more rapidly than is possible with software running on commodity hardware that accesses disk via the operating system. Message flows to each client endpoint can be individually monitored and managed, and message delivery to fast and slow consumers can be managed independently without affecting each other's performance.

For all these benefits, the true effectiveness of a middleware appliance depends on its implementation. Centralized distribution architectures are only practical if the routing and filtering components are performant and scalable. With this in mind, we set out to assess the performance of a middleware appliance. Since Solace's messaging appliance has been rapidly gaining ground in the financial services sector, claiming microsecond latency and throughput of over ten million messages per second, Solace Systems' 3260 message router was an ideal candidate. We designed a set of tests to determine whether a centralized distribution architecture can provide the performance and

Figure 2: Latency test

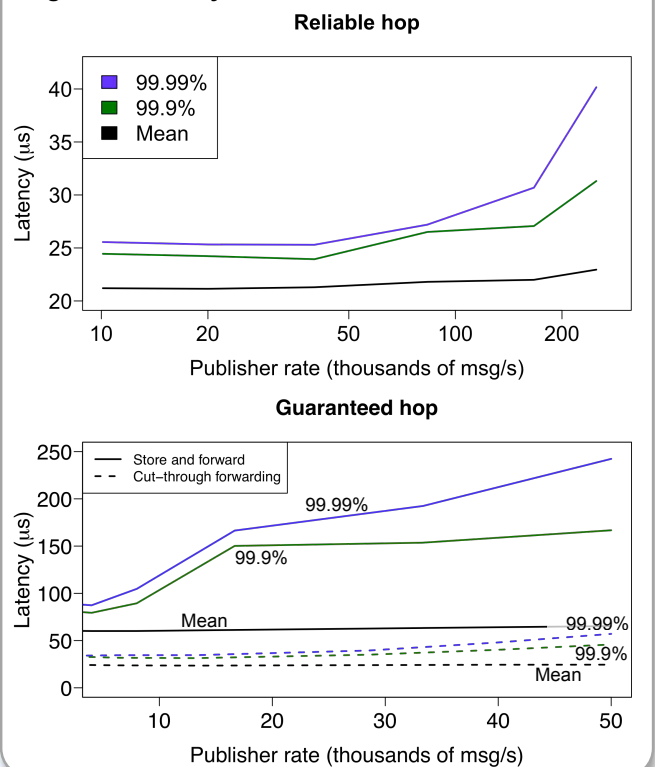
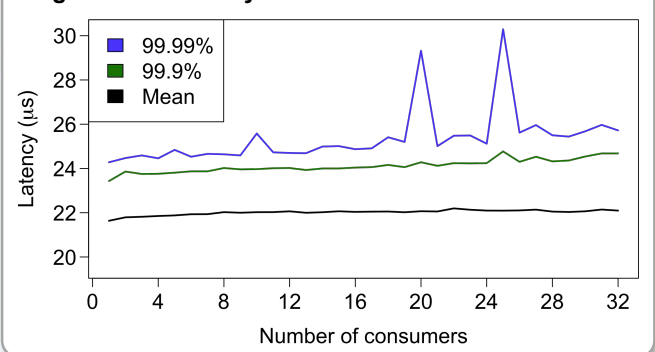


Figure 3: Scalability test



SCENARIO	OBJECTIVE	PRODUCERS	CONSUMERS
Latency test	Determine the platform's latency when streaming rates are received using reliable data delivery and then when orders are sent in response using guaranteed data delivery.	One process that simulates a feed handler.	One process that simulates an algorithmic trading engine, sending one order for every five rate updates received. Another process that simulates an order gateway.
Scalability test	Determine the platform's ability to efficiently filter and distribute high volumes of data over a large number of data streams with low latency using reliable data delivery.	One process that acts as a pricing engine that publishes individual margin-adjusted rate streams.	Thirty-two processes that subscribe to different subsets of the published data streams.
Flexibility test	Determine the platform's ability to accommodate both high-throughput and low-throughput consumers simultaneously, using reliable messaging.	One process that acts as a pricing engine.	One process that consumes data at a high rate. Another slow-consuming process. A third slow-consuming process that uses the router's message eliding feature.

flexibility required to meet the needs of FX trading applications.

Software architecture

A simplified version of an FX trading architecture based on a messaging appliance is shown in Figure 1. One or more FX ECNs provide rate feeds and order gateway interfaces. The bank uses feed handlers and order gateways to communicate with the ECN and other internal trading platform components. An algorithm engine is used to aggregate prices from multiple liquidity sources and run automated trading strategies against the available liquidity. The feed handler and order gateway components would typically be run in an active-passive high-availability configuration. To improve readability, the redundant components required for high availability are not shown in Figure 1.

The pricing engine receives raw or aggregated rates and generates synthetic cross rates for a wide range of forward dates using interpolated tenors. It may also generate option prices across a number of strike prices and expiration dates. Typically, the pricing engine would be run in a load-balanced active-active configuration to provide both resilience and scalability. In some cases, the pricing engine may be interactive, providing rates when a request for a quote is received. Alternatively, the pricing engine may be designed to publish a continuous stream of rates, calculating and sending new prices when raw rate updates are received. The rates distributed will be tradable for a short period of time, frequently less than one second.

The approach of automatically generating and distributing multiple rate streams is straightforward from an application design standpoint, but it requires the underlying messaging platform to filter and route large volumes of data to different types of consumers. "Fast consumers" will receive price data and send orders at high rates. "Slow consumers" may be applications that are unable to keep up with high data rates or are at the other end of wide area network connections that limit the rate at which data can be received. Various other applications will fall between these two extremes.

Test scenarios

The tests were designed to measure key concerns relevant to FX trading platforms:

- the latency and throughput characteristics of the messaging infrastructure;
- the performance impact of increasing the number of consumers;
- the efficiency of distribution for a large number of topics; and
- the impact of slow and fast consumers upon each other.

A logical view of the components in the test environment and the measurement points is shown in Figure 1. The components shown in grey with dashed outlines were not included in the tests. Henceforth, references to the feed handler, order gateway, algo engine and pricing engine will indicate the test components developed to simulate those functions.

For all tests, TS-Associates' Application Tap card and TipOff monitoring appliance were used to timestamp the test messages as they passed through the test components. This configuration avoided clock synchronization problems while minimizing the impact of measurement on the test results. More details of the test approach can be found at:

<http://www.catena-technologies.com/reports/hp-test-approach.pdf>.

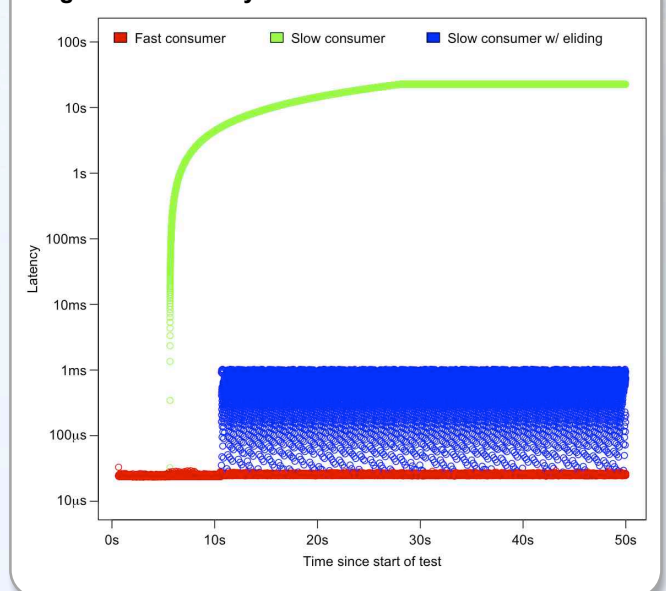
The test results are shown as graphs of the average latency, calculated as the mean over all messages received during the test period, and percentile latency thresholds. The 99.9% latency threshold is the latency measurement exceeded by no more than 0.1% of all messages sampled during a test. Similarly, only 0.01% of messages exceeded the 99.99% latency threshold. This approach was used to show how the extreme latency values were distributed in various test conditions.

The first test (the Latency test) simulated automated order generation triggered by real-time rates received from liquidity providers. Simulated rates were generated by the feed handler and sent to the algo engine component using reliable delivery. The algo engine was configured to send an order message (using guaranteed message delivery) to the order gateway in response to every fifth rate update it received. Latency was measured separately for each hop. This test exercised the basic messaging functionality and helped demonstrate whether the performance would support the use of guaranteed delivery for pre-execution order management.

Two guaranteed delivery modes were tested: store-and-forward and cut-through forwarding. The store-and-forward mode persists messages to non-volatile memory across a fault-tolerant pair of Solace routers before delivering them to subscribers. Cut-through forwarding delivers the messaging in parallel with the persistence operation, thus achieving lower latency.

The second test (the Scalability test) simulated distribution of rates from a pricing engine to many consumers using a large number of distinct topics. The goal of this test was to measure the messaging appliance's ability to filter and distribute high volumes of data efficiently over a large number of different data streams. A maximum of 32 consumers were run, each of which subscribed to a unique set of 500 topics, receiving a total of 2,000 messages per second – i.e. four messages per topic per second.

Figure 4: Flexibility test



The tests results showed average latency of 22-25 μ s in reliable mode, 66 μ s in guaranteed store-and-forward mode, and 25 μ s in guaranteed cut-through mode. Adding consumers or topics did not noticeably affect the system performance.

The third test (the Flexibility test) measured the ability of the messaging platform to accommodate both fast and slow consumers for reliable data distribution. The pricing engine was the same as the Scalability test, but only three consumers were used: one “fast” consumer and two “slow” consumers configured to accept a maximum of 500 messages per second, simulating an application that cannot keep up with high message rates. The Solace router was configured to “elide” messages to one of the slow consumers. This feature caps the maximum rate at which the router will deliver messages to that subscriber. When messages are published more rapidly than the subscriber can handle, the router will only deliver the most recently sent messages, automatically discarding older ones as necessary.

Test results

The results of the latency tests are shown in Figure 2. Mean latency between the feed handler and the algorithm engine using reliable message delivery did not exceed 23 microseconds (μ s) as the publishing rate increased to 250,000 msg/sec, which was determined to be the maximum rate that a single-threaded publisher could sustain on the test hardware. The 99.99% level increased smoothly at a relatively low rate.

Between the algorithm engine and the order gateway, guaranteed delivery with cut-through forwarding demonstrated performance similar to reliable delivery. The mean latency remained consistently below 25 μ s, and the 99.99% level

increased only moderately at higher message rates. Using the guaranteed store-and-forward delivery mode, the mean latencies remained below 66 μ s. This was higher than with cut-through delivery mode because the router synchronously persists messages before delivering them.

The scalability test results, presented in Figure 3, show that adding subscriptions and consumers to the system at moderate message rates had very little effect on latency. Average latency rose from 21.6 μ s with one consumer, 500 subscriptions, and 2,000 messages per second to 22.1 μ s with 32 consumers and a commensurate increase in subscriptions and message rate. More variability was seen at the 99.99% level, though latency remained below 30.3 μ s in all cases. It should be noted that the peaks seen in the graph did not consistently appear at the same number of consumers in all test runs. As only two test servers were available, it was not possible to increase the number of consumers beyond 32 without running out of CPU capacity.

The results of the flexibility test are shown in Figure 4. This differs from the other graphs in that each individual latency measurement is plotted, rather than showing aggregated results. The fast subscriber’s latency is plotted in red and shows consistently low latency. The slow consumer is plotted in green, and shows a rapidly rising and continuously increasing latency, which only levels off when the router has queued so many messages for the subscriber that it begins to discard new messages. The eliding slow

consumer is plotted in blue. Because the router is configured to restrict message delivery to 1,000 messages per second per topic and only deliver the newest message for each topic, its latency never significantly exceeds 1 ms.

It should be noted that in all tests, the limit of the test servers to publish or consume data was reached well before the router’s capacity was reached.

Conclusion

High performance, scalability, and consistent latency are vital in FX trading environments. As tested, the Solace router showed precisely these qualities. It easily accommodated the test systems’ maximum load and maintained predictable levels of latency, even as slow consumers were introduced.

Beyond reliable and guaranteed messaging, Solace’s messaging appliance supports many other features, including horizontal scaling, efficient distribution over wide area networks, hardware-based resilience, and web streaming capabilities. While all of these are useful for foreign exchange trading and other financial services applications, another full-scale testing effort would be necessary to do them justice. Until then, we will be watching to see what new innovations the

Catena Technologies thanks TS-Associates for supplying the TipOff® and Application Tap® hardware to support this testing. This report was prepared in conjunction with Solace Systems, Inc. All trademarks in this document belong to their respective owners.

© 2012 Catena Technologies Pte Ltd
 Catena Technologies is an independent financial technology consulting firm based in Singapore that provides architecture, design, and implementation services to support high performance computing solutions.

www.catena-technologies.com

OVERALL RESULTS

Test	Configuration	Max. pub. rate	Avg. latency	99.9% latency	99.99% latency
Latency test	Reliable hop	250,000 msg/s	23.0 μ s	31.3 μ s	40.2 μ s
Latency test	Guaranteed – store and forward	50,000 msg/s	65.4 μ s	166.8 μ s	242.3 μ s
Latency test	Guaranteed – cut-through forwarding	50,000 msg/s	24.5 μ s	45.9 μ s	57.1 μ s
Scalability test	32 consumers	64,000 msg/s	22.1 μ s	24.7 μ s	25.7 μ s
Flexibility test	Fast consumer	1,000 msg/s	25.2 μ s	27.4 μ s	28.6 μ s
Flexibility test	Eliding consumer	1,000 msg/s	526 μ s	1,026 μ s	1,028 μ s
Flexibility test	Slow consumer	1,000 msg/s	16,600 ms	22,600 ms	22,700 ms

TECHNICAL SPECIFICATIONS

Messaging Appliance		Test Servers	
Model	3260M Message Router	CPU	2 x Intel® Xeon® X5550 2.67GHz 4 cores per socket
Operating system	SoIOS-TR Version 5.3.1.17	Memory	12 GB
	Test Network	Operating system	CentOS 5.2
Network switch	Arista 7148SX	Benchmark software	Custom developed C test applications
TipOff® appliance	TipOff® Stack 2.0.7 Application Tap® firmware 1.1.8; driver 1.0.6	Network interface	Solarflare SFN5122
		Kernel bypass	OpenOnload 20100604-u2