

Low Latency Market Data

Are proprietary protocols needed?



By **Rolf Andersson**, CEO, Pantor Engineering

The FIX Protocol as well as software implementing the protocol (aka "FIX engines") have from time to time been said to have a performance that is too low for use where very high throughput and/or very low latency is required. Performance characteristics of the protocol as well as engine implementations have been discussed in previous articles (Kevin Houston's article "Is FIX

Really Slow?" FIXGlobal Vol 2, Issue 12, December 2009 and John Cameron's "Evolution of the FIX Engine" Vol 2, Issue 7, September 2008). Granted, there are slow implementations in use and the classic FIX tag=value syntax is too verbose for some use cases: e.g., market data.

Recent developments within FPL such as the release of the FAST Protocol

and efforts within the FIX 5.0 usage sub-committee to support alternative recovery state models will enable FIX to be used in high throughput, low latency scenarios. This article reviews the various sources of latency and demonstrates that a FIX over FAST implementation can be used in place of a proprietary protocol to provide very high performance and low latency.

An overview of latency sources

There are a number of latency sources that contribute to the total latency for producing, transferring and consuming market data. The impact of different sources varies widely between implementations. The following sources will be discussed:

- Message processing overhead – the encoding and decoding between transfer format and the internal representation suitable for the processing required in a specific application, as well as safe-storing messages to support recovery of lost messages;
- Communication processing overhead – the network processing associated with sending and receiving messages;
- Scheduling delay – the delay in reacting on a request to send a message or reacting on a notification that a message has been received.
- Transfer delay – the time elapsed between the start and the end of a packet containing one or more messages.
- Propagation delay – a function of the physical distance between two communicating parties and the speed of light in the medium used to communicate.

These latency sources are to a large extent similar in behavior irrespective of the choice of external protocol, but there are differences as discussed below.

Message processing overhead

The overhead of processing a traditional FIX message is negatively affected by a number of aspects:

- Message content – FIX messages contain a host of information for the benefit of different communicating parties.
- Message format – the FIX message format contains

redundant information about the message structure. Text is used to represent all data.

- Recovery semantics – the FIX recovery mechanism is based on message sequencing per session and a contract that a receiver can re-request messages.

Communication processing overhead

The communication overhead depends on the number and size of network packets. Each transferred packet incurs some processing both at the sending and receiving end. One or more messages are transferred in each network packet. Smaller messages mean that less data has to be copied and that more messages can be transferred in each network packet.

Scheduling delay

The scheduling delay is largely independent of message size but implementations may make use of the fact that smaller messages can be packed more tightly together and be transferred in fewer network packets.

Transfer delay

The transfer delay is affected by the size of the messages to be transferred. Each network packet carries a fixed overhead of at least 42-54 bytes depending on the network protocols used. Smaller messages result in lower transfer latency in two ways; network packets will generally be smaller and they will be fewer during peaks.

Propagation delay

The propagation delay is independent of the format and size of messages so there is no difference between FIX and other protocols in this regard.

FIX message processing details

Message Content

Specific rules of engagement are normally agreed upon between two communicating parties, but the current FIX specification contains a number of mandatory fields that may not be needed in a specific context. The inclusion of extra fields increases the message processing



Rolf Andersson,
CEO, Pantor Engineering

overhead as well as the transfer latency, so making more fields optional will allow smaller messages to be used in specific contexts.

Message format

The structure information included in traditional FIX messages is removed in FIX over FAST. A message template is used by the communicating parties to agree on the layout of each message type. Traditional FIX message data is represented as text, whereas in FIX over FAST, numeric data is represented in binary. The binary representation of numeric data in itself results in a space saving of 50 percent and also decreases the encoding and decoding overhead of numeric fields.

Recovery semantics

The traditional FIX recovery mechanism requires a sender to store the sent messages to be able to resend them at a later time if the receiver requests some range of messages to be resent. The safe-storing of messages incurs extra message processing overhead and may render a FIX application unusable in scenarios with stringent latency requirements. The FIX 5.0 working group is considering alternatives that will allow a sender to perform limited or no safe-storing of messages. The receiver

is instead required to use some alternate mechanism of recovering from message loss. This is in line with how ITCH and other proprietary interfaces implement recovery.

A summary of latency contributions

As the table above shows, the value

LATENCY SOURCE	LATENCY RANGE
Message processing overhead (encoding or decoding a message)	100 nanoseconds - 100 microseconds (depending on protocol and implementation)
Communication processing overhead (wire to userland or userland to wire)	1- 50 microseconds (depending on platform and configuration)
Scheduling delay	1- 100 microseconds (depending on platform and configuration)
Transfer delay	100 nanoseconds - 1 millisecond (depending on packet size and line speed)
Propagation delay	10 nanoseconds - >100 milliseconds (depending on physical distance)

range for each of the latency sources is large. As a consequence, there is no single latency source that in general has much greater impact than other sources. It all depends on the specific situation. The propagation delay is not an issue when the communicating parties are co-located. The transfer delay is at the single digit microsecond level or below if 1 or 10 Gigabit interface speeds are used. The scheduling delay will be in the low microseconds if the application code is correctly written and the environment is correctly set up. There are hardware solutions that provide single digit microsecond communication delays. Lastly, it has been previously shown that encoding and decoding of FIX as well as proprietary protocols can be implemented very efficiently.

Comparing ITCH to FIX over FAST

The ITCH protocol has been in production for a number of years. Variants of the ITCH protocol as well as ITCH-like protocols are offered by Nasdaq OMX, CHI-X and BATS. FAST has been in production use since early 2006 and was originally developed by FPL during 2005 to provide efficient support for high volume market data flows. FAST 1.2 is the current version which was released in March 2009. The

primary focus in the development of version 1.2 was to further improve the support for FIX flows. It has been argued by some that ITCH is better suited for high volume market data because of its small messages and its low encoding and decoding overhead.

The traditional FIX market data messages carry a much larger space overhead and are significantly more expensive to encode and decode. A test was conducted based on real-world data to determine the bandwidth and processing characteristics of the two protocols.

The test configuration

A real INET ITCH feed was chosen as a basis for the ITCH versus FIX over FAST comparison. FIX over FAST message types were created based on the ITCH message types rather than using the traditional FIX market data message types. The out of band recovery semantics of the ITCH feed was used for the FIX over FAST implementation as well. Mandatory FIX fields that had no corresponding field in the ITCH feed and had no meaningful default value were removed. Live ITCH market data was then used to re-encode an ITCH data stream as well as a FIX over FAST data stream. The network packet boundaries of the original ITCH stream were preserved in the re-encoded ITCH and FIX over FAST streams to allow close comparison even though more messages could have been packed into each network packet in the FIX over FAST case.

The test results

Packet sizes (including network

protocol headers) and encoding and decoding latencies were measured based on network traces containing approximately 600 million ITCH messages. The results were in line with what has been observed in other comparisons: The FIX over FAST feed had 50% lower bandwidth utilization on average and up to 65% lower utilization in peak situations. With heavily optimized implementations, the raw encoding and decoding speeds of both protocols were in the range of 10 million messages per second (less than 100 nanoseconds per message). The difference between the encoding and decoding overheads of ITCH and FIX over FAST was marginal, with FIX over FAST being approximately 25% faster.

Conclusions

There are many proprietary protocols other than ITCH, but it was chosen as a representative protocol that has been touted as “lean and mean.” ITCH exists in multiple independent versions that each addresses the needs of a specific market. Its simplicity is appealing but its lack of flexibility makes it costly to deploy and re-deploy as the variants in use are similar but different. The tests performed show that FIX over FAST, not only more than keeps up with ITCH in terms of performance, but offers an advantage in terms of lower bandwidth utilization. In addition, FIX over FAST offers a repository and framework for expressing variant versions of an interface. The perceived greater simplicity of protocols like ITCH will increasingly be offset by the availability of high performance, high quality FIX over FAST implementations. Some may have other reasons for choosing a proprietary protocol like ITCH instead of FIX over FAST but bandwidth and processing efficiency are no longer valid reasons.

Any thoughts about this or other articles?
Please send any comments direct to:
editorial@fixglobal.com