IJCSE International Journal of Computer Sciences and Engineering Open Access
Review Paper Volume-3, Issue-5 E-ISSN: 2347-2693

Implementation insight for High Performance Messaging Solution

Muralidaran Natarajan[1*], Nandlal L. Sarda[2] and Sharad C. Srivastava[3]

[1*,3]BIT Mesra, India
[2]IIT Bombay, India

www.ijcseonline.org

Received: Apr/22/2015 Revised: May/01//2015 Accepted: May/20/2015 Published: May/30/ 2015

***Abstract*—**This paper addresses the messaging needs of decomposed multi-tier applications to support high performance. In the transformation exercise, the applications are getting decomposed into multiple logical tiers and get deployed in clusters to take advantage of computing power available in multi-core commodity hardware. In such deployments the communication between the tiers or layers within the deployment is critical for such transformations, as there are millions of small packets moving across these tiers and gets processed at different stages. It is important that a suitable messaging middleware is put to use for the success of such transformation exercise to address the transportation of messages across tiers. This study provides an insight to the developer community on, (a) various communication protocols current and emerging ones that can be used given a problem situation and the deployment nuances of these protocols (b) messaging models and middleware available for such deployment (c) key factors that are to be kept in mind for selection of such commercially available messaging middleware and (d) finally, approach towards deployment of such middleware and key parameters that are available for fine tuning to achieve the desired scalability and latency.

***Keywords*—**Messagin middleware; Messaging for high performance; Messaging models; Ultra-low latency messaging; ; Thin stream

I. INTRODUCTION

Historically, the real-time applications that are used for high performance systems like banking, financial systems and avionics systems have remained as specialized custom developments to meet the stringent QoS needs[1] [2]. Due to the tightly coupled nature of these applications with the hardware platforms as well as the operating environment, the development requires specialized resources and is time-consuming, expensive and so do maintenance [3]. The vertical scalability is achieved by hardware upgrades to a limited extent but the horizontal scalability is custom-built which limits the time to market.

In the next generation of real-time applications, which are driven by the straight through processing demands of the market with inter-connected systems, performance expectations of business have increased multi-fold. The QoS expectations such as near 100% Uptime with response times of the order of microseconds, have transformed the application architectures to cater to the increasing demands for processing power and memory bandwidth and to meet the business needs [2][4].

With the Message Passing Interface (MPI) becoming the standard for high performance and parallel computing, lot of research has been happening on the techniques that can be used for message passing. Based on these researches, the commercial messaging products are continuously getting evolved. In multi-tiered architecture, the performance of the messaging system is the key to guarantee delivery, throughput and response time of transactions.

This paper discusses our review on the various messaging protocols, messaging models with the techniques that are used for large scale business services. The study includes: (a) messaging protocols such as Transmission Control Protocol (TCP), User Datagram protocol (UDP) in unicast, multicast and broadcast modes and new protocols for extended services (b) Various models such as point-to-point, Topic based publish-subscribe (c) Messaging middleware and protocols for meeting the business needs such as Ultra-Low Latency and High Throughput mission critical transaction processing, Large scale information distribution with resilience.

II. NEED OF MESSAGING SOLUTION FOR SCALABILITY

This section of this paper discusses the essentials of messaging in the transformed systems using the DSP models for very high throughput and low latency.

A. Need for Scalable Messaging

With the advent of high performance computing in major industries such as finance, telecom, aviation majority of the real-time applications are enabled for straight through processing (STP). The exchange of goods as well as funds is based on sequence of events, seamlessly completing the entire transaction with interfaces communicating across the systems. In such a model, the messaging layer is the key component for the systems integration to ensure reliability and accuracy of the business process chain. One complete transaction involves exchange of hundreds of messages. More important and an essential feature is therefore the scalability of the messaging layer for the sustainability of the messaging when a new system is interfaced or when business expands multi-fold.

B. Need for Optimal Resources Utilization and Minimal Overheads for Performance

A scalable messaging, an important aspect of motivation for the messaging choice, normally gets limited by the resources utilization as the application eco-system grows and expands. As we progress towards the world of micro-

second services combined with high throughput, the need for differential techniques using optimal resources is needed. Further these are supposed to have minimal overheads to offer predictable latency and eliminate outliers. The traditional method of offering reliability with connection oriented transport suffers from the drawback that the resource usage increases linearly with increase in number of processes. The connection-less datagram service maintains near constant resource usage and overheads. The connection-less protocols therefore become natural choice for larger systems that can handle losses with application-level services and speed is of paramount importance [5].

C. *Ultra-Low Latency and High Throughput for Thin Stream Applications with Reliability*

With the growing end-user driven applications and internet proliferation, the applications are highly interactive requiring higher levels of QoS (Quality of Service) [2]. The data produced by such real-time applications are bursty with high rate of small packet sizes. The user experience in such applications is driven by the response time. The reliable protocols work well on continuous, uniform data streams as these are built with optimal timing for re-transmission and congestion control to maximize the throughput; Offering both ultra-low latency and high throughput for thin stream applications has always remained a challenge petlun [6].

D. *Guaranteed Delivery in Mission Critical Transaction Processing Applications*

Large real-time transaction processing systems require high throughput with guaranteed delivery and latency of the order of micro-seconds [2]. Such a demanding business requirement would need the messaging protocol to ensure that there are no duplications, no losses, no corruption and in case of any error, reporting and correcting time in sub-microseconds. In case of any of these above situations the system will go into recovery and the same will directly affect the predictability of delivery and latency.

III. MESSAGIN PROTOCOLS

This section discusses the messaging protocols. The common end-to-end transport layer protocols today are Transmission Control Protocol (TCP) and User Datagram Protocol (UDP). The choice of the messaging protocol varies with the nature of services offered, the application architecture and the system infra-structure. There are also new protocols that extend the range of services to suit specific application categories [7] [8].

- Transmission Control Protocol:Reliable and in-order delivery of data with adaptive rate control.
- User Datagram Protocol:Unreliable but faster delivery with datagrams.
- New Protocols:New protocols such as SCTP, DCCP extend the transport services customized to handle specific needs.

A. *Reliable Transmission Control Protocol (TCP)*

A reliable, connection oriented transport TCP is the most commonly used protocol, is primarily designed for transmission of bulk data from source to destination as fast as possible to achieve maximum throughput. Because of reliability and robustness, it is used for transactions processing in many of the interactive applications. It is an end-to-end protocol and is widely supported by all ISP firewalls.

1) *TCP Services*

The transport services offered by TCP are: (a) Reliability with retransmission (b) In-order delivery (c) Congestion control by adjusting the send rate (d) Flow control to adapt to receiver's capacity (e) Error control with checksums. To provide these services, the protocol keeps track of state while transmitting each packet. For keeping track, a set of information is to be included in the packet header for every transmission. This results in large overheads as compared to other simpler protocols. As the retransmissions, adaptive rate control and ordering are designed to be handled per connection; applications based on TCP can support connections from various types of devices [9].

Reliability: TCP ensures delivery by way of positive acknowledgements using sequence numbers. Sender transmits the sequence number information in every packet and receiver sends an ACK with the next in-order sequence number expected. In case a particular sequence number is missed, the receiver keeps sending the ACKs for the received data with the expected in-order sequence number, till the missed one is received. Once the missed one is received, the receiver sends the ACK with the next in-order packet expected that is after the successfully received packets. If the sender does not receive ACK for a given packet within a certain time (RTO-Re-transmit timeout), the packet is re-transmitted.

In-Order Delivery: The protocol presents the data to the receiver in the same order as generated by the sender. With the sequence number that is transmitted with every packet, the duplicates, misses, damages, out-of-order packets are handled and the data is presented to the receiver and there is no pairing of read/write operation. This makes it a true streaming protocol suitable for high volume interactive and bulk transfer applications.

Congestion Control: If the transmission rate is very high, the packets may get queued at the routers and can get discarded. The situation is further worsened by causing the sender to initiate re-transmits and this condition is referred to as 'Congestion Collapse'. TCP uses four congestion control algorithms viz. Slow start, Congestion Avoidance, Fast Retransmit and Fast Recovery.

Flow Control: TCP uses the sliding window protocol to adjust to the receiver's capacity. Every packet received, contains the receiver's window advertisement. This becomes the upper bound for the sender's sliding window.

Error Control: Every TCP packet carries a checksum which helps the receiver to detect the errors. TCP also detects duplicate packets and drops already received packets.

Reliable delivery of data in a packet switched network supporting high volume real-time applications is a complex task. The above mentioned transport services make TCP the suitable protocol. However, the current generation high volume mission critical applications require ultra-low latency, combined with high throughput. These applications are normally characterized as "thin data stream" due to the following reasons: (a) the size of the data packet is much smaller than the Maximum Transmission Unit (MTU) size (b) fast retransmission is not triggered as the packet inter-arrival time (IAT) is high. In such applications, the TCP retransmission is initiated by timeout rather than by feedback to trigger fast retransmission. Also, the protocol invokes exponential back off when multiple losses occur and this further increases the latency.

2) Tuning TCP for Ultra-low latency in thin stream interactive DSP applications

The high-performance, high volume applications are thin stream, the data stream is bursty in nature and performance requirements are of the order of micro-seconds. This section discusses the TCP settings that need to be modified for thin stream DSP applications to provide ultra-low latency [10].

Socket buffer sizes: The size of the socket buffers decides the amount of data can be batched and sent to the receiver to improve the throughput. Similarly, at the receiver, batches the network operations to receive the data. It is recommended to maintain these buffer sizes at optimal level and not very high to have bounded latency. Also, the send buffer size should be set within the limits of the receiver buffer to minimize the losses.

Exponential back-off: If the number of packets in-flight that are un-acknowledged is less than the thresholds required for triggering fast re-transmit, the exponential back off factor is disabled. If not, the factor is applied as the chances for faster re-transmit is high and the bandwidth consumption needs to be maintained at the optimal level, which is applicable for thick-stream applications. Disabling the back off for thin streams improves the response time by reducing the latency. In thin streams, fast-retransmit does not result in bandwidth issues.

Fast re-transmit and Selective ACK(SACK): The receiver when receives a packet that has a sequence number that is higher than the expected sequence number, sends a duplicate ACK with the sequence number of the packet expected. After the configured number of duplicate ACKs, the receiver requests re-transmit and the sender re-transmits the packets without waiting for the timeout. Also sends a SACK to enable the sender to send only missing/dropped packets and not the whole stream again. For thin streams, the number of Duplicate ACKs required to trigger fast re-transmit is set to a very low value so that the delay is maintained within limits for thin stream interactive applications, in case of any data loss. Enabling of SACK improves the efficiency further as only necessary packets are re-transmitted.

Redundant bundling TCP to reduce latency: This technique is proposed by Petlund et.al. to reduce the latency[86]. It involves redundant transmission of unacknowledged data by copying the data from send buffer. In case of any data loss, the next packet received would already contain the data. Thus latency is reduced by the redundant sending of data.

TCP is a widely used protocol for interactive real-time applications as well as high volume bulk data applications. While the protocol is designed for high throughput, the settings and tuning methods described above improve the response time for thin-streams by trading off bandwidth due to faster re-transmissions. However, this does not pose a problem as these thin-stream applications rarely use the full bandwidth available. The changes are transparent to the receiver and effective only when the stream is thin and not affecting the TCP behavior when the stream is not thin. Thus the bulk transfer applications will continue to have the benefits offered by TCP.

B. User Datagram Protocol (UDP)

UDP is a connection-less, unreliable, simple message oriented protocol. It is faster than TCP because of the minimal transport services that it offers. Voice and Video traffic are common handled using UDP [11]. VoIP application, in which latency and jitter are required to be minimal and application handles misses and damages, UDP is used; but is not used for applications that require a very high level of reliability. When loss on a network is negligible, UDP is the most suited protocol. Unix Network File System (NFS) is an example one such application that uses UDP when it operates on LAN with its own mechanisms for handling reliability tuned to the application needs. UDP is widely used in applications such as SNMP, TFTP, DNS, Internet telephony and multi-media applications that require speed [11].

1) UDP Services

Unicast, Multicast and Broadcast: UDP support all three forms of data transfer – Unicast, Multicast and Broadcast mode of transmission. Depending on the application's need and the system infra-structure, services can be used.

Datagram service: The protocol supports delivery of datagrams and preserves message boundaries; useful for internet based request-response applications.

Stateless: It is a connection less and stateless protocol and hence resources usage is optimal; suitable for internet based applications with very large number of clients

Data integrity: Provides checksums for data integrity; but no error correction.

No re-transmission delays: Does not support guaranteed delivery, error correction and hence no retransmission delays; suitable for application that cannot tolerate jitters but can handle losses.

2) UDP with application layer reliability services for VoIP/multi-media applications

As discussed in the previous section, UDP is used by applications that require speed but can handle losses and damages. However, there are applications that prefer to use UDP for speed but also require trivial level of reliability. In some cases, the extended transport protocol services such as SCTP, DCCP can be used. In certain categories of applications, it may be desirable to build the necessary services at the application layer with UDP, to provide services specific to the application's need. There are frameworks available such as Enet, UDT with UDP as the underlying protocol [13].

Enet: It is a small library designed for on-line gaming support. It is a simple, robust interface that offers partial reliability with in-order delivery of data. Retransmissions are based on timeout. Delivery can be configured to be message-oriented or stream-oriented. It cannot be considered as a middleware platform as the level of abstraction is very minimal [14].

UDT: It is a comprehensive library designed for high-speed nets and provides various features based on UDP. The congestion control algorithm enables UDT to utilize high bandwidth links effectively. Partial reliability and in-order delivery are supported. It uses timer-based, selective acknowledgement to save bandwidth. However, at very low bandwidth it ACKs every packet that reduces the latency. Retransmissions are managed like TCP but with additionally using negative acknowledgements. Though reliability requirements can be built, building a complete set of services on UDP will be on similar lines as TCP and will result in the same performance if not worse. For such applications, it is normally preferred to tune the TCP settings to offer the desired performance.

3) UDP for sending information to many users in mission-critical e-commerce applications:

The previous paragraph discussed the use of UDP for point-to-point to communication i.e. in unicast mode. The other two modes multicast and broadcast are normally used for sending out information to many users (one to many) without having to write to each user individually.

UDP Broadcast: UDP Broadcast allows sending of packets to a particular network eg. all the machines/devices on a local network. It is a primitive facility and there is no way of segregating the messages into groups. ISPs block the UDP broadcast traffic, as it can congest the entire network. Broadcast can also be a "Directed Broadcast" to send to all hosts on a remote network. Such broadcasting is the primary mode of operation for the physical layer in devices like shared Ethernet, Wireless links and Optical networks.

UDP Multicast: [15] Similar to UDP broadcast, but users subscribe to specific multicast groups, therefore are not bombarded with all the broadcast messages.

With the growth of internet, E-commerce applications, information services have grown extensively and it has become the major medium of corporate communication. Large scale broadcasting of news and events and band width hungry multi-media based live transmissions poses major challenges to the network operators and application designers. One solution that will allow broadcasting to multiple users without escalation in network traffic is multicasting. Hence the best protocol for scalability assurance is multicast. The multicasting technology offers the following advantages: (a) Improved efficiency through reduction of network traffic and server load (b) Improved performance because of elimination of redundant traffic (c) Facilitates Distributed Applications Architecture.

Multicast has three essential components: (a) Multicast addressing (b) Multicast Group Management Protocol (IGMP) (c) Multicast Routing. Multicast addresses follow Class – D addressing scheme. Any packet sent to a multicast address is forwarded to all the users who have joined that multicast group. The router and the switch selectively duplicate and transmit the data; data can be transmitted to multiple hosts belonging to the group without affecting the hosts that are not part of the group. IGMP is a simple protocol for supporting the subscribers to a multicast group like joining a group, leaving a group, routers to query a membership etc. Multicast routing protocols enable the routing of multicast packets across the routers using optimal paths discovered. Multicasting is the most commonly used technology for video conferencing. Multicast is a mode of group communication. With multicast, any machine can join and leave a group dynamically and all members of a group receive all the packets, without any filtering. Due to the benefits, substantial work has gone into the implementation of multicast protocols such as single source multicast, reliable multicast and end-to-end multicast [16][17] [18] [19]. As multicast does not guarantee delivery and does not support access control or security, it is not used for mission-critical transaction processing.

4) Challenges with multicast

In large volume systems, as the business expands, the challenges associated with multicast correspondingly increase. Network traffic does not remain steady as it is completely driven by the business activities. When the business activities are high, information that is multicast increases and hence the traffic pattern is dynamic in nature. Consistency in response time/latency can be achieved with multicast only when the message rate and size remain almost constant [20].

In case of customized reliable Multicast, slow subscribers become a problem, as the publishers have to be sent NACK (Negative Acknowledgement). The processing of the

NACK increases the processing overhead for the publishers and affects the other subscribers. This additional overhead increases the latency spike in the application.

C. *New Protocols*

With the growth of internet applications, new protocols that extend the services and the flexibility of the transport layer to suit very specific application categories. This section discusses a few of such protocols.

Stream Control Transmission protocol (SCTP): The SCTP was originally designed for transporting Public Switched Telephone Network (PSTN) signaling traffic over Internet protocol (IP) networks by the IETF signaling transport (SIGTRAN) working group. SCTP supports a range of functions that is critical to message-oriented signaling transport but over time, features that are useful for other applications also [8][21].

Datagram Congestion Control Protocol (DCCP): The DCCP provides congestion control on UDP like TCP but without reliability. This is useful for time-sensitive applications that need to control delivery timing, without having to implement congestion control at the application [22].

Game Transport Protocol (GTP): The GTP is designed for the transmission of event data used by multimedia on-line game (MMOG) with minimal latency. GTP uses a packet-based window scheme instead of a byte-based window scheme, suitable for small-sized event data. It also does session management and an adaptive retransmission using GTP control blocks. Although it is a specialized protocol for MMOG, it can also be used for other on-line multimedia applications [23].

D. *Messaging Models*

With the above messaging protocols, messaging models emerged towards meeting the non-functional requirements of business systems. This section provides information on those models [24]:

Request-response for reliable one-to-one interactive applications: This is one of the basic models used for message between two applications/modules. It can be a two way communication and hence when implemented over a reliable protocol TCP, it is very powerful for guaranteed delivery and a variety of other transport features such as rate adaptation to support connections from various devices. The messaging solutions, combine queuing with Request-Response to support large systems that support high volume of transactions by overcoming the limits of point-to-point communication.

Scalable Publish-Subscribe for many-to-many loosely coupled applications: This is a model in which the publishers (senders) and subscribers (receivers) are loosely coupled; the publisher need not know about the subscriber and is also independent of the number of subscribers. This model provides much higher scalability than Request-Response model. In messaging systems that support large volume, the publisher sends to an intermediate broker and subscribers register themselves with the broker for the category of messages that they require. The filtering of messages for the subscriber could be based on topics or content or both. The choice of protocol for the intermediate broker TCP or multicast is dependent on the reliability needs of the business requirement. The decoupling nature of publish-subscribe model makes it more suitable for low latency requirements as it allows for quick re-configuration of the path across multiple network domains, when issues are detected in the network.

IV. MESSAGING MIDDLEWARE EVALUATION

The growing need for messaging solutions towards scalability and security resulted in software vendors providing products with interfaces between applications and between modules in large business systems. The enterprise backbone of the middleware enables integration across heterogeneous IT platforms with distributed computing environments, allowing real-time data exchange asynchronously without the need for a synchronous sender-receiver. The middleware program supports features like store and forward with powerful persistence that guarantees delivery even though receiver is not up when the sender sends the data. These features are configurable and therefore can be turned on-off depending on the criticality and the business need. Being standards based, the middleware greatly eliminates proprietary dependencies, minimizes the deployment cost and time to market.

This subsection provides insight into selection of a suitable messaging middleware based on the followingaspects:

- Features of messaging middleware
- Selection criteria for high throughput ultra-low latency systems
- Implementation considerations for a typical DSP
- Key parameters tuningfor high throughput and ultra-low latency
- A simulation exercise to arrive at appropriate SSB and RSB setting

A. *Features of Messaging Middleware*

This section describes the features of the COTSmessaging middleware that are essential [25][26] for supporting any business application. Evaluation must ensure that the support for these features is mandatory.

1) The middleware to provide comprehensive support for **messaging standards** such as JMS, J2EE, JTA XA API, XML, SOAP, HTTP, HTTPS, SSL and TCP/IP.
2) The message queuing or publish-subscribe, is expected to support high volume and low latency on demand. The middleware to support high performance enabling features such as connection pooling, in-memory storage, efficient persistence and horizontal scalability to deliver **Scalability**QoS [27].
3) To support High Availability(HA), Fault Tolerance (FT) and load balancing to meet the ResilienceQoS of the business.[28].

4) To meet the confidentiality and SecurityQoS of business, messaging products should support strong, industry-standard authentication, authorization, transport security and data security. For authentication and authorization, support interfacing with LDAP, NT/UNIX realms and databases. The transport security is provided with SSL for both TCP and HTTP and digital certificates. The data security is guaranteed with strong encryption algorithms. Security standards include SSL, TLS, JCE, LDAP and PKCS [29].
5) To support wide variety of IT platforms running various Operating Systems and devices such as web, mobile devices with these solutions.
6) Multi-protocol support: The message broker in a middleware supports multiple protocols, to suit the needed quality of service levels and deliver the best performance in terms of throughput and latency. After selecting a messaging middleware, the application architect must choose the protocols needed by the application suiting the business requirement and the operating environment. The most commonly used TCP/IP based protocols used by message brokers are [30] [31].
   - AMQP (Advanced Message Queuing Protocol) to support reliability and interoperability
   - It provides a wide range of features messaging, reliable queuing, topic based publish-subscribe, dynamic routing, transaction processing and security. It is a good choice for building large scale, reliable and resilient mission-critical messaging applications.
   - MQTT (Message Queue Telemetry Transport) to support high latency, low bandwidth networks
   - It provides publish-subscribe model with low footprint suitable for mobile and "Internet Of Things" applications with message pushing needs, to support many thousands of concurrent device connections such as mobile notifications, weather updates, market updates etc.
   - STOMP (Simple/Streaming Text Oriented Messaging Protocol) to support text based messaging.
   - It does not operate with topics or queues, but deals through "destination string". Since there are no standard specifications, the vendor implementations are not standardized and there are different flavours. However, it is simple and light-weight making it suitable for simple browser/web update applications.

B. *Selection Criteria for High Throughput, Ultra-low Latency Systems*

This section discusses the key functional as well as non-functional factors based on which messaging middleware are to be evaluated for a high throughput, ultra-low latency mission critical DSP system.

1) **Message throughput:** The applications that handle high volumes with thousands of connected users/devices need to handle millions of messages/second. Connection and message handling capacity is the foremost selection criteria.
2) **Ultra-low latency:** Time critical interactive applications; require the response time to be of the order of micro-seconds for a good user experience. The facilities supported by the middleware for the protocols, the intermediate in-memory storage, speed of persistence/recovery and the parameters towards achieving low latency for transaction execution need to be evaluated.
3) **Guaranteed delivery:** Mission critical applications require assured delivery. Towards the same the middleware needs to offer support for 'No loss', 'No Duplication', 'No Corruption'. Commercial products offer 3 levels of QoS: persistent, non-persistent and transactional.[32].
4) **Scalability and Resiliency:** For mission critical systems, the scalability and high availability/fault tolerance features of the messaging system are studied to suit the requirements. This will help in deciding the ease of capacity upgrade when the volume increases and system uptime in case of failures. With the growth of IOT, messaging technologies have extended their services with multicast, as multicast is the best way to guarantee scalability. Support for multicast protocol is to be included as part of the scalability evaluation criterion.
5) **Choice of platform for the middleware:** Industry reports often indicate that there is a variation in performance across various hardware platforms. The evaluation needs to be done for choosing the right combination of middleware, the hardware and the application services.
6) **Choice of protocol:** Depending on the application requirements, the appropriate protocol is to be chosen for each layer and tests are to be done for the above mentioned factors with fine tuning of the parameters. The cases where guarantee of every message is of utmost importance, TCP with its customized parameters is to be evaluated. There could be needs where the message delivery to the users is highly time sensitive but with compromised reliability. In such cases, the multicast features or Reliable UDP unicast are to be evaluated thoroughly.

C. *Implementation Approach for DSP*

Once the messaging middleware choice is done as described in the previous section, the next step is to configure the middleware for the implementation. The messaging middleware could be commercial products like Tibco, 29West, Fiorano MQ, Reuters RMDS or an open source platform like AMQP etc.,. The implementation of any standardized middleware is discussed with respect to a layered architecture that is designed to have a lean business tier for best performance as elaborated in chapter 5. Design of a high volume, ultra-low latency real-time processing system involves choosing the right network transports considering the QoS requirements. Different layers would require the appropriate solution to be chosen depending on the service needs. It therefore means to choose a hybrid of

network transports and implement with the techniques that would be suited for that layer of the system. Integration is a complex task that would require planned architecting. The middleware deployed needs to be configured with the chosen network transport for each layer.

1) **UDP for input channels:** The communication layer receives the input from various channels. These operate in highly parallel mode as the data is stateless at this stage. The input channel receives streams of data from various sources. The sources could be of varying network capacity and speed. The streams from these sources are validated for the stateless conditions using the details in metadata repository. This layer accepting the data from users can operate using connection-less protocol such as UDP for high performance.
2) **TCP or Reliable Unicast for acknowledged data:** After validation, data accepted for processing is forwarded to the 'Business Processing Layer'. Once accepted, the transactions are expected to be guaranteed and processed with low latency. This message passing is necessarily using a reliable transport as the validated and accepted data is presented for business processing. It is therefore recommended to use Reliable Unicast or TCP for reading the input without much delay and to write the data to the 'Business Processing Layer'. If the stream is classified as thin, then the appropriate protocol supported by the middleware to have low latency needs to be chosen; in middleware like 29West LBM, this is supported as 'Latency bounded TCP'. Reliable Unicast is the customized protocol built on UDP supported by messaging middleware for reliability with low latency.
3) **TCP or Reliable Unicast to handover to 'Response Service':**The 'Business Processing Layer' reads the stream, using reliable protocol such as TCP as the reliability and the order of input messages are significant. For ultra-low latency the data is processed in flight by passing through the rule engine; store-and-forward is avoided. In parallel, the validated but unprocessed data stream is sent to the 'Data Object Store'. In high performance systems, this data store could reside in memory to be used during the course of business processing. The processed data is sent to the 'Response Service' using reliable protocol.
4) **TCP for persistence:**The processed information is sent to the 'Enterprise Persistent Store' in parallel using a 'Daemon Service' to ensure no impact on real-time processing and timely response. Reliable 'TCP' is the protocol suited for the requirements of this layer for reading the input and sending the output to guarantee transactions.
5) **Reliable multicast for transmitting to 'Business Processing Layer' (Specialized variation Implementation):** However, there is a specific category of applications where timeliness of the input is the most important criterion such as avionic systems, where data from sensors that is to be processed by the alerting modules. In such systems, reliable multicast is the preferred option to transmit the data to the 'Business Processing Layer'. The reliable multicast offered by the middleware technologies optimizes the traffic and the reliability is offered by having more than one receiver. In case of any miss detected by the receiving process, other receivers are contacted and not the source to fill the gap and to maintain the order of the latest information. Contacting other receivers for missing information reduces the load on the single source. Due to the performance advantages, the multicast implementation by the messaging middleware is continuously getting evolved to have scalable ultra-low latency. The topic and content based filtering make the implementation further light and the detection/recovery is handled by the subscribers. In the business layer, the receiving processes subscribe to the topics/contents for their need so that the load of messages is optimum and CPU cycles are not wasted in rejecting unwanted messages. The number of publishers-subscribers, topics and the size of messages need to be configured based on the test runs for the best performance.
6) **Latency Bound TCP or Multicast for 'Response Sender':** The 'Response Sender' layer runs multiple threads/processes to maximize the throughput as the data is stateless. This layer is configured to send the data in parallel to the consumers using 'Reliable TCP' and unreliable multicast depending on the service. The response to a transaction execution which is sent over interactive channel is sent using 'Reliable TCP' and in latency sensitive applications that are thin stream using 'Latency bounded TCP'. The broadcast information which is common to the entire community of users is normally sent using unreliable multicast as it is light-weight in terms of resources consumption and latency. Also such information is time-sensitive and the next update cycle would satisfy the latest information needs, rather than re-sending the obsolete information. The point-to-point delivery mechanism is avoided as slow consumers will cumulatively increase the latency and will overall impact the performance. The multicast that is based on publish-subscribe technology with topic/content based filtering that is most suitable for all services other than interactive individual transaction response.

The configuration parameters of the messaging middleware have a profound impact on the messaging performance. The parameters are fine-tuned towards achieving high throughput combined with ultra-low latency and ensuring reliability by avoiding losses. These parameters are to be tested for performance in the benchmark setup under various scenarios as the impact is due to the combination of parameters. The fine-tuning of the configurable parameters is explained in the next sub section.

D. Key Parameters Tuning for High Throughput and Ultra-low Latency

This section discusses the impact of key parameters supported by common middleware platforms on

performance. The values suggested are applicable for a high performance system that handles messages of the order of 40K/second with the lab tests conducted.

1) **Receiver socket buffer size:** TCP receive buffer size sets the buffer allocated to the socket of the receiver. TCP being a reliable, flow control based protocol, the sender will send till there is space in the receive buffer of the receiver. It is set equal to or more than the send buffer of the sender. Increasing the size of this buffer improves the throughput but affects the response time i.e. negative impact on low latency requirements.
2) **Sender socket buffer size:**TCP send buffer size sets the buffer allocated to the socket of the sender. TCP being a reliable, flow control based protocol, the sender will send till there is space in the receive buffer of the receiver. It is set equal to or less than the receive buffer of the sender. For optimal performance, it is recommended that the sender operates within the limits of the receiver capacity to minimize data loss and latency. Increasing the size of this buffer in line with receive socket buffer improves the throughput but affects the response time i.e. negative impact on low latency requirements.

These two parameters have a significant impact on performance. The simulation exercise done for a typical DSP application is described in sub section 6.4.5.

3) **Packets rate interval:** This option is not valid for TCP; however, in case of multicast, controls the rate at which information is transmitted. Along with buffering this option helps to improve the performance by minimizing congestion.
4) **Packets rate limit:**This option helps to set the bandwidth limit such as 10Mbps, 100Mbps, 200Mbps etc. Along with packet rate interval, this option assists in achieving maximum throughput. While configuring messaging middleware, this is set to a higher value of the order of 1000Mbps for high throughput to accommodate data as well as re-transmissions. Default value is of the order of 10Mbps.
5) **Batching size:**This is the buffering option at the messaging layer level as against the socket buffer size which is at the kernel level to increase the throughput. This can be increased from 2K to 32K depending on the throughput need. If the kernel level option is used, this should be turned off/fine-tuned so that latency is controlled.
6) **Batching interval:**This option should be used along with Batching size option.
7) **Data buffering for transport session:**This parameter refers to the maximum amount of buffered data to be retained at the source for re-transmission. It is increased to minimize the losses. This parameter is increased from 25Mb to 600Mb.
8) **Window for re-transmission for reliable multicast:**The reliable multicast involves negative ACKs for re-transmission requests either to the source or the other multicast receivers. The re-transmission is implemented as reliable unicast and this option sets the re-transmission window to avoid congestion. This is increased to reduce the unrecoverable loss.This is set to be of the order of 512Mb as against default value of 25Mb.
9) **Initial and subsequent NAK delay:**This option sets the delay for the initial NAK in case of data loss for reliable multicast. In low latency applications, the initial NACK is set to a lower value than the default value.The value is set to 10ms as against the default value of 50ms for response critical applications. Subsequent regeneration requests are delayed to avoid unrecoverable burst loss errors. This parameter is increased from 1 second to 60 seconds.
10) **Max Burst lost size:**Loss, more than this size is declared as unrecoverable loss. To minimize unrecoverable losses and to assist in recovery, the value of this parameter is increased. Along with the increased delay for retransmission requests, the burst loss size is also increased.

E. Simulation Exercise for Tuning SSB and RSB

This sub section describes the simulation conducted to arrive at the various buffer sizes to achieve the desired throughput with optimal latency for a high performance DSP.Table 6.1 shows the various configurations with the resultant throughput, latency and the recommended settings.

1) Run 1 is the baseline experiment for measuring the throughput and latency with the initial configuration for sender and receiver tasks.
2) Runs 2 and 3 are for fine tuning the transmit buffer size (TRM_BUF) and to verify the impact on throughput and latency. In run 2 and 3, the transmit buffer size is reduced from the baseline configuration.
   - In run 2, the transmit buffer of the sender is reduced from 64K to 8K. With this change, the response time improves but the throughput reduces.
   - Further reduction in the transmit buffer of the sender, in run 3 from 8K to 512, improves the response time due to reduced memory consumption with no major impact on throughput compared to Run 2.
3) Runs 4, 5 and 6 are for fine tuning the Send (SSB) and Receive (RSB) socket buffers.
   - In run 4, the SSB of the sender is increased from 4K to 8K. The throughput increases but the response time degrades.
   - In run 5, the TRM_BUF of the sender and receiver are maintained at 512, the RSB of the receiver (16K) is made higher than the SSB of the sender (4K). This setup gives optimal throughput and latency with reduced memory consumption.
   - In run 6, the SSB and RSB of the sender are increased by 4x and SSB and RSB of the receiver are increased by 8x. Though this improves the throughput, due to high memory consumption, response time suffers.

	Pkt_size = 512 ; No of Pkts = 1M									
	Sender		Receiver				RTT LATENCY			
RUN	TRM_ BUF	SSB	RSB	TRM_ BUF	SSB	RSB	AVG	90%	99%	Throughput
1	64K	8K	64K	64K	64K	1M	1.4240	2.0240	2.9860	77.06 Kmsgs/sec, 315.6 Mbps
2	8K	4K	4K	64K	64K	1M	0.5352	0.6800	1.2220	42.14 Kmsgs/sec, 172.6 Mbps
3	512	4K	4K	64K	64K	1M	0.2958	0.3360	0.7300	37.39 Kmsgs/sec, 153.1 Mbps
4	512	8K	8K	64K	64K	1M	0.3704	0.3920	1.9430	52.39 Kmsgs/sec, 214.6 Mbps
5	512	4K	8K	512	8K	16K	0.2974	0.3390	1.5230	39.25 Kmsgs/sec, 160.8 Mbps
6	512	16K	32K	512	64K	128K	5.8230	0.6460	173.50	65.44 Kmsgs/sec, 268 Mbps

Table 6.1 Buffer Size Fine Tuning for Performance

The recommendation for desired throughput with optimal latency is therefore to have TRM-BUF, SSB and RSB size for reduced memory consumption with RSB of the receiver to be greater than the SSB of the sender.

V. CONCLUSION

A Perspective is presented on various messaging protocols, extended protocols and new protocols of latest market trends based on the study and survey of COTS middleware. These are analyzed based on the category of business applications that they are intended to support. With growing requirement for ultra-low latency applications, low latency requirements for thin streams are discussed in detail. With these basics, the selection criteria for any messaging middleware for high throughput, ultra-low latency applications is discussed which would provide guidance for the system designer. Also, the implementation of typical high volume data stream processing application based on a layered architecture is analyzed. The analysis covers the messaging models and the protocols for various business services covering both stateless and state-full processing to maximize the performance. In a messaging middleware, configurable parameters play a key role in meeting the performance requirements of various types of services. The fine tuning ofthesekey parameters is studied and discussed to help the developer community with respect to TCP/multicast protocols towards offering reliability, high throughput and low latency.

ACKNOWLEDGEMENTS

The authors would like to thank the support of National Stock Exchange of India Limited for the lab facilities for the research work.

REFERENCES

[1] A. R. Alkhawaja, L. L. Ferreira and M. Albano, "Message Oriented Middleware with QoS Support for Smart Grids", INForum 2012 - Conference on Embedded Systems and Real Time, 2012.

[2] Holger Wunderlich, Diego Cardalliaguet, Russ Heald, Tomokuni Shimizu and Dirk Ziesemann, "IBM, Building Multi-Tier Scenarios for Web Sphere Enterprise Applications", An IBM Redbook Publications, 2003.

[3] EDS, "Financial Services Legacy Modernization", EDS White Paper, 2007.

[4] Shameem Akhter and Jason Roberts, "Multi-Core Programming", An Intel Press Publications, ISBN 0-9764832-4-6,2006.

[5] A Friedley, T Hoefler and ML Leininger, "Scalable High Performance Message Passing over InfiniBand for Open MPI", e-reports.ext.llnl.gov, 2007.

[6] Andreas Petlund, "Improving latency for interactive, thin-stream applications over reliable transport", Doctoral Thesis, http://urn.nb.no/URN:NBN:no-24274, October 2009.

[7] Ross Carter, "Microsoft Real-Time Communications : Protocols and Technologies", Microsoft TechNet Library, July 2003.

[8] Jan Newmarch, "Introduction to Stream control Transmission Protocols", Linux Journal, September 2007.

[9] Mark Allman, "Improving TCP Performance over Satellite Channels", Doctorol Thesis, 1997.

[10] A Petlund, K Evensen and P Halvorsen, "Improving application layer latency for reliable thin-stream game traffic", Netgames `08Proceedings of the 7th ACM SIGCOMM Workshop on Network and System Support for Games, 2008, pp.91-96.

[11] Mark A. Miller, "Voice Over IP Technologies : Building the Convergent Network", John Willy & Sons Inc., NY,ISBN:0764549073, 2002.

[12] Manoj Bhatia, Jonathan Davidson, Satish Kalidindi, Sudipto Mukherjee and James Peters, "VoIP: An In-Depth Analysis", Publication by CISCO Press, October, 2008, pp.145-168.

[13] PPK Lam, SC Liew, "UDP-Liter: an improved UDP protocol for real-time multimedia applications over wireless",1st International Symposium on Wireless Communication Systems, 2004, pp.314-318.

[14] Mike Diehl, "Network Programming with Enet", Linux Journal, 2012.

[15] PK Chrysanthis, V Liberatore and K. Pruhs, "Middleware Support for Multicast-based Data Dissemination: A working Reality", Proceedings of the Eighth International Workshop on IEEE Explore Object-Oriented Real-Time Dependable Systems, (WORDS 2003), 2003, pp.265-272.

[16] A. Tripathi, A.K. Gupta and Dr. D. Arora, "Comparative Analysis of Quality Services of Dense and CBT Mode of Multicast Routing Strategies", International Journal of Scientific and Research Publications, Volume 3, Issue 2, February 2013.

[17] D. Zappala and A. Fabbri, "Using SSM Proxies to Provide Efficient Multiple- Source Multicast Delivery",Global Telecommunications Conference, GLOBECOM '01, IEEE,Volume 3, 2001, pp.1590-1594.

[18] L. Rizzo, L. Vicisano, "A Reliable Multicast data Distribution Protocol based on software FEC techniques", The fourth IEEE Workshop on High-Performance Communication Systems, 1997, pp.116-125.

[19] Y.Chu, S. G. Rao, S. Seshan and H.Zhang, "Enabling Conferencing Applications on the Internet using an Overlay Multicast Architecture", SIGCOMM'01, 2001,pp.55-67.

[20] "The Advantage of Using Hardware-Based TCP Fanout for High-Performance Messaging", http://www.solacesystems.com,Solace Systems.

[21] "Stream Control Transmission Protocol: Past, Current, and Future Standardization Activities",IEEE Communications Magazine, April 2011.
[22] Mohammad Nayeem Teli, Endrit Thanasi, "Analysis of Datagram Congestion Control Protocol (DCCP)", Colorado State University, 2005.
[23] S Pack, E Hong, Y Choi, I Park, "Game Transport Protocol: A Reliable Lightweight Transport Protocol for Massively Multiplayer On-line Games (MMPOGs)",Proceedings of SPIE 4861, Multimedia Systems and Applications, 2002.
[24] E. C. Eugène and L.A.A. Frejus, "Asynchronous Message Exchange System between Servers based on Java Message Service API", International Journal of Computer Science Engineering (IJCSE), Volume 1, Issue 2, November 2012, pp.144-152.
[25] "Fiorano MQ Enterprise Messaging", Fiorano, 2015.
[26] "Messaging Middleware – a Technical Reference Guide for Designing Mission-Critical Middleware Solutions", Microsoft, SQL Server 2012, 2012.
[27] G Chen, Y Du, P Qin and L Zhang, "Research of JMS Based Message Oriented Middleware for Cluster", International Conference on Computational and Information Sciences, 2012, pp.1628-1631.
[28] H Abie, RM Savola and I Dattani, "Robust, secure, self-adaptive and resilient messaging middleware for business critical systems", Computation World, 2009, pp.153-160.
[29] RS Wu, SM Yuan, "A Pluggable Security Framework for Message Oriented Middleware", 5th WSEAS International Conference on Applied Computer Science, April 2006, pp.1045-1050.
[30] Andrew Foster, "Messaging Technologies for the Industrial Internet and Internet of Things", Prism Tech Corp. White paper, May 2015.
[31] D. Sangvikar, V. Tekale, "Multi Protocol Cross Platform Communication", IJERT,Volume 3, Issue 5, May 2014.
[32] P Tran and P. Greenfield, "Behavior and Performance of Message-Oriented Middleware Systems", 22nd International Conference on Distributed Computing Systems Workshops,2002, pp.645-650.