The transmission of broadcast data, such as financial data and news feeds, is accelerated over a communication channel using data compression and decompression to provide secure transmission and transparent multiplication of communication bandwidth, as well as reduce latency. Broadcast data may include packets having fields. Encoders associated with particular fields may be selected to compress those particular fields.
2001/0047473 A1 11/2001 Fallon
2002/0097172 A1 7/2002 Fallon
2003/0142874 A1 7/2003 Schwartz
2003/0191876 A1 10/2003 Fallon
2006/0015650 A1 1/2006 Fallon
2006/0181441 A1 8/2006 Fallon
2006/0181442 A1 8/2006 Fallon
2006/0184687 A1 8/2006 Fallon
2006/0184696 A1 8/2006 Fallon
2006/0190644 A1 8/2006 Fallon
2006/0195601 A1 8/2006 Fallon
2006/0195602 A1 8/2006 Fallon

FOREIGN PATENT DOCUMENTS

EP 0164677 12/1985
EP 0185998 6/1986
EP 0283798 9/1988
EP 0595406 7/1994
EP 0405572 14/1994
EP 0587437 2/2002
GB 2162025 1/1986
JP 605198 2/1994
JP 9188009 7/1997
JP 11140376 6/1999
WO WO 9429852 12/1994
WO WO 9502873 1/1995
WO WO 9748212 8/1997
WO WO 9908146 2/1999
WO WO 02/39591 5/2002

OTHER PUBLICATIONS

Tridgell, Andrew; “Efficient Algorithms for Sorting and Synchronization”; A thesis submitted for the degree of Doctor of Philosophy at The Australian National University; Feb 1999; pp. iii-106
Jones, et al.; “Lossless data compression for short duration 3D frames in position emission tomography”; IEEE Conference Record Nuclear Science Symposium and Medical Imaging Conference; vol. 3; pp. 1831-1834
Maier, Mark W.; “Algorithm Evaluation for the Synchronous Data Compression Standard”; University of Alabama; pp. 1-10
ALDC: Adaptive Lossless Data Compression; IBM; 1994
ALDC-Macro: Adaptive Lossless Data Compression; IBM Corporation; 1994
ALDC1-208: Adaptive Lossless Data Compression; IBM Corporation; 1994
ALDC1-40: Adaptive Lossless Data Compression; IBM Corporation; 1994
ALDC1-SS: Adaptive Lossless Data Compression; IBM Corporation; 1994
Craft, David J.; “Data Compression Choice No Easy Call”; Computer Technology Review; vol. XIV, No. 1; Jan. 1994
Costlow, Terry; “Sony designs faster, denser tape drive”; Electronic Engineering Times; May 20, 1996, pp. 86-87
Wilson, Ron; “IBM ups compression ante”; Electronic Engineering Times; Aug. 16, 1993; pp. 1-94
“IBM Announces New Feature for 3480 Subsystem”; Tucson Today; vol. 12, No. 337, Jul. 25, 1989
Syngress Media, Inc.; “CCA Citrix Certified Administrator for MetaFrame 1.8 Study Guide”; 2000
Cisco Systems; “Cisco IOS Data Compression”; 1997; pp. 1-10
Craft, D. J.; “A fast hardware data compression algorithm and some algorithmic extensions”; IBM J. Res. Develop.; vol. 42; No. 6; Nov. 6, 1998; pp. 733-746
Rustici, Robert; “Enhanced CU-SeeMe” 1995, Zero In Technologies, Inc.
Data Compression Applications and Innovations Workshop; Proceedings of a Workshop held in Conjunction with the IEEE Data Compression Conference; Snowbird, Utah; Mar. 31, 1995
Sattler, Michael; “Internet TV with CU-SeeMe”; Sams.Net Publishing; 1995; First Edition
IBM Microelectronics Conmed Fall ’93 Booth Location.
Han, et al.; “CU-SeeMe VR Immersive Desktop Teleconferencing”; Department of Computer Science; Cornell University; To appear in ACM Multimedia 1996
Non-Final Office Action in Inter Partes Reexamination of U.S. Pat. No. 6,604,158, U.S. Appl. No. 95/000,486, issued Nov. 12, 2009, 199 pgs.
Expert Report of Dr. James A. Storer on Invalidity filed on behalf of some of the defendants [Includes Appendices—Exhibits A-K (Exhibit A has been redacted pursuant to a protective order)] filed in Realtime Data, LLC v Packeteer, Inc., et al., Civil Action No. 6:08-cv-00144-LED, U.S. District Court for the Eastern District of Texas, Jun. 10, 2009, 1090 pgs.
Presentation to 35 U.S.C. Section 282 Disclosures, Realtime Data, LLC v Packeteer, Inc., et al., District Court for the Eastern District of Texas, No. 6:08cv144, filed May 8, 2009, 3 pgs.
Declaration of Patrick Gogerty, Realtime Data, LLC v Packeteer, Inc., et al., District Court for the Eastern District of Texas, No. 6:08cv144, executed May 8, 2009, 3 pgs.
Blue Coat Defendants’ Notice Pursuant to 35 U.S.C. Section 282 Disclosures, Realtime Data, LLC v Packeteer, Inc., et al., District Court for the Eastern District of Texas, No. 6:08cv144, filed Dec. 11, 2009, 7 pgs.
Expand Networks’ 35 U.S.C. Section 282 Disclosures, Realtime Data, LLC v Packeteer, Inc., et al., District Court for the Eastern District of Texas, No. 6:08cv144, filed Dec. 11, 2009, 4 pgs.
Expand Networks’ 35 U.S.C. Section 282 Disclosures (Amended), Realtime Data, LLC v Packeteer, Inc., et al., District Court for the Eastern District of Texas, No. 6:08cv144, filed Dec. 11, 2009, 5 pgs.
Defendant Citrix Systems, Inc.’s Notice of Obviousness Combinations Pursuant to Court Order, Realtime Data, LLC v Packeteer, Inc., et al., District Court for the Eastern District of Texas, No. 6:08cv144, filed Dec. 11, 2009, 3 pgs.
Order of United States Magistrate Judge regarding Motion to Limit the Number of Prior Art References to be Asserted at Trial, Realtime Data, LLC v Packeteer, Inc., et al., District Court for the Eastern District of Texas, No. 6:08cv144, filed Dec. 21, 2009, 6 pgs.
Expanding Defendants’ Notice of Obviousness Combinations Pursuant to Court Order, Realtime Data, LLC v Packeteer, Inc., et al., District Court for the Eastern District of Texas, No. 6:08cv144, filed Dec. 22, 2009, 3 pgs.
Blue Coat Systems, Inc. and 7-Eleven, Inc.’s Notice of Obviousness Combinations to be Used at Trial, Realtime Data, LLC v Packeteer, Inc., et al., District Court for the Eastern District of Texas, No. 6:08cv144, filed Dec. 22, 2009, 30 pgs.
Defendant Citrix Systems, Inc.’s Notice of Other Prior Art References Within the Scope of the References Discussed at the Dec. 17, 2009 Hearing, Realtime Data, LLC v Packeteer, Inc., et al., District Court for the Eastern District of Texas, No. 6:08cv144, filed Dec. 29, 2009, 6 pgs.
Docket Listing downloaded Mar. 12, 2010 for Realtime Data, LLC v Packeteer, Inc., et al., District Court for the Eastern District of Texas, No. 6:08cv144, filed Apr. 18, 2008, 165 pgs.
Preliminary Data Sheet, 9600 Data Compressor Processor, Hi/fn, 1997-99, HIFN 000001-68, 68 pgs.
Specification Update, 9751 Compressor Processor, Hi/fn, 1997-99, HIFN 000208-221, 14 pgs.
9711 to 7711 Migration, Application Note, Hi/fn, 1997-99, HIFN 000354-361, 8 pgs.


* cited by examiner
FIG. 1
FIG. 2
SELECT GLOBAL STATE SYSTEM

PROCESS INPUT DATA TO ACQUIRE COUNTS OF N-TUPLE SEQUENCES IN EACH GLOBAL STATE

APPLY PREDETERMINED COUNT THRESHOLD

FOR A GIVEN GLOBAL STATE, GENERATE A SUBSTATE FOR EACH N-TUPLE WHOSE COUNT EXCEEDS PREDETERMINED COUNT THRESHOLD

FIG. 3
FIG. 5
FIG. 6
SYSTEM AND METHOD FOR DATA FEED ACCELERATION AND ENCRYPTION

CROSS REFERENCE TO RELATED APPLICATIONS

This is a continuation of patent application Ser. No. 10/434, 305, filed on May 7, 2003, which is a Continuation-in-Part of U.S. patent application Ser. No. 09/699,987, filed on Oct. 3, 2001, which claims the benefit of U.S. Provisional Application No. 60/237,571, filed on Oct. 3, 2000, each of which are fully incorporated herein by reference. In addition, this claims the benefit of U.S. Provisional Application No. 60/378,517, filed on May 7, 2002, which is fully incorporated herein by reference.

TECHNICAL FIELD

The present invention relates generally to systems and method for providing data transmission, and in particular, to systems and method for providing accelerated transmission of data, such as financial trading data, financial services data, financial analytical data, company background data and news feeds, advertisements, and all other forms or information over a communication channel using data compression and decompression to provide data broadcast feeds, bi-directional data transfers, and all other forms of communication with or without security and effectively increase the bandwidth of the communication channel and/or reduce the latency of data transmission.

BACKGROUND

The financial markets and financial information services industry encompass a broad range of financial information ranging from basic stock quotations, bids, order, fulfillment, financial and quotations to analyst reports to detailed pricing of Treasury Bills and Callable bonds. Users of financial information can now generally be divided into three segments—Traders, Information Users and Analytics Users, although some users constitute components from one or more of these categories.

Traders utilize data from financial markets such as NASDAQ, the American Stock Exchange, the New York Stock Exchange, the Tokyo Exchange, the London Exchange, the Chicago Options Board, and similar institutions that offer the ability to buy and sell stocks, options, futures, bonds, derivatives, and other financial instruments. The need for vast quantities of information is vital for making informed decisions and executing optimal transactions.

Thus given the importance of receiving this information over computer networks, an improved system and method for providing secure point-to-point solution for transparent multiplication of bandwidth over conventional communication channels is highly desirable.

For example, with the introduction of Nasdaq's next generation trading system SuperMontage, Nasdaq will offer market data users an unparalleled view into the activity, liquidity, and transparency of the Nasdaq market.

For example, currently Nasdaq provides each market participant's best-attributed quotation in each stock in which it makes a market. This system known as SuperMontage allows Nasdaq to accept multiple orders from each market participate in each stock for execution within SuperMontage. Nasdaq offers that data, with multiple levels of interest from individual market participants, through new data services.

Nasdaq provides this data on both an aggregated and a detailed basis for the top five price levels in SuperMontage. This data is currently offered through market data vendors and broker/dealer distributors via the following four entitlement packages:

- QuoteViewSM Each SuperMontage participant’s best bid and offer, as well as the best bid and offer available on SuperMontage.
- DepthViewSM The aggregate size, by price level, of all Nasdaq market participants’ attributed and unattributed quotations/orders that are in the top five price levels in SuperMontage.
- PowerViewSM Bundled QuoteView and DepthView.
- TotalViewSM PowerView plus all Nasdaq market participants’ attributed quotations/orders that are in the top five price levels in SuperMontage, in addition to the aggregate size of all unattributed quotes/orders at each of the top five price levels.

The NASDAQ SuperMontage trading system has been cited to be representative of trend for explosive growth in the quantity of information for all emergent and future trading and financial information distribution systems. Increases in processing power at the end user sites will allow traders, analysts, and all other interested parties to process substantially larger quantities of data in far shorter periods of time, increasing the demand substantially.

The ever increasing need for liquidity in the financials markets, coupled with the competitive pressures on reducing bid/ask spreads and instantaneous order matching/fulfillment, along the need for synchronized low latency data dissemination makes the need for the present invention ever more important. Depth of market information, required to achieve many of these goals requires orders of magnitude increases in Realtime trade information and bid/ask pricing (Best, 2nd best, . . . ).

A fundamental problem within the current art is the high cost of implementing, disseminating, and operating trading systems such as SuperMontage within the financial services industry. This is in large part due to the high bandwidth required to transfer the large quantities of data inherent in the operation of these systems. In addition the processing power required to store, transmit, route, and display the information further compounds cost and complexity.

This fundamental problem is in large part the result of utilizing multiple simultaneous T1 lines to transmit data. The data must be multiplexed into separate data streams, transmitted on separate data lines, and de-multiplexed and checked. Software solutions have high latency and cost while hardware solutions have even higher cost and complexity with somewhat lower latency. In addition the synchronization and data integrity checking require substantial cost, complexity, inherent unreliability, and latency. These and other limitations are solved by the present invention.

Further compounding this issue is a globalization and consolidation taking place amongst the various financial exchanges. The emergence of localized exchanges (ECNs—Electronic Computer Networks) coupled with the goal of 24 hour/7 day global trading will, in and of itself, drive another exponential increase in long haul international bandwidth requirements, while ECNs and other localized trading networks will similarly drive domestic bandwidth requirements. Clearly long haul links are orders of magnitude more expensive than domestic links and the value and significance of the present invention is at least proportionately more important.

Information users range from non-finance business professionals to curious stock market investors and tend to seek basic financial information and data. Analytical users on the
other hand, tend to be finance professionals who require more arcane financial information and utilize sophisticated analytical tools to manipulate and analyze data (e.g., for writing option contracts).

Historically, proprietary systems, such as Thomson, Bloomberg, Reuters and Bridge Information, have been the primary electronic source for financial information to both the informational and analytical users. These closed systems required dedicated telecommunications lines and often product-specific hardware and software. The most typical installations are land-based networking solutions such as T1, or ISDN, and satellite-based "wireless" solutions at speeds of 384 kbps.

Latency of financial data is critical to the execution of financial transactions. Indeed the more timely receipt of financial data from various sources including the New York Stock Exchange, American Stock Exchange, National Association of Securities Dealers (NASDAQ), Options Exchange, Commodities Exchanges, and Futures presents a fundamental advantage to those who trade. Latency is induced by the long time taken to receive and decrypt the compressed and encrypted data prior to transmission, along with the associated time to decrypt and decompress. Often current methods of encryption and compression take as much or substantially more time than the actual time to transmit the uncompressed, unencrypted data. Thus another problem within the current art is the latency induced by the act of encryption, compression, decryption, and decompression. The present invention overcomes this limitation within the current art.

Modern data compression algorithms suffer from poor compression, high latency, or both. Within the present art algorithms such as Lempel-Ziv, modified/embellished Lempel-Ziv, Binary Arithmetic, and Huffman coding are essentially generic algorithm having a varied effectiveness on different data types. Also small increases in compression to decompress and encrypt data prior to transmission, along with the associated time to decrypt and decompress. Often current methods of encryption and compression take as much or substantially more time than the actual time to transmit the uncompressed, unencrypted data. Thus another problem within the current art is the latency induced by the act of encryption, compression, decryption, and decompression. The present invention overcomes this limitation within the current art.

Modern data compression algorithms suffer from poor compression, high latency, or both. Within the present art algorithms such as Lempel-Ziv, modified/embellished Lempel-Ziv, Binary Arithmetic, and Huffman coding are essentially generic algorithm having a varied effectiveness on different data types. Also small increases in compression to decompress and encrypt data prior to transmission, along with the associated time to decrypt and decompress. Often current methods of encryption and compression take as much or substantially more time than the actual time to transmit the uncompressed, unencrypted data. Thus another problem within the current art is the latency induced by the act of encryption, compression, decryption, and decompression. The present invention overcomes this limitation within the current art.

Modern data compression algorithms suffer from poor compression, high latency, or both. Within the present art algorithms such as Lempel-Ziv, modified/embellished Lempel-Ziv, Binary Arithmetic, and Huffman coding are essentially generic algorithm having a varied effectiveness on different data types. Also small increases in compression to decompress and encrypt data prior to transmission, along with the associated time to decrypt and decompress. Often current methods of encryption and compression take as much or substantially more time than the actual time to transmit the uncompressed, unencrypted data. Thus another problem within the current art is the latency induced by the act of encryption, compression, decryption, and decompression. The present invention overcomes this limitation within the current art.

Modern data compression algorithms suffer from poor compression, high latency, or both. Within the present art algorithms such as Lempel-Ziv, modified/embellished Lempel-Ziv, Binary Arithmetic, and Huffman coding are essentially generic algorithm having a varied effectiveness on different data types. Also small increases in compression to decompress and encrypt data prior to transmission, along with the associated time to decrypt and decompress. Often current methods of encryption and compression take as much or substantially more time than the actual time to transmit the uncompressed, unencrypted data. Thus another problem within the current art is the latency induced by the act of encryption, compression, decryption, and decompression. The present invention overcomes this limitation within the current art.

Modern data compression algorithms suffer from poor compression, high latency, or both. Within the present art algorithms such as Lempel-Ziv, modified/embellished Lempel-Ziv, Binary Arithmetic, and Huffman coding are essentially generic algorithm having a varied effectiveness on different data types. Also small increases in compression to decompress and encrypt data prior to transmission, along with the associated time to decrypt and decompress. Often current methods of encryption and compression take as much or substantially more time than the actual time to transmit the uncompressed, unencrypted data. Thus another problem within the current art is the latency induced by the act of encryption, compression, decryption, and decompression. The present invention overcomes this limitation within the current art.

Modern data compression algorithms suffer from poor compression, high latency, or both. Within the present art algorithms such as Lempel-Ziv, modified/embellished Lempel-Ziv, Binary Arithmetic, and Huffman coding are essentially generic algorithm having a varied effectiveness on different data types. Also small increases in compression to decompress and encrypt data prior to transmission, along with the associated time to decrypt and decompress. Often current methods of encryption and compression take as much or substantially more time than the actual time to transmit the uncompressed, unencrypted data. Thus another problem within the current art is the latency induced by the act of encryption, compression, decryption, and decompression. The present invention overcomes this limitation within the current art.

Modern data compression algorithms suffer from poor compression, high latency, or both. Within the present art algorithms such as Lempel-Ziv, modified/embellished Lempel-Ziv, Binary Arithmetic, and Huffman coding are essentially generic algorithm having a varied effectiveness on different data types. Also small increases in compression to decompress and encrypt data prior to transmission, along with the associated time to decrypt and decompress. Often current methods of encryption and compression take as much or substantially more time than the actual time to transmit the uncompressed, unencrypted data. Thus another problem within the current art is the latency induced by the act of encryption, compression, decryption, and decompression. The present invention overcomes this limitation within the current art.
The volume of real-time data that is required to operate any major financial institution is staggering by comparison. To deal with this issue only critical account and transaction information is currently processed by co-locations in real-time. In fact, many institutions use batch mode processing where the transactions are only repeated “backed up” at the co-locations some time period later, up to 15 minutes or longer. The limitation of highly significant bandwidth and/or long delays with co-location processing and long latency times is solved by the present invention.

Thus given the importance of receiving financial information over computer networks, an improved system and method for providing secure point-to-point solution for transparent multiplication of bandwidth over conventional communication channels is highly desirable.

As previously stated, these and other limitations within the current art are solved by the present invention.

SUMMARY OF THE INVENTION

The present invention is directed to systems and methods for providing accelerated data transmission, and in particular to systems and methods of providing accelerated transmission of data, such as financial trading data, financial services data, financial analytical data, company background data, news, advertisements, and all other forms of information over a communications channel utilizing data compression and decompression to provide data transfer (secure or non-secure) and effectively increase the bandwidth of the communication channel and/or reduce the latency of data transmission. The present invention is universally applicable to all forms of data communication including broadcast type systems and bi-directional systems of any manner and any number of users or sites.

These and other aspects, features and advantages, of the present invention will become apparent from the following detailed description of preferred embodiments that is to be read in connection with the accompanying drawings.

BRIEF DESCRIPTION OF PREFERRED EMBODIMENTS

FIG. 1 is a block diagram of a system in which the present invention may be implemented for transmitting broadcast data;

FIG. 2 is a block diagram of a system and method for providing accelerated transmission of data over a communication channel according to an embodiment of the present invention;

FIG. 3 is a flow diagram illustrating a method for generating compression/decompression state machines according to one aspect of the present invention;

FIG. 4 is a diagram illustrating an exemplary encoding table structure according to the present invention, which may be generated using the process of FIG. 3.

FIG. 5 is a diagram of a system/method for providing content independent data compression, which may be implemented for providing accelerated data transmission according to the present invention; and

FIG. 6 is a diagram of a system/method for providing content independent data decompression, which may be implemented for providing accelerated data transmission according to the present invention.

DETAILED DESCRIPTION OF PREFERRED EMBODIMENTS

The present invention is directed to systems and methods for providing accelerated transmission of broadcast data, such as financial data and news feeds, over a communication channel using data compression and decompression to provide secure transmission and transparent multiplication of communication bandwidth, as well as reduce the latency associated with data transmission of conventional systems.


In general, the term “accelerated” data transmission refers to a process of receiving a data stream for transmission over a communication channel, compressing the broadcast data stream in real-time (wherein the term “real time” as used herein collectively refers to substantially real time, or at real time, or greater than real time) at a compression rate that increases the effective bandwidth of the communication channel, and transmitting the compressed broadcast data over the communication channel. The effective increase in bandwidth and reduction of latency of the communication channel is achieved by virtue of the fast than real-time, real-time, near real time, compression of a received data stream prior to transmission.

For instance, assume that the communication channel has a bandwidth of “B” megabytes per second. If a data transmission controller is capable of compressing (in substantially real time, real time, or faster than real time) an input data stream with an average compression rate of 3:1, then data can be transmitted over the communication channel at an effective rate of up to 3*B megabytes per second, thereby effectively increasing the bandwidth of the communication channel by a factor of three.

Further, when the receiver is capable decompressing (in substantially real time, real time, or faster than real time) the compressed data stream at a rate approximately equal to the compression rate, the point-to-point transmission rate between the transmitter and receiver is transparently increased. Advantageously, accelerated data transmission can mitigate the traditional bottleneck associated with, e.g., local and network data transmission.

If the compression and decompression are accomplished in real-time or faster, the compressed, transmitted and decompressed data is available before the receipt of an equivalent uncompressed stream. The “acceleration” of data transmission over the communication channel is achieved when the total time for compression, transmission, and decompression, is less than the total time for transmitting the data in uncompressed form. The fundamental operating principle of data acceleration is governed by the following relationship:
As stated in Equation [1] above, if the time to compress, transmit, and decompress a data packet is less than the time to transmit the data in original format, then the delivery of the data is said to be accelerated.

In the above relationship, a fundamental premise is that all information is preferably fully preserved. As such, lossless data compression is preferably applied. While this disclosure is directed to transmission of data in financial networks, for example, the concept of “acceleration” may be applied to the storage and retrieval of data to any memory or storage device using the compression methods disclosed in the above-incorporated U.S. Pat. Nos. 6,195,024 and 6,309,424, and U.S. application Ser. No. 10/016,355, and the storage acceleration techniques disclosed in the above-incorporated application Ser. No. 09/481,243 and 09/266,394.

Returning to Equation [1], data acceleration depends on several factors including the creation of compression and decompression algorithms that are both effective (achieve good compression ratios) and efficient (operate rapidly with a minimum of computing processor and memory resources). Rearranging the terms of Equation [1] we can see that the total time to transmit data in an “accelerated” form (transmit compressed data is the sum of the original time to transmit the data in an uncompressed fashion divided by the actual compression ratio achieved, plus the time to compress and decompress the data.

\[
T_{\text{Transmit Accelerated}} = \frac{T_{\text{Transmit w/o Compression}}}{CR} + T_{\text{Compress}} + T_{\text{Decompress}}
\]

Where:

\(CR\) = Compression Ratio

Thus the latency reduction is the simple arithmetic difference between the time to transmit the original data minus the total time to transmit the accelerated data (per Equation 2 above), resulting in:

\[
\frac{T_{\text{Latency Reduction}}}{T_{\text{Transmit Accelerated}}} = \frac{T_{\text{Transmit w/o Compression}} - (T_{\text{Transmit Accelerated}})}{T_{\text{Transmit Accelerated}}}
\]

And finally the achieved “Acceleration Ratio” is defined as:

\[
\frac{T_{\text{Transmit Accelerated}}}{T_{\text{Transmit Accelerated}}}
\]

A number of interesting observations come to light from these relatively simple algebraic relationships and are implemented within the present invention:

Compression Ratio The present inventions achieve a consistent reduction in latency. The data compression ratio is substantial and repeatable on each data packet.
parent increase in the transmission bandwidth of the various communication channels used, without requiring modification of existing network infrastructure.

Referring now to FIG. 2, a block diagram illustrates a system/method for providing accelerated transmission of data according to one embodiment of the present invention. More specifically, FIG. 2 illustrates embodiments of a broadcast data server (transmitter) and client system (receiver) for implementing accelerated transmission and real-time processing of broadcast data. Each state machine (comprising one or more different broadcast types) is processed by data server 21 prior to transmission to client 23 over a communication channel 24. The data server 22 utilizes a processor 25 (e.g., microprocessor, digital signal processor, etc.) for executing one or more decompression algorithms 26 for decompressing (in real-time) the broadcast data 21 prior to transmission. In preferred embodiments, compression is achieved using Huffman or Arithmetic encoding, wherein one or more state machines 27-27n are constructed based on a-priori knowledge of the structure and content of one or more given broadcast and data feeds.

As explained in further detail below, each state machine 27-27n comprises a set of compression tables that comprise information for encoding the next character (text, integer, etc.) or sequence of characters in the broadcast data feed, as well as pointers which point to the next state (encoding table) based on the character or character sequence. As explained in greater detail below, a skeleton for each state machine 27-27n (nodes and pointers) is preferably built by finding sequences of characters (n-tuples) that frequently appear in an input. Once a skeleton has been determined, a large set of data is processed through the system and counts are kept of character n-tuples for each state. These counts are then used to construct the compression tables associated with the state machine to provide statistical compression. The compressed data is transmitted over the communication channel 24 via a communication stack using any suitable protocol (e.g., RTP (real-time protocol) using RTCP (real-time control protocol), TCP/IP, UDP, or any real-time streaming protocol with suitable control mechanisms).

Similarly, the client 23 comprises a processor 30 for executing one or more decompression algorithms 31. Depending on the data feed type, one of a plurality of decomposition state machines 32-32n is used to decompress the compressed data stream received by the client 23 via communication stack 34. Each state machine 32-32n comprises a set of decompression tables 33-33n that comprise information for decoding the incoming character (or symbol) or sequence of symbols in the compressed broadcast data feed, as well as pointers which point to the next state based on the symbol or sequence. For each compression state machine 27-27n in the data server, a corresponding decompression state machine 32-32n is needed in the client 23 to decompress the associated data stream.

Advantageously, a compression/decompression scheme according to the present invention using Huffman or Arithmetic encoding provides secure transmission via de facto or virtual "encryption" in a real-time environment. Indeed, virtual encryption is achieved by virtue of the fast, yet complex, data compression using Huffman tree, for example, without necessarily requiring actual encryption of the compressed data and decryption of the compressed data. Because of the time-sensitive nature of the market data, and the ever-changing and data-dependent nature of the arithmetic scheme, decryption is virtually impractical, or so complex and useless as to render the data worthless upon eventual decoding.

However, data compression using Huffman or Arithmetic encoding yields encoded data that is very difficult to decode than current encryption schemes such as plain text or simple bit shuffling codes as currently used by broadcast service providers. An attacker must have the compression model and the tables used to compress the data stream to be able to obtain useful information from it. Thus, at one level of security, the client-side decompression tables are preferably stored in encrypted form and are decrypted on being loaded into the processor 30 (e.g., general purpose processor, DSP, etc.) using an encryption/decryption key that is validated for a subscribing user. In this manner, a client will be unable to use the tables on other processors or sites or after terminating a service contract.

Since Huffman compression uses the same bit code for a character each time it appears in a given context, an attacker with a very large data set of compressed and uncompressed data could possibly reconstruct the tables, assuming the overall model were known. Arithmetic compression, on the other hand, generates different bit patterns for the same character in the same context depending on surrounding characters. Arithmetic encoding provides at least an order of magnitude more difficult to recover the tables from the compressed and uncompressed data streams.

The following is a detailed discussion of a compression scheme using Huffman or Arithmetic encoding for providing accelerated transmission of broadcast data according to one aspect of the present invention. It is to be appreciated that the present invention is applicable with any data stream whose statistical regularity may be captured and represented in a state machine model. For example, the present invention applies to packetized data streams, in which the packets are limited in type format and content.

In one embodiment using Huffman or Arithmetic encoding, each character or character sequence is encoded (converted to a binary code) based on the frequency of character or character sequence in a given "context". For a given context, frequently appearing characters are encoded with fewer bits while infrequently appearing characters are encoded with more bits. High compression ratios are obtained if the frequency distribution of characters in most contexts is highly skewed with few frequently appearing characters and many characters seldomly (or never) appear.

Referring now to FIG. 3, a flow diagram illustrates a method for generating compression/decompression state machines according to one aspect of the present invention. The "context" in which a character (or character sequence) is encoded in a given broadcast stream is based on a "global state" that represents packet type and large-scale structure and the previous few characters. The first step in building a compression scheme involves selecting a global state system based on the packet structure of the broadcast model (step 40). More specifically, a global state system is constructed based on a priori knowledge of the data stream model, e.g., the packet type frequency and structure of the broadcast model. By way of example, one model for financial data may comprise four global states representing: a beginning of packet, an options packet, a NYSE (New York Stock Exchange) packet and some other packet type. Further, additional codes may be added to the encoding tables to indicate global state transitions (e.g., for an end of packet code in the broadcast model). If there is internal structure to packets, such as a header with different statistics than the body, additional global states could be added.

Once a global state system is selected, training samples from an associated data stream are passed through the global model to acquire counts of frequencies of the occurrence of
n-tuple character sequences ending in each of the model states (step 41). In a preferred embodiment, the n-tuples comprise character sequences having 1, 2 and 3 characters. Using the acquired counts, sub-states (or "local states") of the pre-defined global states are constructed based on previous characters in the data stream. A local state may depend on either none, 1, 2, or 3 (or more) previous characters in the stream. To provide a practical limitation, a predetermined count threshold is preferably applied to the count data (step 42) and only those sequences that occur more often than the count threshold are added as local states (step 43). For example, if a three-character sequence does not occur sufficiently frequently, the count for the last two characters is tested, etc.

It is to be understood that any character sequence length "n" may be implemented depending on the application. The longer the allowed character sequence, the more memory is needed to store the encoding tables and/or the lower the count threshold should be set.

As samples of the data are passed through the state model, character (and transition code) counts for each context are accumulated. These counts are used to build the Huffman or Arithmetic coding tables. The construction of the global and local models is an iterative process. The count threshold for forming local states can be adjusted depending on the application. For instance, a larger threshold will result in less local states but less compression as well. Further, a comparison of statistics in local or global states may suggest adding or deleting global states.

The construction of the global model requires knowledge of the data stream packet structure. The construction of the local states is automatic (once the threshold is set).

FIG. 4 is a diagram of an exemplary state diagram (or encoding table structure) according to the present invention, which may be generated using the process of FIG. 3.

As noted above with reference to FIGS. 1 and 2, a compression scheme according to the present invention may be implemented in any system to provide accelerated data transmission to multiple client site systems. Preferably, the client site systems may connect at any time, so minimal immediate history may be used (since a newly connected site must be able to pick up quickly). A system according to an embodiment of the present invention uses statistical compression (Huffman or Arithmetic coding) using fixed (or adaptive) tables based on the statistics of a data feed sample. As noted above, it has been determined that the statistical compression schemes described herein are well adapted for use with structured data streams having repetitive data content (e.g., stock symbols and quotes, etc.) to provide fast and efficient data compression/decompression.

The following discussion provides further details regarding the preparation of statistical-based encoding tables and their use for compression/decompression according to the present invention. During a data compression process, the selection of which encoding table to use for compression is preferably based on up to n (where n is preferably equal to 3) preceding characters of the message. In an exemplary broadcast model tested by the present inventors, a data stream comprises messages that begin with an ID code in the range 0-31 with the remainder of the message being characters in the range 32-127. It was found that approximately half of the messages in a given sample began with ID code 0x0c and half of the remainder began with ID code 0xf. Thus, a separate encoding table is preferably used for a message ID code. Further, separate table sets are used for messages beginning with 0x0c and with 0xf, with the remaining messages lumped together in another table.

Each table has an additional termination code. The termination code in a "start table" indicates the end of a compressed block. The termination code in all other tables indicates the end of the message. Thus, the start table comprises 35 entries and all other tables have 37 entries.

Using one table for each 3-character context would require prohibitive amounts of memory. For example, a complete one-character context would require 334*3*3*256 = 2,592,000 bytes. Then, a complete two-character context would require 324*3*3*256 = 3,145,728 bytes. And finally, a complete three-character context would require 324*3*3*256 = 3,048,516 bytes. Preferably, as described above, the application of a count threshold at each context size reduces the amount of tables. Only when a context occurs at greater than the threshold rate in the sample will a table be created for that context.

Each table entry includes a link to the next table to be used. For instance, in an "abc" context table, the entry for next character "d" would point to the "bed" table, if such table was created. If such table was not created, the entry for next character "d" would point to the "cd" table, if such table existed. If no "cd" table exists, the "d" table would be used and if that fails, a base table for the message type would be used.

For a client site system to pick up the broadcast feed at any time, clearly identifiable synchronization points are preferably included in the compressed data stream. In a preferred embodiment, data is compressed in blocks with each block comprising some number of complete messages. Preferably, each compressed block ends with at least four bytes with each bit being logic 1 and no interior point in the compressed block will comprise 32 consecutive 1 bits. The compressed block preferably begins with two bytes giving the decompressed size of the block shifted to guarantee that the first byte of the compressed block is not all 1's. Thus, to achieve synchronization, the client site system can scan the input compressed data stream for 4 bytes of 0xff, wherein the next byte not equal to 0xff is deemed the start of a compressed block. In other words, the receiver will accumulate the compressed data until at least a sequence of 4 bytes each having a value of 0xff is detected in the input stream, at which point decompression will commence on the compressed input stream.

In another embodiment of the present invention, if a compressed block is more than 6 bytes longer than the uncompressed data, the data block is transmitted uncompressed preceded by the shifted two-byte count with the high bit set and trailed by 4 bytes of 0xff.

The following is a discussion of a method for preparing Huffman Tables according to one aspect of the present invention. The Huffman codes generated by a conventional optimal algorithm have been modified in various ways in accordance with the present invention. First, in order that there not be 32 consecutive one bits in the data stream except at the end of a compression block, a termination code in each table comprises all 1 bits.

Further, to reduce space required for decompression tables, and ensure no sequence of 32 1 bits, each code is preferably decoded as follows:

a) The first 7 bits are used to index into a table. If the character code is no more than 7 bits, it can be read directly;

b) otherwise, some number N of initial bits is discarded and the next 7 bits are used to index a second table to find the character.

Based on these steps, preferably, no character code can use more than 14 bits and all codes of more than 7 bits must fit into the code space of the N initial bits. If N is 3, for instance, then no code can use more than 10 bits.
To achieve this, the code space required for all optimal codes of more than 7 bits is first determined, following by a determining the initial offset $N$. Every code comprising more than $N+7$ bits is preferably shortened, and other codes are lengthened to balance the code tree. It is possible that this may cause the code space for codes over 7 bits to increase so that $N$ may need to be decreased. Preferably, this process is performed in a manner that causes minimal reduction in the efficiency of the codes.

The above modifications to conventional optimal algorithm yields codes in which no non-termination code ends in more than 71 bits, no non-termination code begins with more than 61 bits, no termination code is more than 141 bits and no non-termination packet start code begins with more than 51 bits. Thus, in the middle of a packet, a sequence of no more than 13 bits of logic 1 can occur, while, at the end of a packet, a sequence of no more than 26 bits of logic 1 can occur.

In another embodiment of the present invention, Arithmetic compression can be used instead of Huffman encoding. The tables for Arithmetic encoding are preferably constructed such that a sequence of 32 bits of logic 1 will not occur in the interior of a message (which is important for a random sign-on in the middle of the stream).

Arithmetic compression provides an advantage of about 6% better compression than Huffman and uses half as much memory for tables, which allows the number of tables to be increased. Indeed, the addition of more tables and/or another level of tables yields more efficient compression. Although Arithmetic compression may take about 6 times as long as Huffman, this can certainly be improved by flattening the subroutine call tree (wherein there is a subroutine call for each output bit.)

In summary, a compression scheme according to one aspect of the invention utilizes a state machine, wherein in each state, there is a compression/decompression table comprising information on how to encode/decode the next character, as well as pointers that indicated which state to go to based on that character. A skeleton of the state machine (nodes and pointers) is preferably built by finding sequences of characters that appear often in the input. Once the skeleton has been determined, a large set of data is run through the system and counts are kept of characters seen in each state. These counts are then used to construct the encode/decode tables for the statistical compression.

Other approaches may be used to build the skeleton of the state machine. A very large fraction of the traffic on a certain feed consists of messages in the digital data feed format, which is fairly constrained. It may be possible to build by hand a skeleton that takes into account this format. For instance, capital letters only appear in the symbol name at the beginning. This long-range context information can be represented with our current approach. Once a basic skeleton is in place, the structure could be extended for sequences that occur frequently.

The above-described statistical compression schemes provide content-dependent compression and decompression. In other words, for a given data stream, the above schemes are preferably structured based on the data model associated with the given stream. It is to be appreciated, however, that other compression schemes may be employed for providing accelerated data transmission in accordance with the present invention for providing effectively increased communication bandwidth and/or reduction in latency. For instance, in another embodiment of the present invention, the data compression/decompression techniques disclosed in the above-incorporated U.S. Pat. No. 6,195,024, entitled "Content Independent Data Compression Method and System," may be used in addition to, or in lieu of, the statistical based compression schemes described above.

In general, a content-independent data compression system is a data compression system that provides an optimal compression ratio for an encoded stream regardless of the data content of the input data stream. A content-independent data compression method generally comprises the steps of compressing an input data stream, which comprises a plurality of disparate data types, using a plurality of different encoders. In other words, each encoder compresses the input data stream and outputs blocks of compressed data. An encoded data stream is then generated by selectively combining compressed data blocks output from the encoders based on compression ratios obtained by the encoders. Because a multiplicity of different data types may be present within a given input data stream, or data block, to it is often difficult and/or impractical to predict the level of compression that will be achieved by any one encoding technique. Indeed, rather than having to first identify the different data types (e.g., ASCII, image data, multimedia data, signed and unsigned integers, pointers, etc.) comprising an input data stream and selecting a data encoding technique that yields the highest compression ratio for each of the identified data types, content-independent data compression advantageously applies the input data stream to each of a plurality of different encoders to, in effect, generate a plurality of encoded data streams. The plurality of encoders are preferably selected based on their ability to effectively encode different types of input data. Ultimately, the final compressed data stream is generated by selectively combining blocks of the compressed streams output from the plurality of encoders. Thus, the resulting compressed output stream will achieve the greatest possible compression, regardless of the data content.

In accordance with another embodiment of the present invention, a compression system may employ both a content-dependent scheme and a content-independent scheme, such as disclosed in the above-incorporated application Ser. No. 10/016,355. In this embodiment, the content-dependent scheme is used as the primary compression/decompression system and the content-independent scheme is used in place of, or in conjunction with, the content dependent scheme, when periodically checked “compression factor” meets a predetermined threshold. For instance, the compression factor may comprise a compression ratio, wherein the compression scheme will be modified when the compression ratio falls below a certain threshold. Further, the “compression factor” may comprise the latency of data transmission, wherein the data compression scheme with be modified when the latency of data transmission exceeds a predetermined threshold.

Indeed, as explained above, the efficiency of the content-dependent compression/decompression schemes described herein is achieved, e.g., by virtue of the fact that the encoding tables are based on, and specifically designed for, the known data model. However, in situations where the data model is may be modified, the efficiency of the content-dependent scheme may be adversely affected, thereby possibly resulting in a reduction in compression efficiency and/or an increase in the overall latency of data transmission. In such a situation, as a backup system, the data compression controller can switch to a content-independent scheme that provides improved compression efficiency and reduction in latency as compared to the primary content-dependent scheme.

In yet another embodiment of the present invention, when the efficiency of a content-dependent scheme falls below a predetermined threshold based on, e.g., a change in the data structure of the data stream, the present invention preferably
comprises an automatic mechanism to adaptively modify the encoding tables to generate optimal encoding tables (using the process described above with reference to FIG. 3).

FIG. 5 is a detailed block diagram illustrates an exemplary content-independent data compression system 110 that may be employed herein. Details of this data compression system are provided in U.S. Pat. No. 6,195,024, which is fully incorporated herein by reference. In this embodiment, the data compression system 110 accepts data blocks from an input data stream and stores the input data block in an input buffer or cache 115. It is to be understood that the system processes the input data stream in data blocks that may range in size from individual bits through complete files or collections of multiple files. Additionally, the input data block size may be fixed or variable. A counter 120 counts or otherwise enumerates the size of input data block in any convenient units including bits, bytes, words, and double words. It should be noted that the input buffer 115 and counter 120 are not required elements of the present invention. The input data buffer 115 may be provided for buffering the input data stream in order to output an uncompressed data stream in the event that, as discussed in further detail below, every encoder fails to achieve a level of compression that exceeds an a priori specified minimum compression ratio threshold.

Data compression is performed by an encoder module 125 that may comprise a set of encoders E1, E2, E3 . . . En. The encoder set E1, E2, E3 . . . En may include any number “n” (where n may=1) of those lossless encoding techniques currently well known within the art such as run length, Huffman, Lempel-Ziv Dictionary Compression, arithmetic coding, data compaction, and data null suppression. It is to be understood that the encoding techniques are selected based upon their ability to effectively encode different types of input data. It is to be appreciated that a full complement of encoders are preferably selected to provide a broad coverage of existing and future data types.

The encoder module 125 successively receives as input each of the buffered input data blocks (or unbuffered input data blocks from the counter module 120). Data compression is performed by the encoder module 125 wherein each of the encoders E1 . . . En processes a given input data block and outputs a corresponding set of encoded data blocks. It is to be appreciated that the system affords a user the option to enable/disable any one or more of the encoders E1 . . . En prior to operation. As is understood by those skilled in the art, such feature allows the user to tailor the operation of the data compression system for specific applications. It is to be further appreciated that the encoding process, may be performed either in parallel or sequentially. In particular, the encoders E1 through En of encoder module 125 may operate in parallel (i.e., simultaneously processing a given input data block by utilizing task multiplexing on a single central processor, via dedicated hardware, by executing on a plurality of processor or dedicated hardware systems, or any combination thereof). In addition, encoders E1 through En may operate sequentially on a given unbuffered or buffered input data block. This process is intended to eliminate the complexity and additional processing overhead associated with multiplexing concurrent encoding techniques on a single central processor and/or dedicated hardware, set of central processors and/or dedicated hardware, or any achievable combination. It is to be further appreciated that encoders of the identical type may be applied in parallel to enhance encoding speed. For instance, encoder E1 may comprise two parallel Huffman encoders for parallel processing of an input data block.

A buffer/counter module 130 is operatively connected to the encoder module 125 for buffering and counting the size of each of the encoded data blocks output from encoder module 125. Specifically, the buffer/counter 130 comprises a plurality of buffer/counters BC1, BC2, BC3 . . . BCn, each operatively associated with a corresponding one of the encoders E1 . . . En. A compression ratio module 135, operatively connected to the output buffer/counter 130, determines the compression ratio obtained for each of the enabled encoders E1 . . . En by taking the ratio of the size of the input data block to the size of the output data block stored in the corresponding buffer/counter BC1 . . . BCn. In addition, the compression ratio module 135 compares each compression ratio with an a priori-specified compression ratio threshold limit to determine if at least one of the encoded data blocks output from the enabled encoders E1 . . . En achieves a compression that exceeds an a priori-specified threshold. As is understood by those skilled in the art, the threshold limit may be specified as any value inclusive of data expansion, no data compression or expansion, or any arbitrarily desired compression limit. A description module 138, operatively coupled to the compression ratio module 135, appends a corresponding compression type descriptor to each encoded data block which is selected for output as to indicate the type of compression format of the encoded data block. A data compression type descriptor is defined as any recognizable data token or descriptor that indicates which data encoding technique has been applied to the data. It is to be understood that, since encoders of the identical type may be applied in parallel to enhance encoding speed (as discussed above), the data compression type descriptor identifies the corresponding encoding technique applied to the encoded data block, not necessarily the specific encoder. The encoded data block having the greatest compression ratio along with its corresponding data compression type descriptor is then output for subsequent data processing or transmittal. If there are no encoded data blocks having a compression ratio that exceeds the compression ratio threshold limit, then the original unencoded input data block is selected for output and a null data compression type descriptor is appended thereto. A null data compression type descriptor is defined as any recognizable data token or descriptor that indicates no data encoding has been applied to the input data block. Accordingly, the unencoded input data block with its corresponding null data compression type descriptor is then output for subsequent data processing or transmittal.

Again, it is to be understood that the embodiment of the data compression engine of FIG. 5 is exemplary of a preferred compression system which may be implemented in the present invention, and that other compression systems and methods known to those skilled in the art may be employed for providing accelerated data transmission in accordance with the teachings herein. Indeed, in another embodiment of the compression system disclosed in the above-incorporated U.S. Pat. No. 6,195,024, a timer is included to measure the time elapsed during the encoding process against an a priori-specified time limit. When the time limit expires, only the data output from those encoders (in the encoder module 125) that have completed the present encoding cycle are compared to determine the encoded data with the highest compression ratio. The time limit ensures that the real-time or pseudo real-time nature of the data encoding is preserved. In addition, the results from each encoder in the encoder module 125 may be buffered to allow additional encoders to be sequentially applied to the output of the previous encoder, yielding a more optimal lossless data compression ratio. Such techniques are discussed in greater detail in the above-incorporated U.S. Pat. No. 6,195,024.

Referring now to FIG. 6, a detailed block diagram illustrates an exemplary decompression system that may be
employed herein or accelerated data transmission as disclosed in the above-incorporated U.S. Pat. No. 6,195,024. In this embodiment, the data compression engine 180 accepts compressed data blocks received over a communication channel. The decompression system processes the input data stream in data blocks that may range in size from individual bits through complete files or collections of multiple files. Additionally, the input data block size may be fixed or variable.

The data compression engine 180 comprises an input buffer 155 that receives as input an uncompressed or compressed data stream comprising one or more data blocks. The data blocks may range in size from individual bits through complete files or collections of multiple files. Additionally, the data block size may be fixed or variable. The input data buffer 55 is preferably included (not required) to provide storage of input data for various hardware implementations. A descriptor extraction module 160 receives the buffered (or unbuffered) input data block and then parses, lexically, syntactically, or otherwise analyzes the input data block using methods known by those skilled in the art to extract the data compression type descriptor associated with the data block.

The data compression type descriptor may possess values corresponding to null (no encoding applied), a single applied encoding technique, or multiple encoding techniques applied in a specific or random order (in accordance with the data compression system embodiments and methods discussed above).

A decoder module 165 includes one or more decoders D1 . . . Dn for decoding the input data block using a decoder, set of decoders, or a sequential set of decoders corresponding to the extracted compression type descriptor. The decoders D1 . . . Dn may include those lossless encoding techniques currently well known within the art, including: run length, Huffman, Lempel-Ziv Dictionary Compression, arithmetic coding, data compaction, and data null suppression. Decoding techniques are selected based upon their ability to effectively decode the various different types of encoded input data generated by the data compression systems described above or originating from any other desired source.

As with the data compression systems discussed in the above-incorporated U.S. Pat. No. 6,195,024, the decoder module 165 may include multiple decoders of the same type applied in parallel so as to reduce the data decoding time. An output buffer or cache 170 may be included for buffering the decoded data block output from the decoder module 165. The output buffer 70 then provides data to the output data stream. It is to be appreciated by those skilled in the art that the data compression system 180 may also include an input data counter and output data counter operatively coupled to the input and output, respectively, of the decoder module 165. In this manner, the compressed and corresponding decompressed data block may be counted to ensure that sufficient decompression is obtained for the input data block.

Again, it is to be understood that the embodiment of the data decompression system 180 of FIG. 6 is exemplary of a preferred decompression system and method which may be implemented in the present invention, and that other data decompression systems and methods known to those skilled in the art may be employed for providing accelerated data transmission in accordance with the teachings herein.

It is to be appreciated that a data transmission acceleration system according to the present invention offers a business model by which market data vendors and users in the financial information services industry can receive various benefits. For example, the present invention affords transparent multiplication of bandwidth with minimal latency. Experiments have shown that increased bandwidth of up to 3 times can be achieved with minimal latency. Furthermore, proprietary hardware, including chip and board designs, as well as custom embedded and application software and algorithms associated with accelerated data transmission provide a cost-effective solution that can be seamlessly integrated with existing products and infrastructure. Moreover, the data acceleration through “real-time” compression and decompression affords a dramatic reduction in ongoing bandwidth costs. Further, the present invention provides mechanism to differentiate data feeds from other vendors via enriched content or quantity of the data feed.

In addition, a data compression scheme according to the present invention provides dramatically more secure and encrypted feed from current levels, thus, providing the ability to employ a secure and accelerated virtual private network over the Internet for authorized subscribers or clients with proprietary hardware and software installed.

Moreover, the present invention offers the ability to reduce a client’s ongoing monthly bandwidth costs as an incentive to subscribe to a vendor’s data feed service.

The present invention is readily extensible for use on a global computer network such as the Internet. This is significant since it creates a virtual private network and is important for the market data vendors and others due to its reduced cost in closed network/bandwidth solutions. In effect, the data vendors get to “ride for free” over the world’s infrastructure, while still providing the same (and enhanced) services to their customers.

In yet another embodiment of the present invention a highly optimized data compression and decompression system is utilized to accelerate data transfers for data transmission feeds. This type of compression achieves very high compression ratios (over 10:1) on financial data feeds such as Nasdaq Quote Dissemination Service Data (NQDS) and SuperMontage Services. The information utilized to develop the methods described herein for Nasdaq has been garnered solely from public knowledge through specifications available from the Nasdaq Trader and Nasdaq websites. The techniques disclosed herein are broadly applicable to all financial data feeds and information or trading services.

Three types of encoding are utilized dependent upon the data feeds and packet structure. In the event that a data field is unrecognizable then content independent data compression is preferably used, as previously discussed herein.

Variable Length Encoding

The basic unit of the compression process is the code. Each message field or set of set of fields being compressed together is assigned one or more codes in the range 0 . . . N. The code for a single character field is the ASCII value of the field minus 32 since all characters are in the range 32 to 127.

For various reasons, additional (escape) codes may be added to those for field values. For example, the category field has an escape code to indicate the end of a block and another to allow encoding of messages, which do not match the current format.

A basic technique used is variable rate encoding of symbols. In this approach, different amounts of the output bits are used to transmit the codes within a set. Higher frequency codes use less output bits while lower frequency codes use more output bits. Thus the average number of bits is reduced. Two methods of accomplishing this are used. The faster method uses a variant of Huffman coding while the slower method uses a form of Arithmetic coding.

In Huffman coding, each code is represent by an integral number of bits. The code sizes are computed using the stan-
dard algorithm and then (possibly) adjusted to facilitate table driven decoding (for instance, limiting codes to at most 16 bits). In the table driven decoding method used, there is a 256 element base table and two 256 element forwarding table. At each step, the next 8 bits of the input are used to index into the base table. If the code is represented in no more than 8 bits, it will be found directly. Otherwise, there will be a forwarding entry indicating which forwarding table to use and how many input bits to discard before using the next 8 bits as an index. The entry determining the result also indicates how many bits of the input to discard before processing the next field.

In arithmetic coding, the message is essentially represented as the (approximate) product of fractions with base 16384. The numerators of the fractions are proportional to the frequencies with which the codes appear in the training data. The number of output bits used to represent a code is the base 2 logarithm of the fraction. Thus codes which appear in almost all messages may be represented with fractions of a bit.

Single Character Codes
For arithmetic coding, all single character fields are encoded as the ASCII value -32 + the number of escape codes. For Huffman coding, certain single character message fields are encoded in the same way. These include:

- MM Trade Desk
- Quote Condition
- Inside Indicator
- Quote Type

Other single character fields, which have a single value that occurs most of the time, are encoded as multiple character fields (see next). In Huffman coding the smallest representation for a code is 1 bit. By combining these fields, we may encode the most common combination of values in 1 bit for the whole set. These include:

- Message Category + Message Type
- Session Identifier + Originator ID
- PMM + Bid Price Denominator + Ask Price Denominator (Quotes)
- Inside Status + Inside Type
- Inside Bid Denominator + Inside Bid MC
- Inside Ask Denominator + Inside Ask MC
- UPC Indicator + Short Sale Bid Tick
- Market of Origin + Reason

Small Set Multiple Character Codes
Multiple character fields with a small number of common values and certain combinations of single character fields are encoded based on the frequency of the combinations. A list of common combinations is used together with an escape code. The common combinations are encoded using the corresponding code. All other combinations are encoded by the escape code followed by the (7 bit) ASCII values for the characters in the combination. The fields include the field sets above for Huffman coding as well as the following for both approaches:

- Retransmission Requester
- MM Location
- Currency Code

Large Set Multiple Character Codes
Multiple character alphabetic or alphanumeric fields for which a large number of values are possible (Issue Symbol and MMID/MPID) are encoded as follows. Trailing spaces for Issue Symbols are deleted. Then the result is encoded using:

- Variable length codes for a list of the most common values together with escapes for the possible lengths of values not in the list.
- A table for the first character of the field.
- A table for subsequent characters in the field.

If a value is in the list of most common values, it is encoded with the corresponding code. Otherwise, the value is encoded by sending the escape code corresponding to the (truncated) length of the value, followed by the code for the first character, which is then followed by codes for the remaining characters.

Absolute Numeric Values
Numeric fields are transmitted by sending a variable length code for the number of significant bits of the value followed by the bits of the value other than the most significant bit (which is implicitly 1). For example, 27 (a 5 bit value) would be represented by the code for a 5 bit value followed by the 4 least significant bits (11). These fields include:

- Short Bid Price
- Long Bid Price
- Short Bid Size
- Long Bid Size
- Short Ask Size
- Long Ask Size
- Short Inside Bid Size
- Long Inside Bid Size
- Short Inside Ask Size
- Long Inside Ask Size

Relative Numeric Values
Numeric fields expected to be close to the value of numeric values occurring earlier in the message are encoded by encoding the difference between the new value and the base value as follows:

If the difference in non-negative and less than ⅛ of the base value, the difference is encoded by sending a variable length code for the number of significant bits of the difference followed by the bits of the difference other than the most significant bit (which is implicitly 1). Otherwise, the new value is encoded by sending a variable length code for the number of significant bits of the value followed by the bits of the value other than the most significant bit (which is implicitly 1). The difference significant bit codes and the value significant bit codes are mutually exclusive. The following fields are encoded using the difference compared to the field in parentheses:

- Short Ask Price (Bid Price)
- Long Ask Price (Bid Price)
- Short Inside Bid Price (Bid Price)
- Long Inside Bid Price (Bid Price)
- Short Inside Ask Price (Inside Bid Price)
- Long Inside Bid Price (Bid Price)
- Long Inside Ask Price (Inside Bid Price)

Differences
Both time and Message Sequence Number are encoded as the difference between the new value and a previous value within the compression block. This is transmitted using a code giving the sign of the difference and the number of significant bits in the absolute value of the difference followed by the bits of the absolute value other than the first.

Date
Each message within a compression block is expected to have the same date. The base date is transmitted at the beginning of the block as 7 bits of year, 4 bits of month and 5 bits of day of the month. If the date of a message is different than
that of the block, a special escape code is used in place of the encoding of the sequence number and time. This is followed by the year, month and day as above followed by the time in seconds (17 bits) and the sequence number (24 bits).

Message Sequence Number and Time

Message time is converted to seconds after midnight. For all retransmitted messages (Retransmission Requester not “O”), the time is transmitted as a 17-bit value followed by the Message Sequence Number transmitted as a 24-bit value. If the date is not the same as the block date, a time value of 0x1f1ff is used as an escape code.

For the first original transmission message in a block, the Message Sequence Number and time are transmitted in the same way.

For arithmetic coding of all other original transmission messages in a block, the Message Sequence Number is transmitted as the encoded change from the Message Sequence Number of the preceding original transmission message. Similarly, the time of all other original transmission messages is encoded as the difference from the previous original transmission message. An escape code in the Message Sequence Number Difference Table is used to indicate that the date is not the same as the block date.

Since almost all sequence number changes are 1 and almost all time changes are 0, we can save a bit (while Huffman coding) by encoding time and sequence number together.

This is done as follows: The most common values for both time and sequence number changes are 0 and 1 so there are three possibilities for each: 0, 1 and something else. Together this yields nine possibilities. An escape code is added to indicate a date different from the block date. To transmit the sequence number and time, the code corresponding the correct combination is first sent and then, if the time difference is not 0 or 1, the difference code for time followed by the difference code for sequence number (if required) is sent.

Unexpected Message Types

For administrative messages or non-control messages of unexpected category or type, the body of the message (the part after the header) is encoded as a 10-bit length field followed by the characters of the body encoded as 7-bit ASCII. Any Quotation message with an unexpected Inside Indicator value will have the remainder of the message encoded similarly.

Termination Code and Error Detection

Each compression block is terminated by an escape code of the message header category or category-type table. If this code is not found before the end of the block or if it is found too soon in the block, an error is returned. It is highly unlikely that a transmission error in the compressed packet could result in decoding so as to end at the same place as the original. The exception to this would be errors in transmitting bits values such as date, time or sequence number or the least significant bits of encoded values or changes. For additional error detection, a CRC check for the original could be added— one utilizing Arithmetic and the other utilizing Huffman techniques.

The Arithmetic routines typically use 40% more CPU time than the Huffman routines and achieve approximately 15% better compression. On average the compression ratio for the SuperMontage data rate (9.0 Megabits/sec) utilizing Arithmetic Mode, yielded a value of 9.528 with a latency under 10.0 ms. This effectively says the that the NQDS feed operating at a SuperMontage rate could be transmitted over one T1 line! Further overall latency can be reduced from 500 msec to something approaching 10 milliseconds if routing delays are reduced. Since the amount of data is substantially less, it will be easier and much more cost efficient to reduce routing delays. Further, since the quantity of transmitted bits is substantially smaller, the skew amongst transmitted packets will also be proportionately lower.

The average compression ratio for the standard NQDS data rate (221 Kbits/sec) was 9.3925 for the Arithmetic Mode with a latency under 128 ms. The higher latency is due to the time required to accumulate data for blocking. Since the present invention allows for very high compression ratios with small blocks of data, the latency can be reduced substantially from 128 msec without a loss in compression ratio. This effectively says that the existing NQDS feed could be transmitted over one-half of a 56 Kilobit/sec modem line. Other advantages of using data acceleration according to the invention is that such methods inherently provide (i) a high level of encryption associated with the Arithmetic Mode (with no subsequent impact on latency) and (ii) error detection capability of the decompression methods at the end user site. The first benefit produces additional levels of security for the transmitted data and the second benefit guarantees that corrupted data will not be displayed at the end user site. Furthermore, the need to dynamically compare the redundant data feeds at the end user site is eliminated.

In yet another embodiment of the present invention the aforecited algorithms and all other data compression/decompression algorithms may be utilized in a data field specific compiler that is utilized to create new data feed and data stream specific compression algorithms.

A data field description language is utilized to define a list of possible data fields and parameters along with associated data compression encoders and parameter lists. In one embodiment of the invention the data fields are defined utilizing the following convention:

Thus start list and end list are reserved identifiers however any suitable nomenclature can be utilized.

In this simple embodiment of the present invention the list is then submitted to a data compression compiler that accepts the data field list and creates two output files. The first is a data compression algorithm set comprised of data field specific
The method comprising:

receiving an encoded message in a data packet of the financial data stream using a data decoding engine, wherein multiple decoders applying a plurality of lossless decompression techniques are applied to an encoded message.

The method comprising:

decoding the data field with a selected lossless decoder utilizing content independent data decompression, if the descriptor indicates the data field is encoded utilizing content independent data compression.

In this more sophisticated embodiment the encoders are selected based upon the data fields and their specific ordering.

In yet another embodiment of the present invention the sets of ordered data fields can be assigned to sets by set name, giving the ability for nesting of sets to facilitate ease of coding.

In yet another embodiment of the present invention the optional parameters to each encoder are utilized to share parameters amongst the same or different data fields.

Although illustrative embodiments have been described herein with reference to the accompanying drawings, it is to be understood that the present invention is not limited to those precise embodiments, and that various other changes and modifications may be affected therein by one skilled in the art without departing from the scope or spirit of the invention. All such changes and modifications are intended to be included within the scope of the invention as defined by the appended claims.

What is claimed is:

1. A method of decoding one or more encoded messages of a data packet in a financial data stream using a data decoding engine, wherein multiple decoders applying a plurality of lossless decompression techniques are applied to an encoded message, the method comprising:

receiving an encoded message in a data packet of the financial data stream having a plurality of data fields associated with the encoded message and one or more descriptors comprising one or more values, wherein the one or more descriptors indicate data field types of the data fields and lossless encoders used to encode the data fields, and further wherein the lossless encoders are selected based on analyses of content of the data fields;

analyzing the encoded message to identify a descriptor;

selecting one or more lossless decoders for a data field associated with the encoded message, wherein the selecting is based on the descriptor and a description file, and further wherein the description file comprises data field types and associated lossless decoders;

decoding the data field with a selected lossless decoder utilizing content dependent data decompression, if the descriptor indicates the data field is encoded utilizing content dependent data compression; and

decoding the data field with a selected lossless decoder utilizing content independent data decompression, if the descriptor indicates the data field is encoded utilizing content independent data compression.

2. The method of claim 1, wherein the one or more lossless decoders are further selected based upon the specific ordering of the data field in the encoded message.

3. The method of claim 1, wherein the descriptor comprises values corresponding to a single applied decompression technique or multiple decompression techniques applied in a specific order.

4. The method of claim 1, further comprising initiating the method of decoding one or more encoded messages of a data packet in a financial data stream using a synchronization point, wherein the financial data stream includes a plurality of synchronization points.

5. The method of claim 4, wherein the one or more encoded messages of a data packet are included in a broadcast to a plurality of client systems.

6. The method of claim 1, wherein decoding the data field comprises packet independent data decoding.

7. The method of claim 1, further comprising providing one or more global state machines and one or more adaptive local state machines.

8. The method of claim 7, further comprising:

storing in one or more of the global state machines data fields that are likely to repeat in the financial data stream based on a priori knowledge of the data stream; and

storing in one or more of the adaptive local state machines the decoded data field such that the data field is available to decode one or more other data fields.

9. The method of claim 8, further comprising resetting one or more of the adaptive local state machines at a determinate point of the data packet.

10. The method of claim 1, wherein the time of receiving and decoding the one or more encoded messages of a data packet is less than the time to receive the one or more encoded messages of a data packet in unencoded form.

11. The method of claim 1, wherein the method of decoding one or more encoded messages of a data packet in a financial data stream achieves an expansion ratio of at least 1:10.

12. The method of claim 1, wherein the method of decoding one or more encoded messages of a data packet in a financial data stream is performed in real-time.

13. A system for encoding a plurality of data blocks to create an encoded data packet in a financial data stream, wherein multiple encoders applying a plurality of lossless compression techniques are applied to a plurality of data blocks, the system comprising:

an input interface that receives a data block from the plurality of data blocks;

a memory with a fixed table of data blocks based on a priori knowledge of the financial data stream and an adaptive table of data blocks;

a data encoding engine operatively connected to said input interface and said memory having a computer readable program code of instructions executable by the data encoding engine, said instructions comprising instructions to:

analyze content of the data block to determine a data block type;
select one or more lossless encoders for the data block based on the data block type and a computer file, wherein the computer file indicates data block types and associated encoders; encode the data block with a selected lossless encoder using a data block in said adaptive table identified by said selected lossless encoder, if available, otherwise using a data block in said fixed table identified by said selected lossless encoder; and store the data block in said adaptive table, such that the data block is available to encode one or more other data blocks; and an output interface operatively connected to said data encoding engine that outputs the encoded data packet with a descriptor comprising one or more values, wherein the descriptor indicates the selected one or more lossless encoders.

14. The system of claim 13, wherein the data block corresponds to a data field of a message.

15. The system of claim 13, wherein the memory resets the adaptive table at a determinate point of the data packet.

16. The system of claim 13, wherein the system for encoding includes a plurality of synchronization points in the financial data stream for initiating decoding the financial data stream.

17. The system of claim 13, further comprising instructions executable by the data encoding engine to encode one or more data blocks with a selected lossless encoder utilizing content independent data compression, if the data block type is associated with a lossless encoder utilizing content independent data compression.

18. The system of claim 13, wherein the data encoding engine performs packet independent data encoding.

19. The system of claim 13, wherein the instructions to encode the data block with a selected lossless encoder comprises using a difference between the data block and a data block in the adaptive table.

20. The system of claim 13, wherein the system for encoding a plurality of data blocks to create an encoded data packet in a financial data stream achieves a compression ratio of at least 10:1.

21. The system of claim 13, wherein the system for encoding a plurality of data blocks to create an encoded data packet in a financial data stream operates in real-time.

22. A system for encoding a plurality of data blocks to create an encoded data packet in a financial data stream, wherein multiple encoders applying a plurality of lossless compression techniques are applied to a plurality of data blocks, the system comprising:

an input interface that receives a data block;

a data encoding engine operatively connected to said input interface having a computer readable program code of instructions executable by the data encoding engine, said instructions comprising instructions to:

analyze content of the data block to determine a data block type;

select one or more lossless encoders based on the data block type and a computer file, wherein the computer file indicates data block types and associated encoders;

encode the data block with a selected lossless encoder utilizing content dependent data compression, if the data block type is recognized as associated with a lossless encoder utilizing content dependent data compression;

and encode the data block with a selected lossless encoder utilizing content independent data compression, if the data block type is not recognized as associated with a lossless encoder utilizing content dependent data compression; and

an output interface operatively connected to said data encoding engine that outputs a descriptor comprising one or more values in the encoded data packet in the financial data stream, wherein the descriptor indicates the one or more selected lossless encoders.

23. The system of claim 22, wherein the data block corresponds to a data field of a message.

24. The system of claim 22, wherein the system for encoding includes a plurality of synchronization points in the financial data stream for initiating decoding the financial data stream.

25. The system of claim 22, wherein the data encoding engine performs packet independent data encoding.

26. The system of claim 22, further comprising one or more global state machines and one or more adaptive local state machines operatively connected to said data encoding engine.

27. The system of claim 26, wherein the one or more global state machines store data blocks that are likely to repeat in the financial data stream based on a priori knowledge of the data stream and the one or more adaptive local state machines store the received data block such that the data block is available to encode one or more other data blocks.

28. The system of claim 27, wherein the one or more adaptive local state machines reset at a determinate point of the data packet.

29. A method of encoding a plurality of data blocks to create an encoded data packet in a financial data stream using a data encoding engine, wherein multiple encoders applying a plurality of lossless compression techniques are applied to a plurality of data blocks, the method comprising:

receiving a data block from the plurality of data blocks;

analyzing content of the data block to determine a data block type;

selecting one or more lossless encoders based on the data block type and a computer file, wherein the computer file indicates data block types and associated encoders;

encoding the data block with a selected lossless encoder utilizing content dependent data compression, if the data block type is recognized as associated with a lossless encoder utilizing content dependent data compression;

encoding the data block with a selected lossless encoder utilizing content independent data compression, if the data block type is not recognized as associated with a lossless encoder utilizing content dependent data compression; and

providing a descriptor for the encoded data packet in the financial data stream, wherein the descriptor indicates the one or more selected lossless encoders for the encoded data block.

30. The method of claim 29, wherein the data block corresponds to a data field of a message.

31. The method of claim 29, further comprising including a plurality of synchronization points in the financial data stream for initiating decoding the financial data stream.

32. The method of claim 31, wherein the encoded data packet in the financial data stream is broadcast to a plurality of client systems.

33. The method of claim 32, wherein the encoded data packet is a User Datagram Protocol (UDP) data packet.

34. The method of claim 29, wherein the encoding comprises packet independent data encoding.

35. The method of claim 29, further comprising providing one or more global state machines and one or more adaptive local state machines.
36. The method of claim 35, further comprising: storing in one or more of the global state machines data blocks that are likely to repeat in the financial data stream based on a priori knowledge of the data stream; and storing in one or more of the adaptive local state machines the received data block such that the data block is available to encode one or more other data blocks.

37. The method of claim 36, further comprising resetting one or more of the adaptive local state machines at a determinate point of the encoded data packet.

38. The method of claim 29, wherein encoding the data block utilizing content dependent data compression comprises using a difference between data blocks in the encoded data packet.

39. The method of claim 29, wherein the time of encoding the plurality of data blocks and transmitting the encoded data packet is less than the time to transmit the plurality of data blocks in unencoded form.

40. The method of claim 29, wherein the plurality of data blocks includes one or more of stock, options, and futures information.

41. The method of claim 29, wherein the descriptor comprises values corresponding to a single applied compression technique or multiple compression techniques applied in a specific order.

42. The method of claim 29, wherein the method of encoding a plurality of data blocks to create an encoded data packet in a financial data stream is performed in real-time.

43. A method of encoding a plurality of data blocks to create an encoded data packet for a financial data stream using a data encoding engine, wherein multiple encoders applying a plurality of lossless compression techniques are applied to a plurality of data blocks, the method comprising: providing a fixed table of data fields based on a priori knowledge of the financial data stream; providing an adaptive table of data fields; receiving a message from the one or more messages; analyzing content of a data field in the message to determine a data field type; selecting one or more lossless encoders for the data field based on the data field type and a computer file, wherein the computer file indicates data block types and associated encoders; encoding the data field with a selected lossless encoder using a data field in said fixed table identified by said selected lossless encoder, if available, otherwise using a data field in said adaptive table identified by said selected lossless encoder; storing the data field in said adaptive table, such that the data field is available to encode one or more other data blocks; and providing a descriptor for the encoded data field, wherein the descriptor indicates the selected one or more lossless encoders for the encoded data field.

44. The method of claim 43, wherein the data block corresponds to a data field of a message.

45. The method of claim 43, further comprising resetting the adaptive table at a determinate point of the encoded data packet.

46. The method of claim 43, further comprising including a plurality of synchronization points in the financial data stream for initiating decoding the financial data stream.

47. The method of claim 43, wherein encoding the data block comprises packet independent data encoding.

48. The method of claim 43, further comprising encoding one or more of the plurality of data blocks with a selected lossless encoder utilizing content independent data compression, if the data block type is associated with a lossless encoder utilizing content independent data compression.

49. The method of claim 43, wherein encoding the data block with a selected lossless encoder comprises using a difference between the data block and a data block in the adaptive table.

50. The method of claim 43, wherein the method of encoding a plurality of data blocks to create an encoded data packet in a financial data stream achieves a compression ratio of at least 10:1.

51. The method of claim 43, wherein the method of encoding a plurality of data blocks to create an encoded data packet in a financial data stream is performed in real-time.

52. A method of encoding one or more messages to create an encoded data packet for a financial data stream using a data encoding engine, wherein multiple encoders applying a plurality of lossless compression techniques are applied to a plurality of data fields of a message, the method comprising: providing a fixed table of data fields based on a priori knowledge of the financial data stream; providing an adaptive table of data fields; receiving a message from the one or more messages; analyzing content of a data field in the message to determine a data field type; selecting one or more lossless encoders for the data field based on the data field type and a computer file, wherein the computer file indicates data block types and associated encoders; encoding the data field with a selected lossless encoder using a data field in said adaptive table identified by said selected lossless encoder, if available, otherwise using a data field in said fixed table identified by said selected lossless encoder; storing the data field in said adaptive table, such that the data field is available to encode one or more other data fields; and providing a descriptor for the encoded data packet, wherein the descriptor indicates the selected one or more lossless encoders for the encoded data field.

53. The method of claim 52, further comprising resetting the adaptive table at a determinate point of the data packet.

54. The method of claim 52, further comprising including a plurality of synchronization points in the financial data stream for initiating decoding the financial data stream.

55. The method of claim 52, wherein encoding the data field comprises packet independent data encoding.

56. The method of claim 52, further comprising encoding one or more data fields with a selected lossless encoder utilizing content independent data compression, if the data field type is associated with a lossless encoder utilizing content independent data compression.

57. The method of claim 52, wherein encoding the data field with a selected lossless encoder comprises using a difference between the data field and a data field in the adaptive table.

58. The method of claim 52, wherein the method of encoding one or more messages to create an encoded data packet for a financial data stream is performed in real-time.

59. The method of claim 52, wherein the method of encoding one or more messages to create an encoded data packet for a financial data stream achieves a compression ratio of at least 10:1.

60. A system for encoding one or more messages to create an encoded data packet in a financial data stream, wherein multiple encoders applying a plurality of lossless compres-
sion techniques are applied to a plurality of data fields of a message; the system comprising:
an input interface that receives a message, wherein the message comprises a plurality of data fields;
a data encoding engine operatively connected to said input interface having a computer readable program code of instructions executable by the data encoding engine, said instructions comprising instructions to:
analyze content of a data field of the message to determine a data field type;
select one or more lossless encoders based on the data field type and a description file, wherein the description file indicates data field types and associated encoders;
encode the data field with a selected lossless encoder utilizing content dependent data compression, if the data block type is recognized as associated with a lossless encoder utilizing content dependent data compression; and
encode the data field with a selected lossless encoder utilizing content independent data compression, if the data block type is not recognized as associated with a lossless encoder utilizing content dependent data compression; and
an output interface operatively connected to said data encoding engine that outputs a descriptor in the encoded data packet in the financial data stream, wherein the descriptor indicates the selected one or more lossless encoders.

61. The system of claim 60, wherein the system for encoding includes a plurality of synchronization points in the financial data stream for initiating decoding the financial data stream.

62. The system of claim 61, wherein the encoded data packet is included in a broadcast to a plurality of client systems.

63. The system of claim 60, wherein the data encoding engine performs packet independent data encoding.

64. The system of claim 60, further comprising one or more global state machines and one or more adaptive local state machines operatively connected to said data encoding engine.

65. The system of claim 64, wherein the one or more global state machines store data fields that are likely to repeat in the financial data stream based on a priori knowledge of the data stream and the one or more adaptive local state machines store the data field such that the data field is available to encode one or more other data fields.

66. The system of claim 65, wherein the one or more adaptive local state machines reset at a determinate point of the data packet.

67. The system of claim 60, wherein the instructions to encode the data field utilizing content dependent data compression comprises using a difference between the content of data fields in the encoded data packet.

68. The system of claim 60, wherein the time of encoding the one or more messages and transmitting the encoded data packet in the financial data stream is less than the time to transmit the one or more messages in unencoded form.

69. The system of claim 60, wherein the one or more messages include one or more of stock, options, and futures information.

70. The system of claim 60, wherein the one or more messages include financial news.

71. The system of claim 60, wherein the system for encoding one or more messages to create an encoded data packet in a financial data stream achieves a compression ratio of at least 10:1.

72. The system of claim 60, wherein the system for encoding one or more messages to create an encoded data packet in a financial data stream operates in real-time.

73. A method of decoding one or more encoded data packets of a financial data stream using a data decoding engine, wherein multiple decoders applying a plurality of lossless decompression techniques are applied to an encoded data packet, the method comprising:

receiving an encoded data packet from the financial data stream having one or more descriptors comprising one or more values, wherein the one or more descriptors indicate lossless encoders used to encode data blocks associated with the encoded data packet, and further wherein the lossless encoders are selected based on analyses of content of the data blocks;

analyzing the encoded data packet of the financial data stream to identify a descriptor;

selecting one or more lossless decoders for a data block associated with the data packet, wherein the selecting is based on the descriptor;

decoding the data block with a selected lossless decoder utilizing content dependent data decompression, if the descriptor indicates the data block is encoded utilizing content dependent data compression; and

decoding the data block with a selected lossless decoder utilizing content independent data decompression, if the descriptor indicates the data block is encoded utilizing content independent data compression.

74. The method of claim 73, wherein the data block corresponds to a data field associated with a message in the encoded data packet of the financial data stream.

75. The method of claim 73, wherein the descriptor comprises values corresponding to a single applied decompression technique or multiple decompression techniques applied in a specific order.

76. The method of claim 73, wherein decoding the data block utilizing content independent data decompression occurs prior to decoding the data block utilizing content dependent data decompression.

77. The method of claim 73, further comprising initiating the method of decoding one or more encoded data packets of a financial data stream using a synchronization point, wherein the financial data stream includes a plurality of synchronization points.

78. The method of claim 77, wherein the one or more encoded data packets of the financial data stream are broadcast to a plurality of client systems.

79. The method of claim 78, wherein the one or more encoded data packets are User Datagram Protocol (UDP) data packets.

80. The method of claim 73, wherein decoding the data block comprises packet independent data decoding.

81. The method of claim 73, further comprising providing one or more global state machines and one or more adaptive local state machines.

82. The method of claim 81, further comprising:

storing in one or more of the global state machines data blocks that are likely to repeat in the financial data stream based on a priori knowledge of the data stream; and

storing in one or more of the adaptive local state machines the decoded data block such that the data block is available to decode one or more other data blocks.

83. The method of claim 82, further comprising resetting one or more of the adaptive local state machines at a determinate point of the data packet.
84. The method of claim 73, wherein decoding the data block utilizing content dependent data decompression comprises using a difference between data blocks in the encoded data packet.

85. The method of claim 73, wherein the time of receiving and decoding the one or more encoded data packets is less than the time to receive the one or more encoded data packets in unencoded form.

86. The method of claim 73, wherein the one or more encoded data packets include one or more of stock, options, and futures information.

87. The method of claim 73, wherein the one or more encoded data packets include financial news.

88. The method of claim 73, wherein the method of decoding one or more encoded data packets of a financial data stream achieves an expansion ratio of at least 1:10.

89. The method of claim 73, wherein the method of decoding one or more encoded data packets of a financial data stream is performed in real-time.

90. The method of claim 73, wherein the one or more encoded data packets of the financial data stream is transmitted to one or more client systems utilizing TCP/IP.

91. A method of decoding one or more encoded data packets in a financial data stream using a data decoding engine, wherein multiple decoders applying a plurality of lossless decompression techniques are applied to an encoded data packet, the method comprising:

- providing a fixed table of data blocks based on a priori knowledge of the financial data stream;
- providing an adaptive table of data blocks;
- receiving an encoded data packet from the financial data stream having one or more descriptors comprising one or more values, wherein the one or more descriptors indicate lossless encoders used to encode data blocks associated with the encoded data packet, and further wherein the lossless encoders are selected based on analyses of content of the data blocks;
- analyzing the encoded data packet to identify a descriptor; selecting one or more lossless decoders for a data block associated with the encoded data packet, wherein the selecting is based on the descriptor;
- decoding the data block with a selected lossless decoder using a data block in said adaptive table identified by said selected lossless decoder, if available, otherwise using a data block in said fixed table identified by said selected lossless decoder; and
- storing the decoded data block in said adaptive table, such that the decoded data block is available to decode one or more other data blocks.

92. The method of claim 91, wherein the data block corresponds to a data field associated with a message in the encoded data packet of the financial data stream.

93. The method of claim 91, further comprising resetting the adaptive table at a determinate point of the encoded data packet.

94. The method of claim 91, further comprising initiating the method of decoding one or more encoded data packets in a financial data stream using a synchronization point, wherein the financial data stream includes a plurality of synchronization points.

95. The method of claim 91, wherein decoding the data block comprises packet independent data decoding.

96. The method of claim 91, further comprising decoding one or more data blocks with a selected lossless decoder utilizing content independent data decompression, if the descriptor indicates the data block is encoded utilizing content independent data compression.

97. The method of claim 91, wherein decoding the data block with a selected lossless decoder comprises using a difference between the data block and a data block in the adaptive table.

98. A system for decoding one or more encoded data packets of a financial data stream, wherein multiple decoders applying a plurality of lossless decompression techniques are applied to an encoded data packet, the system comprising:

- an input interface that receives an encoded data packet from the financial data stream having one or more descriptors comprising one or more values, wherein the one or more descriptors indicate lossless encoders used to encode data blocks associated with the encoded data packet, and further wherein the lossless encoders are selected based on analyses of content of the data blocks;
- a data decoding engine operatively connected to said input interface having a computer readable program code of instructions executable by the data decoding engine, said instructions comprising instructions to:
  - analyze the encoded data packet of the financial data stream to identify a descriptor;
  - select one or more lossless decoders for a data block associated with the encoded data packet, wherein the selecting is based on the descriptor;
  - decode the data block with a selected lossless decoder utilizing content dependent data decompression, if the descriptor indicates the data block is encoded utilizing content dependent data decompression; and
  - decode the data block with a selected lossless decoder utilizing content independent data decompression, if the descriptor indicates the data block is encoded utilizing content independent data compression; and
- an output interface operatively connected to said data decoding engine that outputs data from the data packet.

99. The system of claim 98, wherein the data block corresponds to a data field associated with a message in the encoded data packet of the financial data stream.

100. The system of claim 98, wherein the descriptor comprises values corresponding to a single applied decompression technique or multiple decompression techniques applied in a specific order.

101. The system of claim 98, wherein the system for decoding initiates decoding the one or more encoded data packets of the financial data stream using a synchronization point, and further wherein the financial data stream includes a plurality of synchronization points.

102. The system of claim 98, wherein the data decoding engine performs packet independent data decoding.

103. The system of claim 98, further comprising one or more global state machines and one or more adaptive local state machines operatively connected to said data decoding engine.

104. The system of claim 103, wherein the one or more global state machines store data blocks that are likely to repeat in the financial data stream based on a priori knowledge of the data stream and the one or more adaptive local state machines store the decoded data block such that the data block is available to decode one or more other data blocks.

105. The system of claim 104, wherein the one or more adaptive local state machines reset at a determinate point of the encoded data packet.

106. The system of claim 98, wherein the system for decoding one or more encoded data packets of a financial data stream achieves an expansion ratio of at least 1:10.

107. The system of claim 98, wherein the system for decoding one or more encoded data packets of a financial data stream operates in real-time.
108. A system for decoding one or more data packets in a financial data stream, wherein multiple decoders applying a plurality of lossless decompression techniques are applied to an encoded data packet, the system comprising:

- an input interface that receives an encoded data packet from the financial data stream having one or more descriptors comprising one or more values, wherein the one or more descriptors indicate lossless encoders used to encode data blocks associated with the data packet, and further wherein the lossless encoders are selected based on analyses of content of the data blocks;
- a memory with a fixed table of data fields based on a priori knowledge of the financial data stream and an adaptive table of data blocks;
- a data decoding engine operatively connected to said input interface and said memory having a computer readable program code of instructions executable by the data decoding engine, said instructions comprising instructions to:
  - analyze the encoded data packet to identify a descriptor;
  - select one or more lossless decoders for a data block associated with the encoded data packet, wherein the selecting is based on the descriptor;
  - decode the data block with a selected lossless decoder utilizing content independent data decompression, if the descriptor indicates the data block is encoded utilizing content independent data decompression, and further wherein the description file comprises data field types and associated lossless decoders;
  - store the decoded data block in said adaptive table, such that the decoded data block is available to decode one or more other data blocks; and
- an output interface operatively connected to said data decoding engine that outputs data from the data packet.

109. The system of claim 108, wherein the data block corresponds to a data field associated with a message in the encoded data packet of the financial data stream.

110. The system of claim 108, wherein the memory resets the adaptive table at a determinate point of the data packet.

111. The system of claim 108, wherein the system for decoding initiates decoding the one or more data packets in a financial data stream using a synchronization point, and further wherein the financial data stream includes a plurality of synchronization points.

112. The system of claim 108, wherein the data decoding engine performs packet independent data decoding.

113. The system of claim 108, further comprising instructions executable by the data decoding engine to decode one or more data blocks with a selected lossless decoder utilizing content independent data decompression, if the descriptor indicates the data block is encoded utilizing content independent data compression.

114. The system of claim 108, wherein the instructions to decode the data block with a selected lossless decoder comprises using a difference between the data block and a data block in the adaptive table.

115. A system for decoding one or more encoded messages of a data packet in a financial data stream, wherein multiple decoders applying a plurality of lossless decompression techniques are applied to an encoded message, the system comprising:

- an input interface that receives an encoded message in a data packet from the financial data stream having a plurality of data fields associated with the encoded message and one or more descriptors comprising one or more values, wherein the one or more descriptors indicate data field types of the data fields and lossless encoders used to encode the data fields, and further wherein the lossless encoders are selected based on analyses of content of the data fields;
- a memory with a fixed table of data fields based on a priori knowledge of the financial data stream and an adaptive table of data fields;
- a data decoding engine operatively connected to said input interface and said memory having a computer readable program code of instructions executable by the data decoding engine, said instructions comprising instructions to:
  - analyze the encoded message to identify a descriptor;
  - select one or more lossless decoders for a data field associated with the encoded message, wherein the selecting is based on the descriptor and a description file, and further wherein the description file comprises data field types and associated lossless decoders;
  - decode the data field with a selected lossless decoder using a data field in said adaptive table identified by said selected lossless decoder, if available, otherwise using a data field in said fixed table identified by said selected lossless decoder; and
  - store the decoded data field in said adaptive table, such that the decoded data field is available to decode one or more other data fields; and
- an output interface operatively connected to said data decoding engine that outputs data from the data packet.

116. The system of claim 115, wherein the memory resets the adaptive table at a determinate point of the data packet.

117. The system of claim 115, wherein the system for decoding initiates decoding the one or more encoded messages using a synchronization point, wherein the financial data stream includes a plurality of synchronization points.

118. The system of claim 115, wherein the data decoding engine performs packet independent data decoding.

119. The system of claim 115, wherein the instructions to select one or more lossless decoders is further based upon the specific ordering of the data field in the encoded message.

120. The system of claim 115, further comprising instructions executable by the data decoding engine to decode the data field with a selected lossless decoder utilizing content independent data decompression, if the descriptor indicates the data block is encoded utilizing content independent data compression.

121. The system of claim 115, wherein the instructions to decode the data block with a selected lossless decoder comprises using a difference between the data field and a data field in the adaptive table.

122. The system of claim 115, wherein the system for decoding one or more encoded messages of a data packet in a financial data stream achieves an expansion ratio of at least 1:10.

123. The system of claim 115, wherein the system for decoding one or more encoded messages of a data packet in a financial data stream operates in real-time.