An Internet Multicast System for the Stock Market

N. F. Maxemchuk
D. H. Shur

AT&T Labs - Research

ABSTRACT

We are moving toward an international, twenty-four hour, distributed, electronic stock exchange. The exchange will use the global Internet, or internet technology. This system is a natural application of multicast because there are a large number of receivers that should receive the same information simultaneously.

The data requirements for the stock exchange are discussed. The current multicast protocols lack the reliability, fairness, and scalability needed in this application. We describe a distributed architecture and a timed reliable multicast protocol, TRMP, that has the appropriate characteristics.

We consider three applications:

1. A unified stock ticker of the transactions that are being conducted on the various physical and electronic exchanges. Our objective is to deliver the the same combined ticker reliably and simultaneously to all receivers, anywhere in the world.
2. A unified sequence of buy and sell offers that are delivered to a single exchange or a collection of exchanges. Our objective is to give all traders the same fair access to an exchange independent of their relative distances to the exchange or the delay and loss characteristics of the international network.
3. A distributed, electronic trading floor that can replace the current exchanges. This application has the fairness attributes of the first two applications and uses TRMP to conduct irrefutable, distributed trades.

1. Introduction

An exchange is any organization, association or group which provides or maintains a marketplace where securities, options, futures or commodities can be traded. There are hundreds of exchanges around the world. Yahoo lists 107 stock exchanges.

Traditional stock exchanges are centralized. The exchange is located in a single physical place and all data (both market and trades) flows through a single system. The centralized system is responsible for transaction reporting and exchanging assets.

Electronic exchanges, like the NASDAQ, allow remotely located traders to connect with the centralized system. The traders send and receive information, trades, etc., over dedicated access lines and private networks. Connections are expensive, and thus limit who can be connected.

The Internet allows almost anyone to connect to electronic exchanges at very low cost. Internet communication technology is enabling the restructuring of the stock exchanges and a number of Internet based, private stock trading systems are emerging. Private stock trading systems interface into other trading systems, such as the NASDAQ. The private system may satisfy buy and sell requests among their own subscriber base, or may pass on the trade to the larger exchange.

Electronic stock markets present opportunities, as noted in these excerpts from the prepared remarks by Frank G. Zarb, Chairman and CEO of the National Association of Securities Dealers Inc. before the

National Press Club in Washington, D.C. on Wednesday, June 23, 1999:

> *In a very few years, trading securities will be digital, global, and accessible 24 hours a day.*
>
> *People will be able to get stock price quotations instantly and instantly execute a trade any time of day or night, anywhere on the globe, with stock markets linked and almost all-electronic.*
>
> *As for stock markets, they will see global alliances, mergers, and new electronic ventures. That will give companies listed on these markets access to pools of capital internationally, not just domestically, and consumers will be able to invest in a worldwide list of companies as easily as trading locally.*
>
> *This 21st century stock market will be multi-dealer, computer-screen based, technology-driven and open to all - all because people will have access to information that they want to act on.*

Electronic exchanges also create technical challenges, as noted in these excerpts from a New York Times article on Sept. 23, 1999, of an interview with Arthur Levitt, the chairman of the Securities and Exchange Commission:

> *Levitt seems most concerned that if trading continues to migrate to the new electronic market systems, investors may not get the best prices. Information about orders and transactions across the entire market are not now available in one place. Technology, he said, allows the creation of a central system in which investors will be fully informed about prices everywhere, from the New York Stock Exchange, the NASDAQ, the American Stock Exchange and the new systems.*
>
> *He stressed that he was asking for the development of a technology that would allow all orders to be shown to investors, not an institution or place where all orders would be executed.*

We are proposing an architecture and protocol which can meet these challenges. Three applications that use this system are described in section 7.

In the first application, trades that are executed on hundreds of trading floors, distributed globally, are merged into a single information stream and presented to millions or tens of millions of investors, internationally. Our objective is to provide the same information stream to all of the investors at the same time, no matter where in the world they or the trading floors are located.

In the second application, the buy and sell orders from millions of investors are merged into a single stream and presented to a single exchange, or simultaneously presented to several exchanges. The exchanges operate under their own rules. Currently, investors that are in the same city as an exchange have an advantage over investors on the other side of the world. If both investors see a ticker at the same time and submit a bid, the bid from the closer investor may reach the exchange several seconds earlier on the average. Our objective is to give all of the investors fair access to all of the exchanges.

In the third application we use a distributed rule to operate on the buy, sell, and cancel orders to create a distributed exchange. Distributed exchanges can operate under several rules for executing trades and can be among the trading floors in the first two applications.

The three applications are implemented using Internet multicast [1] technology. Multicast has recently been applied to the Swiss stock exchange [2,3] and the New York stock exchange [3] using the Isis [4] and Horus [5] tool kits. Both of these systems distribute the ticker from a single exchange over a limited geographical area. These systems are a subset of our first application, which distributes an ordered sequence of the tickers from many exchanges, internationally. The previous systems do not address the multiple source sequencing problem or the propagation delay, network delay and losses that occur in an international network.

We obtain reliable, ordered message delivery by using a modified version of the reliable broadcast protocol, RBP[6]. RBP was the first reliable broadcast protocol and has since become the reliable multicast protocol, RMP. RMP provides a unique sequence of the messages from multiple sources to multiple destinations that lose different messages. It uses a negative acknowledgement strategy to reduce the number of control messages. RMP is described in section 2.

RMP is an event driven protocol, control messages are transmitted when other messages are received. The modified version, TRMP, is time driven and guarantees that all of the receivers obtain the same sequence of messages more quickly than the original protocol. We can guarantee faster delivery because not receiving a negative acknowledgement has much more significance in a time driven protocol than in an event driven protocol. For example, consider a scheduled railroad with impatient commuters who have cell phones. If the network controller does not receive a complaint within a few minutes of a train's scheduled departure, he can be reasonably certain that the train left on time. If events occur at regular intervals, like busses that run every ten minutes, not receiving a negative acknowledgement is almost as significant as it is for a scheduled event. The change to a timed protocol and other modifications of RMP are described in section 5.

RMP is intended for groups with a few tens of receivers and does not scale to the tens of millions of receivers in the stock market application. We use two hierarchical techniques, described in section 4, to resolve the scaling issues.

The first hierarchical technique distinguishes between local regions and the international network. The loss, delay, and number of receivers is significantly different in these two segments of the network, which makes it reasonable to use different recovery techniques. TRMP is used in the international network and an alternative recovery procedure, such as the reliable multicast transport protocol, RMTP [7], or the gossip protocols used in bimodal multicast[3] may be used in the local region. Some of our applications do not require recovery in the local regions, as discussed in sections 3 and 7. By synchronizing and remulticasting the data at the boundary between the international network and the local regions we compensate for differences in network delay, including the signal propagation time, and the recovery time for lost messages. Remulticasting has been used in the cooperative, resilient multicast protocol, CRMP[8], to restore timing to video multicasts that recover lost packets.

The second hierarchical technique reduces the number of TRMP receivers in a group by partitioning the receivers into sub-groups that span different distances: perhaps national, continental and intercontinental regions. TRMP uses a "logical" loop of receivers. The resulting hierarchy consists of loops of loops similar to the hierarchy of physical loops in John Pierce's classical paper "How far can loops go?"[9].

In section 6 we consider the security requirements and trust relationships in the various stock market applications. We reduce the problems to known, solved, cryptographic problems. There is active research on many of these security problems, and advances can be applied directly to the stock market applications.

2. RMP

RMP was originally implemented in 1983 to build a distributed database, with redundant data, on an Ethernet[10]. The computers on the Ethernet independently lost messages due to buffer overflows caused by competing processes. The protocol predated the first multicast implementations[1] and was called the reliable broadcast protocol, RBP.

RBP has three characteristics that distinguished it from earlier protocols:

1. Every receiver places the messages from all of the sources in the same sequence.
2. Every receiver eventually knows that every other receiver has a message.

3. It operates with as few as one control message per source message.

RBP was the first broadcast protocol to use negative acknowledgements, NACK's, to reduce the number of control messages. When there aren't any losses RBP sends only one control message per data message, independent of the number of receivers. The number of messages that are transmitted when there are losses is derived in [11]. The number of control messages used by RBP should be compared with earlier reliable protocols, that require at least as many control messages as receivers when there are no losses, without providing the first two characteristics.

RBP was applied to the Internet multicast network and called RMP[12]. NASA maintains a WEB site for recent work on RMP at http://research.ivv.nasa.gov/RMP/.

RMP has two parts. The first part operates on multicast messages during normal operation and guarantees delivery and ordering of the messages from the sources. In TRMP we use much of the first part of RMP. The second part is a reformation protocol that reorganizes the broadcast group and guarantees database consistency after failures and recoverys. In TRMP we use a reformation procedure that increases the availability of the stock market system, but does not guarantee database consistency. The TRMP reformation protocol is described in section 5.3.

In an RMP system there are *n* sources and *m* receivers, as shown in figure 1. The sources and receivers may be the same or different. The objective is for the *m* receivers to place the messages from the *n* sources in the same order, regardless of which receivers fail to receive which messages. This is accomplished by having a single receiver acknowledge source messages. The acknowledgement assigns the message a sequence number and all of the receivers place the messages in that order. We guarantee that every receiver has all of the messages by sequentially passing the responsibility to acknowledge messages to each receiver and requiring a receiver to acquire all of the preceding messages before acknowledging a new message. The receiver that can acknowledge source messages is called the token site.

A message from source *s* contains the label (s, M_s) to signify that it is the M_s^{th} message from source *s*. Source *s* transmits message M_s at regular intervals until it receives an acknowledgement or decides that the token site has failed. If a source decides that the token site has failed it initiates the reformation process.

The receivers take turns acknowledging messages from sources by passing the token. Each of the receivers is assigned a unique number from 0 to $m-1$. When the token site at receiver number *r* sends an acknowledgement, the control message is multicast and serves four separate functions:

1. It is an acknowledgement to source *s* that message (s, M_s) has been received by the receiver group.
2. It informs all of the receivers that message (s, M_s) is assigned the global sequence number *t*.
3. It is an acknowledgement to the previous token site, receiver $[r-1] \bmod m$, that the token was successfully transferred to *r*.
4. It is a message to the next token site, receiver $[r+1] \bmod m$, inviting it to accept the token.

Token site *r* periodically sends acknowledgement *t* until it receives acknowledgement $t+1$, which acknowledges that receiver $[r+1] \bmod m$ accepted the token. If an acknowledgement isn't received in a specified number of attempts, receiver *r* decides that receiver $r+1$ is inoperable and initiates a reformation process. In order to prevent unnecessary reformations, receiver $r+1$ transmits a token acknowledgement message when there are no source messages to acknowledge.

As soon as *r* sends acknowledgement *t* it gives up the right to acknowledge new source messages, even though it is not certain that $[r+1] \bmod m$ has received the token. This guarantees that at most one receiver assigns sequence numbers to source messages.

Receiver $[r+1] \bmod m$ does not accept the token transferred by acknowledgement *t* until it has all of the acknowledgements and source messages that were acknowledged up to and including *t*. Once a receiver accepts the token it responds to all retransmission requests.

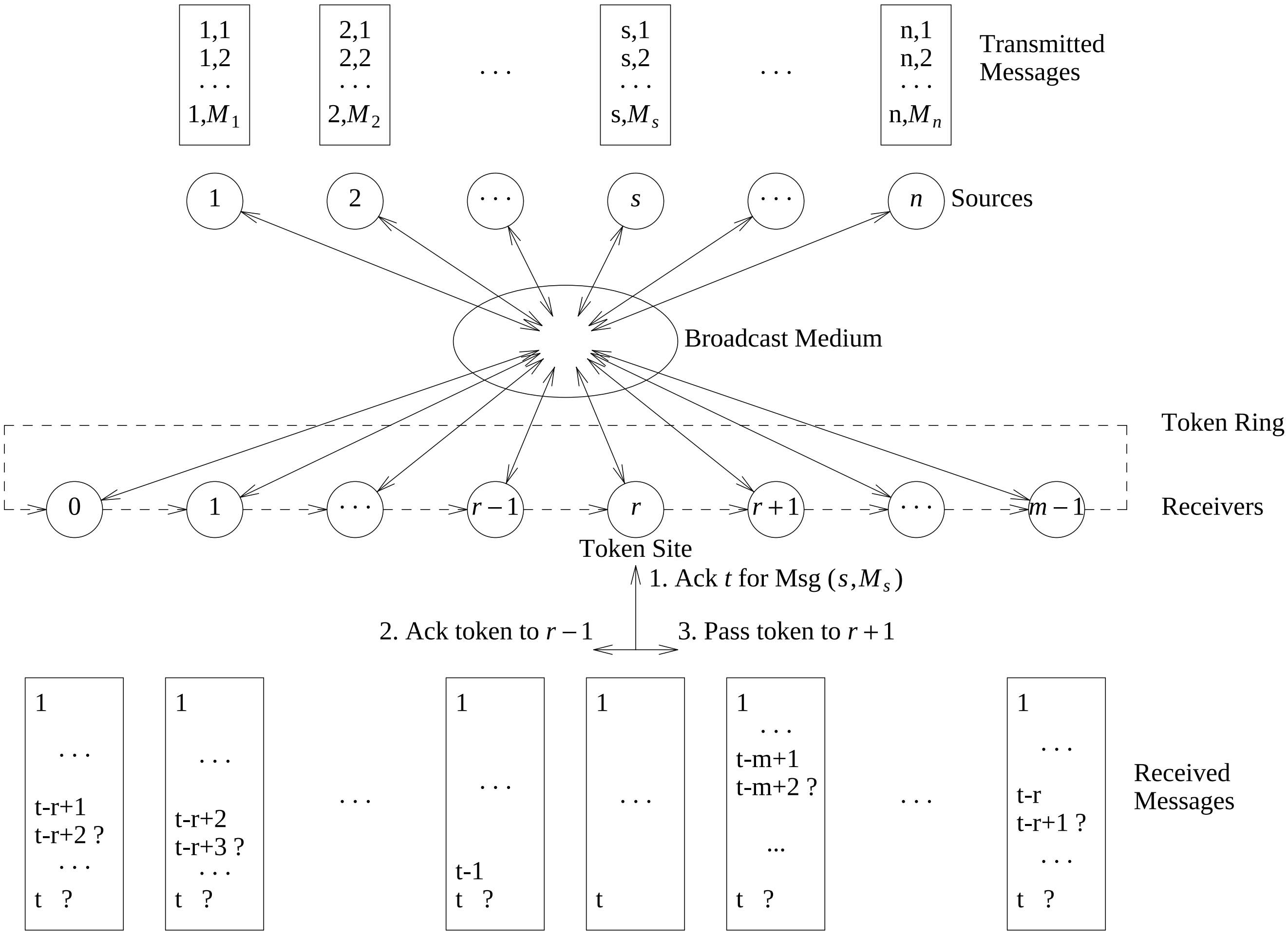

Figure 1. The Reliable Broadcast Protocol

The receivers use a negative acknowledgement protocol and explicitly request retransmissions. If an acknowledgement is received with a larger sequence number than expected, the receiver requests the missing acknowledgements. If an acknowledgement is received for a source message (s, M_s) that has not been received, the receiver requests the missing source message.

The sources and previous token sites use a positive acknowledgement protocol and implicitly request retransmissions. If a source retransmits a message that has been acknowledged, the implication is that the source failed to receive the acknowledgement. If a previous token site retransmits a token passing message, the implication is that the site failed to receive the token passing acknowledgement.

When a receiver passes the token, it does not stop servicing retransmission requests until it receives the acknowledgement for passing the token. This guarantees that at least one site can respond to all retransmission requests.

RMP guarantees that every receiver eventually receives the acknowledged messages and that every receiver eventually knows that every other receiver has received these messages. When a source message is acknowledged, the receiver that sent the acknowledgement has that message and all of the acknowledged source messages that preceded it. We can also infer that the previous token sites had all of the messages that were needed to accept their latest token. Therefore, when acknowledgement t is transmitted from receiver r:

— receiver r has all of the messages up to and including the t^{th} source message,

— receiver $(r-1) \mod m$ has all of the messages up to and including the $(t-1)^{th}$ source message,

— $\cdots$, and

— receiver $(r-m+1) \mod m$ has all of the messages up to and including the $(t-m+1)^{th}$ source message.

Since $(r-m) \mod m = r$, *all of the receivers have all of the messages up to and including* $(t-m+1)^{th}$ *source message.*

A similar line of reasoning allows us to determine what all receivers know about the other receivers. When the t^{th} acknowledgement is transmitted receiver r knows that all of the receivers have all or the messages up to and including $(t-m+1)^{th}$ source message. As before,

— receiver $(r-1) \mod m$ knows that all receivers have all of the messages up to and including the $(t-m)^{th}$ source message,

— $\cdots$, and

— receiver $(r-m+1) \mod m$ knows that all of the receivers have all of the messages up to and including the $(t-m+2)^{th}$ source message.

Since $(r-m) \mod m = r$, *all of the receivers know that all of the receivers have all of the messages up to and including the* $(t-m+2)^{th}$ *source message*.

It can take a long time to recover missing messages in an event driven system that uses negative acknowledgements. When there are no new source messages to acknowledge, receivers that missed the latest source messages or acknowledgements do not detect their loss. In reference [6] additional techniques are used to recover missing messages when the sources are idle. These techniques are not needed in TRMP.

3. Requirements of the Application

The requirements of a global stock market are very different from those of a replicated database on a local area network. Some of the requirements are addressed by the original RMP. For instance, the electronic stock exchanges that are used in the Swiss exchange [2] and the New York stock exchange[3] explicitly require that messages be delivered in the same order at every receiver. This characteristic is easier to obtain in exchanges with a single message source than in our applications. However, RMP is designed to sequence messages from multiple sources and satisfies this requirement.

There are, however, other requirements that make it necessary to modify RMP and the flat, logical loop architecture. In the first stock market application we distribute a unified ticker to millions of receivers that join and leave the application frequently. RMP cannot operate effectively in this environment because

— a network with millions of receivers must pass the token millions of times before the reception guarantees are realized,

— when receivers join or leave the group frequently, a protocol that uses the original reformation process will spend most of its time reorganizing the receiver list, rather than ordering messages, and,

— most messages will be lost by some receivers so that the number of recovery messages may eliminate the advantages of the small number of control messages in RMP.

In addition to the large number of receivers, the number of messages in a unified stock ticker may be greater than the RMP acknowledgement strategy or the receivers can accommodate. In the core of the network there may be periods when the average arrival rate of messages will be greater than the average time that it takes to pass a token. During these periods, the queue of messages waiting to be acknowledged increases. Eventually, the backlog of unacknowledged messages will result in unacceptable delays and lost messages. In section 5 we describe changes in RMP that eliminate the constraints on message volume.

At the edge of the network, the amount of data in the composite ticker will exceed the amount of data that can be transmitted to most receivers. We cannot require that all of the users have the same access rates to the networks, and we should not constrain the network by the least capable receiver. All of the electronic systems, including the antiquated NASDAQ system, compensate for receiver differences by filtering the information to receivers. In section 4.3 we discuss striping.

The stock market applications introduce a number of fairness requirements that are different from most other applications. One requirement is for receiver fairness. The data should be delivered to all of the receivers simultaneously so that the customers have the same opportunity to act on the information. The electronic networks for the Swiss exchange and the New York stock exchange guarantee the same approximate delivery time at every receiver. It is easier to make this guarantee for networks that operate over limited distances, than for our international network. In the international network there are greater differences in network delay and loss rates between source-destination pairs than in a regional network. In addition, in the international network, the speed of light may give some receivers an unfair advantage. We provide receiver fairness by preventing the customers from getting access to the original multicast and simultaneously remulticasting delayed message sequences in each local area.

In addition to receiver fairness, there is source fairness. Every source should have an equal opportunity to have its data received promptly. This should hold even though some sources are near the receiver, and have few message losses and others are far and must retransmit several times. Although RMP was not designed for transmitter fairness, it achieves a degree of fairness by rotating the token site around the network. The token site is the portal that sources send their messages through. In our international network the portal continuously moves around the world, rather than remaining closer to some sources than others.

The trust and security issues on the stock market are completely different from the original application of RMP. It is unreasonable to expect individuals who are competing in a stock market to cooperate to recover missed messages or to provide fair access to one another. The difference in the value of the data that is exchanged in our applications makes security a more important issue in some applications than in others. The security concerns that must be addressed include:

— constraining transmission access to authorized sources,

— preventing early reception of the data stream,

— limiting reception to authorized receivers,

— preventing spoofing or adding to the message sequence, and

— preventing denial of service attacks.

The security issues of our three applications are addressed in section 6.

There is one requirement that makes some stock market applications easier to implement than many of the earlier applications of RMP. In some stock market applications data quickly becomes obsolete. For instance, a new trading price makes the price from a few seconds earlier less useful. The bimodal multicast, that is used in the NYSE, implicitly takes advantage of this characteristic by not recovering "older" messages. The ability to disregard old messages prevents that protocol from becoming overloaded. This requirement justifies not recovering messages at the edge of the network in some of our applications, as in section 7.1.

It is obvious that the stock market system should be available most of the time. This is one of the primary reasons for restricting the size and membership of token passing groups in RMP and changing the reformation process so that the entire system is not affected when network receivers fail or are added. The modified reformation process is described in section 5.3.

Finally, the messages that are lost by different receivers on a multicast tree are correlated. When a message is lost or corrupted on a link of the tree, all of the downstream receivers lose the message and there may be a NACK implosion [13,8,14,15]. NACK implosion is reduced in our multiple loop architecture by having multiple token sites and limiting the number of receivers that recover messages from any particular site, as described in section 4. In section 5.2 we show how to further reduce the number of NACK's in RMP.

4. Architecture

The scaling and trust issues in the stock market applications lead to a change in the basic architecture of RMP. In the original RMP architecture, figure 1, all of the sources and receivers are equally trusted and all of the receivers participate in the token passing loop. The modified architecture is multi-layered, as shown in figure 2. The primary sources, S_P, are trusted more than the secondary sources, S_S, although the S_P may not be completely trusted. The S_P transmit messages on a global multicast tree. The receivers $R_T = \{R_P, R_S\}$ are owned by the network. The R_T participate in TRMP and are trusted to recover missing messages and to send the messages to all of the customers, *R*, at the same time. The large numbers of *R* are not trusted at all. If the quantity of data transmitted in an application exceeds the message handling ability of the R_T's the entire reliable multicast segment of the architecture is replicated, as described in section 4.3.

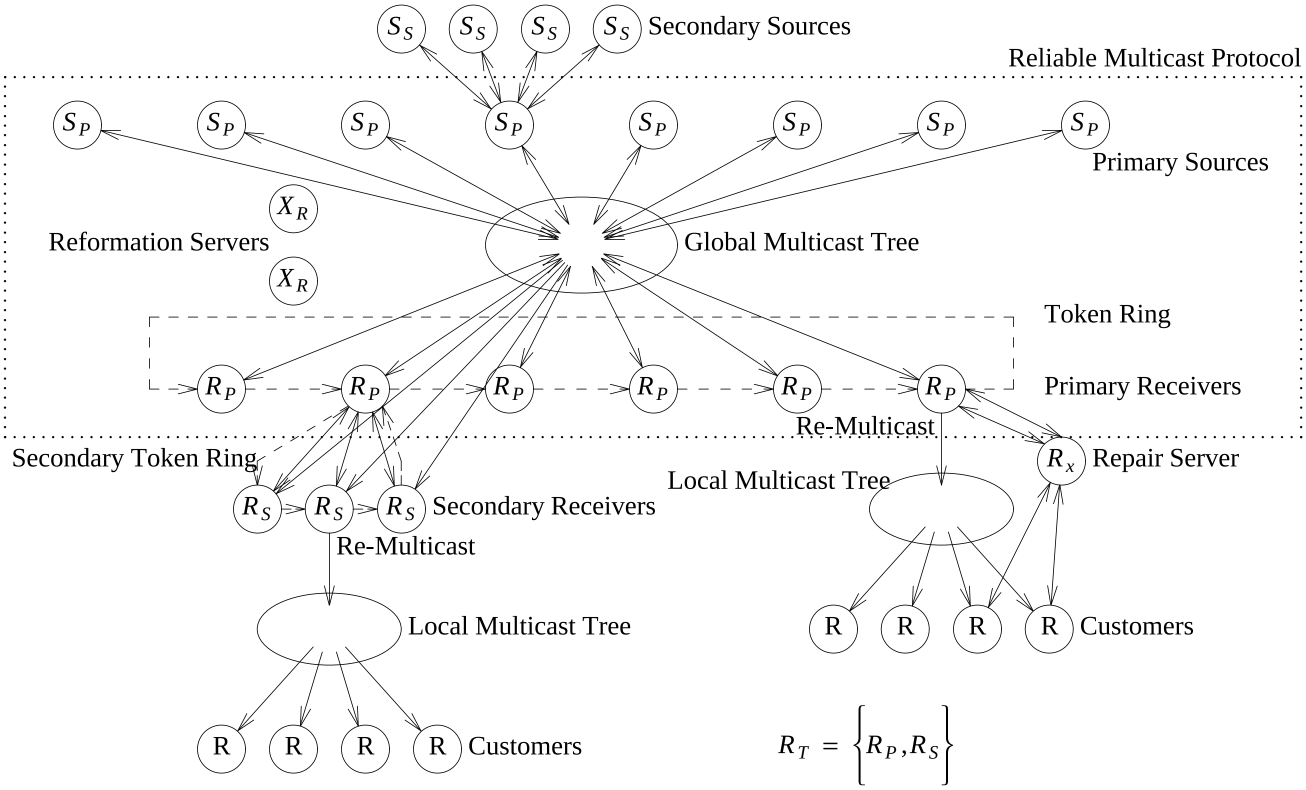

Figure 2. The general multicast architecture for the stock market applications.

In the upper layer of the receiver architecture the R_P use TRMP to recover messages over the long distance segments of the network that have large delays and high loss rates. This TRMP group should be limited to a few tens of receivers. When there are more recovery points, the logical token loop in TRMP is divided into a primary loop that is intercontinental, secondary loops, the R_S, that cover specified regions of the globe, and possibly tertiary loops that cover more restricted areas. This structure is similar to that proposed for physical loops in 1972 [9]. In a multiple loop configuration, TRMP is modified as described in section 4.2 so that each message is assigned a single sequence number.

In the lower layer of the architecture, we must supply the information to a large number of *R* who may join and leave the system frequently. The logical loops in TRMP are not appropriate in this environment since the *R* cannot be trusted to assist one another and frequent changes in the receiver set require frequent reformations. Instead, the R_T delay messages by a fixed time after they are acknowledged then retransmit the message sequence on a multicast tree to the R.

In the customer layer the propagation delay is insignificant, the network delays are small, and packets are lost less frequently. The raw remulticast messages may be sufficient for some applications. Applications that require recovery in this layer may use many of the reliable multicast protocols that have been proposed[15]. In fact, different reliable multicast protocols can be used in different areas of the network.

The separation between the upper and lower layers of the architecture provides a means of balancing the cost and quality of the network. On the one hand we would like to use the public Internet to reach a large number of customers economically, but on the other hand we would like to provide delay and bandwidth guarantees that are not currently available on the public network. A compromise is to use a private network for the upper layers of the architecture, that carry the data over the longest distances, and to use the public Internet for the remulticasts to the customers. The set of R_T receives data on the private network and remulticasts it on the public network. The lines in the private network are leased and managed in the same way as lines in an international corporate network. Bandwidth on the backbone network is not shared with the public network, and the quality is guaranteed. As network providers are able to guarantee the quality of service on virtual private networks [16,17], or when the public network is able to guarantee the quality of service across the many ISP's, the private lines may be replaced with shared facilities.

4.1 Receivers

The receivers R_T, take part in TRMP to recover and sequence the source messages. The acknowledged messages in the sequence are timestamped, and the R_T remulticast the sequence to the R after a delay that is sufficient to guarantee that all of the R_T have the message. The R_T are trusted to not to favor specific customers by giving them early access to the data.

Individual customers may join and leave an application frequently, but the set of R_T is stable. The R do not take part in the token passing protocol. Therefore, it is only necessary to reform a token group when an R_T fails or a new R_T is installed.

There is no limit on the number of R that can receive the multicast signal from an R_T. However, in order to provide receiver fairness the region of an R_T's remulticast should be limited by the loss characteristics of the network and the delay from that R_T to its R's. In addition, remulticast regions should overlap so that if an R_T fails an R can obtain the messages from an alternate R_T.

Increasing the number of receivers R_P in the primary token loop, m, decreases the size of the remulticast regions and improves the quality of the data delivered to the R. However, as m increases, the time to detect failed R_P increases. If m becomes very large the protocol becomes susceptible to NACK implosion because of the the correlated losses. We use a layered architecture, with secondary receivers, R_S, to improve the quality of the data delivered to the R without as large an increase in m.

The R_S receive the source messages and acknowledgements on the global multicast tree, just like the R_P, and remulticast the same sequence of messages. However, the R_S do not acknowledge source messages and do not take part in the primary token loop. When an R_S detects a missing acknowledgement or source message it requests the message from a specific R_P that is assigned to support it, rather than from the token site. Therefore, the R_S do not increase the time required to detect a failed R_P nor do they add to the NACK implosion on the global multicast tree.

A similar, hierarchical strategy for reducing NACK implosion has been used on a tree architecture in CRMP [18] and RMTP-II [19]. The tree architecture has layers of retransmit servers that focus on a single source at the root of the retransmit tree. The token site in our primary loop does not focus on a specific source, as noted in the source fairness criterion. Therefore, the multiple loop architecture is better suited for multiple sources than the hierarchical tree.

An issue that is not addressed in this paper is the best method for superimposing logical loops on the multicast tree to reduce the number of NACK's that are generated. While most mappings of multiple loops onto the multicast tree reduce the number of NACK's, some mappings appear to be more effective than others. In reference [20] we describe some simulations of international, multicast recovery architectures.

The R_S in a region pass a secondary token between themselves to detect failures and guarantee that all of the source messages are received by all of the R_S. The secondary tokens are numbered to correspond with the primary tokens and an R_S does not pass the token until it has the acknowledgements and source messages up to that token number. A site that receives a secondary token can make the same inferences about the sites in the secondary group as a site that received the primary token could make about sites in the primary group. Each secondary token ring includes an R_P on the primary token ring. That R_P can infer the state of the secondary group and transfer that information to the primary group.

As we increase the number of receivers in a secondary group we encounter the same problems that we encountered as we increased *m*. Two layers of receivers can only reduce the number of token passes required to detect failures or guarantee delivery, L_2, by the square root of the number of passes in a single layer architecture, L_1. If we have a group of 100 receivers $L_1 = 100$. If we organize the 100 receivers into 10 secondary groups of 10 receivers, each secondary group requires 10 token passes to circulate the token. The primary group also has 10 members. Therefore, we require 10 token passes to detect a failure and $L_2 = 10$. If we have 10,000 receivers, $L_1 = 10{,}000$ and $L_2 = 100$.

We can generalize the layered receiver architecture. With *i* layers, $L_i = \sqrt[i]{L_1}$. The land surface of the earth is about $57x10^6$ square miles. If we place 10,000 receivers uniformly over the surface, the maximum distance to a receiver is less than 50 miles, which is most likely adequate to provide our delay and loss guarantees. Since our stock market systems won't have to provide uniform access to the entire land surface of the earth, two layers of receivers should be more than sufficient to limit our token rings to a few 10's of receivers.

Within a region we can provide two grades of service, best effort and guaranteed delivery. Best effort only delivers the messages that are not lost in the regional area to the *R*. Guaranteed delivery recovers all of the messages. We guarantee delivery by colocating retransmit servers, R_x, with the R_P or R_S. When an *R* detects a missing sequence number, it can request the message from R_x.

4.2 Sources

The sources, S_P, are trusted to enter valid data messages into the reliable multicast group. There are authentication [21,22,23] and certificate granting systems that can extend trust to a large number of sources. However, we can keep tighter security with a smaller number of participants. In addition, TRMP requires that each receiver maintain state information for each source, such as the next expected message number, and some of the cryptographic techniques that we will use require receivers to share a secret with each source. It is difficult to maintain source specific information when the number of sources is large.

While the constraint on the number of sources isn't as severe as the constraint on the number of receivers, there should not be millions or ten's of millions of sources in an TRMP group. In order to keep the number of sources that participate in TRMP to a few thousand, the sources can also be layered: The S_P participate in TRMP; The S_P trust a set of secondary brokers, S_S; And, each S_S trusts a set of customers.

The set $S_T = \{S_P, S_S\}$ are trusted by the network. The degree of trust depends upon the security mechanisms that are used in the network, as discussed in section 6. The S_T can be owned by the network and operate as a firewall between the public Internet and the private backbone network, they can be privately owned sources that transmit data on the private backbone, such as the stock exchanges described in section 7.1, or they can be privately owned firewalls, such as licensed stock brokers. The S_T that operate as firewalls are responsible for verifying the authenticity of the customers or the data, as described in section 7.2. The S_T may each use different password or authentication systems. If the S_T are owned by the network, the network owner must determine if the different systems compromise the security of the network. If the S_T are licensed brokers, they may be required to act as insurers and accept responsibility for any data that they enter into the network.

4.3 Striping

The amount of data in the composite stock ticker will almost always exceed the bit rate that can be transmitted to an individual customer, *R*. The amount of data in the composite ticker may also exceed the amount of data that can be processed by an R_T in the repair group. Both of these problems are addressed by "striping" stocks into common groups. However, the stripes that are used to solve the two problems have different widths.

In the core of the network, stocks are organized in stripes of related stocks. A stripe is limited to a group of stocks that have few enough messages to be processed by all of the R_T. Each stripe uses a different multicast address and the entire network of R_T's is replicated for each stripe. Since the R_T are provided by the network, rather than the customers, we can assume that all of the processors are similar - no weak links - and that they are among the more powerful processors that are available. There should be a relatively small number of wide stripes in the core of the network.

An R_T organizes the data that it receives on a core stripe into narrower stripes of more closely related stocks and transmits each edge stripe on a different remulticast address. The amount of data on an edge stripe is limited by the least capable of the *R* and the more capable *R* receive multiple stripes. For instance, if the least capable *R* have 56 Kbps modems, R_T organizes the data into 56 Kbps stripes. A customer with a 56 Kbps modem may select any one remulticast address and receive information on a small group of stocks, while a broker with a 45 Mbps connection may simultaneously view the information on the stocks in 800 different stripes. There should be a relatively large number of narrow stripes at the edge of the network.

5. TRMP

The fairness requirements, time constraints, the quantity of data transferred, the distance spanned, the number of users served, and the availability requirements of the stock market applications require modifications of RMP. The timed reliable multicast protocol, TRMP, is the modified RMP. The major changes are

1. multiple loops,
2. delayed delivery,
3. time driven, rather than event driven, token passing,
4. NACK reduction, and
5. the reformation protocol.

Multiple loops result in multiple tokens. In order to guarantee that each source message is only acknowledged once, and that there is a unique sequence of messages, only one of the tokens transfers the right to acknowledge messages. The remaining tokens are circulated to test the receivers and to determine when we can guarantee that all of the receivers have specific messages.

In TRMP the R_T wait Δ_A between the time a message is acknowledged and the time that it is remulticast. The R_T have synchronized clocks. The clocks can be synchronized by a network protocol [24,25], or by receiving a timing signal from a satellite, such as the global positioning system (GPS), and adjusting for differences in the propagation delay. When the token site acknowledges a source message, it timestamps the acknowledgement. The R_T remulticast a message Δ_A after the message's time stamp. The delay Δ_A compensates for the difference in reception times at the different R_T, caused by differences in delay from the source and the time to recover missing messages. A system is fair when all of the R_T simultaneously retransmit the message to their local customers.

The propagation delay around the circumference of the earth is about 150 milliseconds and the network delays between receivers are at least a few hundred milliseconds. Therefore, we cannot transfer the token more than 2 or 3 times a seconds. If we only acknowledge a single source message each time that the token is passed, the message arrival rate in some of the stock market applications will exceed the limits of the protocol. In TRMP, the token is transferred periodically, every τ_t seconds, and acknowledges all of the unacknowledged source messages, including any messages that were missed by the previous token sites. TRMP does not have an upper bound on the message rate. The t^{th} token passing message acknowledges a sequence of k source messages, where k is variable. The k messages are assigned sequence numbers s to $s+k-1$.

We can acknowledge multiple source messages in an event driven protocol rather than passing the token periodically. However, periodic token transfers reduce the number of token transfers until we can guarantee that all of the operable receivers have the message from m, the number of receivers in the logical token loop, to "1". There is an informal proof of this property in section 5.1.

Another advantage of periodic token transfers is that the R_T detect a failed token site when a token isn't transferred on time. In the event driven RMP, sources detect a failed token site when a message isn't acknowledged. Removing failure detection and reporting responsibility from the sources makes it possible to operate with less trusted sources. In section 7.1 we note that this can be an important characteristic in an environment where all trading floors are not equally trusted.

NACK implosion is considered a serious problem in reliable multicast protocols. Our layered architecture reduces NACK implosion, as described in section 4.1. TRMP has an additional mechanism to reduce NACK implosion, described in section 5.2. The mechanism is based on token passing, and is not available in other reliable multicast protocols. This NACK reduction mechanism increases the delivery delay and will not be used in the stock market applications.

In the original reformation protocol communications in the entire system stopped during reformation. The stock market applications require a high availability, and it is necessary to keep the communications disruptions to a minimum. In the reformation protocol described in section 5.3, communications may only stop on a portion of the system. In addition, the reformations take much less time, so that communications disruptions are shorter.

5.1 Periodic Token Passing

In RMP a receiver does not know that it has missed an acknowledgement until it receives a higher numbered acknowledgment. RMP is event driven, and higher numbered acknowledgements aren't transmitted until the token site receives the next source message. The probability that a receiver is unaware of a missed acknowledgement is a function of the number of additional acknowledgements that have been transmitted, and decreases with time. There is a tradeoff between Δ_A and the fraction of the operable R_T that remulticast the message simultaneously. However, we cannot guarantee that all of the operable R_T can remulticast the message for any Δ_A.

In TRMP we can make the much stronger claim that all of the operable R_T remulticast the message simultaneously when $\Delta_A \geq \tau_t$. The claim holds when τ_t, the token passing time, satisfies the inequality $\tau_t \geq (2n_{\max} + 1/2)\,T_R$. In this relationship, T_R is the time between retransmission requests and $n_{\max}$ is the maximum number of recovery attempts before declaring a failure. When a single source and destination use a positive acknowledgement protocol, we cannot guarantee that the message is delivered to an operable receiver in less than $n_{\max}\,T_R$. Therefore, the guaranteed delivery time in our multicast network is only 2.17 times that on a point-to-point link, when $n_{\max} = 3$.

In dedicated networks we make T_R greater than the round trip delay through the network. However, in packet networks the delay is a random variable and can be virtually unbounded. The penalty for making T_R smaller than some of the round trip delays is that we occasionally declare an operable receiver inoperable and perform an unnecessary reformation. The penalty for increasing T_R is that we increase Δ_A, the delay until we obtain the stock information. Since reformations are not an expensive operation in our system we should not try to make T_R long enough to eliminate all unnecessary reformations.

Figure 3 is the extended finite state machine for TRMP. The transitions are labeled with the conditions that initiate the transition. The "*"'ed labels are the state variables that are modified when a transition occurs. All of the receivers are in state 1 at time t_e when the acknowledgement Ack(e) is scheduled to be transmitted. A receiver that does not receive Ack(e) before time $t_e + T_R/2$ moves to state 2. ($T_R/2$ is the nominal bound on the one-way network delay.) A receiver that has received the acknowledgement moves to state 4.

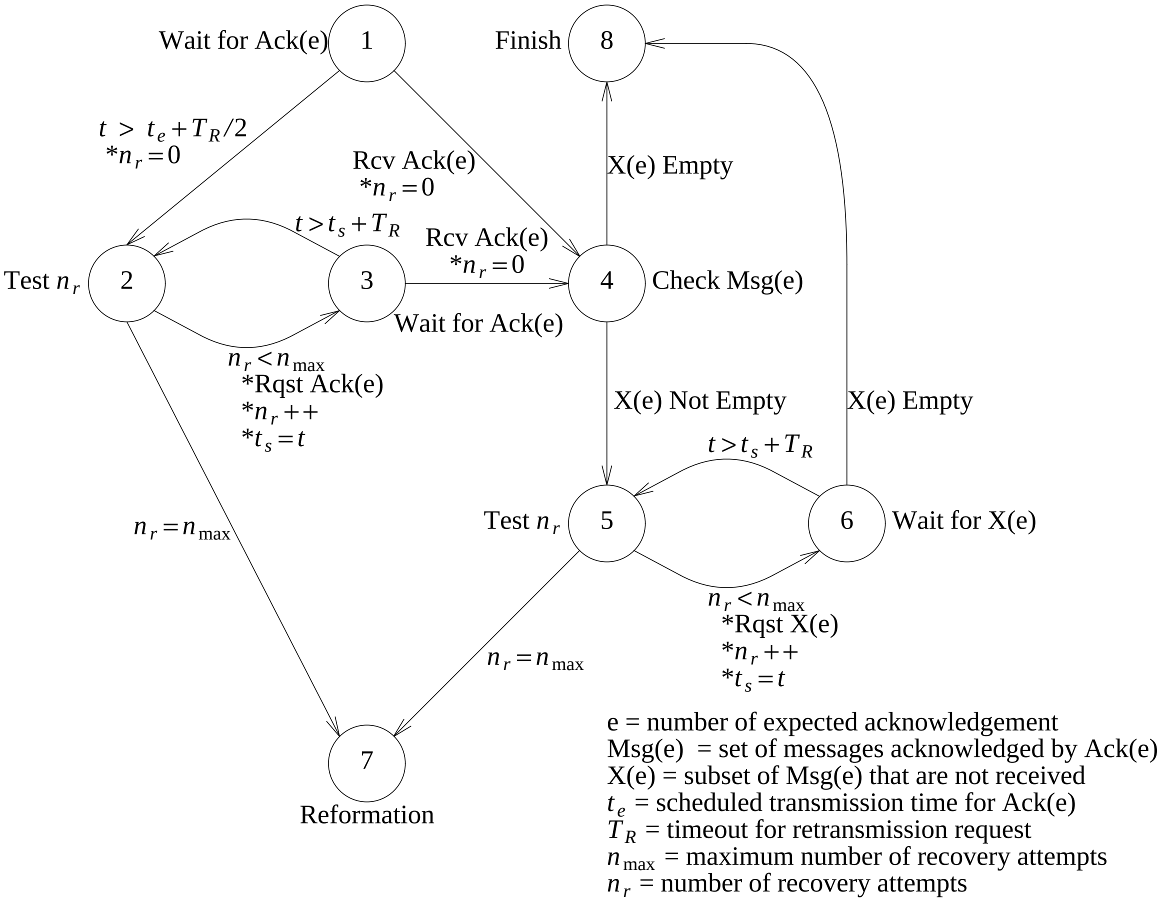

Figure 3. An extended finite state machine representation of acknowledgement processing at a receiver in TRMP

A receiver in state 2 that has made $n_{\max}$ or fewer requests, requests Ack(e) and waits in state 3. If Ack(e) is received within T_R seconds, the receiver moves to state 4, otherwise it returns to state 2. If the number of requests equals $n_{\max}$, the receiver declares that the token site has failed has moves to state 7. In most standards for positive acknowledgement protocols $n_{\max} = 3$.

By time $t_e + (n_{\max} + 1/2)\,T_R$ a receiver has either passed through state 4, or has entered a reformation. If the messages that are acknowledged by Ack(e) have been received, the receiver moves from state 4 to state 8, otherwise it moves to state 5. The operation of states 5 and 6 are the same as states 2 and 3. Within time $n_{\max}\,T_R$ after entering state 4, a receiver is in state 8 or 7.

Therefore, by time $t_e + (2n_{\max} + 1/2)\,T_R$ all of the receivers are in state 8, or one or more of the receivers have declared that the token site has failed. When $\Delta_A \geq (2n_{\max} + 1/2)\, T_R$, either every operable R_T has the acknowledged messages and remulticasts them simultaneously, or the system is being reformed. If the token passing interval $\tau_t \geq (2n_{\max} + 1/2)\, T_R$ we can guarantee that, if the system is not being reformed, no receivers have to recover Ack(e) or Msg(e) at $t \geq t_{e+1}$ (where $t_{e+1} = t_e + \tau_t$). Since the next token site has recovered all of the preceding messages and acknowledgements by time t_{e+1}, it sends Ack(e+1) on time.

There are a number of things that we can do to make Δ_A smaller:

1. Instead of counting $n_{\max}$ independently in states 2 and 5, we can test the sum of the retrys in both states. For instance, if we allow a maximum of 5 retrys in both states 2 and 5, rather than a maximum of 3 retrys in state 2 and 3 retrys in state 5, Δ_A is reduced by T_R. The probability of entering state 7 when the token site has not failed may also be reduced by limiting the sum of the retrys.

2. We can schedule $t_{e+1} < t_e + (n_{\max} + 1/2)\, T_R$. Most of the time the next token site will be ready to transmit on time, but occasionally it will be late. When the token site transmits the acknowledgement later than scheduled, the retransmit timers at the receiver start before the acknowledgement is available and the probability of entering state 7 when the token site has not failed, increases.

3. We can recover both Ack(e) and Msg(e) in states 2 and 3, even if only one is missing. This increases the amount of data retransmitted, but cuts out half of the retrys.

These protocol modifications will be investigated in a future work. Our objective is to recover all of the messages as quickly at all of the multicast receivers as we can between a single source to destination.

5.2 NACK Reduction

The standard mechanism for reducing NACK implosion in reliable multicast systems is to limit the subset of receivers that request a missing message, but to multicast the missing message to all of the receivers. In subsequent intervals of time different subsets of receivers can request the missing message until all of the receivers have had an opportunity. If a receiver requests a message and a receiver that is scheduled to request the message in a later interval receives the multicast, the later receiver does not request the retransmission, and the number of NACK's is reduced. This strategy is particularly useful in the Internet where the multicast is transmitted on a tree and many receivers miss the same message.

In TRMP the receiver that accepts the token must have all of the previous messages. Therefore, when we define subsets of receivers that request missing messages, we must guarantee that a receiver has an opportunity to recover a missing message before it becomes the token site.

The simplest and most economical method of defining the subsets in TRMP is to have one receiver in each subset, the next token site. A receiver only requests missing messages before it becomes the token site, The disadvantage with this approach is that a site may have to wait an entire token rotation before it can recover a missing message. The number of token passes before we can guarantee that all of the operable TRMP receivers have a message increases from 1 to *m*.

We can reduce the time until a site recovers a missing message by giving several sites the opportunity to request the missing message The maximum time until a receiver can request a missing message is minimized when we space those sites equidistant around the token ring. Specifically, if token *t* is sent by receiver *r*, define sets of receivers $S_{i,t} = \left\{ (r + i + 1 + j * k_p) \bmod m \text{ for } 0 \leq j \leq (m - i - 1)/k_p \right\}$, for $i = 0, 1, \ldots, k_p - 1$, where there are *m* receivers numbered 0 to $m - 1$ in the token group. A receiver in $S_{i,t}$ can request the acknowledgement sent during interval *t*, or the source messages that it acknowledged, during interval $t + i$, if they are still missing. With this assignment, we can guarantee that every receiver has a message within k_p token passes.

If m/k_p is an integer, each receiver requests any missing messages in the interval it is scheduled to accept the token, and every k_p^{th} interval after that. In the other $(k_p - 1)^{th}$ intervals, the receiver listens in case one of its missing messages is requested by another receiver.

Since receivers in the later sets do not request a missing message when receivers in earlier sets request that message and the retransmission is received, the average number of requests for retransmission is clearly reduced. If, on the average, there are more sites that miss the message than there are sets, we can further reduce the average number of requests by putting fewer receivers in the sets that make the initial requests than in the sets that make later requests. We can "tune" the number of receivers in each set so that, on the average, the probability of a request is the same in each subset. We can also reduce the number of requests by placing receivers in different sets if they are likely, because of their positions on the original multicast tree, to miss the same messages.

A problem with limiting the number of receivers that transmit a NACK is that it increases the average delay until a missing message is acquired. For this reason we do not recommend this NACK reduction mechanism in the stock market applications. In a different application we may replace fairness with a penalty for delay, and this NACK reduction mechanism may be useful.

5.3 Reformation

The reformation process is initiated by the positive acknowledgement protocols that are part of RMP or TRMP. In positive acknowledgement protocols the source assumes that the receiver has failed if it does not receive an acknowledgement after a specified number of retransmission attempts. In RMP, token site failures are detected when an S_P fails to receive an acknowledgement for a message or an R_T cannot recover a missing message or acknowledgement. A failure in the next token site is detected when the current token site cannot pass the token. Since there are no control messages transmitted by the R_T when there are no messages from the S_P, we must depend on the S_P to detect token site failures in a quiet system.

TRMP transfers the token periodically. In this protocol we do not depend on the S_P to detect failures. Instead, all of the R_T detect a token site failure if the token is not passed on time. In RMP the S_P could act maliciously and disable the system by continuously putting it into a reformation, or could neglect to restart a system that has lost the token. In TRMP the S_P can be less trusted, with the advantages that are noted in section 7.

The RMP reformation protocol is completely distributed. When any sources or receivers detect a failure, they initiate an election protocol to select a single site to lead the reformation. The leader initiates a three phase commit protocol in which it determines which receivers are operable, which acknowledged messages can be committed, it forms a new token list, and finally gives one receiver the token. No source messages are acknowledged during the reformation. If there are a large number of receivers, and the set of operable receivers changes frequently, the system availability can be very poor.

The reformation protocol in the stock market applications is centralized. When a receiver detects a failure it notifies a reformation server, X_R. There are redundant X_R in case one fails. The X_R is responsible for forming a new token loop on the particular primary or secondary loop that has changed. There is no election protocol.

To form a new token loop, the X_R performs a straightforward loop bypass or insertion. All of the receivers in a logical loop are numbered. If X_R receives a report that receiver r has failed, X_R instructs receiver $(r-1) \mod m$ to transfer the token to $(r+1) \mod m$, and gives the token to $(r+1) \mod m$. If either or both of these receivers have failed, X_R selects the next or prior receiver that has not failed, and instructs those receivers to bypass r. When receiver r recovers, it contacts X_R and asks to be reinstalled in the token list. X_R notifies $(r-1) \mod m$, or the previous operating receiver, to pass the token to r, and instructs r to pass the token to $(r+1) \mod m$, or the next operating receiver.

The numbering scheme on a sub-loop is internal to the X_R. The receivers pass the tokens using the network address for the other receivers. The receivers that X_R calls r and $(r+1) \mod m$ are at the network addresses A_1 and A_2. During normal operation, the receiver at A_1 transfers the token to the receiver at A_2. By using network addresses, the strategy for bypassing or re-inserting receivers can also be used to change the system when new receivers are added or when an existing receiver is retired. If a new receiver, at address A_3 is added to the token loop following *r*, X_r must increase *m* and all of the receiver numbers greater than *r* by one, but must only notify receivers A_1, A_2, and A_3 about the addition to the token loop.

The primary reason for using this reformation procedure is that it is less disruptive of the information flow than the original protocol. The centralized protocol restores a lost token more quickly than the distributed protocol. The centralized protocol doesn't have an election phase to determine X_R. Furthermore, since the centralized procedure fixes one fault at a time, it only communicates with two receivers rather than using a three phase commit procedure to determine and order all of the operable receivers. In fairness, the original reformation protocol was more concerned with guaranteeing the state of a distributed database than with resuming communications as quickly as possible.

A second reason why the reformation process is less disruptive is the because the network is organized into a hierarchy of loops, rather than a flat structure. When a failure occurs on a secondary loop, the primary loop continues to operate and most of the system continues to acknowledge and order source messages during a reformation. The affected loop catches up as soon as the failed token site is bypassed. The only time that we stop acknowledging source messages is when the token is lost on the primary loop. There are fewer components on any of the sub-loops with the hierarchical structure than the flat structure. Therefore, any sub-loop, and in particular the primary loop, enters a reformation process less often.

Another reason for using the centralized protocol is that the reformation server in this system must have access to more information than in the RBP applications. The reformation server must know the structure of the sub-loops in order to perform a simple bypass. If a bypassed receiver *r* is at a junction between two sub-loops, X_R must assign the responsibility for joining the sub-loops to a surviving receiver. In addition, the reformation operation must adjust the multicast regions (time to live fields) and possibly change the remulticast addresses, so that every customer receives at least two multicasts on different addresses. The information about, and state of, the system is easier to maintain in a small number of X_R's than in all of the R_T's.

We have ignored a number of difficult design problems. In the stock market system, particularly on the intercontinental loop, there may be an advantage to a particular ordering of the receivers. Determining the time to live fields and addresses used for the re-multicasts and deciding which receivers assume remulticast responsibilities for others is also difficult. At present we assume that the design is performed manually. However, as the system grows this process should be automated. In the CRMP system, the topologies were designed manually until enough experience was obtained to determine which parameters should be optimized[20]. The same practice should be followed in the stock market system.

6. Security

In section 3 we listed the following security concerns:

1. constraining transmission access to authorized sources,
2. preventing early reception of the data stream,
3. limiting reception to authorized receivers,
4. spoofing or adding to the repaired sequence, and
5. denial of service.

The first two concerns address the core of the system, where TRMP operates. The global multicast group in the core is $G_M = \{S_P, R_T\}$. The final three concerns address the remulticast data at the edge of the system.

In this section we expand on these five concerns and map them onto known networking or cryptographic problems. There are existing solutions for each of these problems and there is also active research on many of them. We have not specified any problem that has to be uniquely solved for our applications. Although, we can make good use of time-lock puzzles [26,27] that release information after a delay. The advantage in reducing our security issues to known problems is that we can use any refinements in the solutions to these problems and we can also make use of future standards.

The first security concern is that a source outside our set of S_P will transmit messages that are placed in the sequence of messages that are remulticast. In a stock market applications one trader may find advantage in providing other traders with mis-information. We can address this concern with cryptographic techniques, networking techniques, or both.

In our architecture there are only a few hundred S_P. If each S_P shares a secret key with the group of R_T and encrypts its messages, we can operate G_M on the public Internet. When each S_P has a different key, the encryption also identifies the source. This reduces the amount of trust that we must place in the S_P, since an S_P that is inserting mis-information cannot masquerade as another source.

Alternatively, we can operate G_M on a private network, or a virtual private network. The set of R_T receives data on the private network and remulticasts it to the customers on the public network. In applications where there are many more sources than the S_P, the S_P act as firewalls [28] between the public network and the private network, and verify the right of the customers to place a message on the private network. Network providers currently prevent external access to international, corporate private networks and are proposing techniques to protect virtual private networks.

The second security concern is that an unauthorized receiver will eavesdrop on G_M. The R_T delay the multicast messages before they are remulticast to R so that all of the receivers get the messages simultaneously. Clearly, in the stock market applications, investors can take advantage of obtaining information on trades before other investors. This concern can also be addressed by networking or cryptographic techniques.

If the S_P encrypt their transmissions with a key that can only be decrypted by the R_T, the information is protected from the R and can be transmitted on the public network. The R_T are owned by the network and are trusted not to divulge the messages early. If the S_P each use a separate secret, they do not have to be trusted not to divulge the information from the other S_P before the delay imposed on the R_T. Reducing the degree of trust of the S_P may be significant in a distributed stock ticker if the trading floors are given direct access to the core network, as noted in section 7.

The original multicasts are only available on the core of the network. Therefore, the private networking techniques that constrain transmission on the core network also prevent receivers that do not have access to the core from gaining early access to the information. If the network provider is trusted to prevent eavesdropping, the degree of security that is obtained with a firewall can be equal to other cryptographic techniques.

The third security concern is to restrict access to the data that is remulticast by the R_T. There are electronic stock markets, like NASDAQ, that require the R to be part of a private network in order to protect access to the data. However, our objective is to make our system accessible to the general population, less expensively, by using the public Internet to connect the R. A stock market application can sell the remulticast sequence over the public Internet by the month, like a subscription for a newspaper. To sell the sequence, the R_T decrypt the messages from the S_P, then re-encrypt the entire sequence with a new key. The key is sold to each of the R, and is changed when the subscriptions expire.

The decryption key is sold to a large number of customers, and we must discourage someone who buys the key from giving it to others. The problem of selling and distributing a remulticast key has been encountered in an earlier electronic publishing system [29,30]. In that system the decryption key is included in a program that is sent to each subscriber. A subscriber pays for the service with his credit card, and the key in the program is masked by the credit card number. In order to give away access to the data, a customer must give away his credit card number. While this does not prevent a person from giving away the program, it should discourage most people.

The final two security concerns are similar to the first concern, but occur on the remulticast groups on the public Internet. A malicious user may insert false messages into the sequence or may flood the multicast group to prevent others from receiving the repaired sequence.

The conventional cryptographic approach for dealing with pretenders is digital signatures. Since there are a large number of untrusted receivers, the digital signature should use public key cryptography [31,32]. In a public key system only the remulticast source can sign the message, but any receiver can verify that the message is legally signed. Unfortunately, public key systems require more computation than we should perform in this application.

We can use the message numbering in our system as a partial alternative to signatures. In most instances it is much easier to insert messages than it is to delete messages. The numbering provides a means of detecting added messages. We cannot tell which of the duplicate numbers are real, but we can decide that some messages are forgeries and not act on any of the information.

We cannot use cryptographic techniques to prevent an attacker from flooding the remulticast group with a large number of phony messages. Although we may detect the attack, the attacker can deny service to the R. Since there is only one source, R_T, in each remulticast group, we can eliminate illegal transmitters by configuring routers to only multicast the signal from a particular source on a particular port.

7. Applications

The three stock market applications are

1. a unified ticker of the transactions from a number of physical and electronic trading floors,
2. a merged stream of buy and sell orders, and
3. a distributed trading floor.

The first application has a relatively small number of sources and a very large number of receivers. The second application has a very large number of sources and a relatively small number of receivers. The third application has the same number of sources and receivers, both of which may be large.

In the first two applications we are primarily interested in fairness. TRMP is used to create a level playing field for investors independent of their location. In the third application we are also interested in providing the same sequence of messages to every receiver, so that receivers can independently determine which transactions have occurred. In addition, the third application requires guarantees that a specified number of operable receivers have witnessed a transaction, so that the transactions can survive system failures.

7.1 Unified Ticker:

In the unified ticker every investor receives a list of the trades on every trading floor. The objective is to create a level playing field where all of the investors have the same information on trades. The investors receive the list in the same order, at the same time, no matter where in the world they are located.

The sources, S_P, are the trading floors. The trading floors operate independently under their own rules and customs. Some may be physical places, others may be Internet servers, and still others may be the distributed trading floors described in the third application.

The core network is a private network. The bandwidth on this network is guaranteed, and firewalls protect the network from mischief. The trading floors are inside the firewall and multicast their list of trades directly on the the TRMP group. Each trading floor shares a different secret key with the group of receivers, R_T, and encrypts the messages that it places in the multicast sequence.

The encryption serves two functions: First, it acts as a signature of the trading floor that has entered the data, and second, it prevents a trading floor from acquiring and using the information from the other floors before it is available on the unified ticker. The second function is important because the trading floors may not be equally trusted. Many of the new electronic exchanges have not had time to establish trust, and the different floors in an international system have different regulatory agencies, with different rules and penalties. The trading floors are only trusted to transmit an honest and timely accounting of their trades and not to divulge their own trades before they are reported on the unified ticker. We assume that this degree of cooperation can be enforced by the regulatory agencies or by the fear of being excluded from the unified ticker.

The multicast receivers, R_T, are owned by the network operator and are trusted to protect the unified ticker until it is scheduled to be distributed. At Δ_A after the timestamp all of the R_T decrypt a source message and remulticast that message in their regional areas. The remulticasts are outside the firewalls of the private network. If the unified ticker is being sold, the R_T re-encrypt the ticker with a distribution key, as described in the previous section.

The remulticast messages may be lost. An R detects lost messages by gaps in the sequence numbers. Missing messages can be acquired from R_x. The data transmitted in a unified ticker is temporal and many customers may only be interested in the most recent stock prices. Once the next value of a stock is received, there is no reason for these customers to retrieve the previous price. There may be other customers who wish to accurately plot the stock price to predict trends. These customers may retrieve the missing transactions.

There is an expense associated with maintaining R_x. The network provider can recover the expense by selling two levels of service, one with and one without retransmissions. It is likely that the same messages will be lost by many receivers in a region. Therefore, retransmissions should also be multicast, rather than sent by point-to-point communications to select receivers. The retransmissions can be restricted to a subgroup of the original R by encrypting them with a different key than the original multicast. The retransmit key can be sold separately, but by the same technique as the multicast key, so that only receivers that pay for the higher level of service can decrypt the retransmitted messages.

The amount of data in a unified stock ticker will be significantly greater than the amount of data that a typical user can receive. This problem is addressed by organizing the data into "stripes" of related stocks. Different customers may have very different rate connections to the network, and the size of the stripes is determined by the least capable receivers that are supported. If the least capable receiver is 56 Kbps, the stocks in a stripe are restricted so that their composite data rate is unlikely to exceed 56 Kbps. Each stripe is transmitted on a different multicast address. A trader with a 56 Kbps modem can only select one stripe at a time while a trader with a 1.5 Mbps line may simultaneously follow the stocks on 25 stripes.

Striping may also be necessary in the backbone network if the data rate of the composite ticker exceeds the throughput of the R_T. The R_T are owned by the network. It is unlikely that some R_T will act as severe bottlenecks and reduce the size of the stripes much below the size required by the other R_T. The stripes in the backbone network will be much wider than those at the edge of the network. The entire multicast infrastructure, R_P's, R_S's, and R_x's, is duplicated for each stripe. The S_P's transmit transactions involving different stocks on the appropriate stripe.

7.2 Unified Orders

The unified order system is a sequence of offers to buy or sell stocks at a given price. The offers can be directed to a specific exchange or can be open to all participating exchanges. Our objective is to give all of the traders a fair opportunity to place their bids in the sequence of offers.

If the buy and sell offers are directed to a single exchange, the order of the sequence may be binding on the trades that occur. If the offer is open to all exchanges, the offer may just be an invitation for a broker to close a deal.

This system is the inverse of the unified ticker. There are many sources and a small number of receivers. In the degenerate case there is one receiver, a single trading floor. It may seem wasteful to circulate the token among the R_P that are distributed around the world, just to give the sequence to a single trading floor in a single location. However, the circulating token provides fairness.

If all of the sources transmit their offers directly to the trading floor, the sources that are in the same city as the exchange have an advantage over sources on the other side of the world. First, the propagation and network delays may be seconds shorter for the closer source, and second, the average number of transmission retrys may be significantly smaller for the closer sources.

In a conventional communication system, if two traders in different parts of the world simultaneously try to enter a bid, the trader in the same city as the trading floor will almost always have its bid registered first. With a rotating token, the portal that allows messages to enter the system spends equal amounts of time at different locations on the globe. The trader that gets into the system fastest depends on where the portal is located when the traders decide to enter their offers, and not where the trading floor is located.

The sources in this application include brokers and individual traders as well as other trading floors. These sources cannot be trusted to the same extent as the trading floors in the unified ticker. The sources may make offers without the proper resources, or may transmit a large number of messages to disrupt the system. In addition, there may be too many of these sources for the R_T to have a different shared secret with each.

Both of these problems are solved by not giving the sources direct access to the multicast group. The sources, S_P, are either owned by the network or are completely trusted. These are the only sources inside the network firewall. The sources, S, must present credentials to the S_P that they own the stock that they would like to sell or that they have the funds that they would like to spend. Alternatively, the S may have accounts with the S_P, in which case they must prove their identity by an agreed upon password system. If there are too many S for the network based S_P to track, there can be secondary sources, S_S, that trust the S and are trusted by the S_P.

7.3 Distributed Trading Floor

The third application uses TRMP to construct a distributed, international trading floor. The participants may be individual traders, brokers, or the other trading floors. This trading floor may also be one of the sources that reports trades in the unified ticker. All of the participants enter buy, sell or stop orders and see the same sequence of orders from all of the participants. Depending on the sequence, each participant knows which trades have occurred.

This application has many of the problems of the previous two applications. There are large numbers of sources and receivers, none of which is trusted. Both the S_P and R_T are network based and are distributed around the world to provide fair entry and distribution of the data. The S_P verify the credentials of the S and enter the bids. Based upon the RMP sequence and the rules of the particular trading floor, an arbiter declares which trades have been made and reports the trade on an appropriate ticker. By making the token site the arbiter we guarantee that the arbiter has the most complete acknowledged sequence of buy and sell orders.

There are a number of different rules that we can use to make trades. Some of the differences between the rules are semantic. If one participant offers to buy a stock at price A and another offer to sell the stock at price $B < A$, should the trade be made at A, B, or somewhere in between. Other differences in the rules are a matter of style. Some floors may post buys and sells, others may be run a single round, high bid auction, where the highest bidder gets to buy the stock at the price offered by the second highest bidder. Other floors may be modeled after the Amsterdam flower auction.

TRMP offers very strong guarantees that can be used to make trades reliably even when there are system failures. For instance, assume that the token site is the arbiter. An R_P that has the token can report a tentative trade that is based upon the bids that have been acknowledged, and the bids that are about to be acknowledged. The next token site guarantees that two operable R_P have recorded the trade. By waiting N token passes after a trade is reported, before committing the trade, we can guarantee that information on the trade will not be lost when there are up to N failures.

Note that guaranteeing that N operable receivers have a message is different from guaranteeing that all of the operable receivers have a message. We may suspect that there are more than N operable receivers, but at any instant in time we cannot guarantee that there are more than N operable receivers. By passing the token we can guarantee $N-1$ token passes later that there were N operable receivers.

8. Conclusions

We have described an architecture and protocols for an Internet-based global stock exchange. The architecture creates a hierarchy of transmitters and receivers that address the security concerns of the stock market and resolve the scaling problems associated with RMP. RMP has been modified for this application. The principle change converts RMP from an event driven protocol to a time driven protocol, TRMP. TRMP guarantees that all of the operable receivers have an acknowledged message within a single token passing time.

In this work the token passing time is only 2.17 times the time required to guarantee delivery in a positive acknowledgement protocol that operates between a single source and destination. In section 5.1 we describe techniques that may reduce the guaranteed delivery time toward that required for the single source-destination pair.

There are other open issues involving the protocol. In this work the events in the protocol are scheduled to occur at particular times. In the introduction we state that the protocol should operate nearly as well when we know the interval between events as when we know the times of the events, but we have not addressed the use of relative timing. Timing accuracy is also an open issue. In the stock market application, we can justify using moderately expensive GPS units to obtain very accurate timing for the network owned components. However, as we apply the protocol to other applications we should determine the effects of the timing inaccuracies that occur when we derive timing from the data stream.

On a more general level, there are fewer tools for analyzing, verifying correctness, or performing conformance testing on timed protocols [33,34,35,36] than on protocols that can be described as state machines. This is becoming an increasingly important issue. As we use protocols to provide quality of service guarantees, the protocols must use time. Timed protocols operate over the continuous dimension of time as well as a finite state space, and are more difficult to reason about than finite state machines. The types of tools that have been designed for finite state machines must be extended to continuous time.

There are also open issues related to the architecture. In this paper we assume that the network based components are manually placed and organized into a hierarchy of logical loops. Placement of components is a complicated issue that is determined by the distribution of customers, the topology of the underlying network, the expected growth patterns, and many other factors. Until we have a better understanding of all of the factors that determine the best solution, a human designer will most likely design better networks. However, our network is likely to become large and we expect it to change frequently, so we must at least consider automated design assistance.

At present we bypass single failed components. We must consider the advantage of performing more complicated reconfigurations when components fail or are restored. We should also consider moving receivers between the different levels of the hierarchy of loops. In a network where the hierarchy of loops span different distances, it may make more sense to replace a failed receiver by a nearby receiver, in a lower level of the hierarchy, than to bypass it by two distant receivers in the same level of the hierarchy.

Finally, there are operational issues that must be addressed. We consider a static system in which all of the components are at the same technological level and none of the components are evolving to the next technology. In fact, we must consider both migrating the current stock markets to this technology and have a plan for replacing components as new technologies become available. These issues are discussed in reference [37] as they apply to the electronic system that is currently being used for NASDAQ.

REFERENCES

[1] S. Deering, "Multicast Routing in Internetworks and extended LAN's," ACM SIGCOMM '88, Aug. 1988, Stanford, Calif., pp 55-64.

[2] R. Piantoni, C. Stancescu, "Implementing the Swiss Exchange trading system," Int. symp on Fault-Tolerant Computing, 1997. FTCS-2, June 24-27, 1997, pp. 24-27.

[3] K. P. Birman, M. Hayden, O. Ozkasap, Z. Xiao, M. Budiu, Y. Minsky, "Bimodal multicast," ACM Trans. on Comp. Syst., Vol. 17 , No. 2, 1999, pp. 41-88.

[4] K. Birman, R. van Renesse, "Reliable distributed computing with the Isis toolkit," IEEE Computer Society Press, 1994.

[5] R. van Renesse, K. Birman, S. Maffeis, "Horus : a flexible group communication system," Communications of the ACM, Vol.39, No. 4 (April 1996), pp. 76-83.

[6] J-M. Chang, N. F. Maxemchuk, "Reliable Broadcast Protocols," ACM Transactions on Computer Systems, Vol. 2, No. 3, Aug. '84, pp. 251-273.

[7] S. Paul, K. Sabnani, J. Lin, S. Bhattacharyya, "Reliable Multicast Transport Protocol (RMTP)," IEEE JSAC, vol. 15, no. 3, Apr. 1997, pp. 407-421.

[8] N. F. Maxemchuk, K. Padmanabhan, S. Lo, "A Cooperative Packet Recovery Protocol for Multicast Video," Int. Conf. on Network Protocols, Oct. 29-31 1997, Atlanta Georgia. pp. 259-266.

[9] J. R. Pierce, "How far can loops go," IEEE Trans on Comm., Vol. COM-20, No.3, June 1972, pp 527-530.

[10] J-M. Chang, "Simplifying Distributed Database Systems Design by Using a Broadcast Network," Proc SIGMOD '84, pp 223-233, June 1984.

[11] N. F. Maxemchuk, J-M. Chang, "Analysis of the Messages Transmitted in a Broadcast Protocol," Proc ICC '84, pp 1263-1267, May, 1984.

[12] B. Whetten, T. Montgomery, S. Kaplan, "A High Performance Totally Ordered Multicast Protocol,," Theory and Practice in Distributed Systems, number 938 in LCNS, Springer Verlag, 1994.

[13] S. Pejhan, M. Schwartz, D. Anastassiou, "Error Control Using Retransmission Schemes in Multicast Transport Protocols for Real-Time Media", IEEE/ACM Trans. on Networking, vo. 4, no. 3, June 1996, pp. 413-427.

[14] L. Delgrossi, C. Halstrick, R. G. Hertwich, H. Stuttgen, "HeiTP - a transport protocol for ST-II," Proc of IEEE Globcom '92, 1992, pp. 1369-1332

[15] C. Diot, W. Dabbous, J. Crowcroft, "Multipoint Communications: A Survey of Protocols, Functions and Mechanisms," IEEE JSAC, Apr. 1997, vol. 15, no. 3, pp. 277-290.

[16] P. B. Busschbach, "Toward QoS-Capable Virtual Private Networks," Bell Labs Technical Journal, vol. 3, no. 4, pp. 161-175, Oct-Dec 1998.

[17] H.T. Kung, S.Y. Wang, "TCP Trunking," Harvard University, Boston, Massachusetts, Nov. 1998.

[18] K. D. Gunther, "Prevention of Deadlocks in Packet-Switched Data Transport Networks," IEEE Trans. on Commun., vol. COM29, no. 4, Apr. 1981, pp. 512-524.

[19] B. Whetten, G. Taskale, "An overview of reliable multicast transport protocol II," IEEE Network Mag., Jan/Feb 2000, pp 37-47.

[20] D. Rubenstein, N. F. Maxemchuk, D. Shur, "A Centralized Approach to Network Repair Service for Multicast Streaming Media," Proceedings of NOSSDAV 2000, June 26-28, 2000, Chapel Hill,

N.C., pp. 173-182.

[21] J. G. Steiner, C. Neuman, J. I. Schiller, "Kerberos: An Authentication Service for Open Network Systems," USENIX Winter Conference, Feb. 9-12, 1988, Dallas Texas, pp. 191-202.

[22] B. Clifford, T. Tso, "Kerberos: An Authentication Service for Computer Networks," IEEE Communications Magazine, September 1994, Vol. 32, No. 9, pgs. 33-38.

[23] L. Harn, H-Y. Lin, "Key Management for Decentralized Computer Network Services," IEEE Trans. on Commun., Vol. 41, No. 12, Dec. 1993, pp. 1777-1779.

[24] D. L. Mills, "Internet Time Synchronization: The Network Time Protocol," IEEE Trans. on Communications, Vol. 39, pp. 1482-1493, Oct. 1991.

[25] D. L. Mills, "Improved Algorithms for Synchronizing Computer Network Clocks," IEEE/ACM Trans. on Networking, Vol. 3, No. 3, pp. 245-254, June 1995.

[26] G. Di Crescenzo, R. Ostrovsky, S. Rajagopolan, "Conditional Oblivious Transfer and Time Released Encryption," Proceedings of Eurocrypt ’99, Springer-Verlag, pp. 74-89.

[27] R. Rivest, A. Shamir, D. Wagner, "Time-lock puzzles and time-released Crypto," MIT Laboratory for Computer Science report, Mar. 10, 1996.

[28] B. Cheswick, "The design of a secure Internet gateway," Proc. of Usenix Summer Conference, (Anaheim, California), pp. 233--237, Jun. 1990.

[29] N. F. Maxemchuk, "Electronic Document Distribution," ATT Technical Journal, vol. 73, no. 5, Sept. 1994, pg 73-80.

[30] J. T. Brassil, S. Low, N. F. Maxemchuk, "Copyright Protection for the Electronic Distribution of Text Documents," Invited Paper - Proceedings of the IEEE, July 1999, vol. 87, no. 7, pp. 1181-1196.

[31] M. Willett, "A Tutorial on Public Key Cryptography" Computers & Security I (1982) pp. 72-79, North-Holland Publishing Company.

[32] W. Diffie, "The First Ten Years of Public-Key Cryptography," Proceedings of the IEEE, Vol. 76, No. 5, May 1988, pp. 560-577.

[33] P. A. Miller, P. N. Turcu, "Generic Signaling Protocol: Architecture, Model and Services." IEEE Trans. on Communications, May 1992, Vol. 40, No. 5, pp. 957-966.

[34] P. A. Miller, P. N. Turcu, "Generic Signaling Protocol: Switching, Networking and Interworking." IEEE Trans. on Communications, May 1992, Vol. 40, No. 5, pp. 967-979.

[35] R. Alur, D. L. Dill, "A Theory of Timed Automata," "Theoretical Computer Science," v. 126 pp. 183-235 (1994).

[36] M. Yannakakis, D. Lee, "An Efficient Algorithm for Minimizing Real-Time Transistion Systems," Formal Methods in System Design 11, 1997, pp 113-136, Kluwer Academic Publishers.

[37] B. Oki, M. Pfluegl, A. Siegel, D. Skeen, "The Information Bus: an architecture for extensible distributed systems," Proc. of the 14th ACM symp. on Operating systems principles, 1993, pp. 58-68.