An Overview of the VMTP Transport Protocol

Carey L. Williamson
David R. Cheriton
Computer Science Department
Stanford University
Stanford, CA 94305

Abstract
Changes in communication use and technology have motivated a desire for a new transport protocol. For example, communication in modern distributed systems demands low latency transaction-oriented communication, rather than stream-oriented communication as in the past. The performance and functionality of current standard transport protocols has become a major limitation in the move to higher speed networks and larger scale, more sophisticated distributed systems.

This paper provides an overview of the Versatile Message Transaction Protocol (VMTP) developed to address these limitations. We then present measurements of VMTP performance in actual use in the V distributed system, showing that its performance matches our objectives.

1 Introduction
With the development and increasing use of distributed systems, computer communication use is changing. In the past, communication had a distinct stream orientation. File transfer, remote terminal access, and electronic mail were the dominant network activities, and the main concerns were the throughput for streaming data and low cost for idle connections (for remote terminal access). A distributed systems environment, on the other hand, imposes new demands and requirements on the transport service. In particular, the emphasis is on providing fast response for request-response communication, such as page-level file access and remote procedure call (RPC) [1].

There are also significant changes taking place in the communication substrate. Past networks were based on telephone technology and low performance computer switching nodes. Communication bandwidth, node buffer space and node processing power have been the critical resources. Today, local area networks with multi-megabit data rates, low delay, and low error rates are common. In addition, the future promises high-speed, low error rate, wide-area fiber optic channels [12].

This work was sponsored in part by the Defense Advanced Research Projects Agency under contract N00039-83-K-0431 and by Digital Equipment Corporation, the National Science Foundation, AT&T Information Systems, Bell-Northern Research, and IBM.

plus high-performance switching nodes and gateways that take advantage of the low cost of memory, processors, and multiprocessor technology.

These changes in use and substrate argue for a new generation of communication systems [4]. The design of protocols, networks and network interfaces should be rethought from first principles in the context of the new environment. Refining existing designs and tuning conventional wisdom in the area is inadequate, given the magnitude of changes that have taken place. At the same time, one needs to be extremely careful to identify the key problems of the current standard protocols.

We see two major categories of deficiencies in current transport protocols: performance and functionality. First, current transport protocols provide poor response for transaction-oriented communication. Their stream-orientation requires explicit connection setup and teardown, adding a significant overhead to simple request-response interactions. We regard data throughput as a less important problem with current transport protocols.

Current transport protocols also suffer from poor performance due to overruns. With high performance networks, most packet loss is due to packets being dropped by the host interface when packets are arriving faster than it can handle them. Window-based flow control techniques alone are inadequate, since they specify only how many packets to send, and not when (i.e., when) to send them. Thus, windowing techniques only work to solve the overrun problem when the window size is less than or equal to the burst size that the intermediate node or receiver can handle.

Finally, current transport protocols do not support multicast, datagram service, security, priority and various other communication services required by sophisticated distributed systems. In fact, they only support pair-wise, streamed reliable data interchange. Multicast is of growing importance in distributed systems [8] and is being developed for the Internet [10, 11]. A datagram facility with the same naming, transmission and reception facilities as the normal transport level is a powerful asset for real-time and parallel applications. We argue elsewhere[2] that VMTP provides the functionality and performance necessary for high-performance distributed systems.

The following section describes the basic VMTP design. Section 3 provides some performance measurements of the protocol in use in the V distributed system. This paper gives only a brief overview of the VMTP protocol. The interested reader and potential implementor should consult the full protocol specification, available as RFC 1045[6].
2 VMTP Design Overview

VMTP provides transport communication between network visible entities via message transactions. VMTP entities are identified by 64-bit identifiers which are unique, stable, and independent of host-address. VMTP assumes an underlying network (or internetwork) service providing datagrams.

A message transaction consists of a Request message sent by a client entity to one or more server processes followed by zero or more Response messages sent back to the client by the server processes. In the common case, a client sends a request to a single server and receives a single response. Multicast is realized by sending to a group of servers. Datagram support is provided by indicating in the request that no response is expected. In addition, VMTP offers a streaming mode in which an entity can issue a stream of requests, receiving the responses back asynchronously. Using this facility, VMTP can achieve the same streaming performance over long delay lines as other protocols specifically designed for streaming, such as TCP[8].

2.1 Packet Format

There are only two packet types in VMTP: Request and Response. A VMTP packet consists of a fixed size 64-byte header and a data segment of zero or more bytes. A single bit in the packet header indicates whether it is a Request or Response packet. The single, fixed packet format simplifies processing: there are no options to parse and interpret, there are no variable length fields to deal with, and there is a clear separation of data and control.

The header, shown in Figure 1, contains protocol control information, client and server entity identifiers, a transaction identifier, and a 32-byte (user level) message control block (MCB). The MCB allows small amounts of data (e.g., procedure parameters for RPC) to be exchanged very efficiently. Larger amounts of data are transferred using the data segment portion of the packet, with the length of the segment indicated in the packet header.

![Figure 1: VMTP Packet Format](image)

A VMTP packet also contains a trailing 32-bit checksum, facilitating calculation of the checksum during packet transmission. The conventional approach of placing the checksum in the header requires that the transmitting host access every word of the header and data to calculate checksum before transmission begins.

2.2 Request-Response Model

VMTP is designed to support efficient transaction-oriented or RPC-style communication, and thus is based on the request-response model of interaction. The request-response model uses implicit connection setup, and timer-based connection management. Thus, connections are set up on demand, rather than by an explicit action by the client as is required in TCP. The request-response approach offers three main advantages over a handshake-based approach: minimal packet exchanges for communication, lower delay for initial data delivery, and minimal cost for inactive connections.

To initiate a message transaction, the client increments its transaction identifier and transmits the request message. A response message is normally received fairly soon thereafter, acknowledging the request and providing the user-level response data. Responses are matched to requests based on the client entity and transaction identifier. If no response arrives, the client times out and retransmits the request message with a control bit set requesting an immediate acknowledgement. The client retransmits the request periodically, giving up in failure if no response or acknowledgement of the request is received within some number of retransmissions.

On receipt of a request, a server locates a transaction record corresponding to the requesting client and initializes it with the parameters of the request and its associated transaction, and delivers the request to the server process. If there is no transaction record for the client, the server host allocates and initializes a new record.

The server process handles the request and sends a response back to the client. The response is saved as part of the transaction record so that it is available for retransmission if so requested by the client. Prior to sending the response, the transaction record is used to recognize and filter out retransmissions of the request, sending back an acknowledgment if requested by the client. After sending the response, the transaction record is retained until there is no danger of receiving delayed duplicates, following the delta-T protocol of Watson [15]. At that point, the server is free to reuse the transaction record.

With an active client, a new request is typically received from the client before the transaction record is reclaimed. Each new request from a client to the server implicitly acknowledges the last response sent to this client by the server. For this type of client, the transaction record is reused by each new request from the client, operating with the efficiency of having a connection record in TCP. However, once the client becomes idle or shifts its activity to another server, the current server host is able to reclaim the transaction record without any communication with the client. In contrast, with TCP, an explicit action to close the connection is required.
2.3 Packetizing a Message: Packet Groups

A VMTP message transaction may contain up to 16 kilobytes of data. Such large data segments are transmitted as a packet group, a sequence of VMTP packets, each containing part of the total data segment. Each packet in the packet group contains the same client, server, and transaction identifier, and the same MCB.

To facilitate selective retransmission and buffering, a data segment in a packet group is viewed as a sequence of segment blocks, each 512 bytes in length (except for the last, which may be only partly full), allowing the portions of the segment in a packet group to be specified by a 32-bit mask. The packet delivery mask in the packet header is used to indicate the portion of the data segment contained in each packet. The maximum number of blocks per packet is determined by the network maximum packet size. (Two, in our case, on a 10Mb Ethernet.)

In the normal case, a message is sent as a single packet group. That is, its data segment size is less than 16 kilobytes. User level messages that are larger than 16 kilobytes are divided into a succession of packet groups. In this case, the offset of a packet group within the message is indicated by its transaction identifier relative to the transaction identifier of the first packet group in the message.

The minimal VMTP message transaction contains no data segment: it uses only the minimal 64-byte header for both Request and Response.

2.4 Selective Retransmission

VMTP provides selective retransmission and rate-based flow control to minimize packet loss due to overruns and to provide resilience to lossy communication channels. The data segment in each packet group is represented by a 32-bit delivery mask. The receiver of a packet contains a cumulative delivery mask of blocks received for a given transaction. On timeout with an incomplete packet group, an acknowledgement is sent indicating the blocks received, and triggering retransmission of the missing blocks. The receiver can request any portion of the packet group to be retransmitted. For example, if every other packet is dropped, the receiver can request retransmission of exactly those portions, rather than all packets, or all packets starting from the first dropped packet.

2.5 Rate-based Flow Control

VMTP also allows rate-based flow control, spacing out packets in a packet group with inter-packet gaps to reduce the arrival rate at the receiver, if necessary. Clients and servers may explicitly communicate their desired inter-packet gap times, and may make adjustments based on selective retransmission requests. For example, if the receiver requests retransmission of every 4th packet, the sender can reasonably increase the inter-packet gap. Moreover, the sender can periodically attempt to reduce the interpacket gap when no packet loss is occurring to ensure it is transmitting at the maximum rate that the receiver can handle. Thus, selective retransmission provides feedback to indicate that the rate of transmission is too high and also minimizes the performance penalty arising from the resulting overrun.

2.6 Additional Features

VMTP provides a rich collection of optional facilities that expand its functionality and efficiency in various situations. These facilities include nested calls, conditional message delivery, co-resident addressing, and streaming.

These optional facilities are carefully designed to provide critical extensions to the basic facilities without imposing a significant performance penalty, especially on common-case processing. The refinement of the VMTP design in concert with its implementation has ensured that these optional facilities do in fact impose a minimal overhead.

VMTP was also designed to keep the complexity of managing protocol timers to a minimum. The protocol requires only one timer per client with an outstanding transaction, regardless of the number of packets in that transaction. Upon expiration of the timer, the client either retransmits the request (only the last packet in the request if the request is a multi-packet group), or aborts the transaction with a failure indication. With streaming, the timer is maintained only for the last outstanding transaction. Similarly, on the server host, there is only one timer per active (or recently active) client. A timeout at the server results in either a retransmission of the response, or the discarding of the client transaction record.

3 Performance Evaluation

There are two implementations of VMTP at this time: one in the Unix kernel, and one in the V Distributed System [5], an experimental distributed operating system developed at Stanford. There are more applications and services using VMTP as part of the V system, so we use this system as the basis for our performance evaluation of the protocol.

The V system is a message-based distributed operating system designed for use on workstations connected by local networks. V provides transparent network interprocess communication, network file access, multiprocess programs, remote execution, and process migration, as well as a multicast facility useful for distributed scheduling, decentralized naming, and other distributed applications. The system is based on the V kernel [3], a communication-oriented kernel that supports an efficient request-response form of message passing tuned for remote procedure call behavior. All network communication in V uses VMTP. More extensive measurements of our experience with VMTP in this environment are presented in an earlier paper [7].

3.1 Method of Measurement

The performance measurements presented in the following sections were made using the program timeicep to measure basic elapsed times for network interprocess communication, similar to measurements that have been previously published [3]. Timeicep creates a user-level client process on one machine, and a user-level server process on another, and exchanges messages between them as fast as possible. The user can specify the size of the data segment for the message transaction. The mean message transaction time is computed from the number of messages exchanged (typically 1000 to 10000), and the elapsed time for the test. Each test is repeated several times, and the results averaged to minimize the effect of random variations.
3.2 Response Time and Throughput

Table 1 lists the elapsed times for the V Send operation with various amounts of segment data returned, and the corresponding data rate that this supports. These times were measured with the timeipc program running on two SUN 3/75's connected by an idle 10 Mbit Ethernet. Thus the elapsed time in the table is the time between the client's sending of the initial request and receiving of the final response, as measured at the application level on the client machine.

The first row gives the time for a basic Send-Receive-Reply message transaction exchanging only the 32 bytes of user data in the VMTP header. The remaining rows reflect the effect of increasing the data segment size in a message transaction, both in increased elapsed time and increased effective data transfer rate.

The basic elapsed time for a V message transaction of 2.23 milliseconds suggests that VMTP and the V kernel implementation provide a fairly minimal delay for a Request-Response interaction.

The doubling in data rate with increasing data segment size from 1 kilobyte to 16 kilobytes indicates the benefit of the packet group mechanism. Using the full packet group rather than 1K at a time reduces the total number of packets sent, eliminates the end-to-end delay for the response or acknowledgement for each packet, and allows for an overlapping of processing at the sender and the receiver.

This suggests that strict per-packet acknowledging of packets, as used in some RPC systems, may be too inefficient, even in a local area network environment. In an environment with much greater end-to-end delay, the multi-packet group would dramatically outperform a per-packet acknowledgement scheme. The relatively modest increase in data rate between 12 and 16 kilobytes indicates that supporting larger data segment sizes in VMTP would not lead to significantly higher data rates, assuming low propagation delays. Note that the 16 kilobyte unit for VMTP was chosen in part to support the 8-16 kilobyte data sizes recommended by Lazowska et al. [14] for network file access.

3.3 Selective Retransmission

To evaluate the importance and effectiveness of selective retransmission, timeipc measurements were made on VMTP implementations with and without the selective retransmission feature, under various degrees of packet loss. (A kernel configured to randomly drop network packets is part of our robustness test suite for the VMTP protocol. The packet loss factor is settable by the user.) The results are plotted in Figure 2.

Table 1: V IPC Elapsed Time (SUN-3/75 to SUN-3/75)

<table>
<thead>
<tr>
<th>Operation (data in Kbytes)</th>
<th>Time (milliseconds)</th>
<th>Data Rate (Mbits/sec.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2.23</td>
<td>-</td>
</tr>
<tr>
<td>1</td>
<td>3.75</td>
<td>2.18</td>
</tr>
<tr>
<td>4</td>
<td>10.5</td>
<td>3.12</td>
</tr>
<tr>
<td>8</td>
<td>16.8</td>
<td>3.91</td>
</tr>
<tr>
<td>12</td>
<td>23.2</td>
<td>4.24</td>
</tr>
<tr>
<td>16</td>
<td>29.3</td>
<td>4.47</td>
</tr>
</tbody>
</table>

With selective retransmission, the performance degrades gradually as the level of random packet loss increases. Performance does however still improve with the size of the packet group. Without selective retransmission, performance drops significantly as soon as 1 packet in a packet group is lost. Large packet groups are no longer advantageous when full retransmission is required on packet loss.

3.4 Rate-based Flow Control

Overrun arises because of the significant speed differences between machines and network interfaces of different "generations". Every indication is that this generation gap problem will continue in the future, with higher speed networks making good interfaces expensive. We illustrate overrun in our environment using the SUN-3 to the SUN-2 communication. The SUN-3 is approximately two to three times faster. In an internetwork environment, overrun could also occur at a gateway if buffering is inadequate. For instance, the speeds of the incoming and outgoing links may differ dramatically, or multiple conversations may be competing for the limited buffers available.

To measure the presence and cost of overrun in our environment, timeipc was used to transfer data from a SUN-3/75 to a SUN-2 with a 3Com network interface. This particular network interface has only two receive buffers, resulting in significant packet loss if more than two packets arrive back to back from the network. Figure 3 depicts the cost of overrun as the size of a packet group increases. The cost of overrun is expressed as the percentage of extra packets (selectively) retransmitted to successfully complete the transfer.

Without rate-based flow control, the SUN-2 always misses the third and sixth packets in a packet group. The overrun effect is significant enough that the effective data rate actually decreases when going from 1K packet groups (1.30 Mbits/sec) to 4K packet groups (1.07 Mbits/sec). The throughput achievable with 16K packet groups is 1.68 Mbits/sec.
With rate control, a delay or interpacket gap is inserted between the consecutive transmissions of each packet in the packet group. This effectively spaces out the packets so they are less likely to be dropped at the interface to the slower machine. Figure 3 shows the effect on overrun cost of increasing the size of the inter-packet gap. As the inter-packet gap size approaches a full packet time (approximately 900 microseconds), the overrun problem disappears, and the number of packet retransmissions required approaches zero.

![Figure 3: Cost of Overrun With and Without Rate Control](image)

Implementing interpacket gaps in software requires a clock with fine granularity. In practice, gaps of 1 millisecond or less are often needed. Our clock granularity (10 ms) is not sufficient for this purpose. Thus, for this experiment, the interpacket gaps are inserted by the Ethernet driver using a simple spin loop. The number of iterations in the loop is specified as a parameter to the packet sending routine. The gap time (in microseconds (usec)) is estimated from the number of iterations of the loop and the MIPS rating of the SUN-3. Also, static rather than dynamically adjusted gaps are used.

With rate control, the maximum throughput achievable from SUN-3 to SUN-2 increases from 1.68 Mbits/sec to 2.73 Mbits/sec (for 16K packet groups). For comparison purposes, the SUN-2 to SUN-2 transfer rate (without flow control) is 2.20 Mbits/sec.

Our measurements of 16 kilobyte transfers from a SUN 3 to a SUN 2 indicate that, without rate control, overrun occurs. Rate control significantly improves the performance when speed mismatches arise, as is common in most network environments. When packet loss occurs, the selective retransmission facility minimizes the cost of recovery. Under repeated packet loss, selective retransmission is essential to make the transfer possible at all.

4 Conclusions

Changes in communication use and the underlying communication technology present the opportunity and the necessity of developing a new generation of protocols, including a transport protocol. Our focus on fast response rather than throughput has led us to design VMTP around the request-response paradigm, rather than the more conventional streaming model of standard protocols. It has also caused the design to incorporate a number of extended features in order to cover a wide range of real-time communication requirements.

The measurements presented here provide an indication of the performance characteristics of VMTP and the high-performance networks that make use of request-response protocols. VMTP provides comparable throughput to existing transport protocols, as well as lower latency and additional functionality.

The measurement of elapsed times for packet groups indicates that this facility provides double the data rate compared to single packet request-response for large file reads. The selective retransmission and rate-control mechanisms in VMTP reduce the packet loss due to overruns when transmitting these multi-packet blasts.

References

Distributed Process Groups in the V Kernel.

[9] DARPA.

Host Extensions for IP Multicasting.

Multicast Routing in Internetworks and Extended LANs.
In SIGCOMM '88. ACM SIGCOMM, August 1988.


The VMP Network Adapter Board (NAB):
High-Performance Network Communication for Multiprocessors.

File Access Performance of Diskless Workstations.

Timer-based Mechanisms in Reliable Transport Protocol Connection Management.