```
*       closes the socket on receiving the entire file.
*/
```

Appendix B

Programming Interface for the SCE

We now describe the programming interface to the SCE in the context of SunOS. We preserve the socket interface offered by BSD-based SunOS and allow the application to use the same TCP socket to communicate with the SCE.

The application has to supply the CPDS ("Specify Parameters" primitive) to the SCE by a "setsockopt()" system call before attempting to open the connection with a "connect()" system call. An error is returned by "connect()" if this sequence is not followed. The application can update the CPDS ("Update Parameters" primitive) during the course of the connection by another "setsockopt()". It can get the uptodate CPDS from the SCE ("Status Query" primitive) by executing a "getsockopt()" system call. Both the update and query can be done during the course of the connection only, and not after it is closed by the application. We have not implemented the "Inform" primitive in this version.

A sample application program which transfers a file from a source to multiple destinations is shown below. A more non-trivial relative of this application, which can use the SCE, is a "multicast put" operation in the File Transfer Protocol, as mentioned earlier.

Source application:

```
/*    This is very similar to a file transfer using a unicast TCP connection. So we are presenting the
 *    multicast specific part of it only. Other parts are described tersely.
 */
/*    Open a TCP socket (an Internet stream socket).
 */
/*    Set the Time-To-Live value for multicast packets sent out on this socket. (this operation needed for
 *    IP multicast, which otherwise sets ttl to a default value of 1).
 */
    ttl = 8;
    if (setsockopt(sockfd, IPPROTO_IP, IP_MULTICAST_TTL, &ttl, sizeof(ttl)) == -1)  {
        printf("client: error in setsockopt %d\n", errno);
        exit (1);
    }
/*    The CPDS we use is as follows:
 *          struct cpds {
 *                  u_short MnDGC;
 *                  u_short MnDOP;
 *                  } ds;
 *    This can modified and used as needed. A full scale version would include all the parameters
 *    described in Appendix A.
 */
    ds.MnDGC = 2;
    ds.MnDOP = 2;
    if (setsockopt(sockfd, IPPROTO_SCE, SCE_OPTS, &ds, sizeof(ds)) == -1)  {
        printf("client: error in setsockopt %d \n", errno);
        exit (1);
    }
/*    The application then executes "connect()" to the multicast address, on success, starts sending data as
 *    in the unicast case. "connect()" returns an error (with errno set to EINVAL) if the right sequence
 *    of operations is not followed.
 */
```

Destination application:

```
/*    The destination application opens a TCP socket, makes itself a member of the multicast group and
 *    listens on the socket. On successful connection establishment, it receives the multicast data, and
```

12)	MnDOP	Minimum percentage of group members desired to be receiving data correctly for the duration of connection.
13)	MxDOP	Maximum percentage of group members desired to be receiving data correctly for the duration of connection.
14)	Timeout	Delay value specified by application before SCE can timeout and possibly generate control packet for delivery to TCP.
15)	NEGOTIATE	Flag indicating desire of application to be updated about connection status by SCE.
16)	Group Membership	Count or list of hosts currently in group.
17)	SCE State	State of connection.

Appendix A

We describe here the interface primitives to the multicast-specific services provided by the SCE. These are modeled along the lines of those provided to TCP services [21]. The existing application/system interface (sockets in case of BSD) can be used by the application to communicate with the SCE.

SCE service request primitives

	Primitive	Parameters
1)	Specify	source port, destination port, destination address
	Parameters	[, control parameter data structure]
	Purpose:	Specify control parameters for multicast connection to specified destination address and port.
2)	Update	local connection name, control parameter data
	Parameters	structure
	Purpose:	Update parameters for specified connection.
3)	Status	local connection name
	Query	
	Purpose:	Query connection status.

SCE service response primitives

	Primitive	Parameters
1)	Inform	local connection name, status
	Purpose:	Reports connection status.
2)	Update	local connection name, response
	Response	
	Purpose:	Reports update request response.
3)	Status	local connection name, status
	Response	
	Purpose:	Reports connection status to application on request.
4)	Error	local connection name, description
	Purpose:	Reports service request or internal error.

SCE service parameters

1)	Source Port	Identifier for local TCP/SCE multicast application.
2)	Destination Port	Identifier for remote TCP/SCE multicast application.
3)	Destination Address	Multicast IP address for group.
4)	Control Parameter Data Structure (CPDS)	Data Structure containing control parameters like Minimum Desired Group Cardinality (MnDGC), Maximum Desired Group Cardinality (MxDGC), Desired Group Members (DGM), Minimum Degree Of Participation (MnDOP), Maximum Degree of Participation (MxDOP), Timeout, NEGOTIATE flag.
5)	Local Connection Name	Identifier for multicast connection, alloted by TCP.
6)	Response	Indicates success or failure of “Update Parameters” request.
7)	Status	Data structure indicating multicast-specific status of connection. Contains local connection name, CPDS, group membership, SCE state.
8)	Description	Supplementary information in an error message.
9)	MnDGC	Minimum number of hosts desired to be in group at connection setup.
10)	MxDGC	Maximum number of hosts desired to be in group at connection setup.
11)	DGM	Specific hosts desired to be in group.

cast TCP approach suffers due to the overhead of having to manage multiple connections, thus causing its connection establishment time to increase for increasing group sizes.

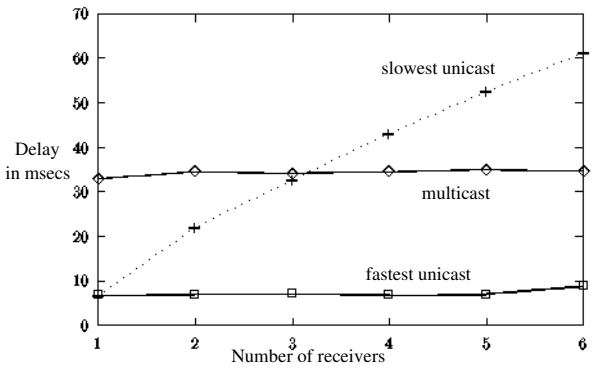

Figure 5: Connection establishment time vs. Number of receivers

The SCE approach thus offers connection throughput comparable to the unicast approach, and better connection establishment delays for larger groups. It also offers the benefits of using multicast instead of unicast, namely not needing the individual group member identity to be known prior to connection setup, and avoiding duplicate transmission of data for achieving multicast.

7 Concluding Remarks

We presented a new architecture for implementing reliable multicasting in existing network architectures. This approach has the advantage of conserving the semantics of the existing protocol layers. It permits applications to supply a reliability definition for the multicast connection to the SCE sublayer which then regulates the status of the connection using this definition. The semantics of the multicast connection can also be easily changed by just altering the SCE sublayer, without affecting the other layers. Finally, we discuss an implementation of this architecture for the TCP/IP protocol suite and present some performance results using this implementation

We are now exploring the use of this protocol to provide for enhanced application functionality and performance. Among these are the provision of scalable World-Wide-Web service using multicast delivery [13] and the provision of a "multicast put" function within FTP to allow easy file replication.

As discussed previously, the RMTS provided by the SCE architecture derives many of its features from the transport protocol being used. The SCE architecture that uses TCP can thus be criticized for two problems: 1) the lack of scalability due to the possibility for acknowledgement implosion [8] when the number of receivers is large; and 2) its inability to provide efficient transaction-oriented communication. These are however drawbacks stemming from TCP and not the SCE architecture.

We are currently exploring the provision of a scalable SCE-based RMTS through the use of a state-exchange transport protocol such as the one described in [14]. Also we will be considering how multicast transaction-oriented communication can be built around the use of T/TCP [15]. In the process of constucting these different types of multicast transport protocols, we aim to understand the full scope of the SCE architecture.

References

[1] S. Casner, "FAQ on the Multicast Backbone," file://venera.isi.edu/mbone/faq.txt, May 1993.

[2] S. Deering, "Host Extensions for IP Multicasting," RFC 1112, Aug. 1989.

[3] B. Rajagopalan, "Reliability and scaling issues in multicast communication", Proceedings of ACM SIGCOMM, pp. 188-198, 1992.

[4] C. Topolcic, Editor, "Experimental Internet Stream Protocol, Version 2 (ST-II)," RFC 1190, Oct. 1990.

[5] R. Aiello, E. Pagani and G. P. Rossi, "Design of a Reliable Multicast Protocol," Proceedings of IEEE INFOCOM, pp. 75-81, 1993.

[6] S. Armstrong, A. Freier and K. Marzullo, "Multicast Transport Protocol," RFC 1301, Feb. 1992.

[7] R. Braudes and S. Zabele, "Requirements for Multicast Protocols," RFC 1458, May 1993.

[8] J. Crowcroft and K. Paliwoda, "A Multicast Transport Protocol," Proceedings of ACM SIGCOMM, pp. 247-256, 1988.

[9] "XTP Protocol Definition Revision 3.6," Protocol Engines Incorporated, PEI 92-10, Mountain View, CA, 11 Jan. 1992.

[10] T.A. Joseph and K.P. Birman, "Reliable Broadcast Protocols," in *Distributed Systems,* (S. Mullender, Ed.), ACM Press, 1989.

[11] A. Tanenbaum, *Computer Networks*, Prentice Hall, 1988.

[12] R. Talpade and M.H. Ammar, "Single Connection Emulation: An architecture for reliable multicast transport service," Technical Report, ftp://ftp.cc.gatech.edu/pub/coc/tech_reports/1994/GIT-CC-94-47.ps.Z.

[13] R. Clark, and M.H. Ammar, "Providing Scalable Web Services Using Multicast," Proc of 2nd Intl. Workshop on Services in Distributed and Networked Environments, June 1995.

[14] A.N. Netravalli, W.D. Roome and K. Sabnani, "Design and implementation of a high-speed transport protocol," IEEE Trans Commun, v. 38 n. 11, Nov 1990, p 2010-2024.

[15] R. Braden, "T/TCP - TCP Extensions for Transactions Functional Specification," RFC 1644, July 1994.

track of connection related information by using the TCP header in the data packet. It gives a single copy of the data packet to the IP for multicasting.

The SCE schedules a timer event for each data packet multicast and updates its database on the basis of control packets received before timeout. Errored acknowledgments are discarded quietly by the SCE. All other control packets except acknowledgments and connection close packets are discarded by the SCE in this phase. On timeout, if the number of group members that acknowledged previously unacknowledged data is at least equal to the DOP desired, the SCE generates an acknowledgment and returns it to the TCP (this set of members is called the current set). It otherwise does nothing, so the TCP times out and takes necessary action, which usually involves retransmitting the data packet.

If the DOP requirement is satisfied, the SCE uses appropriate values in the fields of the control packet that it passes to the transport layer. These values are such that the transport layer is assured that the connection is working correctly. The relevant fields are the acknowledgment field, the sequence number of the control packet and the window size that the receiver supposedly advertises. The acknowledgement field is set to the least acknowledgement value returned by a member from the current set. It is possible for a group member to acknowledge only part of the multicast data as TCP sequence numbers bytes and not packets. The window size is set equal to the smallest window size of all received packets. This practice is maintained throughout the connection. The sequence number of the control packet is set to any appropriate value, and is incremented whenever needed.

Connection close phase

On sending out the connection close packet on the connection, the SCE does not forward a connection close to the TCP until the number of group members acknowledging all mulitcast data satisfy the DOP requirement. Only when this condition is satisfied does the SCE forward a corresponding connection close to the TCP, causing it to generate a last acknowledgement which is multicast to close the connection cleanly at both ends.

6 Experimental Results

We used our implementation of the SCE with TCP/IP to transfer data reliably to a group of receivers. Our experimental setup consisted of a multicast source running on a Sun Sparc20 and six potential receivers: two Sparc1's, two Sparc20's and two SparcLX's. The Sparc20's and SparcLX's were on the same subnet as the source, while the Sparc1's were on a different subnet, connected to the source subnet through a multicast-capable router.

In the first experiment we used our protocol to transfer a file from the source to four receivers (the SparcLX's were not used in this experiment). For each receiver we measured the time from when it received the source's connection request message until it received the source's connection close message. This is a measure of the time a particular receiver is occupied with this connection. We compared this time with how long each receiver would be occupied had the source transfered the files using four unicast TCP connections.

Figure 4 shows the receiver-occupied time as a function of the total data transfered to each receiver. This is shown for unicast and multicast approaches, and for the slowest and fastest receivers. This figure shows that the throughput experienced by receivers is the same regardless of whether multicast or unicast is used. Of course the multicast scheme has the advantage of consuming less network bandwidth.

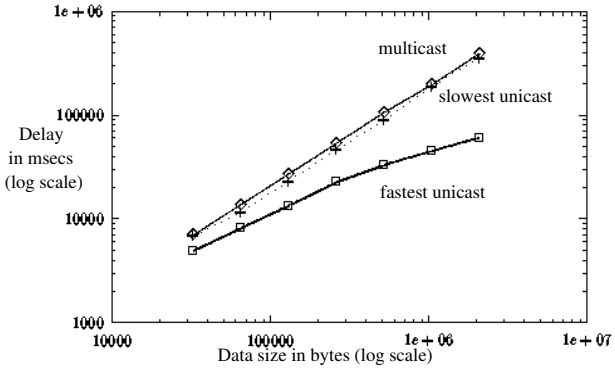

Figure 4: Delay for 4 receivers vs. Data size

In the second experiment we used our protocol to connect to a varying number of receivers. Our protocol was used by an application which forked a process that initiated the multicast connection. Our benchmark was an application that used unicast TCP, and therefore, had to fork multiple processes, each initiating a unicast TCP connection to a particular receiver.

Figure 5 plots the measured time as a function of the number of receivers. In the multicast case we measure the time from when the application forks a child process to setup a multicast connection, to when the child process completes the connection open and returns from the "connect" system call. In the unicast case, we measured the time from when the application forks the first child process, to 1) the time when the last child process returns from the "connect" system call; and 2) the time when the first child process returns from the "connect" (note that the first process to return may not be the first one that was forked). The plots indicate that the connection establishment time for the SCE approach remains approximately the same regardless of the number of receivers. The uni-

greater than the highest sequence number sent by that destination TCP. This does not affect the destination TCP as no data is ever sent by it on that connection back to the source.

Our implementation supports two multicast control parameters:

- Timeout: The amount of time that the SCE should wait during connection establishment phase to collect connection accepts from the group members.
- Degree of Participation(DOP):as described in section 3.

In the remainder of this section we give a brief discussion of the SCE design and operation in the TCP/IP context.

5.1 SCE Interactions

The SCE exists as a sublayer between TCP and IP. As such, it has to interact between these two protocols. It goes through three phases similar to the TCP; connection establishment, data transfer and connection close. We take a more in-depth look at the interactions between the SCE and the TCP and IP during these phases, and also the possible situations that the SCE has to contend with. It is important to note that the SCE uses the checksum in the TCP header to detect errors in received control packets, and discards any packets with errors quietly.

SCE state diagram

Figure 3 presents an overview of the SCE state diagram. The SCE always starts a new connection in the SCE_CLOSED state. It makes a transition to the SCE_SYNSENT state on forwarding a SYN packet received from the TCP. It then changes to the SCE_ESTBLD state if the connection is set up (depending on the DGC requirement being satisfied), otherwise returns to the SCE_CLOSED state. Data transfer, and most other interactions, take place in the SCE_ESTBLD state. The SCE finally moves to the SCE_FWT1 state on forwarding a FIN packet received from the TCP, and then returns to SCE_CLOSED after forwarding an acknowledgment for the FIN to the TCP

Connection establishment phase

This is the first phase that the SCE has to go through for any connection. The SCE receives a connection request packet from the TCP. A connection request packet is identified to require multicast communication if its destination address is a class D (multicast) IP address. It initializes its database and other variables using the information contained in the TCP header. This connection request packet is then handed over to the IP for multicasting to the group identified by the class D IP address supplied by the TCP. The application supplies the multicast control parameters to the SCE at the beggining of this stage.

The SCE receives the connection accept packets from the IP, updates its database and increments the count of the number of accepts received. The amount of time that the SCE waits for these accepts is set initially to an application-specified value (Timeout). The SCE then compares the number of accepts received against the DGC, returning an accept to TCP if the number of connection accepts is greater and returns nothing otherwise. Only connection accept packets are accepted by the SCE at this stage, all other packets are quietly dropped. No further connection accepts are accepted by the SCE once it has finished waiting for them.

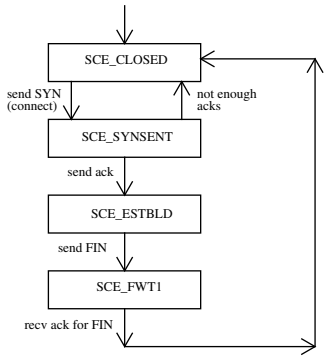

Figure 3: SCE state diagram

As the last part of connection establishment, the SCE gets the acknowledgment for the connection accepts from the TCP. The sender and receiver in a TCP connection exchange their initial sequence numbers during connection establishment phase. So each of the receiver needs to receive an acknowledgment for its specific connection accept. This is done by having the SCE unicast the acknowledgment to each of the receivers individually, after having set its value to reflect the sequence number of the connection accept received from the particular receiver. This completes the three-way handshake connection establishment procedure at both the source and the destinations. All succeeding multicast packets have their acknowledgment fields set to a value greater than the initial sequence numbers of all received connection accepts. All nodes that do not receive a connection request or connection acknowledgement, or whose connection accept is not received by the source, are not deemed to be part of the connection multicast group. It is possible that such nodes will receive packets belonging to the multicast connection some time later, these packets will be dropped and any responses to them will be ignored by the SCE at the multicast source.

Data Transfer phase

The SCE gets data packets from the TCP and keeps

nation addresses, port numbers, starting sequence numbers, current window sizes and variables, connection state and pointers to corresponding transport and network layer connection databases. The SCE updates this database on the basis of information it gleans from outgoing multicast packets it gets from the transport layer, and the decisions it makes on processing incoming control packets from the group members.

Acknowledgment processing

In a typical sequence of events, the SCE receives a packet (e.g., a data packet or a connection request) from the transport layer. This packet is then forwarded onto the network layer which would then multicast it to a destination group. Members of the destination group may acknowledge receipt of the packets. These acknowledgments are intercepted by the SCE as they arrive at the (multicast) source.

Once the SCE collects the "appropriate" number of acknowledgments, it forwards a single acknowledgment to the transport layer. Whether an appropriate number is collected is determined by criteria supplied to the SCE by the application layer (e.g., the connection success criteria in section 3). Lacking the appropriate number of acknowledgments to a packet, the SCE does not forward any indication to the transport layer; we rely on the transport layer's own mechanisms to time-out and take appropriate action.

Timeout Threshold Estimation for SCE

When the SCE times-out after collecting acknowledgements sent (by multiple destinations) in response to a packet, it forwards a (self-generated) acknowledgement to the transport protocol (TP) only if the connection requirements (CRs) are satisfied. Otherwise the TP itself will timeout and retransmit on not getting an acknowledgement from the SCE. This mechanism will function correctly only if the timeout threshold (TT) estimate for the TP is greater that the SCE's. Otherwise, the TP may timeout (and retransmit) even though the SCE has not, and this might lead to multiple copies of the same packet being received by the destinations. This in turn will lead to the destinations sending duplicate acknowledgements, thus causing wastage of bandwidth. The difference in the TP and SCE TT estimates should be a function of the processing time needed by the SCE to generate and forward an ack to the TP.

There are two basic approaches for the SCE to obtain the timeout threshold estimate for a packet:
1) Implement its own version of TT estimation mechanism. This introduces more complexity into the SCE, however makes its TT estimate independent of the transport layer's. A certain degree of duplication of functionality and implementation in the TP and SCE is also introduced.
2) Obtain the current value for timeout threshold from the overlying TP and use it after any needed processing. The transport protocol TT estimate should be easily accessible for this approach to be feasible. The "processing" of the TP timeout threshold estimate will include making sure that it is appropriately decremented before being used by the SCE.

We adopt the second approach because it simplifies the implementation of the SCE and achieves basically the same effect as the first approach. This works well in situations where the transport protocol has its own adaptive timeout threshold estimation.

Group management

The SCE has to maintain the connection group for the duration of the multicast connection. This group is initially formed out of all the destinations that participate successfully in connection set up. The size and constituency of this group may change during the course of the multicast connection. A group member may be dropped if it consistently does not acknowledge received packets for prolonged periods. The SCE can determine the definition of the prolonged period, or alternatively, the definition can be supplied by the application prior to connection establishment. The SCE may choose to ignore a dropped member or initiate a connection close with it separately. A group member may also be re-admitted into the group if it starts responding correctly after some interval of time.

5 Experiences with using SCE for the TCP/IP protocol suite

We implemented the SCE for TCP as the transport layer and IP as the network layer. The TCP/IP code used is derived from the SunOS Release 4.1.3C implementation.

The code at the source end consists of an SCE implementation with no modifications necessary to the TCP or IP code. Rather than introduce an SCE layer in the destinations, we found that three simple patches to the TCP code were sufficient there. One patch ensured that all multicast connection related TCP packets emerging from a destination had that destination's individual IP address (as opposed to multicast group address) in the source field of the packets. The second was needed to ensure that TCP does not throw away packets received with multicast destination addresses. A recent SunOS update (Release 4.1.3_U1) modified TCP so that it would filter out any received packets with class D (multicast) IP addresses. The last patch permitted the destination TCP to accept multicast packets (in the ESTABLISHED and consequent phases of the connection) with acknowledgement values

sider a RMTS that provides reliable flow of information in one direction, from the source to the destinations. Note that packets still flow in two directions as part of multicast transport connections. However, only control packets flow back from the destinations to the source. If the destinations want to send information back to the source, this can be done through a separate unicast transport layer connection.

We propose that the SCE be incorporated at both the source and destinations. However, because of the one-way information flow, we expect that the SCE functionality at the destinations will be minimal. In fact, in our implementation discussed in section 5, we were able to do without an SCE layer at the destinations by adding minor patches to the transport layer protocol at the destinations.

Reliability and Group Dynamics

In unreliable (datagram) multicasting, it is possible to have destinations join and leave a multicast group by simply adding or removing the multicast group address from the list of addresses they recognize. Similar to traditional TCP, we make the assumption that individual packets do not represent meaningful application data units by themselves. Rather, the sequence of packets received from conection establishment until connection close need to be concatenated to form an application data unit. It is, therefore, not meaningful for destinations to join existing connections. We adopt the following model of group dynamics which only allows destinations to drop out but not join a connection in progress:

- The initial membership of a group is "discovered" at connection establishment time. All destinations that respond are made part of the group and a list of these members is maintained at the source (in the SCE). Initial membership can be controlled by the application through the control parameters passed to the SCE. The source may be a member of the group.
- Destinations are allowed to drop out of an existing group. No receivers are allowed to join after the connection is established.
- The SCE can detect that a destination has dropped out when it stops receiving acknowledgments for packets it is transmitting. Note that not receiving acknowledgments is not a positive indication that the receiver has dropped out. It could be just an indication of congestion or other temporary network condition. The SCE will, therefore, have to observe the lack of returned acknowledgments for a prolonged period of time before it declares that a receiver has dropped out.
- The application would specify to the SCE a *degree of participation* percentage. This indicates to the SCE that the connection should continue as long as the indicated percentage of destinations is still in the group. (The degree of participation could be 100%, in which case if any receiver drops out, the connection is aborted.)

4 SCE Functions

We now examine the functionality that needs to be built into the SCE in order to be able to support a reliable multicast service. This functionality is restricted to handling multicast traffic only, regular unicast traffic is allowed to pass through to the next layer after ascertaining its nature. Thus, introducing reliable multicasting capability into the architecture causes minimal overhead (of a conditional check) for regular unicast traffic.

SCE interfaces

The SCE maintains three interfaces: one each to the application layer, the transport layer and the network layer. The application/SCE interface is used to convey multicast control parameters as described in the previous section.

The SCE's interface to both the transport and network layers is such that the existence of the SCE is transparent to both these layers. The transport and network layers in the original network architecture communicate with each other usually by means of system calls. In order to support reliable multicasting using the SCE however, these layers now communicate through the appropriate SCE routine. In turn, the SCE invokes the appropriate network layer output routine for outbound traffic, and the appropriate transport layer input routine for inbound traffic. The syntax of the transport and network layer calls is not changed. Thus these protocol routines can still be invoked with the same syntax as before by other protocol routines.

Multicast connection information maintained by the SCE

The SCE determines the status of the connection on the basis of the number of group members currently participating in it. It keeps track of each group member by maintaining certain member-specific information. This database is initialized with all the specific group members that successfully participate in connection set up. All member-specific connection related variables (e.g., starting sequence number, window size, maximum segment size, value of last acknowledgment sent by member) are stored for each group member with whom a connection was initially established. The SCE then updates the database for each individual member on the basis of responses it receives.

In addition to the group member-specific information, the SCE also needs to maintain connection-specific information. This is similar to the information maintained by the transport layer for a connection. Typical variables stored include but are not limited to the source and desti-

maintaning synchronization between group members. RAMP/MGA consists of three components: RAMP, MGA and modified routing algorithms. It also has the notion of a "root" node which controls address management, service registration and group membership maintanance. It involves path setup and tear down and uses a NAK-based selective retransmission scheme and rate based flow control. AMTP is one of the earliest multicast transport protocols and needs to be used over a network layer supporting unreliable multicasting, like IP. XTP combines the transport and the network layers and provides the functionality of both in one layer. It offers two levels of multicast: unreliable and semi-reliable.

Our proposed architecture is distinguished from the approaches above mainly by not replacing but rather enhancing the transport layer. This has many benefits as described earlier. An important aspect of our approach is that it derives many of its features (good and bad) from the existing transport protocol.

It is also important to relate our work to work on reliable multicast protocols done within the distributed computing community [10]. The semantics of our reliable communication are derived from those usually associated with reliable transport layer protocols [11]. Additional semantics such as all-or-nothing atomic multicast or ordering of multicast messages from multiple sources to a common receiver can be built on top of our RMTS interface. For example, the atomic multicast transport protocol described in [10] calls for an initiator to send a message to all sites in the destination group. Receiving sites, in turn, multicast newly received messages to all members of the destination group. Both the initiator of the multicast and the receiving sites can use our multicast protocol to efficiently transport their messages,

3 The Reliable Multicast Transport Service

In this section we discuss aspects of our envisioned reliable multicast transport service (RMTS) which we construct by introducing the SCE. It should be noted that by using an existing reliable transport protocol, many of the service characteristics are dictated by the connection-oriented transport service being provided. We, therefore, focus on the aspects of the RMTS that pertain to its multicast nature.

Service Interface Components

The RMTS is made up of two components: 1) the interface to the service of the (existing) transport protocol and 2) the interface to the multicast-specific services provided by the SCE layer. We assume that the former will remain unmodified and propose a new interface only in order to communicate with the SCE layer (see [12] for the specification of an interface in the TCP/IP BSD system).

Connection-Orientation

The reliable multicast protocol we construct is connection-oriented. Prior to any data transmission, a connection has to be established from the source of the multicast to the set of targeted destinations. At the conclusion of the conversation the connection has to also undergo a termination phase. In the single-destination scenario connection establishment is successful if the destination accepts the connection.

In the multicast scenario, we say that the connection establishment is successful if particular *success criteria* are met. The criteria for a particular connection attempt should be conveyed to the SCE through the SCE/application interface prior to a connection attempt being initiated through the SCE/transport protocol interface.

Examples of connection success criteria are:

- At least *n* destinations accept the connection.
- At least the destinations identified explicitly accept the connection.
- At least *n* destinations accept the connection within a specified time-out period.

Other criteria can also be specified to the SCE by the application. For example, the application may specify that it wants to connect to "at most *m* destinations".

Addressing

We assume that the network layer uses multicast group addressing to identify groups of destinations (e.g., class D addressing in IP). At the transport layer, a particular destination set is defined by the combination of a multicast network layer address and a transport layer address (e.g., TCP port number). A connection request is issued to this multicast address and the source has no way of knowing a-priori how many destinations will respond. The likelihood of the success of a connection, therefore, relies on the application possessing some knowledge (through out-of-band means) of what is a reasonable success criteria for a particular connection request.

As a connection is established and as data is transferred, the source will keep track of the status of the destinations through control packets it receives from them. We will, therefore, need to have the individual destinations identified in all the packets received by the source. (Although this may seem obvious at this point, it is potentially troublesome in the context of TCP/IP as discussed in section 5.)

One-Way Information Flow

In typical multicast conversations, information from a source flows to a set of destinations in a multicast group. As part of this conversation, the destinations may also want to send information back to the source. We only con-

using multicast group addresses. The transport layer uses the services of the network layer to provide two possible interfaces to the application layer, a datagram interface and a connection-oriented reliable interface. The connection-oriented reliable interface, however, does not provide multicast capability because of the limitations of the transport layer protocol. Applications desiring to perform reliable multicasting in this architecture are forced to build reliability features on top of the datagram transport protocol interface.

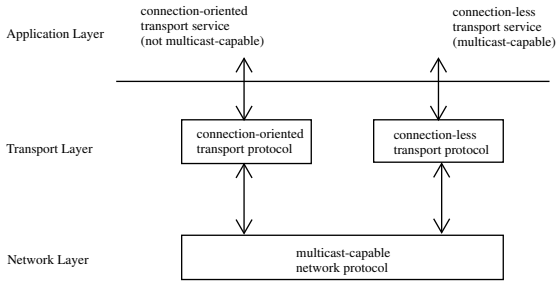

Figure 1: Existing network architecture for multicast

One observation about the architecture shown in figure 1 is that many of the features that one would program into a reliable multicast protocol are already present in the existing connection-oriented transport protocol. Our proposed architecture is based on this observation and is shown in figure 2. From the application layer viewpoint, the major difference is that our enhanced architecture provides a reliable multicast transport interface. This interface is in fact provided by keeping the existing (unicast) reliable transport protocol and introducing the single connection emulation layer (SCE) to adapt the transport protocol's expectation of single connection network layer interface to the multicast capabilities of the network layer. The SCE receives acknowledgements and other control packets from the multicast group members and aggregates them for forwarding to the overlying transport layer.

In our proposal we also allow for a direct application/ SCE interface. This interface is used to communicate parameters that can be used to control multicast-specific variables of the multicast connection. Note that this is required since the transport layer remains in its original form and is not multicast capable. An important set of control parameters are used to describe the reliability requirements of a connection. These are discussed in more detail in section 3.

Our proposed architecture has several advantages:

- By retaining the original transport layer protocol we are able to continue use of well-understood and mature networking technology. Also we are able to continue use of well-understood interfaces and therefore make for easier application development and testing.
- The architecture separates transport layer issues from multicasting issues in a modular fashion. Thus it is now easier to modify the semantics of multicast connections by changing the SCE without affecting the original transport layer protocol.

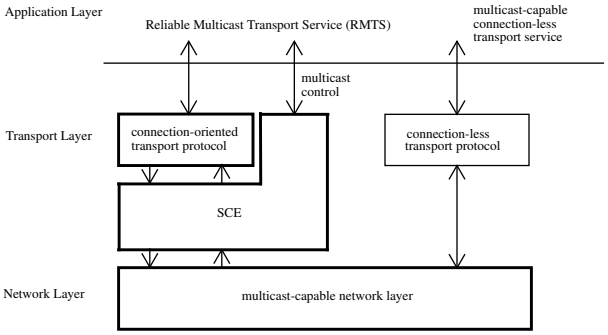

Figure 2: Proposed network architecture using SCE

2.1 Comparison with other approaches

One of the first successful forays into multicasting over internets involved using IP and was implemented at the network layer [2]. This has been used widely to implement the "MBONE" virtual network. The main drawback here is that it offers only unreliable service, and it is up to the application using it to make sure that the transmission is reliable, which means introducing transport layer functionality into the application. This is certainly not desirable. There are several ways to provide a reliable multicast transport service. These range from introducing reliable multicast at the network layer, to building new transport layer protocols which support reliable multicast, to combining the functionality of both the transport and network layers into a new layer.

Reliability at the network layer has been introduced in RM [3], ST-II [4] and URGC [5]. RM offers mechanisms for partial multicast and supports reverse communication from group members to source by a "gather" operation. It involves constructing and maintaining trees for each group. ST-II guarantees end-to-end delay and bandwidth, and involves reserving bandwidth for all connections at setup time. RM and ST-II are more on par with IP multicasting and are meant to replace it. URGC offers reliable communication between processes in the same group, and is suitable for supporting operations of a distributed operating system that manipulates replicated resources.

Multicasting at the transport layer has been suggested in MTP [6], RAMP/MGA [7], AMTP [8] and XTP [9]. MTP has the concept of a "master" node or process which controls many aspects of the multicast communication, including group membership and performance. It involves

Single Connection Emulation (SCE): An Architecture for Providing a Reliable Multicast Transport Service*

Rajesh Talpade and *Mostafa H. Ammar*

College of Computing, Georgia Institute of Technology, Atlanta, GA 30332
e-mail: {taddy, ammar}@cc.gatech.edu

Abstract

We present a novel architecture for providing a reliable multicast transport service over existing protocol stacks. These protocol stacks ordinarily support reliable unicast transport layer connections over a network layer which is capable of providing an unreliable multicasting service. We propose the addition of a new Single Connection Emulation (SCE) sublayer between the unicast transport layer and the multicast network layer. This added layer mimics the single destination network layer interface to the transport layer and interfaces with the multicast network layer to provide the necessary multicast functionality. The new architecture also enables interactions between applications and the SCE, thus allowing the applications to control the semantics of the reliable multicast connection. We discuss the design issues that need to be considered when such a sublayer is to be introduced. We also discuss an implementation of this new approach using the TCP/IP protocol stack.and present some preliminary experimental results.

1 Introduction

Multicasting has become one of the focal areas of research in networking over the past few years due to the growth of applications which require information to be distributed simultaneously to different destinations. While applications like audio/video multicast (e.g., MBONE [1]) can withstand a certain degree of loss of data, other applications require that the multicast data be received reliably by the multiple destinations. We consider the problem of providing a reliable multicast transport layer protocol in a network with an architecture similar to that of the Internet. Such an architecture includes a single destination, connection-oriented, reliable transport protocol along with a datagram-based network protocol which supports multicast addressing and dynamic group membership. An unreliable datagram multicast service is thus available from the network layer.

In this paper, we propose a new architecture for reliable multicasting by introducing a single connection emulation (SCE) sublayer between the single destination transport layer and the (multicast capable) network layer. This added layer mimics the single destination network layer interface to the transport layer and interfaces with the multicast network layer to provide the necessary multicast functionality. We have successfully implemented this architecture using the Transmission Control Protocol (TCP) as the transport layer and the Internet Protocol (IP) as the network layer. The objective of this paper is to explain the principles on which SCE is based, and present the design issues involved in introducing SCE as a sublayer in an existing network architecture.

We present an overview of the SCE architecture and compare it to existing reliable multicasting protocols in section 2. The reliable multicast transport service which we aim to construct by introducing the SCE is discussed in section 3. Section 4 provides an examination of the functionality that needs to be built into the SCE for it to be able to support a reliable multicast service. An implementation of this architecture using the TCP/IP protocol stack is presented in section 5. Section 6 presents some experimental results. The paper is concluded in section 7.

2 Overview of the SCE Architecture

Our proposed architecture is designed to be a reliable multicast upgrade (or enhancement) of existing protocol architectures. Although we feel that the proposed ideas are applicable in many situations, we focus on upgrading the architecture shown in figure 1. Three layers exist in such an architecture (which is similar to that in the Internet): an application layer, a transport layer and a network layer. The network layer is connection-less and does not provide any reliability guarantees. It is also capable of best-effort transmission and routing of multicast packets addressed

* Work supported by ARPA contract N00174-93-K-0105