Facilitating Robust Multicast Group Management

Avijit Sen Mazumder
Dept of Computer Science
UC-Santa Barbara
Santa Barbara, CA 93106
avijit@cs.ucsb.edu

Kevin Almeroth
Dept of Computer Science
UC-Santa Barbara
Santa Barbara, CA 93106
almeroth@cs.ucsb.edu

Kamil Sarac
Dept of Computer Science
Univ of Texas at Dallas
Richardson, TX 75083
ksarac@utdallas.edu

ABSTRACT

Multicast is a key service for many audio and video applications, yet it continues to be a challenging Internet service to deploy. While much attention has been given to a number of problems associated with multicast, two closely related problems in particular have received almost no attention. First, there is essentially no host support for dealing with the variety of ways to join a multicast group, and second, there is a complete absence of group join success or failure feedback from the network. The lack of a straightforward way to robustly join a multicast group and then to know whether the join has been successful is possibly the biggest limitation for multicast application developers today. In this study we develop a robust solution to determine the existence and nature of multicast service in the network. We consider two options: (1) the use of existing protocol features to extract the required information, and (2) the introduction of new protocol extensions to directly query the network. Our results indicate that while a truly robust group join is possible only when there is network support, these additional changes are difficult to deploy. Our evaluation implements a proof-of-concept prototype to determine the existence and nature of the multicast service in the network.

Categories and Subject Descriptors

C.2.2 [**Computer-Communication Networks**]: Network Protocols–Protocol Architecture

General Terms

Reliability, Performance, Management

Keywords

multicast, group management, debugging, protocol feedback

1. INTRODUCTION

IP multicast provides both scalable and bandwidth efficient mechanisms to support multi-receiver networking applications [1]. The commercial deployment of multicast, however, has been slower than expected [6]. Some of the reasons for this slowness include: difficulties with group management, challenges with efficient and practical multicast address allocation, security vulnerabilities, and a lack of multicast network management tools. In this paper we focus on addressing the issues of multicast group management and robust group join problems.

In the original IP multicast service model, referred to as Any Source Multicast (ASM), sources send their data to a multicast group defined by a Class D IP address. Receivers join the group to become members and receive multicast data sent to the group address. More recently, a simplified multicast service model called Source Specific Multicast (SSM) [2] has been introduced. In SSM, multicast groups are referred to as *channels* where a channel is identified by the IP address of the transmitting host *and* the multicast group address. In both cases, the underlying network builds multicast distribution trees connecting sources and receivers. The process by which a receiver is added to the multicast tree is called a *group-join*. A successful group-join results in proper data reception at the receiver site. If a group-join fails, however, the receiver cannot join the multicast tree and does not receive any data. The robustness of group-joins, therefore, determines the robustness of the overall multicast service in the Internet.

IP multicast is designed to provide a simple interface for applications to join and receive data from a multicast group. The join process is straightforward and requires a receiver to open a socket and set a socket option [9]. No acknowledgment is provided to confirm the success of a group-join. This simplicity makes it easy to join a group, but creates problems in the case of failures. If a join is sent but no content is received, the receiver cannot really be sure about the reason for the failure. It may be that the source is currently inactive or it may be that the join request failed. Within the current IP multicast service architecture, the receiver does not have a mechanism to learn the reason why multicast content was not received. In order to achieve a *robust group-join*, the user application *must* be able to determine whether a group join has succeeded or not.

The current *best effort* approach to joining a multicast group is fragile due to a number of problems that include protocol mismatch, version mismatch, and partial deployment. Other than successfully receiving content from a source, there is no way that an application can determine what might have happened to its join request. The effect of this ambiguity is twofold. First, an application never knows whether there is even a problem. And if there is a problem, there is no meaningful information about the cause. Second, in the absence of proper feedback, the application cannot easily determine when to fall back to a different mechanism to receive the content, e.g., to use Automatic Multicast Tunneling (AMT) [13] or an Application Layer Multicast (ALM) solution [15].

NOSSDAV'05, June 13–14, 2005, Stevenson, Washington, USA.

In this paper, we propose two solutions for making group joins more robust. The first is to gather information about the nature of multicast support using *existing protocol mechanisms*. In this approach, we limit ourselves to existing protocol messages to determine the availability and nature of the multicast service supported by the underlying network. Because the existing protocols were not designed to provide feedback, the result provides less than full knowledge about what the network supports. However, the gathered information in combination with existing multicast debugging techniques makes it possible to determine the availability of multicast.

The second solution that we propose for making group joins more robust involves the introduction of additional functionality to collect network feedback about the fate of group-joins sent into the network. While this approach requires modifications to the multicast infrastructure, the result would help us achieve robust group-joins and a more robust multicast architecture. In this way, a user application would be notified at once about the failure of a join request. Additionally, optional information about the nature of the failure could be delivered to the user. This information could in turn be forwarded to a network administrator to assist in debugging. While this solution provides complete feedback, the deployment of the required modifications is a challenge. The tradeoff is either that we have less-than-full knowledge about the join process or we have full knowledge which necessitates making changes to the multicast infrastructure.

As part of our contribution in this paper, we attempt to evaluate the *invasive* solution (requiring modifications) as well as the *non-invasive* solution (not requiring modifications) and their implementation complexities. For the non-invasive solution, we have implemented a tool called the *Multicast Detective*. We also elaborate on how non-invasive techniques can be integrated into a multimedia content application, e.g., Real Network's Helix platform[1]. One lesson learned from using our tools is that a point of failure located far from the user application is harder to identify using non-invasive means.

The remainder of this paper is organized as follows. In Section 2 we provide the background and motivation for our study. In Section 3 we describe our Multicast Detective solution. Section 4 elaborates on our implementation and evaluates the proposed architecture. The paper is concluded in Section 5.

2. BACKGROUND AND MOTIVATION

In the current IP multicast architecture, the multicast group join process is initiated by a user application opening a socket and setting a multicast-specific socket option. This apparent simplicity at the application layer along with the lack of any join success/failure feedback leave the application in a non-deterministic waiting state without proper information about potential failures in the join process. Hidden by this application-perceived simplicity is a multicast group join function involving several steps. In this section we describe these steps, the different points of failure, and the problems associated with the current group-join process.

Figure 1 shows the different components in the multicast group join process. The two endpoints of a multicast tree are the *user application* (#1 in Figure 1) and the *source application* (#6 in Figure 1). The user application interfaces with the host socket interface, which in turn uses one of several multicast group management protocols. These include any of the three Internet Group Management Protocol (IGMP) versions [7, 3] or either of the two Multicast Listener Discovery (MLD) protocol versions [5, 14]. The *IP Stack* (#2 in Figure 1) will support some of these protocols, and through these protocols, allow an application to join or leave a multicast group dynamically. The *first major challenge* for the user application is to determine the group membership protocols supported by the Host IP Stack. This information is necessary for the application to be able to use the proper socket options to join (and leave) a multicast group.

The *second challenge* arises when the Host IP Stack sends a membership report packet towards the first hop router. The membership report travels through one or more *Switches* (#3 in Figure 1) on its way to the first hop router. If IGMP snooping [8] is enabled on these switches, the specific protocol version of the membership report becomes important. An IGMP snooping switch that understands IGMPv2 may suppress IGMPv3 (and MLDv1 and MLDv2) packets resulting in a lost group join request. In addition, until recently, IGMP snooping was considered a protocol optimization and there was no IETF standard for how a switch should handle group membership messages. Only recently have researchers begun work to solve this problem [8]. As a result, a large percentage of group-join problems are because of incorrect switch operation.

Once the IGMP group membership report arrives at the *First Hop Router* (#4 in Figure 1), it is interpreted according to the router's capability. As a result, different failure conditions may occur. A "multicast unaware" router simply drops the membership message. Similarly, an IGMPv2 enabled router ignores IGMPv3 membership reports. And finally, problems can occur for the MLD family of group membership protocols because the router may not understand IPv6. Thus, even though a large number of problems are possible, there is no mechanism for sending feedback to the Host IP stack. Similarly, the user application will be unaware of any problems and will likely simply wait for data to be delivered via the open socket.

If no problems arise and the first hop router is already part of the multicast distribution tree, the join request will complete successfully. Of course, the user application may continue to wait for data if the source application does not happen to be transmitting. On the other hand, if the first hop router is not yet part of the multicast distribution tree, additional steps must be completed.

If the first hop router is not already part of the multicast distribution tree, it must graft itself onto the tree. In this process, routers use multicast routing protocol(s) to establish and maintain a *multicast forwarding tree* between the source(s) and receiver(s) (#5 in Figure 1). In this study, we consider the current pre-dominant multicast routing protocol called Protocol Independent Multicast-Sparse Mode (PIM-SM) [4] for establishing the forwarding trees. In PIM-SM, when an IGMP membership report is received, the first hop router initiates a PIM-Join request towards the source or a designated local router called the Rendezvous Point (RP). If the PIM-Join request eventually reaches the source application's edge router or reaches an intermediate router with forwarding state for the requested group, the router is successfully grafted into the multicast tree.

The *third challenge* concerns problems that arise related to PIM-SM. Previous monitoring studies [11] have shown that while there are "islands" of multicast connectivity, not every island is reachable from every other island. In this case, a PIM-Join request destined to a remote source may be dropped before reaching either the source or an already-connected intermediate router. In addition, configuration problems, or a complete lack of multicast capability, can cause PIM-Join requests to be mis-routed or dropped completely.

Given the numerous problems that can occur with the multicast group join process, a user application waiting on a socket to deliver data cannot be sure whether a lack of data is due to (1) silence of

[1] http://www.helixcommunity.org/

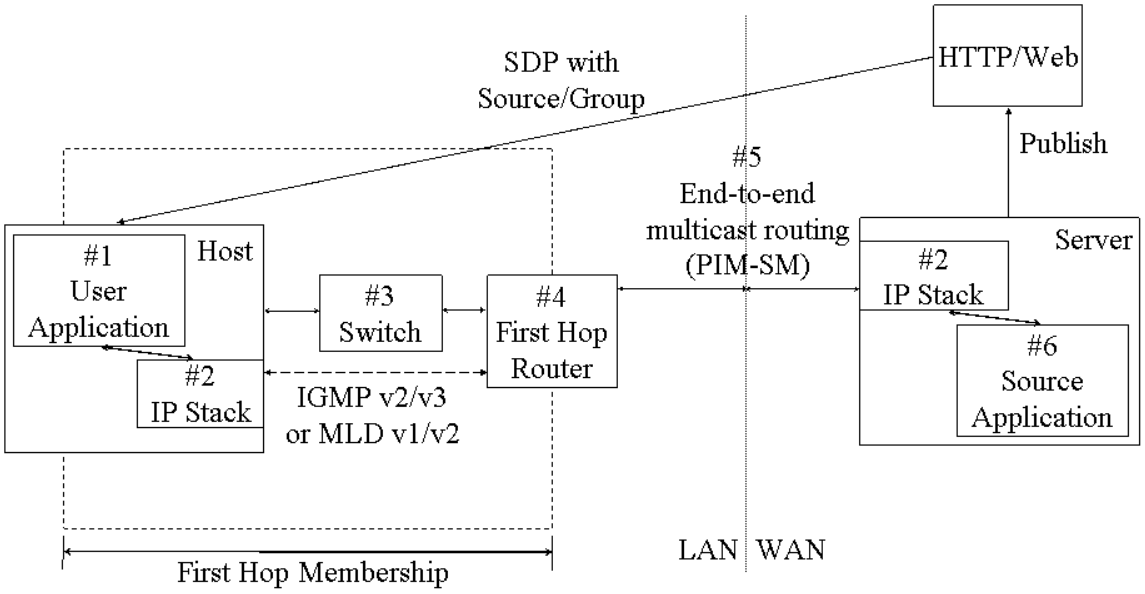

Figure 1: The multicast group join process.

the source, or (2) any of the number of possible multicast protocol problems. These problems create the *fourth challenge*. Fundamentally, multicast signaling is implemented only among network elements that do not have a feedback loop and without feedback to user applications. Because Any Source Multicast (ASM) allows any source to transmit to a multicast group address and does not provide receivers a way of knowing the set of sources, it becomes extremely difficult for a receiver to verify whether there are *any* active sources and whether data is being delivered from *every* active source.

The evolving nature of IP multicast in the Internet along with the inherent disconnect between sources and receivers makes the task of joining a multicast group quite difficult. To summarize, the major problems associated with joining a group are: (1) a lack of feedback to detect multicast support at the edge of the network, (2) membership protocol message exchange failure between end hosts, switches and routers, (3) multicast routing protocol errors, and (4) a lack of application-layer coordination among group senders and receivers.

The solution to these problems will likely include enhancing an application's knowledge of multicast service availability in the network. At a minimum, a proactive, network-aware application should at least be able to determine the multicast capabilities of the network components within the scope of the local subnet. The application can then avoid waiting on a futile join request when the underlying network does not support multicast. In the first place, if it can be determined quickly that multicast does not exist, an application might then be able to use an alternative [15, 13].

Determining multicast capability of the underlying network is not always easy. The different types of multicast service, namely Any Source Multicast (ASM) and Source Specific Multicast (SSM) make the determination complicated. For example, the network could support ASM but not SSM; vice versa; both; or neither. Given that there are numerous options, we believe there is an obvious need for an intelligent middleware to provide a unified interface for seeking multicast content.

Several studies have been conducted to address the multicast heterogeneity problem [12, 10]. Swan et. al. propose a multicast session layer called Aspen [12]. Aspen tries to simplify the Application Programming Interface (API) for multicast by hiding the details of whether the network uses ASM, SSM or some application layer multicast protocol. Aspen allows users to specify session parameters which are used internally to build appropriate distribution tree. Our proposed solution would be complementary to the Aspen approach in that it would give deterministic feedback on whether multicast is working. This determination is a key step for Aspen in that it must try each alternative and determine that a failure has occurred before trying the next alternative.

In the MAGNA [10] architecture, Mathew et. al. suggest a software agent at the Presentation/Session layer to provide support for streaming multimedia in heterogeneous environments. MAGNA is a much broader platform that deals with a variety of network as well as content issues. And while MAGNA identifies the need for robust group management, the details of how this is accomplished are not described. Our solution is again complementary and could easily be included as part of the MAGNA architecture.

Both of these platforms could therefore benefit from a solution that provides information about the multicast capabilities of the network. Our proposed *Multicast Detective* would provide precisely this necessary information.

3. THE MULTICAST DETECTIVE

The multicast group join process is fragile because there are multiple points of potential failure along the path between the user application and the source. Our goal is to provide feedback on whether this process seems to be working correctly. We propose two different approaches to improve the robustness of the group join process. Both approaches develop mechanisms to allow the user application to detect and potentially identify the reason for a join failure. The first approach attempts to do this without making any changes to existing multicast infrastructure. The second approach leverages a set of proposed changes to existing protocols in order to develop a feedback mechanism between the network and applications.

Figure 2 depicts the generic architecture for our multicast capability detection tool, called the *Multicast Detective*. This tool initiates a series of protocol queries and infers the extent of multicast

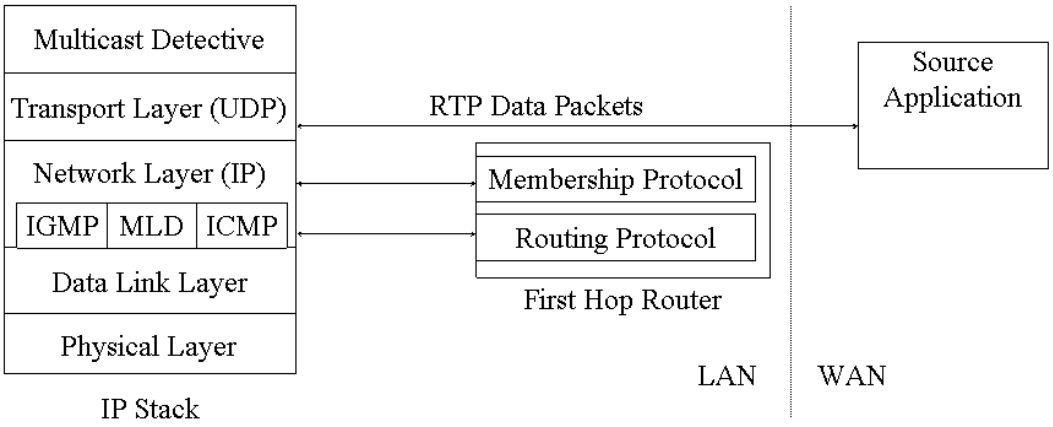

Figure 2: The *Multicast Detective* architecture.

support based on the responses. The goal is to verify the functional correctness of both individual protocols as well as the communication between different protocols. Treating individual components separately helps to isolate problems into logical areas and facilitates in-depth investigation. The Multicast Detective begins by first evaluating multicast support within the local LAN and then gradually expands the scope of diagnosis towards the source.

As Figure 2 shows, the Multicast Detective is developed using a receiver-centric approach for diagnosis of multicast capability. It performs three types of tests. In the first, it simply joins existing multicast groups known to constantly source traffic (e.g., session announcement protocol (SAP) group or the Multicast Grid Beacon group) to determine if multicast connectivity to at least some sources exists. The second set of tests consists of issuing multicast membership messages to test the capabilities of the host and the LAN. The third set of tests consists of protocol messages to determine whether there is a multicast routing protocol running on the first hop router and then whether there is multicast support beyond the first hop.

There are two main components to verify multicast functionality: (1) exercising multicast group join membership and routing protocol message exchange, and (2) testing multicast packet delivery. The Multicast Detective uses both techniques to gather as much information as possible about multicast operation. By exercising multicast membership and routing protocols, the Multicast Detective directly evaluates the robustness of group join and tree management functions. The reception of multicast data can be used to indirectly verify correctness and directly verifying the overall success of the join process.

The techniques used by the Multicast Detective can be run as a stand-alone tool or can be integrated into an intelligent, network-aware application. In the next section we consider two implementations, one stand-alone and the other integrated into an existing open-source content delivery system. The goal is to understand whether the functionality of the Multicast Detective can be packaged as a library such that future multicast application developers can avoid knowing about the overly complex details of multicast operation and instead are given a more robust multicast socket facility.

4. IMPLEMENTATION AND EVALUATION

In this section we present our work to gather information about whether the multicast group join process is working correctly or not. As mentioned previously, there are two main approaches to provide this information: (1) using existing multicast protocol messages, and (2) introducing additions to existing protocols. In the first instance, we leverage existing multicast network facilities to determine whether group-joins are likely to succeed. For this case, we have developed two slightly different implementations. First, we have developed a stand-alone version of our solution called the Multicast Detective, a tool similar to the Internet2 Detective[2]. Second, we have integrated some of the Multicast Detective techniques into an open source multimedia content delivery platform called Helix. In our evaluation of these solutions, we have found both to be most effective only within the local subnet of a group receiver. In Section 4.3, we describe the limitations of our first two approaches and propose a set of primitives/functions to existing protocols in order to provide deterministic feedback to applications about whether a specific group-join was successful.

4.1 The Multicast Detective

The Multicast Detective is a stand-alone tool that gathers information about a network's multicast capability using a host's existing socket API. Since part of our solution requires creating and sending several IGMP and ICMP messages into the network, the Multicast Detective uses raw sockets for creating and communicating these messages. The advantage of this approach is that it does not have application-specific constraints. For example, we can assume administrative privileges which are necessary to use raw sockets. The membership (IGMP or MLD) and routing (PIM-SM) protocols are the most critical components of the group join process. An important feature is that the availability of raw sockets helps us run different tests related to these protocols.

While the Multicast Detective can effectively determine if multicast exists, it does not directly make multicast application development more robust. However, one option is to turn the Multicast Detective into a library and have other applications execute it as a stand-alone application. This would allow applications to know if there is a problem while resolution of the problem is left to network administrators. This is our second approach and is described in the next section.

The initial version of the Multicast Detective is a command line tool running on an IPv4-based Windows XP platform. The tool runs a series of diagnostic queries to infer multicast network characteristics within the local network. The following list describes the series of diagnostic tests that are run by the Multicast Detective. On the basis of the outcomes of these tests, the tool determines the multicast capabilities available in the network.

[2]http://detective.internet2.edu/

Step 1. The Multicast Detective, similar to the Internet2 Detective, joins the NLANR Multicast Beacon Group. The tool checks to see if any data packets are received from the group. After receiving data from the group, the tool checks the IP address of the source to make sure that the source and the receiver do not reside in the same subnet and/or in the same domain. Because there are almost always sources transmitting to this group, if no data packets are received, we conclude that the group join failed. Data that has been successfully received indicates that the membership protocol is working in the local subnet and the routing protocol has worked for at least some sources in the group. There is no guarantee however, that multicast is working for *all* sources sending to the group.

Step 2. The tool issues an SSM join with a known group and source address by setting the proper socket options and then listening for data packets. Currently we use an Internet2 Multicast Working Group SSM source to serve as the target group. If the socket API fails while setting the source specific option, we can conclude that host IP stack does not support SSM. If the socket API call returns successfully but we do not receive any data packets, we can assume there is a problem somewhere beyond the host.

Step 3. The tool sends ICMP ping messages to the All-Multicast-Routers and All-SSM-Routers group addresses and listens for responses. The results of Steps 2 and 3 determine the capability of the host and network up to the first hop router. A successful ping response from the All-Multicast-Router group indicates that the first hop router is capable of accepting multicast membership requests for ASM. Similarly a response to the All-SSM-Router group ping demonstrates the router's capability of accepting SSM membership messages.

Step 4. In this step the Multicast Detective looks more closely at the message exchanges for a group-join. The tool creates and sends an IGMP join message followed by an IGMP leave message for an arbitrary group address and then listens for a Group-Specific-Query from the router. This technique not only ensures that IGMP packets can be exchanged seamlessly between the host and the first hop router, but it also verifies that the membership protocol is operating correctly. A Group-Specific-Query confirms that switches along the path to the first hop router are not blocking group-join messages. However, one problem that can still arise is for switches on the path to treat multicast packets as broadcast packets and deliver the data to all switch ports. While not affecting the group member's ability to receive data, it might create congestion on switch ports that never explicitly had a member of the multicast group.

Step 5. In order to test the multicast routing protocol, the tool sends an ICMP ping message to the All-PIM-SM-Routers group and listens for a response. It also constructs and sends a PIM Hello message from the host pretending to be an PIM-SM enabled router. A ping response verifies that the first hop router is listening to the All-PIM-SM-Routers group. In addition, if the tool receives a PIM Hello response, we can conclude that PIM-SM at the router is operational and active for the interface on which the host resides. The earlier version of PIM-SM specifies that a PIM Hello response must contain an RP-set which includes the location of the domain's RP. If this information is returned, it is a good indication that multicast is working in at least the local domain.

4.2 Integrated Client–Helix Player

Helix is a state-of-the-art open-source multimedia content delivery platform lead by Real Networks. The challenge with implementing the Multicast Detective techniques in Helix is that Helix is not able to use low-level network probing techniques that require administrative privileges. This limitation severely constrains the number of possible network diagnostic steps that can be integrated directly into the Helix player. One option is to run only the set of Multicast Detective diagnostics during installation of the player to determine multicast network capabilities. This approach, however, is sub-optimal, given that network conditions could and often do change. Another alternative is that when the user application fails to receive content, the Multicast Detective could be started in parallel and diagnostic information then collected through a Helix-based API.

Assuming neither of these choices is possible, a subset of techniques employed by the Multicast Detective could be integrated into a Helix client and run without administrative privileges or access to raw sockets:

1. As a user application, the Helix player can join the Multicast Beacon Group and listen for data packets as described earlier.

2. The Helix player can also join other groups in a process similar to joining the multicast beacon group. The application could join the Simple Network Time Protocol (SNTP) group or the the Session Announcement Protocol (SAP) group and listens for data packets.

3. The player can determine whether the host IP stack supports IGMPv3 by attempting to set the source specific join option and observing the result. In this way, the player can determine the availability of SSM support in at least the host IP stack.

The Multicast Detective provides a comprehensive way to gather information about multicast operation in the first hop and beyond. This information could help user applications understand what kind of multicast support exists. On the other hand, the Multicast Detective falls short in terms of locating and isolating potential multicast problems if the problems occur beyond the first hop router. While the Multicast Detective can be used to detect the existence of multicast support, it cannot be used as a feedback mechanism for a specific group-join attempt. Such feedback is critical to truly improve the robustness of the group join process.

4.3 Multicast Infrastructure Changes

We believe that some of the limitations of the Multicast Detective can be overcome by introducing changes to existing multicast protocols. The main goal of these changes is to improve the robustness of group joins by including necessary functionality as part of the multicast protocols themselves. We propose a set of changes in membership and routing protocols to improve the robustness of the join process:

Group Join Feedback. A group membership acknowledgment (IGMP-ACK and MLD-ACK) message would help to complete the feedback loop between the application and first hop router. A group membership acknowledgment issued by the designated multicast router in response to unsolicited membership reports would improve the robustness for host-to-router communication at receiver sites.

Group Membership-Routing Protocol Feedback. A handshake mechanism between IGMP/MLD and PIM-SM would allow any PIM-SM-related problems to be propagated back to the joining host through the group join feedback message. For example, if the first hop router had no route to the source address or no RP was configured, this information could be communicated back to the host as a join failure.

Group Membership Status Query. Another solution would be to create an IGMP/MLD Join-Status query/response mechanism.

The receiver host could use the Join-Status query to learn the result of a join message. The first hop router would consult the forwarding state entry for the indicated group and send a Join-Status response message back to the querier. This solution would require multicast forwarding state to include a new field to indicate the result of the join request.

PIM-SM Per-Hop Join Acknowledgments. To improve the robustness of PIM-SM, the join request message could be acknowledged at each hop. After the new forwarding path successfully grafts to the existing forwarding tree (or reaches the source of the tree), a PIM-Join ACK message would be forwarded downstream to the first hop router. For a failure situation, this message could be similar to an ICMP host/network unreachable error.

Support for Automatic Tunneling. The network can support auto tunneling of multicast traffic using traditional unicast communication from the edge of the multicast island to the receiver [13].

Receiver-to-Source Signaling. This modification would not necessarily require a change in the multicast infrastructure, but would require a change in sources and receivers. A simple source receiver signaling protocol could be developed for direct contact between a receiver and a transmitting source. In this way, the receiver would be able to verify that packets were indeed being generated by the source. If the receiver is not receiving packets, it could use an alternate means of delivery.

5. CONCLUSIONS AND FUTURE WORK

We have in our work addressed one of the most important problems with multicast: determining the existence of multicast support within the network and identifying ways to make the group join process more robust. Achieving robust multicast group joins without changes in the network or socket API is virtually impossible. Without changes, the best we can hope to gain is limited knowledge about whether or not problems exist along the join path. Even this small bit of information is useful for improving the robustness of the multicast join process. There is more work to be done. An eventual goal for multicast should be a robust solution that simply returns an error via the socket API if any part of the join is unsuccessful. While this solution would require significant changes within the network architecture, it is an important requirement for making multicast truly usable.

6. REFERENCES

[1] K. Almeroth. The evolution of multicast: From the MBone to inter-domain multicast to Internet2 deployment. *IEEE Network*, 14(1):10–20, January/February 2000.

[2] S. Bhattacharyya. An overview of source-specific multicast (SSM). Internet Engineering Task Force (IETF), RFC 3569, July 2003.

[3] B. Cain, S. Deering, I. Kouvelas, B. Fenner, and A. Thyagarajan. Internet group management protocol, version 3. Internet Engineering Task Force (IETF), RFC 3376, October 2002.

[4] S. Deering, D. Estrin, D. Farinacci, V. Jacobson, G. Liu, and L. Wei. PIM architecture for wide-area multicast routing. *IEEE/ACM Transactions on Networking*, pages 153–162, April 1996.

[5] S. Deering, W. Fenner, and B. Haberman. Multicast Listener Discovery (MLD) for IPv6. Internet Engineering Task Force (IETF), RFC 2710, October 1999.

[6] C. Diot, B. Levine, B. Lyles, H. Kassem, and D. Balensiefen. Deployment issues for the IP multicast service and architecture. *IEEE Network*, 14(1):78–88, January/February 2000.

[7] W. Fenner. Internet group management protocol, version 2. Internet Engineering Task Force (IETF), RFC 2236, November 1997.

[8] F. S. M. Christensen, K. Kimball. Considerations for IGMP and MLD Snooping Switches. Internet Engineering Task Force (IETF), Internet Draft, (work in progress), February 2005.

[9] D. Makofske and K. Almeroth. *Multicast Socket Practical Guide For Programmers*. Morgan Kaufmann, New York, 2003.

[10] R. Mathew. Providing seamless access to multimedia content in heterogeneous environment. Master's thesis, Dept of Computer Science, University of California, Santa Barbara, September 2004.

[11] K. Sarac and K. Almeroth. Application layer reachability monitoring for ip multicast. *Elsevier Computer Networks Journal*, 48(2), June 2005.

[12] A. Swan and L. Rowe. Aspen: A multicast session layer. In *Open Mash Consortium*, March 2004.

[13] D. Thaler, M. Talwar, A. Aggarwal, V. L., and P. T. Automatic ip multicast without explicit tunnels (AMT). Internet Engineering Task Force (IETF), Internet Draft, (work in progress), February 2005.

[14] R. Vida and L. Costa. Multicast Listener Discovery Version 2 (MLDv2) for IPv6. Internet Engineering Task Force (IETF), RFC 3810, June 2004.

[15] S. S. Y.-H. Chu, S. Rao and H. Zhang. Enabling conferencing applications on the internet using an overlay multicast architecture. In *Proceedings of ACM SIGCOMM*, San Diego, California, August 2001.