Topology Discovery Service for Router-Assisted Multicast Transport*
CMPSCI TR 01-55

Jonathan K. Shapiro[a] Jim Kurose[a] Don Towsley[a] Stephen Zabele[b]
[a]Department of Computer Science
University of Massachusetts at Amherst
{jshapiro,kurose,towsley}@cs.umass.edu
[b]Litton-TASC Inc.
gzabele@tasc.com

January 9, 2002

Abstract

Many existing proposals for introducing network support for multicast transport require the establishment of signaling paths among adjacent active routers in a session. We present a general-purpose, lightweight protocol to establish a signaling overlay among sparsely deployed active nodes. The resulting overlay is tied to the underlying multicast route and adaptive to changes in its topology. In addition, we make this overlay available to other protocols by means of efficient communication primitives to provide reliable signaling between neighboring active routers. Our protocol and associated services can serve as a building block for a variety of active multicast services and greatly simplify their development. We describe its applicability for implementing several previously proposed services.

1 Introduction

Several proposals have emerged in recent years for the deployment of active services in routers to support multicast transport. In the earliest proposals, the augmented router functionality was targeted at facilitating reliable multicast [6, 8, 13, 11, 15, 7], a problem that has resisted a scalable solution using purely end-to-end techniques. Other work [17] promotes active filtering services to improve the performance of specific applications. Recent efforts [3, 4] have sought to generalize previous work by defining a simple set of services (feedback aggregation and suppression, subcast, directed multicast), invocable from the edges of the network, that can be composed to support a range of multicast transport protocols.

A requirement for all of the proposals mentioned above is that control messages traverse the reverse multicast path from receivers to the source, receiving hop-by-hop processing along the way. Implementing such a signaling path is challenging because receiver-to-source unicast traffic need not follow the reverse multicast path. Furthermore, the realities of incremental deployment make it unlikely that all routers along the reverse path would be active (i.e. capable of processing such control messages). The existence of this common problem, combined with the widespread adoption of source path messages (described below) to solve it, presents an opportunity to meet a fundamental requirement of many protocols with a common

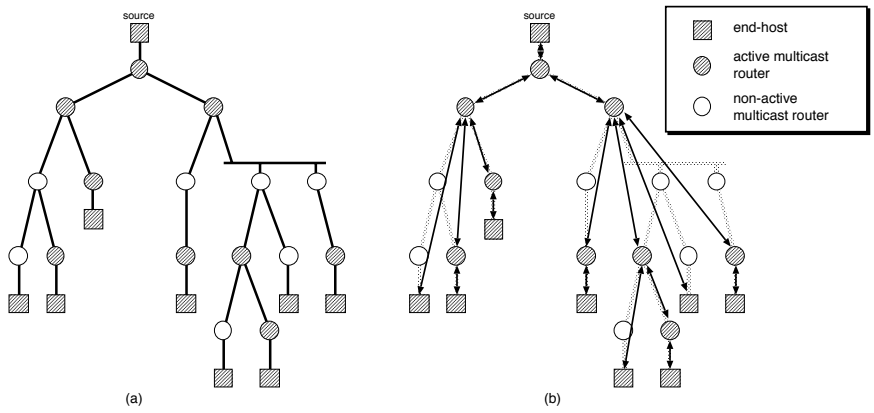


Figure 1: A multicast tree with sparsely deployed active routers (a) and the corresponding overlay formed among the active nodes (b).

architectural component.

The fundamental problem for reverse-path signaling is one of topology discovery. Under sparse deployment conditions, as depicted in Fig. 1-a, adjacent active nodes may be separated by one or more non-active routers, which can be relied upon to forward packets but not to provide any additional service. A mechanism is required to allow adjacent active nodes to discover each other and to form a signaling overlay as shown in Fig. 1-b. The topology of the overlay must be congruent to the underlying multicast tree and must adapt to changes in multicast routing due to member joins and leaves. Once established, such an overlay allows control messages to travel active-hop-by-active-hop along the forward and reverse multicast paths.

In this work, we seek to isolate the requirements of overlay formation and maintenance from other active services by providing a lightweight topology discovery mechanism that can serve as a building block for these other services. To this end, we present a preliminary design for an Active Topology Discovery Protocol (ATDP) to maintain bidirectional reliable connections between each active node and its upstream and downstream neighbors in a source-based multicast tree. ATDP uses these connections to provide an efficient active-hop-by-active-hop signaling service for other protocols operating over the same tree. By treating overlay formation independently, we hope to obviate its redundant implementation in the various active transport services for which it is required. In addition, we find that the resulting overlay provides a useful abstraction for developing higher level transport protocols, greatly simplifying their design. This advantage is retained even if all routers in the network are active.

The remainder of this paper is organized as follows. In the next section we review related work. Section 4 presents the ATDP topology discovery protocol internals. Sections 5 and 6 describe the services that ATDP offers to higher level protocols. We present two use cases in section 7, showing how ATDP can support two protocols that implement very different active services. We conclude in section 8 with a short summary and a view to future work.

2 Related Work

It is widely believed that a scalable solution to reliable multicast transport is difficult to achieve without some form of network support. Feedback implosion at the sender and the exposure of receivers to redundant retransmissions severely limit the applicability of a naive approach based on negative acknowledgements

*This work was supported in part by the Defense Advanced Research Projects Agency (DARPA) under grant number N66001-99C-8614.

(NAKs). More sophisticated approaches, such as RMTP [9], attempt to organize receivers into a recovery hierarchy that limits both exposure and feedback at the source. To be effective, however, the hierarchy must group receivers according to topological proximity, which is difficult for end-hosts to determine accurately without significant control overhead. Scalable versions of both the NAK-based and hierarchical approaches are possible with the introduction of modest additional services at active routers within the network.

PGM [6] implements a NAK-based reliable multicast protocol, relying on network support to prevent feedback implosion and receiver exposure. Active routers suppress redundant NAKs by forwarding only the first NAK for a given packet sequence number toward the source. Subsequent NAKs for the same sequence number are dropped, but the downstream interfaces on which they arrive are marked. The scope of retransmissions from the source is then limited to only those marked interfaces. Receivers observe an exponential back-off prior to sending a NAK, which allows the upstream active router to suppress redundant NAKs by multicasting a confirmation message on the interface of an arriving NAK.

In addition to its introduction of network services to support transport, PGM is highly relevant to our work because it introduced the use of *source path messages* (SPMs) for topology discovery. An SPM is a packet periodically sent to a multicast address that is treated specially by active routers so as to allow discovery of upstream neighbors. An SPM is marked with the IP router-alert option, signaling routers along its path to inspect its contents and, in the case of active nodes, perform some additional processing. The SPM contains a field with the the address of the last active node that forwarded it, initialized to the source address. On the arrival of an SPM at an active node, the node stores the value of this field—indexed by source and group addresses—and overwrites the field with its own address on the forwarded copies. The effect of the SPM traversing the multicast tree is to define reverse signaling paths from the receivers to the source constructed from those active nodes present on the forward multicast path. Periodic multicast of fresh SPMs allows the signaling path to adapt to changes in the underlying tree, and, since parent addresses are stored as soft-state, unused branches to be pruned automatically.

Other network-supported reliable multicast schemes have been proposed [8, 13, 11, 15, 7]. Space does not permit us to review them all here. It is worth observing, however, that LMS [10] differs significantly from PGM by enlisting the network to support a hierarchical approach. PGM and LMS offer different tradeoffs between storage requirements at active nodes and the degree to which receivers participate in error recovery. Like PGM, however, LMS relies on the use of SPMs to construct a signaling overlay.

The observation that different transport protocols can make use of network support has motivated recent efforts [3, 4, 5, 14] to define a set of general but simple functions that can be variously composed to support a range of transport protocols. A general active services architecture such as the Generic Router Assist (GRA) effort of the IETF would, for example, allow diverse reliability protocols such as PGM and LMS to coexist in the network. Scalable single-rate multicast congestion control protocols that track the worst-performing receiver, exemplified by [12] and [1], can be easily supported by a generic feedback aggregation service. The scope of such efforts go well beyond what we propose here. For example, SPMs serve the additional purpose of declaring the services to be used by the session and initializing network state. However, the establishment of a signaling overlay remains an essential requirement. Our work can thus be considered a building block for the GRA architecture.

A recent proposal by Braden and Lindell [2] for a generic architecture to support hop-by-hop signaling to install state along network paths is strongly related to our own work. The authors state the need for topology discovery to allow signaling between adjacent active nodes and identify a basic mechanism similar to SPMs. Our work our work is more directly focused on multicast and compliments this proposal by introducing communication primitives explicitly designed to fascilitate multicast protocols.

Other related work has an application-layer flavor. Zabele and Stanzione [17, 16] have proposed an active filtering service in routers to significantly reduce the network traffic in distributed interactive simulations (DIS). Large-scale publish-subscribe applications like DIS are characterized by large numbers of

both senders and receivers, hence, many source-based trees and relatively few multicast addresses. From a signaling perspective, such applications are interesting because the required overlay for a multicast group is the mesh formed formed by the union of many source-based trees. Our work is complementary in that we have designed communications primitives to provide the abstraction of a signaling mesh on top of a multicast architecture that supports only source-based routing.

3 Assumptions and Terminology

We assume the use of source-based multicast trees in this document, although the protocol presented can be readily extended to shared trees. We use the term *active node* to refer to any node running the topology discovery protocol described in this document. The source and receivers in a session are all assumed to be active, as is some subset of routers in the associated multicast tree. Unless otherwise qualified, the terms *parent* and *child* refer to the nearest upstream and downstream active nodes, respectively. Note that a parent active node is not necessarily identical to the immediate parent in a multicast tree, since there may be intervening routers that are not active. We will sometimes refer to adjacent active nodes as *peers* when we wish to deemphasize their parent-child relationship.

4 Topology Discovery

The Active Topology Discovery Protocol (ATDP) maintains, for each (s, g) pair[1], bidirectional reliable connections between each active node and its parent and children. We assume that in addition to ordinary multicast forwarding capability, an active router provides the ability to determine the ingress interface of an arriving packet and the ability to modify fields in the multicast packet header prior to forwarding. In particular, it is necessary to make different modifications to the copy forwarded over each outgoing interface. ATDP uses this ability to implement *source path messages* (SPMs) sent over the forward multicast path.

SPMs originate at each multicast source and are sent periodically to the group address by ATDP.[2] Each SPM packet contains a PARENT field that is overwritten by the active node with a contact IP address prior to forwarding. In this way, each node learns the identity of its parent and, as SPMs are multicast repeatedly, learns if the parent has changed. Also, the SPM contains an outgoing interface field that contains the outgoing virtual interface (VIF) identifier of each forwarded copy. This field allows a child to identify the correct parent downstream interface when signaling its parent.

The use of two overwritten fields in the SPM reflects a somewhat subtle design issue that is worthy of comment. An active router needs to know how to contact its parent (contact address) and be able to inform the parent of the associated downstream interface. In a naive SPM implementation, the IP address of the physical egress interface at the parent might provide both pieces of information. However, this technique has two drawbacks. First, a router must assume that two children with the same downstream interface reside on the same subtree and hence can both be reached by a single multicast packet forwarded on that interface. However, this assumption is incorrect when multicast routing uses a mix of tunneled and native multicast. For example, a router might forward an SPM through two tunnels by transmitting two encapsulated copies on a single physical interface. Although the two copies depart on the same physical interface, any signaling protocol should treat them as if they followed two disjoint downstream paths. Second, it may be desirable to separate the contact address from the interface identifier to allow design flexibility. An intriguing possibility,

[1] We use the term *(s, g) pair* to describe a tuple of source and group addresses.

[2] Note that SPMs may also serve as "keep alive" messages to prevent multicast routing protocols from pruning branches of the tree due to inactivity.

for example, is to minimize the load on the router due to signaling traffic by directing contact to a colocated server with a privileged interface to the router.

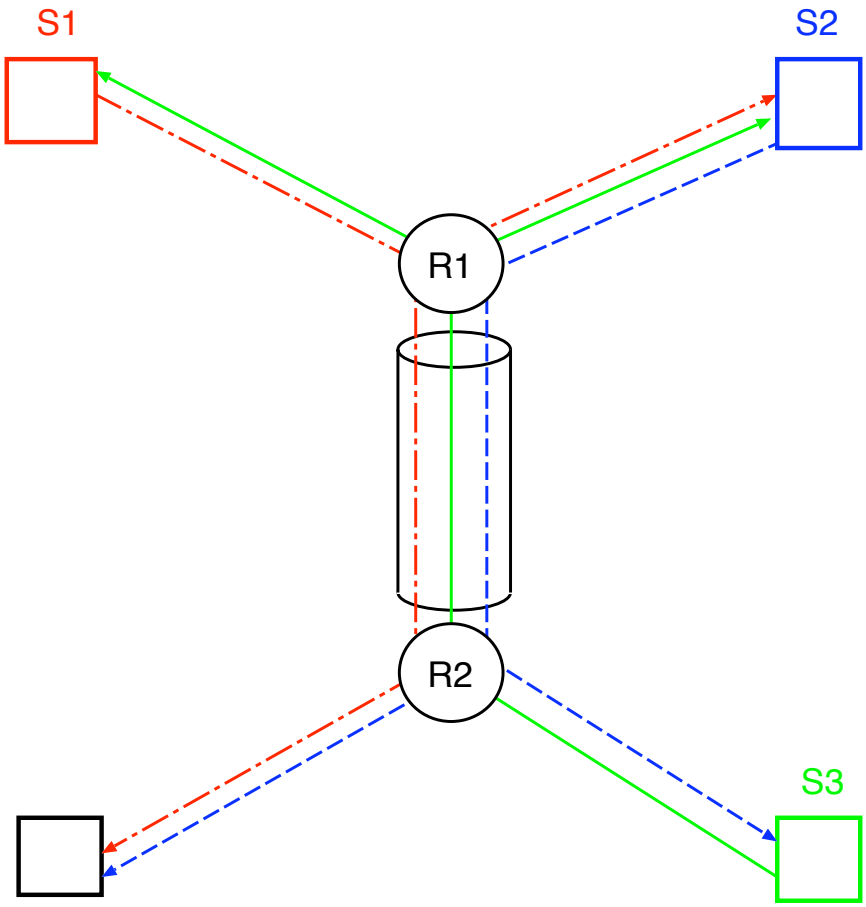


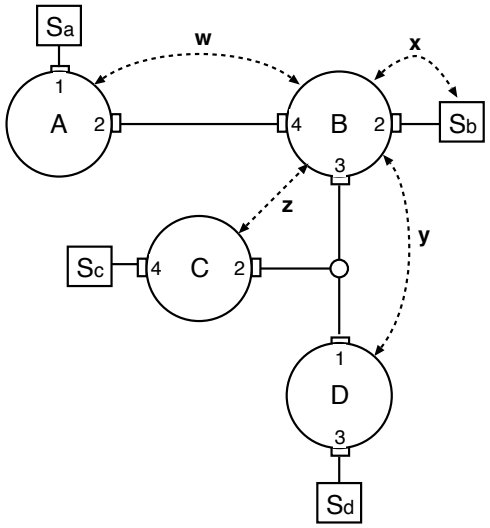
Figure 2: A single connection between R_1 and R_2 multiplexing signals for three different (s, g) pairs. Here we assume that each end host (squares) is a receiver for a group g and that three of the end-hosts (S_1, S_2 and S_3) are senders.

Having learned of its parent via an SPM, the active node will attempt to establish a reliable bidirectional communication channel—which we refer to as an *ATDP channel*—with that parent. ATDP makes this channel available to other local protocol modules for hop-by-hop reliable data exchange. Furthermore, as shown in Fig. 2, ATDP multiplexes signaling traffic between two hosts over a single channel, even if those hosts have parent-child relationships in multiple source-based trees. Thus, if the ATDP channel is connection-oriented, the connection is shut down only if it is no longer used for any (s, g) pair. For the implementation suggested here, TCP will be used to guarantee reliable communication between active nodes. However, it should be straightforward to replace TCP with a UDP-based reliable protocol in the future.

4.1 Internal State Description

The TDF daemon maintains two tables, an example of which is shown in Fig. 3. The *state table* contains an entry for each (s, g) pair for which an SPM has been received. A state table entry contains the source address, group address, a peer record for the parent, and a list of peer records for any attached children. A

peer record is a tuple containing a peer's address, the VIF IDs associated with both local and remote peers, and an ATDP channel identifier. The state table of router *B* in Fig. 3 contains a record for four source-based trees all using group address *g*. The first record, for example, describes the local topology of the tree rooted at S_a in the neighborhood of router *B*. We observe that *B* has discovered that router *A* is its parent and that routers *C* and *D* as well as end-host S_b are its children.



state table at B

```
Sa  g  (A,  4, 2, w)  {(C, 3, 2, z), (Sb, 2, 1, x), (D, 3, 1, y)}
Sb  g  (Sb, 2, 1, x)  {(A, 4, 2, w), (C,  3, 2, z), (D, 3, 1, y)}
Sc  g  (C,  3, 2, z)  {(A, 4, 2, w), (Sb, 2, 1, x)}
Sd  g  (D,  3, 1, y)  {(A, 4, 2, w), (Sb, 2, 1, x)}
```

connection table at B

```
w   A  ... {(Sa, g, 1), (Sb, g, 0), (Sc, g, 0), (Sd, g, 0)}
x   Sb ... {(Sb, g, 1), (Sc, g, 0), (Sd, g, 0)}
y   D  ... {(Sa, g, 0), (Sb, g, 0), (Sd, g, 1)}
z   C  ... {(Sa, g, 0), (Sb, g, 0), (Sc, g, 1)}
```

Figure 3: A simple topology with four active routers and four end-hosts. Each end-host is a sender and a receiver for group g. ATDP channels are shown as dotted lines. The state table at router B shows how four ATDP channels support 14 logical peering relationships.

The *connection table* is used to keep track of currently open ATDP channels. Each entry in this table contains the connection id, peer address, the socket used for the connection[3], and list of route records. A *route record* is a tuple containing the source address, group address, and a parent flag which, if set, indicates that the remote node is the parent in the context of the indicated (s, g) pair. For example, the first row of the connection table in Fig. 3 shows that the ATDP channel *w* between *A* and *B* is reused for signaling in four source-based trees.

ATDP channels can be reused for multiple source-based trees and the route record list provides a form of reference counting for this purpose. We adopt the convention that the child, having initiated the connection to the parent in response to an SPM, is responsible for tearing it down. When the final entry in the route record list is removed, if its parent flag indicates that the local peer is the child, the peer terminates the connection. A timeout mechanism allows the parent to prune any connections to an unresponsive child.

[3]For simplicity of presentation, socket identifiers have been omitted from the connection table in Fig. 3

These data structures are sufficient to maintain the signaling overlay and, as we will see, can be queried by other locally executing protocol modules to provide convenient access to connected peers. All data in these structures is stored as soft state and must be periodically reset by the arrival of SPMs.

To operate correctly, ATDP must maintain three invariant properties with respect to its internal structures.

Invariant 1 *The connection table contains at most one entry for any given remote peer address.*

Invariant 2 *For any entry in the state table, the parent peer does not appear in the list of children.*

Invariant 3 *The setting of the parent flag in the connection table is consistent with the parent-child relationships described in the state table.*

4.2 Protocol State Machine

Conceptually, the state diagram in Fig. 4 runs for each (s, g) and can run on all internal active nodes as well as at receivers. A node is in the unconnected state when it does not know its parent and is in the connected state when it both knows about its parent and has established an ATDP channel with that parent. In the wait state, the node knows about its parent, but has yet to establish the ATDP channel.

There are four message types: SPM, SIGNAL_DATA, CHILD_ACK, and REQUEST_DISCONNECT.

- SPMs are sent along the forward multicast path and allow nodes to discover their parents as described above. When processing an SPM, a node records its parent and writes its own contact IP address into the SPM.parent field.
- SIGNAL_DATA messages encapsulate signaling information sent between active peers via the ATDP channel. In the absence of actual signaling information, periodic empty SIGNAL_DATA messages from children to parents ensure that parent-child connections can be reliably torn down if the child dies.
- CHILD_ACK messages indicate to the parent that a child has successfully established an ATDP channel There may be other messages associated with setting up and tearing down the parent-child connections in addition to CHILD_ACK. For example, if TCP is used for these connections then in some cases there will be SYN and SYN_ACK messages, but these operate below the ATDP state machine and we do not show them here. Due to ATDP channel reuse, some invocations of `connect(SPM.parent)` will merely increment a reference count and not result in a new connection. For the parent to provide reliable notification of the new child in the case when a channel is reused, the child must send a CHILD_ACK message over the existing channel. To simplify the protocol, we require the child to send a CHILD_ACK even when creating a new connection.
- REQUEST_DISCONNECT allows a parent to inform its children that it has disconnected from its own parent (due to an SPM timeout). If this message were not sent, the children would presumably timeout eventually anyway and tear down their connections; this message allows them to do so sooner.

A possible race condition can occur if two nodes attempt to establish a connection to each other at the same time. This condition can happen, for example, when the nodes simultaneously receive SPMs from each other for two different (s, g) combinations. When TCP is used reliable connections, the handshakes initiated by each node can occur in parallel, unobserved by ATDP. Thus, two ATDP channels could be formed between these two nodes. The function `TCPCheck` (Fig. 4) tests for this race condition and ensures that only one of these two connections is retained by forcing the peer with the numerically greater IP address to destroy the connection it created. Note that this function must be called in two places—by a parent that has just received a CHILD_ACK from a new child, and by a child that has just successfully opened a TCP connection to its parent.

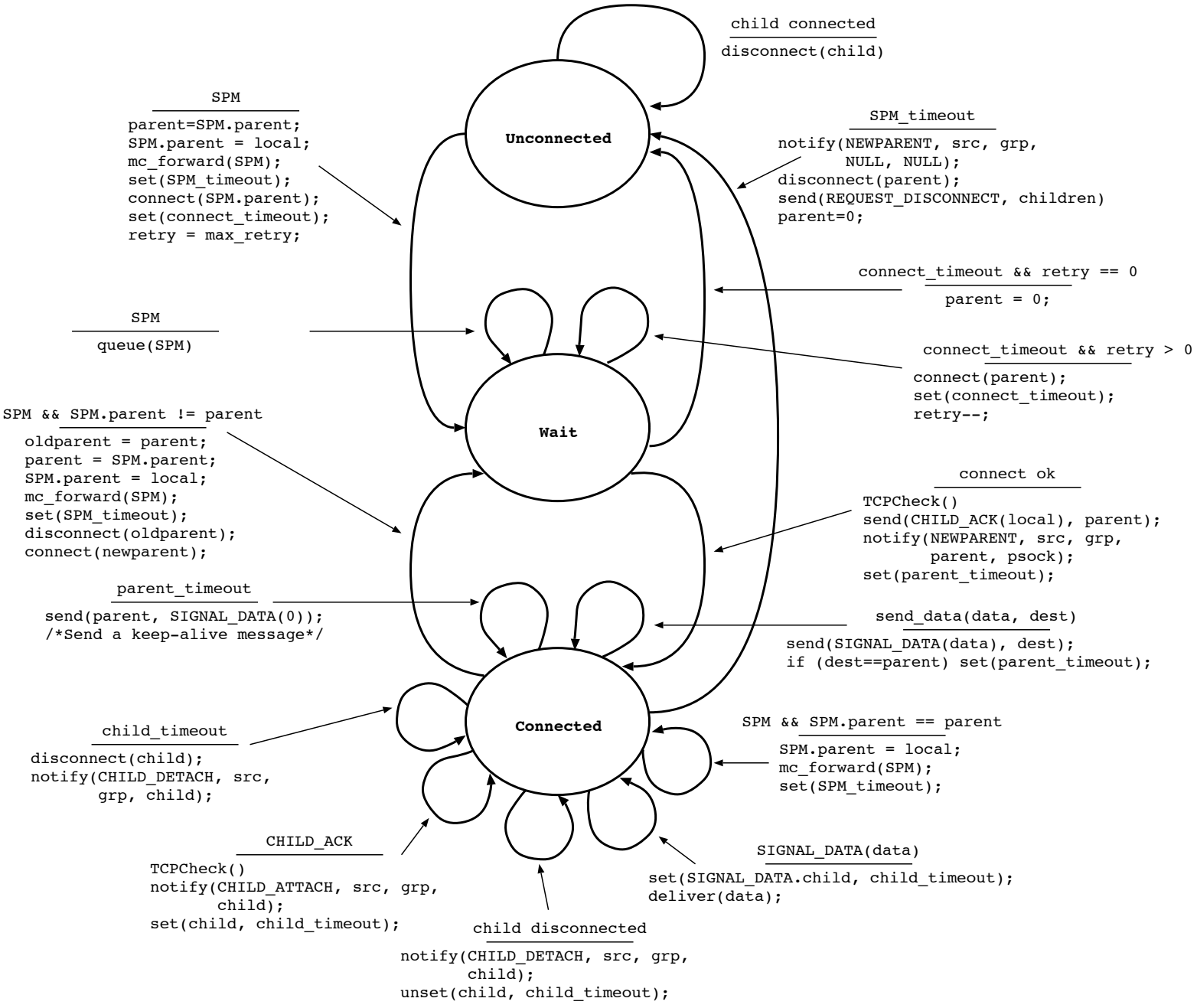


Figure 4: ATDP State Machine.

5 Notifications

ATDP can provide notifications to other locally executing modules in the event of local topology changes. We do not specify the mechanism for providing such local notifications, but rather identify the events of interest and the information that must be communicated for each. Below, the term *local node* refers to the node receiving notifications from a locally executing ATDP daemon.

- `Parent_Changed(src, grp, parentIP, remoteVIF, localVIF)` This notification is sent when the initial parent for a source-based tree is discovered or when a change in the upstream parent has been detected. ATDP notifies other modules *after* reliable communication has been established with this parent.

- `Child Attached(src, grp, childIP, remoteVIF, localVIF)` This notification is sent when a child node has established reliable communication with the local node.

- `Child Detached(src, grp, childIP, remoteVIF, localVIF)` This notification is sent when a child node explicitly tears down its connection or when the local node terminates a connection to a child due to a timeout.

6 Communications Support for Local Protocol Modules

In addition to providing notifications to other protocol modules, ATDP provides interfaces for querying the contents of the state table and for reliably sending and receiving data. The goal of these interfaces is to provide a useful abstraction for addressing control messages to peers based on their topological relationships. Our approach is to use a simple but flexible query engine to select sets of peers according to a variety of criteria and to address messages to the set of peers that match a given query.

One very important design consideration is that it is sometimes convenient to consider the union of source-based trees for a given multicast address when addressing peers.[4] Consequently, it is quite easy to compose a query that selects a peer based on a match in more than one state table entry. When such a query is used to address a message, we say that the message is destined for multiple *logical peers* at a single *physical peer*. For example, in Fig. 2 a query at router R_1 selecting all children associated with group g will return two logical peers associated with the sessions (S_1, g) and (S_2, g), respectively. However, both logical peers reside on the same physical peer, R_2. It is certainly is desirable to transmit a message only once over the ATDP connection to the physical peer. However, whether that message should be interpreted as one message destined for the physical peer or multiple messages destined for the logical peers is application-dependent. Thus, ATDP itself, cannot provide consistent semantics for delivering a message destined to multiple logical peers on a single router, but must instead provide sufficient information to allow the layer above to identify all logical peers to which the message applies if necessary. ATDP appends a fixed-length header to each message for this purpose, which the receiver can use to query its own local state table to find the matching logical peers.[5]

The following two methods permit the retrieval of a set of peers from the state table:

[4] The active filtering protocol we mention in Section 2 provides one example of this requirement.

[5] It is important that the header be fixed length since the number of logical peers is potentially very large. For example, a large-scale distributed simulation with 1000 sources transmitting on a single multicast address can have up to 1000 source-based trees traversing the path between two active nodes, each of which yields a single logical peer.

Argument	Description
`peer`	Peer contact IP address or wild-card
`vif`	Local VIF identifier
`src`	Source of multicast or wild-card
`grp`	Multicast address
`direction`	1 (parent), 0 (child), or wild-card
`mode`	Determines interpretation of first argument. Possible values are `default` or `exclude`.

Table 1: Arguments for state table queries and peer addressing.

Query	Result
`atdpGet(*, s, g, 1, default)`	retrieves the parent for a given (s, g).
`atdpGet(*, s, g, 0, default)`	retrieves all children for a given (s, g).
`atdpGet(*, s, g, *, default)`	retrieves all peers for a given (s, g).
`atdpGet(*, *, g, *, default)`	retrieves all peers for a given group.
`atdpGet(p, s, g, 0, exclude)`	retrieves all children for (s, g) excluding p.
`atdpGetVif(v, *, g, 1, default)`	retrieves all parents for group g whose data arrives on v.
`atdpGetVif(v, *, g, 0, default)`	retrieves all children for group g for whom data departs on v.

Table 2: Examples of the use of wild-cards to construct state table queries

- `atdpGet(peer, src, grp, pFlag, mode)`
- `atdpGetVif(vif, src, grp, pFlag, mode)`

As Table 1 shows, wild-card values are permitted for certain arguments, which permits peers to be specified quite flexibly. Table 2 shows some examples of potentially useful queries.

The following methods,

- `atdpSend(src, grp, peer, pFlag, mode, msg)`
- `atdpSendVif(src, grp, vif, pFlag, mode, msg)`,

reliably send a message to one or more peers. The first five arguments address the message, selecting the set of peers that would be returned by sending the same arguments to `atdpGet` or `atdpGetVif`.

Finally, the method

- `atdpRcv(&peer, &src, &grp, &pFlag, &remoteVif, &localVif, &msg, &len)`,

receives a single message from a peer. On return, arguments contain the message itself and relevant information taken from the message header. `Peer` refers to the address of the message sender, while `src` refers to the source in a source-based tree. The source and parent-flag values may be wild-cards depending on how the message was addressed at the sender. The receiver can recover the locally matching logical peers using `atdpGet` by reversing the designations *local* and *remote* for the VIF arguments, inverting the value of the parent flag (if it is not a wild-card), and explicitly supplying the sender's peer address.

7 Two Signaling Examples

In this section, we illustrate the use of ATDP's communication methods to implement two very different protocols.

7.1 PGM NAK Aggregation

In PGM, receivers unicast NAKs towards the source, with each active router along the signaling path recording sequence number of the requested packet and marking the interface on which the NAK arrived. If no previous repair request for the sequence number has been seen, the NAK is forwarded to the parent. Additionally, a NAK confirmation (NCF) message is multicast downstream on the marked interface, allowing other receivers to suppress redundant NAKs.

PGM NAK-handling could easily be implemented using ATDP's communication methods. Since PGM operates on a source-based or shared unidirectional tree, the source (or rendezvous point) would be explicitly specified in all ATDP methods.

1. A NAK is delivered to the PGM module via $atdpRcv$. In addition to the message itself, the method specifies the source and group addresses, the peer sending the NAK, and the associated remote and local VIFs–in particular, local interface v. The direction parameter is set to 1 since the message was sent from child to parent.

2. $atdpSend(*, s, g, 1, NAK)$ sends the NAK to the parent if no previous NAK has been seen.

3. $atdpSendVif(s, g, v, 0, NCF)$ sends a confirmation to all children reachable via interface v.

7.2 Active Filter Installation

Consider a large-scale publish-subscribe application, say, a multi-player game using a single multicast group, where each player is both a sender and receiver. Active filtering in some routers provides one way to limit each receiver's exposure to updates for players outside its field of vision and to reduce the traffic caused by such unwanted data. The union of parent-child relationships in all source-based trees for this applications forms a signaling mesh for the multicast group. Receivers send subscription messages upstream toward sources.

Suppose each active router provides the following *ingress filtering* service. A subscription filter (indexed by group address) is associated with each VIF specifying the union of all downstream subscriptions. When a multicast packet arrives at the VIF, the router compares the packet with the filter to determine if any downstream interest exists. If not, then the packet is immediately discarded.[6]

A protocol for establishing ingress filters must ensure that subscription state propagates upstream to all sources to prevent data for which no interest exists from being transmitted at all. Thus, active routers inform their parents about the union of subscriptions received from children. Routers should also inform their children when the aggregate subscription state changes, allowing downstream routers to suppress redundant subscription signaling. A protocol for installing active filters might be built on ATDP in the following way:

1. A subscription message Q arrives via $atdpRcv$. Parameters of interest are the group address g, the child p who sent the message, the local VIF v, and the direction parameter f. If $f = 0$ then this message was sent by a parent and no further signaling is required.

2. If Q has been sent by a child ($f = 1$), then the router updates a local filter database entry for (g, v) and determines if the aggregate subscription state has changed. If the aggregate state has *not* changed, no further signaling is required.

[6] Ingress filtering alone may allow unnecessary traffic one some links since a packet must arrive at a router with no downstream interest before it can be filtered. The scheme can be improved by adding *egress filtering*, which prevents packets from leaving a router on a VIF with no downstream interest.

3. In response to a change in the aggregate state, the router encodes the new state into a subscription message Q' and sends it to all upstream peers using $atdpSend(p, *, g, 1, exclude, Q')$. Note that if the peer p is also a parent to the local router, it need not be informed of the change in state.

4. The message Q' is also reflected to other children using $atdpSend(p, *, g, 0, exclude, Q')$.

8 Conclusion

We have presented ATDP, a protocol to facilitate signaling in network-assisted multicast protocols using a lightweight protocol to dynamically organize sparsely deployed active routers into a signaling overlay that tracks changes in the underlying multicast topology. ATDP exploits the local topological information acquired by each active node in this process to support efficient signaling among adjacent active nodes. Furthermore, ATDP provides practical communication primitives that can simplify the design of a wide range of higher-level multicast protocols.

The design presented here is preliminary and several interesting directions are available for further work. An important, but straightforward, extension is to allow higher-level protocols to piggy-back initialization data on the SPMs sent by ATDP. We would also like to replace TCP for reliable signaling with a UDP-based protocol. Along similar lines, it may be reasonable to support unreliable signaling between neighboring peers, perhaps making use of the forward multicast path in some cases. This extension might allow ATDP to provide additional communication services such as subcast. Finally, while we have established the usefulness of ATDP for several proposed active services, the search is ongoing for new protocols that can make use its features.

References

[1] Supratik Bhattacharyya, Don Towsley, and Jim Kurose. The loss path multiplicity problem for multicast congestion control. In *Proc. IEEE Infocom'99*, 1999.

[2] B. Braden and B. Lindell. A two-level architecture for internet signaling. Technical report, IETF Internet Draft, 2000.

[3] B. Cain and D. Towsley. Generic multicast transport services: Router support for multicast applications. In *Proc. Networking 2000*, May 2000.

[4] Brad Cain, Tony Speakman, and Don Towsley. Generic router assist (gra) building block motivation and architecture. Technical report, IETF, 2001. draft-ietf-rmt-gra-arch-02.txt.

[5] K. L. Calvert, J. Griffoen, B. Mullins, A. Sehgal, and S. Wen. Concast: Design and implementation of and active network service. *IEEE Journal on Selected Areas in Communications*, 19(3), 2001.

[6] T. Speakman et al. Pgm reliable transport protocol. Technical report, IETF, 2000.

[7] S. Kasera, S. Bhattacharyya, M. Keaton, D. Kiwior, S. Zabele, and D. Towsley. Scalable fair reliable multicast using active services. *IEEE Network Magazine*, 14(1), Jan-Feb 2000.

[8] Brian N. Levine and J. J. Garcia-Luna-Aceves. Improving internet multicast with routing labels. In *Proc. ICNP-97*, 1997.

[9] J. Lin and S. Paul. Rmtp: A reliable multicast transport protocol. In *Proc. IEEE Infocom'96*, 1996.

[10] Christos Papadopoulos and Emmanouil Laliotis. Incremental deployment of a router-assisted reliable multicast scheme. In *Proc. of Networked Group Communications (NGC2000)*, 2000.

[11] Christos Papadopoulos, Guru Parulkar, and George Varghese. Lms: A router assisted scheme for reliable multicast. *Submitted to IEEE/ACM Transactions on Networking*, 2001. http://pollux.usc.edu/ chrisp/papers/LMS-ToN.ps.

[12] Luigi Rizzo. Pgmcc: a tcp-friendly single-rate multicast congestion control scheme. In *Proc. SIGCOMM'00*, 2000.

[13] Li wei H. Lehman, Stephen J. Garland, and David L. Tennenhouse. Active reliable multicast. In *Proc. Infocom'98*, 1998.

[14] S. Wen, J. Griffoen, and K. L. Calvert. Building multicast services from unicast forwarding and ephemeral state. In *Proc. Openarch 2001*, 2001.

[15] Koichi Yano and Steven McCanne. The breadcrumb forwarding service: A synthesis of pgm and express to improve and simplify global ip multicast. *ACM Computer Communication Review*, 30(2), 2000.

[16] S. Zabele, B. Braden, M. Keaton, and B. Lindell. Active multicast information dissemination. Submitted for Publication.

[17] Stephen Zabele and Thomas Stanzione. Interest management using an active networks approach. In *Proceedings of the Simulation Interoperability Workshop*, March 2000.