     Multicast Support for Nimrod : Requirements and Solution Approaches

Status of this Memo

   This memo provides information for the Internet community. This memo
   does not specify an Internet standard of any kind. Distribution of
   this memo is unlimited.

Abstract

   Nimrod does not specify a particular solution for multicasting.
   Rather, Nimrod may use any of a number of emerging multicast
   techniques. We identify the requirements that Nimrod has of a
   solution for multicast support. We compare existing approaches for
   multicasting within an internetwork and discuss their advantages and
   disadvantages. Finally, as an example, we outline the mechanisms to
   support multicast in Nimrod using the scheme currently being
   developed within the IETF - namely, the Protocol Indpendent Multicast
   (PIM) protocol.

Table of Contents

1 Introduction

   The nature of emerging applications such as videoconferencing, remote
   classroom, etc. makes the support for multicasting essential for any
   future routing architecture. Multicasting is performed by using a
   multicast delivery tree whose leaves are the multicast destinations.

   Nimrod does not propose a solution for the multicasting problem.
   There are two chief reasons for this. First, multicasting is a non-
   trivial problem whose requirements are still not well understood.
   Second, a number of groups (for instance the IDMR working group of
   the IETF) are studying the problem by itself and it is not our
   intention to duplicate those efforts.

   This attitude towards multicasting is consistent with Nimrod's
   general philosophy of flexibility, adaptability and incremental
   change.

   While a multicasting solution per se is not part of the "core" Nimrod
   architecture, Nimrod does require that the solution have certain
   characteristics. It is the purpose of this document to discuss some
   of these requirements and evaluate approaches towards meeting them.

   This document is organized as follows. In section 2 we discuss why
   multicasting is treated a little differently than unicast despite the
   fact that the former is essentially a generalization of the latter.
   Following that, in section 4 we discuss current approaches toward
   multicasting . In section 5, we give an example of how Nimrod
   multicasting may be done using PIM [DEF+94a]. For readers who do not
   have the time to go through the entire document, a summary is given
   at the end.

   This document uses many terms and concepts from the Nimrod
   Architecture document [CCS96] and some terms and concepts (in section
   5) from the Nimrod Functionality document [RS96]. Much of the
   discussion assumes that you have read at least the Nimrod
   Architecture document [CCS96].

2 Multicast vs Unicast

   We begin by looking at the similarities and differences between
   unicast routing and multicast routing. Both unicast and multicast
   routing require two phases - route generation and packet forwarding.
   In the case of unicast routing, Nimrod specifies modes of packet
   forwarding; route generation itself is not specified but left to the
   particular routing agent. For multicasting, Nimrod leaves both route
   generation and packet forwarding mechanisms unspecified. To explain
   why, we first point out three aspects that make multicasting quite
   different from unicasting :

o Groups and group dynamism. In multicasting, the destinations are part
  of a group, whose membership is dynamic. This brings up the following
  issues :

   - An association between the multicast group and the EIDs and
      locators of the members comprising that group. This is especially
      relevant in the case of sender initiated multicasting and policy
      support.

   - A mechanism to accommodate new group members in the delivery in
      response to addition of members, and a mechanism to "prune" the
      delivery in response to departures.

o State creation. Most solutions to multicasting can essentially be
  viewed as creating state in routers for multicast packet forwarding.
  Based on who creates the state, multicasting solutions differ. In
  multicasting, we have several options for this - e.g., the sender, the
  receivers or the intermediate routers.

o Route generation. Even more so than in unicast routing, one can choose
  from a rich spectrum of heuristics with different tradeoffs between a
  number of parameters (such as cost and delay, algorithmic time
  complexity and optimality etc.). For instance, some heuristics produce
  a low-cost tree with high end-to-end delay and some produce trees that
  give the shortest path to each destination but with a higher cost.
  Heuristics for multicasting are a significant research area today, and
  we expect advances to result in sophisticated heuristics in the near
  future.

   Noting that there are various possible combinations of route
   generation, group dynamism handling and state creation for a solution
   and that each solution conceivably has applications for which it is
   the most suitable, we do not specify one particular approach to
   multicasting in Nimrod. Every implementation of Nimrod is free to
   use its own multicasting technique, as long as it meets the goals and
   requirements of Nimrod. However, for interoperability, it is

necessary that certain things are agreed upon - for instance, the
structure of the forwarding information database that they create (we
discuss this in more detail in section 4).

Thus, we do not discuss the details of any multicast solution here,
only its requirements in the context of Nimrod. Specifically, we
structure the discussion in the remainder of this document on the
following two themes :

   o What are the goals that we want to meet in providing multicasting in
     Nimrod, and what specific requirements do these goals imply for the
     multicast solution?

   o What are some of the approaches to multicasting being discussed
     currently, and how relevant are each of these approaches to Nimrod?

3 Goals and Requirements

   The chief goals of Nimrod multicasting and their implications on
   solution requirements are as follows:

1. Scalability. Nimrod multicasting must scale in terms of the size of
   the internetwork, the number of groups supported and the number of
   members per group. It must also support group dynamism efficiently.
   This has the following implications for the solution:

   o Routers not on the direct path to the multicast destinations should
     not be involved in state management. In a network with a large
     number of routers, a solution that does involve such routers is
     unlikely to scale.

   o It is likely that there will be a number of applications that have
     a few members per group (e.g., medical imaging) and a number of
     applications that have a large number of members per group (e.g.,
     news distribution). Nimrod multicasting should scale for both
     these situations. If no single mechanism adequately scales for
     both sparse and dense group memberships simultaneously, a
     combination of mechanisms should be considered.

   o In the face of group membership change, there must be a facility
     for incremental addition or deletion of "branches" in the
     multicast tree. Reconstructing the tree from scratch is not likely
     to scale.

     o It is likely that we will have some well-known groups (i.e., groups
       which are more or less permanent in existence) and some ephemeral
       groups. The dynamics of group membership are likely to be
       different for each class of groups, and the solution should take
       that into account as appropriate.

2. Policy support. This includes both quality of service (QOS) as
   well as access restrictions, although currently, demand is probably
   higher for QOS. In particular, every path from a source to each
   destination in the multicast group should satisfy the requested
   quality of service and conform to the access restrictions. The
   implications for the multicasting solution are :

   o It is likely that many multicasting applications will be cost
     conscious in addition to having strict quality of service bounds
     (such as delay and jitter). Balancing these will necessitate
     dealing with some new parameters - e.g., the tree cost (sum of the
     "cost" of each link), the tree delay (maximum, mean and variance
     in end-to-end delay) etc.

   o In order to support policy-based routing, we need to know where the
     destinations are (so that we can decide what route we can take to
     them). In such a case, a mechanism that provides an association
     between a group id and a set of destination locators is probably
     required.

   o Some policy constraints are likely to be destination specific. For
     instance, a domain might refuse transit service to traffic going to
     certain destination domains. This presents certain unique problems
     - in particular, for a single group, multiple trees may need to be
     built, each tree "servicing" disjoint partitions of the multicast
     destinations.

3. Resource sharing. Multicasting typically goes hand in hand with large
   traffic volume or applications with a high demand for resources.
   These, in turn, imply efficient resource management and sharing if
   possible. Therefore, it is important that we place an emphasis on
   interaction with resource reservation. For instance, Nimrod must be
   able to provide information on which tree resources are shareable and
   which are not so that resource reservation may use it while allocating
   resources to flows.

4. Interoperability. There are two issues in this context. First, the
   solution must be independent of mechanisms that provide the solution
   with information it needs. For instance, many multicast solutions
   (e.g., PIM) make use of information supplied by unicast routing
   protocols. The multicast solution must not be dependent on which
   unicast protocol is used.

Second, a multicast solution must interoperate with other multicast
solutions in the construction of a delivery tree. This implies some
kind of "agreement" at some "level". For instance, the agreement
could be that everybody use the same structure for storing forwarding
information in the routers. Since the delivery tree is defined by the
nature of forwarding information in the routers and not by the
particular mechanism used to create that information, multiple
implementations can coexist.

4 Approaches

The approaches to multicasting currently in operation and those being
considered by the IETF include the following :

1. Distance vector multicast routing protocol (DVMRP)[DC90]. This
   approach is based upon distance-vector routing information distribution
   and hop-by-hop forwarding. It uses Reverse Path Forwarding (RPF)[DM78]
   - a distributed algorithm for constructing an internetwork broadcast
   tree. DVMRP uses a modified RPF algorithm, essentially a truncated
   broadcast tree, to build a reverse shortest path sender-based multicast
   delivery tree. A reverse shortest path from s to d is a path that uses
   the same intermediate nodes as those in the shortest path from d to
   s (If the paths are symmetric (i.e., cost the same) in either
   direction, the reverse shortest path is same as the shortest path.)
   An implementation of RPF exists in the current Internet in what
   is commonly referred to as the MBONE. An improvement to this is in the
   process of being deployed. It incorporates "prune" messages to
   truncate further the routers not on the path to the destinations and
   "graft" messages to undo this truncation, if later necessary.

   The main advantage of this scheme is that it is simple. The major
   handicap is scalability. Two issues have been raised in this
   context[BFC93]. First, if S is the number of active sources and G
   the number of groups, then the state overhead is O(GS) and might be
   unacceptable when resources are limited. Second, routers not on a
   multicast tree are involved (in terms of sending/tracking prune and
   graft messages) even though they might not be interested in the
   particular source-group pair. The performance of this scheme is
   expected to be relatively poor for large networks with sparsely
   distributed group membership. Furthermore, no support for policies
   or QOS is provided.

2. Core Based Trees (CBT)[BFC93]. This scheme uses a single tree shared
   by all sources per group. This tree has a single router as the core
   (with additional routers for robustness) from which branches emanate.
   The chief distinguishing characteristic of CBT is that it is receiver
   initiated, i.e., receivers wishing to join a multicast group find the
   tree (or its core) and attach themselves to it, without any

   participation from the sources.

   The chief motivation behind this scheme is the reduction of the state
   overhead, to O(G), in comparison to DVMRP and PIM(described below).
   Also, only routers in the path between the core and the potential
   members are involved in the process. Core-based tree formation and
   packet flow are decoupled from underlying unicast routing.

   The main disadvantage is that packets no longer traverse the shortest
   path from the source to their destinations. The performance in
   general depends on judicious placement of cores and coordination
   between them. Traffic concentration on links incident to the core is
   another problem. There is also a dependence on network entities (in
   other administrative domains, for instance) for resource reservation
   and policy routing.

3. Protocol Independent Multicasting (PIM)[DEFJ93]. Yet another approach
   based on the receiver initiated philosophy, this is designed to reap
   the advantages of DVMRP and CBT. Using a "rendezvous point", a
   concept similar to the core discussed above, it allows for the
   simultaneous existence of shared and source-specific multicast trees.
   In the steady state, data can be delivered over the reverse shortest
   path from the sender to the receiver (for better end-to-end delay) or
   over the shared tree.

   Using two modes of operation, sparse and dense, this provides
   improved performance, both when the group membership in an
   internetwork is sparse and when it is dense. It is however, a
   complex protocol. A limitation of PIM is that the shortest paths are
   based on the reverse metrics and therefore truly "shortest" only when
   the links are symmetric.

4. Multicast Open Shortest Path First (MOSPF)[Moy92]. Unlike the
   abovementioned approaches, this is based on link-state routing
   information distribution. The packet forwarding mechanism is
   hop-by-hop. Since every router has complete topology information,
   every router computes the shortest path multicast tree from any
   source to any group using Dijkstra's algorithm. If the router
   doing the computation falls within the tree computed, it can
   determine which links it must forward copies onto.

MOSPF inherits advantages of OSPF and link-state distribution, namely localized route computation (and easy verification of loop-freedom), fast convergence to link-state changes etc. However, group membership information is sent throughout the network, including links that are not in the direct path to the multicast destinations. Thus, like DVMRP, this is most suitable for small internetworks, that is, as an intra-domain routing mechanism.

5. Inter-Domain Policy Routing (IDPR)[Ste]. This approach uses link-state routing information distribution like MOSPF, but uses source-specified packet forwarding. Using the link-state database, the source generates a policy multicast route to the destinations. Using this, the IDPR path-setup procedure sets up state in intermediate entities for packet duplication and forwarding. The state contains information about the next-hop entities for the multicast flow. When a data packet arrives, it is forwarded to each next hop entity obtained from the state.

Among the advantages of this approach are its ability to support policy based multicast routing with ease and independence (flexibility) in the choice of multicasting algorithm used at the source. IDPR also allows resource sharing over multiple multicast trees. The major disadvantage is that it makes it relatively more difficult to handle group membership changes (additions and deletions) since such changes must be first communicated to the source of the tree which will then add branches appropriately.

We now discuss the applicability of these approaches to Nimrod. Common to all of the approaches described is the fact that we need to set up state in the intermediate routers for multicast packet forwarding. The approaches differ mainly on who initiates the state creation - the sender (e.g., IDPR, PIM), the receiver (e.g., CBT, PIM) or the routers themselves create state without intitiation by the sender or receivers (e.g., DVMRP, MOSPF).

Nimrod should be able to accommodate both sender initiated as well as receiver initiated state creation for multicasting. In the remainder of this section, we discuss the pros and cons of these approaches for Nimrod.

Nimrod uses link-state routing information distribution (topology maps) and has four modes of packet forwarding - flow mode, Connectivity Specification Chain (CSC) mode, Connectivity Specification Sequence (CSS) mode and datagram mode [CCS96].

An approach similar to that used in IDPR is viable for multicasting using the flow mode. The source can set up state in intermediate routers which can then appropriately duplicate packets. For the CSC, BTES and datagram modes, an approach similar to the one used in MOSPF is applicable. In these situations, the advantages and disadvantages of these approaches in the context of Nimrod is similar to the advantages and disadvantages of IDPR and MOSPF respectively.

Sender based trees can be set up using an approach similar to IDPR and generalizing it to an "n" level hierarchy. A significant advantage of this approach is policy-based routing. The source knows about the policies of nodes that care to advertise them and can choose a route the way it wants (i.e., not depend upon other entities to choose the route, as in some schemes mentioned above). Another advantage is that each source can use the multicast route generation algorithm and packet forwarding scheme that best suits it, instead of being forced to use whatever is implemented elsewhere in the network. Further, this approach allows for incrementally deploying new multicast tree generation algorithms as research in that area progresses.

CBT-like methods may be used to set up receiver initiated trees. Nimrod provides link-state maps for generating routes and a CBT-like method is compatible with this. For instance, a receiver wishing to join a group may generate a (policy) route to the core for that group using its link-state map and attach itself to the tree.

A disadvantage of sender based methods in general seems to be the support of group dynamism. Specifically, if there is a change in the membership of the group, the particular database which contains the group-destination mapping must be updated. In comparison, receiver oriented approaches seem to be able to accommodate group dynamism more naturally.

Nimrod does not preclude the simultaneous existence of multiple approaches to multicasting and the possibility of switching from one to the other depending on the dynamics of group distributions. Interoperability is an issue - that is, the question of whether or not different implementations of Nimrod can participate in the same tree. However, as long as there is agreement in the structure of the state created (i.e., the states can be interpreted uniformly for packet forwarding), this should not be a problem. For instance, a receiver wishing to join a sender created tree might set up state on a path between itself and a router on the tree with the sender itself being unaware of it. Packets entering the router would now be additionally forwarded along this new "branch" to the new receiver.

In conclusion, the architecture of Nimrod can accommodate diverse
approaches to multicasting. Each approach has its disadvantages with
respect to the requirements mentioned in the previous section. The
architecture does not demand that one particular solution be used,
and indeed, we expect that a combination of approaches will be
employed and engineered in a manner most appropriate to the
requirements of the particular application or subscriber.

5 A Multicasting Scheme based on PIM

The Inter-Domain Multicast Routing (IDMR) working group of the IETF
has developed a specification for a new multicast scheme, namely,
Protocol Independent Multicasting (PIM) for use in the Internet
[DEF+94a, DEF+94b]. In this section, we decribe how the schemes
mentioned therein may be implemented using the facilities provided by
Nimrod.

We note that the path setup facility provided in Nimrod makes it very
conducive to PIM-style multicasting; despite the length of the
description given here, we assure the reader that it is quite simple
to implement PIM style multicasting in Nimrod.

Before reading this section, we recommend that the reader acquire
some familiarity with PIM (see [DEF+94a, DEF+94b]).

5.1 Overview

The PIM architecture maintains the traditional IP multicast service
model of receiver-initiated membership and is independent of any
specific unicast routing protocol (hence the name).

A significant aspect of PIM is that it provides mechanisms for
establishing two kinds of trees - a shared tree, which is intended
for low "cost" multicasting and a source-based tree, intended for low
delay multicasting.

A shared tree is rooted at a rendezvous point (RP), which is
typically a prespecified router for the multicast group in question.
In order to establish a shared tree, a designated router (DR) for a
host wishing to join a group G initiates a flow setup from the RP for
G to the DR. A source S wishing to send to a group G initiates a flow
setup between S and the RP for group G. At the conclusion of these
flow setups, packets can be forwarded from S to H through the RP. For
details on the protocol used to implement this flow setup please
refer to [DEF+94b].

After the shared tree has been setup, a recipient for group G has the
option of switching to a source-based shortest path tree. In such a
tree, packets are delivered from the source to each recipient along
the shortest path. To establish a source-based shortest path tree,
the DR for H looks at the source S of the packets it is receiving via
the shared tree and establishes a flow between S and the DR. The flow
is established along the shortest path from the DR to S (Thus,
strictly speaking, it is the reverse shortest path that is being
used.) Subsequently, packets can be forwarded from S to H using this
shortest path and thereby bypassing the RP. For details on the
protocol used to implement source-based trees in PIM please refer to
[DEF+94b].

When a host wishes to leave a multicast group, its designated router
sends a prune message towards the source (for source-based trees) or
towards the RP (for shared trees). For details on this and other
features of PIM please refer to [DEF+94b].

In Nimrod, PIM is implemented as follows (we refer to PIM based
multicast as Nimpim). In order to join a shared tree, an endpoint
(or an agent acting on behalf of the endpoint) wishing to join a
group G queries the association database for the EID and locator of
the RP for G (for well-known groups the association may be
configured). It is required that such an association be maintained
for every multicast group G. The endpoint gets a route for the RP and
initiates a multicast flow setup to the RP (a multicast flow setup is
similar to an unicast flow setup described in [CCS96] except for one
feature - when a multicast flow setup request reaches a node that
already has that flow present, the request is not forwarded further.
The new flow gets "spliced" in as a new branch of the existing
multicast tree). Similarly, the source establishes a flow to the RP.
The RP creates state to associate these two flows and now packets can
be forwarded to the endpoints from the source. Note that each flow
setup may be "hierarchical" and involve many subflows. All this,
however, is transparent to Nimpim. For details on management of
hierarchical flows please refer to [CCS96].

To create the source-based tree, the representative for a recipient
node N obtains the EID or locator of the source from the data packets
and initiates a multicast flow setup to the source. The route agent
for the node N uses its map in order to calculate the shortest path
from the source to N. The flow request is sent along the reverse of
this path. We note that the "shortness" of the path is constrained
by the amount of routing information available locally. However,
since the map is available locally, one can find the actual shortest
path from the source to N and not use the shortest path from N to S.
Thus, with Nimrod one can actually surmount a shortcoming of PIM with
relative ease.

We now discuss some more details of Nimpim. We start with a
description of multicast flow setup. This is the "basic"
functionality required to implement multicasting. Having this
"building-block" spelt out, we use this to specify the establishment
of the shared tree (in section 5.3) and the establishment of a
source-based tree (in section 5.4).

We only discuss sparse-mode multicasting, as described in [DEF+94a]
here. Further, to simplify the discussion, we assume a single
Rendezvous Point per group. Finally, we "address" all entities in
terms of their EIDs alone for reasons of conciseness - the locators
could be used in conjuction to reduce the overhead of database
lookups.

5.2 Joining and Leaving a Tree

Nimpim uses two control packets in order to setup a flow - the Nimrod
Multicast Flow-Request packet (NMFReq) and the Nimrod Multicast
Flow-Reply packet (NMFRep).

The NMFReq packet is a control packet identified by a prespecified
"payload type". The protocol-specific part of this packet includes
the following fields (except for the Code field, these fields are
present in the Unicast Flow-Request packet too) :

1. S-EID : The EID of the initiator of the flow.

2. T-EID : The EID of the target of the flow.

3. Flow-id : A label denoting the flow.

4. Direction : The direction of the flow - whether from the initiator
   to the target (FORW) or from the target to the initiator (REVERSE)
   or both (BOTH).

5. Code : Denotes whether the packet is for joining a flow
   (NMFReq-Join) for leaving a flow (NMFReq-Prune).

6. Source Route : A sequence of node locators through which the packet
   must travel.

The processing of the NMFReq by a forwarding agent at node N is
similar to that of the unicast flow request (see [CCS96]), except for
the fact that now we provide the ability for the new flow to "splice"
onto an existing delivery tree or "un-splice" from an existing
delivery tree. Specifically,

o If the Code is NMFReq-Join then the algorithm executed by the
  forwarding agent for node N is shown in Figure 1.

o If the Code is NMFReq-Prune then the algorithm is executed by the
  forwarding agent at node N is shown in Figure 2.

The NMFRep packet is used to accept or reject an NMFReq-Join or
NMFReq-Prune. The packet format is the same as that for unicast flow
request. However, an NMFRep packet is generated now by the first
node N that grafts the new flow to the existing tree. This may be
different from the target of the NMFReq.

It is required that a leaf router keep track of all hosts currently
joined to the group and send a prune message only if there is no host
in the local network for the group.

The NMFReq - NMFRep exchanges constitute a procedure for joining a
multicast delivery tree (when the Code is Join) and for leaving a
multicast delivery tree (when the Code is Prune). We term these
procedures Tree-Join and Tree-Leave respectively; we shall be using
these procedures as "building-blocks" in the construction of shared
trees (section 5.3) and of source-based trees (section 5.4).

```
begin
if the flow-id F in NMFReq-Join is in flow-list then
    if T-EID in NMFReq-Join = target in flow state for F then
        if Direction in NMFReq-Join is REVERSE or BOTH then
           Add the node preceding N in source route to child list for F
        else
           discard packet
    else
       discard packet
else
   begin
     install state for F in N, i.e.,
        assign parent(F) = node succeeding N in source route
        assign child(F)  = node preceeding N in source route
        assign target(F) = T-EID in NMFReq-Join
     forward NMFReq-Join to parent(F)
   end
end.
```

Figure 1: Algorithm executed by a forwarding agent for node N when
when it receives an NMFReq-Join.

```
begin
  if the flow-id F in NMFReq-Prune is in flow-list
  then begin
       delete previous hop in source route from child list for F, if exists
       if child list for F is empty
       then begin
             delete the flow-id and state associated with it
             forward to next hop in source route
           end
       else discard packet
       end
  else forward to next hop in source-route
end.
```

Figure 2: Algorithm executed by a forwarding agent for node N when it
receives an NMFReq-Prune.

5.2.1 An Example

   An example of how a tree is joined is given here with the help of
   Figure 3. In the figure, bold lines indicate an existing tree.
   Representative R on behalf of host H joins the tree by sending an
   NMFJoin-Req towards a target T. When used in the shared tree mode,
   the target is the RP and when used in the source tree mode, it is the
   source (root) of the multicast tree. Suppose that a host H wants to
   join the multicast tree. The following steps are executed :

Step 1. A representative R of H queries the route agent for a route
    from T to R. It obtains the route T - C- B - A - R. It builds a
    NMFJoin-Req packet with source route as R, A, B, C, T and flow
    as F forwards it to A.

Step 2. A looks for flow F in its installed flow database and
    doesn't find it. It installs state for F (makes R a child and
    B a parent in the multicast tree) and sends the NMFJoin-Req packet
    to B.

Step 3. B looks for flow F in its installed flow database and finds it.
    It adds B to its child list and constructs an NMFJoin-Rep packet and
    sends it to A.

Step 4. A forwards the packet to R and the tree joining is complete.
    Branch B-A-R is now added to the tree.

5.3 Establishing a Shared Tree

     There are two parts to establishing a shared tree - the receiver-to-
     RP communication wherein the receiver joins the delivery tree rooted
     at RP and the sender-to-RP communication wherein the RP joins the
     delivery tree rooted at the sender.

```
                                           T
                                         +---+
                                         |   |\
                                         +---+ \
                                          /     \
                                         /       \
                                    C   /         \ X
                                  +---+             +---+
                                  |   |             |   |
                                  +---+             +---+
                                      \
                                       \
                                        \
          R      join-req               join-req   \ B
        +---+ - - - - ->  +---+ - - - - -> +---+
        |   |<------------|   |<-----------|   |
        +---+  join-rep   +---+  join-rep  +---+
          |                 A                  \
          |                                     \
          |                                      \     Y
         ( )                                      +---+
          H                                       |   |
                                                  +---+
```
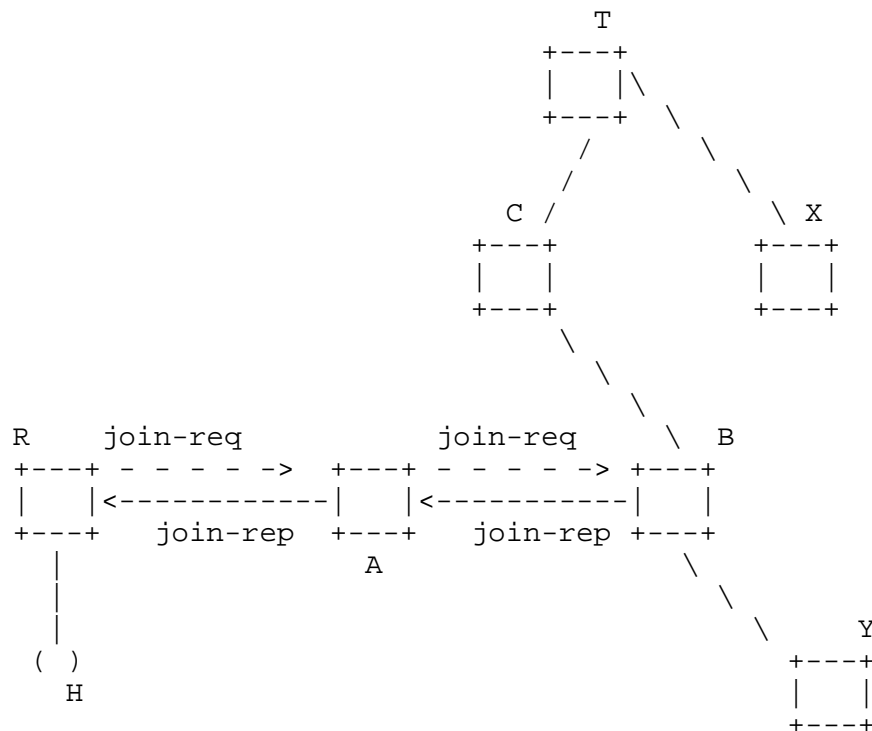
Figure 3: Illustration for the example describing joining an existing
multicast tree.

     Receiver-RP Communications: When an endpoint wishes to join a
     multicast group G, the endpoint representative obtains the Rendezvous
     Point EID for G. We assume that the association database contains
     such a mapping. For details on how the association database query is
     implemented, please refer [CCS96].

     The representative also obtains the flow-id to be used for the flow.
     The flow-id is constructed as the tuple (RP-EID, G) or an equivalent
     thereof. Note that the flow-id must be unique to the particular
     multicast flow. This is not the only method or perhaps even the best
     method for obtaining a flow id. Alternate methods for obtaining the
     flow-id are discussed in section 5.5.

The representative then initiates a Tree-Join procedure.

The NMFReq packet fields are as follows:

   o S-EID : The EID of the endpoint wishing to join.

   o T-EID : The RP EID (obtained from the Association Database).

   o Flow-id : The flow-id for this group (obtained as mentioned
     above).

   o Direction : REVERSE (from the RP to the receiving endpoint).

   o Code : Join.

   o Source Route : Reverse of the route obtained from the map agent
     for a query "from RP-EID to Receiver-EID".

At the first node already containing this Flow-id or the RP, an
NMFRep packet is generated. The S-EID, T-EID, Direction and Flow-id
fields are copied from the NMFReq packet and the Code is set to
Join-Accept or Join-Refuse as the case may be. The source route is
reversed from the NMFReq packet.

Sender-RP Communications: When an endpoint wishes to send to a
multicast group G, the endpoint representative obtains the Rendezvous
Point EID for G. We assume that the association database contains
such a mapping. For details on how the association database query is
implemented, please refer [CCS96].

The representative also obtains the flow-id to be used for the flow.
The flow-id is constructed as the tuple (Sender-EID, G) or an
equivalent thereof. Note that the flow-id must be unique to the
particular multicast flow. This is not the only method or perhaps
even the best method for obtaining a flow id. Alternate methods for
obtaining the flow-id are discussed in section 5.5.

The representative then sends a RP-Register Message to the RP. This
register message is equivalent to the PIM-Register described in
[DEF+94b]. The RP-Register message contains the group G and the
flow-id (obtained as discussed above) and the sender EID.

The RP then initiates a Tree-Join with the Sender EID as the target.
The NMFReq fields are as follows :

   o S-EID : RP-EID.

   o T-EID : Sender EID (copied from RP-Register Message).

    o Flow-id : The flow-id field from RP-Register Message.

    o Code : Join.

    o Direction : REVERSE.

    o Source Route : Reverse of the route obtained from map agent
      query "from Sender-EID to RP-EID".

   The NMFRep fields are obvious.

   Shared Tree Data Forwarding: Packets sent from the source for group G
   contain the Flow-id used by the sender(s) and receiver(s) for setting
   up the delivery tree. The packets from the sender are sent to the RP
   where they are multicast, using the state created for the flow, into
   the delivery tree rooted at the RP to all of the receivers that did a
   Tree-Join.

5.4 Switching to a Source-Rooted Shortest Path Tree

   There are two parts involved in switching to a Source-Rooted Shortest
   Path Tree - the receiver-source communications wherein the receiver
   joins a multicast delivery tree rooted at the source and the
   receiver-RP communications wherein the receiver leaves the shared
   tree.

   Receiver-Source Communications: An endpoint E that is receiving
   packets through the shared tree from source S has the option of
   switching to a delivery tree rooted at the source such that packets
   from S to E traverse the shortest path (using whatever metric).

   The endpoint representative of E obtains the flow-id to be used for
   the flow. The flow-id is constructed equivalently to the tuple
   (Source-EID, G). Note that the flow-id must be unique to the
   particular multicast flow. This is not the only method or perhaps
   even the best method for obtaining a flow id. Alternate methods for
   obtaining the flow-id are discussed in section 5.5.

   The representative for E initiates a Tree-Join toward S with NMFReq
   fields as follows:

   o S-EID : EID of the Endpoint E.

   o T-EID : EID of the source.

   o Flow-id : Flow id for the multicast (obtained as mentioned above).

   o Code : Join.

o Direction : REVERSE.

o Source Route : To obtain the route, the route agent is queried for
  a shortest path route (based on the chosen metric, typically, the
  delay) from the source to the endpoint. We note that the quality
  of the route is constrained by the amount of routing information
  available, directly or indirectly, to the route agent. The Source
  Route is the reverse of the route thus obtained.

A comment on the difference between the shortest-path trees obtained
using the RPF tree as in [DEF+94b, DC90] and the trees that are be
obtained here. When using the RPF scheme, the packets from the
source S to the endpoint E follow a path that is the shortest path
from E to S. This is the desired path if and only if the path is
symmetric in either direction. However, in the mechanism described
here for Nimrod, the packets do follow the "actual" shortest path
from S to E whether or not the path is symmetric.

The NMFRep fields are obvious.

Receiver-RP Communications: After the receiver has joined the
source-rooted tree, it can optionally disassociate itself from the
shared tree. This is done by initiating a Tree-Leave procedure.

The representative sends a NMFReq packet toward the RP with the
fields as follows.

o S-EID : The EID of the endpoint wishing to leave the shared tree.

o T-EID : The RP-EID.

o Flow-id : The flow-id it used to join the shared tree.

o Code : Prune.

o Direction : REVERSE.

o Source Route : Obtained as for the Tree-Join.

The prune packet is processed by the intermediate forwarding agents
as mentioned in section 5.2. When the receiver gets back the NMFRep
packet, the receiver has left the shared tree.

Source Tree Data Forwarding: Packets from the source contain the
flow-id that was used to join the source tree for a given multicast
group. Forwarding agents simply use the state created by the Tree-
Join procedure in order to duplicate and forward packets toward the
receivers.

5.5 Miscellaneous Issues

   Obtaining the Flow-Id: In the above scheme the flow-id for a
   particular multicast group G was obtained by combining the RP-EID and
   the group set-id (G-SID) (in case of shared tree) or by combining the
   Source-EID and the G-SID (in case of source-based tree). A
   disadvantage of this approach is that the bit-length of EID/SID is
   potentially high (more than 64 bits) and thus the flow-id could be
   very long. While there do exist bit-based data structures and search
   algorithms (such as Patricia Trees) that may be used for an efficient
   implementation, it is worth considering some other methods in lieu of
   using the EID/SID combination. We describe some methods below :

1. For shared trees, the flow-id for a particular group G may be stored
   and updated in the association database. Since we have to use the
   association database anyway to obtain the RP-EID, these does not cause
   much additional burden.

   However, this cannot be used efficiently for source-based trees because
   we need a flow-id for each combination of Source and Group.

2. The flow-id for shared trees could be done as above. When the sender
   does an RP-Register, it could send the RP the flow-id that it wishes to
   be used by receivers when they switch to a source-based tree. This
   could be included in the RP-Register message. The RP could then
   multicast that flow-id to all receivers in a special packet. When the
   receivers wish to switch, they use that flow-id.

   This needs the definition of the "special" packet.

3. The flow-id is handed out only by the source (for source-based trees)
   or the RP (for shared trees). The receivers use a "dummy" flow-id in
   the NMFReq when doing a Tree-Join. The correct flow-id to be used is
   returned in the NMFRep message generated by the forwarding agent where
   the new branch meets the existing tree. Forwarding agents in the path
   of the NMFRep packet update the state information by rewriting the
   dummy flow-id by the correct flow-id contained in the NMFRep packet.

   This requires the re-definition of the NMFRep packet. Note that now
   there must be space for two flow-ids in the NMFRep packet - one for the
   "dummy" flow-id and the other for the "correct" flow-id that must
   replace the dummy flow-id.

   We claim that each of the above schemes achieves synchronization in
   the flow-id in various parts of the internetwork and that each flow-
   id is unique to the multicast delivery tree. A formal proof of these
   claims is beyond the scope of this document.

Dense Mode Multicast: The PIM architecture [DEF+94a] includes a
multicast protocol when the group membership is densely distributed
within the internetwork. In this mode, no Rendezvous Points are used
and a source rooted tree is formed based on Reverse Path Forwarding
in a manner similar to that of DVMRP [DC90].

We do not give details of how Nimrod can implement Dense Mode
Multicast here.

Multiple RPs: Our discussion above has been based on the assumption
that there is one RP per group. PIM allows more than one RP per
group. We do not discuss multiple-RP PIM here.

6 Security Considerations

Security issues are not discussed in this memo.

7 Summary

o Nimrod does not specify a particular multicast route generation
  algorithm or state creation procedure. Nimrod can accommodate diverse
  multicast techniques and leaves the choice of the technique to the
  particular instantiation of Nimrod.

o A solution for multicasting within Nimrod should be capable of:

  - Scaling to large networks, large numbers of multicast groups and
    large multicast groups.

  - Supporting policy, including quality of service and access
    restrictions.

  - Resource sharing.

  - Interoperability with other solutions.

o Multicasting typically requires the setting up of state in intermediate
  routers for packet forwarding. The state setup may be initiated by the
  sender (e.g., IDPR), by the receiver (e.g., CBT), by both (e.g., PIM)
  or by neither. The architecture of Nimrod provides sufficient
  flexibility to accommodate any of these approaches.

o A receiver-initiated multicast protocol, PIM, is being designed by the
  IDMR working group of the IETF. The facilities provided by Nimrod make
  the use of PIM as a multicast protocol quite straightforward.

8 References

[BFC93]   A. J. Ballardie, P. F. Francis, and J. Crowcroft. Core based
            trees. In Proceedings of ACM SIGCOMM, 1993.

[CCS96]   Castineyra, I., Chiappa, N., and M. Steenstrup, "The Nimrod
            Routing Architecture", RFC 1992, August 1996.

[DC90]   S. Deering and D. Cheriton. Multicast routing in datagram
            internetworks and extended lans. ACM Transactions on Computer
            Systems, pages 85--111, May 1990.

[DEF+94a]   Deering, S., Estrin, D., Farinacci, D., Jacobson, V., Liu,
            C., and L. Wei, "Protocol Independent Multicast (PIM) :
            Motivation and Architecture, Work in Progress.

[DEF+94b]   Deering, S., Estrin, D., Farinacci, D., Jacobson, V., Liu,
            C., and L. Wei, "Protocol Independent Multicast (PIM) :
            Sparse Mode Protocol Specification, Work in Progress.

[DEFJ93]   Deering, S., Estrin, D., Farinacci, D., and V. Jacobson,
            "IGMP router extensions for routing to sparse multicast
            groups, Work in Progress.

[DM78]   Y. K. Dalal and R. M. Metcalfe. Reverse path forwarding of
            broadcast packets. Communications of the ACM, 21(12), pages
            1040--1048, 1978.

[Moy92]   Moy, J., "Multicast Extensions to OSPF, RFC 1584, March 1994.

[RS96]   Ramanathan, S., and M. Steenstrup, "Nimrod functional and
            protocol specifications, Work in Progress.

[Ste]   Steenstrup, M., "Inter-domain policy routing protocol
            specification: Version 2", Work in Progress.

9 Acknowledgements

   We thank Isidro Castineyra (BBN), Charles Lynn (BBN), Martha
   Steenstrup (BBN) and other members of the Nimrod Working Group for
   their comments and suggestions on this memo.

10 Author's Address

   Ram Ramanathan
   BBN Systems and Technologies
   10 Moulton Street
   Cambridge, MA 02138

   Phone: (617) 873-2736
   EMail: ramanath@bbn.com