Internet Draft
May 1995
Expires November 1995

M. Steenstrup
Nimrod Working Group

A Perspective on Nimrod Functionality

Status of this Memo

This document is an Internet Draft. Internet Drafts are working documents of the Internet Engineering Task Force (IETF), its Areas, and its Working Groups. Note that other groups may also distribute working documents as Internet Drafts.

Internet Drafts are draft documents valid for a maximum of six months. Internet Drafts may be updated, replaced, or obsoleted by other documents at any time. It is not appropriate to use Internet Drafts as reference material or to cite them other than as a "working draft" or "work in progress".

Please check the 1id-abstracts.txt listing contained in the internet-drafts directories on ftp.isi.edu, ds.internic.net, munnari.oz.au, nic.nordu.net, or ftp.is.co.za to learn the current status of any Internet Draft.

Distribution of this Internet Draft is unlimited. Please send all comments to nimrod-wg@bbn.com.

Abstract

We describe the Nimrod routing functionality, expressed in terms of the relationships among a set of distributed databases of routing information together with the procedures for constructing, accessing, and acting upon database contents.

Contents

1 Introduction

In this document, we present an overview of the routing functionality implied by the Nimrod routing architecture. Nimrod is a scalable routing architecture designed to support a dynamic internetwork of arbitrary size, to provide service-specific routing in the presence of multiple constraints, and to admit incremental deployment throughout an internetwork. The key features of Nimrod include representation of internetwork connectivity and services in the form of maps at multiple levels of abstraction; source- and destination-controlled route generation and selection based on maps and traffic service requirements; and source- and destination-controlled message forwarding according to the routes selected. A complete description of the Nimrod routing architecture appears in [1].

At the most general level, one may view any routing system as a set of basic functions which are producers and consumers of certain databases of routing information. These routing functions and their associated routing information include:

1. Assembling, distributing, and collecting information necessary for route generation and selection. This information includes internetwork connectivity and services, traffic service requirements, and locations of traffic sources and destinations.
2. Generating and selecting routes, based on the collected information.
3. Establishing in routers information necessary for forwarding messages, based on the selected routes.
4. Forwarding messages along these routes.

Routing systems may, however, differ in the details of the mechanisms that provide a particular routing function. As Nimrod has been designed for routing in large, heterogeneous, and dynamic internetworks, its basic routing functions include additional mechanisms for reducing the quantity of routing information that must be distributed, processed, and stored throughout an internetwork; for discovering and accommodating changes in routing information caused by physical changes in an internetwork; and for protecting the integrity of routing information.

The purpose of this document is to illustrate how the Nimrod architecture might be realized as a set of routing mechanisms and hence to provide a basis from which Nimrod protocols and algorithms can be developed. This document is not intended to be the definitive Nimrod functional specification. (Note that this version supersedes those previously distributed within the Nimrod working group.) The proposed routing mechanisms are based on Nimrod architectural requirements and engineering cost-benefit tradeoffs. In the remainder of this document, we describe each of these mechanisms in some detail, and we justify the selection of these particular mechanisms. We begin with a brief review of the Nimrod architecture, followed by a general discussion of Nimrod as a set of databases and functions, and then proceed to describe the individual Nimrod routing mechanisms.

2 Overview of the Nimrod Architecture

Before Nimrod routing can be applied within an internetwork, the internetwork must be represented in terms of the two basic Nimrod entities: *nodes* and *endpoints*. The internetwork's physical assets, such as routers, point-to-point links, and multiaccess networks, must be captured in Nimrod *maps* comprising interconnected nodes. Traffic sources and destinations must be cast as Nimrod endpoints. Nimrod entities possess attributes (e.g., location with respect to the maps, interconnectivity with other entities, service information) which are important for routing.

2.1 Clustering and Abstraction

Ideally, Nimrod maps should be constructed so as to satisfy the following two primary, and potentially conflicting, goals:

1. Minimize the amount of routing information maintained throughout an internetwork.
2. Maintain routing information sufficient to generate routes that meet traffic service requirements.

To satisfy these goals, Nimrod employs two complementary map construction procedures, namely *clustering* of internetwork physical assets into nodes and *abstraction* of attributes of the component physical assets resulting in node attributes.

The objective of clustering is to reduce the number of entities visible to Nimrod routing. Nodes are usually formed by clustering physical assets possessing similar attributes. These attribute similarities might be in terms of, for example, qualities of service, restrictions on access to services, or ownership. Such clustering results in a reduction in the amount of information necessary to characterize these physical assets, without a reduction in information detail. However, an internetwork's physical assets may be diverse enough so that clustering according to attribute similarity produces no significant reduction in the number of entities visible to Nimrod routing. In this case, alternative clustering criteria (e.g., geographical locality of physical assets) may be employed.

Clustering may be applied repeatedly, such that physical assets are first clustered into nodes, and then nodes are themselves clustered into larger nodes, and so on. Iterative clustering further reduces the number of entities visible to Nimrod routing and results in a hierarchical organization of nodes with a single top-level *universal* node containing all other entities. In the clustering hierarchy, the clustering criteria applied at different levels may not necessarily be the same.

The objective of abstraction is to reduce the amount of information required to characterize an entity visible to Nimrod routing. Nodes whose component physical assets possess different attributes rely on information abstraction in order to reduce the number of attributes used to characterize them. Abstraction procedures include, for example, eliminating attributes possessed by only a small percentage of the component physical assets or expressing attributes in terms of ranges of values exhibited by these physical assets. Multiple abstraction procedures may be applied to produce the attributes of a given node.

Nimrod does not mandate the choice of clustering and abstraction procedures to invoke in an internetwork. Rather, this choice is a local one under the control of the network managers for the various portions of the internetwork, and hence network managers may develop procedures that best suit their needs. We note that the specific clustering and abstraction procedures employed in an internetwork may have a significant effect on the quality of routes generated and on the cost of routing information maintenance. Hence, network managers should exercise care in selecting and using these procedures and may wish to experiment with several different ones before making a final decision. Although clustering and abstraction procedures may be fully automated, we recommend allowing manual intervention in order to enable network managers to make cost-benefit tradeoffs appropriate for their particular networks.

2.2 Nimrod Entities

All of the Nimrod routing functions are performed with respect to an internetwork's representation in terms of the basic Nimrod routing entities, namely nodes and endpoints. Each Nimrod entity has a set of attributes, each of which may be established through one or more of the following methods:

1. Manual configuration.
2. Automatic acquisition during initialization.
3. Active measurement.
4. Abstraction of attributes of component nodes.

Those attributes that must be configured are labelled as such.

A node is a set of contiguous internetwork physical assets. It may be formed by clustering physical assets directly or by clustering existing nodes. If the given node itself comprises component nodes, the routing system employed to route traffic within or across the node is Nimrod routing. Otherwise, this routing system may be any other. A node's attributes include its:

1. *Node identifier* (NID). An NID is a location-independent referent for a node. It is globally unique, relatively short, fixed-length (and likely lacking in internal structure) bit string used by Nimrod-capable devices to communicate their node's identity prior to acquisition of the node's locator. (Must be configured.)

2. *Locator*. A node's locator is a globally unique label describing the node's position in the clustering hierarchy. It consists of the global locator of the node's enclosing node in the clustering hierarchy, concatenated with a local locator, called an *element*, unique among all component nodes of the enclosing node.

3. Pool of locator elements that may be assigned to its component nodes and constraints on forming component relationships with nodes. (Must be configured.)

4. Constraints on forming *associations* with endpoints. An association is a relationship formed between a node and an endpoint, such that the endpoint acquires the node's locator. A node may be associated with multiple endpoints, and an endpoint may be associated with multiple nodes. (Must be configured.)

5. Constraints on forming *adjacencies* with other nodes. An adjacency is a neighbor relationship formed between two nodes that are physically joined. The neighbor relationship need not be symmetric. For example, nodes X and Y may agree to a relationship in which Y is adjacent to X, but X is not adjacent to Y. (Must be configured.)

6. Credentials of the node's manager, used in forming node adjacencies, assigning locators to component nodes, and forming endpoint associations. (Must be configured.)

7. Maps consisting of the node's current adjacencies and service offerings.

An endpoint is a traffic source, destination, or both that is visible to other Nimrod entities through association with one or more Nimrod nodes. Examples of endpoints include hosts and routers or even processes within hosts and routers. An endpoint's attributes include its:

1. *Endpoint identifier* (EID). An EID is a location-independent referent for an endpoint. It is globally unique, relatively short, fixed-length (and likely lacking in internal structure) bit string used by Nimrod-capable devices to communicate endpoint identity. (Must be configured.)

2. *Names.* A endpoint name is a globally unique, variable-length, structured, ASCII string used primarily by humans to refer to the endpoints (e.g., DNS names). (Must be configured.)

3. Constraints on forming associations with nodes. (Must be configured.)

4. Locators. An endpoint's locators are obtained from the nodes with which the endpoint is associated. Therefore, as an endpoint may be associated with more than one node, it may obtain more than one locator.

5. Credentials of the endpoint and its managers, used respectively in routing information authentication and in forming node associations. (Must be configured.)

6. Traffic service requirements from the perspectives of the endpoint as source and as destination.

3 Nimrod Routing Functions and Databases

At the core of Nimrod lies a set of distributed databases containing routing information that is constructed, accessed, and acted upon by the routing functions. These databases and their relationships to the routing functions are as follows:

1. *Node attributes.* Each node has a set of attributes (as discussed in the previous section) used in forming node adjacencies and endpoint associations, in constructing maps, and in assigning locators to component nodes.

2. *Endpoint attributes.* Each endpoint has a set of attributes (as discussed in the previous section) used in forming node associations and in selecting routes.

3. *Endpoint/locator associations.* Nimrod endpoint locators are used in generating routes between and in forwarding messages toward those endpoints. Endpoint/locator associations are stored and accessed based on globally unique labels (e.g., names) for the endpoints. We use the shortened term *association* database to refer to the Nimrod database of endpoint/locator associations stored within an internetwork.

4. *Maps.* Each Nimrod node has a set of maps describing its traffic service offerings and adjacencies to other nodes and used in generating routes.

5. *Routes.* Routes are generated in response to requests on behalf of traffic sessions between endpoints. Route generation works within the constraints of the service requirements specified by the endpoints and the services and adjacencies advertised in maps. Each route is expressed in terms of nodes and their corresponding services and is used to construct forwarding information to be installed in routers.

6. *Forwarding information.* Nimrod traffic forwarding is *path-oriented*, where a path is defined by the forwarding state stored in routers according to the route selected and is under the control of the source and destination endpoints.

Databases relevant to but not maintained by Nimrod are the pools of NIDs that may be assigned to nodes and EIDs and names that may be assigned to endpoints.

Each of the Nimrod routing functions uses portions of the contents of one or more of the Nimrod databases. In a dynamic internetwork, the procedures for updating and retrieving the contents of Nimrod databases will be performed frequently. Therefore, each Nimrod database should be organized to optimize the performance of these procedures along the dimensions of delay, internetwork resource consumption, fault tolerance, and load balancing.

Nimrod databases will be maintained by a combination of routers, hosts, and special-purpose physical devices. The use of special-purpose devices means that routers and hosts do not have to assume all of the processing and memory load related to routing. For example, as route generation is a computationally intensive procedure, some network managers may elect to use dedicated devices, distinct from routers, whose sole purpose is to generate Nimrod routes.

For most Nimrod databases, we suggest distributing database contents over several physical devices throughout an internetwork. In a large internetwork, one may in fact have no other choice; the memory required for a single Nimrod database (e.g., the association database) may exceed the storage capacity of any one device. Also, we suggest distributing database contents with partial redundancy, such that each database entry is stored in more than one device. Distributed organization of Nimrod database contents helps to reduce the database maintenance and query-response costs borne by any one physical device. Partial redundancy helps to increase the availability of database contents and to reduce the costs of the average database query-response; it may also increase the cost of the average database update, however.

3.1 Nimrod Database Consistency

Each Nimrod database contains routing information crucial for successful communication between endpoints. Inconsistencies between the actual state of the internetwork and the state as reflected by Nimrod database contents may result in impaired communication between a pair of endpoints and, in the worst case, may completely disrupt communication among all endpoints. Thus, minimizing the number as well as the consequences of such inconsistencies must be a primary objective of the Nimrod database maintenance procedures.

Inconsistencies between database contents and actual internetwork state may result from delays incurred in updating database contents following internetwork state changes. Many of the Nimrod databases are volatile and hence require mechanisms for keeping the contents current, in order to prevent propagation and use of stale routing information. Database maintenance includes rapid and reliable updating with new information as well as removal of old information. We recommend that each Nimrod database be maintained as a cache, such that each entry has a finite lifetime and may be removed from the database when it expires. Cache entry lifetimes will depend upon the expected duration of the usefulness of the cached information.

Inconsistencies between database contents and internetwork state may also result from errors introduced directly into database contents. Errors in Nimrod database contents may be injected inadvertently, through faults in the transmission media or in physical device memory, through misconfiguration, or through incorrect implementation of the database maintenance procedures. Errors may also be injected intentionally by malicious parties, through distribution of fictitious database updates and responses to queries (by capturing and corrupting existing database messages or by generating new messages) or through modification of database maintenance procedures.

Updating and retrieval of Nimrod database contents involve frequent communication of routing information over an internetwork and hence expose this routing information to numerous potential opportunities for error introduction. Therefore, the protocols that carry out these procedures should attempt to protect the routing information from introduced errors. In particular, the protocols for communicating information to or from a Nimrod database should permit the intended recipient of that information to:

1. Authenticate the information.

2. Detect corruption of the information.

3. Determine whether the information received is newer than any related information the recipient already possesses.

4. Indicate to the sender the receipt of acceptable or unacceptable information.

These protocols should also permit the sender to retransmit to the intended recipient any information that it preceives the recipient has failed to receive successfully.

We note that while Nimrod requires consistency between database contents and internetwork state, it does not require different physical devices to maintain identical views of internetwork state: for example, two different routers might maintain different maps for the same node, both of which are consistent with the physical assets of that node. Furthermore, Nimrod does not require consistency in route selection across different physical devices: for example, two different routers might select routes to the same destination such that each router is included in the other's route. The underlying path-oriented nature of message forwarding in Nimrod enables loop-free forwarding in the presence of such inconsistencies in route selection among routers.

3.2 Nimrod Agents

Within an internetwork, each Nimrod database is stored in a set of physical devices. Each physical device containing a portion of a Nimrod database executes a set of functionality, called a Nimrod *agent*, for manipulating the database contents. A single device may contain portions of more than one Nimrod database and hence may contain more than one Nimrod agent. Each Nimrod agent is a Nimrod endpoint and thus has an EID and a locator. Note that if we were to separate an agent from its database, the agent would have to know the locator for the database. We believe that such a separation introduces unnecessary complications, and so we recommend that agents and their databases be collocated.

For each Nimrod database, certain agents are responsible for maintaining specific portions. Such an agent is designated as an *authoritative source* for that portion of the database. A specific portion of a database may have multiple authoritative sources. Some types of databases are such that not all of their corresponding agents need to be authoritative sources. Usually, however, an agent is an authoritative source for some portion of a database but may also obtain and cache information learned from authoritative sources for other portions of that same database. In addition to receiving unsolicited database updates, a Nimrod agent may also refresh its database by querying other agents of the same type for their database contents.

Nimrod agents and databases are organized according to the clustering hierarchy, such that each node has a set of agents that act on its behalf to answer or propagate database queries. The node on whose behalf an agent acts is called the agent's *scope*, and a single agent may have multiple scopes. Moreover, an agent may be an authoritative source for a database portion that is independent of the agent's scope. Such is the case with association agents, as we discuss in section 4.7.

The Nimrod agents and their corresponding functions are as follows:

1. *Node representatives.* Each Nimrod node must have one or more representatives which maintain the database of the node's attributes and act on its behalf. A node representative is responsible for forming adjacencies with other nodes; assigning locators to component nodes; collecting maps from component nodes; and constructing its node's maps. Node representatives are authoritative sources for the maps of the nodes they represent.

2. *Endpoint representives.* Each Nimrod endpoint must have one or more representatives which maintain the database of the endpoint's attributes and act on its behalf. An endpoint representative is responsible for forming associations with nodes; discovering locators of the endpoints with which its endpoints wish to communicate; requesting routes to those endpoints and ensuring that the routes satisfy its endpoints' service requirements; initiating path establishment along the selected routes; and forwarding data along the established paths. Endpoint representatives are authoritative sources

for the locators and service requirements of the endpoints they represent.

3. *Association agents.* Each node may have one or more association agents responsible for answering or propagating association queries from endpoint representatives associated with the given node. Some association agents are authoritative sources for specific endpoint/locator associations while others may not be authoritative sources for any endpoint/locator associations.

4. *Route agents.* Each node may have one or more route agents responsible for collecting maps from nodes throughout the internetwork and for generating and dispensing routes based on endpoint service requirements and node connectivity and services advertised in maps. Route agents are authoritative sources for the routes they generate.

5. *Forwarding agents.* Each node must have one or more forwarding agents responsible for initiating neighbor relationships with forwarding agents in other nodes; requesting routes; installing forwarding information in routers; forwarding messages along established paths; and controlling traffic flow into and out of the node according to the node's access restrictions. While each of these functions could be performed by a different type of agent, we have elected to concentrate them in the forwarding agents, in order to minimize the number of different agent types performing the Nimrod routing functions. Forwarding agents are authoritative sources for the portion of the paths that traverse them.

Agents acting on behalf of a node need not reside within that node. Nevertheless, we suggest locating all Nimrod agents (and their databases) close to the entities on whose behalf they act. Such location minimizes delay and internetwork resource consumption when updating the databases corresponding to those entities and in responding to queries from other agents in the vicinity of those entities. A Nimrod agent residing external to the node on whose behalf it acts must be configured with location information for that node, and in some cases for ancestral and descendant nodes as well, in order communicate with other agents that act on behalf of the same node. We recommend placing all agents within the node on whose behalf they act, and henceforth we describe agent behavior from this perspective.

The Nimrod routing functions involve communication among different Nimrod agents, in order to update or to retrieve information from Nimrod databases. Usually, an agent requires information from a specific database but does not care which of that database's agents provides the information. Nimrod, however, does provide two different types of database queries, *direct* and *indirect*, which enable the querying agent to make its own tradeoffs in query cost versus quality of information obtained. A direct query must be answered by an authoritative source for the specified portion of the specified database. An indirect query may be answered by any agent for the specified database. Indirect queries are likely to consume fewer internetwork resources than direct queries, because they can often be answered by agents that are close to the querying agent. Indirect queries may yield stale information, however, because they are likely to be answered using cached information previously obtained from authoritative sources.

The type of routing information and hence its expected duration of usefulness, the cost to obtain the routing information, and the success rate of past queries will determine the type of query to place in a particular situation. For example, each agent may execute a database refresh procedure by querying another agent for its latest information concerning a particular database. An agent executes a database refresh following its activation or its detection of a discrepancy between internetwork state and its current database contents. The requesting agent may prefer to query an authoritative source for the information but may be willing to accept information from any agent. It is the responsibility of the network manager to set the rules for determining which type of query an agent, under the manager's jurisdiction, should place in a given circumstance.

Distributing and maintaining agent location information (EID, locator, and in some cases additional location information such as IP address) for inter-agent communication may be expensive when there are many agents in an internetwork. Hence, we provide an inter-agent message forwarding scheme that does not require the sending agent to specify a particular recipient agent but rather a description of where the message should be sent. In particular, the sending agent specifies:

1. Its EID and locator.
2. The type of message: update or query. If a query, the type: direct or indirect.
3. The type of database to be updated or queried.
4. The type of recipient agent.
5. The *key*, which in most cases is the NID or locator of the node for the updated or requested routing information. For association updates or queries, the key is the name of the endpoint for which information is updated or requested.

The above information is sufficient to direct a message toward an appropriate recipient agent, without having to specify the EID and locator of a particular recipient agent. Such inter-agent message forwarding may require the participation of many intervening agents, each of which knows how to to direct the message toward an appropriate ultimate recipient agent. Agents may also use EID, locator, and other location information as well as Nimrod paths when communicating, provided this information is available, in order to achieve more efficient inter-agent message forwarding.

4 Nimrod Entity Initialization and Operation

The Nimrod routing functions must work well in a large dynamic internetwork. Hence, these functions should include procedures that enable Nimrod agents to perform much of their own initialization, without requiring human intervention. An agent performs initialization procedures following its activation and may also perform some of these procedures following reconfiguration or movement. The initialization procedures include:

1. Neighbor discovery.
2. Agent discovery.
3. Locator acquisition for nodes and endpoints.
4. Node map construction.
5. Association database maintenance.

Although Nimrod provides automatic initialization procedures, network managers retain ultimate control over Nimrod entities and agents, and thus can organize their networks in order to achieve specific routing behavior. Network managers cluster physical assets into Nimrod nodes, determine the abstraction algorithms used in map construction, and load Nimrod agent functionality into physical devices. Moreover, network managers may configure each of the attributes of any Nimrod entities under their jurisdiction. In the remainder of this section, we discuss initialization and subsequent operating procedures for Nimrod nodes, endpoints, and agents.

4.1 Node Formation

Node formation begins when a network manager clusters a contiguous set of physical assets or existing nodes into a new node. Clustering defines the node boundary, indicating which physical devices belong to the node and which are external to the node. In particular, clustering identifies the node's boundary routers, namely those that have direct physical connectivity to routers external to the node. Physical connections that extend to external routers may be considered part of one node or shared by both nodes. For example, a network manager may wish to define the node boundary to include the outbound portion of an external physical connection but not the inbound portion of that same connection.

A Nimrod node must have boundary forwarding agents at the entry and exit points of the node for forwarding Nimrod traffic into and out of the node; at least one node representative for forming adjacencies with other nodes and for generating maps; and at least one endpoint representative for internal hosts that want to take advantage of Nimrod forwarding. The node might also have its own route and association agents, but these are not mandatory. After defining the node boundary, the network manager may have to install agents for the node. If the node has been formed by clustering existing nodes, the network manager may not need to install any new agents. Instead, it may only need to add a scope to existing agents.

For each new agent it installs in a node, the network manager must assign the agent a type, an EID, and a label for each of its scopes. Each assigned scope label is an NID that uniquely identifies a node and acts as a temporary node locator before the node acquires its actual locator. A method for assignment of NIDs that does not require management of a separate label space is to assign the node's NID to be the EID of one of a node's representatives.

As we have previously noted, forwarding agents must reside in physical devices within the node. Each of the node's boundary routers must contain a forwarding agent, designated as a *boundary* forwarding agent for that node, which defines the boundary of the node and controls traffic flow across that boundary. Forwarding agents contained within other routers in the node are designated as *interior* forwarding agents for that node. If the node has no component nodes, then it need only have boundary forwarding agents. If the node comprises component nodes, then it may have interior as well as boundary forwarding agents. Its interior forwarding agents include those component nodes' boundary forwarding agents that are not boundary forwarding agents for the specified node.

Each node must have at least one node representative. We currently restrict all of a node's representatives to reside within the node, thus simplifying the automatic initialization procedures. The network manager must assign the following node attributes to be maintained by the node's representatives:

1. Constraints defining the type of nodes with which the node may form an adjacency. For example, the network manager may specify that a node should only become a neighbor of nodes managed by particular providers.
2. Constraints defining the type of endpoints with which the node may form an association.
3. The pool of locator elements that may be assigned to component nodes and constraints defining the type of nodes that may become components of the node.

In addition, the network manager may also configure the node's maps, including adjacencies and service attributes. This service information may include not only services in terms of quality and access restrictions, but also service-related properties such as the credentials of the node's manager and the types of nodes to which the services should be advertised. *Transit* services apply across the node between entry and exit points. *Inbound* services apply from entry points to interior points. *Outbound* services apply from interior points to exit points. Service quality includes but is not limited to offered delay, throughput, reliability, ordering, and security. Access restrictions include but are not limited to who may access the service, for what purposes, when, and for what fee. All of the node's representatives apply the same abstraction procedures for determining their node's service attributes, as described in section 4.6.2.

Adjacency constraints, association constraints, and offered services must be consistent among all of the node's representatives to ensure consistency in neighbor and endpoint acquisition and in map formation. The component node locator element pool, however, is different for each of the node's representatives. Specifically, each node representative maintains a block of available locator elements for component nodes, which is disjoint from the blocks maintained by the other representatives. Thus, different node representatives cannot assign the same locator element to different component nodes. Moreover, different node representatives do not normally assign different locator elements to the same component node, except in the presence of node partitions (see section 4.6.5). The reason is that at any one time, a single representative of a non-partitioned node is responsible for acquiring its node's locator (as described in section 4.5.1).

4.2 Endpoint Configuration

Endpoint definition begins when a network manager assigns an EID and an endpoint representative to a new endpoint. An endpoint's representative may reside within or outside of the endpoint it represents. If there are no endpoint representatives already in the vicinity of the new endpoint, the network manager installs such an endpoint representative, assigning it a set of endpoints on whose behalf it will act and an EID. If there is already an endpoint representative in the vicinity of the new endpoint, the network manager need not install a new one. Instead, the network manager may add the new endpoint to the existing endpoint representative's set of represented endpoints. An endpoint representative represents itself as an endpoint. Also, all other types of agents contain endpoint representative functionality as well but are not likely to represent endpoints other than themselves.

Each endpoint must have at least one endpoint representative, and a single endpoint representative may act on behalf of multiple endpoints. Endpoint representatives form associations with nodes on behalf of their endpoints, as we describe in section 4.5.2, and they themselves become associated with these same nodes. The network manager must assign the following endpoint attributes to be maintained by the endpoint's representatives:

1. An EID.
2. A set of endpoint names. (DNS names are one such example.)
3. Constraints on the types of nodes with which the endpoint may form an association. For example, an endpoint owned by a given company may only be permitted to connect to physical assets owned

by that company.

4. The set of multicast groups of which the endpoint is automatically a member. (Nimrod multicast communication is discussed further in [3].)

For all endpoints represented by external endpoint representatives, there is currently no standard protocol for communicating an endpoint's traffic service requirements to other locations, such as an external endpoint representative. Support of such a facility in Nimrod would require participation by hosts, but we do not want to mandate any host changes to support Nimrod initially. Therefore, for an external endpoint representative, the network manager must configure the endpoint's requested services, from the perspective of traffic source and traffic destination. This service information may include not only service requirements in terms of quality and access restrictions, but also service-related properties such as the credentials of the endpoint's managers and the classification (e.g., research, commercial) of its traffic. Service quality includes but is not limited to requested delay, throughput, reliability, ordering, and security. Access restrictions include but are not limited to the names of service providers to prefer or to avoid and the amount the endpoint is willing to pay for the service.

The two endpoint label databases, EIDs and names, are not maintained by Nimrod. Nevertheless, we provide an example of the organization of these databases. This example is based on a hierarchical management organization which culminates with a central organization such as the Internet Assigned Numbers Authority (IANA). These management entities would be responsible for maintaining the databases of endpoint labels and for assigning endpoint labels from these databases. At the top, the IANA would assign blocks of endpoint labels to network managers. Each network manager would assign blocks of these labels to hosts under its jurisdiction, and each host and router would assign labels to endpoints resident within it. There may also be more levels of management entities between the IANA and the network managers. Additional facilities might include the following. Whenever its local supply of available endpoint labels was exhausted, a network manager could request a new block of labels from the IANA Moreover, it might reclaim endpoint labels that were no longer in use.

4.3 Neighbor Discovery

Once all of its node's boundary forwarding agents have been installed, the network manager activates all installed agents for that node. In addition to manual activation, agents are also automatically activated whenever the physical devices in which they reside become operational.

Neighbor discovery is the first function performed by activated Nimrod agents within a node. Agents acting on behalf of a node but not resident within the node do not participate in neighbor discovery. Agents resident in Nimrod nodes that do not execute Nimrod internally need not perform neighbor discovery but instead may proceed directly to agent discovery.

Neighbor discovery is the initial step in the formation of adjacencies between nodes (see section 4.6.1) and the initial step in the formation of associations between endpoints and nodes (see section 4.5.2). It enables agents to determine whether there are other Nimrod agents reachable within one "hop" and if so which agents. A hop may constitute a direct link (point-to-point or multiaccess) between two agents, or it may include many intervening links and routers.

When the hop is not a direct link, the agent must be configured with location information (e.g., IP address) for the agent at the other end of the hop, so that the neighbor discovery messages reach the appropriate agent. Even when the hop is a direct link, the agent may require some location information configuration or discovery prior to Nimrod neighbor discovery. Specifically, an agent must know the type of link attached to each of the interfaces of the device in which it resides, so that it can invoke the appropriate low-level protocols. For example, if an agent resides in a device attached to an Ethernet that also uses IP, that agent may use a combination of ARP, dynamic host configuration, and router discovery to determine its IP address and the IP address of a neighboring router. These functions must be executed prior to Nimrod neighbor discovery.

A Nimrod agent periodically issues a neighbor discovery message over each of its external interfaces. These neighbor discovery messages contain the issuing agent's type, EID, locator (when available), set of scope labels expressed as NIDs, locators for those scopes (when available), and in some circumstances additional location information (e.g., IP address). When a Nimrod agent X receives a neighbor discovery message, it knows that it has a neighboring agent Y. From then on, X expects to continue to receive periodic neighbor discovery messages from Y. Failure to receive neighbor discovery messages from a known neighbor indicates a reachability problem, caused by a physical component's malfunction or movement.

4.4 Agent Discovery

Agent discovery is the second function performed by activated Nimrod agents within a node. It enables agents with the same scope to learn how to reach each other and hence to communicate before Nimrod inter-node routing is fully operational. Agents acting on behalf of a node but not resident within the node do not participate in agent discovery. The location information required by and for these agents must be preconfigured. Agents resident within a node may be configured with location information pertaining to other agents in that node, or they may participate in agent discovery.

Agent discovery enables agents with a particular scope to determine autonomously the set of *key* agents for the specified node and how to reach them, independently of Nimrod inter-node routing. Key agents include boundary forwarding agents, node representatives, endpoint representatives, and if present, route agents and association agents. The advertisements of key agents with the same scope are distributed such that all agents with the same scope receive the advertisements. Each key agent generates an advertisement message containing the issuing agent's type, EID, locator (when available), set of scopes expressed as NIDs, locators corresponding to those scopes (when available), and in some circumstances additional location information (e.g., IP address). Note that this information is identical to the information distributed in neighbor discovery messages. Interior forwarding agents do not generate agent advertisements. Boundary forwarding agents also include in their advertisements the scopes (in terms of NIDs and, when available, locators) of neighboring exterior boundary forwarding agents, learned through neighbor discovery. Node representatives use this neighbor information to determine whether to form adjacencies with other nodes.

In a Nimrod node that does not execute Nimrod routing internally, the agent discovery procedure should be tailored to the information distribution mechanisms native to that node. Consider the following examples. Broadcasting agent advertisements is the method of choice, if the node is a multiaccess network. Flooding agent advertisements may be appropriate, provided the node supports flooding. Multicasting agent advertisements may be the most efficient strategy, if the node supports multicast distribution. In this case, all agents in the node are configured with a multicast address specific to those agents, which they use when distributing agent advertisements. Moreover, specific types of agents may also have their own multicast addresses (e.g., all boundary forwarding agents for a node may share a multicast address).

4.4.1 Flooding Agent Advertisements

In a Nimrod node that executes Nimrod internally, the agent discovery procedure is a hop-by-hop reliable flooding procedure (according to the reliability criteria described in section 3.1) among agents within a node. Each of the key agents for the node periodically generates an advertisement and distributes it to each of its neighboring agents that shares in common at least one of its scopes. Periodic distribution ensures that all agents with the same scope keep current with respect to reachability of key agents.

Upon receipt of an advertisement, an agent determines whether any of the scopes listed in the advertisement coincide with at least one of its scopes. If so, the agent then determines whether the advertisement is more recent than any thus far received from the issuing agent. If the advertisement is new, the agent accepts the advertisement and stores the advertised agent information locally. An agent discards an advertisement if none of the scopes coincide or if the advertisement is old. Interior forwarding agents for a node are the only type of agents that redistribute advertisements received from neighboring agents. When an interior forwarding agent receives a new advertisement, it distributes the advertisement to all neighboring agents sharing at least one of the scopes in the advertisement, except the neighboring agent from which the advertisement was received.

For each acceptable advertisement it receives, an agent records the neighbor from which the advertisement arrived, thus establishing forwarding information indicating how to reach the agent in the advertisement. Note that this procedure is similar to distance-vector routing, with the exception that the routes thus learned are the first and not necessarily the shortest. With this procedure, no persistent routing loops may form. The reasons are as follows: each agent knows of at most one way to reach a particular key agent, and no agent distributes an advertisement to the neighbor from which it was received. Temporary reachability problems may occur, however, if an agent or a link between agents fails.

The following procedure ensures that agents with the same scope quickly learn when a key agent becomes unreachable. If an agent Q loses connectivity with a neighboring agent R, all key agents learned through R become unreachable from the perspective of Q. In this case, Q floods a message indicating the set of agents that are now unreachable from its perspective. Each agent X receiving this unreachability message compares the list of unreachable agents contained in the message and with the list of agents supposedly reachable through the neighbor from which the message was received. If no agent is present on both lists, then X reaches the key agents through other neighbors and so X may safely discard the unreachability message. If at least one agent is present on both lists, then X can no longer reach those key agents. In this case, X constructs an unreachability message containing all such agents and floods it to all neighbors except the one from which the original unreachability message was received. X also removes its forwarding information pertaining to the unreachable agents.

With the above procedure, all agents affected by a connectivity problem between agents learn which key agents they can no longer reach. The periodic distribution of agent discovery messages ensures that agents will learn of alternate routes to currently unreachable agents, if such routes exist. Moreover, by installing multiple key agents of each type throughout a node, a network manager may minimize the probability that a particular agent is unable to reach any key agents of a particular type with the same scope.

4.4.2 Communication without Locators

Nimrod agents need to be able to communicate, whether or not they have acquired locators. When Nimrod nodes are initializing, inter-agent communication may traverse several nodes lacking locators. Hence, inter-agent query messages always record the NIDs of all nodes that forward the message. This sequence of collected NIDs can be reversed by the destination agent to form a route back to the source, so that the destination agent can return its response. When nodes possess locators, inter-agent communication may use Nimrod inter-node routing, such that inter-agent messages are sent over Nimrod paths.

4.5 Locator Acquisition

All Nimrod nodes and endpoints require locators for Nimrod inter-node routing. These locators are expressed with respect to a particular node clustering hierarchy. Each node has exactly one locator, whereas an endpoint may be associated with several locators. Entity locators are assigned during entity initialization following activation, reconfiguration, or movement. Although they may be preconfigured as part of the entity attributes, locators are usually acquired by exchanging information with other agents. In an internetwork in which physical assets are not stationary, automatic locator acquisition is desirable as it enables a Nimrod entity to obtain a locator without human intervention.

The locator acquisition procedure depends upon knowledge of the existence and scope of certain agents, and hence agent discovery must precede locator acquisition. When acquiring a locator for an entity, an entity representative always attempts to obtain a locator at the lowest possible level in the clustering hierarchy, in order to minimize the number of new Nimrod entities visible at higher levels in the hierarchy.

4.5.1 Node Locators

A single node representative should be responsible for acquiring a locator for its node, so that the node does not acquire multiple locators. Any method for determining this node representative should permit node representatives to make the same decision without pairwise communication. One such example is to choose the the single representative to be the one with the lowest-numbered EID of all mutually reachable representatives for that node. Node representatives can use agent discovery to learn which has the lowest-numbered EID.

Node representatives may act independently to obtain a locator for their node, if network connectivity problems cause them to become mutually unreachable. In this case, each node representative that believes it has the lowest-numbered EID of all mutually reachable node representatives attempts to acquire a locator for its node. Thus, multiple locators may be acquired for the same node, because multiple node representatives may attempt locator acquisition concurrently.

Each node representative that acquires a locator for its node distributes this locator in its periodically-distributed agent advertisements. Once the connectivity problems disappear, all representatives for the same node learn of the multiple node locators through agent advertisements. Upon discovering the existence of multiple locators for the same node, the node representatives independently choose a single node locator, for example, the one with the smallest numerical value (when the locator is viewed as a bit string). Those node representatives that have acquired redundant node locators must return these locators to representatives of the node from which the locators were acquired.

To acquire a locator on behalf of its node, a node representative begins by determining the node from which the node's locator should be acquired. Usually, the network manager configures the NID of this node within each node representative. To accommodate mobile nodes which may need to acquire new locators as they move, Nimrod should also provide a method for a node representative to select the node from which its node's locator should be acquired. An example of such a method is for the node representative to choose the node as the one whose locator is the longest prefix common to all external boundary forwarding agents (learned during agent discovery).

The following locator acquisition procedure is described from the perspective that the node representative from which the locator should be acquired is not collocated with the node representative that is attempting to acquire a locator. First, the node representative sends a locator request message (containing the NID and, if available, the locator of the node from which the locator should be acquired) to a boundary forwarding agent for its node, which in turn relays the message to an external boundary forwarding agent. Upon receipt of a node locator request, the external boundary forwarding agent relays the message toward a representative for the specified node. The message may pass through several nodes before reaching an appropriate node representative. Once the representative for the specified node receives the locator request message, it determines, according to its node's component constraints, whether to honor the request. Hence, the locator request message must contain information that enables the recipient node representative to evaluate the request.

The recipient node representative sends the requesting node representative a locator response message. If it elects to honor the request, the recipient node representative responds with a node locator, unique among the locators of its node's component nodes. Upon receiving the response, the requesting node representative must then determine, according to its node's component constraints, whether to accept the new node locator. Hence, the locator response message should contain information that enables the requesting node representative to evaluate the response. Note that the requesting node representative indirectly acts on behalf of all nodes contained within it in deciding whether to accept or reject the new node locator.

Once a node representative acquires a new locator for its node, it notifies all agents who share its scope to update their locators accordingly. Specifically, the node representative floods a special advertisement, called a locator update, that contains the node's new locator. Upon receiving a locator update, each agent for the node updates its locator. Each interior forwarding agent receiving a locator update then acts on behalf of the node for which it is a boundary forwarding agent, to forward the locator update toward a node representative within that node, and so on. Thus, when a node representative acquires a new locator for its node, the notification of the locator change trickles down to all lower-level nodes contained within the specified node. External boundary forwarding agents learn of the new node locator through the periodic exchange of neighbor discovery messages.

Thus, all affected endpoints and all adjacent nodes learn of the new locator. Endpoint representatives use the new locator information to determine whether they should associate their endpoints with the new node. External boundary forwarding agents notify their node representatives which in turn use the new locator information as a signal to determine whether to reconstruct their node's map to include an adjacency with the new node.

4.5.2 Endpoint Locators

An endpoint representative attempts to acquire a set of locators for each of its endpoints. It begins by determining the nodes from which an endpoint's locators should be acquired. If these are not configured, the endpoint representative uses the locator of each neighboring forwarding agents (obtained during agent discovery) to indicate the nodes with which to attempt associations.

For each node selected and for each endpoint that has not yet been associated with a set of locators, the endpoint representative executes the following procedure. The endpoint representative begins by sending a locator request message (containing the NID and, if available, the locator of the selected node) to a forwarding agent which in turn relays the message toward a representative for the specified node. The recipient node representative then determines, according to its node's association requirements, whether to honor the request. Hence, the locator request message must contain information that enables the recipient node representative to evaluate the request.

The recipient node representative informs the endpoint representative of its decision about the locator request. If it elects to honor the request, the recipient node representative responds with its node's locator. Upon receiving the response, the endpoint representative must then determine, according to its association requirements, whether to accept the new node association. Hence, the locator response message should contain information that enables the endpoint representative to evaluate the response. Once an endpoint representative acquires a new node association, it updates the endpoint/locator associations for the affected endpoints (as described in section 4.7.1).

4.6 Map Construction

Maps represent the interconnectivity of and the services provided by the individual physical assets of an internetwork, with respect to a particular clustering hierarchy. Each node has a *basic map* and an *abstract map* derived from the basic map. A node's basic map depicts the component nodes and their adjacencies to each other, to the enclosing node, and to external nodes. It also describes the transit, inbound, and outbound services provided by each of these component nodes. Note that the basic map is empty for nodes with no visible component nodes.

A node's abstract map depicts the node and its adjacencies to external external nodes. It also describes the transit, inbound, and outbound services provided by the node. (We provide examples of how to represent this service information concisely in section 4.6.2.) An abstract map represents the node's services and connectivity to external nodes that the node's manager elects to advertise externally. It does not necessarily represent the actual services or external connectivity provided by the node's physical assets.

Each node representative is an authoritative source for its node's basic and abstract maps and distributes these maps on behalf of its node. A node representative constructs its node's basic map by assembling the abstract maps distributed to it by the component nodes. A node representative constructs its node's abstract map using information obtained from one or more of the following sources:

1. Abstraction of basic map information.
2. Configuration of node service attributes.
3. Measurement of qualities of service (e.g., delay, throughput, or reliability) across the node and to adjacent nodes).

The procedures used to create abstract maps need not be the same for every node. Here is an example of how a node representative might derive the service attributes for its node's abstract map from its node's basic map. Using information provided in the basic map, the node representative first generates a set (not necessarily exhaustive) of routes between each node entry point and exit point. The node representative knows, through the adjacency formation procedure described in section 4.6.1 below, which inbound and outbound adjacencies for component nodes compose which inbound and outbound adjacencies for its node. For each route generated, the node representative determines the service attributes, based on those advertised in the abstract maps of the component nodes. The node representative then decides how to abstract the service attributes of these routes for its node's abstract map.

4.6.1 Adjacency Formation

A single node representative should be responsible for forming adjacencies between its node and external nodes. An external node may be external to the enclosing node, or it may be the enclosing node itself (in the case in which the enclosing node has a forwarding agent that is not contained within a component node). As with locator acquisition, we recommend the node representative with the lowest-numbered EID among those mutually reachable node representatives as the one to initiate adjacency formation. At the lowest level, adjacencies are formed by clustering physical connections from the given node to external nodes. Adjacencies may also be formed, using a node's basic map, by clustering adjacencies connecting component nodes to external nodes.

When forming adjacencies by clustering physical connections, a node representative determines which links will compose which adjacencies. First, the node representative obtains, through agent discovery, candidate component physical connections. These are physical connections from boundary forwarding agents for this node to boundary forwarding agents of external nodes. Then, the node representative groups these physical connections according to which of the external boundary forwarding agents share common scopes. This information defines the nodes with which adjacency formation will be attempted.

When forming adjacencies by clustering existing adjacencies, a node representative determines which lower-level adjacencies will compose which higher-level adjacencies. First, the node representative obtains, from the node's basic map, candidate external adjacencies. These are adjacencies between the node's component nodes and external nodes. Then, the node representative groups these adjacencies according to which of their adjacent nodes are components of the same enclosing node. This information defines the nodes with which adjacency formation will be attempted.

For each node and candidate adjacency determined through the grouping procedures described above, the node representative executes the following adjacency formation procedure. The node representative begins by sending an adjacency request message to a representative for the specified node. Each adjacency request message must contain information that enables the recipient node representative to determine, according to its adjacency requirements, whether to honor the request.

The recipient node representative informs the requesting node representative of its decision about the adjacency formation request. Upon receiving a positive response to an adjacency formation request, the requesting node representative must then determine, according to its adjacency requirements, whether to accept the new adjacency. Hence, the adjacency response message should contain information that enables the requesting node representative to evaluate the response.

Once a node representative definitely decides to form a new adjacency, it informs all other node representatives with the same scope, by distributing to them an adjacency update containing the NID and locator for the adjacent node, the set of lower-level adjacencies or physical links included in the adjacency, and the set of boundary forwarding agents in its node included in the adjacency. Thus, all of the node's representatives learn of the existence of the new adjacency. (Note that as all representatives for the node use the same clustering and abstraction algorithms, each could also independently and consistently determine which physical links or lower-level adjacencies and which forwarding agents were included in the new adjacency.) The node representative also sends the adjacency update to each boundary forwarding agent for its node. Boundary forwarding agents use this information when setting up paths through the node to an adjacent node, as described in section 6.

4.6.2 Abstraction of Service Information

The set of all transit, inbound, and outbound services for a node is the set of services available on all transit, inbound, and outbound routes. A node's abstract map may include all of this service information. The quantity of service information, however, may be unacceptably large for a node with many adjacencies, many routes, or many different service attributes for these routes. To reduce the size of its node's abstract map, a node representative should use concise representations of service attributes. For example, suppose several pairs of node entry and exit points share the same services on the routes connecting them. In an abstract map, these service attributes need only be listed once together with the relevant entry and exit pairs, rather than individually for each pair.

The node representative may also reduce the size of its node's abstract map by applying abstraction procedures to the service attributes related to particular entry or exit points or to all such points. In any case, the abstracted service attributes should reflect the actual services provided, in order to maximize the chance that the routes generated using abstract maps will actually satisfy the traffic service requirements. As part of the abstraction procedure, a node representative determines the number of distinct sets of service attributes to be advertised in its node's abstract map. The node representative assigns each set a label to be used in place of the descriptions of the attributes in the set, when specifying routes as described in section 5.1.

The following are example abstraction algorithms for reducing the quantity of service information in a node's abstract map:

1. Eliminate from the abstract map those services that are available only on a small number of transit, inbound, or outbound routes.
2. Eliminate from the abstract map those services that have severe restrictions (e.g., available only to a small number of users, for short periods of time, or for an unusual purpose).
3. For transit, inbound, and outbound services provided by the node, abstract the services provided by the component routes. Example abstraction techniques include:
   (a) A pessimistic view of the actual services. Such a view might be formed by choosing from the actual services the intersection of the service restrictions (when service restrictions are expressed in terms of what is permitted rather than what is not permitted), the worst (e.g., maximum delay, minimum throughput) value for each quality of service, and the maximum fee for service.
   (b) An optimistic view of the actual services. Such a view might be formed by choosing from the actual services the union of the service restrictions (when service restrictions are expressed in

terms of what is permitted), the best (e.g., minimum delay, maximum throughput) value for each quality of service, and the minimum fee for service.

(c) An "average" view of the actual services. Such a view might be formed by choosing from the actual services a subset of the service restrictions that appears to apply to the majority of users, the mean value of each quality of service, and the mean value of the fee for service.

(d) A range of the actual services. Such a view might be formed by combining the pessimistic and optimistic views.

We do not mandate the use of specific service abstraction algorithms for Nimrod. Instead, the abstraction algorithms applied in different portions of an internetwork may be completely different. In general, we expect that network managers will choose the services to advertise for their networks so as to maximize their expected revenues. This means trading off the cost incurred in and the income generated by providing a particular service. We expect most networks to offer a set of basic low-cost services to all users as well as special and probably more expensive services to particular groups of users.

4.6.3 Map Updates

A node representative updates its node's basic and abstract maps whenever they change. Specifically, a node representative updates its node's basic map whenever it receives a new abstract map from a component node. The node representative updates its node's abstract map in response to a change in the node's adjacencies or services resulting from a new basic map, a change in the reachability of boundary forwarding agents (learned through agent discovery) in the case of a node with no basic map), reconfiguration of the node's service information, or new measurements of service. All representatives of a given node have access to the same basic map and use the same abstraction algorithms, and hence can construct the same abstract map for the node.

Recall that each set of service attributes for a node includes information concerning the types of nodes to which the attribute set should be advertised. Thus, a node's basic and abstract maps may have sets of service attributes with restrictions on their distribution. Although each node has a single basic map (which may be empty) and a single abstract map, the advertised portion of a map depends upon the intended recipient. Hence, different recipients may obtain different subsets of the same map. The reason is that a node representative adheres to the distribution restrictions associated with its node's service attributes when distributing its node's maps. Thus, in an internetwork, a route agent requesting a map from a node receives from a representative of that node only those parts of the node's maps that are distributable according to the distribution restrictions. In the extreme case, if the distribution restrictions on each set of service attributes associated with a given node preclude a route agent from learning of these attributes, then the node is invisible to the route agent. We note that restricted distribution of a map cannot be guaranteed unless all recipients of that map honor its distribution restrictions.

A node representative distributes its node's basic and abstract maps in response to map queries, as described in section 4.6.4. It also automatically distributes its node's abstract map, to node representatives and route agents acting on behalf of the node's enclosing node, whenever there is a change in the portion of the map distributable within that node. Only one of a node's representatives should initiate automatic distribution of the node's new abstract map, in order to minimize the amount of internetwork resources consumed in map distribution. We recommend the node representative with the lowest-numbered EID, in the set of mutually-reachable representatives for the node, as the one to distribute maps automatically.

The subset of a node's abstract map (which may be the entire abstract map) distributed during automatic map distribution contains the sets of service attributes that may be advertised throughout the enclosing node. A node representative begins automatic map distribution by sending the abstract map to one of its boundary forwarding agents, who also acts as a forwarding agent for the enclosing node. That forwarding agent distributes the map to one of that node's representatives and route agents. The recipient node representative distributes the map to all node representatives with the same scope. These node representatives may then determine whether the information contained in the new abstract map from the component node results in a change in their node's abstract map. The recipient route agent in turn distributes the map to all route agents with the same scope, who may then use the information to generate routes within the node.

In the worst case, a change in a low-level node's abstract map may force changes in the abstract map of each of its ancestral nodes in the clustering hierarchy. We expect such changes to be rare, especially in nodes whose physical assets are multiply connected. Nevertheless, to control the amount of processing and transmission capacity required to distribute maps from nodes that change often, we recommend limiting the rate at which a node representative may generate successive map updates for its node.

4.6.4 Map Queries

Route agents query node representatives for maps they intend to use in generating routes. A route agent sends a map request message toward the node from which a map is desired. When the map request reaches a boundary forwarding agent for that node, the forwarding agent relays the request to a representative for that node. The node representative then determines how it will respond to the map request, according to the type of map request (described below) and the distribution restrictions that apply to the requestor. Hence, the map request message should contain information that enables the recipient node representative to evaluate the request.

The node representative responds to the route agent with the largest subset of the requested map that is consistent with its distribution restrictions. This subset may be empty if the node representative cannot fill the request. Whenever the node representative issues a negative response to a map request, it indicates the reason for the response (either it does not have the requested map or its distribution restrictions preclude distribution of the requested map to the route agent).

Although the information provided by an abstract map is usually sufficient for route generation, a route agent may desire more detailed information than that provided by a node's abstract map. In this case, the route agent requests a basic map of the node, which provides information about the node's internal structure in terms of component nodes and their services. Basic maps are large, and hence node representatives distribute basic maps only in response to requests for them.

To enable a route agent to control the amount of internetwork resources it requires to obtain and store basic maps, Nimrod offers a choice of basic map queries along the dimensions of *complete* or *partial* basic maps containing *full* or *abbreviated* abstract maps. A basic map is complete if it contains abstract maps for all component nodes and partial if it contains abstract maps for a proper subset of component nodes. To obtain a partial map, a route agent must specify which component nodes to include in the map. An abstract map is full if it contains service information and abbreviated if it does not contain service information.

Route agents use basic map queries to gain a more detailed view of a node or a portion of a node and thus to generate a route that meets specific service requirements. The following is an example of the use of map queries. A route agent desiring more information about a node requests a complete abbreviated map of that node, i.e., a complete basic map composed of abbreviated abstract maps for all component nodes. Information contained in this basic map provides the route agent with "pointers" to the component nodes. In response to the receipt of this basic map, the route agent then requests a partial full map, i.e., a partial basic map composed of full abstract maps for specific component nodes. The route agent may also place direct queries for some abstract maps of specific component nodes. Queries may be placed for increasingly detailed information, using basic map contents to locate component nodes, and so on.

4.6.5 Node Partitions

With Nimrod, a node is a contiguous collection of internetwork physical assets such that when all such assets are functioning properly, traffic can flow between any two endpoints contained within the node by traversing assets that belong only to that node. We call this the *contiguity* property of a node. If enough physical assets fail, however, a node may no longer possess the contiguity property; in this case, the node is *partitioned.*

A node partition may affect the reachability of only a small number of endpoints associated with a node, but it may also significantly impair the ability of the node to provide transit services. In particular, a entry point and an exit point may no longer be connected by intra-node routes. Only when a node's transit service is affected by a partition must this information be advertised in the node's abstract map. Note that a partition may isolate a set of endpoints associated with a node, so that they are not reachable from any entry point. This form of partition may not necessarily have an impact on transit service, and hence such a partition may be invisible from outside of the node.

Node representatives monitor the connectivity between entry and exit points using their node's basic map and using agent discovery, and they update their node's abstract map when the connectivity changes. Hence, node representatives can detect the existence of node partitions. A partitioned node is composed of disjoint *subnodes*, each of which may be viewed as a distinct "personality" of the node. A subnode possesses the contiguity property, and contains at least one node representative and one boundary forwarding agent. Communication between endpoints within two different sub-nodes of the same partitioned node requires a route that leaves and subsequently re-enters the node.

Representatives of the partitioned node that are contained within a particular subnode maintain a basic map and construct an abstract map from the perspective of the portion of the node they can directly reach, i.e., the subnode. The node representative with the lowest-numbered EID among those mutually reachable within the subnode initiates map distribution. Recipients of maps from the subnodes of a partitioned node can distinguish the maps by using the EIDs of the node representatives issuing the maps. Hence, a partitioned node is likely to have several different maps active simultaneously. Moreover, in the case of a partitioned node, the response to a map query should contain all known abstract or basic maps of the node, i.e., one from each subnode.

When constructing a route to an endpoint in a partitioned node, a route agent should attempt to generate a set of routes, one from the source node to each subnode of the partitioned node. The reason is that the node's basic and abstract maps distributed by the subnodes do not indicate whether a given endpoint is reachable from a given subnode. Thus, the route agent should generate multiple routes, to maximize the chance of generating a route that reaches the specified endpoint.

4.7 Association Database Maintenance

Endpoint representatives associate nodes and hence locators with the endpoints on whose behalf they act, as we discussed in section 4.5.2. They are also responsible for updating the association database with these new endpoint/locator associations. The association database is maintained by association agents, each of which must participate in updating and retrieval of association database contents. Each association agent caches endpoint/locator associations learned from authoritative sources and also maintains the set of endpoint/locator associations for which it is an authoritative source. However, not all association agents are authoritative sources, and not all nodes must have association agents. We expect that the Domain Name System (DNS) will play the part of the association agents in a Nimrod implementation. Nevertheless, we describe the behavior of association agents as separate entities, in case Nimrod is to be implemented in an environment without a DNS.

Most Nimrod databases are organized according to the nodes about which they contain information. Thus, to contact an authoritative source for a database portion corresponding to a particular node, an agent merely sends its query toward that node; upon receipt of the query, an appropriate agent acting on behalf of the node will generate a reply. Although association agents like other agents possess and act on behalf of nodes, the authoritative sources for associations are not organized according to nodes, for the following reason. The principal function of the association database is to answer queries about the set of nodes currently associated with a given endpoint. In many cases, the querying agent may have no idea where in an internetwork the endpoint currently resides, particularly if the endpoint is highly mobile.

Authoritative sources for specific endpoint associations must be accessible using endpoint names only. Each endpoint must have a globally unique label that can be used as a key for finding an authoritative source for that endpoint's associations. We recommend using as keys endpoint labels that have structure. (Unstructured labels could be used, but it makes the search procedure most complicated.) For the purposes of discussion, we assume that endpoint names, which possess structure derived from the management organization that assigns labels to endpoints (as described in section 4.2), will be used as keys. The names of endpoints with a common manager will share a common prefix. These prefixes can be used to direct a query toward the correct authoritative source, as we describe below.

Authoritative sources for specific endpoints' associations should be situated in well-known stationary locations under the control of the endpoints' managers. Although not mandatory, such organization of authoritative sources significantly simplifies the management of the association database. Moreover, authoritative sources for specific endpoints' associations should ideally reside close to those endpoints so as to minimize the costs of updating the endpoint's associations. In the presence of highly mobile endpoints, however, such organization of authoritative sources may not be possible. Nevertheless, if a manager has a priori knowledge of the vicinity in which a set of its endpoints may be found, that manager may be able to place authoritative sources within that vicinity.

The following example serves to illustrate how one might organize authoritative sources within an association database. Consider an endpoint name as a string composed of m substrings, one for each level of management over the endpoint. Beginning from the left of the label and successively concatenating substrings, each string thus formed identifies a prefix to a block of endpoint labels. The prefix corresponds to a particular manager. This correspondence must then be extended to a particular node in the clustering hierarchy and hence a set of association agents. One may simply select the node in which the manager resides in order to make the correspondence.

To construct linked chains of managers leading to authoritative sources for endpoint associations, one begins with the universal node. For each value of the first substring of a label endpoint, one selects the node containing the manager corresponding to that prefix. One must ensure that all association agents in the universal node point toward the selected node for locating authoritative sources for endpoints whose label prefix equals that substring. One then repeats the procedure at the association agents in the selected nodes, but with the label prefix equal to the concatenation of the corresponding first and second substrings. The procedure concludes when all substrings but the last have been included in the label prefix, thus forming a linked chain of nodes following the management organization. At this point, each node at the end of a linked chain is the authoritative source for all endpoints whose label prefix equals the concatentation of the first $m - 1$ substrings.

4.7.1 Association Updates

When a representative for an endpoint issues an association update message for that endpoint, it distributes that update to association agents within its vicinity. Updating proximate association agents helps to reduce the cost of nearby endpoints' association queries for that endpoint's locators. The endpoint representative also distributes the association update to the authoritative sources for that endpoint's associations, which may not necessarily reside within the given vicinity. Implications of updating associations when dealing with mobile endpoints are not discussed here but instead are treated in [2].

Association updates propagate up the node clustering hierarchy toward the universal node and then back down along the management hierarchy toward the authoritative sources. Association update distribution begins when an endpoint representative issues an association update for one of its endpoints and sends it to a neighboring forwarding agent with the same scope. That forwarding agent in turn distributes the association update to an association agent with the same scope, if such an agent exists, and otherwise distributes the update to a boundary forwarding agent with the same scope. In the latter case, the boundary forwarding agent acts as a forwarding agent for the enclosing node and in turn distributes the association update to an association agent with that scope, if such an agent exists, and so on.

Upon receipt of an association update from a forwarding agent, an association agent distributes the update to all other association agents with the same scope. Association agents receiving association updates from other association agents never redistribute those updates. Each association agent updates its association database according to the contents of the association update. The association agent that received the update from a forwarding agent determines whether it is an authoritative source for the given endpoint's associations. If it is an authoritative source, the association agent concludes update distribution. Otherwise, the association agent uses the endpoint name to determine whether its scope is within the linked chain of nodes pointing toward the authoritative sources for the endpoint's association.

If the association agent's scope is not within the linked chain of nodes, the association agent distributes the update to a boundary forwarding agent with the same scope. In this case, the boundary forwarding agent acts as a forwarding agent for the enclosing node and in turn distributes the association update to an association agent with that scope, if such an agent exists, and so on.

If the association agent's scope is within the linked chain of nodes, however, the association agent sends the update toward the next node in the chain, where it will be received by a boundary forwarding agent for that node. In this case, the boundary forwarding agent distributes the update to an association agent with the same scope, and so on. If the update encounters no intersections with the linked chain of nodes, it will eventually propagate up to the universal node, which is the first node on every such chain. Thus, with this update distribution procedure, the association update is guaranteed to reach a node in the linked chain of nodes leading to an authoritative source.

4.7.2 Association Queries

Any agent may make an association query. In most cases, it will be an endpoint representative requesting the set of locators associated with the endpoints with which its endpoints communicate. An endpoint representative issuing an association query distributes it to a forwarding agent with the same scope. That forwarding agent in turn distributes it to an association agent with the same scope, if such an agent exists, and otherwise distributes it to a boundary forwarding agent with the same scope. In the former case, if it cannot answer the query, the association agent forwards the query to a boundary forwarding agent with the same scope. In either case, the boundary forwarding agent then acts as a forwarding agent for the enclosing node and in turn distributes the association update to an association agent with that scope, if such an agent exists, and so on.

We note that association queries follow the same distribution patterns as association updates. An indirect association query may be answered by any association agent that has the information requested. However, a direct association query may only be answered by an authoritative source. In the worst case, a direct association query may have to travel all the way up to an association agent in the universal node and back down to an authoritative source.

5 Routes

When one endpoint wishes to communicate with another endpoint, the representative of the endpoint initiating communication issues an association query in order to determine the set of locators associated with the other endpoint. (In this section, we discuss unicast route generation only; multicast issues are discussed in [3].) We expect that the source endpoint's representative will usually be the one initiating communication, but the destination endpoint's representative may also initiate communication. The decision will be a function of the service requirements of the endpoints (e.g., the endpoint paying for the service may wish to dictate the route used for the session).

After obtaining the locators of the endpoints, the endpoint representative initiating communication determines whether there exists a satisfactory path for the session traffic. If such a path exists, the endpoint representative uses it (as described in section 6). If no such path exists, the endpoint representative attempts to obtain a route and establish a path based on the route. If the session has specific service requirements, the initiating endpoint representative requests a route for the session from a route agent. This request contains the source and destination endpoint locators and the traffic service requirements. If the session has no strict traffic service requirements, in terms of quality or access restrictions, a route request may not be necessary. In this case, an adequate route description may consist of the source and destination endpoint locators only.

When responding to a route request, a route agent first searches its route database for a set of feasible routes and, if unsuccessful, attempts to generate a set of feasible routes. Feasible routes are those that meet the service requirements of the session and are consistent with the services of the intermediate nodes contained in the routes. Before including a node within a route, the route agent ensures that the node's service attributes are consistent with the session's service requirements. The route agent then records the label for the set of service attributes to be used within the route specification, described in section 5.1.

Once the route generation procedure concludes, the route agent responds to the requesting endpoint representative either with a set of feasible routes or with an indication that no feasible route could be found. The endpoint representative may select one or more routes from this set to carry the session traffic; Nimrod supports single-path as well as multiple-path routing.

Nimrod permits both localized and distributed route generation. We expect that route generation will normally be localized at a single route agent acting on behalf of a source or destination. Such localization of route generation helps to increase the source's or destination's control over route selection and to confine the costs of route generation to those agents acting on behalf of the traffic using the route. In some circumstances, an endpoint representative might request routes from different route agents and concatenate them to form the whole route; this is an example of distributed route generation.

For Nimrod, we do not mandate the use of specific route generation procedures. Instead, each route agent may apply route generation procedures of its own choosing. Such flexibility enables network managers to make their own cost-performance tradeoffs in route generation and to experiment with new route generation algorithms in their networks. Nevertheless, there are well-defined inputs and outputs common to any such route generation procedure. Inputs include a set of abstract maps and the source and destination endpoint locators and service requirements. Outputs include a set of feasible routes or an indication that a feasible route could not be found.

5.1 Representation

A route agent generates routes at the minimum node granularity necessary to meet the session's service requirements, subject to the node granularity of the advertised maps at its disposal. Different portions of a single route may be generated and specified at different granularities. A route specification need only include those nodes that must be used in order to satisfy the traffic service requirements. At the minimum, the route specification must include the source and destination node locators. Moreover, each node in the specification must be accompanied by the label of the set (which may be null) of service attributes that resulted in its inclusion in the route; this service information is used to guide message forwarding across these nodes. The portion of a route between two successive nodes in a route specification is called a *route segment*.

The following is an example of route specification granularity. If the source or destination endpoint requests a particular service provider, then the node containing that service provider should be included in the route. If there are several equivalent ways of reaching that service provider, the source or destination may wish to leave that segment of the route unspecified, so that intermediate forwarding agents can choose among multiple alternate routes for that segment.

Fine granularity routes concentrate control over and cost of route selection at a route agent acting on behalf of a source or destination and minimize the number of independent forwarding decisions exercised at intermediate forwarding agents. Coarse granularity routes relinquish some control over and distribute the cost of route selection to intermediate forwarding agents along the route, and hence may require some distributed route generation to fill in the segments of the route between the specified nodes. However, coarse granularity routes are more fault tolerant than fine granularity routes, because the intermediate forwarding agents can quickly repair failed routes by obtaining regenerated segments from intermediate route agents. Moreover, coarse granularity routes provide a means for load balancing, because the intermediate forwarding agents can distribute the load over multiple routes between specified entities within a given segment.

6 Message Forwarding and Path Management

Nimrod supports two distinct data message forwarding modes: *flow* and *datagram*. (The Nimrod architecture document [1], distinguished four separate forwarding modes, highlighting their differences.

However, we note that each of those forwarding modes may be classified as an instance of one of the two forwarding modes described here.) For each mode, a forwarding agent's "next-hop" forwarding decision is dictated by the information stored in its forwarding database and by information contained within the message to be forwarded.

Both of the Nimrod forwarding modes rely on the existence of underlying *paths* to fill in route segments. A path is based on the route selected for a traffic session and corresponds to forwarding state stored in endpoint representatives and forwarding agents along the route. Each path has a label which is unique within all of the endpoint representatives and forwarding agents along the path, but is not necessarily globally unique. Distinct traffic sessions may use the same path, and distinct paths may be based on the same route.

A path may connect one or more source endpoints and one or more destination endpoints. The *context* of a path is the lowest node in the clustering hierarchy that contains all of the path's sources and destinations. Hence, Nimrod can support unicast and multicast paths using the same path management procedure. (In this section, we discuss unicast paths only; multicast issues are treated in [3].)

Paths may be set up from source to destination or from destination to source. Each path has an *initiator* and a *target*. We expect that most paths will be set up from the source endpoint to the destination endpoint. Hence, the initiator usually begins the path setup procedure on behalf of the source endpoint, and the target usually accepts or rejects a path on behalf of the destination endpoint.

Endpoint representatives and forwarding agents execute a path management procedure to install path state in and remove path state from their forwarding databases. Each endpoint representative and forwarding agent maintains a portion of the forwarding database pertaining to those paths that originate, terminate, or pass through it. Forwarding state is established in endpoint representatives and forwarding agents such that traffic may travel in either direction of along a path. Nimrod permits multiple traffic sessions to use the same path, thus helping to contain the amount of internetwork resources consumed in setting up, tearing down, and managing resources for paths. Path labels enable forwarding agents to multiplex and demultiplex multiple traffic sessions over a single path.

6.1 Flow Mode

Flow mode requires the establishment of session-specific forwarding state in endpoint representatives and certain forwarding agents along the routes selected for a traffic session. With flow mode, each session is assigned one or more paths, derived from the selected routes. A path corresponds to forwarding state stored in endpoint representatives and forwarding agents along a route. Each path has a label which is unique within all of the endpoint representatives and forwarding agents along the path, but is not necessarily globally unique. Distinct traffic sessions may use the same path, and distinct paths may use the same route. The minimum forwarding state required for flow-mode forwarding includes linkages between the path label and the path's previous- and next-hops and any service guarantees for traffic control. In flow mode, data messages carry the path label which guides the message forwarding decisions at forwarding agents along the path.

Flow mode provides fast and consistent message forwarding according to path labels, once the session-specific state is established in the forwarding agents. This state, however, consumes memory in forwarding agents, and the procedure for its installation may impose some delay before data messages can be forwarded successfully to their destinations unless data is sent along with path establishment messages. Flow mode should be used when session-specific state is necessary to meet traffic service requirements or when the cost of state installation and maintenance can be amortized over many messages. Hence, we recommend flow mode for traffic sessions requiring guaranteed service or consisting of several messages.

6.2 Datagram Mode

Datagram mode does not require the establishment of any session-specific forwarding state. In datagram mode, data messages carry a description of the selected route, which guides the message forwarding decisions at forwarding agents along the route. Each forwarding agent at the beginning of a route segment makes an independent forwarding decision for that segment, and hence the session source and destination relinquish some control over message forwarding. On the other hand, datagram mode provides robust forwarding, in the sense that the intermediate forwarding agents can base their message forwarding decisions on the current state of their portion of the internetwork. Datagram mode should be used when the state of the internetwork is unpredictable or when the cost of forwarding state installation and maintenance is unacceptable. Hence, we recommend datagram mode for traffic sessions traversing highly changeable portions of an internetwork or consisting of few messages.

6.3 Forwarding State and Multilevel Paths

Endpoint representatives and forwarding agents try to form paths by piecing together existing paths rather than by establishing new paths. This method provides the lowest-cost message forwarding in terms of the amount of route generation and forwarding state installation required. In a busy internetwork, there are likely to be many existing paths, and hence we expect this mechanism to be much less expensive than individually establishing and maintaining paths for each traffic session.

Nimrod paths are inherently multilevel. Each path begins with a single path, ${p^1}_1$, derived from the selected route between the source and destination endpoints for the traffic session. (The superscript indexes the level of the path, where the top level is 1. The subscript indexes the path for the corresponding route segment of the higher-level path.) This path comprises multiple contiguous paths, ${p^2}_1, \ldots, {p^2}_n$, one for each of the n segments of the route on which ${p^1}_1$ is based. Each ${p^2}_j$ itself comprises multiple contiguous paths corresponding to each of its segments, and so on. In general, for each ${p^i}_j$ composing ${p^{i-1}}_k$, the initiator and target of ${p^i}_j$ maintain linkages to the path ${p^{i-1}}_k$, which helps to guide forwarding along the successive segments of ${p^{i-1}}_k$.

We now describe how a new traffic session uses paths at multiple levels, distinguishing the actions in flow mode and datagram mode where appropriate. Whenever an endpoint representative needs to use Nimrod forwarding for traffic from one of its endpoints, it always checks whether there already exists a satisfactory path for the session. This is true whether the new session desires flow or datagram mode forwarding. If a satisfactory path exists and if the endpoint representative is at the source endpoint, the endpoint representative includes the path label in the endpoint's messages and forwards them along the existing path. If no such path exists, however, the endpoint representative attempts to obtain a feasible route for the session. Note that route generation might not be required and that a feasible route might include only the source and destination locators.

After obtaining a feasible route, the endpoint representative proceeds to determine where to install the necessary forwarding state.

Flow mode: The endpoint representative becomes the initiator of a new path, ${p^1}_1$, and generates a path setup message.

If the route contains more than the source and destination locators, the endpoint representative then checks whether there already exists a satisfactory path for the session from itself to the next node in the route specification. Provided such a path, ${p^2}_1$, exists, the endpoint representative proceeds as follows:

Flow mode: The initiator of ${p^2}_1$ (which is also the initiator of ${p^1}_1$) links ${p^1}_1$ and ${p^2}_1$ in its forwarding database and sends ${p^1}_1$'s path establishment request to the target of ${p^2}_1$. Upon reception of the path establismen message, the target of ${p^2}_1$ also links ${p^1}_1$ and ${p^2}_1$ in its forwarding database.

Datagram mode: The initiator of ${p^2}_1$ sends the data message to the target of ${p^2}_1$.

If no satisfactory path, ${p^2}_1$, yet exists between the first two nodes in the route specification, the endpoint representative attempts to form such a path by piecing together existing paths. First, the endpoint representative determines the context of the prospective path, ${p^2}_1$. Second, it determines the locator of the node, X, immediately below the context and containing the next node on the route. This locator is used to limit the search for component paths, confining it to paths with targets for which the locator of X is a prefix. The endpoint representative then searches through its existing paths for the path whose destination locator is the longest match against the locator of the next node in the route specification. If the locator of X is a prefix of the selected path, the selected path becomes ${p^3}_1$. The endpoint representative must then attempt to complete the remainder of ${p^2}_1$ by piecing together other paths, ${p^3}_i$. If the locator of X is not a prefix of the selected path, the endpoint representative then seeks a path up to the source node's enclosing node, as there are likely to be more existing paths between higher-level entities.

To this end, the endpoint representative checks whether there already exists a satisfactory path whose target is an exit point of the source node's enclosing node. If such a path, ${p^3}_1$, exists, the endpoint representative proceeds to search for a path to the next node on the route. If no such path exists, the endpoint representative attempts to obtain a feasible route and establish a path, ${p^3}_1$, to the source node's enclosing node.

Once the endpoint representative establishes ${p^3}_1$, the target of ${p^3}_1$ then attempts to find a path to the next node on the route in order to piece together ${p^2}_1$. If there is no such path from the source node's enclosing node to the next node on the route, the above procedure may need to be repeated for successively higher-level ancestors of the source node, up to but not including the context of the prospective path ${p^2}_1$. In the worst case, upon reaching the context of ${p^2}_1$, there might not be a suitable existing path to the next node, and a route might need to be requested to that node.

This iterative path formation procedure is performed by the target of each path thus selected, which then becomes the initiator for the path for the next segment, and so. The procedure terminates after attaining the last node in the route, possibly linking together paths at many different levels.

In the presence of multilevel paths, each data message carries a nested sequence of path labels, in order to enable all forwarding agents involved in the paths to forward the message correctly. Intermediate forwarding agents update the path labels in the message, according to the linkages between paths stored in their forwarding databases. For example, upon receipt of a data message, the target of path ${p^j}_i$ finds it is linked to path ${p^{j-1}}_k$ which in turn is linked to path ${p^j}_{i+1}$ and hence is the initiator of ${p^j}_{i+1}$. This forwarding agent strips off the label for ${p^j}_i$ and replaces it with the label for ${p^j}_{i+1}$, before forwarding the message along that path.

6.3.1 Path Labels

Path label collisions result in path establishment failure. We want to be able to assign relatively short path labels in a way that reduces the probability of collisions of path labels from different paths.

Below we describe an example mechanism for selecting path labels that are not globally unique but that are unlikely to collide. It is a variant of the "run and twiddle" search technique. An endpoint representative or forwarding agent begins by choosing a "random" number for the first path label and then selects additional labels in sequence until it experiences a "collision" with another path. At this point, the endpoint representative or forwarding agent again selects a new "random" starting point for its path labels.

An inexpensive way of choosing a "random" starting point is to use the lowest-order n bits of its time-of-day clock and then reverse their order. The low-order clock bits are unlikely to be the same in all endpoint representatives and forwarding agents, because they change so rapidly and because we are not planning on synchronizing clocks to this granularity. Hence, the low-order clock bits are a reasonable choice for assigning different initial path labels, but they do not prevent label collisions. Consider the case in which there are two forwarding agents in the same vicinity with low-order clocks differing only in the value of the least-significant bit. Although these forwarding agents would initially have different path labels, their incremental path label selections from these starting points would have a high probability of colliding, because their starting points were so close. Bit reversal helps to spread out the initial path label starting points for the two forwarding agents, so that selection of subsequent path labels by each of these agents is less likely to result in a collision.

We can further reduce the chance of path label collisions by using the following procedure. Endpoint representatives and forwarding agents record not only the path label for every path they establish but also the sequence of path labels stored in the path establishment message. Thus, when a forwarding agent or target receives a path establishment message with a path label already in use, it can attempt to use the rest of the sequence of path labels contained in the message to distnguish the two paths. Therefore, path collisions only occur when two paths have the same sequence of path labels.

6.3.2 An Example of Path Setup

The example below illustrates flow mode's installation of forwarding state. Suppose two endpoints wish to establish a traffic session. Let *U:V:W:X* be the locator for the source endpoint, and let *U:V:A:B* be the locator for the destination endpoint. Thus, the context for any potential path between these two endpoints is *U:V*.

Suppose that no satisfactory path for the session yet exists between *U:V:W:X* and *U:V:A:B*. The first-hop forwarding agent then becomes the initiator for a new path f. Suppose that there are no special service requirements for the session and so the route for f consists of only the source and destination nodes.

To assemble f, the initiator of f attempts to find a satisfactory path from *U:V:W:X* to any node in *U:V:A*. Suppose no such path yet exists. In this case, the initiator of f has reached the context *U:V* without finding a satisfactory path and must then look for a satisfactory path to an exit point of *U:V:W*. Suppose such a path g exists. The initiator of g (which is also the initiator of f) links f and g and sends the path establishment message for f to the target of g, which in turn also links f and g.

Suppose there are no direct paths from the target of g to *U:V:A:B*. Then the target of g looks for a satisfactory path to any node in *U:V:A*. Suppose no such path yet exists. In this case, the target of g requests a route to *U:V:A* and receives a route terminating at *U:V:A:C*. The target of g becomes the initiator of the path e to *U:V:A:C*, which in turn consists of paths $d_1, \ldots, d_5$, one for each route segment of e. The initiator of e links f and e, and the initiator of each d_i links e and d_i. Furthermore, the initiator of e sends the path establishment message for f to the target of e.

The target of e then looks for a satisfactory path to *U:V:A:B*. Suppose such a path h exists. The initiator of h (which is also the target of e) links f and h and then sends the path establishment message for f to the target of h, which in turn also links f and h. Thus, path installation for f requires only a single route request for the path, e, and new forwarding state installed at eight forwarding agents.

6.4 Path Management Protocol

Nimrod can execute a path management protocol to install and remove forwarding state in endpoint representatives and forwarding agents. A skeleton of such a protocol is sketched below.

The path management protocol uses three messages: *setup*, *accept*, and *teardown*. All such messages travel along the path to which they refer. Path management protocol messages may be used to collect and return performance monitoring information for a path (e.g., path delay and throughput) as well as to set up and tear down a path. Monitoring operates in two modes: *collection* and *return*. Hence, monitored information may appear in a message as information being collected or information being returned. Moreover, a message may simultaneously collect and return information for different directions of a path.

All path management protocol messages (and in fact all Nimrod messages except data messages) are covered by integrity and authentication checks and are sent using a reliable transaction protocol (with positive and negative acknowledgements) between endpoint representatives and successive forwarding agents along the path. This helps to reduce the amount of network resources consumed by retransmissions in lossy environments and also helps to determine problems (e.g., incompatible Nimrod protocol versions) for unsuccessful message transmissions.

The setup message is generated by the path initiator and is used to establish forwarding state in forwarding agents. It contains:

1. The label for the path.
2. An indication of whether the path is source- or destination-initiated.
3. Service requirements (e.g., delay, throughput, reliability).
4. The EIDS of the source and destination endpoints and endpoint representatives. (The EID for the destination endpoint's representative is available in the destination endpoint's entry in the association database.)
5. The route, in terms of the locators of the nodes through which it passes (must at least include the source and destination nodes) and the label of the service attributes used across each node in the route.
6. Monitored information for the path (e.g., path delay, path throughput), updated at each node in the node.
7. Optionally, data.

The accept message is generated by the path target and is used to indicate successful path establishment from initiator to target. It contains the label for the accepted path and any monitored information for the path collected during setup.

Accept messages travel backwards along paths. A path may be used for data transport before an accept message is generated by the target or received by the initiator. Note that a source as initiator may wish to wait for an accept message from the target before sending data on a path. For example, if the source pays for all packets sent, whether or not they are successfully received at the destination, it may want to wait to make sure that the path is successfully established before sending data to the destination.

The teardown message may be generated by any endpoint representative or forwarding agent on the path and is used to remove forwarding state. It contains the label for the path to tear down, the reason for the teardown, and any monitored information collected along the path.

Teardown messages travel in both directions along paths and may result from any of the following:

1. Loss of a lower-level path which is a component of the specified path.
2. A timeout. Paths have a maximum lifetime to ensure that forwarding state for failed paths is eventually removed.
3. An inconsistency between the service requirements for the traffic session and the services provided by the node. (To make such a consistency check, a forwarding agent must have knowledge of the route for each path.)

4. Preemption in favor of another path. Endpoint representatives and forwarding agents are free to implement their own preemption mechanisms, but no particular ones are required for Nimrod.

Note that teardown messages may be issued in response to unsuccessful path setup attempts as well as to irreconciliable changes in existing paths.

6.4.1 Path Acceptance

Each setup message in flow mode and each data message in datagram mode contains the route specification and additional service requirements, such as resource reservation requests. Intermediate boundary forwarding agents and the target are responsible for determining the acceptability of received setup or datagram messages. Acceptability is in part based on the perceived consistency between the route specification and service requirements contained in the message and the service attributes of each node traversed.

When a forwarding agent deems a setup message unacceptable, it sends a teardown message back toward the initiator. A target may also deem a setup message unacceptable if there is perceived inconsistency between the route specification and service requirements contained in the message and the service requirements of the target endpoint. Each target that accepts a setup message informs the initiator. If there is an inconsistency with the target endpoint's service requirements, the target takes one of two actions, depending upon whether the target is the path's destination or source:

1. If the target is at the destination endpoint, it returns to the initiator a teardown message containing its endpoints' destination service requirements. The initiator is then responsible for obtaining a route and setting up a path that is consistent with both the source and destination service requirements.
2. If the target is at the source endpoint, it returns to the initiator a teardown message indicating that it will generate its own route. The target is then responsible for obtaining a route and setting up a path that is consistent with the source service requirements and the destination service requirements contained in the setup message.

When a forwarding agent deems a datagram message unacceptable, it may inform the source endpoint representative. This action is not strictly necessary for datagrams, but may help network management to determine why messages are not getting through to their destinations.

7 Security Considerations

The protocols used to transport Nimrod control information must include mechanisms for source authentication, error detection, and recency determination that will help to prevent the introduction of incorrect control information, as we discussed in section 3.1. In particular, these mechanisms will help to prevent those denial of service attacks resulting from masquerading sources of control information, corruption of control information, and message replays.

This document is not a protocol specification and therefore does not contain a detailed description of the security mechanisms used in Nimrod.

8 Author's Address

Martha Steenstrup
BBN Systems and Technologies
10 Moulton Street
Cambridge, MA 02138
Phone: (617) 873-3192
Email: msteenst@bbn.com

References

[1] I. Castiñeyra, J. N. Chiappa, and M. Steenstrup. *The Nimrod Routing Architecture.* March 1995. Working Draft.

[2] S. Ramanathan. *Mobility Support for Nimrod: Requirements and Approaches.* February 1995. Working Draft.

[3] S. Ramanathan. *Multicast Support for Nimrod: Requirements and Approaches.* February 1995. Working Draft.