

# **TESTING IPV6 ROUTERS WITH THE NEIGHBOR DISCOVERY PROTOCOL**

by

Jenny Koo  
B.A.Sc Electrical and Computer Engineering  
University of British Columbia, 1999

PROJECT SUBMITTED IN PARTIAL FULFILLMENT OF  
THE REQUIREMENTS FOR THE DEGREE OF  
MASTER OF ENGINEERING

In the School  
of  
Engineering Science

© Jenny Koo 2004

SIMON FRASER UNIVERSITY

Fall 2004

All rights reserved. This work may not be  
reproduced in whole or in part, by photocopy  
or other means, without permission of the author.

## APPROVAL

**Name:** Jenny Koo  
**Degree:** Master of Engineering  
**Title of Project:** Testing IPv6 Routers with the Neighbor  
Discovery Protocol

**Examining Committee:**

**Chair:** **Dr. Marek Syrzycki**  
Professor in the School of Engineering Science

---

**Dr. Stephen Hardy**  
Senior Supervisor  
Professor in the School of Engineering Science

---

**Dr. Craig Scratchley**  
Supervisor  
Lecturer in the School of Engineering Science

---

**Dr. Nicholas Malcolm**  
Supervisor  
Project Manager at Agilent Technologies

**Date Defended/Approved:** November 26, 2004

# SIMON FRASER UNIVERSITY



## PARTIAL COPYRIGHT LICENCE

The author, whose copyright is declared on the title page of this work, has granted to Simon Fraser University the right to lend this thesis, project or extended essay to users of the Simon Fraser University Library, and to make partial or single copies only for such users or in response to a request from the library of any other university, or other educational institution, on its own behalf or for one of its users.

The author has further granted permission to Simon Fraser University to keep or make a digital copy for use in its circulating collection.

The author has further agreed that permission for multiple copying of this work for scholarly purposes may be granted by either the author or the Dean of Graduate Studies.

It is understood that copying or publication of this work for financial gain shall not be allowed without the author's written permission.\

Permission for public performance, or limited permission for private scholarly use, of any multimedia materials forming part of this work, may have been granted by the author. This information may be found on the separately catalogued multimedia material and in the signed Partial Copyright Licence.

The original Partial Copyright Licence attesting to these terms, and signed by this author, may be found in the original bound copy of this work, retained in the Simon Fraser University Archive.

W. A. C. Bennett Library  
Simon Fraser University  
Burnaby, BC, Canada

## **ABSTRACT**

The Neighbor Discovery Protocol (NDP) is a fundamental component of Internet Protocol version 6 (IPv6) networks. An IPv6 network is made up of a collection of nodes, each of which is either a router or a host, connected via a physical network. NDP serves as the foundation that allows IPv6 to run transparently over any physical network. On a given network node, NDP functions as a mechanism for discovering the existence and identity of neighboring hosts and routers, for determining a neighbor's data link layer address given its IPv6 address, and for producing a unique IPv6 address based on the node's data link layer address.

Because of the importance of the role NDP plays in IPv6 networks, testers of IPv6 routers should include the verification of NDP in their test plan. This report discusses the steps a tester needs to perform in order to verify the operation of NDP on an IPv6 router. In addition, this report also outlines the ways in which NDP can be used to facilitate the configuration of both testers and routers in setting up test scenarios, such as traffic generation and real-time measurements, which are not focused on NDP.

## **ACKNOWLEDGEMENTS**

A trolley of thank yous to Nick Malcolm for letting me work on the NDP module in the Agilent N2X application as a part of my MEng project, for closely supervising my work, and for giving me all your support during these busy months. You are the best manager in the world!

A bucket of thank yous to Dr. Stephen Hardy and Dr. Craig Scratchley for agreeing to be my academic supervisors, and for looking over my outlines and abstracts to make sure that I was on the right track. I hope I didn't overflow your inboxes with all my emails.

A sack of thank yous to Dr. Marek Syrzycki for agreeing to be the chair for my presentation. I hope I hadn't misspelled your name anywhere.

A load of thank yous to Raj Pabla and Penny Simpson for helping me through the process and the forms for submitting my project, applying for graduation, etc. I hope I wasn't too big of a pest with all my questions, emails and phone calls.

A world of thank yous to my family and friends for giving me all your support and for being there when I needed you. Now that school is coming to an end, I promised that I will get more sleep and be less grumpy. Okay, I will try anyways.

# TABLE OF CONTENTS

<b>Approval</b> .....	<b>ii</b>
<b>Abstract</b> .....	<b>iii</b>
<b>Acknowledgements</b> .....	<b>iv</b>
<b>Table of Contents</b> .....	<b>v</b>
<b>List of Figures</b> .....	<b>vii</b>
<b>List of Tables</b> .....	<b>viii</b>
<b>List of Acronyms</b> .....	<b>ix</b>
<b>1 Introduction</b> .....	<b>1</b>
<b>2 Overview of NDP</b> .....	<b>3</b>
2.1 Goals of NDP .....	3
2.2 NDP Hosts and Routers.....	4
2.3 NDP Messages and Addresses .....	4
2.3.1 Message Format .....	5
2.3.2 Message Addresses.....	5
2.3.3 Message Types .....	8
2.4 Router and Prefix Discovery.....	10
2.5 Next-hop Determination .....	11
2.6 Link Layer Address Resolution.....	12
2.7 IP Address Autoconfiguration.....	12
2.8 Other Features of NDP .....	13
2.8.1 Neighbor Unreachability Detection .....	13
2.8.2 Duplicate Address Detection .....	16
2.8.3 Router Redirect .....	16
2.9 Current Status of NDP .....	17
<b>3 Testing an IPv6 Router's NDP Implementation</b> .....	<b>18</b>
3.1 Fundamentals of Protocol Testing.....	18
3.1.1 Conformance Testing .....	18
3.1.2 Scalability Testing .....	19
3.2 A Model of a NDP Tester .....	20
3.2.1 Goals of a Protocol Tester.....	20
3.2.2 Static Representation of the Model.....	23
3.2.3 Dynamic Representation of the Model.....	27
3.3 Test Scenarios .....	30
3.3.1 Initial Connection, Router Discovery, Prefix Discovery and Address Autoconfiguration .....	31
3.3.2 Address Resolution, Neighbor Unreachability Detection and Duplicate Address Detection .....	34
3.3.3 Next-hop Determination.....	37
3.3.4 Router Redirect .....	40

3.4	Application of the Test Scenarios .....	42
3.4.1	Conformance Tests .....	42
3.4.2	Scalability Tests .....	43
3.5	Future Extensions to the Model.....	44
<b>4</b>	<b>Using NDP as a Test Configuration Tool.....</b>	<b>46</b>
4.1	Fundamentals of Performance Testing .....	46
4.1.1	Traffic Generation.....	47
4.1.2	Real-time Measurements .....	47
4.2	Configuration Scenarios.....	48
4.2.1	Automatic Interface Discovery .....	48
4.2.2	Automatic Address Setup.....	49
4.3	Extensions to the Protocol Tester Model for Performance Testing .....	49
<b>5</b>	<b>Conclusion .....</b>	<b>52</b>
	<b>References .....</b>	<b>54</b>

## LIST OF FIGURES

Figure 2.1: An Ethernet link with connected NDP hosts and routers .....	4
Figure 2.2: NDP packet protocol hierarchy .....	5
Figure 2.3: State transition diagram for neighbor reachability states .....	15
Figure 3.1: Class diagram of the NDP test model .....	24
Figure 3.2: Sequence diagram for router discovery and address autoconfiguration.....	28
Figure 3.3: Sequence diagram for initial connection .....	32
Figure 3.4: Sequence diagram for address resolution, neighbor unreachability detection, and duplicate address detection.....	35
Figure 3.5: Sequence diagram for next-hop determination .....	38
Figure 3.6: Sequence diagram for router redirect.....	41
Figure 4.1: Collaboration diagram for performance testing scenarios .....	50



## LIST OF TABLES

Table 2.1: Summary of reachability states for on-link neighbors .....	14
---	----

## LIST OF ACRONYMS

ARP	Address Resolution Protocol
ATM	Asynchronous Transfer Mode
DHCP	Dynamic Host Configuration Protocol
ICMP	Internet Control Message Protocol
IETF	Internet Engineering Task Force
IPv4	Internet Protocol version 4
IPv6	Internet Protocol version 6
ISP	Internet Service Providers
MTU	Maximum Transmission Unit
NA	Neighbor Advertisement
NBMA	Non-broadcast Multiple Access
NDP	Neighbor Discovery Protocol
NS	Neighbor Solicitation
POS	Packets over SONET
PTOP	Point-to-point
RA	Router Advertisement
RFC	Request For Comments
RS	Router Solicitation
UML	Unified Modelling Language

# 1 INTRODUCTION

The Neighbor Discovery Protocol (NDP) is one of the most popular data link layer protocols for Internet Protocol version 6 (IPv6) networks<sup>1</sup>. Data link layer protocols are used by network nodes on the same link to exchange information with each other, and, as such, play a fundamental role in maintaining the healthy operation of a network. Therefore, a router participating in NDP exchanges in a network is expected to adhere to standardized NDP operating procedures, and to not become a hindrance to the network as the utilization to both the network and the router itself increases.

NDP is designed to be run over IPv6 networks of any link layer type. It provides the process for neighboring network nodes to resolve a neighbor's link layer address based on its IPv6 address, to discover neighboring routers that are willing to participate in packet forwarding, to automatically configure IPv6 addresses, to determine the reachability or availability of a neighbor, and to detect when duplicate addresses are configured on the link.

This report examines the ways in which NDP can be used to test an IPv6 router. Chapter 2 provides an overview of NDP, including the addressing scheme that is used to transmit messages, the types of messages that are exchanged and the meaning of various fields within each message, the basic operations and the message handshake that is used to carry out each operation, and the current status of the protocol itself. Chapter 3 covers the fundamentals of protocol testing, including an explanation of

---

<sup>1</sup> The Internet Protocol (IP) is the network layer protocol used in the Internet. The existing versions of IP include version 4, which is widely deployed today, and version 6, which is rapidly gaining acceptance as the IP protocol of the choice for emerging networks.

conformance and scalability testing. It also portrays the way in which the NDP implementation on a router can be tested, and presents a model of a NDP protocol tester. In addition, the chapter also outlines a list of basic NDP test scenarios, and applies the tester model to each scenario to illustrate how the testing can be executed. Possible future improvements to the model are listed at the end of the chapter. Chapter 4 covers the fundamentals of performance testing, including the basics of traffic generation and real-time measurements. The chapter also describes the ways in which NDP can be used as a tool for setting up performance tests. Chapter 5 provides a conclusion of the report.

The goal of this report is to provide the reader with a good understanding of the basics of NDP, and the requirements for performing protocol and performance tests on a router or a system of routers.

## 2 OVERVIEW OF NDP

This chapter provides an overview of the NDP, to serve as background information before examining the specifics of testing a NDP-capable IPv6 router.

### 2.1 Goals of NDP

NDP is a data link layer protocol<sup>2</sup> used to define the interaction amongst IPv6 network nodes connected by a single link. Specifically, NDP specifies the procedures for allowing a node to resolve a neighboring node's corresponding link layer address from its IPv6 address, to discover all routers residing on the same link, and to automatically configure the node's IPv6 address based on its link layer address. NDP also allows a node to detect the availability or reachability of a neighboring node, to determine the uniqueness of its IPv6 address within its connected link, and to be informed by a router of a better path for a particular traffic destination.

NDP is intended to be run on networks whose network layer protocol is IPv6. NDP is designed such that it may be run over any data link layer types, including multicast links, point-to-point links and non-broadcast multiple access (NBMA) links<sup>3</sup>.

NDP replaces, for IPv6 networks, the functionality of the Address Resolution Protocol (ARP) and various features of version 4 of the Internet Control Message Protocol (ICMPv4) in IPv4 networks.

---

<sup>2</sup> Data link layer protocols are protocols used to describe the interactions amongst network interfaces connected to the same physical link.

<sup>3</sup> Examples of multicast, point-to-point, and NBMA links include Ethernet, Packet Over SONET (POS), and Asynchronous Transfer Mode (ATM), respectively. For more information about the different data link layer types, refer to Chapter 11, "Neighbor Greeting and Autoconfiguration," in [14].

## 2.2 NDP Hosts and Routers

Since NDP is a data link layer protocol, the only other nodes that a NDP node is aware of are those located on directly connected links. Each NDP node is either an IPv6-capable host or router. The difference between a host and a router is that a router has the ability to forward IPv6 packets onto different links and networks on behalf of other nodes, while a host can only act as a packet source and sink. In general, a given link usually contains at least one host and one router.

Figure 2.1 shows a typical Ethernet link and its connected hosts and routers. When NDP is enabled, the hosts and router exchange NDP messages with one another, but not with the Internet or any other network.

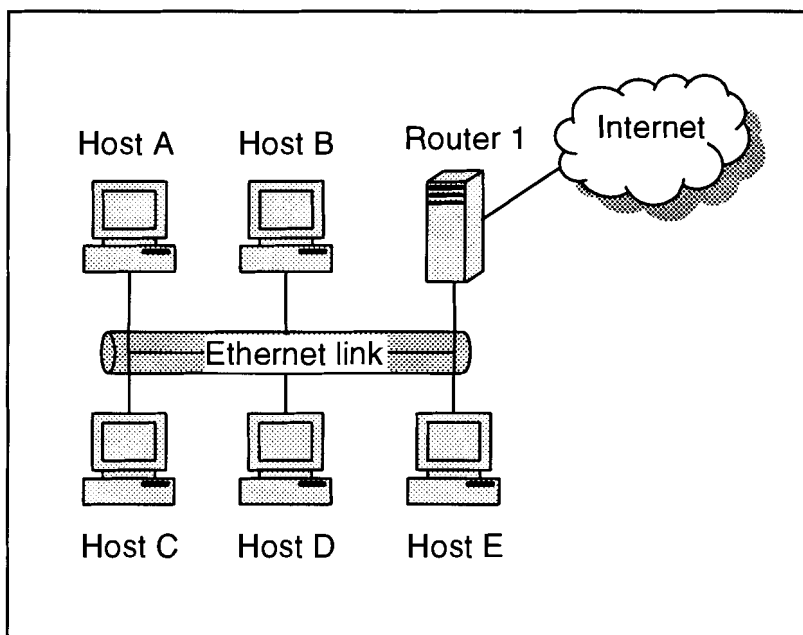


Figure 2.1: An Ethernet link with connected NDP hosts and routers

## 2.3 NDP Messages and Addresses

This section gives a description of the format of NDP messages, the types of addresses found in NDP messages, and the various types of NDP messages. This

summary of NDP messages is provided to facilitate the explanation of the major functionalities of NDP and their corresponding operations in the next section.

### 2.3.1 Message Format

NDP-capable nodes operate by exchanging NDP messages via version 6 of the Internet Control Message Protocol (ICMPv6). A NDP packet is a specific type of ICMPv6 packet. Each NDP packet contains an ICMPv6 header plus NDP-specific fields and options, and is encapsulated by an IPv6 header. The resultant IPv6 packet is, in turn, encapsulated by a link-layer-specific header and trailer. Figure 2.2 shows this hierarchy of headers in a NDP packet.

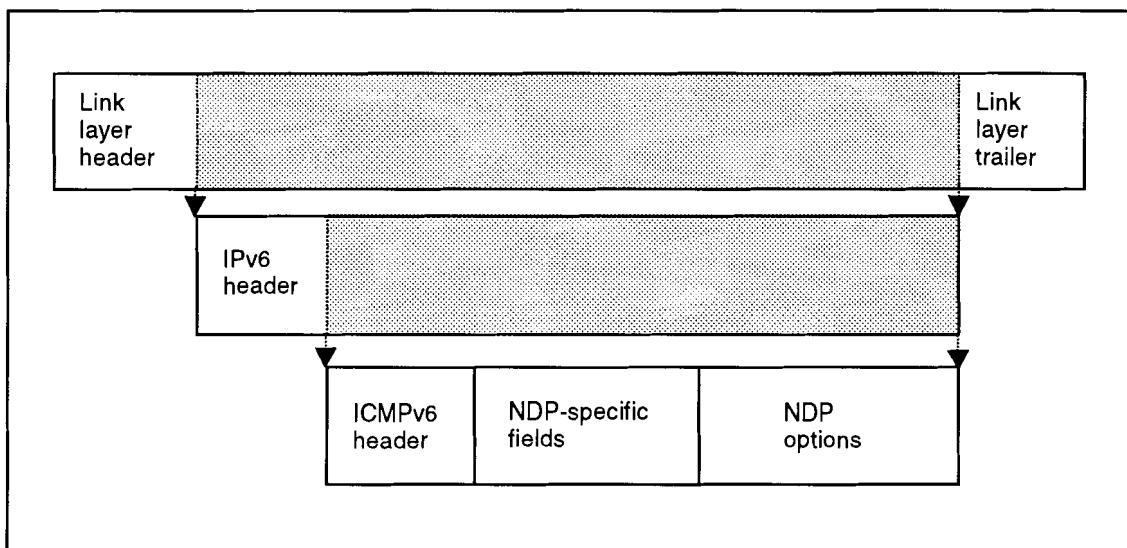


Figure 2.2: NDP packet protocol hierarchy

### 2.3.2 Message Addresses

Each NDP message contains at least two pairs of addresses: the source and destination IPv6 addresses and link layer addresses. These addresses allow NDP messages to be transmitted and received amongst nodes connected to the same link. Some NDP messages also contain a target IPv6 address, in addition to the source and

destination addresses. A target address is used to specify a node or an address in question when NDP nodes query each other about particular addresses or redirect packets to a specific node.

A number of different types of addresses are used in the IPv6 source and destination address fields, including global unicast addresses, link-local addresses, and multicast addresses.

### **2.3.2.1 Global Unicast Addresses**

A global unicast IPv6 address is used to label a single network interface, and has a scope that spans the whole Internet. Each global unicast address must be unique within the Internet, though more than one global unicast address can be associated with the same network interface.

A special global unicast address is the *unspecified address*, which is basically an IPv6 address containing all zeroes. The *unspecified address* is used as the source IPv6 address for certain types of NDP packets that do not require the source of the message to be known.

### **2.3.2.2 Link-Local Addresses**

Link-local addresses are unicast IPv6 addresses that have significance only within a single link. Each network interface connected to a given link must have a link-local address, in addition to one or more global unicast addresses, and each link-local address must be unique to that link.

Link-local addresses are formed by appending the well-known link-local prefix, 0xfe80 (in hexadecimal), to the *Interface Identifier* for each interface. The *Interface*



*Identifier* must be 64 bits long, and must be constructed using the IEEE EUI-64 format<sup>4</sup>. The exact composition of the *Interface Identifier* is specific to each link type, but the link layer address is used in most cases. *Interface Identifiers* have link-only scope, and each *Interface Identifier* must be unique on its connected link.

### 2.3.2.3 Multicast Addresses

Multicast addresses are used by the sender to send messages to multiple nodes simultaneously. NDP messages use a number of well-known multicast addresses with link-only scope, including the *all-nodes multicast address*, the *all-routers multicast address*, and the *solicited-node multicast address*.

The *all-nodes multicast address*, with a value of FF02::1, is used to send packets from one node on a link to all other nodes on the same link. The *all-nodes multicast address* is used by NDP for advertisement messages (refer to section 2.3.3 on Message Types).

On the other hand, the *all-routers multicast address*, with a value of FF02::2, is used to send packets from one node on a link to all other routers (but not hosts) on the same link. The *all-routers multicast address* is used for Router Solicitation (RS) messages.

Unlike the aforementioned multicast addresses, the *solicited-node multicast address* does not have a fixed value, but is, instead, a function of a given unicast address<sup>5</sup>. The *solicited-node multicast address* is used to direct Neighbor Solicitation (NS) messages to specific neighbor targets.

---

<sup>4</sup> For more information on the IEEE EUI-64 format, refer to [17].

<sup>5</sup> For a specification of this function, refer to section 2.7.1 in [10].

### 2.3.3 Message Types

There are five different types of NDP messages:

1. *Neighbor Solicitation (NS)* messages are sent by a node to request information from a specific targeted neighbor. The source address of a NS message is either one of the sender node's global unicast addresses or the *unspecified address*, while the destination address is either the global unicast address of the targeted neighbor or the *solicited-node multicast address* derived from it. NS messages also contain the "Targeted Address" field, used to specify the IPv6 address of the targeted neighbor.

NS messages are used for link layer address resolution (section 2.6), neighbor unreachability detection (section 2.8.1), and duplicate address detection (section 2.8.2).

2. *Neighbor Advertisement (NA)* messages are sent by a node either in response to a NS message or unsolicited when the sender node wants to inform its neighbors of any changes to its link layer address. The source address of a NA message is one of the global unicast addresses of the sender node, while the destination address is either the source address of the corresponding NS message, or the *all-nodes multicast address* if the NS message source address is unspecified or if the NA message is unsolicited. NA messages contain the "Targeted Address" field, which stores the IPv6 address of the sender node, and whose value should match that of the associated NS messages. NA messages also include the "Target Link layer Address" option.

NA messages are used for link layer address resolution (section 2.6), neighbor unreachability detection (section 2.8.1), duplicate address detection (section 2.8.2), and for unsolicited information advertisements.

3. *Router Solicitation (RS)* messages are sent by a host to request for information from routers on the link that are willing to forward packages on its behalf. Such routers are called advertising routers and their interfaces that are connected to the link are called on-link advertising interfaces. The source address of a RS message is either one of the sender node's unicast addresses or the *unspecified address*, while the destination address of the message is the *all-routers multicast address*.

RS messages are used for router and prefix discovery (section 2.4) and stateless address autoconfiguration (section 2.6).

4. *Router Advertisement (RA)* messages are sent by advertising routers to hosts in response to their RS messages. RA messages are also sent unsolicited by advertising routers to all hosts on the link when the advertised information has changed. The source address of a RA message is the link-local address of the router's advertising interface, while the destination address of the message is either the source address from the corresponding RS message, or the *all-nodes multicast address* if the RA message is unsolicited. RA messages also include a number of important options, including the "MTU" option, which specifies the Maximum Transmission Unit (ie. the size of the largest packet) allowable on the link, and the "Prefix" option, which specifies all the network prefixes<sup>6</sup> configured on the link.

---

<sup>6</sup> A prefix refers to the network portion of an IP address.

RA messages, like RS messages, are used for both router and prefix discovery (section 2.4) and stateless address autoconfiguration (section 2.6).

5. *Redirect* messages are sent by routers to hosts to redirect the hosts to use different routers for forwarding data packets destined for specific destinations. The source address of a Redirect message is the link-local address of the sender router's on-link interface, while its destination address is the source address of the data packet that triggered the redirect. Redirect messages also contain the "Targeted Address" field, which stores the IPv6 address of the router that is a more suitable next hop for the traffic, and the "Destination Address" field, which stores the destination IPv6 address of the data packets that should be redirected.

Redirect messages are used solely for the router redirect functionality (section 2.8.3).

## 2.4 Router and Prefix Discovery

Router discovery is performed by each host to determine all the routers that are connected to the same link (i.e. are on-link), and are willing to forward traffic on its behalf. The IPv6 addresses of the discovered routers are used to populate the Default Router List, a data list present on each NDP host, needed for next-hop determination (see section 2.5).

Prefix discovery is performed to determine the list of prefixes that are on-link or can be used for stateless address autoconfiguration (see section 2.7). These prefixes are used to populate the Prefix List (another data list present on each NDP host), which is also used for next-hop determination.

To perform router and prefix discovery, a host broadcasts a RS message to all on-link routers. On receipt of the RS message, each on-link router responds with a RA message containing the list of prefixes known by the router to be on-link or may be used for address autoconfiguration in the “Prefix” option. When the sender host receives a RA message, it extracts the source address of the packet and inserts the address into its Default Router List. It also extracts the prefixes in the “Prefix” option, and inserts them into its Prefix List.

Router and prefix discovery are usually performed when a host connects to a link and when NDP is first enabled on a host. In the case when the configuration on a router changes, the router broadcasts unsolicited RA messages to all its on-link neighbors so that the neighbors can update their caches and lists with the new information.

## **2.5 Next-hop Determination**

Each NDP node maintains a Destination Cache that stores a mapping of the destination IPv6 addresses of recently sent packets to the IPv6 addresses of the next-hop neighbors best-suited for forwarding the packets to their respective destinations. Next-hop neighbors are either on-link routers or the actual destination hosts of on-link destinations.

To perform next-hop determination for a destination IPv6 address, a node first checks the address against the list of on-link prefixes in its Prefix List to determine whether the destination is on-link. If the destination is on-link, then the destination address is the next-hop address. On the other hand, if the destination is not on-link, the sender node selects a router from its Default Router List to use as the next-hop neighbor. If the Default Router List is empty, the sender assumes that the destination is on-link.

Next-hop determination is usually performed the first time traffic is sent to a given destination, and when a next-hop neighbor is determined to be unreachable (see section 2.8.1).

## **2.6 Link Layer Address Resolution**

In order to send or forward IP traffic to a next-hop neighbor, the sender node must know the link layer address of the next-hop neighbor before directing the packets. Each NDP node keeps track of the mapping of the IPv6 address to the link layer address for all neighboring nodes in its Neighbor Cache.

To perform address resolution, a node sends a NS message with the “Target Address” field containing the IPv6 address of the on-link node whose link layer address is being queried (i.e. the target node). When the target node receives the NS message, it responds with a NA message containing its link layer address. Upon receipt of the NA message, the sender node extracts the target link layer address and stores it in its Neighbor Cache.

In the case when the link layer address of a node changes, the node sends unsolicited NA messages to all of its on-link neighbors to inform them of the new address so that their Neighbor Caches can be updated.

## **2.7 IP Address Autoconfiguration**

Since IPv6 addresses are 128 bits long, they are much more time-consuming and error-prone to enter manually in a host's or router's network configuration, when compared to 32-bit IPv4 addresses. NDP offers a mechanism to automatically generate and configure unique IPv6 addresses for each interface on a link.

Before IPv6 addresses can be generated, prefix and router discovery must be performed (see section 2.4). On receipt of the RA messages generated by routers during prefix and router discovery, for each prefix extracted by the sender host in the “Prefix” option of the RA messages, if the “Autonomous Address-Configuration” flag is set in the RA message, then the prefix is used to generate a unique IPv6 address for the receiving interface. Given a specific prefix, a unique IPv6 address can be generated by appending the interface’s *Interface Identifier*. Each interface can have multiple IPv6 addresses associated with it.

This method of generating IPv6 addresses using NDP, as mentioned in the previous paragraph, is called stateless address autoconfiguration. There is another method of generating unique IPv6 addresses, called stateful address autoconfiguration, in reply to: which an external protocol such as the Dynamic Host Configuration Protocol for IPv6 (DHCPv6) is used<sup>7</sup>. The decision of whether to use stateless or stateful address autoconfiguration is determined by the “Managed Address Configuration” flag in the received RA messages, where a set flag indicates that stateful autoconfiguration should be used, and an unset flag indicates that stateless autoconfiguration should be used. However, regardless of whether stateless or stateful address autoconfiguration is used, the resultant address should always be verified for uniqueness on the link using duplicate address detection (see section 2.8.2).

## **2.8 Other Features of NDP**

### **2.8.1 Neighbor Unreachability Detection**

Neighbor Unreachability Detection is a mechanism in NDP for a node to verify the reachability status of each of its neighbors. Each neighbor in a node’s Neighbor

---

<sup>7</sup> The specifics of DHCPv6 is out of the scope of this report, but may be obtained from [9].

Cache has a reachability state associated with it. There are five reachability states, including INCOMPLETE, REACHABLE, STALE, DELAY, and PROBE. Table 2.1 provides a summary of the meaning of each state.

State	Summary
INCOMPLETE	Link layer address resolution has just been invoked for the neighbor
REACHABLE	The neighbor has been confirmed to be reachable
STALE	A predetermined duration has expired without any reachability confirmation for the neighbor
DELAY	A data packet has just been forwarded to the STALE neighbor
PROBE	A NS message has just been sent to verify the reachability of the neighbor, and a response has not yet been received

**Table 2.1: Summary of reachability states for on-link neighbors**

When link layer address resolution is invoked by a node (see section 2.6) for an on-link IPv6 address, a new neighbor entry is added to the Neighbor Cache, with the reachability state set to INCOMPLETE. Once the solicited NA messages for link layer address resolution is received by the sender node, the state for the new neighbor entry becomes REACHABLE. When a period of time, equal to a randomized multiple of the “Reachable Time” field in received RA messages, has elapsed with no reachability confirmation from upper layer protocols or from received solicited NA messages, the state for the neighbor entry becomes STALE. The neighbor entry remains in the STALE state until a data packet is forwarded to the neighbor, at which time the entry goes into the DELAY state. After a fixed duration<sup>8</sup>, if no reachability confirmation is received, the entry goes into the PROBE state and a NS message is sent to seek a direct response from the neighbor. If a corresponding NA message is received, then the neighbor entry goes back to the REACHABLE state; otherwise, the entry is removed from the Neighbor

<sup>8</sup> For the specification that quantifies “a fixed duration,” see section 10 in [12].



Cache. Reachability confirmations generally come from upper layer protocols, and the reception of such confirmations will change the neighbor entry back to the REACHABLE state, with its timeout timer reset. Figure 2.3 shows the state transition diagram for a neighbor's reachability status.

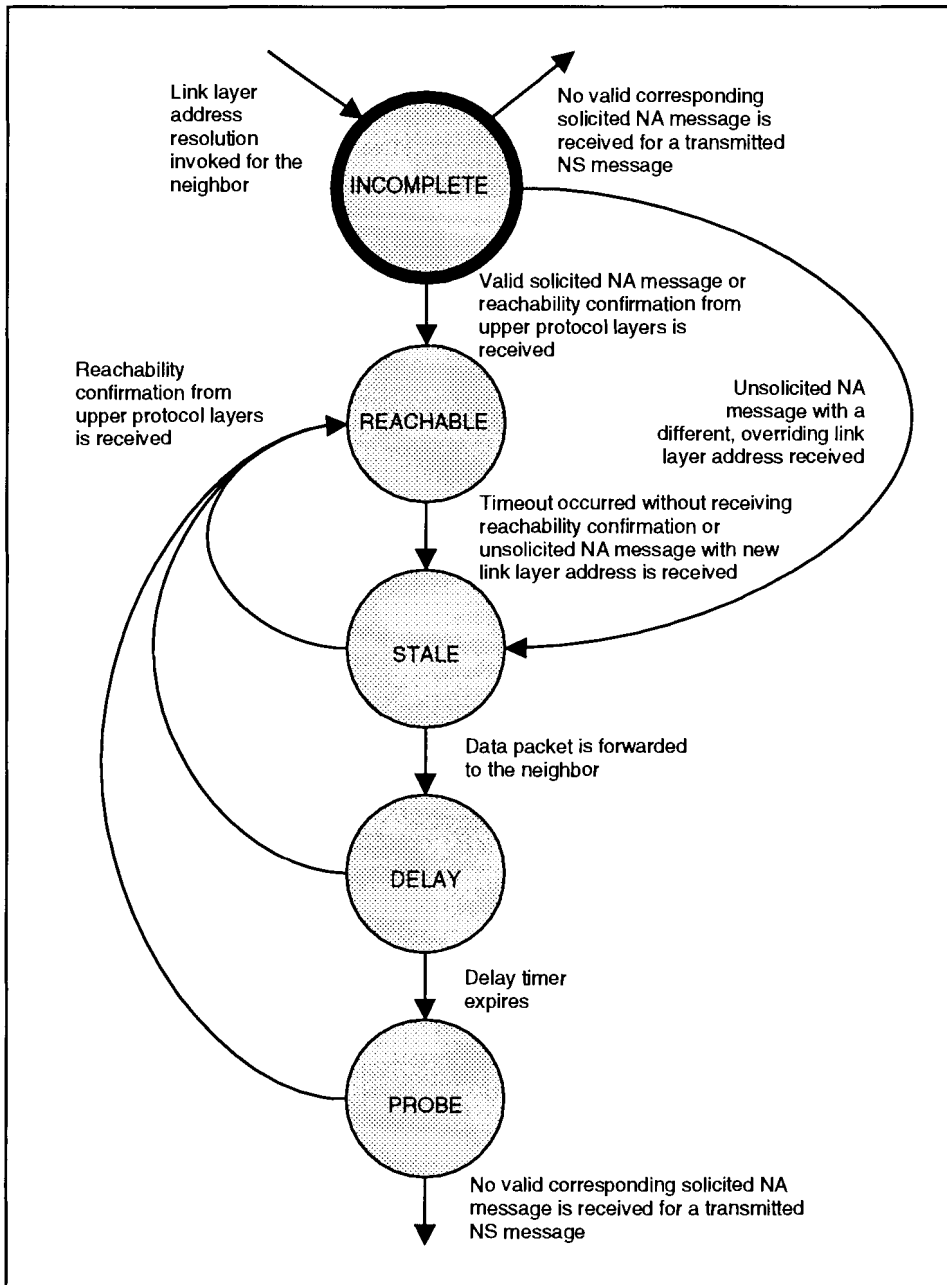


Figure 2.3: State transition diagram for neighbor reachability states

## 2.8.2 Duplicate Address Detection

Duplicate address detection is the mechanism in NDP by which a node determines if one of its newly created IPv6 addresses is unique on its connected link. Duplicate addresses are not allowed on a link.

To perform duplicate address detection for a new address, a node sends a NS message with the *unspecified address* as the IPv6 source address, and the new address as the target address. If no valid corresponding solicited NA messages are received, then the new address is deemed unique and can be assigned to the node's interface. On the other hand, if a valid corresponding solicited NA message is received from a neighbor, indicating that the new address is already used by that neighbor, then the new address is not unique; in such a case, the new address is discarded, and manual intervention is usually required to configure another new address (which must also go through the duplicate address detection process).

Duplicate address detection is performed for newly formed link-local addresses when a link becomes enabled, and for all new global unicast IPv6 addresses created using stateless or stateful address autoconfiguration, or using manual configuration.

## 2.8.3 Router Redirect

The router redirect scenario usually occurs when a host uses a default router as its next-hop router (see section 2.4). If a shorter path through a different router on the link exists for a given destination, or if the destination of the packets is actually another host on the same link, the default router will attempt to redirect the sender host's traffic by sending a Redirect message with the "Target Address" field set to the IPv6 address of the preferred router.

On receipt of the Redirect message, the sender host will update its Destination Cache with the new next-hop address for the given destination.

## **2.9 Current Status of NDP**

Even though NDP was designed to support all data link layer types, due to its requirement of a link layer broadcast or multicast mechanism, non-broadcast-based link layer types like Packet Over SONET (POS), Asynchronous Transfer Mode (ATM), or frame relay mostly use other mechanisms to perform the functionality offered by NDP. Nonetheless, NDP is the neighbor discovery protocol of choice for IPv6-over-Ethernet-type networks. As of the writing time of this report, NDP is fully implemented on most network equipment manufacturers' devices, and is commonly deployed by service providers in IPv6 networks containing Ethernet interfaces, most prominently in Europe and Asia.

## **3 TESTING AN IPV6 ROUTER'S NDP IMPLEMENTATION**

This chapter examines the specifics of testing the NDP implementation of an IPv6 router. In particular, the chapter begins with an overview of the fundamentals of protocol testing, including an explanation of what constitute conformance and scalability testing. Next, a proposed architectural model of a NDP tester is presented, followed by a listing of various basic NDP test scenarios and an illustration of how this tester model can be applied in these tests.

### **3.1 Fundamentals of Protocol Testing**

In general, a protocol is a specification of the procedures and the messages exchanged amongst one or more communicating network entities or nodes. There are two important basic components to testing an implementation of a network protocol such as NDP: conformance testing and scalability testing<sup>9</sup>.

#### **3.1.1 Conformance Testing**

In essence, conformance testing is the verification of a protocol implementation's adherence to the published standards. Conformance to standards is the only way to guarantee that one implementation of a protocol can interoperate with other implementations, or that one company's network equipment can work together with its customer's or competitor's equipment. Such interoperability capabilities are important in

---

<sup>9</sup> Conformance and scalability testing are the most common types of protocol testing requested by test engineers. Other types of protocol testing include functional testing, interoperability testing, and negative testing. Since these types of tests are not as common, they will not be covered by this report. More details on all types of protocol testing can be found in [4].

today's large global networks, which are made up of different types of equipment from companies all over the world.

The basis of conformance testing is to provide a pre-determined inputs or stimuli to an implementation, and then verify that its output is as expected according to the standards. For NDP, the main standard was published by the Internet Engineering Task Force (IETF) as Request For Comments (RFC) 2461, "Neighbor Discovery for IPv6" [12]. As evident from the overview of NDP in chapter 2 of this report, NDP mostly involves the exchange and processing of messages. Therefore, a test for the conformance of a NDP implementation can be performed by sending various NDP messages (i.e. the input) to the implementation, and then verifying the subsequent messages received (i.e. the output) from the implementation.

### **3.1.2 Scalability Testing**

As today's networks are made up of hundreds of thousands of nodes and links, it is essential that a router, including all of its protocol implementations, is able to handle a large volume of input, to store the millions of sets of network information it needs to properly forward traffic, and to process and generate the corresponding amount of output. Scalability testing involves simulating a real network environment with hundreds of thousands of nodes, and testing a router's ability to function normally in such an environment. Scalability testing also verifies that a router's performance does not deteriorate significantly as the volume of protocol traffic, and, hence, the amount processing required of the router, increases.

To test a NDP router for scalability, the router can be placed in a simulated environment containing a large number of neighbors, and then its performance in sending and processing all necessary NDP messages can be measured.

## 3.2 A Model of a NDP Tester

In this section, the general goals of a protocol tester are outlined, followed by a detailed explanation of a proposed model of a NDP protocol tester.

### 3.2.1 Goals of a Protocol Tester

A protocol tester's job is to simulate the operation of one or more network nodes against the router or the system under test, and to verify that the protocol implementation in the system under test is functioning as expected. A protocol tester should also make it easy to determine when a system under test is behaving correctly or incorrectly, and to trace through the series of procedures being performed or messages being exchanged to determine the cause of any problems that occur.

In theory, a router's protocol implementation can be tested using other routers or hosts. However, routers or hosts are normally designed for performance and efficiency, and, thus, do not have extra resources left over to support the tracing, logging, and general flexibility needed in a dedicated tester. In addition, to simulate the high-density environment for scalability testing, hundreds of thousands of routers and hosts would be required; such a large test bed is too expensive to maintain, and is not feasible.

For dedicated testers to be able to provide a realistic test environment that is easy to set up and to use for testing against real routers, the protocol implementation within a tester should vary from that within a real router or host in several ways:

1. *A tester should provide a way to create and store a large number of instances of a protocol simulation quickly and easily.* This capability is necessary in order to perform scalability tests with as little test equipment as possible. For example, instead of requiring five hundred routers, all with NDP enabled and hooked up onto the same data link, one tester port simulating five hundred

NDP routers can be used. Also, the test protocol simulations should be designed in a way such that all or large portions of the simulations can be created and configured all at once.

2. *A tester should provide a way to simulate large surrounding networks and topologies on a single port.* Because today's Internet consists of large networks and topologies, in order for router tests to be as realistic and relevant as possible, they should be operating under such an environment. For any router that is to be deployed within the Internet, it not only has to conform to the standards for any protocols that it runs, but it also has to be able to handle all the protocol operations correctly while forwarding traffic for large networks at close to its full bandwidth capability. Ideally, one tester port should be able to simulate as many hosts, routers and networks as possible in order to be relevant and efficient at the same time.
3. *A tester's protocol implementation does not need to perform all the processing and store all the data required of a protocol implementation in a real router or host.* A tester should be able to simulate many protocol instances in large networks and topologies using a small number of test ports, which are necessary for realistic and scalable tests. Since a tester is not a real router, and is not expected to forward traffic, it doesn't need to process and store all the routing information that a real router uses to make forwarding decisions; as a result, a tester should be able to use fewer resources than a real router.
4. *Multiple protocol simulations on a tester should pretend to have done all the required protocol interactions with each other, without actually doing them.*  
When simulating more than one protocol instance on a port, a tester should

allow its protocol instances to share data with one another without the need for protocol packets to be sent over any physical links. In this manner, the protocol instances can be assumed to have performed all protocol operations, exchanged all the necessary messages, and be completely up-to-date with one another without actually having performed the work. This capability also allows protocol implementations on a tester to be more resource-efficient than that on a real router.

5. *Protocol simulations should allow protocol operations to be explicitly triggered and controlled.* In a real router, once a protocol instance is configured and enabled, all protocol operations should be carried out by the router automatically with little or no intervention needed by the router operator. On the other hand, a protocol implementation within a tester should provide explicit control of the protocol operations, in addition to the automatic control of a real router. When testing a real router using a tester, a test engineer can start by having the protocol instances on the tester operate automatically, while monitoring the logs and trace messages for anything out of the ordinary. When the test engineer wishes to test or to trace through a problem in a particular part of the protocol on the router, such as the reception of an unexpected or a malformed protocol message, the test engineer can explicitly trigger various operations on the tester to gauge the reaction of the router.

A dedicated tester offers its value by allowing a test engineer to create as large and complicated a setup and using as little equipment and as little configuration time as possible, all the while being adequately thorough in testing every aspect of the operation of a router or a system of routers.



### 3.2.2 Static Representation of the Model

With the goals of a tester listed in the previous section in mind, a model of a NDP protocol tester is presented in this section. The model is illustrated using the Unified Modelling Language (UML)<sup>10</sup>.

Figure 3.1 shows a UML class diagram, or static representation, of the NDP protocol tester model. The main component within the model is the `NdpHostPool` class, which is a representation of a group of simulated NDP hosts within the same link. Each `NdpHostPool` object is uniquely identified within the system by an integer handle. The number of simulated NDP hosts contained within a `NdpHostPool` object is a user-configurable value. Also, to allow fast set-up of the simulated NDP hosts, all NDP-related properties, like whether NDP is enabled, are shared by all hosts and specified only once per host pool, regardless of the number of hosts in each pool. In addition, all addresses, including link layer addresses (L2 addresses in the diagram), and *Interface Identifiers*, which must be unique to each NDP host, are specified as a patterned range of addresses using a “first address” and a modifier (i.e. a value to be added or subtracted from the current address to obtain the next address); for example, for a host pool containing 10 simulated hosts, if the “first address” value is 1 and the modifier is positive 3, then the addresses for the 10 simulated hosts are 1, 4, 7, 10, 13, 16, 19, 22, 25, and 28. As for the link-local and global unicast addresses for the simulated hosts within a pool, they can be obtained by appending each simulated host’s *Interface Identifier* to the link-local prefix or the global prefixes listed in the `PrefixList` object.

---

<sup>10</sup> For a specification of UML, refer to [13].

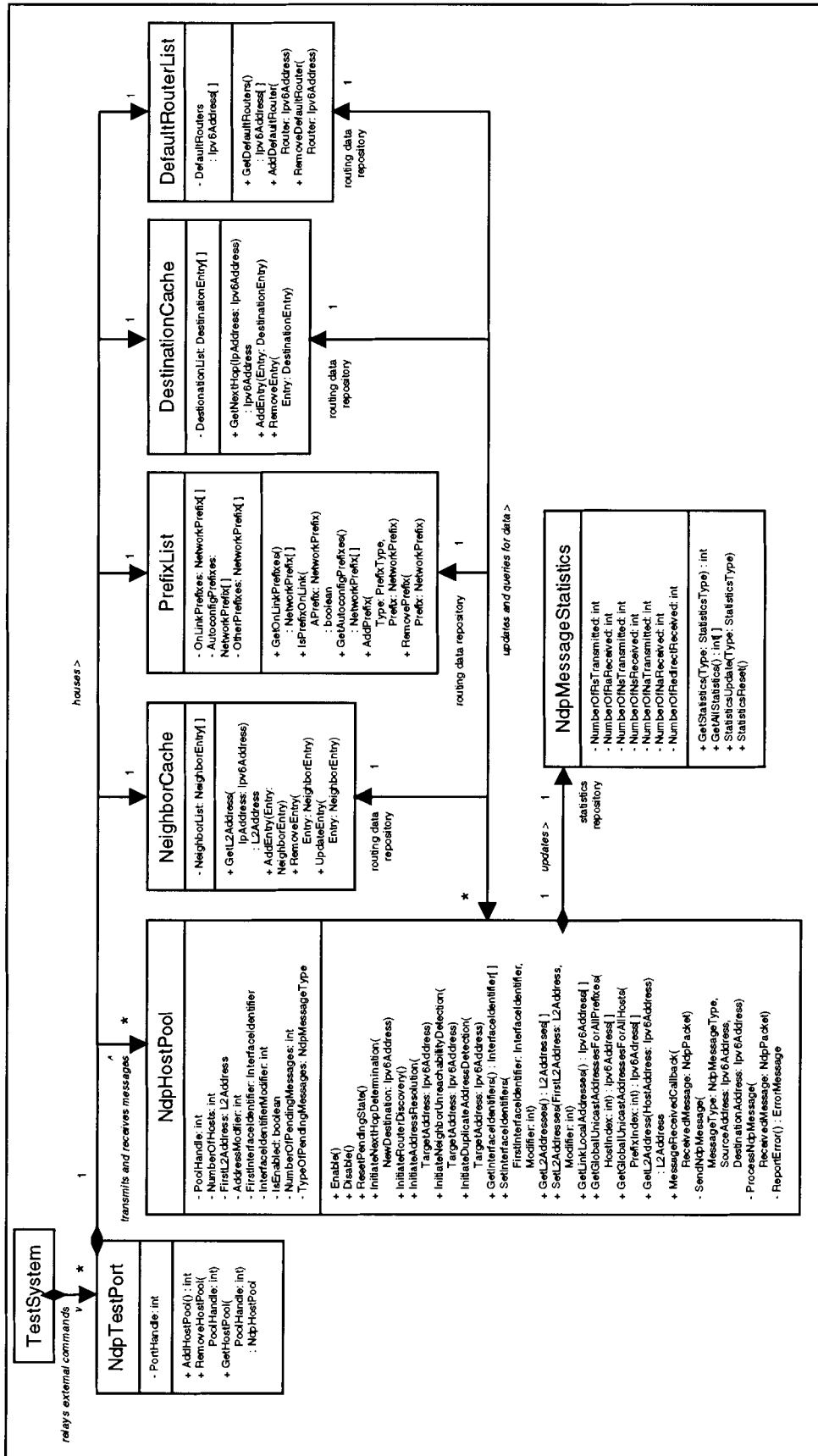


Figure 3.1: Class diagram of the NDP test model

The public member operations in the `NdpHostPool` class include methods like `InitiateRouterDiscovery` and `InitiateAddressResolution`, which allow the user to explicitly trigger NDP protocol operations for specific tests, and `GetInterfaceIdentifiers` and `SetInterfaceIdentifiers`, which allow the user and other system components to access the host pool's parameters in a controlled manner. Each of these operations is applied to every simulated NDP host in the host pool. For instance, a call to `InitiateRouterDiscovery` will start the router discovery process on all the simulated hosts in the pool, such that one RS message is sent and one RA message is expected to be received on behalf of each simulated host. While the host pool is waiting for all the RA messages, the `NumberOfPendingMessages` member variable will indicate the number of simulated hosts yet to receive a RA message, and the `TypeOfPendingMessages` member variable will indicate that the host pool is currently expecting RA messages. A new NDP operation should not be allowed to start on the host pool until the current operation is complete, which is denoted by the `NumberOfPendingMessages` variable having a value of zero. If the user determines that a fault has occurred in the router being tested, such that the expected NDP messages will not be received by the `NdpHostPool`, then he can explicitly reset the `NumberOfPendingMessages` and `TypeOfPendingMessages` variables using the `ResetPendingState` member operation. After the reset has been triggered, the user will be free to start any new NDP operations on the NDP host pool.

Each `NdpHostPool` object also contains a `NdpMessageStatistics` object, which is used to store the counts of the number of each type of NDP message transmitted and received by the `NdpHostPool` object since the `NdpHostPool` object was enabled. The message counts can also be reset to zero by calling the `StatisticsReset` member

operation within the NdpMessageStatistics object. Other member operations include methods to update and query the message counts.

NdpHostPool objects are created by and stored within NdpTestPort objects. Each NdpTestPort object may contain zero or more NdpHostPools, all of which are assumed to be residing on the same data link. Each NdpTestPort object also contains a NeighborCache object, a PrefixList object, a DestinationCache object, and a DefaultRouterList object. These four cache and list objects store the data obtained by the simulated hosts when exchanging NDP messages with and performing NDP operations against a router. To save on the amount of resources needed to store these objects, these caches and lists only contain data with regards to the router under test, and not that with regards to the simulated hosts themselves. The information contained within these four objects is common to and is shared by each NdpHostPool object within the same NdpTestPort object.

The public member operations for the NeighborCache, PrefixList, DestinationCache, and DefaultRouterList classes mainly consist of methods to add, modify, remove, and query for the information contained within each class.

The TestSystem class represents all the other components in the tester that are not related to NDP. Also, the TestSystem class provides the interface between the user and the NDP-related objects. Each TestSystem object may contain zero or more NdpTestPorts, and is responsible for relaying the commands and messages between the NDP-related objects and the user or other parts of the tester.

### 3.2.3 Dynamic Representation of the Model

The previous section examined the static representation of the NDP protocol tester model. In this section, the dynamic representation of the model is presented and analyzed.

Figure 3.2 shows a UML sequence diagram for the NDP protocol tester model. Specifically, the diagram illustrates the operations amongst instances of classes like `NdpTestPort` and `NdpHostPool`, and external actors such as the user or the router. The diagram also shows when each class is activated via the vertical bars underneath the class name, the synchronous commands issued by each class via the solid arrows, and the asynchronous messages passed amongst the classes and the external actors via the stick arrows.

The sequence diagram in Figure 3.2 illustrates the sequence of messages and events that occur when the user initiates a command to start the router discovery and IPv6 address autoconfiguration operation on a particular NDP host pool. In the diagram, the “User” actor triggers the start of the sequence by initiating the command to the test system, which acts as the interface between the user and the rest of the NDP-related components. The test system then finds and invokes the `InitateRouterDiscovery` command in the `NdpHostPool` object that the user specifies using the pool object’s unique handle.

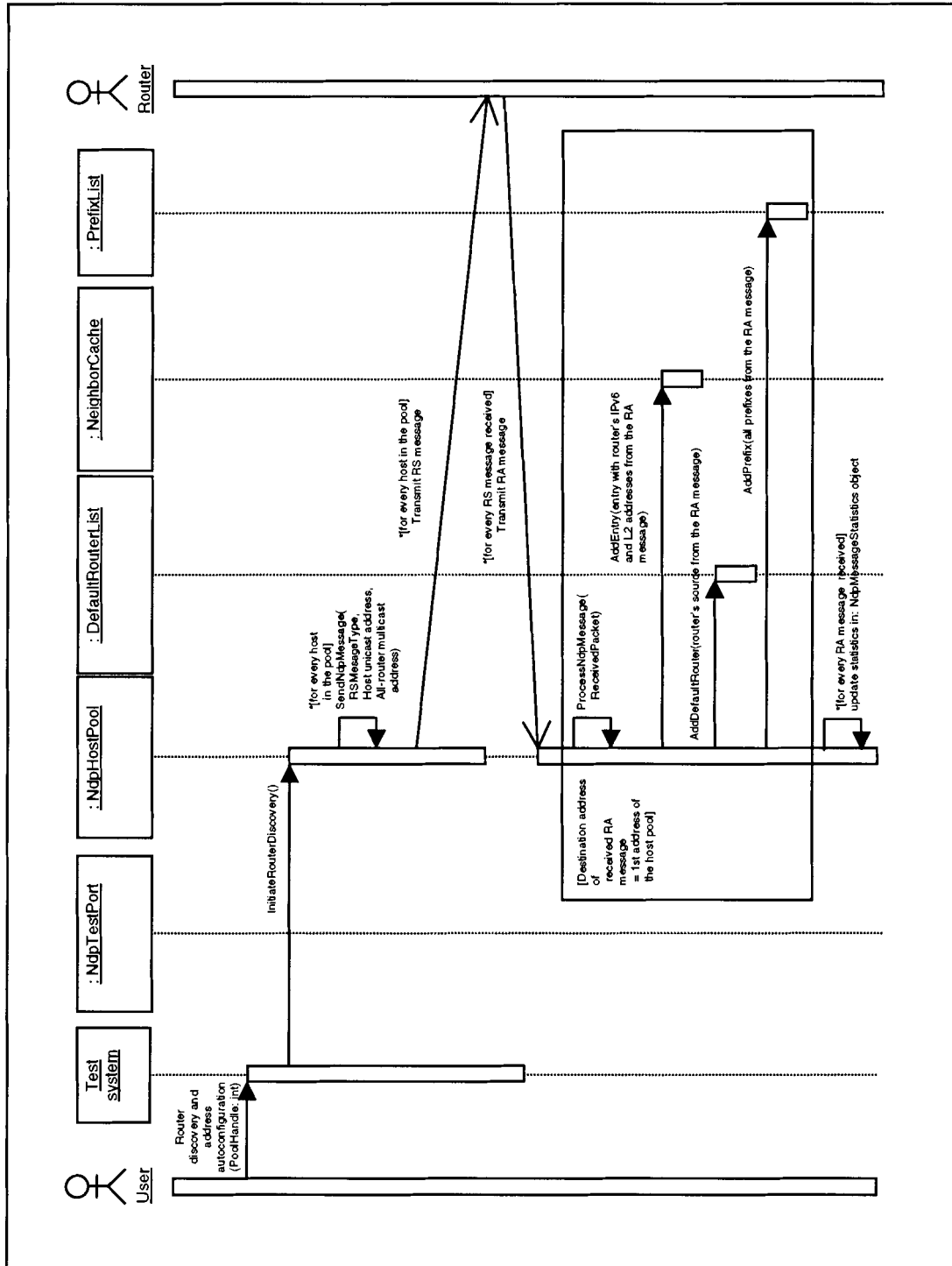


Figure 3.2: Sequence diagram for router discovery and address autoconfiguration

Next, the NdpHostPool object builds and sends a RS message for each simulated NDP host in the pool, with the source address of the message being one of the simulated host's global unicast addresses, and the destination address being the *all-routers multicast address*. The NdpHostPost object should also update its NdpMessageStatistics object with the number of RS messages that was sent. At this point, the NdpHostPool object is in a pending state, waiting for the RA message for each corresponding RS message to be sent by the router under test. When the RA message corresponding to the RS message that was sent on behalf of the first simulated host in the pool is received<sup>11</sup> (i.e. the destination address of the received RA message equals the "first address" of the host pool's address range), the NdpHostPool object invokes its own ProcessNdpMessage method. The ProcessNdpMessage method will, in turn, update the NeighborCache object with the source IPv6 and link layer addresses of the router, the DefaultRouterList object with the source IPv6 address of the router, and the PrefixList object with the advertised prefixes, all of which are extracted from the received RA message. The NeighborCache, DefaultRouterList, and PrefixList objects that are updated should be the ones belonging to the same NdpTestPort as the NdpHostPool object. Since all the simulated NDP hosts within the host pool are assumed to reside on the same data link, the information advertised by the router under test in its RA messages must be the same for each RS message that was sent<sup>12</sup>; as a result, only one of the RA messages needs to be processed and used to update the associated caches and lists, thereby allowing the tester to save on the amount of processing resources needed, and also allowing the tester's performance to remain relatively similar regardless of the number of hosts being simulated within the pool object.

---

<sup>11</sup> An alternative method of implementing this functionality is to process the first RA message that was received by the NdpHostPool object instead of the RA message whose address matches the "first address" of the NdpHostPool object.

<sup>12</sup> This fact can be tested using a large number of host pools, each with only one simulated host.

In addition to the procedures for processing the received RA messages, each received RA message must also be counted in the NdpHostPool's NdpMessageStatistics object, regardless of which simulated host within the pool the RA message was destined. However, since the amount of processing needed to increment a statistics count is almost negligible, the requirement to process every RA message in this case should not be a detriment to the tester's performance.

Figure 3.2 illustrates a typical usage scenario for the NDP protocol tester model. The chain of events is often started by the user invoking a command to perform a specific test. A NdpHostPool object is generally the recipient of the command, at which point it often builds and sends the required NDP messages to the router under test. If and when the corresponding NDP messages sent by the router are received, the corresponding test port's NDP-related caches and lists are updated with the new data from the messages, and the message statistics counts are incremented accordingly. Though not indicated in the sequence diagram, if, at any point, the chain of events is interrupted by unexpected behaviour from the router, the proper error information should be displayed to the user so that the user can use the information to pinpoint the problem.

### **3.3 Test Scenarios**

In the previous section, a model for a NDP protocol tester was presented. A static representation, in the form of a UML class diagram, outlined the basic components of the model. A dynamic representation, in the form of a sequence diagram, examined the interactions amongst those components.

In this section, the NDP protocol tester model is applied to various test scenarios to demonstrate how the tester can be used to perform protocol conformance and scalability testing on a router or a system of routers. This application of the model is



illustrated using a series of sequence diagrams similar to that in Figure 3.2. The test scenarios include the following:

1. initial connection, router discovery, prefix discovery, and address autoconfiguration
2. address resolution, neighbor unreachability detection and duplicate address detection
3. next-hop determination, and
4. router redirect

Each scenario focuses on the interaction of one NdpHostPool object with the rest of the system and with the router being tested. Together, these scenarios make up the fundamental protocol operations of NDP, and target many of the NDP-related functionality of a router that needs to be tested.

### **3.3.1 Initial Connection, Router Discovery, Prefix Discovery and Address Autoconfiguration**

The test scenario presented in the sequence diagram in Figure 3.3 shows the series of events and interactions during the initial connection phase of a NdpHostPool object. It is similar to the scenario presented in Figure 3.2 in the previous section, which illustrates the events and interactions during router discovery. Prefix discovery and address autoconfiguration are performed as a result of both scenarios.

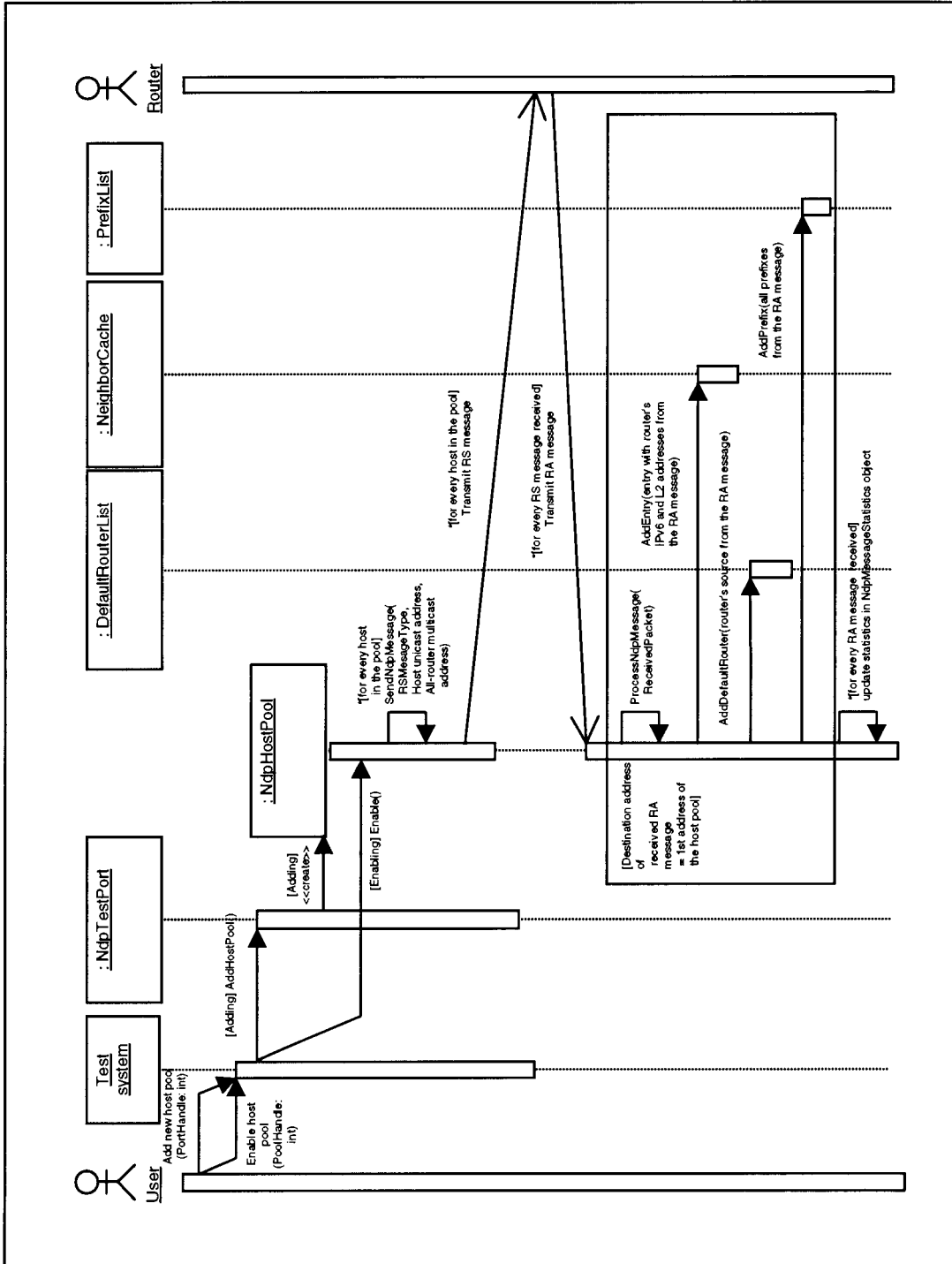


Figure 3.3: Sequence diagram for initial connection

The only difference between the scenarios in Figure 3.2 and Figure 3.3 is the way in which the user triggers the start of the events. For the scenario in Figure 3.2, the user explicitly invokes the command to start the router discovery process. For the scenario in Figure 3.3, the user initiates the connection either by adding a new NDP host pool, or by enabling an existing one. When adding a new host pool, the test system executes the AddHostPool command in the corresponding NdpTestPort object, which then creates the new NdpHostPool object and stores it internally. On the other hand, when enabling an existing host pool, the test system finds and directly sends the command to the correct NdpHostPool object itself.

Once the command reaches the NdpHostPool object, the rest of the events and interactions in the initial connection scenario are identical to that in the router discovery scenario. The NdpHostPool object sends a RS message on behalf of each simulated NDP host, and waits for the corresponding RA messages to be sent by the router. Only one received RA message is processed and is used to update the NeighborCache, DefaultRouterList, and PrefixList objects on the associated NdpTestPort, while all RS messages transmitted and all RA messages received are updated in the NdpHostPool's NdpMessageStatistics object.

The expected behaviour from the router under test in both scenarios is that for every RS message sent by the NdpHostPool object, the router responds with a corresponding RA message in a timely manner<sup>13</sup>. The RA message that is processed should pass the message validation procedures described in section 6.1.2 in [12]. Also, the RA message should contain the correct information reflecting the source IPv6 and link layer addresses of the router, and the prefixes that are on-link or may be used for

---

<sup>13</sup> The specifications for quantifying “a timely manner” can be found in [12].

address autoconfiguration. The tester is responsible for verifying that the router has performed all of the above, and for reporting any discrepancies that occur.

### **3.3.2 Address Resolution, Neighbor Unreachability Detection and Duplicate Address Detection**

Figure 3.4 shows the sequence diagram for address resolution, neighbor unreachability detection, and duplicate address detection. The chain of events that occurs for each of these three scenarios is similar, but not identical. Also, for each of these three operations, different source and destination addresses are used in the NS messages that are sent to the router under test.

The sequence begins when the user triggers the address resolution, neighbor unreachability or duplicate address detection command in the test system, specifying both the unique handle of the host pool and the target address for the operation. The target address for both address resolution and neighbor unreachability detection should be one of the addresses in the NeighborCache. On the other hand, the target address for duplicate address detection may be an address in the NeighborCache or any address that the user wishes to test.

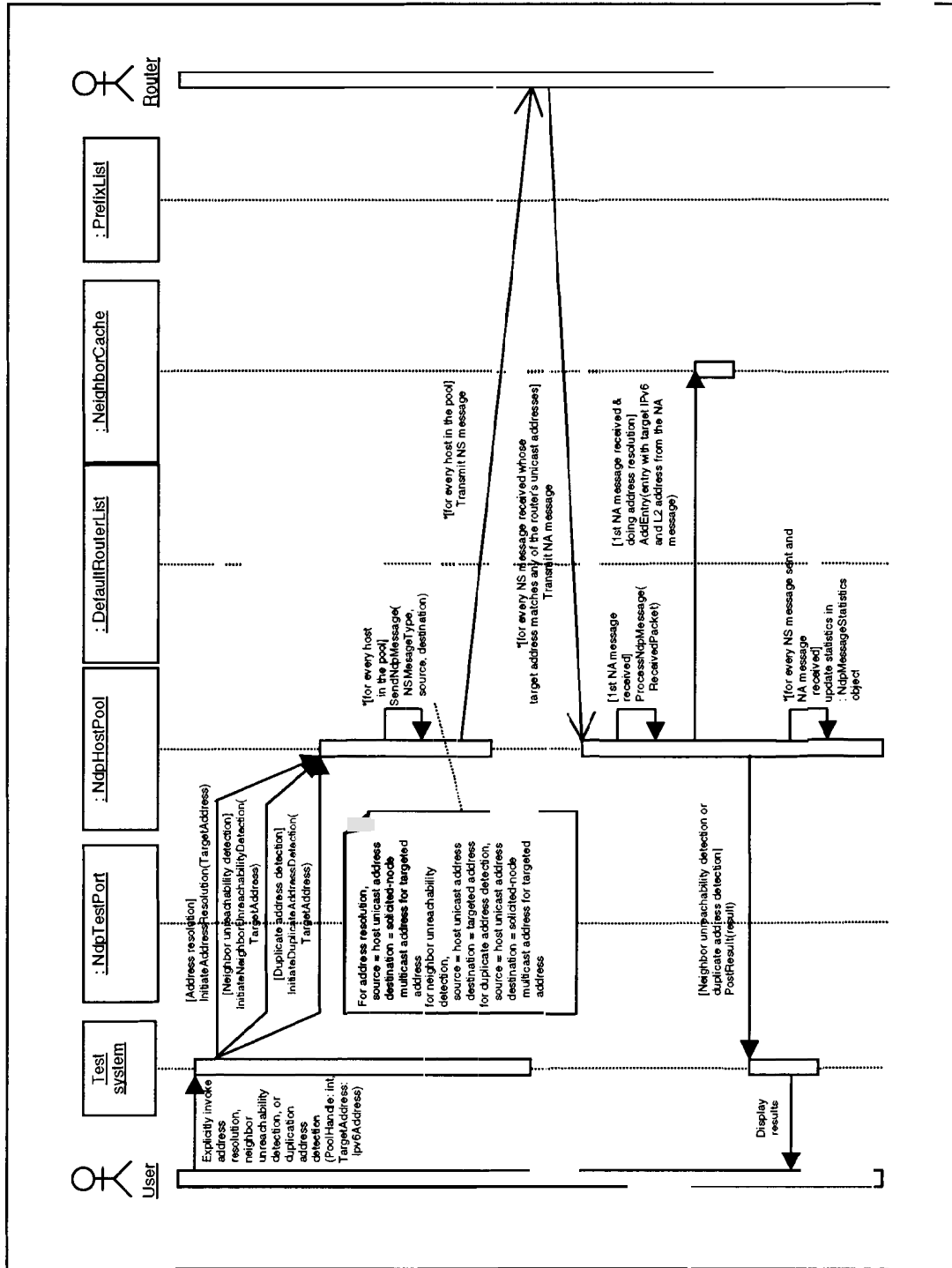


Figure 3.4: Sequence diagram for address resolution, neighbor unreachability detection, and duplicate address detection

Upon receipt of the user command, the test system finds and relays the command to the specified NdpHostPool object. For each NDP host that it simulates, the NdpHostPool object builds a NS message and sends it to the router. For address resolution, the source address of the NS message is one of the simulated host's global unicast addresses, and the destination address is the *solicited-node multicast address* of the target address. For neighbor unreachability detection, the source address of the message is one of the simulated host's global unicast addresses, and the destination address is the target address. For duplicate address detection, the source address of the message is the *unspecified address*, while the destination address is the *solicited-node multicast address* of the target address. Upon successful transmission of the NS messages, the associated count in the NdpMessageStatistics object is updated.

At this point, the NdpHostPool object goes into a pending state, and waits for the corresponding NA messages to be sent by the router. The TypeOfPendingMessages member variable is set to the NS message type, and the NumberOfPendingMessages is set to the number of expected NA messages yet to be received. If address resolution is being performed, all NA messages are expected to be sent by the router, and the NdpHostPool object will remain pending until either all messages are received, or the user explicitly resets the pending state.

On the other hand, if either neighbor unreachability detection or duplicate address detection is being performed, NA messages are expected to be received only if the target address matches one of the router's own addresses; otherwise, they should not be received. For this scenario, like that for address resolution, the NdpHostPool object can still exit out of the pending state if either all messages are received, or the user explicitly resets the pending state. Unlike address resolution, if after a fixed duration of time, the NdpHostPool object is still in the pending state, it will automatically

exit out of the pending state. The exact amount of time in this fixed duration may be a predetermined constant in the system, or it may be configurable by the user<sup>14</sup>.

Regardless of the cause, as the NdpHostPool exits the pending state, it will post the outcome of the neighbor unreachability detection or the duplicate address detection process to the user. An affirmative or a negative result may be desired, depending on the target address that was supplied.

The expected behaviour for the router in this test scenario is that it responds to the NS messages with corresponding NA messages in a timely manner, provided that it is the neighbor being targeted in the test. If it is not the targeted neighbor, it is expected to not respond with any NA messages. Also, any NA messages sent by the router should pass the message validation procedures as described in section 7.1.2 in [12], and contain the router's link layer address in the "Target link-layer address" option.

### **3.3.3 Next-hop Determination**

Under normal circumstances, the scenario for next-hop determination does not directly involve any interactions with the router under test. Instead, the tester relies on the existing information that was gathered in previous NDP operations in the DestinationCache, PrefixList, DefaultRouterList, and NeighborCache objects. Message exchange and interaction with the router only comes into play when the information needed by the tester cannot be found in the existing data in the cache and list objects.

---

<sup>14</sup> This fixed duration of time for the NdpHostPool object to automatically exit out of the pending state should be longer than the timeout for a pending NA message, as defined in [12].

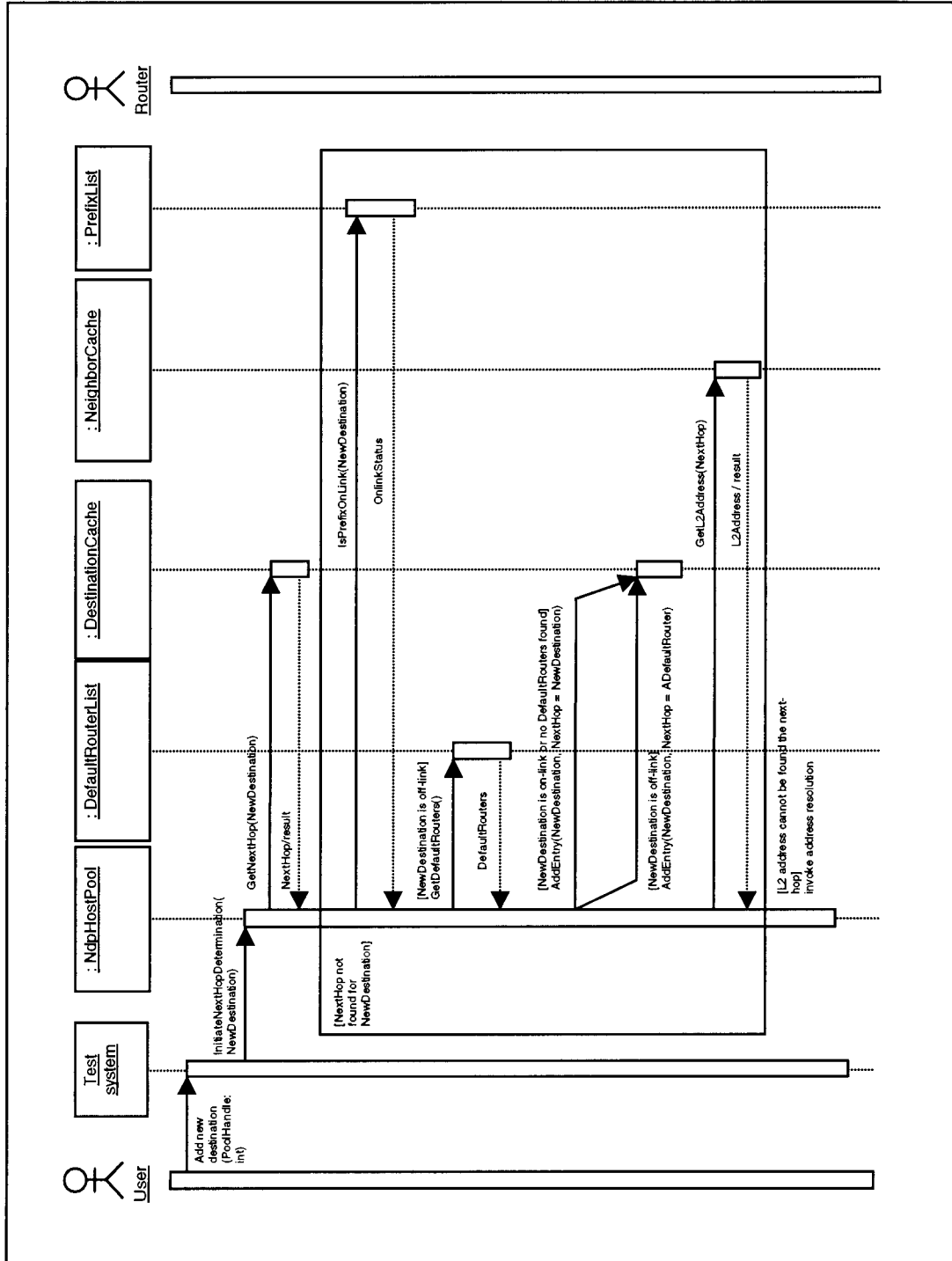


Figure 3.5: Sequence diagram for next-hop determination



The scenario diagram for next-hop determination is shown in Figure 3.5. Like previous scenarios, the chain of events starts off with a command from the user. In this case, the command is to add a new destination for sending traffic. Upon receipt of the user command, the test system finds and invokes the `InitiateNextHopDetermination` command in the specified `NdpHostPool` object.

The `NdpHostPool` object first checks with the `DestinationCache` object to see if the new destination indicated by the user has already been cached in the object, and, if so, to retrieve the corresponding next-hop address. If the new destination cannot be found within the `DestinationCache`, the `NdpHostPool` object queries the `PrefixList` object to see if the new destination is on-link. If the new destination is on-link, then the next-hop address is set to the address of the new destination. If the new destination is off-link, then the `NdpHostPool` object retrieves the address of a default router from the `DefaultRouterList` object, and sets it as the next-hop address. If no default routers are found in the `DefaultRouterList` object, then the new destination is assumed to be on-link.

Once the next-hop address is obtained, the `NdpHostPool` object queries the `NeighborCache` to retrieve the link layer address corresponding to the next-hop address. When the query is successful, and a link layer address is found, the new destination is ready to be used for sending traffic. However, if the query is unsuccessful, the `NdpHostPool` object will have to invoke the address resolution procedure, as described in section 3.3.2. The only difference between the subsequent series of interactions that occur and the interactions shown in the address resolution sequence diagram in Figure 3.4 is the fact that there should be no command being triggered by the user and the test system. Instead, the chain of events starts at the `NdpHostPool` object.

Assuming that the address resolution step is not needed, there is no real expected behaviour for the router under test in this next-hop determination scenario.

However, the router is expected to have kept the NeighborCache, DefaultRouterList and PrefixList objects up-to-date via solicited or unsolicited NA and RA messages, such that the link layer address retrieved at the end of the scenario actually corresponds to the new destination and can be used to send traffic without any errors.

### **3.3.4 Router Redirect**

Figure 3.6 shows the sequence diagram for the router redirect test scenario. Unlike the previous scenarios, the router redirect scenario does not start with an explicit user command. Instead, it is generally triggered as a result of the user sending data packets from the tester to the router under test when the router is not the ideal next-hop neighbor for the destination of the traffic. This condition is true when another router that is closer to the destination reside on the same link, or when the destination is actually a neighboring node on the link.

If the user does not explicitly provide a destination link layer address for the data packets, the tester may ask the NdpTestPort object to extract a valid next-hop address for the traffic destination from its DestinationCache, and the corresponding link layer address from its NeighborCache. If a valid next-hop cannot be found, then the next-hop determination process (as described in the previous section) is invoked.

If the user does not explicitly provide a source link layer address for the data packets, the tester will use the link layer address of the NDP simulated host whose global unicast IPv6 address matches the data packets' source IPv6 address.

Once the link layer addresses are determined, the user can start transmitting the data packets to the router via the test port. Since, in this scenario, the router is not the ideal next-hop for the given destination, it should send a Redirect message to the simulated host acting as the source of the packets.

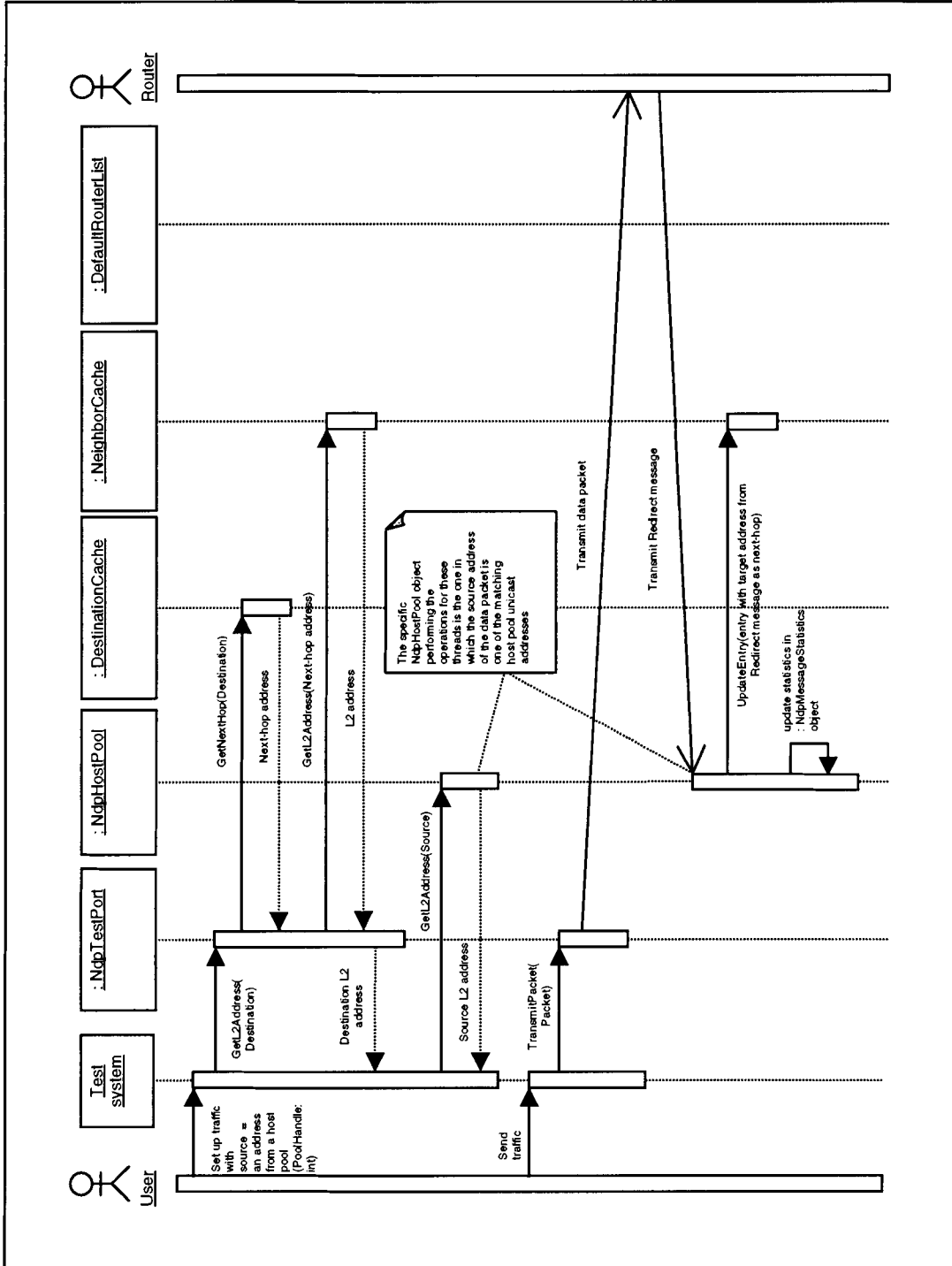


Figure 3.6: Sequence diagram for router redirect

When the Redirect message is received, the NdpHostPool object containing the source simulated host updates the DestinationCache object associated with its parent NdpTestPort object with the new next-hop address for the destination. The NdpHostPool object also increments the received Redirect message count in its NdpMessageCount member object.

Assuming that a better next-hop neighbor exists, the router under test is expected send a Redirect message in a timely manner after it receives the data packets. The NdpHostPool object does not go into a pending state since it is not aware of the need to redirect the traffic until it actually receives the Redirect message. The Redirect message sent by the router should pass the message validation procedures listed in section 8.1 of [12]. Also, the tester should verify that the Redirect message contains the address of the better next-hop neighbor in its “Target Address” field, and the destination address of the data packets that triggered the redirect in its “Destination Address” field.

### **3.4 Application of the Test Scenarios**

The test scenarios presented in the previous section can be applied to perform both conformance tests and scalability tests.

#### **3.4.1 Conformance Tests**

To perform conformance tests on the router under test for verification of adherence to the protocol standards, the test scenarios can be run using one test port and one NdpHostPool object containing only one simulated NDP host. Using this minimal setup, the router’s basic protocol functionality is tested without any extra complications like requiring the router to process and send multiple NDP messages at once. As long as the router provides the expected behaviour in servicing the simulated

host, the router's protocol implementation and processing algorithm can be deemed to adequately follow the published standards.

At any point during a test scenario, if anything unexpected occurs in the tester's interaction with the router, the number of contributing factors is reduced in this simple configuration, making it easier for the user to locate the source of and the reason behind the unexpected behaviour.

### **3.4.2 Scalability Tests**

To perform scalability tests on the router under test for the test scenarios, multiple test ports, each with multiple NdpHostPool objects containing many simulated NDP hosts should be used. Ideally, before starting any scalability tests, the corresponding conformance test should have already been performed with the expected behaviour verified. Otherwise, should an error occur during the test scenario, the large number of simulated hosts involved and messages being sent and received will only serve to complicate the situation, making it harder to analyze the problem.

When executing the test scenarios using multiple simulated hosts, the tests should start with a small number of test ports and simulated host pools (each with a small number of simulated hosts) such that success is expected. Then, the tests should be repeated with a gradually increasing number of ports and simulated hosts until an error or any unexpected behaviour occurs. Examples of errors include expected NDP messages not being sent by the router, or are not sent within the message timeout period<sup>15</sup>. The resulting number of simulated hosts used can be considered to be beyond the scalability limit for the particular router under test. Variations of the scalability tests can be created by increasing only one or two of the parameters (i.e. fixing the number of

---

<sup>15</sup> Default NDP message timeout periods are defined in [12].

ports and increasing the number of NdpHostPool objects per port, or just increasing the number of simulated hosts per host pool) to gauge the router's reaction under different circumstances.

Further scalability tests can be performed by blasting the router with non-NDP-related traffic while the NDP test scenarios are being executed. These types of tests are relevant since routers are expected to handle traffic and protocol messages sent by multiple sources simultaneously.

In all of the aforementioned scalability test scenarios, the tester is expected to be able to handle much higher scalability numbers than the router under test, as the amount of processing that it is required to do is significantly less than for the router.

### **3.5 Future Extensions to the Model**

The NDP protocol tester model presented in this chapter is designed to perform the most basic functionality needed to test the NDP-related behaviour of a router. As such, the model can be extended to cover more details in each test scenario or to cover other test scenarios.

The following list contains features that can be added to the existing tester model in order to widen the scope of the tests that can be performed:

- a timing module and event scheduler can be introduced to handle the message timeouts, retransmissions, and list entry expirations as described in [12] and [18]
- more details can be added to the model to describe how events and interactions are traced, and how logs, error messages, and statistics are displayed to the user

- more capability can be added to the model in testing a system of routers, as the model is currently designed to handle only one router on the data link connected to the test ports (for example, the pending state within a NdpHostPool object only expects advertisements from one router, since having more than one router would increase the number of pending messages by a factor of  $n$ , with  $n$  being the number of real routers on the link)
- the ability to simulate NDP routers (in NdpRouterPool objects), in addition to NDP hosts, can be included to allow the model to test real hosts as well as real routers

## **4 USING NDP AS A TEST CONFIGURATION TOOL**

The previous chapter explored the ways in which the NDP implementation on a tester can be used for testing a router's conformance to the NDP protocol standards and its scalability in the presence of large NDP-enabled IPv6 networks. In this chapter, the ways in which a tester's NDP implementation can be used to facilitate the configuration of performance tests involving traffic generation and real-time measurements are examined.

In addition, this chapter shows how the model of a NDP tester presented in the previous chapter can be expanded to include the components of performance testing. The resulting model presents a more complete picture of the elements essential to a dedicated tester of a router.

### **4.1 Fundamentals of Performance Testing**

The performance of a router placed in the large and high-traffic networks of today's Internet is of particular concern to router manufacturers and Internet Service Providers (ISP's). Accordingly, a router's performance should be thoroughly tested and validated before the router can be deployed in a live network.

The two basic components of performance testing include traffic generation and real-time measurements. In essence, traffic generation is the process of blasting a router or system under test with packets at one end, and real-time measurements is the process of collecting and analyzing the packets forwarded by the router at the other end.



### **4.1.1 Traffic Generation**

To fully test the capability of a router, different types of traffic should be sent to the router to gauge the router's ability to handle various traffic scenarios. For a dedicated tester, a number of parameters should be considered when generating test traffic, including the number of different flows of traffic to be generated<sup>16</sup>, the size of the packets being generated, the rate and distribution at which the packets are sent, and the contents of the packets being generated.

Each of these parameters may be affected by the result of protocol exchanges between the tester and the router. In terms of NDP, since NDP is primarily used for exchanging neighbor addresses and address-related information, the data gathered by NDP can be used to automatically populate the address fields of generated traffic such that the resultant packets would be forwarded by the router. The automatic setup of valid protocol-negotiated addresses in data packets is very important for tests involving large networks, plenty of nodes, and a constantly fluctuating environment.

### **4.1.2 Real-time Measurements**

Real-time measurements involve gathering the packets forwarded by the router, tracking various statistics on the received packets, and making analysis on the packets and the statistics based on previous knowledge of the nature of the traffic that was sent to the router. The most common types of statistics gathered on traffic forwarded by a router include packet and octet counts, packet loss, traffic throughput, latency, and jitter.

In general, protocol implementations on testers do not directly affect the process of making real-time measurements on router traffic. Nevertheless, for NDP, running NDP operations and, in particular, scalability test scenarios against a router while

---

<sup>16</sup> a flow of traffic is generally distinguished by the source of the traffic, the destination of the traffic, and a set of service agreements or handling procedures from the network carrying the traffic

generating and measuring large volumes of traffic may affect the results of the real-time measurements. In fact, such a test scenario of sending both control (i.e. protocol) and data packets simultaneously to a router may be closer to the actual types of traffic the router is expected to handle in real network. Also, the address information gathered by NDP may be used to identify the flows of traffic that are sent to the router, and for which statistics should be tracked.

## **4.2 Configuration Scenarios**

As mentioned, the ways in which NDP can be used to aid performance testing mostly involve using the information gathered by NDP to set the content of generated traffic. This section presents two ways in which the address information learned by NDP can be used to automatically fill in the address fields of data packets.

### **4.2.1 Automatic Interface Discovery**

In order for a router to forward any packets, the packets themselves should contain valid link layer and IPv6 addresses that allow the router to recognize and process the packets. In particular, for most link layer types, the packets forwarded to the router should contain the router's link layer address, or some derivation of it, as the link layer destination address.

NDP router discovery and address resolution procedures allow a tester to automatically learn the router's unicast IPv6 and link layer addresses. Thus, once these operations have been executed, and the test port has been updated with the most current address information, the port's Destination Cache, Default Router List, and Neighbor Cache can be queried to retrieve the applicable next-hop (i.e. the router's) link layer address for a given packet destination. The next-hop determination and router redirect scenario diagrams, in Figure 3.5 and Figure 3.6 respectively, illustrate the way

the test system retrieves the relevant information from the NDP caches and lists that can be used to fill in the link layer destination address of the data packets.

#### **4.2.2 Automatic Address Setup**

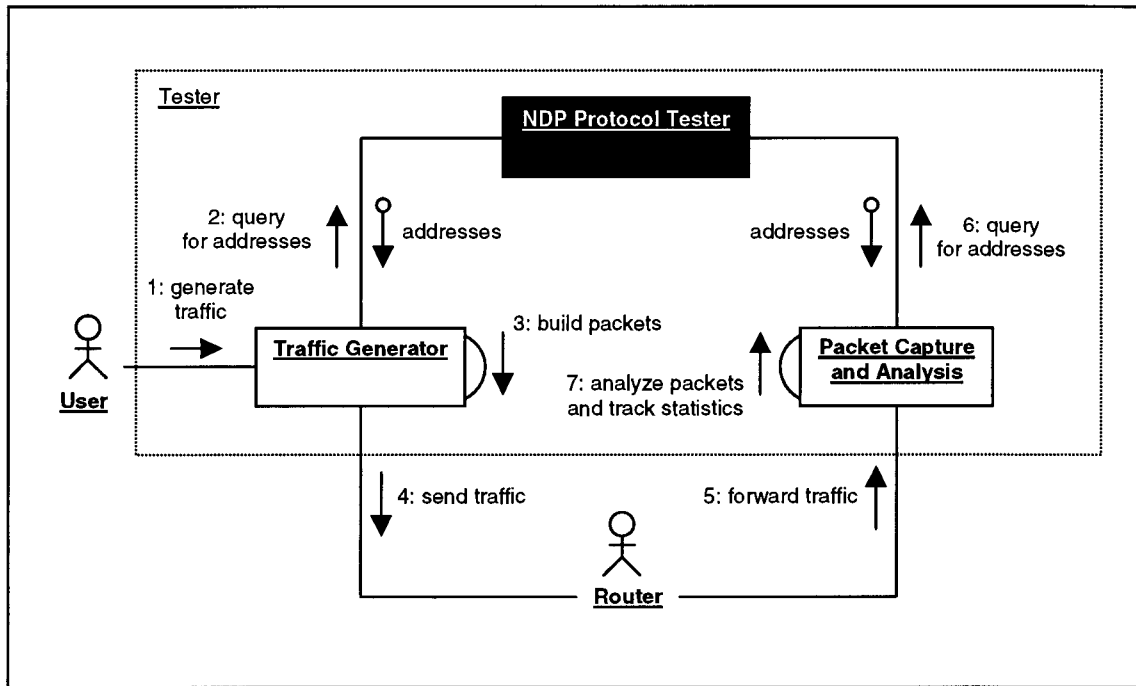
A different test configuration problem that can be solved by NDP involves the address autoconfiguration process. In particular, since IPv6 addresses are long and hard to set up, using the automatically generated unicast addresses for simulated NDP hosts as the source or destination IPv6 addresses in data packets saves the user the trouble of manually editing the addresses. Having the packets addressed as such emulates traffic being sent from one simulated host through the router to another simulated host.

To perform automatic address setup, the address autoconfiguration procedures should be carried out first so that valid and unique IPv6 addresses can be automatically assigned to each simulated host. The IPv6 addresses for a simulated host can be produced by appending its *Interface Identifier* to all the prefixes marked for autoconfiguration in the associated test port's Prefix List. After address autoconfiguration is done, traffic can be generated using the simulated hosts' new IPv6 addresses as either the source or destination address, depending on the intended path of the traffic.

### **4.3 Extensions to the Protocol Tester Model for Performance Testing**

In the previous chapter, a model for testing the NDP implementation on a router was examined. However, the associated test scenarios that were presented mostly focussed on protocol conformance and scalability testing.

The NDP protocol tester model can be extended to provide for the automatic interface discovery and automatic address setup scenarios. Figure 4.1 shows a collaboration diagram with an extended version of the tester model, and the interactions amongst the components of the model, the user, and the router under test.



**Figure 4.1: Collaboration diagram for performance testing scenarios**

The new extended model of the tester contains two new components (or classes): the traffic generator component and the packet capture and analysis component. The third component, the NDP protocol tester, basically consists of all the elements of the protocol tester model presented in the previous chapter (see Figure 3.1); it is illustrated as a black box that can be used by the other tester components to query for NDP-negotiated address information. Together, the traffic generator, the packet capture and analysis, and the NDP protocol tester make up the more complete model of a router tester.

The operations listed in Figure 4.1 are focussed on the two scenarios, automatic interface discovery and automatic address configuration, in which the tester's NDP implementation is used as a configuration tool. Before the sequence of events begins, the NDP protocol tester is expected to have fully communicated with the router and stored all the necessary information in its data caches and lists. The process begins when the user indicates a desire to generate traffic for testing and analyzing the performance of the router. When the traffic generator receives the command, it queries the NDP protocol tester for the addresses of the simulated hosts and their next-hops. Then, it builds the data packets using the returned information (and possibly other configuration values from the user), and sends the packets out to the router.

Assuming that the router's NDP operations had proceeded as expected and that the information it gave to the NDP protocol tester is correct, the router is expected to forward the packets to the tester's receiving ports. When the packets are received, the packet capture and analysis module at the receiving port queries the NDP protocol tester to make sure the packet addresses are expected, and then performs statistics tracking and analysis on the packets. The results of the analysis should be presented back to the user to paint a picture of the performance of the router.

## 5 CONCLUSION

In this report, the ways in which an IPv6 router's NDP capabilities can be tested are explored.

NDP is a data link layer protocol used in IPv6 networks for dynamically exchanging address information amongst neighbors connected on the same link. NDP runs on top of both ICMPv6 and IPv6, and makes use of both multicast and unicast transmissions for message exchange. The primary operations of NDP include router and prefix discovery, traffic next-hop determination, link layer address resolution, IP address autoconfiguration, neighbor unreachability detection, duplicate address detection, and router redirect. All of these operations are performed by the exchange of NS, NA, RS, RA, and Redirect messages amongst neighbors.

Protocol testing on a router involves checking for the router's conformance to published protocol standards, and for the router's ability to scale as the number of neighbors and the size of the surrounding networks increase. Ideally, a dedicated tester is used to perform the testing on the router. Dedicated testers are designed to provide simulations of different network environments and network entities for various test scenarios, and to implement detailed monitoring and reporting of the router's behaviour during the tests.

To test a router's conformance to the NDP standards, a dedicated tester can be used to simulate neighboring NDP hosts and to perform each of the primary NDP operations with the router. Most of the NDP operations expect the router to respond with a specific NDP message in a timely manner when given a particular stimulus such as the

receipt of another NDP message. The message sent by the router should pass the message validation procedures as defined in the standards, and contain the information sought by the tester. To test the router's scalability, the same conformance tests can be performed, but with the number of simulated hosts and the size of the surrounding networks gradually increasing.

When testing the performance of a router by sending it data packets and by capturing the packets forwarded by the router, the NDP capabilities of a tester can be used as an address configuration tool. In particular, the address information gathered during the router discovery, address resolution and address autoconfiguration processes, including the link layer address of the router and the generated IPv6 addresses of the simulated hosts, can be used to set the address fields of the data packets such that they would be forwarded by the router. These methods of automatic neighbor discovery and automatic address configuration allow the user of the tester to avoid having to search for and enter the IPv6 addresses into the traffic configuration manually.

A model is used in this report to illustrate one approach to the design of a dedicated tester. The model is also used to demonstrate the series of events and interactions that can occur when a tester performs the typical NDP operations against a router, and the way the router's behaviour can be verified. As the protocol standards and the nature of the Internet evolve, the model can be extended to support future NDP testing requirements.

## REFERENCES

- [1] Agilent Technologies, "Test Automation for Network Routing Devices," White Paper, December 2001.  
<http://advanced.comms.agilent.com/n2x/docs/whitepapers/pdfs/feature.pdf> [September 25, 2004].
- [2] Agilent Technologies, "Testing Forwarding Performance," Application Notes, June 2004.  
<http://advanced.comms.agilent.com/n2x/docs/appnotes/enterprise/TestForwPerf/5989-1093EN.pdf> [September 25, 2004].
- [3] Agilent Technologies, "True Router Performance Testing," Application Notes, May 2000.  
<http://advanced.comms.agilent.com/n2x/docs/appnotes/routing/TrueRouterPerformanceTesting.htm> [September 25, 2004].
- [4] Agilent Technologies, *Journal of Internet Test Methodologies*, Edition 2.6, September 2004.  
<http://advanced.comms.agilent.com/n2x/docs/journal/Journal.htm> [September 25, 2004].
- [5] Conta, A. and S. Deering, "Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification," Request For Comments 2463, Internet Engineering Task Force, December 1998.  
<http://www.ietf.org/rfc/rfc2463.txt?number=2463> [June 3, 2004].
- [6] Crawford, M., "Transmission of IPv6 Packets over Ethernet Networks," Request For Comments 2464, Internet Engineering Task Force, December 1998.  
<http://www.ietf.org/rfc/rfc2464.txt?number=2464> [June 3, 2004].
- [7] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification," Request For Comments 2460, Internet Engineering Task Force, December 1998.  
<http://www.ietf.org/rfc/rfc2460.txt?number=2460> [June 3, 2004].
- [8] Donzé, François, "IPv6 Address Autoconfiguration," Volume 7, Issue 2, The Internet Protocol Journal, June 2004.  
[http://www.cisco.com/en/US/about/ac123/ac147/current\\_issue/ipv6\\_autoconfig.html](http://www.cisco.com/en/US/about/ac123/ac147/current_issue/ipv6_autoconfig.html) [August 11, 2004].
- [9] Droms, R., J. Bound, B. Volz, T. Lemon, C. Perkins, and M. Carney, "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)," Request For Comments 3315, Internet Engineering Task Force, July 2003.  
<http://www.ietf.org/rfc/rfc3315.txt?number=3315> [June 3, 2004].
- [10] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture," Request For Comments 2373, Internet Engineering Task Force, July 1998.  
<http://www.ietf.org/rfc/rfc2373.txt?number=2373> [June 3, 2004].
- [11] Holub, Allen I., "Allen Holub's UML Quick Reference," Version 2.1.0, April 2004.  
<http://www.holub.com/goodies/uml/> [October 26, 2004].



- [12] Narten, T. E. Nordmar, and W. Simpson, "Neighbor Discovery for IP Version 6 (Ipv6)," Request For Comments 2461, Internet Engineering Task Force, December 1998. <http://www.ietf.org/rfc/rfc2461.txt?number=2461> [May 5, 2004].
- [13] Object Management Group, "UML 2.0 Superstructure Specification," August 2003. <http://www.omg.org/docs/ptc/03-08-02.pdf> [October 26, 2004].
- [14] Perlman, Radia. *Interconnections, Second Edition, Bridges, Routers, Switches, and Internetworking Protocols*. Reading, Massachusetts: Addison Wesley Longman, Inc., 2000.
- [15] Sun Microsystems, "IPv6 Neighbor Discovery," IPv6 Administration Guide. <http://docs.sun.com/db/doc/817-0573/6mqc65bb6?a=view> [May 7, 2004].
- [16] Telecom Lab Italia, "The Neighbor Discovery Protocol," Technical Report. <http://www.ngnet.it/e/ipv6proto/ipv6-proto-6.php> [May 7, 2004].
- [17] The Institute of Electrical and Electronics Engineer (IEEE), "Guidelines for 64-bit Global Identifier (EUI-64) Registration Authority," March 1997. <http://standards.ieee.org/db/oui/tutorials/EUI64.html> [June 3, 2004].
- [18] Thomson, S. and T. Narten, "IPv6 Stateless Address Autoconfiguration," Request For Comments 2462, Internet Engineering Task Force, December 1998. <http://www.ietf.org/rfc/rfc2462.txt?number=2462> [June 3, 2004].