
**HP Computer Systems
Training Course**

**— HP-UX Network
Administration II**

Student Workbook

**Version A.03
H1690S Student
Printed in USA 06/99**

Notice

The information contained in this document is subject to change without notice.

HEWLETT-PACKARD PROVIDES THIS MATERIAL "AS IS" AND MAKES NO WARRANTY OF ANY KIND, EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. HEWLETT-PACKARD SHALL NOT BE LIABLE FOR ERRORS CONTAINED HEREIN OR FOR INCIDENTAL OR CONSEQUENTIAL DAMAGES (INCLUDING LOST PROFITS IN CONNECTION WITH THE FURNISHING, PERFORMANCE OR USE OF THIS MATERIAL WHETHER BASED ON WARRANTY, CONTRACT, OR OTHER LEGAL THEORY).

Some states do not allow the exclusion of implied warranties or the limitations or exclusion of liability for incidental or consequential damages, so the above limitations and exclusion may not apply to you. This warranty gives you specific legal rights, and you may also have other rights which vary from state to state.

Hewlett-Packard assumes no responsibility for the use or reliability of its software on equipment that is not furnished by Hewlett-Packard.

This document contains proprietary information which is protected by copyright. All rights reserved. No part of this document may be photocopied, reproduced or translated to another language without the prior consent of Hewlett-Packard Company.

UNIX UNIX is a registered trademark in the United States and other countries, licensed exclusively through X/Open Company Limited.

Worldwide Customer Educational Services
Professional Services Organization
100 Mayfield Avenue
Mountain View, CA 94043 U.S.A.

© Copyright 1998 by the Hewlett-Packard Company

Contents

Overview

Course Description	1
Student Performance Objectives	1
Student Profile and Prerequisites	3
Curriculum Path	3
Agenda	3

Module 1 — Introduction

Objectives	1-1
1-1. SLIDE: What You Already Know	1-2
1-2. SLIDE: New Components in This course	1-5
1-3. SLIDE: Coverage in Depth	1-8
1-4. SLIDE: Tasks You Will Be Able to Perform	1-10

Module 2 — Serial Line Protocols

Objectives	2-1
2-1. SLIDE: Point-to-Point Links	2-2
2-2. SLIDE: Point-to-Point Protocol Features	2-4
2-3. SLIDE: Point-to-Point Protocol Overview	2-6
2-4. SLIDE: PPP Format	2-8
2-5. SLIDE: Phases of a PPP Link	2-12
2-6. SLIDE: LCP Protocol	2-14
2-7. SLIDE: IPCP Protocol	2-16
2-8. SLIDE: PPP and SLIP	2-18
2-9. SLIDE: Serial-Line Throughput Considerations	2-20
2-10. SLIDE: PPP Configuration Files	2-22
2-11. SLIDE: PPP Configuration Overview	2-24
2-12. SLIDE: Create Device Files for the Serial Ports	2-26
2-13. SLIDE: Increase the Number of IP Tunnels	2-28
2-14. SLIDE: Configuring Your Modem	2-29
2-15. SLIDE: Configuring Outbound Connections	2-31
2-16. SLIDE: Configuring Inbound Connections	2-34
2-17. SLIDE: Testing the Outbound Connection	2-36
2-18. SLIDE: Migrating from SLIP to PPP	2-38
2-19. SLIDE: IP Routing over PPP	2-40
2-20. SLIDE: Security over PPP	2-43
2-21. LAB: Point-to-Point Protocol	2-46

Module 3 — Troubleshooting of Common TCP/IP Problems

Objectives	3-1
3-1. SLIDE: The TCP/IP Network Protocol Stack	3-2

3-2.	SLIDE: Connection Type	3-5
3-3.	SLIDE: Recognizing a Bad Physical Connection	3-7
3-4.	SLIDE: MTU-Size	3-9
3-5.	SLIDE: IP Addressing Faults	3-11
3-6.	SLIDE: Network and Broadcast Addresses	3-13
3-7.	SLIDE: Subnetting Errors	3-14
3-8.	SLIDE: Routing Faults	3-16
3-9.	SLIDE: Reverse Name Resolution Failure	3-18
3-10.	SLIDE: Multihomed Hosts	3-20
3-11.	LAB: Finding and Resolving Network Faults	3-22
3-12.	SLIDE: The <code>syslog</code> Subsystem	3-25
3-13.	SLIDE: Computer Facilities	3-27
3-14.	SLIDE: <code>syslog</code> Computer Levels	3-29
3-15.	SLIDE: <code>syslog</code> Message Recipients	3-31
3-16.	SLIDE: <code>/etc/syslog.conf</code>	3-32
3-17.	LAB: System Logging Daemon— <code>syslogd</code>	3-33
3-18.	SLIDE: The Network Tracing and Logging Subsystem	3-35
3-19.	SLIDE: The Network Tracing and Logging Daemon	3-37
3-20.	SLIDE: The Network Tracing and Logging Formatter	3-40
3-21.	SLIDE: Filter Configuration File	3-42
3-22.	SLIDE: Using <code>nettladm</code>	3-44
3-23.	LAB: Using <code>nettl</code> , <code>netlfmt</code> , and <code>nettladm</code>	3-46

Module 4 — Routing Tables and Dynamic Routing Protocols

Objectives	4-1	
4-1.	SLIDE: Routing Tables	4-2
4-2.	SLIDE: IP Routing Algorithm	4-4
4-3.	SLIDE: IP Routing	4-6
4-4.	SLIDE: Static Routing	4-8
4-5.	SLIDE: Proxy ARP	4-10
4-6.	SLIDE: Setting Up a Proxy ARP Client	4-12
4-7.	SLIDE: ICMP Redirect	4-14
4-8.	SLIDE: Router Discovery Protocol	4-15
4-9.	SLIDE: Overview of Dynamic Routing	4-17
4-10.	SLIDE: Autonomous Systems	4-19
4-11.	SLIDE: Vector Distance Protocol	4-21
4-12.	SLIDE: Link State Routing Protocol	4-23
4-13.	SLIDE: <code>gated</code>	4-24
4-14.	SLIDE: Routing Information Protocol	4-25
4-15.	SLIDE: Example of RIP and EGP	4-27
4-16.	SLIDE: Common Configuration Clauses	4-28
4-17.	SLIDE: RIP Definition	4-30
4-18.	SLIDE: Point-to-Point Interfaces	4-32
4-19.	SLIDE: Troubleshooting <code>gated</code>	4-33
4-20.	LAB: RIP Configuration—Default Rerouting	4-35
4-21.	SLIDE: OSPF Concepts	4-38
4-22.	SLIDE: Example of OSPF Configuration	4-40
4-23.	SLIDE: OSPF Internal Router	4-41
4-24.	SLIDE: OSPF Area Border Router	4-43
4-25.	SLIDE: Exporting Routes	4-44

Module 5 — Managing the NIS Environment

Objectives	5-1
5-1. SLIDE: The Network Information Service	5-2
5-2. SLIDE: Generating NIS Maps with <code>make</code>	5-4
5-3. SLIDE: The Automounter and NIS	5-6
5-4. SLIDE: Overview of Creating New NIS Maps	5-8
5-5. SLIDE: Step 1: Editing the NIS Source Files	5-9
5-6. SLIDE: Step 2a: Editing <code>/var/yp/Makefile</code>	5-10
5-7. SLIDE: Step 2b: Editing <code>/var/yp/Makefile</code>	5-12
5-8. SLIDE: Step 3: Creating New NIS Maps	5-14
5-9. SLIDE: Propagating NIS Maps with <code>ypxfr</code>	5-16
5-10. SLIDE: <code>ypxfr</code> Examples	5-17
5-11. LAB: NFS Automount Using NIS Maps	5-19

Module 6 — NIS+ Administration

Objectives	6-1
6-1. SLIDE: Overview of NIS+	6-2
6-2. SLIDE: Advantages of NIS+ over NIS	6-3
6-3. SLIDE: Disadvantages of NIS+	6-5
6-4. SLIDE: Structure of the NIS+ name space	6-6
6-5. SLIDE: Structure of a NIS+ Domain	6-7
6-6. SLIDE: How NIS+ Information is Stored and Propagated	6-9
6-7. SLIDE: NIS+ Tables	6-11
6-8. NIS+ Authentication and Authorization	6-12
6-9. NIS Compatibility Mode	6-14
6-10. Planning the NIS+ name space	6-16
6-11. Setting Up the Root Master Server	6-18
6-12. Populate the NIS+ Tables on the Master Server	6-21
6-13. Adding Administrators to the NIS+ admin Group	6-24
6-14. Setting Up NIS+ Client Hosts	6-26
6-15. Setting UP NIS+ Replica Servers	6-29
6-16. Initializing NIS+ Client Users	6-32
6-17. Setting Up a NIS+ Subdomain	6-34
6-18. Using BIND With NIS+	6-37
6-19. Allowing a NIS+ User Authenticated Access to Another Domain	6-39
6-20. Managing NIS+ Objects and Their Properties	6-41
6-21. Managing NIS+ Tables	6-45
6-22. User, Group, Host and Domain Management in NIS+	6-48
6-23. Security Management in NIS+	6-53
6-24. LAB: Administering NIS+	6-55

Module 7 — Network and NFS Performance

Objectives	7-1
7-1. SLIDE: Network Performance	7-2
7-2. SLIDE: Protocol Overhead and Efficiency	7-3
7-3. SLIDE: Fragmentation of IP Datagrams	7-5
7-4. SLIDE: Using Path MTU Discovery	7-7
7-5. SLIDE: Switching PMTU Discovery	7-9

7-6.	SLIDE: Optimizing Memory Buffers	7-11
7-7.	SLIDE: Performance Measurement Tools	7-14
7-8.	SLIDE: <i>ndd</i> -- A Network Tuning Tool	7-18
7-9.	SLIDE: Remote Monitoring Management Information Base (RMON MIB)	7-20
7-10.	SLIDE: Embedded Advanced Sampling Environment (HP EASE)	7-22
7-11.	LAB: Using Network Performance Tools	7-24
7-12.	SLIDE: Understanding NFS Architecture	7-25
7-13.	SLIDE: NFS Operations	7-28
7-14.	SLIDE: Typical NFS Bottlenecks	7-30
7-15.	SLIDE: Synchronous Writing	7-31
7-16.	SLIDE: Changing the Number of biods and nfsds	7-33
7-17.	SLIDE: Timeouts and Retransmissions	7-35
7-18.	SLIDE: Attribute Caching	7-39
7-19.	SLIDE: Attribute Caching Options	7-41
7-20.	SLIDE: Server Disk I/O Impact	7-43
7-21.	SLIDE: Strategies for Applications on NFS Servers	7-44
7-22.	LAB: Optimizing NFS Performance	7-46

Module 8 — Dynamic Host Configuration Protocol

Objectives	8-1	
8-1.	SLIDE: Dynamic Host Configuration Protocol	8-2
8-2.	SLIDE: Review of BOOTP	8-4
8-3.	TEXTPAGE: BOOTP Header	8-6
8-4.	SLIDE: The <i>bootptab</i> Configuration File	8-7
8-5.	SLIDE: States of a DHCP Client	8-13
8-6.	SLIDE: DHCP Architecture on HP-UX	8-15
8-7.	SLIDE: DHCP Pool and Device Groups	8-17
8-8.	SLIDE: Configuring a DHCP Server	8-19
8-9.	SLIDE: The <i>dhcptab</i> Configuration File	8-22
8-10.	SLIDE: Configuring an HP-UX DHCP Client	8-27
8-11.	SLIDE: DHCP Troubleshooting	8-29
8-12.	SLIDE: Updating Network Configuration	8-31
8-13.	LAB: DHCP Configuration	8-34

Module 9 — Configuring and Maintaining the Domain Name Service

Objectives	9-1	
9-1.	SLIDE: Hostname Resolution	9-2
9-2.	SLIDE: Reverse Name Resolution	9-3
9-3.	SLIDE: Overview of DNS Configuration	9-4
9-4.	SLIDE: Standard Resource Record Format	9-6
9-5.	SLIDE: DNS Record Types	9-8
9-6.	SLIDE: The <i>db</i> Files	9-10
9-7.	SLIDE: The <i>db.cache</i> File	9-12
9-8.	SLIDE: The <i>db.127.0.0</i> File	9-14
9-9.	SLIDE: The <i>db.domain_name</i> File: Name Resolution	9-15
9-10.	SLIDE: The <i>db.network_number</i> File: Address Resolution	9-17
9-11.	SLIDE: The MX Record Type	9-20
9-12.	SLIDE: The <i>named.boot</i> File	9-22
9-13.	SLIDE: Building DNS Data Files	9-25
9-14.	SLIDE: Debugging BIND	9-28

9-15.	SLIDE: <code>nslookup</code> Options and Applications	9-30
9-16.	SLIDE: Examining Query and Response Packets	9-32
9-17.	SLIDE: Simulating the Resolver with <code>nslookup</code>	9-34
9-18.	SLIDE: Zone Transfers with <code>lsc</code>	9-36
9-19.	SLIDE: Maintaining DNS	9-37
9-20.	SLIDE: DNS Client Setup: Resolvers	9-39
9-21.	SLIDE: Host Name Resolution Order	9-41
9-22.	LAB: Configuring and Maintaining the DNS	9-47

Module 10 — Configuring `sendmail`

Objectives	10-1	
10-1.	SLIDE: <code>sendmail</code> Overview	10-2
10-2.	SLIDE : The Many Roles of <code>sendmail</code>	10-4
10-3.	SLIDE: UNIX Message Format	10-7
10-4.	SLIDE: UNIX Address Formats	10-9
10-5.	SLIDE: Address Encoding and Decoding	10-12
10-6.	SLIDE: <code>sendmail</code> as a Daemon	10-13
10-7.	SLIDE: <code>sendmail</code> as a Routing Facility	10-15
10-8.	SLIDE: The Configuration File <code>sendmail.cf</code>	10-17
10-9.	SLIDE: Configuration File Basics	10-19
10-10.	SLIDE: Rulesets and Mailers	10-21
10-11.	SLIDE: Defining a Mailer	10-23
10-12.	SLIDE: Ruleset Example: Avoiding Spam	10-26
10-13.	SLIDE: Simple Mail Transport Protocol (SMTP)	10-30
10-14.	SLIDE: Local and Remote SMTP	10-32
10-15.	SLIDE: Overview of <code>sendmail</code> Configuration	10-35
10-16.	SLIDE: Site Hiding	10-36
10-17.	SLIDE: DNS and e-mail	10-38
10-18.	SLIDE: The MX Algorithm	10-40
10-19.	SLIDE: MX and the Firewall	10-42
10-20.	SLIDE: Sending Mail Through a Firewall	10-43
10-21.	SLIDE: Internal Domain Addresses	10-44
10-22.	SLIDE: Configuring a Mail Hub	10-45
10-23.	SLIDE: Mail Servers and Clients	10-48
10-24.	SLIDE: <code>/etc/hosts</code> – Based Name Resolution	10-50
10-25.	SLIDE: Basic <code>sendmail</code> Checks	10-51
10-26.	SLIDE: <code>sendmail</code> Diagnostic Options	10-54
10-27.	SLIDE: The Mail Statistics File	10-56
10-28.	SLIDE: <code>sendmail</code> 's Logfile	10-58
10-29.	SLIDE: <code>sendmail</code> 's Mail Queue	10-62
10-30.	SLIDE: Checking Error Messages	10-65
10-31.	LAB: Configuring a Mail Hub	10-67

Appendix A — IBM AIX Operating System

Objectives	A-1	
A-1.	SLIDE: AIX UNIX Network Configuration	A-2
A-2.	SLIDE: Configuring More AIX Networking Parameters	A-4
A-3.	SLIDE: Main AIX Configuration Files	A-5
A-4.	SLIDE: Configuring BIND under AIX	A-7
A-5.	SLIDE: The AIX BIND Boot File	A-9

A-6.	SLIDE: Debugging BIND under AIX	A-11
A-7.	SLIDE: <code>sendmail</code> under AIX	A-13
A-8.	SLIDE: <code>sendmail</code> Configuration under AIX	A-14
A-9.	SLIDE: Compiling and Rereading the <code>sendmail.cf</code> File	A-15
A-10.	SLIDE: Creating System Mail Aliases	A-16
A-11.	SLIDE: Managing the Mail Queue	A-18
A-12.	SLIDE: Managing Mail Logging	A-19
A-13.	SLIDE: Managing the NLS Configuration File	A-21
A-14.	SLIDE: AIX: Serial Line Communications	A-22
A-15.	SLIDE: Steps for Configuring an AIX SLIP Connection	A-23
A-16.	SLIDE: Network Tuning under AIX	A-26
A-17.	SLIDE: Setting Buffer Sizes	A-27

Appendix B — Sun Solaris and SunOS Operating System

Objectives	B-1	
B-1.	SLIDE: Solaris UNIX Network Configuration	B-2
B-2.	SLIDE: Solaris Network Configuration — Scripts	B-4
B-3.	SLIDE: Configuring BIND under Solaris 2.X	B-6
B-4.	SLIDE: The <code>/etc/nsswitch.conf</code> file	B-7
B-5.	SLIDE: Debugging BIND under Solaris	B-8
B-6.	SLIDE: <code>sendmail</code> Configuration under Solaris	B-10
B-7.	SLIDE: Components of Solaris <code>sendmail</code>	B-11
B-8.	SLIDE: <code>sendmail</code> Configuration under SunOS 4.1	B-13
B-9.	SLIDE: Components of SunOS <code>sendmail</code>	B-14
B-10.	SLIDE: Solaris 2.3 Serial Line Communications	B-15
B-11.	SLIDE: Setting Up Solaris PPP	B-16
B-12.	SLIDE: The <code>/etc/asppp.cf</code> Configuration File	B-18
B-13.	SLIDE: Starting PPP and Error Logging	B-20
B-14.	SLIDE: Network Tuning Under Solaris UNIX	B-22
B-15.	SLIDE: <code>ndd</code> Command	B-24
B-16.	SLIDE: <code>snoop</code> Command	B-26
B-17.	SLIDE: NIS+ Administration Under Solaris 2.3	B-28
B-18.	SLIDE: NIS+ Tables	B-29
B-19.	SLIDE: Setting Up the Root Domain NIS+ Server	B-30
B-20.	SLIDE: Populating NIS+ Files	B-32
B-21.	SLIDE: Administering Access Rights within NIS+	B-35
B-22.	SLIDE: Setting Up an NIS+ Client	B-38

Appendix C — Digital UNIX

Objectives	C-1	
C-1.	SLIDE: Digital UNIX Network Configuration	C-2
C-2.	SLIDE: Digital UNIX Network Configuration — Scripts	C-3
C-3.	SLIDE: Configuration of BIND under Digital UNIX	C-5
C-4.	SLIDE: The <code>/etc/svc.conf</code> file	C-7
C-5.	SLIDE: Debugging BIND under Digital UNIX	C-8
C-6.	SLIDE: <code>sendmail</code> Configuration under Digital UNIX	C-9
C-7.	SLIDE: Components of Digital UNIX <code>sendmail</code>	C-10
C-8.	SLIDE: Serial Line Communications	C-13
C-9.	SLIDE: The <code>slattach</code> command	C-15
C-10.	SLIDE: The Protocol Analyzer <code>tcpdump</code>	C-17

Solutions

Figures

Tables

2-1.	2-39
------	-------	------

Overview

Course Description

This is a five-day lecture and lab course designed for experienced HP-UX network administrators. It concentrates on practical problems encountered in everyday administration of complex networks. This course is a follow-on to the HP-UX Network Administration I course (H6294S), but it also is appropriate for experienced administrators of any UNIX network.

Student Performance Objectives

Introduction

- On completion of this module you will be able to do the following:
 - Describe how this course relates to the HP-UX Network Administration I course.
 - Discuss the topics that are covered in this course.

Serial Line Protocols

- Configure and maintain connections using the point-to-point protocol (PPP) and its associated daemon, `pppd`.
- Manage Internet protocol (IP) routing over PPP.
- Apply appropriate PPP security functions to a connection.
- Troubleshoot a PPP connection.
- Migrate serial line Internet protocol (SLIP) connections to PPP.

Troubleshooting of Common TCP/IP Problems

- Detect, characterize, and correct common TCP/IP faults.
- Use `tracing` and `logging` to identify faults that vary with system conditions.

Routing Tables and Dynamic Routing Protocols

- Upon completion of this module you will be able to do the following:
 - Apply at least two different methods to the problem of maintaining routing tables.
 - Determine the table maintenance method most appropriate to a given network environment.

Managing the NIS Environment

- Upon completion of this module you will be able to do the following:
 - Discuss NIS maps and their function within NIS.
 - Generate new NIS maps.
 - Configure the automounter to use NIS maps.

NIS+ Administration

- Describe the purpose of NIS+ and its name space.
- Plan the NIS+ name space.
- Set up the NIS+ name space.
- Administer NIS+.

Network and NFS Performance

- Isolate and characterize network bottlenecks.
- Isolate and characterize NFS bottlenecks.

Dynamic Host Configuration Protocol

- List the characteristics of the BOOTP protocol.
- Describe the features of the BOOTP protocol header.
- Understand BOOTP protocol vendor formats.
- Understand Dynamic Host Configuration Protocol.
- List DHCP client states.
- Understand DHCP architecture on HP-UX.
- Describe DHCP keywords.
- Configure DHCP client and server.

Configuring and Maintaining the Domain Name Service

- Describe DNS (Domain Name Service) features and operation.
- Explain how DNS resolves hostnames into IP addresses, and vice versa (reverse name resolution).
- Describe in detail how DNS data files are built and maintained.

- Set up a complete Internet domain.

Configuring sendmail

- List at least three different mail address formats.
- Describe the operation of SMTP.
- Make at least one valid modification to a ruleset in the `sendmail.cf` configuration file.
- Set up site hiding to support mail clients and servers.
- Configure forwarding.
- Configure a mail relay.

IBM AIX Operating System

- Document the differences between IBM (AIX) and HP-UX.

Sun Solaris and SunOS Operating System

- Document the differences between SunOS, Solaris, and HP-UX.

Digital UNIX

- Document the differences between Digital UNIX and HP-UX.

Student Profile and Prerequisites

Students attending this course should have experience administering networked UNIX systems and should have attended the HP-UX Network Administration I course (H6294S).

Students should be familiar with:

the physical components of local area networks
the functions of the Network File System (NFS)
the functions of the Network Information System (NIS)
the functions of the Distributed Naming Service (DNS)
user administration in `sendmail`

Curriculum Path

This course is a follow-on to HP-UX Network Administration I (H6294S).

Agenda

Monday a.m. • Introduction

	<ul style="list-style-type: none"> • Serial Line Protocols <ul style="list-style-type: none"> — lecture and lab • Troubleshooting Common TCP/IP Problems <ul style="list-style-type: none"> — first lecture
Monday p.m.	Troubleshooting Common TCP/IP Problems <ul style="list-style-type: none"> • first lab • <code>syslog</code> lecture and lab • <code>nettl</code> lecture and lab
Tuesday a.m.	Routing Tables and Dynamic Routing Protocols <ul style="list-style-type: none"> • routing lecture and lab • OSPF lecture
Tuesday p.m.	<ul style="list-style-type: none"> • Routing Tables and Dynamic Routing Protocols (OSPF lab) • Managing the NIS Environment (lecture and lab)
Wednesday a.m.	NIS+ Administration (lecture and lab)
Wednesday p.m.	Network and NFS Performance (lectures and labs)
Thursday a.m.	Dynamic Host Configuration Protocol (lecture and lab)
Thursday p.m.	Configuring and Maintaining the Domain Name Service (lecture)
Friday a.m.	<ul style="list-style-type: none"> • Configuring and Maintaining the Domain Name Service (lab) • Configuring <code>sendmail</code> (lecture)
Friday p.m.	Configuring <code>sendmail</code> (lab)

Module 1 — Introduction

Objectives

On completion of this module you will be able to do the following:

- Describe how this course relates to the HP-UX Network Administration I course.
- Discuss the topics that are covered in this course.

1-1. SLIDE: What You Already Know

What You Already Know

In HP-UX Network Administration I, you learned how to do the following tasks:

- Connect and configure networks.
- Configure and administer naming services.
- Manage a simple **sendmail** service.
- Install and update software over a network.
- Manage printing services.
- Back up a system over a network.

a6461

Student Notes

In the previous course, “HP-UX Network Administration” (H6294S), you learned how to do the following:

- Connect and configure networks.

The following topics were discussed:

- network mapping
- LAN 9000 configuration, manual configuration and using System Administration Manager (SAM)
- **/etc/networks** and **/etc/hosts**
- routing tables
- system as router

- subnetting
- troubleshooting
- Configure and administer naming services.

The following naming services were discussed:

- ARPA/Berkeley services
- daemons and servers
- **inetd**
- ports and sockets
- **rwhod**
- **rwho** and **ruptime**
- Manage a simple **sendmail** service.
 - user agent, routing facility, delivery agent, receiving agent, and mailer
 - standalone hosts, mail servers, and mail clients
 - aliasing and autoforwarding
 - undeliverable messages and statistic file
 - **mail.log** and **/etc/mail/sendmail.cf**
- Install and update software over a network.
 - HP Software Distributor
 - Software Distributor server
 - software depot security
 - cold network install
 - install boot server
- Manage printing services.
 - HP Distributed Print Service (HPDPS)
 - spoolers, queues, supervisors and printers
 - event notifications
- Back up a system over a network.

- **fbackup/frecover**
- HP OpenView OmniBack

1-2. SLIDE: New Components in This course

New Components in This Course

In this course, you will become familiar with and gain experience in using the following sets of rules and software:

- The point-to-point protocol (PPP)
- The NFS automounter
- The Network Information Service Plus (NIS+)
- The dynamic host configuration protocol (DHCP)

a6462

Student Notes

New components in this course include modules devoted specifically to these topics: point-to-point protocol (PPP), network file system (NFS) automounter, NIS+, and dynamic host configuration protocol (DHCP).

- Where local area network (LAN) or packet-switched wide area network (WAN) connectivity is not possible, network connections can be established over serial links. Point-to-point protocol can support this link, as can transfer control protocol/Internet Protocol (TCP/IP) and serial line Internet protocol (SLIP).

You will be instructed how to perform the following tasks:

- Configure and maintain connections using the PPP and its associated daemon, `pppd`.
- Manage IP routing over PPP.
- Apply appropriate PPP security functions to a connection.

- Troubleshoot a PPP connection.
- Migrate serial line Internet protocol (SLIP) connections to PPP.
- The network information service (NIS) allows you to administer the configuration of many network hosts from a central location. Common configuration information can be managed centrally, and propagated throughout the network.

NIS provides hostname resolution services, and also manages many other types of configuration data, such as user account names and passwords, groups of users, mappings of network services to port numbers and protocols, and much more.

In order for a system to make use of its remote file resources, NFS provides an **automounter**, which mounts the needed file system components on an *as-needed* basis. The **automounter**, in turn, makes use of NIS *maps* that inform it of the location of the resources and the conditions under which they should be mounted.

In addition, you will be instructed how to do the following:

- Describe NIS maps and their function within NIS.
- Generate new NIS maps.
- Configure the **automounter** to use NIS maps.
- NIS+ is a new service designed to replace NIS. NIS+ is a distributed database system that allows you to store configuration information on a master server and propagate that information to all hosts on your network. NIS+ employs a hierarchical domain structure called the **NIS+ namespace**. You will learn to configure and administer the servers and clients in an NIS+ namespace.

Additionally, you will be directed how to do the following:

- Describe the purpose of NIS+ and its namespace.
- Plan the NIS+ namespace.
- Set up the NIS+ namespace.
- Administer NIS+.
- The dynamic host configuration protocol permits Internet protocol (IP) addresses to be pooled, released for use by network devices only as needed, and returned to the pool for use by other devices. This module will enable you to do the following:
 - List the characteristics of the BOOTP protocol.
 - Describe the elements of the BOOTP header and BOOTP protocol vendor formats .
 - Describe BOOTP protocol vendor formats.
 - Describe DHCP.
 - List DHCP client states and DHCP keywords.

- Describe DHCP architecture in HP-UX.
- Configure DHCP for clients and servers.

1-3. SLIDE: Coverage in Depth

Coverage in Depth

- TCP/IP troubleshooting and tools
- routing table maintenance and dynamic routing protocols
- network and NFS performance
- configuring and maintaining the domain name service (DNS)
- site-level configuration of `sendmail`

a6463

Student Notes

In addition to discussing the new components just mentioned—PPP, NFS automounter, NIS+, and DHCP—this course also focuses on the following topics:

- TCP/IP troubleshooting and tools

TCP/IP networks are robust when correctly configured and when all components are operating properly, but hardware failures and configuration errors can take a considerable toll on network performance and stability. You will learn the most common errors and points of failure in a TCP/IP network, and how to detect and correct such errors and failures.

- routing table maintenance and dynamic routing protocols

The correct operation of network routers and certain kinds of network switches depends completely on the currency of the routing tables used by these devices. Because a network is usually subject to frequent change, maintaining accurate routing tables is a nontrivial problem. You will learn about a variety of routing protocols and determine which is most appropriate for your network environment.

- network and NFS performance

The subject of network performance and tuning is a large and complex matter. You will become familiar with network performance limitations and practical solutions based on configuration alternatives.

- domain name service (DNS) configuration and maintenance

DNS resolves hostnames into addresses (name resolution) and addresses into hostnames (reverse name resolution) to meet the needs of users and applications. You will extend what you have learned about BIND into a full knowledge of how to maintain DNS for use in your network.

- site-level configuration of `sendmail`

The `sendmail` facility routes mail to users and applications, either local or remote, by means of the simple mail transfer protocol (SMTP).

1-4. SLIDE: Tasks You Will Be Able to Perform

Tasks You Will Be Able to Perform

You will learn how to do a variety of tasks, including the following:

1. Establish and maintain PPP links over serial connections.
2. Maintain routing tables under dynamic routing protocols.
3. Configure NIS+.
4. Configure NFS and its automounter to use NIS+ maps.
5. Characterize and solve network performance problems.
6. Support a variety of devices employing the dynamic host configuration protocol (DHCP).
7. Configure and maintain the domain name service (DNS).
8. Configure and maintain **sendmail**.

a6464

Student Notes

You will be able to do the following tasks:

- Establish and maintain PPP links over serial connections.

You will configure and maintain connections using the point-to-point protocol and its associated daemon, **pppd**. You will be introduced to managing IP routing over PPP and applying appropriate PPP security functions to a connection. You will also troubleshoot a PPP connection and migrate serial line Internet protocol (SLIP) connections to PPP.

- Maintain routing tables under dynamic routing protocols.

You will be given the opportunity to apply at least two different methods to the problem of maintaining routing tables. You will be able to determine the table maintenance method most appropriate to a given network environment.

- Configure NIS+.

You will be able to describe the purpose of NIS+ and its namespace; plan the NIS+ namespace; set up the NIS+ namespace, and administer NIS+.

- Configure NFS and its automounter to use NIS+ maps.

NIS maps and their function within NIS, generating new NIS maps, and configuring the Automounter to use NIS maps will be discussed.

- Characterize and improve network performance problems.

You will be shown how to isolate and characterize network bottlenecks, and to isolate and characterize NFS bottlenecks.

- Support a variety of devices employing DHCP.

You will be able to discuss the characteristics of the BOOTP protocol, elements of the BOOTP header, and vendor formats. You will also be able to describe the dynamic host configuration protocol and list its client states, describe DHCP architecture in HP-UX, list DHCP keywords, and configure DHCP for clients and servers.

- Configure and maintain the domain name service (DNS).

DNS features and operation will be discussed. How DNS resolves hostnames into IP addresses, and reverse name resolution will be covered. You will be able to describe in detail how DNS data files are built and maintained. In addition, you will set up a complete Internet domain.

- Configure and maintain `sendmail`.

At least three different mail address formats will be discussed. You will be able to describe the operation of SMTP, make at least one valid modification to a ruleset in the `sendmail.cf` configuration file, set up site hiding to support mail clients and servers, and configure forwarding and a mail relay.

Module 2 — Serial Line Protocols

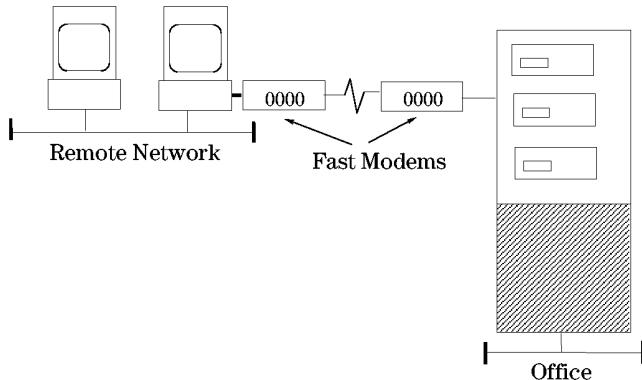
Objectives

Upon completion of this module, you will be able to do the following:

- Configure and maintain connections using the point-to-point protocol (PPP) and its associated daemon, `pppd`.
- Manage Internet protocol (IP) routing over PPP.
- Apply appropriate PPP security functions to a connection.
- Troubleshoot a PPP connection.
- Migrate serial line Internet protocol (SLIP) connections to PPP.

2-1. SLIDE: Point-to-Point Links

Point-to-Point Links



Inexpensive connectivity among remote networks

a6465

Student Notes

Where coax cabling is restricted, network connections can be made with serial lines. Serial lines are relatively inexpensive and easy to install. A point-to-point link (PPL) extends the Internet protocol (IP) network over them. It permits the use of hard-wired, dial-out, or dial-in serial (`tty`) lines. PPL establishes a connection over the serial line, then runs an encapsulation protocol to transfer packets between the IP network and the serial line.

Several types of Internet protocols specify a standardized set of rules defining how two machines will communicate with each other, via their local software over serial lines.

The most common implementations of a point-to-point link include the following:

Serial line Internet protocol (SLIP)	Transmission control protocol/Internet protocol (TCP/IP) for transmitting IP datagrams over serial lines such as RS-232 cables and dial-up telephone lines that connect two systems.
Compressed SLIP (CSLIP)	A type of SLIP in which the header is compressed.
Point-to-point protocol	A successor to SLIP, PPP supports route-to-router and host-to-network connections over synchronous and asynchronous lines. PPP often includes error detection and correction schemes.

As of HP-UX 10.30, the point-to-point protocol is fully supported. SLIP connections can also be set up under an option within HP's implementation of PPP.

2-2. SLIDE: Point-to-Point Protocol Features

The Point-to-Point Protocol (PPP) Features

HP-UX PPP features

- on demand dial-up connections
- link traffic control
- minimal transmission overhead

Not supported:

- traffic encryption

a6466

Student Notes

Features of the Point-to-Point Protocol

On demand dial-up connections	Network routers require an expensive leased line to provide a constant connection. PPP establishes an asynchronous link over standard telephone lines when traffic demands. PPP can then hang up the connection to save the telephone costs. This kind of link is transparent to applications.
Link traffic control	PPP can enhance security provided to other systems on the local network with a selective-isolation packet filtering <i>firewall</i> gateway. The HP-UX system administrator has complete control over all traffic allowed to cross the link.

Minimal
transmission
overhead

PPP compresses high data link control (HDLC) address control, protocol fields, and TCP headers, thereby improving both throughput and interactive response on typical modem connections. In its interface to HP-UX, TCP/IP implements the *fast queue* scheme, in which interactive packets have priority to transmit on a congested link.

NOTE:

Point-to-point protocol does not support traffic encryption.

2-3. SLIDE: Point-to-Point Protocol Overview

Point-to-Point Protocol Overview

- common encapsulation protocol for all point-to-point links
- negotiation of physical link options
variable, but defined MTU size
- optional authentication
- error detection/error correction
- network type definition
several network protocols simultaneously possible
- negotiation of network-layer options
dynamic addressing scheme
- optional compression

a6467

Student Notes

The point-to-point protocol is a common encapsulation protocol for all point-to-point links. PPP is based on the International Standards Organization/Open Systems Interconnection (ISO/OSI) data link layer HDLC.

PPP facilitates the following features:

- It allows different network layer protocols to exist on the same physical link.
- It provides negotiation of the network-layer independent options.
- It provides negotiation of the network-layer dependent options.
- It supports different compression protocols.
- It provides error detection.
- It allows dynamic addressing.

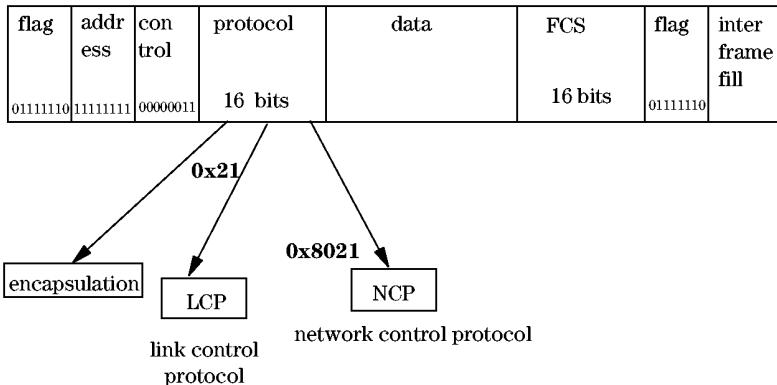
The point-to-point connection must adhere to the following constraints:

- 8 bits per character
- no parity check

- full duplex

2-4. SLIDE: PPP Format

PPP Format



a6468

Student Notes

PPP is comprised of three main components. According to these components, a PPP packet can transport a

- method for encapsulating datagrams over serial links
- link control protocol (LCP) for establishing, configuring, and testing the data-link connection
- family of network control protocols (NCP)s for establishing and configuring different network layer protocols

Dividing the PPP protocol into different components allows PPP to

- Minimize encapsulation overhead for a single datagram. (Data link information need not be transported within each datagram.)
- Establish, configure, and test the point-to-point link, independently of the network layer protocol.

- Be more flexible when configuring network parameters such as maximum transfer unit (MTU) size.

Encapsulation

PPP provides an encapsulation protocol over both bit-oriented synchronous and asynchronous links. It also provides for the multiplexing of different network layer protocols simultaneously over the same link. By default, only 8 additional octets are necessary to form the encapsulation.

In environments where bandwidth is at a premium, the encapsulation can be shortened to as few as 2 octets. To support high-speed hardware implementations, PPP ensures that the default encapsulation header and information fields fall on 32-bit boundaries and allows the trailers to be padded to an arbitrary boundary.

Link Control Protocol

PPP defines more than just an encapsulation scheme. In order to be portable to a wide variety of environments, PPP provides link control protocol (LCP).

LCP is used to

- handle varying limits on sizes of packets
- authenticate the identity of its peer on the link
- determine when a link is functioning properly
- detect a looped-back link and other common configuration errors
- terminate the link

Network Control Protocol

A network control protocol allows the configuration of parameters that depend on the used network protocol. For instance, assignment and management of IP addresses, which is a problem even in LAN environments, is especially difficult over circuit-switched point-to-point links (such as dial-up modem servers).

Each network protocol has its own control protocol. In the case of TCP/IP, the Internet protocol control protocol (IPCP) controls IP specific parameters.

The frame fields have the following meaning:

Flags

Flag is a single octet and indicates the beginning or end of a frame. Its binary sequence is 01111110. Flag is a frame separator. Only one flag is required between two frames.

Address

Address is a single octet and contains the binary sequence 11111111, which represents the All-Stations address. PPP does not assign individual station addresses. The use of other address lengths and values may be defined at a later time.

Control

Control is a single octet and contains the binary sequence 00000011. Frames with other control field values should be discarded.

Protocol

Protocol is two octets and its value identifies the protocol encapsulated in the data field. Depending on the component protocol, numbers are divided into the following ranges:

0x0xxx to 0x3yyy	network layer protocol
0x4xxx to 0x7yyy	protocols with low volume traffic which have no associated NCP
0x8xxx to 0xbyyy	associated network control protocols (NCPs)
0xcxxx to 0xfyyy	link control protocols

Examples

Network Layer Protocols

0x0021	Internet protocol
0x0023	OSI network layer
0x0025	Xerox NS IDP
0x0027	DECnet Phase IV
0x002b	Novell IPX

Network Control Protocols

0x8021	Internet protocol control protocol
0x8023	OSI network layer control protocol
0x8025	Xerox NS IDP control protocol
0x8027	DECnet Phase IV control protocol
0x802b	Novell IPX control protocol

Link Control Protocols

0xc021	link control protocol
--------	-----------------------

0xc023	password authentication protocol
0xc025	link quality report
0xc223	challenge handshake authentication protocol

Data

The data field contains the datagram for the protocol specified in the protocol field. The default maximum length of the data is 1500 octets. By negotiation, consenting PPP implementations may use other values for the maximum data length.

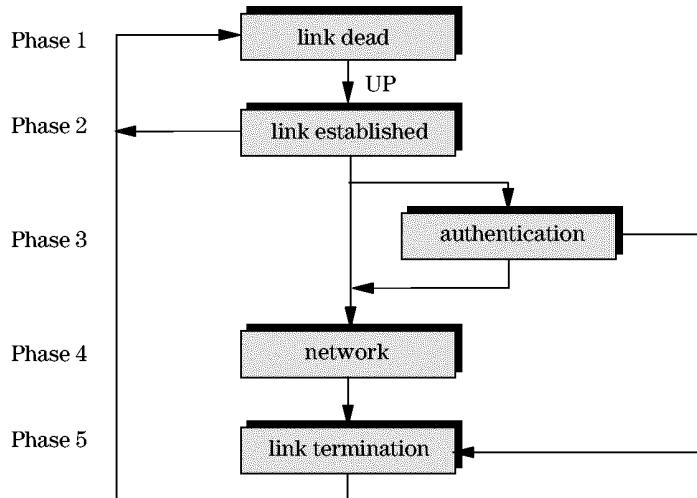
On transmission, the data may be padded with an arbitrary number of octets up to the maximum length. It is the responsibility of each protocol to distinguish padding octets from real information.

Frame Check Sequence

Frame check sequence (FCS) is two octets. The FCS field is calculated over all bits of the address, control, protocol and data fields not including the flag separators.

2-5. SLIDE: Phases of a PPP Link

Phases of a PPP Link



a6307

Student Notes

According to the different components of PPP, a PPP link goes through several phases.

- | | |
|---------|---|
| Phase 1 | Link Dead—A link is dead when the physical layer is down. |
| Phase 2 | Link Establishment—A carrier detection, a start-up program, or some other action causes an advance to the link establishment phase. The link control protocol establishes the connection by exchanging configuration packets. Only configuration options that are independent of particular network layer protocols are configured by LCP. Any non-LCP packet received during this phase is silently discarded. |
| Phase 3 | Authentication—Optionally, authentication takes place after the link has been established. The use of the authentication protocol must be negotiated during phase 2, the link establishment phase. |

Phase 4

Network layer protocol—Configuration of individual network layer protocols is handled by separate network control protocols during the network layer protocol phase.

Each network layer protocol must be configured separately by the appropriate network control protocol.

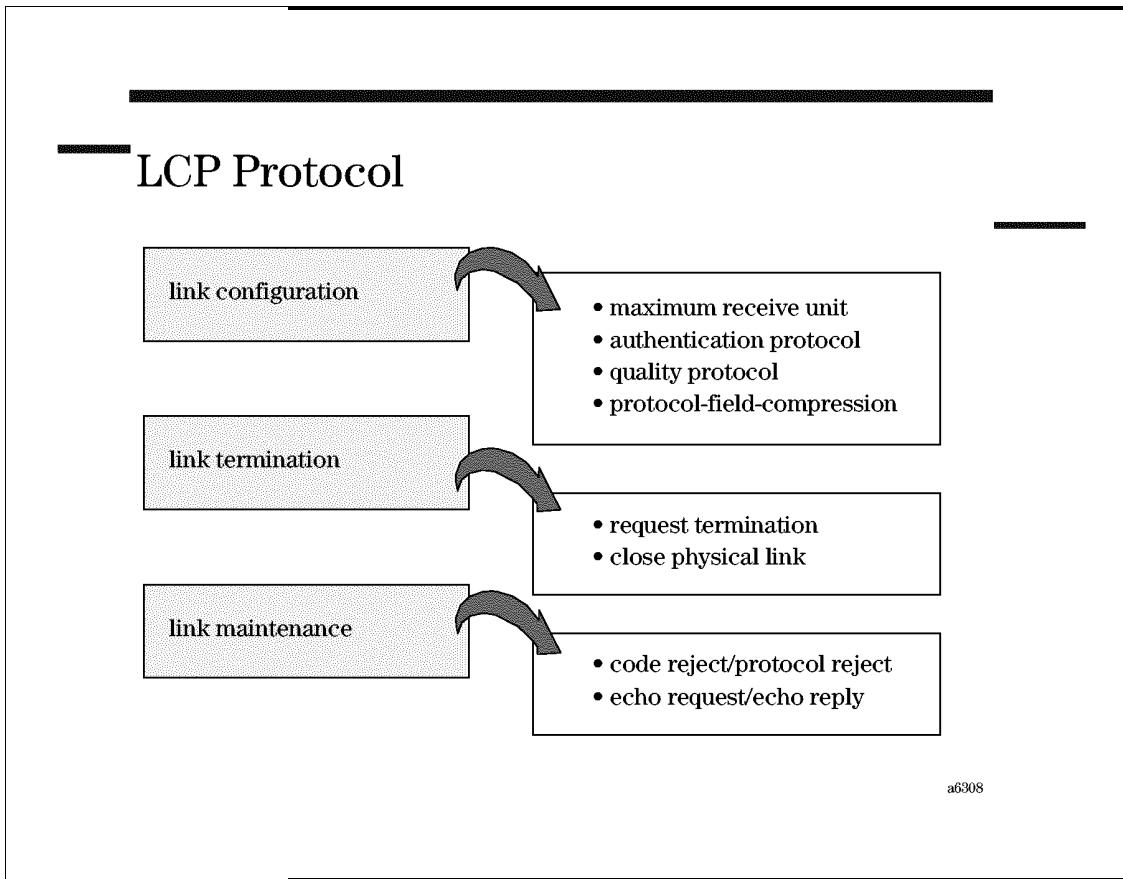
Phase 5

Link termination—The link may be terminated at any time for reasons that include

- the request of a human user
- a physical event such as the loss of a carrier
- an authentication failure
- a link quality failure
- the expiration of an idle-period timer

The link control protocol is used to close the link through an exchange of terminate packets. When the link is closing, PPP informs the network layer protocols so that they can take appropriate action.

2-6. SLIDE: LCP Protocol



Student Notes

LCP supports the following operations:

- Operations for link configuration

configure-request

To open a connection, a **configure-request** must be submitted first. An options list contains configuration options that the sender wishes to negotiate. Configuration options can be

- maximum receive unit
- async-control-character-map
- authentication-protocol
- quality-protocol
- magic-number (identification number of a certain system)
- RESERVED
- protocol-field-compression
- address-and-control-field-compression

configure-ack	If every configuration option received in a configure-request is both recognizable and acceptable, then a configure-ack is sent back as an acknowledgment.
configure-nak	If configuration options received in a configure-request are recognizable, but not acceptable, then a configure-nak is sent back. After a configure-nak , the sender can send a new configure-request with new values for the configure options.
configure-reject	A configure-reject is sent back if, in a configure-request , configure options are not recognizable or are not acceptable.

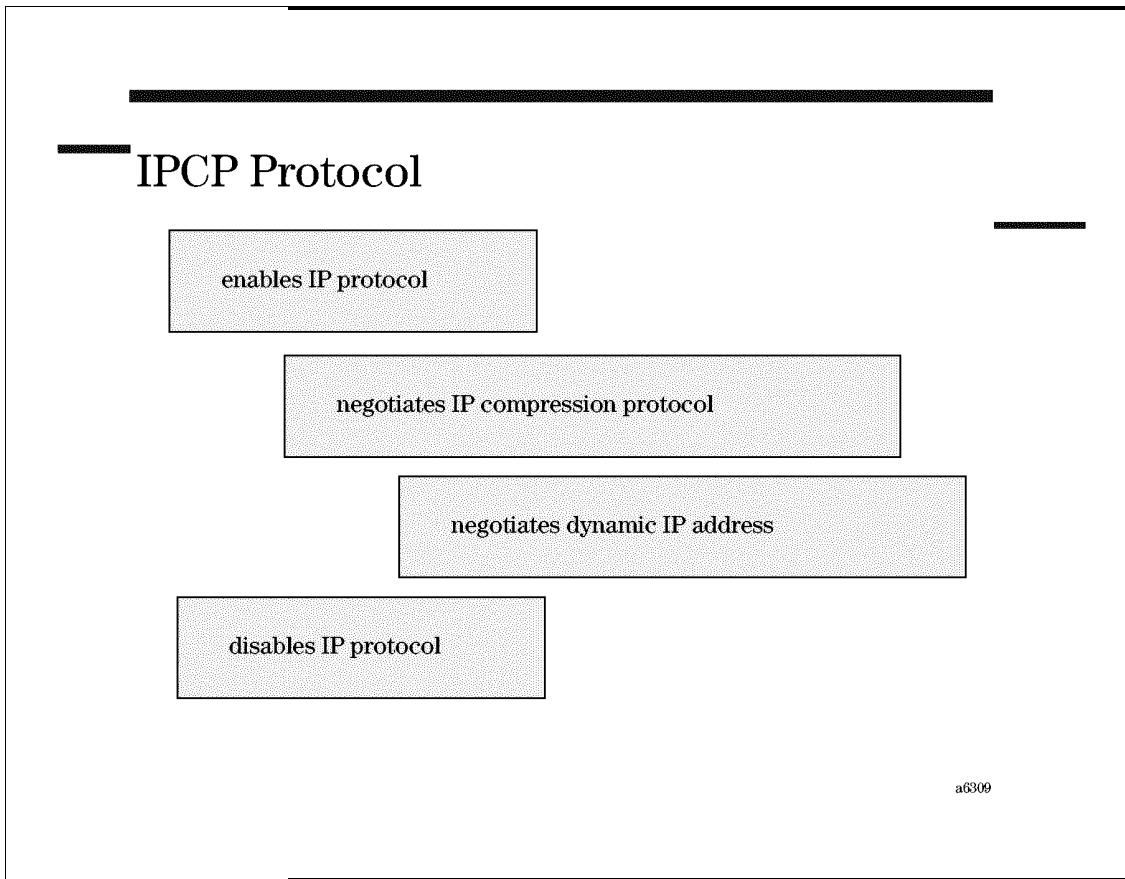
- Operations for link termination

terminate-request	A terminate-request requests the termination of a link.
terminate-ack	A terminate-ack acknowledges this request. The physical link now can be closed.

- Operations for link maintenance

code-reject	A code-reject indicates that a received LCP operation was faulty or incomplete.
protocol-reject	A protocol-reject indicates that the peer was attempting to use a protocol that is not supported.
echo-request	LCP includes echo-request in order to provide a data link layer loopback mechanism for use in exercising both directions of the link.
echo-reply	Upon reception of an echo-request , an LCP packet must be answered with the echo-reply .
discard-request	This operation is for debugging purposes only. LCP includes a discard-request code in order to provide a data-link layer data-sink mechanism for use in exercising the local-to-remote direction of the link. A discard receiver simply throws away discard-requests .

2-7. SLIDE: IPCP Protocol



Student Notes

The Internet protocol control protocol (IPCP) is responsible for configuring, enabling, and disabling the IP protocol modules on both ends of the point-to-point link.

IPCP supports the configuration of the following options:

IP address	This configuration option provides a way to dynamically negotiate the IP address to be used on the local end of the link. It allows the sender of the <code>configure-request</code> either to state the desired IP address or to request that the peer provide the information.
------------	--

IP-compression protocol

This configuration option provides a way to negotiate the use of a specific compression protocol. By default, compression is not enabled. The current supported compression protocol is *Van Jacobson Compressed TCP/IP*.

2-8. SLIDE: PPP and SLIP

PPP and SLIP

SLIP connectivity can be provided, but important PPP features will not be available:

- link control protocol (LCP) and Internet protocol control protocol (IPCP)
- password authentication protocol (PAP) and challenge handshake authentication protocol (CHAP)
- link quality monitoring (LQM)
- asynchronous control character mapping or the escape option

Additional restrictions apply.

a646119

Student Notes

When linking with a peer host that cannot use the point-to-point protocol, PPP offers SLIP option. SLIP is a framing convention for arranging IP packets on a link. Other PPP options available when running SLIP include automatic dialing, idle line hangup, packet filtering, and the `exec` option. Most other management facilities can be invoked with SLIP.

However, SLIP provides few of the advanced facilities available with the PPP protocol. SLIP connectivity can be provided, but some important PPP features are not available:

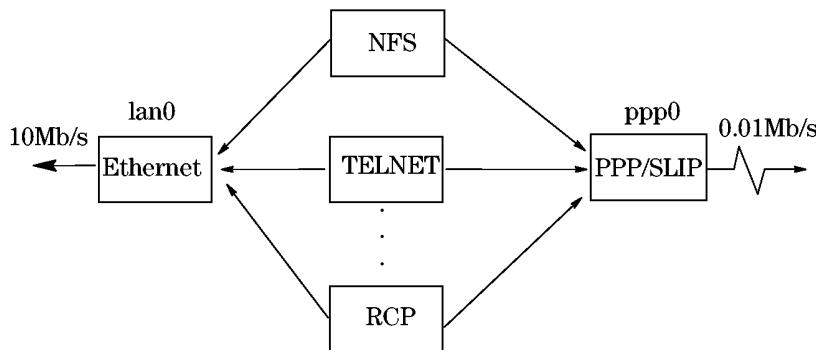
- negotiations using LCP and IPCP
- authentication using password authentication protocol (PAP) and challenge handshake authentication protocol (CHAP)
- link quality monitoring (LQM)
- asynchronous control character mapping or the escape option

The following connection restrictions exist when running PPP with the SLIP option:

- Links must be asynchronous.
- Connections must be completely transparent to all 8-bit character values.
- Flow control selection must be hardware only or no flow control at all.
- Link must not interpret passing data as flow control.

2-9. SLIDE: Serial-Line Throughput Considerations

Serial-Line Throughput Considerations



a6469

Student Notes

The performance of network file system (NFS), X, and highly interactive software suffers from the slower services a serial line provides. In some extreme cases, NFS service requests can time out before a serial line connection can send and receive the information.

You will not reach LAN speeds. NFS works, but can be slow. Although interactive programs tend to have bursts of high speeds, overall performance is slower, but tolerable.

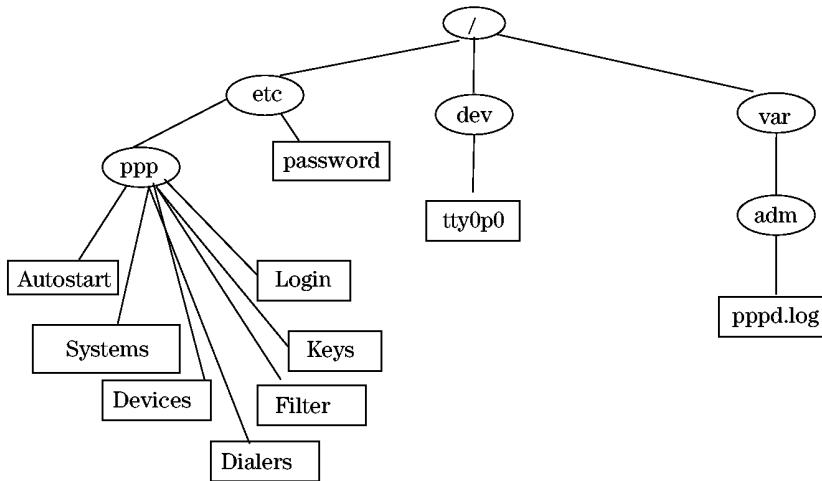
Modem protocols can affect serial line timing:

- Protocol spoofing—The local machine can fool the remote that it has already received and acknowledged the packet, when, in fact, it may not yet have left the modem.
- Compression—Textual data can typically be compressed up to 80%, while binary data can be compressed only about 5%. Compression is usually performed by the modem, thereby improving apparent data rates.

- Ping-pong protocol—Some modems do not pass in a symmetric fashion. Instead, they use a high speed channel in one direction and a low speed channel in the reverse. These modems usually monitor the traffic and switch the higher speed channel to service the majority of the data flow. This restricts the acknowledgment packet speed during a transfer, slowing down the link.

2-10. SLIDE: PPP Configuration Files

PPP Configuration Files



a64610

Student Notes

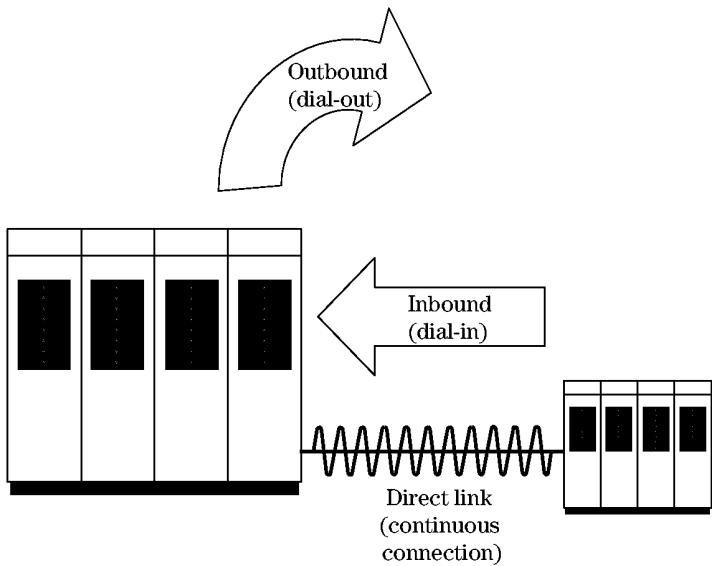
A `pppd` process on a local system negotiates with the `pppd` process on a remote or peer system to establish the PPP connection. The following files are used for PPP connections:

- `/etc/ppp/Autostart` starts `pppd` for on-demand outbound calls.
- `/etc/ppp/Systems` contains the hostname or IP address of peers and how to connect with peers.
- `/etc/ppp/Devices` associates dialer types with physical devices and speeds. `pppd` examines the file when it places a call. If no suitable speed is found or if all devices associated with that speed are busy, `pppd` tries again later.
- `/etc/ppp/Dialers` describes how to dial each type of modem that is available for outbound calls.
- `/etc/passwd` is used to verify the login of an inbound connection.

- **/etc/ppp/Login** is a shell script that is run after a successful login. Login starts **pppd** on the local system to communicate with **pppd** on the peer to negotiate and establish a PPP connection.

2-11. SLIDE: PPP Configuration Overview

PPP Configuration Overview



a64611

Student Notes

- Dialing in to an HP 9000
 - Users at remote supported terminals or personal computers can establish dial-in IP connections with logins. For a login connection, the user establishes a modem link to an HP 9000 and logs in as usual. After login, the `Login shell` script is run. The script starts `pppd` on the local system, which communicates with the `pppd` on the peer. The two `pppds` negotiate and establish a PPP connection.
 - For a connection without a login, the user simply dials in to a preset HP 9000 serial line where the serial protocol is already running on the line.
- Dialing out from an HP 9000
 - HP 9000 users can establish dial-out IP connections with or without login. This can be to any remote host that runs a supported serial IP protocol. For a login connection, the user simply invokes `pppd` at the HP-UX shell prompt. `pppd` establishes a modem link to the

specified host and logs in. **pppd** runs a command on the remote machine to initiate the specified serial IP protocol and establish the connection

- For a connection without login, the user's action is the same. After the user invokes **pppd**, a preset line is assigned to that user. The user has full network access to the remote host.
- Configuration Quick Reference

The following steps show how to configure outbound and inbound PPP connections.

— Outbound

1. Create device file with dial-out or direct connect minor number for serial port.
2. Increase the number of IP tunnels if needed (16 are created by default). (*optional*)
3. Configure your modem.
4. Create entry in **/etc/ppp/Devices**. Optionally, add entries to **/etc/ppp/Dialers** or **Dialers.local**.
5. Define dial-out connection in **/etc/ppp/Systems**.
6. Create **/etc/ppp/Autostart**.

— Inbound

1. Create device file with dial-in minor number for serial port.
2. Increase the number of IP tunnels if needed (16 are created by default). (*optional*)
3. Configure your modem.
4. Add user accounts to **/etc/passwd** to allow incoming connections.
5. Create **Login** shell script.

2-12. SLIDE: Create Device Files for the Serial Ports

Create Device Files for the Serial Ports

Device file template

O	x	I	I	P	P	H	M
---	---	---	---	---	---	---	---

a64612

Student Notes

Create device files in the `/dev` directory for serial ports. Each serial port used for `pppd` may require one to four device files. Depending on the hardware attached, you may also need files for the dial-in, dial-out, or direct connect minor number.

Use the System Administration Manager (SAM) to create device files. Creating the device files with appropriate minor numbers is important.

The following is an example of an entry in `/dev` for the device `tty0b1`, a dial-in modem with hardware flow control enabled.

```
major number = 1  
minor number = 0x0000012
```

The minor number corresponds to the template 0xIIPPHM, which is resolved as follows:

- II Two hexadecimal digits (8 bits) to indicate the instance of the serial interface. The instance of the serial interface is determined by running the program `ioscan -f` and looking in the *I* column. On a Series 700, port A will generally be 0, and port B will generally be 1.
- PP Two hexadecimal digits (8 bits) to indicate the port number of this device on the serial interface. On a Series 700 serial port, it will always be 0.
- H One hexadecimal digit (4 bits) to control diagnostic access and hardware flow control (HP J2094A only). Bit 0 controls RTS/CTS hardware flow control.
 - bit 0 = 0** disables the hardware flow control
 - bit 0 = 1** enables the hardware flow control
- M One hexadecimal digit (4 bits) to determine port access type. Values for each bit are
 - bit 3 = TI/ALP**
 - bit 2 = 0** means simple protocol (U.S.A.)
 - bit 2 = 1** means CCITT protocol (Europe)
 - bits 0 and 1 = 00** direct, **01** dial-out modem, **10** dial-in modem, **11** unused

You can also use the `mknod` command to create device files.

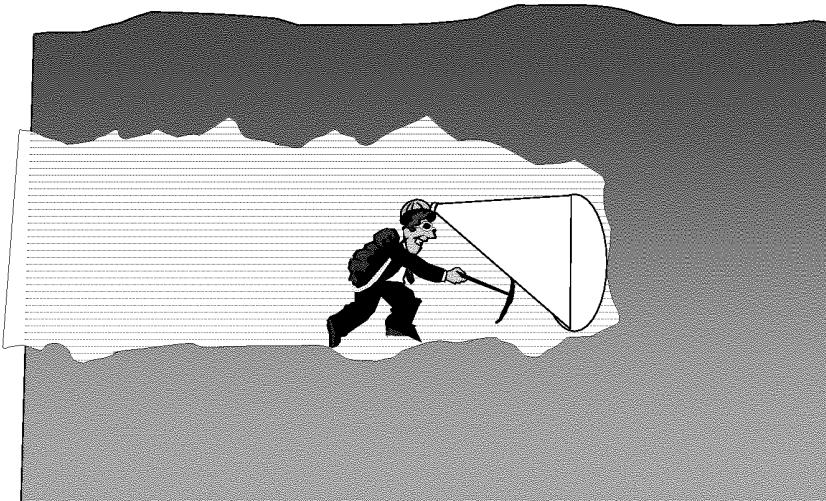
For example,

```
mknod /dev/ttypd10 c 1 0x000001 #dial-in
```

The `termio(7)` manpage describes how to derive the minor number of the device file for dial-in, dial-out, and direct connections. Major and minor numbers may be different from system to system. See `termio(7)` for more details about configuring a serial device.

2-13. SLIDE: Increase the Number of IP Tunnels

Increase the Number of IP Tunnels



a64613

Student Notes

`pppd` uses IP tunnels to pass packets between the serial port and IP layer. During bootup, `/sbin/init.d/ppp` creates 16 IP tunnels (the default value is defined in `/etc/ppp/tunconf`) which will support 16 concurrent `pppd` processes. Each IP tunnel creates three device files. Each `pppd` process facilitates a PPP connection. The kernel can support a maximum of 64 IP tunnels. If you need more than 16 IP tunnels, you can increase the value of `NTUN` in `/etc/ppp/tunconf` and reboot the system.

For more information about configuring IP tunnels, see manpage `tun(4)`.

2-14. SLIDE: Configuring Your Modem

Configuring Your Modem

- Use the highest available speed.
- Enable error correction.
- Enable data compression.
- Enable hardware flow control.
- Set the modem to answer after one ring.
- Disable result codes for incoming calls.
- Enable extended result codes for outgoing calls.
- Assert Carrier Detect for remote modem's carriers.

a64614

Student Notes

The following are general recommendations for configuring modems to use with PPP:

- Choose the highest asynchronous serial speed that both the modem and computer can support.
- Enable error correction. If possible, choose CCITT V.42 for compatibility with CCITT V.42bis data compression.
- Enable data compression. HP recommends CCITT V.42bis for higher maximum compression ratios and handling of precompressed data streams.
- Enable flow control. Use hardware instead of software flow control if possible.
- Use default modem parameters for purposes outside PPP protocol, including other inbound applications.

- Set up dialers for UNIX-to-UNIX copy (UUCP) or PPP outbound connections. Dialers' PPP-specific register settings ensure that a general purpose modem used for PPP will work as intended during a PPP session.
- Allow bidirectional connections over any available modem. Try the following suggestions to configure bidirectional modems and to allow modem use by other applications, such as **uucp**, **tip**, **cu** and remote logins.
 - Set **s0** register value to 1 to answer after one ring.
 - Lock the DTE interface to the selected speed. Many modems do not support a speed register, but you can store the current value with the **&W** command.
 - Disable modem printing of result codes when processing incoming calls. You may have to disable result codes too. Start the dialer with **ATE1** or its equivalent.
 - Set the modem to print extended result codes, like Busy, when dialing out. Better dialer performance is possible. Enter this in the **/etc/ppp/Dialers** file, if necessary.
 - Set the modem so it asserts only the carrier detect (CD) signal when the carrier is established with another modem.
 - Set the modem to disconnect the call and restore its saved values when the computer deasserts the data terminal ready (DTR) signal.

HP recommends that you verify that your modem connection is working correctly before configuring a PPP connection. This will detect any hardware, software, or modem configuration problems that are not related to the protocol.

To verify that the modem connection is working, use the following command to connect to a remote system.

```
cu -l device_file -s speed
```

You should be able to connect to a remote system using the **cu** command before configuring **pppd**. The remote system displays a login prompt and users can login to the remote system, verifying that the modem line is functional.

2-15. SLIDE: Configuring Outbound Connections

Configuring Outbound Connections

- **/etc/ppp/Devices**
- **/etc/ppp/Dialers**
- **/etc/ppp/Systems**
- **/etc/ppp/Autostart**

a64615

Student Notes

The following files are used for outbound calls:

- **/etc/ppp/Devices** associates dialer types with physical devices and speeds. **pppd** examines the file when it places a call. If no suitable speed is found or if all devices associated with that speed are busy, **pppd** tries again later.
- **/etc/ppp/Dialers** describes how to dial each type of modem attached to the HP-UX system that is to be made available for outbound calls.
- **/etc/ppp/Systems** contains the hostname or IP address of peers and how to connect with peers.

- `/etc/ppp/Autostart` starts `pppd` for on-demand outbound calls.

/etc/ppp/Devices

The **Devices** file associates dialer types with physical devices and speeds. Entries in the file have the following format:

dialer device speed [optional_parameters]

If the PPP connection is a direct connection, the dialer field should be *Direct*. Refer to the `ppp.Devices(4)` manpage for more information.

/etc/ppp/Dialers

The **Dialers** file describes how to dial each modem that is attached to the local system. The file **Dialers** is installed with PPP. Entries in the file have the following format:

dialer chat_script

Refer to the `ppp.Dialers(4)` manpage for more information.

You can create a separate list of modem descriptions in a file named **Dialers.local**. Entries in **Dialers.local** take precedence over entries in **Dialers**. You can use the entries from the **Dialers** file to guide you as you create new entries in **Dialers.local**. See the `ppp.Dialers(4)` manpage for more information about setting up dialer entries.

Since your modem may be used for other purposes besides PPP (for example, UUCP or for interactive users), it is best to set the modem's default parameters to accommodate dial-in applications and have outgoing UUCP or PPP dialers change them if necessary. The PPP-specific register settings in **Dialers** or **Dialers.local** ensure that an otherwise general-purpose modem will work as well as possible with PPP for the duration of the PPP session.

/etc/ppp/Systems

The **Systems** file contains the hostname or IP address of peer systems and how to connect to them. Entries in the file have the following format:

name when device speed phone_number chat_script

Refer to the `ppp.Systems(4)` manpage for more information.

/etc/ppp/Autostart

All outbound PPP connections are started through the user-generated **/etc/ppp/Autostart** file. When the local system is booted, a **pppd** process is started for the outbound link to the remote system. There is no example **Autostart** file; you must create it. It contains the command line necessary to start **pppd**:

pppd local_host:remote_host daemon_mgt_opt link_mgt_opt idle_timer

Refer to the **pppd(1)** manpage for information on command line options.

2-16. SLIDE: Configuring Inbound Connections

Configuring Inbound Connections

- **/etc/passwd**
 - create login entry for PPP user(s)
 - entry runs a **Login** shell script
- **Login** shell script
 - script executes **pppd** with appropriate options
 - check permissions on script

a64616

Student Notes

On machines that accept only incoming calls, **pppd** does not need to be started at boot time, since **pppd** is started when a PPP login occurs. Machines that both initiate and receive calls must start **pppd** at boot time, and must also prepare accounts for incoming connections. User accounts must be created in the **/etc/passwd** file for the system to be able to accept incoming calls.

When the local system receives a login, it does the following:

- Verifies the password by comparing it to the entry in the **/etc/passwd** file.

- If the login is successful, the **Login** shell script is run. **Login** starts **pppd** on the local system which will communicate with the **pppd** on the peer. The two **pppds** will negotiate and establish a PPP connection.

Create Accounts in /etc/passwd

- Create the **Login** shell script.

A PPP user's login shell script can be located anywhere and named anything.

- Check Permissions.

Following the creation of the **Login** shell script, make sure the script is executable with the following command:

```
# chmod 755 Login
```

2-17. SLIDE: Testing the Outbound Connection

Testing the Outbound Connection

- Reboot your system or start **pppd** manually.
- Check run status of **pppd**.
- Telnet to a system on the other end of the PPP link.
- Check **/var/adm/pppd.log** to confirm link status.

a64617

Student Notes

When PPP is installed, configured, running, and connected on both ends of the link, users should be able to access each peer machine using any TCP/IP application, such as **telnet**, **ftp**, etc. Once you have configured the PPP connection on both the local and remote systems, follow these steps to test the outbound connection.

1. Either reboot your machine or run **/etc/ppp/Autostart** to start **pppd**.
2. Check to make sure **pppd** was started:

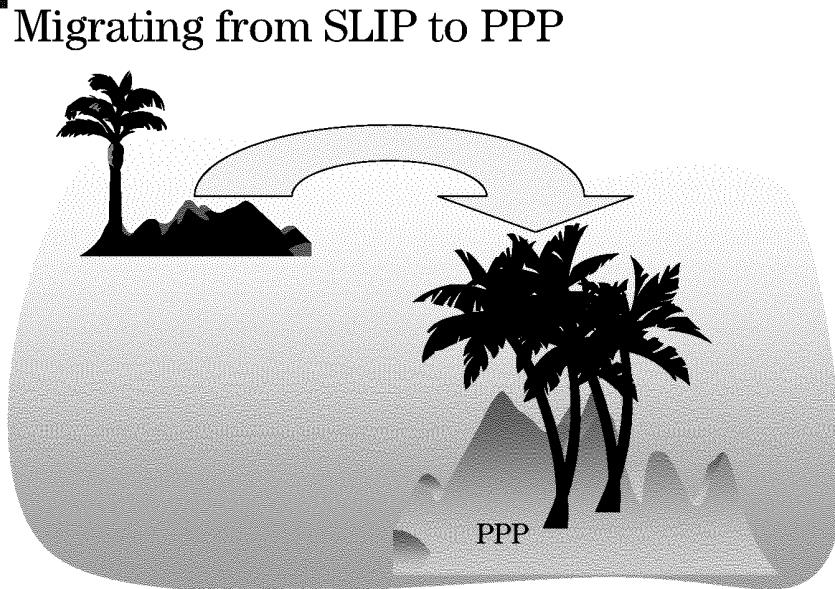
```
# more /var/adm/pppd.log
8/4-14:14:43-14902  PPP
8/4-14:14:43-14902  Version 2.0
8/4-14:14:43-14902  du0: pppd  robin:lark  auto  idle  150
```

3. Use **telnet** to bring up the link and type **^D**(Ctrl-Shift-D) to exit the login. There will be a half minute pause while the local system dials the phone; the modems establish a

carrier; the **Systems' chat** script completes; the answering **pppd** is started on the remote system; and the two **pppds** negotiate.

If either machine is connected to a local area network, you can set up IP routing on the two networks so that hosts on either network can communicate with hosts on the other, using machines on the ends of the PPP links as routers.

2-18. SLIDE: Migrating from SLIP to PPP



a64618

Student Notes

There are four main files used to set up serial line Internet protocol connections with the SLIP product:

- `/etc/ppl/ppl.remotes`
- `/etc/uucp/Systems`
- `/etc/uucp/Devices`
- `/etc/uucp/Dialer`

Table 2-1. Relationship of SLIP and PPP Configuration Information

SLIP Files	SLIP Fields	PPP Files	PPP Fields
/etc/ppl/ppl.remotes	remote_host local_host mask_protocol type uucp_name parity speed serial_line phone modem_control login_info command_name		
/etc/uucp/Systems	sysname time;retry device;protocol speed phone chat_script	/etc/ppp/Systems	name when device speed phone chat_script
/etc/uucp/Devices	device_type device_file calling_unit speed dialer_token	/etc/ppp/Devices	dialer device speed option
/etc/uucp/Dialers	modem_type chat_script	/etc/ppp/Dialers	dialer chat_script

The /etc/ppl/ppl.remotes file is the key to creating SLIP configuration files. The information for every dialout or direct entry can be extracted from the /etc/ppl/ppl.remotes file to construct the PPP configuration file. The dial-in entries in the ppl.remotes file have no corresponding place in the PPP configuration files.

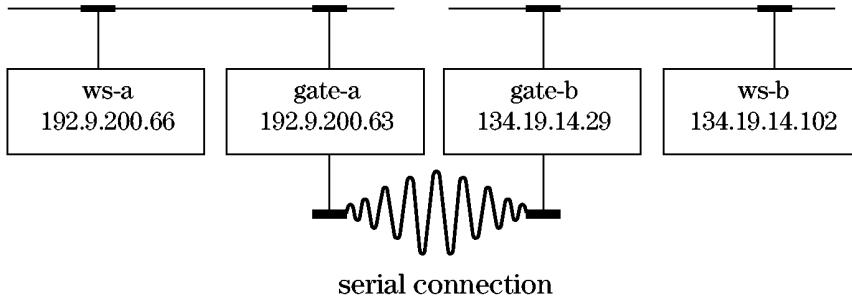
The /etc/uucp/Dialers file and /etc/ppp/Dialers file have identical formats. The /etc/ppp/Dialers.ex file is delivered with PPP-RUN. To avoid potential migration issues, the /etc/uucp/Dialers file is copied to the /etc/ppp/Dialers file during the swinstall process. If there is a need to move any entries from the /etc/ppp/Dialers.ex file to the /etc/ppp/Dialers file, it is up to the system administrator. It is also possible to create a /etc/ppp/Dialers.local file for system-specific dialers.

For more information about PPP configuration files, refer to the following manpages:

```
ppp.Dialers (4)
ppp.Devices (4)
ppp.Systems (4)
```

2-19. SLIDE: IP Routing over PPP

IP Routing over PPP



a64620

Student Notes

The UNIX host's IP implementation sees PPP as a point-to-point network connection between two known addresses. If neither end-point resides on an IP-based local area network (LAN), packets simply flow in both directions as soon as the PPP connection is established. When the PPP link connects a remote host to a LAN, provides a connection between two LANs, or connects a host or a LAN to the worldwide Internet, the systems involved must be concerned with routing.

Establishing static routes when the machines boot is cumbersome, because any topology change requires that all the machines involved, even remotely, be reconfigured by editing their boot-time shell scripts. An alternate to static routes is to use some automated system for updating all the hosts' routing tables, usually implemented as a daemon running on all the hosts involved. Many networks use the router Internet protocol (RIP) and run `gated` on their UNIX systems. If you use `gated`, you should invoke `pppd` with the netmask argument set to the same value as used on the LAN, if that subnet mask is different from the default for the class of your network.

- connecting a host to a LAN

There are two conventional approaches for connecting a set of standalone hosts to a LAN via PPP.

- separate network

The method that causes the least confusion is to assign a network or subnet for use by all remote machines.

- address resolution protocol (ARP) table manipulation

If a separate subnet number is unavailable for use by remote-access machines, it is possible to assign the remote machines addresses on the same subnet number as the departmental LAN.

- connecting two LANs

Suppose gate-a (192.9.200.63) and ws-a (192.9.200.66) are on one LAN, with gate-a supporting a modem. Also suppose gate-b (134.19.14.29) and ws-b (134.19.14.102) are on another LAN across town, with gate-b supporting a modem.

gate-b's **pppd** should be started as

```
pppd gate-b:gate-a auto idle 130
```

and gate-b should have a route similar to

```
route add net 192.9.200.0 gate-a 1
```

ws-b should have a route similar to

```
route add net 192.9.200.0 gate-b 1
```

Similarly, gate-a's **pppd** should be started as

```
pppd gate-a:gate-b auto idle 130
```

and gate-a should have a route similar to

```
route add net 134.19.14.0 gate-b 1
```

or, depending upon the structure of LAN b, maybe

```
route add net 134.19.0.0 gate-b 1
```

ws-a should have a route such as

```
route add net 134.19.14.0 gate-a 1
```

or, again depending on the structure of LAN b, perhaps

```
route add net 134.19.0.0 gate-a 1
```

- connecting a host or LAN to the Internet

If your LAN is connected to the Internet, or if you have arranged an account at a point of presence (POP) of a PPP or SLIP Internet connectivity vendor (say `foo.net`), then you should arrange for your default route to point through the gateway at the other end of the PPP connection. If `hotel` supported a modem to call a POP, it would start its PPP daemon as

```
pppd hotel:pop.foo.net auto idle 240
```

and would arrange a route as

```
route add default pop.foo.net 1
```

Any hosts on the LAN *behind* `hotel` would set a route similar to

```
route add default hotel 1
```

A machine's default route should point to the next machine along its path *outward* to the Internet. If `hotel` were a remote machine dialing into one machine in a complex corporate internet, its default route should point to that hub machine, expecting that the hub will deal with the issues of routing `hotel`'s packets to their destination, whether on the LAN or elsewhere on the corporate internet or onto the Internet.

2-20. SLIDE: Security over PPP

Security over PPP

- packet filtering
- time-to-call restrictions
- dialback
- CHAP authentication

a64619

Student Notes

It is impractical to impose thorough security policies on each internal host of the networks linked by a PPP connection. But PPP's strong security features support a variety of techniques that strengthen your network's ability to prevent loss. In most cases, a single connection can be supported by more than one of PPP's security features. For example, a connection might use any of the following:

Packet Filtering

- static packet filtering

Establish a security policy before you write a packet filter. A security policy is a statement based on thorough analysis of access needs, vulnerabilities, and real or perceived threats to your assets. You must identify the types of network traffic associated with these issues before you can create a packet filter that supports your security policy.

- security policies

In general, all security policies are based on one of two opposing strategies. Both types of policies are supported by PPP filters.

The first strategy permits a few specific services and blocks everything else. If you follow this philosophy, a service will be unavailable if you commit an error of omission. This is a fail-safe, or closed, policy.

The second strategy blocks only specific services and permits everything else. If you begin from this premise, an error of omission may leave you unintentionally vulnerable when a fragile service is not blocked.

- filter file rulesets

When **pppd** starts, the software checks for a filter file. A filter file contains rulesets for filtering packets.

- filters

A ruleset is made up of one to four filters that regulate the response to a packet. The filter's actions are defined by its initial keyword. Each type of filter can be used one time per connection.

- filter stanzas

Each filter is composed of a filter name followed by one or more stanzas (rules). Each packet passing through the interface is compared to the rules in the stanzas until a match is found, completing the filter operation. The ordering of the stanzas is extremely important.

Time-to-Call Restrictions

The second field on each line in the **Systems** file specifies the times **pppd** is allowed to attempt to establish connections. Two benefits of time restrictions are

- Assurance that there will be personnel on each end of the link to monitor the connection attempt.
- Assurance that connections take place when the most favorable telephone calling rates apply.

See manpage **ppp.Systems(4)** for details. The **when** field is very flexible. It can be configured by day and by hour, with many different combinations allowed for the same connection.

Dialback

PPP supports the ability to maintain a connection when calling a modem that has a dial-back security feature. The **Systems** file chat script option **\M** allows this by disabling delivery of **SIGHUP** to **pppd**. This signal usually results from loss of **Carrier Detect** and tells **pppd** to abruptly disconnect from the active session.

- dial-back process

Typically, an answering modem with dial-back capability responds to a call by taking the following steps:

1. Challenges incoming callers with a prompt string.
2. Accepts the input identifying the caller.
3. Hangs up the call.
4. Calls a number associated with the caller's identification.
5. Re-establishes a carrier.

The calling modem might then demand the same type of identification before allowing remote data to flow through its serial interface to the local system.

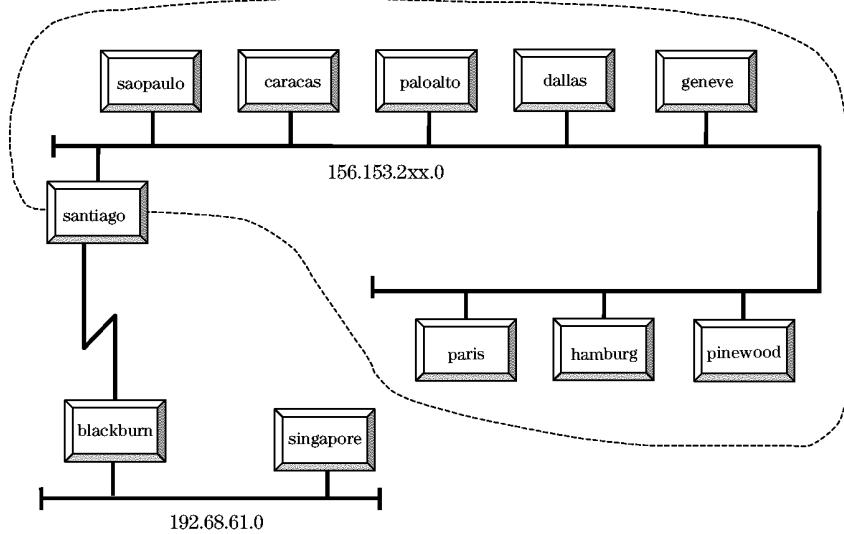
CHAP Authentication

PPP implements both the password authentication protocol (PAP) and the challenge handshake authentication protocol (CHAP). If `pppd` is invoked with any of the authentication options, it demands that the peer (either calling or called) authenticate itself. The `ppp.Auth(4)` file contains pairs of either names and secrets for CHAP negotiation, or usernames and passwords for PAP negotiation. If a peer provides a name or username, its password must match that found in the `Auth` file or the authentication phase fails and the connection is terminated. Each name/secret pair in the `Auth` file may be followed by address patterns restricting the peer's negotiated IP address. If an address restriction is specified for a particular name and the peer's negotiated IP address does not match the restriction address patterns, `pppd` terminates the connection.

The `rechap` interval option instructs `pppd` to periodically (every interval seconds) challenge the peer to authenticate itself. If the peer fails the new challenge, the link is terminated.

2-21. LAB: Point-to-Point Protocol

Configuration for Point-to-Point Lab



a646149

Student Notes

For additional information on the configuration of the point-to-point protocol see manpages `pppd(1)`, `ppp.Auth(4)`, `ppp.Devices(4)`, `ppp.Dialers(4)`, `ppp.Filter(4)`, `ppp.Keys(4)`, `ppp.Systems(4)`, and "Installing and Administering PPP, Edition 1", B2355-90137.

Directions

In this lab, we will create a point-to-point link between two computers on different subnets. We will

- Connect the designated serial ports with a null-modem cable.
- Then, on each of the systems linked by the cable:
 - Check that the chosen serial port is read/write.

- Edit `/etc/ppp/Autostart` to include a line like

```
pppd local_IP:remote_IP auto dedicated ignore-cd
```

- Make the file executable.

- Edit `/etc/ppp/Systems` to include a line like

```
remote_IP Any serial_port baud_rate 0
```

- Edit `/etc/ppp/Devices` to include a line like

```
Direct serial_port baud_rate
```

- Test the connection with `/etc/ping`.

Point-to-Point-Protocol Configuration

For all systems:

- Configure as in the above figure or as the instructor may otherwise direct. Note that in the following instructions, the node names used are from the figure. Your instructor will *map* these into names used at your site, as well as indicate which systems will be used as the PPP gateways and which nodes will be *moved* to the 192.68.61 net.
- Verify that
 - `/etc/hosts` is current and complete. Generally, this means adding the IP addresses and hostnames of the systems on the net 192.68.61 side of the PPP link.
 - only static routing is being used, that is, no `gated`, `routed`, or `rdpd` daemons are running).
- The systems will need to adjust their routing so that they can reach the other networks (192.68.61 or 156.153.2xx).
 - Run `/sbin/set_parms initial` on the net 192.68.61.0 systems to change hostnames, IP addresses, and default routes. The net 192.68.61.0 non-PPP systems should use `blackburn` as the default gateway. The net 192.68.61.0 PPP system should use `santiago` as the default gateway.

NOTE: You may need to delete the existing default route first.

Upon completion, verify local network connectivity using `ping`.

- Replace `<ip_addr_of_santiago>` and `<ip_addr_of_blackburn>` in the following examples with the actual IP address of the node `santiago` or `blackburn` respectively. The net 156.153.2xx.0 non-PPP systems will need to modify `/etc/rc.config.d/netconf` to add route statements in for the 192.68.61.0 network. Add statements similar to the following:

```
ROUTE_DESTINATION[1]=192.68.61.0
ROUTE_MASK[1]="""
ROUTE_GATEWAY[1] =
ROUTE_COUNT[1]=1
ROUTE_ARGS[1]=""
```

In addition, interactively set up this route using the command

```
# route add net 192.68.61.0 <ip_addr_of_santiago> 1
```

The net 156.153.2xx.0 PPP system will need to modify `/etc/rc.config.d/netconf` to add route statements in for the 192.68.61.0 network. Add statements similar to the following:

```
ROUTE_DESTINATION[1]=192.68.61.0
ROUTE_MASK[1]="""
ROUTE_GATEWAY[1] =
ROUTE_COUNT[1]=1
ROUTE_ARGS[1]=""
```

In addition, interactively set up this route using the command

```
# route add net 192.68.61.0 <ip_addr_of_blackburn> 1
```

Upon completion, verify local network connectivity using `ping`.

1. On the two PPP systems:

Verify that the `PPP-RUN` fileset is loaded and that the PPP drivers are in the kernel. Normally, this is done when loading either the core networking bundle or the LAN/9000 networking product.

2. Attach a null modem cable between the two systems that will be linked via the PPP connection. For ease of configuration, it is suggested that you use the same serial port number on both systems (e.g., RS-232 Interface (#1)).

3. Using SAM, add the serial interface to each of the systems. As this is a *hard-wired* interface you can configure this port as a *terminal*.

4. Check the device just created on each of the systems.

5. Turn off the **getty** for the newly created device on each of the systems. Change the word *respawn* to *off* for the created **tty** in **/etc/inittab**.

For example, change **a0:3:respawn:/usr/sbin/getty -h tty0p0 38400**
to **a0:3:off:/usr/sbin/getty -h tty0p0 38400**.

Then, have **init** reread **/etc/inittab** to kill off the running **getty**.

6. Check out the basic link. On one system, type

```
# cat < /dev/tty0p0
```

While on the other system, type

```
# echo hello world > /dev/tty0p0
```

7. Create the PPP configuration files in **/etc/ppp**.

NOTE:

The keywords *Any* and *Direct* must be capitalized and the file
/etc/ppp/Autostart must be executable.

In the following examples, replace **<remote_ip_address>** with the IP address of the node on the remote side of the PPP link. For example, if node **santiago** is being configured, use the IP address of **blackburn**.

8. Start the PPP daemon.
9. Check to make sure everything started OK. Make sure the connection is established. The network interface name is `du0`.
10. It is also a good idea to establish some baseline parameters for the link at this time:
11. To shut down the daemon use a `kill` signal. Do *not* use a `kill -9`, as this may leave the tunnelling device in a busy state and require a system reboot to clear.
12. Once the link has demonstrated its reliability, try changing the data rate in the `/etc/ppp/Devices` and `/etc/ppp/Systems` files on both systems to other values, stop and restart the PPP daemons, and observe what happens.

Module 3 — Troubleshooting of Common TCP/IP Problems

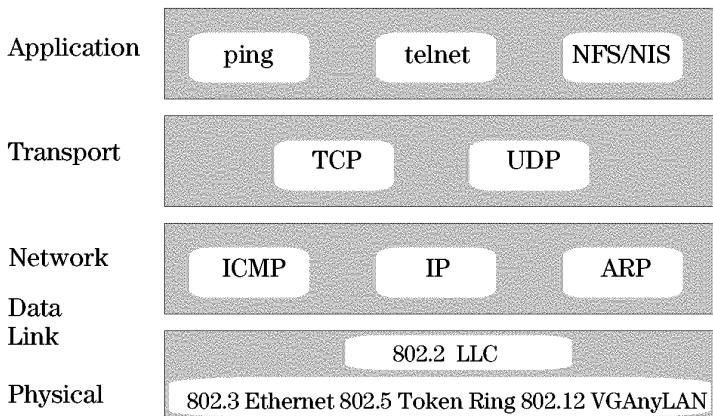
Objectives

Upon completion of this module, you will be able to do the following:

- Detect, characterize, and correct common TCP/IP faults.
- Use **tracing** and **logging** to identify faults that vary with system conditions.

3-1. SLIDE: The TCP/IP Network Protocol Stack

The TCP/IP Network Protocol Stack



a63028

Student Notes

Understanding the network protocol stack is a good prerequisite for searching for faults in a network. The key to proper network troubleshooting is to identify in which network layer the problem exists. Unless you are reasonably certain where the fault exists, always search from the bottom up, that is, from the physical layer to the application layer.

We will cover the ten most common faults in each of the network layers. Understanding these faults should help you

- understand the network architecture
- search through the network layers for a fault
- apply the appropriate troubleshooting commands
- find the most common faults quickly

In this chapter we will step through each network layer from the bottom up. Faults that depend on special applications, such as network file system (NFS), (NIS), or (DNS) will be covered in other modules.

TCP/IP protocols do not cover all seven layers of the International Standards Organization/ (ISO/OSI) reference model. Here is a short repetition of the network protocol stack:

The transmission control protocol/Internet protocol (TCP/IP) network protocol stack is divided into four layers.

Application Layer

The application layer is the layer at which application programs run, both client and server (or daemon). It can be further divided into

- (ARPA) services, which are the standard TCP/IP services, such as `telnet`, `ftp`, the Berkeley `r` commands `rlogin`, `rcp`, and `rsh` (`remsh` for HP-UX), and others; these tend to access the transmission layer directly, via the Berkeley Socket Interface.
- Remote procedure call (RPC) based services, such as NFS and NIS, which use the two new layers not shown in the diagram above, eXternal Data Representation (XDR) and RPC, rather than directly accessing the Transmission layer protocols.

Most applications are divided into client and server sides. With many services, the client program, such as `telnet`, runs on the *local* system and uses the lower layer protocols to contact the server daemon, such as `telnetd`, to provide the requested service.

Transport Layer

This layer provides two transmission protocols to the application layer programs. These are used to facilitate separate client/server communication channels via the *port* numbering scheme. The Berkeley socket interface is used by applications to gain access to TCP or UDP ports. This layer provides the multiplexing/demultiplexing of separate logical channels across the single Internet layer channel.

Transmission Control Protocol (TCP)

TCP provides a reliable, stream-oriented, connection-oriented transport mechanism. It allows for an end-to-end virtual circuit between the client program and the server daemon that is two-way, flow-controlled, and flexible. Both the client and the server using TCP have the connection opened at the same time, and both view it as a stream-oriented input/output device, just like a file.

Reliability is guaranteed via the positive acknowledgment with retransmission mechanism, which ensures that the messages handed down by the application and divided into segments (up to 64 KB per segment), are clearly marked, transmitted, and acknowledged. In addition, a flow-control mechanism ensures that the rate of retransmissions in a congested network is adjusted appropriately.

User Datagram Protocol (UDP)

UDP provides a datagram delivery service to application layer programs; it offers an unreliable, connectionless transport mechanism. Since it provides no guarantee of delivery, it is used by applications that either do not care about guaranteed delivery (such as the `users` program) or that provide their own acknowledgment/retransmission facility (such as NFS/ NIS). UDP offers an optional `checksum` for data integrity.

Network Layer

The network layer provides the ability to interconnect multiple networks. It is comprised of one main protocol, Internet protocol (IP), and two supporting protocols, ICMP, and ARP.

Internet Protocol (IP)

The Internet protocol (IP) is responsible for three major functions: addressing, routing, and fragmentation/reassembly. It is an unreliable, best-effort connectionless datagram delivery mechanism.

Internet Control Message Protocol (ICMP)

ICMP is a protocol which is actually encapsulated inside IP. It is used to pass error and control messages between IP software across the Internet. It is typically used to provide flow-control by a gateway to return a datagram to a sending host indicating that the desired host/network is unreachable, and is used by the `ping` program to provide an echo request/reply mechanism for troubleshooting and performance measurement.

The Address Resolution Protocol (ARP)

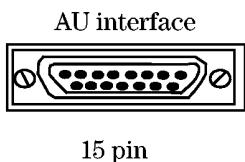
ARP is used by IP when IP determines that the destination host is located on a directly connected network (not point-to-point). It uses a cache in kernel memory (the `arp` cache) to provide the hardware or MAC-level address (the Ethernet address in the case of an Ethernet network) of a host on the local network, given the logical IP address of the desired host. The cache keeps the most recently referenced IP <->Ethernet addresses, but if the desired mapping is not found, ARP uses an *arp broadcast* to solicit the requested information from the hosts on the local network.

The Physical Data Link Layer

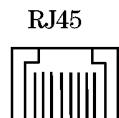
The physical data link layer provides the actual physical means of moving data across the network. Many different network layer protocol are used by UNIX® networks, including Ethernet, token ring, FDDI, SLIP, and X.25.

3-2. SLIDE: Connection Type

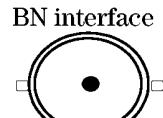
Connection Type



AU interface
15 pin



RJ45
10BASE-T cable



BN interface
10BASE2 cable

a64621

Student Notes

Ethernet cards support three types of LAN interface ports:

- AUI or DIX interface—A 15-pin AUI interface, which is connected with an AUI cable to an external medium attachment unit (MAU), a transceiver.

A transceiver can be connected to

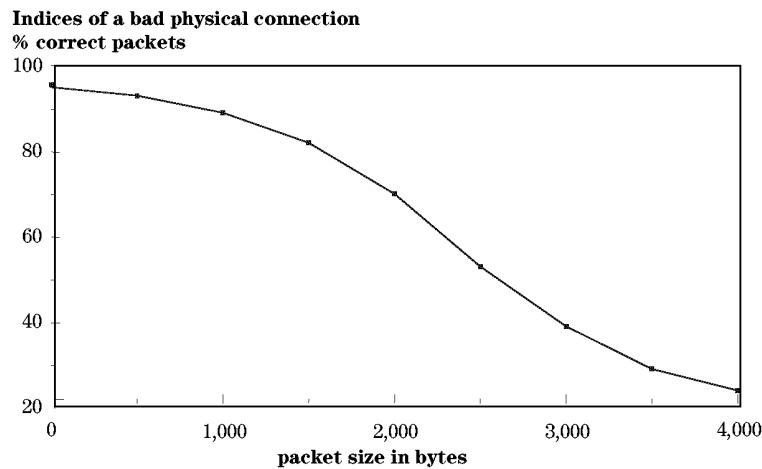
- thick Ethernet or yellow cable (10Base5)—The cable is attached by an N-Series connector or Vampire Tap.
 - thin Ethernet (10Base2)—The correct cable is RG58, which is attached by a BNC connector.
 - twisted pair (10BaseT)—It is attached by an 8-pin RJ45 port.
- BNC interface—The card offers an internal built-in transceiver. The cable is directly attached to BNC port of the card.

- RJ45 interface—The card offers an internal built-in transceiver. The cable is directly attached to the TP port of the card.

Different Ethernet cards integrate one, two, or all three types of the above ports.

3-3. SLIDE: Recognizing a Bad Physical Connection

Recognizing a Bad Physical Connection



a64622

Student Notes

Any of the following are possible reasons for a bad physical connection:

- The wrong cable type has been used—For example, you have used RG59 with 60 Ohm impedance. The correct cable for thin Ethernet is RG58 with 50 Ohm impedance.
- The Ethernet is not correctly terminated—Both ends must be terminated by a 50 Ohm resistance.
- The cable is defective or damaged.
- The adapter, receiver or tap has a bad contact.
- The communication line is bad—for example, the telephone linkage.

How can you recognize a bad physical connection? One way is to ping the remote host with increasing packet sizes and determine the portion of lost packets. If the portion of lost packets increases with the packet size, the fault will most likely be a bad physical connection.

Add new text to describe how to adjust the packet sizes.

The proper command format is

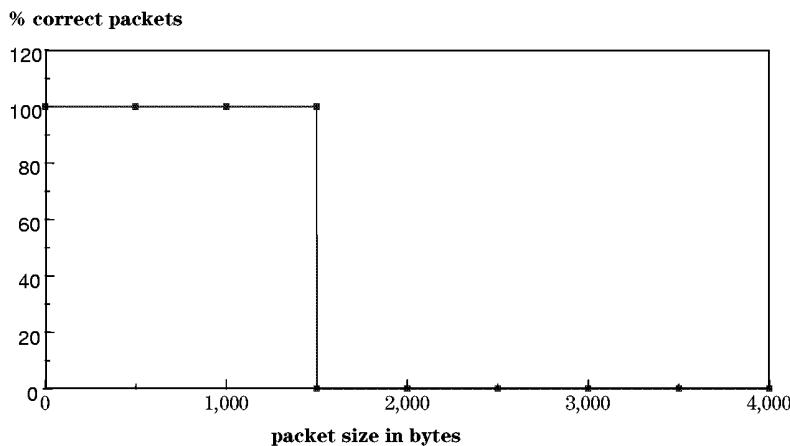
```
# ping remote_host packet_size
```

The default packet size is 64 bytes. The minimum is 8 bytes, and the maximum is 4095 bytes.

(See **ping(1)** for details.)

3-4. SLIDE: MTU-Size

MTU-Size



a64623

Student Notes

Fragmentation and reassembly are performed by IP when a datagram traverses several intermediate networks with differing maximum transmission units (MTUs). For instance, the MTU for Ethernet is 1500 bytes, for token bus is 8192, and for serial line IP (SLIP), it may be as small as 128 bytes. When a gateway forwards a datagram from one interface with a larger MTU than the outgoing network MTU, the gateway IP fragments the incoming datagram into smaller fragments that will fit the MTU of the outgoing network. The destination machine performs the reassembly by starting a timer when it gets the first fragment. If the timer expires before all fragments are received, the surviving fragments are discarded.

Reasons for an MTU-size fault include the following:

- A bridge is the most common cause for an MTU size fault. In the ISO/OSI reference model, a bridge is a connector in layer 2. In layer 2 there is no possibility to fragment and reassemble packets. For example, a bridge that connects a token ring with an Ethernet cannot fragment a token ring packet with a size of 4096 bytes into three Ethernet packets with a size of 1500 bytes. In this case, the MTU size in the token ring must also be set to 1500 bytes.

- Corresponding to the ISO/OSI reference model, an IP router fragments IP datagrams. For some routers an option must be set to allow or disallow fragmenting. Determine if there is an option for the IP driver to fragment datagrams.
- A router may have an error. Rarely, some PCs that work with an IP router software don't fragment an IP datagram. Such a router must be treated as a bridge. The MTU size must be the same in all physical segments.
- The IP software offers a flag, **DON'T FRAGMENT**. Some implementations (for example for the **BOOTP**- or **DHCP**- protocols) set this flag. In this case, the router is not allowed to fragment IP datagrams. This fault is very hard to detect, because most applications do not set this flag, yet still work properly.

NOTE: Path MTU discovery always uses the **DON'T FRAGMENT** flag.

3-5. SLIDE: IP Addressing Faults

IP Addressing Faults

- **IP Addresses**

Class A 1-127 nnn.hhh.hhh.hhh

Class B 128-191 nnn.nnn.hhh.hhh

Class C 192-223 nnn.nnn.nnn.hhh

Class D 224-239 mmm.mmm.mmm.mmm

- **Reserved Addresses**

0 default destination for routing

127.0.0.1 loopback or localhost

255.255.255.255 broadcast address - own network address is unknown

a64625

Student Notes

Although the IP address scheme is very easy, many mistakes are made with IP addressing.

It is important to note the following:

- The first byte of a class C address ranges from 192–223. Values from 224–239 belong to a class D, a reserved class for multicasting.
- The network address 127 is reserved for loopback.
- The network address 255.255.255.255 is reserved for broadcasting if the network address is unknown. This is the case for boot devices such as network printers or X-stations.
- When a network number is unknown, a host can use 0 as a substitute. Addresses beginning with 0 have special meanings:

0.0.0.0 This host on this network

0.0.0.x

Host x on this network

Such addresses are useful at boot time, when the booting system does not yet know its own IP address.

3-6. SLIDE: Network and Broadcast Addresses

Network and Broadcast Addresses

- Network IDs

All bits in the local portion are set to 0. **00000000** (0 decimal) represents the network address.

15.0.0.0

150.143.0.0

196.8.197.0

Always specify 4 bytes, even for a network ID:

RFC converts

150.143	to	150.0.0.143
---------	----	-------------

196.8.197	to	196.8.0.197
-----------	----	-------------

- Broadcast address

All bits in the local portion are set to 1. **11111111** (decimal 255) represents the broadcast address.

15.255.255.255

150.143.255.255

196.8.197.255

a646106

Student Notes

The **hostid** is the last (farthest right) portion of an IP address.

In a network address, all bits of the **hostid** are equal to 0. When specifying a network address, *always* write all 4 bytes completely.

The RFCs define the following address segments.

- When a three-part address is specified, the last part is interpreted as a 16-bit quantity and is placed in the right-most two bytes of the network address.
- When a two-part address is supplied, the last part is interpreted as a 24-bit quantity and is placed in the ending three bytes of the network address.

A broadcast address is an address that has all bits of the **hostid** equal to 1. A broadcast cannot be used as a **hostid**.

3-7. SLIDE: Subnetting Errors

Subnetting Errors

A subnet mask

- Extends a network ID from left to right.
- Reserved subnets are
 - all bits equal to zero in the extended network portion
 - all bits equal to one in the extended network portion
- Use option **net** for **route** command.

a64626

Student Notes

A subnet mask always extends the network address. The larger the network address becomes, the smaller the **hostid** becomes. Although the netmask is not defined in the RFC, it should extend the network address with continuous bits from left to right. Many routers accept only continuous bit netmasks. A single byte results in the netmask value

128, 192, 224, 240, 248, 252, 254, 255

Only Internet providers should group different network addresses to a **supernet**. For static routing within a subnet, always use the keyword **net**.

Example:

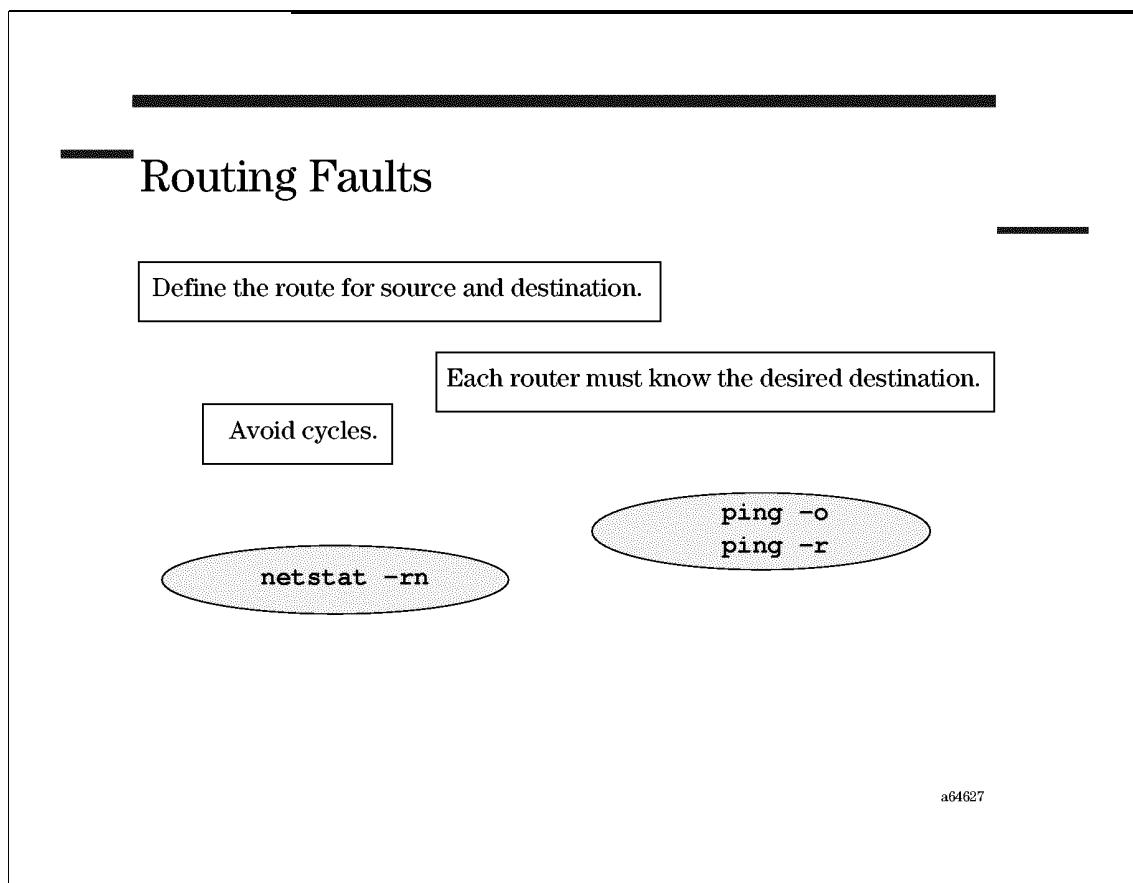
```
route add net 15.138.128.0 15.138.192.6 1
```

- All rules for IP addresses remain the same for subnetting.
- All bits equal to 0 in the **hostid** represent the network address.

- All bits equal to 1 in the **hostid** represent the broadcast address.
- All bits equal to 0 in the extended network portion of the IP address represent the entire subnet as an external sees the network, and are therefore reserved.

Although a broadcast is not supported over a router, all bits equal to 1 in the extended network portion represent a broadcast address for all subnets and therefore are reserved.

3-8. SLIDE: Routing Faults



Student Notes

Static routing is very trivial but can cause a large number of entries for a large network.

- For a perfect connection, a route must be very well defined—both back and forth. Unfortunately, this is often overlooked in WAN connections.
- For a certain destination, a route table entry can announce only the *next* router.
- All routers on a path must know the next router to a desired destination. This includes default routers.
- Avoid cycles in a routing path.

A facility for analyzing routing problems is **traceroute** (located in **/usr/contrib/bin**). It determines the path by provoking an ICMP error message. **traceroute** sends IP packets with increasing values for hop count. The first hop count is 0. Therefore the first router is forced to send an ICMP error message, which identifies the first router. The second hop count is 1 and

identifies the second router, and so on. For a perfect connection, **traceroute** must work in both directions.

3-9. SLIDE: Reverse Name Resolution Failure

Reverse Name Resolution Failure

- /etc/hosts
 - 15.138.135.2 koeln k4711
 - rlogind, remshd, rshd
 - mountd, nfsd
 - rlpd
 - rlpdaemon
- /etc/hosts.equiv
 - k4711

a64629

Student Notes

Addressing a host in TCP/IP protocols is done exclusively with IP addresses. No hostname is sent to the recipient. Many programs need a reverse name resolution for proofing the hostname against authentication. Therefore, the IP address of the sender is mapped to a hostname. The mapping can be resolved by domain name service (DNS), network information service (NIS), or the local /etc/hosts file.

In any case, use the canonical form of the mapping because, in practicality, alias names are never checked against the database.

Reverse name resolution is necessary for

rlogind, remshd, rshd	in /etc/hosts.equiv and \$HOME/.rhosts
mountd, nfsd	in /etc/exports and /etc/xtab.
rlpd	in /etc/hosts.lpd or /etc/hosts.equiv

rlpdaemon in /var/spool/lp/.rhosts, /etc/hosts.equiv or /var/adm/inetd.sec

- Other applications

— Network Nodemanager

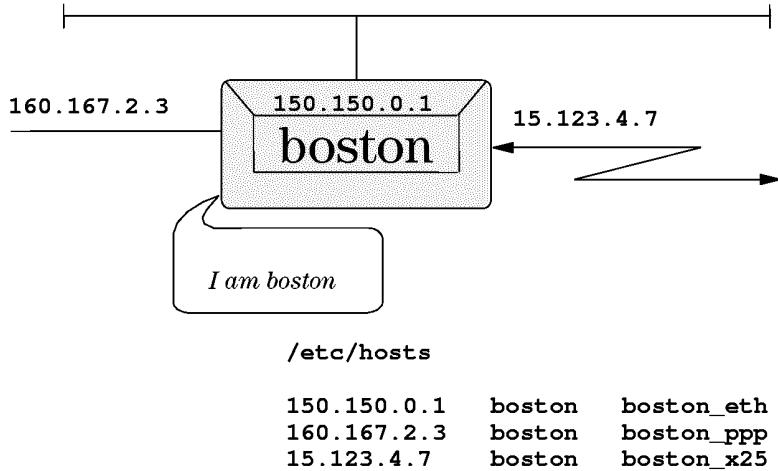
— arp

— netstat

Keep in mind that other programs will hang if the NIS or DNS reverse name resolution is not available. This is true for `arp` or `netstat`. These, and most other commands, display the hostname instead of the IP address. The hostname is resolved by reverse name resolution.

3-10. SLIDE: Multihomed Hosts

Multihomed Host



a64630

Student Notes

A multihomed host is a host with several interfaces and, therefore, several IP addresses.

A hostname must be mapped to an IP address and vice-versa. But which IP address is the best to map?

Name resolution always depends on a **sortlist**. This **sortlist** is not always best for all clients. On the other hand, for reverse name resolution, it is better to receive the same hostname for both interfaces. This will be very important if several paths to a destination exist.

For applications depending on reverse name resolution, it is nearly impossible to handle several names for the same host. This is especially true for

- network file system (NFS)
- Network Node Manager (NNM)

The following rule does not solve all problems, but still is a good rule:

For reverse name resolution, always supply the same hostname for all interfaces. This hostname is the canonical form or the official name of the host.

In order to distinguish among different interfaces, add an individual alias name for each interface.

Example:

```
/etc/hosts      150.150.0.1  boston  boston_eth  #  ethernet interface  
                  160.167.2.3  boston  boston_ppp   #  serial line interface  
                  15.123.4.7   boston  boston_x25  #  X25 WAN interface
```

Try to transfer these rules for DNS accordingly.

3-11. LAB: Finding and Resolving Network Faults

Purpose

To enable students to recognize faults more easily through the use of common troubleshooting commands.

Directions

The theme of this lab is to present students with a network of hosts whose operation is compromised by several faults, each at different levels of the TCP/IP protocol stack.

1. Complete the following:
 - a. How can you find the network adapters available on your system?
 - b. For each of the interfaces determine the following:
 - i. the hardware address
 - ii. the link layer level protocol
 - iii. the hardware state
 - iv. the IP address
 - v. the subnet mask
 - vi. the broadcast address
 - vii. the MTU size
2. ping the following addresses:
 - a. the network broadcast address
 - b. a broadcast address with all bits equal to 1.
 - c. an address with all bits equal to 0

Which hardware address(es) will your ARP cache contain?

3. Answer all parts of the following:

- a. Determine how many collisions occurred on your Ethernet segment.

The **spray** command is comparable to the **ping** command, but is used to help provide performance measurements of the RPC based applications.

- b. Activate the **rpc.sprayd** server on your system.
- c. Send 10000 packets to a neighbor's system using the **spray** command.
- d. What was the effect on the collision rate?

4. Perform the following tasks:

- a. Determine the hostname and nodename of your system.
- b. Temporarily change the nodename to *guardian*.
- c. What, if anything, has happened to your hostname?
- d. Cross check your hostname with your actual IP address and the reverse.
- e. Set your nodename back to its original value.
- f. Cross check your hostname with your actual IP address and the reverse.

5. Login as root on your system. Issue the command:

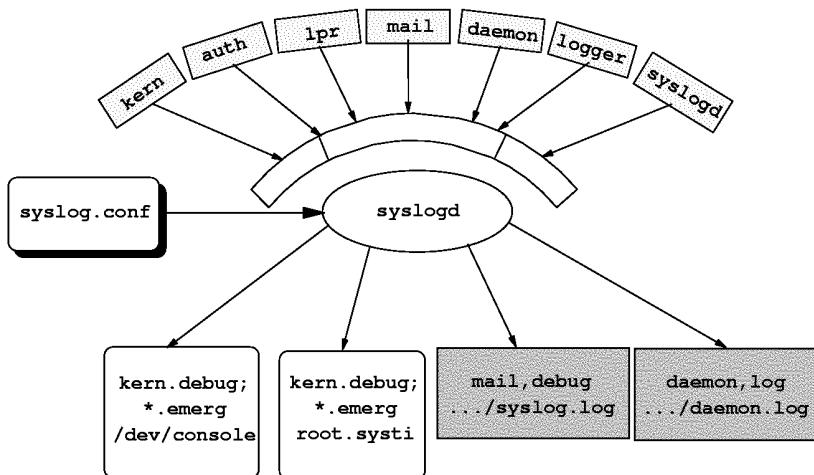
```
# telnet localhost
```

- a. Determine which sockets are being used by **telnet**.
- b. Determine which sockets are the telnet clients and which are the servers.

6. Use the command `arp -a` to get a list of the hardware addresses of systems connected to the local network.
 - a. Test the link level connectivity to that system.
 - b. Can this test be used across a router?

3-12. SLIDE: The **syslog** Subsystem

The **syslog** Subsystem



a64631

Student Notes

Several facilities like the kernel or the Internet servers log informational and error messages through **syslog**. You can monitor these messages by running **syslogd**. For different facilities messages are gathered at different levels and directed to files, users, the console, or to a **syslogd** running on another host.

syslogd is started at startup time in run level 2 by:

```
/sbin/rc2.d/S220syslogd ---> /sbin/init.d/syslogd
```

The process ID (PID) of **syslogd** is written to

/etc/syslog.pid

syslogd gathers information from

/dev/klog a buffer for kernel messages

/dev/log.un a UNIX domain socket for facilities using the **syslog** system call

/dev/log named pipe for facilities using the **syslog** system call but not supporting the UNIX domain sockets

514/udp a UDP port for messages from another **syslogd**

3-13. SLIDE: Computer Facilities

Computer Facilities

- **kern** kernel messages
- **user** applications messages
- **mail** mail messages
- **daemon** network daemon messages
- **auth** authorization messages
- **lpr** spooler messages
- **syslog** **syslog** messages
- **local0** own messages

a64632

Student Notes

Several facilities use the **syslog** subsystem to log their messages. In general, each application can use the **syslog** subsystem by calling the **syslog** system call.

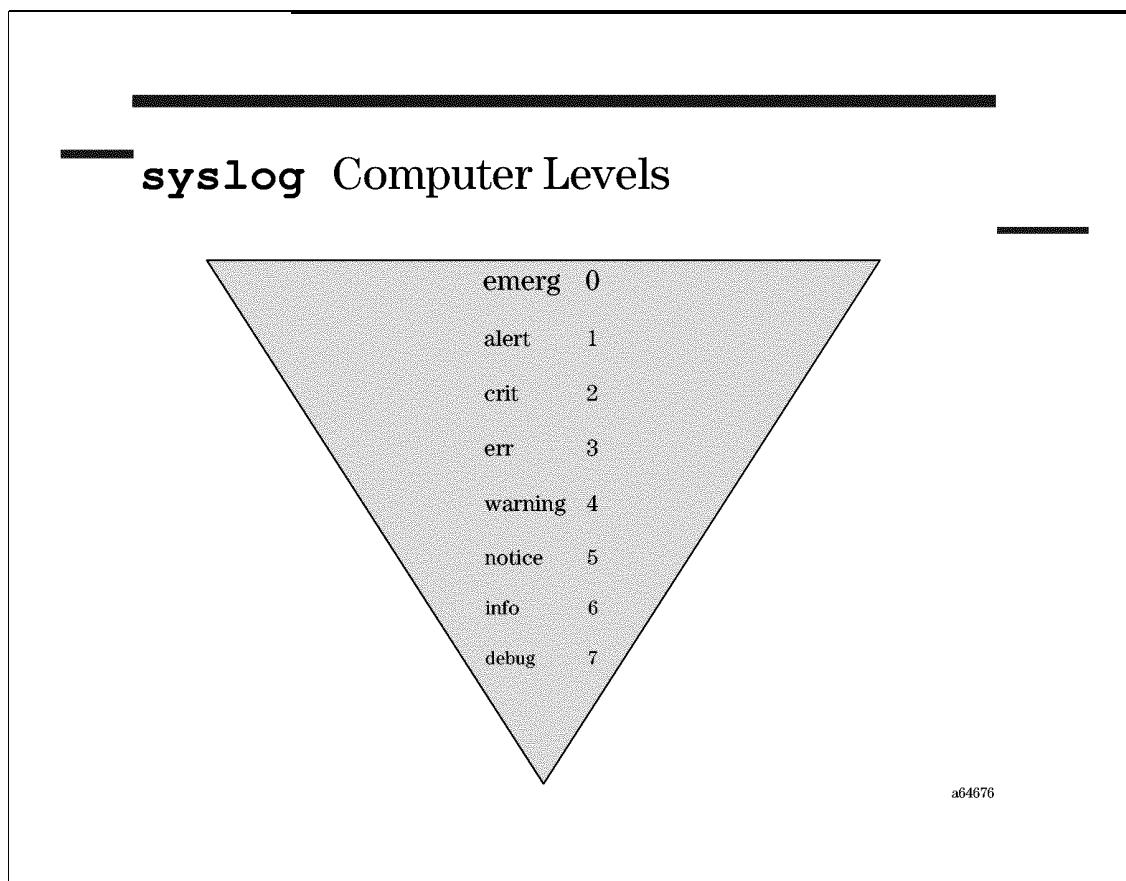
Facility	Definition
kern	Messages generated by the kernel.
user	Messages generated by user processes. This is the default subsystem for an application process. Each application can log messages by the syslog system call. The system call must be implemented by the programmer.
mail	Messages generated by the mail system.
daemon	Messages generated by network daemons such as inetd , ftpd , bootpd , automount , named , gated .

auth	Messages generated by the authorization system such as login , su , getty .
lpr	Messages generated by the line printer spooling system such as lp , lpdaemon .
syslog	Messages generated by syslogd itself.
local0	Messages reserved for local use—similarly, for local1 through local7 .

Many network daemons support options for extending **syslog** information:

```
inetd -l
automount -v
ftpd -l
named -d debug level
bootpd -l debug level
```

3-14. SLIDE: syslog Computer Levels



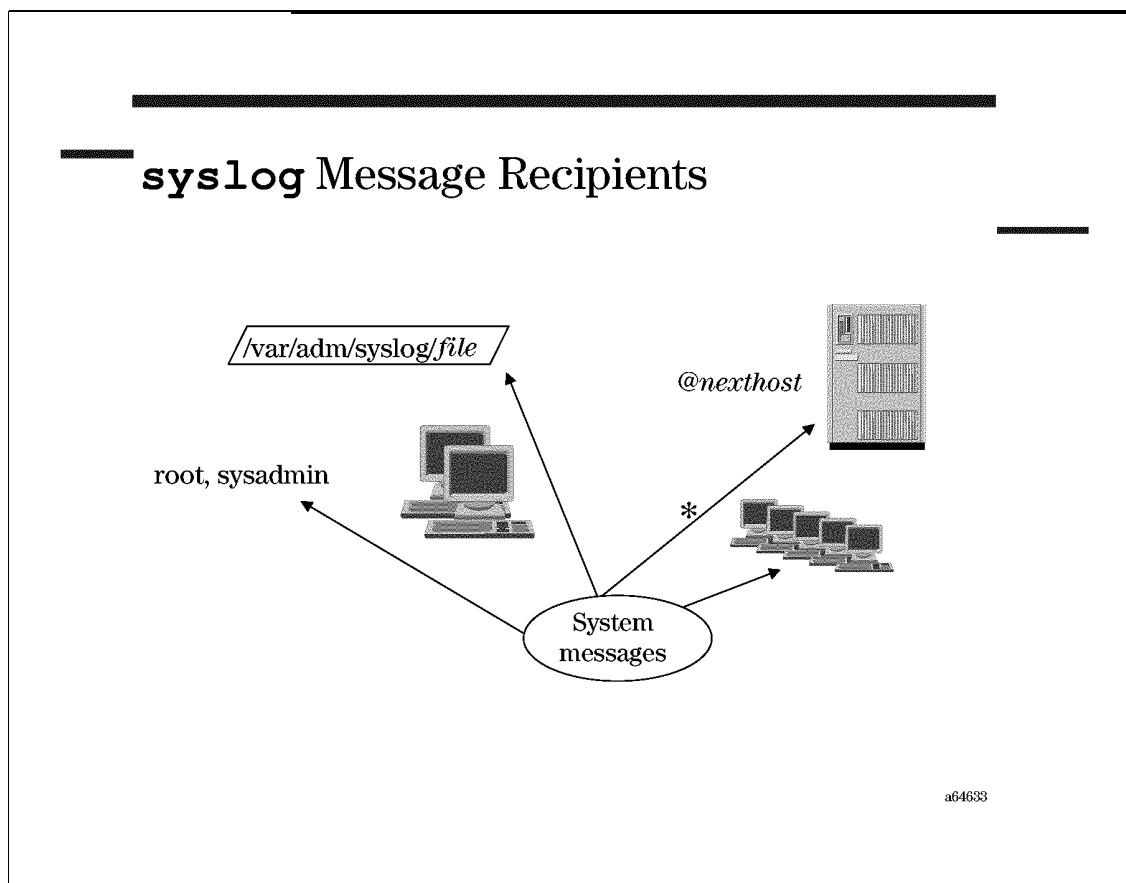
Student Notes

emerg	Panic or emergency condition—normally broadcasted to all users.
alert	Condition requiring immediate correction.
crit	Critical condition, such as hard device errors.
err	Common errors.
warning	Warning messages.
notice	Conditions that are not error conditions, but may need special handling or care.
info	Informational messages.
debug	Messages that contain only information when debugging a program.

none Exclusion of certain facilities in an entry specified by a wildcard.

Syslog levels are arranged hierarchically. The debug level includes all messages; the information level includes all messages except the debug level.

3-15. SLIDE: syslog Message Recipients



Student Notes

A **syslog** message can be written to any of the following:

- A file beginning with a leading slash—The file is opened in **append** mode. If the file does not exist, it is created.
- A host preceded by an @ character—Selected messages are forwarded to the **syslogd** on the named hosts.
- A user terminal—Several users can be informed by a comma-separated list of users. Selected messages are written to the user terminal if the user is logged in.
- An asterisk—Selected messages are written to the terminals of all logged-in users.

3-16. SLIDE: /etc/syslog.conf**/etc/syslog.conf****Syntax**

```
facility1.level1[;facility2.level2 ...]      recipient
```

Example

```
mail.debug      /var/adm/syslog/mail.log
*.info;mail.none /var/adm/syslog/syslog.log
*.alert        /dev/console
*.alert        root,sysadmin
*.alert        @monitor_host
lpr.emerg       *
```

a64677

Student Notes

The `/etc/syslog.conf` configuration file allows the system administrator to specify where the various systems' processes messages will get sent.

In the above example, on the slide, all mail system messages are written to `/var/adm/syslog/mail.log`. All messages at the info level and above, except for mail messages, are written to the file `/var/adm/syslog/syslog.log`.

All messages at the alert level and above are logged to the console. They are also logged to the users `root` and `sysadmin` if they are logged in as well as forwarded to the `syslogd` of `monitor_host`. Messages of the `lp` spooler system are sent to all logged-in users' terminals.

When specifying that a facility sends its messages to a log file you must use the character between the facility list and the log file, not a blank, i.e. `mail.debug /var/adm/syslog/mail.log`

3-17. LAB: System Logging Daemon—**syslogd**

Directions

Some subsystems send their protocol and error messages to the **syslogd** daemon. The network daemons and the mail system use these subsystems. These message often contain important hints as to what errors may be occurring on the system.

1. Which option is used with the **ftp** server to enable it to log messages to the **syslogd** daemon?

In which configuration file do you have to configure the **ftpd** server?

Enable the **ftp** server to log to **syslogd**.

2. Configure the system logging daemon, **syslogd**, for the following subsystems:

- a. Messages generated by network daemons with a level of INFO and greater should be sent to the file **/var/adm/syslog/daemon.log**, while those with a level of WARNING or greater should be sent to the console.
- b. Messages generated by the mail subsystem with a level of DEBUG and greater should be sent to the file **/var/adm/syslog/mail.log**.
- c. Choose a partner. Configure your system to send all messages generated by the **auth** (authentication) subsystem to the **syslogd** on your partner's system.

Do not delete any existing entries.

3. Start an **dtterm** with the **-C** option. The **-C** option causes all console messages to be sent to the **dtterm**.

- a. Determine the process ID (PID) of the **syslogd** daemon.
- b. Which signal do you have to use so that **syslogd** will read the modified configuration file **/etc/syslog.conf** again?
- c. Send the appropriate signal to the **syslogd** daemon to have it reread its configuration file.

4. Connect to your own system using **ftp**. Log in as the user **cracker** with the password **BaDdOoD**. The user **cracker** should not have a valid account on the system.

5. Use the following command to send the bogus mail:

```
# echo "breaking in" | mailx cracker@localhost
```

Notice that nothing was added to **/var/adm/syslog/daemon.log**. Instead this information was appended to **/var/adm/syslog/mail.log**.

6. In its simplest form, the following logger command could be used to enter information into the **syslog** subsystem.

```
# logger "Attention: Cracker suspected"
```

The default subsystem and priority would be **user.notice** and, based on the default configuration of the **syslogd** daemon, a message similar to the following would appear in **/var/adm/syslog/syslog.log**:

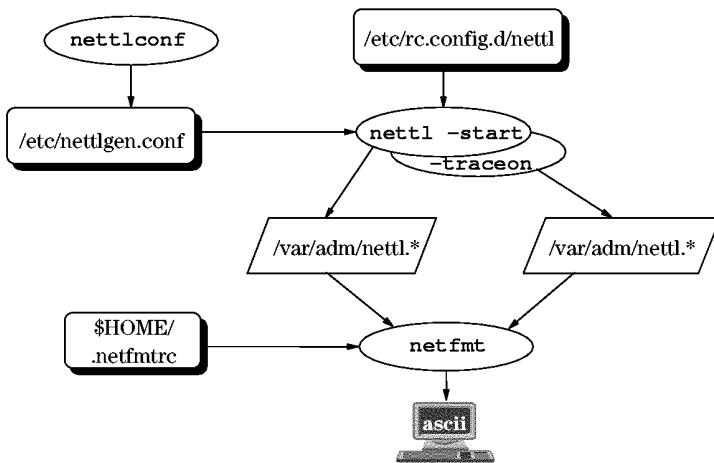
```
Dec 23 18:24:41 localhost root: Attention: Cracker suspected
```

7. The following logger command should forward a message to your partner's **syslog** daemon:

```
# logger -p auth.crit "Cracker Attack"
```

3-18. SLIDE: The Network Tracing and Logging Subsystem

The Network Tracing and Logging Subsystem



a64678

`nettl` is a tool used to control the network tracing and logging facility. It is used for the following actions:

- to log network activities, such as state changes, errors, and established connections
- to trace all inbound and outbound packets going through the network

The system startup script `/sbin/rc2.d/S300nettl` activates `nettl`, providing that the `NETTL` parameter is set to 1 in the `/etc/rc.config.d/nettl` configuration file. With the exception of the `-status` option, only users with superuser capability can use this command.

The tracing and logging facility is initialized with the `-start` option; `nettl` is started for logging only, which means that only default `ERROR` and `DISASTER` events are logged. Log messages are written to `/var/adm/nettl.log`. Disaster messages are also written to the console.

Tracing is activated with the option **-traceon**. Before tracing can be activated, **nett1** must be already active with the **-start** option. Trace records are written to **/var/adm/tracename.TRCn**, where *n* is a number, starting with 0. If the file exists, it will be appended. To start a fresh trace file, turn off tracing, then turn it on again, using a different tracename file.

Both trace and log files are binaries that can be viewed with the **netfmt** format command. Both **nett1** and **netfmt** allow filters to reduce output.

/usr/sbin/nett1conf maintains the database file **/etc/nett1gen.conf**, which is a database used to store system logging information along with a description of each subsystem that uses the network tracing and logging facilities. System administrators can use the **nett1conf** command to customize the system logging parameters stored in the database, such as console logging behavior, the system log file name, the maximum system log file size, and the amount of memory required by the tracing and logging facilities.

nett1conf is also called during system startup to change the database to reflect the values of any relevant environment variables in the **/etc/rc.config.d/nett1** file.

3-19. SLIDE: The Network Tracing and Logging Daemon

The Network Tracing and Logging Daemon

- Start network logging.
`nettl -start`
- Start network tracing.
`nettl -traceon hdrin -entity all -f mytracefile`
- Stop network tracing.
`nettl -traceoff -entity all`
- Determine `nettl` status.
`nettl -status`

a64679

`nettl` recognizes the following options, some of which can be abbreviated as described below:

-start (-st) Initialize the tracing and logging facility, and start default logging as determined by the `/etc/nettlgen.conf` file. A `nettl -start` command is performed during system startup if the NETTL variable in the `/etc/rc.config.d/nettl` file has a value of 1.

-stop (-sp) Terminate the tracing and logging facility.

-entity *subsystem* Limit the action of `-log`, `-traceoff`, or `-traceon` to the specified protocol layers or software modules specified by *subsystem*. Examples of LAN subsystems:

```
ns_ls_driver
  ns_ls_nfs
  ns_ls_icmp
  ns_ls_nft
  ns_ls_igmp
  ns_ls_ni
  ns_ls_ip
  ns_ls_tcp
```

```

ns_ls_loopback
  ns_ls_udp
  ns_ls_netisr
  ns_ls_x25

```

-log class Control the class of log messages that are enabled for the subsystems specified by the **-entity** option. *class* specifies the logging class. Classes can be specified as keywords or as a single numeric mask depicting which classes to log. Available classes are:

informative	i 1
warning	w 2
error	e 4
disaster	d 8

informative

describes routine operations and current system values

warning

indicates abnormal events possibly caused by subsystem problems

error

signals an event or condition which was *not* affecting the overall subsystem or network operation, but may have caused an application program to fail

disaster

signals an event or condition that *did* affect the overall subsystem or network operation, caused several programs to fail, or the entire node to shut down

-status (-ss) Report the tracing and logging facility status. Additional options include

log	log status information
trace	trace status information
all	trace and log status information

-tracemax maxsize (-tm) Tracing uses a circular file method. When one file fills up, a second is used. Two trace files can exist on a system at any given time. *maxsize* specifies the maximum size in kilobytes of both trace files combined.

-traceon kind(-tn) Start tracing on the specified subsystem. The tracing and logging facility must have been initialized by **nettl -start** for this command to have any effect. *kind* determines which events should be traced.

hdrin (0x80000000)	Inbound Protocol Header
hdrout (0x40000000)	Outbound Protocol Header
pduin (0x20000000)	Inbound Protocol Data Unit (including header and data)

pduout (0x10000000)	Outbound Protocol Data Unit (including header and data)
proc (0x08000000)	procedure entry and exit
state (0x04000000)	protocol or connection states
error (0x02000000)	invalid events or condition
logging (0x01000000)	special kind of trace that contains a log message
loopback (0x00800000)	packets whose source and destination system is the same

For more information, refer to the manpage **nett1(1M)**.

3-20. SLIDE: The Network Tracing and Logging Formatter

The Network Tracing and Logging Formatter

- Format network logging.
`netfmt -f /var/adm/nettl.LOG00 -v`
- Format network tracing.
`netfmt -f mytracefile.TRC00 -v -n -N > asciifile
vi asciifile`

a64680

`netfmt` is used to format binary trace and log data gathered from the network tracing and logging facility. Formatting options are specified in an optional filter configuration file, `$HOME/.netfmtrc`, or with the file specified with the `-c` option. This file allows you to specify the options that control input data formatting. It determines which filters are used to decide what input data is to be discarded and what is to be formatted.

Some important options used with `netfmt` include:

- `-f filename` Specifies the input file containing the binary log or trace data.
- `-c filterfile` Specifies the file containing formatter filter configuration commands.
- `-p` Performs syntax check only on the `config_file` specified by the `-c` option.
- `-F` Follow the input file. Instead of closing the input file when end of file is encountered, `netfmt` keeps it open and continues to read from it as new data arrives.
- `-v` Verbose output.
- `-n` Shows port numbers and network addresses as numbers rather than symbols.

- N Enables formatting of the protocol headers like Ethernet/IEEE802.3, SLIP, IP, ICMP, IGMP, TCP, UDP, and RPC. Remaining user data is formatted in hexadecimal and ASCII.

Options -v, -n, and -N can also be specified in the filter configuration file.

For more information see manpage **netfmt(1M)**.

3-21. SLIDE: Filter Configuration File

Filter Configuration File

`$HOME/.netfmtrc` is the default filter

- global filters

```
formatter verbosity nice
formatter filter kind hdrin
formatter filter time_from 09:00:00 04/22/97
```

- subsystem filters

```
filter dest 08-00-09-3a-4b-3c
filter tcp_sport 23
filter ip_daddr 15.138.139.25
```

a64634

The default filter configuration file is `$HOME/.netfmtrc`. A different configuration file can be specified by using the `-c` option and specifying the different file name. The configuration file contains two kinds of filters:

- Global filters control all subsystems. Global filters all begin with the keyword formatter.
- Subsystem filters pertain only to specific subsystems.

Input data which does not match a filter is discarded. Examples of global filters include:

`formatter verbosity nice`

Nice formatting. Same as the `-N` option.

`formatter verbosity normal`

Default formatting.

`formatter verbosity terse`

Format each packet in 1 line. Same as the `-1` option.

`formatter filter Process_ID pid`

Only packets sent to or received from the specified `pid`.

`formatter filter kind (hdrin, hdrouout,...)`

Only events of specified `kind` are formatted. `kind` is the same value that you can specify at `nett1`. For

formatter filter time_from 09:00:00 04/22/97	reducing data, it is preferable to filter packets with nettl .
formatter filter time_through 17:00:00 04/22/97	Filter only those packets sent during the specified time.

Subsystem filters are provided to filter data for different subsystems, such as LAN (TCP/IP), X25, or OTS (ISO/OSI). LAN filtering occurs at each network layer. If a packet matches a filter within a layer, it is passed up to the next layer. The packet must pass every layer to pass through the entire filter. Filtering starts with layer 1 and ends with layer 5. If no filter is specified for a particular layer, that layer is *open* and all packets pass through. Filters at each layer are logically ORed. Filters between layers are logically ANDed.

Examples of LAN subsystem filters are:

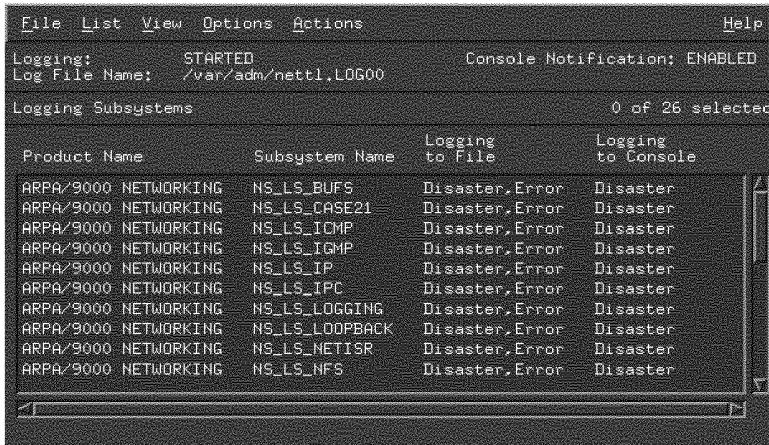
- filter connection local_ip_addr:port remote_ip_addr:port
- filter dest dest.hardware_address
- filter source source.hardware_address
- filter ip_daddr dest.ip_address
- filter ip_saddr source.ip_address
- filter ip_proto protocol_number
- filter tcp_dport destination_port
- filter tcp_sport source_port
- filter udp_dport destination_port
- filter udp_sport source_port

More values can be listed separately with a comma (.). An exclamation mark (!) excludes value, and an asterisk (*) matches any value.

For more on filters, see manpage **netfmt(1M)**.

3-22. SLIDE: Using nettladm

Using nettladm



The screenshot shows the nettladm application window. At the top, it displays "Logging: STARTED" and "Log File Name: /var/adm/nettl.LOG00". The title bar includes "File List View Options Actions Help". Below this, a status message says "Console Notification: ENABLED". The main area is titled "Logging Subsystems" and shows a table with 0 of 26 selected rows. The columns are "Product Name", "Subsystem Name", "Logging to File", and "Logging to Console". All entries show "Disaster, Error" for both logging levels.

Product Name	Subsystem Name	Logging to File	Logging to Console
ARPA/9000 NETWORKING	NS_LS_BUFS	Disaster, Error	Disaster
ARPA/9000 NETWORKING	NS_LS_CASE21	Disaster, Error	Disaster
ARPA/9000 NETWORKING	NS_LS_ICMP	Disaster, Error	Disaster
ARPA/9000 NETWORKING	NS_LS_IGMP	Disaster, Error	Disaster
ARPA/9000 NETWORKING	NS_LS_IP	Disaster, Error	Disaster
ARPA/9000 NETWORKING	NS_LS_IPC	Disaster, Error	Disaster
ARPA/9000 NETWORKING	NS_LS_LOGGING	Disaster, Error	Disaster
ARPA/9000 NETWORKING	NS_LS_LOOPBACK	Disaster, Error	Disaster
ARPA/9000 NETWORKING	NS_LS_NETISR	Disaster, Error	Disaster
ARPA/9000 NETWORKING	NS_LS_NFS	Disaster, Error	Disaster

a64635

Student Notes

The network tracing and logging administration manager, **nettladm**, command is a tool used to administer network tracing and logging. It provides an interactive user interface to the **nettl**, **netfmt**, and **nettlconf** commands.

The **nettladm** command starts a menu-driven program that makes it easy to perform network tracing and logging tasks with only limited specialized knowledge of HP-UX.

nettladm recognizes the following options:

- l Shortcut to enter the *Logging Subsystems* (logging) area. This is the default.
- t Shortcut to enter the *Tracing Subsystems* (tracing) area.

- c filter_file** Use the contents of the filter file as the default set of subsystem formatting criteria when creating reports within the *Create Report* area. The defaults can be overridden through the interface screens.

For more information, see manpage **nettladm**.

Warnings

- Changes to logging and tracing levels and states are not preserved across system reboots or stops and restarts from outside the **nettladm** command. Permanent changes must be made to the **/etc/nettlgen.conf** file using the **nettlconf** command. Changes to console logging and all logging startup parameters are preserved.
- the **nettladm** command allows the specification of all log classes and all trace kinds for all subsystems. However, many subsystems do not support all log classes and all trace kinds. No error or warning will be issued if a subsystem does not support a log class or trace kind.
- The **nettladm** command reads the **/etc/nettlgen.conf** and **/var/adm/conslog.opts** files each time it is run. If the files become corrupted, this command will not work.

Files

/etc/nettlgen.conf Tracing and logging subsystem configuration file.

/var/adm/conslog.opts Default console logging options filter file as specified in **/etc/nettlgen.conf**.

3-23. LAB: Using nettl, netlfmt, and nettladm

Directions

1. Determine if the trace and logging subsystem is activated on your system.

What is the name of the log file?

What is its maximum size?

Read the entries in the log file.

2. Use **nettl** to capture packets transiting the network interface. For example, trace an ICMP Echo Request (**ping**) packet to a remote destination and its reply.

To avoid capturing too many packets, use **nettl** only on packets transiting the network driver. Do not capture state, exit code, and so forth.

Finally, save all the collected data put in the file **/tmp/trace**.

3. Format the gathered trace data. If too much data exists in the output and makes it difficult to read, set up a filter so that only packets to or from your destination will be shown.

To limit the information displayed by the **netlfmt** command, you will need to create a configuration file that defines which traces are of interest. In your home directory, create a file called **.netlfmtrc** whose contents are:

```
filter ip_saddr  
filter ip_daddr
```

Note that there are two variables defined here. The first, **ip_saddr**, says you are interested in packets that have the associated IP address as an IP packet source address. The second, **ip_daddr**, similarly defines an interest in packets that have the associated IP address as an IP packet destination address.

To view the stored trace information, use the command **netfmt**.

```
# netfmt -N -f /tmp/trace.TRC0
```

4. Fill in the following table with your results.

	ECHO Header	ECHOREPLY Header
Tracing Subsystem		
Ethernet Protocol		
HW Source Address		
HW Destination Address		
Source Service Access Point		
Destination Service Access Point		
Network Protocol Number		
IP Source Address		
IP Destination Address		
Incoming/Outgoing Packet		
Application Port/ Protocol		
Application Information		

5. You can also view the tracefile from within **nettladm**.
 1. Select *List -> Tracing Subsystems* from the menu bar to work with the tracing subsystem.
 2. Then select *File -> Create Report*
 3. From the Create Report dialog, select *Select Trace File* if the correct trace file is not listed.
This will allow you to *browse* the file systems to select the correct file.
 4. Select *Modify Criteria* which will take you to the Modify Report Criteria for Tracing dialog. Here you can further define which components you wish to view. Under *Subsystems* exclude all subsystems except *LAN/9000 NETWORKING NS_LS_DRIVER* and then press **[OK]**.
 5. Under *Trace Kind*, include only incoming and outgoing protocol data unit and then press **[OK]**.
 6. Finally, press **[OK]** in the Modify Report Criteria for Tracing dialog.
 7. Select *Print Report* and choose a detailed report format with any desired report options.
 8. Choose the screen as the destination. Then press **[OK]**.

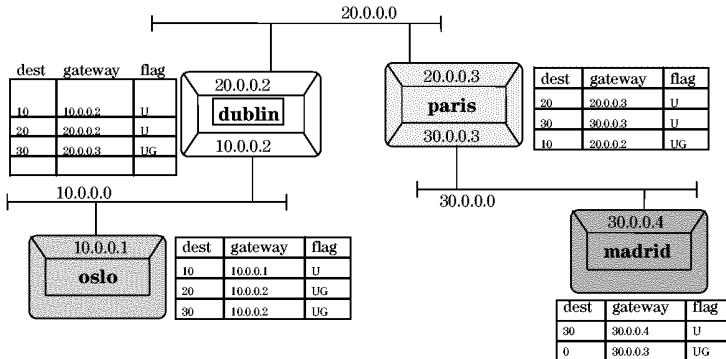
Module 4 — Routing Tables and Dynamic Routing Protocols

Objectives

- Upon completion of this module you will be able to do the following:
 - Apply at least two different methods to the problem of maintaining routing tables.
 - Determine the table maintenance method most appropriate to a given network environment.

4-1. SLIDE: Routing Tables

Routing Tables



a63057

Student Notes

A router is a device that has multiple network interfaces. These interfaces transfer Internet protocol (IP) packets from one network or subnet within an inter-network to another . When two systems communicate, data packets are sent from source to destination. If source and destination do not share the same network, packets must traverse one or more routers.

Within the transmission control protocol/internet protocol (TCP/IP) protocol family, the Internet protocol (IP) is responsible for routing packets from network to network via routers.

The source system and each router on the path to the destination need to know the next hop to choose the correct way. IP routing chooses the path over which a datagram is sent by consulting the routing tables to find the correct next hop.

Each entry in a routing table specifies only the next hop for each destination. A system does not know the complete path to a destination. This means that an entry in a routing table points to a router that can be reached across a single network.

An entry in a routing table contains a pair of addresses:

- a destination address
- the IP address of the next router to receive packets

A destination can be

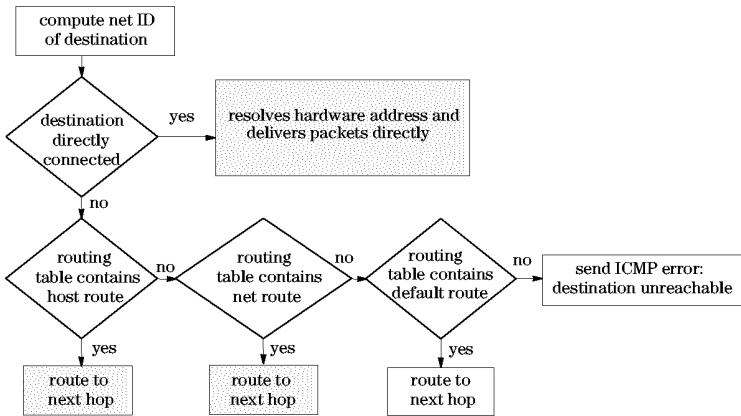
- a host IP address (host routing)
- a network address (net routing)
- a default address 0

It is common for hosts and most routers to specify a default router address for all destinations with unknown routes.

The IP protocol consults only the routing table. IP doesn't create a routing entry itself. Correct routing depends on correct routing tables on ALL involved systems, from source to destination and back.

4-2. SLIDE: IP Routing Algorithm

IP Routing Alogorithm



a63058

Student Notes

Entries in a routing table are consulted in a fixed order.

1. For a destination address, IP protocol extracts the network address.
2. If the network address of the destination is the same as one of the network addresses of the router's interfaces, the network is **directly** connected. On a directly connected network, a router delivers packets directly by determining the hardware address of the destination. Normally, a hardware address request is made if, and only if, the destination network address corresponds to the interface network address. This is very important for understanding proxy address resolution protocol (ARP).
3. Next priority is the host specific route, if it matches the destination address.
4. Otherwise, a network specific route is searched.
5. If no entry matches the destination, the IP packet is sent to the default router.

6. If none of the above criteria are met, an Internet control message protocol (ICMP) network unreachable error is declared, if no default entry exists.

4-3. SLIDE: IP Routing

IP Routing

What does IP do?

- IP routes IP packets according to routing table

What does IP not do?

- IP does not create routing table entries

What methods are used to maintain routing tables?

- static routing
 - manual with command `route add`
 - *tricky* routing with Proxy ARP
- default router entries with router discovery protocol
- dynamic routing with dynamic router protocols
 - RIP, HELLO, OSPF
 - EGP, BGP

Warning: Don't mix different methods on same host!

a63059

Student Notes

The IP protocol routes a packet only if an appropriate entry for the destination can be found in the routing table.

The IP protocol neither creates nor maintains the routing table.

An ICMP redirect can be considered an exception. In reality, it doesn't create a new entry, but rather improves network performance.

A variety of methods exist for maintaining routing tables. On a specific host, only one method can be applied. In any given network, different methods are usually used together.

- static routes created with the `route` command
- proxy ARP, a special case of routing
- dynamic routing

Dynamic routers support protocols that dynamically search for networks. Once a network is found, it is added to the routing table dynamically. Dynamic routers check connectivity periodically, and in case of a loss, delete the routing entry.

Dynamic routing on a UNIX[®] system is implemented with the `gated` or `routed` command.

CAUTION: Never use `gated` with other maintenance commands.

4-4. SLIDE: Static Routing

Static Routing

```
route add destination gateway hop
route add host 30.0.0.1 10.0.0.2 1
route add net 20.0.0.0 10.0.0.2 1
route add default 10.0.0.1 1
```

When is static routing used?

- hosts with only one interface
- simple networks
- nodes without dynamic router protocols
(example: X-terminals, network printers, and PCs)
- segments with only one router (default router)

a63060

Student Notes

Static routing is a very important method for maintaining routing tables. It is used with the following:

- end systems with only one interface
- simple networks
- special devices, such as X-terminals, network printers, personal computers, etc.
- segments with only one router (default router)

Static routing has the following advantages:

- It is easy to configure.
- There is no protocol overhead.

Disadvantages of static routing include the following:

- It requires manual configuration.
- Changes in network topology must be changed on all hosts.

- There is no flexibility for
 - path optimization
 - router failures

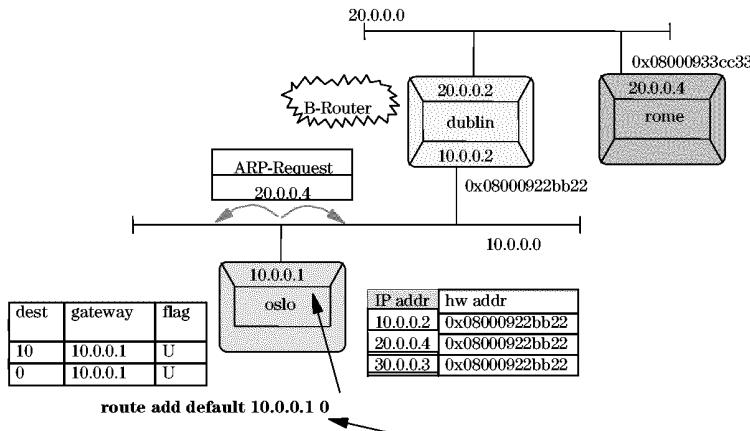
NOTE:

To avoid endless loops, ping-pong packets, and undeliverable packets, each router decrements the time-to-live parameter (hopcount) in the IP protocol header. If the hopcount is equal to zero the router removes the datagram from the net.

Older route commands specify the initial value of the hopcount with the hop argument. It is difficult to estimate the appropriate hopcount. Therefore, newer implementations set the initial hopcount to *maximum*, if the hop argument of the route command is greater than zero.

4-5. SLIDE: Proxy ARP

Proxy ARP



a63061

Student Notes

Proxy ARP involves two components:

- A client sends an ARP request for a network ID to a network that is not directly connected to the client.
- A proxy ARP server responds to such an ARP request with its own hardware address if it knows a path to the requested destination.

Proxy ARP allows the dynamic connections of remote systems without the need for the update of the routing tables on other systems, but instead the one associated as the proxy ARP server.

Normally, IP addresses are translated to hardware addresses, using ARP only for hosts directly connected to the same network. This means the source and destination have the same network ID.

A proxy ARP client can request a hardware address for any destination.

To set up a proxy ARP client for a given destination, specify at the **route** command:

- the IP address of your system's own interface
- hop count 0

In most cases the destination will be default.

On the other end of the network, a proxy address must be configured to a medium access control (MAC) address. It will send its own hardware address to the requester. Frames to this IP address will be sent to the server. The server may then forward them to the remote system.

On a proxy ARP client, the ARP cache contains the same hardware address, (the hardware address of the proxy ARP server), for several IP addresses outside its own network.

The server is a proxy for the remote IP address. It tells the network that it can accept frames for the remote IP address and deliver them by responding to the ARP requests.

The main advantage of proxy ARP is that all systems on the network do their own routing. They know that in order to send an IP frame to the remote's IP address, they must put it on the same wire that is connected to the proxy ARP server's network adapter. A network can be dynamically changed; routers can be dynamically exchanged without reconfiguring the clients. In addition, more than one *default router* can be added to the network.

4-6. SLIDE: Setting Up a Proxy ARP Client

Setting Up a Proxy ARP Client

```
ROUTE_DESTINATION[index]=value  
ROUTE_MASK[index]=value  
ROUTE_GATEWAY[index]=value  
ROUTE_COUNT[index]=value  
ROUTE_ARGS[index]=value
```

a64698

Student Notes

Configuration Overview

When **gated** starts, it reads a configuration file to find out how each protocol should be used to manage routing. By default, it uses the configuration file called **/etc/gated.conf**.

Creating the configuration file is usually the responsibility of the system administrator. The configuration file may include up to eight sections (called classes) of configuration statements. Statements can be further defined with optional clauses.

The eight classes of configuration statements are

- **Directives** are statements that are immediately acted upon by the **gated** parser.
- **Trace** statement controls **gated** tracing options.
- **Options** statements define global **gated** options.
- **Interface** statements define router interface options.
- **Definition** statements identify the autonomous system that the router belongs to, the router ID, and *martian* addresses (any addresses for which routing information should be ignored).

- **Protocol** statements enable or disable gated protocols and set protocol options.
- **Static** statements define static routes or default routers that are installed in the kernel routing table.
- **Control** statements define routes that are imported to the router from other routing protocols and routes that the router exports to other routing protocols.

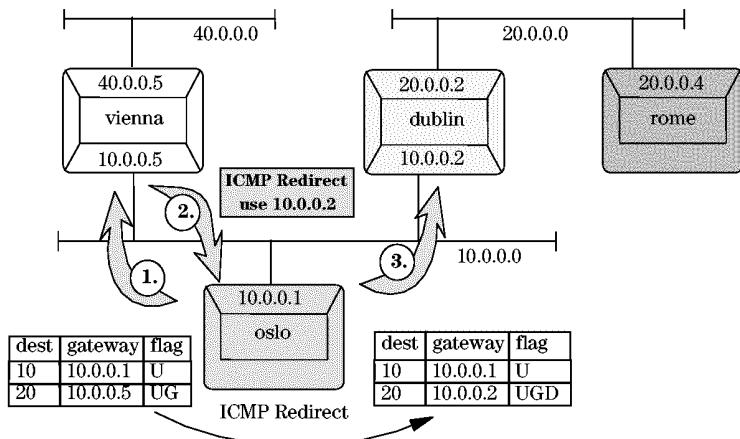
Type `man 4 gated.conf` at the HP-UX prompt for a description of each configuration class and to determine which statements belong to which class. With version 3.5 of `gated`, the two statements previously in the Trace class (`tracefile` and `traceoptions`) have been combined into one `traceoptions` statement. So, the `tracefile` statement has been eliminated. Also, some of the global options have been removed, some new global options have been added, and options have been added for some of the protocols. For details about the new syntax, type `man 4 gated.conf` at the HP-UX prompt.

Following are some routing variables that must be set in `/etc/gated.conf`. The index indicates the network interface to which the routing variables are assigned.

- `ROUTE_DESTINATION[index]=value`
- `ROUTE_MAST[index]=value`
- `ROUTE_GATEWAY[index]=value`
- `ROUTE_COUNT[index]=value`
- `ROUTE_ARGS[index]=value`

4-7. SLIDE: ICMP Redirect

ICMP Redirect



a63062

Student Notes

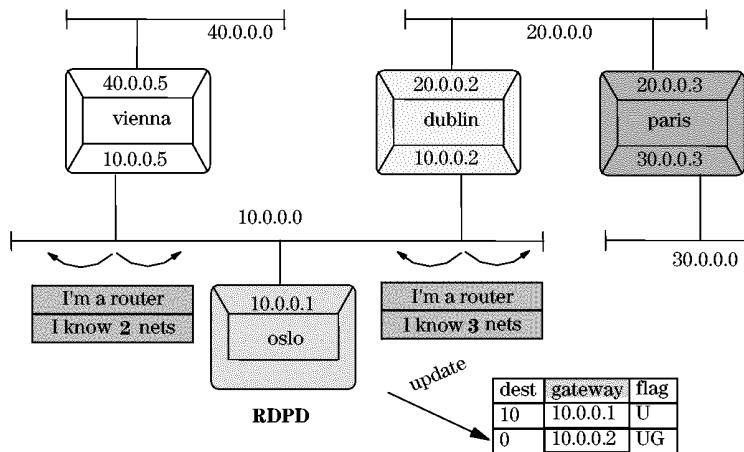
IP never adds an entry to the routing table. However, in one instance, the IP network layer does change an entry.

When a router (*vienna*) detects a host (*oslo*) that uses a nonoptimal route, it sends an Internet control message protocol (ICMP) redirect message to the host, requesting that the host change its route to another router (*dublin*).

The advantage of **ICMP redirect** is that it allows a host to know only the address of one router on the local network. ICMP redirects actually do not maintain the routing table; rather, they increase network performance.

4-8. SLIDE: Router Discovery Protocol

Router Discovery Protocol



a63063

Student Notes

Router discovery protocol (RDP) is useful for finding default routers. A router that supports RDP (Request for Comment (RFC) 1256) sends an ICMP router advertisement. An RDP end system listens to these advertisements and automatically discovers all directly connected routers.

If there are multiple candidates for default routers, the router with the highest preference value is selected. The remaining candidates act as backups. The RDP end systems select a new router if one fails. RDP is not a true routing protocol, because the ICMP router advertisements do not advertise the networks that are reachable through routers.

Simultaneously adding default routes to RDP via the `route` command can cause unpredictable results and should be avoided.

On HP-UX, RDP is implemented with the `rdpd` daemon, which is started during boot time initialization.

Set the `RDPD` variable in the `/etc/rc.config.d/netconf` configuration file.

NOTE: UNIX systems do *not* send ICMP router advertisements.

4-9. SLIDE: Overview of Dynamic Routing

Overview of Dynamic Routing

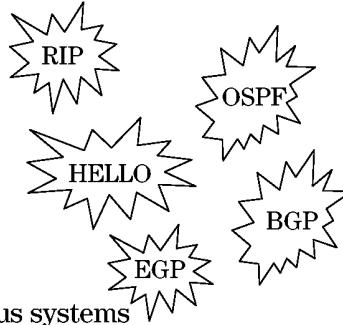
Dynamic routing protocols are designed for different purposes:

Internal gateway protocols

- find paths dynamically
- test reachability
- detect network failures
- optimize routing path
- reconfigure routes automatically

External gateway protocols

- maintain links between autonomous systems



a646132

Student Notes

Dynamic routing eliminates the need to reset routes manually. When network failures occur, routes are automatically rerouted. Routing protocols are designed to find a path between network nodes. If multiple paths exist for a given destination, the shorter paths are usually chosen. Each protocol has a cost or a metric that it applies to each path. In most cases, the lower the cost or metric for a given path, the more likely a protocol will choose it.

There are several dynamic routing protocols for different purposes.

Interior gateway protocols find paths within a domain like a company. In the Internet, such a domain is called an **autonomous system**. Routers inside an autonomous system know only their networks. Networks outside an autonomous system are reached via default routers and core routers.

- Interior gateway protocols find paths within a domain.
 - Interior gateway protocols are

RIP routing information protocol

HELLO

open shortest path first

- Exterior gateway protocols maintain links between the autonomous systems.

- Exterior gateway protocols are

EGP exterior gateway protocol

BGP border gateway protocol

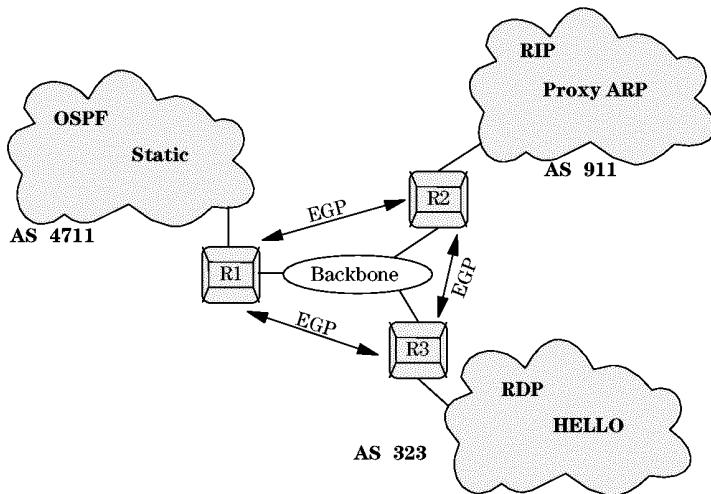
- The two main methods for finding routes dynamically are:

- link state routing protocol

- vector distance routing protocol

4-10. SLIDE: Autonomous Systems

Autonomous Systems



a63065

Student Notes

For routing purposes, networks and gateways are logically grouped into autonomous systems. An autonomous system (AS) is a set of networks and gateways that are administered by a single entity. Companies and organizations that wish to connect to the Internet and form an AS must obtain a unique AS number from the InterNIC assigned numbers authority.

An **interior** gateway protocol is used to distribute routing information within the autonomous system. Alternately, you can configure IP routes manually with the `route` command (static routing, proxy ARP) or with RDP. For end systems in subnets with only one router (gateway) to the rest of the network, configuring a default route is usually more efficient than running the dynamic routing daemon.

An **exterior** gateway protocol is used to distribute general routing information about an autonomous system to other autonomous systems.

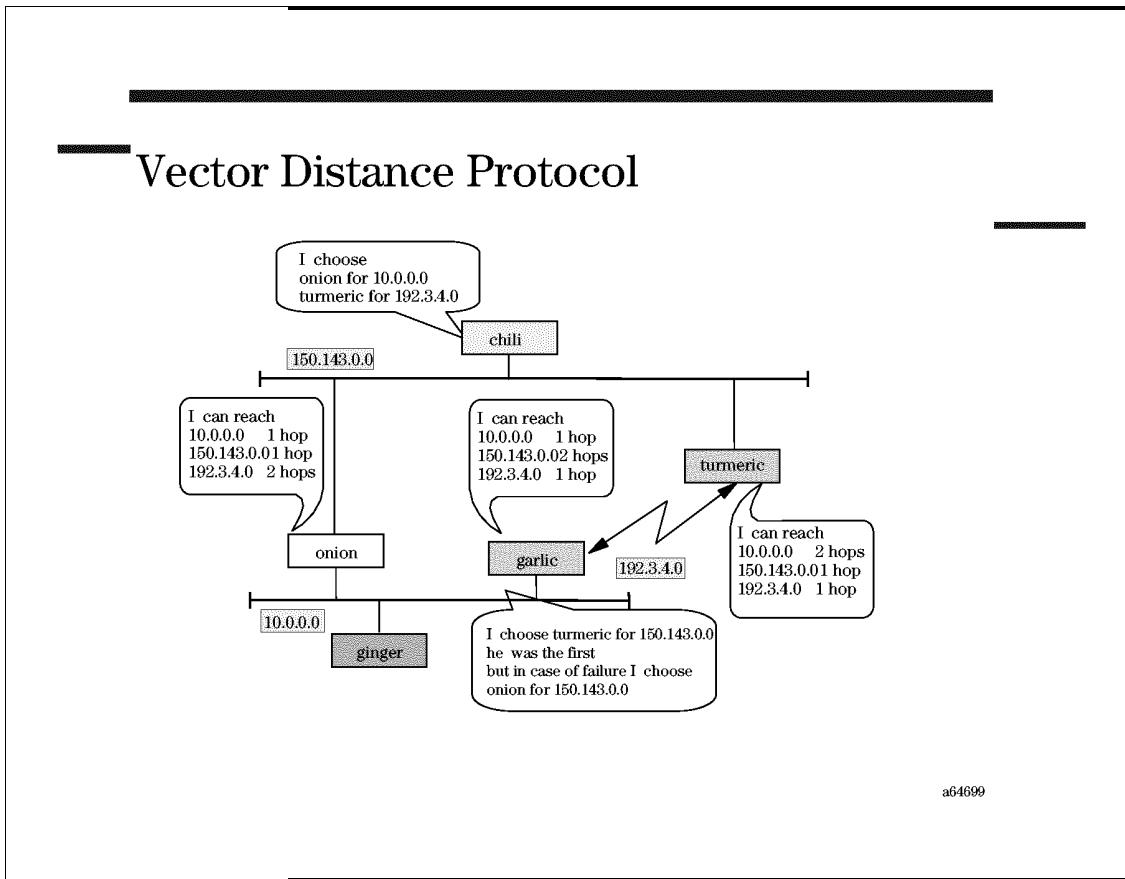
Dividing networks into autonomous systems keeps route changes inside the autonomous system from affecting other autonomous systems. When routes change within an autonomous

system, the new information need not be propagated outside the autonomous system if it is irrelevant to gateways outside.

In this course we will discuss

- RIP as a vector distance protocol
- EGP as exterior gateway protocol
- OSPF as a link state protocol

4-11. SLIDE: Vector Distance Protocol



Student Notes

The algorithm behind vector distance routing is very simple. Each router participating at a vector distance routing protocol periodically sends its routing table to all directly connected routers. The information contains

- the destination of a route
- the distance to a route

The distance is often measured in hops. As a distance parameter, RIP uses the hopcount to determine the shortest path to a destination.

HELLO uses the delay as a distance parameter. Don't use HELLO in new networks, as it is an older protocol.

A recipient of a routing table updates its own routing entries:

- when a new route is propagated
- when it receives a route to a known destination with a smaller distance

- when the distance with the same path has changed

When a router has learned of a destination with a distance of N , it propagates the distance with a distance of $N+1$ towards its neighbor's next period.

In large networks the message size increases with the number of networks. A router knows only the distance to a network, but does not know the topology of the network.

When routes change rapidly, information spreads very slowly, resulting in inconsistencies.

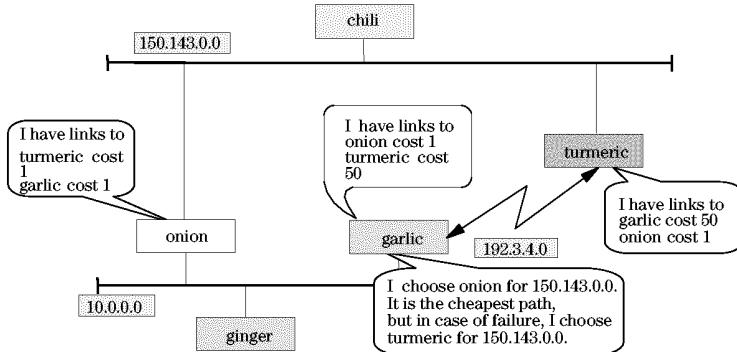
Examples of vector distance routing protocols are

- RIP
- HELLO
- GGP (no longer used)

4-12. SLIDE: Link State Routing Protocol

Link State Routing Protocol

My default gateway is onion.



a646100

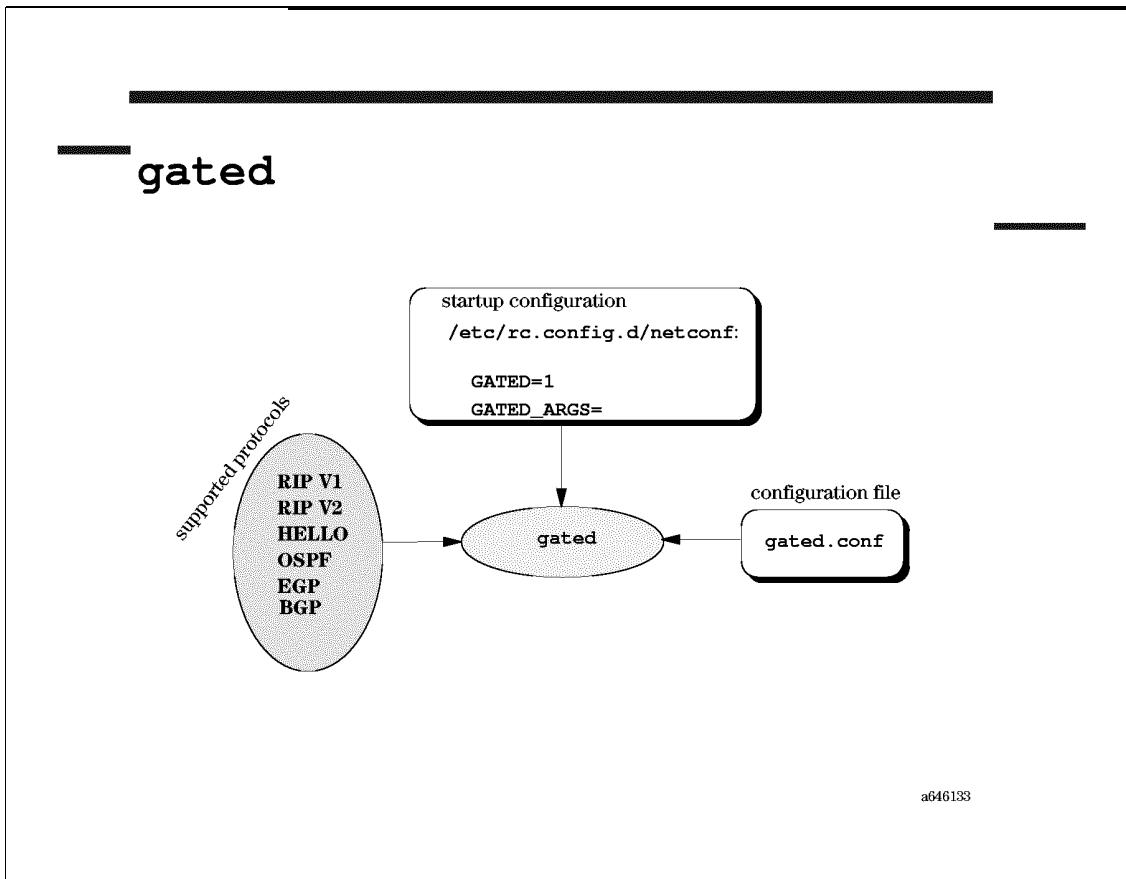
Student Notes

In a link state routing protocol, routers exchange link state advertisements (LSA). The primary information exchanged in one LSA message is

- the networks attached to a link
- the cost of the link

Each router on the network transmits a packet that describes its local links to all other routers. The distributed database is then built from the collected descriptions. If a link fails, updated information floods the network, allowing all routers to recalculate their routing tables at the same time. Using the database information, each router constructs its own complete topology database.

Instead of sending lists of destinations, a router actively tests the status of all directly connected neighbors and propagates the link status to *all* other routers.

4-13. SLIDE: gated**Student Notes**

`gated` supports many routing protocols that allow routers to build and maintain dynamic routing tables. However, `gated` also supports RIP, which can run on end systems (systems with only one network interface) as well as routers. The `gated` daemon can be configured to perform all or any combination of the supported protocols.

`gated` supports RIP, OSPF, HELLO, EGP, and BGP.

4-14. SLIDE: Routing Information Protocol

Routing Information Protocol

RIP

- is a vector distance protocol
- measures hopcount as distance
- broadcasts every 30 sec
- deletes a route after 180 sec of unreachability
- RIP VERSION 2
- supports subnetmasking
- supports multicasting
- supports authentication

a63069

Student Notes

Routing information protocol (RIP) is a common routing protocol used within an autonomous system. Currently two versions of RIP implementations exist:

- version 1, as defined in RFC 1058
- version 2, as defined in RFC 1388

RIP is a typical vector distance protocol. RIP uses the hopcount as distance parameter to determine the shortest path to a destination. Hopcount is the shortest number of routers a packet must pass through to reach its destination.

If a path is directly connected, it has the lowest hopcount. RIP will propagate the hopcount for directly connected routers. If the path passes through a single router, the hopcount increases to 2. Hopcount can increase to a maximum value of 16, which is RIP's *infinity metric*, an indication that a network or node cannot be reached. Hosts with only one local area network (LAN) interface can use the RIP protocol with *gated* to passively listen to routing information when there is more than one router on the LAN. If *gated* finds two or more network interfaces, the node both listens to and broadcasts RIP information.

RIP version 1 broadcasts the routing tables every 30 seconds. A route becomes invalid if a router fails more than 180 seconds. Subnetmasking is not supported in RIP version 1.

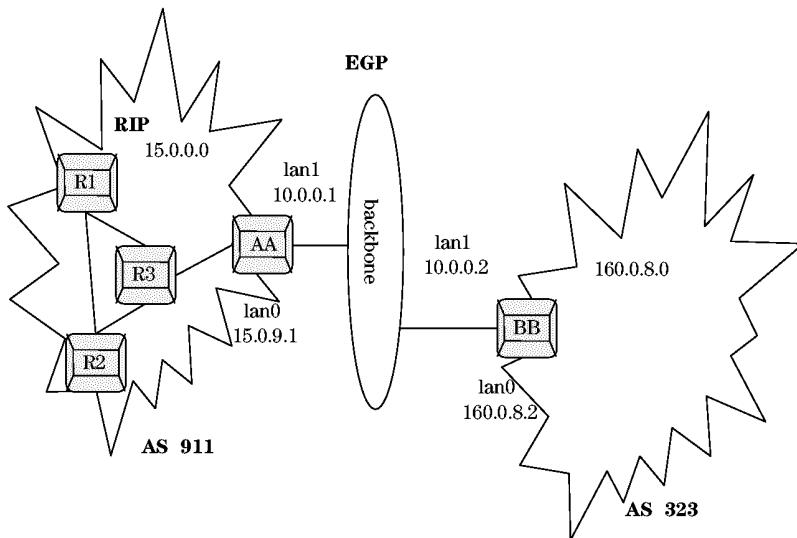
Main extensions in RIP version 2 are

- subnetmasking
- multicasting

Other extensions such as `route tag` or `authentication` are not supported by `gated 3.0.3`.

4-15. SLIDE: Example of RIP and EGP

Example of RIP and EGP



a646134

Student Notes

The following configuration files refer to the example above.

In autonomous system 911, RIP is used as interior gateway protocol. In autonomous system 323, another method is used for example static routing. For exchanging routing information via EGP, internal routing methods are irrelevant.

EGP is merely a reachability protocol. Routers use EGP messages to advertise the addresses that are directly reachable through them. Although EGP propagates distances, EGP does not interpret metrics such as cost or distance for route selection.

EGP can be used for exchanging routing information, but not for dynamically finding routes.

4-16. SLIDE: Common Configuration Clauses

Common Configuration Clauses

```
traceoptions "/var/adm/gated.trc" size 1m files 2 route;  
interfaces { lan0 lan1 passive ; } ;  
rip yes;  
static { default gateway 15.0.9.1 ; } ;
```

gated.conf

a646101

Student Notes

The `/etc/gated.conf` configuration file consists of a sequence of statements terminated with a semicolon (`:`). There are eight classes, or clauses, of statements. The most common classes are

- | | |
|---------------------------|--|
| <code>trace_file</code> | Trace information is appended to the trace file according to the trace options. To replace an old <code>trace_file</code> , specify <code>replace</code> . |
| <code>traceoptions</code> | Specify the desired level of tracing output. Useful <code>traceoptions</code> are <i>general</i> , <i>rip</i> , <i>size</i> , <i>files</i> , <i>route</i> , <i>replace</i> , and <i>protocol</i> . |
| <code>interfaces</code> | The <code>interfaces</code> clause binds several attributes to existing interfaces. As a rule, each interface should be defined with the attribute <i>passive</i> . This prevents the <code>gated</code> from deleting interfaces from the routing table, if <code>gated</code> receives no routing protocol information from that interface. More information on such interfaces is presented in a subsequent slide, "Point-to-Point Interfaces." |
| <code>static</code> | For some destinations, it may be necessary to announce routers that cannot be discovered by a dynamic routing protocol. As a rule, this is the case for |

default routers. The *retain* qualifier ensures that the entry is not deleted when **gated** exists. If the static statement is specified but not an export statement, then the route is not passed on as a route to other routers. The static clause must be placed behind all protocol clauses.

NOTE:

Do not propagate routes with the **route** command when running the **gated**. Doing so leads to unpredictable results.

4-17. SLIDE: RIP Definition

RIP Definition

```
traceoptions "/var/adm/gated.trc" size 1m files 2 route ;
interfaces { lan0 lan1 lan2 passive ; } ;

# use RIP version 1
rip yes ;
# use RIP version 2
rip yes { interface lan0 lan1 lan2 version 2 broadcast ;
traceoptions packets ; } ;
static { default gateway 15.0.9.1 ; } ;
```

gated.conf on R3

a646102

Student Notes

To configure RIP, the following statements should be included in `/etc/gated.conf` file.

rip yes Tells `gated` to enable the RIP protocol. Default is RIP version 1 for all interfaces.

interface lan0 ... Defines attributes depending on the interface such as:
noripin `gated` does not process any RIP information received through the specified interface
noripout specifies that `gated` does not send any RIP information

version 1 Specifies that RIP version 1 packets are sent. Version 2 packets are sent only in response to a version 2 poll packet.

version 2	Specifies that RIP version 2 packets are sent to the RIP multicast address or to the broadcast addresses. You can specify how the packets are sent with the multicast or broadcast clauses.
broadcast	Specifies that RIP packets are always generated. If the RIP is enabled and there is more than one interface, broadcast is assumed. For single end system (host), broadcast normally is never set. Specifying broadcast with only one interface is useful only when propagating static routes or routes learned from other protocols.

NOTE: The order in **gated.conf** clauses is very important. A wrong order can cause much confusion. For a complete **rip** clause, refer to manpage **gated.conf(4)**.

For some examples of properly-written **gated.conf** files, see the contents of the directory / **usr/examples/gated**.

4-18. SLIDE: Point-to-Point Interfaces

Point-to-Point Interfaces

```
traceoptions "/var/adm/gated.trc" size 1m files 2 route ;  
  
interfaces { options strictintfs ;  
    interface lan0 passive ;  
    interface du0 simplex passive ;  
    define 10.0.0.2 pointopoint 10.0.0.1 ;  
};  
  
autonomous system 911 ;  
  
...
```

a646135

Student Notes

Point-to-point interfaces need to be explicitly defined because such interfaces may not be present when `gated` is started. `gated` considers it an error to reference a nonexistent interface in the configuration file.

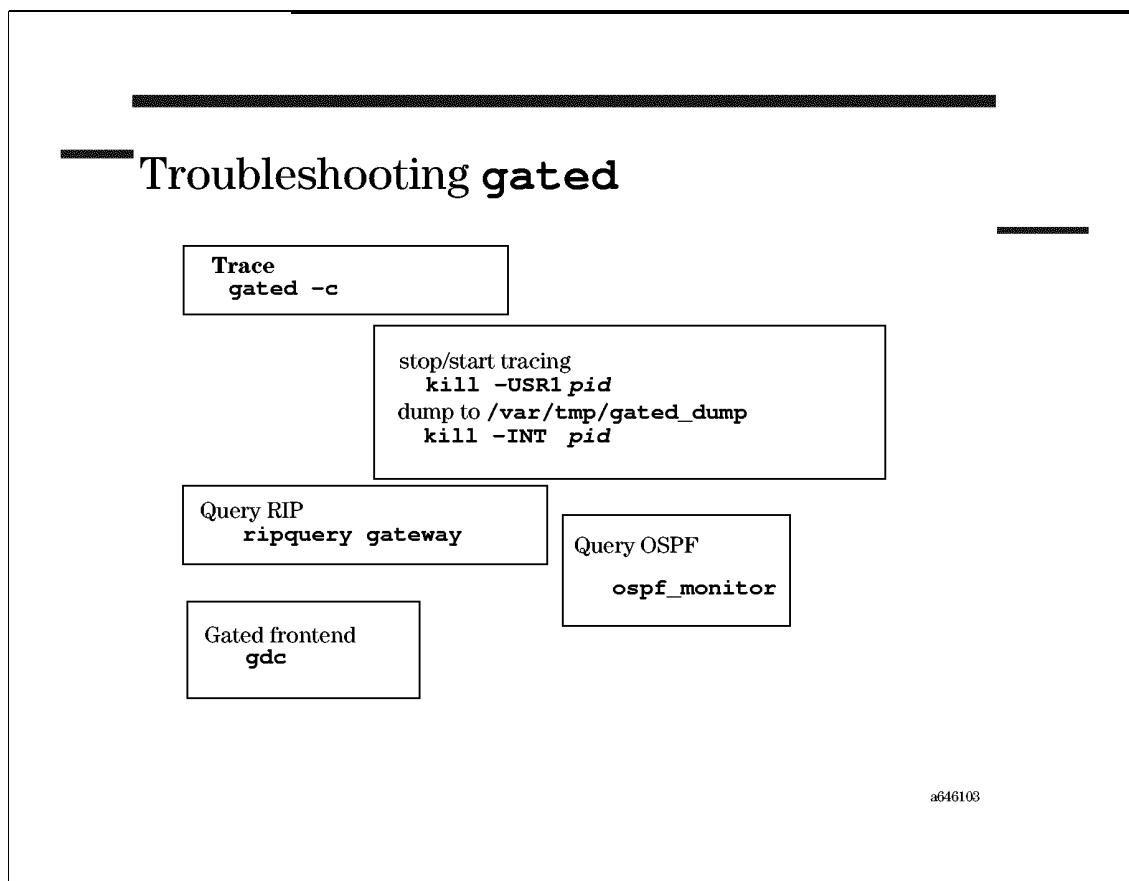
The `define` clause allows specification of such an interface so it can be referenced in the configuration file.

The definition for a point-to-point connection within the interface statement is

`define remote_address pointopoint local_address`

The option `simplex` for `ni0` defines an interface as unable to hear its own broadcast packets.

4-19. SLIDE: Troubleshooting **gated**



Student Notes

Several commands exist for troubleshooting **gated**:

To check the syntax of `gated.conf`, issue the command:

```
gated -C
```

Signals for **gated** include

- **SIGUSR1**—stop/start tracing
- **SIGINT**—dump routing information to `/var/tmp/gated_dump`

Query RIP Routing Information

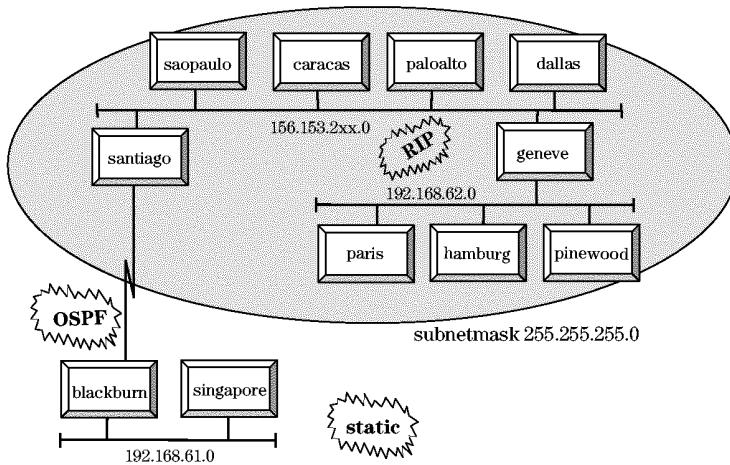
ripquery is a support tool that can be used to query **gated** for RIP routing information. **ripquery** sends out two types of commands: a **POLL** command or a RIP request command. **gated** responds to a **POLL** command by listing all routes learned from RIP that are in its routing table. This does not include the interface routes for the local network or routes from other protocols that are announced via RIP on that interface. It does include routes from other protocols that are being imported by **gated** on the node.

Query OSPF Routing Information

ospf_monitor is a support tool that can be used to query OSPF routers for information on OSPF routing tables, interfaces and neighbors, as well as data on AS external databases, link-state databases, error logs, and input/output statistics. Running the **ospf_monitor** command displays a prompt that allows you to enter interactive commands. See manpage **ospf_monitor** for details on using this tool.

4-20. LAB: RIP Configuration—Default Rerouting

RIP Configuration – Default Rerouting



a646215

Purpose

The lab is designed to promote student understanding of routing tables and dynamic routing protocols. Students should understand the importance of routing tables, differences between internal and external routing protocols, and be able to demonstrate simple configuration and use of the `gated` dynamic routing daemon.

Directions

For this lab, refer to the configuration shown . Notice that you will be using a combination of static and dynamic routing.

All non-gateway systems on the 192.168.61, 192.168.62, and 156.153.2xx networks will manage routing by using `gated` and the routing information protocol (RIP). End systems on networks with only one possible route out from the local network more typically use either static or perhaps proxy address resolution protocol (ARP) mechanisms. This lab is using `gated` on all systems to facilitate learning how dynamic routing works.

The systems **geneve** and **santiago** will perform two functions. First they will act as routers, and second they will exchange dynamic routing information via the **gated** daemon and RIP.

For all systems:

1. The network should be configured as in the above figure or as the instructor may otherwise direct. Note that in the following instructions the node names used are from the figure. Your instructor will map these into names used at your site, as well as indicate which systems will be used as the gateways, and which nodes will be moved to the 192.168.61 or 192.168.62 networks as appropriate.
2. Verify that **/etc/hosts** is current and complete. This will generally mean adding the IP addresses and hostnames of the systems on the net 192.168.61 and net 192.168.62 side of the gateway systems.
3. The systems will need to adjust their routing so that they can reach the other networks (192.168.61, 192.168.62, or 156.153.2xx). This is done through dynamic routing, so you will need to remove any non-local static routes from your system. Use the **route** command with the **delete** option. If you modified **/etc/rc.config.d/netconf** there to include those routes, you will need to delete them there also.
 - a. Run **/sbin/set_parms initial** on the net 192.168.61.0 and 192.168.62.0 systems to change hostnames, IP addresses, and default routes. The net 192.168.61.0 systems other than **blackburn** should use **blackburn** as their default gateway. The net 192.168.62.0 systems other than **geneve** should use **geneve** as their default gateway. Systems **santiago** and **blackburn** should use **blackburn** and **santiago** respectively as their default gateways.

NOTE: You may need to delete the existing default route first.

Upon completion, verify local network connectivity using **ping**.

The systems **paris**, **hamburg**, and **pinewood** will need to be moved. System **geneve** will have to configure its second interface. Note that you can use SAM to do this.

4. Use the file **/var/tmp/gated.trc** as the trace file for **gated**. Be sure to clear it each time **gated** is restarted.

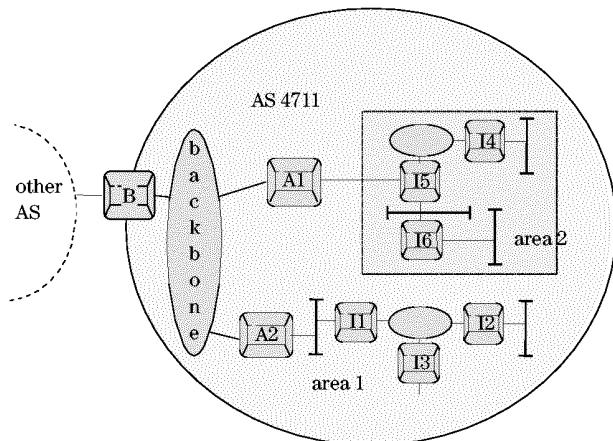
Trace all general information as well as information about RIP.

5. Configure all of the systems on the 192.168.62 network with the following in mind:
 - a. Which version(s) of RIP can be used?
 - b. Which systems have to broadcast their routing information via RIP?
 - c. Which systems may passively receive RIP packets?
 - d. For all systems, define a default route. Why is it necessary to define a default route?
6. **blackburn** and **santiago** exchange their information via RIP. What do the interface statements look like for a point-to-point link? Use RIP on the serial link.
7. Check the syntax of your **gated** configuration file, **/etc/gated.conf**, before starting the **gated** daemon.
8. Start your **gated** daemon. Watch the routing tables and trace file.

Is everything proceeding correctly? Be patient, as it will take some time before the daemons on the other systems in the room are running and communicating. Dynamic routing takes a little time before all of the routes are propagated.

4-21. SLIDE: OSPF Concepts

OSPF Concepts



a646104

Student Notes

OSPF is a link-state routing protocol designed to distribute routing information between routers in a single autonomous system (AS).

Areas

OSPF allows routers, networks, and subnetworks within an AS to be organized into subsets called areas.

An area is a grouping of logically contiguous networks and hosts. Instead of maintaining a topological database of the entire AS, routers in an area maintain the topology only for the area in which they reside. Therefore, all routers that belong to an area must be consistent in their configuration of the area.

The topology of an area is hidden from systems that are not part of the area. The creation of separate areas can help minimize overall routing traffic in the AS.

Internal Routers

Routers that have all their directly-connected networks in the same area are called internal routers (I1–I6).

Area Border Routers

Routers that connect one AS to another are called AS boundary routers. AS boundary routers exchange routing information with routers in other autonomous systems. An AS boundary router learns about routes outside of its attached AS through exchanges with other routing protocols (such as EGP) or through configuration information. Each AS boundary router calculates paths to destinations outside of its attached AS.

Backbone

The set of routers that exchange OSPF protocol packets between areas in an autonomous system is called the backbone.

There are two levels of routing in the AS:

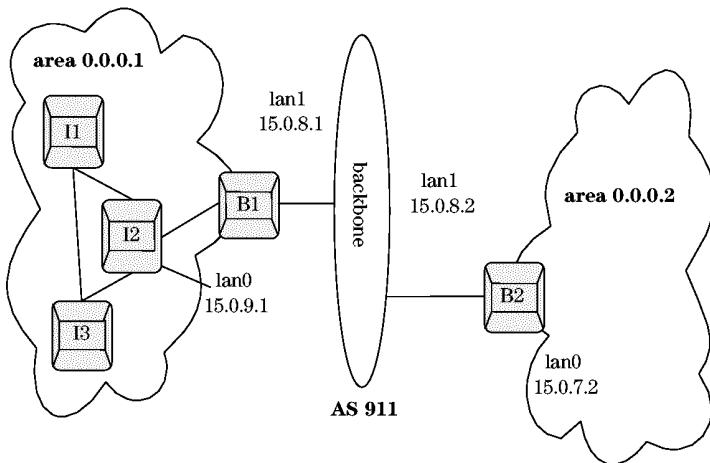
Intra-area routing	The source and destination of a packet both reside in the same area. Routing is handled by internal routers.
Inter-area routing	The source and destination of a packet reside in different areas. Packets travel an intra-area route from the source to an area border router, then travel an inter-area route on a backbone path between areas, then finally travel another intra-area route to the destination.

NOTE:

The OSPF protocol should be enabled for routers only. Once the OSPF protocol is enabled for a system, the system is treated as a router, not a host, by other routers.

4-22. SLIDE: Example of OSPF Configuration

Example for OSPF Configuration



a69077

Student Notes

The following slides refer to the configuration files associated with the above example. In autonomous system 911, OSPF is used as interior gateway protocol.

The two areas 0.0.0.1 and 0.0.0.2 are linked via the backbone.

4-23. SLIDE: OSPF Internal Router

OSPF Internal Router

```
...
ospf yes { area 0.0.0.1
{ interface lan0 cost 5
  { enable ;
    priority 15 ;
    hellointerval 120 ;
    routerdeadinterval 300 ;
    retransmitinterval 30 ; } ;
  interface lan1 cost 5
  { enable ;
    priority 15 ;
    hellointerval 120 ;
    routerdeadinterval 300 ;
    retransmitinterval 30 ; } ;
...
} ;
```

a646136

Student Notes

To configure OSPF on system I2, `/etc/gated.conf` should include the following:

ospf The `ospf` clause activates the OSPF protocol. An internal router is associated with one area. Both interfaces `lan0` and `lan1` belong to the same area.

Interface options

cost	The cost clause can optionally be specified to define a cost of sending a packet on the interface. The outbound side of each router interface is associated with a configurable cost. Lower cost interfaces are more likely to be used in forwarding data traffic. Cost values are assigned at the discretion of the network or system administrator. While the value is arbitrary, it should be a function of throughput or capacity of the interface: the higher the value, the lower the throughput capacity. Thus, the interfaces with the highest throughput or capacity should be assigned lower cost values than other interfaces. Interfaces from networks to routers have a cost of 0.
priority	Specifies the priority of the router to become the designated router. When two routers attached to a network both attempt to become the designated router, the one with the highest router priority value takes precedence.
hellointerval	Specifies the number of seconds between transmissions of OSPF Hello packets. Smaller intervals ensure that changes in network topology are detected faster, however routing traffic can increase. Default: None (you must specify a value) Range: 0–255
routerdeadinterval	Specifies the number of seconds that hello packets are not received from a router before it is considered <i>down</i> or <i>inactive</i> by its neighbors. This value should be some multiple of the hellointerval value. Default: None (you must specify a value) Range: 0–65535
retransmitinterval	Specifies the number of seconds between link state advertisement retransmissions for adjacencies belonging to this interface.

4-24. SLIDE: OSPF Area Border Router

OSPF Area Border Router

```

ospf yes {
    area 0.0.0.1
    {
        interface lan0 cost 5
        {
            enable ;
            priority 15 ;
            hellointerval 120 ;
            routerdeadinterval 300 ;
            retransmitinterval 30 ; } ;
    } ;
    backbone
    {
        interface 15.0.8.1
        {
            enable ;
            priority 20 ;
            transitdelay 30 ;
            hellointerval 120 ;
            routerdeadinterval 300 ;
            retransmitinterval 30 ; } ;
    } ;
}

```

a646105

XXXXXX XY X

1

© 1998 Hewlett-Packard Co

Student Notes

An area border router connects multiple areas. Whenever an area border router is configured, backbone information must be provided. The OSPF backbone distributes routing information between areas. Backbones are defined with the same statements and clauses as areas. The **backbone** statement is used to define a router as a backbone router.

The **backbone** statement must follow the **area** statement(s) in the **/etc/gated.conf** file.

The **transitdelay** keyword in the **backbone** definition specifies the estimated number of seconds to transmit a **link stat update** over this interface. **transitdelay** takes into account transmission and propagation delays and must be greater than 0.

NOTE:

Backbones must be directly-connected, or contiguous.

4-25. SLIDE: Exporting Routes

Exporting Routes

```
...  
export proto ospfase  
{  
    proto direct ;  
    proto ospf ;  
} ;
```

gated.conf on B1

```
...  
export proto egp as 323  
{  
    proto rip { all metric 1 ; } ;  
    proto direct { all metric 1 ; } ;  
} ;
```

gated.conf on AA

a646137

Student Notes

Export statements determine which routes are exported from the **gated** forwarding table to the routing protocols. You can also restrict which routes are exported and assign metrics (values used for route selection) to be applied to the routes after they have been exported.

The format of the export statement varies according to the protocol to which you are exporting routes and the original protocol used to build the routes you are exporting.

For exporting to EGP and OSPF, an export statement always must be specified.

export This export statement exports to OSPF the route that was imported to the
proto **ospfase** **gated** forwarding table in the example above. The exported routes were
originally learned by:

other OSPF **proto ospf ;**
information

direct connected **proto direct ;**
interfaces

NOTE:

EGP also transfers a distance parameter in its protocol header. **metric** specifies the outgoing metric for all routes propagated by this node referring to a route learned by **rip** or connected directly. Although not explicitly mentioned in the manual, a metric parameter must be specified. The value need not represent the real hopcount.

Module 5 — Managing the NIS Environment

Objectives

- Upon completion of this module you will be able to do the following:
 - Discuss NIS maps and their function within NIS.
 - Generate new NIS maps.
 - Configure the automounter to use NIS maps.

5-1. SLIDE: The Network Information Service

The Network Information System

- manages configuration data for networked clients
- makes use of a distributed database
- increases the reliability of networks by ensuring the accuracy of client configuration data

a646120

Student Notes

The Network Information Service (NIS) stores and manages configuration data for networked clients by means of a distributed database. It is typical of many network configurations that many clients make use of identical configuration files. NIS relieves the client systems and their administrators of the burden of keeping these files current by storing copies of the configuration files on NIS servers. NIS clients access these files rather than making use of local copies. This has several advantages:

- NIS clients need not use up local storage to keep copies of these files.
- The configuration files can be maintained in one location, from which accurate and up-to-date copies may be disseminated across the network.

Examples of configuration files maintained under NIS are `/etc/hosts`, `/etc/passwd`, and `/etc/group`. Data provided by many other files may also be managed under NIS.

NIS Maps and Their Distribution

A NIS server maintains clients' data in data files called **maps**. These maps are not identical to the files from which their data is drawn; instead, they are actually indices built from the configuration files. This module describes in detail how to build a particular set of NIS maps used by the automounter, a component of the Network File System (NFS).

The NIS server itself may be one of two types:

- a **master server**, on which the master copy of each map is maintained, or
- a **slave server**, which receives updated copies of the master copy from the primary server and responds to client requests for information.

This distribution arrangement improves network performance by making certain that copies of accurate information are always reasonably *close* to clients who need them, while ensuring that all the information is updated from a common source.

5-2. SLIDE: Generating NIS Maps with make

Generating NIS Maps with **make**

- Uses the makefile **/var/yp/Makefile**.
- Creates or rebuilds maps from ASCII configuration files.
- Maps may then be distributed to other servers for use by clients.

a646121

Student Notes

The **make** command uses the makefile **/var/yp/Makefile** to build one or more NIS maps (databases) on a master NIS server. If no arguments are specified, **make** (when executed within the directory **/var/yp**) either creates maps, if they do not already exist, or rebuilds maps that are not current. These maps are constructed from ASCII configuration files like these:

```
/etc/group  
/etc/hosts  
/etc/netgroup  
/etc/networks  
/etc/passwd  
/etc/protocols  
/etc/publickey  
/etc/rpc  
/etc/services  
/etc/vhe_list
```

Other sources may be used as well, as we will see in the case of creating NIS maps for the NFS automounter.

If any maps are supplied on the command line, `make` creates or updates those maps only. Permissible names for maps are the filenames within `/etc` listed above.

In addition, specific maps can be named—such as `netgroup.byuser` or `rpc.bynumber`. The names employed for automounter maps will be discussed later.

NOTE:

The `make` command is used, instead of `ypmake` (see *ypmake(1)*). The makefile `/var/yp/Makefile` no longer calls the `ypmake` script; now it actually constructs the maps. All NIS commands have been modified to use the makefile instead of `ypmake`.

5-3. SLIDE: The Automounter and NIS

The Automounter and NIS

- The automounter typically is used in large networks where NIS also is used.
- NIS maps can be used to replace local files on each client, thus simplifying overall administration.
- The HP-UX NIS makefile provides rules for the **auto_master** map; rules for **auto_home** can be created from these.

a646122

Student Notes

The automounter function is provided by a daemon (**automount**) that automatically and transparently mounts NFS file systems as needed. It monitors attempts to access directories that are associated with an automount map, along with any directories or files that reside under them. When a file is to be accessed, the **automount** daemon mounts the appropriate NFS file system.

NIS Administration of Clients' Automounters

The automounter can be set up and administered on each client in the network environment, but the overhead of administration can be greatly reduced if NIS is used to provide centralized maps to drive the automounter.

The automount Command

The **automount** command (running as a daemon) attempts to locate a NIS map called **auto_master** if it is started with no arguments, which suggests how closely related the automounter and NIS are.

Each automounter process is started via the **/usr/sbin/automount** command, and must be given two main arguments:

- which directory (or directories) to watch
- a map, which is either a local file or a NIS map, and which provides mounting instructions

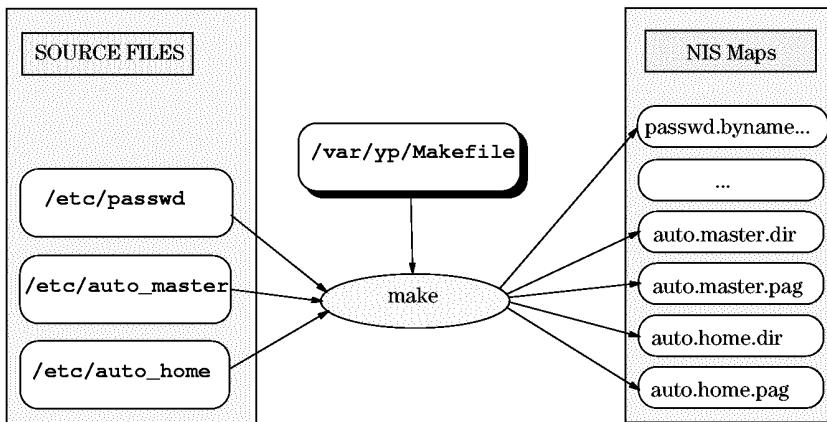
Types of Automounter Maps

There are three basic types of automounter maps: the *built-in* map, the *direct* map, and the *indirect* map. Each system may start up several automounters, one with the built-in map, one with the direct map, and possibly several indirect maps, with each monitoring the usage of one directory. If a system is using more than one of these maps, it is likely that a *master* map will be created and used instead, since this simplifies the start-up. With a master map, only one **automount** command need be specified, since the master map will include the information to start the specified automounters each with its own map.

The master map (**/etc/auto_master**) and the indirect map for home directories (**/etc/auto_home**) are frequently used automounter maps. When managed under NIS, the **auto_home** map can be used to allow any user in a given network environment to login to any system and have their **home** directories NFS mounted to the system they are using, thus allowing users to have their files follow them throughout the network.

5-4. SLIDE: Overview of Creating New NIS Maps

Overview of Creating New NIS Maps



a646123

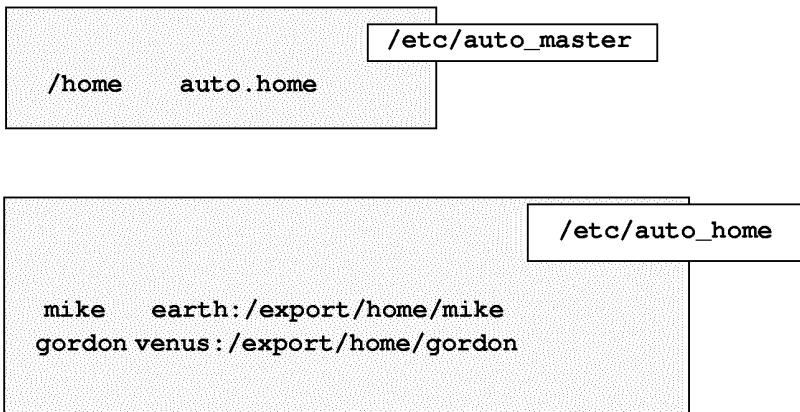
Student Notes

The automounter maps are a good example of the procedure for creating NIS maps. The `make` command creates the NIS maps from the ASCII source files under the control of `/var/yp/Makefile`. This makefile must contain an entry for each NIS map to be created by `make`.

The HP-UX NIS makefile `/var/yp/Makefile`, as shipped, contains rules for creating the `auto_master` file, but it makes no reference to `auto_home`. To place `auto_home` under the control of NIS, it is necessary to add rules to generate the necessary maps. This can be managed easily by copying all the `auto_master` rules in the `Makefile`, duplicating the text with a paste operation and editing the copied content so that it refers to `auto_home`. It's also necessary to modify the `auto_master` file on the NIS primary server to refer to the `auto_home` map.

5-5. SLIDE: Step 1: Editing the NIS Source Files

Step 1: Editing the NIS Source Files



a63006

Student Notes

STEP 1 - Edit the NIS Source Files

To produce NIS maps called `auto.master` and `auto.home`, you must create the initial source files `auto_master` and `auto_home`.

Note that the maps use a dot, but the source files use an underscore. In newer automount implementations, the master file is named `auto_master`. The NIS map still is called `auto.master`.

In the master file, use the NIS nickname `auto.home` to redirect the automount daemon to the NIS map rather than to the local file.

```
# example auto.master source file
/net      -hosts          # use built-in map
/-       /etc/auto.direct  # use local /etc/auto.direct file
/home    auto.home        # use NIS auto.home map
```

5-6. SLIDE: Step 2a: Editing /var/yp/Makefile

Step 2a: Editing /var/yp/Makefile

- Append `auto.home` to all entry.

/var/yp/Makefile

```
all: passwd group hosts networks rpc services protocols \
    netgroup aliases publickey netid vhe_list auto.master auto.home
```

- Duplicate `auto_master.time` target and substitute `master` by `home`.

```
$ (YPDBDIR) /$ (DOM) /auto_home.time: $(DIR) /auto_home
@{sed -e "s/^[\ | ]*/\g" -e "/^#/d" -e s/#.*$$// < \
$(DIR) /auto_home $(CHKPIPE) | \
$(MAKEDBM) - $(YPDBDIR) /$ (DOM) /auto.home;
@touch $(YPDBDIR) /$ (DOM) /auto_home.time;
@echo "updated auto.home".
@if [ ! $ (NOPUSH) ]; then $(YPPUSH) -d $ (DOM) auto.home; fi
@if [ ! $ (NOPUSH) ]; then echo "pushed auto.home"; fi
```

a646124

Student Notes

Step 2a - Edit /var/yp/Makefile

- Append `auto.home` to the `all` entry.

```
all: passwd group hosts networks rpc services protocols \
    netgroup aliases publickey netid vhe_list auto.master auto.home
```

- Duplicate `auto_master.time` target and substitute `home` for `master` in the copied text.

```
$ (YPDBDIR) /$ (DOM) /auto_master.time: $(DIR) /auto_master
@{sed -e "s/^[\ | ]*/\g" -e "/^#/d" -e s/#.*$$// < $(DIR) /auto_master $(CHKPIPE)) | \
$(MAKEDBM) - $(YPDBDIR) /$ (DOM) /auto.master;
@touch $(YPDBDIR) /$ (DOM) /auto_master.time;
@echo "updated auto.master";
@if [ ! $ (NOPUSH) ]; then $(YPPUSH) -d $ (DOM) auto.master; fi
@if [ ! $ (NOPUSH) ]; then echo "pushed auto.master"; fi
```

```
$(YPDBDIR) / $(DOM) / auto_home.time: $(DIR) / auto_home
@{sed -e "s/^[\ | ]*/;/g" -e "/^#/d" -e s/#.*$$// < $(DIR) / auto_home $(CHKPIPE)) | \
    $(MAKEDBM) - $(YPDBDIR) / $(DOM) / auto.home;
@touch $(YPDBDIR) / $(DOM) / auto_home.time;
@echo "updated auto.home";
@if [ ! $(NOPUSH) ]; then $(YPPUSH) -d $(DOM) auto.home; fi
@if [ ! $(NOPUSH) ]; then echo "pushed auto.home"; fi
```

We will continue editing **/var/yp/Makefile** file on the next slide.

5-7. SLIDE: Step 2b: Editing /var/yp/Makefile

Step 2b: Editing /var/yp/Makefile

```
/var/yp/Makefile

3. Duplicate auto_master target and substitute master by home

auto.home:
@if [ ! ${NOPUSH} ]; then $(MAKE) $(MFLAGS) -k \
$(YPDBDIR)/$(DOM)/auto_home.time DOM=$(DOM) DIR=$(DIR); \
else $(MAKE) $(MFLAGS) -k $(YPDBDIR)/$(DOM)/auto_home.time \
DOM=$(DOM) DIR=$(DIR) NOPUSH=${NOPUSH};fi
```

a646125

Student Notes

Step 2b - Edit /var/yp/Makefile (Continued):

3. Duplicate `auto_master` target and substitute `home` for `master` in the copied text.

```
auto.master:
@if [ ! ${NOPUSH} ]; then $(MAKE) $(MFLAGS) -k \
else $(MAKE) $(MFLAGS) -k $(YPDBDIR)/$(DOM)/auto_master.time \
DOM=$(DOM) DIR=$(DIR) NOPUSH=${NOPUSH};fi
```

```
auto.home:  
@if [ ! $ (NOPUSH) ] ; then $(MAKE) $(MFLAGS) -k \  
$(YPDBDIR) /$(DOM)/auto_home.time DOM=$(DOM) DIR=$(DIR) ; \  
else $(MAKE) $(MFLAGS) -k $(YPDBDIR) /$(DOM)/auto_home.time \  
DOM=$(DOM) DIR=$(DIR) NOPUSH=$(NOPUSH) ;fi
```

This completes the addition of entries to **/var/yp/Makefile**.

5-8. SLIDE: Step 3: Creating New NIS Maps

Step 3: Creating New NIS Maps

- NIS is not active :
 - configure NIS with
 - sam
- OR
- ```
domainname mydomain
ypinit -m
vi /etc/rc.config.d/namesvrs
```

- NIS is active:
  - create new NIS maps:

```
cd /var/yp
make Makefile
```

a646138

## Student Notes

### STEP 3 - Create New Maps

In order to activate the new maps you have to create the NIS maps.

1. If your NIS system is already running, just build the maps with the `make` command:

```
cd /var/yp
make Makefile
```

2. If your NIS system is not active, issue these commands:

```
domainname my_domain
ypinit -m
```

You can use **sam** or the shell commands. If you use the shell commands, you must edit the NIS variables in the configuration file **/etc/rc.config.d/namesvrs** in order to activate NIS at next boot time. To do this, change the following entries to the values shown:

```
NIS_MASTER_SERVER=1
NIS_CLIENT=1
NIS_DOMAIN=domain_name
```

## 5-9. SLIDE: Propagating NIS Maps with **ypxfr**

### Propagating NIS Maps with **ypxfr**

- copies NIS maps from an NIS server
- automatically checks to see if the local map is out-of-date
- Example:

```
/usr/sbin/ypxfr auto.master
```

a646126

## Student Notes

The **ypxfr** command copies a NIS map (database) to the local host from a NIS server by using the NIS services. A map can be copied regardless of its age, or it can be copied depending on whether its modification time (order number) is more recent than that of the local map.

The **ypxfr** command creates a temporary map in the directory `/var/yp/domain` where *domain* is the NIS domain. The **ypxfr** command fills the map with entries from the specified map, obtains the map parameters (master and order number), and loads them. It then clears the old version of the map and moves the temporary map to the cleared location.

For example, to copy an updated automounter map from the master server to a slave server, log in as root on the slave server and issue this command:

```
/usr/sbin/ypxfr auto.mapname
```

## 5-10. SLIDE: **ypxfr Examples**

### **ypxfr Examples**

- Because NIS maps tend to become out of date at different times, it is useful to automate the update process through the use of cron scripts
  - `/var/yp/ypxfr_1perday`
  - `/var/yp/ypxfr_2perday`
  - `/var/yp/ypxfr_1perhour`

a646127

## **Student Notes**

To maintain consistency between NIS servers, **ypxfr** should be executed periodically for every map in the NIS. Different maps change at different rates. For example, the **services.byname** map may not change for months at a time, and might therefore be checked for changes only once a day, such as in the early morning hours. However, **passwd.byname** may change several times per day, so hourly checks for updates might be more appropriate.

A **crontab** file can perform these periodic checks and transfers automatically. Rather than having a separate **crontab** file for each map, **ypxfr** requests can be grouped in a shell script to update several maps at once. Example scripts (mnemonically named) are in **/var/yp**:

- **ypxfr\_2perday**
- **ypxfr\_1perday**
- **ypxfr\_1perhour**

They serve as reasonable rough drafts that can be changed as appropriate. For example, the default version of **ypxfr\_1perhour** contains:

```
@(#)ypxfr_1perday: $Revision: 1.1.211.1 $ $Date: 96/10/09 11:27:17 $
#
(c) Copyright 1987, 1988 Hewlett-Packard Company
(c) Copyright 1985 Sun Microsystems, Inc.
#
ypxfr_1perday - Do daily Network Information Service map check/updates

/usr/sbin/ypxfr group.bygid
/usr/sbin/ypxfr groupbyname
/usr/sbin/ypxfr networks.byaddr
/usr/sbin/ypxfr networksbyname
/usr/sbin/ypxfr protocolsbyname
/usr/sbin/ypxfr protocolsbynumber
/usr/sbin/ypxfr rpcbynumber
/usr/sbin/ypxfr servicesbyname
/usr/sbin/ypxfr ypservers
/usr/sbin/ypxfr vhe_list
```

---

## 5-11. LAB: NFS Automount Using NIS Maps

### Directions

In this lab, two systems make up a group. Both systems establish a NIS domain: one as the NIS client; the other as the NIS server.

Each NIS domain has two users: Romeo and Juliet.

Physically, Romeo's home directory is located on the NIS server, and Juliet's home directory is located on the NIS client.

Each physical home directory is located in the subdirectory `/export/balcony`

`/export/balcony/juliet` on the NIS client for Juliet

`/export/balcony/romeo` on the NIS server for Romeo

The *watch* point for the `automount` daemon is `/balcony` where the `automount` will create the symbolic links. To avoid conflicts with the `/home` directory and existing accounts, we will use `/balcony` as the home directory for Romeo and Juliet instead of `/home`.

Juliet's login directory is `/balcony/juliet`

Romeo's login directory is `/balcony/romeo`

In other words the `/etc/passwd` entries for the home directories should be

`/balcony/juliet` for Juliet

`/balcony/romeo` for Romeo.

In reality these directories are symbolic links created by the `automount` daemon.

#### 1. Define the Necessary User Environment

On the NIS server, define the two users. Suggestion: SAM is a good tool to use for defining the users, but you must adjust the home directories and the `/etc/passwd` file manually.

Remember, although both Juliet and Romeo are configured in the NIS `passwd` map on the NIS server, the physical home directory of Juliet is located on the NIS client, and the physical home directory of Romeo is located on the NIS server.

As the automounter at 11:00 does not look for `auto.master` maps, modify the `/etc/rc.config.d/nfsconf` file:

```
AUTOMOUNT=1
AUTO_MASTER=''/etc/auto_master''
AUTO_OPTIONS=''-f $AUTO_MASTER''
edit /etc/auto_master
+ auto.master
```

2. Using either `vipw` or `vi`, edit the `/etc/passwd` file to change the home directories to `/balcony/juliet` and `/balcony/romeo`, respectively.

3. For Juliet, copy the entire home directory `/export/balcony/juliet` from the NIS server to the NIS client. You can best accomplish this task by using a remote shell and the `tar` command:

#### 4. Create the Source Files for the NIS Automount Maps

Configure the automount configuration files in the following way:

On the NIS client, mount `/export/balcony/juliet` at `/balcony/juliet`. On the NIS server, mount `/export/balcony/romeo` at `/balcony/romeo`.

These files will serve as source files only for the NIS maps. The automount daemon should request both maps via NIS.

For configuration files, use the following:

`/etc/auto_master` and `/etc/auto_home`

Assuming that `paris` is the NIS client and `geneve` is the NIS server, what will your configuration files look like?

5. Extend the files `/var/yp/Makefile` so that the `/etc/auto_home` file will be built as the `auto.home` NIS map.

## 6. Configure Each System as an NFS Server

Both the NIS client and the NIS server function as the NFS server for Juliet's and Romeo's home directories, respectively. On both systems, export the appropriate file systems. Then activate the NFS server daemons on each system. This is done automatically if you use a tool such as SAM.

## 7. Configure NIS

On each system configure the NIS domain, including both the NIS client and the NIS server.

(If you want to continue with the Shakespearean theme, Montague, Capulet, Benvolio, Mercutio, Tybalt, and Verona are good selections.)

## 8. Activate the Automount Daemon

On both systems, determine if the automount daemon is running. If it is running, stop it. Do not use the signal 9 to stop the automount daemon!

Restart the automount by using the `-v` option. You want the automount daemon to read the NIS master map instead of the local `/etc/auto_master` file.

If you get an error, look at the `var/adm/syslog/*` file.

What does the `-v` option do?

## 9. Test Your Automount Configuration

On both systems login as Juliet and as Romeo. On the NIS client create a test file to check the correct configuration.

Do you see the test file on the NIS server also?

Are you logged into the correct home directory?

Where are the home directories physically located?

Using the mount command, look at the mount table.

---

# Module 6 — NIS+ Administration

## Objectives

Upon completion of this module, you will be able to do the following:

- Describe the purpose of NIS+ and its name space.
- Plan the NIS+ name space.
- Set up the NIS+ name space.
- Administer NIS+.

---

## 6-1. SLIDE: Overview of NIS+

### Overview of NIS+

- Configuration data is maintained for large numbers of hosts.
- Data is contained in a distributed, replicable database.
- Data is accessible from any host.
- Centrally stored data is propagated throughout the network.

a64636

## Student Notes

NIS+ allows you to maintain configuration information for many hosts in a set of distributed databases. You can read or modify these databases from any host in the network, if you have the proper credentials and access permissions. Common configuration information, which would have to be maintained separately on each host in a network without NIS+, can be stored and maintained in a single location and propagated to all of the hosts in the network.

## 6-2. SLIDE: Advantages of NIS+ over NIS

### Advantages of NIS+ over NIS

- NIS+ supports the NIS+ name space, a hierarchical domain structure.
- NIS+ may be expanded as your organization grows.
- NIS+ is not limited by subnet boundaries.
- NIS+ is secured by keyed authentication and Data Encryption Standard (DES).
- NIS+ is modifiable from any host.
- NIS+ tables are updated incrementally. There is no need to push whole tables to the replica servers.
- NIS+ tables may contain many columns. Entries may be searched for on the basis of their row/column positions.

a64637

## Student Notes

NIS+ has the following advantages over NIS:

- NIS+ supports a hierarchical domain structure called the **NIS+ name space**. You can create a separate domain for each workgroup or department in your organization. Each domain can be managed independently of the others. Hosts in any domain may have access to information in all the other domains in the name space.
- The NIS+ name space can grow with your organization. Because information may be distributed over multiple domains, each with its own servers, the size of the NIS+ name space is not limited by the capacity of any single server.
- NIS+ is not limited by subnet boundaries. NIS+ clients do not broadcast requests, so you do not need a server on every subnet.
- NIS+ is secure. It uses a private key/public key authentication scheme with **DES** encryption. Every user and host in the name space has its own unique credentials, and you can decide

which users and hosts will be allowed to read or modify the information in each NIS+ domain.

- You can modify the information in a NIS+ table from any host in the name space. Modifications are made directly to the NIS+ table, so you do not have to rebuild the table from a file.
- Replica servers in NIS+ domains receive each table update as it is made. You do not have to push whole tables to the replica servers.
- A NIS+ table may contain many columns, and you can search for entries based on the information in any column.

---

## 6-3. SLIDE: Disadvantages of NIS+

### Disadvantages of NIS+

- difficult to administer
  - requires well-trained administrators
  - different administrative paradigm from NIS
- databases are not automatically backed up to flat files
  - backup strategy required: databases first must be dumped to flat files, and then the flat files are backed up

a64638

## Student Notes

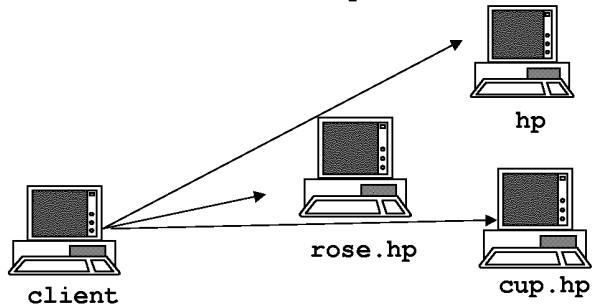
NIS+ has the following disadvantages:

- NIS+ is difficult to administer. It requires dedicated system administrators trained in NIS+ administration. NIS+ administration is very different from NIS administration.
- The NIS+ databases are not automatically backed up to flat files. The system administrator must create and maintain a backup strategy for NIS+ databases, which includes dumping them to flat files and backing up the files.

## 6-4. SLIDE: Structure of the NIS+ name space

### Structure of the NIS+ Name space

- The hierarchical name space is similar to DNS.
- Information in NIS+ servers is visible to authorized users and applications across the entire domain hierarchy.
- Each server contains information only pertaining to its own section of the overall name space.



a64639

### Student Notes

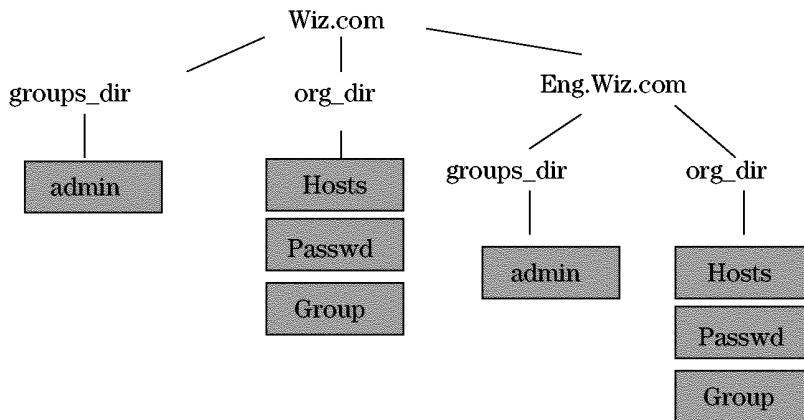
A NIS+ name space may be *flat*, consisting of a single domain, or it may be *hierarchical*, like the Domain Name Service (DNS) domain structure. Every name space has exactly one **root domain**. All other domains are subdomains of the root domain.

The master server of the root domain is the **root master server**. Master servers of subdomains are called **non-root master servers**. NIS+ backup servers are called **replica servers**. Replica servers are the NIS+ equivalent of slave servers in an NIS domain. The Replica servers of the root domain are called **root replica servers**, and the replica servers of the non-root domains are called **non-root replica servers**. A server may serve more than one domain, but it is not recommended.

All NIS+ servers must also be NIS+ clients. The root master and root replica servers are clients of the root domain. However, a non-root server serves a subdomain of the domain in which it is a client. The domain in which it is a client is the parent domain of the domain it serves.

## 6-5. SLIDE: Structure of a NIS+ Domain

### Structure of a NIS+ Domain



a64640

### Student Notes

A NIS+ domain is a NIS+ directory whose name is the domain name. A NIS+ directory is not an HP-UX directory. You must use `nisls(1)` command to see the directory structure of a NIS+ domain.

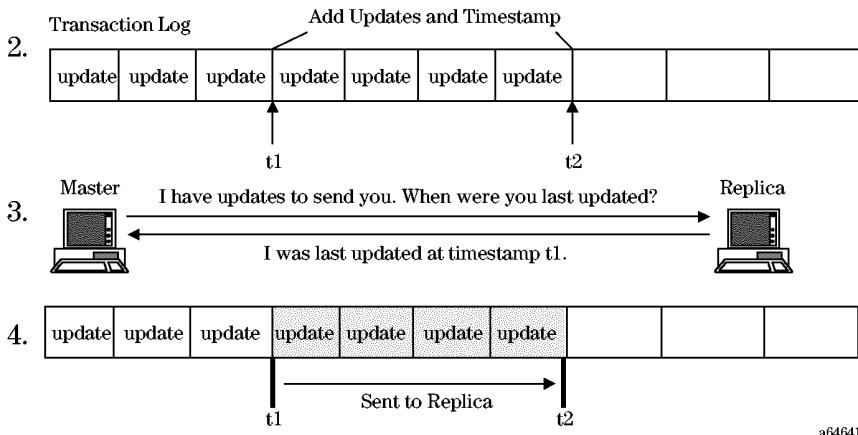
Each NIS+ domain contains two NIS+ subdirectories, call `groups_dir` and `org_dir`. The `groups_dir` subdirectory contains NIS+ groups, which are like HP-UX groups except they are used only to determine access to NIS+ objects. The `org_dir` subdirectory contains all the standard NIS+ tables. Any other NIS+ directories are subdomains.

The full name of a NIS+ object includes the names of all the NIS+ directories in its directory path. For example, the full name of the hosts table in the `Wiz.com.` domain is `hosts.org_dir.Wiz.com.` To specify an entry in this table, you need to specify enough column values to uniquely identify it. For example, to identify a host whose economical name in the `cname` column is `romney`, you would specify `[cname=romney]` , `hosts.org_dir.Wiz.com.` If the default domain on the local host is `Wiz.Com.`, you can leave off the domain name and type just `hosts.org_dir` or `[cname=romney]`, or `hosts.org_dir`. Domain names always end in a period, except when you are setting the default domain with the `domainname` command.

## 6-6. SLIDE: How NIS+ Information is Stored and Propagated

### How NIS+ Information is Stored and Propagated

- Upon receiving an update, the master server waits 10 seconds for any other updates that may arrive.



### Student Notes

NIS+ information is stored in the `/var/nis` directory. On a server, the `/var/nis/data` subdirectory, or the `/var/nis hostname` subdirectory (where `hostname` is the name of the local host), contains the NIS+ directories and tables that make up the domain.

You can make changes to the NIS+ objects from any NIS+ client in the name space, if you are authenticated and have the proper access permissions. Whenever anyone makes a change to a NIS+ object, the change is sent to all the replica servers. NIS+ sends only the changes to replica servers, not whole tables. A transaction log in the `/var/nis` directory on each server keeps track of all the changes that have been made. To keep the transaction log from growing too large, you must checkpoint the domain regularly. When you checkpoint the domain (with the `nisping(1M)` command), the changes in the transaction log on all the servers are incorporated into the tables on the disk, and the transaction log is cleared.

A NIS+ client may get information from any domain in the name space, if it is authenticated and has the proper access permissions. Each NIS+ client has a file called `nis_cold_start` in the `/var/nis` directory, which contains the internet addresses of servers the client can contact for NIS+ information. Because all servers are also clients, every server has a cold start file, too. A

NIS+ client does not *bind* to a server the way a NIS client does. It contacts a server directly to request information. If a client requests information about a domain from a server that does not serve the domain, the server tells the client how to find a server that serves the domain. As the client learns about other servers, it adds the information to a cache file called *nis\_shared\_dircache* in the `/var/nis` directory, and it uses the information to find servers later.

## 6-7. SLIDE: NIS+ Tables

### NIS+ Tables

| Key |         | Value                 |
|-----|---------|-----------------------|
| NIS | hpnecdo | 15.56.216.199 hpnecdo |
|     | hpnecsm | 15.56.216.200 hpnecsm |

```
NIS> ypmatch hpnecdo hostsbyname
```

| Name        | Year | Winner     | Classification |
|-------------|------|------------|----------------|
| Derby       | 1992 | LittleFeat | StockCar       |
| Derby       | 1991 | Elvis      | FormulaOne     |
| Stakes      | 1994 | Dabulls    | FunnyCar       |
| AlmadenOpen | 1992 | Elvis      | FormulaOne     |

```
NIS+> nisgrep -h Winner=Elvis races.org_dir.Wiz
```

a64642

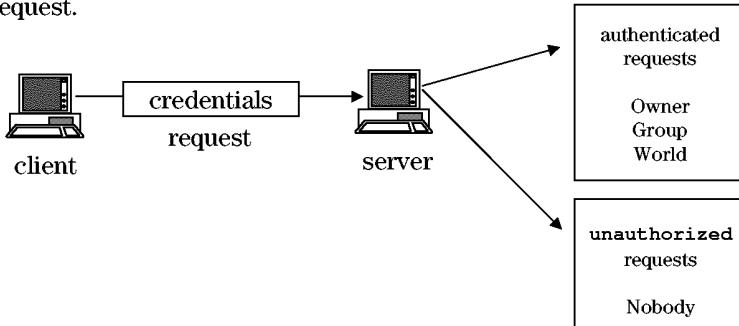
### Student Notes

There are many features that are provided by NIS+ tables. These tables have column entry structure, search paths are accepted, they can be linked and there are different ways in which they can be set up. However, you can create your own table and there are several preconfigured ones provided for you.

## 6-8. NIS+ Authentication and Authorization

### NIS+ Authentication and Authorization

When a NIS+ client sends a request to a server, it automatically includes the credentials of the user or workstation making the request.



The NIS+ server that checks those credentials places the request in one of two categories: *authenticated* or *unauthorized*.

a64643

### Student Notes

Authentication is the process by which NIS+ determines who you are. To be an authenticated NIS+ user, you must have an entry in the `cred` table, and your password must decrypt your secure RPC key, which is stored in the `cred` table. When you log in and supply your password, NIS+ identifies you as a NIS+ principal. If you are a non-root user, your NIS+ principal name is `loginname.domainname`. For example, if you log in as user `ming` in domain `Wiz.Com`, your NIS+ principal name is `ming.Wiz.Com`. If you are a root user, NIS+ identifies you by the host name where you logged in, and your NIS+ principal name is `hostname.domainname`. For example, if you logged in as `root` to host `garlic` in the `Eng.Wiz.Com.domain`, your NIS+ principal name is `garlic.Eng.Wiz.Com`.

The `cred` table stores two types of credentials: `Local` and `DES`. A `Local` credential associates a NIS+ principal name with a user ID. Only non-root users have `Local` credentials. A `DES` credential contains the secure RPC keys for authenticating a NIS+ user. Both root and non-root users may have `DES` credentials. Each NIS+ principal has only one `DES` credential, in his or her home domain, but may have `Local` credentials in many domains. Authorization is the process by which NIS+ determines what you are allowed to do with NIS+ objects. Every NIS+ object has a permissions string that determines who can read, modify, create, or destroy

it. This permissions string is similar to the HP-UX file permissions string that grants read, write, and execute permissions to HP-UX users. NIS+ grants four types of permissions:

- **(r)ead**
- **(c)reate**
- **(m)odify**
- **(d)estroy**

It grants permissions to four types of users:

- **nobody**
- **owner**
- **group**
- **world**

User **nobody** is the group of all unauthenticated users. If you have no entry in the **cred** table, NIS+ identifies you as user **nobody** and assigns you a user ID of -2. The owner of a NIS+ object is typically the NIS+ principal who created it. However, you can change the owner of a NIS+ object with the **nischown(1)** command. The group is the NIS+ group that owns the object. NIS+ groups are stored in the **groups\_dir** subdirectory under each domain directory.

A NIS+ domain typically has an admin group consisting of the NIS+ principals who administer the domain. Not every NIS+ object has a group owner. For more information on NIS+ groups, type **man 1 nisgrpadm** at the HP-UX prompt. The world is the group of all authenticated NIS+ principals. If you are an authenticated NIS+ principal, but you are not the owner of an object, and you are not a member of the NIS+ group that owns the object, then you will have whatever access permissions are granted to the world. Every NIS+ directory has an owner and permissions associated with it.

Every table has an owner and permissions. Entries and columns in a table have all the permissions the table has, but you can assign more permissions to an entry or column than the table has. An entry's owner and group owner may be different from the owner and group owner of the table to which the entry belongs. When a NIS+ object is created, it inherits a default owner and permissions. Most NIS+ commands have an option for overriding these defaults. You can also change these defaults. Type **man 1 nisdefaults** at the HP-UX prompt for more information.

After a NIS+ object has been created, you can change its owner with the **nischown(1)** command, its group owner with the **nischgrp(1)** command, and its permissions with the **nischmod(1)** command.

---

## 6-9. NIS Compatibility Mode

### NIS Compatibility Mode

- NIS+ compatibility mode allows NIS+ servers to answer requests from both NIS+ and NIS clients.
- Utilities allow the transfer of contents of a NIS map into a NIS+ table and vice-versa.

a64644

### Student Notes

A NIS+ server may serve NIS clients, by running in NIS compatibility mode. NIS compatibility mode is intended as a migration tool, to allow you to migrate your servers from NIS to NIS+ without having to migrate all your clients to NIS+ at the same time. NIS compatibility mode has the following disadvantages:

- NIS compatibility mode is less secure than regular mode. NIS clients cannot be authenticated by NIS+, so NIS compatibility mode allows unauthenticated clients to read the `passwd` table.
- If you have links or concatenation paths in NIS+ tables, NIS clients will not be able to follow them.
- NIS clients may read information only in their default domain. They cannot read information in other domains in the name space.
- Every NIS client must have a server on its local subnet, unless its server name has been set with the `ypset` command.

If any server in a NIS+ domain is running in NIS compatibility mode, all servers for that domain must run in NIS compatibility mode. All servers in a NIS+ domain must be NIS+ servers. You cannot mix NIS servers and NIS+ servers in the same domain.

---

## 6-10. Planning the NIS+ name space

### Planning the NIS+ Name Space

- Determine the number of NIS+ domains needed.
- Determine the number of NIS+ servers needed.
- Determine which hosts will be NIS+ servers.

a64645

### Student Notes

This section explains how to plan your NIS+ name space. It tells you how to perform the following tasks:

- Determine the number of NIS+ domains you need.
- Determine the number of NIS+ servers you need.
- Determine which hosts will be NIS+ servers.

#### Determine the Number of NIS+ Domains You Need

For many sites, all hosts can belong to the same domain, and it is not necessary to set up a hierarchical name space with multiple domains. However, you might want to create multiple domains for the following reasons:

- NIS+ works most efficiently when each domain contains fewer than 10,000 table entries. 10,000 table entries translates roughly into about 1000 users. Therefore, you should create enough domains so that no domain contains more than 1000 users.
- If your site is divided into multiple administrative departments, with a different system administrator for each department, you should allow each system administrator to maintain a separate NIS+ domain. If your site is divided into multiple administrative departments, and each department requires different configuration data and allows access to different users and hosts, you should create a separate NIS+ domain for each administrative department.

## Determine the Number of NIS+ Servers You Need

The following are guidelines for determining the number of NIS+ servers you need in your domain:

- You must configure one master server per NIS+ domain.
- Configure at least one replica server per NIS+ domain, but no more than 10 replica servers per domain.
- A server may serve more than one domain, but it is not recommended.

## Determine Which Hosts Will Be NIS+ Servers

You must go through the following steps in planning a NIS+ domain:

1. Choose servers that are reliable and highly available.
2. Choose fast servers that are not used for CPU-intensive applications. Do not use gateways or terminal servers as NIS+ servers. Do not use NFS or database servers as NIS+ servers.
3. Choose servers with sufficient disk space. NIS+ data is stored in `/var/nis`. The `/var/nis` directory requires approximately 5 Kbytes of disk space per client of the domain. For example, if a domain has 1000 clients, `/var/nis` requires about 5 Mbytes of disk space. You should add an additional 10-15 Mbytes to allow transaction logs to grow. So, for a server in a domain of 1000 clients, you should allocate 15-20 Mbytes of disk space.
4. Choose servers with sufficient memory. The minimum amount of memory required for a NIS+ server is 64 Mbytes, but servers of medium and large domains should have at least 128 Mbytes.

---

## 6-11. Setting Up the Root Master Server

### Setting Up the Root Master Server

1. Log in as root.
2. Set the default domain name.
3. Set \$PATH to include /usr/lib/nis.
4. **nisserver -r** or **nisserver -r -Y**
5. Respond to script prompts.
6. Verify creation of the domain with **nisls -lR**.
7. If the server was previously a NIS server or client, reset these values in /etc/rc.config.d/namesrvrs to 0:  
**NIS\_MASTER\_SERVER, NIS\_SLAVE\_SERVER, NIS\_CLIENT**
8. Run **nisping -Ca** once daily.

a64646

### Student Notes

Before you perform this task, make sure no one is using the host that will be the root master server. The **nisserver** script copies the **/etc/nsswitch.nisplus** file to **/etc/nsswitch.conf**. This may render the host unusable until the NIS+ tables are populated and NIS+ is operational.

1. Log in as root to the host that will be the root master server.
2. Issue the following command to set the default domain name:

```
/usr/bin/domainname default_domain
```

The domain name must have at least two components, for example, **wiz.com**. Do not type a period at the end of the domain name.

3. Set the PATH variable to include /usr/lib/nis. If you are running the C shell, type the following command:

```
setenv PATH $PATH:/usr/lib/nis
```

If you are running the Bourne or Korn shell, type the following commands:

```
PATH=$PATH:/usr/lib/nis
export PATH
```

4. Issue the following command to set up the root master server:

```
nisserver -r
```

If you want the server to run in NIS compatibility mode so that it can serve NIS clients, add the **-Y** option.

```
nisserver -r -Y
```

The **nisserver** script asks you if the information it has is correct. You can change it by typing **n**. The script then allows you to change each piece of information. To make a change, just type the correct information after the incorrect information and press **[Return]**. You cannot change the security level. When the **nisserver** script asks you for a password, type the root password. The **nisserver** script will use the root password to create credentials for the local host in the **cred** table.

5. To verify that the **nisserver** script created the root domain successfully, issue the following command:

```
nisls -lR
```

The **nisls** command should list the domain name, the **org\_dir** and **groups\_dir** NIS+ directories, the admin group, and all the standard.

6. If the host was previously a NIS server or client, set the **NIS\_MASTER\_SERVER**, **NIS\_SLAVE\_SERVER**, and **NIS\_CLIENT** variables to 0 in the **/etc/rc.config.d/namesvrs** file.

7. Create a cron job that runs **nisping -Ca** at least once a day, during a time when the network is not busy. The following example crontab file runs **nisping -Ca** once a day, at 3:00 AM. It directs standard output and standard error from the **nisping** command to the file **/tmp/nisplog.0**.

```
3 * * * /usr/lib/nis/nisplog -Ca > /tmp/nisplog.0 2& >1
```

The **nisping -Ca** command causes all servers of the domain to update their tables with the changes in the transaction log and to clear the transaction log. If you do not issue the **nisping -Ca** command regularly, your transaction log may grow too large, and you may not have enough disk space to checkpoint it.

After you run the **nisserver** script, the local host is set up as the root master server and as a client of the default domain. However, the domain tables are still empty.

For more information, see the following man pages: **nisserver** (1M), **domainname**(1M), **nisls**(1), **nsswitch.conf**(4), **crontab**(1), **rpc.nisd** (1M), and **nis**(1).

---

## 6-12. Populate the NIS+ Tables on the Master Server

### Populate the NIS+ Tables on the Master Server

1. If working from existing files, copy them into a working directory.
2. Remove any entries from the **passwd**, **aliases** or **hosts** files that you do not want distributed across the name space (**root** and other system entries such as **bin**).
3. Remove any fully qualified DNS domain names from **hosts** table.
4. In automounter map names, replace periods with underbars.
5. Run the **nispopulate** script with appropriate options.
6. Verify the tables with **niscat**.
7. Checkpoint the domain with **nisping -Ca**.
8. Reboot the host to force all processes to read the new **/etc/nsswitch.conf** file.

a64647

### Student Notes

You can populate NIS+ tables from files or from NIS maps. Before you populate the master server's tables, you must run the **nisserver** script to create the tables.

---

**NOTE:**

The **nispopulate** script may fail if there is insufficient **/tmp** space on the system. To keep this from happening, you can set the environment variable **TMPDIR** to a different directory. If **TMPDIR** is not set to a valid directory, the script will use the **/tmp** directory instead.

- Log in as root to the master server.
- Populate the NIS+ tables from files, create a temporary directory, and copy the files into it from the **/etc** directory:

```
mkdir /nis+files
cd /etc
```

```
cp auto_home auto_master group hosts mail/aliases netgroup \
networks passwd protocols rpc services TIMEZONE ../nis+files
```

- In the temporary directory, remove the entry for root from the `passwd` file. Remove all other entries with user ID 0 (zero) from the `passwd` file. Remove all the system entries such as `root` and `bin`, with user IDs of less than 100, from the `passwd` file. This should be done both for security and for interoperability with NIS and other NIS+ implementations. Remove any other entries from the `passwd`, `aliases`, or `hosts` files that you do not want distributed across the name space.
- Replace any periods in automounter map names with underbars. For example, if your master map is called `auto.master`, rename it to `auto_master`. If you will populate your NIS+ tables from NIS maps, make sure your NIS map names contain no periods. If you will populate your NIS+ tables from files, make sure your file names contain no periods.
- To populate the NIS+ tables from files, issue the following command:

```
nispopulate -F -p /nis+files -d domainname
```

The `-p` option specifies the path to the files.

To populate the NIS+ tables from NIS maps, issue the following command:

```
nispopulate -Y -h NIS_server_name -a NIS_server_address \
-y NIS_domain -d domainname
```

- The `nispopulate` script asks you if the information it has is correct. You can change it by typing `n`. The script then allows you to change each piece of information. To make a change, just type the correct information after the incorrect information and press `Return`.

If you are populating files on the root master server, you do not need the `-d domainname` option, because the default domain is the domain the host will serve. However, on subdomain master servers, it is very important to specify the domain, because it is different from the default domain.

- To verify that your tables have been populated successfully, issue the **niscat** command for several standard tables. The following example lists the contents of the NIS+ **passwd** table:

```
niscat passwd.org_dir.domainname
```

Issue the following command to checkpoint the domain:

```
nisping -Ca
```

---

**NOTE:**

If you do not have enough swap or disk space, the server will be unable to checkpoint properly, but it won't notify you. To ensure that the checkpoint completed successfully, issue the **niscat** command to list the contents of a table:

```
niscat rpc.org_dir
```

If you do not have enough swap space, you will see the following error message:

```
can't list table: Server busy, Try Again.
```

To fix this problem, increase the swap space and checkpoint the domain again.

---

- Reboot the host to force long-running processes to read the new **/etc/nsswitch.conf** file. (The **nisserver** script copies **/etc/nsswitch.nisplus** to **/etc/nsswitch.conf**.)

The **nispopulate** script populates the **cred** table from the **passwd** and **hosts** files or NIS maps. Every NIS+ principal except the root user on the root master server is given the default NIS+ password, which is **nisplus**. (The credentials for the root user on the root master server were created using the root password.) When you run the **nisclient** script to initialize a client host in the root domain, the **nisclient** script will change the client's credentials to use the client's root password instead of the default NIS+ password. For more information, see the following man pages: **nispopulate(1M)**, **nispasswd(1)**, **niscat(1)**, **nisping(1M)**, and **nis(1)**.

## 6-13. Adding Administrators to the NIS+ admin Group

### Adding Administrators to the NIS+ admin Group

- Add the principals to the group:
  - `nisgrpadm -a admin.domainname NIS+_principal`
- List the members of the group:
  - `nisgrpadm -l admin.domainname`

a64648

## Student Notes

The following procedures are used to add administrators to the NIS+ admin group:

- Type the following command to add NIS+ principals to the admin group:

```
nisgrpadm -a admin.domainname NIS+_principal [NIS+_principal ...]
```

For example, to add users **ming** and **sara** to the admin group in the **Wiz.Com.domain**, you would type the following command:

```
nisgrpadm -a admin.Wiz.Com. ming.Wiz.Com. sara.Wiz.Com.
```

- Issue the following command to list the members of the admin group to make sure the administrators you added are there:

```
nisgrpadm -l admin.Wiz.Com.
```

The members of the NIS+ admin group may make any modifications to the NIS+ objects in the domain that the group owner is allowed to make. See NIS+ Authentication and Authorization for an explanation of NIS+ permissions. It is useful to add non-root users to the admin group, because they can administer the domain while logged into any host in the name space. Root users must be logged in as root to a specific host in order to be recognized as members of the admin group. The admin group is a NIS+ group stored in the **groups\_dir** subdirectory of the domain directory. It is not one of the HP-UX groups stored in the **/etc/group** file or the NIS+ group table. For more information, type **man nisgrpadm(1M)** or **man 1 nis** at the HP-UX prompt.

---

## 6-14. Setting Up NIS+ Client Hosts

### Setting Up NIS+ Client Hosts

1. Look for the current credential on the host (with **nisgrep**). If none, create the credential with **nisclient -co**.
2. Log into the host that will be the client and set the NIS+ domain name with **/usr/bin/domainname domainname**.
3. Set to 0 the **NIS\_\*** variables in **/etc/rc.config.d/namesrvrs**.
4. Set **\$PATH** to include **/usr/lib/nis**.
5. Initialize the client with **nisclient -i -h master\_server\_name**.
6. Verify client-server communication with **nisls** and **niscat**.
7. Reboot the host.

a64649

### Student Notes

Before you set up a NIS+ client host, the master server must be set up and running, and the client must have an entry in the NIS+ hosts table on the master server. Also, make sure no one is using the client host. The **nisclient** script copies the **/etc/nsswitch.nisplus** file to **/etc/nsswitch.conf**. This may render the host unusable until NIS+ is operational.

- Log into the master server and issue the following command to determine whether the client host has NIS+ credentials in the domain's **cred** table:

```
nisgrep client_hostname cred.org_dir.domainname
```

The **nispopulate** script creates credentials for every host that is in the **/etc/hosts** file or NIS hosts map when the command is run. If the client host did not have a hosts entry when **nispopulate** was run, it will not have credentials in the **cred** table.

- If the **nisgrep** command returns nothing, issue the following command on the master server to add a credential for the client host:

```
nisclient -co client_hostname
```

When prompted for a password, type the default password, **nisplus**.

- Log in as root to the host that will be the NIS+ client.
- Issue the following command to set the NIS+ domain name:

```
/usr/bin/domainname domainname
```

- If the host was previously a NIS server or client, set the **nis\_master\_server**, **nis\_slave\_server**, and **nis\_client** variables to 0 in the **/etc/rc.config.d/namesvrs** file.
- Set the PATH variable to include **/usr/lib/nis**. If you are running the C shell, type the following command:

```
setenv PATH $PATH:/usr/lib/nis
```

If you are running the Bourne or Korn shell, type the following commands:

```
PATH=$PATH:/usr/lib/nis
export PATH
```

- Issue the following command to initialize the client host:

```
nisclient -i -h master_server_name
```

If the master server's internet address is not in the client's `/etc/hosts` file, the `nisclient` script prompts you for the master server's internet address. The `nisclient` script will prompt you for the secure RPC password for root. Type the default NIS+ password, `nisplus`. The `nisclient` script will then prompt you for the root password on the client host. After you type the password, the `nisclient` script will change the client host's entry in the `cred` table to use the root password instead of the default password.

- Issue the following command on the new NIS+ client host to verify that the host can receive information from the NIS+ server:

```
nisls
```

The `nisls` command should list the domain name and the `org_dir` and `groups_dir` NIS+ directories.

- To verify that your client can get information from NIS+ tables, issue the `niscat` command for several standard tables. The following example lists the contents of the NIS+ `passwd` table:

```
niscat passwd.org_dir
```

- Reboot the host to force long-running processes to read the new `/etc/nsswitch.conf` file. For more information, see the following man pages: `domainname(1)`, `nisaddcred(1M)`, `nisclient(1M)`, `nisls(1)`, and `nis(1)`.

---

## 6-15. Setting UP NIS+ Replica Servers

### Setting Up NIS+ Replica Servers

1. Log in as root on the host that will be the replica server.
2. Set **\$PATH** to include **/usr/lib/nis**.
3. If this is to be a root replica server, set it up as a client in the root domain. If not, set it up as a client in the parent domain of the domain it will serve.
4. Issue **rpc.nisd** (or **rpc.nisd -Y** if the master server is operating in NIS compatibility mode).
5. Set **nisplus\_server** to 1 in **/etc/rc.config.d/namesrvrs**. Reset **NIS\_\*** variables to 0.
6. Initialize with **nisserver -R -h replica\_host\_name**.
7. Checkpoint the domain by issuing **nisping -a** from the master server.

a64650

### Student Notes

Before you can set up a replica server, the master server must be set up and running, and the hosts table on the master server must contain an entry for the host that will be a replica. When you run the **nisserver** script to initialize a replica server, the NIS+ tables on the master server are copied to the replica. Copying the tables can take anywhere from a few minutes to a couple of hours, depending on the size of your tables, the network load, and the system load on the master and replica servers.

1. Log in as root to the host that will be a replica server.
2. Set the PATH variable to include `/usr/lib/nis`. If you are running the C shell, type the following command:

```
setenv PATH $PATH:/usr/lib/nis
```

If you are running the Bourne or Korn shell, type the following commands:

```
PATH=$PATH:/usr/lib/nis
export PATH
```

3. If the host will be a root replica server, set it up as a client in the root domain. If the host will be a non-root replica server, set it up as a client in the parent domain of the domain it will serve.
4. Type the following command if the master server is not running in NIS compatibility mode:

```
rpc.nisd
```

Type the following command if the master server is running in NIS compatibility mode:

```
rpc.nisd -Y
```

If the master server is running in NIS compatibility mode, its replica servers must also run in NIS compatibility mode.

5. Set the `nisplus_server` variable to 1 in the `/etc/rc.config.d/namesvrs` file:

```
nisplus_server=1
```

If the host was previously a NIS server or client, set the `nis_master_server`, `nis_slave_server`, and `nis_client` variables to 0.

6. Log in as root to the master server.
7. Type the following command to initialize the replica server:

```
nisserver -R -h replica_host_name
```

The `nisserver` script asks you if the information it has is correct. You can change it by typing `n`. The script then allows you to change each piece of information. To make a change, just type the correct information after the incorrect information and press `[Return]`.

8. Type the following command on the master server to checkpoint the domain and copy the domain directories to the new replica server:

```
nisping -a
```

9. To verify that the replica server is operating correctly, issue the following command:

```
nisstat -H replica_host_name
```

The **nisstat** command should return a list of statistics about the replica server.

It is recommended that you have only a few replicas per domain, for performance reasons. Do not configure more than 10 replicas per domain. If your domain includes sites that are distant from the master server, configure replica servers at the distant sites to avoid unnecessary network traffic. For more information, see the following man pages: **nissserver(1M)**, **rpc.nisd(1M)**, **nisping(1M)**, **nisstat(1M)**, and **nis(1)**.

---

## 6-16. Initializing NIS+ Client Users

### Initializing NIS+ Client Users

1. Log into any NIS+ client as an ordinary user.
2. Update your credential with **/usr/lib/nis/nisclient -u**.  
This will permit you to change your password from the default (**nisplus**) to be the same as your login password (which you must type in when prompted).
3. Use **nispasswd** for future password changes; it changes both the secure RPC password used by NIS+ and your login password.

a64651

### Student Notes

Tell all of your non-root users to perform this task. This task sets a user's secure RPC password to be the same as the user's login password.

- Log into any NIS+ client host using your non-root user login.
- Issue the following command:

```
/usr/lib/nis/nisclient -u
```

- The **nisclient** script will prompt you for the secure RPC password. Type the default password, **nisplus**. The **nisclient** script will then prompt you for your login password. After you type your login password, the **nisclient** script will change your **cred** table entry to use your login password instead of the default password.

To change your password in the future, use the **nispasswd** command, which changes your login password and your secure RPC password at the same time.

For more information, see the following man pages: **nisclient(1M)**, **nispasswd(1)**, and **nis(1)**.

---

## 6-17. Setting Up a NIS+ Subdomain

### Setting Up a NIS+ Subdomain

1. Login as **root** to the host that will be master server for the subdomain.
2. Follow steps 2-5 in “Setting Up the NIS+ Replica Servers.”
3. Log into the master server for the parent domain of the new subdomain and initialize the new master server with **nisserver** and the appropriate options and arguments.
4. Log into the new master as root and populate its tables from files or from NIS maps.
5. Verify the creation of the new subdomain with **nisls -lR subdomainname**.
6. Run **nisping -Ca** once daily.
7. Set up clients, replica servers and client users as necessary.

a64652

### Student Notes

Before you can set up a subdomain, the parent domain must be set up, and its master server must be running. The master server for the parent domain must have an entry in its hosts table for the master server of the new subdomain.

1. Log in as root to the host that will be the master server for the subdomain.
2. Set the PATH variable to include **/usr/lib/nis**. If you are running the C shell, type the following command:

```
setenv PATH $PATH:/usr/lib/nis
```

If you are running the Bourne or Korn shell, type the following commands:

```
PATH=$PATH:/usr/lib/nis
```

```
export PATH
```

3. Set up the host as a client in the parent domain. For example, if the root domain is **Wiz.Com.**, and you are setting up a subdomain called **Eng.Wiz.Com.**, make the host a client in the **Wiz.Com.domain**.
4. Type the following command if the new master server will not run in NIS compatibility mode:

```
rpc.nisd
```

If the new master server will be required to serve NIS clients, type the following command to run the server in NIS compatibility mode.

```
rpc.nisd -Y
```

5. Set the *nisplus\_server* variable to 1 in the **/etc/rc.config.d/namesvrs** file:

```
NISPLUS_SERVER=1
```

If the host was previously a NIS server or client, set the *nis\_master\_server*, and *nis\_client* variables to 0.

6. Log in as root to the master server for the parent domain of the new subdomain. For example, if the new subdomain is called **Eng.Wiz.Com.**, log in as root to the master server for the **Wiz.Com.domain**.
7. Issue the following command if the master server of the new subdomain will not run in NIS compatibility mode:

```
nisserver -M -d subdomain_name -h subdomain_master_server_name
```

If the master server of the new subdomain is required to serve NIS clients, issue the following command to set up the master server in NIS compatibility mode:

```
nisserver -M -Y -d subdomain_name -h subdomain_master_server_name
```

8. Log in as root to the master server of the new subdomain.
9. Populate the new master server's tables from files or from NIS maps. Be sure to specify the **-d domainname** option in the **nispopulate** command.
10. To verify that the subdomain was created successfully, issue the following command:

```
nisls -lR subdomain_name
```

The **nisls** command should list the subdomain name, the **org\_dir** and **groups\_dir** NIS+ directories.

11. Create a **cron** job that runs **nisping -Ca** at least once a day, during a time when the network is not busy. The following example crontab file runs **nisping -Ca** once a day, at 3:00 AM. It directs standard output and standard error from the **nisping** command to the file **/tmp/nisping.log**.

```
0 3 * * * /usr/lib/nis/nisping -Ca > /tmp/nisping.log 2& >1
```

The **nisping -Ca** command causes all servers of the domain to update their tables with the changes in the transaction log and to clear the transaction log. If you do not issue the **nisping -Ca** command regularly, your transaction log may grow too large, and you may not have enough disk space to checkpoint it.

12. Initialize the clients of the new subdomain.
13. Set up one or more replica servers for the new subdomain.
14. Initialize the client users of the new subdomain.

Every time you create a master server, you create a new subdomain. You can create as many domains as you need. You can create subdomains beneath subdomains. It is recommended that you keep your name space hierarchy as simple as possible and that you keep an accurate map of your name space. For more information, see the following man pages: **nisserver(1M)**, **rpc.nisd(1M)**, **nispopulate(1M)**, **nisclient(1M)**, and **nis(1)**.

---

## 6-18. Using BIND With NIS+

### Using BIND with NIS+

- Use the name service switch to define BIND (DNS) as the hostname resolution service to do the following:
  - Configure the NIS+ client to query BIND.
  - Configure the NIS+ server to return BIND information to clients.

a64653

### Student Notes

A NIS+ client can consult BIND (DNS), NIS, NIS+, or the `/etc/hosts` file when it needs to resolve a host name to an IP address. The name service switch determines where an NIS+ client will look for host information.

Some clients, like PCs, cannot use the name service switch. If your domain includes PC clients, you can configure the NIS+ server to query BIND when its NIS+ hosts table does not contain the information that a client requests. The server then returns the information to the client through NIS+.

Only NIS+ servers running in NIS compatibility mode may consult BIND to answer client queries. This section tells you how to perform the following tasks:

- Configure a NIS+ client to query BIND.
- Configure a NIS+ server to return BIND information to clients.

If your NIS+ clients support the Name Service Switch, configure the clients to query BIND. If your NIS+ clients do not support the name service switch, configure their server to return BIND information.

---

**NOTE:** BIND must already be configured and running before you can configure a NIS+ client or server to query it.

---

## To Configure a NIS+ Client to Query BIND

1. Log in as root to the NIS+ client host.
2. Use a text editor to open the `/etc/nsswitch.conf` file, and find the line that begins with `hosts`. It probably looks something like this:

```
host: nisplus [NOTFOUND=return] files
```

Change the `hosts` line so that it looks like this:

```
hosts: dns [NOTFOUND=continue] nisplus [NOTFOUND=continue] files
```

The NIS+ client will now query BIND first for host information. If the BIND server is down or unreachable, it will query NIS+. Then, if no NIS+ server is available, it will consult its local `/etc/hosts` file. For more information, type `man 4 nsswitch.conf` at the HP-UX prompt.

## To Configure a NIS+ Server to Return BIND Information to Clients

Only servers running in NIS compatibility mode may return BIND information to clients through NIS+.

- Log in as root to the NIS+ server.
- In the `/etc/rc.config.d/namesvrs` file, set the `EMULYP` variable to `"-Y -B"`, as follows:

```
EMULYP='`-Y -B'`
```

- Kill the `rpc.nisd` daemon, and restart it with the `-Y` and `-B` options:

```
ps -ef | grep rpc.nisd
kill PID
rpc.nisd -Y -B
```

For more information, type `man 1M rpc.nisd` at the HP-UX prompt.

---

## 6-19. Allowing a NIS+ User Authenticated Access to Another Domain

---

### Allowing a NIS+ User Authenticated Access to Another Domain

---

1. Use **nismatch** and **nisaddent** to copy the user's **passwd** table entry from home domain to the other domain.
2. Check that the user ID in the **passwd** table in the other domain is unique; change it if necessary.
3. Use **nisaddcred** to create a new credential for the user in the other domain.

a64654

### Student Notes

A user's home domain is defined as the domain where the user has a DES credential in the **cred** table. (Each NIS+ principal has a DES credential in only one domain.) If a user needs to be authenticated in another domain, the user must have a Local credential in that domain. In domains where the user does not have a Local credential, the user is treated as **nobody**.

1. From any NIS+ client host, issue the following commands to copy the **passwd** table entry from the user's home domain to the remote domain where the user needs authenticated access:

```
nismatch name=username passwd.org_dir.user's_homedomain \
> tempfile
nisaddent -a -f tempfile passwd remote_domainname
```

2. If necessary, change the user ID in the entry to ensure that it is unique in the **passwd** table of the remote domain. Each user ID may occur only once in a **passwd** table. See To Modify an Entry in a NIS+ Table.

3. From any NIS+ client host, issue the following command:

```
nisaddcred -p UID -P loginname.domainname local remote_domainname
```

The argument following the **-p** option is the user's user ID from the NIS+ **passwd** table in the remote domain where the user needs authenticated access. The argument following the **-P** option is the user's NIS+ principal name and must end with a period.

The *remote\_domainname* argument is the domain where the credential will be created (the domain where the user needs authenticated access).

The following example allows NIS+ principal **sara.Eng.Wiz.Com** to be authenticated in domain **Sales.Wiz.Com**:

```
nisaddcred -p 7899 -P sara.Eng.Wiz.Com. local Sales.Wiz.Com.
```

You must have create permission for the **cred** table and the **passwd** table in the remote domain in order to complete this task.

For more information, see the following man pages: **nisaddcred(1M)**, **nismatch(1)**, and **nisaddent(1M)**.

---

## 6-20. Managing NIS+ Objects and Their Properties

---

### Managing NIS+ Objects and Their Properties

---

1. List an object's properties with **niscat**.
2. Change the default properties of a new object by modifying the **NIS\_DEFAULTS** environment variable.
3. Change an object's permissions with **nischmod**.
4. Change an object's ownership with **nischown**.

a64655

## Student Notes

### To List Properties of NIS+ Objects

To list the object properties of any NIS+ directory, table, table entry, group, or link, issue the following command from a NIS+ client host:

```
niscat -o NIS+_object
```

For example, to list the object properties of the `passwd` table entry for user `jane` in the default domain, you would issue this command:

```
niscat -o '[name=jane],passwd.org_dir'
```

The `niscat -o` command gives you information about the object, including its owner, group owner, and permissions. If the NIS+ object is a table, the `niscat -o` command gives the number of columns in the table, the names of the columns, and the permissions for each column. For more information, type `man niscat(1)` at the HP-UX prompt.

## To Change the Default Properties for New NIS+ Objects

Whenever you create a new NIS+ object (a directory, table, table entry, group, or link), it inherits a set of default properties (owner, group owner, permissions, time to live, and so on). You can override the default object properties by setting the `nis_defaults` environment variable. For more information, type `man sam(1M)`.

1. Issue the `nisdefaults` command to find out the current default values:

```
nisdefaults
```

2. If you are using the Korn or Bourne shell, issue the following command:

```
NIS_DEFAULTS=access=perms:owner=owner:group=group:ttl=time
export NIS_DEFAULTS
```

If you are using the C shell, issue the following command:

```
access=perms:owner=owner:group=group:ttl=time
```

You do not have to specify all four values. For example, you could change just the default owner and group owner, as in the following example:

```
setenv NIS_DEFAULTS owner=garlic.Eng.Wiz.Com.:group=admin.Eng.Wiz.Com.
```

You can also set the default group owner by setting the `nis_group` environment variable, but if the `nis_defaults` variable specifies a default group owner, it overrides the `nis_group` variable.

The time to live (`ttl`) applies only to NIS+ directories and groups. It tells NIS+ clients when to purge the information in their caches and get new information from a server. (To change the `ttl` value for an existing NIS+ object, use the `nischttl(1)` command.) For more information, see the following man pages: `nisdefaults(1)`, `nischttl(1)`, `sam(1M)`, and `nis(1)`.

## To Change the Permissions for NIS Objects

To change the permissions of a NIS+ directory, table, table entry, group, or link, issue the `nischmod` command from a NIS+ client host.

The following example changes the permissions for the group table in the **Wiz.Com.domain**. It gives user **nobody** no permissions, owner and group owner full permissions, and world read permission only.

```
nischmod n=,og=rmcd,w=r group.org_dir.Wiz.Com.
```

The following example gives user **nobody** read permission for the *groups\_dir* directory in the default domain and takes away modify, create, and destroy permission from the group owner:

```
nischmod n+r,g-mcd groups_dir
```

To change permissions for a table column, use the **nistbladm -u** command.

The following example changes the permissions on the *passwd* column of the **passwd** table in the default domain. It gives **nobody**, group, and world no permissions and takes away create and destroy permissions from the owner.

```
nistbladm -u passwd=ngw=,o-cd passwd.org_dir
```

In order to change the permissions for a NIS+ object, you need modify permission for that object. The actual permissions for an entry or column are the entry or column permissions plus the permissions for the table. For example, if the **passwd** table has permissions **---rmcdrmcd---**, and the **passwd** column of the table has permissions **r-----**, the actual permissions for the **passwd** column are **r---rmcdrmcd---**.

---

**NOTE:** The **cred** table must allow read permission to user **nobody** in order for NIS+ to start up.

---

The **cred** table must allow read permission to user **nobody** in order for NIS+ to start up. For more information, see the following man pages: **nischmod(1)**, **nistbladm(1)**, **sam(1M)**, and **nis(1)**.

## To Change the Ownership of NIS+ Objects

To change the owner of a NIS+ directory, table, table entry, group, or link, issue the **nischown** command from a NIS+ client host. The following example changes the owner of the **passwd** table entry for user **sid** to **sid.Sales.Wiz.Com.**:

```
nischown sid.Sales.Wiz.Com. '[name=sid],passwd.org_dir'
```

The following example makes **sid.Sales.Wiz.Com.** the owner of his own **cred** table entries:

```
nischown sid '[cname=sid.Sales.Wiz.Com.],cred.org_dir'
```

In this example, the owner **sid** is not a fully qualified NIS+ principal name. NIS+ will append the default domain to **sid** when it processes the command. The **cred** table contains two entries for **sid.Sales.Wiz.Com.**: a Local credential and a **DES** credential. The command in this example will change the ownership of both entries, because both entries have the same value in the **cname** column. To change the group owner of a NIS+ directory, table, table entry, group, or link, issues the **nischgrp** command. The following example changes the group owner of the **Sales.Wiz.Com.** directory to **admin.Sales.Wiz.Com.**:

```
nischgrp admin.Sales.Wiz.Com. Sales.Wiz.Com.
```

The following example changes the group owner of all the entries in the hosts table to the **admin** group in the default domain:

```
nischgrp admin '[]hosts.org_dir'
```

To change the ownership of a NIS+ object, you need modify permission for the object. You cannot change the owner or group owner of a table column, because it is always the same as the owner and group owner of the table. For more information, see the following man pages: **nischown(1)**, **nischgrp(1)**, and **nis(1)**.

## 6-21. Managing NIS+ Tables

### Managing NIS+ Tables

- Search a NIS+ table with **nisgrep** and **nismatch**.
- Add an entry to a NIS+ table with **nistbladm -a** or **nisadddent**.
- Remove an entry from a NIS+ table with **nistbladm -r**.
- Modify an entry to a NIS+ table with **nistbladm**.
  - You also can dump the entire table, edit it, and then reload it.
  - You also may use SAM.

a64656

## Student Notes

### To Search a NIS+ Table

Issue one of the following commands from any NIS+ client host:

```
nisgrep 'column_name=regular_expression' tablename
nismatch column_name=text_string tablename
```

For example, the following command returns all the entries from users in the **passwd** table whose home directories are under **/users**: **nisgrep 'home=/users/\*' passwd.org\_dir**

If you do not specify a column name, the first column of the table is searched. The following command returns the **Local** and **DES** credentials for NIS+ principal **liz.Eng.Wiz.Com.** from the **cred** table:

```
nismatch liz.Eng.Wiz.Com. cred.org_dir
```

The **nismatch** command can search only columns that were defined as searchable when the table was created. The **nisgrep** command can search any column in a table. To get the name of a column, or to determine whether a column is searchable, issue the following command:

```
niscat -o tablename.org_dir
```

The **nisgrep** command can search on regular expressions, but the **nismatch** command can search only for exact matches of text strings. The **nisgrep** command is slower than the **nismatch** command. You must have read permission on the table or the entries you are searching for, or NIS+ will not display the entries. For more information, see the following man page: **nismatch(1)**.

## To Add an Entry to a NIS+ Table

1. Issue the following command from any NIS+ client host:

```
nistbladm -a column_name=value column_name=value ... tablename
```

The following example adds an entry to the hosts table:

```
nistbladm -a cname=romney name=romney.Eng.Wiz.Com \
addr=15.14.13.12 comment="acb, pillar R4" hosts.org_dir
```

2. Issue the following command to make sure the entry was added successfully:

```
nismatch column_name=value tablename
```

The following example searches the hosts table for the entry for host **romney**:

```
nismatch cname=romney hosts.org_dir
```

If the entry exists, and if you have read access to it, the **nismatch** command will return the entry.

In the **nistbladm -a** command, you must specify the value for every column. To leave a column blank, specify no value after the equal sign (=). The following example adds an entry to the group table without specifying a password:

```
nistbladm -a name=staff passwd= gid=10 members=root group.org_dir
```

To get the names of the columns in a table, issue the following command:

```
niscat -o tablename.org_dir
```

You must have create permission for the table in order to add an entry to it.

For more information, see the following man pages: **nistbladm(1)**, and **niscat(1)**.

## To Remove an Entry from a NIS+ Table

Issue the following command from any NIS+ client host:

```
nistbladm -r column_name=value column_name=value ... tablename
```

The following example removes an entry from the hosts table:

```
nistbladm -r cname=romney addr=15.14.13.12 hosts.org_dir
```

In the **nistbladm -r** command, specify as many column values as you need to identify a single entry. If the criteria you specify identify more than one entry, NIS+ displays an error. If you want to remove all entries matching a set of criteria, use the **-R** option instead of the **-r** option. The following example removes both the **Local** and **DES** credentials for principal **liz.Eng.Wiz.Com.** from the **cred** table:

```
nistbladm -R cname=liz.Eng.Wiz.Com. cred.org_dir
```

To get the names of the columns in a table, issue the following command:

```
niscat -o tablename.org_dir
```

You must have destroy permission for the table or for the entries you want to remove. For more information, see the following man pages: **nistbladm(1)**, and **niscat(1)**.

## To Modify an Entry in a NIS+ Table

You can use either of two methods to modify a table entry:

- You can use **nistbladm(1)** to modify the entry directly.
- You can use **nisadddent(1M)** to dump the table to a file, and you can modify the file. Then, you can use **nisadddent** to update the NIS+ table from the file.

---

## 6-22. User, Group, Host and Domain Management in NIS+

---

### User, Group, Host and Domain Management in NIS+

---

- Add hosts, groups, and users to a NIS+ domain with **nistbladm**, **nisaddcred**, and **nisgrpadm**. Adding users also requires use of **nisadddent** and **passwd**.
- Create or remove a NIS+ group with **nisgrpadm**.
- Add members to or remove them from a NIS group with **nisgrpadm**.

a64657

## Student Notes

### To Add a Host to a NIS+ Domain

1. Issue the following command, from any NIS+ client host, to add the new host to the NIS+ hosts table:

```
nistbladm -a cname=hostname name=hostname addr=IPaddress \
comment=comment hosts.org_dir.domainname
```

You must have create permission for the **hosts** table to use this command.

You must create one **hosts** table entry in which the *cname* and *name* columns are both set to the official host name. If you wish to configure aliases for the host name, create entries in which the *cname* column contains the official host name and the *name* column contains the alias.

If the domain is the default domain, you do not have to specify the domain name, as in the following example:

```
nistbladm -a cname=romney.Eng.Wiz.Com name=romney.Eng.Wiz.Com \
addr=15.14.13.12 comment= hosts.org_dir
```

**2. Issue the following command to add a DES credential for the new host to the **cred** table:**

```
nisaddcred -p unix.hostname@domainname -P hostname.domainname \
des domainname
```

If you do not specify the domain name as the last argument, the credential is created in the default domain, as in the following example:

```
nisaddcred -p unix.romney@Eng.Wiz.Com -P romney.Wiz.Com. des
```

The argument following the **-p** option is the host's secure RPC netname and does not end with a period. The argument following the **-P** option is the host's NIS+ principal name and must end with a period.

When NIS+ prompts you for a password, enter the root password of the new host.

You must have create permission for the **cred** table to use this command.

**3. If you want to allow the root user on this host to administer the NIS+ domain, add the host to the domain's admin group. Issue this command:**

```
nisgrpadm -a hostname.domainname admin_groupname.domainname
```

The admin group for most domains is called **admin**, as in the following example:

```
nisgrpadm -a romney.Eng.Wiz.Com. admin.Eng.Wiz.Com.
```

You must have modify permission for the admin group in order to add members to it.

**4. Set up the host as a client of the NIS+ domain to which you just added the host's data and credentials.**

For more information, see the following man pages: **nistbladm(1)**, **nisaddcred(1M)**, **nisgrpadm(1)**, and **nis(1)**.

## To Create or Remove a NIS+ Group

To create a NIS+ group, type the following command on any NIS+ client host:

```
nisgrpadm -c groupname
```

The following example creates a NIS+ group called engineers in the **Sales.Wiz.Com.domain**:

```
nisgrpadm -c engineers.Sales.Wiz.Com.
```

To remove a NIS+ group, type the following command on any NIS+ client host:

```
nisgrpadm -d groupname
```

The following example removes the NIS+ group called **engineers** from the **Sales.Wiz.Com.domain**:

```
nisgrpadm -d engineers.Sales.Wiz.Com.
```

NIS+ groups are not the same as the HP-UX groups stored in the **group.org\_dir** table or the **/etc/group** file. NIS+ groups are used to determine group ownership of NIS+ objects. NIS+ objects allow certain access permissions to their group owners. NIS+ groups are stored in the **groups\_dir** subdirectory of the domain directory. To create a NIS+ group, you must have create permission for the **groups\_dir** directory. To remove a group, you must destroy permission for group or for the **groups\_dir** directory. For more information, see the following man page: **nisgrpadm(1)**.

## To Add or Remove Members of a NIS+ Group

- To add members to a NIS+ group, type the following command on any NIS+ client host:

```
nisgrpadm -a groupname group_member [group_member...]
```

The following example adds the host principal **thyme.Wiz.Com.** and the NIS+ group **tempadmin.Wiz.Com.** to the group **admin.Wiz.Com.**:

```
nisgrpadm -a admin.Wiz.Com. thyme.Wiz.Com. @tempadmin.Wiz.Com.
```

- To remove members from a NIS+ group, type the following command on any NIS+ client host:

```
nisgrpadm -r groupname group_member [group_member...]
```

The following example removes the user principal **amy.Wiz.Com.** and all principals in the **Eng.Wiz.Com.** domain from the group **admin.Wiz.Com.**:

```
nisgrpadm -r admin.Wiz.Com. amy.Wiz.Com. *.Eng.Wiz.Com.
```

- To list the current members of a NIS+ group, type the following command on any NIS+ client host:

```
nisgrpadm -l groupname
```

A NIS+ group member may take any of the following forms:

|                       |                                                                             |
|-----------------------|-----------------------------------------------------------------------------|
| principal             | Any host or user principal (for example, <code>amy.Wiz.Com.</code> )        |
| @group                | Another NIS+ group (for example, <code>@tempadmin.Wiz.Com.</code> )         |
| <code>*.domain</code> | All principals in a NIS+ domain (for example, <code>*.Eng.Wiz.Com.</code> ) |

## To Add A User to a NIS+ Domain

- Issue the following command, from any NIS+ client host, to add the new user to the NIS+ `passwd` table:

```
nistbladm -a name=loginname passwd= uid=userID gid=groupID \
gcos=user_info home=home_dir shell=shell shadow= \
passwd.org_dir.domainname
```

You must have create permission for the `passwd` table to use this command.

If the domain is the default domain, you do not have to specify the domain name, as in the following example:

```
nistbladm -a name=sara passwd= uid=7899 gid=20 \
gcos="Sara Sena,,x77555," home=/home/sara shell=/bin/ksh \
shadow= passwd.org_dir
```

- Issue the following commands to add `Local` and `DES` credentials for the new user to the `cred` table:

```
nisaddcred -p UID -P loginname.domainname local domainname
nisaddcred -p unix.UID@domainname -P loginname.domainname des \
domainname
```

If you do not specify the domain name as the last argument, the credentials are created in the default domain, as in the following example:

```
nisaddcred -p 7899 -P sara.Eng.Wiz.Com. local
nisaddcred -p unix.7899@Eng.Wiz.Com -P sara.Eng.Wiz.Com. des
```

The user ID must not belong to any other user in the `passwd` table. The argument following the `-P` option is the user's NIS+ principal name and must end with a period.

When the **nisaddcred** command prompts you for a password, enter a temporary password for the user.

You must create permission for the **cred** table to use this command.

- Issue the following command to change the user's password:

```
passwd -r nisplus loginname
```

When the **nispasswd** command prompts you for a password, type the same password you typed when you created the user's DES credential in step 2.

You can ignore the message that tells you what to do if the user's login password is different from the user's secure RPC password. If you followed the steps in this section, the user's two passwords are the same.

- Issue the following command to make the user the owner of the user's **passwd** table entry:

```
nischown username.domainname '[name=username],passwd.org_dir.domainname'
```

The following example changes the ownership of a **passwd** table entry in the default domain:

```
nischown sara.Eng.Wiz.Com. '[name=sara],passwd.org_dir'
```

- Add the user to the primary group you specified when you added the user to the **passwd** table.

- Issue the following command to dump the current NIS+ group table to a file:

```
nisaddent -d group > filename
```

- Use a text editor to add the new user to the appropriate group in filename.

- Issue the following command to merge the contents of the temporary file into the NIS+ group table:

```
nisaddent -m -f filename group
```

You must have modify permission for the group table to add a user to a group.

- Create the user's home directory, and make the user the owner of it, as in the following example:

```
mkdir /export/home/sara
chown sara /export/home/sara
```

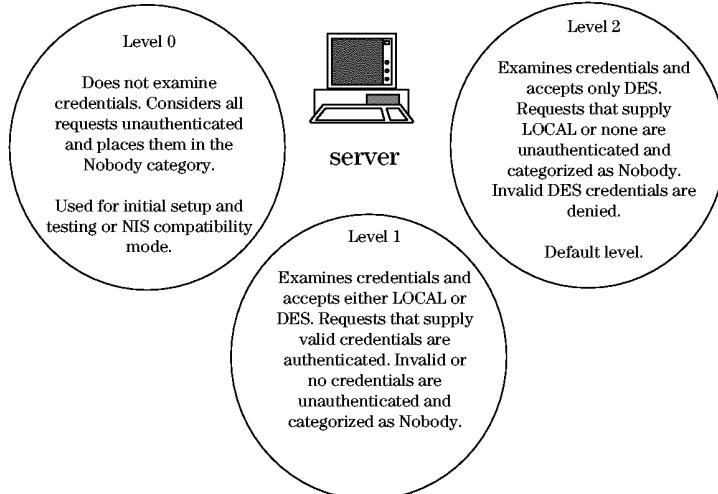
- If you are using the automounter to mount users' home directories, add the new user's home directory to the **auto\_home** table.
- Tell the new user to log in with the password you specified in steps 2 and 3 and change passwords with the **nispasswd** command.

For more information, see the following man pages: **nistbladm(1)**, **nisaddcred(1M)**, **passwd(1)**, **nisaddent(1)**, **sam(1M)**, and **nis(1)**.

## 6-23. Security Management in NIS+

### Security Management in NIS+

NIS+ servers can be run in three different security levels:



a64658

### Student Notes

An NIS+ server can operate at any one of three security levels. At each level, different access rights are granted to principals, depending on the types of credentials they submit in order to have their service requests authenticated and granted.

### Access Rights

There are four access rights:

|                |                                            |
|----------------|--------------------------------------------|
| <b>read</b>    | read the contents of an object             |
| <b>modify</b>  | modify the contents of an object           |
| <b>destroy</b> | destroy an object in a table or directory  |
| <b>create</b>  | create new objects in a table or directory |

## Authorization Categories

There are four classes of authorization:

|               |                                                                                                                              |
|---------------|------------------------------------------------------------------------------------------------------------------------------|
| <b>Owner</b>  | The owner of the object.                                                                                                     |
| <b>Ggroup</b> | The NIS+ group to which the object belongs.                                                                                  |
| <b>world</b>  | All NIS+ principals who are authenticated by NIS+.                                                                           |
| <b>nobody</b> | Requests made without credentials. It does not include requests made under invalid credentials; these are denied altogether. |

## Credentials

Credentials may be either **LOCAL** or **DES**.

- A **LOCAL** credential is the user ID (**uid**) of a NIS+ principal. It may be stored in any domain; in fact, in order to be able to log into domains other than their home domains, the **cred** tables of the remote domains must contain the **LOCAL** credentials of such users.
- A **DES** credential is a securely encrypted *key* to access the actual credential information itself, which is stored in the server's **cred** table. The **DES** credential is structured in such a way as to ensure that the credential cannot be forged.

## Security Levels

These credentials are used in different ways at each of the three security levels:

- |   |                                                                                                                                                                                                                                                                                                                                     |
|---|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0 | At this level, security is minimal. Any principal, even without a credential, will be granted full access rights. Level 0 is intended primarily for use in testing and initial set-up of the NIS+ name space.                                                                                                                       |
| 1 | Used for testing and debugging, this level will authenticate a request made under either credential. It is not to be used on networks which may be accessed by untrusted clients.                                                                                                                                                   |
| 2 | This is the default setting, and it is the highest level of security currently available under NIS+. Only those requests submitted under valid <b>DES</b> credentials are authenticated. Requests made under <b>LOCAL</b> credentials (or no credentials at all) are assigned the access rights granted to the <b>nobody</b> class. |

---

**NOTE:** NIS+ security must be set up when the name space is first established. It cannot be added later.

---

## **6-24. LAB: Administering NIS+**

## **Directions**

The instructor will organize the students into groups of three, since three systems will be needed for this lab. Each group will have a domain called `class#.hp`, where the instructor will assign the group a number (the lab instructions assume number 3). The three systems are:

- |       |                                                                                          |
|-------|------------------------------------------------------------------------------------------|
| hostA | The Root Master Server for the root domain ( <b>class3.hp</b> )                          |
| hostB | A client for domain <b>class3.hp</b> , and the server for subdomain <b>sub.class3.hp</b> |
| hostC | A client for subdomain <b>sub.class3.hp</b>                                              |

# **Setup Root Master Server**

1. Delete references to **hostC** and **hostA** in **/etc/hosts** on **hostA** and **hostB**, respectively.
  2. Set the **domainname** for the NIS+ domain.
  3. Configure the **PATH** variable to contain the NIS+ directory.
  4. Initialize the NIS+ root server.
  5. Edit startup configuration file.

## **Populate NIS+ Tables on Master Server**

1. Make a temporary directory for the NIS+ files.
2. Change directory to `/etc` (location of files to be used for NIS+).
3. Copy files to the `/nis+files` directory.
4. Prepare the files for conversion to NIS+ tables.
5. Populate the NIS+ tables based on files (`-F` option) in the `/nis+files` directory.
6. Test access to NIS+ tables.
7. Perform a checkpoint (`-C` option) for all (`-a` option) NIS+ tables.

## Add Administrator to NIS+ admin Group

1. All users that will be allowed to modify the NIS+ tables (and their entries) must be added to the NIS+ `admin` group. Add appropriate users to the `admin` group (assume `user1` is to be added).

## Setup a NIS+ Client

1. Verify that the client system is recognized as an authorized client by the server.
2. If the client is not authorized, then arrange for its proper authorization.
3. From the client, set the `domainname`.
4. Configure the PATH variable to contain the NIS+ directory.
5. Initialize the NIS+ client.

6. Verify client access to NIS+ server.
7. Verify client access to NIS+ tables.
8. Edit client startup configuration file.

## **Setup a NIS+ Subdomain**

1. On the subdomain system (`hostB`), start the NIS+ RPC daemon.
2. Login to the master server for the parent domain (in this case the `root` domain).
3. Initialize the NIS+ subdomain server.

## **Populate Table for the NIS+ Subdomain**

1. Exit from master of parent domain (`hostA`) back to the master of the subdomain (`hostB`).

2. Make a temporary directory for the NIS+ files on the subdomain server (`hostB`).
3. Copy files to the `/nis+files` directory.
4. Prepare the files for conversion to NIS+ tables.
5. Populate the NIS+ tables based on files (-F option) in the `/nis+files` directory.
6. Verify subdomain was created successfully.
7. Verify client access to NIS+ tables.
8. Edit client startup configuration file.

## Setup a NIS+ Client for Subdomain

1. Verify that the client system is recognized as an authorized client by the subdomain server.
2. From the client (`hostC`), set the `domainname` and the `PATH` variable.
3. Initialize the NIS+ client.
4. Verify client access to NIS+ server and to NIS+ tables.
5. Edit client startup configuration file.

---

# Module 7 — Network and NFS Performance

## Objectives

Upon completion of this module you will be able to do the following:

- Isolate and characterize network bottlenecks.
- Isolate and characterize NFS bottlenecks.

## 7-1. SLIDE: Network Performance

### Network Performance

- Modern reliance on networks means network performance is an increasingly important component of overall performance.
- This is largely due to the growth of popular network-loading applications:
  - NFS
  - X/Windows
  - Diskless clients

a646203

### Student Notes

Network performance has become an important issue for modern network administrators because of the growing reliance on distributed applications and services such as NFS and X/Windows. Virtually every modern UNIX® system network utilizes such services, and the delays that may be experienced by network users due to overloaded or badly-tuned networks are the source of much frustration.

## 7-2. SLIDE: Protocol Overhead and Efficiency

### Protocol Overhead and Efficiency

| Protocol        | Overhead<br>in bytes | Max frame<br>size in bytes | Description                                                                                                               |
|-----------------|----------------------|----------------------------|---------------------------------------------------------------------------------------------------------------------------|
| IEEE 802.3      | 21                   | 1,518                      | 6 source, 6 destination 2 length, 3 LLC, 4 CRC.                                                                           |
| FDDI (RFC 1390) | 28                   | 4,500                      | 3 802.2 LLC, 5 802.2 SNAP, 4 source, 4 destination, 4 misc., 8 FCS, The source and destination addresses may be 12 bytes. |
| ATM (RFC 1483)  | 8                    | 53                         | 4 CRC, 2 length, 1 CPI, 1 CPCS-UU. For AAL-5 multiprotocol encapsulation. The ATM cell header is only 5 bytes.            |
| TCP (RFC 793)   | 20                   | 65,535                     | 8 sequence numbers, 4 ports, 2 window, 6 other. RFC 1106 extends the window size to over one gigabyte.                    |
| IP (RFC 791)    | 20                   | 65,535                     | 8 IP addresses, 12 other.                                                                                                 |

a646107

### Student Notes

There are two major overhead components which, while necessary for the correct operation of a network, can reduce that network's efficiency:

- the headers required to satisfy the operational requirements at each layer of the OSI standard, and
- the fragmentation and reassembly of packets necessitated by different MTU (maximum transmission unit) limits across various network segments

Generally speaking, network efficiency drops as the data volume transmitted per packet decreases relative to the growth in header content or the level of fragmentation required, either by the path itself or the nature of the application.

An infamous example is the *telnet* virtual-terminal application, which transmits just one user keystroke per packet and requires an echo of that keystroke before further keystrokes can be transmitted. Also, the echoed character is acknowledged in yet another packet. Even a simple

TELNET session running on a LAN without WAN overhead can cause 192 bytes to be transmitted for each keystroke.

Generally, this is not a problem for *telnet* users, since human typists cannot generate the levels of data that are typical of file exchanges and other high-traffic applications, and humans are willing to accept modest levels of latency in response which would be intolerable for an application with near realtime requirements (voice or video, for example).

### 7-3. SLIDE: Fragmentation of IP Datagrams

## Fragmentation of IP Datagrams

- packet size too small
  - wasting of network resources
  - inefficient throughput
- optimal throughput reached
  - largest size without fragmentation
  - = minimum of the MTU of each hop

a646108

## Student Notes

Fragmentation and reassembly are performed by IP when a datagram traverses several intermediate networks with differing maximum transmission units (MTU). For example, the MTU for Ethernet is 1500 bytes; for Token Bus, 8192; and for serial line IP (SLIP) it may be as small as 128 bytes. When a gateway forwards a datagram from one interface with a larger MTU than the outgoing network MTU, IP on the gateway fragments the incoming datagram into smaller fragments which will fit the MTU of the outgoing network. Reassembly is performed on the destination machine, not on any intermediate gateways. The destination machine starts a reassembly timer when it gets the first fragment; if the timer expires before all fragments are received, the surviving fragments are discarded.

IP fragmentation can have a large negative effect on network performance for several reasons:

- IP fragments are not optimally filled.
- If a single IP fragment is lost, all other IP fragments have to be retransmitted.

- The risk of losing just one of the fragments requires the sending transport to retransmit the entire data.
- The router needs time for fragmentation.
- Additional CPU overhead is required to fragment data.

For optimal throughput the MTU size should be as large as possible.

Fragmentation overhead occurs when the network protocol has a packet too large for the target medium. Think about how an application would send 9,000 bytes of data.

There are two reasons why you want to avoid fragmenting:

- to avoid the additional CPU overhead required to fragment and reassemble the packets
- the risk of losing just one of the fragments requires the sending transport to retransmit the entire datagram, not just the lost fragment

For example, protocols based on the OSI model add headers and trailers around their data. The OSI layer may have a different maximum transmission unit (MTU) requiring fragmentation and reassembly.

## 7-4. SLIDE: Using Path MTU Discovery

### Using PMTU Discovery

- PMTU discovery
  - determines the optimal MTU size
  - defined by RFC 1191
- set the **Don't Fragment** flag (DF)
  - causes an ICMP *Destination Unreachable* message
  - converges to the optimal MTU size

a646204

## Student Notes

For optimal throughput of data the MTU (packet) size should be as large as possible. However, the MTU size used by interfaces along any particular path cannot be known in advance.

Therefore a packet that is too large must be fragmented by routers. This leads to an additional CPU load on the router itself and split fragments, with several disadvantageous consequences:

- If one fragment gets lost, the whole datagram will have to be repeated.
- The last of the split fragments will not be of optimal size.

In a recursion this can lead to more inefficiently-sized small packets.

This is why the current practice is to use the lesser of

- 576
- owner's interface MTU

as the path maximum transmission unit (PMTU) for any destination outside the local network.

In many cases, this results in the use of smaller datagrams than necessary, because many paths have a PMTU greater than 576. A host sending datagrams much smaller than the PMTU allows is wasting network resources and probably getting suboptimal throughput.

The best throughput is reached when datagrams are of the largest size that does not require fragmentation anywhere along the path from the source to the destination. This datagram size is referred to as the PMTU, and it is equal to the minimum of the MTUs of each hop in the path. RFC 1191 was defined to determine the optimal MTU and is called PMTU Discovery.

PMTU discovery is a technique for determining the maximum transmission unit (MTU) of an arbitrary path dynamically without fragmentation.

Both the host and router implementations should include PMTU discover, but only the host implementation is necessary for proper operation.

The implementation in fact is very simple. A source host initially assumes that the PMTU of a path is the known MTU of its own interface. In the IP datagram, the **Don't Fragment (DF)** bit is set. If a datagram is dropped by a router because it is too big to be forwarded without fragmenting, the originating system will receive an ICMP *Destination Unreachable* message with a code meaning *fragmentation needed and DF set*.

A router compatible with RFC 1191 must include the MTU of that next-hop network in the low-order 16 bits of the ICMP header field that is labelled *unused* in the ICMP specification. The high-order 16 bits remain unused, and *must* be set to zero.

If a router message doesn't support RFC 1191, the host should invoke searching for an accurate PMTU estimate by continuing to send datagrams with the DF bit while varying their sizes.

RFC 1191 makes a series of suggestions regarding how to search for the best PMTU but doesn't dictate how to converge. In any case the algorithm must lead to a converging solution.

## 7-5. SLIDE: Switching PMTU Discovery

### Switching PMTU Discovery

- Display the current PMTU sizes in the routing table.

```
netstat -r
```

- PMTU strategies.

1 - Set the **Don't Fragment (DF)** bit on all outbound datagrams.

2 - Set the **DF** bit only under certain conditions (default).

0 - Disable PMTU Discovery.

- Change the PMTU strategy to Strategy 1.

```
ndd -set /dev/ip ip_pmtu_strategy 1
```

a646109

### Student Notes

You may examine the current PMTU sizes with **netstat -r**. If you feel that they are in need of examination or adjustment, the HP-UX command **ndd** controls the behavior of PMTU Discovery.

Because of problems encountered with some firewalls, hosts, and low-end routers, IP provides for selection of either of two discovery strategies, or for completely disabling the algorithm. The tunable parameter **ip\_pmtu\_strategy** controls the selection. To examine the current setting, enter this command:

```
ndd -get /dev/ip ip_pmtu_strategy
```

These are the strategic alternatives:

- Strategy 1** All outbound datagrams have the *Don't Fragment* bit set. This should result in notification from any intervening gateway that needs to forward a datagram down a path that would require additional fragmentation. When the ICMP *Fragment Needed* message is received, IP updates its MTU for the remote host. If the responding gateway does not provide next hop information, then IP will reduce the MTU to the next lower value taken from a table of popular media MTUs.
- Strategy 2** When a new routing table entry is created for a destination on a locally connected subnet, the *Don't Fragment* bit is never turned on. When a new routing table entry for a non-local destination is created, the *Don't Fragment* bit is not immediately turned on. Instead,
- An ICMP *Echo Request* of full MTU size is generated and sent out with the *Don't Fragment* bit on.
  - The datagram that initiated creation of the routing table entry is sent out immediately, without the *Don't Fragment* bit. Traffic is not held up waiting for a response to the *Echo Request*.
  - If no response to the *Echo Request* is received, the *Don't Fragment* bit is never turned on for that route; IP won't time-out or retry the `ping`. If an ICMP *Fragmentation Needed* message is received in response to the *Echo Request*, the Path MTU is reduced accordingly, and a new *Echo Request* is sent out using the updated Path MTU. This step repeats as needed.
  - If a response to the *Echo Request* is received, the *Don't Fragment* bit is turned on for all further packets for the destination, and Path MTU discovery proceeds as for Strategy 1.

Assuming that all routers properly implement Path MTU Discovery, Strategy 1 is generally better—there is no extra overhead for the ICMP *Echo Request* and response. Strategy 2 is available only because some routers, or firewalls, or end hosts have been observed simply to drop packets that have the DF bit on without issuing the *Fragmentation Needed* message. Strategy 2 (the system default) is more conservative in that IP will never fail to communicate when using it.

**Strategy 0** Disable PMTU Discovery.

To apply a different strategy (Strategy 1, for example), enter this command:

```
ndd -set /dev/ip ip_pmtu_strategy 1
```

## 7-6. SLIDE: Optimizing Memory Buffers

### Optimizing Memory Buffers

- determine the network memory load
  - maximum usable network memory:
  - dropped IP fragments:
  - TCP socket receive buffer size:
  - TCP flow control limits:
- tune network memory buffers
  - maximum usable network memory:
  - TCP socket receive buffer size:
  - TCP flow control limits:

```
ndd -get /dev/ip ip_reass_mem_limit
netstat -s -p ip
ndd -get /dev/tcp tcp_recv_hiwat_def
ndd -get /dev/tcp tcp_xmit_hiwat_def
ndd -get /dev/tcp tcp_xmit_lowat_def

ndd -set /dev/ip ip_reass_mem_limit 2000000
ndd -set /dev/tcp tcp_recv_hiwat_def 32768
ndd -set /dev/tcp tcp_xmit_lowat_def 8192
```

a646110

### Student Notes

It is possible that not enough network memory has been allocated, such as the memory needed to hold fragmented messages in the IP layer. IP messages may be fragmented into smaller parts when the message is sent through the system. The fragments must be held in memory for some time so that the entire message can be reassembled, because the fragments arrive at the destination at different times and possibly out of order.

Normally, fragmentation reassembly memory is limited arbitrarily so that incomplete messages do not consume all of the memory, which could cripple the system. During stressful networking activity, some fragments may never be delivered because they are typically dropped in transit—for example, due to a collision or resource limitations on an intermediate system. However, fragments may also not be delivered (*dropped*) if there is insufficient fragmentation reassembly memory on the destination system during periods of high network activity. This can degrade performance due to retransmissions of data.

To determine if IP fragments have been dropped see the output from the command  
`netstat -sp ip`.

If the problem is due to a large number of fragments dropped after time-out, consider increasing the amount of memory that may be devoted to packet reassembly. There is an upper bound on the number of bytes IP will use for this purpose, and if the limit is reached reassembly lists are purged until the space required for the new fragments becomes available. The default size is 2,000,000 bytes. To check the current limit:

```
ndd -get /dev/ip ip_reass_mem_limit
```

To reset it to a new value:

```
ndd -set /dev/ip ip_reass_mem_limit 1500000
```

It is also possible that there are difficulties with the buffer sizes allowed for use by TCP. To check the maximum size for the receive window (default 32,768 bytes):

```
ndd -get /dev/tcp tcp_recv_hiwater_def 16384
```

To reset it to a new value:

```
ndd -set /dev/tcp tcp_recv_hiwater_def
```

The behavior for transmitting TCP packets may be adjusted by turning *write-side* flow control on or off, depending upon the current condition of the output buffer. If the buffer is full, flow control is initiated for writing to that buffer. Effectively, the buffer size is increased or decreased by changing this *trigger level*. To check the current setting (default 32768 bytes):

```
ndd -get /dev/tcp tcp_xmit_hiwater_def
```

To reset it to a new value:

```
ndd -set /dev/tcp tcp_xmit_hiwater_def 16384
```

To check the lower limit at which *write-side* flow control should be turned off (default 8192 bytes):

```
ndd -get /dev/tcp tcp_xmit_lowater_def
```

To reset it to a new value:

```
ndd -set /dev/tcp tcp_xmit_lowater_def 4096
```

Note that there are special buffer-control facilities that may be tuned to fast or slow links. To learn more about these:

```
ndd -h parameter
```

where *parameter* is one of the following:

|                      |                                                                         |
|----------------------|-------------------------------------------------------------------------|
| tcp_rec_hiwater_lfp  | maximum receive window size for fast links                              |
| tcp_rec_hiwater_lnp  | maximum receive window size for fast links                              |
| tcp_xmit_hiwater_lfp | the amount of unsent data that triggers TCP flow control for fast links |
| tcp_xmit_hiwater_lnp | the amount of unsent data that triggers TCP flow control for slow links |
| tcp_xmit_lowater_def | the amount of unsent data that relieves TCP flow control                |
| tcp_xmit_lowater_lfp | the amount of unsent data that relieves TCP flow control for fast links |
| tcp_xmit_lowater_lnp | the amount of unsent data that relieves TCP flow control for slow links |

Further information about **ndd** will be presented later in this module.

## 7-7. SLIDE: Performance Measurement Tools

### Performance Measurement Tools

|                                                   |                                                                                                                                                                                              |
|---------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>ping</code>                                 | provides simple statistics on packet transfer rates as well as assisting in troubleshooting                                                                                                  |
| <code>netstat -i</code> and <code>lanadmin</code> | give per-interface statistics including number of packets in and out, number of parity errors, and number of collisions; allow calculation of collision rate = (collisions / output packets) |
| <code>netstat interval</code>                     | allows for continuous display of <code>netstat</code> statistics; used to monitor activity on a live, ongoing basis                                                                          |
| <code>netstat -f inet</code>                      | shows currently active socket connections                                                                                                                                                    |
| HP GLANCE/GPM                                     | graphical monitoring tools                                                                                                                                                                   |

a646111

### Student Notes

#### `ping`

The well-known `packet internet-groper` command is typically used for troubleshooting, but also provides statistics on packet transfer rates. The HP-UX version automatically operates in *statistics mode*, while other vendor-based versions (such as Sun) require the use of the `-s` option. In either case, the number of packets and the packet size may be specified. By default, packets of size 64 bytes are continuously sent, until the administrator types `[CTL]C`. The resultant report includes a count of packets sent, packets received, and timing data, and looks something like:

```
earth# ping mars
PING mars: 64 byte packets
64 bytes from 193.118.172.70: icmp_seq=0. time=9. ms
64 bytes from 193.118.172.70: icmp_seq=1. time=5. ms
64 bytes from 193.118.172.70: icmp_seq=2. time=5. ms
64 bytes from 193.118.172.70: icmp_seq=3. time=6. ms
64 bytes from 193.118.172.70: icmp_seq=4. time=5. ms
```

c

```
----mars PING Statistics----
5 packets transmitted, 5 packets received, 0% packet loss
round-trip (ms) min/avg/max = 5/6/9
```

*Important things to watch out for:* if the remote host returns most of the packets but misses out a few, it is possible that the remote host is badly overloaded, but it is usually the case that a local connector or interface is intermittently malfunctioning.

#### **netstat -i, landiag, and lanadmin**

The **-i** option to **netstat** displays per-interface statistics for this system SINCE BOOT TIME, an unfortunate fact which makes it more difficult to measure activity during a specific test. **landiag** and **lanadmin** offer similar services to display and modify per-interface parameters, and can be told to clear the statistics counters.

*Important things to watch out for:* include the error rate for input and output packets: this should be less than 0.025% of the packets input or output; and the collision rate, which should be less than 10% on average; however, any collision rate greater than 5% on a reasonable sample is cause for concern!

#### **netstat interval**

By running **netstat** with the *interval* option, where *interval* is the number of seconds to pause between reports, it is possible to observe the flow of packets into and out of a system's network interfaces, and also to see evidence of recent collisions.

*Important things to watch out for:* This tool can be started up on suspected busy networking systems before commencing network load testing, and will provide important data on per-host network activity.

#### **netstat -f inet**

This option of the **netstat** command displays a report of current internet socket connections. For each such connection, a line is printed which includes the type of protocol, the length of the receive and send queues, the local and foreign addresses, and the state of the connection.

*Important things to watch out for:* on an overloaded or congested network, the *Send-Q* field for some of the connections may be non-zero; this field is normally zero for most if not all connections nearly all of the time.

## **GLANCE and GPM**

HP's GLANCE/GPM package offers a state-of-the-art graphical user interface tool to monitor the system and network performance of your HP 9000 systems.

## 7-7. SLIDE: Performance Measurement Tools (Continued)

### Performance Measurement Tools (Continued)

|                     |                                                                                                                               |
|---------------------|-------------------------------------------------------------------------------------------------------------------------------|
| <b>netstat -s</b>   | displays per-protocol statistics; can be used on a gateway to indicate data integrity problems                                |
| <b>nfsstat</b>      | displays server- and client-side NFS and RPC statistics; invaluable in determining probable cause of NFS performance problems |
| <b>nettl/netfmt</b> | can be used to trace all LAN packets ( <b>nettl</b> ) and to display report of data thus gathered ( <b>netfmt</b> )           |
| <b>spray</b>        | used to saturate the network interface of another host; used for load testing                                                 |

a646112

### Student Notes

#### **netstat -s on a Gateway**

This option of the **netstat** command displays per-protocol statistics, most of which are only of marginal importance when tackling network performance problems. However, the possibility of data corruption is a performance issue especially on a gateway, since damaged packets normally require retransmission, and increase the load on the network.

*Important things to watch out for:* any *bad checksum* counts greater than zero; try **netstat -s | grep checksum**. These counts should normally be zero or at most hundredths of a percent. If the counts are higher, there is a problem with the gateway, probably some form of memory corruption within the gateway or its network interface.

#### **nfsstat [-cmnrsz]**

The **nfsstat** command provides the best information on the usage and behaviour of your NFS systems. This meter reports client-side and server-side statistics for both NFS and for the

underlying RPC protocol layer. It is discussed in more detail later in this section. The major indicator of poor NFS performance is observed in the client-side RPC report, generated by: **nfsstat -rc**. The retransmission rate can be calculated as:

$$\text{retransmission rate} = (\text{timeouts} / \text{calls})$$

If this rate is consistently greater than 5% the client is experiencing poor response to its NFS requests. The reason is usually either an overloaded network, or an overloaded NFS server. If the value of the *timeout* and *badxid* fields are roughly equal (within a factor of 2 or 3), the problem is probably due to an overloaded NFS server. If the value of the *timeout* field is much larger than the *badxid* field, the problem is probably an overloaded network. This area is discussed in more detail later in this section. The **-c/-s** options are used to display only client/server side information, the **-r/-n** options are used to display only RPC- or NFS-based statistics, and the **-z** option is used to zero the counters. The **-m** option can be used to display per-mount performance information. This can provide important pointers to which NFS server is causing timeouts and retransmissions.

*Important things to watch out for:* A retransmission rate greater than 5%.

#### **nettl**

This command is a tool used to capture network events or packets. Logging is a means of capturing network activities such as state changes, errors, and connection establishment. Tracing is used to capture or take a snapshot of inbound and outbound packets going through the network, as well as loopback or header information.

#### **netfmt**

This command is used to format binary trace and log data gathered from the network tracing and logging facility **nettl**. The binary trace and log information can be read from a file or from standard input.

#### **spray**

The **spray** command is used to send a large burst of packets to another host. It is used to compare the *dropped packet* rate of two systems, and as a method of saturation-testing a host or the network as a whole.

---

## 7-8. SLIDE: **ndd** -- A Network Tuning Tool

### **ndd** – A Network Tuning Tool

- The command **ndd** allows you to examine and set kernel driver parameters for TCP/IP.

- View all the supported parameters.

```
ndd -h supported | more
```

- View the parameters within each protocol family.

```
ndd network_device
```

```
ndd /dev/ip
```

- Examine a particular value.

```
ndd -get network_device parameter
```

```
ndd -get /dev/ip ip_forwarding
```

- Set a particular value.

```
ndd -set network_device parameter value
```

```
ndd -set /dev/ip ip_send_redirects 1
```

a646113

### Student Notes

The **ndd** command allows direct manipulation of the TCP/IP suite of kernel drivers and allows such mechanisms as IP forwarding or debugging to be selected. The command has two forms. The first utilizes the \? flag to display the parameters available for each protocol family, the second form allows the parameter to be set to a supplied value. The following example displays the IP protocol and sets **ip\_debug** to be turned on.

```
ndd /dev/ip \?
?
(ip_ill_status (read only)
 ip_ipif_status (read only)
 ip_ire_status (read only)
 ip_rput_pullups (read and write)
 ip_forwarding (read and write)
(ip_respond_to_address_mask_broadcast (read and write)
(ip_respond_to_echo_broadcast (read and write)
(ip_respond_to_timestamp (read and write)
(ip_respond_to_timestamp_broadcast (read and write)
(ip_send_redirects (read and write)
(ip_forward_directed_broadcasts (read and write)
(ip_debug (read and write)
(ip_mrtdebug (read and write)
(ip_ire_cleanup_interval (read and write)
(ip_ire_flush_interval (read and write)
(ip_ire_redirect_interval (read and write)
(ip_def_ttl (read and write)
(ip_forward_src_routed (read and write)
(ip_wroff_extra (read and write)
(ip_ire_pathmtu_interval (read and write)
(ip_icmp_return_data_bytes (read and write)
(ip_send_source_quench (read and write)
(ip_path_mtu_discovery (read and write)
(ip_ignore_delete_time (read and write)
(ip_ignore_redirect (read and write)
(ip_output_queue (read and write)
(ip_broadcast_ttl (read and write)
(ip_icmp_err_interval (read and write)
(ip_reass_queue_bytes (read and write)

ndd -set /dev/ip ip_debug 1
```

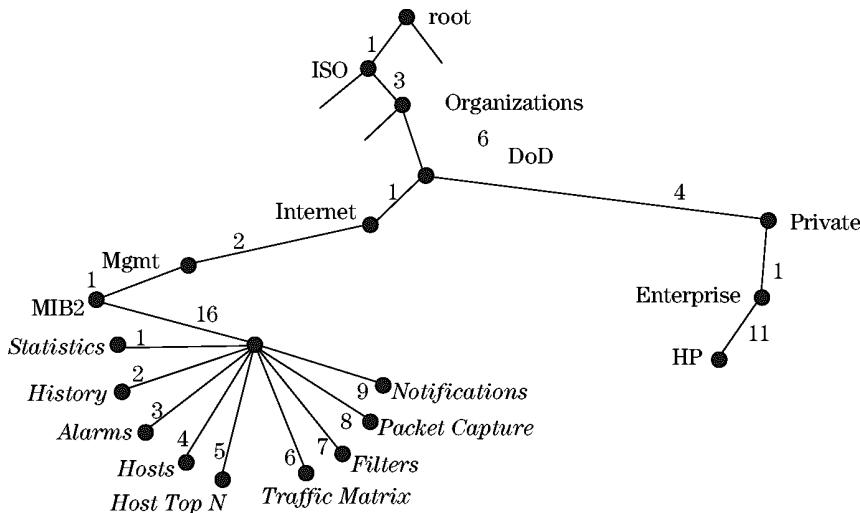
To make your system a firewall gateway, IP forwarding must be turned off, as shown by:

```
ndd -set interface ip_forwarding
```

Windows size of TCP/IP packets, debugging and many other parameters can be switched using the **ndd** command. The administrator should consult the manual references for more information.

## 7-9. SLIDE: Remote Monitoring Management Information Base (RMON MIB)

### Remote Monitoring Management Information Base (RMON MIB)



### Student Notes

A Management Information Base (MIB) is a data structure and command structure for controlling a network resource by means of the SNMP (Simple Network Management Protocol) agent. Many MIBs are standardized to allow various vendors' devices to be managed by diverse network management systems.

The MIB is not a physically distinct database, but rather it is a concept that includes configuration and status values normally available on the agent system. MIB includes objects dealing with IP internetworking routing variables.

The RMON MIB is such an industry standard structure. It provides the basis for managing segment monitors. A segment monitor is a network device which is capable of intercepting all traffic on a LAN (rather than just the traffic intended for the usual sort of network device). Usually, a segment monitor is a special purpose device. Its *promiscuous* mode of operation must be used with some care because of the potential for security breaches.

Such devices are extremely useful in examining network traffic in general and client-server exchanges in particular. For example, a probe can *listen* for such traffic and store a sample of it in a file. This file, in turn, may be analyzed by special purpose programs or scripts to summarize and display the components of such traffic.

However, although RMON is standardized and very precise, it can be complex and expensive. Oddly enough, its very precision, which depends upon intercepting and processing vast amounts of information, can place extraordinary loads on the instrumentation it requires and perhaps even on the very network it is monitoring.

---

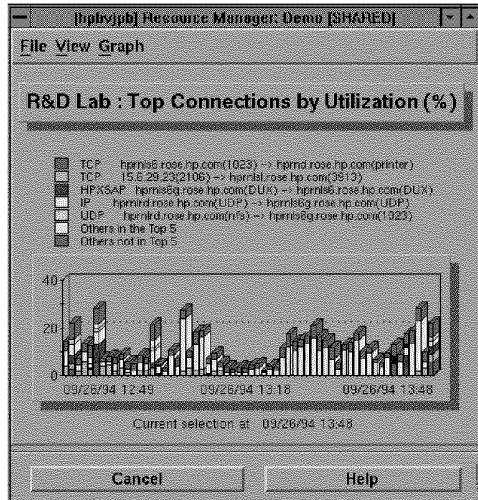
*NOTE:*

---

This slide was reproduced with permission. Copyrighted 1996 material from Prentice Hall Inc.

## 7-10. SLIDE: Embedded Advanced Sampling Environment (HP EASE)

### Embedded Advanced Sampling Environment (HP EASE)



a646115

### Student Notes

Traffic management deals with the use of the entire network and concentrates on understanding and managing the flow of network traffic and predicting future traffic demands. The HP Embedded Advanced Sampling Environment (EASE) technology was designed to be an inexpensive and scalable network-wide traffic management method for identifying and solving problems caused by high network utilization.

The monitoring technique consists of continuous statistical sampling of network traffic on each segment and collecting this data to create a network-wide picture of segment utilization.

The HP EASE agent is embedded in network devices such as hubs or switches, or implemented in a stand-alone probe. Since monitoring is performed by statistical sampling rather than capturing every packet sent across the segment, HP EASE is scalable for networks using high speed technologies such as 100VG or 100Base-T, and large, multisegmented networks. This sampling method also ensures that the traffic overhead incurred by gathering data does not degrade network performance.

HP EASE provides the network manager with important information about networks by allowing useful measures such as total number of bytes received or transmitted and total number of error packets to be predefined. This data is gathered by the HP EASE agent continuously. When the data is analyzed over short periods such as the last minute or last hour, real-time fault isolation can be performed. When the data is gathered over longer periods such as days, weeks, or even months, the overall health of the network can be determined. This trend analysis allows the network manager to engage in more proactive network management activities, impacting network planning, design, and performance.

---

*NOTE:* This slide was reproduced with permission. Copyrighted 1996 material from Prentice Hall Inc.

---

## **7-11. LAB: Using Network Performance Tools**

## **Directions**

1. Use `ping` to obtain transfer statistics on 10 packets of 750 bytes to a host on your LAN. Try the same thing by pinging to a host on the other LAN. If still available, also try to ping to a host on the other side of a SLIP link.
  2. Examine the statistics for your network interfaces by using the `netstat -i` command. Run `netstat` with an interval of 2 seconds in one window, while in another window `spray` another host in the classroom, and observe the activity.
  3. As a group, try to overload your netork by selective spraying. Pair up with another pair of students. Start `netstat` with a 2 second interval in one window. Use the `spray` command to transfer 800,000 packets to the other team's system. Observe the effect. Especially look for collisions. When finished, use the `kill` command to terminate your `spray` if necessary.
  4. Determine the overflow of the UDP socket buffer.
  5. Determine the default buffer size for TCP sockets. Does increasing the TCP socket buffer also increase the network performance?

## 7-12. SLIDE: Understanding NFS Architecture

### Understanding NFS Architecture

- client components
  - **mount**
  - **biod**
  - **lockd**
  - **statd**
- server components
  - **mountd**
  - **nfsd**
  - **lockd**
  - **statd**

a646205

## Student Notes

### NFS

In most local networks today, file sharing is the dominant network-intensive application. The typical NFS network environment comprises several file servers and many clients. The flow of data and attribute information among them could result in a heavy load on the network.

### NFS Version 2 Components

Understanding NFS architecture is very important to improve NFS performance.

Version 2 of Sun's NFS software is defined by the two protocols which implement it: the **mount** protocol, and the **network file system** protocol. Both protocols are based on Sun's XDR and RPC protocols.

## The `mountd` Daemon

The `rpc.mountd` daemon implements the mount protocol on the server side. On the client side the `mount` command establishes the connection to the `mountd`.

The `mount` protocol provides the client with the ability to establish the initial connection to the server's file system. The client-side `mount` command contains RPC-client code to call the server's `rpc.mountd` daemon, which validates the requested `mount` operation, and if authorized, returns the *file handle* of the mounted directory. The client kernel then caches the file handle for subsequent use.

The network file system protocol supports the basic operations on files, including the ability to read, write, create, delete, and list the attributes of server files and directories.

## The `nfsd` Daemon

Most of the server side logic of NFS version 2 is implemented in kernel-resident code, but external daemon processes are used as *front-ends*, which allows for more effective throughput via multi-threading.

The `rpc.mountd` daemon provides the initial file handle to the client in response to the client's `mount` request. This daemon is started by the `/sbin/init.d/nfs.server` script at boot time.

The `nfsd` daemon processes respond to client file/directory access requests. The server system typically runs several such processes, which are scheduled in a pseudo round-robin fashion; when an `nfsd` has completed one request, it is placed at the end of the queue where it awaits a new request.

## The `biod` Daemon

Nearly all of the client side of the NFS version 2 software is implemented in kernel-resident code, but the `biod` daemon processes are used as a front-end so that they may be scheduled. This is done to improve NFS performance.

The `biod` daemon processes (typically 4) are started on a client system in order to improve NFS throughput. The `biod` processes perform pre-fetch operations for NFS client processes which are doing data-related I/O (each user client process doing attribute-related operations performs its own RPC calls).

When a client process reads from an NFS-mounted file, it does the read itself. However, the kernel then has one of the `biod` processes send more read requests to the server, effectively pre-fetching data which the user process will probably subsequently need. This allows the client system to initiate several RPC calls at the same time.

`biod` processes are also used to effectively *batch up* client write requests. It is not absolutely necessary to run `biod` processes, but the client system's performance may suffer!

Note that the maximum data-transfer buffer size is fixed at 8KB in version 2.

## File Locking

Voluntary NFS file locking, which needs to be built into the client programs, utilizes the `rpc.lockd` and `rpc.statd` daemons on both client and server. The `rpc.lockd` daemon

handles file lock requests, while the `rpc.statd` daemon is the network status monitor daemon, which ensures consistency between locking clients and servers. In particular, the status daemons allow locks to be properly reset after a crash.

The `rpc.lockd` and `rpc.statd` daemons are not shown on the slide.

## Synchronous Writing

Because of the stateless nature of the NFS server-side, client write requests must be synchronous; that is, when the client requests a write, the server must complete the operation by writing the data to server disk before the client can be notified. This has a significant impact on NFS performance, and enabling asynchronous writes in a data-intensive environment may have a large impact on improving performance, albeit at the risk of loss of data integrity. Safe asynchronous writing is promised in NFS version 3.

## 7-13. SLIDE: NFS Operations

### NFS Operations

The following actions may be performed by a client process on a server file/directory;

|                 |                                                           |
|-----------------|-----------------------------------------------------------|
| <b>create</b>   | create a node; may be a file or a symbolic link           |
| <b>fsstat</b>   | get info on the filesystem (used by <b>bdf</b> , etc)     |
| <b>getattr</b>  | get file attributes, such as type, size, mode             |
| <b>link</b>     | create a hard link                                        |
| <b>lookup</b>   | search a directory for a specific name, return a handle   |
| <b>mkdir</b>    | create a directory                                        |
| <b>null</b>     | do nothing; used for verification and timing              |
| <b>read</b>     | read an 8KB block of data                                 |
| <b> readdir</b> | read a directory entry                                    |
| <b>readlink</b> | chase a symbolic link on the server                       |
| <b>remove</b>   | remove a link                                             |
| <b>rename</b>   | change the file's directory name entry                    |
| <b>rmdir</b>    | remove a directory                                        |
| <b>root</b>     | get the root-level directory of this mount                |
| <b>setattr</b>  | change file/directory attributes such as date/times, size |
| <b>symlink</b>  | create a <b>symlink</b> in the remote file system         |
| <b>write</b>    | write an 8KB block of data                                |

a646206

## Student Notes

### NFS Operations

The NFS protocol defines 18 operations which may be performed by client processes against the files and directories on the server. Most of these operations are only occasionally used, but five tend to dominate the activity of most NFS environments:

- **lookup**
  - **getattr**
  - **setattr**
- and
- **read**
  - **write**

The former are used to manipulate a file's *attributes* while the latter manipulate a file's *contents*. The actual number and relative percentage of calls sent by a client, and those received by a server, may both be displayed with the **nfsstat** command.

## **The Anatomy of a Typical NFS Transaction**

When a client program opens an NFS file on the server, it must issue both **lookup** and **getattr** operations on *each directory* in the path, in order to check for the existence of, and the permissions on, the intermediate directories and the target file. It then issues the **read** request to request the first block of data. When finished, the client then issues a **setattr** to change the last access time for the file. A similar set of operations are involved in writing.

## **Attribute-Intensive versus Data-Intensive Activity**

As you can see from the anatomy of a typical transaction, most NFS operations involve obtaining or changing the *attributes* of a file, rather than its contents. Such *attribute-intensive* behaviour is typical of most NFS environments today, since most sites are operating on files of moderate size. Some sites, however, such as CAD and image-processing environments, typically operate on very large files (1MB-1GB in size), and thus have NFS activity dominated by *data-intensive* traffic. The **nfsstat -sn** command may be used to seek an indication as to the nature of your environment (discussed later in this section).

### **Attribute-Intensive Environments**

This is the typical NFS environment, and most of the NFS-related network traffic is being used to carry relatively small packets containing attribute information. This does not place an excessive load on the typical 10Mbit Ethernet bandwidth. Nonetheless, a figure of 20 to 25 clients per Ethernet network is considered to be a sensible limit. Version 2 of NFS provides for *attribute caching* on the client side, which further improves performance.

### **Data-Intensive Environments**

NFS sites which are data-intensive in nature place a considerably greater load on the network bandwidth. For example, a typical 8KB data transfer moves about 75 times as much data as an attribute packet transfer, and requires the transmission of several Ethernet packets. In such environments, the typical limit is about 10 to 15 clients, and more advanced network technologies are required (such as FDDI, 100Mbit/sec Ethernet or ATM networks). For data-intensive environments the largest path MTU size is very important to reduce the number of IOPS (I/O operations per second).

## 7-14. SLIDE: Typical NFS Bottlenecks

### Typical NFS Bottlenecks

The following typical bottlenecks contribute to poor NFS performance:

- synchronous writing
- number of  **nfsd/biod**
- timeouts and retransmissions
- attribute caching
- server disk I/O impact
- placement of the NFS server(s)
- network buffers
- At a simpler level, the problem is usually either *network too busy* or *server too busy*.
- Most commonly, the problem is caused by the server disk bandwidth.

a646116

## Student Notes

### NFS Bottlenecks

When poor NFS performance is indicated, the bottlenecks are usually one of those listed above. Most of the time, the case boils down to either *network too busy* or *server too busy*, and when the server is indicated, it is usually due to disk bandwidth on the server.

Each bottleneck we will discuss in detail below.

## 7-15. SLIDE: Synchronous Writing

### Synchronous Writing

- synchronous writing
  - default NFS uses synchronous writing
  - necessary for compatibility
- asynchronous writing
  - caches write requests on NFS server
  - improves dramatically NFS performance
  - not compatible to local write requests
  - increases risk of losing data
- exporting file systems asynchronously
  - `exportsfs -i -o async /directory`

a646207

## Student Notes

### Synchronous Writing

When a write operation is performed by a client, the client must wait for the server to acknowledge that the write has taken place. On the server side, when the client requests a write, the server must complete the operation by writing the data to server disk before the client can be notified.

This is because of the stateless nature of the NFS server-side. Client write requests must be synchronous. This has a significant impact on NFS performance.

### Asynchronous Writing by the Server

Synchronous writing considerably slows overall NFS performance, since the server must immediately perform NFS writes so as to avoid as much hold-up to the client. This in turn reduces the efficiency of caching on the server, and makes for poorer server performance.

On HP-UX it is possible to enable asynchronous writes. A data-intensive environment may have a large impact on improving performance, albeit at the risk of loss of data integrity. Safe asynchronous writing is promised in NFS version 3. Therefore, if the risk of the loss of data can be tolerated, the server should export the file system with the `async` option in the `/etc(exports` file, as in:

```
/users -async
```

HP's `-async` option is a write request passed to the system's memory that is acknowledged before the data is written to disk. The system will write the data in memory to disk.

The risk of mounting asynchronously is the chance of losing data if the NFS Server CPU crashes. If the NFS Server's disk crashes, data will be lost in the CPU and from the disk.

---

**NOTE:**

---

The NFS Client always sends data to the server with every `write` operation. An NFS Client crash will lose data not written to the NFS server.

## 7-16. SLIDE: Changing the Number of **biods** and **nfsds**

### Changing the Number of **biods** and **nfsds**

- Start several **biods**:
  - **biod 4**
- Start several **nfsds**.
  - **nfsd 4**
- Change the values of **nfsd** and **biod** at system startup.
  - **vi /etc/rc.config.d/nfsconf**
  - **NUM\_NFSD=4**
  - **NUM\_NFSIOD=4**

a646117

## Student Notes

- NFS client

Block I/O daemons (**biods**) facilitate requests to the servers, by buffering requests together. The normal default is 4 **biods** per client. Usually, this number is sufficient for most clients, however it should be increased if **netstat -s -p udp** indicates *socket overflows*.

- NFS server

Network File Server daemons (**nfsds**) handle requests from the clients. Default number of **nfsds** is 4.

Recommendation: Roughly 3 **nfsds** should be configured per disk on large multiple-CPU systems.

By not having enough **nfsds** configured, clients have to make the same request multiple times before the server will return a response. This slows down NFS performance.

Configuring too many **nfsds** can leave some of them idle, or it can affect overall system performance.

**netstat -s -p udp** indicating *socket overflows* would suggest that more **nfsds** would help.

## 7-17. SLIDE: Timeouts and Retransmissions

### Timeouts and Retransmissions

- Mount options `timeo` (default 7) and `retrans` (default 3) to control the level of impatience of the client.
- ```
# nfsstat -rc
Display client rpc statistics.
calls      badcalls      retrans  badxid timeout  wait
78627      1979       983      2480    3981      0
```
- Calculate `retransmission rates`.

$$\text{retransmission rate} = \frac{\text{timeout}}{\text{calls}}$$
 If `retransmission rate > 5 percent`:
 - If approximately the same, the server is slow.
 - If much less, the network is slow.
 - Change the timeout option.


```
mount -o timeo=20 earth:/users /users
```

a646208

Student Notes

NFS Retransmissions

NFS requests utilize RPC *on top of* UDP, so the responsibility for ensuring the accurate and reliable transmission of data is taken on by the NFS protocol itself. This is achieved by client-side NFS. The client process uses two mount option parameters, `timeo` and `retrans` to control retransmissions. `timeo` is expressed as an integer and represents the number of tenths of a second for a timeout—that is, when a client issues a request, it expects a response within `timeo/10` seconds. If no such response is received, a *minor* timeout occurs. The timeout interval is doubled, and the request is retransmitted. This process is repeated until the retransmission count reaches `retrans`, at which point a *major* timeout occurs. After a *major* timeout, a message is displayed such as:

```
NFS server earth not responding, still trying
```

At this point, the initial timeout period is doubled, the count of retransmission attempts is zeroed, and a new major cycle begins. Note that if the file system were *hard mounted* this

behaviour continues until the server responds. However, for *soft* mounts, the cycle ends at the first major timeout.

Reasons for Timeouts and Retransmissions

Timeouts and retransmissions can have three main causes:

- an overloaded or slow server
- an overloaded or defective network
- slow network

It is important to distinguish between a slow and a overload network. For example a slow serial line can lead to a timeout although it is not overloaded. In such a case it is very easy to solve the problem by increasing the timeout parameter.

Increasing the timeout parameter on an overloaded network decreases the retransmission. On an overloaded network it is very difficult to determine the right retransmission time. However, increasing the timeout parameter too much may decrease the NFS performance. It may cause too long a wait before retransmission if a packet is truly lost.

How can we determine which is the reason for timeouts?

Measuring Timeouts and Retransmissions

The **nfsstat** command provides important clues as to the cause of poor NFS performance when run on the client side. The **-rc** options to **nfsstat** display the client-side **rpc** statistics, and an analysis of the figures displayed usually gives a good indication as to whether the bottleneck is due to poor network performance, or to poor server response.

Output of **nfsstat -rc**

The client output of the **nfsstat** command displays the following fields:

calls	The total number of RPC calls made.
badcalls	The total number of calls rejected by the RPC layer.
retrans	The number of times a call had to be retransmitted due to a timeout while waiting for a reply from the server.
badxid	The number of times a reply from a server was received which did not correspond to any outstanding call.
timeout	The number of times a call timed out while waiting for a reply from the server.
wait	The number of times a call had to wait because no client handle was available.

Retransmission Rate

This is the percentage of calls which result in a retransmission, due to a timeout on receiving the expected response. It is calculated by dividing the number of timeouts by the number of calls. Although opinion differs as to what constitutes an acceptable rate, a value in excess of 5% is certainly a strong indication of poor responsiveness to client requests.

Server Overload

Each client request is identified by a specific and unique `transmission id`, or `xid` for short. This `xid` is used to allow the client to match up the reply to the request originally sent. A `badxid` occurs when a duplicate reply is received from a server. This happens because the server is overloaded, causing the client to retransmit its request, and the server is *eventually* responding to all requests, including the duplicated retransmissions from the client. Thus, if the `badxid` and timeout counts are both of the same order of magnitude (within a multiple of 2 or 3 of each other) then poor server performance is probably indicated.

Network Overload

Alternatively, the responsiveness of the network itself is indicated if the retransmission rate is unacceptable, and if `badxid` is very small compared to the number of timeouts. This is because a busy network will result in many packets getting dropped, both from client and server, so the client is likely to get only one response to its request, but with many intervening timeouts. The absence of `badxids` means that the server is responding acceptably quickly to what it gets, but that it is hard work to get the packets through.

Client Mount Options

The default value for the `timeo` option is 7 (or .7 seconds) while the default retransmission value is 3. The `timeo` option may be increased in an attempt to accommodate slow servers. For example, the `mount` command:

```
mount -o timeo=20 earth:/users /users
```

would reduce the level of impatience of the client, who would therefore be more tolerant of slow responses from the server. This would reduce the load on both the network and the server, since fewer retransmissions would take place.

Use `nfsstat -m` to Identify Slow Servers

The `-m` option to `nfsstat` provides per-mount statistics on lookups, reads and writes. The information given includes the smoothed round-trip time in milliseconds (`srtt`) as well as the average deviation (`dev`) and current expected response time (`cur`). For example, the following output indicates relatively poor response time for one of the mounts:

```
$ nfsstat -m
/usr/man from venus:/usr/man (Addr 193.118.172.66)
Flags: hard read size=8192, write size=8192, count=5
Lookups: srtt=13 (32ms), dev=7 (35ms), cur=5 (100ms)
Reads: srtt=21 (52ms), dev=11 (55ms), cur=8 (160ms)
Writes: srtt=79 (197ms), dev=14 (70ms), cur=16 (320ms)
All: srtt=28 (70ms), dev=18 (90ms), cur=12 (240ms)
```

```
/cdrom from neptune:/cdrom (Addr 193.118.172.34)
Flags: ro,hard read size=8192, write size=8192, count=5
Lookups: srtt=130 (32ms), dev=7 (35ms), cur=5 (100ms)
Reads: srtt=210 (52ms), dev=11 (55ms), cur=8 (160ms)
Writes: srtt=790 (197ms), dev=14 (70ms), cur=16 (320ms)
All: srtt=280 (70ms), dev=18 (90ms), cur=12 (240ms)
```

The Given Example

In our example, the retransmission rate is just over 5% and the relative closeness of `badxid` and `timeout` means that server performance problems are indicated.

7-18. SLIDE: Attribute Caching

Attribute Caching

```
# nfsstat -s
Server rpc statistics:
calls      badcalls      nullrecv      badlen      xdrcall
  0          0            0            0            0

Server nfs statistics
calls      badcalls
2015657      0
null      getattr      setattr      root      lookup      readlink      read
774 0%    140751 7%   28639 1% 0 0%    309442 15% 53811 3% 743967 37%
wrcache   write       create       remove      rename      link       symlink
0 0%    569183 28% 91173 5% 8862 0% 9902 0%    381 0%   155      0%
mkdir     rmdir       readdir      fsstat
482 0%   390 0%    55923 3% 1822 0%

• symlink > 10% : too much symlink chasing
• getattr > 60% : clients not attribute caching
```

a646209

Student Notes

Other Server-side Performance Indicators

The **nfsstat** command can be used to illuminate possible server-side performance problems. The demographics of which type of operations are being requested may indicate whether the NFS environment is attribute-intensive or data-intensive. Certain types of operation, if indicated as occurring with too much frequency, also indicate problem areas.

nfsstat -s Indicators

Some of the counts displayed for the RPC and NFS reports on the server side can be used to indicate a problem area.

calls The total number of RPC calls received.

badcalls The total number of calls rejected by the RPC layer. The sum of **badlen** and **xdrcall** as defined below.

nullrecv	The number of times an RPC call was not available when it was thought to be received.
badlen	The number of RPC calls with a length shorter than a minimum-sized RPC call.
xdrcall	The number of RPC calls whose header could not be XDR decoded.

Server-side NFS: Data-Intensive versus Attribute-Intensive

The counters here display the count and percentage of each of the NFS operations discussed earlier, as well as the total count. By adding the percentage for

- **read**
- **write**

and comparing that to the total percentage of

- **getattr**
- **lookup**
- **setattr**

you can get some measure of how data-intensive or attribute-intensive your environment is. In the example above, 65% of the client requests received are for **read/write** while only 23% are for **getattr/setattr/lookup**, indicating a rather data-intensive site.

NOTE:

Don't be concerned about the percentage of attribute caching if no data are exchanged. Of course, under such circumstances the percentage for attribute caching is very high without special meaning.

Server-side NFS: Other Indicators

If **symlink** is greater than 10%, clients are making excessive use of symbolic links on the filesystems being exported by the server. You should replace the symbolic links with a directory, and export it on the server. The client should mount both the original filesystem, and the directory which contains whatever was being pointed to.

If **getattr** is greater than 60%, you should check to make sure that attribute caching is being used by all your clients.

7-19. SLIDE: Attribute Caching Options

Attribute Caching Options

Option	Function	Default
<code>acregmin=n</code>	Min. lifetime for file attributes	3 sec.
<code>acregmax=n</code>	Max. lifetime for file attributes	60 sec.
<code>acdirmmin=n</code>	Min. lifetime for dir. attributes	30 sec.
<code>acdirmmax=n</code>	Max. lifetimes for dir. attributes	160 sec.
<code>actimeo=n</code>	Sets all above to n	
<code>noac</code>	Turns off all attribute caching	

a646210

Student Notes

mount Command's Attribute Caching

The attributes of a file are stored in the client's memory. When a client makes a request for data that is on an NFS server, the system will first check the client's cache to see if it has a copy of the data. The data may not be in the buffer, but the buffered attributes might still remain. This saves the system a series of `rpc` calls asking the server for the attribute information before requesting the actual data. The less attribute caching that is done, the higher the Inputs/Outputs Per Second (IOPS) per client. The attribute options are:

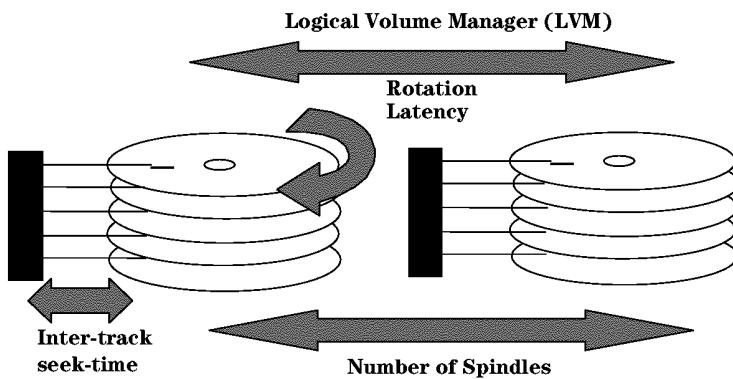
Attribute Caching Options

Option	Function	Default
acregmin=n	Min. lifetime for file attributes	3 sec.
acregmax=n	Max. lifetime for file attributes	60 sec.
acdirmax=n	Max. lifetimes for dir. attributes	160 sec.
actimeo=n	Sets all above to n	
noac	Turns off all attribute caching	

When you mount a filesystem, you explicitly specify with the attribute caching options how long you want the client system to cache information about the file(s). The values that the options are configured to be depends on how often you think the file attribute information changes. If the file is accessed very frequently, you might use the **noac** option. If the file rarely changes (for example read-only), then you may want to change the minimum lifetime options to very large values.

7-20. SLIDE: Server Disk I/O Impact

Server Disk I/O Impact



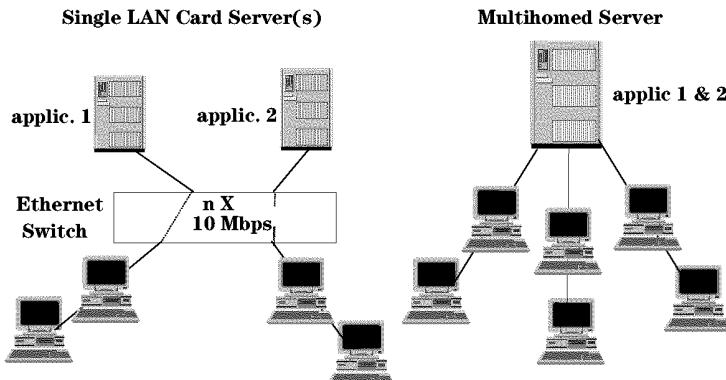
a646211

Student Notes

The slowest part of a system tends to be data transfers between the system and the disk devices. Making sure that the disks aren't above 80% full will keep the disk's performance from degrading.

7-21. SLIDE: Strategies for Applications on NFS Servers

Strategies for Applications on NFS Servers



a646118

Student notes

If key applications exist across several servers, and these are required across your organization's LAN, it may be useful to attach the servers and the subnets to an Ethernet switch. This is especially useful in cases where there is significant client-to-client traffic across subnets.

Mounting from a MultiHomed Server

Clients mounting from a multi-homed server should ensure that they use the hostname/IP address for the server's interface on *their* network. Although it is perfectly legal to mount from the *other side* of the gateway, doing so will require the IP layer of the server's kernel code to internally forward the packets (unnecessarily) to the other interface.

Multiple LAN cards can increase IOPS. The example shows clients bound to a single LAN card, effectively creating a segmented topology.

System LAN Card Support

Systems	Number of LAN cards supported
700	2
E Series	4
F Series	2
G Series	4
H Series	7
I Series	7
K 100,2xx	4
K 4xx	8
T Series	32
V Series	16

Avoid Heavy Client/Server Loads Across Subnets

Wherever possible, try to place heavy clients on the same subnet as their server. If the client and server are located on different subnets, all traffic must pass through the router(s) between them. If a router is itself overloaded, packet loss may result in the retransmission of large (8KB) reads and writes. Should this situation be unavoidable, reduce the size of the read and/or write buffers to limit the loss of large buffers during peak periods of activity.

Stepping-Stone Mounts

Client systems should avoid *stepping-stone* mounts. This is when the client mounts one NFS directory on top of a directory which is itself part of another NFS-mounted filesystem. If stepping-stone mounts are used, any reference to a file/directory below the lowest-level mount may require NFS operations on several servers.

7-22. LAB: Optimizing NFS Performance

Directions

Form a group of two-system teams by pairing with another set of students—for example, those at the system either beside you, in front of, or behind you in the class. One system is the NFS server the other the NFS client. For a useful and comparable measurement, it is important not to disturb each other. Agree with the other team when to use the same physical network segment.

Choose a filesystem with at least 10 free MBytes. Assume the directory `/local/mnt` is located in this filesystem. In your lab you can rename this directory if you choose to do so.

On both the client and server sides, create a directory with the name `/local/mnt`.

On the client side create a directory called `/remote/mnt`, as well as a file `/local/mnt/bigfile` with approximate 8 MB. On HP-UX systems you can copy the UNIX kernel for this purpose.

```
cp /stand/vmunix /local/mnt/bigfile
```

After all data transfers, measure the required time with `time` command. At a minimum, take each measurement twice. Transcribe all results to the table below.

1. Local copy: On the NFS client create a shellscript `perf` and change the mode of `perf` to executable.

In the script, measure the local copying on the client by copying `/local/mnt/bigfile` to `/remote/mnt/bigfile`. Time the execution of `perf`. Finally, remove the file `/remote/mnt/bigfile`.

2. Theoretical transfer rate: Calculate the minimum time used for transferring the `/local/mnt/bigfile` via your network.

Next, reduce this value with about 6 percent for protocol headers. What do you get for the UNIX kernel with a size of 8.8 MB?

3. Remote copy with FTP: Using `ftp`, copy the local file from the client to the server. First, choose the ascii mode for transferring the file, then switch to the binary mode. Again take each measurement at least twice.

4. Synchronous writing: Varying the Number of **biod** and **nfsd**-daemon

Configure your NFS environment. On the NFS server, export the directory **/local/mnt** with root access for the NFS client. Start the NFS servers using the following criteria:

First start only 1 **nfsd**:

With **ps** determine the number of **nfsds** that are running. Kill all **nfsds** and restart the **nfsd** daemon:

Don't forget to start the **mountd**, if it is not running. (Of course, for configuring NFS you can use a tool such as SAM.)

In the same way as you did on the server kill all **biod** on the client, and restart only one **biod** daemon:

On the NFS client, mount the **/local/mnt** directory from the server at the mount point **/remote/mnt**

Measure the file transfer time at least twice:

Now, vary the number of **biods** and **nfsds**:

5. Asynchronous writing: Reduce the number of **nfsd** and **biod** daemons to 4.

On the NFS server re-export the filesystem with asynchronous writing. How much does the performance increase now?

6. Attribute caching: On the NFS server, export the **/usr** filesystem as read-only:

On the NFS client, mount **/usr** directory from the server to **/rmnt** as read-only, too. Zero the NFS statistic on the client:

Measure the amount of time the **find** command uses in stepping through the **/rmnt** subdirectories.

Look at the NFS statistics. Write the **nfsstat** output to **/tmp/cache.out**. Again, zero out the NFS statistics. Then unmout **/rmnt** and remount it, but this time do not use attribute

caching. How long does it take the **find** command to go through the **/rmnt** subdirectories this time? Write the **nfsstat** output to **/tmp/nocache.out**. Compare the cache output with the nocache output.

7. Calculating NFS timeouts: Run the **nfsstat -m** command and try to identify the best and worst server response time. Use the results obtained above to fill in the following table:

	num biod	num nfsd	test1	test2	transfer rate
1)	local copy	-	-		
2)	theorectical rate				
3a)	ftp, ascii mode	-	-		
3b)	ftp, binary mode	-	-		
4a)	NFS, write	1	1		
4b)	NFS, write	1	4		
4c)	NFS, write	4	4		
4d)	NFS, write	6	6		
4e)	NFS, write	8	8		
-	NFS, read	4	4 (8.7) (8.6)		(1050 KB/sec)
5	NFS, write, async	4	4		

Module 8 — Dynamic Host Configuration Protocol

Objectives

Upon completion of this module you will be able to do the following:

- List the characteristics of the BOOTP protocol.
- Describe the features of the BOOTP protocol header.
- Understand BOOTP protocol vendor formats.
- Understand Dynamic Host Configuration Protocol.
- List DHCP client states.
- Understand DHCP architecture on HP-UX.
- Describe DHCP keywords.
- Configure DHCP client and server.

8-1. SLIDE: Dynamic Host Configuration Protocol

Dynamic Host Configuration Protocol

Dynamic Host Configuration Protocol ...



... allows dynamic IP address configuration

- fixed address assignment... ...**manual** configuration
- infinite lease time... ...**automatic** configuration
- limited lease time... ...**dynamic** configuration

... enhancement of BOOTP Protocol

- upwards compatible to BOOTP Protocol



a646212

Student Notes

In addition to the Bootstrap Protocol (BOOTP), the Dynamic Host Configuration Protocol (DHCP) provides advanced IP address allocation and management for TCP/IP LAN computing environments.

DHCP also supports more configuration options than BOOTP. DHCP clients can include TCP/IP network printers, X terminals and Microsoft Windows machines.

DHCP is an enhancement of the BOOTP and therefore supports new DHCP clients as well as existing BOOTP clients.

DHCP supports three types of address assignment:

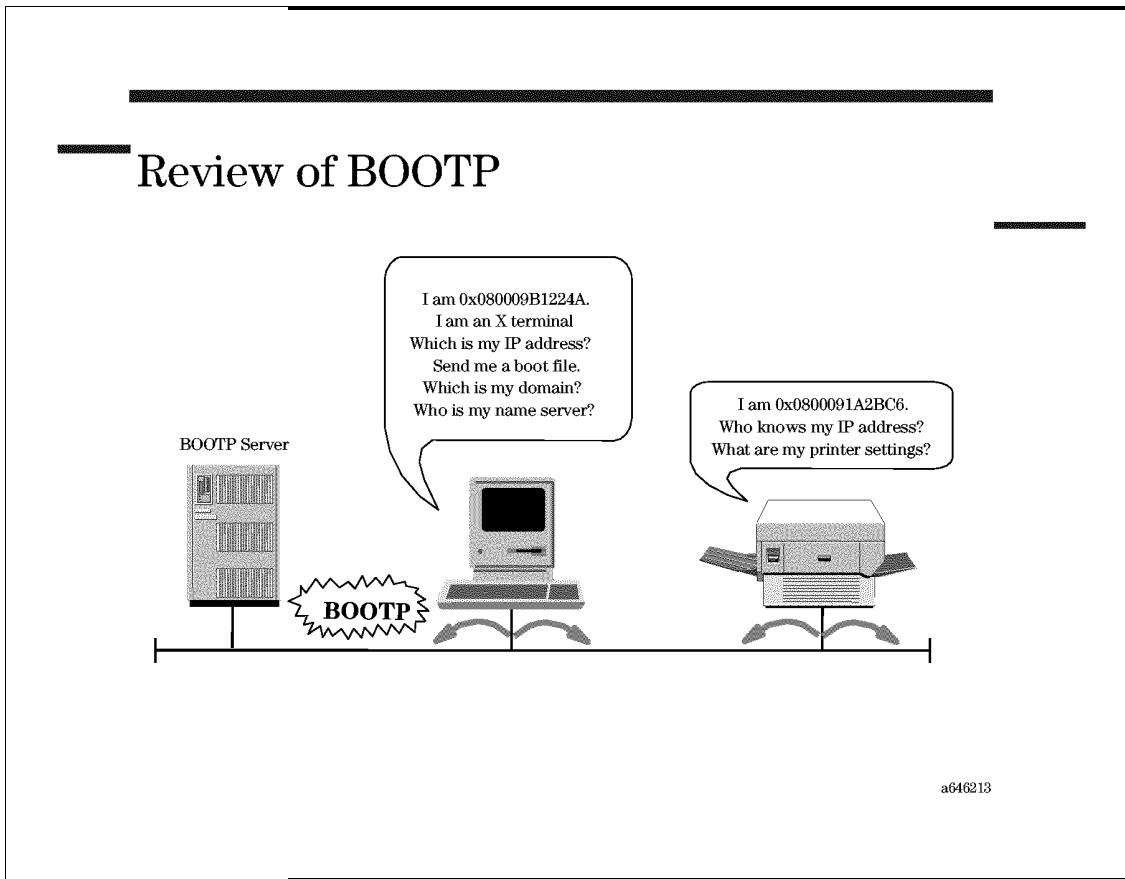
Manual configuration (fixed address assignment)	Fixed, manually assigned IP address for each MAC-address. Fully compatible with BOOTP.
Automatic configuration (infinite lease time)	The DHCP server automatically assigns an IP address when a computer first attaches to the network. From this point the address remains permanent.
Dynamic configuration (limited lease time)	The DHCP server leases an IP address for a limited time. The IP address assignment can be completely dynamic.

All three types of assignment can be mixed within one server.

The DHCP Server offers advanced IP address management for TCP/IP LAN computing environments. By automating IP address allocation, the server offers a level of automation not available with today's client-server bootstrap protocol. DHCP preserves your investment in BOOTP by supporting new and existing BOOTP clients, as well as new DHCP clients. DHCP further improves upon previous versions of BOOTP by supporting more configuration options.

There is a new command line tool `dhcptools(1M)` that further simplifies the management of DHCP devices. `dhcptools` provides access to DHCP-related operations for the `bootpd` server. The options provide control for dumping internal data structures, generating a hosts file, previewing client address assignment, reclaiming unused addresses, tracing packets, and validating configuration files.

8-2. SLIDE: Review of BOOTP



Student Notes

The BOOTP Protocol provides a client with following information:

IP address

BOOTP clients like network printers, X terminals or NFS diskless clients broadcast a BOOTP request asking a BOOTP server for an IP address. The BOOTP request contains at least the client's hardware address.

The server manages a BOOTP database (`/etc/bootptab`) that contains for each hardware address a fixed IP address for the BOOTP client.

In addition to the IP address the boot reply packet contains additional information for the client:

- common client configuration parameters, such as subnet mask
- the name of a boot file that the client loads in the second phase via the `ftp` protocol. The boot file can be a any file—for example:
 - the object code of the UNIX® kernel for an NFS diskless client
 - the object code of an X server program for an X terminal
 - or just an ASCII file for a network printer, containing printer network configuration parameters

8-3. TEXTPAGE: BOOTP Header

Student Notes

The BOOTP protocol has some fixed protocol fields. Most information is vendor specific and can be customized for use by each vendor.

Following is an explanation of the fields of the BOOTP header:

OPERATION	REQUEST or REPLY
HTYPE	Network hardware type.
HLEN	Hardware address length.
HOPS	Number of <code>bootpd</code> routers the packet passed.
TRANSACTION ID	Identification number for the request generated by the client.
SECONDS	Number of seconds since the client started to boot.
CLIENT IP ADDRESS	Client IP address from an earlier request or the (EPROM) configuration. Otherwise set to zero.
YOUR IP ADDRESS	IP address responded by the server
SERVER IP ADDRESS or SERVER HOST NAME	Earlier request or the (EPROM) configuration. Otherwise set to zero. If these fields are not zero, only the appropriate server will send a response.
BOOT FILE NAME	Name of a file which will be loaded via <code>tftp</code> .
VENDOR SPECIFIC AREA	Optional information which can be passed from the server to the client. This area is vendor specific.

8-4. SLIDE: The bootptab Configuration File

The bootptab Configuration File

```
defaulthost:\n    ht=ether:\\\n    vm=rfc1048:\\\n    sm=255.255.224.0:\\\n    gw=15.83.63.54\n\nxterm:\\\n    tc=defaulthost:\\\n    ha=08000901A2BC6:\\\n    ip=15.83.32.1:\\\n    dn=edu.hp.com:\\\n    ds=15.244.81.15:\\\n    bf=C2300A:\\\n    hd=/usr/lib/X11\n\nljprinter:\\\n    tc=defaulthost:\\\n    ha=080009B1224A:\\\n    ip=15.83.32.8:\\\n    T144="hpnp/nplaser1.cfg"
```

/etc/bootptab

a646150

Student Notes

Upon startup, Internet Bootp **bootpd** reads its configuration files to build its internal database, then listens for boot request packets. **/etc/bootptab** and **/etc/dhcptab** are the default configuration files. **bootpd** rereads its configuration file when it receives a hangup signal, **SIGHUP**, or when it receives a boot request packet and detects that the configuration file has been updated. The configuration uses two-character, case-sensitive tag symbols to represent host parameters. These parameter declarations are separated by colons (:). The general format is:

hostname:tg=value:...:tg=value:...:tg=value:...

where **hostname** is the actual name of a DHCP/BOOTP client.

The followings tags are valid for both BOOTP and DHCP entries:

ba

This tag specifies that **bootpd** should broadcast the boot reply to the client. As a boolean tag, it causes **bootpd** to send the boot reply on the configured broadcast address of each network interface. You can also assign the tag an IP address value, which specifies the specific IP or broadcast address for the boot reply.

bf=filename

This tag specifies the filename of the boot file that the client should download. The client's boot request, and the values of the **hd** (see below) and **bf** symbols, determine the contents of the boot file field in the boot reply packet. If the client specifies an absolute path name (in its boot request), and that file is accessible on the server machine, **bootpd** returns that path name in the reply packet. If the file is not accessible, the request is discarded; no reply is sent. If the client specifies a relative path name, **bootpd** constructs a full path name by appending the relative path name to the value of the **hd** tag, and tests to determine if the full path name is accessible. If the full path name is accessible, it is returned in the boot reply packet; if not, the request is discarded. If the **tftp** pseudouser exists, **bootpd** treats all path names (absolute or relative) as being relative to the home directory of **tftp** and checks there first. If the file is not accessible under the **tftp** home directory or the **tftp** pseudouser does not exist, **bootpd** checks for the file relative to **/**.

bs=size

This tag specifies the size of the boot file in 512-octet blocks, or the keyword **auto**, which causes the server to automatically calculate the boot file size at each request.

ci=client_ID

This tag specifies the client identifier of the client. The parameter **client_ID** can be either a hexadecimal integer, or a string contained in double quotes. **client_ID** is a unique identifier that the DHCP client may use to identify itself to the server. If present, the client identifier supersedes the hardware address, so a client and an entry will only match in one of two situations: one, they both have the same client identifier, or two they both have the same hardware address and neither has a client identifier. If a request has a client identifier, then that is used to match

<i>dn=domain_name</i>	the client up with an entry in the bootp configuration file. One common client ID used is to concatenate the hardware type (e.g. 0x01 for Ethernet) with the hardware address.
<i>ds=ip_address_list</i>	This tag specifies the domain name of the client for Domain Name Server resolution (see RFC 1034).
<i>ef=filename</i>	This tag specifies the IP addresses of RFC 1034 Domain Name Servers.
<i>gw=ip_address_list</i>	Specifies the name of an extensions file. The file, retrievable via tftp , contains information which can be interpreted in the same way as the 64-octet vendor-extension field within the BOOTP response. The maximum length of the file is unconstrained. All references to an extensions filename within the file are ignored.
<i>ha=hardware-address</i>	This tag specifies the IP addresses of gateways for the client's subnet. If one of multiple gateways is preferred, it should be listed first.
<i>hd=home-directory</i>	This tag specifies the hardware address of the client. The hardware address must be specified in hexadecimal; optional periods and/or a leading 0x can be included for readability. The ha tag must be preceded by the ht tag either explicitly or implicitly (see tc below).
<i>hn</i>	This tag specifies a directory name to which the boot file is appended (see the bf tag above). The default value of the hd tag is /.

<i>ht=hardware-type</i>	This tag specifies the hardware type of the subnet. The most common are ether , ieee802 , and tr .
<i>ip=ip-address</i>	This tag specifies the IP address of the DHCP/BOOTP client.
<i>lp=ip_address_list</i>	This tag specifies the IP addresses of Berkeley 4BSD printer servers.
<i>md=merit_dump_file</i>	This tag specifies the name of a file to dump the core of a client.
<i>na=ip_address_list</i>	This tag specifies the IP address(es) of RFC 1001/1002 NetBIOS name server(s) in order of preference.
<i>nb=ip_address_list</i>	This tag specifies the IP address(es) of RFC 1001/1002 NetBIOS datagram distribution server(s) in order of preference.
<i>nc=NetBIOS_node_type</i>	Specifies the NetBIOS node type code. Allows NetBIOS over TCP/IP clients to be configured as described in RFC 1001/1002. NetBIOS_node_type can be an unsigned decimal, octal, or hexadecimal integer corresponding to one of the client types as follows: 0x1 or B-node for B-node; 0x2 or P-node for P-node; 0x4 or M-node for M-node; 0x8 or H-node for H-node.
<i>nd=string</i>	This tag specifies the NetBIOS over TCP/IP scope parameter for the client as specified in RFC 1001/1002.
<i>nt=ip_address_list</i>	This tag specifies the IP addresses of Network Time Protocol servers. Servers should be listed in order of preference.
<i>rl=ip_address_list</i>	This tag specifies the IP addresses of RFC 887 Resource Location Protocol servers.
<i>rp=root_path</i>	This tag specifies a path name to be mounted as a root disk.
<i>sm=subnet-mask</i>	This tag specifies the client's subnet mask. subnet-mask is specified as a single IP address.

tc=template-host

This tag indicates a table continuation. Often, many host entries share common values for certain tags such as domain servers. Rather than repeatedly specifying these tags, a full specification can be listed for one host entry and shared by others via the **tc** mechanism. The template-host is a dummy host that does not actually exist and never sends boot requests.

*sr=destination_ip_address
gateway_ip_address ...*

This tag specifies a list of static routes that the client should put in its routing cache. Each route consists of a pair of IP addresses. The first address is the destination address, and the second is the router. Use the **gw=** option to specify the default route (0.0.0.0) as it is not a legal destination address. The **ss=ip_address** tag specifies the IP address of a swap server.

Tnnn=generic-data

This is a generic tag where **nnn** is an RFC 1533 option field tag number. Use this option to configure RFC1533 options not currently supported with **bootpd** tag names. This option allows one to immediately take advantage of future extensions to RFC 1533. The generic-data data can be represented as either a stream of hexadecimal numbers or as a quoted string of ASCII characters. The length of the generic data is automatically determined and inserted into the proper fields of the RFC 1541-style boot reply.

to=offset

This tag specifies the client's time zone offset in seconds from UTC. The time offset can be either a signed decimal integer or the keyword **auto** which uses the server's time zone offset. Specifying the **to** symbol as a boolean has the same effect as specifying **auto** as its value.

ts=ip_address_list

This tag specifies the IP addresses of RFC 868 Time Protocol servers.

yd=NIS-domain-name

Specifies the name of the client's NIS domain.

ys=ip_address_list

Specifies the IP address(es) of NIS servers available to the client. Servers should be listed in order of preference.

Vnnn=generic-data

This is a generic tag for vendor specific information where *nnn* is a vendor-defined option field tag number. The generic-data data can be represented as either a stream of hexadecimal numbers or as a quoted string of ASCII characters. The length of the generic data is automatically determined and inserted into the vendor specific field of the RFC 1541-style boot reply.

xd=ip_address_list

This tag specifies the IP addresses of systems that are running the X Window System Display Manager and are available to the client. Addresses should be listed in order of preference.

xf=ip_address_list

This tag specifies the IP addresses of X Window System font servers available to the client. Servers should be listed in order of preference.

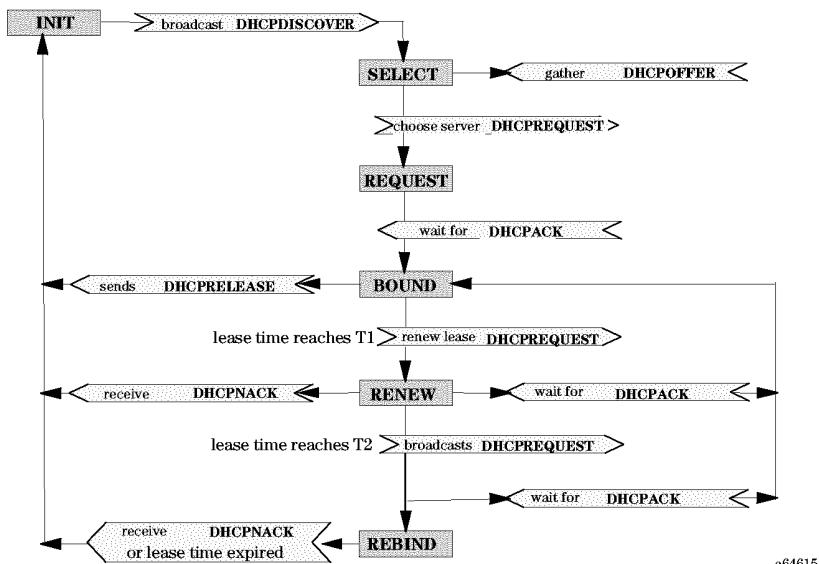
T144=filename

Specifies a configuration file name, which can include, for instance, SNMP-related parameters or an access list for remote systems. The name must be enclosed in quotation marks. This is optional. If it exists, *tftpd* transfers it to the network-based printer.

For more tags and details see man **bootpd(1M)**.

8-5. SLIDE: States of a DHCP Client

States of a DHCP Client



a646151

Student Notes

As its name implies, DHCP provides for the dynamic assignment of IP addresses.

When a client asks for an IP address using DHCP, it steps through several states. Each state transition is caused by a message to or from a server.

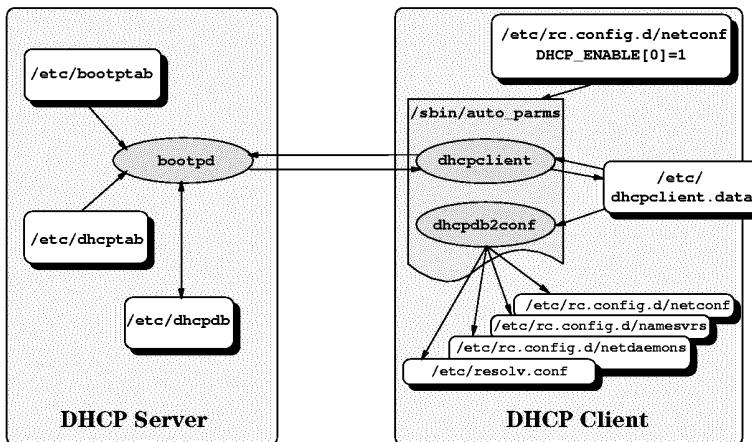
INIT state When the client boots, it broadcasts a **DHCPDISCOVER** request for acquiring an IP address. Within the **DHCPDISCOVER** request the client usually includes suggestions for a certain IP address and parameters which it received from a former **REQUEST**. An HP-UX DHCP client reads these parameters from the **/etc/dhcpclient.data**. After the broadcast the client changes to the **SELECT** state.

SELECT state The DHCP client waits for one or more **DHCPOFFER** responses. Among several responses the client selects one offer, which is confirmed with a **DHCPREQUEST**. Waiting for an acknowledgement, the client switches to the **REQUEST** state.

REQUEST state	The requested server leases the IP address to the client for a defined lease time and acknowledges it to the client with a DHCPACK .
BOUND state	After the acknowledgment, the client is in the BOUND state. Clients with a permanent mass storage can store the IP address in order to request the same address next boot. In the BOUND state a timer starts. When the timer reaches the time T1, the client sends a further DHCPREQUEST for extending the lease time. T1 default is 50% of the lease time. The client is in the RENEW state.
RENEW state	If the DHCPREQUEST is acknowledged with a DHCPACK by the server the client changes to the BOUND state again. After reaching T2, the client requests DHCPREQUEST for extending the lease time. Default T2 is 87.5% of the lease time. The client is in the REBIND state.
REBIND state	If the DHCPREQUEST is acknowledged with a DHCPACK by the server, then the client changes to the BOUND state. If the DHCPREQUEST is NOT acknowledged and the lease time is reached, or the server sends a DHCNACK , then the client must shutdown his network interface and return to the INIT state.

8-6. SLIDE: DHCP Architecture on HP-UX

DHCP Architecture on HP-UX



a646152

Student Notes

The `auto_parms` tool lets you configure the system identity and basic configuration parameters. This tool invokes the `dhclient` command, which broadcasts to find a DHCP server. The server, in turn, provides a default set of networking parameters.

`auto_parms` is a system initialization script whose primary responsibility lies in handling first-time boot configuration and ongoing management of the DHCP lease. `auto_parms` is invoked at boot time by the `/sbin/rc` script. Initially, it loads a list of available Ethernet interfaces. For each interface, `auto_parms` consults `/etc/rc.config.d/netconf` and examines the variable `DHCP_ENABLE[index]`. If `DHCP_ENABLE[index]` is set to 0, `auto_parms` does not attempt to request a lease on the interface designated by index.

If `DHCP_ENABLE[index]` does not exist in `/etc/rc.config.d/netconf`, `auto_parms` assumes that it can attempt the DHCP request over the interface. For each interface `auto_parms` runs a command called `/usr/lbin/dhcpclient`. `dhcpclient` contacts the server to get an IP address lease. This command has the ability to broadcast out onto the network prior to the network interface being enabled. The `dhcpclient` also serves as a daemon process that sleeps until the time that it needs to renew the IP address lease. At that time it re-contacts the server where it got the original lease, in order to extend it. The `dhcpclient` command is not intended to be run by users directly.

Once a lease is secured, the information supplied with the lease is used to initialize key networking parameters.

Network parameters provided by the DHCP server are written to the file `/etc/dhcpclient.data`.

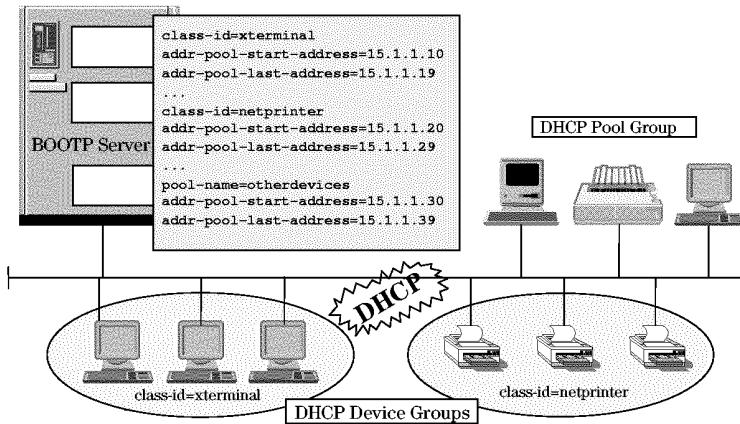
If a client system was initially set up using an IP address that was leased by a DHCP server, that client checks the client database `/etc/dhcpclient.data` to ensure that the lease is still valid at each next boot. The client starts the daemon process `dhcpclient -m` that maintains and renews that lease while the system is running. To ensure stable IP addresses, choose a long or infinite lease time. For all subsequent boots, the data supplied by a DHCP lease is assumed to be definitive and recognized as such by `auto_parms`. Note that in a non-mobile environment where DHCP is being used for IP address management, the lease information will not change from boot to boot under normal conditions. This is accomplished by `auto_parms` ensuring that the `dhcpclient` is placed in *lease maintenance mode* prior to exiting.

If a system cannot contact the DHCP server from which it originally got the IP address lease, it will try to contact other DHCP servers in order to determine if it has been moved to a different network. If this is the case, the system will write a message to the `auto_parms` log file `/etc/auto_parms.log` indicating that it has detected a move to a new subnet and that it is attempting to request a new lease. If the new lease request is successful, new networking configuration values supplied by the DHCP server will automatically be applied.

`bootpd` serves both BOOTP and DHCP. Its configuration files are `/etc/bootptab` and `/etc/dhcptab`. If a DHCP client leases an IP address `bootpd` writes all attributes to `/etc/dhcpdb`.

8-7. SLIDE: DHCP Pool and Device Groups

DHCP Pool and Device Groups



a646153

Student Notes

DHCP allows you to configure pool or device groups, specifying a unique IP address range for each group configured. Each device in a specific group is automatically assigned an available IP address from its group upon requesting booting information.

A pool group defines a range of IP addresses on the same subnet and characteristics of that group. A pool group is independent of the type of device requesting an IP address.

A device group allows different types of clients to receive different parameters from the server. For example, you may want one group to contain nothing except printers, and you may want another group to contain a certain type of terminal. As typical characteristic a device group requires a class ID specifying the client class. DHCP allows you to exclude certain addresses within a group if you wish to make them unavailable for assignment. You also have the capability of defining many values for the devices of a group, including address lease times, DNS servers, NIS servers and many other optional parameters.

8-8. SLIDE: Configuring a DHCP Server

Configuring a DHCP Server

1. Designate DHCP server.
2. Allocate a pool of unused IP addresses.
3. Register hostnames to IP addresses.
4. Add a pool or device group.
 - Choose group name
 - Enter subnet address
 - Enter subnet mask
 - Enter address pool
 - Determ device class (for device groups only)
 - Choose address lease time
 - Fill in additional parameter
5. Enable **bootpd**.

```
# dhcptools -h fip=15.0.0.1 no=10 sm=<subnet_mask> hn=xterm7  
# cat /tmp/dhcphosts >> /etc/hosts  
# sam
```



a646154

Student Notes

To set up a DHCP server you either can use SAM or edit the **/etc/dhcptab** file, and follow the steps below. Note that only one DHCP server per network subnet is required. The main parameters are described below. For additional items see the next slide.

1. Designate a system to act as the DHCP server for your network. This should be a system that is *always* available to its clients.
2. Allocate a set of currently unused IP addresses (preferably a contiguous block of addresses). For example: 15.1.48.50 -15.1.48.80

3. Pre-assign and register hostnames to the IP address allocated above. Using the **-h** option with the **dhcptools** command may be useful. For example, the following line:

```
dhcptools -h fip=15.1.48.50 no=30 sm=255.255.255.0 hn=devlab##
```

This command will create a file **/tmp/dhcphosts** that can be incorporated into your **/etc/hosts** or DNS/NIS database.

4. Use the SAM application to configure the DHCP services on this server. To do this:

- a. Start the interactive SAM application by typing **sam**.
- b. Double click on the icon Networking and Communications.
- c. Double click on the icon Bootable Devices.
- d. Double click on the icon DHCP Device Groups Booting From this Server. You should now see a screen that lists any DHCP groups already defined. There may not be any if DHCP is not already configured.
- e. To add the new group of IP addresses which you allocated click on the Action menu item and choose Add DHCP Group. This should bring up a form with parameters to fill in.
- f. Now you will need to fill in the information on this screen. Some information may require additional research if you are not familiar with the terms or with your network.

Group Name: This can be any name that isn't already defined as a DHCP group—for example **mygroup**.

Subnet Address: This is the portion of an IP address that is not masked off by the subnet mask. If you don't want to figure this out, then just enter one of the IP addresses in the range you picked along with the correct subnet mask and SAM will take care of the calculation. For example: 15.1.48.50.

Subnet Mask: This depends on the *class* of your network, and basically determines how an IP address is separated into a network number and a host specific number. Press F1 in this field for more information. For example: 255.255.255.0.

Subnet Address Pool: Press this button to select the range of IP addresses that you allocated in Step 2. A new screen will be displayed where you can enter the Start and End address. If there are addresses within the range that you picked that you do not want allocated via DHCP, you can use the Reserved Addresses button to specify those (or ranges of them).

Allow Any Device Class:	The SAM default is to allow any type of DHCP device to use the group of IP address you are configuring. This may be undesirable if you use a different method, a different DHCP server, or group for managing systems such as PCs. If you want this range of addresses to be used only by HP-UX systems, then unselect this button, and enter the text: "HewlettPackard.HP-UX" in the text field provided.
Automatic Allocation to Bootp Clients	Leave this option disabled. Enabling it will cause problems for bootp devices such as printers and terminals which rely only on their preconfigured server to respond to their boot request.
Accept New Clients:	Leave this option enabled.
Address Lease Time:	The lease time should be set sufficiently long so that if a client system is temporarily out of service (off) for a time, its lease will not expire too soon. Infinite leases will never expire and disable the IP-address reclamation features of DHCP. For example: 2 weeks.
Boot file name:	You can leave this field blank.
Additional Parameters	There are many parameters that can be specified in this screen for such things as the default routers, time server, DNS server, and NIS domain. You can specify as much or as little as you like in this area. Only the more basic parameters are actually used by installation and boot configuration tools.
Callback Routines:	None is necessary.
<p>g. Once the parameters are all filled in, then press OK on the Add DHCP Group screen. SAM will then make the modifications to the /etc/dhcptab file.</p> <p>h. You will now want to use the Action menu to Enable boot Server (if it is not already enabled).</p> <p>5. Now, new systems that are installed with HP-UX 10.20 (or newer) or booted with a pre-installed 10.20 (or newer) version of HP-UX should contact this server to get an IP address lease and other network information provided by the server.</p>	

8-9. SLIDE: The dhcptab Configuration File

The dhcptab Configuration File

```
dhcp_pool_group:\n    pool-name=otherdevices:\\\n        addr-pool-start-address=15.1.1.30:\\\\n        addr-pool-last-address=15.1.1.39:\\\\n        subnet-mask=255.255.255.0:\\\\n        lease-time=1800:\\\\n        lease-policy=accept-new-clients:\\\\n        allow-bootp-clients=TRUE:\\\\n        hn:\\\\n            gw=15.1.1.254:\\\\n            sr=192.6.126.0 15.1.1.253:\\\\n            tr=55:\\\\n            tv=90\n\ndhcp_device_group:\\n    class-id=xterminal:\\n        addr-pool-start-address=15.1.1.10:\\\\n        addr-pool-last-address=15.1.1.19:\\\\n        subnet-mask=255.255.255.0
```

/etc/dhcptab

a646155

Student Notes

The **bootpd** provides BOOTP as well as DHCP. For backwards compatibility, the **bootpd** reads two configuration files, **bootptab** and **dhcptab**.

The **bootptab** file contains configuration information for old BOOTP clients as well as DHCP clients with fixed IP addresses. The **bootptab** file also contains configuration for relay agents. The **dhcptab** file contains configuration information for DHCP groups, where clients are assigned IP addresses from a pool of currently unused addresses.

You can manually edit both files if desired, although most of your work will probably be performed using SAM.

The configuration file **/etc/dhcptab** has a format similar to the **/etc/bootptab** file, with a keyword followed by one or more tag symbols. These tag symbols are separated by colons (:). The general format is:

keyword: tg=value:...: tg=value:...: tg=value:...

where *keyword* is one of four allowed (non-case-sensitive) symbols and *tg* is a two or more (case-sensitive) character tag symbol. Most tags must be followed by an equals-sign and a value as above. Some can also appear in a boolean form with no value.

Keyword entries are separated from one another by newlines; a single host entry may be extended over multiple lines if each continued line ends with a backslash (\).

The four currently recognized keywords are:

`dhcp_pool_group`

This keyword is followed by tags defining a group of IP addresses to give out to clients on the same subnet, and the characteristics of that group. In addition to the tags defined for DHCP groups, all of the two-letter tags for `bootp` entries may also be used (except for `ht`, the hardware type tag, `ha`, the hardware address tag, or `ci`, the client ID tag).

Required tags are:

```
subnet-mask  
addr-pool-start-address  
addr-pool-last-address
```

`dhcp_device_group`

This keyword is used to define a group of IP addresses on a subnet much like `dhcp_pool_group`, but with one exception: all clients in a device group must have the same client class specified by the tag `class-id`. This allows different types of clients to receive different parameters from the server.

Required tags are:

```
class-id  
subnet-mask  
addr-pool-start-address  
addr-pool-last-address
```

`dhcp_default_client_settings`

This keyword is followed by tags to be applied to all groups. These tag values can be overridden for a specific group if that tag is defined for that specific group. This keyword simply saves one from entering the same tag for every group. Thus, most tags that may be used for

```
dhcp_pool_group
dhcp_device_group
```

also may be used here.

The tag descriptions specify if a tag may not be used here.

`dhcp_server_setting`

This keyword is followed by tags that specify a few general behaviors for the `dhcp` server as a whole.

The currently supported tags for `dhcp_server_settings`:

- *call-on-unrequited=filename* This tag specifies an executable file filename that will be called when the server receives a request to which it cannot send a response. Certain arguments will be passed in; the call executed will be:
- *filename client-id htype haddr [gateway]* where client-ID is the client ID in hex if present, or 00 if there is no client ID. `htype` is the hardware type as per the ARP section of the *Assigned Numbers* RFC. `haddr` is the hardware address in hex. `gateway` is the IP address of the `bootp` relay agent. If the packet was not relayed, then this field is absent.

The currently supported tags for `dhcp_pool_group`, `dhcp_device_group`, and `dhcp_default_client_settings`:

- *class-name=classname* Administrative name only to refer to a device group. It is only applicable to `dhcp_device_group`. The only use that `bootpd` makes of this field is in logging errors found in the configuration of the group.
- *pool-name=poolname* This tag specifies a name to refer to a pool group. It is only applicable to `dhcp_pool_group`. The only use that `bootpd` makes of this field is in logging errors found in the configuration of the group.
- *class-id=client-class* This tag specifies the client-class that clients must have to be assigned to this group. This tag is required for `dhcp_device_group` and is inappropriate for any other keyword. Some DHCP clients send out a client-class that identifies a class that a client belongs to. For an IP address to be assigned from a device group address pool, not only must the client be on the right subnet, it must send a request with a client-class that matches that defined for the class-id. An example for a client-class is `xterminal`.

- *subnet-mask=mask*

This tag specifies the subnet mask for the addresses in the group being defined. It is specified as an IP address.

- *addr-pool-start-address=IP-address* This tag specifies the lowest address in the pool group to be assigned. This tag is required for both `dhcp_device_group` and `dhcp_pool_group`.
- *addr-pool-last-address=ip-address* This tag specifies the highest address in the pool group to be assigned. For the server, no two group address ranges may overlap.
- *reserved-for-other=ip-address-list* This tag is followed by one address that falls in the range of the group. This address is reserved, and will not be assigned to any clients by the DHCP server. This tag may be repeated to reserve more addresses in the same group.
- *lease-time=seconds* This tag specifies the time in seconds that a lease should be given to each client. The word *infinite* may be used to specify leases that never expire. The default is *infinite*. Note that if a client asks for a shorter lease than is configured for it, it will get that shorter lease time.
- *lease-grace-period=percent* This tag specifies the time after a lease expires during which that lease will not be assigned to a new client. *percent* is the percentage of the configured lease time that this grace period lasts. The default is 5%.
- *tr=seconds* This tag specifies the DHCP IP lease renewal time (T1). This is the time interval from lease assignment until the client attempts to renew the lease. RFC 1541 states that T1 defaults to half the lease duration.
- *tv=seconds* This tag specifies the DHCP IP lease *rebind* time (T2). This is the time interval from lease assignment until when the client attempts to obtain a new lease from any server. RFC 1541 states that T2 defaults to 0.875 times the lease duration.
- *lease-policy=policy* This tag specifies whether or not the assigning of new leases can be done. If policy is set to `reject-new-clients` then no new clients can get a lease, and only clients with existing leases will get a response. The default is `accept-new-clients`.
- *allow-bootp-clients=boolean*

This tag specifies whether or not `bootp` clients can be members of the group being defined. The default is FALSE. If boolean is TRUE, then an IP address may be assigned to a client that doesn't have an entry in the `bootptab` file and that is on the same subnet as the group being defined. This address is treated as an infinite lease, and a boot reply is sent to the client. This tag is not appropriate for `dhcp_device_group`, since `bootp` don't have a client class.

- *call-on-assignment=filename* This tag specifies the fully qualified filename to be called when an IP address has been assigned to a new client.

The following calls will be made:

- `filename client-id htype haddr ipaddr subnet-mask lease-expiration [hostname]`
where `client-id` is the client ID in hex if present, or 00 if there is no client ID.
- `htype` is the hardware type as per the ARP section of the *Assigned Numbers* RFC. `haddr` is the hardware address in hex. `ipaddr` is the IP address of the subnet mask for the client represented as an IP address.
- `lease-expiration` is the `bootpd` internal representation of when the lease will expire, whose value is `ffffffffff`, which represents an infinite lease. If there is a hostname associated with this address, then it is the final argument.
- `call-on-decline=filename` This tag specifies the fully qualified filename to be called when an IP address has been declined by a new client.

The following calls will be made:

- `filename client-id htype haddr ipaddr subnet-mask`
`call-on-release=filename` This tag specifies the fully qualified filename to be called when an IP address has been released by a client.

The following calls will be made:

- `filename: client-id htype haddr ipaddr lease-expiration`
`call-on-lease-extend=filename` This tag specifies the fully qualified filename to be called when an IP address lease for a client has been extended.

The following calls will be made:

- `filename: client-id htype haddr ipaddr subnet-mask lease-expiration`

Other provided tags are the same as for BOOTP clients and described above.

8-10. SLIDE: Configuring an HP-UX DHCP Client

Configuring an HP-UX DHCP Client

- Use sam
 - Networking and Communications
 - Network Interface Cards
 - Action: Modify
 - Enable DHCP
- Reboot

```
# vi /etc/rc.config.d/netconf
INTERFACE_NAME[0]=lan0
IP_ADDRESS[0]=15.1.1.10
SUBNET_MASK[0]=255.255.255.0
BROADCAST_ADDRESS[0]=""
DHCP_ENABLE[0]=1
# reboot
```

a646156

Student Notes

If a system has been set up without using DHCP, but you would like to start using it, the following steps may be taken. There are two methods for enabling DHCP on a system that is not currently using it:

Using SAM to Enable DHCP

The first method is to use SAM.

1. As root, run SAM.
2. Double click on Networking and Communications.
3. Double click on Network Interface Cards.
4. Highlight the card on which you wish to enable DHCP. Go to the Actions pull-down menu and select Configure.

5. Single click on the Enable DHCP button.

6. Single click on OK and exit SAM.

Your system will now begin using DHCP after the next reboot.

NOTE: All the current networking parameters will be overridden with new values supplied by the DHCP server.

If for some reason the system cannot contact a DHCP server during the next reboot, it will continue to use its current networking parameters. If you suspect that your system had a problem contacting the DHCP server, you can examine the `auto_parms` log file (`/etc/auto_parms.log`) to determine if the lease request was successful. The second method for enabling DHCP over a particular network interface is to use a text editor to edit the `/etc/rc.config.d/netconf` file. In the header of this file, you will find some brief instructions regarding a variable named `DHCP_ENABLE`. This variable is tied by an index number to an individual network interface. Consider the following block:

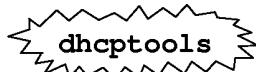
```
INTERFACE_NAME [0]=lan0
IP_ADDRESS [0]=15.1.50.76
SUBNET_MASK [0]=255.255.248.0
BROADCAST_ADDRESS [0]=""
DHCP_ENABLE [0]=1
```

Here, the variables are instructing the system to use the `lan0` interface when attempting to contact a DHCP server. Similarly, if the lease request is successful, the above `IP_ADDRESS` variable would be updated to reflect the new value supplied by the DHCP server. If the `DHCP_ENABLE` variable was set to 0 or if the variable did not exist, no DHCP operations would be attempted over the corresponding network interface. If the variable `DHCP_ENABLE` does not exist for a particular interface, the SAM tool will display a grayed out DHCP enable button. Correspondingly, you could disable DHCP over a particular interface by setting the variable to 0 . Again, as in the first method, the system will only begin using DHCP after the next reboot.

8-11. SLIDE: DHCP Troubleshooting

DHCP Troubleshooting

- dump bootpd data
`dhcptools -d`
- generate hostfile
`dhcptools -h fip=15.0.0.1 no=10 sm=255.0.0.0 hn=name?`
- preview address assignment
`dhcptools -p ht=ether ha=0800091A2B3C sn=15.0.0.0`
- reclaim a client's IP address
`dhcptools -r ip=15.0.0.1 ht=ether ha=0800091A2B3C`
- start packet tracing
`dhcptools -t ct=50`
- stop packet tracing
`dhcptools -t ct=0`
- validate configuration files
`dhcptools -v`



a646157

Student Notes

A command line tool known as `dhcptools(1M)` is available to provide access to DHCP-related options for the `bootpd` server.

The `dhcptools` options provide control for the following tasks:

- dumping internal data structures
- generating a hosts file
- previewing client address assignment
- reclaiming unused addresses
- tracing packets
- validating configuration files

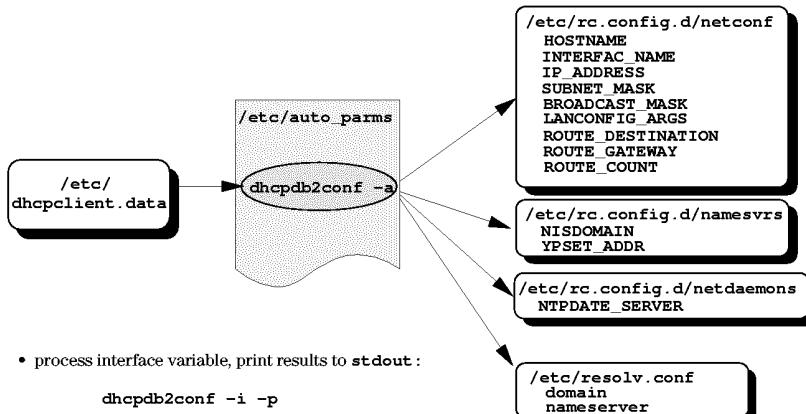
Options:

- d Dump internal **bootpd** data to output files
`/tmp/dhcp.dump.bootptab` containing fixed address clients
`/tmp/dhcp.dump.dhcptab` containing global and group configuration
`/tmp/dhcp.dump.other` containing miscellaneous **bootpd** internal data.
- h Generate hosts file in `/etc/hosts` format.
The output file is `/tmp/dhcphosts`.
The file can be incorporated into a name database in advance of **bootpd** server activation so that the server can automatically allocate a hostname along with an IP address to a DHCP client.
The hostname of the first entry is derived from the `hostname_template`. Each subsequent host entry contains a unique IP address and `hostname` derived from the `first_IP_address`, `subnet_mask`, and `hostname_template`. The wildcards permitted in the `hostname_template` are `*#?`. The `*` means to use a character selected sequentially from the range `[a-z, 0-9]`. The `#` means to use a digit selected sequentially from the range `[0-9]`. The `?` means to use a letter selected sequentially from the range `[a-z]`. A maximum of 3 wildcards can be specified. If a `domain_name` is specified, it will be appended to the hostname. The maximum `number_of_entries_to_generate` is 1000.
- p Preview a client's address assignment based on current conditions for the **bootpd** server to standard output. The `subnet-identifier` tells **bootpd** the subnet for which the client is requesting an IP address. Optionally, the user may request a specific IP address and lease duration using the parameters `lease-time` and `requested-IP-address`.
- P Preview a client's address assignment based on current conditions for the **bootpd** server. This option is the same as -p except that the client is identified by a unique `client-identifier`.
- r Reclaim a client's IP address for re-use by the **bootpd** server. This option is intended for limited use by the **bootpd** administrator to return an allocated but unused IP address to a DHCP allocation pool. The option may be useful to clear the **bootpd** database of old entries (for example, clients retired from service while holding an unexpired IP address lease). Do not reclaim an address that belongs to an active client. The `IP_address`, `hardware_address`, and `hardware_type` can be obtained from the **bootpd** database file.
- t Establish packet tracing for **bootpd**. This will trace the inbound and outbound BOOTP/DHCP packets for the local **bootpd** server. The output file is `/tmp/dhcptrace`. The packet trace count can be a value from 0 to 100. To query the current count, use `dhcptools -t`. To turn off packet tracing use `dhcptools -t ct=0`.
- v Validate **bootpd** configuration files. The default configuration files that will be validated are `/etc/bootptab` and `/etc/dhcptab`. When a `bootptabfile` or `dhcptabfile` is specified, the full pathname is required. The output file for validate is `/tmp/dhcpvalidate`.

For more information see man pages `dhcptools(1M)`.

8-12. SLIDE: Updating Network Configuration

Updating Network Configuration



a646158

Student Notes

`dhcddb2conf` provides a means of translating a client DHCP database into a set of standard configuration file variables. DHCP database is found in `/etc/dhcpclient.data` and set by `/usr/lbin/dhcpcclient`. A DHCP client database can contain settings for such items as IP address, hostname, and default gateway. Settings depend on the information sent by the DHCP server.

Using `dhcddb2conf`, you can do the following:

- list the contents of the database to the screen
- create a set of configuration staging files
- execute direct edits on existing configuration files using the values contained in the client database.

The files and variables that can be processed are the following:

```
/etc/rc.config.d/netconf

    HOSTNAME
    INTERFACE_NAME [index]
    IP_ADDRESS [index]
    SUBNET_MASK [index]
    BROADCAST_MASK [index]
    LANCONFIG_ARGS [index]
    ROUTE_DESTINATION [index]
    ROUTE_GATEWAY [index]
    ROUTE_COUNT [index]

/etc/rc.config.d/namesvrs

    NISDOMAIN
    YPSET_ADDR

/etc/rc.config.d/netdaemons

    NTPDATE_SERVER

/etc/resolv.conf
    domain
    nameserver
```

The options have the following meanings:

- a Using the results of the specified filter, directly apply the variable definitions to the existing configuration files.
- c Create a set of staging files using the results of the selected filter(s). Each variable processed will be applied to its corresponding configuration file. Specifically, `dhcpdb2conf` will generate a copy of the existing configuration file. As an example, `/etc/rc.config.d/netconf` will be copied to `/etc/rc.config.d/netconf.dhcp`. Once this staging file has been created, the variable that is being processed will be applied to the newly created file. Be careful: Using the -c option will override any existing values that are currently set in the system's configuration files.
- d Process the *DNS* variable set.
- h Process *hostname*
- i Process the *interface* variable set.
- n Process the *NIS* variable set.
- p Print results to the screen (`stdout`), which is the default action if neither -c or -a are specified.
- r Process the *route* variable set.
- s Specify the variable set index.

-t Process *ntpdate_server*.

Typically **dhcpdb2conf** is called from **auto_parms** with the **-a** option.

8-13. LAB: DHCP Configuration

Directions

For each subnet, configure a DHCP server.

Subtasks:

- Configure the server (with SAM)
- Configure one client to use a static address (with SAM).
- Configure the other clients to use dynamically assigned addresses (with SAM and `dhcptools`).

All other remaining workstations are DHCP clients.

Configure your DHCP server with the following attributes:

For the answers to this lab we will use our classroom layout. In our example `paloalto` is the DHCP server for the subnet 150.150.192.0. Use `dallas` for the fixed address assignment.

For each subnet determine the DHCP server. All other remaining workstations are DHCP clients.

Configure your DHCP server with attributes requested in the following questions:

1. For one of the DHCP clients use a fixed, pre-assigned IP address. Determine the hardware address of this client.
2. Use SAM for configuring this client on your boot server. What information do you need to provide?
3. Which configuration file has SAM updated ?

4. Look at the newly created entry in the configuration file. What are the tags for the following?

Host Name:
Internet Address:
Subnet Mask:
Host Identification Method:
Station Address:
Adapter Device Type:

Default Router:
Static Router:
Broadcast Address:
Send Hostname to Device:

5. For the remaining clients configure a DHCP pool. Again use SAM to fill in information below:

Group Name:
Subnet Address:
Subnet Mask:
Subnet Address Pool:
Lease Time:

Default Router:
Static Router:
Broadcast Address:
Send Hostname to Device:

Renewal Time T1:
Rebind Time T2:

6. Which configuration file has SAM updated this time?

7. Look at the newly created entry in the configuration file. What are the tags for in the following?

Group Name:
Subnet Address:
Subnet Mask:
Subnet Address Pool:
Lease Time:

Default Router
Static Router:
Broadcast Address:
Send Hostname to Device:

Renewal Time T1:
Rebind Time T2:

8. Using **dhcptools** dynamically, create new hostnames in the form of

clienta, clientb, clientc ...

In which file are the names created?

Append this file to your existing **/etc/hosts** file.

9. Using SAM, enable your **bootpd** daemon.

10. Using **dhcptools**, preview a client's address assignment

(a) for the fixed IP assignment (b) for any other hardware address.

11. Start a trace for `bootpd`.

12. On the client enable DHCP and reboot the client.

13. Does your configuration work correctly?

Stop the trace on the DHCP server.

In which file do you find the traced packets?

Do you understand the output?

14. On the DHCP client, determine the values that the client has gotten from the DHCP server.

Be sure to check the appropriate variables in the configuration files. Are they all correct?

Module 9 — Configuring and Maintaining the Domain Name Service

Objectives

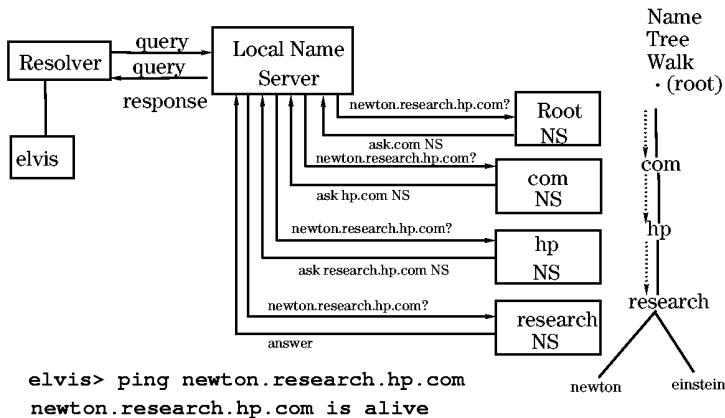
Upon completion of this module you will be able to do the following:

- Describe DNS (Domain Name Service) features and operation.
- Explain how DNS resolves hostnames into IP addresses, and vice versa (reverse name resolution).
- Describe in detail how DNS data files are built and maintained.
- Set up a complete Internet domain.

9-1. SLIDE: Hostname Resolution

Hostname Resolution

Resolution of name `newton.research.hp.com` from machine `elvis`



a64659

Student Notes

Resolver is the interface that clients use to access the name service.

Two types of queries are supported by name servers:

- *Recursive*, where a single name server handles the complete query on behalf of a client (which could be resolvers or other name servers). If a recursive query is requested by a client, it is the responsibility of the name server to return an answer.
- *Iterative*, where a name server queries another, gets a response containing either the answer or information about other name servers.

In the example, the resolver requests a recursive query from the local name server and the local name server requests iterative queries from all other name servers starting from the root domain.

9-2. SLIDE: Reverse Name Resolution

Reverse Name Resolution

- Certain programs (such as **rlogind**) require *Reverse Address Resolution*--that is, for security reasons, address and name need to match or possible *SPOOF*
 - Given an address, the hostname corresponding to the address needs to be determined
- Reverse Name Resolution is accomplished by treating the address as a name and fitting it into the domain tree using a fictitious top-level domain
 - The top level domain **arpa** and the first level domain **in-addr** are used for reverse name resolution
- The IP address is inverted to fit the tree: the host part (last octet) appears as the leaf node of the tree
 - Compare with names: rightmost octet of an IP address corresponds to the host whereas in a fully qualified domain name, the leftmost component is the host

a646128

Student Notes

Example:

```
newton.research.hp.com    # hostname  
130.207.5.61            # IP address
```

Reverse Address Resolution to fit the DNS tree:

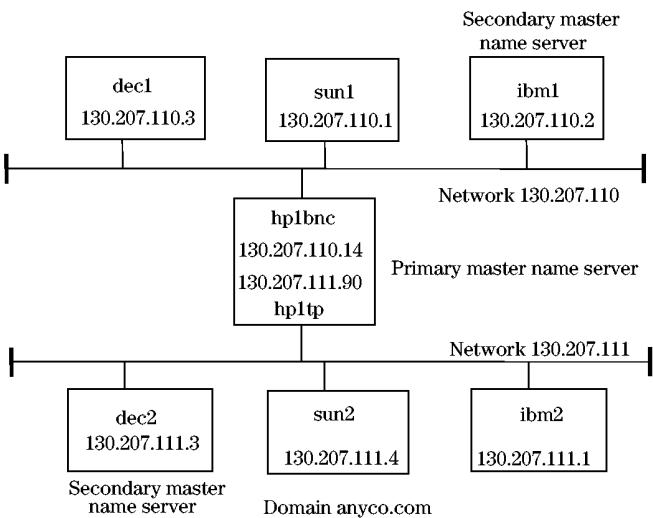
```
61.5.207.130.in-addr.arpa
```

Nodes in the special Internet **in-addr.arpa** domain, used for reverse name resolution, are named after numbers in the dotted octet representation of IP addresses.

Every subdomain under **in-addr.arpa** can have up to 256 subdomains, which includes the set of possible IP addresses.

9-3. SLIDE: Overview of DNS Configuration

Overview of DNS Configuration



a64660

Student Notes

Before DNS is setup, the network managed by the DNS domain must be identified and hosts acting as primaries and secondaries must be defined. The figure shows the network layout in a sample domain called `anyco.com`.

The figure shows a network with one primary master name server and two secondary master name servers.

At this point, the administrator must also set up a complete host file (if it doesn't exist already) on the primary. It must include all the local names, and their IP addresses, that this server will be responsible for.

NOTE:

There will only be one primary server for a domain, but there may be multiple secondaries—usually designated as one per segment. A secondary can be used to reduce the load of a primary and to carry on if a primary is down. It can do anything a primary can do, except update other secondaries.

Following is a sample file for our network:

```
127.0.0.1      localhost

# Network 130.207.110

130.207.110.1  sun1
130.207.110.2  ibm1
130.207.110.3  dec1

# Network 130.207.111

130.207.111.1  ibm2
130.207.111.3  dec2
130.207.111.4  sun2

# Gateway machine with two interfaces

130.207.110.14 hp1 hp1bnc
130.207.111.19 hp1 hp1tp
```

Note that the gateway machine is a multihomed machine—that is, it has more than one interface. DNS expects all the interfaces to have the same hostname and to use an alias to differentiate among them.

The administrator must also know what the parent domain is, and where the root name servers are.

A consideration to be faced is what visibility to give users in your domain. If only the Internet root name servers are available, you must rely on your connection to the Internet. An attractive alternative is to establish internal root servers, so that users can obtain complete information on the internal name space structure without dependence on the Internet. They also afford scalability when growing into a large network.

9-4. SLIDE: Standard Resource Record Format

Standard Resource Record Format

- The format of DNS data files is standardized for UNIX platforms.
- Each data file requires a record of type *start of authority* (SOA).
 - SOA indicates that this name server is authoritative for the zone.
 - There can be one and only one SOA in a data file for a zone.
- Each line of a data file is called a *resource record* (RR).
- Most resource records have the current zone (*origin*) appended to their names, if they are not terminated by a ":".
- Resource records contain the following fields, separated by a white space
[name] [ttl] class Record Type Record Specific Data
- Example: an address record

```
sun1      999999      IN      A      130.207.110.1
```

a64661

Student Notes

DNS data files are standardized by the standard resource record (RR) format.

There is no difference among the several UNIX® platforms, so one can build up DNS data files on one system and copy them to the others.

The first entry in a data file is the start of authority (SOA) record. SOA indicates that this name server is the authoritative name server for the *zone* indicated (the name field).

The *zone* is a portion of the domain name space, with a subtlety to be noted. A zone contains data and domain names that are contained in a domain, except for data and domain names that are delegated to another name server. For example, the top level domain **com** contains subdomains **hp** and **sun**, among others. Authority for **hp** and **sun** reside within those subdomains. The domain **com** contains all the data in **com**, **hp**, and **sun**. The zone **com** contains only the data in **com**.

A sample SOA record follows. The character ; starts a comment:

```
anyco.com. 604800 IN SOA hostname person_in_charge (
    132      ; serial number of revision
    4000     ; refresh: how often to check with primary
    600      ; retry period in case refresh failed
    36000    ; expiration time for this data
    604800) ; default number of seconds to be used for
              minimum time to live interval
```

hostname indicates the host where this data was created. *person_in_charge* is the e-mail address of a person responsible for administering this domain.

For the RR record, fields within brackets [] are optional.

<i>name</i>	Name of the domain. If it is terminated with a dot, it is a fully qualified name. Otherwise, it is considered a relative name and the current domain is appended to it
<i>ttl</i>	Time-to-live (in seconds) for this data
<i>class</i>	Record class. Currently, only one class is used—IN. It is used for any TCP/IP-based Internet.

9-5. SLIDE: DNS Record Types

DNS Record Types

DNS supports several record types:

A	Name to IP address mapping
CNAME	Alias to canonical name
HINFO	Information about host
NS	Domain name to name server
MX	Mail address to mail exchange address
PTR	IP address to host name mapping
SOA	Start of authority information

a64662

Student Notes

Most entries in db files are called DNS resource records. Meanings of various fields in the resource record are:

<i>Record Type</i>	<i>Description</i>
A	Record contains a name followed by an address.
CNAME	Record contains a name followed by an alias (canonical name). When the name is referenced, the alias address is returned.
HINFO	Record contains a host name followed by the host's hardware type and the operating system.
NS	Record contains a domain name, followed by the name of a server that contains detailed information about the domain.

MX	Record contains a domain name followed by a mail exchange server name. The server either sends mail directly to a recipient, forwards it to another server closer to the final destination, or gateways it to another mail transport such as UUCP.
PTR	Record contains an IP address, followed by a hostname. This provides pointer information for reverse address resolution.
SOA	Record contains a domain name, followed by the primary name server for authoritative information about the domain. Subsequent fields give the mail address of the person in charge of the data, and various parameters to define polling interval, etc., for secondary name servers. At least one secondary name server should be employed for a reasonably robust mail system.

In addition, a resource record may contain an entry of the form

\$ORIGIN <domainname>

which changes the origin for a relative domain name (what gets appended to a non-fully-qualified name) to the defined domain name.

These entries may also contain some special characters:

- . free-standing dot in name field refers to the current domain name
- @ free-standing @ in name field refers to the current origin
- ;
starts a comment
- * signifies wild-carding
- () group data that crosses a line

9-6. SLIDE: The db Files

The db Files

- **db.cache**
- **db.127.0.0**
- **db.domain_name** (db.anyco)
- **db.network_number** (db.130.207)

a64663

Student Notes

The first step in setting up a name server is to translate the host table into equivalent DNS data, using four separate db files. These files are described as follows:

<i>File Name</i>	<i>Description</i>
db.cache	This file holds the information on root name servers needed to initialize cache of Internet domain name servers.
db.127.0.0	This file contains the special address that hosts use to direct traffic to themselves. This assures no surprises when lookups to 127.0.0.1 are used.
db.domain_name	This file maps hostnames to addresses. In the domain hp.com , this file would be called db.hp .

db.*network_number*

This file maps addresses to hostnames, where *network_number* is the network address. Thus there is one **db.*network_number*** file for each network address. An example would be **db.15.3.41**.

9-7. SLIDE: The db.cache File

The db.cache File

```
db.cache
; BIND data file for initial cache data for root domain servers.

.
99999999 IN NS ns.nic.ddn.mil.
.
99999999 IN NS ns.nasa.gov.
.
99999999 IN NS erp.umd.edu.
.
99999999 IN NS a.isi.edu.
.
99999999 IN NS aos.brl.mil.
.
99999999 IN NS c.nyser.net.
ns.nic.ddn.mil. 99999999 IN A 192.67.67.53
ns.nasa.gov. 99999999 IN A 128.102.16.10
.
99999999 IN A 192.52.195.10
terp.umd.edu. 99999999 IN A 128.8.10.90
a.isi.edu. 99999999 IN A 26.3.0.103
99999999 IN A 128.9.0.107
aos.brl.mil. 99999999 IN A 128.20.1.2
99999999 IN A 192.5.25.82
c.nyser.net. 99999999 IN A 192.33.4.12
```

a64664

Student Notes

The db.cache file contains name server records (NS) for the root name servers. It must be updated periodically, as the name servers do change from time to time. A good practice would be to update monthly or bi-monthly, by **ftp'ing to ftp.rs.internic.net**.

- Root name servers provide name service for the "." (root) domain.
- The **NS** record must also have a matching **A** record with the IP address of the root servers.

At startup, **named** cycles through the list of root name servers, contacts one of them and gets the up-to-date information about root name servers, if need be.

hosts_to_named creates a db.cache file, which is a template only.

- The template must be edited and the identity of the root name servers must be filled in.

As name servers cannot cache data forever, an appropriate time must be selected. This *time to live*, or TTL, defines when a given cache must be flushed and refreshed. A large TTL will give fast response, but the data is more likely to be out of date. A short TTL will assure good data

consistency, but might give performance problems as caches are frequently refreshed. A TTL of 345600 seconds (4 days) is recommended for top-level domain servers.

9-8. SLIDE: The db.127.0.0 File

The db.127.0.0 File

```
db.127.0.0
@ IN SOA hp1.anyco.com. root.hp1.anyco.com. (
    1           ; Serial
    10800       ; Refresh every 3 hours
    3600        ; Retry every hour
    604800      ; Expire after a week
    86400 )     ; Minimum ttl of 1 day
IN NS hp1.anyco.com.

1 IN PTR localhost.
```

a646129

Student Notes

The db.127.0.0 local host reverse address file is used for IP address-to-host name translation.

If someone looks up 127.0.0.1, they get back the name localhost.

The IP address 127.0.0.1 exists on every host, and no server is assigned to manage the 127 domain. Therefore each name server must manage resolution of the domain 0.0.127.in-addr.arpa. itself.

The loopback address 127.0.0.1 is used for much internal interprocess communication. Many X-clients applications use this facility.

9-9. SLIDE: The db.domain_name File: Name Resolution

The db.domain_name File: Name Resolution

```
db.anyco: Name Resolution Data File
@      IN      SOA    hpl.anyco.com. root.hpl.anyco.com. (
                           1           ; Serial
                         10800       ; Refresh every 3 hours
                           3600        ; Retry every hour
                         604800     ; Expire after a week
                           86400 )    ; Minimum ttl of 1 day
                           IN      NS      hpl.anyco.com.
Localhost IN      A      127.0.0.1
sun1      IN      A      130.207.110.1
solaris   IN      CNAME  sun1
ibm1      IN      A      130.207.110.2
dec1      IN      A      130.207.110.3
hpl       IN      A      130.207.110.14
          IN      A      130.207.111.90
ibm2      IN      A      130.207.111.1
dec2      IN      A      130.207.111.3
sun2      IN      A      130.207.111.4
hp1bnc   IN      A      130.207.110.14
hp1tp    IN      A      130.207.111.90
hp1bnc   IN      CNAME  hp1
hp1tp    IN      CNAME  hp1
```

a646130

Student Notes

The SOA record is obligatory for each data file.

Recall that the @ stands for the current origin. The current origin is the domain name which is written in the `named.boot` file.

The `NS` record type lists by name a server responsible for a given domain. All subdomains are announced by an `NS` entry. Domains above a name server regularly are not announced. In order to switch to another domain or a domain above, the query is always sent to a root name server.

- The format of an `NS` resource record is as shown below:

[name] [ttl] class NS Name-server-name

- The name field lists the domain serviced by the name server. If missing, it defaults to the last name listed.
- One `NS` record must exist for each authoritative server for the domain.

Defining Further Subdomains

- Within `anyco.com` there exists a subdomain `sales` resulting in the complete domain name `sales.anyco.com`. The name server for `sales` is `rob.sales`. In terms of a complete domain name `rob.sales.anyco.com`, `rob` is the name server for `sales.anyco.com`. By the NS entry a subdomain and its name server is defined. In order to address the name server `rob.sales` a second entry A is necessary for name resolution.

The A record lists the address for a given machine.

- The format of an A record is as shown as follows

`[name] [time-to-live] class A ip_address`

Both `name` and `time-to-live` are optional.

- The name field is the machine name and address is the IP address
- One A record must exist for each address of the machine

Multihomed Hosts

Multihomed hosts are machines with two or more interfaces, and this gives certain advantages. When one name server is connected to multiple networks, it can be reached directly by all servers on those networks, whether or not routers are running. Addresses are also sorted by the name server, eliminating the need to use aliases to get the best response for NFS mounts.

9-10. SLIDE: The db.network_number File: Address Resolution

The db.network_number File: Address Resolution

```
db.<network-number>: Address Resolution Data File (db.130.207)
@      IN      SOA    hp1.anyco.com. Root.hp1.anyco.com. (
                      1           ; Serial
                      10800      ; Refresh every 3 hours
                      3600       ; Retry every hour
                      604800     ; Expire after a week
                      86400 )    ; Minimum ttl of 1 day
          IN      NS     hp1.anyco.com.
1.110   IN      PTR    sun1.anyco.com.
          IN      PTR    solaris.anyco.com.
2.110   IN      PTR    ibm1.anyco.com.
3.110   IN      PTR    dec1.anyco.com.
14.110  IN      PTR    hp1.anyco.com.
          IN      PTR    hp1bnc.anyco.com.
90.111  IN      PTR    hp1.anyco.com.
          IN      PTR    hp1tp.anyco.com.
```

a64666

Student Notes

The **SOA** and **NS** records are similar to the name resolution data file.

Note that the origin (@) of this file is 207.130.in-addr.arpa.

The **PTR** record allows addresses to point to hostnames.

Canonical Names and Aliases

Compare the **sun1** record with the name resolution data file.

The record in the name resolution data file is:

```
sun1      IN      A      130.207.110.1
solaris1  IN      CNAME  sun1
```

solaris1 is an alias for **sun1**, whereas **sun1** is the canonical name.

In the reverse name (or address) resolution data file the appropriate entry for 1.110 returns two names:

1.110	IN	PTR	sun1.anyco.com
	IN	PTR	solaris.anyco.com

sun1.anyco.com. is the canonical name and MUST be the first entry.
solaris.anyco.com. is the alias name and is the second entry.

NOTE: Although both names are returned by the reverse name resolution function `gethostbyaddr()`, in most cases only the first, that is the canonical, form is used by application programs. Don't change the order of canonical forms in the data files for name resolution and reverse name resolution respectively.

Multihomed Hosts

Multihomed hosts should have only *one* name. This is absolutely important for network management programs. It also benefits commands like `rlogind`, `lpd`, or `mountd`, because you need not distinguish between both interfaces in the `/etc/hosts.equiv`, `.rhosts`, `/etc/hosts.lpd` or `/etc(exports` files. Therefore both interface IP addresses return the same canonical name. In order to distinguish both interfaces you can add the alias name for both IP addresses as a second entry. Again, in most cases the second entry is ignored.

Below you see the comparison of both name resolution and reverse name resolution.

Name resolution data file:

hp1	IN	A	130.207.110.14
	IN	A	130.207.111.90
hp1bnc	IN	A	130.207.110.14
hp1tp	IN	A	130.207.111.90

Reverse name resolution data file:

```
14.110 IN PTR hp1.anyco.com.  
          hp1bnc.anyco.com.  
90.111 IN PTR hp1.anyco.com.  
          hp1tp.anyco.com.
```

9-11. SLIDE: The MX Record Type

The MX Record Type

```
*.*    IN    MX      5    hp1b.hpl.hp.com.  
IN     MX      10   relay.hp.com.  
IN     MX      10   hp.com.
```

a64667

Student Notes

MX records are added to `db.root` and `db.domainname`, and are used for two main purposes:

1. To arrange that one host *back up* another by receiving mail for it when it is down.
2. To arrange that mail addressed to remote networks be relayed through the appropriate gateways.

Referring to the example below, the first field indicates a single server or a group of servers (by use of the wildcard). The `IN` means Internet (a given) and `MX` indicates the record type. The integer is a preference, the lower number being preferred. This priority eliminates the possibility of looping when a server is unavailable. The trailing dot (.) is important to tell the system NOT to append another `hp.com` to the domain name.

The second record is required because the first wildcard will not match `hp.com` or any other subdomains of `com`.

```
*.*          IN  MX   5    hp1b.hpl.hp.com.  
*.hp.com.    IN  MX   10   hp2b.hpl.hp.com.
```

9-12. SLIDE: The named.boot File

The named.boot File

```
/etc/named.boot:  
;  
; type      domain    host/file    backupfile  
;  
directory /etc/nameserver ; running directory for named  
  
primary   0.0.127.IN-ADDR.ARPA    db.127.0.0  
primary   anyco.com            db.anyco  
primary   207.130.IN-ADDR.ARPA    db.130.207  
cache     .                      db.cache
```

The **named.boot** file is a mechanism for pointing the server to its data files.

a64668

Student Notes

The directory entry specifies a working directory for zone data files.

- All subsequent file names are assumed relative to this directory.

The primary entries relate a domain name with a zone data file and define the server as authoritative for that domain.

- The source file names the actual data files.

The cache entry identifies a file containing names and addresses of authoritative servers for the root domain.

- This provides the initial **bootstrap** information.

The domain named in the primary lines is used as the origin for relative names in the zone files.

As networks become more complex, more organization of directories will make system administration easier. One method is to keep separate primary and secondary directories. The boot file in the slide could be expanded as follows to handle the primary/secondary division of files:

```
directory /etc/nameserver ; running directory for named
;
;non-specific zone files
primary    0.0.127.IN-ADDR.ARPA      db.127.0.0
cache      .
;
;primary zone files
primary    anyco.com                  primary/db.anyco
primary    107.130.IN-ADDR>ARPA       primary/db.130.207
;
;secondary zone files
secondary   otherco.com              151.092.208.15 secondary/bak.otherco
secondary   208.092.151.IN-ADDR.ARPA  151.092.208.15 secondary/bak.151.092.208
```

Other possible types are:

- | | |
|-------------------|--|
| forwarders | Define hosts to which unresolved queries are sent, maintaining a cache.
Forwarders contact other domains directly. |
| slave | Define hosts to which unresolved queries are sent, but which DO NOT contact other domains directly. Queries are sent to forwarders only and no cache is maintained. Note the term slave is no longer used. Instead, the directive option is used: options forward-only . |
| domain | Specify a default domain such as IBM or HP. This directive is placed in the resolv.conf file. |

NOTE:	When using a forwarder, db.cache will have the forwarder name listed, NOT the Internet root servers.
--------------	---

A complete list of directive options follows:

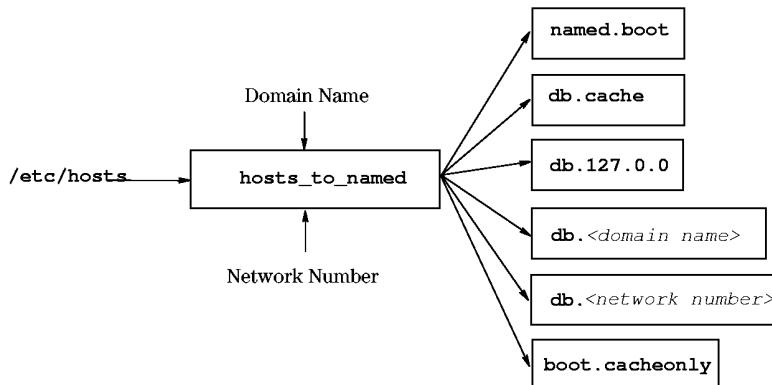
- | | |
|------------------------------|--|
| options forward-only | Prevents the name server from resolving domain names independently of a forwarder. |
| options no-recursion | Prevents the name server from performing recursive resolution of domain names. |
| options no-fetch-glue | Prevents the name server from fetching missing glue when constructing a response. |
| options query-log | Logs all queries received by the name server. |

**options fake-
iquery** Tells the name server to respond to old-fashioned inverse queries with a fake answer instead of an error.

9-13. SLIDE: Building DNS Data Files

Building DNS Data Files

HP-UX provides the script `hosts_to_named` for building DNS data files
`hosts_to_named -d domain -n network_number [options]`



a64669

Student Notes

If you want to set up a BIND name server you can proceed as follows:

1. Create the directory to contain the DNS data files.

```
# mkdir /etc/named.data
```

2. Change the current directory to `/etc/named.data`.

```
# cd /etc/named.data
```

3. Run `hosts_to_named`, to collect information about networks `130.207.110` and `130.207.111`. Note this is called `h2n` in some UNIX systems.

```
hosts_to_named -d anyco.com -n 130.207 -b /etc/named.boot
```

hosts_to_named translates the host table **/etc/hosts** into DNS data files. Therefore it is important that you insure the correct contents of the host table before running **hosts_to_named**.

Some options :

-b /etc/named.boot	Name of the named boot file.
-d domain	Name of the domain.
-n network-number	Network number, without trailing zeros.
-z namesrv-ip-addr	Creates boot file boot.sec.save for secondary server.
-Z namesrv-ip-addr	Creates boot file boot.sec for secondary server.
-r	Creates db.root for an isolated network (that is, a network not connected to the Internet).

You can use the **-n** option more than once if your name server should serve more than one network.

Use the **-d** option more than once if the name server serves more than one domain.

The **-b** option is used to provide the name for the **named** boot file. The file name **/etc/named.boot** is expected by **named** during startup. (This can be overridden by the **-b** option of **named**). If you don't use the **-b** option you have to move the boot file to the default location **/etc** after creation.

If you want to create a secondary server you can give the **-z** or the **-Z** options. Now the secondary server's boot file is created automatically and you simply have to copy it to the appropriate system and rename it to **named.boot**. If you want the secondary server to store the data files on its disk, then you have to use the file **boot.sec.save**. If the data should not be stored locally you have to use **boot.sec**.

With **-r** you create name server data indicating that the name server is authoritative for the root of the domain tree. The file created is **db.root**. Use this only when your network is isolated from the ARPA Internet.

The options are too numerous to list all of them. Please refer to manual pages.

NOTE: Build a config file (**-f** option) to define parameters.

1. This saves retyping when you must reconfigure.
 2. This is a good record of how DNS has been configured.
-

Files created by **hosts_to_named** are:

named.boot	Default boot file for named
boot.cacheonly	Boot file for Cache Only Server
db.cache	Cache data file template
db.127.0.0	Local reverse address file
db.<domainname>	Data file for name resolution
db.<networknumber>	Reverse data file for address resolution

The output generated by running **hosts_to_named** on the **/etc/hosts** file shown before is shown below:

```
# hosts_to_named -d anyco.com -n 130.207
Translating host to lower case ...
Collecting network data ...
130.207
Creating list of multihomed hosts ...
Creating "A" data (name to address mapping) for net 130.207 ...
Creating "PTR" data (address to name mapping) for net 130.207 ...
Creating "MX" (mail exchanger) data ...
Building default named.boot file ...
Building default db.cache file ...

WARNING: db.cache must be filled in with the name(s) and address(es) of
the rootserver(s)
```

Building default boot.cacheonly for caching only servers ... done

The files created are listed below:

```
# ls
boot.cacheonly  db.130.207  db.cache  named.boot
db.127.0.0      db.anyco
```

9-14. SLIDE: Debugging BIND

Debugging BIND

- To stop the `named` process:
`# sig_named kill`
- To examine `named`'s current database:
`# sig_named dump`

This dumps the current database and cache into a disk file:

`/var/tmp/named_dump.db`

- If `named` is started with the debug option or debug level is activated with `sig_named`, it writes debug messages into a disk file
`/var/tmp/named.run`
- Create statistics with the following
`# sig_named stats`
- The file `/var/tmp/named.stats` will be created if needed, or appended to with current information.

a646131

Student Notes

`named` can be started in debug mode using the following command:

`named -d <number>`

The number determines the level of debugging: 1 is least prolific and 7 is most prolific.

The commands available with `sig_named` are

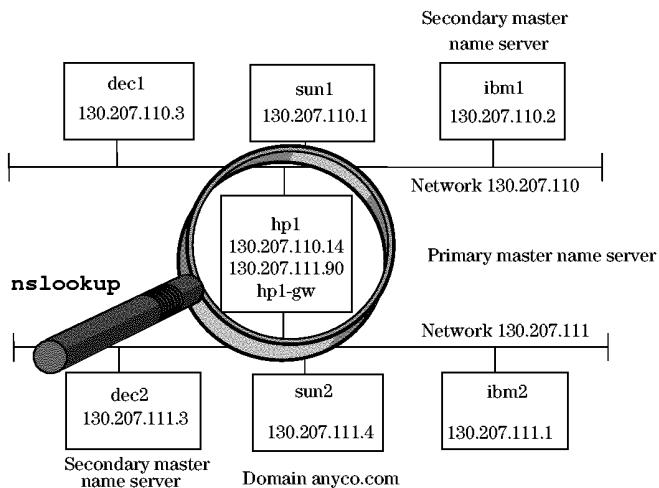
<code>debug</code>	level of debugging can be activated or increased, without restarting <code>named</code>
<code>[+] debug-</code>	
<code>level</code>	
<code>dump</code>	creates a dump of name server data base
<code>kill</code>	used to terminate <code>named</code>
<code>restart</code>	used to restart <code>named</code>

stats

creates statistics

9-15. SLIDE: nslookup Options and Applications

nslookup Options and Applications



a64670

Student Notes

The options for **nslookup** are selected using the **set option** command. The options are as follows:

<i>Option</i>	<i>Description</i>
[no] debug	Debugging is turned off by default. If turned on, timeouts and response packets are displayed.
[no] defname	By default, nslookup adds the default domain name to names without a dot in them. Before search existed, the default domain was only added to names without any dots in them. Now nslookup can implement the pre-search behavior (search off, defname on), or it can implement search behavior (search on).

[no] search	This option overrides the <code>defname</code> option. By default, <code>nslookup</code> appends the domains in the search list to names that do not end in a dot.
[no] recurse	Recursive service is called by default.
[no] d2	Debugging at level is turned off by default. If turned on, query packets are displayed along with the normal debug display.
[no] vc	By default, <code>nslookup</code> makes queries using UDP packets instead of over a virtual circuit (TCP), matching usual BIND resolver behavior.
[no] ignoretc	By default, truncated packets are not ignored, matching normal BIND resolver behavior. If the <i>truncated</i> bit is set, <code>nslookup</code> will retry the query using TCP.
port=53	DNS service is on port 53. Other ports can be used for debugging purposes.
querytype=A	By default, A (address) resource records are looked up. In addition, if you type in an IP address (and the querytype is A or PTR), then the address is inverted, IN-ADDR.ARPA is appended, and pointer data is looked up instead.
class=IN	Internet is the only class of interest.
timeout=5	If the name server does not respond within 5 seconds, the query is resent with a timeout of 10 seconds, then 20, and then 40.
retry=4	Send the query 4 times with the timeouts defined above.
root=a.root-servers.net	The <code>root</code> command, under <code>nslookup</code> , switches the default server to <code>a.root-servers.net</code> . Change the default <code>root</code> server with <code>set root=server</code> .
domain=mgr.hp.com	The default domain when the <code>defname</code> option is on.
srchlist=mgr.hp.com	If <code>search</code> is on, this domain is appended to names not ending in a dot.

Note that `nslookup` can be directed to query another server by using the `server` command (`server anotherserver`). The command `lserver` can return to the original default server.

9-16. SLIDE: Examining Query and Response Packets

Examining Query and Response Packets

```
# nslookup
      Default name server:    F00.hp.com
      Address:                10.1.2.3
> set debug
> node1.hp.com
```

The system will respond with details and if possible the IP address of this node.

a64671

Student Notes

The slide shows the coding of responses from a queried server. If the matching queries are to be displayed, add the command `set d2`.

Displays of query-response sets follow the format below:

Header Section

The caption *HEADER:* is followed by a series of parameters;

- *opcode* = QUERY
- *id* = *unique_integer_identifier*
- *rcode* = *response_code*, which would be one of:
 - no error

- **server failure**
- **name error**
- **not implemented**
- **refused**
- *header flags*: consisting of the following:
 - **response or query**
 - **auth. answer**—response is authoritative, not from cache
 - **want recursion**—let the name server do all the work
 - **recursion avail.**—the server can handle the recursion tasks

The last four entries are counters, indicating how many records are in each of the next four sections.

Question Section

The caption *QUESTION*: (there is always only one question per query) is followed by *queried_server, type = A, class = IN*.

Answer Section

ANSWERS: followed by one or more answers of the form *queried_server, Internet address = internet_addr*

Authority Section

AUTHORITY RECORDS: followed by the servers and addresses providing authoritative answers, when required.

Additional Section

ADDITIONAL RECORDS: followed by additional records as required, such as addresses to authoritative servers.

9-17. SLIDE: Simulating the Resolver with nslookup

Simulating the Resolver with nslookup

```
# nslookup
Default name server:      default_domain
Address:      default_server_address
> set norecurse
> set nosearch
> queried_name_server
> queried name
```

a64672

Student Notes

It is sometimes helpful to send out a query packet to test configuration in a manner independent of the normal BIND name server query. `nslookup` can be made to send out the same query packet to simulate a BIND name server. Keep in mind that a name server does not do recursion or implement searches. Thus `set norecurse` (shorthand `set norec`) and `set nosearch` (shorthand `set nosea`) must remove these defaults from the `nslookup` operation. Other than these differences, the query is a normal `nslookup` query.

An example of an `nslookup` query follows:

```
# nslookup - relay.hp.com
Default Name Server:  relay.hp.com
Address:  15.255.152.2

> set norec
> set nosearch
> ftp.wustl.edu
```

Name Server: relay.hp.com
Address: 15.255.152.2

Name: ftp.wustl.edu
Served by:
- WUGATE.WUSTL.EDU
 128.252.120.1
 WUSTL.EDU
- WUARCHIVE.WUSTL.EDU
 128.252.135.4
 WUSTL.EDU
- ADMIN.STARNET.NET
 199.217.253.10
 WUSTL.EDU
- NEWS.STARNET.NET
 199.217.253.11
 WUSTL.EDU
- ALPHA.NOC.USL.EDU
 130.70.40.25
 WUSTL.EDU

9-18. SLIDE: Zone Transfers with `ls`

Zone Transfers with `ls`

```
# nslookup
Default name server:      default_domain
Address:      default_server_address
> ls mayfield.hp.com > filename
```

a64673

Student Notes

Information about an entire zone can be transferred (displayed) using the `ls` command. It is helpful when one wants to check spelling of a remote host's name, counting hosts, or troubleshooting in general.

The `-t datatype` argument is the most useful. It allows filtering according to data types such as `MX`, `A`, `any`.

The other use of the zone transfer is to copy a zone database from a primary name server to update a secondary name server. This is initiated by sending a `HUP` signal to the secondary, which in turn starts child processes on the primary and secondary. These child processes offload the respective servers, and work with a snapshot of the database to eliminate problems in data inconsistency. A child name server started by a secondary server to pull a copy of a zone is called `named-xfer` instead of `named`.

9-19. SLIDE: Maintaining DNS

Maintaining DNS

- When data in any of the zone files is changed, inform the server.
 - Change the serial number in the **SOA** resource record.
- Inform **named** to reload the file.

To make a change in a zone file, perform the following:

- Modify the proper DNS database file with node name or address change.
- Modify the serial number in **SOA** field of the same DNS database file.
- Secondaries use the serial number to decide if new data must be downloaded from the primary.
 - Secondaries initiate a zone-transfer, if data has changed.
- Notify the name server of the change. **sig_named restart** (Secondaries can be forced to see the change, or wait for next zone transfer and serial number will show change.)

a646167

Student Notes

BIND name servers are manipulated by using signals. These signals are described below:

HUP	Restart the name server. This is normally used to restart a server after modifying its boot file or a database file.
INT	Dump a copy of the name server's internal database to /usr/tmp/named_dump.db .
ABRT	Append the name server's statistics to /usr/tmp/named.stats .
USR1	Append debugging information to /usr/tmp/named.run .
USR2	Turn off debugging.
WINCH	Toggle logging all queries with syslog. Logging takes place at priority LOG_INFO .

When hosts are added or deleted from zone files, the primary server must be notified.

The primary server can be notified by sending a hangup signal to **named**.

- Recall that **named**'s process ID is stored in **named.pid**. On HP systems call:

```
# kill -HUP `cat /var/run/named.pid`
```

or

```
# sig_named restart
```

Secondaries should also be made aware that data has changed so that zone transfer can be initiated.

- Accomplished by changing the serial number in the zone's **SOA** record.

9-20. SLIDE: DNS Client Setup: Resolvers

DNS Client Setup: Resolvers

- Clients use the resolver libraries to contact the name server.

```
# more /etc/resolv.conf
domain anyco.com
nameserver 130.207.110.14
nameserver 130.207.110.3
```
- If the search option is not used, the search list will contain only the local domain name. The search list can be changed by listing the desired domain search path keyword with spaces or tabs separating the names.

a64674

Student Notes

Resolver is a set of routines in the C library, used by clients to access the DNS system.

`/etc/resolv.conf` is read by the resolver to find out the name of the local domain and the location of name servers.

Name servers indicate the IP address of a name server that the resolver should query.

- resolver library queries them in the order listed, with a timeout period
- up to three name servers can be used

If no domain entry is present, domain name is assumed to be set, as discussed in the previous section.

If the user does not like the search order determined by the domain name assumed above, HP allows the search list to be set explicitly via a `search` directive in `resolv.conf`. The key word `search` is followed by from one to six domain names in the order you want them

searched. Most resolver queries will be attempted using each component of the search path in turn until a match is found. Note that this process may be slow and that it generates a lot of network traffic if the servers for the listed domains are not local, and that queries time out if no server is available for one of the domains.

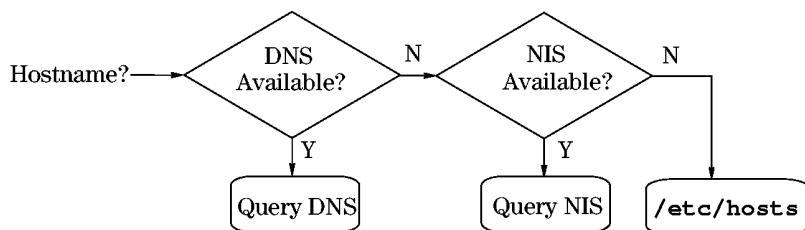
A new feature in HP-UX is the **nsswitch.conf** command. This controls the order in which the resolver consults the various naming services. It can be used to configure resolution order for the **hosts**, **services**, **networks**, **protocols**, **rpc**, and **netgroup** databases.

The HP-UX resolver also supports the **BIND 4.9.3** search list behavior and the **options ndots:n** directive.

ndots sets the minimum number of dots a domain name argument must have so that the resolver will look it up before applying the search list. The default **n** of 1 might be raised to 2 if users are accustomed to typing partial domain names.

9-21. SLIDE: Host Name Resolution Order

Hostname Resolution Order



a64675

Student Notes

The slide shows how the name resolution is performed on HP-UX systems. The resolution order is configurable in the name service switch data base, and therefore the slide represents only an example of one possible configuration.

If the `/etc/nsswitch.conf` file does not exist, or if no `source` is specified, or if the entry is

```
hosts: dns nis files
```

The default search order is as follows:

1. BIND (DNS)
2. NIS
3. `/etc/hosts` file

Note that if BIND is set up (`/etc/resolv.conf` exists), the following directives determine whether or not further queries are made. These directives changed from the last DNS version to the current:

root directive	last — current
SUCCESS=	return — return
NOTFOUND=	return — continue
UNAVAIL=	continue — continue
TRYAGAIN=	return — continue

Some more sample entries in `/etc/nsswitch.conf` are shown below:

```
hosts: dns [NOTFOUND=continue] nis [NOTFOUND=continue] files
```

```
hosts: dns [NOTFOUND=continue] nis
```

```
hosts: nis [NOTFOUND=continue] files
```

Note: `/etc/nsswitch.conf` can contain only *one* of these lines.

The First Example

```
hosts: dns [NOTFOUND=continue] nis [NOTFOUND=continue] files
```

causes a host to query DNS (BIND) first. If DNS does not have the requested information, or if DNS is not up and running, the host consults its NIS server. If NIS does not have the requested information too, or if NIS is not up and running, the host queries its `/etc/hosts` file.

The Second Example

```
hosts: dns [NOTFOUND=continue] nis
```

causes a host to query DNS (BIND) first. If DNS does not have the requested information, or if DNS is not up and running, the host consults its NIS server. If NIS does not have the requested information too, or if NIS is not up and running, the search is not continued.

The Third Example

```
hosts: nis [NOTFOUND=continue] files
```

causes a host to query NIS first. If NIS does not have the requested information, or NIS is not up and running, the host looks up its `/etc/hosts` file.

NOTE:

HP recommends that you maintain at least a minimal `/etc/hosts` file that includes important addresses like `routers`, `gateways`, `diskless boot servers`, and your host's own IP address. HP recommends that you include the word `files` in your `/etc/nsswitch.conf` file to help ensure a successful system boot when DNS and NIS are not available.

The Syntax of `/etc/nsswitch.conf`

This file contains a single line with the following syntax:

```
hosts: source [status=action status=action ...] source ...
```

source

A name service where host information can be found. Possible values are as follows:

dns Berkeley Internet Name Domain (BIND) service

nis The Network Information service (NIS)

files The `/etc/hosts` file

status

The result of querying the ***source*** for host information. Possible values are as follows:

SUCCESS The query was successful, and the information was found.

NOTFOUND The ***source*** responded to the query, indicating that it did not have the requested information.

UNAVAIL The query failed to connect to the source, the source was not running, or the source was not working properly.

TRYAGAIN The ***source*** was busy. Retrying the query might be successful.

action

The ***action*** to be taken based on the ***status*** of the query. Possible values are as follows:

return End the search and return control to the calling process, without querying the next source in the list.

continue Continue the search by querying the next ***source*** in the list.

The following parameters must be all lowercase: `hosts`, `dns`, `nis`, and `files`. The other parameters are not case-sensitive. Blank lines and lines beginning with # or white space are treated as comment lines and are ignored. The word `hosts:` must be the first thing in the line, with no spaces preceding it.

Checking the Current Configuration with nslookup

To check the name service switch configuration that your system is currently using, start **nslookup** and issue the built-in **policy** command, as follows:

```
# nslookup
```

```
Default Name Server: hp1.anyco.com
```

```
Address: 130.207.110.14
```

```
> policy
```

```
#Lookups = 3  
dns [RCCC] nis [RCCC] files [RCRC]
```

```
>
```

To stop the **nslookup** program, type **exit** or **ctrl****[d]**

This example shows the default configuration (**hosts: dns nis files**). The letters in square brackets describe the **actions** **Return** or **Continue**. They represent the values of the four statuses: **SUCCESS**, **NOTFOUND**, **UNAVAIL**, and **TRYAGAIN**. In this example, the **status=action** pair configured for DNS and NIS are:

```
SUCCESS=return
```

```
NOTFOUND=continue
```

```
UNAVAIL=continue
```

```
TRYAGAIN=continue
```

For the **nsswitch.conf** file containing the line

```
hosts: dns [NOTFOUND=continue] files
```

the **policy** command displays the following:

```
#Lookups = 2  
dns [RCCR] files [RRRR]
```

The **nslookup** command may display unexpected names or IP addresses. This may happen if the name could not be found in the first instance (DNS for example) and therefore is obtained from another source (for instance **etc/hosts** or NIS).

```
# nslookup sun1
```

```
Name Server: hp1.anyco.com
```

```
Address: 130.207.110.14
```

```
Name: sun1
```

```
Address: 130.207.110.1
```

You can trace the name lookup sequence if you use the **swtrace** option with **nslookup**.

```
# nslookup
```

```
Default Name Server: hp1.anyco.com
```

```
Address: 130.207.110.14
```

```
> set swtrace
```

> sun1

Name Server: hp1.anyco.com

Address: 130.207.110.14

lookup source is DNS

Name Server: hp1.anyco.com

Address: 130.207.110.14

*** No address (A) records available for hpeso

Switching to next source in the policy

lookup source is NIS

Default NIS Server: dec2

Address: 130.207.111.3

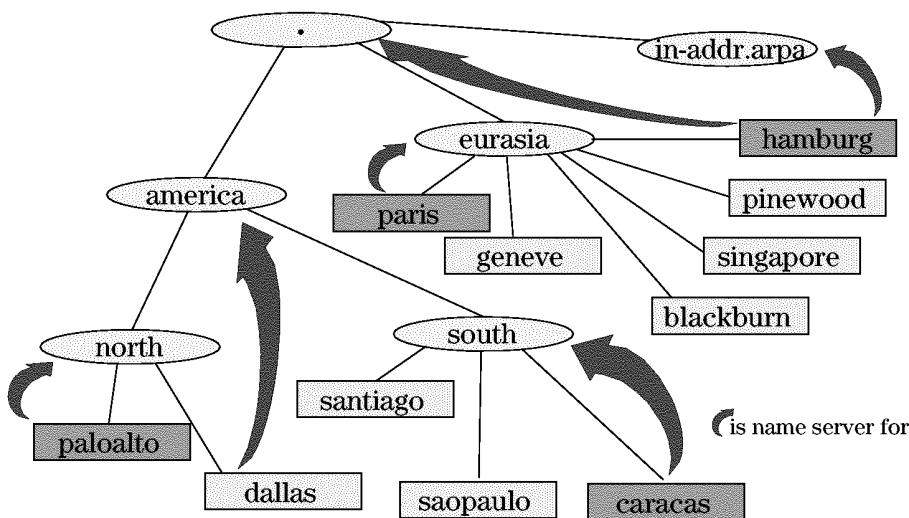
Name: sun1

Address: 130.207.110.1

For more information on the name service switch, see **switch** in section 4 of the *HP-UX Reference Manual*

9-22. LAB: Configuring and Maintaining the DNS

Configuring and Maintaining the DNS



Directions

You will set up your machines to use a complete Internet domain name space. On the domain map above is the layout of the domain name space.

In this example we have four domains:

```
america
north.america
south.america
eurasia
```

We suggest you use the following layout when setting up the domain name servers:

paloalto is the name server for **north.america**.

caracas is the name server for **south.america**.

paris is the name server for **eurasia**.

dallas is the name server for **america**.

hamburg is the root name server as well as the name server for the complete reverse name resolution **150.in-addr.arpa**.

Although it is not typical to set up the reverse name resolution on the root name server, we do this for simplification here.

Note that **hamburg** resolves all IP ranges for reverse name resolution. Again, this is not typical, but guarantees fewer problems within this lab. This is valid especially when starting up the domain.

In your lab layout it may be necessary to change the roles of certain systems. Ask your instructor to do so.

All questions and answers are based on the lab layout shown in the slide.

1. Copy your **/etc/hosts** file to **/etc/hosts.old**. From the **/etc/hosts** file remove all entries except

- your own hostname
- loopback or local host entry
- all hostnames within your domain

2. Create a directory **/etc/nameserver**.

3. Use the command **hosts_to_named** to create the appropriate BIND configuration files in the **/etc/nameserver** directory.

4. Change to **/etc/nameserver** and create the DNS data file with the command **hosts_to_named**.

5. `hosts_to_named` creates entries only for a flat domain space. Edit the configuration files and add, delete or change the appropriate entries under following considerations:

- which one is the root name server
- which domain is your name server responsible for
- which one is the name server for the reverse name resolution
- does your name server have to know other subdomains
- which are the hostnames returned by the reverse name resolution
- check your serial number

What do the following files look like?

- loopback data file
- cache file
- boot file
- name resolution data file
- reverse name resolution data file

6. Create the necessary client configuration file. The client configuration file is not only necessary on clients but also on each name server.

7. Check the name service switch configuration that your system is currently using. Your host should use the DNS name server. If an entry is not found, use the local `/etc/hosts` file. NIS should not be used for this lab.

8. Start your DNS name server and check the classroom DNS domain space.

To run **named** at next system startup, configure the appropriate configuration files.

9. Use **nslookup** to verify your system's IP address. You may wish to look up other system names both inside and outside of your domain.

Next, query other types like PTR, SOA or NS entries.

10. Choose some systems as secondary name servers. Which files do you have to edit ?

In our example we choose **pinewood** as a secondary name server for **eurasia**.

11. Which files are created automatically by **named**? Take a look at these files by using the **more** command.

12. On the primary name server, add a dummy entry to the data file and change the serial number.

Send the appropriate signal to re-read the configuration file.

Has the serial number changed on the secondary name server, too?

Look at the created data file.

13. Send the appropriate signal to your local **named** process to have it dump its current database. Compare the database dump to the zone files in **/etc/nameserver** and note any differences.

Module 10 — Configuring sendmail

Objectives

On completion of this module you will be able to do the following:

- List at least three different mail address formats.
- Describe the operation of SMTP.
- Make at least one valid modification to a ruleset in the `sendmail.cf` configuration file.
- Set up site hiding to support mail clients and servers.
- Configure forwarding.
- Configure a mail relay.

10-1. SLIDE: **sendmail** Overview

sendmail Overview

- **sendmail** is the ARPA/Berkeley Services inter-network mailing facility, or MTA (Mail Transport Agent)
- based on BSD 4.2
- developed by Eric Allman at University of California Berkeley
- implements the Advanced Research Projects Agency (ARPA) Simple Mail Transfer Protocol (SMTP) mailer over TCP/IP LANs (RFC821 / RFC822)
- Multipurpose Internet Mail Extentions (MIME) (RFC 1521/ RFC 1522) for multimedia support

a646139

Student Notes

sendmail acts as a central *post office* to collect and route mail to local or remote users or user agents (applications), via mailers. It provides the following services:

- Send and receive mail from other hosts on a LAN or gateway.
- Relays incoming and outgoing mail to the appropriate program for delivery or further routing.
- Interpretation of many address formats.
- Can be configured to individual needs by modification of the **sendmail** configuration file.

Multipurpose Internet Mail Extensions

The Internet Engineering Task Force's (IAB) Multipurpose Internet Mail Extensions (MIME) protocol enables the communication of multimedia electronic mail (e-mail) across heterogeneous computing environments. Network users want to be able to exchange

multimedia e-mail as well as word processing, spreadsheet and other documents without encoding them as ASCII text. MIME was originally developed as a protocol for exchanging multipart, multimedia messages on the Internet without losing structure or format data. It is now being incorporated into commercial e-mail products for enterprise computing. The technology reduces all types of text, video, audio and PostScript files to seven-bit ASCII data, using RFC 822-like headers to describe the type of message and media being encoded and ensure they are reassembled correctly at the receiving end.

10-2. SLIDE : The Many Roles of **sendmail**

The Many Roles of **sendmail**

The **sendmail** daemon performs several tasks:

- user agent as a simple front end called by a user
- routing facility called by another mail front end
- delivery agent called by a routing facility
(for example **sendmail**)
- receiving agent started as a daemon
- the **trace** command called by a user
- configuration parser called by a user

a64681

Student Notes

Following are the important parts of the UNIX mail architecture:

User Agent

A user agent is a program or set of programs that provide a user interface for creating, reading, and managing messages. Once a message is created, the user agent passes it to a routing facility for processing. In this context **mail**, **mailx** and **elm** are user agents.

Routing Facility

A routing facility is a program that receives messages from user agents, determines routing requirements and invokes an appropriate delivery agent to deliver the messages. **sendmail** is a routing facility.

Delivery Agent

A delivery agent is a program that accepts messages from a routing facility and either delivers them to a final local destination, or passes them via a communications medium to a receiving agent for further routing or remote delivery. Another name for a delivery agent is *mailer*.

Receiving Agent

A receiving agent is a process that receives messages from a remote delivery agent (mailer) and passes them to a routing facility. The `sendmail` daemon is a receiving agent. It receives messages from a peer delivery agent via a LAN using the SMTP protocol. Once the `sendmail` daemon collects the message, it passes it to `sendmail` for further processing and routing.

Transport Agent

A transport agent is a general term referring to either a receiving agent or delivery agent, depending on the context used.

SMTP

Simple Mail Transfer Protocol (SMTP) is the protocol used for transmitting messages through the ARPA Internet and Local Area Networks using TCP/IP. `sendmail` implements both an SMTP receiving agent and an SMTP delivery agent. RFC821 is the specification for SMTP.

In General:

`sendmail` collects the message from one of the following:

- `stdin` as user agent, trace command, or parser called by the user
- the argument list called by another mail front end
- SMTP port 25 as receiving agent for mail from remote delivery agents
- `sendmail`'s mail queue

References

- RFC821: specifies the method for transporting e-mail messages from one host system to another.
- RFC822: specifies a standard for the content of an e-mail message. RFC822 standardizes the names and usage of mail header lines, insuring that mail systems have a uniform syntax and interpret header files consistently, while allowing local extensions that will not interfere with interoperability.
- RFC821/822 are the `sendmail` equivalents for the X.400 P1/P2 protocols. It's easy to remember: they're RFC's, they were written in '82, and they map P1 and P2 protocols. That is, RFC-82-1 and RFC-82-2.
- RFC-976 UUCP Mail Interchange Format Standard
- RFC-1148 Mapping between X.400 (1988)/ISO10021 and RFC 822
- `sendmail(1M)`: Manual page for `sendmail` lists command line and configuration file options.
- *Installing and Administering ARPA Services*, Hewlett-Packard manual, part number B1014-90001. Provides detailed information about `sendmail`, particularly on command options and for deciphering header lines and configuration files.
- `sendmail` by Bryan Costales with Eric Allman and Niel Rickert. Publisher: O'Reilly & Associates, Inc., Sebastopol, California (ISBN 0-937 175-056-2). First edition, October 1993. This book covers major versions of `sendmail` and IDA `sendmail`, a European version.
- *!%@:: - A Directory of Electronic Mail Addressing and Networks*, by Donnalyn Frey and Rick Adams. Publisher: O'Reilly & Associates, Inc., Sebastopol, California. (ISBN 1-56592-031-7) Third Edition, August 1993. A guidebook to all the Internet networks worldwide with information on how to reach them. An essential handbook for regular UNIX mailers.

10-3. SLIDE: UNIX Message Format

UNIX Message Format

A mail message has three main elements:

- envelope
 - Containing SMTP routing information (RFC821)
- ARPA header
 - Containing a series of standard text lines incorporating addresses, routing, date, subject etc.
- body
 - Containing the message text; usually everything after the first blank line in a message header. Text is usually 7-bit-ASCII.

a64682

Student Notes

A UNIX mail message comprises three elements:

1. Envelope

This contains SMTP routing information (sender address, recipient address, and routing information as defined by RFC821). Users do not typically see the envelope.

2. Header

The ARPA header consists of a series of standard text lines incorporating addresses, routing, date, subject, and so forth, of the message.

The header is terminated by the first *blank* line.

ARPA defines a set of *tokens* that can be specified. The format for the header is:

```
token: <text>
token: <text>
token: <text>
      ----- blank line
```

Programs that generate ARPA headers (such as OpenMail when it sends mail through its `sendmail-interface`) are allowed to add their own *tokens* to the ARPA header. This allows the receiving agents to detect the presence of the tokens in the header and act on them accordingly. The advised standard for when you incorporate your own tokens is that these be preceded by an `x-`. OpenMail goes one step further and has said that all OpenMail specific tokens will start with `X-Openmail-`.

Examples of OpenMail tokens:

```
X-Openmail-Hops: <hop count>
X-Openmail-Creator: <name>
X-Openmail-Subject: <subject>
```

Header lines are useful for troubleshooting. Here's an example:

```
Return-Path: BOOTHROYD_SIMON/HP5003_01@hpod.pwd.hp.com
Received: from hpod.pwd.hp.com by hpodwd.pwd.hp.com with SMTP
          (16.8/15.5+IOS 3.20) id AA05490; Tue, 16 Feb 93 17:08:02 GMT
Received: from by hpod.pwd.hp.com with SMTP
          (16.6/15.5+IOS 3.22) id AA09910; Tue, 16 Feb 93 17:08:13 GMT
From: BOOTHROYD_SIMON/HP5003_01@hpod.pwd.hp.com
X-Openmail-Hops: 3
Date: Tue, 16 Feb 93 17:05:00 +0000
Message-Id: <"dc7c6(+0000vFhfA*">@MHS>
In-Reply-To: <"d03?EdP000000000*">@MHS>
Subject: AE351 - Yes Please!
Sender: BOOTHROYD_SIMON/HP5003_01@hpod.pwd.hp.com
To: geoffb@hpod.pwd.hp.com
```

3. Body

The body contains the message text; usually everything after the first blank line in a message header.

10-4. SLIDE: UNIX Address Formats

UNIX Address Formats

- UUCP: `host[!host]!user`
- ARPA: `user@host[.domain]`
- Relay: `user%domain@host`
- x.400: "pn=user/c=us/admd=attmail/...
 "`%mhs_relay@host[.domain]`
- Mixed Addressing
- x.400 Addressing
- Other Variations

a646140

Student Notes

UNIX mail can use one of three main address formats:

UUCP Addressing — Bang (!) Addressing

UUCP (UNIX-UNIX Copy) requires you to know the exact path to send your mail. You specify all the *hops* in the connection.

Syntax: `host[!host]!user`

Example: `romulan!klingon!jackd`

Read from left to right, this address effectively specifies a complete route, which is relative to the originating and destination hosts.

This says: Send the message to the host `romulan` and then to the host `klingon`, and there you will find the user `jackd`.

ARPA Addressing — At (@) Addressing

Specifies the destination host only, in the form:

Syntax: `user@host`

Example: `jackd@klingon`

This address is absolute, and applies to `jackd` from any host that addresses it. No actual route to `klingon` is contained in this address; this must be resolved by the delivering software.

- **Internet Addressing**

A development of ARPA addressing, specifying a whole domain rather than just the target host. This is also an absolute address.

Syntax: `user@domain`

Example: `jackd@klingon.acme.com`

Relay Addressing — Percent (%) Addressing

You can also specify a host at which the Internet address will be found, if for instance, your host lacks a route to a domain you want to address:

Syntax: `@host [, @host...] : user@domain`

Where `@host` is often referred to as the relay host(s).

Example: `@romulan: jackd@klingon.acme.com`

This says: Send the message to the host `romulan` and from there you will be able to send to the address `jackd@klingon.acme.com`.

In the HP world, this form of the Internet addressing is written using the %.

Syntax: `user%domain@host`

So this example would be re-written as: `jackd%klingon.acme.com@romulan`

Mixed Addressing

These basic address formats can be mixed. This makes for fun when trying to interpret them.

Usually the operator precedence is set to the following:

1. @ (The highest priority)
2. !
3. % (The lowest priority)

Examples:

```
spock!kirk@acme.com
jeremyb%hpopdxb@hplb.hpl.hp.com
bsstrs%gdr.bath.ac.uk% nfsnet-relay.ac.uk@hpl.hpl.hp.com
LSchoor%ntmutual.com.au@hplb.hpl.hp.com
```

X.400 Addressing

Mail can be addressed to X.400 from UNIX by supplying the X.400 address in double quotes, for example:

```
"pn=user/c=us/admd=attmail/ ... "%mhs_relay@ucl.ac.uk"
```

This routes the message to the domain **ucl.ac.uk** (University College, London), where the relay **mhs_relay** forwards it to X.400.

Other Variations

Non-actioned comments can be included in addresses in various ways; this is how to include real names in Unix mail addresses.

Examples:

```
jackd@acme.com (Jack Dee)
Jack Dee<jackd@acme.com>
Jack Dee (JD for short) <jackd@acme.com>
```

10-5. SLIDE: Address Encoding and Decoding

Address Encoding and Decoding

(# l#
)	# r#
'	# m #
:	# c #
.	# f#
/	# s#
@	# a#
!	# x #
%	# p #

a620190

Student Notes

Some characters have a special meaning in UNIX mail, and therefore must be encoded. Conversely, any messages coming into the UNIX mail gateway are decoded to replace characters.

10-6. SLIDE: sendmail as a Daemon

sendmail as a Daemon

- **sendmail** as daemon process's two tasks:
 - **sendmail** as RECEIVING agent bound to port 25
 - **sendmail** as mail scheduler processing undelivered mail

RCPT to: recipient

Invoke **sendmail** daemon with:

`/usr/sbin/sendmail -bd -q interval`

- **-bd** initializes daemon in background
- **-q interval** process mail queue each interval

a646141

Student Notes

To send mail to the network, local mailers invoke their own **sendmail** process to handle the outgoing mail. But to receive incoming mail from the network, and to regularly process any queued up mail in the Mail Queue, you must have an SMTP server (the **sendmail** daemon) running on your local host.

As root, invoke a daemon with:

- The **-bd** option initializes the daemon in background mode.
- The **-q** option specifies how often **sendmail** will look at its wait queue to handle any deferred mail.
- *interval* can be w=weeks, d=days, h=hours, m=minutes, or s=seconds.

Specify the common default interval of 30 minutes with the following:

`/usr/sbin/sendmail -bd -q30m`

The above command is started by the script `/sbin/init.d/sendmail` file if the `sendmail_server` variable is set to 1 in the `/etc/rc.config.d/mailservs` file. On reboot, this will remove any mail queue files resulting from system interruption, run the start command, and log the restart to the syslog file.

10-7. SLIDE: sendmail as a Routing Facility

sendmail as a Routing Facility

Received: *recipient* (intermediate or final)
Date: *date_received*
From: *address_of_originator*
Message-ID: *unique_id@mail_hub_of_originator*
To: *address_of_correspondent*
Cc: *list_of_copied_recipients*
Subject: *subject*

a64684

Student Notes

All mail messages are composed of two distinct parts: the header (containing information such as who the message is from) and the body (the actual text of the message). The most important headers are as follows:

Received:

The **Received:** header conveys information about every site the mail message passes through. This header is first inserted by the originating site, and each following site appends its own header to the list. The result is a chronological list of the path from originator to recipient for tracing functionality.

Date:

The **Date:** header specifies the date and time that the mail message was originally sent. It must be included on all messages.

From:

The `From:` header lists the address of the sender. The four legal formats are

```
From: address
From: <address>
From: Full Name <address>
From: address (comment)
```

Message-ID:

The `Message-ID:` header is used to identify each message with a tag that is unique worldwide. The tag is of the form:

```
<timestramp.queueid@localhost>
```

where

```
    timestamp is current time to the nearest second
    queueid is a unique local queue identifier
    localhost is the fully qualified domain name of the local host
```

To:

The `To:` header lists one or more of the recipients of the mail message.

Cc:

The `Cc:` header is treated by `sendmail` no differently than `To:`. To the user, it implies that there were recipients to whom an informational copy of the message was supplied.

Subject:

The `Subject:` header can be included in mail messages to give the topic of the message. Most mail-reading programs display a portion of the first line of the message as the subject.

10-8. SLIDE: The Configuration File `sendmail.cf`

The Configuration File `sendmail.cf`

- creates an environment for `sendmail`
- specifies how mail will be routed
- specifies how addresses are to be interpreted
- specifies how `sendmail` will rewrite addresses (according to rulesets)

a646142

Student Notes

`sendmail`'s flexibility as a routing facility derives from the fact that most operational parameters are contained in a configuration file (`sendmail.cf`), which can be customized to suit almost any requirements. Hewlett-Packard Company supports a limited number of changes to the file, mainly for localizations and routing, including:

- defining the local domain
- defining a UUCP relay host
- defining an SMTP relay host
- modifying changeable options
- defining the file class of direct SMTP connections
- using the `pathalias` name server

- defining OpenMail mailers

Possible more extensive changes that are not supported include:

- adding new delivery agents
- defining new routing classes
- implementing special routing services
- providing gateway functionality
- using mixed addressing

In theory, the file is self-documenting, although the rulesets defining mailers and address re-writing rules are quite tricky and take a while to get used to. Be very careful that you don't mess up your configuration file by making an erroneous change; it may be difficult to recover. Unless you are very sure of what you're doing, don't make changes.

10-9. SLIDE: Configuration File Basics

Configuration File Basics

Syntax Definition Characters

Letter	Defines
C or F	Classes
D	Macros (Defining a macro)
H	Header line formats
M	Mailers (delivery agents)
O	Options
P	Precedence classes for messages
T	Trusted users
S	Start of rulesets
R	Rewriting rules

a646143

Student Notes

The configuration file is organized as a series of several distinct types of lines. The first character of each line defines the type of the line. Lines beginning with a space or tab character are continuation lines. Blank lines and lines beginning with a hash (#) character are comments.

For example, these lines in `sendmail.cf` mean:

- `Timeout.queuereturn=5d`

Set option `Timeout.queuereturn` equal to 5 days. `Timeout.queuereturn=5d` sets the time length that `sendmail` will try to send mail to a remote machine. If it receives no response within the specified time, `sendmail` returns the mail as undeliverable.

- **Troot**

Tdaemon

Tell **sendmail** that **root** and **daemon** are *trusted users*. When **sendmail** runs, it always checks who invoked it. If the invoker is not one of these *trusted users* it aborts.

- **Dmhp.com**

Define the macro-variable **m** to be **hp.com**. The variable that holds the local domain for this **sendmail** is **m**.

10-10. SLIDE: Rulesets and Mailers

Rulesets and Mailers

`sendmail` performs the following on the receipt of a message:

- defines the delivery agent (defined by rulesets)
- converts the addresses to delivery agent, host, and user triplets
- executes the delivery agents

a646144

Student Notes

- When `sendmail` gets a message (either from its queue, stdin, SMTP...) it basically works-out who to give it to. That is, which *delivery agent* will it give the message to. The configuration file will have a series of rulesets and definitions that will determine this *delivery agent* and how it is to be invoked.
- A *delivery agent* is defined as a *mailer* in the `sendmail.cf` file. Mailer definition lines start with a capital *M*, rulesets begin with a capital *S*, and each line of the ruleset starts with a capital *R*.
- Having determined the delivery agent, `sendmail` must then go through and *convert* all the supplied addresses (`From:`, `To:`, etc.) into the format of the receiving mailer.

- The steps are:

1. Resolve the mail address into a {delivery-agent, host, user} triplet.

`sendmail` looks at each `To:` address to determine its `{delivery agent, host, user}` triplet. That is, each `To:` might resolve to a different triplet (one recipient might be local, one remote, etc.).

`sendmail` determines this *triplet* by applying a series of rules to the `To:` address. These are specified as *rulesets* in the `sendmail.cf` file.

The *triplet* maps you to the delivery agent (or mailer).

2. Convert all addresses to match those of the target mailer.

This conversion depends on whether it's a *sender* or *receiver* address, and `sendmail` allows each mailer definition to specify special tweaks in each case.

3. If necessary, add lines to the message header to enable replies.

4. Fork a child process which executes a delivery agent.

Having determined the *triplet* (and thus the *delivery agent*) and converted all addresses, `sendmail` forks and executes the mailer as per its defined command line.

- If the delivery agent is specified as a remote interprocess communication (IPC), it opens a TCP/IP connection to the SMTP server on the remote host and transfers the message over SMTP.
- It processes a copy of the message for each delivery agent invoked, and possibly for each copy of the message. Some delivery agents only may allow sending a message to one person at a time. Others allow bulk sending).

`sendmail` creates a transcript of each transaction (including error messages) to send to the originator if a message cannot be delivered. If `sendmail` cannot deliver the message at the first attempt, but error status indicates that delivery might be successful, it stores the message in a queue for later retry.

10-11. SLIDE: Defining a Mailer

Defining a Mailer

To define a mailer to sendmail, specify an **M** line:

M=mailer, P=path, F=flags, S=sender, R=recipient, A=argv

<i>mailer</i>	Name of a mailer
<i>path</i>	Path for the mailer binary
<i>flags</i>	Special mailer flags
<i>sender</i>	The number of any special ruleset for rewriting sender addresses
<i>recipient</i>	The number of any special ruleset for rewriting recipient addresses
<i>argv</i>	Argument vector to execute mailer

a646145

Student Notes

Let's look at an example, in this case the entry for the *omxport* mailer that receives OpenMail mail (that has come across the **sendmail** transport, addressed to **openmail@this_machine**).

```
Momxport,  P=/usr/openmail/bin/xport.in,  F=LMn,  A=xport.in $u
```

Here, *omxport* is a descriptive name for the mailer, whose actual name and location is given by the path entry (*/usr/openmail/bin/xport.in*). No special address rewriting rulesets have been specified. The mailer *xport.in* is executed, passing the contents of *u* (the recipient name). The header flags specified have the following meanings:

- L** Line lengths as specified in RFC 821
- M** Message-id: expected
- n** Don't insert **FROM_ address** on the front of messages

This mailer definition is of no use to you until you have defined the necessary rulesets such that **openmail** e-mail addressed to **openmail@<this_machine>** is mapped onto this mailer.

Defining the Rulesets

Address rewriting rulesets are ordered sets of address rewriting rules defined in the configuration file. These rulesets perform two major functions. They tell **sendmail** how to:

- Route mail by specifying how **sendmail** should interpret and resolve recipient addresses to {**delivery agent**, **host**, **user**} triplets.
- Rewrite sender and recipient addresses which are passed in the message as envelope or header information. This ensures that this information can be correctly understood at the destination.

The syntax for specifying a rule is:

R<left-side> <right-side> optional comments

For example:

R\$*<\$+>\$* \$2 strip phrase and <>

This line takes an address of the form:

"John Smith (Marketing Dept) <johns@hpbbn.bbn.hp>"

and outputs just the part within the angle brackets:

"johns@hpbbn.bbn.hp"

Explanation of the Symbols \$*, \$+, ...

- When **\$** is used in rulesets it has additional meanings over and above what has already been discussed:

On the left-hand-side of the rule:

\$*	Match zero or more tokens.
\$+	Match one or more tokens.
\$-	Match exactly one token.
\$=x	Match any token or concatenation of tokens in class <i>x</i> .
\$~x	Match any single token NOT in class <i>x</i> .

On the Right Hand Side of the rule:

\$n	where <i>n</i> is a number (for instance \$1 or \$2). Substitute with the input that matched the nth occurrence of any of \$+, \$-, \$*, \$=x, \$~x.
-----	--

\$>n means to apply this on the right side and then call ruleset *n*.
\$ [host\$] means substitute with the full domain name for this host.
\$@ at the start of a rule means: apply this rule, then stop.
\$: at the start of a rule means: only apply this rule once.

- Defining the {**mailer**, **host**, **user**} triplet

\$#mailer defines *mailer* to be the mailer agent used to handle this mail.
\$@host defines *host* to be the receiving host for this mail.
\$:user defines *user* to be the receiving user for this mail.

- sendmail** rewrites addresses by applying whole rulesets to them. It applies a ruleset by trying each rule in the ruleset on the address. If the rule matches, **sendmail** rewrites the address. It then takes this rewritten address and *reapplies the same rule* until it fails. It then moves to the next address, until all the rules run out. (That's why you have the **\$@** and **\$:** options at the start of the right-side, so you can specify that you only want this line of the ruleset to be applied *once*). Rulesets are defined by numbers starting from 0. And just to keep you on your toes, **sendmail** always starts with ruleset 3. For recipient addresses it then branches to ruleset 0.

- Ruleset 0 is the ruleset that produces the {**da**, **h**, **u**} triplet.
- sendmail** uses rulesets 3,0,4 to determine how to route recipient addresses.

In our *omxport* example, the ruleset to produce the {**da**, **h**, **u**} triplet is:

- Resolve mail to OpenMail from remote OpenMail system

Ropenmail \$#omxport\$@\$w\$:openmail

Current **sendmail** has the additional rulesets 5,90,93,94,95,96,97, and 98.

10-12. SLIDE: Ruleset Example: Avoiding Spam

Ruleset Example: Avoiding Spam

In `sendmail.cf`:

```
# file containing full e-mail addresses of well-known spammers  
F{Spammer} /etc/mail/Spammer
```

```
# file containing domains of well-known spammers  
F{SpamDomains} /etc/mail/SpamDomains
```

In `/etc/mail/Spammer`:

```
john@FOO.BAR  
adam@FOO.BAR
```

In `/etc/mail/SpamDomains`:

```
pests.com  
mailhogs.com
```

a64685

Student Notes

Anti-spamming Rulesets

One of the important features provided by `sendmail` is a new group of rulesets to avoid mail *spamming* (that is, the sending of unsolicited mail to large number of users) and to prevent mail *spammers* from using your host as a mail relay.

`sendmail` provides four new *named* rulesets which can be used to check and reject abusive mail messages. The four rulesets are:

check_mail Ruleset

This accepts or rejects mail according to the sender's address in the SMTP `MAIL FROM` command. This ruleset can be used to reject mail messages from specific users or domains. The ruleset has two associated file macros to define the users and domains from whom you do not want to receive mail.

```
# file containing full e-mail addresses of well-known spammers
F{Spammer} /etc/mail/Spammer

# file containing domains of well-known spammers
F{SpamDomains} /etc/mail/SpamDomains
```

For example: If you want to reject mail messages from only `john@FOO.BAR` and `adam@FOO.BAR`, specify them in `/etc/mail/Spammer`.

```
john@FOO.BAR
adam@FOO.BAR
```

Now you will not receive any mail from these two users.

Another example: If you want to reject mail messages from all the users in a particular domain, then you specify that domain in `/etc/mail/SpamDomains`.

```
pests.com
mailhogs.com
```

Now no mail messages are received from the `pests.com` and `mailhogs.com` domains.

It is also possible to reject mail messages from one particular host, say `DeniedHost.FOO.BAR`. You enter this in the `/etc/mail/SpamDomains` file and you will not receive mail messages from this host.

NOTE:	Mail messages from only the entries in <code>/etc/mail/Spammer</code> and <code>/etc/mail/SpamDomains</code> will be denied, while mail messages from all other domains are allowed.
--------------	--

check_rcpt Ruleset

This checks the recipient address given in the SMTP `RCPT TO` command, to prevent spammers from using your machine as a mail gateway. Associated with this ruleset are three file macros.

```
# file containing IP numbers of machines which can use our relay
F{LocalIP} /etc/mail/LocalIP

# file containing names of machines which can use our relay
F{LocalNames} /etc/mail/LocalNames
```

```
# file containing names of machines we relay to
F{RelayTo} /etc/mail/RelayTo
```

For example: If you want to allow the local clients, say hosts with the IP addresses **127.0.0.1** (local host) and **15.10.43.245** (FOO.BAR) to relay through your mail server, then you could specify their IP address in the file **/etc/mail/LocalIP** file as

```
127.0.0.1
15.10.43.245
```

Therefore mail messages sent from the localhost as well as from FOO.BAR will be accepted.

Instead of IP address, you can specify the hostname as well in **/etc/mail/LocalNames** as

```
localhost
FOO.BAR
```

If you do not want to be a relay for external mail messages originating from a domain other than your domain and not destined for your domain, then you can deny it by using the file **/etc/mail/RelayTo**.

For example, if you want to relay mail messages only for the domain **hp.com**, then you can do so by specifying it in the file **/etc/mail/RelayTo** as,

```
hp.com
```

This means that when a user in the domain **aol.com** tries to send mail to another user in the domain **rocketmail.com** through your mail hub, the hub will deny the mail saying **we do not relay**. However, if you want to be a relay for some external domains then you can do so by specifying either the IP address or hostname in the file **/etc/mail/LocalIP** or **/etc/mail/LocalNames** accordingly.

check_relay Ruleset

This accepts or rejects SMTP connections according to incoming hostname domain or IP address. Associated with this ruleset are two file macros:

```
# file containing IP addresses of hosts denied access to this mail server
F{DeniedIP} /etc/mail/DeniedIP
```

```
# file containing names of hosts denied access to this mail server
F{DeniedNames} /etc/mail/DeniedNames
```

For example: If you want to deny hosts **FOO1.BAR** and **FOO2.BAR**, with IP addresses **15.10.43.248** and **15.10.43.245**, access to your mail server, you can do so by specifying their IP addresses in the file **/etc/mail/DeniedIP** as:

```
15.10.43.248
15.10.43.245
```

or alternatively by specifying these hostnames in the file **/etc/mail/DeniedNames** as

```
FOO1.BAR
FOO2.BAR
```

check_relay's denial function is similar to **check_mail**'s rejection of mail from specific domains and machines. However **check_relay** can also be used to deny mail messages received by a host that is acting as a relay machine.

This ruleset differs from the **check_rcpt** ruleset in the sense that, in **check_rcpt**, the mail not destined to your domain but being relayed will be accepted by the mail hub and later rejected. Using **check_relay**, it is possible to refuse to accept the mail messages from domains specified in the file **/etc/mail/DeniedIP** or **/etc/mail/DeniedNames**.

check_compat Ruleset

This allows/disallows mail transfers between specified sender/recipient pairs. Unlike **check_mail**, **check_rcpt** and **check_relay**, this is called for all deliveries, not just SMTP transactions. **check_compat** is used in the following situations:

1. A set of users who are restricted from sending mail messages to external domains but need to send mail messages to internal domains. Both the sender and recipient addresses are checked to ensure that they are in the local domain.
2. A particular user needs to ensure that they do not receive mail messages from a specific source.
3. A particular host needs to ensure that external senders do not use that host as a mail relay. The mail messages are screened, based on the sender's hostname.

The new rulesets **check_mail**, **check_rcpt**, **check_relay**, and **check_compat** use the information in the file macros given above and do validity checking on SMTP arguments. In case the rulesets resolve to the **\$#error mailer**, the SMTP command is rejected.

For enabling anti-spamming, uncomment the lines between **# Begin Anti-Spamming** and **# End Anti-Spamming** in the new **sendmail.cf** supplied along with **sendmail**. Also, create the files mentioned above with the required entries to enable anti-spamming.

10-13. SLIDE: Simple Mail Transport Protocol (SMTP)

Simple Mail Transport Protocol (SMTP)

- **HELO** *hostname*
- **MAIL FROM:** *sender*
- **RCPT TO:** *recipient*
- **DATA**
- **QUIT**
- **RSET**
- **NOOP**
- **HELP**
- **VRFY** *recipient*
- **EXPN** *recipient*

a646146

Student Notes

SMTP is a simple protocol by which `sendmail` can send/receive a message. (SMTP is specified by RFC 821.)

It is basically a set of commands:

HELO <i>hostname</i>	initiate a session, and identify the <i>sending</i> hostname
MAIL FROM: <i>sender</i>	specify sender of mail
RCPT TO: <i>recipient</i>	specify recipient of mail (use multiple RCPT TO: commands).
DATA	signal the start of the text (terminated by a line with a single dot '.')
QUIT	end <code>sendmail</code>
RSET	reset the system

NOOP	do nothing (No-Operation)
HELP	get help
VRFY <i>recipient</i>	print the recipient's full-name
EXPN <i>recipient</i>	(expand) same as VRFY

NOTE: The *envelope* information is transferred with the **MAIL** and **RCPT** commands, and the *header* and *message-body* information are transferred as **DATA**.

10-14. SLIDE: Local and Remote SMTP

Local and Remote SMTP

Local SMTP supports interactive use of **sendmail**

- **/usr/sbin/sendmail -bs [-v]**

Remote SMTP supports use of telnet with **sendmail**

- **telnet hostname 25**

a646147

Student Notes

Local SMTP

Simply invoke **sendmail** interactively and specify that you will communicate using SMTP (-bs option to **sendmail**):

```
$ sendmail -bs -v
220 hpopdwd.pwd.hp.com HP Sendmail (16.8/15.5+IOS 3.20) ready at Wed,
17 Feb 9313:35:18 GMT:
mail from:geoffb
250 geoffb... Sender ok
rcpt to: geoffb
250 geoffb... Recipient ok
data
354 Enter mail, end with "." on a line by itself
Test message
.
050 geoffb... Connecting to local host (local)...    <- Produced by
```

```

050 geoffb... Executing "/bin/rmail -d geoffb"      <- the '-v' option
050 geoffb... Sent          <- on the cmd line
250 Ok
quit
221 hpopdwd.pwd.hp.com closing connection
$
```

And the mail received:

```

Return-Path: geoffb@hpopdwd.pwd.hp.com
Received: by hpopdwd.pwd.hp.com with SMTP
(16.8/15.5+IOS 3.20) id AA10584; Wed, 17 Feb 93 13:35:49 GMT
Date: Wed, 17 Feb 93 13:35:49 GMT Yo Dude
From: Geoff Benton <geoffb@hpopdwd.pwd.hp.com>
Return-Path: <geoffb@hpopdwd.pwd.hp.com>
Message-Id: <9302171335.AA10584@hpopdwd.pwd.hp.com>
Apparently-To: geoffb@hpopdwd.pwd.hp.com
```

Test message

Remote SMTP

sendmail implements a client and a server for the SMTP specified in RFC 821. **sendmail** and other mail handling programs use this protocol to transfer messages over TCP/IP networks.

The **sendmail** daemon listens for connection requests on the SMTP TCP port (specified in */etc/services* and normally 25), and when it receives such a request it forks an SMTP server to communicate with the **sendmail** program (perhaps another **sendmail**). The message is transferred between them by means of an interactive dialog—the envelope with **HELO**, **MAIL**, **From:**, **To:**, and **RCPT** commands, and the header/body with a **DATA** command.

To invoke a remote **sendmail** via SMTP simply use **telnet** and specify the port 25. Note that this is very helpful when testing SMTP connections.

```

$ telnet hpopd 25
Trying...
Connected to hpopd.pwd.hp.com.
Escape character is '^].
220 hpopd.pwd.hp.com HP Sendmail (16.6/15.5+IOS 3.22) ready at Wed,
17 Feb 93 13:55:00 GMT:
heloh
250 hpopd.pwd.hp.com Hello (hpopdwd.pwd.hp.com), pleased to meet you
mail from: geoffb
250 geoffb... Sender ok
rcpt to: geoffb
250 geoffb... Recipient ok
data
354 Enter mail, end with "." on a line by itself
from: geoffb
to: geoffb, iwb, root

Hi there,
This is an important message
```

```
.
```

```
250 Ok
quit
221 hpopd.pwd.hp.com closing connection
Connection closed by foreign host.
$
```

And the mail received:

```
Return-Path: geoffb@hpopd.pwd.hp.com
Received: from hpopd.pwd.hp.com by hpopdwd.pwd.hp.com with SMTP
          (16.8/15.5+IOS 3.20) id AA10625; Wed, 17 Feb 93 13:55:44 GMT
Return-Path: <geoffb@hpopd.pwd.hp.com>
Received: from hpopdwd.pwd.hp.com by hpopd.pwd.hp.com with SMTP
          (16.6/15.5+IOS 3.22) id AA15683; Wed, 17 Feb 93 13:55:11 GMT
Date: Wed, 17 Feb 93 13:55:11 GMT
From: Geoff Benton <geoffb@hpopd.pwd.hp.com>
Message-Id: <9302171355.AA15683@hpopd.pwd.hp.com>
To: geoffb@hpopd.pwd.hp.com, iwb@hpopd.pwd.hp.com, root@hpopd.pwd.hp.com
```

```
Hi there,
This is an important message
```

10-15. SLIDE: Overview of sendmail Configuration

Overview of **sendmail** Configuration

- site hiding
- **sendmail** and DNS
- mail exchanger
- mail hub

a64686

Student Notes

These are the main configuration tasks for **sendmail**.

10-16. SLIDE: Site Hiding

Site Hiding

- Hide own origin:

DMsite_domain_name

```
#####
# Ruleset 94 -- convert envelope names to masqueraded form#
#####
S94
R$+                      $@ $>93 $1
RS* < @ * LOCAL* > $*      $: $1 < @ $j . > $2
```

- Hide other origins:

CMotherhost.domain

DMfileContaining_hidden_hosts

a64687

Student Notes

Site hiding causes *outgoing* SMTP mail to be labeled as coming from another site rather than from the local host. It serves several purposes:

- Use as a mail server.

A mail server receives and stores mail for several clients. A mail client sending mail should appear as the mail server. The mail client therefore fakes the sender address.

- Use as a mail hub.

Each mail within a certain domain passes a mail hub. Users outside the domain don't see several hosts but only the domain. Outgoing mail is labeled with the domain name as sender. Incoming mail is sent to user@domain.

Passing the mail hub, the mail is distributed to one or several mail servers or to each single host.

Site hiding is also called *masquerading*. In newer **sendmail** versions we mostly use the term *masquerading*.

To configure site hiding or masquerading, edit **sendmail.cf** to include:

DMsite_domain_name

The DM macro only masquerades the header address. In order to change the envelope names, too, uncomment rule **#R\$+** on ruleset 94:

```
#####
### Ruleset 94 -- convert envelope names to masqueraded form  ##
#####

S94
R$+          $@ $>93 $1
R$* < @ *LOCAL* > $*   $: $1 < @ $j . > $2
```

Normally the only addresses that are masqueraded are those that come from this host. Hiding site can be done by each sending host. Another possibility would be, that all clients send first their mail to the mail hub. Passing the mail hub, the mail hub masquerades all relayed mails.

CMotherhost.domain

The effect of this line is that although mail to `user@otherhost.domain` will not be delivered locally, any mail including any `user@otherhost.domain` will, when relayed, be rewritten to have the MASQUERADE (DM macro) address. This can be a space-separated list of names.

If these names are in a file, you can use

FMfile_containing_hided_hosts

to read the list of names from the indicated file.

NOTE:	When sending outgoing mail for testing, DO NOT use the root account. By default, root is not masqueraded.
--------------	--

10-17. SLIDE: DNS and e-mail

DNS and e-mail

DNS provides MX records to

- address mail relays
 - mail can be relayed, if recipient is down
 - hosts assume mail handling responsibilities for other hosts
 - preference values determine order in which MX records are processed
- address mail hubs
 - advanced mail routing features
 - hosts either process or forward mail
 - application for domains with a firewall

a630206

Student Notes

MX records specify a mail exchanger for a domain name: a host that will either process or forward mail for the domain name. *Processing* the mail means either delivering it to the individual it is addressed to, or *gateways* it to another mail transport, like UUCP.

Forwarding means sending it to its final destination or to another mail exchanger closer to the destination via SMTP. Sometimes forwarding the mail involves queueing it for some amount of time, also.

MX records are used for two main purposes:

1. To arrange that one host *back up* another by receiving mail for it when it is down.
2. To arrange that mail addressed to remote networks be relayed through the appropriate gateways.

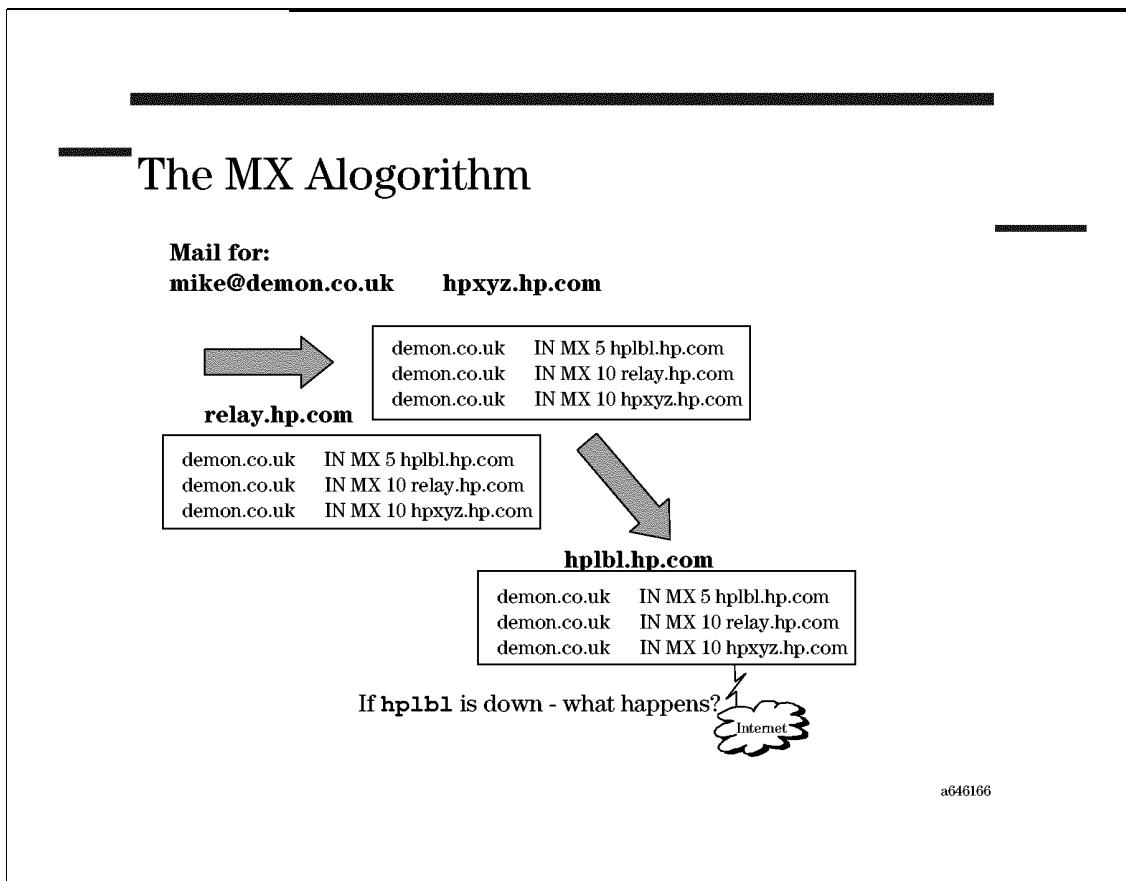
In order to prevent mail routing loops, the MX record has an extra parameter, besides the domain name of the mail exchanger: a *preference* value. The preference value is an unsigned 16 bit integer (between 0 and 65535) that indicates the mail exchanger's priority.

More than one MX record can be employed, to provide multiple backup. The record format is:

```
hostA    IN    MX 0 hostA
         IN    MX 10 hostB
         IN    MX 20 hostC.otherdomain
```

where the first entry indicates the original recipient, IN indicates this is an Internet-type record, *integer* indicates priority (highest 0). The last entry indicates the new recipient.

10-18. SLIDE: The MX Algorithm



Student Notes

MX records are used only if a message address resolves to an IPC mailer, that is, one that uses SMTP over sockets to perform delivery. Instead of attempting to connect directly to the recipient host, `sendmail` first queries the name server, if it is running, for MX records for that host. If the name server returns any, `sendmail` sorts them in preference order, highest preference (lowest number) first.

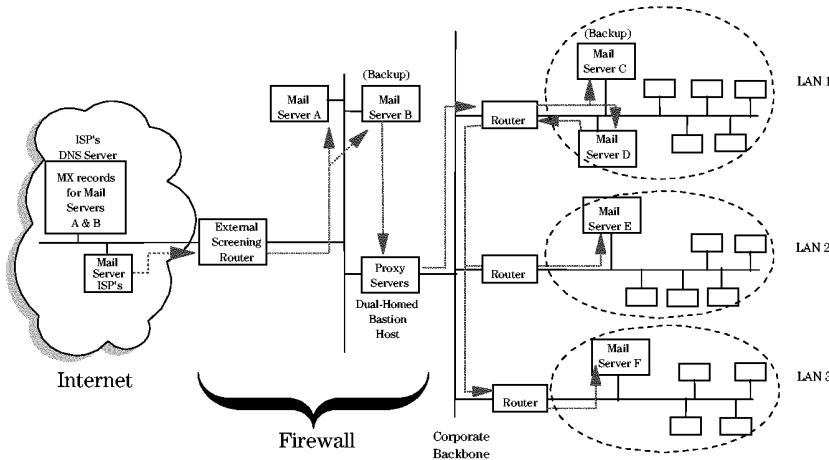
If the *local* host appears in the list, it and any MX hosts with lower preference (higher numbers) are removed from the list. If any MX hosts remain, **sendmail** then tries to connect to each MX host in the list in order, and it delivers the message to the first MX host to which it successfully connects. If that MX host is not the final destination for the message, it is expected that the host will relay the message to its final destination. If **sendmail** tries all the MX hosts in the list and fails, the message is returned to the sender with an error message. If you want **sendmail** to try to connect to the host to which the message is addressed, uncomment the following line in the

/etc/mail/sendmail.cf file: **TryNullMXList**. **sendmail** then tries to connect to the host to which the message is addressed, if any of the following conditions occur:

- The name server returns no MX records.
- The name server is not running.
- The local host is the highest preference mail exchanger in the list.

10-19. SLIDE: MX and the Firewall

MX and the Firewall



a64688

Student Notes

This slide shows a company with two external e-mail servers (A and B). In order to exchange e-mail with users on the Internet, the company must ensure that its Internet service provider (ISP) has its DNS server configured to include MX records for the company's e-mail servers A and B. The reason there are two e-mail servers within the firewall is that one of them functions as a backup e-mail server, to provide high availability. e-mail from the Internet is forwarded from an Internet service provider's e-mail server to the company's e-mail servers. This is done for security reasons because it is not a good idea to have e-mail go directly from the Internet to the internal corporate e-mail servers. From there, the mail is forwarded to the company's internal corporate e-mail servers (C and D), one of which is a backup server. These corporate e-mail servers in turn forward mail to the appropriate e-mail server for LAN 2 and LAN 3. There may be more LANs than are shown in this slide.

10-20. SLIDE: Sending Mail Through a Firewall

Sending Mail through a Firewall

```
hp.com. IN MX 10 tiger.hp.com.  
IN MX 20 mailhub.hp.com.  
IN MX 30 relay.hp.com.
```

a64689

Student Notes

In this configuration, *relay* is a firewall system. Incoming mail from the Internet will arrive on *relay* and will be forwarded to *tiger* or *mailhub* (if *tiger* is down) because both *tiger* and *mailhub* cannot be reached from the Internet while *relay* can be reached.

Outgoing mail to the Internet from inside the organization will go to *tiger* first, or to *mailhub* if *tiger* is down, since that is the *natural* order of precedence, and because all three hosts can be seen from inside the organization.

10-21. SLIDE: Internal Domain Addresses

Internal Domain Addresses

All incoming mail to the domain can be directed to the mail hub using the following technique:

```
*.acme.co.uk.      IN  MX  10  mailhub.acme.co.uk.  
mailhub.acme.co.uk.  IN  A       193.14.104.33
```

a64690

Student Notes

The example in the slide makes all mail addressed to *any host* in the `acme.co.uk` domain go through `mailhub`. All possible recipient addresses must be announced as local on `mailhub`. Also, mail aliases must be set up on `mailhub`, which will expand the recipient's mail address (for example, `fred@acme.co.uk`) to the recipient's mail server address (for example, `fred@mars.acme.co.uk`).

By this means, all mail for `fred`, whether at `acme.co.uk` or `mars.acme.co.uk`, will be directed to `fred`'s mail server.

Of course, the mail administrator must ensure that names are not duplicated in the mail alias database. For example, `fred@mars.acme.co.uk` and `fred@mercury.acme.co.uk` must each have a unique alias on `mailhub`.

10-22. SLIDE: Configuring a Mail Hub

Configuring a Mail Hub

A Mail Hub

- is a host with links to the outside world
- acts as a central funnel for outgoing and incoming mail
- may be a site's Internet mail firewall
- may function as mail gateway translating mail addresses

Configuration:

- Set up an MX record pointing to your *mailhub*.
- Configure site hiding inside your domain.
- Announce *mailhub* to all hosts inside your domain.
- On the *mailhub* define *site hiding name* as local.
- Set up aliases in the *mailhub*.

a64691

Student Notes

Mail Hub

A mail hub is a system with links to the outside world. It is used as a channel for all internal mail going to the outside world and for all incoming mail arriving from the outside world. Typically it is your network's Internet router, and also may have been set up as a firewall router and DNS name server.

In order to set up your network's mailhub it is necessary to perform several steps.

For the steps below assume your domain name would be

internal.domain

and the name of the main hub would be

mailhub.

Setting up the DNS Nameserver

Add a new DNS mail exchanger record to deliver mail which is addressed to the **internal.domain** to the domain itself by giving the domain name the IP address of the mail hub system. For instance, if the mail hub for domain **internal.domain** had address 193.118.172.134, then add the following records to **db.internal.domain**:

```
internal.domain.      IN  MX   5  internal.domain.  
internal.domain.      IN  A    193.118.172.134
```

NOTE:

The domain name for the SOA entry—referenced in the **named.boot** file—here is the same as the MX record. Use fully qualified names for MX records so that the MX record is not expanded by the domain name.

Three entries refer to **internal.domain**:

- | | |
|-----|--------------------------------------|
| SOA | the name of the domain itself |
| MX | the name of a pseudo mail exchanger |
| A | a hostname pointing to an IP address |

All three entries are specified in the same database.

Setting up the Mail Hub

Announce all names for which **mailhub** accepts incoming mail. This may be done in either of two ways:

1. Define the class **w**, which specifies all the alias and alternate names for **mailhub** as well as other names for which **mailhub** will receive mail:

```
Cwlocalhost mailhub mailhub.internal.domain internal.domain
```

2. Place all the alternate names into a file, then define that file as the source for those names:

```
Fw-o /etc/mail/sendmail.cw
```

mailhub reads the file `/etc/sendmail.cf` file to get alternate names for the host. This might be used if you were on a host that **mxed** for a dynamic set of other hosts.

Set up aliases on the mail hub for all your users so that the mail hub knows which actual host to forward mail to for each user.

```
mailhub# vi /etc/mail/aliases
at the end add lines such as
mikea: mikea@herhost
fredf: fredf@hishost
hackers: hackers@yourhost
...
mailhub# newaliases
```

Setting up Mail Hosts Inside the Internal Domain

Configure all mail systems to use *site hiding (masquerading)* so that mail leaving your environment does not mention the *own* host which originated the mail, but instead uses just the domain name. This means that when users in the outside world *reply* to mail your network has sent out, it will be addressed to `user@domainname`, which ties in with the DNS mail exchanger records for the domain name itself. This is accomplished by finding, uncommenting and changing the following line in `sendmail.cf`:

DMinternal.domain

The DM Macro only masquerades the header address. In order to change the envelope names, too, uncomment first rule `#R$+` of ruleset 94:

```
#####
### Ruleset 94 -- convert envelope names to masqueraded form #####
#####

S94
R$+           $@ $>93 $1
R$* < @ *LOCAL* > $*   $: $1 < @ $j . > $2
```

Configure all mail systems to use *mailhub* as an SMTP relay host by modifying the `sendmail.cf` file.

Define the **S** macro to be the hostname of *mailhub*

DSmailhub.internal.domain

10-23. SLIDE: Mail Servers and Clients

Mail Servers and Clients

A Mail Server

- runs `sendmail` as a daemon
- exports `/var/mail` to mail clients via NFS
 - server configuration in `/etc/rc.config.d/mailservs`
`export SENDMAIL_SERVER=1`

A Mail Client

- does not run `sendmail`
- mounts `/var/mail` from the mail server via NFS
- forwards mail automatically to its mail server
- uses site hiding
 - client configuration in `/etc/rc.config.d/mailservs`
`export SENDMAIL_SERVER_NAME="mailserver"`

a64692

Student Notes

Mail Server

A mail server is a system which runs in server mode. This means that it holds mailboxes for other systems and users on the network and that it runs the `sendmail` program in daemon mode. The mail server holds mail files for other systems and users by NFS exporting the mail directory `/var/mail` which the mail clients mount as `/var/mail`. It is advisable to make the `/var/mail` directory a separate file system mount point in a large mail environment. The mail server system may or may not have links to the outside world—that is, it may or may not also be a site's mail hub. In order to set up a mail server system, the following steps should be performed:

1. Become an NFS server exporting `/var/mail`.
2. Edit `/etc/rc.config.d/mailservs` and set the following variables:

```
export SENDMAIL_SERVER=1
export SENDMAIL_SERVER_NAME=""
```

3. OPTIONAL: Add an alias such as `mailserver` to the hostname of this system in the name service in use by the network (NIS, DNS).

Mail Client

A mail client is a system which NFS mounts its mail files from its mail server and which forwards all outgoing mail to its mail server. Since its mail files are not local, there is no need for the mail client to run the `sendmail` program in daemon mode, which offers considerable savings in system resources. To set up a system as a mail client, the following steps should be followed:

1. Become an NFS client, mounting `/var/mail` from the mail server system as `/var/mail` (use the `mailserver` alias if available). For instance, add the following line to `/etc/fstab`:

```
mailserver:/var/mail /var/mail nfs rw,bg 0 0
```

2. Edit `/etc/rc.config.d/mailservs` and set the following variables:

```
export SENDMAIL_SERVER=0
export SENDMAIL_SERVER_NAME="mailserver"
```

10-24. SLIDE: /etc/hosts - Based Name Resolution

/etc/hosts-Based Name Resolution

- mixing DNS and /etc/hosts-based name resolution needs additional attention
- edit `sendmail.cf`

```
Cwlocalhost myname myname.pseudo.domain  
Dj$jw.pseudo.domain
```

a64693

Student Notes

Normally, the \$j macro is automatically defined to be your fully qualified domain name. Sendmail does this by getting your hostname using `gethostname` and then calling `gethostbyname` on the result. For example, in some environments `gethostname` returns only the root of the hostname (such as `foo`); `gethostbyname` is supposed to return the fully qualified domain name (`foo.bar.com`). In some cases, `gethostbyname` may fail to return the fully qualified domain name. This will be the case, if your host `foo` doesn't request the DNS name server but your local `/etc/hosts` file. In order to communicate with the other DNS resolving hosts, you *must* define a fully qualified domain name. This is usually done using:

```
Cwlocalhost myname myname.pseudo.domain  
Dj$jw.pseudo.domain
```

10-25. SLIDE: Basic sendmail Checks

Basic sendmail Checks

- Check **sendmail** installation.
- Check **sendmail** is working.
`ps -ef | grep sendmail`
- Check configuration files.
`sendmail -C filename`
- Send test mail.
`sendmail -v user`

a64694

Student Notes

Check that sendmail is Installed

Usually it is enough to check for the existence of the **sendmail** daemon and configuration file:

```
$ ls -l /usr/sbin/sendmail  
-r-sr-sr-t 1 root mail 176128 May 17 03:18 /usr/sbin/sendmail  
$ ls -l /etc/mail/sendmail.cf  
-r--r--r-- 1 bin bin 75888 May 12 15:35 /etc/mail/sendmail.cf
```

Check that **sendmail** is Running

To check whether or not the local **sendmail** daemon is running:

```
$ ps -ef | grep sendmail  
daemon 116 1 0 13:26:47 ? 0:00 sendmail -bd -q30m -accepting connections
```

To check whether a remote **sendmail** daemon is running you can talk directly to it, by contacting SMTP port 25.

```
$ telnet hpodwa 25  
Trying...  
Connected to hpodwa.pwd.hp.com.  
Escape character is '^]'.  
220 hpodwa.pwd.hp.com Sendmail ready at Wed, 17 Jun 92 18:46:46 +0100:  
he1o  
250 hpodwa.pwd.hp.com Hello (hpodwa.pwd.hp.com), pleased to meet you  
quit  
221 hpodwa.pwd.hp.com closing connection  
Connection closed by foreign host.  
$
```

If your connection is refused, try contacting that remote daemon again from another host. If that works, the remote **sendmail** daemon is up, and there is probably something wrong with the first network connection you tried rather than with **sendmail**. Try **/etc/ping**, **telnet** to a user rather than SMTP, or check **/etc/hosts**.

Check that **sendmail** is Working

- Invoke **sendmail** directly from the command line to mail to yourself (as root):

```
/usr/sbin/sendmail -v root  
Hello, here is a message for Sendmail  
. or <ctrlD>
```

- To send mail to a remote user:

```
/usr/sbin/sendmail -v <user>@<host>
```

- To check that UNIX mail is working:

```
mail root
Hello, here is a message for mail
. or <cntlD>
```

In the last example, `/bin/mail` executes `/usr/sbin/sendmail` to route the mail, which in turn executes `/bin/rmail -d` to deliver local mail.

Check that the Configuration File is Working

If you change the configuration file, save the old working version, and test the new file with the `-C` option. For example:

```
/usr/sbin/sendmail -Ctest.cf
```

10-26. SLIDE: **sendmail Diagnostic Options**

sendmail Diagnostic Options

sendmail has a number of options that can be useful when troubleshooting:

- **\$ sendmail -bt**
- **\$ sendmail -bv**
- **\$ sendmail -q**
- **\$ sendmail -v**

a646148

Student Notes

- **sendmail -bt**

The **-bt** option runs in address test mode, which can be useful to debug configuration files. With OpenMail, you can use this to check UNIX format addresses that don't seem to be recognized by OpenMail:

```
$ /usr/sbin/sendmail -bt
ADDRESS TEST MODE
Enter <ruleset> <address>
> 0 bevan_jeremy/rabbit_ux
rewrite: ruleset 3 input: "bevan_jeremy" "/" "rabbit_ux"
rewrite: ruleset 3 returns: "bevan_jeremy" "/" "rabbit_ux"
rewrite: ruleset 0 input: "bevan_jeremy" "/" "rabbit_ux"
rewrite: ruleset 6 input: "bevan_jeremy" "/" "rabbit_ux"
rewrite: ruleset 6 returns: "bevan_jeremy" "/" "rabbit_ux"
rewrite: ruleset 0 returns: "^V" "openmail" "^W" "hpopdwd" "." "pwd"
"." "hp" ".." "com" "^X" "bevan_jeremy" "/" "rabbit_ux"
```

```
> <CntlD>
```

Where:

```
"^V"  stands for $# as in mailer
"^\$W"      $@          host
"^\$X"      $:          user
```

- **sendmail -bv**

Verify names only. Don't try to collect or deliver mail. This can be useful for validating users or mailing lists, which show the results of aliasing, forwarding, and so forth, on the local system. Here it shows that mail for **rogerw** will be successfully forwarded into OpenMail by a **.forward** file:

```
$ /usr/sbin/sendmail -bv rogerw
williams_roger/lpc_admin@hpod... deliverable, mailer=openmail,
host=hpod.pwd.hp.com, user=williams_roger/lpc_admin
```

- **sendmail -q**

Process the mail queue immediately, instead of waiting for the configured processing interval:

Kill the current **sendmail** daemon:

```
/usr/bin/killsm
```

Process the mail queue immediately:

```
/usr/sbin/sendmail -q
```

To watch the queue being processed, use a verbose **-qv** option (see below).

- **sendmail -v**

Verbose mode. Alias expansions and the complete SMTP transaction used to transfer the message between local and remote hosts are output to **stdout**.

10-27. SLIDE: The Mail Statistics File

The Mail Statistics File

- Logged in **sendmail.st**
- Configured by **sendmail.cf**

a630217

Student Notes

sendmail can keep message statistics, reflecting the amount of message traffic it has processed. These are kept on a per-mailer basis (as defined in the Configuration File), and distinguish between inbound and outbound traffic. Statistics are logged to the file **/etc/mail/sendmail.st**, only if it exists. To view the message statistics, type:

```
$ mailstats

Statistics from Wed Jun 17 17:52:39 1992
M msgsfr bytes_from msgsto bytes_to
0     18       18K      3       3K
2      0        0K      7       9K
7      1        1K      8       8K
8      6       12K      1       0K
```

M designates the mailer involved; the number reflecting its position in the list of mailers in the configuration file. In this case, 0 is the local mailer (**rmail**), 7 is OpenMail's Unix

Gateway (**unix.in**) and 8 is OpenMail's **sendmail** Interface (**xport.in**), as can be found out from the listing below.

The central columns list the number of messages and total size in bytes of all messages passed from that mailer to **sendmail**, and from **sendmail** to that mailer.

To list the mailers defined in the configuration file:

```
$ grep "^\M" /etc/mail/sendmail.cf

Mlocal, P=/bin/rmail, F=DFMPlms, S=10, R=20, A=rmail -d $u
Mprog, P=/bin/sh, F=DFMPlshu, S=10, R=20, A=sh -c $u
Mtcp, P=[IPC], F=CDFMXmu, S=11, R=21, E=\r\n, A=IPC $h
Muutcp, P=[IPC], F=CDFMXmuU, S=11, R=21, E=\r\n, A=IPC $h
Muucp, P=/usr/bin/uux, F=DFMUshu, S=13, R=23, A=uux - $h!rmail ($u)
Mdumbuucp, P=/usr/bin/uux, F=DMUshux, R=23, A=uux - $h!rmail ($u)
Mx400, P=/usr/lib/x400/x4mailer, F=CDMFmn, S=14, R=24, A=x4mailer -f \
      $g $u
Mopenmail, P=/usr/openmail/bin/unix.in, F=DFLMXmnu, S=15, R=25, \
      A=unix.in
Momxport, P=/usr/openmail/bin/xport.in, F=LMn, A=xport.in $u
```

To clear the stats file, as root, type:

```
cp /dev/null /etc/mail/sendmail.st
```

10-28. SLIDE: sendmail's Logfile

— **sendmail's Logfile** —

Check **sendmail's** log file:

/var/adm/syslog/mail.log

Check for **stat** fields containing a value other than **Sent**

Increase log level:

sendmail -bd -q30m -oLloglevel

a64695

Student Notes

- When troubleshooting, it can be helpful to put a line of dashes at the end of the current **syslog** file, so you can easily locate where new entries begin.
- If there are several messages for the same destination in the log file, since no subject is given, you may need to increase the logging level in order to distinguish them.

As root:

- Kill the **sendmail** daemon: **\$ /usr/sbin/killsm**
- Start **sendmail** in verbose mode: **\$ /usr/sbin/sendmail -bd -q30m -oL11**

See log level description below.

The System Log

`sendmail` maintains a log of its activities in the file `/var/adm/syslog/mail.log`. Like most things, this is configurable, however `mqueue/syslog` is an accepted standard. Each line in the log file contains a time stamp, the message ID, and some information logged by `sendmail`, including the message's *stat*.

Here's an example of a system log. (To make it easier to read, some standard fields like the date have been removed, and associated groups of lines have been separated.)

```

14:38:35 [706]: queue run started, interval=30
14:38:35 [706]: dropenvelope, id=(none), flags=1, pid=706
14:38:35 [706]: accepting SMTP connections

15:55:30 [714]: AA00714: message-id=<9206181450.AA00714@spock>
15:55:30 [714]: AA00714: from=rogerw, size=125, class=0, received from
  local host
15:55:30 [714]: dropenvelope, id=(none), flags=1, pid=714
15:55:32 [715]: AA00714: to=ianb,markl, delay=00:00:02, stat=Sent, mailer=local
15:55:32 [715]: dropenvelope, id=AA00714, flags=1, pid=715

16:06:45 [728]: AA00728: message-id=<9206181525.AA00728@spock>
16:06:45 [728]: AA00728: from=rogerw, size=110, class=0, received from
  local host
16:06:45 [728]: dropenvelope, id=(none), flags=1, pid=728
16:06:47 [729]: AA00728: to=simonb@uhura, delay=00:00:03, stat=Sent,
  mailer=tcp, host=uhura, MX host=uhura, address=15.144.34.67
16:06:47 [729]: dropenvelope, id=AA00728, flags=1, pid=729

16:37:36 [735]: AA00735: message-id=<9206181435.AA00735@spock>
16:37:36 [735]: AA00735: from=openmail, size=133, class=0, received from
  local host
16:37:36 [735]: dropenvelope, id=(none), flags=1, pid=735
16:37:41 [736]: AA00735: to=openmail@klingon, delay=00:00:07, stat=Sent,
  mailer=tcp, host=hpopdx, MX host=hpopdx, address=15.146.33.102
16:37:41 [736]: dropenvelope, id=AA00735, flags=1, pid=736

17:20:31 [747]: AA00747: message-id=<H00000650000311e@MHS>
17:20:32 [747]: AA00747: from=Williams_Roger/lpc_admin@spock, size=167,
  class=0, received from local host
17:20:32 [747]: dropenvelope, id=(none), flags=1, pid=747
17:20:34 [745]: dropenvelope, id=(none), flags=1, pid=745
17:20:34 [748]: AA00747: to=markl@spock, delay=00:00:04, stat=Sent, mailer=local
17:20:34 [748]: dropenvelope, id=AA00747, flags=1, pid=748

18:30:11 [813]: AA00813: message-id=<9206181530.AA00813@spock>
18:30:11 [813]: AA00813: from=rogerw, size=22, class=0, received from
  local host
18:30:11 [813]: dropenvelope, id=(none), flags=1, pid=813
18:30:16 [815]: AA00813: to=brown_elizabeth/lpc_uxin, delay=00:00:06,
  stat=Sent, mailer=openmail, host=spock
18:30:17 [815]: dropenvelope, id=AA00813, flags=1, pid=815

```

Definition of each numbered group of entries in the syslog:

706	the startup of the sendmail daemon
714/5	UNIX mail via local mailer from rogerw to ianb and markl
728/9	UNIX mail via TCP from rogerw@spock to simonb@uhura
735/6	OpenMail via TCP from local openmail@spock to openmail@klingon
747/8	OpenMail->UNIX mail via local mailer (to markl@spock)
813/5	UNIX mail->OpenMail via openmail mailer (from rogerw@spock)

Logging can be set by the **-oL** option on **sendmail**'s command line or by specifying the **L** option in the configuration file.

The logging levels can be set as:

0	no logging
1	major problems only
2	message collections and failed deliveries
3	successful deliveries
4	messages being deferred
5	messages being added to the queue
6	unusual but benign events
9	log queue ID to message ID
10	log destination mailer, host, user of each message
11	log Internet address of each connection
12	debugging mode
16	verbose information on the queue

Logging levels up to 10 are most often used. (The example was logged at 11).

The system log file grows with each message that passes through the system. You can manage it by writing a script that copies the log to a save file each night and sets up a new file each night.

Such a script called **newsyslog** could be run each night by putting the following entry into root's **crontab** file:

```
0 0 * * * /var/adm/newsyslog
```

Where **/var/adm/newsyslog** might contain:

```
#!/bin/sh

day='date '+%a''
cd /var/spool/mqueue
rm -f syslog.$day
cp syslog syslog.$day
cat /dev/null > syslog
/usr/sbin/killsm
sleep 30
/sbin/rm -f /usr/spool/mqueue/[lnx]f*
/usr/sbin/sendmail -bi > /dev/null
/usr/sbin/sendmail -bd -q30m
```

10-29. SLIDE: sendmail's Mail Queue

sendmail's Mail Queue

- undeliverable mail is queued in `/var/spool/mqueue`
- queue is processed each `-qinterval`
`sendmail -bd -q30m`
default timeout for undeliverable mail: 3 days
- mail is printed
`mailq`

a64696

Student Notes

The Mail Queue

`sendmail` uses the Mail Queue `/var/spool/mqueue` to hold mail waiting to be processed. Once a message has been sent, it is deleted from the queue. A message that `sendmail` will not be able to deliver—because, say, `sendmail` doesn't know about the remote host—is returned to the originator. A message that cannot be delivered immediately but which is likely to succeed later—say due to a temporarily unavailable connection—stays in the queue for later delivery.

Effectively, when you send a message it is processed and sent straight away. If however it cannot be sent now, it remains on the queue and is re-processed when the next `-q<interval>` has elapsed.

The queue is processed regularly by the `sendmail` daemon, at the interval specified by the `-q` option. This reads and sorts the queue, and attempts to process each message in order of priority. It ignores locked jobs.

The mail queue can be printed with the following commands:

mailq (for HP-UX 11.00)

or

/usr/sbin/sendmail -bp (for HP-UX 10.x or earlier)

For each message in the queue, this lists: queue IDs, message size (in bytes), date the message entered the queue, and the sender and recipient addresses. For example:

```
Mail Queue (2 requests)
--Q-ID-- --Size-- -----Q-Time----- Sender/Recipient-----
AA15841      86  Wed Jun 9 07:08 <janet>
(Deferred: Connection refused)
<ees@vetmed.und.edu>
<ebs@surv.ob.com>
AA02748      16  Thu Jun 18 13:46 <root>
(Deferred)
<williams_roger/lpc_admin@hpopdwa>
```

Each message can generate the following types of files in the mail queue directory (**/usr/spool/mqueue**), where AAnnnn is the message ID:

dfAAnnnn	data file (message body)
qfAAnnnn	job control file (job processing information, including header)
lfAAnnnn	lock file (job synchronizer - exists if job is being processed)
nfAAnnnn	job creation file (exists while message ID is being created)
tfAAnnnn	temporary file (image of qf file, used during a queue rebuild)
xfAAnnnn	transcript file (contains a record of the job)

The base file name is constructed as follows:

A?nnnn

Where:

- *nnnn* is the process id of the creating process
- ? is a letter (starting at *A*) that further ensures a unique name

qf files are useful in checking on the status of messages in the queue.

Here's an example **qf** file:

```
$ more qfAA02771
P2301          <-- Priority of msg in the mqueue
T708872444    <-- creation time (seconds since 1970)
DdfAA02771    <-- data file name
MDeferred     <-- status
S<Williams_Roger/pinewood_ls_hpopd@hpopd.pwd.hp.com>
```

Configuring sendmail

```
Rlauder_mark/lpc_admin@hpopdwa
HReceived: from hpopd.pwd.hp.com by hpopdwa.pwd.hp.com with SMTP
          (16.6/15.5+IOS 3.20) id AA02771; Thu, 18 Jun 92 14:00:44 +0100
H?x?full-name:
H?P?return-path: <Williams_Roger/pinewood_ls_hpopd@hpopd.pwd.hp.com>
HReceived: from by hpopd.pwd.hp.com with SMTP
          (16.6/15.5+IOS 3.22) id AA26125; Thu, 18 Jun 92 13:52:49 +0100
H?F?From: Williams_Roger/pinewood_ls_hpopd@hpopd.pwd.hp.com
Hx-openmail-hops: 1
Hdate: Thu, 18 Jun 92 13:51:49 +0100
Hmessage-id: <P00000d600afa106@MHS>
Hsubject: Test Message
HTo: markl@hpopdwa.pwd.hp.com
```

10-30. SLIDE: Checking Error Messages

Checking Error Messages

- **Service Unavailable**
- *hostname Unknown host*
- **Deferred: Connection refused**
- **Deferred: Connection timed out**
- **Deferred: Not a typewriter**
- **No route to host**
- **No such user**
- **Syntax error: command unrecognized**
- **System file missing**
- **Unknown mailer**

a64697

Student Notes

- **Service Unavailable**

Commonly seen when attempting to start the `sendmail` daemon. The `/etc/services` file on the local system does not contain the appropriate entry for the SMTP service.

- *hostname Unknown host*

The local host does not know about the remote host. Check to see if an entry for the remote host exists in the `/etc/hosts` file on the local host.

- **Deferred: Connection refused**

The remote host is not accepting connection requests for SMTP transactions. Check to see if the remote host is running an SMTP server (`sendmail` daemon) on port 25. Prove this by Telnetting to port 25.

- **Deferred: Connection timed out**

The remote system is down, not responding to ARP requests, or did not respond within the configured timeout set in the **sendmail Configuration File** (usually 5 minutes).

Check to see if the system is available over the network with **/usr/sbin/ping**. With OpenMail, check that SMTP didn't timeout during the transfer of a very large message.

- **Deferred: Not a typewriter**

Seen in the message queue status information when a non-system error occurs, and the message is deferred. This message is nowadays usually erroneous and should be ignored.

- **No route to host**

The local SMTP client is attempting to contact a remote system that is down, or is not responding to ARP requests. Verify that the remote system is up and communicating over the network with **/usr/sbin/ping**.

- **No such user**

sendmail error number 501. A message coming in UNIX mail contains a name with invalid X.400 characters in it or has no surname. Try running **sendmail -bv**.

- **Syntax error: command unrecognized**

sendmail error number 500. Two versions of **sendmail** on different hosts are out of step. The log file should give more information.

- **System file missing**

Make sure the local **/etc/services** file contains an appropriate entry defining TCP port 25 for SMTP.

- **Unknown mailer**

Occurs when a delivery agent encounters problems and returns an error code to **sendmail**. Since **sendmail** is not capable of interpreting error conditions, it simply sends all available information back to the user via the returned message. Probable causes are that the transport agent could not be executed successfully or there are inconsistent mailer flags in the mailer definition in the configuration file.

10-31. LAB: Configuring a Mail Hub

Directions

This lab refers to the last lab of DNS. To finish this lab successfully the DNS example must be configured properly.

We will define two internal mail domains:

- *eurasia*
- *south.america*

The mail hub for *south.america* is: **santiago** The mail hub for *eurasia* is: **geneve**
paloalto and **dallas** represent the world.

One of the remaining hosts without a special function, either

saopaulo, **pinewood**, or **singapore**

should not use DNS name resolution but **/etc/hosts** name resolution instead.

Depending on the function of your system, set up your configuration accordingly.

1. On the DNS primary name servers, set up DNS MX records for these domains:

1. *eurasia* and
2. *south.america*

2. Don't forget to let **named** re-read the configuration files.

3. Announce all alternate names for which the mail hub shall accept mails.

4. Setup mail aliases for all users in your domain in the **/etc/mail/aliases** database. Don't forget to hash your database.
5. Implement Site Hiding on your mail hub.
6. Configure all mail hosts in the domain **eurasia``** and **``south.america** to use Site Hiding.
7. Define the **geneve** as a *smart* mail hub for non-local mail.
8. Restart the **sendmail** daemon.
9. Mail to such a host will be delivered with the fully qualified domain name.
For this define alternate names in the **Cw** class.
10. In order to communicate with the other DNS resolving hosts, define a fully qualified domain name.

11. Try sending mail to the following:

```
mynname  
mynname@myhost  
mynname@myhost.mydomain  
mynname@mydomain  
othername@otherdomain  
worldname@worlddomain
```

When the mail message comes back to you, read the header lines from the bottom up carefully to ensure you understand the routing.

Appendix A — IBM AIX Operating System

Objectives

Upon completion of this module you will be able to do the following:

- Document the differences between IBM (AIX) and HP-UX.

A-1. SLIDE: AIX UNIX Network Configuration

AIX UNIX Network Configuration

Basic steps:

1. Ensure appropriate software is installed. TCP/IP is a separately installed product which is not part of the BOS package.
2. Configure the hardware interfaces installed with the correct protocol drivers for the network you are connecting to.
3. Design network, assign numbers, hostnames, and so forth.
4. Configure appropriate system services.

a646168

Student Notes

After installation of the Base Operating System from the install media, the TCP/IP components of the operating system must be installed. This is easily achieved using the **smit installpkg** option. The components to incorporate are as follows:

bosext2	Contains IEEE data link control Standard Ethernet data link control
bosnet.ncs	Contains Network computing system
bosnet.nfs	Contains the NFS/NIS/RPC programs and libraries
bosnet.snmp	Contains SNMP programs
bosnet.tcpip	Contains TCP/IP applications

Once these software components are in place, the system needs to be configured to utilize the TCP/IP software facilities. This involves the **mktcpip** script which can be front-end driven by

the **smit** interface i.e. **mktcpip**. It simply allows the administrator to set the hostname, Internet address, subnetmask and optional domain name server for the machine. The values are stored within the Object Database and at boot time are configured by the object database manager from the script **/etc/rc.net**. This script can also be edited to provide traditional setting of the interfaces required via the **ifconfig** command.

To set the initial parameters via the command line interface (we recommend the **smit** front end utility be used) the following could be used:

```
mktcpip -h mars -s 192.20.0.1 -m 255.255.255.0 -I en0 -t bnc
```

NOTE:

There are two interfaces that can be specified for Ethernet connections.
en0 corresponds to Ethernet II and **et0** corresponds to IEEE802.3-frame-type Ethernet.

There are two interfaces that can be specified for normal Ethernet cards:
bnc for normal barrel connectors and **dix** for AUX connectors.

A-2. SLIDE: Configuring More AIX Networking Parameters

Configuring More AIX Networking Parameters

1. Use **smit**

```
Communications Applications and Services
|
v
TCP/IP
|
v
Further Configuration
```

2. Edit the files directly.

a646169

Student Notes

After configuring the basic communications under TCP/IP further administration of Domain Name Services, NIS services and so on can be achieved by utilizing the **smit** menu under the TCP/IP section or editing the files directly. **smit** is the preferred option for administrators who are not completely familiar with AIX as it provides on-line help and clear diagnostic messages.

All set-up can be achieved by editing the various files under the **/etc** directory. The following slide details the various files that can be manipulated.

A-3. SLIDE: Main AIX Configuration Files

Main AIX Configuration Files

/etc/rc.net	configures interfaces and addresses from ODM
/etc/rc.bsdnet	alternative Berkeley start-up file
/etc/rc.nfs	starts all NFS processing
/etc/rc.tcpip	starts all TCP/IP related processes

a690251

Student Notes

/etc/rc.net

This script is called by the configuration database manager, **cfmgr** during the boot phase. It interrogates the ODM for the defined interfaces and configures them. It binds the address to each interface and also configures and enables any SLIP interfaces defined. The administrator can edit this file and add traditional **ifconfig** commands to configure the interfaces within the system. All defined and configured interfaces can be detailed with the following command:

```
# lsdev -C -c if
lo0 Available  Loopback Network Interface
en0 Available  Standard Ethernet Network Interface
```

et0 Defined IEEE 802.3 Ethernet Network Interface

/etc/bsdnet

This script can be used in preference to the **rc.net** if the BSD compatibility option is requested by the administrator in the further configuration menu section within **smit**. It contains the familiar **ifconfig**, **hostname** and **route** commands. Generally this file is not used but is useful for SUN/Solaris administrators.

/etc/rc.nfs

This configuration file is responsible for starting all of the NFS services within the system. Configuration within **smit** simply edits this file to turn on/off the various utilities such as NIS and DNS. Again this file can be edited manually to implement NFS within the system.

/etc/rc.tcp

This file starts all of the configured TCP/IP services such as **inetd**, named, the port mapper and so on. Each daemon is started by utilizing the **startsrc** command which sends the System Resource Controller (SRC) a request to start a subsystem or a group of subsystems. Again, the user can manually edit this file or use the **smit** interface to enable and disable various sub-systems.

A-4. SLIDE: Configuring BIND under AIX

Configuring BIND under AIX

The operation of BIND is very similar to that for other versions of UNIX.

Configuration is achieved by a set of standard **awk** scripts that are supplied with BOS.

The named server processes all DNS requests. This process is started by
/etc/rc.tcpip.

a646170

Student Notes

The DNS components of AIX are started from the script **/etc/rc.tcpip**. The process is very similar to that detailed for HP-UX. The set-up and configuration of the files is achieved by running a number of supplied **awk** scripts. Full details and information is available by referencing the manual page for **named**. The following files and scripts are located within the sub-directory **/usr/lpp/tcpip/samples** and should be referenced when setting up DNS under AIX:

- | | |
|-------------------|--|
| named.boot | Contains a sample named.boot file which can be amended and copied to /etc/named.boot . |
| named.data | Contains the sample Domain data file. It is good practice to point named.boot to a new directory such as /var/named and place all of the configuration files within this directory. The default directory is /etc . |
| hosts.awk | Contains a sample awk script that converts the /etc/hosts file into the domain file. The standard output generated should be captured to create the file i.e. hosts.awk /etc/hosts > /etc/named.data |

addrs.awk Creates **named.rev** from **/etc/hosts**

A-5. SLIDE: The AIX BIND Boot File

The AIX BIND Boot File

/etc/named.boot	the default named boot file
Cache	points to file holding root server addresses
Primary	designates server addresses for the named zone
Secondary	designates secondary servers for the named zone
Forwarders	points to other servers capable of performing recursive queries
Slave	forces slave mode for this server
Sortlist	indicates networks that take precedence over other networks

a646161

Student Notes

The **named.boot** file is read by the **named** daemon on start-up from **/etc/rc.tcpip**. The location is always **/etc** unless the **-b** flag is used to redirect the input from another file. The **named.boot** file is similar to those previously discussed but has an optional **sortlist** entry.

sortlist indicates the networks that should take precedence over other networks. Requests for name resolution from a host on the same network as the server receive local network addresses first, addresses on the **sortlist** second, and all other addresses listed last. The **sortlist** line is only acted upon at initial start-up by **named**. Subsequent reloading of the name server ignores this line.

Once the administrator has created these files and moved them into the correct directories, the **named** process must be started via the **smit** fast path entry, **stnamed**. This allows the **named** process to be started immediately, and depending upon response, at subsequent boot times:

Start Using the named Subsystem

Move cursor to desired item and press Enter.

NOW
Next System RESTART
BOTH

To start the **named** daemon from the command line prompt, the following must be entered:

```
startsrc -s named
```

The **named** daemon is a subsystem controlled by the system resource controller (SRC), and as such must be started using the **startsrc** command. This is the case for most daemon processes within AIX.

A-6. SLIDE: Debugging BIND under AIX

Debugging BIND under AIX

Various signals and flags are available to enable debugging of DNS under AIX:

named -dN The **-d** flag with **named** causes debug output to be generated within the file **/var/tmp/named.run**. *N* ranges from 1 to 11 with 11 producing most output.

Signals are available to debug a running **named** process.

SIGHUP	causes named to reread named.boot and reload database
SIGINT	dumps database to /var/tmp/named_dump.db
SIGIOT	dumps statistics for named use into /var/tmp/named.stats
SIGABRT	/var/tmp/named.stats (output is appended)
SIGUSR1	turns on debugging. Each subsequent signal increases level.
SIGUSR2	sends output to /var/tmp/named.run
	turns off debugging completely

a646171

Student Notes

There are various flags and signals available to enable successful debugging of the **named** process. **named** under AIX responds to the same signals as the other versions of UNIX detailed within this course (output being placed under **/var/tmp**). In addition to the signals detailed within the slide, AIX has an additional command to enable trace logging to be started on a daemon process. The following command turns on trace logging for the **named** process. The same result could have been achieved by sending the **named** process signal SIGUSR1.

```
traceson -s named
```

Becoming A Nameserver Client

To set up the AIX system to become a DNS client, the administrator must edit the **/etc/resolv.conf** file manually or use **smit** to create and update entries within the file. The **smit** option within the domain name server component of TCP/IP further configuration allows this to be achieved by executing the **namers1v** command. For example:

```
namerslv -b -i '192.20.0.6' -D 'hp.com.uk'
```

The administrator can also create the file manually. If the machine is to be the main server then the `/etc/resolv.conf` file must exist and be empty. The presence of `/etc/resolv.conf` file tells the local kernel and resolver routines to use the domain protocol. If the `/etc/resolv.conf` file does not exist, the local kernel and resolver routines use the `/etc/hosts` file which causes the `named` process to operate abnormally.

NOTE:

On the domain name server, the `/etc/resolv.conf` file must exist but must be EMPTY.

A-7. SLIDE: sendmail under AIX

sendmail under AIX

Steps required to configure **sendmail** under AIX

1. Configure the **/etc/rc.tcpip** to start **sendmail** daemon.
2. Customize the configuration file **/etc/sendmail.cf**.
3. Define systemwide and domainwide mail aliases in the **/etc/aliases**.
4. Manage the mail queue.
5. Manage the mail log.
6. Customize the **/etc/sendmail.nl** file.

a646172

Student Notes

The **sendmail** package is supplied as standard with the Base Operating System. By default, **sendmail** is started automatically within the **/etc/rc.tcpip** script. If this is not the case, uncomment the **sendmail** command line within the file. Once the daemon has started, the **sendmail.cf** file must be customized for your particular set-up and then compiled.

A-8. SLIDE: sendmail Configuration under AIX

sendmail Configuration under AIX

The `sendmail` file supplied provides rules for TCP/IP and BNU mail forwarding using a domain address structure. IBM suggests the administrator may want to change the following:

HostName class	By default the hostname component is used for hostname resolution.
HostName macro	These two entries can be edited to reflect a different name structure.
Domain macro	Four macros define the separate tokens of the DomainName :

operational logging level
default delivery mode
alias file path
statistics file path

a646173

Student Notes

The default `/etc/sendmail.cf` file as supplied with the base installation contains rules to perform the translation for BNU and TCP/IP networks using a domain address structure. This configuration is usually sufficient for most administrators but can be amended to reflect certain site specific addressing details. The `vi` editor should be used to change the file as spaces and tabs can have different meanings when specifying the delimiter characters.

The rulesets detailed within the file can be found detailed earlier in this module. Once the configuration file has been changed to reflect the site details it must be re-compiled.

A-9. SLIDE: Compiling and Rereading the `sendmail.cf` File

Compiling and Rereading the `sendmail.cf` File

`/usr/sbin/sendmail -bz` Recompiles the `sendmail.cf` file. This creates the file `/etc/sendmail.cfDB`.

The `sendmail` daemon must now be instructed to reread the compiled file. Depending on how the daemon is started, the following commands are used:

`refresh -s sendmail` for `startsrc` invocation

`kill -l 'cat /etc/sendmail.pid'` for `/usr/sbin/sendmail` invocation

a646174

Student Notes

The `sendmail.cf` file is compiled for speed into a binary file using the `sendmail -bz` command. The file created is `/etc/sendmail.cfDB`. However, the `sendmail` daemon now has to be notified of the new file. Depending on how the `sendmail` process was started, either a signal is sent to the `sendmail` program or, the System Resource Controller refreshes the daemon process.

If `sendmail` was started using the system resource controller (`startsrc` command), the following command should be executed:

```
refresh -s sendmail
```

This is the default method of starting the `sendmail` daemon from `/etc/rc.tcpip`. The administrator can however start the process by deleting this entry and running the `sendmail` program directly from the `/etc/rc.tcpip` script. If this later option is chosen, SIGHUP must be sent to the `sendmail` process to force the new file to be read. The pid of the `sendmail` process is stored in the file: `/etc/sendmail.pid`.

A-10. SLIDE: Creating System Mail Aliases

Creating System Mail Aliases

Edit the local `/etc/aliases` file.

If NIS is configured for `sendmail` (OP option set), edit NIS alias file.

Compile the aliases file: `/usr/sbin/sendmail -bi`.

a646159

Student Notes

The `aliases` file allows the administrator to set up aliases for users or groups or users. The `aliases` file is located in `/etc` and is edited by the administrator. For speed, however, the `sendmail` process requires a compiled version of the `aliases` file in DBM format. This is achieved by compiling with the `sendmail -bi` command. This creates two files:

`/etc/aliasesDB/DB.dir`
`/etc/aliasesDB/DB.pag`

If these files do not exist, the `sendmail` command cannot process mail, and will generate an error message. The layout of the alias file conforms to the following:

Alias: Name₁, Name₂, ... Name_X

where *Alias* is the string of characters you want the group to be known as (not including special characters, such as @ or !) and *Name₁* etc., is a series of one or more recipient names. An alias list as indicated should always have an owner specified so that any mail problems

with the alias can be notified to the registered owner. The next example serves to highlight a typical entry:

```
gurus: mike@daemon, dick@hp, garry@compu  
owner-gurus: gerry@hp
```

In addition to the aliases the administrator sets up, the following three aliases must be present under AIX:

MAILER-DAEMON	The ID of the user who is to receive messages addressed to the mailer daemon. Usually assigned to the root user:
postmaster	The ID of the user responsible for the operation of the local mail system
nobody	The ID that is to receive messages directed to programs such as news. This name is initially assigned to /dev/null.

A-11. SLIDE: Managing the Mail Queue

Managing the Mail Queue

Restart the **sendmail** daemon with a different mail check frequency:

1. Edit the **/etc/rc.tcpip** file.
2. Change the **qpi** variable, such as: **qpi=60m**.
3. Stop the **sendmail** daemon with either:

```
stopsrc -s sendmail
```

If daemon was not started
with the **startsrc** command.

```
kill 'cat /etc/sendmail.pid'
```

If daemon was started
manually.

4. Restart the daemon with either:

```
startsrc -s sendmail -a "-bd-q60m"
```

```
/usr/lib/sendmail -bd -q60m
```

a646175

Student Notes

The frequency at which the mail queue is examined is determined by the **-q** flag argument to **sendmail**. By default this is 30 minutes which is set in the **/etc/rc.tcpip** file. To specify a different queue processing interval this value must be changed and the **sendmail** process stopped and restarted as indicated on the slide. Each message in the mail queue contains a time value indicating the time of submission. The administrator can flush messages that have been in the queue for a given period by specifying a shorter message time-out for the queue directly at the command line prompt, hence the following example details how to flush the queue immediately, disregarding all messages over 1 day old:

```
/usr/sbin/sendmail -oT1d -q
```

A-12. SLIDE: Managing Mail Logging

Managing Mail Logging

The **sendmail** command logs mail system activity through the **syslogd** daemon. The **syslogd** daemon must be configured and running for logging to occur.

The **/etc/syslog.conf** file must contain the uncommented line:

```
mail.debug      /usr/spool/mqueue/log
```

To force the **syslogd** daemon to re-read the file, the following must be entered:

```
refresh -s syslogd
```

Cumulative statistics can be recorded by creating a blank
/etc/sendmail.st file.

a646176

Student Notes

The **sendmail** command automatically logs mail activity through the **syslogd** daemon process. This daemon however must be configured to enable this activity. By default the mail debug option is turned OFF. The administrator must enable the facility by uncommenting the corresponding line in **/etc/syslog.conf** and re-starting the **syslogd** daemon using the system resource controller. If the **/usr/spool/mqueue/log** file does not exist, you must create it.

Because, over a period of time, the log file can grow to a very large size, a shell script **/usr/lib/smdaemon.cleanu** is supplied that forces the **sendmail** command to process the queue, and maintains four progressively older copies of log files. This allows logging to start over with a new file and the script is usually invoked by using the **cron** facility.

The **sendmail** command tracks the volume of mail being handled by each of the mailer programs that interface with it (those mailers defined in the **/etc/sendmail.cf** configuration file). To accumulate these statistics, the **/etc/sendmail.st** file must be created manually. The statistics kept in the **/etc/sendmail.st** file are in a database format and cannot be read as ASCII data. The supplied command **/usr/sbin/mailstats** presents the

information in readable form. Statistics are cumulative and can be reset to zero at any time by entering the mailstats program with the -z flag

A-13. SLIDE: Managing the NLS Configuration File

Managing the NLS Configuration File

sendmail under AIX has a facility for handling national language support (NLS).

The **/etc/sendmail.nl** file contains lists of systems that correctly interpret mail messages containing NLS or ISO-8859/1 characters.

Systems that are not detailed within these lists have their messages converted to 7-bit ASCII characters.

Example file:

```
NLS:    ^@.*madrid\.,  
        ^@rome,  
        ^@.*italy\.europe$,  
8859:   .*vienna!$,  
        ^@bangkok\.thailand,  
        ^@tangiers,  
        ^@kinshasa
```

a646177

Student Notes

AIX provides a facility for handling national language support (NLS). The **/etc/sendmail.nl** file contains lists of systems that interpret mail messages containing NLS or ISO-8859/1 characters. If the system receiving a message is not in the lists, all characters in the body of the message are changed to standard 7-bit ASCII characters. If the system is in the list of NLS systems, all extended characters (eighth bit set) in the body of the message are converted to 7-bit escape sequences. These characters can then pass unaltered through mail systems that would otherwise strip the eighth bit from all characters. The receiving mail system can then convert these characters back to the correct characters. The **sendmail.nl** file contains regular expressions to include all addressing methods that can be utilized to access systems in each particular country or domain.

A-14. SLIDE: AIX: Serial Line Communications

AIX: Serial Line Communications

The method of achieving serial line communication within AIX is SLIP.

SLIP Provides asynchronous transmission across dedicated lines RS232 or RS432 serial lines. The SLIP connection can be either direct or via modems.

SLIP is supplied as part of the TCP/IP network support facilities in the base operating system (BOS) runtime package.

a646178

Student Notes

The SLIP package is provided within the network support utilities within the BOS for AIX. It allows the user to administer Internet protocol over direct or modem connected lines. Any serial interface can be configured for SLIP, with **smit** providing a shell script to configure all aspects.

The device used for the SLIP connection must have previously been defined and made available within the ODM.

The UUCP files must be edited to set up the connection details for both direct and modem connected lines. The **/etc/rc.net** file is responsible for configuring and enabling the SLIP connections.

A-15. SLIDE: Steps for Configuring an AIX SLIP Connection

Steps for Configuring an AIX SLIP Connection

1. Define a serial device within the system.
2. Set up the UUCP devices configuration files for the serial line.
3. Define the serial parameters within **smit**.
4. Use **smit** for each of these steps.

a646179

Student Notes

The steps involved in configuring SLIP under AIX are as follows:

1. Define a serial device for the system. The administrator must first set up a serial device that will be used as a SLIP interface within the system. The line should have the **getty** disabled (as the **slattach** process will run on the line) and should be configured so that it is available at system boot. The following command defines a serial port on the first 8 way adaptor card at port 7:

```
# mkdev -c tty -t 'tty' -s 'rs232' -p 'sa2' -w '7' -a ttyprog_action='off'
```

2. Setup the UUCP Devices configuration file for the connection.

The UUCP Devices configuration file needs to be edited to reflect the connection. A direct or modem connection requires a corresponding entry within **/usr/lib/uucp/Devices**. The following option details a simple direct connection on **tty2**:

Direct tty2 - 9600 direct

3. Define the serial parameters within smit.

The last step is to set-up the information regarding the local and remote Internet addresses to be used and The **smit** path to achieve this is :

```
smit tcpip
->Further Configuration
->->Network Interfaces
->->->Network Interface Selection
->->->->Add a Network Interface
->->->->->Add a Serial Line INTERNET Network Interface
```

The following screen is presented. At this point, the administrator must enter the values of the Internet addresses required.

Add a Serial Line INTERNET Network Interface

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

	[Entry Fields]
* INTERNET ADDRESS (dotted decimal)	[195.30.0.1]
* DESTINATION Address (dotted decimal)	[195.30.0.2]
Network MASK (hexadecimal or dotted decimal)	[]
* ACTIVATE the Interface after Creating it?	yes
+	
* TTY PORT for SLIP Network Interface	tty2
BAUD RATE	[9600]
+#	
DIAL STRING	[""ATDT34335 CONNECT""]
Note: 1) If specifying a dialstring, a baudrate is also required.	
2) Update /usr/lib/uucp/Devices before specifying a baud rate.	

The optional dial string is for a modem connection. This string is similar to a chat script and should be exact for the modem and phone number you are specifying. For a direct connection, SLIP automatically retries the connection so that manual intervention is not required. For a modem connection, a terminated telephone connection must be manually redialed.

As with SCO UNIX, the command **slattach** which handles the connection can be run manually by the administrator. Usually this is automated within the **/etc/rc.net**. Once configured, the SLIP interfaces can be interrogated with the **ifconfig** command as with a normal Ethernet interface:

```
# lsdev -C -c if
lo0 Available  Loopback Network Interface
en0 Available  Standard Ethernet Network Interface
et0 Defined    IEEE 802.3 Ethernet Network Interface
sl1 Available  Serial Line Network Interface

# ifconfig sl1
sl1: flags=31<UP,POINTOPOINT,NOTRAILERS>
      inet 194.20.0.1 --> 194.20.0.2 netmask 0xffffffff00
```

A-16. SLIDE: Network Tuning under AIX

Network Tuning Under AIX

The management of memory for network connections is handled by the kernel process **netm** running at a fixed priority.

Netm allocates MBUF buffers dynamically to handle network I/O.

Allocation and size of MBUF Buffers are alterable via the **no** command.

Tuning these values can greatly increase the performance of networked AIX systems.

a646180

Student Notes

The standard interface mechanism that AIX uses to communicate over networks using TCP/IP is the socket mechanism. As an application writes to a socket the data is copied from the users memory address space into the sockets buffer space which resides within the kernel. This socket send buffer is made up of smaller buffers called **mbufs**. These buffers are pinned into system memory and so cannot be paged out by the virtual memory manager.

There are two types of **mbufs** that the kernel administers, they are 256 bytes and 4096 bytes in size. (The larger buffers are known as clusters to avoid confusion.) Depending upon the size of data being transmitted, buffers are allocated from the free pool of buffers in memory.

Having the correct size of **mbufs** available within the memory pool is very beneficial to system performance. If they are configured incorrectly then both system and network performance will be adversely affected. Tuning the values associated with **mbufpools** can be done dynamically but it must be remembered that the space occupied by the buffers is taken from real memory (RAM). The result is that the real memory available for paging applications and data is decreased if the buffer pool is increased. This effect must always be considered.

A-17. SLIDE: Setting Buffer Sizes

Setting Buffer Sizes

The kernel process **netm** allocates buffers dynamically. **netm** runs at a high fixed priority so incorrect allocation of buffers can cause **netm** to thrash.

The **no** command allows several parameters to be set to prevent thrashing:

lowmbuf	controls the minimum number of free buffers for the mbuf pool
lowclust	controls the minimum number of free buffers for the cluster pool
mb_cl_hiwat	controls the maximum number of free buffers the cluster pool can contain
thewall	controls the maximum amount of RAM in Kbytes that can be used.

a646160

Student Notes

The initial size of **mbuf** pools is system dependent and allocated by the kernel. When demands are made on the system for more buffers, they are dynamically increased by a kernel process called **netm** which runs at a fixed process priority (37). Similarly when resource need is decreased, buffers are released back into the free memory pool. Since this process runs at a fixed priority, incorrect allocation can cause **netm** to thrash during peak traffic periods causing poor performance. To provide a mechanism against this, there are a number of parameters that we can set. These parameters are set via the **no** command which must be run as the root user.

To set the maximum number of free clusters for instance we can execute the following command:

```
# no -o mb_cl_hiwat=1500
```

When tuning for network performance, the system administrator must be familiar with the loads that will be imposed on the system. A small server servicing a limited number of ASCII terminals may not require tuning whereas a server supporting many X clients will almost certainly require tuning for optimum performance.

The **netstat** command is used to determine network performance. The **netstat -m** option gives detailed information about the use and availability of the memory buffers.

```
# netstat -m
14 mbufs in use:
    3 mbufs allocated to socket structures
    4 mbufs allocated to protocol control blocks
    5 mbufs allocated to routing table entries
    1 mbufs allocated to socket names and addresses
    1 mbufs allocated to interface addresses
1/8 mapped pages in use
35 Kbytes allocated to network (9% in use)
0 requests for memory denied
0 requests for memory delayed
0 calls to protocol drain routines
```

Notice the line: *1/8 mapped pages in use*. This indicates that there are 8 clusters total in memory, of which 1 is being used.

If the *requests for memory denied value* is not zero, the **mbuf** and/or cluster pools need to be expanded. This value corresponds to dropped packets on the network which has a direct effect on network performance.

The above report can be interpreted along with the **no** tuneable parameters to investigate network performance.

The current **no** settings can be displayed with the option:

```
# no -a
dog_ticks = 60
lowclust = 13
lowmbuf = 72
thewall = 2048
mb_cl_hiwat = 26
compat_43 = 1
```

We can see that the 35 Kbytes allocated to network is within the maximum value (**thewall** = 2048) and that the number of free clusters (8-1 = 7) is less than the maximum water mark (**mb_cl_hiwat** = 13).

Appendix B — Sun Solaris and SunOS Operating System

Objectives

Upon completion of this module you will be able to do the following:

- Document the differences between SunOS, Solaris, and HP-UX.

B-1. SLIDE: Solaris UNIX Network Configuration

Solaris UNIX Network Configuration

Basic Steps:

1. Design network, assign numbers, hostnames, and so forth.
2. Configure appropriate services during installation phase of Solaris.
3. Optionally configure the system to use NIS, NIS+, DNS or a mixture of functions.
4. Configure the `/etc/nsswitch.conf` file for correct file access operation.

a646181

Student Notes

During the installation of the Solaris Operating System from install media, the TCP/IP components of the operating system are installed. The administrator is asked many questions concerning the functionality of the TCP/IP configuration. The user must allocate Internet addresses, subnet masks, chose whether to operate as a NIS/NIS+ server or client and optionally configure for routing. After the installation phase the administrator should configure the system for NIS/NIS+ operation, DNS, PPP, default routing etc. Depending upon the choice of machine configuration, the `/etc/nsswitch.conf` file should be amended for the correct operation. This file contains rules for accessing each of the networking files located in `/etc` so that for instance, should we require host addresses to be first resolved via DNS, then by NIS+ for unresolved names the following entry will dictate the correct search sequence:

```
hosts: dns nisplus [NOTFOUND=return] files
```

There are several pre-configured dummy files located within the `/etc` directory that can be copied over the `/etc/nsswitch.conf` file for various machine configurations. These are as follows:

nsswitch.files	Normal file access within /etc
nsswitch.nis	For NIS operation.
Nsswitch.niplus	For Solaris NIS+ operation.

B-2. SLIDE: Solaris Network Configuration – Scripts

Solaris Network Configuration – Scripts

The main scripts involved in setting up the network at boot phase are as follows:

<code>/etc/rcS.d/S30rootusr.sh</code>	defines the hostname, all interfaces, broadcast addresses and default router if required
<code>/etc/rc2.d/S69inet</code>	configures all tasks that need to be launched before DNS or NIS (+) can be started
<code>/etc/S71rpc</code>	defines and starts NIS or NIS+
<code>/etc/rc2.d/S72inetsvc</code>	starts DNS and configures static routing starts <code>inetd</code>

a646182

Student Notes

The network configuration of the system involves a few scripts and programs. Once in place the system administrator can edit the files directly to effect various changes within the system. The four scripts that make up the Solaris TCP/IP configuration suite are as follows:

`/etc/rcS.d/S30rootusr.sh`

This script configures the loopback interface, and then each subsequent interface that is configured within the system. This is achieved in a compatible SunOS fashion with each interface having a file called `/etc/hostname.XeX` where X is the ascending interface number starting from 0. Each file contains the hostname that is cross referenced within `/etc/hosts` to yield the Internet address for that interface. The netmasks for each Internet family of addresses are read from `/etc/netmasks` which will be gained from NIS/NIS+ if configured in the later script `S72inetsvc`. The nodename for the machine is set from the contents of the file `/etc/nodname`.

/etc/rc2.d/S69inet

This script configures routers, Sets the domainname for the machine, and configures the machine automatically for ip forwarding if more than 1 interface is detected. This must be amended if the machine is to operate as a firewall.

Routing is centered around the presence of the **/etc/defaultrouter** file. This file contains the name of the default routers that should be configured within the dynamic routing table. If there are no default routers but this machine should act as a dynamic router, then a blank empty **/etc/defaultrouter** file should be created.

The script detects the presence of the file and depending upon the number of interfaces configures the machine as a dynamic or passive router using the routed or discovery protocol. The host will NOT be made a dynamic router if only 1 interface is configured within the machine. To alleviate this mechanism, the administrator should create a blank, empty file : **/etc/gateways** whose presence forces the machine to become a router.

/etc/rc2.d/S71rpc

This script starts the **rpcbind** daemon , then starts the NIS/NIS+ server or client processes if configured.

/etc/rc2.d/S72inetsvc

The last script configures all parameters such as netmasks and broadcast addresses for interfaces that need to be resolved using NIS/NIS+. DNS if configured is then started and finally **inetd** is started in standalone mode.

Final amendments that should be considered by the administrator concern the files hosts, networks, netmasks, protocols, services. Particularly if this is not a NIS, NIS+ client or host since the native files within the machine will be used to resolve data queries. The **/etc/nsswitch.conf** file should be amended for the desired mode of operation of the machine.

B-3. SLIDE: Configuring BIND under Solaris 2.X

Configuring BIND under Solaris 2.x

The operation of BIND is very similar to that for other versions of UNIX.

Configuration is achieved by creating the DNS file, the `/etc/resolv.conf` file, and altering `/etc/nsswitch.conf` for the DNS selection of host addresses.

The named server and NIS, NIS+ can processes all DNS requests according to the order specified within `/etc/nsswitch.conf`.

The `named` process is started by `/etc/rc2.d/S72inetsvc`.

a646183

Student Notes

The DNS components of Solaris are started from the script `/etc/rc2.d/S72inetsvc` which is linked to the file `/etc/init.d/inetsvc`. The process of configuring the server to run `named` is very similar to that already described in previous sections. The presence of the `/etc/named.boot` file instigates the launch of the `in.named` process. Again, the `-b` option is available to the command to specify the location of the boot file should a different location be required. The `named.boot` file must be created from scratch as must the files for each component of the database suite. In a multi-platform environment it is easier to use utilities available on other platforms to generate the files and then propagate them onto the target name servers. The layout of the files is identical to those indicated earlier and the student is advised to study earlier modules on this subject.

B-4. SLIDE: The /etc/nsswitch.conf file

The /etc/nsswitch.conf file

The `nsswitch.conf` file dictates the order in which services are interrogated for a given search parameter.

The token `dns` must be present as the first entry within the hosts section of this file in order to use DNS first.

Subsequent values may be added such as NIS+ or files. The order is left to right, for example:

```
hosts:          dns nisplus files [NOTFOUND=return]
```

Clients using NIS will have their addresses resolved via DNS via the `nsswitch.conf` file if the keyword `dns` appears first.

a646184

Student Notes

If we are using DNS to resolve hostnames, the Solaris kernel needs to be informed of the order in which to interrogate the different services that can resolve the name. For instance we may be running NIS+ and DNS on the server. We can resolve the host addresses from either of these services and from the original host files. The `/etc/nsswitch.conf` file presents an ordering of services to use when there is a requirement to resolve a piece of information. The token `dns` must be placed before other subsequent entries in order for DNS to be the first service that is called to resolve the address. Subsequent services can be called if `dns` fails as the line is parsed from left to right.

B-5. SLIDE: Debugging BIND under Solaris

Debugging BIND under Solaris

Various signals and flags are available to enable debugging of DNS under Solaris:

named -d N The **-d** flag with **named** causes debug output to be generated within the file **/var/tmp/named.run** to level N.

Signals are available to debug a running **named** process.

SIGHUP	causes named to reread named.boot and reload database
SIGINT	dumps database to /var/tmp/named_dump.db
SIGUSR1	turns on debugging, each subsequent signal increases level, the output is sent to /var/tmp/named.run
SIGUSR2	turns off debugging completely

The utility **nstest** is an interactive DNS test program also available to test the operation of DNS .

a646185

Student Notes

There are various flags and signals available to enable successful debugging of the **named** process. **named** under Solaris responds to the same signals as the other versions of UNIX detailed within this course (output being placed under **/var/tmp**). In addition to the signals detailed within the slide, Solaris has an additional command to enable DNS to be debugged called **nstest**. Specified without an Internet address argument it queries the name server on the local machine according to the requests the user enters. **nstest** is an interactive command and when invoked, it prints a prompt (">") and waits for user input. DNS queries are formed by typing a key letter followed by the appropriate argument. Each key letter results in a query to the name server. A sample of some of the keys is given:

- A Internet address query
- a Host name query
- c Canonical name query
- U UID query

The following example serves to illustrate the use of **nstest**. It is however limited, and should be used in-conjunction with **nslookup**.

```
# nstest
> a bmw.hp.com.
res_mkquery(0,  bmw.hp.com., 1, 1)
res_send()
HEADER:
    opcode = QUERY, id = 12, rcode = NOERROR
    header flags: rd
    qdcount = 1, ancount = 0, nscount = 0, arcount = 0

QUESTIONS:
    bmw.hp.com, type = A, class = IN

Querying server (# 1) address = 127.0.0.1
got answer:
HEADER:
    opcode = QUERY, id = 12, rcode = NXDOMAIN
    header flags: qr aa rd ra
    qdcount = 1, ancount = 0, nscount = 1, arcount = 0

QUESTIONS:
    bmw.hp.com, type = A, class = IN

NAME SERVERS:
    hp.COM
type = SOA, class = IN, ttl = 1 day, dlen = 37
    origin = ferrari.hp.COM
    mail addr = root.ferrari.hp.COM
    serial = 10001
    refresh = 3 hours
    retry = 1 hour
    expire = 5 days
    min = 1 day
>
```

B-6. SLIDE: **sendmail Configuration under Solaris**

sendmail Configuration under Solaris

The **sendmail** file supplied provides rules for TCP/IP & BNU mail forwarding using a domain address structure for a standard mail client system.

Alternative **sendmail** configuration files are supplied for various mail configurations.

Mail aliases for NIS+ are set using the **aliasadm** command.

Mail aliases are parsed according to values defined within **/etc/nsswitch.conf**.

a646186

Student Notes

The **sendmail** system within Solaris is very similar to that described fully in previous modules. The "Setting up Accounts, Printers and Mail" manual describes the **sendmail** configurations possible within Solaris and should be consulted when administering mail. There are a number of options available within Solaris depending upon the construction of the network layout. Solaris comes supplied with a standard **sendmail** configuration file that should be adequate for any system that is within a single domain or does not connect to any other network.

If however, the host is to act as a mail relay/gateway or mail host, then an alternative **sendmail** configuration file : **/etc/mail/main.cf** should be copied over **sendmail.cf**. Several changes will then need to be administered within this file depending upon the connectivity methods that will be employed to connect to other system networks.

Mail aliases can be set-up as within other systems, but an additional command, **aliasadm**, must be employed if the mail host utilizes NIS+ as its central naming service.

B-7. SLIDE: Components of Solaris **sendmail**

Components of Solaris **sendmail**

<code>/usr/lib/sendmail</code>	the sendmail program
<code>/usr/lib/sendmail -bd -q1h</code>	run directly from <code>/etc/rc2.d/S88sendmail</code> to run sendmail as a daemon process
<code>/usr/lib/sendmail.mx</code>	Alternative sendmail daemon that should be copied to sendmail if DNS hostnames are to be used
<code>/etc/mail/sendmail.cf</code>	standard configuration file
<code>/etc/mail/main.cf</code>	alternate configuration file for a relay host
<code>/etc/nsswitch.conf</code>	switch file for alias location selection

a646187

Student Notes

sendmail is configured and primed at boot by the script `/etc/rc2.d/S88sendmail` which loads **sendmail** as a server process (`sendmail -bd -q1h`) This process is responsible for sending and receiving SMTP requests over the network. If DNS is required to supply the hostname translation, the alternative binary program `/usr/lib/sendmail.mx` must be copied over `/usr/lib/sendmail`. When this program is in place the **sendmail** rule for converting fully qualified names using DNS is activated. As an additional step, the administrator must supply an entry for the mailhost in both the DNS files and the normal, NIS/NIS+ mail alias files.

If the server is to act as mail host from which clients will NFS mount mail directories, or as a mail relay station which forwards and receives packets from other domains and networks, then an alternate `sendmail.cf` file is supplied called `/etc/mail/main.cf` which should be copied over `/etc/mail/sendmail`. If this file is copied the following entries should be modified depending upon your configuration to the outside world:

`DMsmartuucp`

Change this if the relay mailer is not using `uucp` to either `ddn` for Defence Data Network, `ether` for Ethernet.

DRddn-gateway	Replace with the name of the relay host. (DR defines Relay host)
CRddn-gateway	Replace with the name of the relay host. (CR defines Class of relay host)

Aliases are defined either in local files or via NIS, NIS+ or DNS. The `/etc/nsswitch.conf` file dictates the order in which ASCII files in `/etc`, DNS, NIS and NIS+ are accessed to resolve a given data input. The administrator must edit this file to provide suitable access for the mail alias mechanism. The line is read from left to right, that is:

```
aliases: files nisplus dns
```

If aliases are added locally into the `/etc/mail/aliases` file, `sendmail` will automatically call `newaliases` to compile the file into its constituent DBS components. The administrator can simply edit these files directly. If entries are to be added into the NIS+ aliases file then the `aliasadm` command is utilized.

The following options are present:

aliasadm -l	Lists all aliases from the NIS+ table
aliasadm -m name	Lists name from NIS+ table only.
aliasadm -e	Edits existing table to add/delete aliases
alias -a	Add new aliases

The following entry adds the alias gerry to the system:

```
aliasadm -a gerry gerry@ferrari.hp.com "Gerry Nolan"
```

Of course, the `nisaddent` command can be used to simply re-populate the NIS+ table with the original `/etc/mail/aliases` file. For example,

```
/usr/lib/nis/nisaddent -r -f /etc/mail/aliases aliases
```

B-8. SLIDE: sendmail Configuration under SunOS 4.1

sendmail Configuration under SunOS 4.1

Main **sendmail** files under SunOS

/usr/uucb/mail	the mail program
/usr/bin/mail	sendmail/mail delivery interface
/usr/lib/sendmail.main.cf	configuration file for relay/host machine configuration
/usr/lib/sendmail.subsidiary.cf	configuration file for normal sendmail clients
/etc/sendmail.cf	main configuration file populated with one of above
/usr/lib/sendmail.mx	alternative sendmail binary program for DNS servers

a646188

Student Notes

The **sendmail** suite of programs is supplied under SunOS to provide mail connectivity for TCP/IP and UUCP connectivity. Alternative **sendmail** configuration files are supplied for various mail configurations.

The **sendmail** system within SunOS is very similar to that of Solaris 2.X. There are a number of options available within SunOS depending upon the construction of the network layout. SunOS comes supplied with a standard **sendmail** configuration file that should be adequate for any system that is within a single domain or does not connect to any other network.

If, however, the host is to act as a mail relay/gateway or mail host, then an alternative **sendmail** configuration file : /usr/lib/sendmail.main.cf should be copied over **sendmail.cf**. This file contains rule sets for UUCP connectivity, mail forwarding and so on. The /usr/lib/sendmail.subsidiary.cf file is the default **sendmail.cf** file for normal **sendmail** clients. Either of these files must be copied to the /etc/sendmail.cf file for **sendmail** to function correctly.

B-9. SLIDE: Components of SunOS **sendmail**

Components of SunOS **sendmail**

<code>/usr/lib/sendmail</code>	the sendmail program
<code>/usr/lib/sendmail -bd -q1h</code>	run directly from <code>/etc/rc.local</code> to run sendmail as a daemon process
<code>/usr/lib/sendmail.mx</code>	alternative sendmail daemon that should be copied to sendmail if DNS hostnames are to be used
<code>/etc/sendmail.cf</code>	standard configuration file

a646189

Student Notes

sendmail is configured and primed at boot by the script `/etc/rc.local` which clears the `/var/spool/mqueue` directory down for any locks and then loads **sendmail** as a server process (`sendmail -bd -q1h`). This process is responsible for sending and receiving SMTP requests over the network. If DNS is required to supply the hostname translation, the alternative binary program `/usr/lib/sendmail.mx` must be copied over `/usr/lib/sendmail`. When this program is in place the **sendmail** rule for converting fully qualified names using DNS is activated. As an additional step, the administrator must supply an entry for the mailhost in both the DNS files and the normal, NIS and mail alias files. Each client that will receive and send mail to the master mail server must also have an entry for the local mailhost which can be resolved using the `gethostbyname(3)` system call.

The operation, de-bugging and troubleshooting of SunOS **sendmail** is identical to that detailed in Module 3 for HP-UX. The student should refer to this section of the notes.

B-10. SLIDE: Solaris 2.3 Serial Line Communications

Solaris 2.3: Serial Line Communications

Solaris 2.3 supports serial line communication via use of Point to Point Protocol (PPP). Several components and steps are used to configure PPP under Solaris:

The link manager (aspppd)	automates the connection process
The login service (aspppls)	provides the login service (similar to UUCP's uucico)
The main configuration file	located in /etc/asppp.cf
The ppp log file	log of all ppp requests located at /var/adm/log/asppp.log

The PPP software must be loaded on your system!

a646190

Student Notes

Solaris handles Serial Line Internet communication via the Point to Point Protocol. In order for the user to configure this package it must first have been loaded at installation. The software comprises the following three modules:

- SUNWpppk Contains the kernel modules
- SUNWapppr Contains the configuration files
- SUNWapppu Contains the link manager and the login service.

The **pkginfo** command should be used to check the existence of these modules, and if not present, they should be loaded using **pkgadd**. The version of PPP present within Solaris allows various configurations of PPP but the operation remains essentially the same as that described in both the SCO UNIX and HP-UX modules.

B-11. SLIDE: Setting Up Solaris PPP

Setting Up Solaris PPP

1. Configure **/etc/hosts** for the PPP connected hosts.
2. Edit the UUCP database files to reflect the new PPP link.
3. Create an account for the dial-in machine.
4. Edit the PPP configuration file.
5. Start the link manager: **aspppd**.

a630277

Student Notes

There are a number of steps involved with setting up PPP within Solaris. Firstly the IP entries for the machines to be contacted with PPP links must be present within the hosts file (even if NIS is being used). This is because PPP is started before NIS. After entering the Hosts entries, the uucp configuration files need to be set-up to reflect the serial communication line(s) that the links will utilize. The files are:

/etc/uucp/Devices Contains an entry for each physical port that will be connected to a PPP client. For example,

Direct term/b - Any direct

/etc/uucp/Systems Contains an entry for each machine that will be contacted using either PPP or UUCP. The third field of each entry must point to the corresponding entry within **/etc/uucp/devices**.

hp-ppp Any Direct 19200 "" in:--in: pppuser word: mypasswd

**/etc/uucp/
Dialers**

For connections that are not direct, the dialers file entry details the modem chat script that should be sent to initialize the particular modem you are utilizing. This file contains many dialer scripts which can be edited to suit.

Creating a PPP Account

For systems that dial into your host, a PPP account must be created. This account is referenced within the Systems file (pppuser in previous example.) When the target machine logs into the system, provided that the password is correct, a special login shell: /usr/sbin/aspppls is started. The useradd command can be used to add the entry to the password file.

```
# useradd -d / -s /usr/sbin/aspppls -c "General PPP user" pppuser  
# passwd pppuser
```

The password that is set must reflect the value set in the systems file for this machine. Having set this account up, the PPP configuration file needs to be configured for all PPP links. This configuration file brings all of the interface to be used up with the ifconfig command and sets various parameters such as timeout delays, host Internet addresses and so on.

B-12. SLIDE: The /etc/asppp.cf Configuration File

The /etc/asppp.cf Configuration File

The **/etc/asppp.conf** file contains configuration information on all PPP links. It is read at boot time by the **aspppd** daemon process to configure all links.

The file has various required and optional elements, and has the following form:

```
ifconfig ipdptp0 plumb 194.20.0.230 194.20.0.6 up

path
  inactivity_timeout 120 # Timeout value = 2 minutes
  interface ipdptp0      # The interface to use for the host
  peer_system_name hp-ppp # hp-ppp
```

a646191

Student Notes

The configuration file is read at system start by the **aspppd** process to configure all of the valid PPP links within the system. It has various entries, some of which are required, others being optional depending upon the needs of the particular interface. Each interface being used is primed with the local and remote Internet addresses (expressed as dotted 4 byte addresses) or hostnames which are resolved from **/etc/hosts**. The following stanza entries are required within the configuration file.

ifconfig parameters	Instructs the link manager daemon to configure the specified interfaces with the parameters supplied.
path	Specifies the beginning of a series of entries which are to be grouped together for the interface specified. This grouping is delimited by the default or path keywords.
interface parameter	Specifies the interface to be used for this particular entry. The interfaces ipdptp? are static PPP links, where ? starts at 0 and is incremented for each port used. ipdptp* is used to

enable multiple hosts to dial into the system via a common port (modem).

peer_system_name parameter Specifies the name of the remote machine which must match the entry within the uucp systems file.

There are a number of optional parameters which the administrator can add into the PATH stanza to achieve differing operation of the line configured, among these entries are:

debug_level N Turns on debugging at level *N*. (Range 0-9)

inactivety_timeout N The length of time a link is kept up while no activity is seen. If set to 0 the link is never dropped. The default is 120 seconds.

ipcp_compression vj or off Specifies whether Van Jacobson IP compression should be implemented: the default is off.

B-13. SLIDE: Starting PPP and Error Logging

Starting PPP and Error Logging

The PPP daemon can be started manually with the command:

```
/etc/init.d/asppp start
```

At any time, the configuration file can be reread by sending SIGHUP to the **asppp** command.

Logging information is presented within the file:

```
/var/adm/log/asppp.log
```

a690279

Student Notes

Once the configuration file is in place, PPP can be started manually by running the script `/etc/init.d/asppp start`. Optional debugging can be turned on by amending the script and adding the `-d N` flag where *N* is in the range 0 to 9. At any time, if the configuration file is changed, the `aspppd` daemon process can be forced to re-read the file by sending the SIGHUP signal to the process. Logging information for troubleshooting is held in the file `/var/adm/log/asppp.log`, the level of data given depending upon the debug level chosen. If the configuration files are correct, the `ifconfig` command should report the presence of the PPP link i.e.:

```
lo0: flags=849<UP,LOOPBACK,RUNNING,MULTICAST> mtu 8232
      inet 127.0.0.1 netmask ff000000
le0: flags=863<UP,BROADCAST,NOTRAILERS,RUNNING,MULTICAST> mtu 1500
      inet 192.20.0.230 netmask ffffff00 broadcast 192.20.0.255
      ether 8:0:20:b:f4:60
ipdptp0: flags=8d1<UP,POINTOPOINT,RUNNING,NOARP,MULTICAST> mtu 8232
      inet 194.20.0.230 --> 194.20.0.6 netmask ffffff00
      ether 0:0:0:0:0:0
```

The programs, **ping** and **snoop** can then be used to test the integrity of the link. Common sources of errors are incorrect UUCP chat scripts and incorrectly configured **asppp.cf** files.

RIP is automatically available for PPP links. To disable this feature, the **/etc/gateways** command must be created with the following entry specified:

norip where *n* represents the number of the interface to be configured.
ipdptn

B-14. SLIDE: Network Tuning Under Solaris UNIX

Network Tuning Under Solaris UNIX

The Solaris kernel is self tuning and modules are loaded by the kernel as needed. Only a few tuneable parameters can be altered to influence TCP/IP.

Current values can be indicated by use of the **sysdef** command.

The administrator should be aware of two utilities:

- ndd** used to influence the behaviour of IP within the system
- snoop** used to monitor TCP/IP packets

The basic commands such as **nfsping**, **netstat**, and so forth, also are available to the administrator for network tuning.

a630280

Student Notes

The Solaris kernel automatically loads modules as they are required by the system. There are a number of tuneable parameters but they will only need tuning under special circumstances. This is achieved by setting their values in the file **/etc/system**. Most parameters are scaled according to a direct relationship with the maxusers variable and the student should be aware of the implication of changing certain variables.

Direct tuneable parameters are set by entering the following:

```
set Tuneable_value=new_value
```

whereas module variables are set as:

```
set module_name:variable_name=value
```

After setting the parameters within the **/etc/system** file, the system should be re-booted to apply the values assigned.

A few examples are listed. The administrator should consult Appendix A of the *Administering Security, Performance, and Accounting* guide for a more exhaustive list.

tune:npty	Total number of SunOS psuedo-ttys configured. (Default 48)
tune:pt_cnt	Total number of Solaris 2.X pseudo-ttys configured (Default 48)
rfs:rf_maxkmem	Limits amount of total memory allocated to RFS. (Default 0 - No limit)

Run-time values of variables can be indicated by use of the **sysdef -i** command.

B-15. SLIDE: ndd Command

ndd Command

The **ndd** command is available to set/get kernel driver parameters for the TCP/IP family.

To view the parameters available within each protocol family:

```
ndd /dev/ip \?
```

To get a single value from a protocol:

```
ndd -get /dev/ip ip_forwarding
```

To set a value:

```
ndd /dev/ip ip_send_redirects=1
```

Direct manipulation of these parameters is usually achieved in the start-up files.

a646192

Student Notes

The **ndd** command allows direct manipulation of the TCP/IP suite of kernel drivers and allows such mechanisms as **ip** forwarding or debugging to be selected. The command has two forms, the first utilizes the **\?** Flag to display the parameters available for each protocol family, the second allows the parameter to be set to a supplied value. The following example displays the **ip** protocol and sets **ip_debug** to be turned on.

```
# ndd /dev/ip \?
?
(ip_ill_status          (read only)
 ip_ipif_status          (read only)
 ip_ire_status            (read only)
 ip_rput_pullups         (read and write)
 ip_forwarding            (read and write)
 ip_respond_to_address_mask_broadcast (read and write)
 ip_respond_to_echo_broadcast (read and write)
 ip_respond_to_timestamp (read and write)
 ip_respond_to_timestamp_broadcast (read and write)
 ip_send_redirects       (read and write)
 ip_forward_directed_broadcasts (read and write)
 ip_debug                 (read and write)
 ip_mrtdebug              (read and write)
 ip_ire_cleanup_interval (read and write)
 ip_ire_flush_interval   (read and write)
 ip_ire_redirect_interval (read and write)
 ip_def_ttl               (read and write)
 ip_forward_src_routed   (read and write)
 ip_wroff_extra           (read and write)
 ip_ire_pathmtu_interval (read and write)
 ip_icmp_return_data_bytes (read and write)
 ip_send_source_quench   (read and write)
 ip_path_mtu_discovery   (read and write)
 ip_ignore_delete_time   (read and write)
 ip_ignore_redirect       (read and write)
 ip_output_queue           (read and write)
 ip_broadcast_ttl          (read and write)
 ip_icmp_err_interval     (read and write)
 ip_reass_queue_bytes      (read and write)

# ndd -set /dev/ip ip_debug 1
```

The **ndd** command is utilized within the **/etc/rc2.d/S69inet** shell script to configure the system for routing if more than one interface is located within the system. To make your system a firewall gateway, this must be turned off. This can be achieved by finding the following lines within the **S69inet** script:

```
numifs='ifconfig -au | grep inet | wc -l'
numptptifs='ifconfig -au | grep inet | egrep -e '-->' | wc -l'
```

and changing them to:

```
numifs=1
numptptifs=0
```

Windows size of TCP/IP packets, debugging and many other parameters can be switched using the **ndd** command. The administrator should consult the manual references for more information.

B-16. SLIDE: snoop Command

snoop Command

Packet tracing can be achieved within Solaris via use of the **snoop** facility.

snoop can operate in many modes, which can be tailored to suit the administrator's requirement. Verbose mode produces large output, whereas the single line summary-mode-only outputs information about the highest level protocol.

snoop can drop packets if network traffic is consistently high.

snoop can be configured to listen in for discrete traffic only.

a646193

Student Notes

The **snoop** command enables packets to be captured from the network. Discrete traffic or all traffic can be traced, but at times of high utilization of the network, **snoop** will drop packets. Output can be sent to the display or captured to file for later examination. The following example details **snoop** being used in verbose mode to trace traffic between two hosts (ford & ferrari).

```
# snoop -V ford and ferrari
ford -> ferrari      ETHER Type=0800 (IP), size = 60 bytes
    ford -> ferrari      IP D=192.20.0.220 S=192.20.0.100 LEN=40, ID=1540
    ford -> ferrari      TCP D=23 S=1502      Ack=3297206319 Seq=4677992 Len4
    ford -> ferrari      TELNET C port=1502

    ford -> ferrari      ETHER Type=0800 (IP), size = 60 bytes
    ford -> ferrari      IP D=192.20.0.220 S=192.20.0.100 LEN=40, ID=1541
    ford -> ferrari      TCP D=23 S=1502      Ack=3297206319 Seq=4677992 Len8
    ford -> ferrari      TELNET C port=1502

ferrari -> ford      ETHER Type=0800 (IP), size = 1514 bytes
ferrari -> ford      IP D=192.20.0.100 S=192.20.0.220 LEN=1500, ID=4371
```

```

ferrari -> ford          TCP D=1502 S=23      Ack=4677992 Seq=3297206818 Len0
ferrari -> ford          TELNET R port=1502      ford -> ferrari

ford -> ferrari          ETHER Type=0800 (IP), size = 60 bytes
ford -> ferrari          IP D=192.20.0.220 S=192.20.0.100 LEN=40, ID=1542
ford -> ferrari          TCP D=23 S=1502      Ack=3297206319 Seq=4677992 Len8
ford -> ferrari          TELNET C port=1502

```

Filter packets between **ford** and **ferrari** and save to a file for later interrogation:

```
# snoop -o /tmp/file ford and ferrari
```

View packets sequentially from packet 1 to 5 from the input file:

```

# snoop -i /tmp/file -p1,10
1 0.00000      ferrari -> ford      TELNET R port=1502 /dev/le
2 0.03228      ford -> ferrari    TELNET C port=1502
3 0.00035      ferrari -> ford      TELNET R port=1502
4 0.00416      ford -> ferrari    TELNET C port=1502
5 0.21092      ford -> ferrari    ICMP Echo reply

```

Interrogate packet 5 in more detail:

```

# snoop -i /tmp/file -v -p5
ETHER: ----- Ether Header -----
ETHER:
ETHER: Packet 5 arrived at 10:14:35.10
ETHER: Packet size = 98 bytes
ETHER: Destination = 8:0:20:b:f4:60, Sun
ETHER: Source      = 0:80:c7:2e:3e:a3, Xircom Inc.
ETHER: Ethertype = 0800 (IP)
ETHER:
IP: ----- IP Header -----
IP:
IP: Version = 4
IP: Header length = 20 bytes
IP: Type of service = 0x00
IP:     xxx. .... = 0 (precedence)
IP:     ...0 .... = normal delay
IP:     .... 0.... = normal throughput
IP:     .... .0... = normal reliability
IP: Total length = 84 bytes
IP: Identification = 3124
IP: Flags = 0x0
IP:     .0.. .... = may fragment
IP:     ..0. .... = last fragment
IP: Fragment offset = 0 bytes
IP: Time to live = 255 seconds/hops
IP: Protocol = 1 (ICMP)
IP: Header checksum = 2e0c
IP: Source address = 192.20.0.100, ford
IP: Destination address = 192.20.0.220, ferrari
IP: No options
IP:
ICMP: ----- ICMP Header -----
ICMP:
ICMP: Type = 0 (Echo reply)
ICMP: Code = 0
ICMP: Checksum = 18f9
ICMP:

#

```

B-17. SLIDE: NIS+ Administration Under Solaris 2.3

NIS+ Administration Under Solaris 2.3

Solaris 2.3 onwards provides a replacement NIS system: NIS+.

NIS+ can coexist with NIS clients, but offers the following extra functionality:

- domain management similar to DNS
- incremental, instant updates to replica servers
- coexistence of NIS+, NIS & DNS
- extra security mechanisms

a630283

Student Notes

With Solaris 2.3 onwards, the traditional NIS mechanism has been replaced with NIS+. NIS+ takes the normal NIS mechanism a step further by introducing the concept of NIS domain spaces. Each domain space is administered autonomously but doesn't make data exchange more complicated. Clients can still see information within other domains, and separate NIS domains can be administered from inside other domains. NIS+ as in NIS has master-slave relationships (slave servers being termed replica servers) but the method of transferring new information is different. Data changes are passed immediately as incremental changes so that data propagation is much quicker.

NIS+ implements a much more rigid security process. It authenticates and authorizes clients who require service; this allows sensitive information such as the password field in the /etc/passwd file to be excluded from certain client queries. NIS+ can be configured in NIS computability mode so that it will service requests from both NIS and NIS+ servers.

The updating and propagating of data from a master server to a replica server is implemented via a transaction log. This propagation of data is achieved quickly and automatically.

B-18. SLIDE: NIS+ Tables

NIS+ Tables

NIS+ stores information in tables.

Each table has a column entry structure.

By default there are 16 tables; host, password, services, and so forth.

Security mechanisms are available to access particular rows, columns, and tables.

a646194

Student Notes

Each NIS+ domain contains a number of pre-defined tables containing the information held in the `/etc` files. Each of these tables , like conventional NIS, have columns and rows so that ordinary searches for information index into a column to determine the row value. Each table, column and row element can be configured for client access giving a more flexible security mechanism. The tables are generally primed with values at NIS+ configuration time but can be updated and re-created.

B-19. SLIDE: Setting Up the Root Domain NIS+ Server

Setting Up the Root Domain NIS+ Server

1. Set the domain name.
2. Set the root masters switch configuration file.
3. Name the root domain administration group.
4. Initialize the root master server.
5. Start the NIS+ daemon (NIS+ or NIS compatibility mode).
6. Create domain tables.
7. Create DES encryption key for root master server.
8. Create and add root to administration group.
9. Update the root domain's public keys.
10. Start the NIS+ cache manager.

a646162

Student Notes

The set-up procedure for NIS+ is complicated and requires many steps. After setting the domainname (`/etc/defaultdomain` contains the domain name for subsequent boots) the switch configuration file must be edited to reflect NIS+ operation. The `/etc/nsswitch.conf` file dictates the order in which NIS, NIS+, DNS and raw files are accessed to gain the information. A default file is supplied as standard (`/etc/nsswitch.nisplus`) which should be copied to the `/etc/nsswitch` file, a sample is detailed:

```
services: nisplus [NOTFOUND=return] files
networks: nisplus [NOTFOUND=return] files
protocols: nisplus [NOTFOUND=return] files
bootparams: nisplus [NOTFOUND=return] files
```

After setting the name of the administrative group for the NIS+ server (later, administrators who have permission to change NIS files , permissions and operations must be added to this group) by setting the variable `NIS_GROUP`, the master server must be initialized with the `nisinit -r` command. This creates the `/var/nis/hostname` directory where NIS+ information is stored.

NIS Compatibility Mode

If the server is to process NIS requests, the server must now be started in NIS compatibility mode. For example,

```
rpc.nisd -r -Y -B -S 0
```

These options must also be turned on for subsequent reboots within the file `/etc/init.d/rpc`. Within Solaris 2.4 the `-r` flag is now obsolete. If the flag is specified however a warning message is generated but the nis daemon is still started.

The next step is to create the nis domain objects that will populate the `/var/nis/hostname` subdirectory. This is achieved by running the `/usr/lib/nis/nissetup -y` (-Y : NIS Compatible) command. This process adds the file objects only. After completion of the process of setting up the server, these files will be populated by the administrator using the `nisaddent` command. (See later Slide). After adding the encrypted key for secure RPC calls using the `nisaddircred` command, creating an adding membership of all administrators for the NIS+ domain, the NIS+ cache manager is started. NIS+ should now be available within the system.

For a full and complete description of the outlined process, the administering NIS+ and DNS guide should be consulted.

(see Answerbook NIS+ Network Security (Solaris 2.4 Introduction) if using OpenWindows.)

B-20. SLIDE: Populating NIS+ Files

Populating NIS+ Files

1. The NIS+ files are populated using the **nisaddent** command.
2. The contents of any file can be viewed using the **niscat** command.
3. Searches for NIS table data can be implemented via the **nisgrep** and **nismatch** commands.
4. Mechanisms exist to enable migration between NIS and NIS+ file structures.

a646195

Student Notes

The **nisaddent** command is the primary mechanism available to populate the NIS object space with the contents of the files usually located within the **/etc** directory. It allows data to be imported to/from NIS and ASCII maps. Propagation of maps is completed via a transaction log mechanism so there is an option within nisaddent that appends data into an existing file. This makes the log file smaller and hence propagation of data to slave servers quicker. The following example loads the **/etc/hosts** file into the NIS+ object space. The **-r** flag ensures that the original file is replaced and the **-v** flag as ever gives verbose information.

```
/usr/lib/nis/nisaddent -rv -f /etc/hosts hosts
adding /etc/hosts to table hosts.org_dir.quanta.com.
clearing table hosts.org_dir.quanta.com.
adding/updating localhost
adding/updating ferrari
adding/updating ferrari (loghost)
adding/updating rolls
adding/updating rolls (quant01)
....
```

```
.....
adding/updating lotus
adding/updating lotus (quant02)
70 entries added/updated
```

Files can be additionally loaded into other NIS+ domains by providing the domain name. The user must be set-up as an administrator of that domain.

Viewing Files

Files can be viewed by using the **niscat** command. If the NIS system has been started with NIS backwards compatibility in-place then existing NIS clients can still use the familiar **ypmatch** and **ypcat** commands. **niscat** is also used to display information regarding the object properties of the tables. This is useful since it identifies the access rights, domain name and other important information, for example:

```
nisscat -o org_dir
Object Name      : org_dir
Owner           : ferrari.hp.com.
Group           : admin.hp.com.
Domain          : hp.com.
Access Rights   : r---rmcdrmcdr---
Time to Live    : 12:0:0
Object Type     : DIRECTORY
Name : 'org_dir.hp.com.'
Type : NIS
Master Server   :
      Name       : ferrari.hp.com.
      Public Key : Diffie-Hellman (196 bits)
      Universal addresses (6)
      [1] - udp, inet, 192.20.0.220.0.111
      [2] - tcp,  inet, 192.20.0.220.0.111
      [3] - -,  inet, 192.20.0.220.0.111
      [4] - -,  loopback, ferrari.rpc
      [5] - -,  loopback, ferrari.rpc
      [6] - -,  loopback, ferrari.rpc
Time to live : 12:0:0
```

The NISGREP and NISMATCH Commands

For selective information retrieval, the **nismatch** and **nisgrep** commands are available. The **nisgrep** command allows complex regular expression searches to be implemented. Particular columns within files can be specified to pinpoint exact data matches. The following two examples indicate the searching facilities. The first example details a search for all references to the host **ferrari**, while the second details a search on node name only for all hosts beginning with the **f** character.

```
# nisgrep -h ferrari hosts.org_dir
# cname name      addr          comment
ferrari ferrari 192.20.0.220
ferrari loghost 192.20.0.220
ferrari hp220   192.20.0.220
ferrari solaris 192.20.0.220
```

```
# nisgrep -h name="f.*" hosts.org_dir
# cname name      addr          comment
ferrari ferrari 192.20.0.220
ford    ford     192.20.0.100
#
```

B-21. SLIDE: Administering Access Rights within NIS+

Administering Access Rights within NIS+

One of the reasons for the implementation of NIS+ is data security. NIS+ uses a process very similar to UNIX file permissions to implement security within object files. Four access rights can be granted to a file:

read, create , modify and destroy (delete).

Manipulation of these qualities is performed via the **nischown** command.

These access rights are applied to four access groups:

owner, group, world, and nobody

Manipulation of these qualities is performed via the **nistbladm** command.

a646196

Student Notes

The main strategy behind NIS+ is security. NIS offers a security mechanism very similar to UNIX file permissions for each column within each file within the NIS+ object area. By manipulation of these fields, requests for specific data can be granted and denied for 4 classes of users. There are four access privileges: read, create , modify and delete, and there are 4 classes of user : Owner, group, World and Nobody. Access to the NIS+ server is also controlled by using secure RPC protocol via the DES and keyserv processes.

If NIS backwards compatibility is turned on , then security is somewhat compromised since we must grant read access to each column within each file. For purely Solaris sites, then NIS offers an excellent secure NIS environment. The access permissions on columns is administered via the **nistbladm** command while global permissions to access each table is controlled via the **nischmod** command. Existing permissions can be examined with the **niscat** command .

```
# niscat -o hosts.org_dir
Object Name      : hosts
Owner            : ferrari.quanta.com.
Group            : admin.quanta.com.
Domain           : org_dir.quanta.com.
```

```
Access Rights : r---rmcdrmcdr---
Time to Live   : 12:0:0
Object Type    : TABLE
Table Type     : hosts_tbl
Number of Columns : 4
Character Separator :
Search Path    :
Columns        :

[0]      Name       : cname
          Attributes : (SEARCHABLE, TEXTUAL DATA, CASE INSENSITIVE)
          Access Rights : -----
[1]      Name       : name
          Attributes : (SEARCHABLE, TEXTUAL DATA, CASE INSENSITIVE)
          Access Rights : -----
[2]      Name       : addr
          Attributes : (SEARCHABLE, TEXTUAL DATA, CASE INSENSITIVE)
          Access Rights : -----
[3]      Name       : comment
          Attributes : (TEXTUAL DATA)
          Access Rights : -----
```

Changing the Access Permissions to an Object is performed by utilizing the `chown` command. From the previous output, the table permissions for the host object are as follows:

```
Access Rights : r---rmcdrmcdr---
```

The permissions are represented in the order Nobody, Owner, Group, World. The following command allows the world entry to be set to read, write, create and delete.

```
# nischmod w+rccmd hosts.org_dir
```

Individual table columns can also be set for particular access within the four groups. This is achieved via the `nistbladm` command. From the previous `niscat` command we get the following permissions for the four table entries found within the hosts file.

```
Columns        :
[0]      Name       : cname
          Attributes : (SEARCHABLE, TEXTUAL DATA, CASE INSENSITIVE)
          Access Rights : -----
[1]      Name       : name
          Attributes : (SEARCHABLE, TEXTUAL DATA, CASE INSENSITIVE)
          Access Rights : -----
[2]      Name       : addr
          Attributes : (SEARCHABLE, TEXTUAL DATA, CASE INSENSITIVE)
          Access Rights : -----
[3]      Name       : comment
          Attributes : (TEXTUAL DATA)
          Access Rights : -----
```

We can manipulate these permissions simply by specifying the name(s) of the column entries to change, and their new settings. Hence to change the permissions for group nobody to Read, write, create and Destroy, the following command would be required:

```
# nistbladm -u cname=n+rccmd hosts.org_dir.hp.com.
#niscat -o hosts.org_dir
.....
.....
Columns        :
[0]      Name       : cname
          Attributes : (SEARCHABLE, TEXTUAL DATA, CASE INSENSITIVE)
```

Access Rights : rmcd-----

Ownership and group membership of NIS+ objects can be changed with the **nischown** and **nischgrp** commands respectively. A record of all transactions conducted is automatically stored by the NIS+ service on the server. This information can be quite large and as such is stored in a binary file. To access this information the **nislog** command can be utilized. Without any arguments the whole of the log file is produced. The additional **-t** flag allows only the last n transactions of the log file.

```
# nislog -vt 1
NIS Log printing facility.
NIS Log dump :
    Log state : STABLE.
Number of updates      : 551
Current XID           : 470
Size of Log in bytes  : 209548
*** UPDATES ***
@@@@@@@ Transaction @@@@@@@@#00550, XID : 470
Time      : Thu Jun 29 14:37:29 1995

Directory   : org_dir.hp.com.
Entry type  : ADD Entry
Entry timestamp : Thu Jun 29 14:37:29 1995
Principal   : ferrari.hp.com.
Object name  : [ cname = bmw.hp.com., auth_type = DES,
    auth_name = unix.bmw@hp.com ],cred.org_dir.hp.com.
..... Object .....
Object Name   : cred
Owner         : bmw.hp.com.
Group        : admin.hp.com.
Domain       : org_dir.hp.com.
Access Rights : ----r---rmcd----
Time to Live  : 12:0:0
Object Type   : ENTRY
    Entry data of type cred_tbl
    [1] - [16 bytes] 'bmw.hp.com.'
    [2] - [4 bytes] 'DES'
    [3] - [20 bytes] 'unix.bmw@hp.com'
    [4] - [50 bytes] '07f135bb76db42aeb1e14086ac0e78a57bf89e6478780001:'
    [5] - [65 bytes] Encrypted data
```

B-22. SLIDE: Setting Up an NIS+ Client

Setting Up a NIS+ Client

1. Create credentials for the client on the NIS+ Server.
2. Assign the domain on the client.
3. Set the client's switch table.
4. Initialize the client.
5. Restart the secure RPC key daemon.
6. Run the **keylogin** program.
7. Reboot the client.

a646197

Student Notes

Setting up a NIS+ client requires the NIS+ server to have pre-defined credentials in place for both the client machine and the administrator of that client machine. If NIS compatibility mode has been enabled then this is not necessary since clients can access NIS files without having to supply the correct keylogin id. Again, the client must be instructed to use the NIS service by priming the `/etc/nsswitch.conf` file to access NIS+ rather than the native files within the `/etc` directory. To initialize the client, the `nisinit -c` command is invoked. By inclusion of the `-B` flag the client broadcasts for the nearest server which is responded to by the NIS+ server which places its Internet address in the clients cold start file. To make a certain host the NIS+ client, the `-H Hostname` flag is utilized instead of the `-B` flag. The hostname specified must be present within the clients `/etc/hosts` file since NIS+ has not been started at the point at which this contact is made.

If remote clients are not Solaris based then the administrator must configure the NIS+ service to run under NIS compatibility mode. Utilities such as `yppasswd` on those machines will not be compatible with the NIS+ equivalent commands (`nisyppasswd`) and so the administrator should utilize other standard NIS servers and set-up scripts that will propagate data between the NIS & NIS+ servers.

Appendix C — Digital UNIX

Objectives

Upon completion of this module you will be able to do the following:

- Document the differences between Digital UNIX and HP-UX.

C-1. SLIDE: Digital UNIX Network Configuration

Digital UNIX Network Configuration

Basic Steps:

1. Ensure appropriate software is installed.
2. Design network, assign numbers, hostnames and so forth.
3. Configure appropriate services during installation phase of Digital UNIX.
4. Optionally configure the system to use NIS, DNS, or a variety of functions.

a646163

Student Notes

During the installation of the Base Operating System from install media, the TCP/IP components of the operating system also are installed. The administrator is asked many questions concerning the functionality of the TCP/IP configuration. The user must allocate Internet addresses, sub-net masks and optionally configure for routing.

C-2. SLIDE: Digital UNIX Network Configuration — Scripts

Digital UNIX Network Configuration – Scripts

The main scripts involved in setting up the network at boot phase are as follows:

/sbin/init.d/inet runs the **hostname**, **ifconfig** and **iprsetup**

Hostname and interface are configured as defined in the **/etc/rc.config** file.

iprsetup is called for a system with more than one physical network interface

/etc/rc.config configures all network parameters such as hostname, IP addresses, NIS, NFS and DNS

/sbin /init.d/route runs the **route** command

Static routes are configured as defined in **/etc/routes**.

a646198

Student Notes

The network configuration of the system involves a few scripts and programs. As is true for HP-UX, all parameters are configured in the **/etc/rc.config** configuration file. Once in place, the system administrator can edit this file directly to effect various changes within the system. Another method of configuration is to run the **netsetup** utility.

The scripts that make up the Digital UNIX TCP/IP configuration suite are as follows:

/sbin/init.d/inet

Network variables set in **/etc/rc.config** are evaluated by **/sbin/init.d/inet**. This script runs the **hostname** and **ifconfig** command configuring the loopback interface, and then each subsequent interface that is installed within the system. This is achieved in a compatible Digital UNIX fashion with each interface having a set of variables defined in **/etc/rc.config**. The nodename for the machine is set from the contents of the variable **HOSTNAME**.

iprsetup

If a system has more than one physical interface **iprsetup** will be called. **iprsetup** sets or unsets the **ipforwarding** flag. **ipforwarding** is necessary for routing IP-packets from one interface to the other physical interface.

iprsetup -s sets **ipforwarding** to 1.

iprsetup -r resets **ipforwarding** to 0.

iprsetup -d displays the value of the **ipforwarding** flag.

/etc/rc.config

The following variables are defined in **/etc/rc.config**:

```
HOSTNAME=moon
NUM_NETCONFIG=1
NETDEV_0=tu0
IFCONFIG_0=7.10.1.14 netmask 255.255.0.0
```

In case of several physical interfaces, the appended index of **NETDEV_0** and **IFCONFIG_0** is incremented to **NETDEV_1** and **IFCONFIG_1**, and so on.

/etc/routes

This file configures static routes.

Per line, this file contains options that are appended to the route command.

```
-net 192.6.128.0    7.10.1.253
-net 0              7.10.8.254
```

The route command is called in **/sbin/inet.d/route**. If no static routes are defined, the dynamic route daemon **routed** is called instead.

Final amendments that should be considered by the administrator concern the files hosts, networks, netmasks, protocols, and services. This is true particularly if this is not a NIS client, because the native files within the machine will be used to resolve data queries. The **/etc/svc.conf** file should be amended for the desired mode of operation of the machine. For **/etc/svc.conf** see later in this chapter.

C-3. SLIDE: Configuration of BIND under Digital UNIX

Configuration of BIND under Digital UNIX

- Operation of **named** is the same as with other UNIX systems.
- Configuration is achieved by creating:
 - the DNS file
 - the **/etc/resolv.conf** file
 - **/etc/svc.conf** for search order of name resolution
- **named** boot file is located in **/etc/namedb/named.boot**
- Use **bindsetup** to configure DNS configuration files.
- The **named** process is started by **/sbin/init.d/named**.

a646164

Student Notes

The DNS components of Digital UNIX are started from the script **/sbin/rc3.d/S15named**, which is linked symbolically to the file **/sbin/init.d/named**. The process of configuring the server to run named is very similar to that already described in the HP-UX section.

The presence of the variable **BIND_CONF** in the **/etc/rc.config** file launches the **named** process.

Further entries in **/etc/rc.config** are:

```
BIND_CONF=YES  
BIND_SERVERTYPE=PRIMARY
```

The **-b** option is used to the command to specify the location of the boot file in **/etc/namedb/named.boot**.

The **named.boot** file may be created from scratch as must the files for each component of the database suite. The layout of the named configuration files is identical to those indicated earlier.

Instead of editing the configuration files by hand one can run the **bindsetup** command. Default **bindsetup** reads the source file **/etc/namedb/src/hosts** instead of **/etc/hosts**. To translate the local **/etc/hosts** file, copy it to **/etc/namedb/src/hosts**.

C-4. SLIDE: The /etc/svc.conf file

/etc/svc.conf File

Order of interrogated services:

/etc/svc.conf:

```
passwd=yp, local  
group=yp, local  
host=bind, local
```

a646199

Student Notes

The **svc.conf** file dictates the order in which services are interrogated for a given search parameter.

The resolver routines can process all name resolution requests according to the order specified within **/etc/svc.conf**. The syntax is rather simple.

```
passwd=yp,local  
group=yp,local  
hosts=bind,local
```

There are the sources **local**, **bind**, and **yp** corresponding to the nsswitch.conf sources **files**, **dns**, and **nis**, respectively.

There is no documented way to change the behavior as in **nsswitch.conf**.

The token **bind** should be present as the first entry within the hosts section of this file in order to use DNS first.

C-5. SLIDE: Debugging BIND under Digital UNIX

Debugging BIND under Digital UNIX

Signals and flags enable debugging of DNS.

named -d N generates debug output to file **/var/tmp/named.run**

Signals are available to debug a running **named** process.

SIGHUP	causes named to reread named.boot
SIGINT	dumps database to /var/tmp/named_dump.db
SIGUSR1	turns on debugging, each subsequent signal increases level, sends output to /var/tmp/named.run
SIGUSR2	turns off debugging completely

a646200

Student Notes

Various flags and signals are available to enable successful debugging of the **named** process. **named** under Digital UNIX responds to the same signals as the other versions of UNIX detailed within this course (output being placed under **/var/tmp**).

C-6. SLIDE: **sendmail Configuration under Digital UNIX**

sendmail Configuration under Digital UNIX

The **sendmail** file provides rules for

- TCP/IP
- DECNet
- UUCP

Mail forwarding uses a domain address structure for a standard mail client system.

Mail aliases are set using the **newaliases** command.

Mail aliases are parsed according to values defined within **/etc/svc.conf**.

a646165

Student Notes

The **sendmail** suite of programs is supplied under Digital UNIX to provide mail connectivity for TCP/IP, DECNet and UUCP connectivity.

The **sendmail** system within Digital UNIX is very similar to that described fully in previous modules.

A number of options are available within Digital UNIX depending upon the construction of the network layout. Digital UNIX comes supplied with a standard **sendmail** configuration file that should be adequate for any system that is within a single domain or does not connect to any other network.

However, if the host is to act as a mail relay/gateway or mail host, then the **sendmail** configuration file **/var/adm/sendmail/sendmail.cf** has to be adapted. Several changes will then need to be administered within this file depending upon the connectivity methods that will be employed to connect to other system networks.

Mail aliases can be set up as within other systems.

C-7. SLIDE: Components of Digital UNIX sendmail

Components of Digital UNIX **sendmail**

<code>/usr/sbin/sendmail</code>	the sendmail program
<code>/usr/sbin/sendmail -bd -q15m -om</code>	run directly from <code>sbin/init.d/sendmail</code> as a daemon process
<code>/var/adm/sendmail.cf</code>	standard configuration file
<code>/etc/svc.conf</code>	switch file for alias location selection

a646201

Student Notes

sendmail is configured and primed at boot by the script `/etc/rc3.d/S40sendmail` which is linked to `/sbin/init.d/sendmail`. `/sbin/init.d/sendmail` loads **sendmail** as a server process (`sendmail -bd -q15m -om`). As a daemon , this process is responsible for receiving and queueing SMTP requests over the network.

DRddn-gateway Defines the name of the relay host.

Aliases are defined either in local files or via NIS. The `/etc/svc.conf` file dictates the order in which ASCII files in `/etc`, or NIS are accessed to resolve a given data input. The administrator must edit this file to provide suitable access for the mail alias mechanism. The line is read from left to right, that is:

```
aliases=yp,local
```

Aliases are added locally into the `/var/adm/sendmail/aliases` file.

This is only the raw data file. The attended information that defines the aliases is placed into a binary format in the files `/var/adm/sendmail/aliases.dir` and `/var/adm/sendmail/aliases.pag` using the `newaliases` command.

For the change to take effect, the `newaliases` command must be executed each time the `aliases` file is changed.

Running the command `newaliases` is equivalent to the sendmail command `sendmail -bi`.

NOTE:

Digital UNIX does not have a frozen sendmail configuration file,
`sendmail.fc`.

C-7. SLIDE: Components of Digital UNIX **sendmail** (Continued)

Components of Digital UNIX **sendmail** (Continued)

<code>/usr/bin/mail</code>	the mail program
<code>/var/adm/sendmail/sendmail.cf</code>	sendmail configuration file
<code>/usr/sbin/sendmail/</code>	the sendmail program
<code>/usr/sbin/sendmail -bd -q15m -om</code>	run directly from <code>/sbin/inet.d/sendmail</code>
<code>/var/adm/sendmail/sendmail.cf</code>	as a daemon process standard configuration file

a630287

Student Notes

sendmail is configured and primed at boot by the script `/sbin/inet.d/sendmail`, which translates the aliases file to its binary equivalents and then loads **sendmail** as a server process. This process is responsible for sending and receiving SMTP requests over the network.

The operation, de-bugging and troubleshooting of Digital UNIX **sendmail** is identical to that detailed for HP-UX.

C-8. SLIDE: Serial Line Communications

Serial Line Communications

Digital UNIX supports only the SLIP protocol.

Steps to configure SLIP:

1. Ensure the kernel driver for SLIP is installed.
2. Define entries in **/etc/hosts**.
3. Configure the SLIP interface.
4. Run **slattach** for a serial port.

a646202

Student Notes

Digital UNIX handles Serial Line Internet communication via the SLIP Protocol.

Perform the following steps, if you wish to configure your system to use SLIP:

1. Configure the SLIP option to the host's kernel configuration file:

sys/conf/HOSTNAME:

options SL

By default, an entry for SLIP exists in the kernel configuration file.

The name of the SLIP interface is **s10**.

2. Define IP address and hostname in the **/etc/hosts** file for both ends of the SLIP network.

193.1.1.1	host1_sl
193.1.1.2	host2_sl

3. Configure the SLIP interfaces with the **ifconfig command.**

```
On host1_sl:
  ifconfig sl0 193.1.1.1 193.1.1.2

On host2_sl:
  ifconfig sl0 193.1.1.2 193.1.1.1
```

4. Run the **slattach command in order to attach a serial line to a SLIP interface.**

The **slattach** command selects the serial line that will be attached to the SLIP interface. The **slattach** command is also used to configure SLIP options.

The **slattach** command does not specify the SLIP interface to be used. Instead, the first unattached SLIP interface is used. This can be sl0, sl1 or any further SLIP interface which was be configured by the **ifconfig** command. The **slattach** command also specifies the baud rate for the serial connection. The default rate is 9600 baud.

```
slattach tty00 19200
```

The serial line **tty00** is attached to the first free SLIP interface for example **sl0** with a baud rate of **19200**.

The SLIP connection is established as long as the physical connection is ready and the **slattach** command is running. Use the **ps** command to ensure the **slattach** command is running. After the **slattach** command exits due to a network error or the termination of the **slattach** command itself, the command can be executed again to reestablish the SLIP connection.

For a direct connection with a null modem the SLIP connection is always ready. In case of a phone connection, the connection is established by manually dialing the modem on the local system to connect to the modem on the remote system. Once the remote modem answers, the data/talk button should be pressed to allow the modem on the local host to assume control of the connection.

Stopping and Restarting SLIP

The SLIP network is stopped by using the **kill** command to kill the running **slattach** process that has attached a serial line to SLIP.

C-9. SLIDE: The slattach command

The **slattach** command

```
slattach [+cl -c] [+el -e] [+il -i] ttyname [baudrate]
```

Options:

+l -c

enables/disables TCP header compression.

+l -e

enables/disables automatic TCP header compression.

+l -i

enables/disables ICMP traffic suppression.

a630299

Student Notes

The **slattach** command attaches a **tty** line to a SLIP network interface. The default baudrate is 9600 baud.

Only the superuser is allowed to attach a serial line to a network interface.

The **tty** line is attached to the first available network interface sl0, sl1, ...

NOTE: The SLIP interface is already configured with the local and remote addresses of each end of the SLIP connection.

The **slattach** command has the following options:

+c Enables TCP header compression.

-c Disables TCP header compression.

- +e Enables automatic TCP header compression. TCP header compression will be used only if this option is enabled and the remote system is using TCP header compression.
- e Disables automatic TCP header compression.
- +i Enables ICMP traffic suppression. If enabled, ICMP traffic is not allowed to pass over the SLIP connection.
- i Disables ICMP traffic suppression. This is the default.

C-10. SLIDE: The Protocol Analyzer `tcpdump`

```
/usr/sbin/tcpdump [ -d eflnNOpqStvx] [ -c count]
[ -F filter_expression_file]
[ -i interface ] [ -r input_file ] [ -s snaplen ]
[ -w output_file ] expression
```

Example:

```
tcpdump host hpweb
tcpdump host loon and (mars or earth)
tcpdump port ftp or ftp-data
```

a630300

Student Notes

The `tcpdump` utility prints out the headers of packets on a network interface that match the boolean expression. The kernel must be configured with the `packetfilter` option.

NOTE:

To watch either outbound or inbound traffic, the `copyall` mode has to be enabled. `copyall` mode can be enabled with the `pfconfig` command.

`pfconfig +c tu0`

Useful options:

- d Dumps the compiled packet-matching code to standard output and stop.
- e Prints the link-level header on each dump line.
- F Uses `filter_expression_file` as input for the filter expression. Any additional expressions on the command line are ignored.

- i Listens on interface. If unspecified, **tcpdump** searches the system interface list for the lowest numbered, configured up interface excluding loopback.
- n Does not convert host addresses and port numbers to names.
- q Quick output. Prints less protocol information so output lines are shorter.
- r Reads packets from **input_file** created with the -w option.
- s Displays **snaplen** bytes of data from each packet rather than the default of 68. The default of 68 bytes is sufficient for IP, ICMP, TCP, and UDP, but may truncate protocol information from name server and NFS packets.
- v Prints slightly more verbose output, as for example the time to live and type of service information in an IP packet.
- vv Prints even more verbose output, as for example additional fields from NFS reply packets.
- w Writes the raw packets to file. They can be formatted later with the -r option.

Expressions

Select the packets to dump. If no expression is given, all packets on the network are dumped.

The expression consists of one or more primitives. Primitives usually consist of an **id**, name or number preceded by one or more of the following qualifiers:

type Defines the object to which the **id** name or number refers. The following types are allowed: **host**, **net**, and **port**. For example:

```
host yourhost
net 128.3
port 20
```

direction Specifies the particular transfer direction destination (**dst**) **i** or source (**src**).

```
src yourhost
dst net 128.3
```

proto Restricts the output to a particular protocol. The following protocols are possible: **ether**, **fddi**, **ip**, **arp**, **rarp**, **decnet**, **lat**, **moprc**, **mopdl**, **tcp**, and **udp**.

```
ether src foo
arp net 128.3
tcp port 21
```

The default is to analyze all protocols.

Primitives

Primitives may be combined by using the following:

- A parenthesized group of primitives and operators.
- Negation (not)
- Concatenation (and)
- Alternation (or)

To watch either outbound or inbound traffic, you need to have enabled `copyall` mode using the `pfconfig` command. For example, `pfconfig +c Ln0`.

Name server inverse queries are not dumped correctly: The (empty) question section is printed rather than real query in the answer section.

A packet trace that crosses a daylight saving time change produces skewed time stamps (the time change is ignored).

Filter expressions that manipulate FDDI headers assume that all FDDI packets are encapsulated Ethernet packets. This is true for IP, ARP, and DECNET Phase IV, but is not true for protocols such as ISO CLNS. Therefore, the filter may inadvertently accept certain packets that do not properly match the filter expression.

Following are a few examples of filter expressions:

To filter out all packets arriving at or departing from `hpweb`:

```
tcpdump host hpweb
```

To filter out traffic between `loon` and either `mars` or `earth`:

```
tcpdump host loon and \(\ mars or earth \)
```

To print all FTP traffic:

```
tcpdump port ftp or ftp-data
```

For more information see `man tcpdump`.

Solutions

2-21. LAB: Point-to-Point Protocol

1. On the two PPP systems:

Verify that the **PPP-RUN** fileset is loaded and that the PPP drivers are in the kernel. Normally, this is done when loading either the core networking bundle or the LAN/9000 networking product.

Answer:

```
# swlist -l fileset | egrep -i ppp
    Networking.PPP-RUN                                B.11.00      PPP-RUN
# grep tun /stand/system
tun
```

2. Attach a null modem cable between the two systems that will be linked via the PPP connection. For ease of configuration, it is suggested that you use the same serial port number on both systems (e.g., RS-232 Interface (#1)).

Answer:

Did you remember to use the same serial port number?

3. Using SAM, add the serial interface to each of the systems. As this is a *hard-wired* interface you can configure this port as a *terminal*.

Answer:

```
SAM:Peripheral Devices -> Terminals and Modems
[OK] to "No currently configured terminals and/or modems...."
Actions -> Add Terminal ...
Select an available serial port (e.g., RS-232 Interface (#1))
Speed [ 38400 ]
    NOTE: Select a slower speed to start with
          and increase it as link proves its reliability. Keeping
          the initial rate to 38.4 or less also allows the use
          of a simple ASCII terminal for troubleshooting.

[OK]
[OK] to "The terminal has been added."
    Make note of the device file created (e.g., /dev/tty0p0 )
    Exit SAM
```

NOTE:

This SAM will have added a **getty** process to **/etc/inittab** which will need to be turned off in step 5.

4. Check the device just created on each of the systems.

Answer:

```
# ls -l /dev/tty0p0
crw--w--w- 1 root bin 1 0x000000 Dec 9 12:19 /dev/tty0p0
```

5. Turn off the **getty** for the newly created device on each of the systems. Change the word *respawn* to *off* for the created **tty** in **/etc/inittab**.

For example, change **a0:3:respawn:/usr/sbin/getty -h tty0p0 38400**
to **a0:3:off:/usr/sbin/getty -h tty0p0 38400**.

Then, have **init** reread **/etc/inittab** to kill off the running **getty**.

Answer:

```
# init q
```

6. Check out the basic link. On one system, type

```
# cat < /dev/tty0p0
```

While on the other system, type

```
# echo hello world > /dev/tty0p0
```

Answer:

If the basic link is good, you will see "hello world" being echoed on the system running the **cat** command.

NOTE: The actual device names will depend on the devices that were created through SAM (see step 3 above).

7. Create the PPP configuration files in **/etc/ppp**.

NOTE: The keywords *Any* and *Direct* must be capitalized and the file **/etc/ppp/Autostart** must be executable.

In the following examples, replace **<remote_ip_address>** with the IP address of the node on the remote side of the PPP link. For example, if node **santiago** is being configured, use the IP address of **blackburn**.

Answer:

```
# vi /etc/ppp/Autostart
    pppd 'hostname': <remote_ip_address> auto dedicated ignore-cd
# vi /etc/ppp/Systems
```

```
<remote_ip_address> Any tty0p0 38400 0  
# vi /etc/ppp/Devices  
    Direct          tty0p0 38400
```

8. Start the PPP daemon.

Answer:

```
# /sbin/init.d/ppp start  
Creating 16 tunnel device nodes at major 141...  
Starting PPP daemons
```

9. Check to make sure everything started OK. Make sure the connection is established. The network interface name is du0.

Answer:

On the machine initiating the connection, you should see something like

```
# ps -ef | grep pppd  
root 5244 1 0 09:41:01 tty0p0 0:00 pppd  
[connecting, du0, tty0p0] santiago:192.68.61.142 auto dedicat
```

Once the connection is established, you should see something similar to

```
# ps -ef | grep pppd  
root 5244 1 0 09:41:01 tty0p0 0:00 pppd  
[up, du0, tty0p0, 192.68.61.142, 54/54 bps] santiago:19
```

In the above example, the network interface name is du0.

```
# netstat -i  
Name      Mtu Network          Address          Ipkts      Opkts  
lo0       4136 127.0.0.0        localhost        1588       1588  
lan0      1500 156.153.200.0     santiago        72291      2748  
du0       1500 156.153.0.0      santiago        72291       765  
  
# ifconfig du0  
du0: flags=851<UP,POINTOPOINT,RUNNING,MULTICAST>  
      inet 156.153.200.141 --> 192.68.61.142 netmask fffff0000
```

10. It is also a good idea to establish some baseline parameters for the link at this time:

Answer:

```
# ping 192.68.60.142 -n 3  
PING 192.68.60.142: 64 byte packets  
64 bytes from 192.68.60.142: icmp_seq=0. time=69. ms  
64 bytes from 192.68.60.142: icmp_seq=1. time=70. ms  
64 bytes from 192.68.60.142: icmp_seq=2. time=60. ms  
  
----192.68.60.142 PING Statistics----
```

```
3 packets transmitted, 3 packets received, 0% packet loss
round-trip (ms) min/avg/max = 60/66/70
```

11. To shut down the daemon use a **kill** signal. Do *not* use a **kill -9**, as this may leave the tunnelling device in a busy state and require a system reboot to clear.

Answer:

```
# kill 5244
```

If the daemon on one side of the PPP link should be killed, the surviving daemon will continue to reestablish the link. You may see output similar to the following:

```
# ps -ef | grep pppd
root 5244 1 0 09:41:01 tty0p0 0:01 pppd
[disconnected, du0, tty0p0, 54/54 bps] santiago:192.68.61
```

12. Once the link has demonstrated its reliability, try changing the data rate in the **/etc/ppp/Devices** and **/etc/ppp/Systems** files on both systems to other values, stop and restart the PPP daemons, and observe what happens.

Answer:

What did you see?

3-11. LAB: Finding and Resolving Network Faults

1. Complete the following:
 - a. How can you find the network adapters available on your system?
 - b. For each of the interfaces determine the following:
 - i. the hardware address
 - ii. the link layer level protocol
 - iii. the hardware state
 - iv. the IP address
 - v. the subnet mask
 - vi. the broadcast address
 - vii. the MTU size

Answer:

- a. Use either `lanscan` or `netstat -i` to determine which network adapters are available on your system.
- b.
 - i. `lanscan` will display the hardware address.
 - ii. `lanscan` will display the link layer level protocol encapsulation used.
 - iii. `lanscan` will display hardware and software state of the interface.
 - iv. `ifconfig` will display the IP address.
 - v. `ifconfig` will display the subnet mask.
 - vi. `ifconfig` will display the broadcast address.
 - vii. `lanadmin -m NMID` will display the MTU size.

2. `ping` the following addresses:

- a. the network broadcast address
- b. a broadcast address with all bits equal to 1.
- c. an address with all bits equal to 0

Which hardware address(es) will your ARP cache contain?

Answer:

- a. Answers will vary depending on network address. Use `ifconfig` to determine the network broadcast address.

For example: with an unsubnetted class A address, you use `ping 10.255.255.255`.

- b. `ping 255.255.255.255` is the global broadcast address. Typically, routers block broadcasts.
- c. `ping 0.0.0.0` translates to `this_network.this_host`. The IP code converts it to the local host address (127.0.0.1). `pinging` the local node does *not* exercise any actual network interface cards; it uses the software loopback instead.

The `arp -a` command will show that the ARP cache contains the hardware address(es) of all network interfaces that are on the local network, except the local node.

3. Answer all parts of the following:

- a. Determine how many collisions occurred on your Ethernet segment.

The **spray** command is comparable to the **ping** command, but is used to help provide performance measurements of the RPC based applications.

- b. Activate the **rpc.sprayd** server on your system.
- c. Send 10000 packets to a neighbor's system using the **spray** command.
- d. What was the effect on the collision rate?

Answer:

- a. Use **lanscan** to determine what the physical point of attachment (PPA) number of the interface is.

```
# lanscan
  Hardware Station      Crd Hdw   Net-Interface  NM  MAC    HP-DLPI DLPI
  Path     Address      In# State NamePPA       ID  Type   Support Mjr#
  2/0/2   0x0800090B1D26 0    UP    lan0 snap0      1  ETHER Yes    119
```

Then use **lanadmin** to look at the statistics for that interface. From the menu, select **lan**, then **ppa**. If the PPA is correct, simply press return; otherwise enter the correct PPA. After returning to the LAN interface test mode, select **display**. This display will show the collision statistics, among others.

- b. Uncomment the following line in **/etc/inetd.conf**.

```
rpc dgram udp wait root /usr/lib/netsvc/spray/rpc.sprayd 100012 1 rpc.sprayd
```

Send the hangup signal to the **inetd** server using **/usr/sbin/inetd -c** or **kill -HUP** so that **/etc/inetd.conf** will be reread.

- c. Use the RPC client command **spray -c 10000** to communicate with the **rpc.sprayd** server on a neighbor's system (he or she will do likewise to your system).
 - d. Follow the instructions in step a to see what the current interface statistics are. In general, there will be a few more collisions.
4. Perform the following tasks:
- a. Determine the hostname and nodename of your system.
 - b. Temporarily change the nodename to *guardian*.
 - c. What, if anything, has happened to your hostname?
 - d. Cross check your hostname with your actual IP address and the reverse.
 - e. Set your nodename back to its original value.
 - f. Cross check your hostname with your actual IP address and the reverse.

Answer:

- a. `hostname` will display your system hostname which is used by the ARPA and NFS Services. `uname -n` will display your system nodename which is used by uucp and other related programs.
- b. `uname -s` guardian
- c. `hostname` remains unchanged (on HP-UX)
- d. `uname -S <your_hostname>`
- e. Many network problems can be traced to incorrect symbolic name to IP address translation.

First, determine your actual IP address using `ifconfig`

```
# ifconfig lan0
lan0: flags=63 <UP,BROADCAST,NOTRAILERS,RUNNING>
inet <your_ip_address> netmask <your_net_mask> broadcast <your_bcast_addr>
```

Then, use `nslookup` to verify proper mapping of the symbolic name to IP address and IP address to hostname. For example, using either

```
# nslookup <your_hostname>
```

or

```
# nslookup <your_ip_address>
```

should result in the same output.

Using `/etc/hosts` on `<your_hostname>`

Name: `<your_hostname>`

Address: `<your_ip_address>`

and the IP address should match that produced by `ifconfig`.

5. Login as root on your system. Issue the command:

```
# telnet localhost
```

- a. Determine which sockets are being used by `telnet`.
- b. Determine which sockets are the telnet clients and which are the servers.

Answer:

- a. Use the `netstat` command to see what socket connections are open for telnet.

```
# netstat -a | grep telnet
tcp    0    0  *.telnet          *.*                  LISTEN
tcp    0    0  localhost.telnet  localhost.56348     ESTABLISHED
tcp    0    0  <your_hostname>.56316 <your_hostname>.telnet ESTABLISHED
```

- b. The telnet server will be listening at the port defined for it (usually TCP port 23) in the `/etc/services` file. The `netstat` command will attempt to map port numbers to symbolic service names. As clients are not bound to fixed port numbers, they cannot be resolved symbolically.

In the above example, the second line shows an inbound telnet server connection to the localhost from the localhost, while the third line shows an outbound client connection to a remote server connection.

6. Use the command `arp -a` to get a list of the hardware addresses of systems connected to the local network.
- Test the link level connectivity to that system.
 - Can this test be used across a router?

Answer:

- a. Execute the `linkloop` command using one of the link level addresses from the output of the `arp`.

For example:

```
# arp -a
# linkloop 0x<remote_station_address>                                -- OK
Link connectivity to LAN station: 0x<remote_station_address>
```

- b. The link level connectivity test is based on layer two of the ISO/OSI model. Routing is the responsibility of layer three. Therefore, the link level test cannot be performed across a router.

3-17. LAB: System Logging Daemon—`syslogd`

1. Which option is used with the `ftp` server to enable it to log messages to the `syslogd` daemon?

In which configuration file do you have to configure the `ftpd` server?

Enable the `ftp` server to log to `syslogd`.

Answer:

The `ftp` server, `ftpd`, uses the option `-l` (Editor's note: that's ell) to enable logging to the standard system logging daemon `syslogd`. By default `ftpd` does not log.

The **ftpd** server is configured in **/etc/inetd.conf**. There should be a line similar to the following:

```
ftp      stream  tcp  nowait  root  /etc/ftpd      ftpd -l
```

If the line is missing or in error, it should be corrected. Then the Internet daemon, **inetd**, will need to be notified of the change by using:

```
inetd -c
```

When reconfiguring the Internet daemon it is a good idea to check the system log file **/var/adm/syslog/syslog.log** as any **inetd** configuration errors that occur are logged there.

2. Configure the system logging daemon, **syslogd**, for the following subsystems:
 - a. Messages generated by network daemons with a level of INFO and greater should be sent to the file **/var/adm/syslog/daemon.log**, while those with a level of WARNING or greater should be sent to the console.
 - b. Messages generated by the mail subsystem with a level of DEBUG and greater should be sent to the file **/var/adm/syslog/mail.log**.
 - c. Choose a partner. Configure your system to send all messages generated by the **auth** (authentication) subsystem to the **syslogd** on your partner's system.

Do not delete any existing entries.

Answer:

- a. Add the following lines to **/etc/syslog.conf**:

```
daemon.warning      /dev/console  
daemon.info        /var/adm/syslog/daemon.log
```

- b. The line reading

```
mail.debug          /var/adm/syslog/mail.log
```

should already exist in **/etc/syslog.conf**.

- c. Add a line similar to the following to **/etc/syslog.conf**:

```
auth.debug          @remotehost
```

where **remotehost** is actually the name of your partner's system.

3. Start an **dtterm** with the **-c** option. The **-c** option causes all console messages to be sent to the **dtterm**.

- a. Determine the process ID (PID) of the **syslogd** daemon.

- b. Which signal do you have to use so that **syslogd** will read the modified configuration file **/etc/syslog.conf** again?
 - c. Send the appropriate signal to the **syslogd** daemon to have it reread its configuration file.
4. Connect to your own system using **ftp**. Log in as the user **cracker** with the password **BaDdOoD**. The user **cracker** should not have a valid account on the system.

Answer:

```
# ftp localhost
Connected to localhost.edunet.hp.com.
220 localhost FTP server (Version 1.7.193.3 Thu Jul 22 18:32:22 GMT 1993) ready.
Name (localhost:root): cracker
331 Password required for cracker.
Password:
530 Login incorrect.
Login failed.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> quit
221 Goodbye.
```

In **/var/adm/syslog/daemon.log**:

```
Dec 23 17:55:04 localhost ftpd[12301]:
connection from localhost at Tue Dec 23 17:55:04 1997
Dec 23 17:55:09 localhost ftpd[12301]:
User cracker: Login incorrect
```

On the console:

```
Dec 23 17:55:04 localhost ftpd[12301]:
connection from localhost at Tue Dec 23 17:55:04 1997
Dec 23 17:55:09 localhost ftpd[12301]:
User cracker: Login incorrect
```

Note that they are both the same.

5. Use the following command to send the bogus mail:

```
# echo "breaking in" | mailx cracker@localhost
```

Notice that nothing was added to **/var/adm/syslog/daemon.log**. Instead this information was appended to **/var/adm/syslog/mail.log**.

Answer:

For example,

```
Dec 23 18:14:18 localhost sendmail[12336]: SAA12336: to=cracker@localhost,
      delay=00:00:00, mailer=local, stat=User unknown
Dec 23 18:14:19 localhost sendmail[12336]: SAA12336: from=root, size=154,
```

```
class=0, pri=30154, nrcpts=1, msgid=<199712232314.SAA12336@localhost.esr.hp.com>,  
relay=root@localhost  
Dec 23 18:14:19 localhost sendmail[12336]: SAA12336: SAB12336: postmaster  
notify:User unknown  
Dec 23 18:14:19 localhost sendmail[12338]: SAB12336: to=root, delay=00:00:00,  
xdelay=00:00:00, mailer=local, stat=Sent
```

6. In its simplest form, the following logger command could be used to enter information into the **syslog** subsystem.

```
# logger "Attention: Cracker suspected"
```

The default subsystem and priority would be **user.notice** and, based on the default configuration of the **syslogd** daemon, a message similar to the following would appear in **/var/adm/syslog/syslog.log**:

```
Dec 23 18:24:41 localhost root: Attention: Cracker suspected
```

7. The following logger command should forward a message to your partner's **syslog** daemon:

```
# logger -p auth.crit "Cracker Attack"
```

Answer:

The output should be appended to your partner's **/var/adm/syslog/syslog.log** file and appear similar to the following:

```
Dec 23 18:25:46 localhost root: Cracker Attack
```

Since you can specify any combination of subsystem and priority; the logger facility can also be used as a method of testing the syslog configuration. Simply script a series of logger calls that will exercise each of the configuration file entries.

For example,

```
# logger -p auth.alert "Attention: Cracker suspected"
```

would have exercised the following two entries in **/etc/syslog.conf**

```
*.alert          /dev/console  
*.alert          root
```

and resulted in a message being sent to the system console, as well as all root logins.

3-23. LAB: Using nettl, netlfmt, and nettladm

1. Determine if the trace and logging subsystem is activated on your system.

What is the name of the log file?

What is its maximum size?

Read the entries in the log file.

Answer:

When files are created, they have the suffix `.LOG00` or `.TRC0` appended to them, depending on whether they are log or trace files, respectively. The default name is `/var/adm/nettl` (thus logging starts to file `/var/adm/nettl.LOG00`).

The files implement a type of circular buffer, with new data always going into the file appended with `.LOG00` or `.TRC0`. When a file is full, it is renamed to the next higher number in its sequence; that is, `logname.LOG01` or `tracename.TRC1` and a new file named `logname.LOG00` or `tracename.TRC0` is created.

Currently, only two generations of files are possible. Users can specify a file name prefix (`logname` or `tracename`), which must not exceed eight characters, so that the file name plus suffix does not exceed fourteen characters. Longer names are truncated. The maximum size for both trace (or log) files combined is from 100 to 99999 kilobytes (KB). The default value for the combined file sizes is 1000 KB.

2. Use `nettl` to capture packets transiting the network interface. For example, trace an ICMP Echo Request (`ping`) packet to a remote destination and its reply.

To avoid capturing too many packets, use `nettl` only on packets transiting the network driver. Do not capture state, exit code, and so forth.

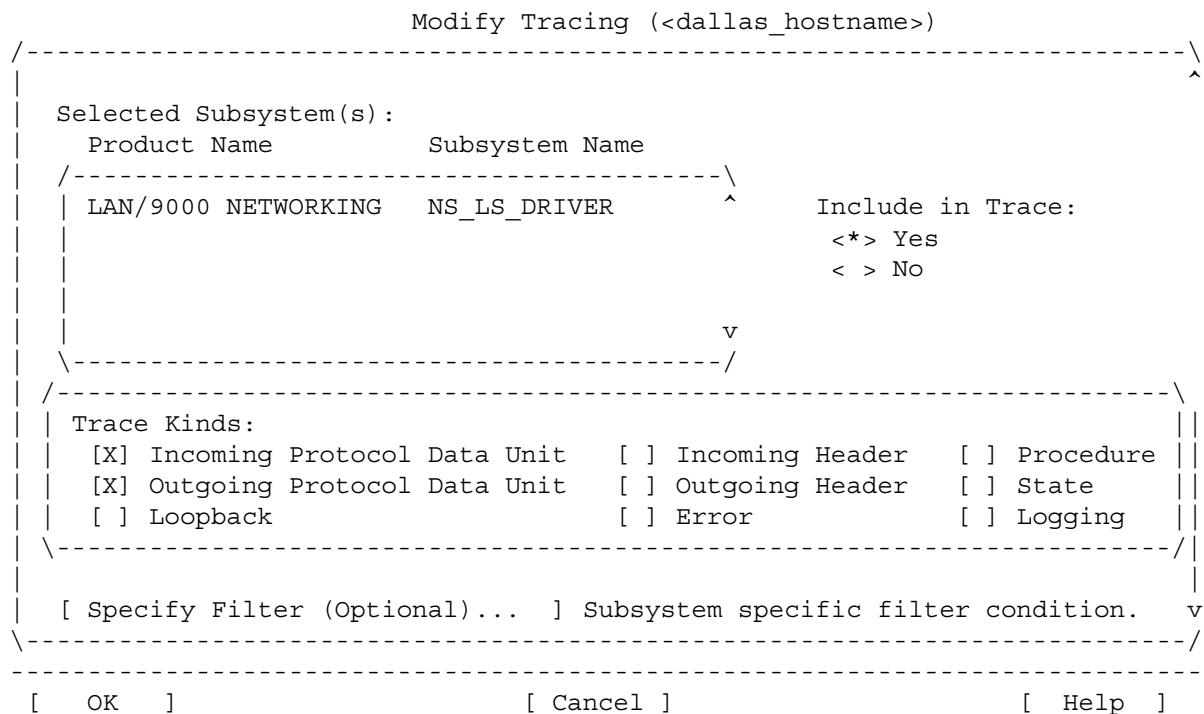
Finally, save all the collected data put in the file `/tmp/trace`.

Answer:

Begin by defining what items you want to trace by invoking `nettladm`.

From the menu bar in the Network Tracing and Logging dialog box, select "List -> Tracing Subsystems". Scroll down to the "LAN/9000 NETWORKING NS_LS_DRIVER" and select it. Then, from the menu bar select "Actions -> Modify Tracing...".

In the Modify Tracing dialog, press the `Yes` button. Be sure to include only incoming and outgoing protocol data units for trace kinds. Then press `OK`.



From the menu bar in the Network Tracing and Logging dialog select "Actions -> Modify Startup Parameters...". Change the current trace file name to `/tmp/trace` in the Modify Tracing Startup Parameters dialog, and then press **OK**.

To start tracing, simply select "Actions -> Start Tracing" in the Network Tracing and Logging dialog.

Now execute the `ping` command: `# ping a4430edc -n 1`

Note that you can check on the current status of the tracings by selecting "Actions -> View Status...".

Once the `ping` command has finished, stop the tracing (otherwise a lot of disk space could be used up fairly quickly) by selecting "Actions -> Stop Tracing" in the Network Tracing and Logging dialog.

3. Format the gathered trace data. If too much data exists in the output and makes it difficult to read, set up a filter so that only packets to or from your destination will be shown.

To limit the information displayed by the `netfmt` command, you will need to create a configuration file that defines which traces are of interest. In your home directory, create a file called `.netfmtrc` whose contents are:

```
filter ip_saddr
filter ip_daddr
```

Note that there are two variables defined here. The first, `ip_saddr`, says you are interested in packets that have the associated IP address as an IP packet source address. The second, `ip_daddr`, similarly defines an interest in packets that have the associated IP address as an IP packet destination address.

To view the stored trace information, use the command **netfmt**.

```
# netfmt -N -f /tmp/trace.TRC0
```

Answer:

```
----- SUBSYSTEM FILTERS IN EFFECT -----  
----- LAYER 1 -----  
----- LAYER 2 -----  
----- LAYER 3 -----  
filter ip_saddr      <dallas_hostname>  
filter ip_daddr      <dallas_hostname>  
----- LAYER 4 -----  
----- LAYER 5 -----  
----- END SUBSYSTEM FILTERS -----
```

```

vvvvvvvvvvvvvvvvvvvvvvvLAN/9000 NETWORKINGvvvvvvvvvvvvvvvvvvvvvvv@%
Timestamp : Tue Dec 30 EST 1997 12:11:25.343533
Process ID : [ICS] Subsystem : NS_LS_DRIVER
User ID ( UID ) : -1 Trace Kind : PDU OUT TRACE
Device ID : 0 Path ID : 0
Connection ID : 0
Location : 00123
=====
===== ETHER =====
Source : 08-00-09-1a-2b-3c [I] [HP ] TYPE: DOD IP
Dest : 08-00-09-a1-b2-c3 [I] [HP ] TRACED LEN: 98
Date : Tue Dec 30 12:11:25.034353 EST 1997
===== IP Header (outbound -- [ICS]) =====
Source: <dallas_hostname>(A) Dest: <paloalto_hostname>(A)
    len: 84      ttl: 255     proto: 1      cksum: 0x3d89      id: 0x5a41
    flags: NONE    tos: 0x0     hdrlen: 20     offset: 0x0     optlen: 0
----- ICMP Header -----
type: ECHO          cksum: 0x1e6f          id: 22323          seq: 0
code: none
----- User Data -----
  0: 34 a9 2b 3d 00 05 37 6f 08 09 0a 0b 0c 0d 0e 0f  4.=..7o.....
  16: 10 11 12 13 14 15 16 17 18 19 1a 1b 1c 1d 1e 1f  .....
  32: 20 21 22 23 24 25 26 27 28 29 2a 2b 2c 2d 2e 2f  !#$%&'()*+,-./
  48: 30 31 32 33 34 35 36 37 -- -- -- -- -- -- -- 01234567.....
^^^^^^^^^^^^^^^^^LAN/9000 NETWORKING^^^^^^^^^^^^^^^^^@%
Timestamp : Tue Dec 30 EST 1997 12:11:25.344317
Process ID : [ICS] Subsystem : NS_LS_DRIVER
User ID ( UID ) : -1 Trace Kind : PDU IN TRACE
Device ID : 0 Path ID : 0
Connection ID : 0
Location : 00123
=====
===== ETHER =====
Source : 08-00-09-a1-b2-c3 [I] [HP ] TYPE: DOD IP
Dest : 08-00-09-1a-2b-3c [I] [HP ] TRACED LEN: 98
Date : Tue Dec 30 12:11:25.034431 EST 1997
===== IP Header (inbound -- [ICS]) =====
Source: <paloalto_hostname>(A) Dest: <dallas_hostname>(A)
    len: 84      ttl: 255     proto: 1      cksum: 0xd78e      id: 0xc03b
    flags: NONE    tos: 0x0     hdrlen: 20     offset: 0x0     optlen: 0
----- ICMP Header -----
type: ECHOREPLY        cksum: 0x266f        id: 22323        seq: 0
code: none
----- User Data -----
  0: 34 a9 2b 3d 00 05 37 6f 08 09 0a 0b 0c 0d 0e 0f  4.=..7o.....
  16: 10 11 12 13 14 15 16 17 18 19 1a 1b 1c 1d 1e 1f  .....
  32: 20 21 22 23 24 25 26 27 28 29 2a 2b 2c 2d 2e 2f  !#$%&'()*+,-./
  48: 30 31 32 33 34 35 36 37 -- -- -- -- -- -- -- 01234567.....
vvvvvvvvvvvvvvvvvvvvvLAN/9000 NETWORKINGvvvvvvvvvvvvvvvvvvvvv@%
Timestamp : Tue Dec 30 EST 1997 12:11:29.880749
Process ID : 22362 Subsystem : NS_LS_DRIVER
User ID ( UID ) : 0 Trace Kind : PDU OUT TRACE
Device ID : 0 Path ID : 0
Connection ID : 0
Location : 00123
=====
```

4. Fill in the following table with your results.

	ECHO Header	ECHOREPLY Header
Tracing Subsystem		

Ethernet Protocol		
HW Source Address		
HW Destination Address		
Source Service Access Point		
Destination Service Access Point		
Network Protocol Number		
IP Source Address		
IP Destination Address		
Incoming/Outgoing Packet		
Application Port/Protocol		
Application Information		

Answer:

The information you provide is based on the sample output above. Your actual values will depend on your traced data.

ECHO Header		ECHOREPLY Header	
Tracing Subsystem	NS_LS_DRIVER	NS_LS_DRIVER	
Ethernet Protocol	ETHER	ETHER	
HW Source Address	08-00-09-1a-2b-3c	08-00-09-a1-b2-c3	
HW Destination Address	08-00-09-a1-b2-c3	08-00-09-1a-2b-3c	
Source Service Access Point	N/A	N/A	
Destination Service Access Point	N/A	N/A	

Network Protocol Type	DOD IP	DOD IP
IP Source Address	<dallas_hostname>	<paloalto_hostname>
IP Destination Address	<paloalto_hostname>	<dallas_hostname>
Incoming/Outgoing Packet	outbound	inbound
Application Port/Protocol	N/A	N/A
Application Information	ICMP Echo Request	ICMP Echo Reply

5. You can also view the tracefile from within `nettadm`.
 1. Select *List -> Tracing Subsystems* from the menu bar to work with the tracing subsystem.
 2. Then select *File -> Create Report*
 3. From the Create Report dialog, select *Select Trace File* if the correct trace file is not listed.
This will allow you to *browse* the file systems to select the correct file.
 4. Select *Modify Criteria* which will take you to the Modify Report Criteria for Tracing dialog. Here you can further define which components you wish to view. Under *Subsystems* exclude all subsystems except *LAN/9000 NETWORKING NS_LS_DRIVER* and then press **OK**.
 5. Under *Trace Kind*, include only incoming and outgoing protocol data unit and then press **OK**.
 6. Finally, press **OK** in the Modify Report Criteria for Tracing dialog.
 7. Select *Print Report* and choose a detailed report format with any desired report options.
 8. Choose the screen as the destination. Then press **OK**.

Answer:

Your gathered information should look similar to that obtained by using `nettfmt`, although undoubtedly there will be many more entries.

4-20. LAB: RIP Configuration—Default Rerouting

1. The network should be configured as in the above figure or as the instructor may otherwise direct. Note that in the following instructions the node names used are from the figure. Your instructor will map these into names used at your site, as well as indicate which systems will be used as the gateways, and which nodes will be moved to the 192.168.61 or 192.168.62 networks as appropriate.
2. Verify that `/etc/hosts` is current and complete. This will generally mean adding the IP addresses and hostnames of the systems on the net 192.168.61 and net 192.168.62 side of the gateway systems.
3. The systems will need to adjust their routing so that they can reach the other networks (192.168.61, 192.168.62, or 156.153.2xx). This is done through dynamic routing, so you will need to remove any non-local static routes from your system. Use the `route` command with the `delete` option. If you modified `/etc/rc.config.d/netconf` there to include those routes, you will need to delete them there also.
 - a. Run `/sbin/set_parms initial` on the net 192.168.61.0 and 192.168.62.0 systems to change hostnames, IP addresses, and default routes. The net 192.168.61.0 systems other than `blackburn` should use `blackburn` as their default gateway. The net 192.168.62.0 systems other than `geneve` should use `geneve` as their default gateway. Systems `santiago` and `blackburn` should use `blackburn` and `santiago` respectively as their default gateways.

NOTE: You may need to delete the existing default route first.

Upon completion, verify local network connectivity using `ping`.

The systems `paris`, `hamburg`, and `pinewood` will need to be moved. System `geneve` will have to configure its second interface. Note that you can use SAM to do this.

4. Use the file `/var/tmp/gated.trc` as the trace file for `gated`. Be sure to clear it each time `gated` is restarted.

Trace all general information as well as information about RIP.

Answer:

Create the `gated` configuration file `/etc/gated.conf`, making sure to include the statement:

```
tracefile "/var/tmp/gated.trc" replace;
```

If this is not done, trace information will continue to be appended.

To trace all general information, as well as information about normal protocol occurrences and routing table changes, append the word "general" in the above statement:

```
tracefile "/var/tmp/gated.trc" replace general;
```

5. Configure all of the systems on the 192.168.62 network with the following in mind:

- a. Which version(s) of RIP can be used?
- b. Which systems have to broadcast their routing information via RIP?
- c. Which systems may passively receive RIP packets?
- d. For all systems, define a default route. Why is it necessary to define a default route?

Answer:

- a. Because we did not choose to subnet either networks 192.168.61 or 192.168.62, either RIP version 1 or version 2 can be used. If the networks had been subnetted, only version 2 can be used, since only version 2 supports subnetting.
- b. Only those systems that will be functioning as internal routing gateways (i.e., **blackburn**, **santiago**, and **geneve**) need to broadcast RIP information. All other systems should passively listen to RIP packets, which is the default.

The **/etc/gated.conf** configuration file for non point-to-point gateways should look something like:

```
tracefile "/var/tmp/gated.trc" replace general;
interfaces {
    interface lan0 lan1 passive ;
}
rip yes {
    interface lan0 lan1 version 2 broadcast ;
}
static {
    default gateway ;
}
```

Explanation:

All interfaces should be marked as passive. This prevents **gated** from deleting the interface from the routing table, if it receives no routing information from it.

RIP version 2 should be broadcast over both interfaces.

The static statement defines **<default_gw>** as the default gateway.

- c. Apart from the **blackburn**, **santiago**, and **geneve** systems, all other systems are passive RIP systems.

The **/etc/gated.conf** configuration file should look like:

```
tracefile "/var/tmp/gated.trc" replace general;
rip yes {
    nobroadcast;
    interface lan0 version 2 ;
}
static {
    default gateway <default_gw> ;}
```

Explanation:

RIP version 2 updates are listened for, but not broadcast, from any interface.

Note that <default_gw> should be replaced by the appropriate default gateway for your particular segment. On the 192.168.61.0 net, use 192.168.61.10, on the 192.168.61.0 net, use 192.168.62.10, and on the 156.153.2xx.0 net, use <dallas_default_gw>.

- d. RIP can determine the known internal routes, but it cannot recognize which should be the default route to use to reach external networks. The static statement defines <dallas_default_gw> as the default gateway.

If you use **gated** for dynamic routing on a system, do not use static routing too. This can confuse the routing tables.

6. **blackburn** and **santiago** exchange their information via RIP. What do the interface statements look like for a point-to-point link? Use RIP on the serial link.

Answer:

The **/etc/gated.conf** configuration file non-point-to-point gateways will look something like:

```
tracefile "/var/tmp/gated.trc" replace general;
interfaces {
    interface all passive ;
    define 192.168.61.10 pointopoint 15.17.132.213 ; };
rip yes {
    interface lan0 du0 version 2 broadcast; };
static {
    <remote_ip_address> gateway <local_ip_address> ;
default gateway ; };
```

Explanation:

Often point-to-point links are not opened or configured when the **gated** daemon is started. **define** introduces interfaces that may not be present when **gated** is started, so they can be referenced in the configuration file.

The first address on the **define** statement references the address of the host on the remote end of the interface; the address specified after this **pointopoint** keyword defines the address on the local side of the interface.

7. Check the syntax of your **gated** configuration file, **/etc/gated.conf**, before starting the **gated** daemon.

Answer:

Use the command **gated -c**.

8. Start your **gated** daemon. Watch the routing tables and trace file.

Is everything proceeding correctly? Be patient, as it will take some time before the daemons on the other systems in the room are running and communicating. Dynamic routing takes a little time before all of the routes are propagated.

Answer:

In `/etc/rc.config.d/netconf` change the line reading `GATED=0` to `GATED=1`

Then use `/sbin/init.d/gated start` to start the gated daemon.

To follow the progress of route propagation use `netstat -rn` and `tail -f /var/tmp/gated.trc`. You may also want to check the `/var/adm/syslog/syslog.log` and `/etc/rc.log` files if you reboot the system.

5-11. LAB: NFS Automount Using NIS Maps

1. Define the Necessary User Environment

On the NIS server, define the two users. Suggestion: SAM is a good tool to use for defining the users, but you must adjust the home directories and the `/etc/passwd` file manually.

Remember, although both Juliet and Romeo are configured in the NIS `passwd` map on the NIS server, the physical home directory of Juliet is located on the NIS client, and the physical home directory of Romeo is located on the NIS server.

As the automounter at 11.00 does not look for `auto.master` maps, modify the `/etc/rc.config.d/nfsconf` file:

```
AUTOMOUNT=1
AUTO_MASTER=''/etc/auto_master''
AUTO_OPTIONS=''-f $AUTO_MASTER''
edit /etc/auto_master
+ auto.master
```

Answer:

User 1:

Name: juliet

Password: juliet

UID: 501

GID: default

Homedirectory: /export/balcony/juliet

Homeshell: /usr/bin/sh

User 2:

Name: romeo

Password: romeo

UID: 502

GID: users

Homedirectory: /export/balcony/romeo

Homeshell: /usr/bin/sh

2. Using either `vipw` or `vi`, edit the `/etc/passwd` file to change the home directories to `/balcony/juliet` and `/balcony/romeo`, respectively.

Answer:

The `/etc/passwd` should look like this:

```
juliet:gripf.grapf:501:20:Juliet
Capulet:/balcony/juliet:/usr/bin/sh
```

```
romeo:kripf$krapf:502:20:Romeo  
Montague:/balcony/romeo:/usr/bin/sh
```

3. For Juliet, copy the entire home directory `/export/balcony/juliet` from the NIS server to the NIS client. You can best accomplish this task by using a remote shell and the `tar` command:

Answer:

```
tar cvf - /export/balcony/juliet | remsh nisClient tar xvf -
```

Delete `/export/balcony/juliet` on the NIS server.

4. **Create the Source Files for the NIS Automount Maps**

Configure the automount configuration files in the following way:

On the NIS client, mount `/export/balcony/juliet` at `/balcony/juliet`. On the NIS server, mount `/export/balcony/romeo` at `/balcony/romeo`.

These files will serve as source files only for the NIS maps. The automount daemon should request both maps via NIS.

For configuration files, use the following:

`/etc/auto_master` and `/etc/auto_home`

Assuming that `paris` is the NIS client and `geneve` is the NIS server, what will your configuration files look like?

Answer:

```
/etc/auto_master:  
  
/balcony      auto.home  
  
/etc/auto_home:  
  
juliet        paris:/export/balcony/juliet  
romeo         geneve:/export/balcony/romeo
```

5. Extend the files `/var/yp/Makefile` so that the `/etc/auto_home` file will be built as the `auto.home` NIS map.

Answer:

Use the student notes as a guide.

6. **Configure Each System as an NFS Server**

Both the NIS client and the NIS server function as the NFS server for Juliet's and Romeo's home directories, respectively. On both systems, export the appropriate file systems. Then

activate the NFS server daemons on each system. This is done automatically if you use a tool such as SAM.

Answer:

/export/balcony/juliet on the NIS Client

/export/balcony/romeo on the NIS Server

7. Configure NIS

On each system configure the NIS domain, including both the NIS client and the NIS server.

(If you want to continue with the Shakespearean theme, Montague, Capulet, Benvolio, Mercutio, Tybalt, and Verona are good selections.)

Answer:

NIS server:

```
domainname mydomain  
ypinit -m  
edit /etc/rc.config.d/namesvrs  
for next reboot
```

NIS client:

```
domainname mydomain  
ypbind  
edit /etc/rc.config.d/namesvrs  
for next reboot
```

Both your **auto.master** and **auto.home** map should have been created as a NIS map, as requested in a earlier question.

Check your NIS configuration:

```
ypcat -k auto.master  
ypcat -k auto.home  
ypcat -k passwd
```

Are your maps well defined?

8. Activate the Automount Daemon

On both systems, determine if the automount daemon is running. If it is running, stop it. Do not use the signal 9 to stop the automount daemon!

Restart the automount by using the -v option. You want the automount daemon to read the NIS master map instead of the local /etc/auto_master file.

If you get an error, look at the var/adm/syslog/* file.

What does the -v option do?

Answer:

The -v option appends automount messages to the syslog daemon.

9. Test Your Automount Configuration

On both systems login as Juliet and as Romeo. On the NIS client create a test file to check the correct configuration.

Do you see the test file on the NIS server also?

Are you logged into the correct home directory?

Where are the home directories physically located?

Using the mount command, look at the mount table.

Answer:

You should see the test file you created. If not, make sure you are logged into the correct home directory.

6-24. LAB: Administering NIS+

Setup Root Master Server

1. Delete references to hostC and hostA in /etc/hosts on hostA and hostB, respectively.

Answer:

- Edit the /etc/host file on hostA to NOT contain hostC.
- Edit the /etc/host file on hostB to NOT contain hostA.

2. Set the domainname for the NIS+ domain.

Answer:

```
# domainname class3.hp
```

3. Configure the **PATH** variable to contain the NIS+ directory.

Answer:

```
# PATH=$PATH:/usr/lib/nis
```

4. Initialize the NIS+ root server.

Answer:

```
# cd /nis+files  
# nisserver -r  
type      y    to verify information is correct  
type      y    to continue  
type      hp   to specify login password for root
```

5. Edit startup configuration file.

Answer:

```
# vi /etc/rc.config.d/namesvrs  
NISPLUS_SERVER=1  
NIS_MASTER_SERVER=0  
NISPLUS_CLIENT=1  
NIS_CLIENT=0
```

Populate NIS+ Tables on Master Server

1. Make a temporary directory for the NIS+ files.

Answer:

```
# mkdir /nis+files
```

2. Change directory to **/etc** (location of files to be used for NIS+).

Answer:

```
# cd /etc
```

3. Copy files to the **/nis+files** directory.

Answer:

```
# cp auto_home auto_master group hosts mail/aliases netgroup networks \  
passwd protocols rpc services TIMEZONE ../nis+files
```

4. Prepare the files for conversion to NIS+ tables.

Answer:

- Remove all **UID=0** and system logins (**UID < 100**) from **passwd**
 - Remove all unwanted entries from **hosts**, **aliases**, **passwd**
 - Remove all fully qualified entries from **hosts**
 - Rename all files containing periods (like **auto.master**) to contain underscores (for instance, **auto_master**)
5. Populate the NIS+ tables based on files (-F option) in the **/nis+files** directory.

Answer:

```
# nispopulate -F -p /nis+files
type      y    to verify information is correct
type      y    to continue
```

6. Test access to NIS+ tables.

Answer:

```
# niscat passwd.org_dir.class3.hp
```

7. Perform a checkpoint (-C option) for all (-a option) NIS+ tables.

Answer:

```
# nisping -Ca
```

NOTE: The NIS+ password assigned for each NIS+ principle user is: **nisplus**

Add Administrator to NIS+ admin Group

1. All users that will be allowed to modify the NIS+ tables (and their entries) must be added to the NIS+ **admin** group. Add appropriate users to the **admin** group (assume **user1** is to be added).

Answer:

```
# nisgrpadm -a admin.class3.hp user1.class3.hp
```

Setup a NIS+ Client

1. Verify that the client system is recognized as an authorized client by the server.

Answer:

On the server, type:

```
# nisgrep hostB cred.org_dir.class3.hp
```

2. If the client is not authorized, then arrange for its proper authorization.

Answer:

The following command can be issued to authorize the client.:

```
# nisclient -co hostB
```

3. From the client, set the **domainname**.

Answer:

```
# domainname class3.hp
```

4. Configure the PATH variable to contain the NIS+ directory.

Answer:

```
# PATH=$PATH:/usr/lib/nis
```

5. Initialize the NIS+ client.

Answer:

```
# nisclient -i -h hostA.class3.hp
type      y      to verify information is correct
type      y      to continue
type      x.x.x.x      to specify server's IP address
type      hp      to specify password for client
```

6. Verify client access to NIS+ server.

Answer:

```
# nisl
```

7. Verify client access to NIS+ tables.

Answer:

```
# niscat passwd.org_dir.class3.hp
```

8. Edit client startup configuration file.

Answer:

```
# vi /etc/rc.config.d/namesvrs
NISPLUS_SERVER=0
NIS_MASTER_SERVER=0
```

```
NISPLUS_CLIENT=1  
NIS_CLIENT=0
```

Setup a NIS+ Subdomain

1. On the subdomain system (**hostB**), start the NIS+ RPC daemon.

Answer:

```
# rpc.nisd
```

2. Login to the master server for the parent domain (in this case the **root** domain).

Answer:

```
# telnet hostA
```

3. Initialize the NIS+ subdomain server.

Answer:

```
# nisserver -M -d sub.class3.hp -h hostB  
type y to verify information is correct  
type y to continue
```

Populate Table for the NIS+ Subdomain

1. Exit from master of parent domain (**hostA**) back to the master of the subdomain (**hostB**).

Answer:

```
# exit
```

2. Make a temporary directory for the NIS+ files on the subdomain server (**hostB**).

Answer:

```
# mkdir /nis+files  
# cd /etc
```

3. Copy files to the **/nis+files** directory.

Answer:

```
# cp auto_home auto_master group hosts mail/aliases netgroup networks \  
passwd protocols rpc services TIMEZONE ..../nis+files
```

4. Prepare the files for conversion to NIS+ tables.

Answer:

- Remove all **UID=0** and system logins (**UID < 100**) from **passwd**

- Remove all unwanted entries from `hosts`, `aliases`, `passwd`
 - Remove all fully qualified entries from `hosts`
 - Rename all files containing periods (like `auto.master`) to contain underscores (for instance, `auto_master`)
5. Populate the NIS+ tables based on files (-F option) in the `/nis+files` directory.

Answer:

```
# nispopulate -F -p /nis+files -d sub.class3.hp
type      y    to verify information is correct
type      y    to continue
```

6. Verify subdomain was created successfully.

Answer:

```
# nislsl -l sub.class3.hp
```

7. Verify client access to NIS+ tables.

Answer:

```
# niscat hosts.org_dir.sub.class3.hp
```

8. Edit client startup configuration file.

Answer:

```
# vi /etc/rc.config.d/namesvrs
NISPLUS_SERVER=1
NIS_MASTER_SERVER=0
NISPLUS_CLIENT=1
NIS_CLIENT=0
```

Setup a NIS+ Client for Subdomain

1. Verify that the client system is recognized as an authorized client by the subdomain server.

Answer:

On the subdomain server (`hostB`), type:

```
# nisgrep hostC cred.org_dir.sub.class3.hp
```

2. From the client (`hostC`), set the `domainname` and the `PATH` variable.

Answer:

```
# domainname class3.hp  
# PATH=$PATH:/usr/lib/nis
```

3. Initialize the NIS+ client.

Answer:

```
# nisclient -i -h hostB.sub.class3.hp  
type      y      to verify information is correct  
type      y      to continue  
type      x.x.x.x      to specify server's IP address  
type      hp      to specify password for client
```

4. Verify client access to NIS+ server and to NIS+ tables.

Answer:

```
# nisl  
# niscat passwd.org_dir.class3.hp
```

5. Edit client startup configuration file.

Answer:

```
# vi /etc/rc.config.d/namesvrs  
NISPLUS_SERVER=0  
NIS_MASTER_SERVER=0  
NISPLUS_CLIENT=1  
NIS_CLIENT=0
```

7-11. LAB: Using Network Performance Tools

1. Use **ping** to obtain transfer statistics on 10 packets of 750 bytes to a host on your LAN. Try the same thing by pinging to a host on the other LAN. If still available, also try to **ping** to a host on the other side of a SLIP link.

Answer:

```
ping nexthost 750 19
```

2. Examine the statistics for your network interfaces by using the **netstat -i** command. Run **netstat** with an interval of 2 seconds in one window, while in another window **spray** another host in the classroom, and observe the activity.

Answer:

```
netstat -i -I lan0 2  
  
spray nexthost
```

3. As a group, try to overload your network by selective spraying. Pair up with another pair of students. Start `netstat` with a 2 second interval in one window. Use the `spray` command to transfer 800,000 packets to the other team's system. Observe the effect. Especially look for collisions. When finished, use the `kill` command to terminate your `spray` if necessary.

Answer:

```
netstat -i -I lan0 2  
  
spray nexthost -c 800000
```

4. Determine the overflow of the UDP socket buffer.

Answer:

```
netstat -s -p udp
```

5. Determine the default buffer size for TCP sockets. Does increasing the TCP socket buffer also increase the network performance?

Answer:

```
ndd -get /dev/tcp tcp_recv_hiwater_dev  
ndd -get /dev/tcp tcp_xmit_hiwater_dev
```

7-22. LAB: Optimizing NFS Performance

1. Local copy: On the NFS client create a shellscript `perf` and change the mode of `perf` to executable.

In the script, measure the local copying on the client by copying `/local/mnt/bigfile` to `/remote/mnt/bigfile`. Time the execution of `perf`. Finally, remove the file `/remote/mnt/bigfile`.

Answer:

```
>cat perf  
cp /local/mnt/bigfile /remote/mnt/bigfile  
  
>sync
```

```
>sync ; time ./perf  
>rm /remote/mnt/bigfile
```

2. Theoretical transfer rate: Calculate the minimum time used for transferring the **/local/mnt/bigfile** via your network.

Next, reduce this value with about 6 percent for protocol headers. What do you get for the UNIX kernel with a size of 8.8 MB?

Answer:

```
10 (MBit/sec) / 8 (Bit/Byte) = 1.25 MB/sec
```

```
8.8 MB / ( 1.25 * 0.94 (MB/sec) ) = 7.5 sec
```

3. Remote copy with FTP: Using **ftp**, copy the local file from the client to the server. First, choose the ascii mode for transferring the file, then switch to the binary mode. Again take each measurement at least twice.

Answer:

```
ftp> ascii  
ftp> put /local/mnt/bigfile /local/mnt/bigfile  
  
ftp> binary  
ftp> put /local/mnt/bigfile /local/mnt/bigfile
```

4. Synchronous writing: Varying the Number of **biod** and **nfsd-daemon**

Configure your NFS environment. On the NFS server, export the directory **/local/mnt** with root access for the NFS client. Start the NFS servers using the following criteria:

First start only 1 **nfsd**:

With **ps** determine the number of **nfsds** that are running. Kill all **nfsds** and restart the **nfsd** daemon:

Don't forget to start the **mountd**, if it is not running. (Of course, for configuring NFS you can use a tool such as SAM.)

In the same way as you did on the server kill all **biod** on the client, and restart only one **biod** daemon:

On the NFS client, mount the **/local/mnt** directory from the server at the mount point **/remote/mnt**

Measure the file transfer time at least twice:

Now, vary the number of **biods** and **nfsds**:

Answer:

```
nfsd 1  
  
kill `ps -efa | grep nfsd | cut -c10-15'  
nfsd 1  
mountd  
exportfs -i -o root=nfsclient /local/mnt  
  
biод 1  
  
kill `ps -efa | grep biод | cut -c10-15'  
biод 1  
mount nfsserver:/local/mnt /remote/mnt  
  
sync ; time ./perf  
  
4b) 1 biод - 4 nfsd  
4c) 4 biод - 4 nfsd  
4d) 6 biод - 6 nfsd  
4e) 8 biод - 8 nfsd
```

5. Asynchronous writing: Reduce the number of **nfsd** and **biод** daemons to 4.

On the NFS server re-export the filesystem with asynchronous writing. How much does the performance increase now?

Answer:

```
exportfs -u /local/mnt  
  
exportfs -i -o async,root=nfsclient /local/mnt
```

6. Attribute caching: On the NFS server, export the **/usr** filesystem as read-only:

On the NFS client, mount **/usr** directory from the server to **/rmnt** as read-only, too. Zero the NFS statistic on the client:

Measure the amount of time the **find** command uses in stepping through the **/rmnt** subdirectories.

Look at the NFS statistics. Write the **nfsstat** output to **/tmp/cache.out**. Again, zero out the NFS statistics. Then unmout **/rmnt** and remount it, but this time do not use attribute caching. How long does it take the **find** command to go through the **/rmnt** subdirectories this time? Write the **nfsstat** output to **/tmp/nocache.out**. Compare the cache output with the nocache output.

Answer:

```
exportfs -i -o /usr  
  
mkdir /rmnt  
mount -o ro nfsserver:/usr /rmnt  
nfsstat -z  
  
time find /rmnt -type f -print > /dev/null  
  
umount /rmnt  
mount -o ro,noac nfsserver:/usr /rmnt  
nfsstat -z
```

Example of output with default attribute caching:

```
real      1:04.2  
  
user       1.5  
  
sys       13.1  
  
nfsstat -c  
  
Client rpc:  
  
calls      badcalls   retrans   badxid    timeout   wait  
newcred  
  
21808      0          0          0          0          0  
0  
  
Client nfs:  
  
calls      badcalls   nclget    nclsleep  
  
21808      0          21808     0  
  
null       getattr    setattr   root      lookup    readlink  
read  
  
0  0%      947  4%    0  0%      0  0%      18944 86%  0  0%  
0  0%  
  
wrcache    write      create    remove    rename    link  
symlink  
  
0  0%      0  0%      0  0%      0  0%      0  0%      0  0%
```

	mkdir	rmdir	readdir	statfs
0 0%	0 0%	1917 8%	0 0%	

Example of output without attribute caching:

```

real      2:13.2
user      1.4
sys       19.8
nfsstat -c

Client rpc:
calls      badcalls   retrans   badxid    timeout   wait
newcred

49820      0          0          0          0          0

Client nfs:
calls      badcalls   nclget    ncleep
49820      0          49820     0

null       getattr    setattr   root      lookup    readlink
read      0 0%        23669 47%  0 0%      0 0%      24233 48%
0 0%        0 0%      0 0%      0 0%      0 0%      0 0%

wrcache   write      create    remove   rename    link
symlink

0 0%      0 0%      0 0%      0 0%      0 0%      0 0%
0 0%      0 0%      0 0%      0 0%      0 0%      0 0%

mkdir      rmdir      readdir  statfs
0 0%      0 0%      1917 3%  1 0%

```

7. Calculating NFS timeouts: Run the `nfsstat -m` command and try to identify the best and worst server response time. Use the results obtained above to fill in the following table:

	num biod	num nfsd	test1	test2	transfer rate
1)	local copy	-	-		
2)	theorectical rate				
3a)	ftp, ascii mode	-	-		
3b)	ftp, binary mode	-	-		

4a)	NFS, write	1	1				
4b)	NFS, write	1	4				
4c)	NFS, write	4	4				
4d)	NFS, write	6	6				
4e)	NFS, write	8	8				
-	NFS, read	4	4	(8.7)	(8.6)		(1050 KB/sec)
5	NFS, write, async	4	4				

Answer:

		num	biiod	num	nfsd	test1	(sec)	test2	(sec)	tranfer	rate
1)	local copy	-	-	2.7	2.6	2.8				3260	KB/sec
2)	theoretical rate	-	-	7.5	-	-				1200	KB/sec
3a)	ftp, ascii mode	-	-	13.6	14.9	13.7				620	KB/sec
3b)	ftp, binary mode	-	-	7.8	7.8	7.9				1150	KB/sec
4a)	NFS, write	1	1	69.1	62.6	61.5				140	KB/sec
4b)	NFS, write	1	4	67.5	63.6	64.5				140	KB/sec
4c)	NFS, write	4	4	17.5	18.4	17.1				500	KB/sec
4d)	NFS, write	6	6	16.8	16.7	17.3				530	KB/sec
4e)	NFS, write	8	8	17.9	17.6	17.8				500	KB/sec
-	NFS, read	4	4	8.7	8.6	8.5				1050	KB/sec
5)	NFS, write, async	4	4	8.6	8.2	8.7				1060	KB/sec

8-13. LAB: DHCP Configuration

- For one of the DHCP clients use a fixed, pre-assigned IP address. Determine the hardware address of this client.

Answer:

On the client call:

```
lanconfig lan0
```

2. Use SAM for configuring this client on your boot server. What information do you need to provide?

Answer:

Host Name:
Internet Address:
Subnet Mask:
Host Identification Method:
Station Address:
Adapter Device Type:
Default Router:
Static Routes:
Broadcast Address:
Send Hostname to Device:

3. Which configuration file has SAM updated ?

Answer:

/etc/bootptab

4. Look at the newly created entry in the configuration file. What are the tags for the following?

Host Name:
Internet Address:
Subnet Mask:
Host Identification Method:
Station Address:
Adapter Device Type:

Default Router:
Static Router:
Broadcast Address:
Send Hostname to Device:

Answer:

Host Name:	dallas is the name of the item.
Internet Address:	ip
Subnet Mask:	sm
Host Identification Method:	no tag, entry is added to bootptab
Station Address:	ha
Adapter Device Type:	ht
Default Router:	gw
Static Router:	sr
Broadcast Address:	ba
Send Hostname to Device:	hn

Notice that SAM has added the following entry to the configuration file /etc/bootptab:

```
dallas:\n    ht=ethernet:\\\n    ha=080009123a4b:\\\n    ip=150.150.192.9:\\\n    hn:\\\n    sm=255.255.255.0:\\\n    gw=150.150.192.5:\\\n    sr=150.150.128.0 150.150.192.10:\\\n    ba=150.150.192.255
```

5. For the remaining clients configure a DHCP pool. Again use SAM to fill in information below:

Group Name:
Subnet Address:
Subnet Mask:
Subnet Address Pool:
Lease Time:

Default Router:
Static Router:
Broadcast Address:
Send Hostname to Device:

Renewal Time T1:
Rebind Time T2:

Answer:

Group Name:	america
Subnet Address:	150.150.192.0
Subnet Mask:	255.255.255.0
Subnet Address Pool:	150.150.192.20 - 150.150.192.30
Lease Time:	30 min
Default Router:	150.150.192.5
Static Router:	150.150.128.0 150.150.192.10
Broadcast Address:	150.150.192.255
Send Hostname to Device:	yes
Renewal Time T1:	60 percent
Rebind Time T2:	95 percent

6. Which configuration file has SAM updated this time?

Answer:

/etc/dhcptab

7. Look at the newly created entry in the configuration file. What are the tags for in the following?

Group Name:
Subnet Address:
Subnet Mask:
Subnet Address Pool:
Lease Time:

Default Router
Static Router:
Broadcast Address:
Send Hostname to Device:

Renewal Time T1:
Rebind Time T2:

Answer:

Group Name:	pool-name
Subnet Address:	only necessary for generating the IP address pool
Subnet Mask:	subnet-mask
Subnet Address Pool:	addr-pool-start-address, addr-pool-last-address
Lease Time:	lease-time
Default Router:	gw
Static Router:	sr
Broadcast Address:	ba
Send Hostname to Device:	hn
Renewal Time T1:	tr
Rebind Time T2:	tv

Notice that SAM has added the entry to the configuration file **/etc/dhcptab**:

```
dhcp_pool_group:\n    pool-name=america:\n        addr-pool-start-address=150.150.192.20:\n        addr-pool-last-address=150.150.192.30:\n        lease-time=1800:\n        allow-bootp-clients=FALSE:\n        hn:\n        subnet-mask=255.255.255.0:\n        gw=150.150.192.5:\n        sr=150.150.128.0 150.150.192.10 :\n        ba=150.150.192.255:\n        tr=60:\n        tv=95
```

8. Using dhcptools dynamically, create new hostnames in the form of

`clienta, clientb, clientc ...`

In which file are the names created?

Append this file to your existing **/etc/hosts** file.

Answer:

```
dhcptools -h fip=150.150.192.20 no=11 sm=255.255.255.0 "hn=client?"  
/tmp/dhcphosts  
cat /tmp/dhcphosts >> /etc/hosts
```

9. Using SAM, enable your **bootpd daemon.**

Answer:

```
Networking and Communications  
    Bootable Devices  
        Fixed-Address Devices  
            Booting from This Server  
                Actions  
                    Enable Boot Server
```

10. Using **dhcptools, preview a client's address assignment**

(a) for the fixed IP assignment (b) for any other hardware address.

Answer:

(a) for the fixed IP assignment

```
dhcptools -p ht=ether ha=080009123a4b sn=150.150.192.0
```

Response Data:

```
hardware type = ethernet  
harware address length = 6  
hardware address = 080009123A4B  
IP address = 150.150.192.9  
lease time = INFINITE  
subnet mask = 255.255.255.0  
bootfile =  
hostname = dallas  
number of DNS servers = 0  
DNS servers =  
number of NIS servers = 0  
NIS servers =  
number of routers = 1  
routers = 150.150.192.5  
time offset = 0  
number of X font servers = 0  
X font servers =  
number of X display servers = 0  
X display servers =
```

(b) for any other hardware address

```
dhcptools -p ht=ether ha=0800091a2b3c sn=150.150.192.0
```

generates:

```
Response Data:  
    hardware type = ethernet  
    hardware address length = 6  
    hardware address = 0800091A2B3C  
    IP address = 150.150.192.20  
    lease time = 1800 seconds  
    subnet mask = 255.255.255.0  
    bootfile =  
    hostname =  
    number of DNS servers = 0  
    DNS servers =  
    number of NIS servers = 0  
    NIS servers =  
    number of routers = 1  
    routers = 150.150.192.5  
    time offset = 0  
    number of X font servers = 0  
    X font servers =  
    number of X display servers = 0  
    X display servers =
```

11. Start a trace for boottpd.

Answer:

```
dhcptools -t ct=100
```

12. On the client enable DHCP and reboot the client.

Answer:

```
vi /etc/rc.config.d/netconf
```

```
DHCP_ENABLE=1
```

```
reboot
```

13. Does your configuration work correctly?

Stop the trace on the DHCP server.

In which file do you find the traced packets?

Do you understand the output?

Answer:

```
dhcptools -t ct=0  
/tmp/dhcptrace
```

14. On the DHCP client, determine the values that the client has gotten from the DHCP server.

Be sure to check the appropriate variables in the configuration files. Are they all correct?

Answer:

```
dhcpdb2conf -p
```

9-22. LAB: Configuring and Maintaining the DNS

1. Copy your `/etc/hosts` file to `/etc/hosts.old`. From the `/etc/hosts` file remove all entries except

- your own hostname
- loopback or local host entry
- all hostnames within your domain

2. Create a directory `/etc/nameserver`.

Answer:

```
mkdir /etc/nameserver
```

3. Use the command `hosts_to_named` to create the appropriate BIND configuration files in the `/etc/nameserver` directory.

Answer:

```
# vi /etc/hosts
```

4. Change to `/etc/nameserver` and create the DNS data file with the command `hosts_to_named`.

Answer:

```
# hosts_to_named -d domain -n network_number -n -b /etc/named.boot
```

Remember, the reverse name resolution created by the option `-n` is only used by the root name server in our lab.

5. `hosts_to_named` creates entries only for a flat domain space. Edit the configuration files and add, delete or change the appropriate entries under following considerations:

- which one is the root name server
- which domain is your name server responsible for
- which one is the name server for the reverse name resolution

- does your name server have to know other subdomains
- which are the hostnames returned by the reverse name resolution
- check your serial number

What do the following files look like?

- loopback data file
- cache file
- boot file
- name resolution data file
- reverse name resolution data file

Answer:

The **loopback** file.

This file is nearly the same on all name servers. As an example, we look at the **loopback** file of name server **hamburg**.

```
/etc/nameserver/db.127.0.0:

@ IN SOA      hamburg.eurasia. root.hamburg.eurasia. (
                      2      ; Serial
                      10800   ; Refresh every 3 hours
                      3600    ; Retry every hour
                      604800  ; Expire after a week
                      86400 ) ; Minimum ttl of 1 day

IN NS      hamburg.eurasia.

1       IN PTR      localhost.
```

Answer:

The cache file:

The cache file on all name servers is the same except on the root name server. The root name server, of course, has no cache file.

The system generates a template for the file, containing comments only. An example of a *filled-out* file follows:

/etc/nameserver/db.cache:

```
.          99999999  IN  NS  hamburg.eurasia.  
hamburg.eurasia. 99999999  IN  A  150.150.128.12
```

Answer:

Special configuration files on the root name server **hamburg**:

/etc/named.boot:

```
directory /etc/nameserver  
primary 0.0.127.IN-ADDR.ARPA db.127.0.0  
primary . db.root primary  
150.IN-ADDR.ARPA db.150
```

/etc/nameserver/db.root:

```
@  IN  SOA  hamburg.eurasia. root.hamburg.eurasia. (  
          2           ; Serial  
          10800        ; Refresh every 3 hours  
          3600         ; Retry every hour  
          604800       ; Expire after a week  
          86400 )      ; Minimum ttl of 1 day  
  
IN  NS  hamburg.eurasia.  
localhost  IN  A  127.0.0.1  
loopback    IN  CNAME  localhost  
america     IN  NS  dallas.america  
eurasia     IN  NS  paris.eurasia  
dallas.america  IN  A  150.150.192.9  
paris.eurasia  IN  A  150.150.128.11  
hamburg.eurasia.  IN  A  150.150.128.12
```

Answer:

data file for reverse name resolution:

/etc/nameserver/db.150:

```
@ IN SOA hamburg.eurasia. root.hamburg.eurasia. (
                2      ; Serial
                10800   ; Refresh every 3 hours
                3600    ; Retry every hour
                604800   ; Expire after a week
                86400 )  ; Minimum ttl of 1 day

IN NS hamburg.eurasia.

3.240.240      IN      PTR      singapore.eurasia.
4.240.240      IN      PTR      blackburn.eurasia.
                  IN      PTR      blackburneth.eurasia.
4.224.224      IN      PTR      blackburn.eurasia.
                  IN      PTR      blackburnsl.eurasia.
4.192.150      IN      PTR      santiago.south.america.
                  IN      PTR      santiagoeth.south.america.
5.224.224      IN      PTR      santiago.south.america.
                  IN      PTR      santiagosl.south.america
6.192.150      IN      PTR      saopaulo.south.america
7.192.150      IN      PTR      caracas.south.america
8.192.150      IN      PTR      paloalto.north.america.
9.192.150      IN      PTR      dallas.north.america.
10.192.150     IN      PTR      geneve.eurasia.
                 IN      PTR      geneveeth.eurasia.
10.128.150     IN      PTR      geneve.eurasia.
                 IN      PTR      genevebnc.eurasia.
11.128.150     IN      PTR      paris.eurasia.
12.128.150     IN      PTR      hamburg.eurasia.
```

```
14.128.150      IN      PTR      pinewood.eurasia.
```

Answer:

configuration files for domain america on the name server dallas:

/etc/named.boot:

```
directory      /etc/nameserver
primary        0.0.127.in-addr.arpa      db.127.0.0
primary        america                  db.america
cache          .                      db.cache
```

/etc/nameserver/db.america:

```
@  IN  SOA   dallas.north.america. root.dallas.north.america.  (
                                2      ; Serial
                                10800    ; Refresh every 3 hours
                                3600     ; Retry every hour
                                604800   ; Expire after a week
                                86400 )  ; Minimum ttl of 1 day
IN  NS   dallas.north.america.

localhost      IN      A      127.0.0.1
north          IN      NS     paloalto
south          IN      NS     caracas
paloalto       IN      A      150.150.192.8
caracas        IN      A      150.150.192.7
; in order to avoid dependencies
; append the own hostname to the name resolution
dallas.north.america. IN      A      150.150.192.9
```

Answer:

configuration files for domain america on the name server dallas:

/etc/named.boot:

```
directory          /etc/nameserver

primary           0.0.127.in-addr.arpa      db.127.0.0

primary           north.america        db.north.america

cache             .                  db.cache
```

/etc/nameserver/db.north.america:

```
@  IN  SOA    paloalto.north.america. root.paloalto.north.america. (
                                2      ; Serial
                                10800   ; Refresh every 3 hours
                                3600    ; Retry every hour
                                604800  ; Expire after a week
                                86400 ) ; Minimum ttl of 1 day

IN  NS    paloalto.north.america.

localhost        IN     A      127.0.0.1
paloalto         IN     A      150.150.192.8
dallas            IN     A      150.150.192.9
```

Answer:

configuration files for south.america on the name server caracas:

/etc/named.boot:

```
directory          /etc/nameserver

primary           0.0.127.in-addr.arpa      db.127.0.0

primary           south.america        db.south.america

cache             .                  db.cache
```

/etc/nameserver/db.south.america:

```
@  IN  SOA    caracas.south.america. root.caracas.south.america. (
                                2      ; Serial
                                10800   ; Refresh every 3 hours
                                3600    ; Retry every hour
```

```

                                604800      ; Expire after a week
                                86400 )      ; Minimum ttl of 1 day

IN      NS      caracas.south.america.

localhost          IN      A      127.0.0.1
santiago          IN      A      150.150.192.5
                           IN      A      150.224.224.5
santiagoeth        IN      CNAME  santiago
; for exact routing to the serial line interface
; define an extra entry
santiagosl         IN      A      150.224.224.5
saopaulo          IN      A      150.150.192.6
caracas            IN      A      150.150.192.7

```

Answer:

configuration files for domain eurasia on the name server paris:

/etc/named.boot:

```

directory          /etc/nameserver
primary           0.0.127.in-addr.arpa      db.127.0.0
primary           eurasia                  db.eurasia
cache              .                      db.cache

```

/etc/nameserver/db.eurasia:

```

@    IN      SOA      paris.eurasia. root.paris.eurasia. (
                                2      ; Serial
                                10800     ; Refresh every 3 hours
                                3600      ; Retry every hour
                                604800    ; Expire after a week
                                86400 )   ; Minimum ttl of 1 day

```

```

IN   NS      paris.eurasia.

localhost          IN   A     127.0.0.1

singapore         IN   A     150.240.240.3

blackburn          IN   A     150.240.240.4

                           IN   A     150.224.224.4

blackburneth       IN   CNAME blackburn

; for exact routing to the serial line interface

; define an extra entry

blackburnsl        IN   A     150.224.224.4

geneve             IN   A     150.150.192.10

                           IN   A     150.150.128.10

geneveeth          IN   CNAME geneve

genevebnc          IN   A     150.150.128.10

paris              IN   A     150.150.128.11

hamburg            IN   A     150.150.128.12

pinewood           IN   A     150.150.128.14

```

Note: MX records are not necessary for name resolution. Therefore, they are not printed here.

6. Create the necessary client configuration file. The client configuration file is not only necessary on clients but also on each name server.

Answer:

```

# vi /etc/resolv.conf

domainname eurasia

nameserver 150.150.128.11

```

7. Check the name service switch configuration that your system is currently using. Your host should use the DNS name server. If an entry is not found, use the local /etc/hosts file. NIS should not be used for this lab.

Answer:

```
# vi /etc/nsswitch.conf:  
  
hosts: dns [NOTFOUND=continue] files
```

8. Start your DNS name server and check the classroom DNS domain space.

To run **named** at next system startup, configure the appropriate configuration files.

Answer:

```
/usr/sbin/named
```

9. Use **nslookup** to verify your system's IP address. You may wish to look up other system names both inside and outside of your domain.

Next, query other types like PTR, SOA or NS entries.

Answer:

```
# nslookup  
  
> set querytyp=PTR  
  
> 150.150.192.9  
  
> set querytype=SOA  
  
> america  
  
> set querytype=NS  
  
> south
```

10. Choose some systems as secondary name servers. Which files do you have to edit ?

In our example we choose **pinewood** as a secondary name server for **eurasia**.

Answer:

```
# vi /etc/named.boot:  
  
directory      /etc/nameserver  
  
primary        0.0.127.in-addr.arpa      db.127.0.0  
  
secondary       eurasia      150.150.128.11    db.eurasia  
  
cache          .           db.cache  
  
# vi /etc/nameserver/db.cache  
  
# vi /etc/nameserver/db.127.0.0.
```

```
# /usr/sbin/named
```

11. Which files are created automatically by **named**? Take a look at these files by using the **more** command.

Answer:

named requests the primary name server 150.150.128.11 (**paris**) for the data file for the domain **eurasia** and writes the data to **db.eurasia**. Don't create this file. It is created by **named**.

```
# more /etc/nameserver/db.eurasia
```

12. On the primary name server, add a dummy entry to the data file and change the serial number.

Send the appropriate signal to re-read the configuration file.

Has the serial number changed on the secondary name server, too?

Look at the created data file.

Answer:

```
sig_named restart      # Re-reads the database on HP
kill -SIGHUP `cat /var/run/named.pid`      # or on every system
```

To change the new entry on the secondary name server immediately, one has to send a signal to the secondary **named** also.

13. Send the appropriate signal to your local **named** process to have it dump its current database. Compare the database dump to the zone files in **/etc/nameserver** and note any differences.

Answer:

```
sig_named dump          # Dumps the database on HP
kill -INT `cat /var/run/named.pid`          # or on every system
more /usr/tmp/named_dump.db
```

The location of the file **named.pid** depends on the system considered. For HP-UX, **/var/run** is used.

There shouldn't be any extra or missing entries in the **named_dump.db** file when compared with the zone files. The database dump file format is an SSR format but the order of the information is often re-arranged from the source zone files, and the database dump may contain many origin lines which were not in the source files. A careful look is necessary to verify that the address and pointer record information is correct.

10-31. LAB: Configuring a Mail Hub

1. On the DNS primary name servers, set up DNS MX records for these domains:
 1. *eurasia* and
 2. *south.america*

Answer:

On **paris**—the name server for **eurasia**—modify the **db.eurasia** file:

```
paris# vi /etc/named.data/db_eurasia
update the version number then add:
eurasia      IN  MX  5  eurasia

eurasia      IN  A   150.150.128.11
```

Accordingly change the name server **caracas** for the domain **south.america**.

2. Don't forget to let **named** re-read the configuration files.

Answer:

Best solution for re-reading the DNS database would be to restart the **named** or otherwise send the SIGHUP signal to it.

```
ps -ef | grep named
kill -HUP named-pid
```

3. Announce all alternate names for which the mail hub shall accept mails.

Answer:

On the mail hub **geneve** and **santiago** respectively edit the **sendmail.cw** file:

```
geneve# vi /etc/mail/sendmail.cw
geneve
localhost
geneve eurasia
eurasia
```

Check the **sendmail.cf** file:

```
Fw-o /etc/mail/sendmail.cf
```

This entry should be uncommented.

Alternatively you can define the **Cw** class.

4. Setup mail aliases for all users in your domain in the **/etc/mail/aliases** database. Don't forget to hash your database.

Answer:

```
geneve# vi /etc/mail/aliases
```

```
mac3 : mac9@singapore
```

```
mac4 : mac9@blackburn
```

```
mac10 : mac7@geneve
```

```
mac11 : mac3@paris
```

```
mac12 : mac9@hamburg
```

```
mac14 : mac9@pinewood
```

```
:wq
```

```
geneve# newaliases
```

5. Implement Site Hiding on your mail hub.

Answer:

See "Site Hiding" on the internal mail hosts below.

6. Configure all mail hosts in the domain **eurasia``** and **``south.america** to use Site Hiding.

Answer:

In **sendmail.cf** configuration file set the **masquerade** macro and change ruleset 94.

```
DMeurasia
```

```
#####
### Ruleset 94 -- convert envelope names to masqueraded form #####
#####
```

```
S94
R$+           $@ $>93 $1
R$* < @ *LOCAL* > $*   $: $1 < @ $j . > $2
```

7. Define the **geneve** as a *smart* mail hub for non-local mail.

Answer:

DSgeneve.eurasia.

8. Restart the `sendmail` daemon.

Answer:

```
myhost# /sbin/init.d/sendmail stop  
myhost# /sbin/init.d/sendmail start
```

Optionally configure one or several hosts of

`saopaulo, pinewood, or singapore`

for using `/etc/hosts`-based name resolution only.

9. Mail to such a host will be delivered with the fully qualified domain name.

For this define alternate names in the `Cw` class.

Answer:

```
Cwlocalhost pinewood pinewood.eurasia.
```

10. In order to communicate with the other DNS resolving hosts, define a fully qualified domain name.

Answer:

```
Dj$w.pseudo.domain
```

11. Try sending mail to the following:

```
myname  
myname@myhost  
myname@myhost.mydomain  
myname@mydomain  
othername@otherdomain  
worldname@worlddomain
```

When the mail message comes back to you, read the header lines from the bottom up carefully to ensure you understand the routing.

Answer:

Any questions?