Sun Solaris 8 Certified Network Administration

Version 3.0.0

Sun Solaris 8 Certified Network Administration

Version 3.0.0

Abstract:

This study guide will help you to prepare for Sun exam 311-043, Solaris 8 Network Administration. Exam topics include LANs, Ethernet interface, ARP & RARP, Internet layer, routing, client-server, DHCP, NTP, IPv6 and troubleshooting.

Find even more help here:

Contents:

Network Model 5
Identify the purpose of each layer in the TCP / IP five layer Model 5
Describe the functionality of the following Network Protocols: TCP, UDP, IP, and ICMP 5
Describe the relationship between the following Network Protocols: TCP, UDP, IP, and ICMP 5
Describe peer-to-peer communication 6
Local Area Networks 6
Identify the role of the following LAN components: Bridge, Repeater, Router, Switch, and Gateway 6
Identify the network topologies 6
Ethernet Interface 7
State the purpose of the Ethernet address 7
Identify the commands to get and set driver configuration 7
ARP and RARP 7
Explain the process of address resolution using ARP and RARP 7
Identify the commands to manage ARP cache 8
Identify the configuration files and scripts used to configure a network interface 8
Internet Layer 8
Describe the following: IP address, Broadcast address, Netmask, Datagram, and Fragment 8
Describe Classless Inter-Domain Routing (CIDR) 9
Identify the file used to set netmasks 10
Identify the features and benefits of the Variable Length Subnet Masks (VLSM) 11
Configure a Network Interface 12
Routing 12
Describe IP routing 12
Identify the Solaris 8 daemons which implement routing protocols 13
Identify the files used to configure routing 13

Identify the purpose of the files used to configure routing 13
Administer the routing table.....14
Transport Layer 14
Identify the features of the TCP and UDP 14
Define the terms connection oriented, connection-less, stateful, and stateless 15
Describe the relationships between port numbers, network services and inetd 15
Client-Server Model.....15
Explain the terms client, server, and service 15
Administer Internet services and RPC services.....15
Collect information about services configured on hosts 17
DHCP 18
State the benefits of DHCP 18
Identify DHCP configuration files 18
State the purpose of DHCP configuration files.....19
Administer DHCP clients and servers 20
Network Management Tools.....20
Identify tools which use the Simple Network Management Protocol (SNMP) 20
Describe the Simple Network Management Protocol (SNMP) 20
Domain Name System 21
Identify the purpose of DNS.....21
Describe address resolution and reverse address resolution.....21
Identify the correct Resource Record syntax.....22
Explain the steps needed to configure DNS 23
Identify the configuration files for DNS 23
State the purpose of DNS configuration files.....24
Network Time Protocol.....24
Describe the NTP features 24
Identify NTP configuration files.....25
State the purpose of NTP files 26

Describe how to configure NTP 26
Network Troubleshooting.....26
Identify common network problems 26
Diagnose network problems.....27
Resolve network problems.....27
IPv6.....28
Describe IPv6 28
Configure an IPv6 network interface 29

Network Model

Identify the purpose of each layer in the TCP / IP five layer Model

Application Layer
User accessed application programs and network services. Examples at this layer would include FTP, telnet, and SMTP, HTTP, IMAP, NFS, NIS, POP etc.
Transport Layer
Connection-oriented TCP and connectionless UDP data transfer occur here. TCP offers flow control, sequencing, and acknowledgements at this layer. UDP is an "unguaranteed", packet based service.
Internet Layer
Here data is fragmented (if necessary) addressed and routed. Encapsulation of data into IP datagrams, happens at this layer, the IP header containing enough information to enable the datagrams to be routed.
Network Interface Layer
Handles error detection and packet framing. Standards like 802.3 and 802.5 arrange bits into understandable data using fields that describe destination and source addresses, data, and error correction.
Hardware Layer
Electrical signals that move raw bits through the ether. Today, twisted pair and fiber optic cabling is the most common transmission medium for network data.

Describe the functionality of the following Network Protocols: TCP, UDP, IP, and ICMP

TCP – Connection-oriented. The hallmark of TCP is the acknowledgements between systems that ensure all of the sent data was received. Can operate two-way (full-duplex)
UDP – Connectionless. UDP packets traverse the network by themselves when they leave the sender. The receiver and sender do not communicate about the status of UDP packets. UDP is faster and easier to implement than TCP.
IP – Determines a packet's path based on the listed destination IP.
ICMP – Communicates system status and error messages using IP datagrams.

Describe the relationship between the following Network Protocols: TCP, UDP, IP, and ICMP

TCP (transport layer) is a protocol that offers flow control, positive acknowledgement with retransmission and guaranteed delivery, and should be used if data transmission must be guaranteed.
UDP (transport layer) does not offer guaranteed delivery. It is simply a datagram delivery service. It is packet based, and delivery is not guaranteed, although it does have an optional checksum, which is usually off, by default.

IP (internet layer) is the protocol that determines the best route for IP datagrams traveling through networks en route to a destination. IP does not build the routing tables (although ICMP redirects may cause routing table changes), but does make the routing decision, relying on kernel, routing table information.
ICMP (internet layer) is a protocol that assists IP with error detection and correction. Notably, the ping command uses ICMP.

Describe peer-to-peer communication

Peer-to-peer is the term used to describe the communication between corresponding network "protocol stack" layers on source and destination hosts. At each layer, data is encapsulated or de-encapsulated and passed on to the next layer. From the source, data is encapsulated and passed "down the stack". At the destination (receiving end), data is de-encapsulated (headers removed) and passed up the stack.

Local Area Networks

Identify the role of the following LAN components: Bridge, Repeater, Router, Switch, and Gateway

Bridge – A link layer device that connects two or more network segments.
Repeater – A simple device that regenerates the signal and prevented signal attenuation.
Router – A device that "routes" packets by examining addresses and selecting optimal paths.
Switch – A link layer device that dedicates traffic between two devices.
Gateway – Interconnects two networks using different protocols.

Identify the network topologies

Bus – One central device, usually a hub, which connects cabling between other machines. The signals they send are shared with every node on the network.
Star – A central hub with a segment of cable running between it and each client and other network hubs. The advantage is that there are never more than 2 hops between any two nodes.
Ring – Each node's output is connected 'serially' to the next node's input. A 'token' must be passed around to allow each device to communicate. Intelligent systems can increase reliability over that of a bus or star configuration.

Ethernet Interface

State the purpose of the Ethernet address

Ethernet is the most popular LAN technology. It consists of units of data (frames/packets) traversing physical cabling and circuitry regulated by a flow-control protocol called CSMA/CD (Carrier Sense Multiple Access with Collision Detect).
An Ethernet address is a 48-bit unique identifier for each network interface or device. The standard is administered by the IEEE and designates the first three octets (in bold) as vendor-specific. An example: **08:00:20:0f:0f:02**, the 48-bit representation is **00001000:00000000:00010100: 00001111:00001111:00000020**

Identify the commands to get and set driver configuration

Getting and setting driver configuration is accomplished with the `ndd` command.

```
# ndd /dev/hme \?
transceiver_inuse       (read only)
link_status             (read only)
link_speed              (read only)
<...>
```

shows the status of features of the adaptor `hme`.

```
# ndd /dev/hme link_status
1
```

queries the parameter link_status and outputs the value 1 (meaning the link is up)

```
# ndd /dev/hme adv_100fdx_cap 1
```

sets the `/dev/hme` adaptors to 100 megabit full duplex.

ARP and RARP

Explain the process of address resolution using ARP and RARP

ARP and RARP (Reverse ARP) are protocols that create a "mapping" between a host's unique MAC address (Ethernet or Physical address) with an "administrator" assigned IP address (known as software or logical address).
ARP maps a 32-bit IP to a 48-bit MAC (or Ethernet) address. Useful when we have an IP address and we need the MAC address of the system that is using the IP address. An ARP broadcast (ARP request) contains the IP address of a "target" system, which, on receiving the broadcast traffic, and "recognizing" its IP address in the ARP request, will return its (the target) MAC address. The ARP reply is "unicast" and directed at the system that generated the ARP request.
A required MAC address will be the MAC address of systems on the local network we may be using to reach distant networks; i.e., local routers.
RARP maps a 48-bit MAC address with a 32-bit IP address.

A RARP broadcast is usually generated by a system, which does not know its own IP address and is attempting to "learn" it from a boot (RARP) server, "booting" Examples of devices that generate RARP requests include diskless clients, jumpstart clients, X terminals, and network printers.

Identify the commands to manage ARP cache

ARP replies received by a host are stored temporarily in cache so that the resolution process does not need to be repeated as frequently.
The ARP cache holds mappings, which may be seen using:
arp –p
or
netstat –p

```
Net to Media Table: IPv4
Device   IP Address               Mask      Flags   Phys Addr
------ -------------------- --------------- ----- ---------------
hme0   moonbug.gvon.co.uk   255.255.255.255       00:00:86:56:c0:83
hme0   phobos.gvon.co.uk    255.255.255.255       00:02:e3:0b:25:26
hme0   tgt.gvon.co.uk       255.255.255.255       08:00:20:a4:e6:82
…….
Lots omitted for the sake of brevity….
```

Identify the configuration files and scripts used to configure a network interface

A host must be properly set up on the network in order for ARP messages to be heard. During the system boot, the init process starts scripts located in `/sbin/rcS`. A script in this directory, `/etc/rcS.d/S30network.sh`, looks for files `/etc/hostname.hme0` and `/etc/inet/hosts` to contain the correct information about the host's name and IP addresses so the interface may be configured correctly. Ultimately, the `/sbin/ifconfig` command is run to configure the interface.
Note **hme0** is the interface in the example. There are several interface types found on Sun hardware, including le0, hme0, qe0, qfe0. There are more.

Internet Layer

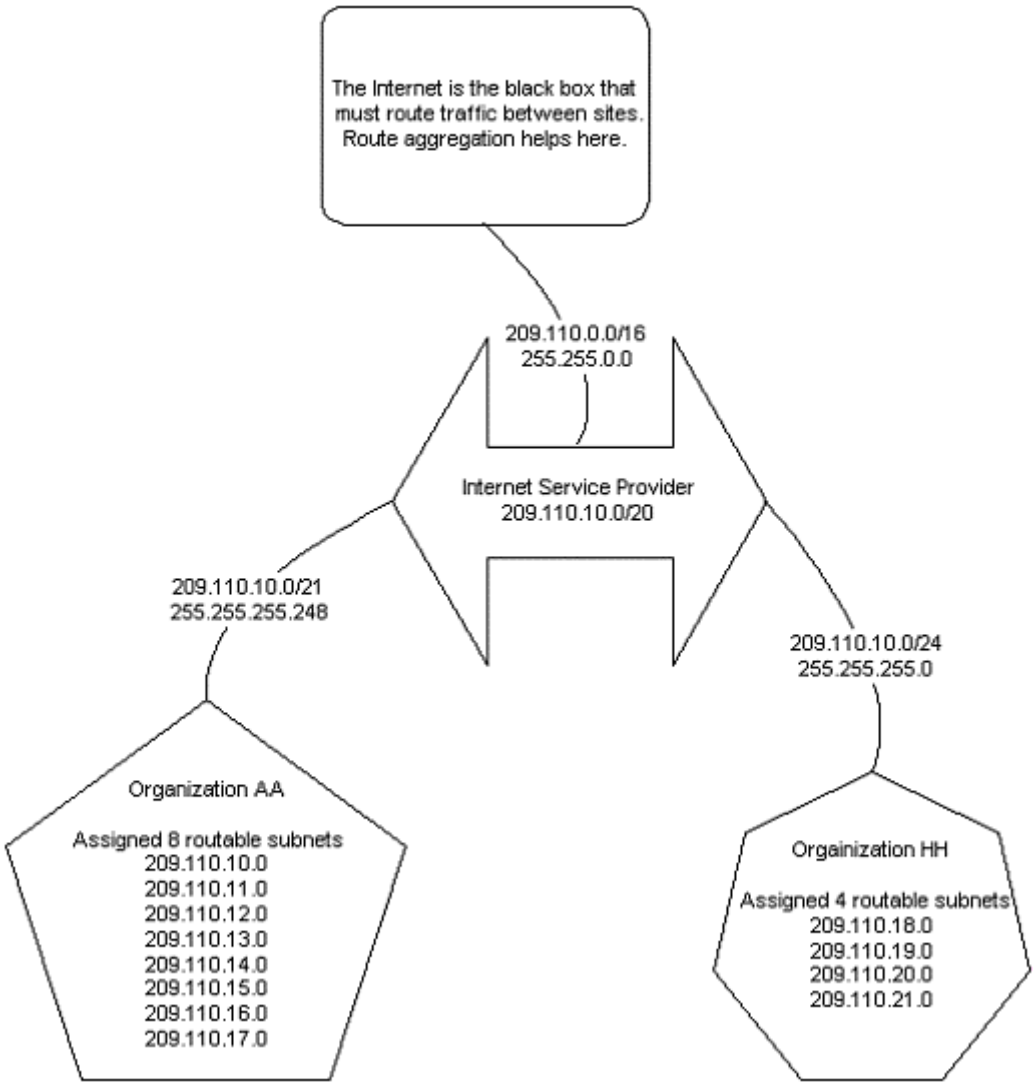
Describe the following: IP address, Broadcast address, Netmask, Datagram, and Fragment

IP is the built-in protocol responsible for the "routing" and "fragmenting" of IP's basic unit of data, namely, the datagram. Datagrams are limited to a "size" dictated by the underlying data link layer's MTU (Maximum Transmission Unit). Ethernet's MTU, for example, is 1500, meaning the biggest IP datagram Ethernet can encapsulate in a single frame is 1500 bytes.

All IP datagrams, including fragments, have a complete IP header, the minimum size being 20 bytes. Fragments may be fragmented again *en route* to a destination, if the MTU of an IP router, through which the IP packet passes, demands it.
The broadcast address is used to address all hosts on a subnet, which are sharing the same broadcast address and netmask.
The netmask parameter determines which portion of an IP address is "host specific" and which part is the network identifier. Using the netmask and the IP address we may calculate the network a host is on, and the "host specific" portion of an IP address. In addition, we may calculate the broadcast address from the netmask and the IP address.

Describe Classless Inter-Domain Routing (CIDR)

Classless Inter-Domain Routing (CIDR) is part of the answer to the rapid growth of the Internet. The standard IPV4 addresses specified in RFC 791 did not account for the large number of hosts many organizations have. The subnetting techniques of IPV4 caused inefficient use of the classful addressing schemes and increased the size of the routing tables needed to direct information. A task force designed and implemented CIDR in RFC-1517, 1518, 1519, and 1520.
CIDR is classless. This means that netmasks are used to create networks of various sizes. Internet addresses may now be written using the notation X.X.X.X/YY. For example, 129.156.1.2/24 would indicate a host address on network with subnet mask 255.255.255.0.
CIDR works because assigned IP address space can be recursively broken down into smaller sizes that fit networks of any design. Large ISPs or network providers are given blocks of addresses (by RIPE, APNIC, ARIN etc.) that they then delegate to other smaller ISPs or private organizations. Since they control a large block, they can aggregate routes to their hosts using the technique of supernetting. This reduces the size of the routing tables and increases performance since fewer lookups need to be performed.



In effect, CIDR allows for hierarchical routing, where subnets within subnets ultimately share a "main router", meaning a reduction in the size of routing tables thanks to this "route aggregation".

Identify the file used to set netmasks

The `/etc/inet/netmasks` file is used to store netmask information. It has the format:

```
network-number          netmask
129.156.0.0             255.255.255.0
```

Note that the network number is the "original" network address, and not a particular subnet. The netmask field enables TCP/IP to calculate the subnet of a given IP address.

Identify the features and benefits of the Variable Length Subnet Masks (VLSM)

Subnet masks logically segment a network to reduce collision domains (and the associated traffic), confine network protocols to segments, and logically associate departments and administrative functions in particular regions.

The default subnet masks are relatively limited because they define networks that are either too large or too small to be practical. VLSM builds on the idea of subnetting by allowing an organization to nest subnets within a particular class of address space.

For example, subnetting the network **10.0.0.0 / 255.255.0.0** would designate the second octet to be treated as a network, yielding networks 10.1.0.0 – 10.255.0.0.

An example address being: 10.1.2.3

```
i.e., Subnet 10.1.0.0     Host 2.3
```

Further subnetting the network with a **24-bit** mask **255.255.255.0** would mark the 3rd octet as being associated with a subnet using 24 bits for the network and only 8 bits for the host portion.

An example address being IP address 10.1.2.3.

i.e., Subnet 10.1.2.0 Host 3

For more granularity and one more level of subnetting, use, for example, a **27-bit** subnet mask 255.255.255.224. This results in subnets of 10.1.2.32 - 10.1.2.224. The remaining 5 bits would be associated with hosts on each of the subnets.

VLSM is implemented using the **`/etc/inet/netmasks`** file in Solaris 8.

Configure a Network Interface

The `/sbin/ifconfig` command is used to configure the network interface in Solaris™.
Basic form: `ifconfig <interface name>`

Parameters can be sent to the interface using syntax like:
`ifconfig <name> inet <ip addr> netmask <mask> broadcast + up`

WHERE: inet is the IP address, mask is the subnet mask, and broadcast + indicates the value that should be computed by the machine using whatever information is provided.

An interface is "plumbed in the kernel" using the command: `ifconfig <name> plumb`, and removed using `ifconfig <name> unplumb`.
Example:

```
# ifconfig hme0 plumb

To enable an interface, bring it "up" using ifconfig.
For example, to bring up interface "hme0", type:
# ifconfig hme0  up

We may plumb in, bring up, and set the netmask and broadcast in a
single command:
# ifconfig hme0 plumb up 194.168.85.1 netmask 255.255.255.0 broadcast +
```

Routing

Describe IP routing

At its most basic level, routing is the method of forwarding datagrams from one network to another by directing "outbound" datagrams through the correct interface (according to "routing table" information), ***en route*** to the destination. For large networks, dedicated equipment is used to communicate cross-network traffic and shield hosts from the task of determining the best route for data to take across a network.
IP routing occurs at the Internet layer. It can be described as either direct or indirect. Direct routing is when the destination host is directly attached to the same network as the sending host. Indirect routing is when the destination host is not on the same network as the sending host and the datagram, when traveling from source to destination, must pass through at least one router.
Routes are stored and referenced by the kernel using routing tables. Several schemes exist for filling these routing tables: static and dynamic. Dynamic routes are usually "learnt" and added to the kernel routing table by Solaris routing daemons.

Identify the Solaris 8 daemons which implement routing protocols

`in.routed` implements RIP for Solaris™. A host is configured with the `-q` option, and a router is configured with the `-s` option.
`in.rdisc` implements RDISC in Solaris™. Hosts are configured using the `-s` option, and routers are configured using the `-r` option.

Identify the files used to configure routing

`/etc/init.d/inetinit` is the script that checks for the existence of certain files in order to make decisions on how to start routing daemons.
If it finds **`/etc/defaultrouter`**, it creates a static, "default" route in the routing table and prevents the starting of `in.routed` or `in.rdisc`.
If the machine is configured for DHCP, or has the **`/etc/notrouter`** file, `ip_forwarding` is disabled and `in.routed -q` or `in.rdisc -s` is started.
If the system (at boot) discovers two or more physical interfaces, and the file **/etc/notrouter** does NOT exist, the system starts the daemons, in.routed –s, and in.rdisc –r and sets the kernel **ip_forwarding** parameter to 1; i.e., it will behave as a router and forward IP datagrams, if it can, based on routing table information.

Identify the purpose of the files used to configure routing

`/etc/defaultrouter` – A file that contains hostname or IP addresses of one or more default routers. The "default router(s)" is used when attempting to route packets to a **non-local** network segment and therefore, **indirect destination**, for which we have no "explicit" route in the routing table. They are in effect "catchall" routers. This file is optional, although usually used.

`/etc/inet/networks` – A file similar to hosts that assigns names to network numbers. This file is optional.
`/etc/gateways` – An optional file that may be used to define additional static routes. Read by `in.routed`. This file is optional.

Administer the routing table

The command `netstat -r` display the routing table:

```
# netstat -r
Routing Table: IPv4
  Destination           Gateway           Flags  Ref   Use   Interface
-------------------- -------------------- ----- ----- ------ ---------
194.168.85.0         194.168.85.51        U       1   398205  hme0
194.168.85.0         194.168.85.53        U       1      0  hme0:53
194.168.85.0         194.168.85.57        U       1      0  hme0:57
194.168.85.0         194.168.85.58        U       1      0  hme0:58
224.0.0.0            194.168.85.51        U       1      0  hme0
default              194.168.85.254       UG      1   178870
127.0.0.1            127.0.0.1            UH     30   3938  lo0
….Lots omitted for the sake of brevity…
```

`Netstat -rn` also prints the routing table, but without host/network name translation:

```
# netstat -rn
```

Routes can be added to the routing table using the route command.

```
route [-fn] add|delete [host|net] destination [gateway [metric]]
Example of adding a default router:
# route add default 194.168.85.254 1
Example of adding an additional static "network" route:
# route add net 62.254.198.0 194.168.85.10 1
```

Transport Layer

Identify the features of the TCP and UDP

UDP is a connectionless protocol. It trades reliability for speed and therefore requires little setup or monitoring by the sender and the receiver. Applications using UDP are responsible for sequencing and message loss. There is no handshake prior to transmission between client and server. A UDP client may transmit a UDP datagram without acquiring the consent, or connecting, with the remote end.
A UDP header has a minimum size of 8 bytes.
Its delivery mechanism may be compared to the way a letter is sent.
UDP is described in RFC 768.

TCP is a connection-oriented protocol. It requires the establishment of a virtual circuit, before transmission can begin, and employs a "positive acknowledgement with retransmission" mechanism that ensures all data is received successfully and retransmitted if necessary.
In effect the TCP client and server establish a connection at the transport layer, through a three-way handshake, prior to the application sending any data.

It uses buffering and full-duplex connection streams. This additional overhead demands more processing power. It's analogous to the phone system, where two callers must be present and agree to talk before communication can begin. In effect, both ends agree to the connection.
TCP is described in RFC 793.

Define the terms connection oriented, connection-less, stateful, and stateless

Connection-oriented – TCP is an example of a connection-oriented protocol.
Connectionless - UDP is an example of a connection-less protocol.
Stateful means that the client and server communicate about the information they send and the quality of the data they receive. Connection state is "known".
Stateless means that no such communication occurs, and the client and server operate independent of each other or the current network condition.

Describe the relationships between port numbers, network services and inetd

A port number is a virtual address that the kernel associates with any service it provides. Clients wishing to interact with a network service will contact the server and specify this special port. A daemon running on the server, inetd, listens for incoming requests, and then contacts the program responsible for the service and instructs it to respond. In effect, inetd reads /etc/inetd.conf, which indicates which services it is to enable. It acquires the "well" known port numbers for these services from the file /etc/services, and "listens" on the indicated port(s).

Client-Server Model

Explain the terms client, server, and service

The terms client, server, and service reference the relationship between these components over a network, and at a level of interaction found between the highest two layers, application and transport. In effect, TCP/IP applications are usually associated with a port number.

Administer Internet services and RPC services

Most Internet services are not started during bootup. Each process is started by inetd on behalf of the client. Internet services are managed using the following files:

Inetd is started by the script `/etc/init.d/inetsvc`.

It reads from a file `/etc/inet/inetd.conf`, which contains information about what network services are at which well-known port, and the location of the process to start.

```
#ident  "@(#)inetd.conf 1.33    98/06/02 SMI"   /* SVr4.0 1.5   */
#
#
# Configuration file for inetd(1M).  See inetd.conf(4).
#
# To re-configure the running inetd process, edit this file, then
# send the inetd process a SIGHUP.
#  <service_name> tli <proto> <flags> <user> <server_pathname> <args>
#
# Ftp and telnet are standard Internet services.
#
ftp     stream  tcp     nowait  root    /usr/sbin/in.ftpd       in.ftpd
telnet  stream  tcp     nowait  root    /usr/sbin/in.telnetd    in.telnetd
#
# Tnamed serves the obsolete IEN-116 name server protocol.
#
#name   dgram   udp     wait    root    /usr/sbin/in.tnamed     in.tnamed
#
# Shell, login, exec, comsat and talk are BSD protocols.
#
#shell  stream  tcp     nowait  root    /usr/sbin/in.rshd       in.rshd
#login  stream  tcp     nowait  root    /usr/sbin/in.rlogind    in.rlogind
#exec   stream  tcp     nowait  root    /usr/sbin/in.rexecd     in.rexecd
#comsat dgram   udp     wait    root    /usr/sbin/in.comsat     in.comsat
#talk   dgram   udp     wait    root    /usr/sbin/in.talkd      in.talkd
<...>
```

The `/etc/services` file identifies and registers well-known port numbers, services, and protocols on the machine. The standards are managed by the NIC, but an administrator can add his or her own at any time.
`/etc/services` has entries that look like:

```
ftp-data     20/tcp
ftp          21/tcp
telnet       23/tcp
smtp         25/tcp
pop3         110/tcp
```

RPC services is a service / port numbering scheme that eliminates the dedicated port requirements of inetd.
RPC services are registered with the `rpcbind` process, and do not require special setup in the `/etc/services` file. Some processes are started at boot time, others at client request. `rpcbind` is started at run level 2 using `/etc/init.d/rpc`. Clients requesting a service interact with rpcbind on port 111. They are then given a unique port number when the associated service runs.
The program `/usr/bin/rpcinfo` can be used to monitor the activities of RPC.

Collect information about services configured on hosts

`netstat` and `rpcinfo` are used to collect information about services on hosts.
`netstat -a` identifies ports on a host, and what connections are established.

```
UDP: IPv4
   Local Address         Remote Address     State
-------------------- -------------------- -------
      *.ntp                                Idle
localhost.ntp                              Idle
myhost.nowhere.net.ntp                     Idle
localhost.domain                           Idle
myhost.nowhere.net.domain                  Idle
      *.*                                  Unbound
      *.65249                              Idle
      *.syslog                             Idle
      *.40227                              Idle
      *.*                                  Unbound

TCP: IPv4
   Local Address        Remote Address    Swind Send-Q Rwind Recv-Q  State
-------------------- -------------------- ----- ------ ----- ------ -------
      *.*                  *.*                0      0 24576      0 IDLE
      *.22                 *.*                0      0 24576      0 LISTEN
myhost.nowhere.net.22 209.110.245.243.2792 31856     0 24616      0 ESTABLISHED
myhost.nowhere.net.22 209.110.245.243.2793 31856     0 24616      0 ESTABLISHED
localhost.domain           *.*                0      0 24576      0 LISTEN
myhost.nowhere.net.domain  *.*                0      0 24576      0 LISTEN
      *.smtp               *.*                0      0 24576      0 LISTEN
      *.587                *.*                0      0 24576      0 LISTEN
      *.3306               *.*                0      0 24576      0 LISTEN
localhost.8080             *.*                0      0 24576      0 LISTEN
localhost.8021             *.*                0      0 24576      0 LISTEN
localhost.8099             *.*                0      0 24576      0 LISTEN
      *.443                *.*                0      0 24576      0 LISTEN
      *.*                  *.*                0      0 24576      0 IDLE


Active UNIX domain sockets
Address  Type          Vnode     Conn  Local Addr      Remote Addr
300009a01a8 stream-ord 30000bc0958 00000000 /usr/local/Zope/pcgi.soc
300009a0828 stream-ord 3000080faa8 00000000 /tmp/mysql.sock
```

`rpcinfo` displays the program number, version, protocol, port number, service, and owner.
`rpcinfo -p <hostname>` shows all RPC services running on a host.
`rpcinfo -u <server> <process>` shows if a specific server is running on a host.
`rpcinfo -b <process>` broadcasts a request to hosts on a network to see if the specified process is running, and displays their machine name and port.
`rpcinfo -d <process>` unregisters a service on a host.

DHCP

State the benefits of DHCP

DHCP eases network IP address management by allowing administrators to dynamically configure network information for clients from a centrally administered server. These benefits reduce cost associated with network management, as well as help alleviate the problem of IP address depletion.
DHCP is described in RFC 2131.

Identify DHCP configuration files

`dhcptab` and `dhcp_network` are the two files used to configure DHCP behaviors.
DHCP options are stored in the `/etc/dhcp/inittab` file.

```
#
#ident  "@(#)inittab    1.3     99/08/16 SMI"
#
# This file provides information about all supported DHCP options, for
# use by DHCP-related programs.  This file should only be modified to
# add support for SITE options or new STANDARD options; no existing
# options should be modified.  Please note that errors introduced into
# this file may cause programs to crash.
#
# Please consult dhcp_inittab(4) for further information.  Note that
# this interface is "Unstable" as defined by attributes(5).
#
Subnet          STANDARD,       1,      IP,         1,   1,      sdmi
UTCoffst        STANDARD,       2,      SNUMBER32,  1,   1,      sdmi
Router          STANDARD,       3,      IP,         1,   0,      sdmi
Timeserv        STANDARD,       4,      IP,         1,   0,      sdmi
IEN116n         STANDARD,       5,      IP,         1,   0,      sdmi
DNSserv         STANDARD,       6,      IP,         1,   0,      sdmi
Logserv         STANDARD,       7,      IP,         1,   0,      sdmi
CookieS         STANDARD,       8,      IP,         1,   0,      sdmi
<...>
```

The DHCP configuration file that allows an administrator to tune some RFC parameters is `/etc/default/dhcpagent`.

```
#
# This file contains tunable parameters for dhcpagent(1M).
#
# All parameters can be tuned for a specific interface by prepending
# the interface name to the parameter name.  For example, to make
# RELEASE_ON_SIGTERM happen on all interfaces except hme0, specify:
#
# hme0
```

```
.RELEASE_ON_SIGTERM=no
# RELEASE_ON_SIGTERM=yes
# By default, when the DHCP agent is sent a SIGTERM, all managed
# interfaces are dropped.  By uncommenting the following
# parameter-value pair, all managed interfaces are released instead.
#
# RELEASE_ON_SIGTERM=yes
<...>
```

You configure the DHCP data store preference (NIS or FILES) using the `/etc/default/dhcp` file.

State the purpose of DHCP configuration files

`dhcptab` contains the macro table used for DHCP clients. It has three fields, Name, Type, and Value.

Name – user-defined name for the record
Type – (s)ymbol or (m)acro
Value – value pairs that define the symbol (delimited by ',') or macro (delimited by ':')

```
Name        Type  Value
net-30       m
                  :Timeserv=10.30.86.2:LeaseTim=259000: \
                  :DNSdmain=gvon.com:DNSserv=194.168.85.1 \
                  194.168.85.2:LeaseNeg
```

`dhtadm` has several flags for creating entries in the `dhcptab` file.

`dhtadm -C` creates a `dhcptab` configuration file
`dhtadm -A` adds a macro definition to `dhcptab`
`dhtadm -M` modifies an existing macro or symbol in `dhcptab`
`dhtadm -D` deletes a macro definition
`dhtadm -R` removes the `dhcptab` file
`pntadm` manages the `dhcp_network` file
`pntadm -C` creates a `dhcp_network` file
`pntadm -A` adds a client entry to `dhcp_network`
`pntadm -D` deletes a specified client entry
`pntadm -P` prints the `dhcp_network` file
`pntadm -R` removes the `dhcp_network` file

An entry in dhcp_network might look like the following:

```
Client_ID   Flags   Client_IP        Server_IP     Lease  Macro
10           00     194.168.85.200   194.168.85.51 0      net-30
```

Administer DHCP clients and servers

Solaris™ includes a GUI program called dhcpmgr, which is a graphical interface to the older dhcpconfig program. It is an easy, step-by-step wizard to guide an administrator through the server configuration process.
`# /usr/sadm/admin/bin/dhcpmgr`
Clients can be setup to use DHCP using `ifconfig <interface> DHCP`, or by touching a file `/etc/dhcp.<interfacename>`
For troubleshooting, DHCP can be started in several debugging modes:
`# /sbin/dhcpagent -d3` (debugging level 3)

Network Management Tools

Identify tools which use the Simple Network Management Protocol (SNMP)

SNMP agents can be set up on many types of network and operating system devices. Information sent out by these agents can be read by vendor applications like HP Openview, Sun Management Center™, Solstice Domain Manager™ and Solstice Site Manager™. OpenSource tools are also available. In particular, UC-Davis' UCD-SNMP.

Describe the Simple Network Management Protocol (SNMP)

Network management includes such concepts as system configuration, fault correction, performance tuning, accounting and security maintenance.
SNMP is a UDP standard that uses standard calls – `Get`, `Set`, and `Trap` – to retrieve or place data into object identifiers (**OID**s). SNMP is based on UDP, because it is generally accepted that a poorly performing device will be able to send out messages relating to the problem quickly in the event of a failure.
The functions of SNMP are:
Get - management consoles will poll all of their SNMP-managed devices in the field to obtain status information. The `gets` update a graphic or table.
Set - should a management console need to change a managed device in some way, it can send out a `set`.
Trap - When a managed device has a failure or needs to communicate with the management station, it sends a `trap`.
Information is described according to the Structure of Management Information (**SMI**), which describes how objects are stored in a management information base (**MIB**). Objects in a MIB are defined according to ASN.1 notation. ASN is the Abstract Syntax Notation.
SNMP is described in RFC 1157.

Domain Name System

Identify the purpose of DNS

DNS is a solution to the problems inherent in managing computer system hostnames. These hostnames must have an efficient way to resolve their corresponding numerical "IP" addresses, and maintain uniqueness on the Internet with respect to individual organizations. (i.e., `saturn.gvon.com` and saturn.sun.`com` share similar hostnames but are unique machines on the Internet). The most widely used version of DNS in UNIX is BIND. Solaris 8 ships with BIND 8.1.2.
See www.isc.org for further information.
DNS is described in RFC 1035 and several other RFC's.
In what follows, the word master and primary are interchangeable. Modern BIND (Version 8 and 9) refer to "master" for what version 4 (still heavily used) called a "primary". The modern term "slave" (BIND versions 8 and 9) is interchangeable with "secondary" for the same reasons.

Describe address resolution and reverse address resolution

A DNS client is called a "resolver" because it passes the DNS query to a DNS server, for the server to resolve either an address to a name mapping (reverse resolution) or a name to an address mapping (forward resolution). The client is in effect a resolver for the application. For example, Netscape passes to the client resolver routines, names like www.gvon.com that need "resolving" for the application.
The traffic between the client and the server is said to be "recursive". When a DNS server asks another DNS server a question, however, the traffic between the two servers is usually "iterative". Servers can, however, also ask for recursion.

Recursive – a recursive request is one that must be satisfied by a nameserver. When a resolver sends a recursive request, the queried nameserver is obliged to return a valid answer. It can't just direct the client (resolver) to another name server.
Iterative – an iterative request attempts to locate a server that has the best information. When a resolver sends an iterative request, the queried nameserver returns its best answer, which may be from its non-authoritative cache or the name of a server it believes may have more information.

In Solaris™, address resolution occurs in the order dictated by the /etc/nsswitch.conf file. This is
hosts: files dns

Or perhaps, if NIS is being used:
hosts: files nis dns
Tools like nslookup and dig can perform both regular and reverse address resolution.

Identify the correct Resource Record syntax

Entries in the name server database are known as resource records. A record has the format:

```
[name] [ttl] [class] [type of record] [record data]
```

The Name field is usually:

- A domain name (or part of) - see NS records
  - uk
  - gvon.com
- A machine name (qualified or not) - see A records
  - hercules
  - hercules.gvon.com
- IP address in reverse (or part of) - see PTR records
  - 51.85.168.194
  - 51.85.168.194.in-addr.arpa

ttl (time to live) indicates how long the record data is to be held in memory (DNS in.named cache) by other DNS servers gleaning the data from the DNS server(s) of the zone
Class for the Internet is almost always IN. There are others but IN is the most important.
Record type indicates the type of information the record provides. Here are the most important record types currently in use:

- **SOA** is the fully qualified address of the master (primary) DNS server for the zone. For example,

```
@ IN SOA auriga.gvon.co.uk. hostmaster.gvon.co.uk. (
76500 ; Serial
10800 ; Refresh every 3 hours
3600 ; Retry every hour
604800 ; Expire after a week
43200 ) ; Default TTL 12 hours
```

- **NS** records specify name servers of the domain in the **name** field of the record. For example,
  gvon.co.uk. in NS auriga.gvon.co.uk.
  gvon.co.uk. in NS centauri.gvon.co.uk.

*Please note that the name field in NS records (gvon.co.uk in above example) is usually blank because the domain name (zone) is usually learned from the name

field of the previous record line, which is usually the SOA record. The previous name field is therefore usually **@**, (current origin), or put simply, the domain/zone of the current zone file; i.e., gvon.co.uk.

- **A** records list the **name (name field)** and **IP address (data field)** of a given host. For example:
  ```
  hercules    in    a    194.168.85.60
  ```

- **PTR** records list IP to name mapping. For example:
  ```
  60.85.168.194    in    ptr    hercules.gvon.co.uk.
  ```

- **MX** records list the **mail servers (smtp servers)** prepared to accept inbound mail for the domain in the name field. For example:
  ```
  gvon.co.uk.  in    mx    5    hercules.gvon.co.uk.
  gvon.co.uk.  in    mx    10   ophelia.gvon.co.uk.
  ```

- **CNAME** records allows for aliases of the data field; i.e., the name field is an alias for the data field. For example:
  ```
  www.gvon.co.uk.    In    cname    voyager.gvon.co.uk.
  ```

If the current origin in gvon.co.uk, one could simply have used:
```
www                in    cname    voyager.gvon.co.uk.
```

Relative name fields have the current origin (@, in the example gvon.co.uk) appended so www becomes www.gvon.co.uk in the cache.

Explain the steps needed to configure DNS

DNS requires some specific information before a host can be configured. Particularly:

- Names of root level servers that have information about GTLD's (Global Top Level Domains); i.e., com, net. org, uk, us, de. There are many more.
- Names of domains (zones) our server is to serve.
- Name to Address translations of all hosts in the domain(s)
- Address to Name translations (reverse lookups) of all hosts
- Any servers that contain information about subdomains we choose to create

Identify the configuration files for DNS

Configuration files for DNS include the:

- `/etc/named.conf` file, the main configuration file
- root.cache file (also called the root hints file)
- zone files, which contain information about the zones we control

- `/etc/resolv.conf` file, which is needed by clients to identify local DNS server(s)
- `/etc/nsswitch.conf` file. Configure it and add dns to the hosts line

State the purpose of DNS configuration files

`/etc/named.conf` identifies the zones we serve, as well as identifying the location of the files that contain zone information for the domains we are authoratative for.
The file contains other configuration related parameters.
The cache (or "hints") file is known as `named.root`, and contains the IP addresses and names of the root servers. This file is used for identifying the authoritative servers of the GTLD's. Currently there are 13 root level servers, running UNIX, of course; what else?
Zone files contain the RR records described above, pertaining to the zones we serve as an authoratative, master (primary) or slave (secondary) server. Please note that both master and slave servers are considered authoratative.
Updates are done on the primary server and pushed out to secondary servers.
In addition, modern BIND (version 8) has a notify feature whereby a master may notify the slave of changes, rather than wait for the slave to check for changes later.

A primary or secondary server can give out authoritative information for a zone.
A client `/etc/resolv.conf` file looks like this:

```
domain gvon.com
nameserver 194.168.85.1
nameserver 194.168.85.2
```

`domain` specifies the local domain information (to append to unqualified hostnames).
`nameserver` specifies the IP of the local DNS server(s).
Solaris™ supports a maximum of three `nameserver` entries.

Network Time Protocol

Describe the NTP features

Network Time Protocol is a method of keeping system times accurate, regardless of the world's time zones. It uses the reliable time standard Universal Time Coordinated (UTC) to synchronize clocks. UTC is an accepted estimate of the current time based on several institutions' calculations.
NTP uses broadcasts and multicasts to learn about time updates. Accurate time is a necessity for systems because it aids in network management, logging and file time stamps, and even as an element of generating encryption keys.

Identify NTP configuration files

NTP is provided in the packages `SUNWntpr` and `SUNWntpu`. The main configuration file for NTP is `/etc/inet/ntp.conf`. An example configuration file looks like:

```
# @(#)ntp.server      1.5     99/09/21 SMI
#
# /etc/inet/ntp.server
#
# An example file that could be copied over to /etc/inet/ntp.conf and
# edited; it provides a configuration template for a server that
# listens to an external hardware clock, synchronizes the local clock,
# and announces itself on the NTP multicast net.
#

#
# * All TrueTime receivers are now supported by one driver, type 5.
#   Types 15 and 25 will be retained only for a limited time and may
#   be reassigned in future.
#
# Some of the devices benefit from "fudge" factors.  See the xntpd
# documentation.

# Either a peer or server.  Replace "XType" with a value from the
# table above.
server 127.127.XType.0 prefer
fudge 127.127.XType.0 stratum 0

broadcast 224.0.1.1 ttl 4

enable auth monitor
driftfile /var/ntp/ntp.drift
statsdir /var/ntp/ntpstats/
filegen peerstats file peerstats type day enable
filegen loopstats file loopstats type day enable
filegen clockstats file clockstats type day enable

keys /etc/inet/ntp.keys
trustedkey 0
requestkey 0
controlkey 0
```

A drift file must also exist. Make sure that `/var/ntp/ntp.drift` exists.
NTP logs messages to the `/var/adm/messages` file. Any updates to the system time are recorded here.
The NTP daemon can be monitored using the `xntpdc` program. It is interactive and can be used to obtain information about configured time servers or other host peers.

State the purpose of NTP files

`/etc/inet/ntp.conf` is the file that lists the preferred timeservers according to strata. Stratum servers, listed highest to lowest in terms of accuracy, determine how precise the time is going to be.
The drift file lists the internal clock's frequency offset and helps keep internal timekeeping more accurate.
NTP clients send NTP broadcast packets, which include their local time, onto the network. NTP servers respond to the packets by inserting their own time. The clients then use these numbers to determine how long the packet was on the network, and sets internal time according to estimates gained from receiving several responses.
`/etc/init.d/xntpd` is the daemon that starts NTP.

Describe how to configure NTP

It is important that a server use external reference servers. See http://www.eecis.udel.edu/~mills/ntp/clock1.htm for a list of clocks to reference.
Copy the template `/etc/inet/ntp.server` to `/etc/inet/ntp.conf`.
Make entries in the conf file similar to:

```
128.50.10.254 prefer
fudge 127.127.XType.0 stratum 0
```

Create the drift file:

```
# touch /var/ntp/ntp.drift.
```

Start the NTP daemon using `/etc/init.d/xntpd start`

Network Troubleshooting

Identify common network problems

There is no book that can teach network troubleshooting fully, although, how to methodically approach troubleshooting, may be successfully taught. Books may also be useful and suggest possible causes of problems (a FAQ for example) and show correct configuration, in terms of how to configure web, mail, DNS, ftp, and samba servers.
I run an ISP and assure you that correct configuration can be documented.
Through experience and experimentation, an administrator can become skilled at solving problems that occur. After all, experience is often the name we give to our mistakes! Certainly, the more complex or obscure problems can prove a challenge even to the experienced network administrator.
Extensive "hands-on" experience, coupled with "good" theory (often learnt from a book, web sites, news groups etc.), enable us to extend our troubleshooting skills.

Try and trace a problem to its source by simplifying the environment and by removing extraneous equipment. Trial and error is a main principle, but it comes back to recreating the problem after changes have been made. Document configuration changes as you work, and summarize the corrections for the final solution.
Often problems occur because of recent configuration changes, so identify recent changes as perhaps the cause of "new" problems.

Diagnose network problems

Network problems can occur at all layers of the TCP/IP model. Faulty wiring can occur at the physical layer. Duplicate addressing, both MAC and IP addresses, cause problems in the middle Network and Internet layers. Poorly configured applications can be the source of many headaches at the highest application layer.
Don't assume anything when troubleshooting network problems. You might want to verify systems with known working cabling and network cards.

Resolve network problems

Sometimes the best tools for fixing network problems are basic tools.
These may include:
ping – uses ICMP "echo request/reply" functions to test connectivity between hosts. It is fast because ICMP packets are encapsulated in an IP packet; i.e., there is no transport layer (TCP/UDP) involvement.

ifconfig – shows the status of configured interfaces and includes information relating to the MTU, IP address, Netmask, Broadcast address, and MAC address.

In Solaris™, there are actually two ifconfig commands. The first, `/sbin/ifconfig`, does not reference NIS+ nsswitch.conf configuration information. The other, `/usr/sbin/ifconfig`, does use name service information in the `nsswitch.conf` file.
arp – when used with the `-a` option, it can display the table of cached hardware addresses on the system. See also **netstat -p**
snoop – used to display on-the-fly network traffic on an interface. It can be an important tool in troubleshooting almost any issue. It's used with the `-v` or `-V` options for verbosity. It writes to a file with the `-o` option, and views a file it created using `-i`.
ndd – used to display and set driver configuration parameters.
netstat – useful for determining the state of the systems interfaces. Displays the routing tables with the `-r` option, interfaces with –i, stats with –s, socket info with –a.
traceroute – a network troubleshooting tool that reveals the state of the network between the client and the destination. It uses IP time-to-live values to try and max

out values of ICMP TIME_EXCEEDED, PORT_UNREACHABLE and ICMP ECHO_REPLY on routers and destination hosts.
Another useful and FREE tool related to "gleaning" and monitoring network related information is ntop by Luca Deri. See http://www.ntop.it/.

IPv6

Describe IPv6

IPv6 (RFC 2460) is another solution to the problem of IP address depletion. The Internet Architecture Board pioneered it in 1991.
IPv6 designates 128 bit unique IP addresses. This provides many more addresses than the IPv4 scheme. Certain types of addresses are assigned to clients automatically once by an administrator, leading to greater "automation" of address allocation. See link and site local addressing.
In addition, because of a simplified header, packets addressed using IPv6 can be routed much faster than a standard IPv4 packet.
Autoconfiguration of the IPv6 protocol comes in two forms: stateful and stateless. A stateful configuration involves another device (like a DHCP server) that must assign the address to the client. Stateless means that a host can generate its own address based on MAC address and other learnt information.
When a host auto-configures its unique address, it creates a locally-linked address from messages received from neighboring hosts. All neighboring devices and systems will receive a broadcast telling them about the new address.
The system's MAC address is used in generating an IPv6 global address. Remember, the MAC is made of two parts, a Company ID (24 bits) and a Vendor ID (24 bits). First, the MAC is converted into binary. Next, the seventh bit from the left, known as the universal bit, is toggled (from 1 to 0 or 0 to 1). Finally, two constant octets -- 0xFF and 0xFE -- are inserted between the Company ID and Vendor ID. The final address is then converted to HEX.
Each address begins with what is known as a FP (format prefix), or address indicator. It can be:

FE8 (local-link)
FEC (site-local)
FF (multicast)

Local Link addresses cannot be routed. Site-Local addresses can be routed.
Multicasts are delivered to all interfaces on the network (similar to a broadcast).
Familiar protocols in IPv4 have incompatible IPv6 cousins. ICMPv6, IGMP and Neighbor Discovery Protocol (which replaces ARP/RARP) are some of them. ICMP and NDP work together to find and assign local routers, advertise routes, and determine neighboring hosts and their addressing schemes.

Configure an IPv6 network interface

Solaris™ 8 supports IPv6 in *dual-stack* mode. This means that the kernel can understand traffic for both IPv4 and IPv6. The operating system cannot support an IPv6-only network scheme at this time.
To configure a Solaris™ 8 system to use IPv6, create an `/etc/hostname6.hme0` file and restart.
The IPv6 equivalent to the `/etc/hosts` file is `/etc/inet/ipnodes`.
Once IPv6 is set up on the system, configuring an interface using ifconfig is as easy as running this command:

```
# ifconfig hme0 inet6
hme0: flags=2000841<UP,RUNNING,MULTICAST,IPv6> mtu 1500 index 2
      inet6 fe80::d01:77ff:fe29:aaeb/10
```

Note that the IP was assigned automatically. The FP `fe80` indicates that the address is locally linked.
Some tools for IPv4 have IPv6 equivalents. These include `netstat`, which can be run as family type ip6:

```
# netstat -f inet6
TCP: IPv6
Local             Remote            Swind Send  Rwind Recv  State If
Address           Address           -Q          -Q
------------------------------ ------------------------------------
localhost.33382   localhost.33319   32768    0  32768 0     ESTABLISHED
localhost.33319   localhost.33382   32768    0  32768 0     ESTABLISHED
```

DNS is configured similarly, except the records are longer. A new record type is the AAAA (quad A).
There is only one routing daemon for IPv6, which is called `in.ripngd`. There are no configuration parameters for this daemon.
The neighbor discovery protocol is implemented by `in.ndpd`. This daemon is started only if a configuration file `/etc/init/ndpd.conf` exists.

Special thanks to
Matthew Kortas
for writing the original Cramsession for this exam, and
Rick Bushnell BA(Hons),PGCE, MBCS, SCNA
for extensive updates to the material.
Please visit Matthew's site at
http://acm.cse.msu.edu/~kortasma
and Rick's site at
http://gvon.com.