information technology & management

viabiliitjit

INTRO TO OPEN SOURCE OPERATING SYSTEMS

ILLINOIS INSTITUTE OF TECHNOLOGY

ITMO456

Networking Linux

Sean Hughes-Durkin

ITMO/IT-O 456 Fall 2017

Information Technology & Management Programs

School of Applied Technology

Objectives

At the end of this lesson students should be able to:

- Describe the purpose & types of networks, protocols, and media access methods
- Explain the basic configuration of TCP/IP
- Configure a NIC to use TCP/IP
- Describe the purpose of host names and how they resolve to IP addresses
- Configure TCP/IP routing

Objectives

At the end of this lesson students should be able to:

- Configure TCP/IP routing
- Identify common network services
- Use command-line and graphical utilities to perform remote administration

Networks and TCP/IP

- **Network**
  - Two or more computers connected with media that can exchange information
- **Local Area Networks (LANs)**
  - Connect computers within close proximity
  - Allow connections to shared resources
- **Wide Area Networks (WANs)**
  - Connect computers separated by large distances
  - Connects to Internet Service Provider

Networks and TCP/IP

- **Internet service provider (ISP)**
  - Company providing internet access
- **Routers**
  - Devices capable of transferring packets between networks
- **Protocol**
  - Set of rules of communication used between computers on a network

Networks and TCP/IP

- **Packets**
  - Packages of data formatted by a network protocol
  - Packets can be recognized by routers and other network devices
- **Media access method**
  - Defines how networked computers share access to the physical medium

Networks and TCP/IP

- **Ethernet**
  - Most common network media access method
  - Ensures packets are retransmitted onto the network if a network error occurs
- **Token ring**
  - (Formerly) popular media access method
  - Controls which computer has the ability to transmit information

The TCP/IP Protocol

- Set of protocols with two core components
  - TCP ensures packets are assembled in correct order, regardless of arrival order
  - IP is responsible for labeling each packet with destination address
- Together, TCP and IP ensure packets travel across the network as quickly as possible without getting lost

The TCP/IP Protocol: Packets

- Data using TCP/IP travels in small chunks called packets
  - Packets each travel independently
  - Each contains necessary address information to reach its destination
  - Highway metaphor is particularly appropriate
    - Flow of packets is accordingly called “traffic”
  - Packets sometimes called datagrams

The TCP/IP Protocol: Packets

- **IP** moves packets from node to node
  - Forwards each packet based on a four byte destination address, the IP number
  - IP routes packets to the destination
- **Transmission Control Protocol**
  - Verifies correct delivery of data from client to server
  - Detects errors or lost data
  - Triggers retransmission until data is correctly and completely received
  - TCP deals with ports (1-65535)

Networks and TCP/IP: UDP

- **User Datagram Protocol**
  - Simplest of the common transport-layer TCP/IP protocols
  - No procedures to correct for out-of-order packets, guarantee delivery, or improve IP but can be faster than more sophisticated tools
  - Runs Domain Name System (DNS), Network File System (NFS), and many streaming media protocols
    - Streaming media protocols not always UDP
  - UDP deals with ports (1-65535)

Networks and TCP/IP: ICMP

- **Internet Control Message Protocol**
  - Simple protocol for communicating data
  - Most often used to send error messages between computers
  - Works at the IP layer – does not deal with ports (TCP/UDP)
  - Ping and traceroute utilize ICMP
    - Some traceroute commands use UDP by default but have an option flag for ICMP

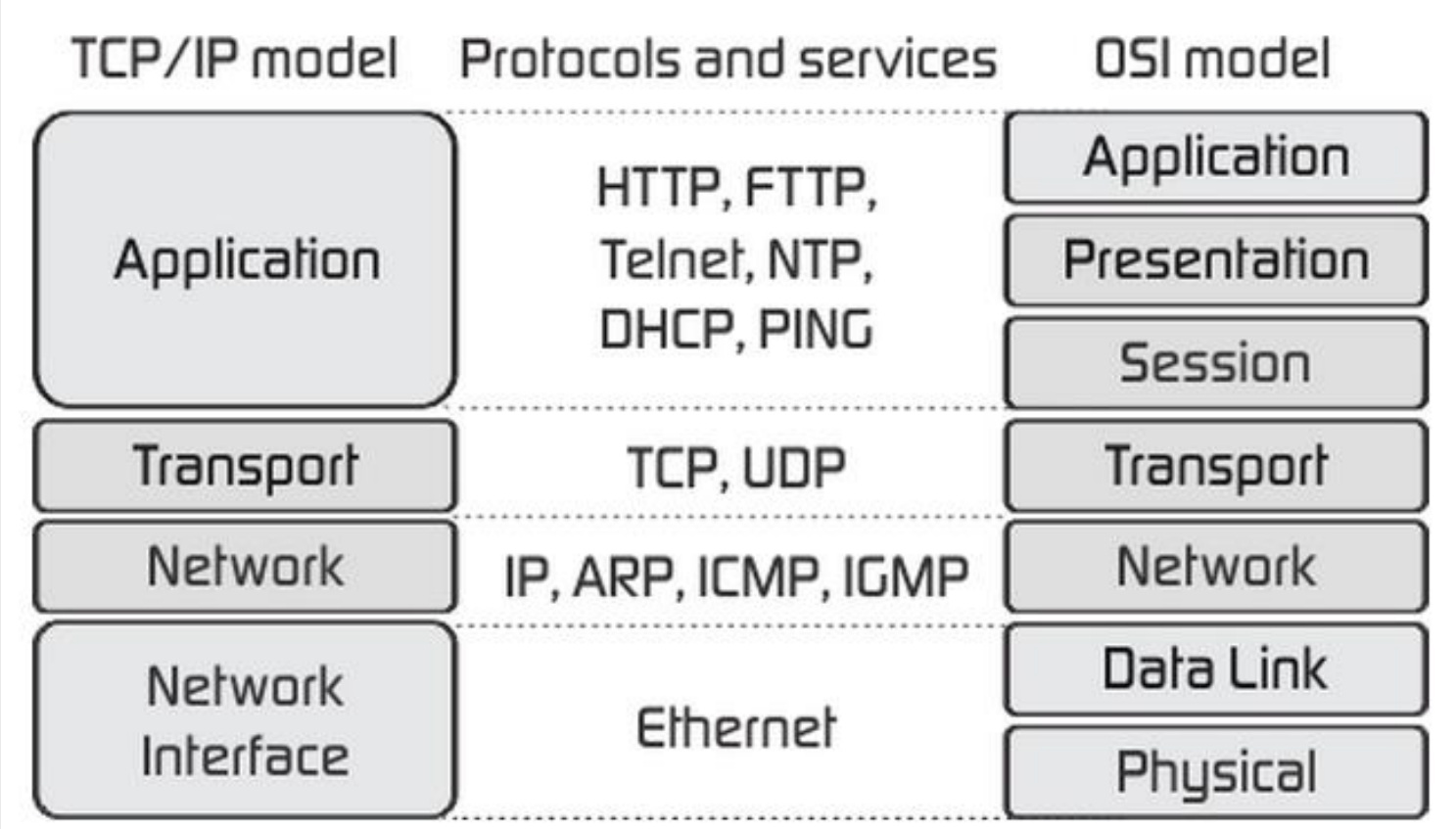
Networks and TCP/IP

- Ethernet
  - The most common media access method used in networks today
- Token Ring
  - (Formerly) popular media access method
  - An IBM standard
- Media access method usually contained within the hardware on the NIC or modem

Networks and TCP/IP

- Ethernet & Token Ring devices have unique Media Access Control (MAC) addresses
  - In Ethernet, 6 bytes expressed in hex
  - Look like this: **`00:A0:CC:24:BA:02`**
  - First 3 bytes represent the manufacturer
    - ie: Cisco, Juniper, Dell, etc
    - http://aruljohn.com/mac.pl

Networks and TCP/IP



Networks and TCP/IP

- Ethernet hardware in Linux typically named **eth*n***, where ***n*** is a number from 0 up
  - Wireless devices named **wlann**
  - No entries in **/dev**

Networks and TCP/IP

- Fedora uses new naming convention due to systemd
  - Wired ethernet interfaces should now be named with a prefix of “en”
    - "eno1", "ens1", "enp2s0" or similar
  - Wireless ethernet interfaces should now be named with a prefix of “wl”
    - “wlp12s0” or similar

Networks and TCP/IP

- From the systemd udev-builtin-net_id.c source code:
  - Two character prefixes based on the type of interface:
    - en Ethernet
    - sl serial line IP (slip)
    - wl wlan
    - ww wwan

The TCP/IP Protocol: Addresses

- To participate on a TCP/IP network, a computer must have a valid IP address
  - As well as a subnet mask
  - To participate on a larger network (Internet), you need to configure a default gateway

The TCP/IP Protocol: IPv4 Addresses

- Internet Protocol (IP) address (IPv4)
  - Unique series of four 8-bit numbers—octets—that represent a computer on a network
  - Identifies a computer on the network
- Internet Protocol (IP) address (IPv6)
  - Next generation; uses 128-bit addresses
- Unicast
  - Directed TCP/IP communication from one computer to another single computer

The TCP/IP Protocol: IPv4 Addresses

- IPv4 addresses composed of two parts:
  - Network ID: Network computer is located on
  - Host ID: Single computer on that network
    - Two computers with different network IDs can have the same host ID
- Only computers with same network ID can communicate without a router
  - Allows administrators to logically separate computers on a network

The TCP/IP Protocol: Subnet Masks

- Subnet mask
  - Define which part of IP address is the network ID and which part is the host ID
  - Series of four 8-bit numbers or *octets*
  - Octet in subnet mask containing 255 is part of network ID
  - Octet in subnet mask containing 0 is part of host ID

The TCP/IP Protocol: Subnet Masks

- ANDing
  - Calculate network and host IDs from an IP address and subnet mask
  - Compare binary bits

The TCP/IP Protocol: Subnet Masks

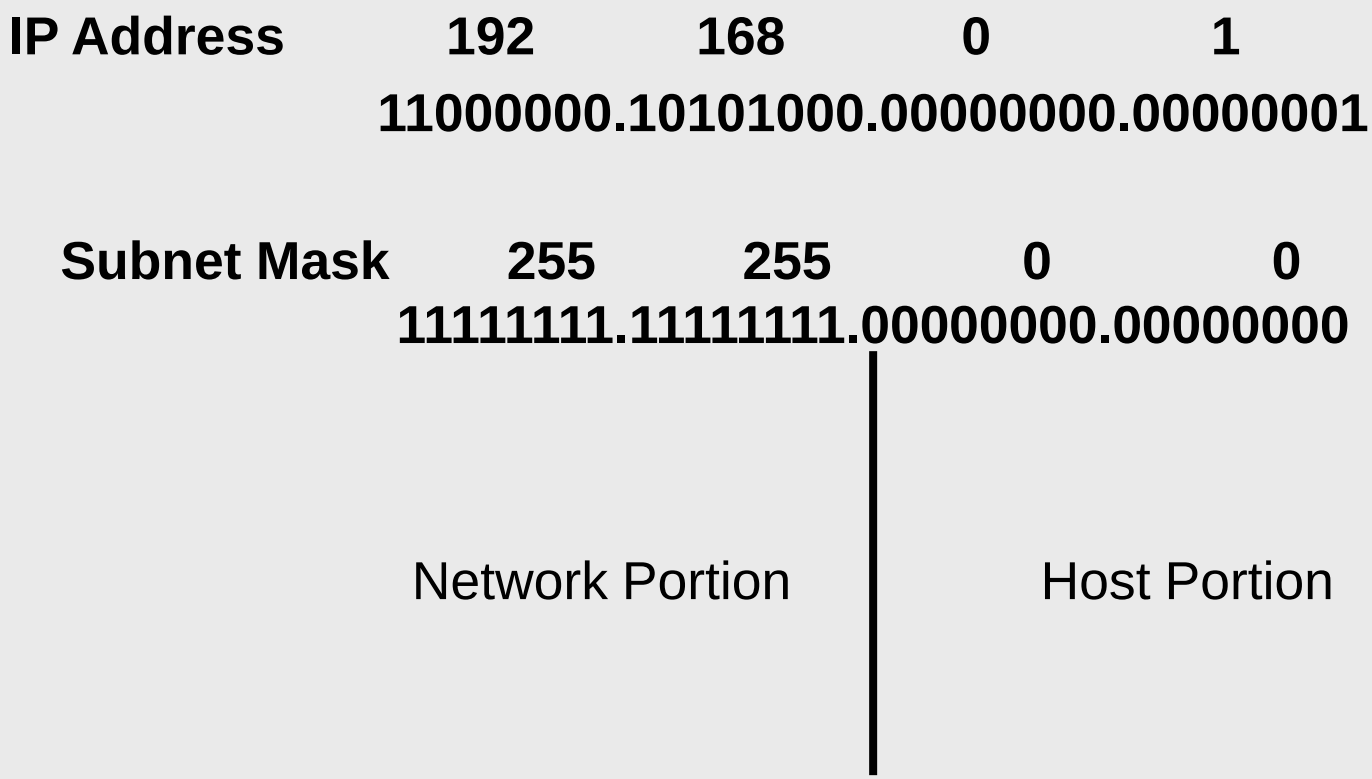


Figure 12-1: A sample IP address and subnet mask (IPv4)

The TCP/IP Protocol: Subnet Masks

- IP addresses that cannot be assigned to a host computer
  - 0.0.0.0 = all networks
    - Default route
  - 255.255.255.255 = all computers
    - Global Broadcast
- 255 in an IP address can specify many hosts
  - i.e. Broadcast addresses
  - Example: 192.168.255.255 refers to all hosts on the 192.168.0.0/16 network

The TCP/IP Protocol: Default Gateway

- The Default Gateway is the IP address on a router that sends packets to remote networks
- Routers can distinguish between different networks
  - Move packets between them
  - Have assigned IP addresses on each attached network

IPv4 Classes and Subnetting

- IP address class defines default subnet mask of associated device
  - All IPv4 address classes identified by first octet

IPv4 Classes and Subnetting

- Class A (First octet 0 – 127)
  - 8 bits for network ID, 24 bits for host ID
  - Assigned to very large companies
- Class B (First octet 128 – 191)
  - 16 bits for network ID, 16 bits for host ID
  - Assigned to larger organizations with several thousand users
- Class C (First octet 192 – 223)
  - 24 bits for network ID, 8 bits for host ID
  - Used for small and home networks

IPv4 Classes and Subnetting

- Multicast
  - TCP/IP communication destined for a certain group of computers
  - Class D addresses
- Subnetting
  - Divide large network into smaller networks
  - Control traffic flow
  - Take bits from host ID, give to network ID

Private Address Ranges

- Non-routable addresses used for local subnets
  - 10.0.0.0 – 10.255.255.255 (16 million +)
  - 172.16.0.0 – 172.31.255.255 (~1 million)
  - 192.168.0.0 – 192.168.255.255 (65,536)
- Used in Network Address Translation (NAT) schemes
  - Allow an entire subnet to share one routable IP address

TCP/IP Classes and Subnetting

Class	Subnet Mask	First Octet	Maximum Number of Networks	Maximum Number of Hosts	Example IP Address
A	255.0.0.0	1-127	127	16,777,214	3.4.1.99
B	255.255.0.0	128-191	16,384	65,534	144.29.188.1
C	255.255.255.0	192-223	2,097,152	254	192.168.1.1
D	N/A	224-239	N/A	N/A	224.0.2.1
E	N/A	240-254	N/A	N/A	N/A

Table 12-1: IPv4 IP address classes

The IPv6 Protocol

- Number of IP addresses using IPv4 is unsuitable for Internet growth
- IPv6 uses 128 bits to identify computers
  - Addresses written using eight 16-bit hexadecimal numbers
  - 2001:0db8:3c4d:0015:0000:0000:adb6:ef12
  - 0000 can be omitted in most notation
  - Above address could also be written:
    - 2001:0db8:3c4d:0015:::adb6:ef12

The IPv6 Protocol

- IPv6 address has two portions
  - First half assigned by ISP: identifies network
  - Last half is link local portion: uniquely identifies computers in a LAN
- Most operating systems today support IPv6
  - Not many networks have adopted IPv6

The IPv6 Protocol

- Addresses consist of 8 groups of 4-digit hex numbers separated by :

  `fed1:0db8:85a3:08d3:1319:8a2e:0370:7334`

- If 1 or more groups of 4 digits is 0000, those groups may be omitted, leaving ::
- IPv6 site-local addresses may be routed within a site but not off-site
- Begin with the hexadecimal number **`fec`**, **`fed`**, **`fee`**, or **`fef`**

Proxy Servers and NAT Routers

- Computers or hardware devices that have an IP address and access to a network
  - Used by other computers to obtain network resources on their behalf
  - Allows computers behind different NAT routers or proxy servers to have the same IPv4 address
- Due to usage of proxy servers / NAT
  - Available IPv4 addresses has remained high and slowed adoption of IPv6

Configuring a Network Interface

- Network device is not located in /dev directory like other devices
  - Character vs Block devices
- Why wouldn't you treat a network device as a block device?
  - Normal file operations (read, write) do not make sense when applied to network interfaces
  - Block drivers operate only in response to request from the kernel

Configuring a Network Interface

- Network devices receive packets asynchronously from the outside
- Block driver is asked to send a buffer toward the kernel
- Network device asks to push incoming packets towards the kernel
  - Kernel interface for network drivers designed for this different mode operation

Configuring a Network Interface

- Network drivers must be prepared to support different administrative tasks
  - Setting IP information
  - Modifying RX/TX parameters
  - Maintaining traffic and error stats
- Linux network driver is a module loaded in the kernel

Configuring a Network Interface

- **`lsmod`**
  - Displays a list of currently loaded modules
- **`lshw`**
  - Displays hardware detected by the system
  - **`lshw -C network`** displays network information such as the driver loaded
  - Can also see driver as a symbolic link for /sys/class/net/$interface/device/driver
- **`modinfo`**
  - Displays detailed information about specific module

Configuring a Network Interface

```
[root@localhost ~]# ip addr show dev ens33
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 00:0c:29:ee:9d:5d brd ff:ff:ff:ff:ff:ff
    inet 192.168.2.187/24 brd 192.168.2.255 scope global dynamic ens33
       valid_lft 76633sec preferred_lft 76633sec
    inet6 fe80::86c1:708f:127d:32e8/64 scope link
       valid_lft forever preferred_lft forever
[root@localhost ~]# ll /sys/class/net/ens33/device/driver
lrwxrwxrwx. 1 root root 0 Nov  7 20:21 /sys/class/net/ens33/device/driver -> ../../../../bus/pci/drivers/e1000
[root@localhost ~]# lshw -C network
  *-network
       description: Ethernet interface
       product: 82545EM Gigabit Ethernet Controller (Copper)
       vendor: Intel Corporation
       physical id: 1
       bus info: pci@0000:02:01.0
       logical name: ens33
       version: 01
       serial: 00:0c:29:ee:9d:5d
       size: 1Gbit/s
       capacity: 1Gbit/s
       width: 64 bits
       clock: 66MHz
       capabilities: pm pcix bus_master cap_list rom ethernet physical logical tp 10bt 10bt-fd 100bt 100bt-fd 1000bt-fd autonegotiation
       configuration: autonegotiation=on broadcast=yes driver=e1000 driverversion=7.3.21-k8-NAPI duplex=full ip=192.168.2.187 latency=0 lin
k=yes mingnt=255 multicast=yes port=twisted pair speed=1Gbit/s
       resources: irq:19 memory:fd5c0000-fd5dffff memory:fdff0000-fdffffff ioport:2000(size=64) memory:fd500000-fd50ffff
[root@localhost ~]# lsmod | grep e1000
e1000                  135168  0
[root@localhost ~]# modinfo e1000
filename:       /lib/modules/4.5.5-300.fc24.x86_64/kernel/drivers/net/ethernet/intel/e1000/e1000.ko.xz
version:        7.3.21-k8-NAPI
license:        GPL
description:    Intel(R) PRO/1000 Network Driver
author:         Intel Corporation, <linux.nics@intel.com>
srcversion:     12EE66F852DF364F1CA5DEF
```

Configuring a Network Interface

- **`insmod`** and **`modprobe`**
  - Loads kernel objects into the Linux kernel
  - Can be used to load NIC drivers
- **`rmmod`**
  - Removes module from kernel
- Older kernels loaded from entries in **`/etc/modprobe.conf`** or **`/etc/modules.conf`**

Configuring a Network Interface

- **`arp`**
  - Displays the MAC addresses that the system currently has stored in the ARP table
  - Can configure static entry
- **`arping`**
  - Command similar to **`ping`** but works at layer 2
  - Not every device will respond to ICMP, but if it has an IP on a network, it will respond to ARP
  - Specify the interface to ARP from using **`-I`** option
  - Can detect duplicate IP addresses on a network

Configuring a Network Interface

```
[root@itmo456 ~ ]# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
192.168.2.100            ether   00:60:e0:51:56:47   C                     eno16777736
[root@itmo456 ~ ]# arping -I eno16777736 192.168.2.100
ARPING 192.168.2.100 from 192.168.2.158 eno16777736
Unicast reply from 192.168.2.100 [00:60:E0:51:56:47]  0.783ms
Unicast reply from 192.168.2.100 [00:60:E0:51:56:47]  0.850ms
Unicast reply from 192.168.2.100 [00:60:E0:51:56:47]  0.831ms
^CSent 3 probes (1 broadcast(s))
Received 3 response(s)
[root@itmo456 ~ ]# arping -I eno16777736 192.168.2.5
ARPING 192.168.2.5 from 192.168.2.158 eno16777736
Unicast reply from 192.168.2.5 [E0:CB:4E:46:6F:EB]  0.855ms
Unicast reply from 192.168.2.5 [E0:CB:4E:46:6F:EB]  1.072ms
Unicast reply from 192.168.2.5 [E0:CB:4E:46:6F:EB]  0.991ms
^CSent 3 probes (1 broadcast(s))
Received 3 response(s)
[root@itmo456 ~ ]# arping -I eno16777736 192.168.2.10
ARPING 192.168.2.10 from 192.168.2.158 eno16777736
Unicast reply from 192.168.2.10 [00:0C:29:A1:68:49]  0.721ms
Unicast reply from 192.168.2.10 [00:22:2D:C5:DC:D8]  4.762ms
Unicast reply from 192.168.2.10 [00:22:2D:C5:DC:D8]  4.762ms
^CSent 2 probes (1 broadcast(s))
Received 3 response(s)
```

Configuring a Network Interface

- **`ifconfig`**
  - Assign TCP/IP configuration to a NIC
  - Also used without arguments to view configuration of all network interfaces in computer
- **`dhclient`**
  - Receive TCP/IP configuration from DHCP or Boot Protocol (BOOTP) server
- Automatic private IP addressing (APIPA)
  - Automatic assignment of IP address in the absence of DHCP and BOOTP
    - assigns a class B IP address from 169.254.0.0 to 169.254.255.255 to the client when a DHCP server is unavailable.

Configuring a Network Interface

```
[root@itm456 ~]# ifconfig eno16777736
eno16777736: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.75.132  netmask 255.255.255.0  broadcast 192.168.75.255
        inet6 fe80::20c:29ff:feab:afd9  prefixlen 64  scopeid 0x20<link>
        ether 00:0c:29:ab:af:d9  txqueuelen 1000  (Ethernet)
        RX packets 2451  bytes 489951 (478.4 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 1767  bytes 250283 (244.4 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
        device interrupt 19  base 0x2000

[root@itm456 ~]# ifconfig eno16777736 192.168.75.50 netmask 255.255.255.0
[root@itm456 ~]# ifconfig eno16777736
eno16777736: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.75.50  netmask 255.255.255.0  broadcast 192.168.75.255
        inet6 fe80::20c:29ff:feab:afd9  prefixlen 64  scopeid 0x20<link>
        ether 00:0c:29:ab:af:d9  txqueuelen 1000  (Ethernet)
        RX packets 2466  bytes 492761 (481.2 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 1777  bytes 252666 (246.7 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
        device interrupt 19  base 0x2000
```

Configuring a Network Interface

- Many desktop/laptop systems typically connect to many different NICs, both wired and wireless
- Network Manager daemon
  - Allows users to quickly connect to wired and wireless networks from desktop environments

Configuring a Network Interface

- If your network has IPv6-configured routers, an IPv6 address is automatically assigned to each NIC
- NICs use Internet Control Message Protocol version 6 (ICMPv6) router discovery messages to probe the network for IPv6 configuration information

Configuring a Network Interface

- **`/etc/sysconfig/network-scripts`**
  - Stores NIC configuration
  - GUI network management tools just rewrite these files
  - Files are **`ifcfg-<interface>`**

Configuring a NIC Interface DHCP

```
[root@itm456 ~]# cat /etc/sysconfig/network-scripts/ifcfg-eno16777736
TYPE="Ethernet"
BOOTPROTO="dhcp"
DEFROUTE="yes"
IPV4_FAILURE_FATAL="no"
IPV6INIT="yes"
IPV6_AUTOCONF="yes"
IPV6_DEFROUTE="yes"
IPV6_PEERDNS="yes"
IPV6_PEERROUTES="yes"
IPV6_FAILURE_FATAL="no"
NAME="eno16777736"
UUID="3fcb1940-cf23-4d9e-a468-d5d5858c18f9"
ONBOOT="yes"
HWADDR="00:0C:29:7B:03:51"
PEERDNS="yes"
PEERROUTES="yes"
```

Configuring a NIC Interface Static

```
[root@itm456 ~]# cat /etc/sysconfig/network-scripts/ifcfg-eno16777736
TYPE="Ethernet"
BOOTPROTO=none
DNS1="8.8.8.8"
DEFROUTE="yes"
IPV4_FAILURE_FATAL="no"
IPV6INIT="yes"
IPV6_AUTOCONF="yes"
IPV6_DEFROUTE="yes"
IPV6_FAILURE_FATAL="no"
NAME="eno16777736"
UUID="3fcb1940-cf23-4d9e-a468-d5d5858c18f9"
ONBOOT="yes"
HWADDR=00:0C:29:AB:AF:D9
IPADDR=192.168.75.75
PREFIX=24
GATEWAY=192.168.75.1
IPV6_PEERDNS=yes
IPV6_PEERROUTES=yes
[root@itm456 ~]# ifconfig eno16777736
eno16777736: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.75.75  netmask 255.255.255.0  broadcast 192.168.75.255
        inet6 fe80::20c:29ff:feab:afd9  prefixlen 64  scopeid 0x20<link>
        ether 00:0c:29:ab:af:d9  txqueuelen 1000  (Ethernet)
        RX packets 2758  bytes 519814 (507.6 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 1909  bytes 267832 (261.5 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

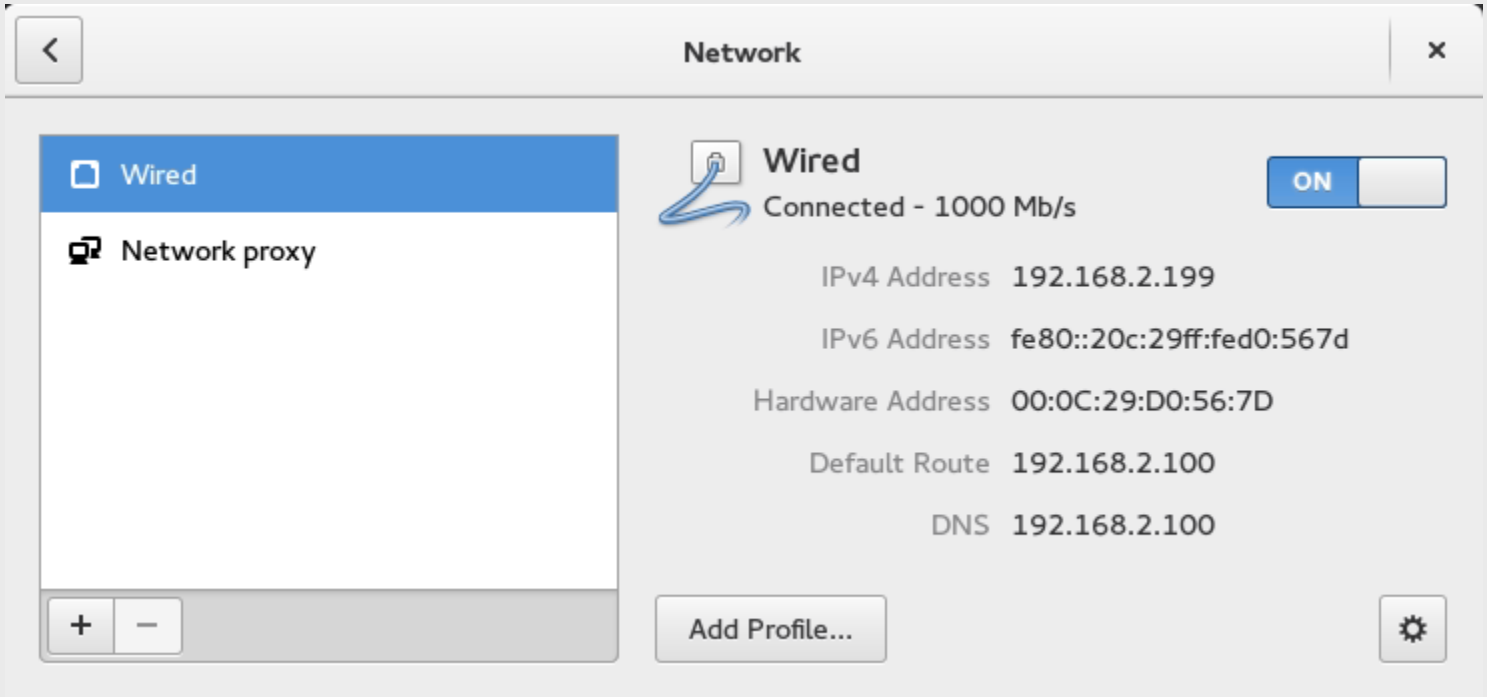
Configuring a Network Interface

- **`ifup`**
  - Configures NIC using **`/etc/sysconfig/network-scripts/ifcfg-`*`interface`*** file
- **`ifdown`**
  - Unconfigures a NIC
- **p**acket **in**ternet **g**roper (**`ping`**)
  - Used to verify routing on a network
  - **`-c`** option limits the number of **`ping`** packets sent

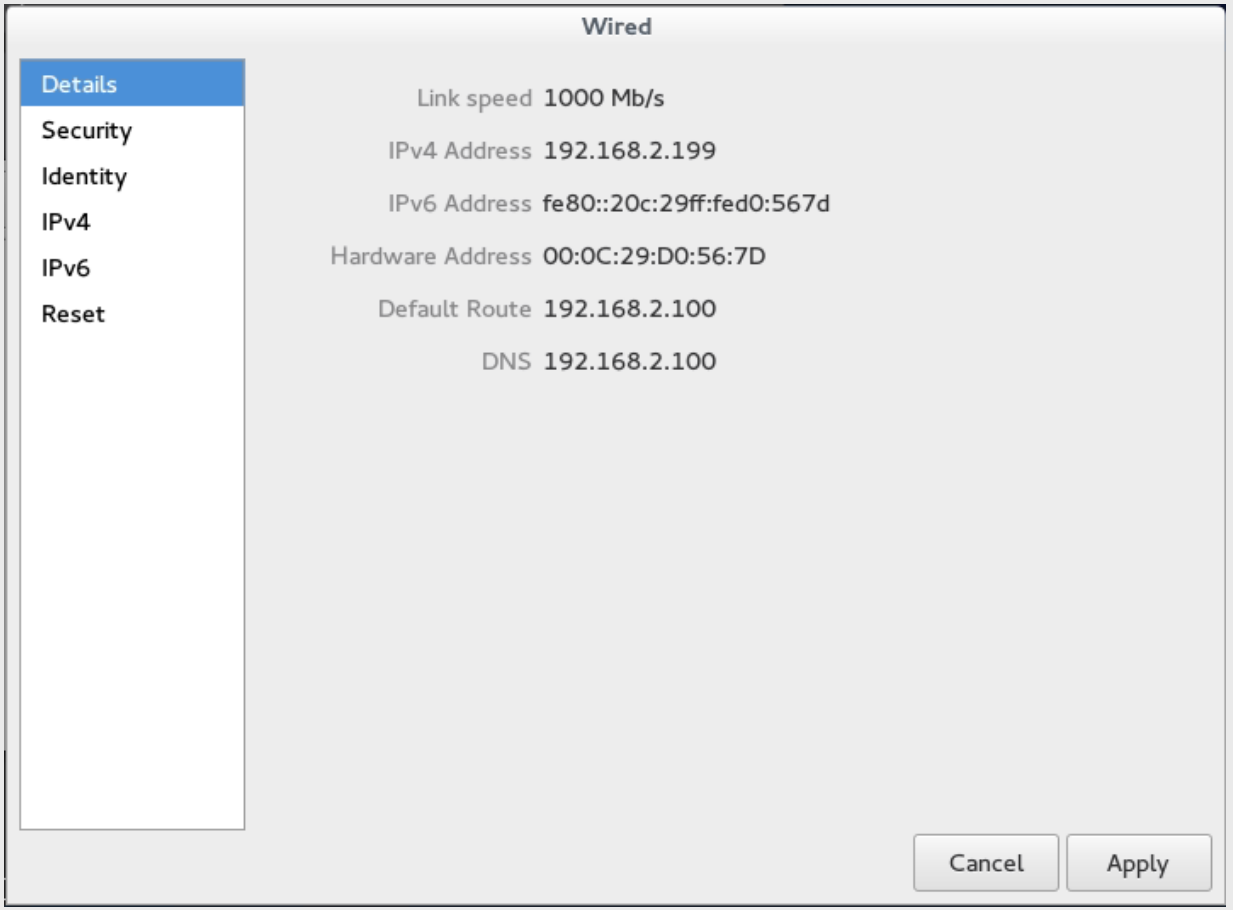
Configuring a Network Interface

- NetworkManager - GUI-based tool
  - Start by clicking the desktop NetworkManager icon
  - Click on the Configure button in window to reach additional settings
  - Window tabs display different configuration information
  - Configuration tabs include:
    - Wired
    - 802.1x Security
    - IPv4 Settings
    - IPV6 Settings

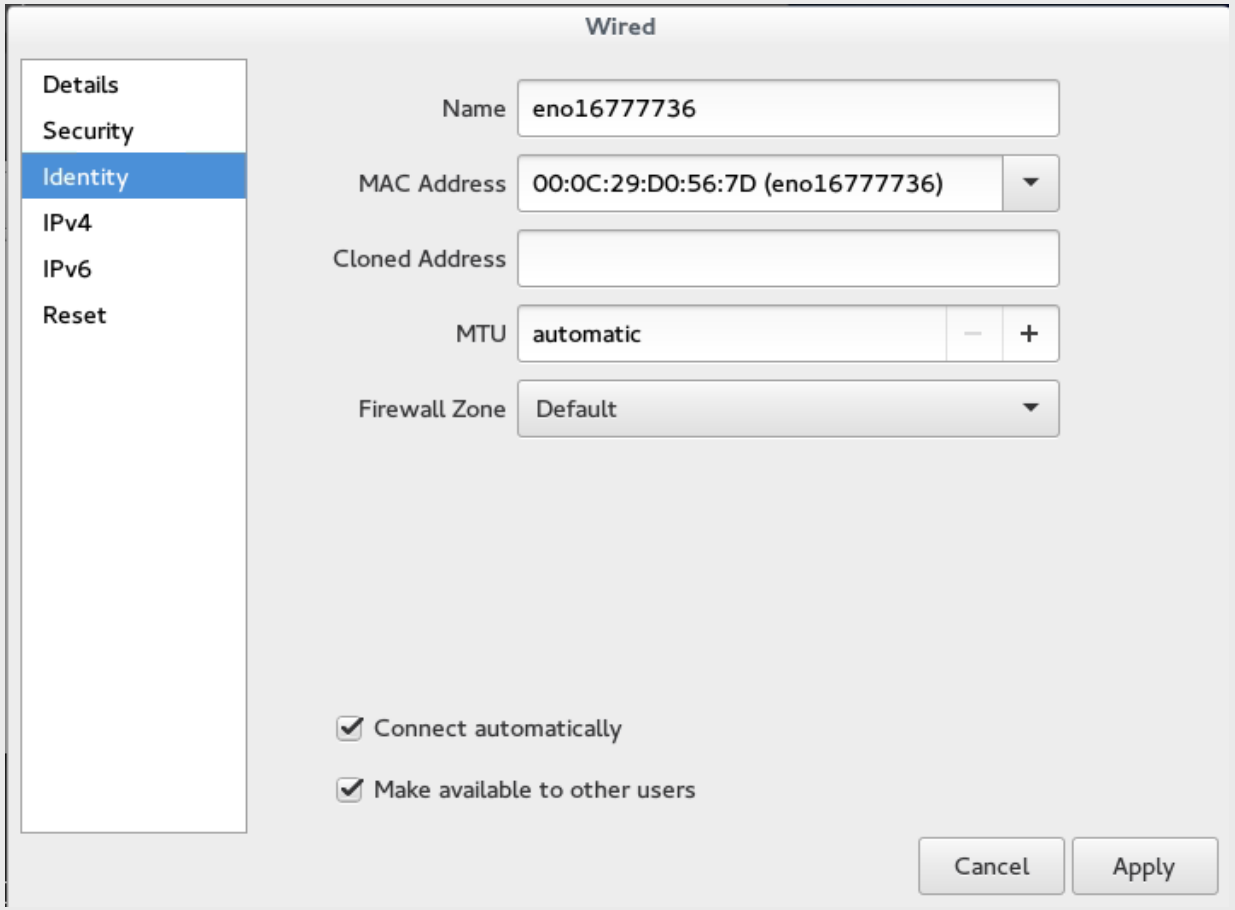
Network Manager



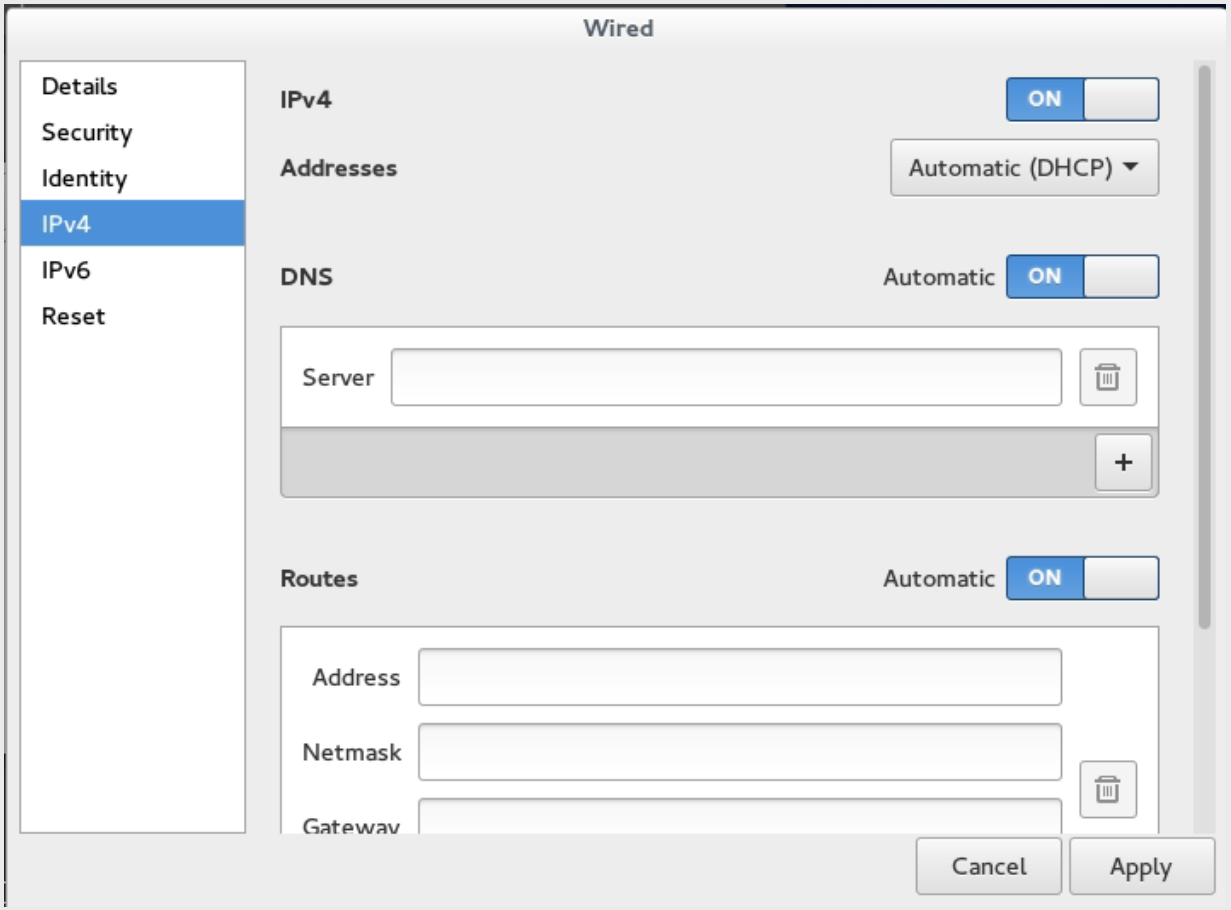
Network Manager



Network Manager



Editing Network Settings



Configuring Networking for Desktops

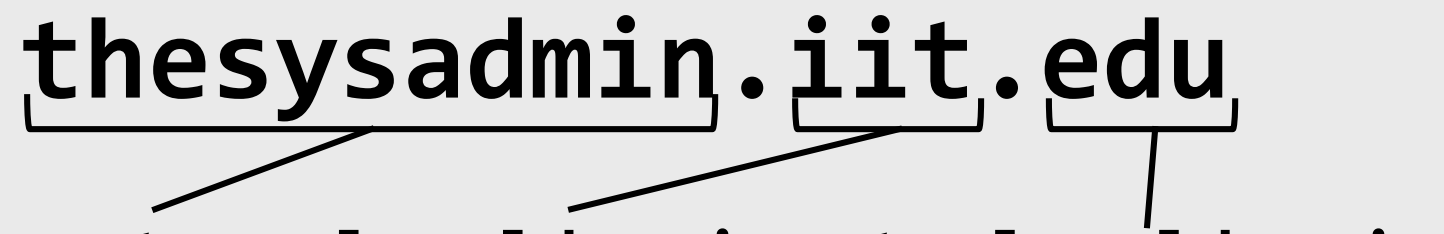
- Minimum Requirements
  - Network interface
  - An IP address
  - An assigned DNS server
  - A route to the Internet.
- Network interface can be
  - wired or
  - wireless

Automatic Network Connection

- Activate network interfaces
- Request DHCP service
- Obtain response from DHCP server, which includes:
  - IP Address
  - Subnet mask
  - Lease time
  - Domain name server
  - Default gateway
- Update of local network settings, as needed

Name Resolution

- Hostnames
  - User-friendly computer name
- Fully Qualified Domain Name (FQDN)
  - Hostname that follows DNS convention
  - Normally three parts: systemname.localdomain.topleveldomain



Name Resolution

- Domain Name Space
  - Hierarchical namespace for host names
  - Top level: **.com**, **.net**, **.org**, **.edu**,**.us**
    - Now for $185,000 can be anything
  - Second level: **microsoft.com**, **iit.edu**
  - Local system name: **ftp.microsoft.com**, **www.iit.edu**
    - Can go down 127 levels: **www.itmo456.itm.sat.iit.edu**
- **hostname** command
  - View or set a computer's host name

Name Resolution

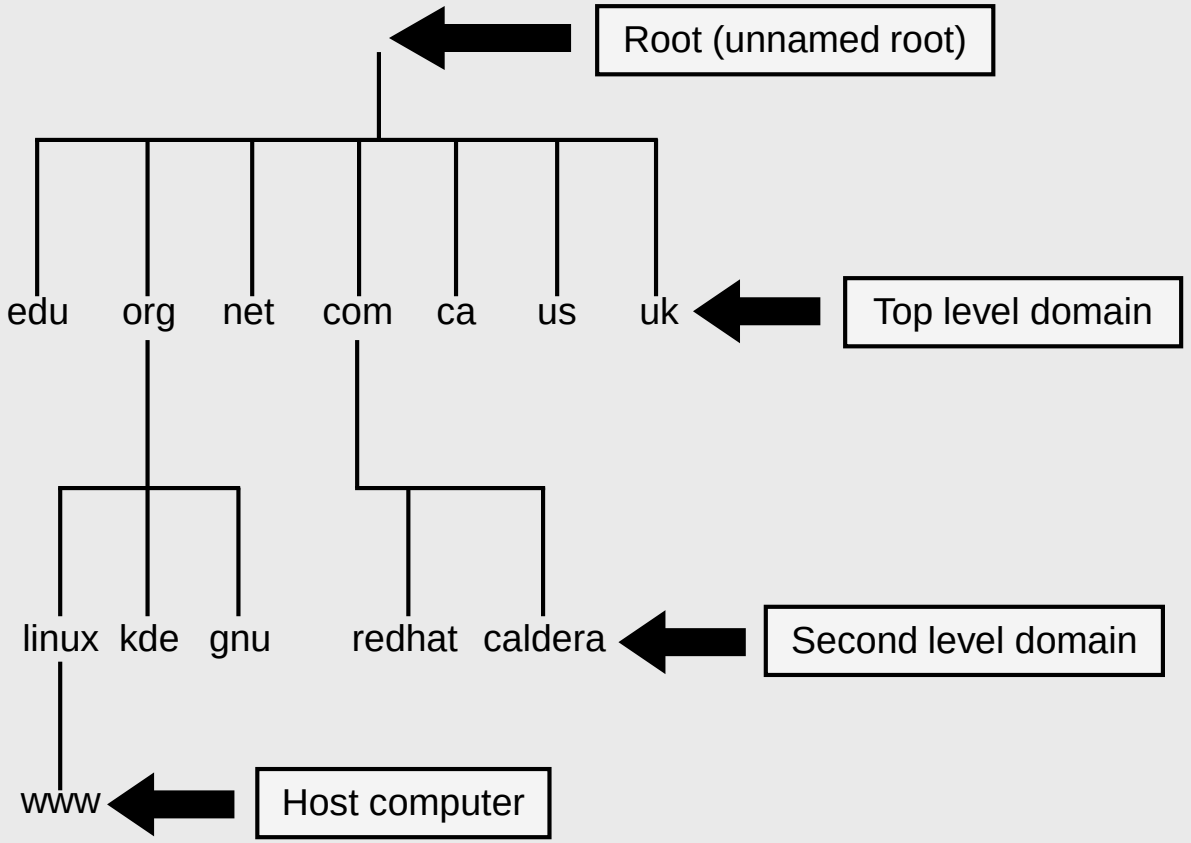


Figure 12-6: The Domain Name Space

Name Resolution

- TCP/IP cannot identify computers via hostnames
  - Must map hostnames to IP addresses
  - Through entries in **`/etc/hosts`** file, or…
  - use DNS: Domain Name System

Name Resolution

- Domain Name System (DNS)
  - Used to resolve FQDNs to the appropriate IP address
  - Translate domain names meaningful to humans into numerical IP addresses
  - System designates authoritative name servers for each top-level domain like **`.com`**, **`.net`**, **`.org`**, **`.us`**, etc.
  - Authoritative name servers resolve through 13 **root name servers**

Name Resolution

- ISPs list FQDNs in second-level DNS servers on the Internet
  - Applications request IP addresses associated with FQDN
  - Root nameserver resolves authoritative nameserver
  - Authoritative nameserver resolves second-level nameserver which resolves FQDN
  - Configure by specifying IP address of the DNS server in **`/etc/resolv.conf`** file

Name Resolution

- **`nslookup`**
  - Query Internet name servers interactively
    - **`nslookup google.com`**
      - Will use assigned DNS servers
    - **`nslookup google.com 8.8.8.8`**
      - Can specify alternate DNS server
  - Local hostnames located in **`/etc/hosts`**
  - Name servers located in **`/etc/sysconfig/network-scripts/ifcfg-`*`interface`*** *and* **`/etc/resolv.conf`**

Routing

- Route table
  - Indicates which networks are connected to network interfaces
- **`route`**
  - Displays the route table
  - Default route located in **`/etc/sysconfig/network-scripts/ifcfg-`*`interface`*** file
  - Static routes located in **`/etc/sysconfig/network-scripts/route-`*`interface`*** file

Routing

- Multihomed hosts
  - Computers with multiple network interfaces
- IP forwarding
  - Forwarding TCP/IP packets between networks
- Routing
  - Forwarding data packets between networks

Routing

- Enabling routing:
  - Place number 1 in:
    - **/proc/sys/net/ipv4/ip_forward** – IPv4
    - **/proc/sys/net/ipv6/conf/all/forwarding** – IPv6
- To enable routing at every boot:
  - Edit **/etc/sysctl.conf** file to include:
    - **net.ipv4.ip_forward = 1** for IPv4
    - **net.ipv6.conf.default.forwarding = 1** for IPv6

Routing

- Large networks may have several routers
  - Packet may travel through several routers
  - May require adding entries in the router table
- **`route add <route>`**
  - Add entries to route table
- **`route del <route>`**
  - Remove entries from route table

Routing

- **`route`**
  - Can add routes and remove routes
    - **`route add -net 10.20.40.0/24 gw 192.168.2.200 dev eno16777736`**
    - **`route add -host 10.20.40.50/32 gw 192.168.2.200 dev eno16777736`**
    - **`route del -net 10.20.40.0/24 gw 192.168.2.200 dev eno16777736`**

Network Gateway

```
[root@itm456 ~]# route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0         192.168.2.100   0.0.0.0         UG    1024   0        0 eno16777736
10.20.40.0      192.168.2.200   255.255.255.0   UG    0      0        0 eno16777736
10.20.40.50     192.168.2.200   255.255.255.255 UGH   0      0        0 eno16777736
192.168.2.0     0.0.0.0         255.255.255.0   U     0      0        0 eno16777736
[root@itm456 ~]# route del -net 10.20.40.0/24 gw 192.168.2.200 dev eno16777736
[root@itm456 ~]# route del -host 10.20.40.50/32 gw 192.168.2.200 dev eno16777736
[root@itm456 ~]# route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0         192.168.2.100   0.0.0.0         UG    1024   0        0 eno16777736
192.168.2.0     0.0.0.0         255.255.255.0   U     0      0        0 eno16777736
```

Net-tools Future

- Net-tools suite has been deprecated and replaced with the iproute suite
- Net-tools consists of the following commands:
  - arp
  - hostname
  - ifconfig
  - netstat
  - rarp
  - route
  - mii-tool

Net-tools Future

- Net-tools maintainers stated that the suite of tools does not support modern features of the kernel
  - Also has not been updated in several years
- Man pages reference net-tools commands to be obsolete and mention what commands replace them
- https://lists.debian.org/debian-devel/2009/03/msg00780.html

Net-tools Future

```
IFCONFIG(8)          Linux System Administrator's Manual          IFCONFIG(8)

NAME
       ifconfig - configure a network interface

SYNOPSIS
       ifconfig [-v] [-a] [-s] [interface]
       ifconfig [-v] interface [aftype] options | address ...

NOTE
       This  program  is obsolete!  For replacement check ip addr and ip link.
       For statistics use ip -s link.

DESCRIPTION
       Ifconfig is used to configure the kernel-resident  network  interfaces.
       It is used at boot time to set up interfaces as necessary.  After that,
       it is usually only needed when  debugging  or  when  system  tuning  is
       needed.

       If  no  arguments  are  given, ifconfig displays the status of the cur-
       rently active interfaces.  If a single interface argument is given,  it
       displays  the  status of the given interface only; if a single -a argu-
       ment is given, it displays the status of  all  interfaces,  even  those
 Manual page ifconfig(8) line 1 (press h for help or q to quit)
```

`iproute` Suite

- iproute suite introduced around kernel 2.2
  - Better and consistent interfaces
  - More powerful that net-suite tools
  - Almost 20 years old
    - Vetted and tested

Querying or Controlling Network Driver

- **`ethtool`**
  - Can query interface information
    - Speed, duplex, auto-neg, supported modes
  - Can modify interface settings
  - Determine if link is connected

Querying or Controlling Network Driver

```
[root@localhost ~]# ethtool ens33
Settings for ens33:
        Supported ports: [ TP ]
        Supported link modes:   10baseT/Half 10baseT/Full
                                100baseT/Half 100baseT/Full
                                1000baseT/Full
        Supported pause frame use: No
        Supports auto-negotiation: Yes
        Advertised link modes:  10baseT/Half 10baseT/Full
                                100baseT/Half 100baseT/Full
                                1000baseT/Full
        Advertised pause frame use: No
        Advertised auto-negotiation: Yes
        Speed: 1000Mb/s
        Duplex: Full
        Port: Twisted Pair
        PHYAD: 0
        Transceiver: internal
        Auto-negotiation: on
        MDI-X: off (auto)
        Supports Wake-on: d
        Wake-on: d
        Current message level: 0x00000007 (7)
                               drv probe link
        Link detected: yes
```

Configuring a NIC Interface

- **`ip`**
  - Can query ARP, IP, and route information
  - Can be used to assign IP address to an interface
  - Can be used to manipulate route table
  - Can be used to manipulate ARP table
  - Can use shorthand
    - ip addr rather than ip address
  - Similar to ifconfig and route commands
  - Need to be root to use command

Configuring a NIC Interface


```
[root@localhost ~]# ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group defaul
t qlen 1
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP gro
up default qlen 1000
    link/ether 00:0c:29:0d:b8:73 brd ff:ff:ff:ff:ff:ff
    inet 192.168.13.142/24 brd 192.168.13.255 scope global dynamic ens33
       valid_lft 1619sec preferred_lft 1619sec
    inet6 fe80::66af:4e33:7eef:3fc7/64 scope link
       valid_lft forever preferred_lft forever
[root@localhost ~]# ip addr show dev ens33
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP gro
up default qlen 1000
    link/ether 00:0c:29:0d:b8:73 brd ff:ff:ff:ff:ff:ff
    inet 192.168.13.142/24 brd 192.168.13.255 scope global dynamic ens33
       valid_lft 1605sec preferred_lft 1605sec
    inet6 fe80::66af:4e33:7eef:3fc7/64 scope link
       valid lft forever preferred lft forever
```

Configuring a NIC Interface

```
[root@localhost ~]# ip addr add 192.168.13.175/24 dev ens33
[root@localhost ~]# ip addr show dev ens33
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP gro
up default qlen 1000
    link/ether 00:0c:29:0d:b8:73 brd ff:ff:ff:ff:ff:ff
    inet 192.168.13.142/24 brd 192.168.13.255 scope global dynamic ens33
       valid_lft 1528sec preferred_lft 1528sec
    inet 192.168.13.175/24 scope global secondary ens33
       valid_lft forever preferred_lft forever
    inet6 fe80::66af:4e33:7eef:3fc7/64 scope link
       valid_lft forever preferred_lft forever
[root@localhost ~]# ip addr del 192.168.13.175/24 dev ens33
[root@localhost ~]# ip addr show dev ens33
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP gro
up default qlen 1000
    link/ether 00:0c:29:0d:b8:73 brd ff:ff:ff:ff:ff:ff
    inet 192.168.13.142/24 brd 192.168.13.255 scope global dynamic ens33
       valid_lft 1369sec preferred_lft 1369sec
    inet6 fe80::66af:4e33:7eef:3fc7/64 scope link
       valid lft forever preferred lft forever
```

Configuring a NIC Interface

```
[root@localhost ~]# ip link set ens33 down
[root@localhost ~]# ip addr show dev ens33
2: ens33: <BROADCAST,MULTICAST> mtu 1500 qdisc fq_codel state DOWN group default
 qlen 1000
    link/ether 00:0c:29:0d:b8:73 brd ff:ff:ff:ff:ff:ff
    inet 192.168.13.142/24 brd 192.168.13.255 scope global dynamic ens33
       valid_lft 1591sec preferred_lft 1591sec
[root@localhost ~]# ip link set ens33 up
[root@localhost ~]# ip addr show dev ens33
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP gro
up default qlen 1000
    link/ether 00:0c:29:0d:b8:73 brd ff:ff:ff:ff:ff:ff
    inet 192.168.13.142/24 brd 192.168.13.255 scope global dynamic ens33
       valid_lft 1798sec preferred_lft 1798sec
    inet6 fe80::66af:4e33:7eef:3fc7/64 scope link
       valid_lft forever preferred_lft forever
[root@localhost ~]# ip neigh
192.168.13.2 dev ens33 lladdr 00:50:56:f5:9e:8d REACHABLE
[root@localhost ~]# ip -s neigh
192.168.13.2 dev ens33 lladdr 00:50:56:f5:9e:8d ref 1 used 8/8/8 probes 4 REACHA
BLE
```

ITMO456

Routing

- **`ip route`**
  - Similar to **`route`** command
  - Can search routing table
    - **`ip route get 10.20.30.50`**
  - Can also add/remove routes
    - **`ip route add 192.0.2.1 via 192.168.2.100 dev eno16777736`**
    - **`ip route add 192.168.6.0/24 via 192.168.2.100 dev eno16777736`**
    - **`ip route del 192.168.6.0/24 via 192.168.2.100 dev eno16777736`**

Network Gateway

```
[root@localhost ~]# ip route
default via 192.168.13.2 dev ens33  proto static  metric 100
192.168.13.0/24 dev ens33  proto kernel  scope link  src 192.168.13.142  metric
100
[root@localhost ~]# ip route add 192.168.100.0/24 via 192.168.13.2 dev ens33
[root@localhost ~]# ip route
default via 192.168.13.2 dev ens33  proto static  metric 100
192.168.13.0/24 dev ens33  proto kernel  scope link  src 192.168.13.142  metric
100
192.168.100.0/24 via 192.168.13.2 dev ens33
[root@localhost ~]# ip route get 192.168.100.10
192.168.100.10 via 192.168.13.2 dev ens33  src 192.168.13.142
    cache
[root@localhost ~]# ip route del 192.168.100.0/24 via 192.168.13.2 dev ens33
[root@localhost ~]# ip route
default via 192.168.13.2 dev ens33  proto static  metric 100
192.168.13.0/24 dev ens33  proto kernel  scope link  src 192.168.13.142  metric
100
```

Routing

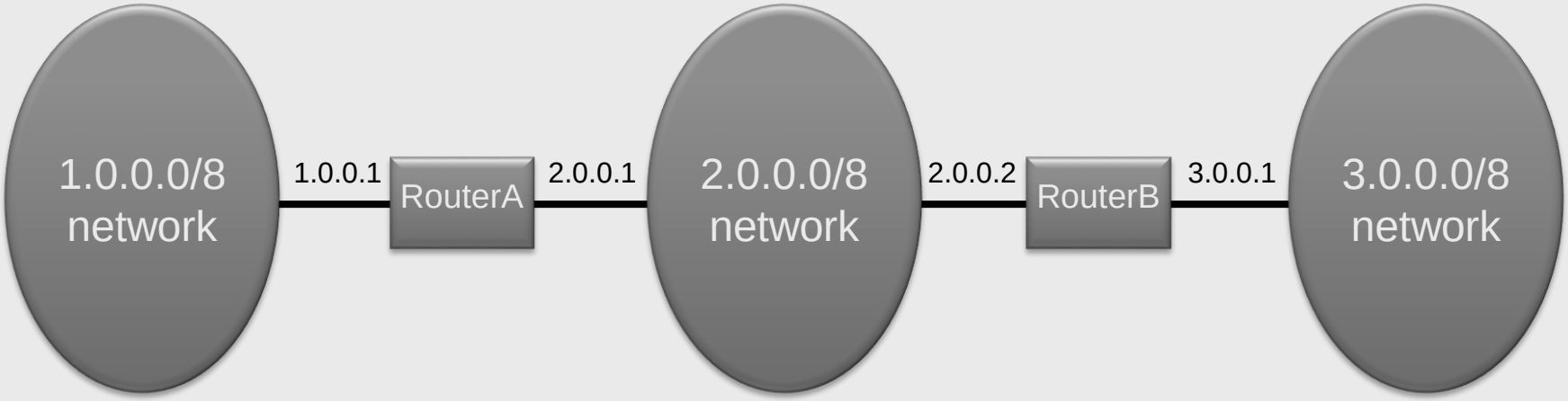


Figure 12-7: A sample routed network

Routing

- Most routers configured with a default gateway
  - For packets addressed to destinations not in route table
- **`traceroute`**
  - Troubleshoot routing
  - Displays routers between current and remote computer

Routing

```
[root@itmo456 ~ ]# traceroute google.com
traceroute to google.com (74.125.225.4), 30 hops max, 60 byte packets
 1  spidey-fw.spideyman.net (192.168.2.100)  0.245 ms  0.310 ms  0.255 ms
 2  96.120.26.237 (96.120.26.237)  15.991 ms  21.126 ms  22.078 ms
 3  te-0-3-0-1-sur03.romeoville.il.chicago.comcast.net (68.85.179.53)  22.077 ms  22.135 ms  22.080 ms
 4  te-0-6-0-1-ar01.elmhurst.il.chicago.comcast.net (68.86.187.145)  25.117 ms te-0-6-0-0-ar01.elmhurst.il.chicago.
comcast.net (68.87.230.105)  25.183 ms te-2-10-0-12-ar01.elmhurst.il.chicago.comcast.net (68.86.187.161)  26.119 ms
 5  he-0-15-0-0-ar01.area4.il.chicago.comcast.net (68.86.189.153)  24.888 ms * *
 6  be-33491-cr02.350ecermak.il.ibone.comcast.net (68.86.91.165)  25.562 ms  25.613 ms  25.539 ms
 7  hu-0-11-0-1-pe04.350ecermak.il.ibone.comcast.net (68.86.83.62)  23.335 ms  14.381 ms  17.416 ms
 8  as15169-3-c.350ecermak.il.ibone.comcast.net (173.167.57.210)  16.216 ms  15.643 ms  14.523 ms
 9  209.85.244.1 (209.85.244.1)  15.644 ms  19.140 ms  14.905 ms
10  72.14.237.109 (72.14.237.109)  15.786 ms  17.592 ms  18.515 ms
11  ord08s12-in-f4.1e100.net (74.125.225.4)  14.343 ms  15.261 ms  21.826 ms
[root@itmo456 ~ ]#
[root@itmo456 ~ ]# traceroute -I google.com
traceroute to google.com (74.125.225.4), 30 hops max, 60 byte packets
 1  spidey-fw.spideyman.net (192.168.2.100)  0.254 ms  0.328 ms  0.305 ms
 2  96.120.26.237 (96.120.26.237)  11.312 ms  12.220 ms  12.375 ms
 3  te-0-3-0-1-sur03.romeoville.il.chicago.comcast.net (68.85.179.53)  12.504 ms  12.480 ms  12.449 ms
 4  te-2-10-0-13-ar01.elmhurst.il.chicago.comcast.net (68.87.235.53)  13.008 ms te-1-1-0-6-ar01.elmhurst.il.chicago
.comcast.net (162.151.44.141)  16.837 ms te-3-12-0-8-ar01.elmhurst.il.chicago.comcast.net (162.151.47.149)  15.658
ms
 5  he-0-15-0-0-ar01.area4.il.chicago.comcast.net (68.86.189.153)  14.787 ms * *
 6  * * *
 7  * hu-0-11-0-1-pe04.350ecermak.il.ibone.comcast.net (68.86.83.62)  22.186 ms  22.828 ms
 8  as15169-3-c.350ecermak.il.ibone.comcast.net (173.167.57.210)  21.769 ms  20.925 ms  20.815 ms
 9  209.85.244.1 (209.85.244.1)  21.780 ms  16.304 ms  16.249 ms
10  72.14.237.109 (72.14.237.109)  15.277 ms  16.007 ms  16.018 ms
11  ord08s12-in-f4.1e100.net (74.125.225.4)  15.871 ms  14.491 ms  15.442 ms
```

Network Services

- Processes that provide some type of valuable service for client computers on network
- Must identify types and features of network services before they can be configured
- Often presented by daemon processes that listen to certain requests
  - Daemons identify packets to which they should respond using a port number

Identifying Network Services

- Port
  - Number uniquely identifying a network service
  - Ensure packets delivered to proper service
  - Range from 0 to 65535
    - Port 0 is reserved and cannot be used
- **`/etc/services`** file
  - Lists ports and associated protocol
- “Well-known” ports
  - Ports from 0 to 1023 used by common networking services

Identifying Network Services

Service	Port
FTP	TCP 20,21
Secure Shell (SSH; includes SFTP)	TCP 22
Telnet	TCP 23
SMTP	TCP 25
HTTP / HTTPS	TCP 80 / TCP 443
rlogin	TCP 513
DNS	TCP 53, UDP 53
Trivial FTP (TFTP)	UDP 69
POP3 / POP3S	TCP 110 / TCP 995
NNTP / NNTPS	TCP 119 / TCP 563
IMAP4 / IMAP4S	TCP 143 / TCP 993

Table 12-2: Common well-known ports

Verifying Connectivity

- **`tcpdump`**
  - Used to monitor network traffic in real-time
  - Can specify interface from **`ip addr`**
  - Uses BPF
    - http://en.wikipedia.org/wiki/Berkeley_Packet_Filter
    - http://biot.com/capstats/bpf.html
  - Must install
    - **`yum install tcpdump`**
  - Common options
    - -n does not resolve IP to name
      - Use another n for port
    - -i specifies interface
    - **`tcpdump -nni eno16777736 host 192.168.2.100`**

Verifying Connectivity

```
[root@itmo456 ~ ]# tcpdump -nni eno16777736 port 4444
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eno16777736, link-type EN10MB (Ethernet), capture size 65535 bytes
11:19:02.769802 IP 192.168.2.158.45627 > 192.168.2.100.4444: Flags [S], seq 3096690555, win 29200, options [mss 146
0,sackOK,TS val 1237141 ecr 0,nop,wscale 7], length 0
11:19:02.770209 IP 192.168.2.100.4444 > 192.168.2.158.45627: Flags [S.], seq 3675761185, ack 3096690556, win 65228,
 options [mss 1460,nop,wscale 7,sackOK,TS val 322910489 ecr 1237141], length 0
11:19:02.770301 IP 192.168.2.158.45627 > 192.168.2.100.4444: Flags [.], ack 1, win 229, options [nop,nop,TS val 123
7142 ecr 322910489], length 0
11:19:02.770530 IP 192.168.2.158.45627 > 192.168.2.100.4444: Flags [P.], seq 1:184, ack 1, win 229, options [nop,no
p,TS val 1237142 ecr 322910489], length 183
11:19:02.770924 IP 192.168.2.100.4444 > 192.168.2.158.45627: Flags [.], ack 184, win 518, options [nop,nop,TS val 3
22910489 ecr 1237142], length 0
11:19:02.826172 IP 192.168.2.100.4444 > 192.168.2.158.45627: Flags [.], seq 1:1449, ack 184, win 520, options [nop,
nop,TS val 322910545 ecr 1237142], length 1448
11:19:02.826266 IP 192.168.2.158.45627 > 192.168.2.100.4444: Flags [.], ack 1449, win 251, options [nop,nop,TS val
1237198 ecr 322910545], length 0
11:19:02.826308 IP 192.168.2.100.4444 > 192.168.2.158.45627: Flags [P.], seq 1449:1822, ack 184, win 520, options [
nop,nop,TS val 322910545 ecr 1237142], length 373
11:19:02.826321 IP 192.168.2.158.45627 > 192.168.2.100.4444: Flags [.], ack 1822, win 274, options [nop,nop,TS val
1237198 ecr 322910545], length 0
11:19:02.835725 IP 192.168.2.158.45627 > 192.168.2.100.4444: Flags [P.], seq 184:310, ack 1822, win 274, options [n
op,nop,TS val 1237207 ecr 322910545], length 126
11:19:02.836030 IP 192.168.2.158.45627 > 192.168.2.100.4444: Flags [P.], seq 310:341, ack 1822, win 274, options [n
op,nop,TS val 1237207 ecr 322910545], length 31
11:19:02.836088 IP 192.168.2.100.4444 > 192.168.2.158.45627: Flags [.], ack 310, win 519, options [nop,nop,TS val 3
22910555 ecr 1237207], length 0
11:19:02.836220 IP 192.168.2.158.45627 > 192.168.2.100.4444: Flags [F.], seq 341, ack 1822, win 274, options [nop,n
op,TS val 1237208 ecr 322910555], length 0
11:19:02.836268 IP 192.168.2.100.4444 > 192.168.2.158.45627: Flags [.], ack 341, win 520, options [nop,nop,TS val 3
22910555 ecr 1237207], length 0
```

ITMO456

Network Troubleshooting

- **nc**
  - NetCat (network concatenate)
  - Must install
    - **yum install nc**
  - Can test if network ports are open

```
[root@itm456 ~]# nc -v 127.0.0.1 22
Ncat: Version 6.47 ( http://nmap.org/ncat )
Ncat: Connected to 127.0.0.1:22.
SSH-2.0-OpenSSH_6.6.1
```

  - Can bind and listen on a specific port
    - Use the **-l** option
      - **nc -l 25**

Common Network Services

- Internet Super Daemon (**`xinetd`**)
  - Initializes and configures many networking services
- Standalone daemons
  - Provide network services directly
    - Log information themselves to subdirectories under **`/var/log`**
  - Configure themselves without assistance
  - **`chkconfig`** or **`ntsysv`** or **`rcconf`** utilities can configure most standalone daemons to start in various runlevels

Configuring **`xinetd`**

- Most services that can be launched from **`xinetd`** will provide explicit instructions for **`/etc/xinetd.conf`** entry

Network Services

- Ubuntu Server 14.0 installs the **`inetd`** daemon by default
- You can install **`xinetd`** from a software repository
- Fedora 24 systems do not contain the **`inetd`** or **`xinetd`** daemons by default
  - You can install **`xinetd`** from a software repository

Common Network Services

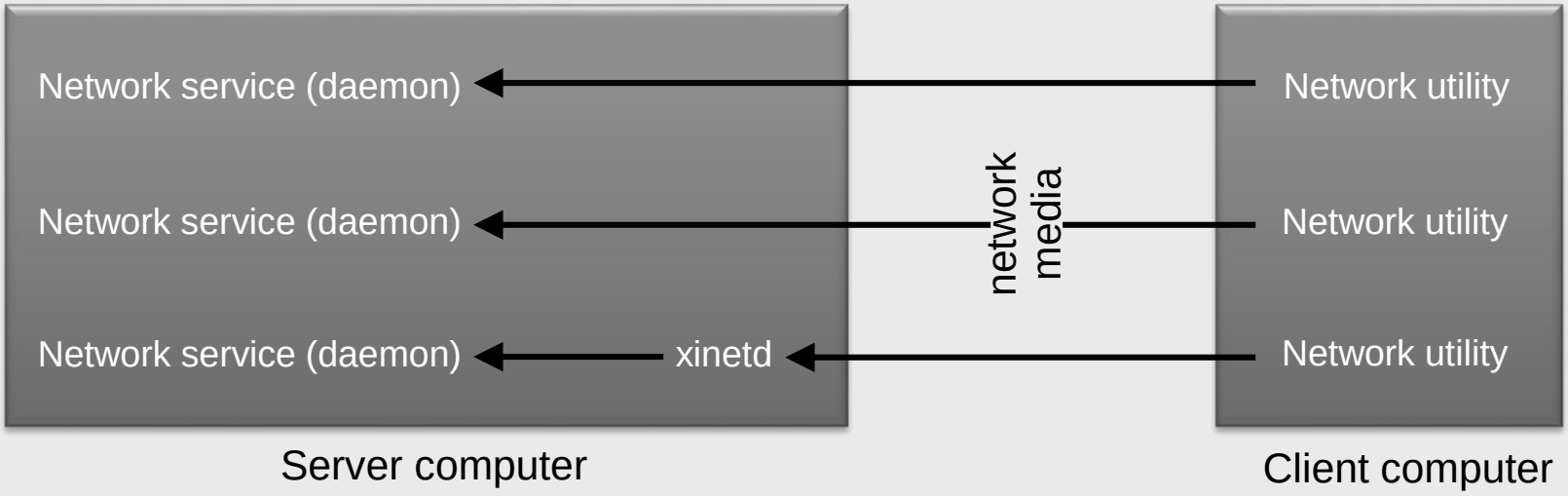


Figure 12-8: Interacting with network services

Remote Administration

- There are several ways to perform command-line and graphical administration of remote Linux servers:
  - Telnet
  - rsh
  - Secure Shell (SSH)
  - Virtual Network Computing (VNC)
  - Webmin

Remote Administration: Telnet

- **`telnet`**
  - Traditionally method to obtain a command-line shell on remote server
  - Receives host name or IP address of remote computer as argument
  - Easiest way to perform remote administration but don't use it!
  - Insecure and NOT installed by default
- Use regular commands and **`exit`** to kill remote BASH shell

Remote Commands

- Remote commands
  - Set of commands used to execute commands on remote systems
  - Not installed by default and insecure—so don't use!
  - **`yum install rsh-server`**
- r commands allow access to remote computers without a password, if remote computer has trusted access

Remote Commands

- **`rlogin`**
  - Open a shell from remote computer on network
- **`rcp`**
  - Copies files between computers
- **`rsh`**
  - Execute a command on a remote computer

Remote Commands

- Trusted access
  - Computers allowed to access a computer without providing a password
  - Does not apply to root user
  - Methods of setting up:
    - Add host names of computers to **`/etc/hosts.equiv`**
    - Create an **`.rhosts`** file in the home directory of each user who should get trusted access

Secure Shell (SSH)

- Secure Shell (SSH)
  - Encrypts information passing between computers
  - Secure replacement for r commands, telnet, and FTP
  - Used to perform command-line administration of remote systems
  - Can forward X11 to access GUI applications

Secure Shell (SSH)

- **`ssh`**
  - Connects to a remote computer running ssh daemon
  - Receives host name or IP address of target computer as argument
  - Accept RSA encryption fingerprint for target computer
  - Can be used to transfer files between computers using the **`sftp`** comand

Secure Shell (SSH)

- **`-X`** option to **`ssh`** can be used to tunnel X Windows information through the SSH connection if you are using **`ssh`** within a GUI environment
- By default, **`sshd`** uses secure challenge-response authentication ensuring that the password is not transmitted on the network
  - Can be changed to Kerberos authentication

Secure Shell (SSH)

- Main types of encryption supported by **`ssh`** daemon:
  - Triple Data Encryption Standard (3DES)
  - Advanced Encryption Standard (AES)
  - Blowfish
  - Carlisle Adams Stafford Tavares (CAST)
  - ARCfour

Virtual Network Computing (VNC)

- Graphical option for administering Linux remotely
  - Other computers run VNC client that connects to VNC server daemon installed on local computer to obtain a desktop environment
- Remote FrameBuffer (RFB)
  - Platform-independent protocol used to transfer graphics, mouse movements and keystrokes across network

Virtual Network Computing (VNC)

- **`vncpasswd`**
  - Configure password for VNC connection
- **`vncviewer`**
  - Connects to VNC server
- NOT secure
  - Can be tunneled over SSH

Virtual Network Computing (VNC)

- On Fedora, install a VNC server by running

  `yum install tigervnc-server`

  - Next, configure the VNC server by creating a VNC server configuration file that listens on a certain display number

Other Remote Access Solutions

- Web-based graphical configuration program that can manage most Linux system and server application configurations
  - Not included with most distributions but widely available
  - Encrypted when run over **SSL/TLS**
  - Will also give you shell access
- Included by default with Turnkey Linux VMs and ISOs

Other Remote Access Solutions

- NX
  - GUI access and runs over SSH (secure)
  - Servers
    - NoMachine (commercial), FreeNX, NeatNX
  - Clients
    - NoMachine (commercial), OpenNX
- XRDP
  - Linux implementation of Windows Remote Desktop Protocol

Summary

- A network is a collection of connected computers that share information
- A protocol is a set of rules that defines the format of information that is transmitted across a network
- Each computer on a TCP/IP network must have a valid IPv4 or IPv6 address

Summary

- IPv4 configuration of a network interface can be specified manually, obtained automatically from a DHCP or BOOTP server, or autoconfigured by the system
- IPv6 configuration of a network interface can be obtained from a router using ICMPv6, from a DHCP server, or autoconfigured by the system

Summary

- On a Fedora system, the **`/etc/sysconfig/network-scripts`** directory contains the configuration for NIC interfaces
- Host names are computer names that are easy for humans to remember; host names that follow the DNS are FQDNs
- Host names must be resolved to an IP address before network communication can take place

Summary

- Routers are devices that forward TCP/IP packets from one network to another; each computer and router has a route table used to determine how TCP/IP packets are forwarded
- Network services are started by the Internet Super Daemon or by stand-alone daemons

Summary

- There are many ways to remotely administer a Linux system, including the **`telnet`** and **`ssh`** commands, VNC, and Webmin

The End…

- Questions?