Some common networking issues found at the client side include:

- **DNS** issues - Can you ping the IP address but not the hostname?
- Firewall issues - A firewall on the client side which is rejecting the return traffic from a network request will cause problems.
- Incorrect network settings.
  - Make sure the IP address is correct. Does it match the **DNS** host name?
  - If the route is wrong or missing, traffic will not get to the other network node.
  - Netmasks determine network routes, thus it is important to have the netmask of your host correct.

To perform basic server troubleshooting, test the network connectivity from the server's point of view.

The **netstat** command lists which daemons are listening on which ports.

```
# netstat -taupe | grep httpd
```

```
tcp 0 0 *:http *:* LISTEN root 23390 5543/httpd
```

The **ss** command is another socket statistics utility. It may be a replacement to **netstat** although it is missing some socket types. A similar command to the **netstat** example shown would be:

```
# ss -ltp | grep httpd
```

Verify that the daemon is running, using the **chkconfig**, **service**, **ps** commands, or the **init** script.

One of the first steps in troubleshooting a server-side daemon should be to check the log files. Log messages will tell you exactly what is wrong, without having to do much debugging.

To perform basic server troubleshooting, test the network connectivity from the server's point of view.

The **netstat** command lists which daemons are listening on which ports.

```
# netstat -taupe | grep httpd
```

```
tcp 0 0 *:http *:* LISTEN root 23390 5543/httpd
```

The **ss** command is another socket statistics utility. It may be a replacement to **netstat** although it is missing some socket types. A similar command to the **netstat** example shown would be:

```
# ss -ltp | grep httpd
```

Verify that the daemon is running, using the **chkconfig**, **service**, **ps** commands, or the **init** script.

One of the first steps in troubleshooting a server-side daemon should be to check the log files. Log messages will tell you exactly what is wrong, without having to do much debugging.

With intermediate server troubleshooting, the server side firewall configuration needs to take into account that:

- New traffic is allowed in.
- Return traffic is allowed in.
- Unwanted traffic is filtered.

Access control systems may also cause troubleshooting problems. Check the settings of tools such as TCP wrappers. (`/etc/hosts.allow` and `/etc/hosts.deny`)

Consult **man 5 hosts_access** for additional details.

Application settings can also cause problems. Make sure there are not any restrictions, blacklists or other application configuration errors.

For advanced server troubleshooting, the `/proc` filesystem has settings that affect the network stack:

- `/proc/sys/net/ipv4/ip_forward`
  Allows for network traffic to be forwarded from one interface to another.
- `/proc/sys/net/ipv4/conf/*/accept_redirects`
  Accepting **ICMP** redirects from a router to find better routes. **This setting has the potential to be exploited by a malicious party to redirect your traffic.**
- `/proc/sys/net/ipv4/icmp_echo_ignore_all`
  Changing this setting will affect the host's visibility to **ICMP** ping packets.
- `/proc/sys/net/ipv4/icmp_echo_ignore_broadcasts`
  This setting will change the host's visibility to broadcast ICMP ping packets.
- `/proc/net/arp`
  Contains the current **arp table.**

These settings are not persistent across reboots. To make the changes persistent, edit the `/etc/sysctl.conf` configuration file, or a **.conf** file in the `/etc/sysctl.d` directory.

The syntax for `/etc/sysctl.conf` matches the path for the file in `/proc/sys` with the **.** character instead of */*.

4.11. Common Server Problems

Common server problems include broken **DNS**, overzealous firewall rules, incorrect network settings, and the daemon not listening on the right interface/port.

Some access control systems require that **Reverse DNS** be properly set up.

When enabling new traffic to pass through a firewall, pay attention to the type of protocol (**UDP** over **TCP** for example) used.

Some protocols break when return traffic comes back from a different **IP** address. Verify that your egress route is correct.

For network monitoring, the **iptraf** tool is a real-time network traffic analyzer. It recognizes the following protocols:

- **IP**
- **TCP**
- **UDP**
- **ICMP**
- **IGMP**
- **IGP**
- **IGRP**
- **OSPF**
- **ARP**
- **RARP.**

Snort is a network intrusion detection system. In addition to being a network monitor, it can help pinpoint unwanted traffic inside of a network.

ntop is an application and web app for monitoring network usage. It can pinpoint bandwidth use, display network statistics, and more.

By the end of this session, you should be able to:

- **Explain network troubleshooting from a client perspective.**
- **Explain network troubleshooting from a server perspective.**

When dealing with network services, one often has to troubleshoot problems which crop up. One should keep in mind that there is a client-side, and a server-side to each network connection:

- Network services require a working network.
- Consider client and server connectivity.

4.4. Simple Client Troubleshooting

The basics of network troubleshooting usually deal with basic connectivity testing. You can use the tools **ping**, **traceroute**, and **nmap** to test connectivity. Remember to test both **DNS** hostnames and **IP** addresses to diagnose **DNS**-related issues.

Test plain-text protocols by using the **telnet** command:

```
$ telnet example.com 80 Trying 192.0.43.10...

Connected to example.com.
Escape character is '^]'.
GET /
<html>
<head><title>welcome to example.com</title></head>
<body>
<h1>welcome to example.com</h1>
</body>
</html>
```

You can also do the same with **SSL** or **TLS** protocols.

```
$ openssl s_client -connect www.example.com:443
```

Use the **arp** command to check the link-layer connectivity.

The **tcpdump** command and **wireshark** tool are useful when you need to dig deeper into a protocol. The command line-based **tcpdump** truncates packets by default and generates **pcap** files.

wireshark uses the graphical interface to capture packets. It can capture and analyze packets in realtime. It is useful to analyze **pcap** files, but you may not want **wireshark** installed on the system you are troubleshooting.

To capture packets with **tcpdump** for use with **wireshark**, use:

```
# tcpdump -i eth0 -s 65535 -w capture.pcap port 22
```