LFS311 Advanced Linux System Administration and Networking

Course Book

LFS311

Advanced Linux System Administration and Networking

Version 3.4

LFS311: Version 3.4

Published by:

the **Linux Foundation**
http://www.linuxfoundation.org

Contents

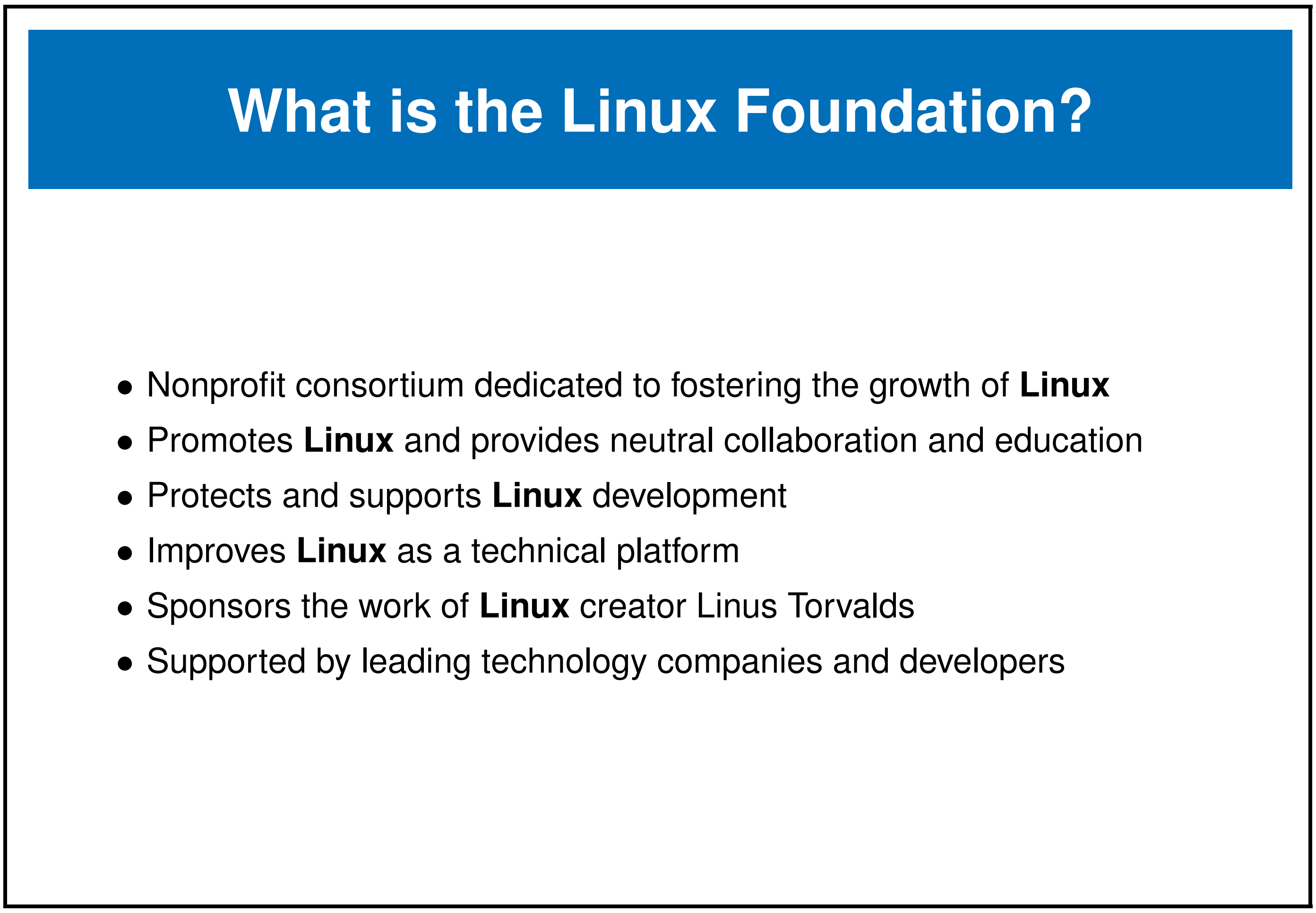
List of Figures

List of Tables

Chapter 1

Introduction

1.1 Linux Foundation



What is the Linux Foundation?

- Nonprofit consortium dedicated to fostering the growth of **Linux**
- Promotes **Linux** and provides neutral collaboration and education
- Protects and supports **Linux** development
- Improves **Linux** as a technical platform
- Sponsors the work of **Linux** creator Linus Torvalds
- Supported by leading technology companies and developers

The **Linux Foundation** is the nonprofit consortium dedicated to fostering the growth of **Linux**. Founded in 2000, it sponsors the work of **Linux** creator **Linus Torvalds** and is supported by leading technology companies and developers from around the world.

The **Linux Foundation** promotes, protects and advances **Linux** by marshaling the resources of its members and the open source development community to ensure **Linux** remains free and technically advanced.

It serves as a neutral spokesperson for **Linux** and generates original research and content that advances the understanding of the **Linux** platform. Its web properties, including `http://www.linux.com`, reach approximately two million people per month. It also fosters innovation by hosting collaboration events (including **LinuxCon**) for the **Linux** technical community, application developers, industry and end users to solve pressing issues facing the **Linux** ecosystem. Through its collaboration programs, end users, developers and industry collaborate on technical, legal and promotional issues.

The **Linux Foundation** sponsors **Linux** creator Linus Torvalds and other key kernel developers so they can work full time on improving **Linux**. It is vitally important that they remain independent.

It also manages the **Linux** trademark, offers developers legal intellectual property protection and a legal defense fund and coordinates industry and community legal collaboration and education, including important work to defend **Linux** against legal threats.

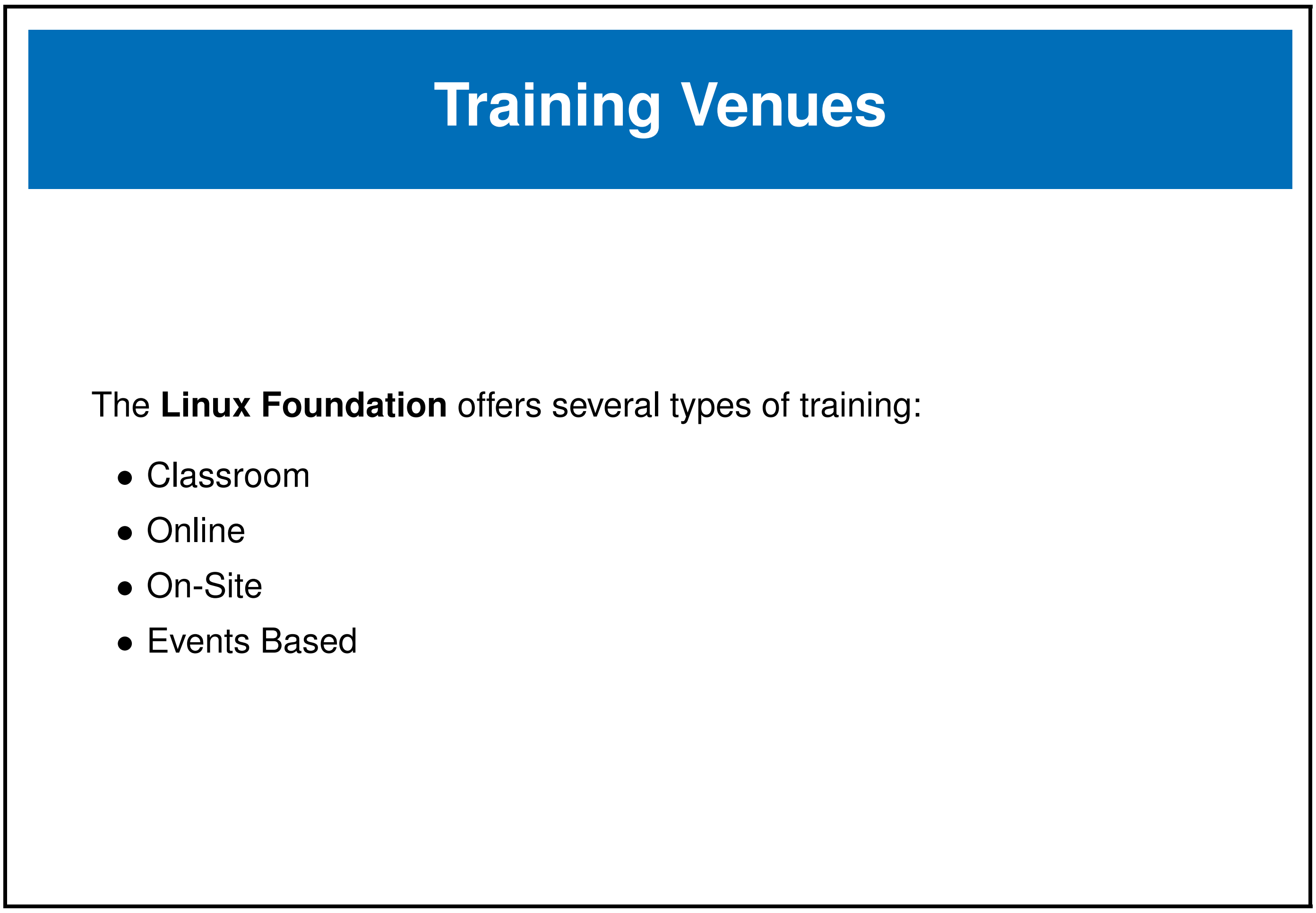
The **Linux Foundation** offers application developers standardization services and support, including the **Linux Standard Base**, that makes **Linux** an attractive target for their development efforts.

Linux Foundation Events

- LinuxCon North America, Europe and Japan
- Linux Kernel Summit
- CloudOpen North America, Japan
- Collaboration Summit
- Container Con
- Embedded Linux Conference North America, Europe
- Automotive Linux Summit
- ApacheCon
- CloudStack
- KVM Forum
- Linux Storage Filesystem and MM Summit
- Vault

The **Linux Foundation** produces technical events around the world. Whether it is to provide an open forum for development of the next kernel release, to bring together developers to solve problems in a real-time environment, to host workgroups and community groups for active discussion, to connect end users and kernel developers in order to grow **Linux** use in the enterprise or to encourage collaboration among the entire community, we know that our conferences provide an atmosphere that is unmatched in their ability to further the platform.

1.2 Linux Foundation Training



Training Venues

The **Linux Foundation** offers several types of training:

- Classroom
- Online
- On-Site
- Events Based

The **Linux Foundation**'s training is for the community, by the community, and features instructors and content straight from the leaders of the **Linux** developer community.

Attendees receive **Linux** training that is distribution-flexible, technically advanced and created with the actual leaders of the **Linux** development community themselves. **Linux Foundation** courses give attendees the broad, foundational knowledge and networking needed to thrive in their careers today. With either online or in person training, the **Linux Foundation** classes can keep you or your developers ahead of the curve on **Linux** essentials.

Training Offerings

Our current course offerings include:

- Linux Programming & Development Training
- Enterprise IT & Linux System Administration Courses
- Open Source Compliance Courses

For more information see `http://training.linuxfoundation.org`.

The **Linux Foundation** also offers a wide range of free **MOOC**s (**M**assively **O**pen **O**nline **C**ourses) offered through **edX** at `https://www.edx.org`. These cover basic as well as rather advanced topics associated with open source.

To find them at the **edX** website, search on `"Linux Foundation"`

Copyright

1.3 Linux Foundation Certifications

LFCS and LFCE

The **Linux Foundation** offers a two-level **Certification Program**:

- **LFCS** (**L**inux **F**oundation **C**ertified **S**ysadmin)
- **LFCE** (**L**inux **F**oundation **C**ertified **E**ngineer)

Full details about this program can be found at http://training.linuxfoundation.org/certification.

This information includes a thorough description of the **Domains** and **Competencies** covered by each exam.

Besides the **LFCS** and **LFCE** exams, the **Linux Foundation** is currently associated with other open source certification programs involving its collaborative projects, for which it has created (or is currently creating) the exams. These include:

- **Certified Openstack Administrator** (**COA**)
  https://www.openstack.org/coa/
- **Node.js Certfied Developer**
  https://nodejs.org/en/blog/announcements/nodejs-certified-developer-program/
- **Kubernetes** with the **Cloud Native Computing Foundation**
  https://www.linux.com/blog/get-trained-and-certified-kubernetes-linux-foundation-and-cncf

Certification/Training Firewall

- The **Linux Foundation** has two separate training divisions:
  - Certification
  - Course Delivery
- These are separated by a **firewall**:
  - Enables third party organizations to develop and deliver **LF** certification preparation classes
  - Prevents using ***secret sauce*** in **LF** courses
  - Prevents ***teaching the test*** in **LF** courses
- Instructors (including today) are guided entirely by publicly available information

The curriculum development and maintenance division of the **Linux Foundation** training department has no direct role in developing, administering, or grading certification exams.

Enforcing this self-imposed **firewall** ensures that independent organizations and companies can develop third party training material, geared to helping test takers pass their certification exams.

Furthermore, it ensures that there are no secret “tips” (or secrets in general) that one needs to be familiar with to succeed.

It also permits the **Linux Foundation** to develop a very robust set of courses that do far more than ***teach the test***, but rather equip attendees with a broad knowledge of many areas they may be required to master to have a successful career in **Linux** system administration.

Preparation Resources

- Before doing anything else, download:
  http://training.linuxfoundation.org/download-free-certification-prep-guide
- Sections on:
  - Domains and Competencies
  - Free Training Resources
  - Paid Training Resources
  - Taking the Exam
- Also get the **Candidate Handbook** at:

http://training.linuxfoundation.org/go/candidate_handbook

- Also read exam-specific information at:

http://training.linuxfoundation.org/certification/lfcs

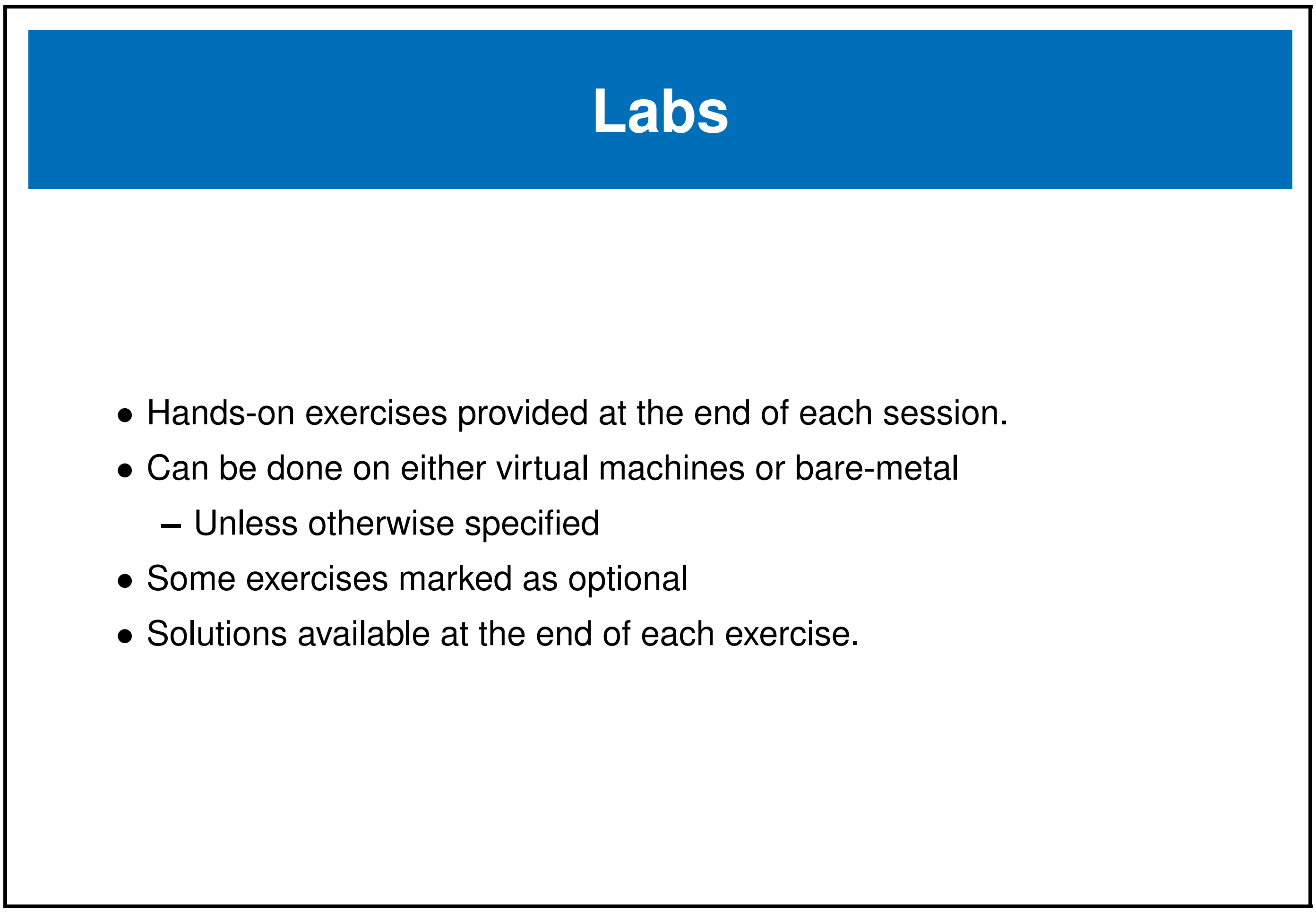
http://training.linuxfoundation.org/certification/lfce

We will discuss some (but not all!) of the issues in the documents quoted above, all of which can also be easily be found through links at the **Linux Foundation** web page at http://training.linuxfoundation.org/certification.

Topics covered include:

- What material is covered in the exam
- Candidate Requirements, including identification, authentication, eligibility, accessibility, confidentiality requirements.
- Exam registration and fees, refund policy, etc.
- **Linux** distribution choices
- Checking your hardware and software environment for suitability for the exam
- How to start and complete the exam
- Exam Interface and format
- Exam results, scoring and re-scoring requests, and retake policy
- Certificate issuance, verification, expiration, renewal etc.
- Accessing tech support

1.4 Laboratory Exercises, Solutions and Resources



Labs

- Hands-on exercises provided at the end of each session.
- Can be done on either virtual machines or bare-metal
  - Unless otherwise specified
- Some exercises marked as optional
- Solutions available at the end of each exercise.

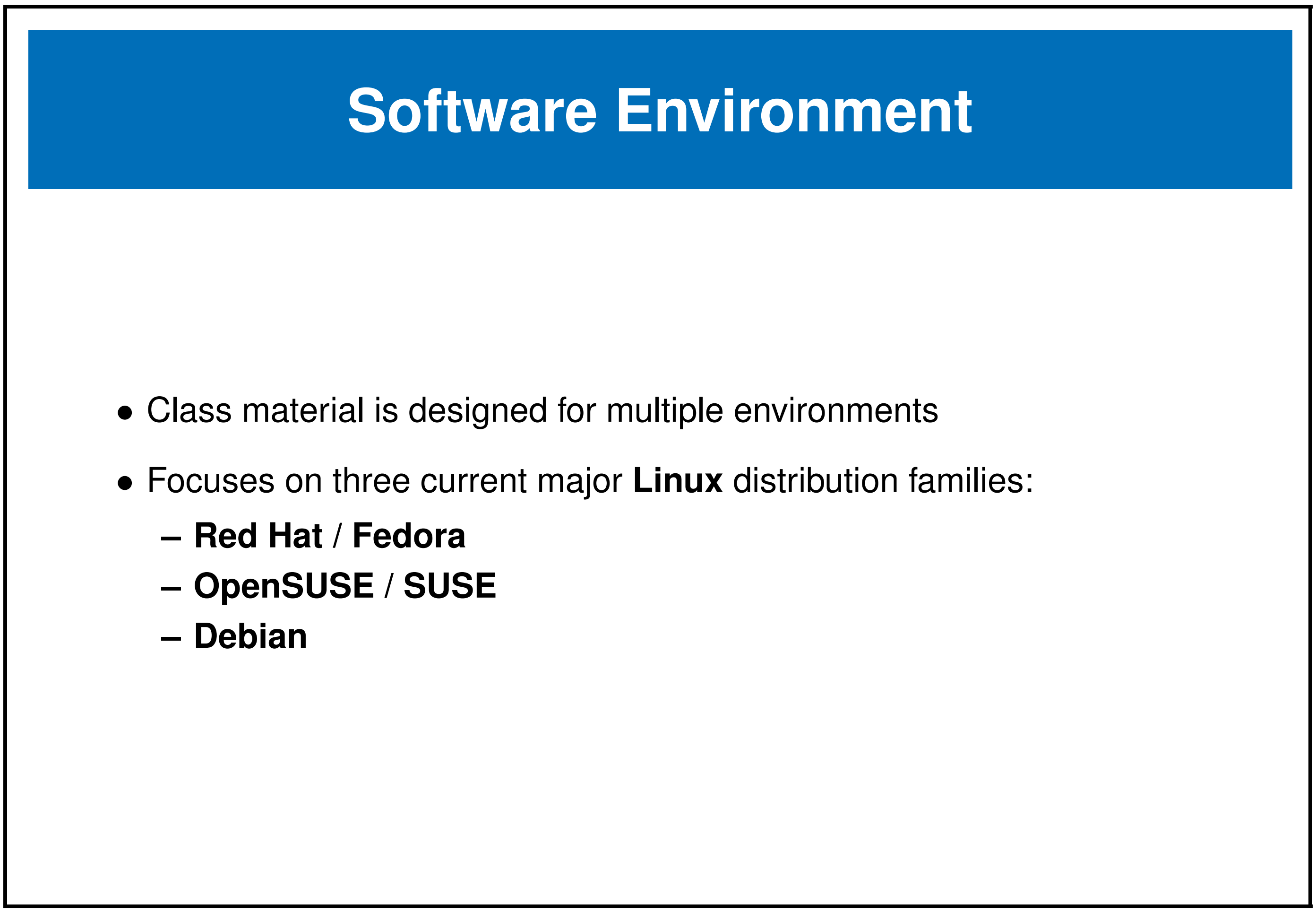
Obtaining Course Solutions and Resources

- If this course has such material, lab exercise solutions, suggestions and other resources may be downloaded from: http://training.linuxfoundation.org/cm/LFS311
- Any errata, updated solutions, etc. will also be posted on that site
- These files may be unpacked with:

```
$ tar jxvf LFS311_*SOLUTIONS*.tar.bz2
```

You will see subdirectories in the SOLUTIONS directory such as s_01, s_10, s_13 for each section that has files you either need or may find of interest. We may refer to these files in future sections.

1.5 Distribution Details



Software Environment

- Class material is designed for multiple environments
- Focuses on three current major **Linux** distribution families:
  - **Red Hat / Fedora**
  - **OpenSUSE / SUSE**
  - **Debian**

The material produced by the **Linux Foundation** is distribution-flexible. This means that technical explanations, labs and procedures should work on most modern distributions and we do not promote products sold by any specific vendor (although we may mention them for specific scenarios).

In practice, most of our material is written with the three main **Linux** distribution families in mind: **Red Hat** / **Fedora**, **Open-SUSE** / **SUSE** and **Debian**. Distributions used by our students tend to be one of those three alternatives, or a product that's derived from them.

Which Distribution to Choose

- Several factors to consider:
  - Has your employer already standardized?
  - Do you want to learn more?
  - Do you want to certify?
- You are encouraged to experiment with more than one distribution

You should ask yourself several questions when choosing a new distribution. While there are many reasons that may force you to focus on one **Linux** distribution versus another, we encourage you to gain experience on all of them. You will quickly notice that technical differences are mainly about package management systems, software versions and file locations. Once you get a grasp of those differences it becomes relatively painless to switch from one **Linux** distribution to another.

Some tools and utilities have vendor supplied front-ends, especially for more particular or complex reporting. The steps included in the text may need to be modified to run on a different platform.

Red Hat / Fedora Family

- Current material based upon the latest release of **Red Hat Enterprise Linux (RHEL)** 7.x at the time of publication, and should work well with later versions.
- Supports **x86**, **x86-64**, **Itanium**, **PowerPC** and **IBM System Z**
- RPM-based, uses **yum** (or **dnf**) to install and update
- Long release cycle; targets enterprise server environments
- Upstream for **CentOS**, **Scientific Linux** and **Oracle Linux**
- **Note: CentOS** is used for demos and labs because it is available at no cost.

Fedora is the community distribution that forms the basis of **Red Hat Enterprise Linux**, **CentOS**, **Scientific Linux** and **Oracle Linux**. **Fedora** contains significantly more software than **Red Hat**'s enterprise version. One reason for this is a diverse community is involved in building **Fedora**; it is not just one company.

The **Fedora** community produces new versions every six months or so. For this reason, we decided to standardize the **Red Hat** / **Fedora** part of the course material on the latest version of **CentOS 7**, which provides much longer release cycles. Once installed, **CentOS** is also virtually identical to **Red Hat Enterprise Linux** (**RHEL**), which is the most popular **Linux** distribution in enterprise environments.

OpenSUSE Family

- Current material based upon the latest release of **OpenSUSE** and should work well with later versions.
- RPM-based, uses **zypper** to install and update.
- **YaST** available for administration purposes
- **x86** and **x86-64**
- Upstream for **SUSE Linux Enterprise Server (SLES)**
- **Note: OpenSUSE** is used for demos and labs because it is available at no cost.

The relationship between **OpenSUSE** and **SUSE Linux Enterprise Server** is similar to the one we just described between **Fedora** and **Red Hat Enterprise Linux**. In this case, however, we decided to use **OpenSUSE** as the reference distribution for the **OpenSUSE** family due to the difficulty of obtaining a free version of **SUSE Linux Enterprise Server**. The two products are extremely similar and material that covers **OpenSUSE** can typically be applied to **SUSE Linux Enterprise Server** with no problem.

Debian Family

- Commonly used on both servers and desktop
- **DPKG**-based, use **apt-get** and front-ends for installing and updating
- Upstream for **Ubuntu**, **Linux Mint** and others
- Current material based upon the latest release of **Ubuntu** and should work well with later versions.
- **x86** and **x86-64**
  - Long Term Release (LTS)
- **Note: Ubuntu** is used for demos and labs because it is available at no cost, as is **Debian**, but has a more relevant user base.

The **Debian** distribution is the upstream for several other distributions including **Ubuntu**, **Linux Mint** and others. **Debian** is a pure open source project and focuses on a key aspect: stability. It also provides the largest and most complete software repository to its users.

Ubuntu aims at providing a good compromise between long term stability and ease of use. Since **Ubuntu** gets most of its packages from **Debian**'s unstable branch, **Ubuntu** also has access to a very large software repository. For those reasons, we decided to use **Ubuntu** as the reference **Debian**-based distribution for our lab exercises.

New Distribution Similarities

- Current trends and changes to the distributions have reduced some of the differences between the distributions.
  - **systemd**, system startup and service management
  - **journald**, manage system logs
  - **firewalld**, firewall management daemon
  - **ip**, network display and configuration tool
- Since these utilities are common across distributions the lecture and lab information will use these utilities.
- If your choice of distribution or release does not support these commands, please translate accordingly.

systemd is used by the most common distributions replacing the **SysVinit** and **Upstart** packages. Replaces **service** and **chkconfig** commands.

journald is a **systemd** service that collects and stores logging data. It creates and maintains structured, indexed journals based on logging information that is received from a variety of sources. Depending on the distribution text based system logs may be replaced.

firewalld provides a dynamically managed firewall with support for network/firewall zones to define the trust level of network connections or interfaces. It has support for IPv4, IPv6 firewall settings and for ethernet bridges. This replaces the **iptables** configurations.

The **ip** program is part of the **net-tools** package and is designed to be a replacement for the **ifconfig** command. The **ip** command will show or manipulate routing, network devices, routing information and tunnels.

These documents may be of some assistance translating older commands to their **systemd** counterparts:

https://fedoraproject.org/wiki/SysVinit_to_Systemd_Cheatsheet

https://wiki.debian.org/systemd/CheatSheet

https://en.opensuse.org/openSUSE:Cheat_sheet_13.1#Services

AWS Free Tier

- **Amazon Web Services** (**AWS**) offers a wide range of virtual machine products (**instances**) which remote users can access in the cloud.
- In particular, one can use their **Free Tier** account level for up to a year and the simulated hardware and software choices available may be all one needs to perform the exercises for **Linux Foundation** training courses and gain experience with **Linux**. Or they may furnish a very educational supplement to working on local hardware, and offer opportunities to easily study more than one **Linux** distribution.
- For kernel-level courses there are some particular challenges. Please download a helpful guide we have prepared to help you experiment with the **AWS** free tier: `https://training.linuxfoundation.org/cm/prep/aws.pdf`.

1.6 Labs

Lab 1

Exercise 1.1 Configuring the System for **sudo**

It is very dangerous to run a **root shell** unless absolutely necessary: a single typo or other mistake can cause serious (even fatal) damage.

Thus, the sensible procedure is to configure things such that single commands may be run with superuser privilege, by using the **sudo** mechanism. With **sudo** the user only needs to know their own password and never needs to know the root password.

If you are using a distribution such as **Ubuntu**, you may not need to do this lab to get **sudo** configured properly for the course. However, you should still make sure you understand the procedure.

To check if your system is already configured to let the user account you are using run **sudo**, just do a simple command like:

```
$ sudo ls
```

You should be prompted for your user password and then the command should execute. If instead, you get an error message you need to execute the following procedure.

Launch a root shell by typing **su** and then giving the **root** password, not your user password.

On all recent **Linux** distributions you should navigate to the `/etc/sudoers.d` subdirectory and create a file, usually with the name of the user to whom root wishes to grant **sudo** access. However, this convention is not actually necessary as **sudo** will scan all files in this directory as needed. The file can simply contain:

```
student ALL=(ALL)  ALL
```

if the user is `student`.

An older practice (which certainly still works) is to add such a line at the end of the file `/etc/sudoers`. It is best to do so using the **visudo** program, which is careful about making sure you use the right syntax in your edit.

You probably also need to set proper permissions on the file by typing:

```
$ chmod 440 /etc/sudoers.d/student
```

(Note some **Linux** distributions may require `400` instead of `440` for the permissions.)

After you have done these steps, exit the root shell by typing `exit` and then try to do `sudo ls` again.

There are many other ways an administrator can configure **sudo**, including specifying only certain permissions for certain users, limiting searched paths etc. The `/etc/sudoers` file is very well self-documented.

However, there is one more setting we highly recommend you do, even if your system already has **sudo** configured. Most distributions establish a different path for finding executables for normal users as compared to root users. In particular the directories `/sbin` and `/usr/sbin` are not searched, since **sudo** inherits the `PATH` of the user, not the full root user.

Thus, in this course we would have to be constantly reminding you of the full path to many system administration utilities; any enhancement to security is probably not worth the extra typing and figuring out which directories these programs are in. Consequently, we suggest you add the following line to the `.bashrc` file in your home directory:

```
PATH=$PATH:/usr/sbin:/sbin
```

If you log out and then log in again (you don't have to reboot) this will be fully effective.

Exercise 1.2 Class Registration

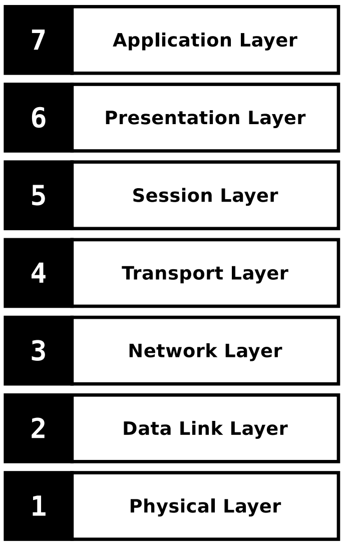
- Register at: `https://training.linuxfoundation.org/surveys/registration`
- The key required to fill the web form will be provided by your instructor.
- This is also the key which will be used for surveys at the end of the class so please write it down on the page that corresponds to the last lab at the end of the book.
- A certificate of completion will be provided after the class is completed.

Chapter 2

Linux Networking Concepts and Review

2.1 OSI Model Introduction and Upper Layers

OSI Model



- Standard language for talking about networking topics
- Not all layers used at all times
- Used to describe other networking technologies (e.g., Internet Protocol Suite)

Figure 2.1: **OSI Model**

The **OSI (Open Systems Interconnection)** model was created to standardize the language used to describe networking protocols. It defines the manner in which systems communicate with one another using abstraction layers.

Each layer communicates with the layer directly above and below.

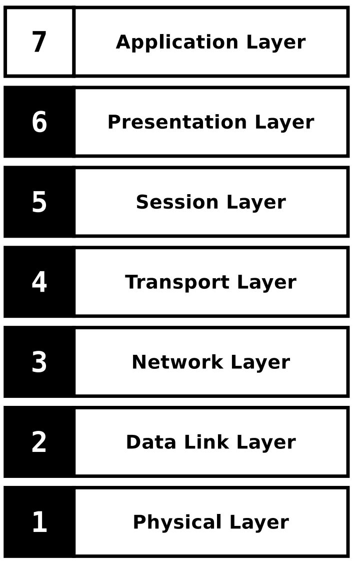
Table 2.1: **OSI Model Layers**

Layer	Name	Description
7	Application	Most networking stacks have a concept of the application layer.
6	Presentation	Most networking stacks combine the presentation layer in layers four five or seven.
5	Session	Sometimes used by different stacks, often combined into other layers (seven or six).
4	Transport	Most networking stacks have a concept of the transport layer.
3	Network	Most networking stacks have a concept of the network layer.
2	Data Link	Most networking stacks have a concept of the data link layer.
1	Physical	Most networking stacks have a concept of the physical layer

There are other models which are used to talk about networking. The most popular networking stack on the Internet today is the **Internet Protocol Suite**.

The **Internet Protocol Suite** can be described using a subset of the **OSI Model**.

OSI Layer 7 Application Layer



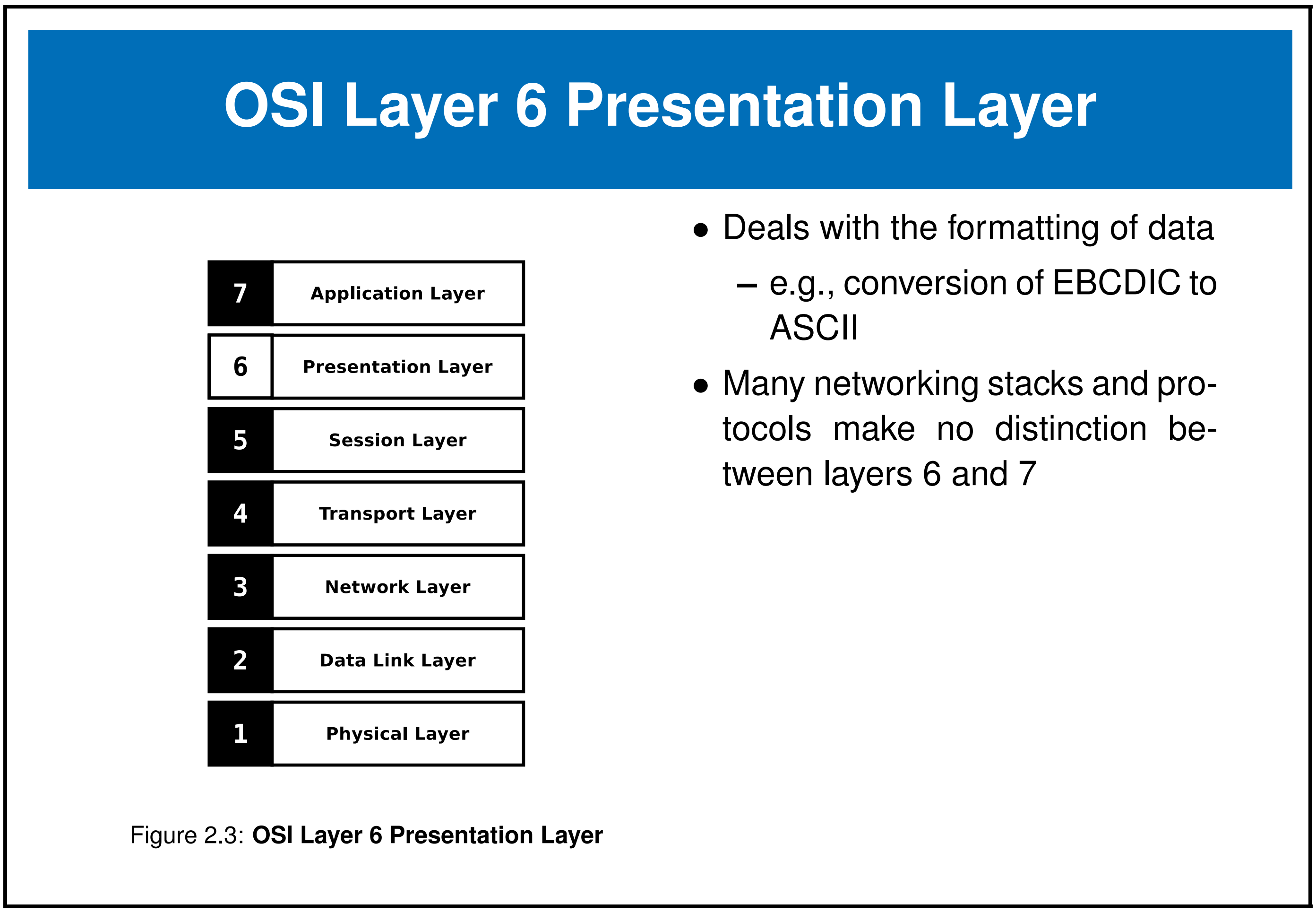
- The highest layer of abstraction
- Protocols at this layer are most familiar to users
- Protocols defined by RFC1123 and elsewhere (`http://tools.ietf.org/html/rfc1123`)

Figure 2.2: **OSI Layer 7 Application Layer**

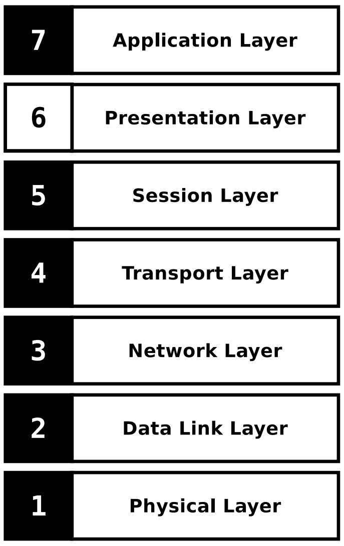
The **Application Layer** is the most well known. This layer is the top of the stack and deals with the protocols which make a global communications network function.

Common protocols which exist in the **Application Layer**:

- **HTTP**: Hypertext Transfer Protocol
- **SMTP**: Simple Mail Transfer Protocol
- **DNS**: Domain Name System
- **FTP**: File Transfer Protocol
- **DHCP**: Dynamic Host Configuration Protocol



OSI Layer 6 Presentation Layer



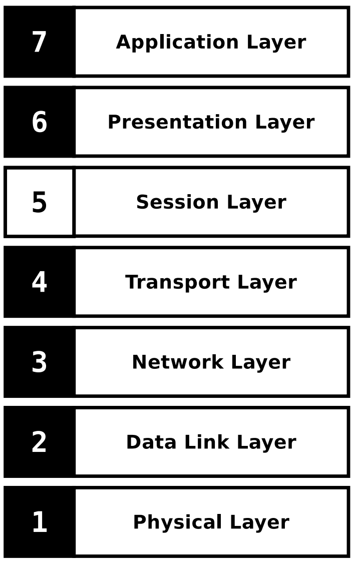
- Deals with the formatting of data
  - e.g., conversion of EBCDIC to ASCII
- Many networking stacks and protocols make no distinction between layers 6 and 7

Figure 2.3: **OSI Layer 6 Presentation Layer**

The **Presentation Layer** is commonly rolled up into a different layer.

For example, the HTTP protocol (an **Application Layer** protocol) has methods for converting character encoding. In other words this **Presentation Layer** step happens at the **Application Layer**.

OSI Layer 5 Session Layer



- Deals with managing of session data
- Used by protocols which need reliable sessions
  - Videoconferencing
  - SOCKS proxy

Figure 2.4: **OSI Layer 5 Session Layer**

The **Session Layer** creates a semi-permanent connection which is then used for communications.

Many of the RPC-type protocols depend on this layer:

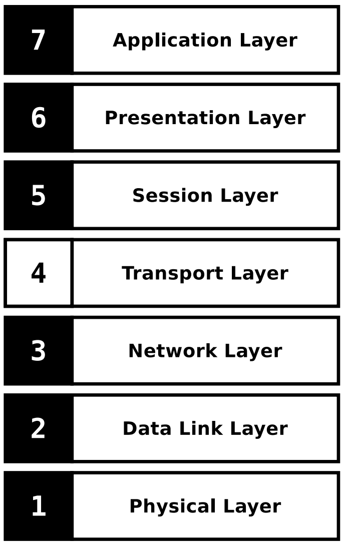
- **NetBIOS**: Network Basic Input Output System
- **RPC**: Remote Procedure Call
- **PPTP**: Point to Point Tunneling Protocol

If an established connection is lost or disrupted, this layer may try to recover the connection.

If a connection is not used for a long time, the session layer may close and reopen it.

2.2 OSI Model Transport Layer

OSI Layer 4 Transport Layer



- Deals with end-to-end communication
- Handles
  - Reliability
  - Flow Control
  - Multiplexing

Figure 2.5: **OSI Layer 4 Transport Layer**

The **Transport Layer** is responsible for the end-to-end communication protocols. Data is properly multiplexed by defining source and destination port numbers. This layer also deals with reliability by adding check sums, doing request repeats, and avoiding congestion.

Common protocols in the **Transport Layer**

- **TCP**: Transmission Control Protocol. The main component of the TCP/IP (**Internet Protocol Suite**) stack
- **UDP**: User Datagram Protocol. Another popular component of the **Internet Protocol Suite** stack
- **SCTP**: Stream Control Transmission Protocol
- Use port numbers to allow for connection multiplexing

Transport Layer Ports

Transport layer protocols use ports to distinguish different types of traffic or to do multiplexing

Ports are classed three different ways

- Well Known (0 - 1023)– assigned by the **IANA** (**I**nternet **A**ssigned **N**umbers **A**uthority)
  - Usually require super-user privilege to be bound
- Registered (1024 - 49151) – assigned by the **IANA**
  - On most systems can be bound by non-super-user privilege
- Dynamic or Ephemeral ports (49152 - 65535)
  - For temporary or private use
  - Also used for source (client) ports in connections

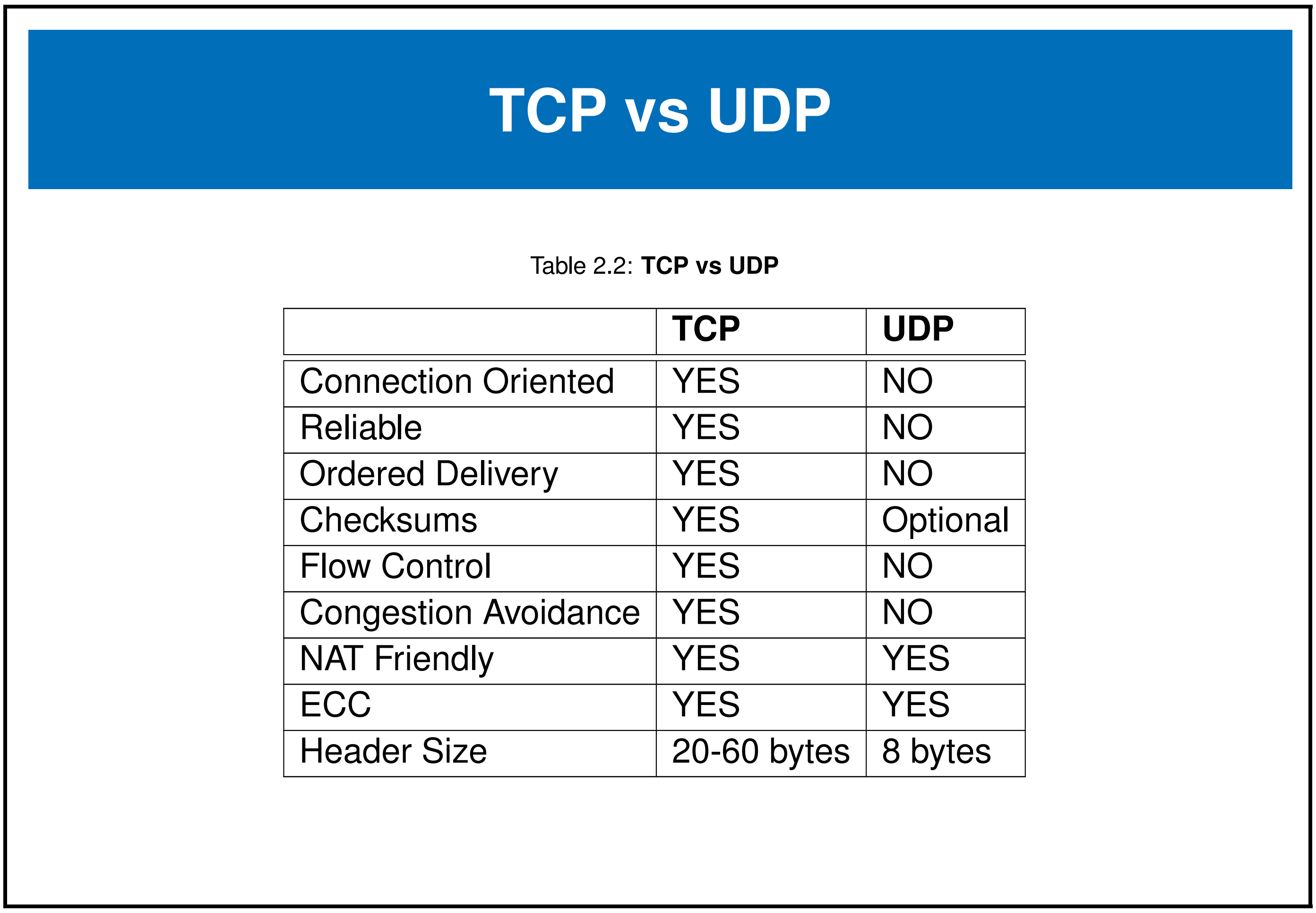
Some of the well-known ports:

- 22 TCP: SSH
- 25 TCP: SMTP
- 80 TCP: HTTP
- 443 TCP: HTTPS

Some of the Registered ports:

- 1194 TCP/UDP: OpenVPN
- 1293 TCP/UDP: IPSec
- 1433 TCP: MSSQL Server

The ephemeral ports are used as source ports for the client-side of a TCP or UDP connection. You can also use the ephemeral ports for a temporary or non-root service.



TCP vs UDP

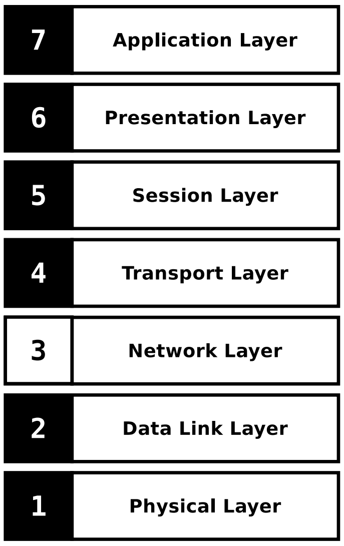
Table 2.2: **TCP vs UDP**

	TCP	**UDP**
Connection Oriented	YES	NO
Reliable	YES	NO
Ordered Delivery	YES	NO
Checksums	YES	Optional
Flow Control	YES	NO
Congestion Avoidance	YES	NO
NAT Friendly	YES	YES
ECC	YES	YES
Header Size	20-60 bytes	8 bytes

TCP is useful when data integrity, ordered delivery and reliability are important. It is the backbone to many of the most popular protocols. UDP is useful when transmission speed is important and the integrity of the data isn't as important, or is managed by an above layer.

2.3 OSI Model Network Layer

OSI Layer 3 Network Layer



- Deals with routing and packet forwarding and managing quality of service
- Connectionless
  - Connection tracking can happen at layers above
- Common protocols
  - IPv4
  - IPv6
- Backbone of the Internet

Figure 2.6: **OSI Layer 3 Network Layer**

The OSI layer 3 is all about routing packets. This layer is responsible for getting the packets to the next hop in the path to the destination.

In many cases the final destination is not adjacent to this machine so the packets are routed based on the local routing table information.

Many routing and control protocols live at this layer. Protocols such as:

- **IP** Internet Protocol
- **OSPF** Open Shortest Path First
- **IGRP** Interior Gateway Routing Protocol
- **ICMP** Internet Control Message Protocol
- and many others.

Internet Protocol

- Originally the datagram service for TCP
- Now transfers many different higher level protocols
- Routes datagrams (packets) based on addressing
- Two major versions of IP
  - IPv4
  - IPv6

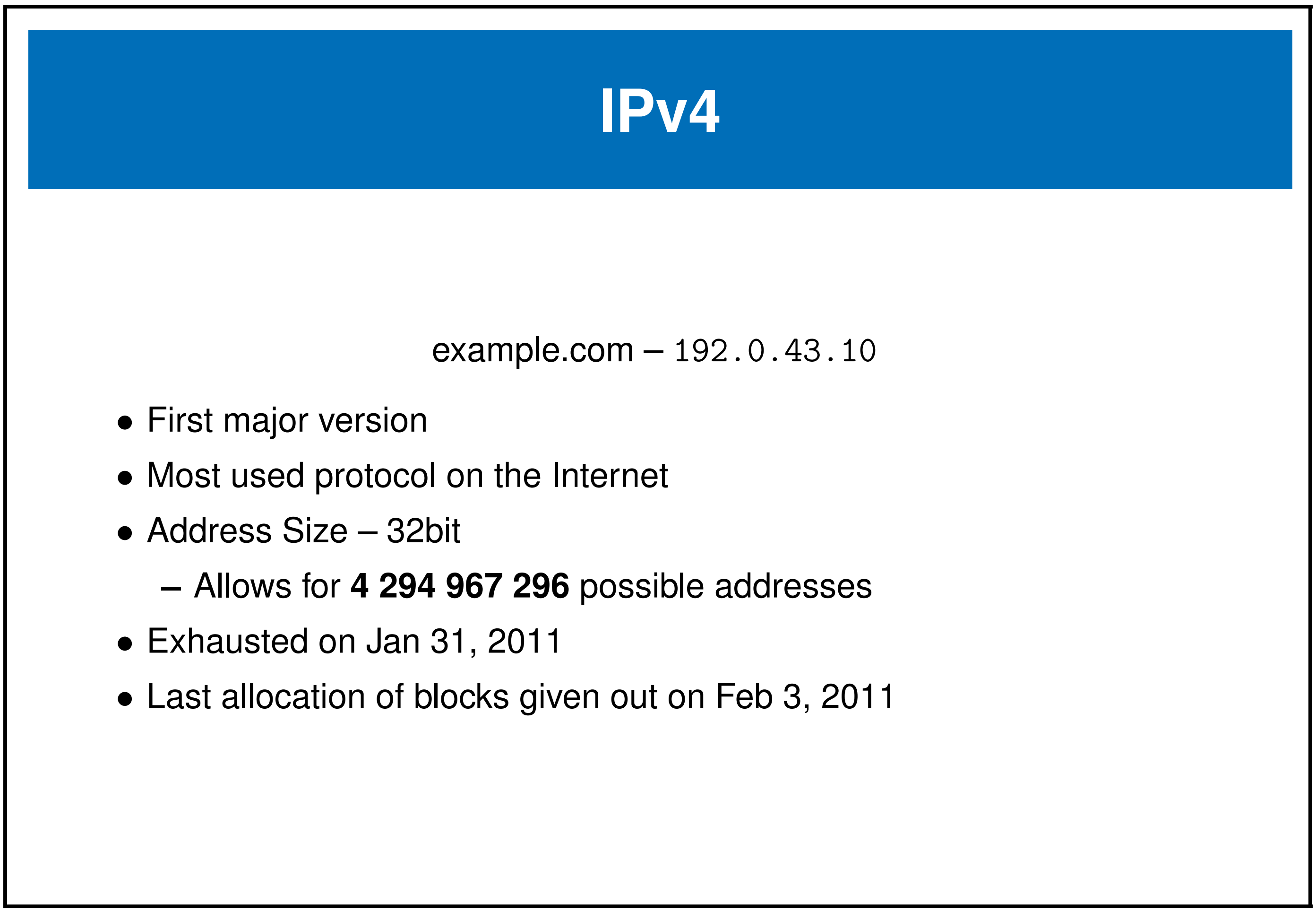
The internet protocol has two main functions:

- Addressing
- Fragmentation

The addressing function examines the address on the incoming packet and decides if the datagram is for the local system or another system. If the address indicates the datagram is for the local system the headers are removed and the datagram is passed up to the next layer in the protocol stack. If the address indicates the datagram is for another machine then it is passed to the next system in the direction of the final destination.

The fragmentation component will split and re-assemble the packets if the path to the next system uses a smaller transmission unit size.

For additional information see `https://www.ietf.org/rfc/rfc791.txt`



IPv4

example.com – `192.0.43.10`

- First major version
- Most used protocol on the Internet
- Address Size – 32bit
  - Allows for **4 294 967 296** possible addresses
- Exhausted on Jan 31, 2011
- Last allocation of blocks given out on Feb 3, 2011

The IPv4 address space exhaustion has been a concern for some time. There are different solutions for mitigating the problem:

- The move from Classed networks to CIDR.
- The invention of NAT.
- The move to IPv6.

IPv6

example.com – `2001:500:88:200::10`

- Successor to IPv4
- Address Size – 128bit
  - Allows for **3.4x10^{38}** possible addresses
- Was designed to deal with the exhaustion of IPv4 addresses
- Also deals with other shortcomings of IPv4

The version 6 of the Internet Protocol (IPv6) addresses several issues with the version 4 (IPv4).

- Expanded Addressing Capabilities
- Header Format Simplification
- Improved Support for Extensions and Options
- Flow Labeling Capability

For additional information see `https://www.ietf.org/rfc/rfc2460.txt`

IP Address Parts

Table 2.3: **IP Address Parts**

IP Addr Dec	10	20	0	100
Netmask Dec	255	255	0	0
IP Addr Bin	00001010	00010100	00000000	01100100
Netmask Bin	11111111	11111111	00000000	00000000
Network Part Bin	00001010	00010100		
Host Part Bin			00000000	00110010

IP addresses have two parts:

- Network Part
- Host Part

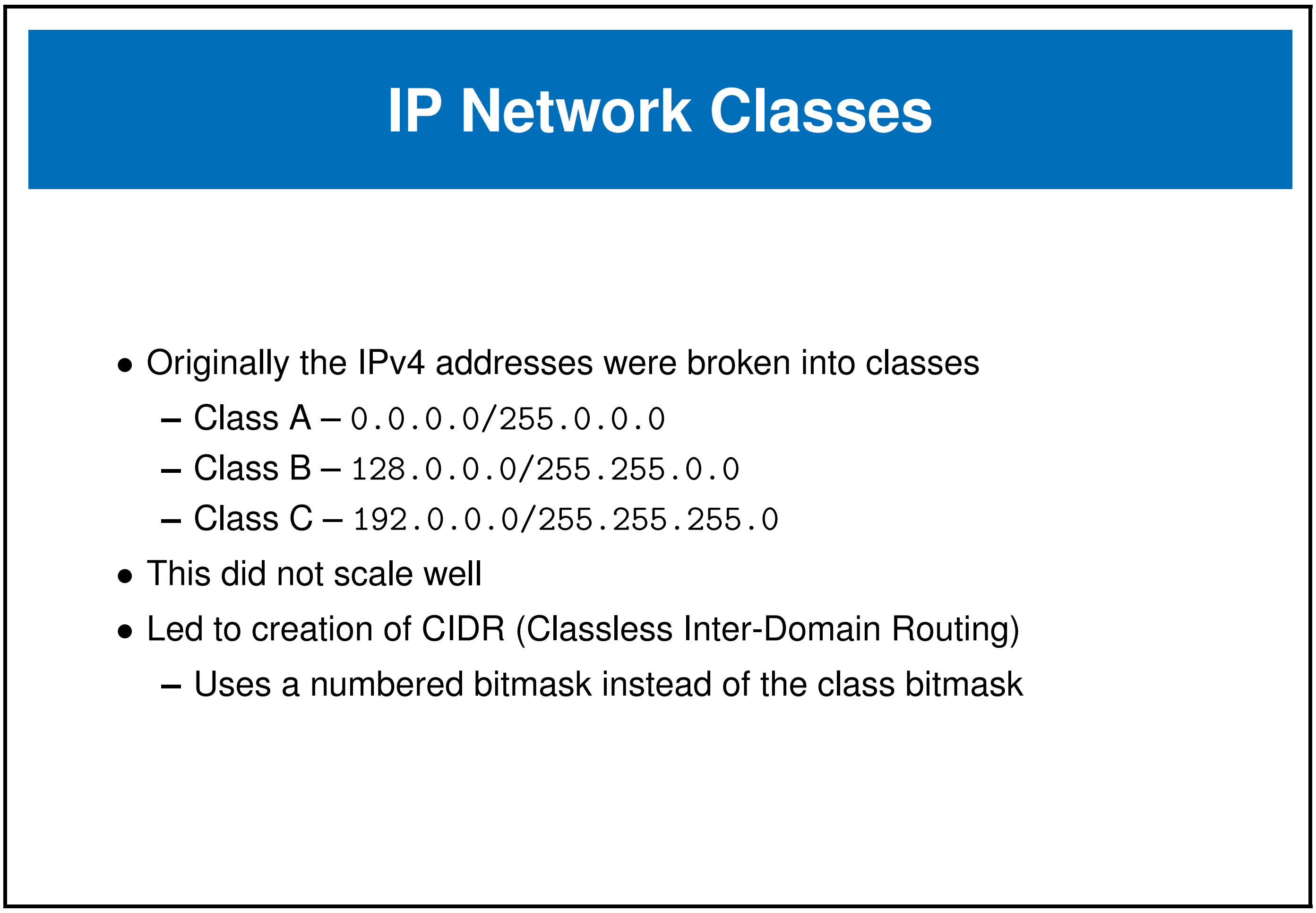
They are distinguished by using a **Netmask**, a bitmask defining which part of an IP address is the network part.

IP Subnetting

Table 2.4: **IP Subnetting**

IP Addr Dec	10	20	0	100
Netmask Dec	255	255	0	0
IP Addr Bin	00001010	00010100	00000000	01100100
Netmask Bin	11111111	11111111	00000000	00000000
Subnet Bin			11111111	
Network Part Bin	00001010	00010100		
Subnet Part Bin			00000000	
Host Part Bin				00110010

IP networks can be broken into smaller pieces by using a subnet. The above example is breaking a network with a netmask of `255.255.0.0` into a smaller subnet with a netmask of `255.255.255.0`.



IP Network Classes

- Originally the IPv4 addresses were broken into classes
  - Class A – `0.0.0.0/255.0.0.0`
  - Class B – `128.0.0.0/255.255.0.0`
  - Class C – `192.0.0.0/255.255.255.0`
- This did not scale well
- Led to creation of CIDR (Classless Inter-Domain Routing)
  - Uses a numbered bitmask instead of the class bitmask

Original classes of networks and subnets did not scale well. Networks which did not fit in a class B were often given a class A. This led to IP addresses going to waste.

Classless Inter-Domain Routing

- Classed Network Netmasks

Table 2.5: **Classed Network Netmasks**

Class A Dec	255	0	0	0
Class A Bin	11111111	00000000	00000000	00000000
Class B Dec	255	255	0	0
Class B Bin	11111111	11111111	00000000	00000000
Class C Dec	255	255	255	0
Class C Bin	11111111	11111111	11111111	00000000

- CIDR Network Netmasks Are More Flexible

Table 2.6: **CIDR Network Netmasks**

CIDR /18 Dec	255	255	192	0
CIDR /18 Bin	11111111	11111111	11000000	00000000

CIDR networks have netmasks which do not have to end on “nibble” boundaries.

IP Routing

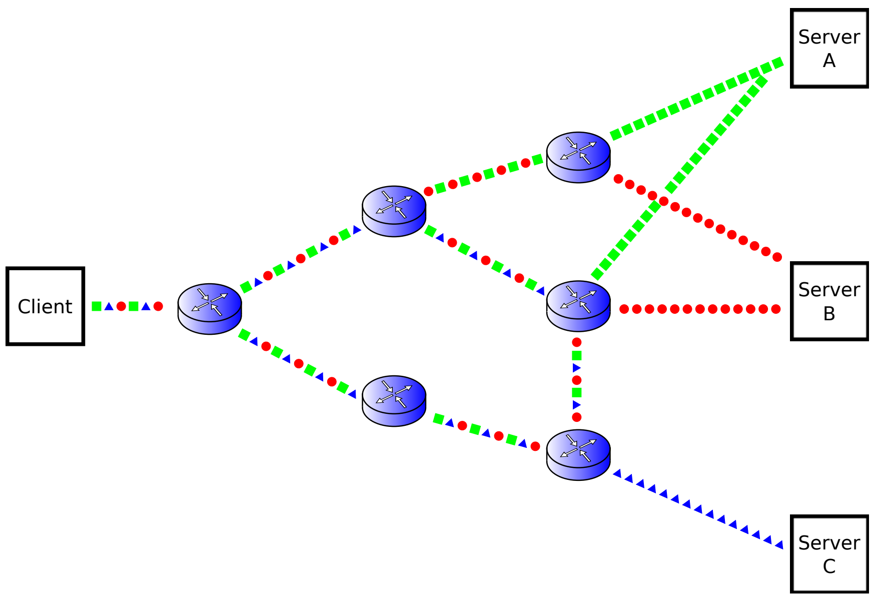


Figure 2.7: **IP Routing**

IP packets can only reach from end to end if they are routed properly.

Network routers inspect packet headers, and make routing decisions based on the destination address.

IP Management Tools

- **ifconfig** – part of the **net-tools** package
  - Historical tool, not very flexible
- **ip** – part of the **iproute** package
  - The new default tool for many distributions
  - Manages layer 2 and 3 settings

Setting an address with **ifconfig**:

```
# ifconfig eth0 10.20.0.100 netmask 255.255.0.0
```

Setting an address with **ip**:

```
# ip addr add 10.20.0.100/16 dev eth0
```

2.4 OSI Model Lower Layers

OSI Layer 2 Data Link Layer

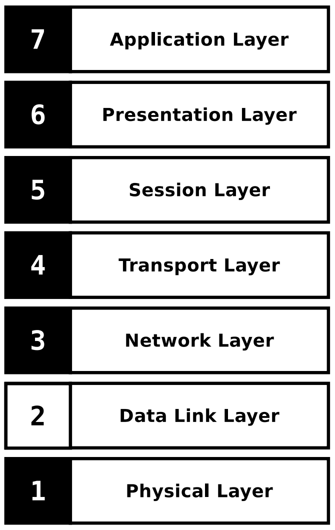


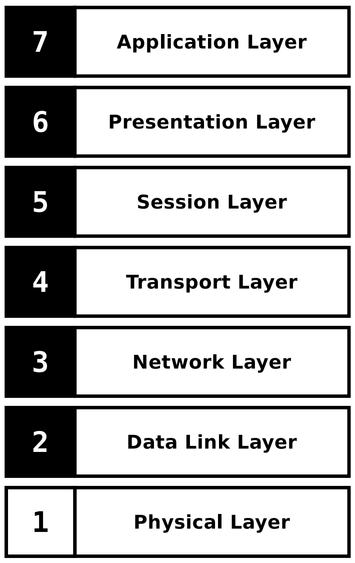
Figure 2.8: **OSI Layer 2 Data Link Layer**

- Deals with transferring data between network nodes
  - Adjacent nodes in a Wide Area Network
  - Nodes on the same Local Area Network Segment

Common **Data Link** protocols:

- **Ethernet**
- **ARP**: Address Resolution Protocol
- **PPP**: Point to Point Protocol
- **STP**: Spanning Tree Protocol

OSI Layer 1 Physical Layer



- Deals with transferring bits over a physical medium
  - Electric pulses over copper cables
  - Laser pulses over fiber optic cables
  - Frequency modulations over radio waves
  - Scraps of paper over carrier pigeons (`http://tools.ietf.org/html/rfc1149`)

Figure 2.9: **OSI Layer 1 Physical Layer**

The **Physical Layer** is the lowest possible layer and deals with the actual physical transfer of information. There are various different protocols, hardware types and standards defined for different types of physical networks (commonly referred to as PHYs).

- **IEEE 802.3**: Copper or fiber connections
- **IEEE 802.11**: Wireless (Wi-Fi) connections
- **Bluetooth**: Wireless connections
- **USB**: Copper connections
- **RS232**: Copper serial connections

2.5 Network Topology

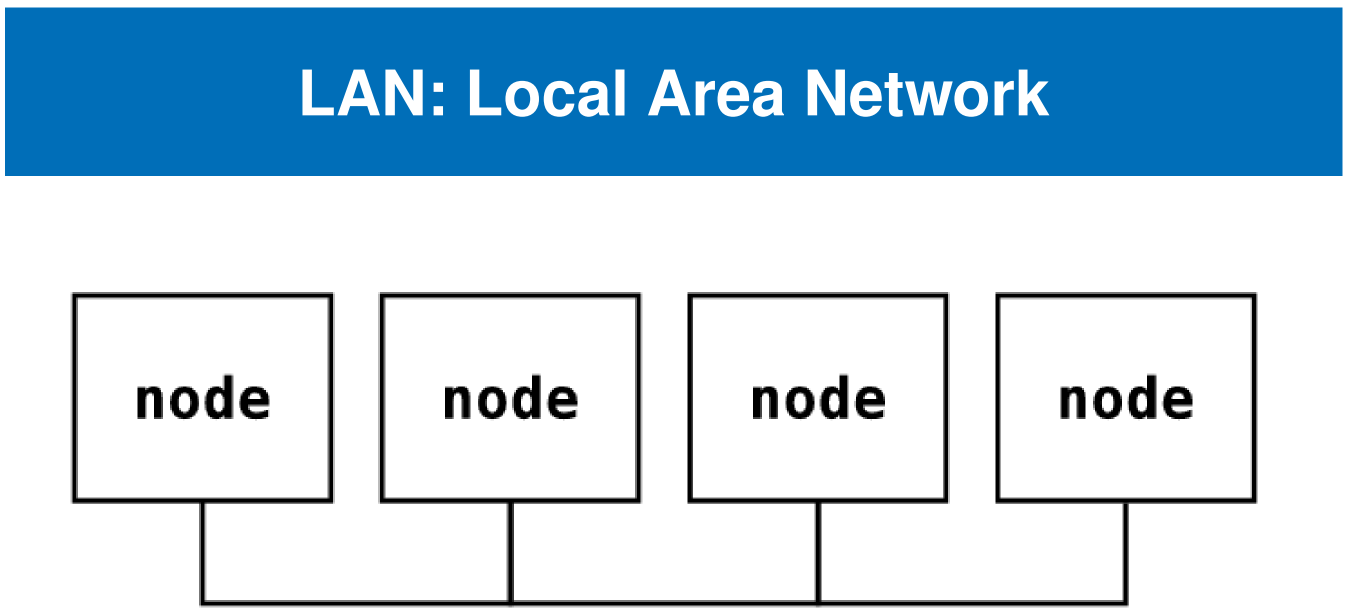


Figure 2.10: **LAN: Local Area Network**

Smaller network segments are sometimes referred to as **collision domains** since there can be only one transmitter on the physical medium at a time. This is common in hub-based networks. Modern switch technology removes the restriction of one at a time but the concept of **broadcast domains** still exists.

- Smaller, locally connected network
- Connected at layer 2 by the same series of switches or hubs
- Node to node communication happens at layer 3 using the same network.
- Layer 2 to layer 3 association may use layer 2 broadcasts and the ARP and RARP protocols to associate MAC addresses with IP addresses.

Bridged Network

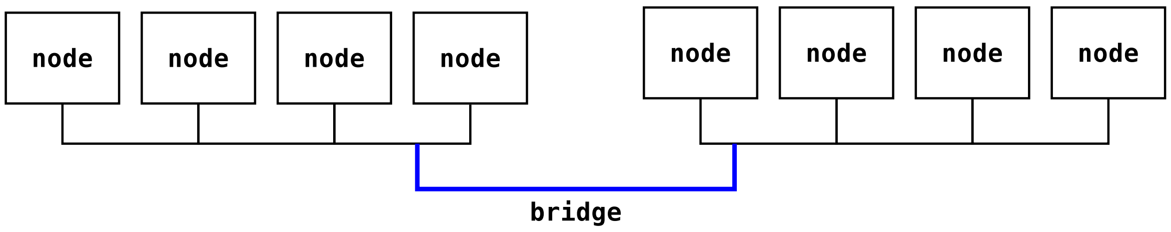


Figure 2.11: **Bridged Network**

A network bridge or repeater accepts packets on one side of the bridge and passes them through to the other side of the bridge. The effect is bi-directional.

- It is a combination of two or more networks at layer 2.
- Bridged networks communicate as a single network.

WAN: Wide Area Network

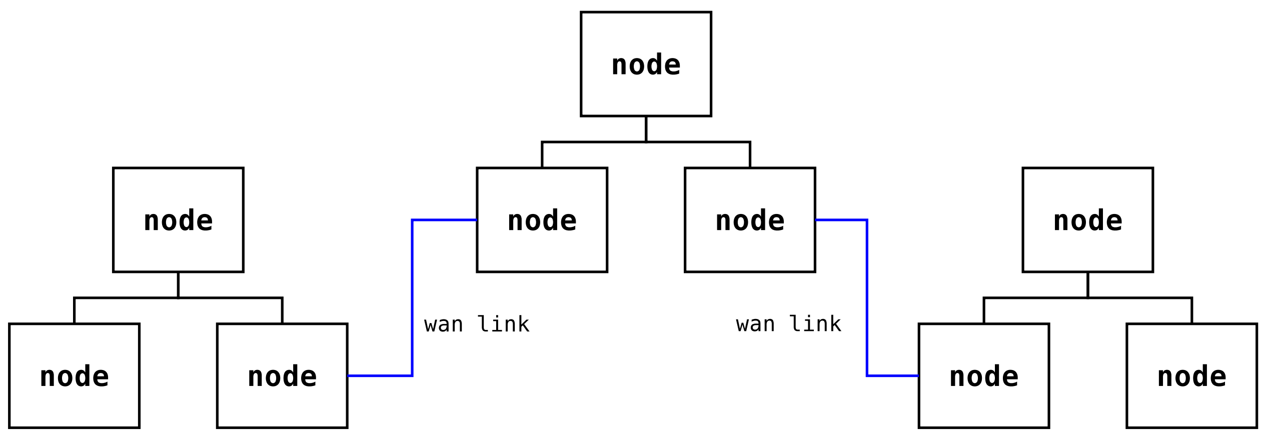


Figure 2.12: **WAN: Wide Area Network**

- Large, geographically diverse network
- Connected at layer 3 from node to node

VLAN: Virtual Local Area Network

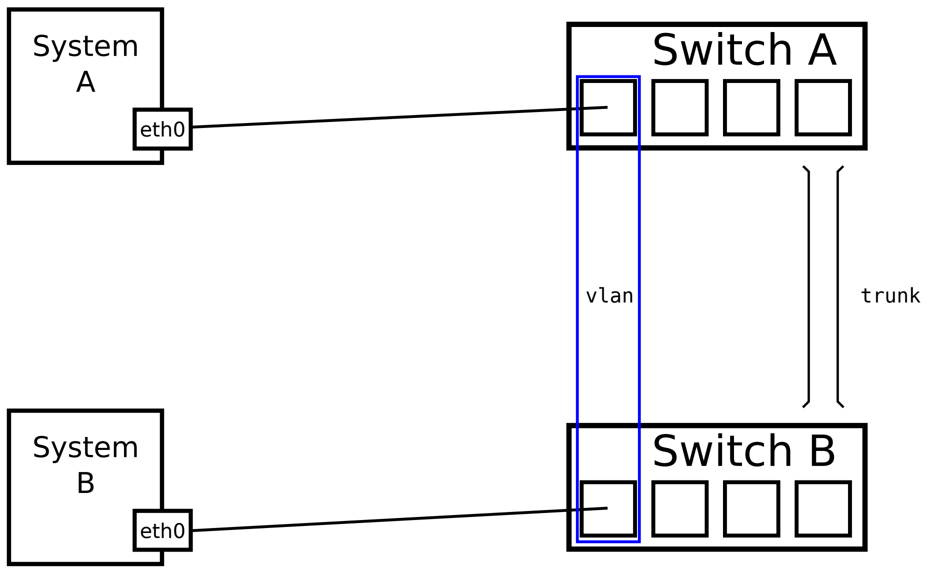
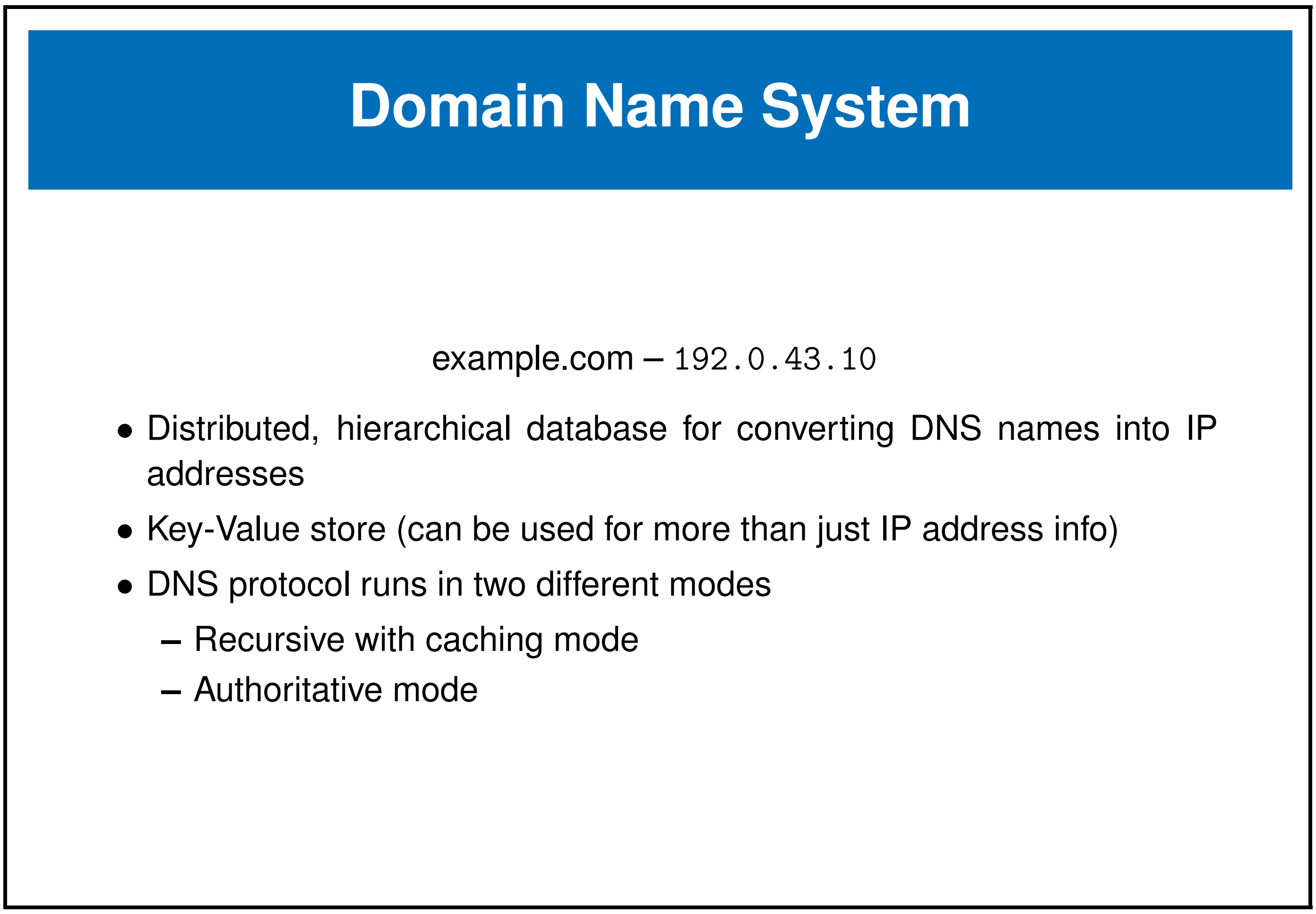


Figure 2.13: **VLAN: Virtual Local Area Network**

- A method for combining two or more separated LANs to appear as the same LAN
- Also a method for securing two or more LANs from each other on the same set of switches

2.6 Domain Name System



Domain Name System

example.com – `192.0.43.10`

- Distributed, hierarchical database for converting DNS names into IP addresses
- Key-Value store (can be used for more than just IP address info)
- DNS protocol runs in two different modes
  - Recursive with caching mode
  - Authoritative mode

When a network node makes a DNS query it most often makes that query against a recursive, caching server. That recursive, caching server will then make a recursive query through the DNS database until it comes to an authoritative server. The authoritative server will then send the answer for the query.

DNS Database

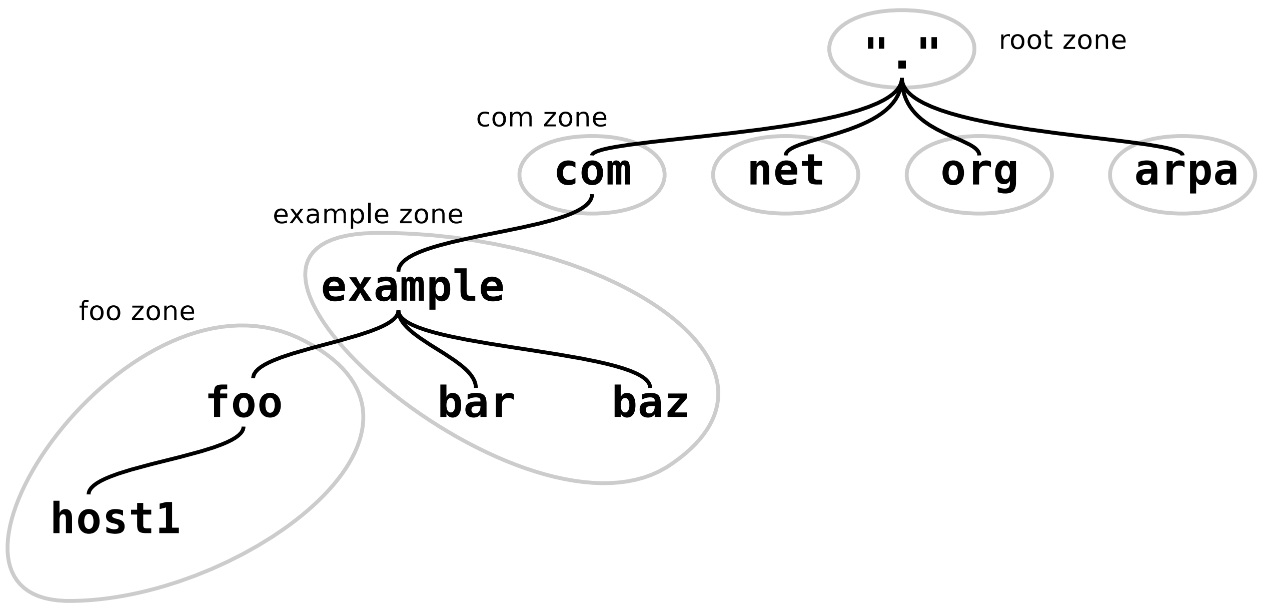


Figure 2.14: **DNS Database**

The DNS database consists of a tree-like, key-value store. The database is broken into tree nodes called **domains**. These domains are managed as part of a **zone**. Zones are the area of the namespace managed by authoritative server(s).

Recursive DNS Query

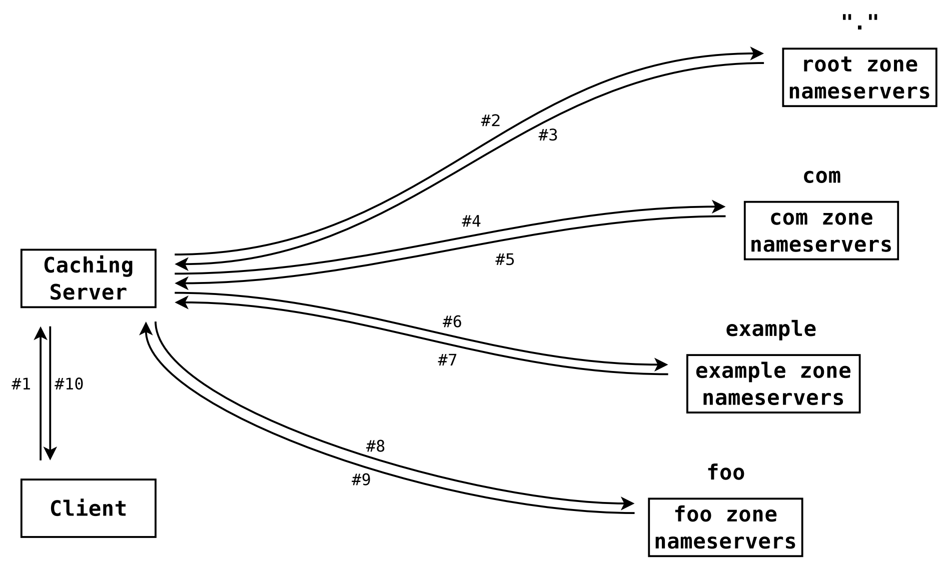


Figure 2.15: **Recursive DNS Query**

A theoretical recursive DNS query for `host1.foo.example.com.` would take the following steps:

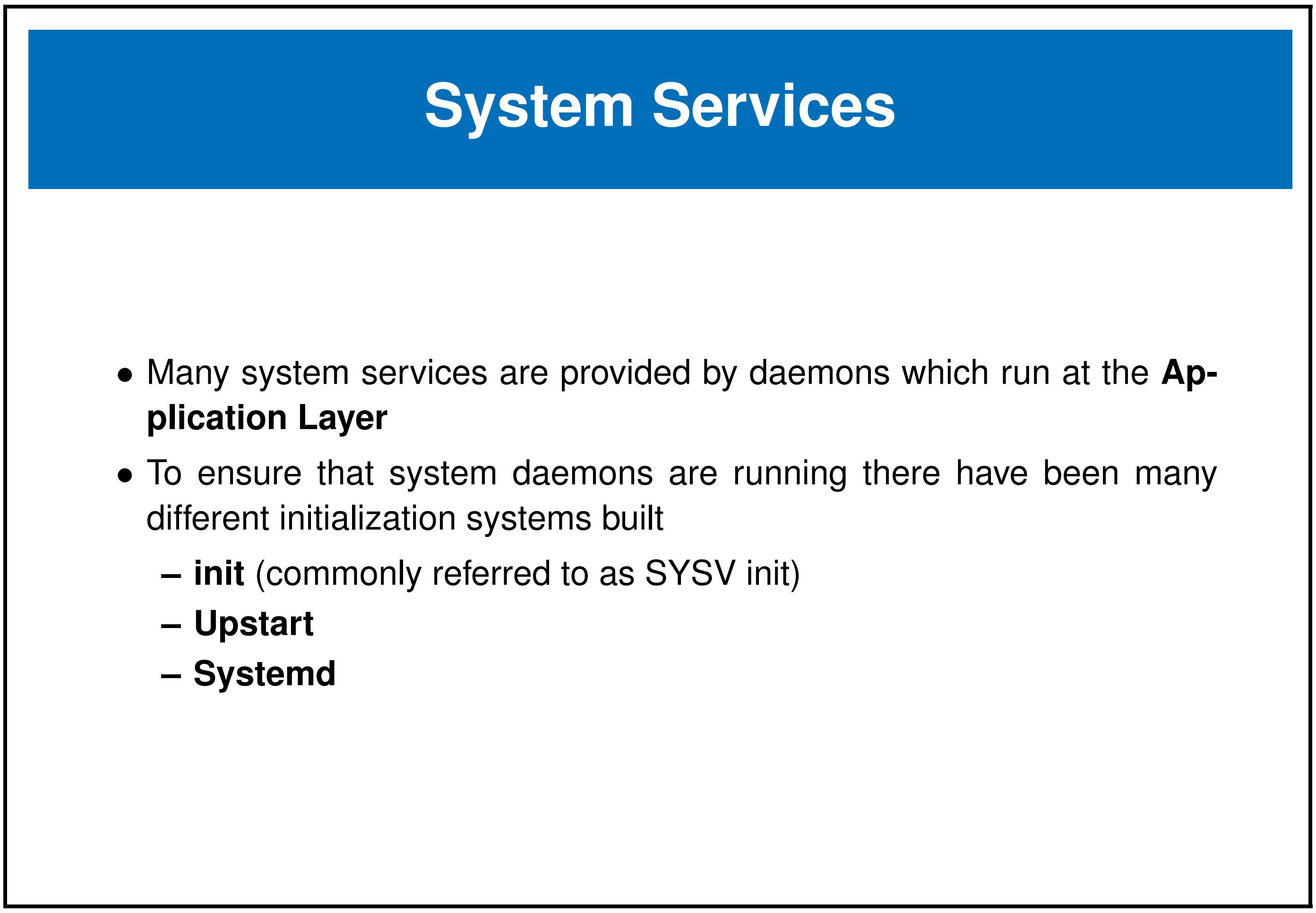
1. Client makes a recursive request to the caching nameserver it has configured.
2. The caching nameserver makes a query for `host1.foo.example.com.` to the root (`"."`) zone nameserver.
3. The root (`"."`) zone nameserver points to the nameserver for the `com.` zone.
4. The caching nameserver makes a query for `host1.foo.example.com.` to the nameserver for the `com.` zone.
5. The `com.` zone nameserver sends a response that points to the nameserver for the `example.com.` zone.
6. The caching nameserver makes a query for `host1.foo.example.com.` to the nameserver for the `example.com.` zone.
7. The `example.com.` nameserver sends a response that points to the nameserver for the `foo.example.com.` zone.
8. The caching nameserver makes a query for `host1.foo.example.com.` to the nameserver for the `foo.example.com.` zone.
9. The `foo.example.com.` nameserver responds as the authoritatively for the address `host1.foo.example.com.` to the caching nameserver.
10. The caching nameserver stores all of these queries and their responses in a cache and responds back to the client with the answer to the original query.

DNS Tools

- Servers
  - **BIND** (**B**erkeley **I**nternet **N**ame **D**omain)
- Clients
  - **dig**: lots of info for debugging
  - **host**: simple interface for DNS queries
  - **nslookup**: deprecated tool

If you are familiar with **nslookup** you should learn one of the other tools. The Internet Systems Consortium (ISC) who maintained **nslookup** has deprecated the tool in favor of **dig** and **host**.

2.7 System Services



System Services

- Many system services are provided by daemons which run at the **Application Layer**
- To ensure that system daemons are running there have been many different initialization systems built
  - **init** (commonly referred to as SYSV init)
  - **Upstart**
  - **Systemd**

System V Init Scripts

- Shell scripts used to start services
- Serial execution
- Dependency management done manually

- The execution order and dependencies for SYSV init scripts are managed by renaming scripts to run in proper order.
- Long boot times due to serial execution