

# TCSS 558: APPLIED DISTRIBUTED COMPUTING

## Processes: Virtualization, Clients & Servers

Wes J. Lloyd  
School of Engineering  
& Technology (SET)  
University of Washington - Tacoma

The diagram illustrates several network topologies: Star (a central node connected to multiple peripheral nodes), Mesh (every node connected to every other node), Ring (nodes connected in a closed loop), Fully Connected (every node connected to every other node), Tree (a hierarchical structure of nodes), Bus (all nodes connected to a single central line), and Line (nodes connected in a straight line). Below these diagrams is a 3D illustration of a central server connected to multiple client laptops.

1

## OBJECTIVES – 2/1

■ Questions from 2/1

■ Assignment 2: Key/Value Store

■ Coming Soon

■ Midterm Thursday February 8

■ 2nd hour - Tuesday February 6 – practice midterm class activity

■ Chapter 3: Processes

■ Chapter 3.1: Threads

■ Threading Models

■ Multithreaded clients/servers

■ Chapter 3.2: Virtualization

■ Chapter 3.3: Clients

■ Chapter 3.4: Servers

February 1, 2024

TCSS558: Applied Distributed Computing [Winter 2024]  
School of Engineering and Technology, University of Washington - Tacoma

L9.2

2

ONLINE DAILY FEEDBACK SURVEY

■ Daily Feedback Quiz in Canvas – Available After Each Class

■ Extra credit available for completing surveys **ON TIME**

■ Tuesday surveys: due by ~ Wed @ 10p

■ Thursday surveys: due ~ Mon @ 10p

TCSS 558 A > Assignments

Winter 2021

Home

Announcements

Assignments

Zoom

Chat

Search for Assignment

Upcoming Assignments

TCSS 558 - Online Daily Feedback Survey - 1/5

Not available until Jan 5 at 1:30pm | Due Jan 6 at 10pm | ~1 pts

February 1, 2024

TCSS558: Applied Distributed Computing [Winter 2024]  
School of Engineering and Technology, University of Washington - Tacoma

L9.3

3

TCSS 558 - Online Daily Feedback Survey - 1/5

Due Jan 6 at 10pm

Points 1

Questions 4

Available Jan 5 at 1:30pm - Jan 6 at 11:59pm 1 day

Time Limit None

Question 1

0.5 pts

On a scale of 1 to 10, please classify your perspective on material covered in today's class:

12345678910

Mostly Review To MeEqual New and ReviewMostly New to Me

Question 2

0.5 pts

Please rate the pace of today's class:

12345678910

SlowJust RightFast

February 1, 2024

TCSS558: Applied Distributed Computing [Winter 2024]  
School of Engineering and Technology, University of Washington - Tacoma

L9.4

4

Slides by Wes J. Lloyd

L9.2

MATERIAL / PACE

- Please classify your perspective on material covered in today’s class (25 respondents):
  - 1-mostly review, 5-equal new/review, 10-mostly new
  - **Average – 6.6** (↓ - *previous 6.77*)
- Please rate the pace of today’s class:
  - 1-slow, 5-just right, 10-fast
  - **Average – 5.8** (↑ - *previous 5.73*)

February 1, 2024

TCSS558: Applied Distributed Computing [Winter 2024]  
School of Engineering and Technology, University of Washington - Tacoma

L9.5

5

FEEDBACK FROM 1/30

- I asked about the 'one-to-one' model in class:
- Regarding the one-to-one threading model, what is the ratio of user threads to kernel threads ?
- Looking online it seems like an application can have multiple threads, and each thread has its own kernel threads. So 'one-to-one' refers to 'one-user-thread-to-one-kernel-thread'
- The "kworker" is a placeholder process for kernel worker threads under which the timer, interrupt processing, and other internal kernel (system) calls run.
- Looking more closely at kworker processes, it appears that kworkers are created for each physical CPU core
- And there are different types of kworkers: events, kdmflush, mm\_percpu\_wq, dio/dm-0
- I counted about 47 distinct kworker processes
- kworker processes appear to be shared. They are on stand-by to run system calls. If there is no system call, they remain idle.

`ps ax | grep kworker`

February 1, 2024

TCSS558: Applied Distributed Computing [Winter 2024]  
School of Engineering and Technology, University of Washington - Tacoma

L9.6

6

## FEEDBACK - 2

- **Helpful links on Linux processes and threads:**
- <https://www.baeldung.com/linux/process-vs-thread>
- <https://www.baeldung.com/linux/monitor-process-thread-count>
- **kworkers in Linux:**
- <https://medium.com/@boutnaru/the-linux-process-journey-kworker-f947634da73>
- <https://askubuntu.com/questions/33640/kworker-what-is-it-and-why-is-it-hogging-so-much-cpu>

February 1, 2024

TCSS558: Applied Distributed Computing [Winter 2024]  
School of Engineering and Technology, University of Washington - Tacoma

L9.7

7

## AWS CLOUD CREDITS UPDATE

- We have been approved to receive AWS CLOUD CREDITS for TCSS 558 – Winter 2024
- Credits will be provided by email request
  - Please include: 12-digit AWS account ID, and AWS account email
- Credits will first be provided for students not in F'23 TCSS562
- Request codes by sending an email with the subject: **“AWS CREDIT REQUEST”** to [wllloyd@uw.edu](mailto:wllloyd@uw.edu)
- Codes can also be obtained in person (or zoom), in the class, during the breaks, after class, during office hours, by appt
- Credit codes are carefully exchanged, and not shared by IM
- For students **unable to create a standard AWS account**:  
Please contact instructor by email -  
*Instructor will work to create hosted IAM user account*

February 1, 2024

TCSS558: Applied Distributed Computing [Winter 2024]  
School of Engineering and Technology, University of Washington - Tacoma

L9.8

8

ASSIGNMENT 1

- **Preparing for Assignment 1:**  
**Intro to Cloud Computing Infrastructure and Load Balancing**
  - Establish AWS Account - Standard account
- **Now posted:**
  - Task 0 - Establish local Linux/Ubuntu environment
  - Task 1 –AWS account setup, obtain user credentials
  - Task 2 – Intro to: Amazon EC2 & Docker: create Dockerfile for Apache Tomcat
  - Task 3 – Create Dockerfile for haproxy (software load balancer)
  - Task 4 – Working with Docker-Machine
  - Task 5 – Submit Results of testing alternate server configs

February 1, 2024

TCSS558: Applied Distributed Computing [Winter 2024]  
School of Engineering and Technology, University of Washington - Tacoma

L9.9

9

TESTING CONNECTIVITY TO SERVER (PG 16-18)

- **testFibPar.sh** script is a parallel test script
- Orchestrates multiple threads on client to invoke server multiple times in parallel
- To simplify coordination of parallel service calls in BASH, **testFibPar.sh** script ignores errors !!!
- To help test client-to-server connectivity, there is also a **testFibService.sh** script that supports 3 tests
- **TEST 1: Network layer test**
  - Ping (ICMP)
- **TEST 2: Transport layer test**
  - TCP: telnet (TCP Port 8080) – security group (fw) test
- **TEST 3: Application layer test**
  - HTTP REST – web service test

Application set

Transportation set

OSI Model Layers

Application  
Presentation  
Session  
Transport  
Network  
Data Link  
Physical

February 1, 2024

TCSS558: Applied Distributed Computing [Winter 2024]  
School of Engineering and Technology, University of Washington - Tacoma

L8.10

10

OBJECTIVES – 2/1

- Questions from 2/1
- **Assignment 2: Key/Value Store**
  - Coming Soon
- Midterm Thursday February 8
  - 2nd hour - Tuesday February 6 – practice midterm class activity
- Chapter 3: Processes
  - Chapter 3.1: Threads
    - Threading Models
    - Multithreaded clients/servers
  - Chapter 3.2: Virtualization
  - Chapter 3.3: Clients
  - Chapter 3.4: Servers

February 1, 2024

TCCS558: Applied Distributed Computing [Winter 2024]  
School of Engineering and Technology, University of Washington - Tacoma

L9.11

11

OBJECTIVES – 2/1

- Questions from 2/1
- Assignment 2: Key/Value Store
  - Coming Soon
- **Midterm Thursday February 8**
  - **2nd hour - Tuesday February 6 – practice midterm class activity**
- Chapter 3: Processes
  - Chapter 3.1: Threads
    - Threading Models
    - Multithreaded clients/servers
  - Chapter 3.2: Virtualization
  - Chapter 3.3: Clients
  - Chapter 3.4: Servers

February 1, 2024

TCCS558: Applied Distributed Computing [Winter 2024]  
School of Engineering and Technology, University of Washington - Tacoma

L9.12

12

OBJECTIVES – 2/1

- Questions from 2/1
- Assignment 2: Key/Value Store
  - Coming Soon
- Midterm Thursday February 8
  - 2nd hour - Tuesday February 6 – practice midterm class activity
- Chapter 3: Processes
  - Chapter 3.1: Threads
    - Threading Models
    - Multithreaded clients/servers
  - **Chapter 3.2: Virtualization**
  - Chapter 3.3: Clients
  - Chapter 3.4: Servers

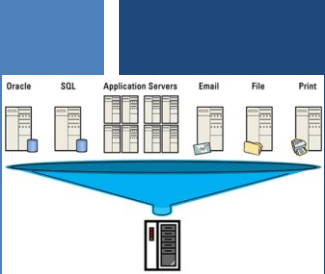
February 1, 2024

TCSS558: Applied Distributed Computing [Winter 2024]  
School of Engineering and Technology, University of Washington - Tacoma

L9.13

13

CH. 3.2:  
VIRTUALIZATION




L9.14

14

## VIRTUALIZATION

- Initially introduced in the 1970s on IBM mainframe computers
- Legacy operating systems run in mainframe-based VMs
- Legacy software could be sustained by virtualizing legacy OSes
- 1970s virtualization went away as desktop/rack-based hardware became inexpensive
- Virtualization reappears in 2000s to leverage multi-core, multi-CPU processor systems
- VM-Ware virtual machines enable companies to host many virtual servers with mixed OSes on private clusters
- Cloud computing: Amazon offers VMs as-a-service (IaaS)

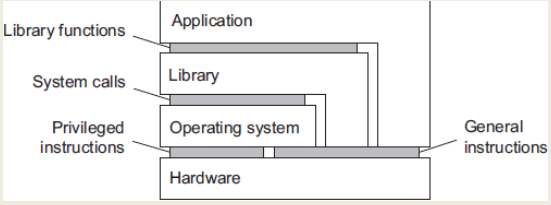


February 1, 2024	TCSS558: Applied Distributed Computing [Winter 2024] School of Engineering and Technology, University of Washington - Tacoma	L9.15
------------------	---	-------

15

## TYPES OF VIRTUALIZATION

- Levels of instructions:**
  - Hardware: CPU**
    - Privileged instructions  
**KERNEL MODE**
    - General instructions  
**USER MODE**
  - Operating system:** system calls
  - Library:** programming APIs: e.g. C/C++, C#, Java libraries
  - Application:**
- Goal of virtualization:**  
mimic these interface to provide a virtual computer



February 1, 2024	TCSS558: Applied Distributed Computing [Winter 2024] School of Engineering and Technology, University of Washington - Tacoma	L9.16
------------------	---	-------

16



TYPES OF VIRTUALIZATION - 2

■ **Process virtual machine**

■ Interpret instructions: (interpreters)  
(JavaVM) byte code → HW instructions

■ Emulate instructions: (emulators)  
(Wine) windows code → Linux code

■ **Native virtual machine monitor (VMM)**

■ Hypervisor (XEN): small OS with its own kernel

■ Provides an interface for multiple guest OSes

■ Facilitates sharing/scheduling of CPU, device I/O among many guests

■ Guest OSes require special kernel to interface w/ VMM

■ Supports **Paravirtualization** for performance boost to run code directly on the CPU

■ Type 1 hypervisor

Application/Libraries

Runtime system

Operating system

Hardware

Application/Libraries

Operating system

Virtual machine monitor

Hardware

February 1, 2024

TCSS558: Applied Distributed Computing [Winter 2024]  
School of Engineering and Technology, University of Washington - Tacoma

L9.17

17

TYPES OF VIRTUALIZATION - 3

■ **Hosted virtual machine monitor (VMM)**

■ Runs atop of hosted operating system

■ Uses host OS facilities for CPU scheduling, I/O

■ Full virtualization

■ Type 2 hypervisor

■ **Virtualbox**

■ *Textbook: note 3.5–good explanation of full vs. paravirtualization*

■ **GOAL:** run all user mode instructions directly on the CPU

■ x86 instruction set has ~17 privileged user mode instructions

■ **Full virtualization:** scan the EXE, insert code around privileged instructions to divert control to the VMM

■ **Paravirtualization:** special OS kernel eliminates side effects of privileged instructions

Application/Libraries

Operating system

Virtual machine monitor

Operating system

Hardware

February 1, 2024

TCSS558: Applied Distributed Computing [Winter 2024]  
School of Engineering and Technology, University of Washington - Tacoma

L9.18

18

Slides by Wes J. Lloyd

L9.9

EVOLUTION OF AWS VIRTUALIZATION

From <http://www.brendangregg.com/blog/2017-11-29/aws-ec2-virtualization-2017.html>

AWS EC2 Virtualization Types

VS:  
Virtualization  
In software

P:  
Paravirtual

VH:  
Virtualization  
In Hardware

H:  
Hardware

Most

Importance

Least

CPU, Memory

Network I/O

Local Storage I/O

Remote Storage I/O

Interrupts, Timers

Motherboard, Boot

#	Tech	Type	With						
1	VM	Fully Emulated		VS	VS	VS	VS	VS	VS
2	VM	Xen PV 3.0	PV drivers	P	P	P	P	VS	VS
3	VM	Xen HVM 3.0	PV drivers	VH	P	P	P	VS	VS
4	VM	Xen HVM 4.0.1	PVHVM drivers	VH	P	P	P	P	VS
5	VM	Xen AWS 2013	PVHVM + SR-IOV(net)	VH	VH	P	P	P	VS
6	VM	Xen AWS 2017	PVHVM + SR-IOV(net, stor.)	VH	VH	VH	P	P	VS
7	VM	AWS Nitro 2017		VH	VH	VH	VH	VH	VS
8	HW	AWS Bare Metal 2017		H	H	H	H	H	H
		Bare Metal		H	H	H	H	H	H

VM: Virtual Machine. HW: Hardware.

VS: Virt. in software. VH: Virt. in hardware. P: Paravirt. Not all combinations shown.

SR-IOV(net): ixgbe/ena driver. SR-IOV(storage): nvme driver.

February 1, 2024

TCSS558: Applied Distributed Computing [Winter 2024]  
School of Engineering and Technology, University of Washington - Tacoma

L9.19

19

AWS VIRTUALIZATION - 2

Full Virtualization - Fully Emulated

Never used on EC2, before CPU extensions for virtualization

Can boot any unmodified OS

Support via slow emulation, performance 2x-10x slower

Paravirtualization: Xen PV 3.0

Software: Interrupts, timers

Paravirtual: CPU, Network I/O, Local+Network Storage

Requires special OS kernels, interfaces with hypervisor for I/O

Performance 1.1x – 1.5x slower than “bare metal”

Instance store instances: 1<sup>ST</sup> & 2<sup>nd</sup> generation- m1.large, m2.xlarge

Xen HVM 3.0

Hardware virtualization: CPU, memory (CPU VT-x required)

Paravirtual: network, storage

Software: interrupts, timers

EBS backed instances

m1, c1 instances

February 1, 2024

TCSS558: Applied Distributed Computing [Winter 2024]  
School of Engineering and Technology, University of Washington - Tacoma

L9.20

20

Slides by Wes J. Lloyd

L9.10

AWS VIRTUALIZATION - 3

- **XEN HVM 4.0.1**
  - Hardware virtualization: CPU, memory (**CPU VT-x required**)
  - Paravirtual: network, storage, **Interrupts, timers**
- **XEN AWS 2013** (*diverges from opensource XEN*)
  - Provides hardware virtualization for CPU, memory, **network**
  - Paravirtual: storage, **Interrupts, timers**
  - Called Single root I/O Virtualization (SR-IOV)
  - Allows sharing single physical PCI Express device (i.e. network adapter) with multiple VMs
  - Improves VM network performance
  - 3<sup>rd</sup> & 4<sup>th</sup> generation instances (c3 family)
  - Network speeds up to 10 Gbps and 25 Gbps
- **XEN AWS 2017**
  - Provides hardware virtualization for CPU, memory, network, **local disk**
  - Paravirtual: remote storage, **Interrupts, timers**
  - Introduces hardware virtualization for EBS volumes (c4 instances)
  - Instance storage hardware virtualization (x1.32xlarge, i3 family)

February 1, 2024

TCSS558: Applied Distributed Computing [Winter 2024]  
School of Engineering and Technology, University of Washington - Tacoma

L9.21

21

AWS VIRTUALIZATION - 4

- **AWS Nitro 2017**
  - Provides hardware virtualization for CPU, memory, network, **local disk, remote disk, Interrupts, timers**
  - All aspects of virtualization enhanced with HW-level support
  - November 2017
  - Goal: provide performance indistinguishable from “bare metal”
  - 5<sup>th</sup> generation instances – c5 instances (also c5d, c5n)
  - Based on KVM hypervisor
  - Overhead around ~1%

February 1, 2024

TCSS558: Applied Distributed Computing [Winter 2024]  
School of Engineering and Technology, University of Washington - Tacoma

L9.22

22

## OBJECTIVES - 2/1

- Questions from 2/1
- Assignment 2: Key/Value Store
  - Coming Soon
- Midterm Thursday February 8
  - 2nd hour - Tuesday February 6 – practice midterm class activity
- Chapter 3: Processes
  - Chapter 3.1: Threads
    - Threading Models
    - Multithreaded clients/servers
  - Chapter 3.2: Virtualization
  - **Chapter 3.3: Clients**
  - Chapter 3.4: Servers

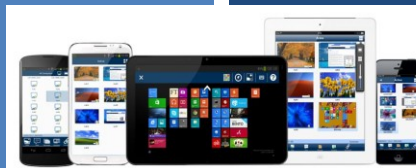
February 1, 2024

TCSS558: Applied Distributed Computing [Winter 2024]  
School of Engineering and Technology, University of Washington - Tacoma

L9.23

23

## CH. 3.3: CLIENTS



L9.24

24

## TYPES OF CLIENTS

- **Thick clients**
  - Web browsers
    - Client-side scripting
  - Mobile apps
  - Multi-tier MVC apps
- **Thin clients**
  - Remote desktops/GUIs (very thin)

February 1, 2024

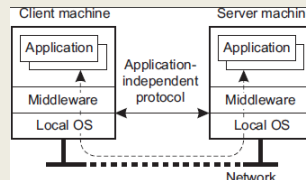
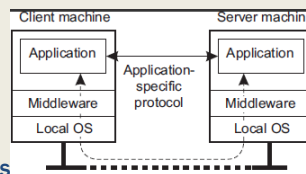
TCSS558: Applied Distributed Computing [Winter 2024]  
School of Engineering and Technology, University of Washington - Tacoma

L9.25

25

## CLIENTS

- **Application specific protocol**
  - Thick clients
  - Clients maintain local data
  - Middleware (APIs)
  - Clients synchronize data with remote nodes
  - Example: shared calendar application
- **Application independent**
  - Thin clients
  - Client acts as a remote terminal
  - Provides interface to user (GUI / UI)
  - Server houses entire application stack



February 1, 2024

TCSS558: Applied Distributed Computing [Winter 2024]  
School of Engineering and Technology, University of Washington - Tacoma

L9.26

26

# X WINDOWS

- Layered architecture to transport UI over network
- Remote desktop functionality for Linux/Unix systems
- X kernel acts as a server
  - Provides the X protocol: application level protocol
  - Xlib instances (client applications) exchange data and events with X kernels (servers)
  - Clients and servers on single machine → Linux GUI
  - Client and server communication transported over the network → remote Linux GUI

February 1, 2024

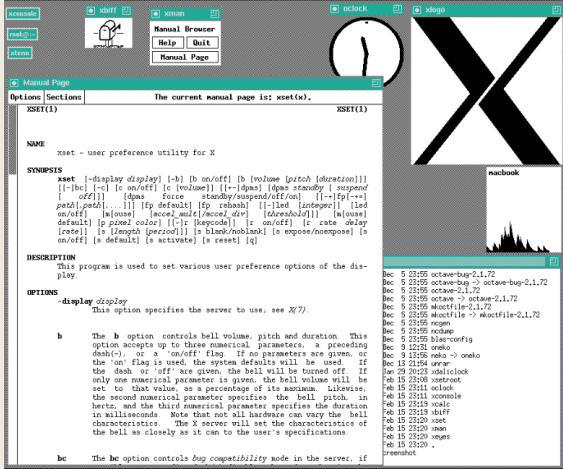
TCSS558: Applied Distributed Computing [Winter 2024]  
School of Engineering and Technology, University of Washington - Tacoma

L9.27

27

# X WINDOWS - 2

- Window manager:
  - Application running atop of X-windows which provides flair
  - Many variants
  - Without X windows is quite bland



February 1, 2024

TCSS558: Applied Distributed Computing [Winter 2024]  
School of Engineering and Technology, University of Washington - Tacoma

L9.28

28

- **Layered architecture**
- **X-kernel: low level interface/APIs for controlling screen, capturing keyboard and mouse events (X window Server)**
- **Provided on Linux as Xlib**
- **Provides network enabled GUI**
- **Layering allows for use for custom window managers**

Application Clients - User Productivity  
OpenOffice.org, Firefox, Gimp

Desktop Environment - Application and File Management  
Gnome/KDE panels, desktop icon managers

Window and Compositing Manager - Placement and Controls Of Windows  
Compiz, Metacity, kwin

Session Manager  
gnome-session, ksmserver

Display Manager - Local X Server Startup and User Authentication  
gdm, kdm, xdm

X Window Server - Display Hardware Management  
Xorg

Network Transports - Client -Server Connections  
TCP/IP, Unix domain sockets

Toolkits  
GTK, Qt, Motif, Xaw

February 1, 2024

TCSS558: Applied Distributed Computing [Winter 2024]  
School of Engineering and Technology, University of Washington - Tacoma

L9.29

29

EXAMPLE: VNC SERVER

- **How to Install VNC server on Ubuntu EC2 Instance VM:**
- `sudo apt-get update`
- `# ubuntu 16.04`
- `sudo apt-get install ubuntu-desktop`
- `sudo apt-get install gnome-panel gnome-settings-daemon metacity nautilus gnome-terminal`
- `# on ubuntu 18.04`
- `sudo apt install xfce4 xfce4-goodies`
- `sudo apt-get install tightvncserver # both`
- **Start VNC server to create initial config file**
- `vncserver :1`

February 1, 2024

TCSS558: Applied Distributed Computing [Winter 2024]  
School of Engineering and Technology, University of Washington - Tacoma

L9.30

30

EXAMPLE: VNC SERVER – UBUNTU 16.04

- On the VM: edit config file: `nano ~/.vnc/xstartup`
- Replace contents as below (Ubuntu 16.04):

```
#!/bin/sh

export XKL_XMODMAP_DISABLE=1
unset SESSION_MANAGER
unset DBUS_SESSION_BUS_ADDRESS

[ -x /etc/vnc/xstartup ] && exec /etc/vnc/xstartup
[ -r $HOME/.Xresources ] && xrdp $HOME/.Xresources
xsetroot -solid grey

vncconfig -iconic &
gnome-panel &
gnome-settings-daemon &
metacity &
nautilus &
gnome-terminal &
```

February 1, 2024

TCSS558: Applied Distributed Computing [Winter 2024]  
School of Engineering and Technology, University of Washington - Tacoma

L9.31

31

EXAMPLE: VNC SERVER – UBUNTU 18.04

- On the VM:
- Edit config file: `nano ~/.vnc/xstartup`
- Replace contents as below (Ubuntu 18.04):

```
#!/bin/bash
xrdp $HOME/.Xresources
startxfce4 &
```

February 1, 2024

TCSS558: Applied Distributed Computing [Winter 2024]  
School of Engineering and Technology, University of Washington - Tacoma

L9.32

32



VNC SERVER - UBUNTU 20.04 - GNOME

```
# install vnc server
sudo apt install tigervnc-standalone-server
sudo apt install ubuntu-gnome-desktop
vncserver :1          # creates a config file
vncserver -kill :1    # stop server
vi ~/.vnc/xstartup     # edit config file

#!/bin/sh
# Start Gnome 3 Desktop
[ -x /etc/vnc/xstartup ] && exec /etc/vnc/xstartup
[ -r $HOME/.Xresources ] && xrdp $HOME/.Xresources
vncconfig -iconic &
dbus-launch --exit-with-session gnome-session &

sudo systemctl start gdm      # start gnome desktop
sudo systemctl enable gdm
vncserver :1                  # restart vnc server
```

February 1, 2024

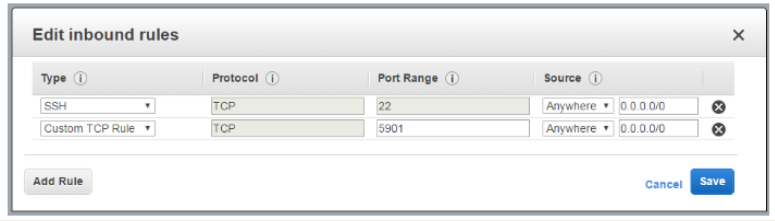
TCSS558: Applied Distributed Computing [Winter 2024]  
School of Engineering and Technology, University of Washington - Tacoma

L9.33

33

EXAMPLE: VNC SERVER - 3

- On the VM: reload config by restarting server
- `vncserver -kill :1`
- `vncserver :1`
- Open port 22 & 5901 in EC2 security group:



February 1, 2024

TCSS558: Applied Distributed Computing [Winter 2024]  
School of Engineering and Technology, University of Washington - Tacoma

L9.34

34

EXAMPLE: VNC CLIENT

- On the client (e.g. laptop):
- Create SSH connection to securely forward port 5901 on the EC2 instance to your localhost port 5901
- This way your VNC client doesn't need an SSH key

```
ssh -i <ssh-keyfile> -L 5901:127.0.0.1:5901 -N -f -l <username> <EC2-instance ip_address>
```

- For example:

```
ssh -i mykey.pem -L 5901:127.0.0.1:5901 -N -f -l ubuntu 52.111.202.44
```

February 1, 2024

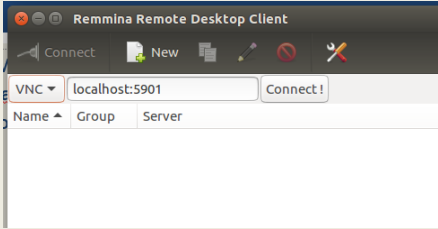
TCSS558: Applied Distributed Computing [Winter 2024]  
School of Engineering and Technology, University of Washington - Tacoma

L9.35

35

EXAMPLE: VNC CLIENT - 2

- On the client (e.g. laptop):
- Use a VNC Client to connect
- Remmina is provided by default on Ubuntu 16.04
- Can “google” for many others
- Remmina login:
- Chose “VNC” protocol
- Log into “localhost:5901”



February 1, 2024

TCSS558: Applied Distributed Computing [Winter 2024]  
School of Engineering and Technology, University of Washington - Tacoma

L9.36

36

REMOTE COMPUTER IN THE CLOUD

■ EC2 instance with a GUI. . .!!!

February 1, 2024

TCSS558: Applied Distributed Computing [Winter 2024]  
School of Engineering and Technology, University of Washington - Tacoma

L9.37

37

THIN CLIENTS

■ Thin clients

■ Besides VNC, there are a variety of other remote desktop protocols:

Remote desktop [protocols](#) include the following:

- [Apple Remote Desktop Protocol \(ARD\)](#) – Original protocol for [Apple Remote Desktop](#) on [macOS](#) machines.
- [Appliance Link Protocol \(ALP\)](#) – a [Sun Microsystems](#)-specific protocol featuring audio (play and record), remote printing, remote USB, accelerated video
- [HP Remote Graphics Software \(RGS\)](#) – a [proprietary protocol](#) designed by [Hewlett-Packard](#) specifically for high end workstation remoting and collaboration.
- [Independent Computing Architecture \(ICA\)](#) – a [proprietary protocol](#) designed by [Citrix Systems](#)
- [NX technology \(NoMachine NX\)](#) – Cross platform protocol featuring audio, video, remote printing, remote USB, H264-enabled.
- [PC-over-IP \(PCoIP\)](#) – a [proprietary protocol](#) used by [VMware](#) (licensed from [Teradici](#))<sup>[2]</sup>
- [Remote Desktop Protocol \(RDP\)](#) – a [Windows](#)-specific protocol featuring audio and remote printing
- [Remote Frame Buffer Protocol \(RFB\)](#) – A framebuffer level [cross-platform](#) protocol that [VNC](#) is based on.
- [SPICE \(Simple Protocol for Independent Computing Environments\)](#) – remote-display system built for virtual environments by [Qumranet](#), now [Red Hat](#)
- [Splashtop](#) – a high performance remote desktop protocol developed by Splashtop, fully optimized for hardware (H.264) including Intel / AMD chipsets, NVIDIA of media codecs, Splashtop can deliver high frame rates with low latency, and also low power consumption.
- [X Window System \(X11\)](#) – a well-established cross-platform protocol mainly used for displaying local applications; X11 is [network transparent](#)

February 1, 2024

TCSS558: Applied Distributed Computing [Winter 2024]  
School of Engineering and Technology, University of Washington - Tacoma

L9.38

38

## THIN CLIENTS - 2

- Applications should separate application logic from UI
- When application logic and UI interaction are tightly coupled many requests get sent to X kernel
- Client must wait for response
- Synchronous behavior and app-to-UI coupling adversely affects performance of WAN / Internet
- **Protocol optimizations:** reduce bandwidth by shrinking size of X protocol messages
- Send only differences between messages with same identifier
- Optimizations enable connections with 9600 kbps

February 1, 2024

TCSS558: Applied Distributed Computing [Winter 2024]  
School of Engineering and Technology, University of Washington - Tacoma

L9.39

39

## THIN CLIENTS - 3

- Virtual network computing (VNC)
- Send display over the network at the pixel level (instead of X lib events)
- Reduce pixel encodings to save bandwidth – fewer colors
- Pixel-based approaches loose application semantics
- Can transport any GUI this way
- **THING**- hybrid approach
- Send video device driver commands over network
- More powerful than pixel based operations
- Less powerful compared to protocols such as X

February 1, 2024

TCSS558: Applied Distributed Computing [Winter 2024]  
School of Engineering and Technology, University of Washington - Tacoma

L9.40

40

TRADEOFFS: ABSTRACTION OF REMOTE  
DISPLAY PROTOCOLS

▪ Tradeoff space: abstraction level of remote display protocols

Pixel-level

VNC

Graphics lib

X11

February 1, 2024

TCSS558: Applied Distributed Computing [Winter 2024]  
School of Engineering and Technology, University of Washington - Tacoma

L9.41

41

TRADEOFFS: ABSTRACTION OF REMOTE  
DISPLAY PROTOCOLS

▪ Tradeoff space: abstraction level of remote display protocols

Pixel-level

VNC

Graphics lib

X11

• Generic – no app context

• Graphics data

• Higher network bandwidth

• Fewer colors

• Utilize graphics compression

• More network traffic

• Application context is available

• UI data/operations

• Lower network bandwidth

• More colors

February 1, 2024

TCSS558: Applied Distributed Computing [Winter 2024]  
School of Engineering and Technology, University of Washington - Tacoma

L9.42

42

Slides by Wes J. Lloyd

L9.21

WE WILL RETURN AT  
5:05PM

43

CLIENT ROLES IN PROVIDING  
DISTRIBUTION TRANSPARENCY

- Clients help enable distribution transparency of servers
- Replication transparency
  - Client aggregates responses from multiple servers
  - Only the client knows of replicas

```
graph LR
    subgraph Client_machine [Client machine]
        direction TB
        C[Client appl]
    end
    subgraph Server_1 [Server 1]
        direction TB
        S1[Server appl]
    end
    subgraph Server_2 [Server 2]
        direction TB
        S2[Server appl]
    end
    subgraph Server_3 [Server 3]
        direction TB
        S3[Server appl]
    end
    C -- "Client side handles request replication" --> S1
    C -- "Client side handles request replication" --> S2
    C -- "Client side handles request replication" --> S3
    C -- "Replicated request" --> S1
```

February 1, 2024

TCSS558: Applied Distributed Computing [Winter 2024]  
School of Engineering and Technology, University of Washington - Tacoma

L9.44

44

CLIENT ROLES IN PROVIDING  
DISTRIBUTION TRANSPARENCY - 2

- Location/relocation/migration transparency
  - Harness convenient naming system to allow client to infer new locations
  - Server inform client of moves / Client reconnects to new endpoint
  - Client hides network address of server, and reconnects as needed
  - May involve temporary loss in performance
- Replication transparency
  - Client aggregates responses from multiple servers
- Failure transparency
  - Client retries, or maps to another server, or uses cached data
- Concurrency transparency
  - Transaction servers abstract coordination of multithreading

February 1, 2024

TCSS558: Applied Distributed Computing [Winter 2024]  
School of Engineering and Technology, University of Washington - Tacoma

L9.45

45

OBJECTIVES – 2/1

- Questions from 2/1
- Assignment 2: Key/Value Store
  - Coming Soon
- Midterm Thursday February 8
  - 2nd hour - Tuesday February 6 – practice midterm class activity
- Chapter 3: Processes
  - Chapter 3.1: Threads
    - Threading Models
    - Multithreaded clients/servers
  - Chapter 3.2: Virtualization
  - Chapter 3.3: Clients
  - Chapter 3.4: Servers

February 1, 2024

TCSS558: Applied Distributed Computing [Winter 2024]  
School of Engineering and Technology, University of Washington - Tacoma

L9.46

46



## CH. 3.4: SERVERS

L9.47

47

## SERVERS

- Cloud & Distributed Systems – rely on Linux
- <http://www.zdnet.com/article/it-runs-on-the-cloud-and-the-cloud-runs-on-linux-any-questions/>
- IT is moving to the cloud. And, what powers the cloud?
  - Linux
- Uptime Institute survey - 1,000 IT executives (2016)
  - 50% of IT executives – plan to migrate majority of IT workloads to off-premise to cloud or colocation sites
  - 23% expect the shift in 2017, 70% by 2020...
- Docker on Windows / Mac OS X
  - Based on Linux
  - Mac: Hyperkit Linux VM
  - Windows: Hyper-V Linux VM

February 1, 2024	TCSS558: Applied Distributed Computing [Winter 2024] School of Engineering and Technology, University of Washington - Tacoma	L9.48
------------------	---	-------

48



## SERVERS - 2

- Servers implement a specific service for a collection of clients
- Servers wait for incoming requests, and respond accordingly
- Server types
  - Iterative: immediately handle client requests
  - Concurrent: Pass client request to separate thread
- Multithreaded servers are concurrent servers
  - E.g. Apache Tomcat
- Alternative: fork a new process for each incoming request
- Hybrid: mix the use of multiple processes with thread pools

February 1, 2024

TCSS558: Applied Distributed Computing [Winter 2024]  
School of Engineering and Technology, University of Washington - Tacoma

L9.49

49

## END POINTS

- Clients connect to servers via:  
IP Address and Port Number
- How do ports get assigned?
  - Many protocols support “default” port numbers
  - Client must find IP address(es) of servers
  - A single server often hosts multiple end points (servers/services)
  - When designing new TCP client/servers must be careful not to repurpose ports already commonly used by others

February 1, 2024

TCSS558: Applied Distributed Computing [Winter 2024]  
School of Engineering and Technology, University of Washington - Tacoma

L9.50

50

COMMON PORTS

packetlife.net

TCP/UDP Port Numbers

7 Echo	554 RTSP	2745 Bagle.H	6891-6901 Windows Live
19 Chargen	546-547 DHCPv6	2967 Symantec AV	6970 Quicktime
20-21 FTP	560 rmonitor	3050 Interbase DB	7212 GhostSurf
22 SSH/SCP	563 NNTP over SSL	3074 XBOX Live	7648-7649 CU-SeeMe
23 Telnet	587 SMTP	3124 HTTP Proxy	8000 Internet Radio
25 SMTP	591 FileMaker	3127 MyDoom	8080 HTTP Proxy
42 WINS Replication	593 Microsoft DCOM	3128 HTTP Proxy	8086-8087 Kaspersky AV
43 WHOIS	631 Internet Printing	3222 GLBP	8118 Privoxy
49 TACACS	636 LDAP over SSL	3260 iSCSI Target	8200 VMware Server
53 DNS	639 MSDP (PIM)	3306 MySQL	8500 Adobe ColdFusion
67-68 DHCP/BOOTP	646 LDP (MPLS)	3389 Terminal Server	8767 TeamSpeak
69 TFTP	691 MS Exchange	3689 iTunes	8866 Bagle.B
70 Gopher	860 iSCSI	3690 Subversion	9100 HP JetDirect
79 Finger	873 rsync	3724 World of Warcraft	9101-9103 Bacula
80 HTTP	902 VMware Server	3784-3785 Ventrilo	9119 MXit
88 Kerberos	989-990 FTP over SSL	4333 mSQL	9800 WebDAV
102 MS Exchange	993 IMAP4 over SSL	4444 Blaster	9898 Dabber
110 POP3	995 POP3 over SSL	4664 Google Desktop	9988 Rbot/Spybot
113 Ident	1025 Microsoft RPC	4672 eMule	9999 Urchin
119 NNTP (Usenet)	1026-1029 Windows Messenger	4899 Radmin	10000 Webmin
123 NTP	1080 SOCKS Proxy	5000 UPnP	10000 BackupExec
135 Microsoft RPC	1080 MyDoom	5001 Slingbox	10113-10116 NetIQ
137-139 NetBIOS	1194 OpenVPN	5001 iperf	11371 OpenPGP
143 IMAP4	1214 Kazaa	5004-5005 RTP	12035-12036 Second Life
161-162 SNMP	1241 Nessus	5050 Yahoo! Messenger	12345 NetBus
177 XDMCP	1311 Dell OpenManage	5060 SIP	13720-13721 NetBackup
179 BGP	1337 WASTE	5190 AIM/ICQ	14567 Battlefield

51

TYPES OF SERVERS

- Daemon server
  - Example: NTP server
- Superserver
- Stateless server
  - Example: Apache server
- Stateful server
- Object servers
- EJB servers

February 1, 2024

TCSS558: Applied Distributed Computing [Winter 2024]  
School of Engineering and Technology, University of Washington - Tacoma

L9.52

52

## NTP EXAMPLE

- **Daemon servers**
  - Run locally on Linux
  - Track current server end points (outside servers)
  - **Example: network time protocol (ntp) daemon**
    - Listen locally on specific port (ntp is 123)
    - Daemons routes local client traffic to the configured endpoint servers
    - University of Washington: time.u.washington.edu
    - Example “`ntpq -p`”
      - Queries local ntp daemon, routes traffic to configured server(s)

February 1, 2024

TCSS558: Applied Distributed Computing [Winter 2024]  
School of Engineering and Technology, University of Washington - Tacoma

L9.53

53

## SUPERSERVER

- **Linux inetd / xinetd**
  - Single superserver
  - Extended internet service daemon
  - Not installed by default on Ubuntu
  - Intended for use on server machines
  - Used to configure box as a server for multiple internet services
    - E.g. ftp, pop, telnet
  - inetd daemon responds to multiple endpoints for multiple services
  - Requests fork a process to run required executable program
- **Check what ports you're listening on:**
  - `sudo netstat -tap | grep LISTEN`

February 1, 2024

TCSS558: Applied Distributed Computing [Winter 2024]  
School of Engineering and Technology, University of Washington - Tacoma

L9.54

54

## INTERRUPTING A SERVER

- **Server design issue:**
  - Active client/server communication is taking place over a port
  - How can the server / data transfer protocol support interruption?
- Consider transferring a 1 GB image, how do you pass a unrelated message in this stream?
  1. **Out-of-band** data: special messages sent in-stream to support interrupting the server (*TCP urgent data*)
  2. Use a separate connection (different port) for admin control info
- **Example: sftp secure file transfer protocol**
  - Once a file transfer is started, can't be stopped easily
  - Must kill the client and/or server

February 1, 2024

TCSS558: Applied Distributed Computing [Winter 2024]  
School of Engineering and Technology, University of Washington - Tacoma

L9.55

55

## STATELESS SERVERS

- Data about state of clients is not stored
- **Example:** web application servers are typically stateless
  - Also function-as-a-service (FaaS) platforms
- Many servers maintain information on clients (e.g. log files)
- Loss of stateless data doesn't disrupt server availability
  - Loosing log files typically has minimal consequences
- **Soft state:** server maintains state on the client for a limited time (*to support sessions*)
- Soft state information expires and is deleted

February 1, 2024

TCSS558: Applied Distributed Computing [Winter 2024]  
School of Engineering and Technology, University of Washington - Tacoma

L9.56

56

## STATEFUL SERVERS

- Maintain persistent information about clients
- Information must be explicitly deleted by the server
- Example:
  - File server - allows clients to keep local file copies for RW
- Server tracks client file permissions and most recent versions
  - Table of (client, file) entries
- If server crashes data must be recovered
- Entire state before a crash must be restored
- Fault tolerance - Ch. 8

February 1, 2024

TCSS558: Applied Distributed Computing [Winter 2024]  
School of Engineering and Technology, University of Washington - Tacoma

L9.57

57

## STATEFUL SERVERS - 2

- Session state
  - Tracks series of operations by a single user
  - Maintained temporarily, not indefinitely
  - Often retained for multi-tier client server applications
  - Minimal consequence if session state is lost
  - Clients must start over, reinitialize sessions
- Permanent state
  - Customer information, software keys
- Client-side cookies
  - When servers don't maintain client state, clients can store state locally in "cookies"
  - Cookies are not executable, simply client-side data

February 1, 2024

TCSS558: Applied Distributed Computing [Winter 2024]  
School of Engineering and Technology, University of Washington - Tacoma

L9.58

58

## OBJECT SERVERS

- **OBJECTIVE:** Host objects and enable remote client access
- Do not provide a specific service
  - Do nothing if there are no objects to host
- Support adding/removing hosted objects
- Provide a home where objects live
- Objects, *themselves*, provide “services”
- Object parts
  - State data
  - Code (methods, etc.)
- **Transient object(s)**
  - Objects with limited lifetime ( $<$  server)
  - Created at first invocation, destroyed when no longer used (i.e. no clients remain “bound”).
  - Disadvantage: initialization may be expensive
  - Alternative: preinitialize and retain objects on server start-up

February 1, 2024

TCSS558: Applied Distributed Computing [Winter 2024]  
School of Engineering and Technology, University of Washington - Tacoma

L9.59

59

## OBJECT SERVERS - 2

- **Should object servers isolate memory for object instances?**
  - Share neither code nor data
  - May be necessary if objects couple data and implementation
- Object server threading designs:
  - Single thread of control for object server
  - One thread for each object
  - Servers use separate thread for client requests
- Threads created on demand    **vs.**    Server maintains pool of threads
- **What are the tradeoffs for creating server threads on demand vs. using a thread pool?**

February 1, 2024

TCSS558: Applied Distributed Computing [Winter 2024]  
School of Engineering and Technology, University of Washington - Tacoma

L9.60

60

## EJB – ENTERPRISE JAVA BEANS

- EJB- specialized Java object hosted by a EJB web container
- 4 types: stateless, stateful, entity, and message-driven beans
- Provides “middleware” standard (framework) for implementing back-ends of enterprise applications
- EJB web application containers integrate support for:
  - Transaction processing
  - Persistence
  - Concurrency
  - Event-driven programming
  - Asynchronous method invocation
  - Job scheduling
  - Naming and discovery services (JNDI)
  - Interprocess communication
  - Security
  - Software component deployment to an application server

February 1, 2024

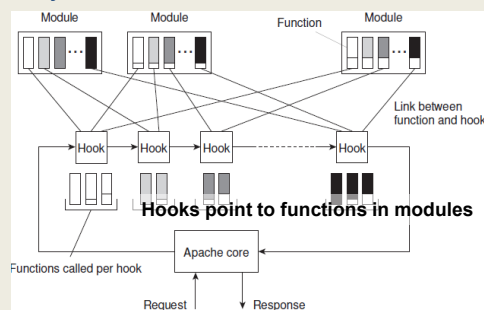
TCCS558: Applied Distributed Computing [Winter 2024]  
School of Engineering and Technology, University of Washington - Tacoma

L9.61

61

## APACHE WEB SERVER

- Highly configurable, extensible, platform independent
- Supports TCP HTTP protocol communication
- Uses hooks – placeholders for group of functions
- Requests processed in phases by hooks
- Many hooks:
  - Translate a URL
  - Write info to log
  - Check client ID
  - Check access rights
- Hooks processed in order enforcing flow-of-control
- Functions in replaceable modules



February 1, 2024

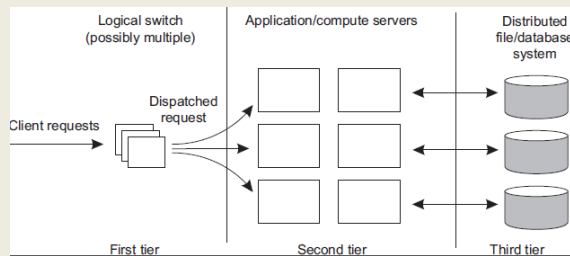
TCCS558: Applied Distributed Computing [Winter 2024]  
School of Engineering and Technology, University of Washington - Tacoma

L9.62

62

## SERVER CLUSTERS

- Hosted across an LAN or WAN
- Collection of interconnected machines
- Can be organized in tiers:
  - Web server → app server → DB server
  - App and DB server sometimes integrated



February 1, 2024

TCSS558: Applied Distributed Computing [Winter 2024]  
School of Engineering and Technology, University of Washington - Tacoma

L9.63

63

## LAN REQUEST DISPATCHING

- Front end of three tier architecture (logical switch) provides distribution transparency – hides multiple servers
- Transport-layer switches: switch accepts TCP connection requests, hands off to a server
  - Example: hardware load balancer (F5 networks – Seattle)
  - HW Load balancer - OSI layers 4-7
- Network-address-translation (NAT) approach:
  - All requests pass through switch
  - Switch sits in the middle of the client/server TCP connection
  - Maps (rewrites) source and destination addresses
- Connection hand-off approach:
  - TCP Handoff: switch hands off connection to a selected server

February 1, 2024

TCSS558: Applied Distributed Computing [Winter 2024]  
School of Engineering and Technology, University of Washington - Tacoma

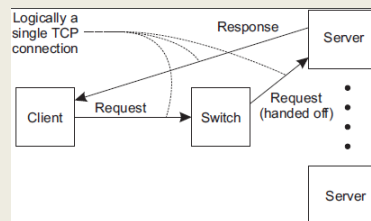
L9.64

64



## LAN REQUEST DISPATCHING - 2

- Who is the best server to handle the request?
- Switch plays important role in distributing requests
- Implements load balancing
- **Round-robin** – routes client requests to servers in a looping fashion
- **Transport-level** – route client requests based on TCP port number
- **Content-aware request distribution** – route requests based on inspecting data payload and determining which server node should process the request



February 1, 2024

TCSS558: Applied Distributed Computing [Winter 2024]  
School of Engineering and Technology, University of Washington - Tacoma

L9.65

65

## WIDE AREA CLUSTERS

- Deployed across the internet
- Leverage resource/infrastructure from Internet Service Providers (ISPs)
- Cloud computing simplifies building WAN clusters
- Resource from a single cloud provider can be combined to form a cluster
- **For deploying a cloud-based cluster (WAN), what are the implications of deploying nodes to:**
  - (1) a single availability zone (e.g. us-east-1e)?
  - (2) across multiple availability zones?

February 1, 2024

TCSS558: Applied Distributed Computing [Winter 2024]  
School of Engineering and Technology, University of Washington - Tacoma

L9.66

66

## WAN REQUEST DISPATCHING

- Goal: minimize network latency using WANs (e.g. Internet)
- Send requests to nearby servers
- Request dispatcher: routes requests to nearby server
- **Example:** Domain Name System
  - Hierarchical decentralized naming system
- Linux: find your DNS servers:

```
# Find you device name of interest
nmcli dev
# Show device configuration
nmcli device show <device name>
```

February 1, 2024

TCSS558: Applied Distributed Computing [Winter 2024]  
School of Engineering and Technology, University of Washington - Tacoma

L9.67

67

## DNS LOOKUP

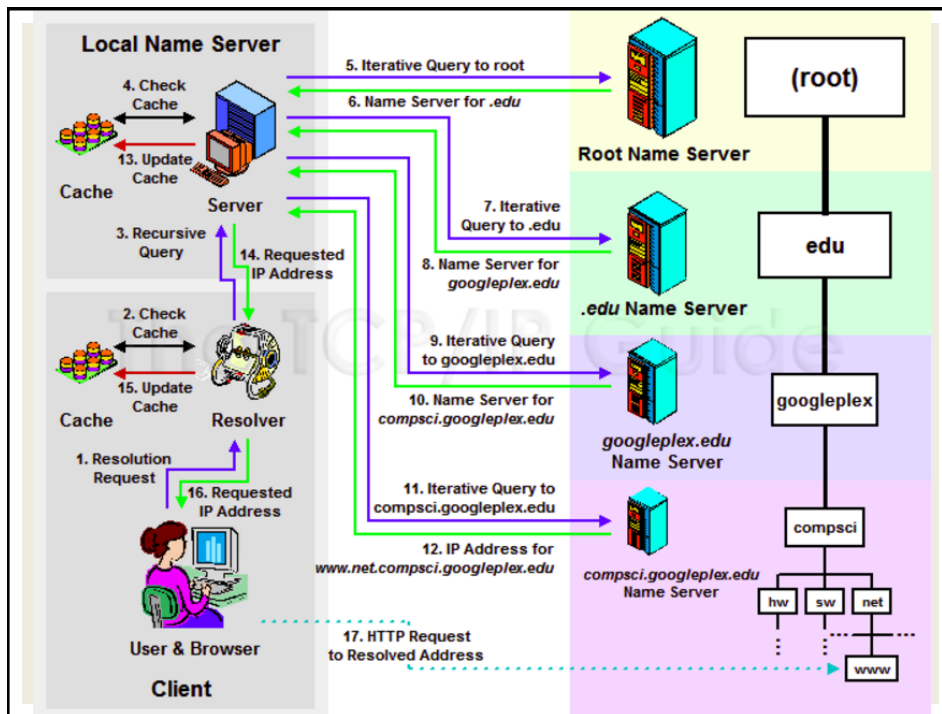
- First query local server(s) for address
- Typically there are (2) local DNS servers
  - One is backup
- Hostname may be cached at local DNS server
  - E.g. [www.google.com](http://www.google.com)
- If not found, local DNS server routes to other servers
- Routing based on components of the hostname
- DNS servers down the chain mask the client IP, and use the originating DNS server IP to identify a local host
- **Weakness:** *client may be far from DNS server used. Resolved hostname is close to DNS server, but not necessarily close to the client*

February 1, 2024

TCSS558: Applied Distributed Computing [Winter 2024]  
School of Engineering and Technology, University of Washington - Tacoma

L9.68

68



69

## DNS: LINUX COMMANDS

- `nslookup <ip addr / hostname>`
- Name server lookup - translates hostname or IP to the inverse
- `traceroute <ip addr / hostname>`
- Traces network path to destination
- By default, output is limited to 30 hops, can be increased

70

DNS EXAMPLE – WAN DISPATCHING

- Ping [www.google.com](http://www.google.com) in WA from wireless network:
  - nslookup: 6 alternate addresses returned, choose (74.125.28.147)
  - Ping 74.125.28.147: Average RTT = **22.458 ms (11 attempts, 22 hops)**
- Ping [www.google.com](http://www.google.com) in VA (us-east-1) from EC2 instance:
  - nslookup: 1 address returned, choose 172.217.9.196
  - Ping 172.217.9.196: Average RTT = 1.278 ms (11 attempts, 13 hops)
- From VA EC2 instance, ping WA [www.google](http://www.google.com) server
- Ping 74.125.28.147: Average RTT 62.349ms (11 attempts, 27 hops)
- Pinging the WA-local server is ~60x slower from VA
- From local wireless network, ping VA us-east-1 google :
- Ping 172.217.9.196: Average RTT=81.637ms (11 attempts, 15 hops)

February 1, 2024

TCSS558: Applied Distributed Computing [Winter 2024]  
School of Engineering and Technology, University of Washington - Tacoma

L9.71

71

DNS EXAMPLE – WAN DISPATCHING

- Ping [www.google.com](http://www.google.com) in WA from wireless network:
  - nslookup: 6 alternate addresses returned, choose (74.125.28.147)

**Latency to ping VA server in WA: ~3.63x**  
WA client: local-google 22.458ms to VA-google 81.637ms

**Latency to ping WA server in VA: ~48.7x**  
VA client: local-google 1.278ms to WA-google 62.349!

- From local wireless network, ping VA us-east-1 google :
- Ping 172.217.9.196: Average RTT=81.637ms (11 attempts, 15 hops)


February 1, 2024

TCSS558: Applied Distributed Computing [Winter 2024]  
School of Engineering and Technology, University of Washington - Tacoma

L9.72

72

# QUESTIONS



February 1, 2024

TCSS558: Applied Distributed Computing [Winter 2024]  
School of Engineering and Technology, University of Washington - Tacoma

L9.73