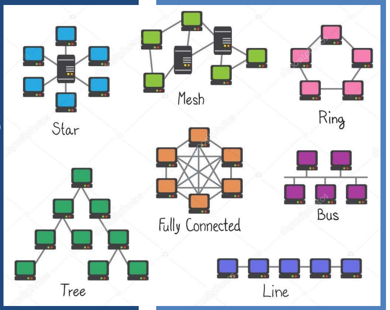
TCSS 558:
APPLIED DISTRIBUTED COMPUTING

Chapter 3 - Processes

Wes J. Lloyd
School of Engineering and Technology
University of Washington - Tacoma

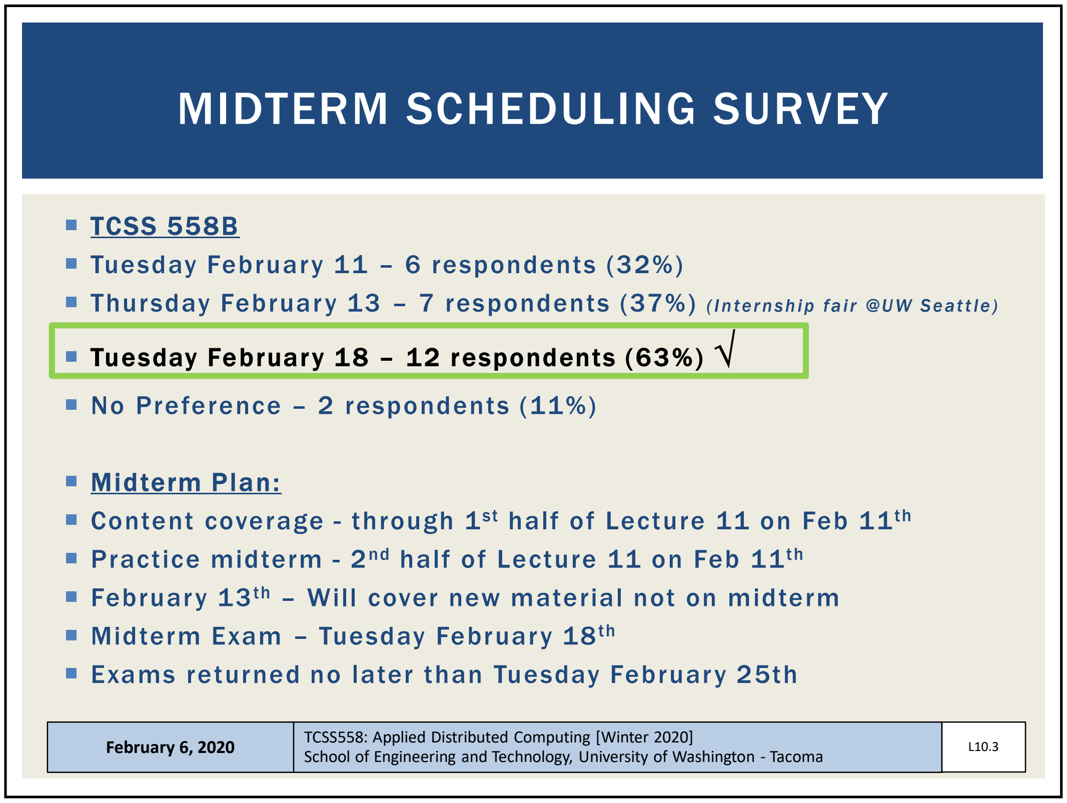


1

OBJECTIVES

- Assignment 0 – questions
- Assignment 1 – questions
- Feedback from 2/4
- Chapter 3.3: Clients – cont'd
- Chapter 3.4: Servers
- Chapter 3.5: Resource Migration

2



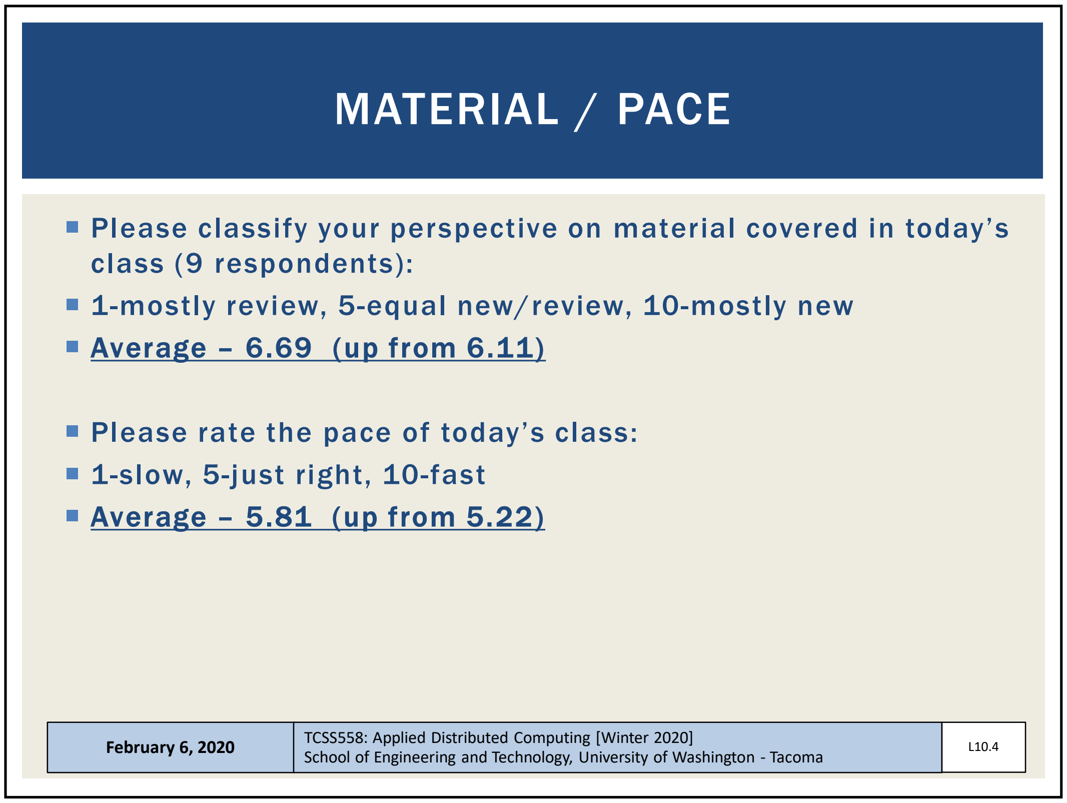
MIDTERM SCHEDULING SURVEY

- **TCSS 558B**
- Tuesday February 11 – 6 respondents (32%)
- Thursday February 13 – 7 respondents (37%) *(Internship fair @UW Seattle)*
- **Tuesday February 18 – 12 respondents (63%)** √
- No Preference – 2 respondents (11%)

- **Midterm Plan:**
- Content coverage - through 1st half of Lecture 11 on Feb 11th
- Practice midterm - 2nd half of Lecture 11 on Feb 11th
- February 13th – Will cover new material not on midterm
- Midterm Exam – Tuesday February 18th
- Exams returned no later than Tuesday February 25th

February 6, 2020 | TCSS558: Applied Distributed Computing [Winter 2020] School of Engineering and Technology, University of Washington - Tacoma | L10.3

3



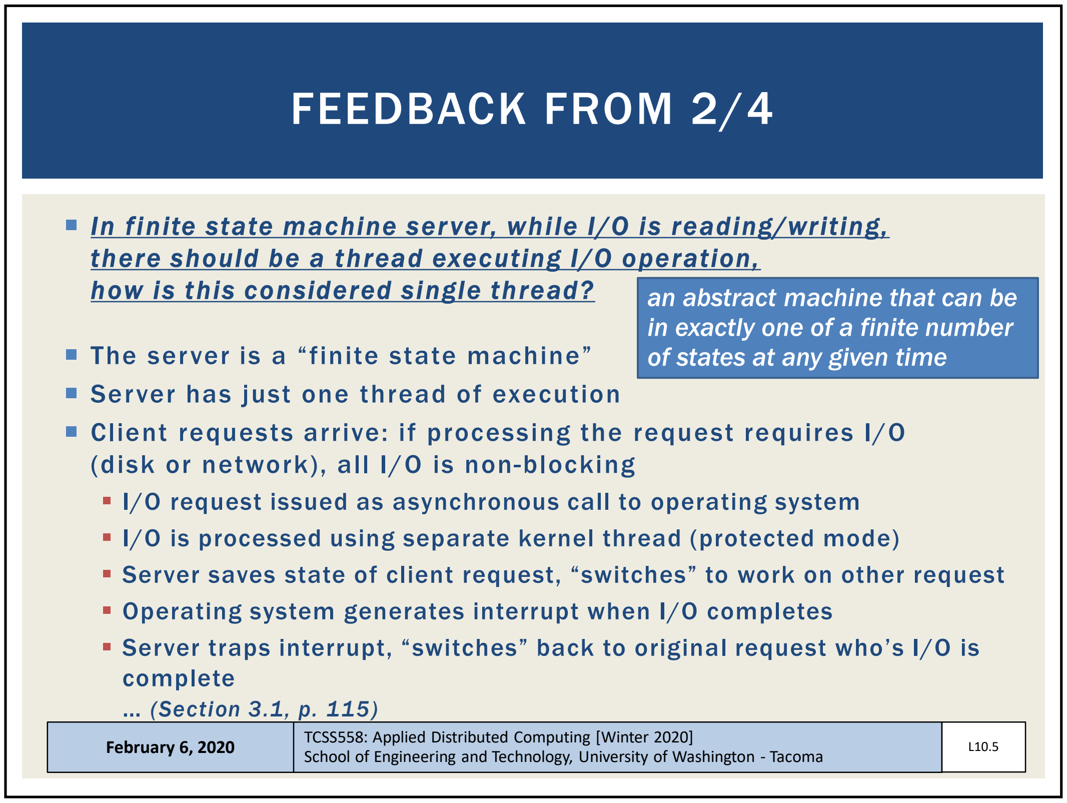
MATERIAL / PACE

- Please classify your perspective on material covered in today's class (9 respondents):
- 1-mostly review, 5-equal new/review, 10-mostly new
- **Average – 6.69 (up from 6.11)**

- Please rate the pace of today's class:
- 1-slow, 5-just right, 10-fast
- **Average – 5.81 (up from 5.22)**

February 6, 2020 | TCSS558: Applied Distributed Computing [Winter 2020] School of Engineering and Technology, University of Washington - Tacoma | L10.4

4



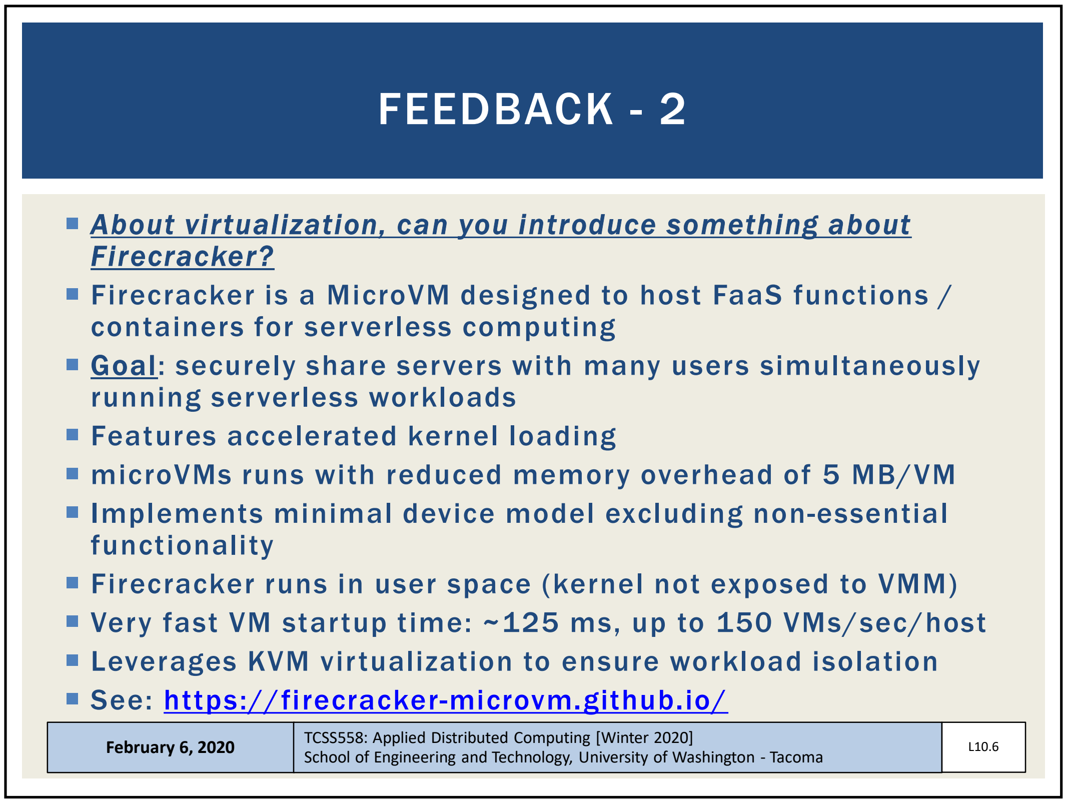
FEEDBACK FROM 2/4

- ***In finite state machine server, while I/O is reading/writing, there should be a thread executing I/O operation, how is this considered single thread?***
- The server is a "finite state machine"
- Server has just one thread of execution
- Client requests arrive: if processing the request requires I/O (disk or network), all I/O is non-blocking
  - I/O request issued as asynchronous call to operating system
  - I/O is processed using separate kernel thread (protected mode)
  - Server saves state of client request, "switches" to work on other request
  - Operating system generates interrupt when I/O completes
  - Server traps interrupt, "switches" back to original request who's I/O is complete
    … *(Section 3.1, p. 115)*

an abstract machine that can be in exactly one of a finite number of states at any given time

February 6, 2020	TCSS558: Applied Distributed Computing [Winter 2020] School of Engineering and Technology, University of Washington - Tacoma	L10.5

5

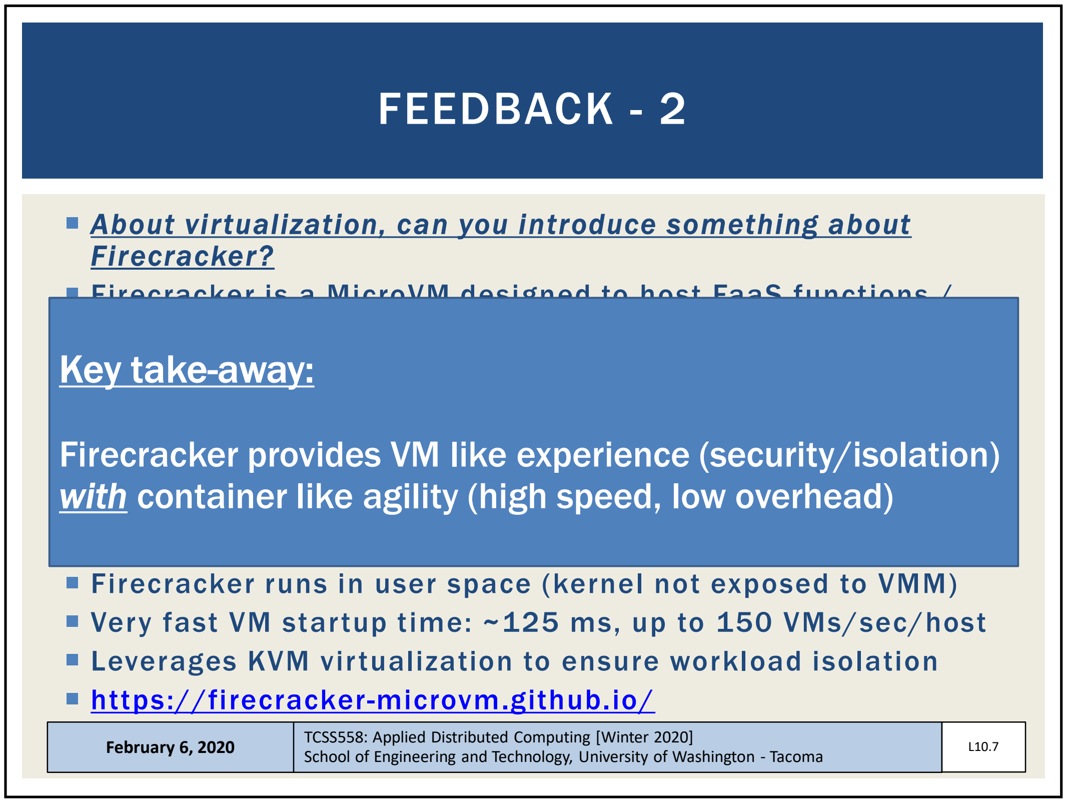


FEEDBACK - 2

- ***About virtualization, can you introduce something about Firecracker?***
- Firecracker is a MicroVM designed to host FaaS functions / containers for serverless computing
- **Goal**: securely share servers with many users simultaneously running serverless workloads
- Features accelerated kernel loading
- microVMs runs with reduced memory overhead of 5 MB/VM
- Implements minimal device model excluding non-essential functionality
- Firecracker runs in user space (kernel not exposed to VMM)
- Very fast VM startup time: ~125 ms, up to 150 VMs/sec/host
- Leverages KVM virtualization to ensure workload isolation
- See: https://firecracker-microvm.github.io/

February 6, 2020	TCSS558: Applied Distributed Computing [Winter 2020] School of Engineering and Technology, University of Washington - Tacoma	L10.6

6



FEEDBACK - 2

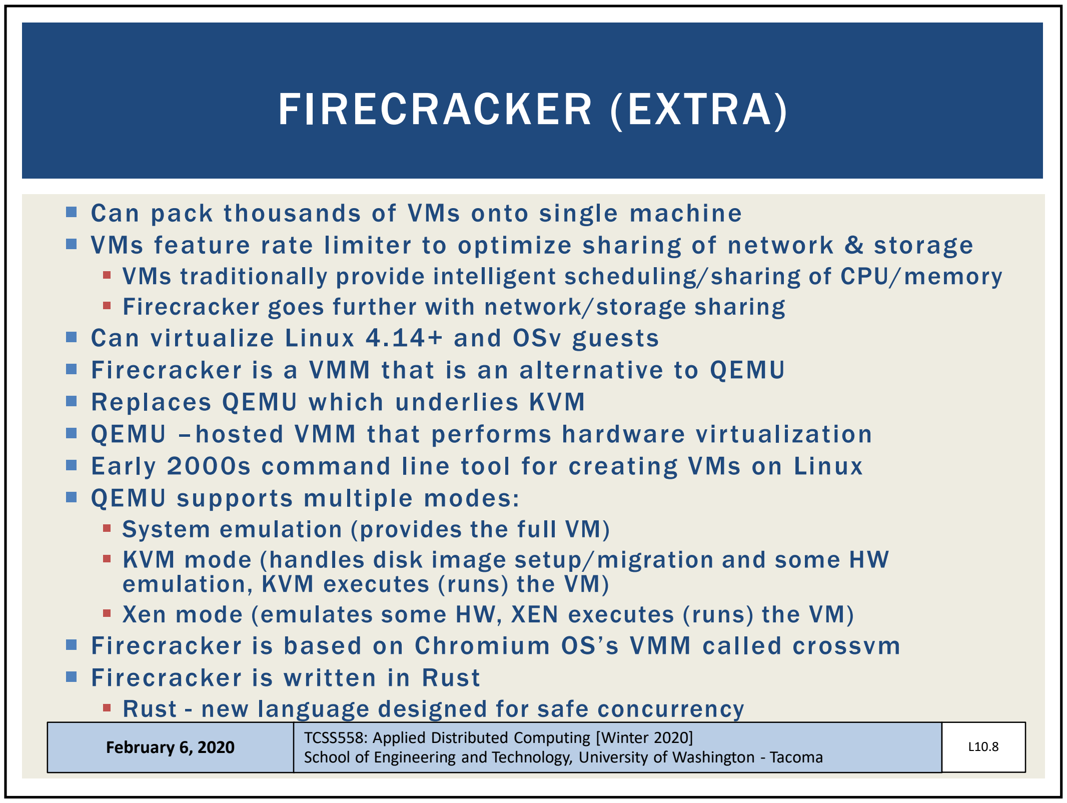
- ***About virtualization, can you introduce something about Firecracker?***
- Firecracker is a MicroVM designed to host FaaS functions /

Key take-away:

Firecracker provides VM like experience (security/isolation) ***with*** container like agility (high speed, low overhead)

- Firecracker runs in user space (kernel not exposed to VMM)
- Very fast VM startup time: ~125 ms, up to 150 VMs/sec/host
- Leverages KVM virtualization to ensure workload isolation
- https://firecracker-microvm.github.io/

7



FIRECRACKER (EXTRA)

- Can pack thousands of VMs onto single machine
- VMs feature rate limiter to optimize sharing of network & storage
  - VMs traditionally provide intelligent scheduling/sharing of CPU/memory
  - Firecracker goes further with network/storage sharing
- Can virtualize Linux 4.14+ and OSv guests
- Firecracker is a VMM that is an alternative to QEMU
- Replaces QEMU which underlies KVM
- QEMU –hosted VMM that performs hardware virtualization
- Early 2000s command line tool for creating VMs on Linux
- QEMU supports multiple modes:
  - System emulation (provides the full VM)
  - KVM mode (handles disk image setup/migration and some HW emulation, KVM executes (runs) the VM)
  - Xen mode (emulates some HW, XEN executes (runs) the VM)
- Firecracker is based on Chromium OS's VMM called crossvm
- Firecracker is written in Rust
  - Rust - new language designed for safe concurrency

8

CH. 3.3: CLIENTS

L10.9

9

TYPES OF CLIENTS

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

February 6, 2020 | TCSS558: Applied Distributed Computing [Winter 2020] School of Engineering and Technology, University of Washington - Tacoma | L10.10

10

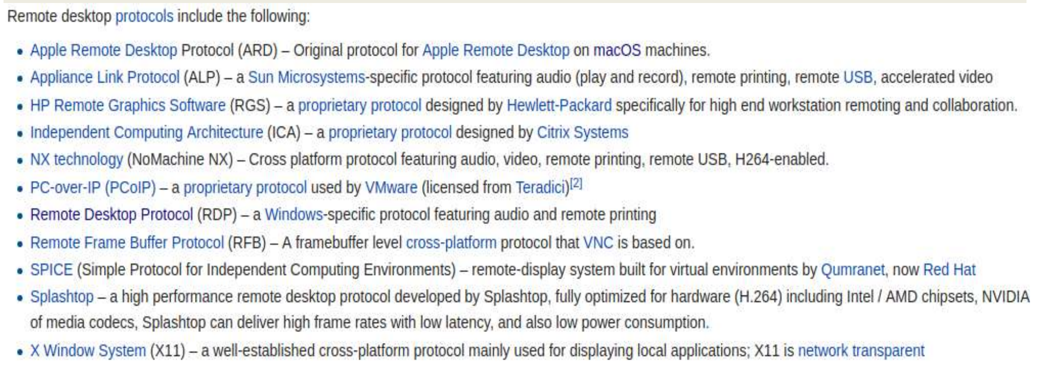
RDP VS. VNC

- VNC sends picture of desktop across network
- Minimal optimizations are employed
  - Send only parts of screen which have changed
  - Limit colors, resolution
- VNC requires more data transfer
- RDP sends instructions on how to draw screen to client
- Client renders image based on instructions and displays it
- Transferring instructions requires much less network bandwidth
- Client computer "understands" image it has created
- Client performs simple operations locally
  - Move windows without sending mouse input to host computer
  - No need to wait for host computer to render moved window
  - No need to wait for response from server
  - Client just calculates and draws results locally

11

THIN CLIENTS

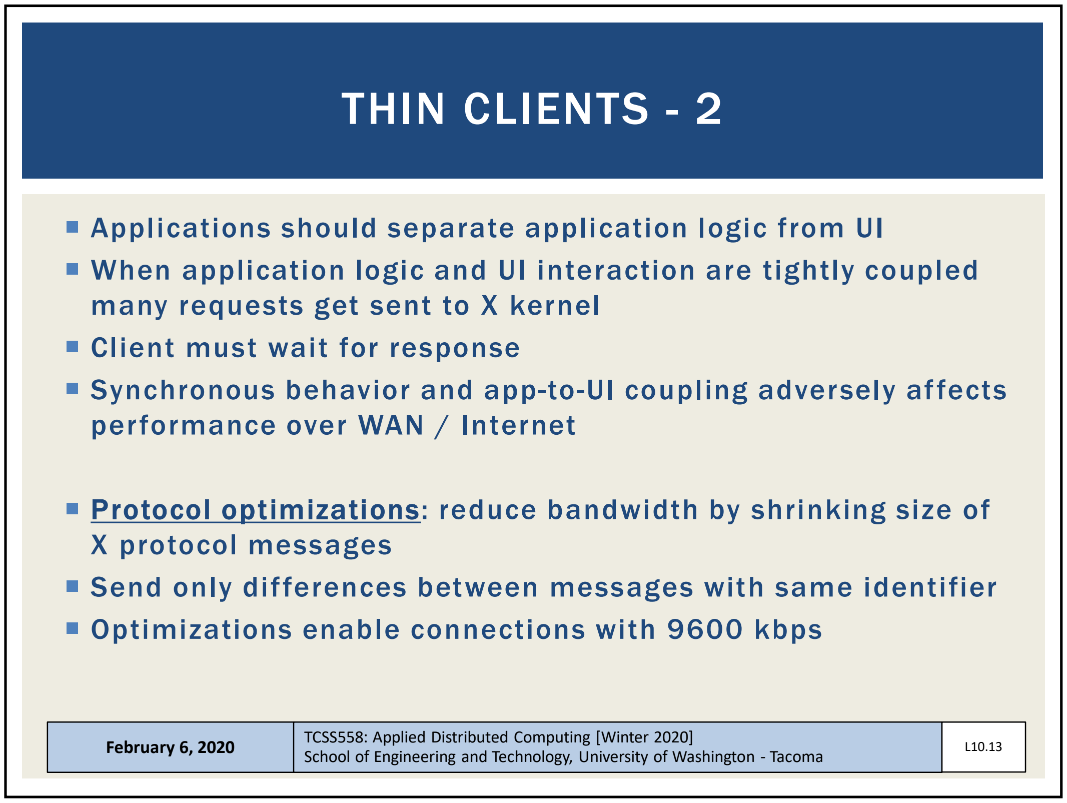
- Thin clients
  - X windows protocol
  - A variety of other remote desktop protocols exist:



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)[2]
- 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

12



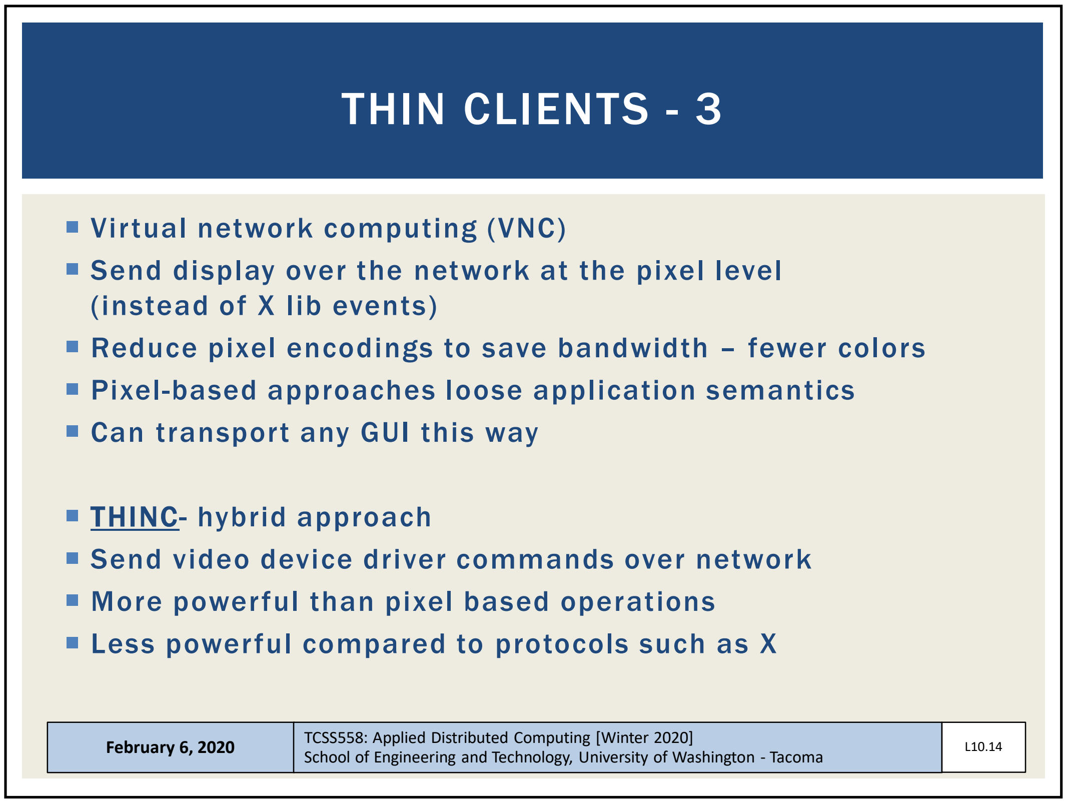
THIN CLIENTS - 2

- 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 over WAN / Internet

- <u>Protocol optimizations</u>: reduce bandwidth by shrinking size of X protocol messages
- Send only differences between messages with same identifier
- Optimizations enable connections with 9600 kbps

February 6, 2020	TCSS558: Applied Distributed Computing [Winter 2020] School of Engineering and Technology, University of Washington - Tacoma	L10.13

13



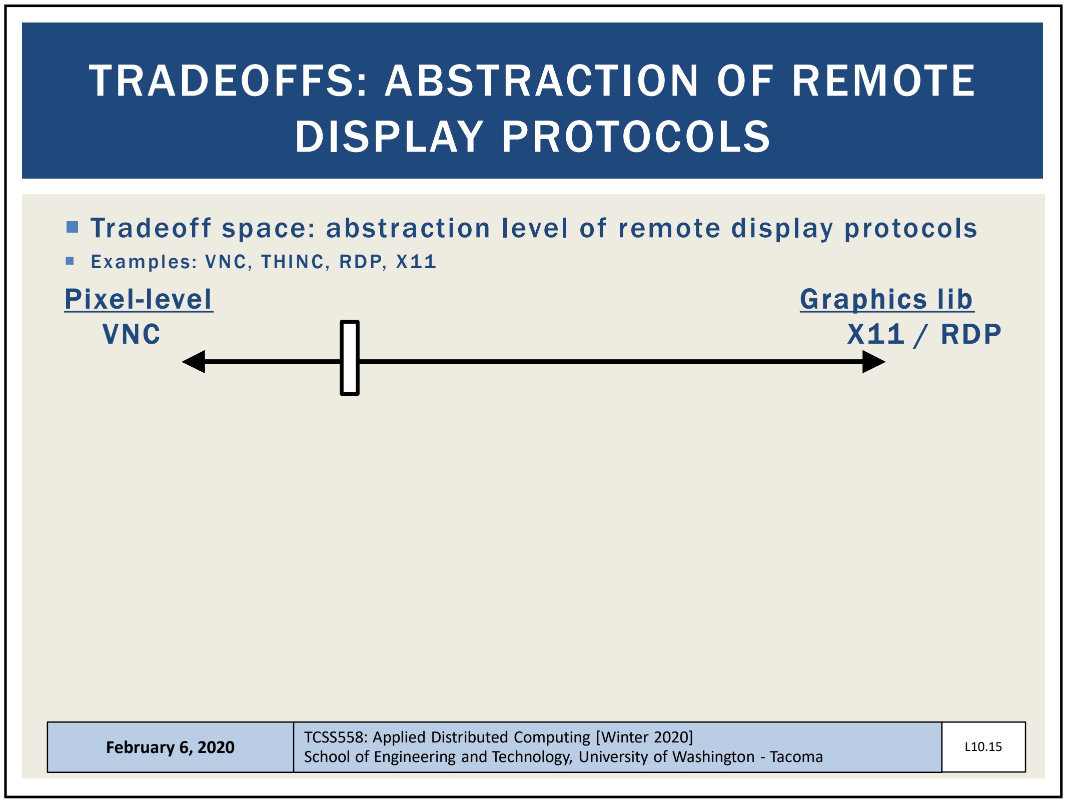
THIN CLIENTS - 3

- 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

- <u>THINC</u>- hybrid approach
- Send video device driver commands over network
- More powerful than pixel based operations
- Less powerful compared to protocols such as X

February 6, 2020	TCSS558: Applied Distributed Computing [Winter 2020] School of Engineering and Technology, University of Washington - Tacoma	L10.14

14

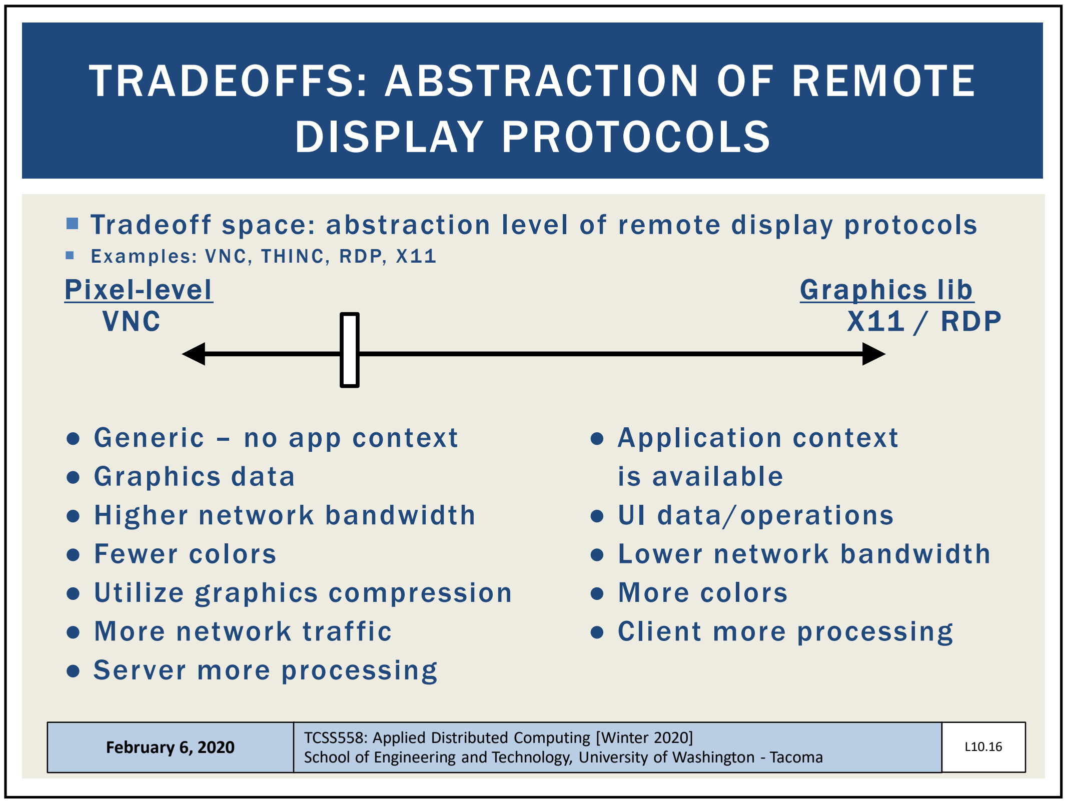


TRADEOFFS: ABSTRACTION OF REMOTE DISPLAY PROTOCOLS

- Tradeoff space: abstraction level of remote display protocols
  - Examples: VNC, THINC, RDP, X11

Pixel-level
VNC
Graphics lib
X11 / RDP

15



TRADEOFFS: ABSTRACTION OF REMOTE DISPLAY PROTOCOLS

- Tradeoff space: abstraction level of remote display protocols
  - Examples: VNC, THINC, RDP, X11

Pixel-level
VNC
Graphics lib
X11 / RDP

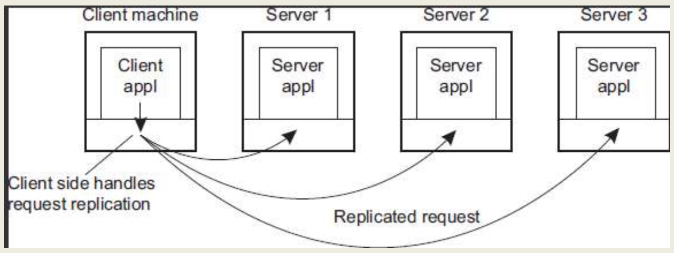
- Generic – no app context
- Graphics data
- Higher network bandwidth
- Fewer colors
- Utilize graphics compression
- More network traffic
- Server more processing

- Application context is available
- UI data/operations
- Lower network bandwidth
- More colors
- Client more processing

16

CLIENT ROLES IN PROVIDING DISTRIBUTION TRANSPARENCY

- Clients help enable distribution transparency of servers
- Replication transparency
  - Client aggregates responses from multiple servers
  - Client application (not users) know of replicas



17

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

18

CH. 3.4: SERVERS

L10.19

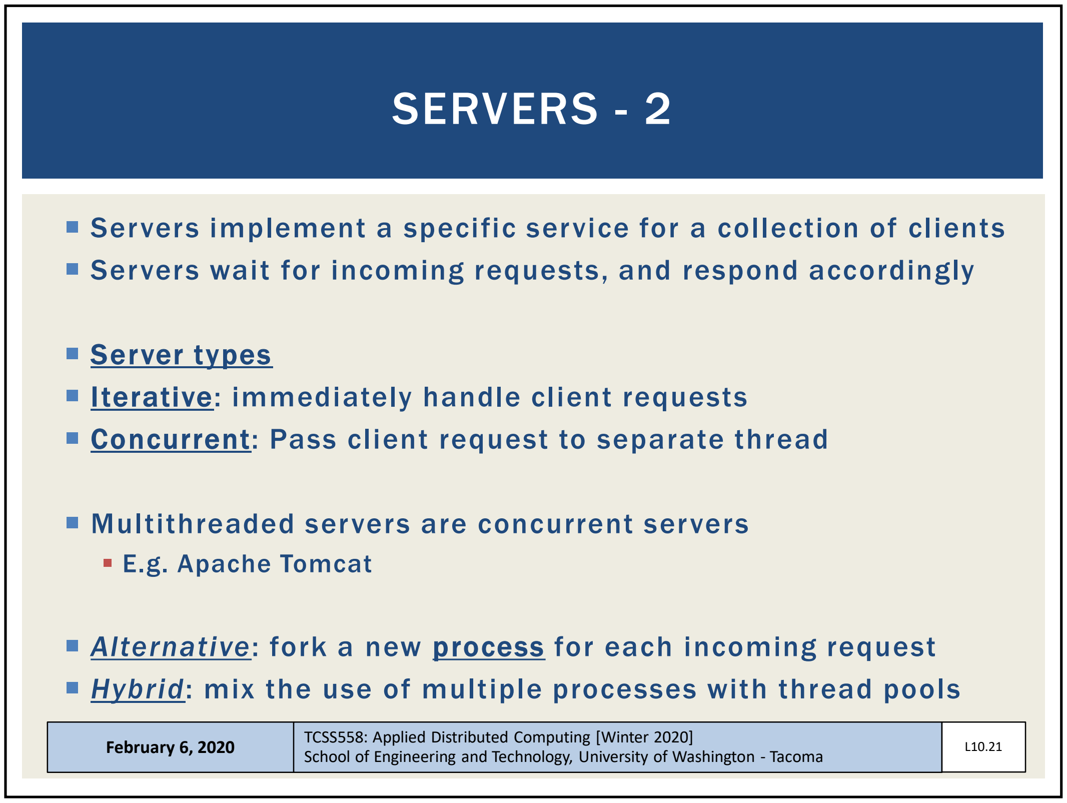
19

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 6, 2020 | TCSS558: Applied Distributed Computing [Winter 2020] School of Engineering and Technology, University of Washington - Tacoma | L10.20

20



SERVERS - 2

- Servers implement a specific service for a collection of clients
- Servers wait for incoming requests, and respond accordingly

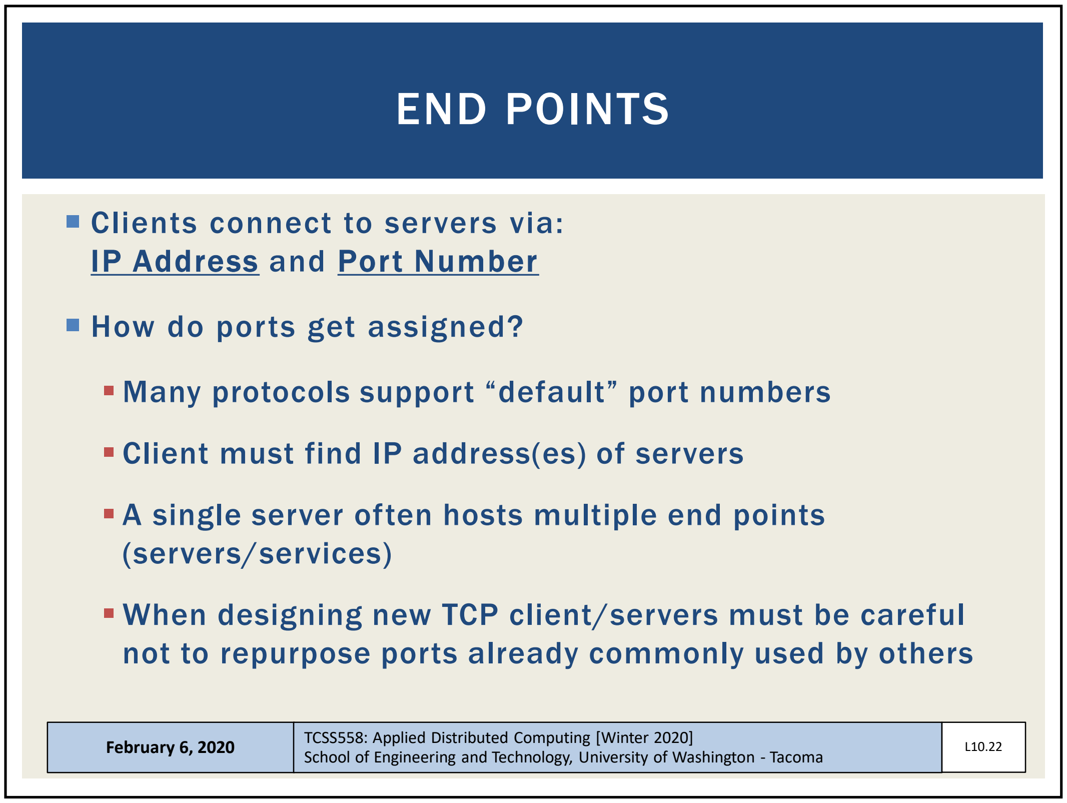
- **Server types**
- **Iterative**: immediately handle client requests
- **Concurrent**: Pass client request to separate thread

- 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 6, 2020	TCSS558: Applied Distributed Computing [Winter 2020] School of Engineering and Technology, University of Washington - Tacoma	L10.21

21



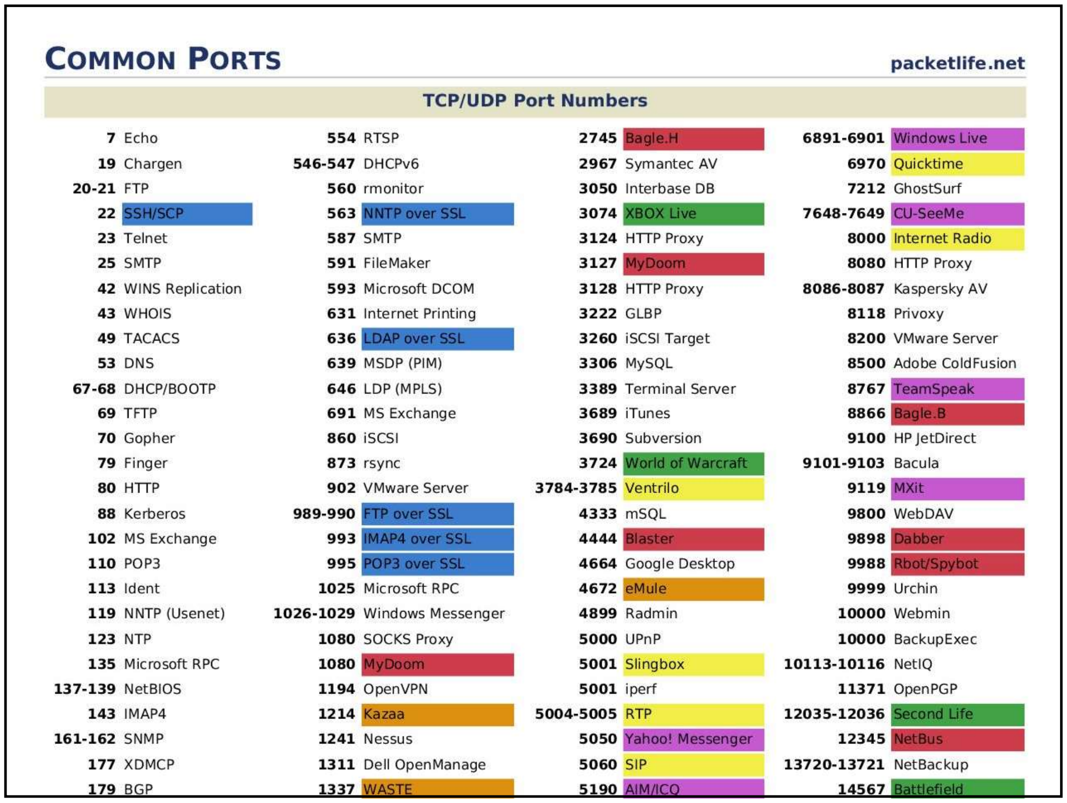
END POINTS

- 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 6, 2020	TCSS558: Applied Distributed Computing [Winter 2020] School of Engineering and Technology, University of Washington - Tacoma	L10.22

22



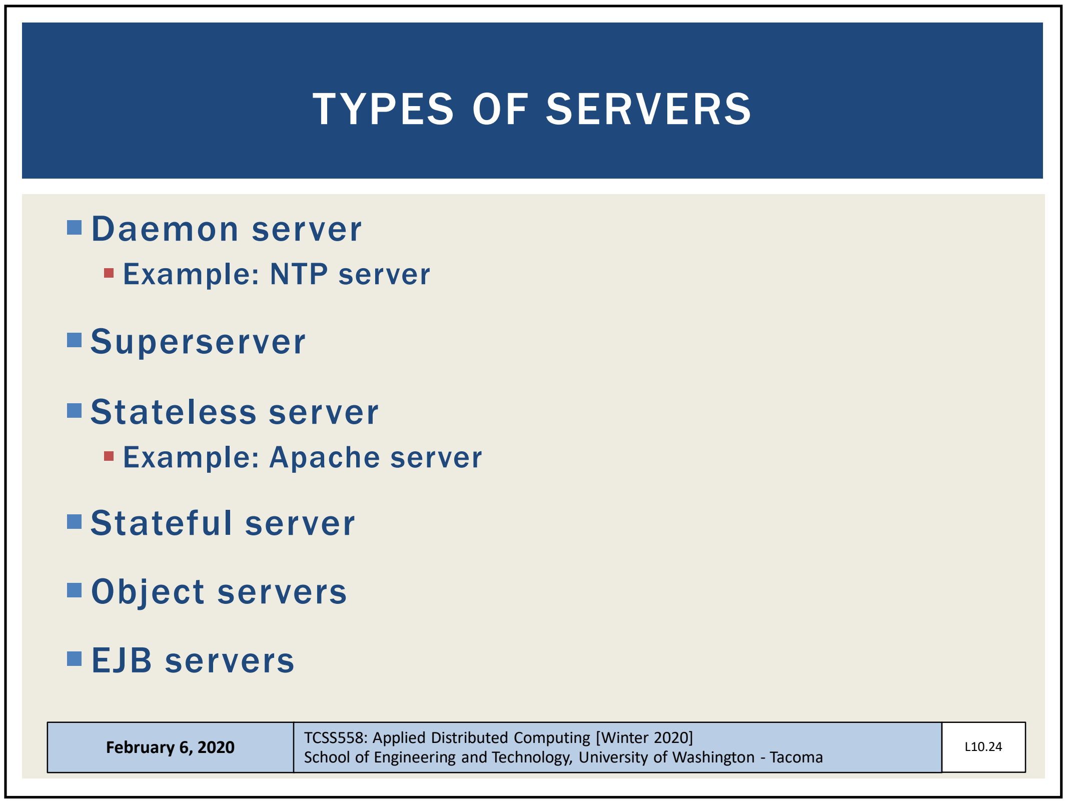
Common Ports

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

23

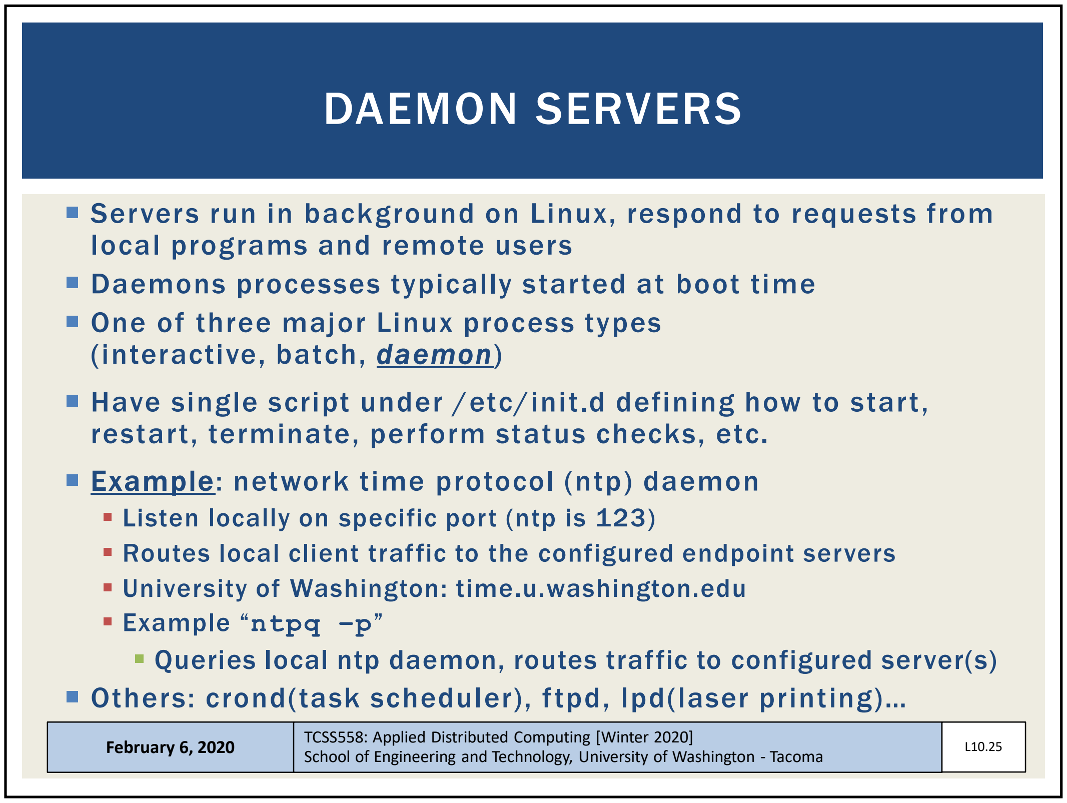


TYPES OF SERVERS

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

February 6, 2020	TCSS558: Applied Distributed Computing [Winter 2020] School of Engineering and Technology, University of Washington - Tacoma	L10.24

24

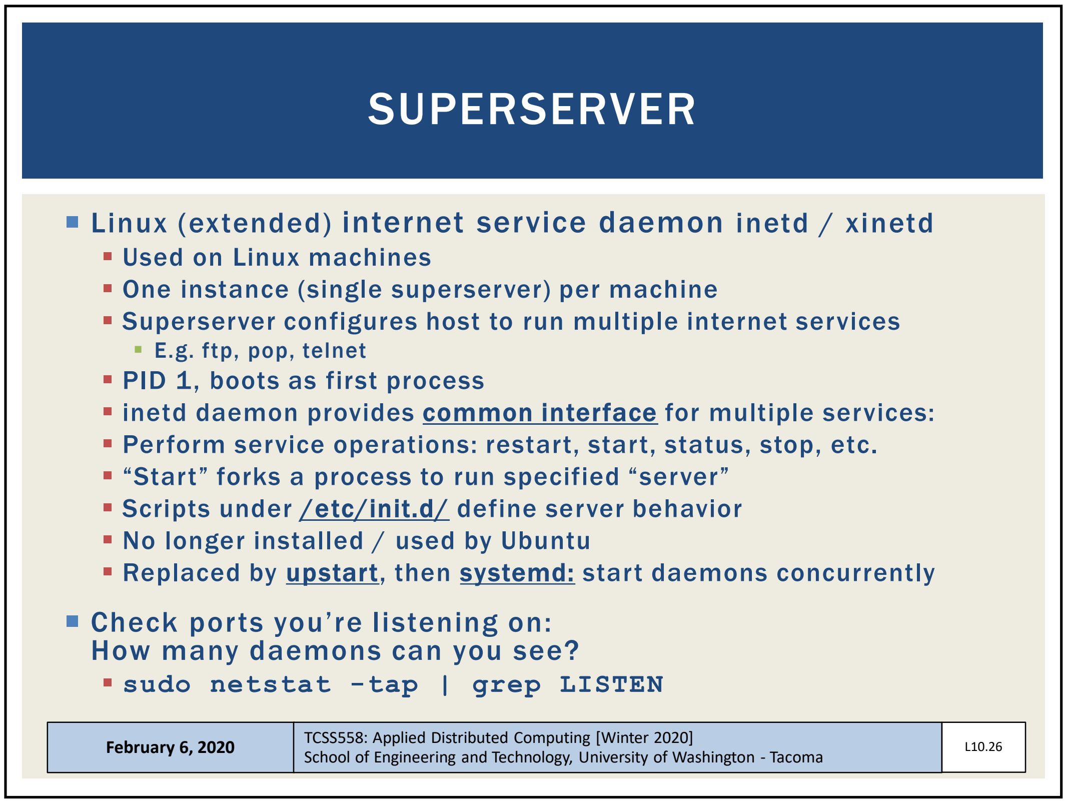


DAEMON SERVERS

- Servers run in background on Linux, respond to requests from local programs and remote users
- Daemons processes typically started at boot time
- One of three major Linux process types (interactive, batch, ***daemon***)
- Have single script under /etc/init.d defining how to start, restart, terminate, perform status checks, etc.
- **Example**: network time protocol (ntp) daemon
  - Listen locally on specific port (ntp is 123)
  - 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)
- Others: crond(task scheduler), ftpd, lpd(laser printing)…

February 6, 2020	TCSS558: Applied Distributed Computing [Winter 2020] School of Engineering and Technology, University of Washington - Tacoma	L10.25

25

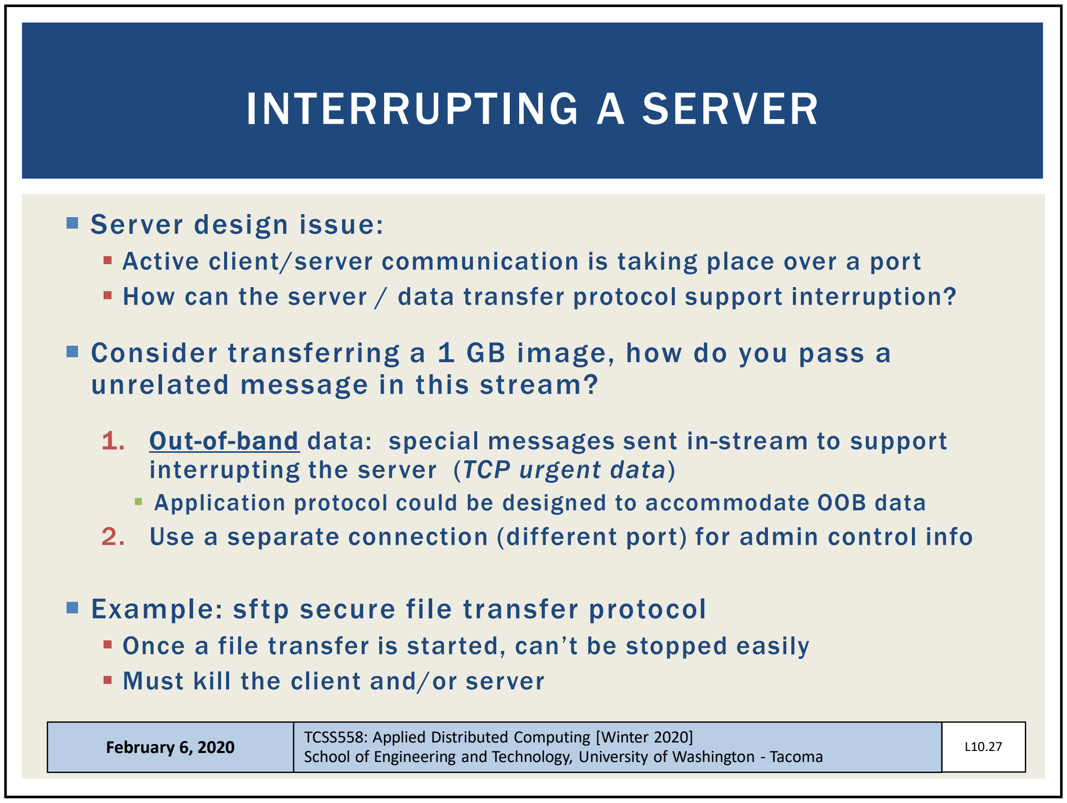


SUPERSERVER

- Linux (extended) internet service daemon inetd / xinetd
  - Used on Linux machines
  - One instance (single superserver) per machine
  - Superserver configures host to run multiple internet services
    - E.g. ftp, pop, telnet
  - PID 1, boots as first process
  - inetd daemon provides **common interface** for multiple services:
  - Perform service operations: restart, start, status, stop, etc.
  - "Start" forks a process to run specified "server"
  - Scripts under **/etc/init.d/** define server behavior
  - No longer installed / used by Ubuntu
  - Replaced by **upstart**, then **systemd:** start daemons concurrently
- Check ports you're listening on:
  How many daemons can you see?
  - `sudo netstat -tap | grep LISTEN`

February 6, 2020	TCSS558: Applied Distributed Computing [Winter 2020] School of Engineering and Technology, University of Washington - Tacoma	L10.26

26

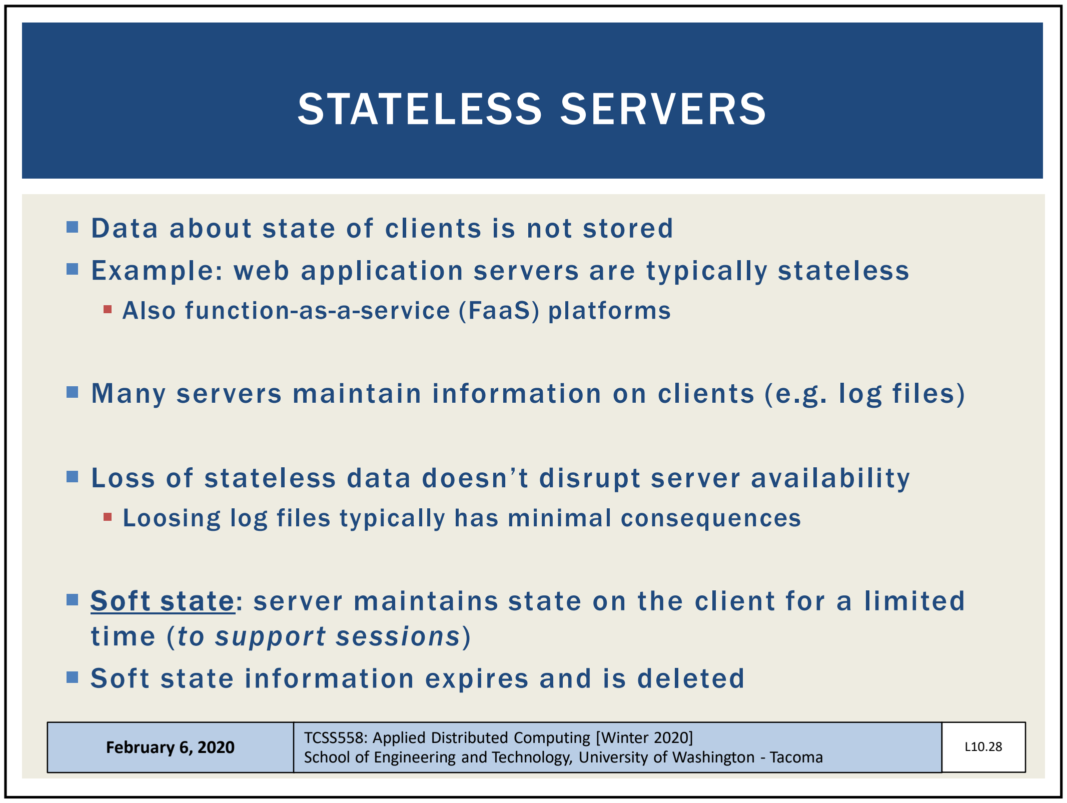


INTERRUPTING A SERVER

- 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*)
     - Application protocol could be designed to accommodate OOB 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 6, 2020	TCSS558: Applied Distributed Computing [Winter 2020] School of Engineering and Technology, University of Washington - Tacoma	L10.27

27

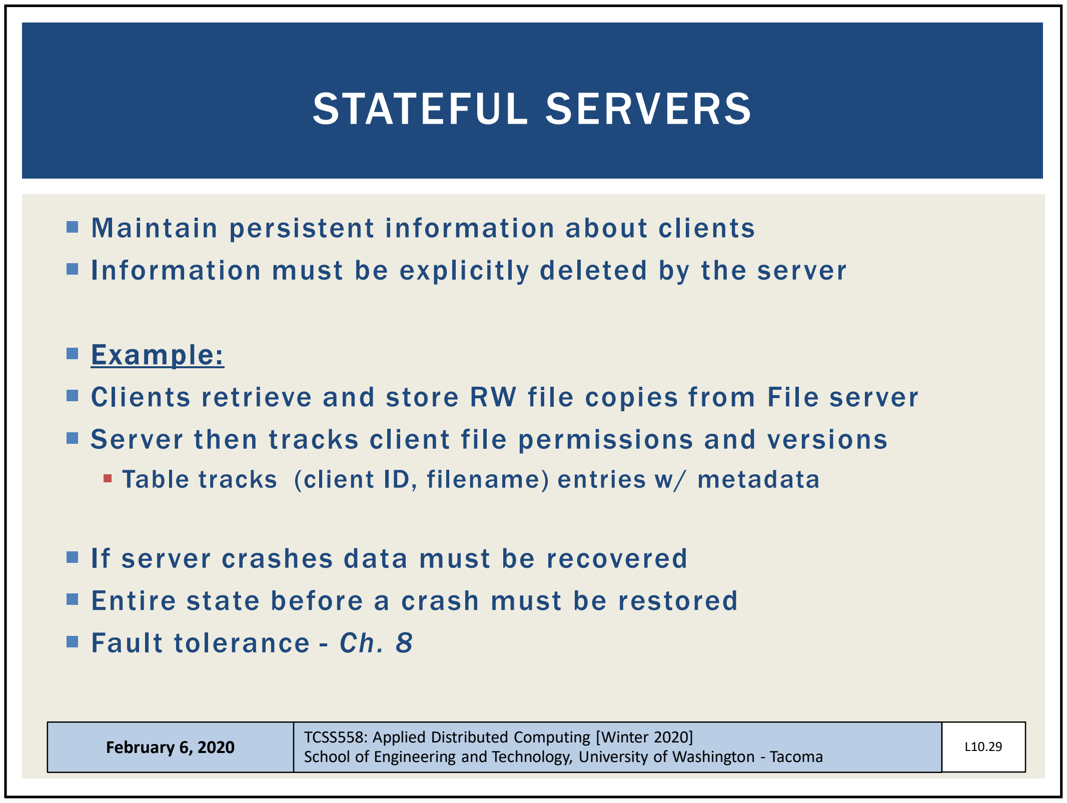


STATELESS SERVERS

- 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 6, 2020	TCSS558: Applied Distributed Computing [Winter 2020] School of Engineering and Technology, University of Washington - Tacoma	L10.28

28



STATEFUL SERVERS

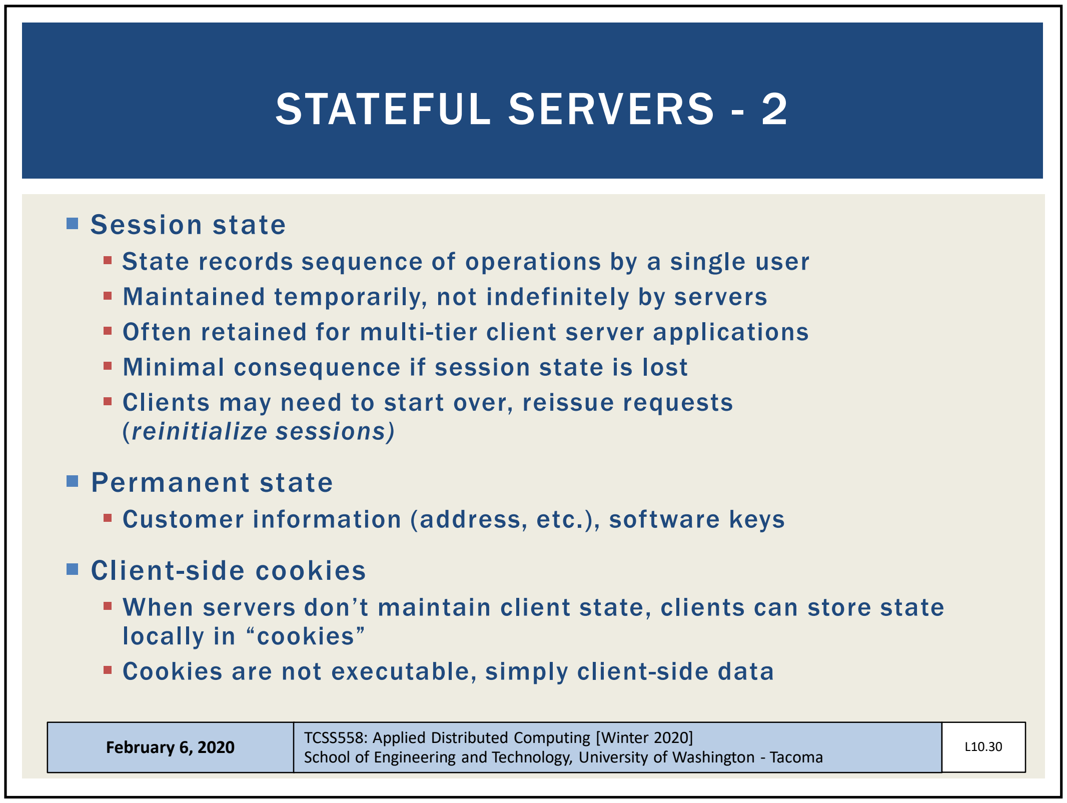
- Maintain persistent information about clients
- Information must be explicitly deleted by the server

- <u>Example:</u>
- Clients retrieve and store RW file copies from File server
- Server then tracks client file permissions and versions
  - Table tracks (client ID, filename) entries w/ metadata

- If server crashes data must be recovered
- Entire state before a crash must be restored
- Fault tolerance - *Ch. 8*

February 6, 2020	TCSS558: Applied Distributed Computing [Winter 2020] School of Engineering and Technology, University of Washington - Tacoma	L10.29

29

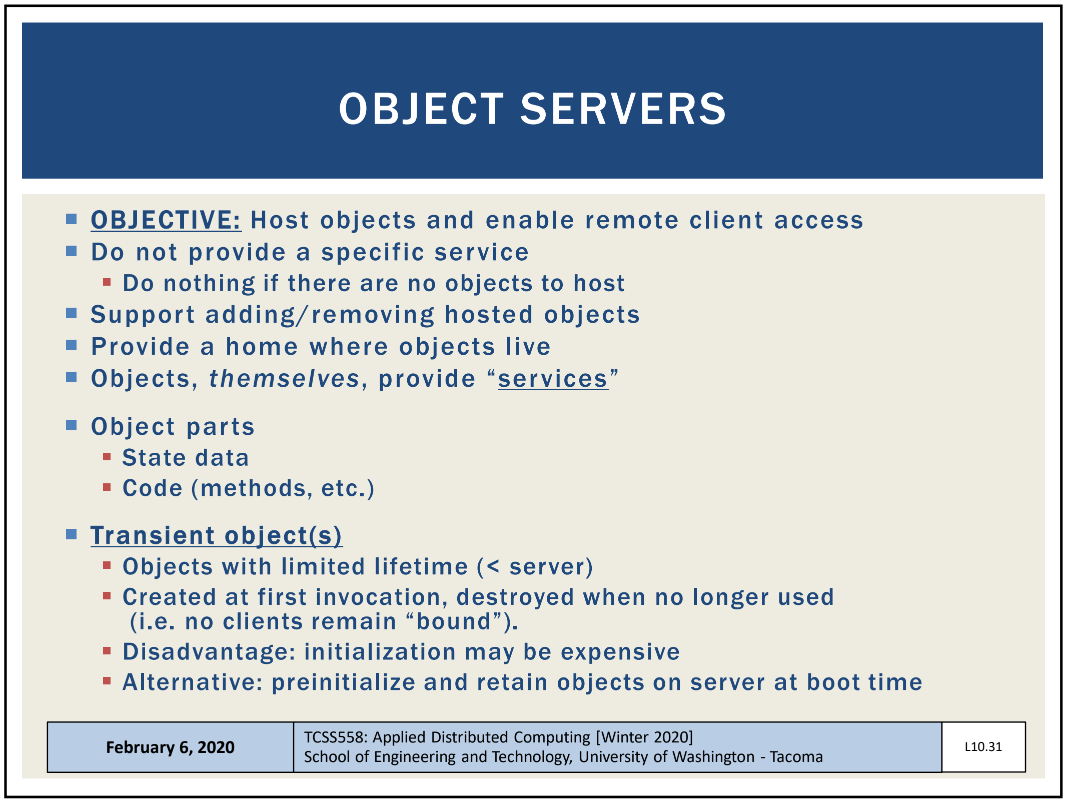


STATEFUL SERVERS - 2

- Session state
  - State records sequence of operations by a single user
  - Maintained temporarily, not indefinitely by servers
  - Often retained for multi-tier client server applications
  - Minimal consequence if session state is lost
  - Clients may need to start over, reissue requests (*reinitialize sessions)*
- Permanent state
  - Customer information (address, etc.), 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 6, 2020	TCSS558: Applied Distributed Computing [Winter 2020] School of Engineering and Technology, University of Washington - Tacoma	L10.30

30

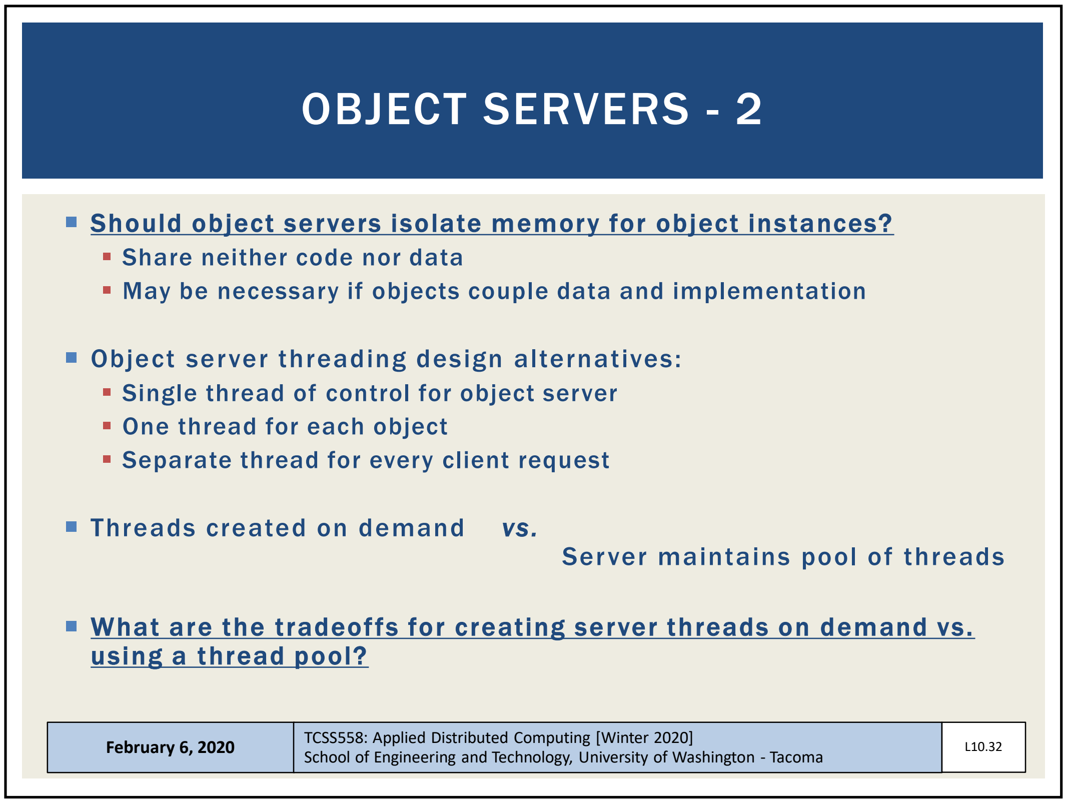


OBJECT SERVERS

- **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 at boot time

February 6, 2020 | TCSS558: Applied Distributed Computing [Winter 2020] School of Engineering and Technology, University of Washington - Tacoma | L10.31

31

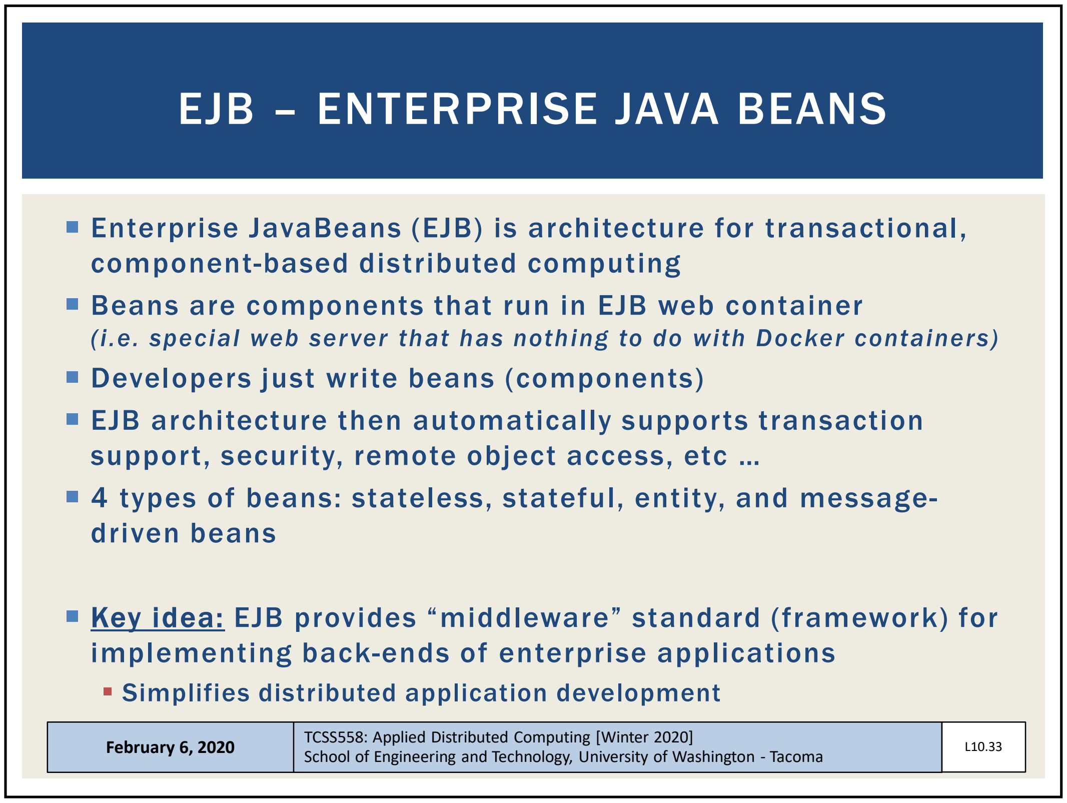


OBJECT SERVERS - 2

- **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 design alternatives:
  - Single thread of control for object server
  - One thread for each object
  - Separate thread for every client request
- 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 6, 2020 | TCSS558: Applied Distributed Computing [Winter 2020] School of Engineering and Technology, University of Washington - Tacoma | L10.32

32



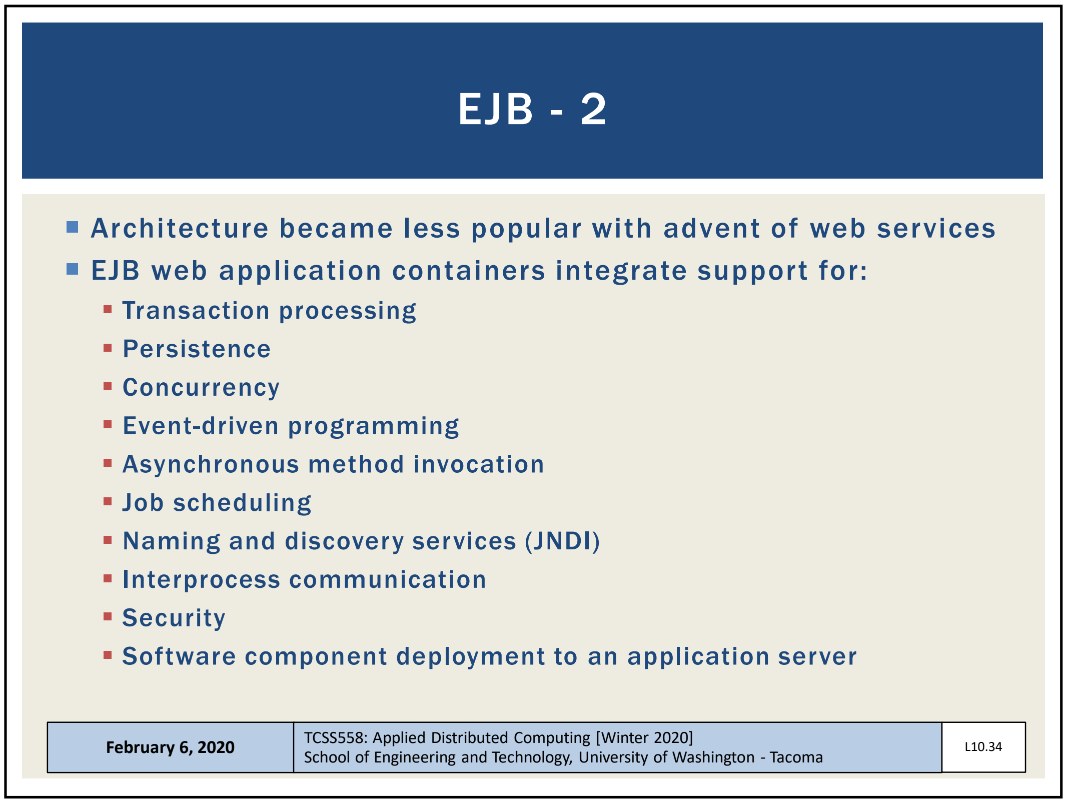
EJB – ENTERPRISE JAVA BEANS

- Enterprise JavaBeans (EJB) is architecture for transactional, component-based distributed computing
- Beans are components that run in EJB web container *(i.e. special web server that has nothing to do with Docker containers)*
- Developers just write beans (components)
- EJB architecture then automatically supports transaction support, security, remote object access, etc …
- 4 types of beans: stateless, stateful, entity, and message-driven beans

- Key idea: EJB provides "middleware" standard (framework) for implementing back-ends of enterprise applications
  - Simplifies distributed application development

February 6, 2020	TCSS558: Applied Distributed Computing [Winter 2020] School of Engineering and Technology, University of Washington - Tacoma	L10.33

33



EJB - 2

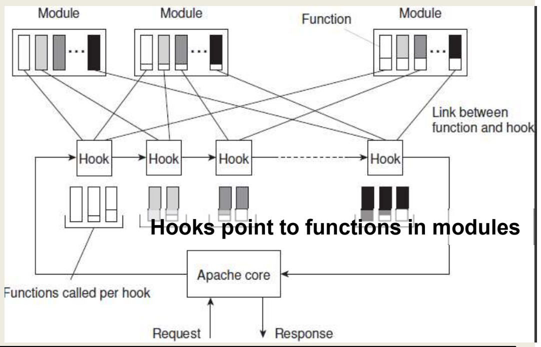
- Architecture became less popular with advent of web services
- EJB web application containers integrate support for:
  - Transaction processing
  - Persistence
  - Concurrency
  - Event-driven programming
  - Asynchronous method invocation
  - Job scheduling
  - Naming and discovery services (JNDI)
  - Interprocess communication
  - Security
  - Software component deployment to an application server

February 6, 2020	TCSS558: Applied Distributed Computing [Winter 2020] School of Engineering and Technology, University of Washington - Tacoma	L10.34

34

APACHE WEB SERVER

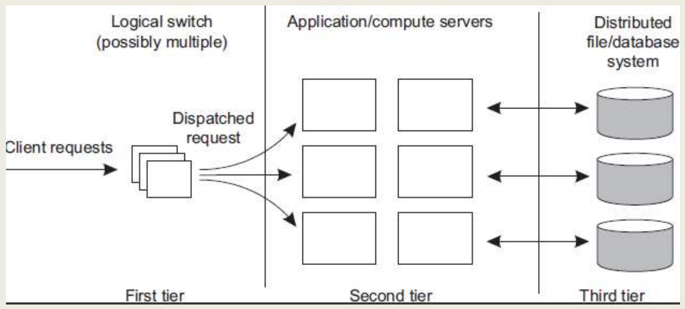
- Highly configurable, extensible, platform independent
- Supports TCP HTTP protocol communication
- Uses hooks – placeholders for group of functions
- Requests processed in phases by hooks
- Many hooks:
  - Translate a URL
  - Write info to log
  - Check client ID
  - Check access rights
- Hooks processed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


Module
Module
Function
Module
Link between function and hook
Hook
Hook
Hook
Hook
Hooks point to functions in modules
Apache core
Functions called per hook
Request
Response

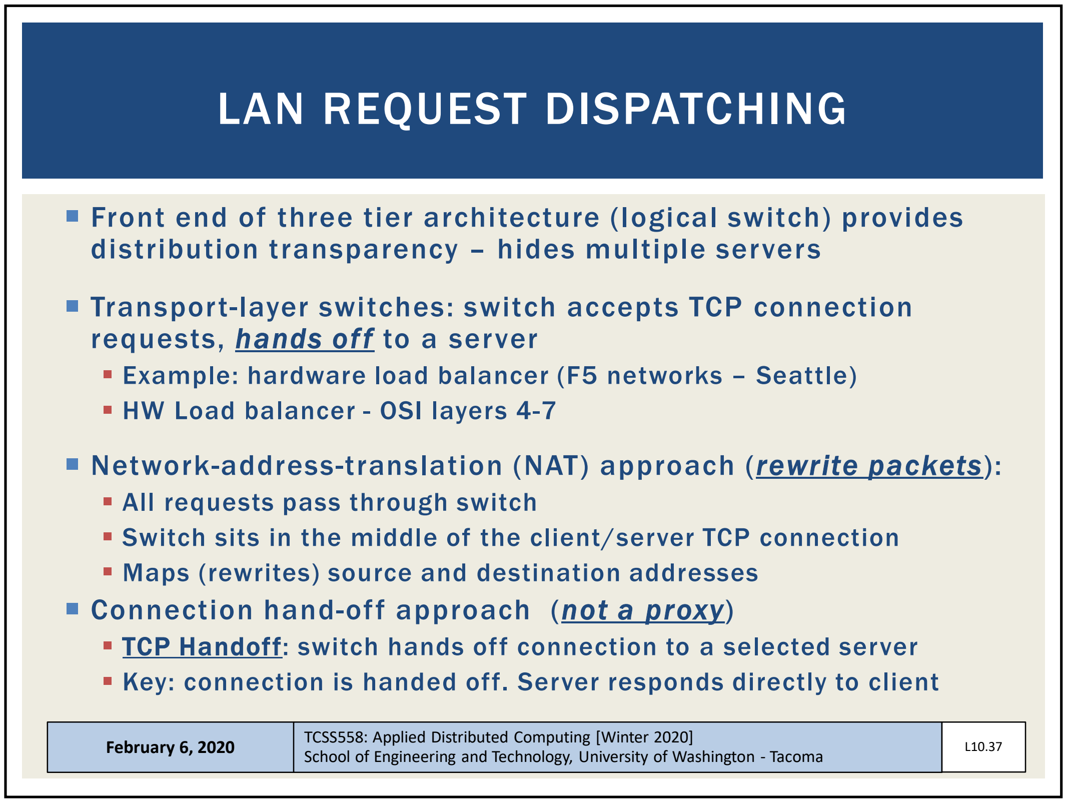
35

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


Logical switch (possibly multiple)
Application/compute servers
Distributed file/database system
Dispatched request
Client requests
First tier
Second tier
Third tier

36

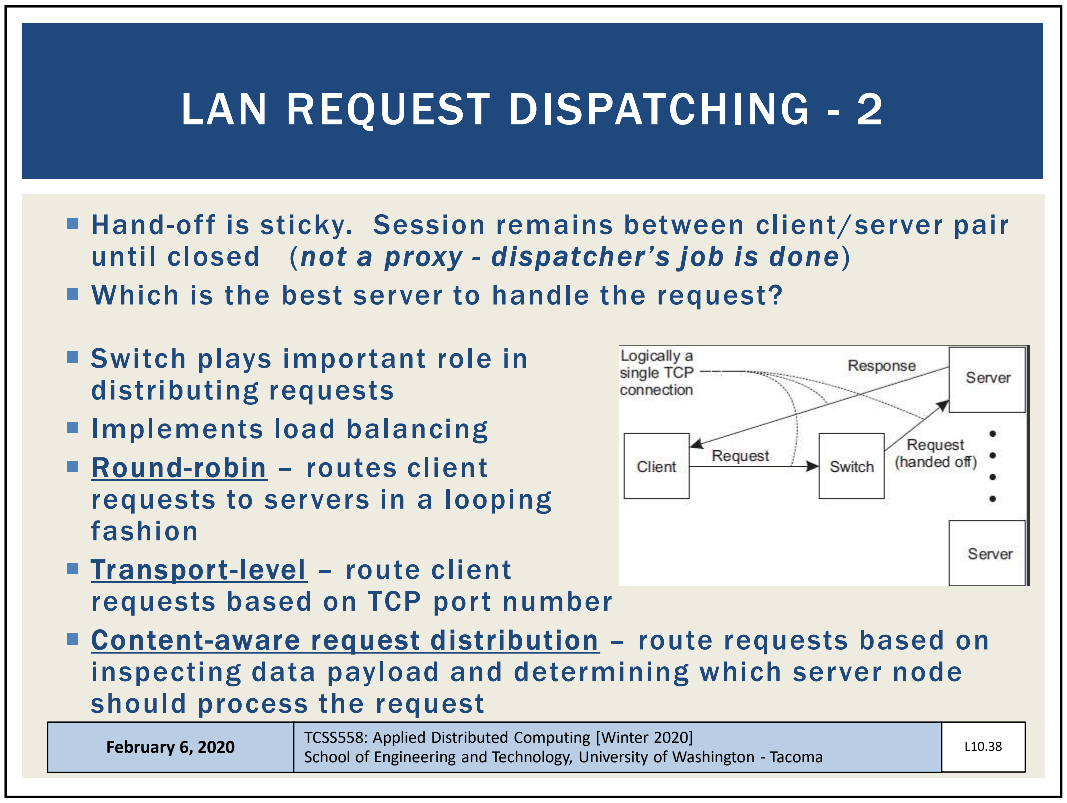


LAN REQUEST DISPATCHING

- 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 (***rewrite packets***):
  - 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 (***not a proxy***)
  - **TCP Handoff**: switch hands off connection to a selected server
  - Key: connection is handed off. Server responds directly to client

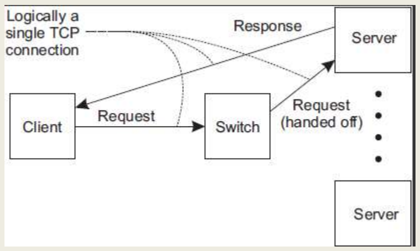
February 6, 2020 | TCSS558: Applied Distributed Computing [Winter 2020] School of Engineering and Technology, University of Washington - Tacoma | L10.37

37



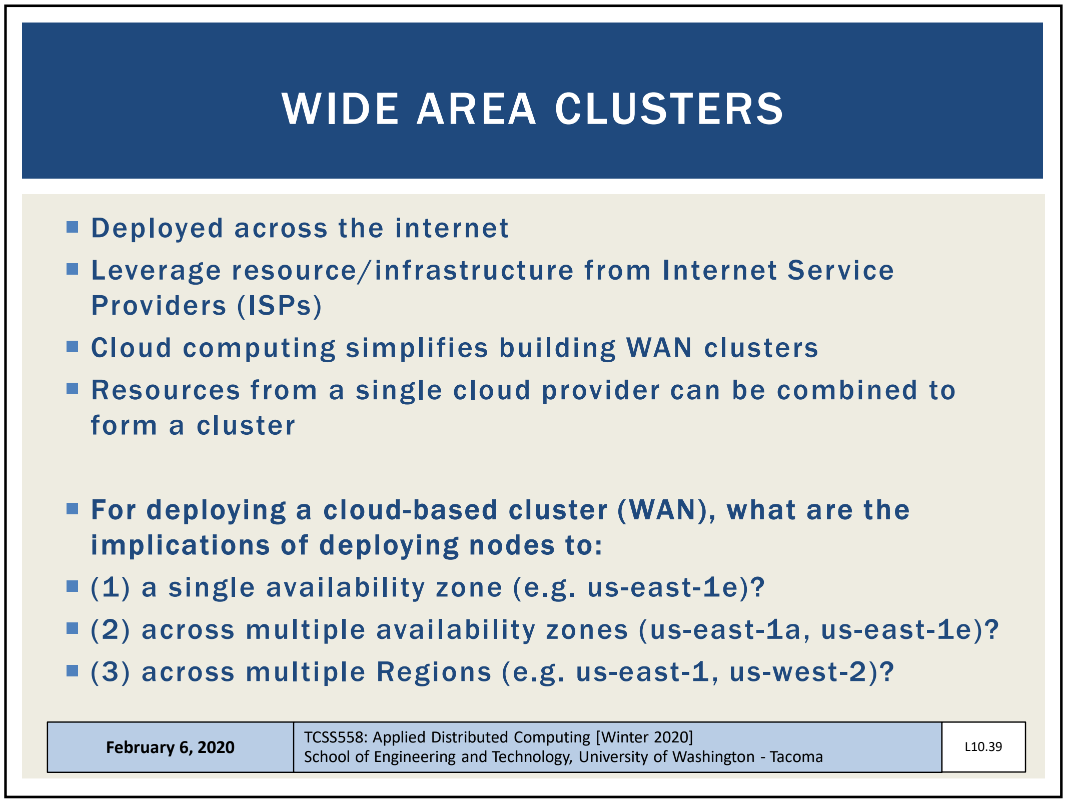
LAN REQUEST DISPATCHING - 2

- Hand-off is sticky. Session remains between client/server pair until closed (***not a proxy - dispatcher's job is done***)
- Which 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 6, 2020 | TCSS558: Applied Distributed Computing [Winter 2020] School of Engineering and Technology, University of Washington - Tacoma | L10.38

38

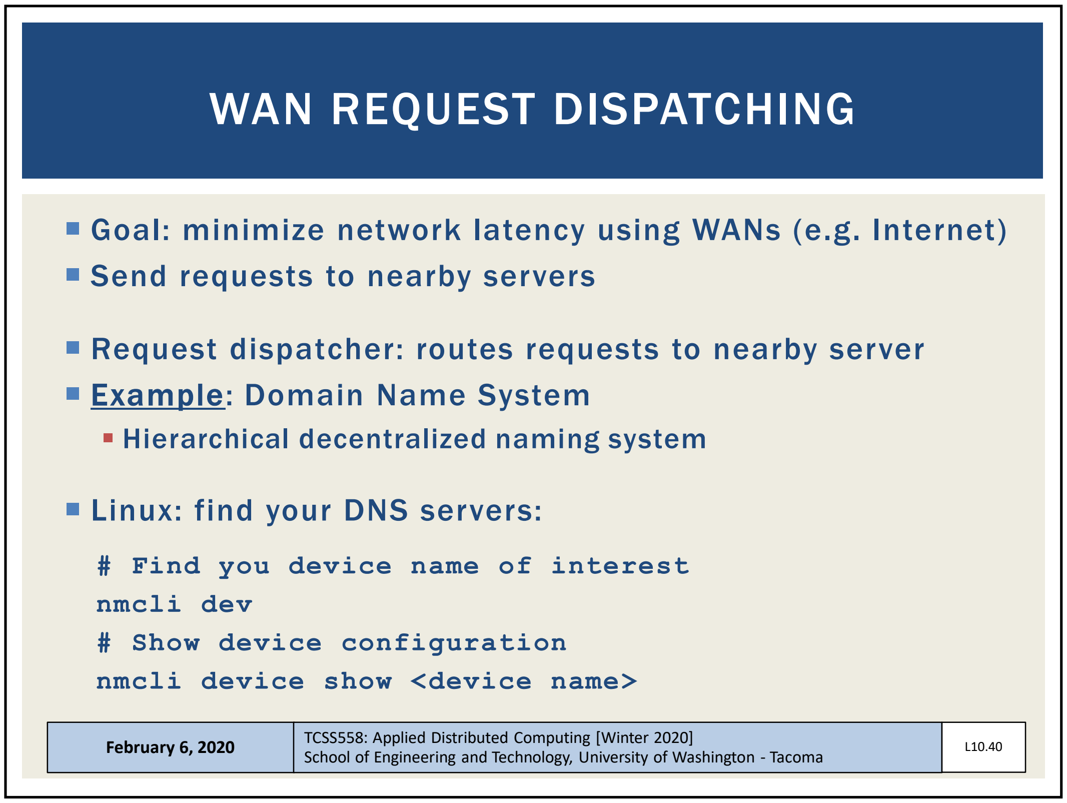


WIDE AREA CLUSTERS

- Deployed across the internet
- Leverage resource/infrastructure from Internet Service Providers (ISPs)
- Cloud computing simplifies building WAN clusters
- Resources 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 (us-east-1a, us-east-1e)?
- (3) across multiple Regions (e.g. us-east-1, us-west-2)?

39



WAN REQUEST DISPATCHING

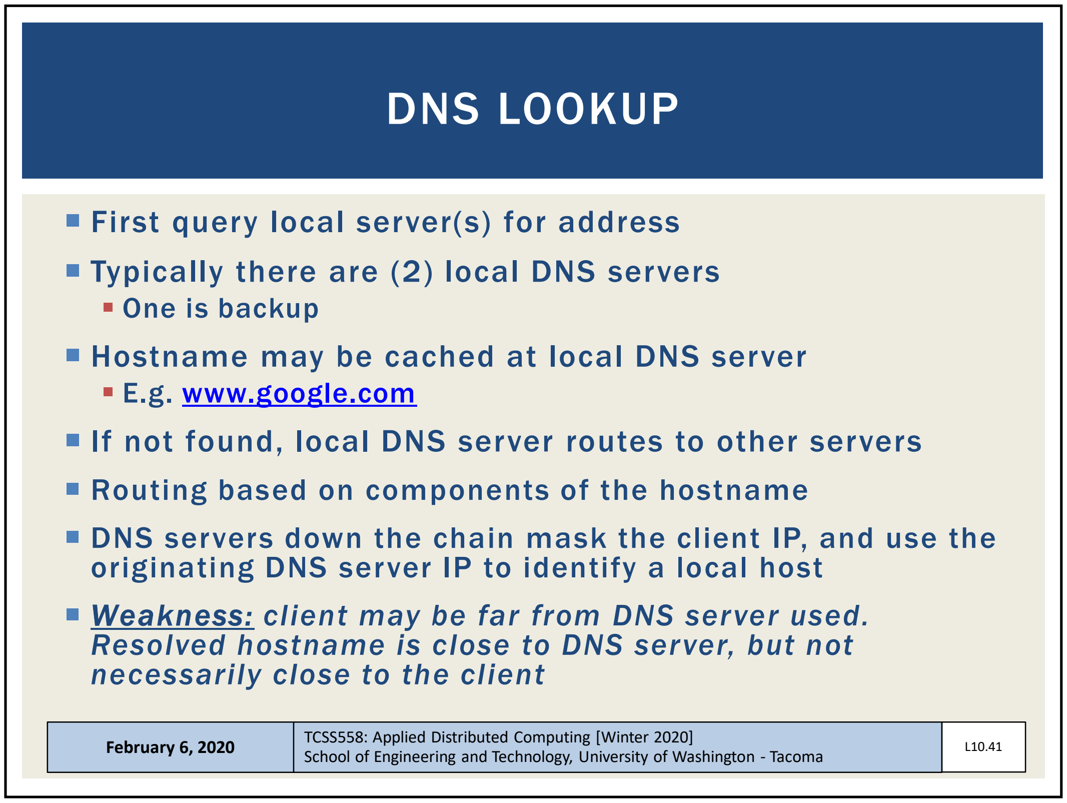
- Goal: minimize network latency using WANs (e.g. Internet)
- Send requests to nearby servers

- 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>
```

40

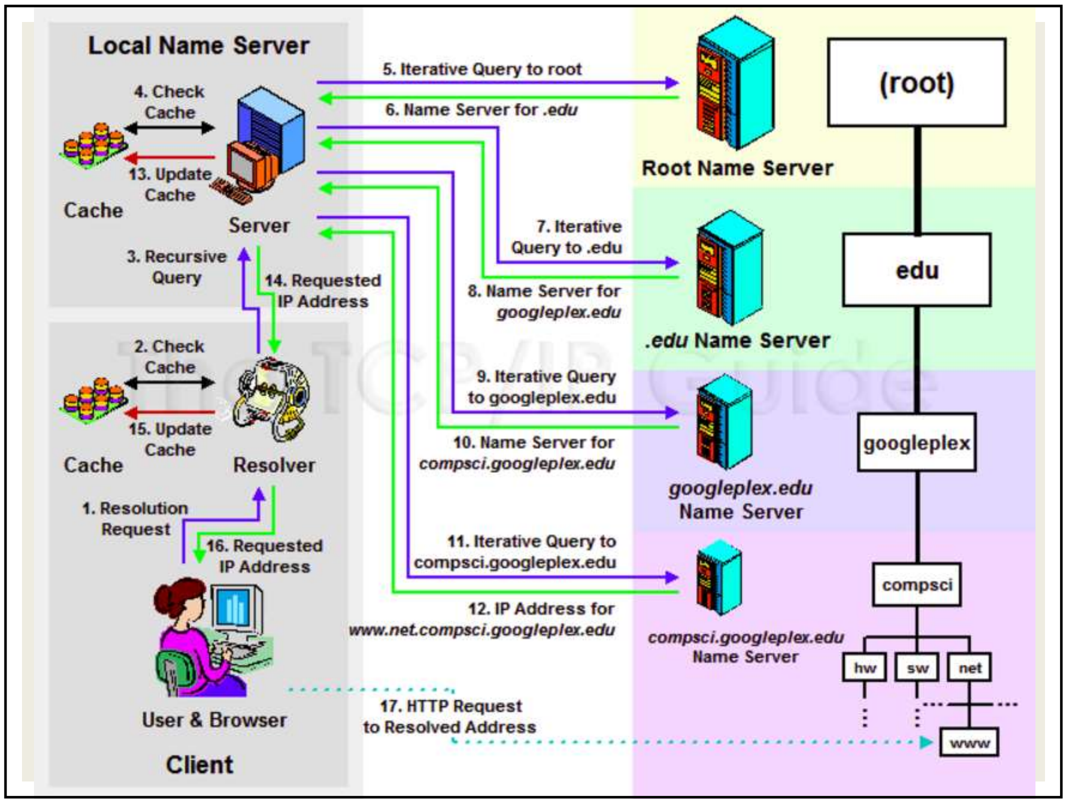


DNS LOOKUP

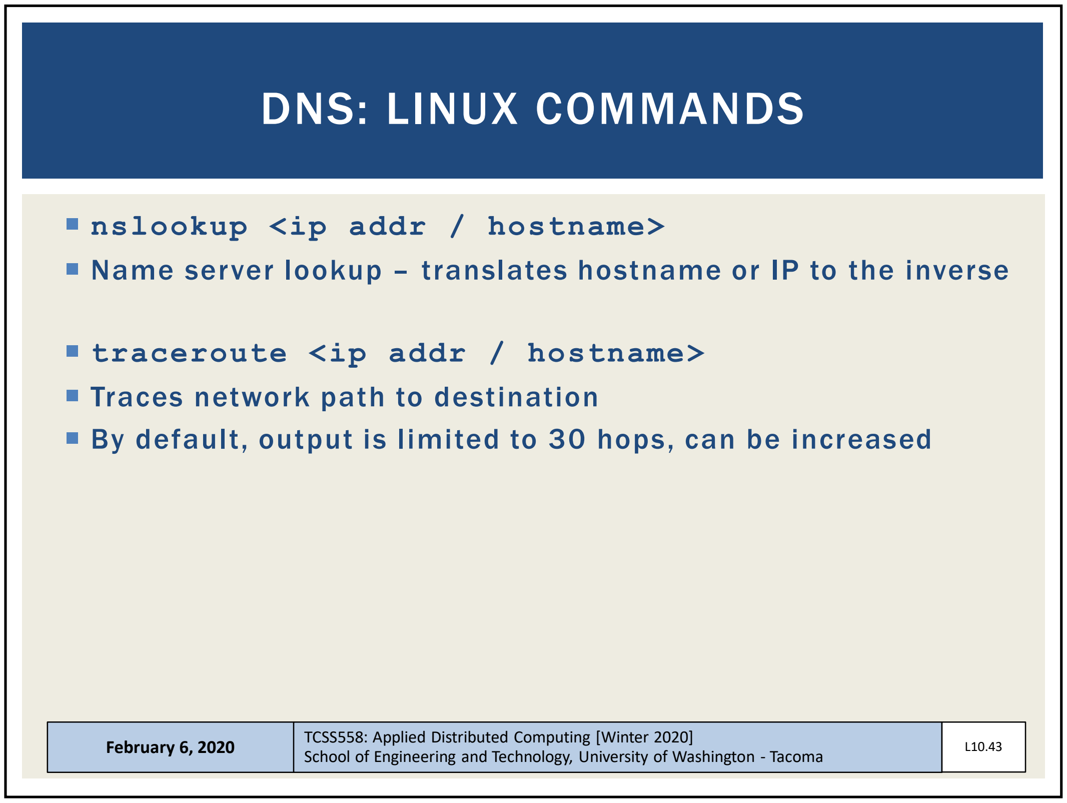
- First query local server(s) for address
- Typically there are (2) local DNS servers
  - One is backup
- Hostname may be cached at local DNS server
  - E.g. www.google.com
- If not found, local DNS server routes to other servers
- Routing based on components of the hostname
- DNS servers down the chain 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 6, 2020	TCSS558: Applied Distributed Computing [Winter 2020] School of Engineering and Technology, University of Washington - Tacoma	L10.41

41


Local Name Server
4. Check Cache
13. Update Cache
Cache
Server
3. Recursive Query
14. Requested IP Address
2. Check Cache
15. Update Cache
Cache
Resolver
1. Resolution Request
16. Requested IP Address
User & Browser
Client
5. Iterative Query to root
6. Name Server for .edu
7. Iterative Query to .edu
8. Name Server for googleplex.edu
9. Iterative Query to googleplex.edu
10. Name Server for compsci.googleplex.edu
11. Iterative Query to compsci.googleplex.edu
12. IP Address for www.net.compsci.googleplex.edu
17. HTTP Request to Resolved Address
Root Name Server
.edu Name Server
googleplex.edu Name Server
compsci.googleplex.edu Name Server
(root)
edu
googleplex
compsci
hw
sw
net
www
The TCP/IP Guide

42



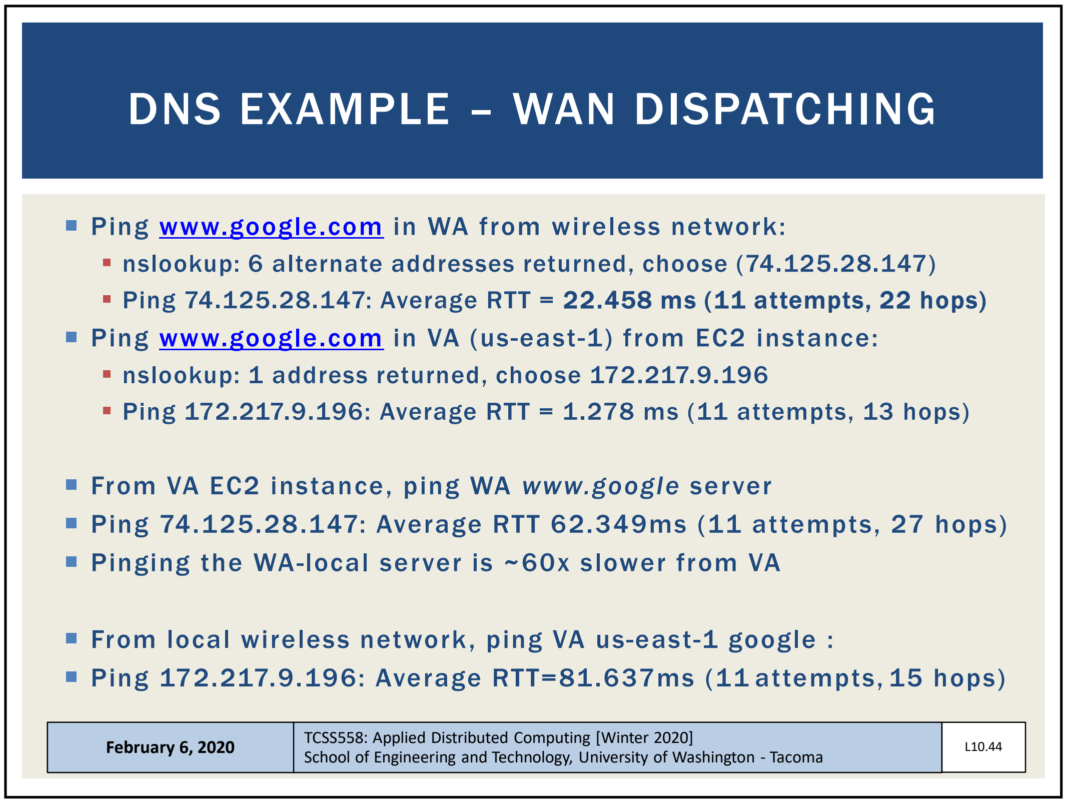
DNS: LINUX COMMANDS

- `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

February 6, 2020 | TCSS558: Applied Distributed Computing [Winter 2020]
School of Engineering and Technology, University of Washington - Tacoma | L10.43

43



DNS EXAMPLE – WAN DISPATCHING

- Ping 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 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* 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 6, 2020 | TCSS558: Applied Distributed Computing [Winter 2020]
School of Engineering and Technology, University of Washington - Tacoma | L10.44

44

DNS EXAMPLE – WAN DISPATCHING

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

Latency to ping "VA" google in WA: ~3.63x

WA laptop: local-google 22.458ms to VA-google 81.637ms

Latency to ping "WA" google in VA: ~48.7x

Virginia ec2 VM: local-google 1.278ms to WA-google 62.349!

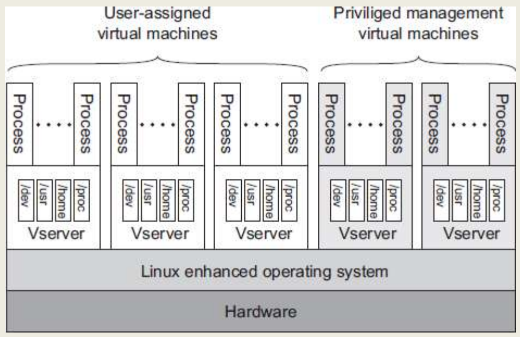
- From local wireless network, ping VA us-east-1 google :
- Ping 172.217.9.196: Average RTT=81.637ms (11 attempts, 15 hops)

45

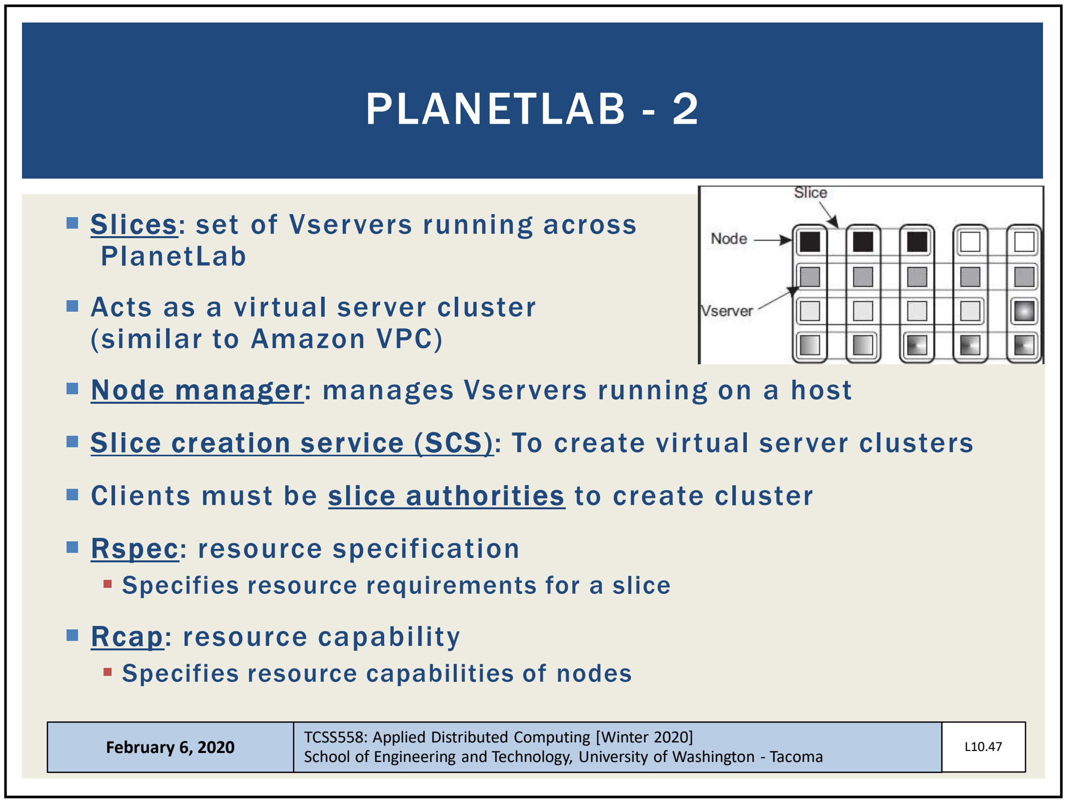
EXAMPLE: PLANETLAB

- Unstructured heterogeneous cluster of servers
- Similar to grid but organized as cluster (no grid middleware)
- Testbed established in 2002 for computer networking and distributed systems research
- Organizations share nodes in the cluster

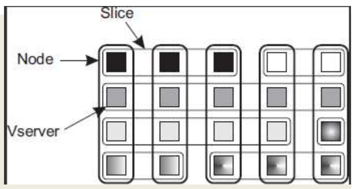
Leverages Linux Vservers Early "containers" similar to Docker


User-assigned virtual machines
Priviliged management virtual machines
Process
Process
/dev
/usr
/home
/proc
Vserver
Linux enhanced operating system
Hardware

46

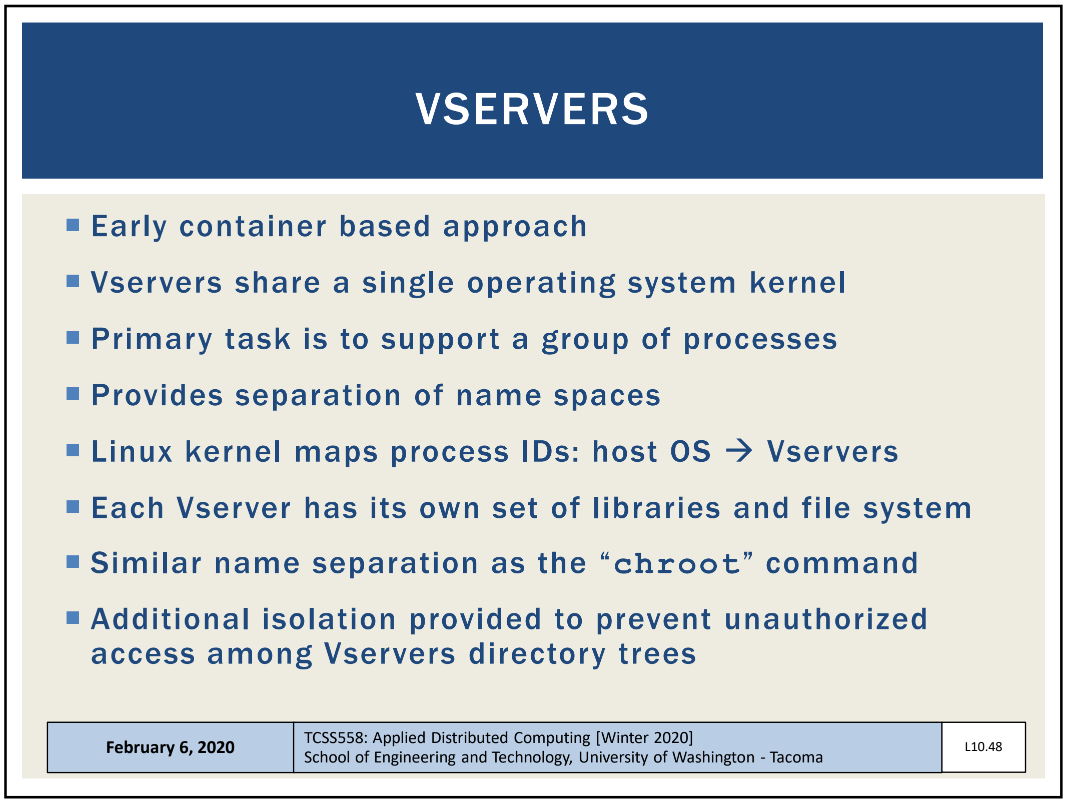


PLANETLAB - 2



- **Slices**: set of Vservers running across PlanetLab
- Acts as a virtual server cluster (similar to Amazon VPC)
- **Node manager**: manages Vservers running on a host
- **Slice creation service (SCS)**: To create virtual server clusters
- Clients must be **slice authorities** to create cluster
- **Rspec**: resource specification
  - Specifies resource requirements for a slice
- **Rcap**: resource capability
  - Specifies resource capabilities of nodes

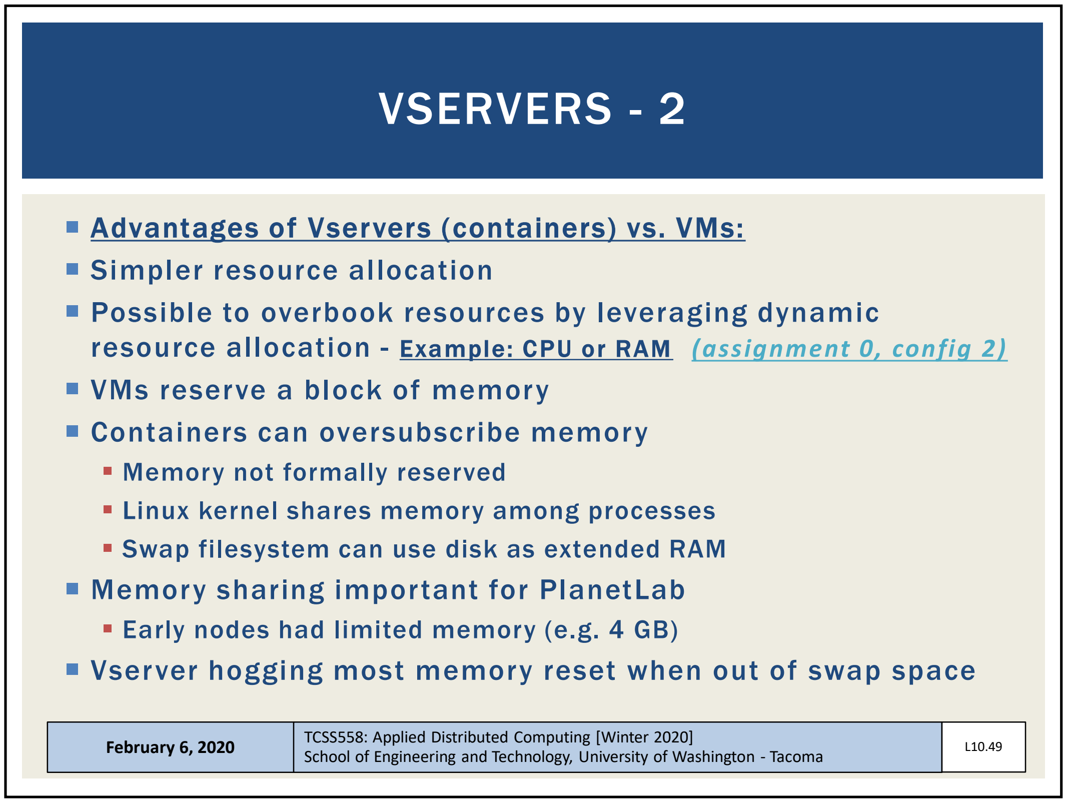
47



VSERVERS

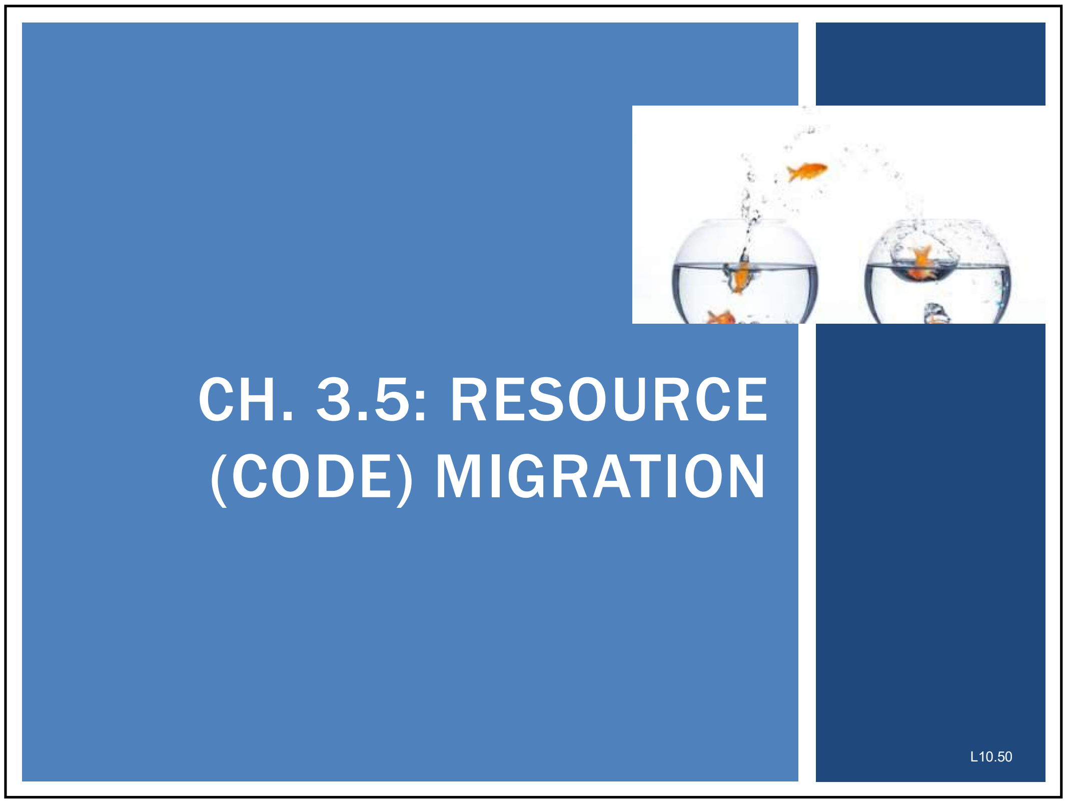
- Early container based approach
- Vservers share a single operating system kernel
- Primary task is to support a group of processes
- Provides separation of name spaces
- Linux kernel maps process IDs: host OS → Vservers
- Each Vserver has its own set of libraries and file system
- Similar name separation as the "`chroot`" command
- Additional isolation provided to prevent unauthorized access among Vservers directory trees

48

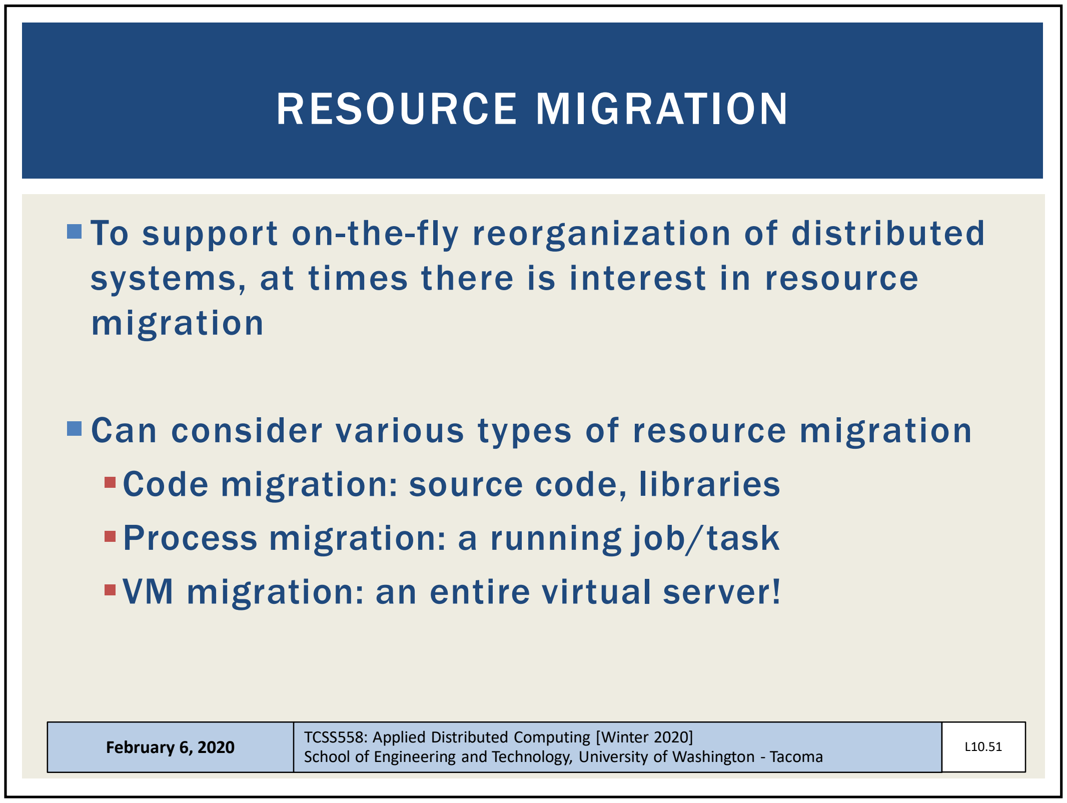


VSERVERS - 2

- **Advantages of Vservers (containers) vs. VMs:**
- Simpler resource allocation
- Possible to overbook resources by leveraging dynamic resource allocation - **Example: CPU or RAM** *(assignment 0, config 2)*
- VMs reserve a block of memory
- Containers can oversubscribe memory
  - Memory not formally reserved
  - Linux kernel shares memory among processes
  - Swap filesystem can use disk as extended RAM
- Memory sharing important for PlanetLab
  - Early nodes had limited memory (e.g. 4 GB)
- Vserver hogging most memory reset when out of swap space



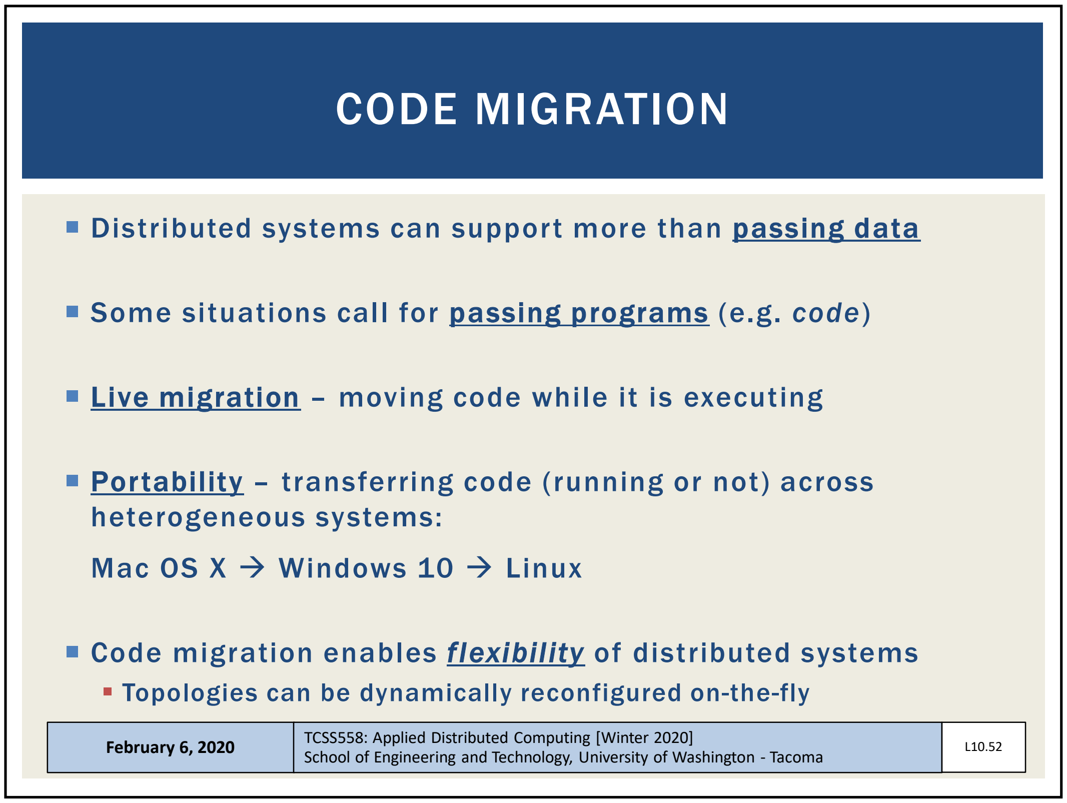
CH. 3.5: RESOURCE (CODE) MIGRATION



RESOURCE MIGRATION

- To support on-the-fly reorganization of distributed systems, at times there is interest in resource migration
- Can consider various types of resource migration
  - Code migration: source code, libraries
  - Process migration: a running job/task
  - VM migration: an entire virtual server!

51

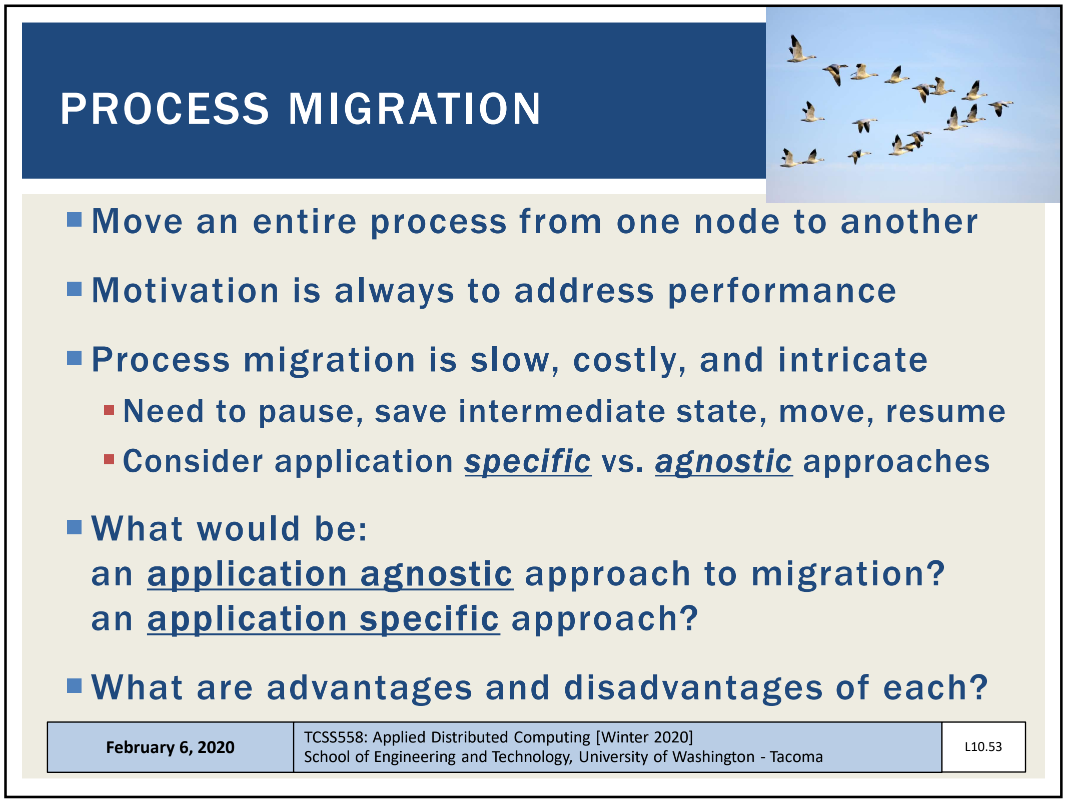


CODE MIGRATION

- Distributed systems can support more than **passing data**
- Some situations call for **passing programs** (e.g. *code*)
- **Live migration** – moving code while it is executing
- **Portability** – transferring code (running or not) across heterogeneous systems:

  Mac OS X → Windows 10 → Linux
- Code migration enables ***flexibility*** of distributed systems
  - Topologies can be dynamically reconfigured on-the-fly

52

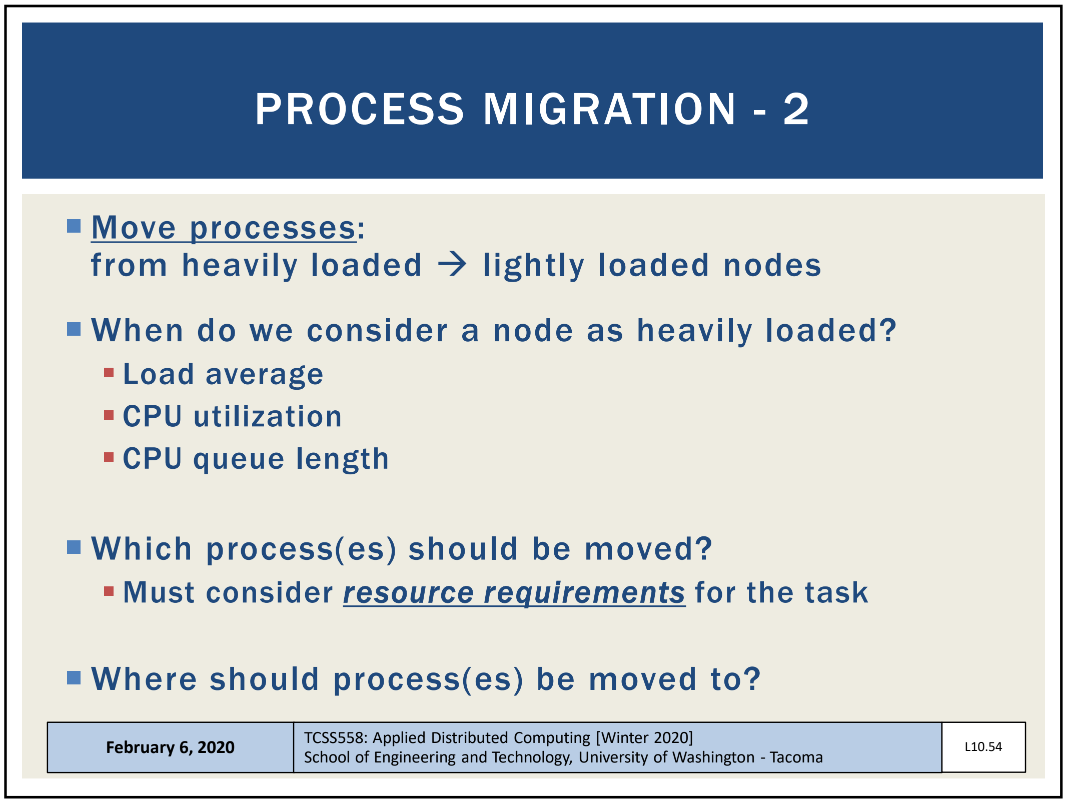


PROCESS MIGRATION

- Move an entire process from one node to another
- Motivation is always to address performance
- Process migration is slow, costly, and intricate
  - Need to pause, save intermediate state, move, resume
  - Consider application ***specific*** vs. ***agnostic*** approaches
- What would be:
  an **application agnostic** approach to migration?
  an **application specific** approach?
- What are advantages and disadvantages of each?

February 6, 2020	TCSS558: Applied Distributed Computing [Winter 2020] School of Engineering and Technology, University of Washington - Tacoma	L10.53

53



PROCESS MIGRATION - 2

- Move processes:
  from heavily loaded → lightly loaded nodes
- When do we consider a node as heavily loaded?
  - Load average
  - CPU utilization
  - CPU queue length
- Which process(es) should be moved?
  - Must consider ***resource requirements*** for the task
- Where should process(es) be moved to?

February 6, 2020	TCSS558: Applied Distributed Computing [Winter 2020] School of Engineering and Technology, University of Washington - Tacoma	L10.54

54

MOTIVATIONS FOR MIGRATION

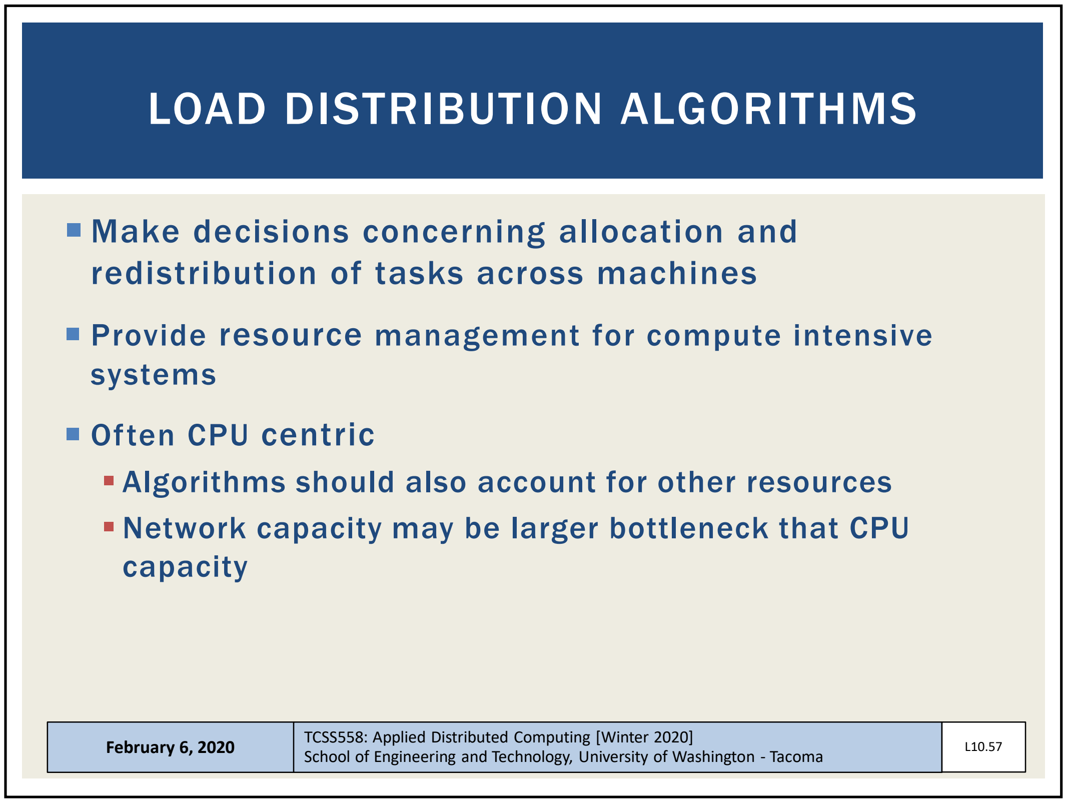
- Can migrate <u>processes</u> or entire <u>virtual machines</u>
- <u>Goals:</u>
  - Off-loading machines: reduce load on oversubscribed servers
  - Loading machine: ensure machine has enough work to do
  - Minimize total hosts/servers in use to save energy/cost
- <u>VM migration:</u>
- Migrate complete VMs with apps to lightly loaded hosts
- Generally, VM migration is easier than process migration
- <u>Is VM migration application specific or agnostic?</u>

55

LINUX CRIU

- Linux (CRIU) Checkpoint restore in userspace
- Linux tool: https://www.criu.org/
- Supports freezing a running application (or part of it) to create a checkpoint to persistent storage (e.g. disk) as a collection of files.
  - This means saving the state of RAM to disk
- Can use checkpoint files to restore and run the application from the point it was frozen at.
- Distinctive feature of CRIU is that it can be run in the user space (CPU user mode), rather than in kernel mode.
- CRIU can save a Docker container's state for migration elsewhere

56

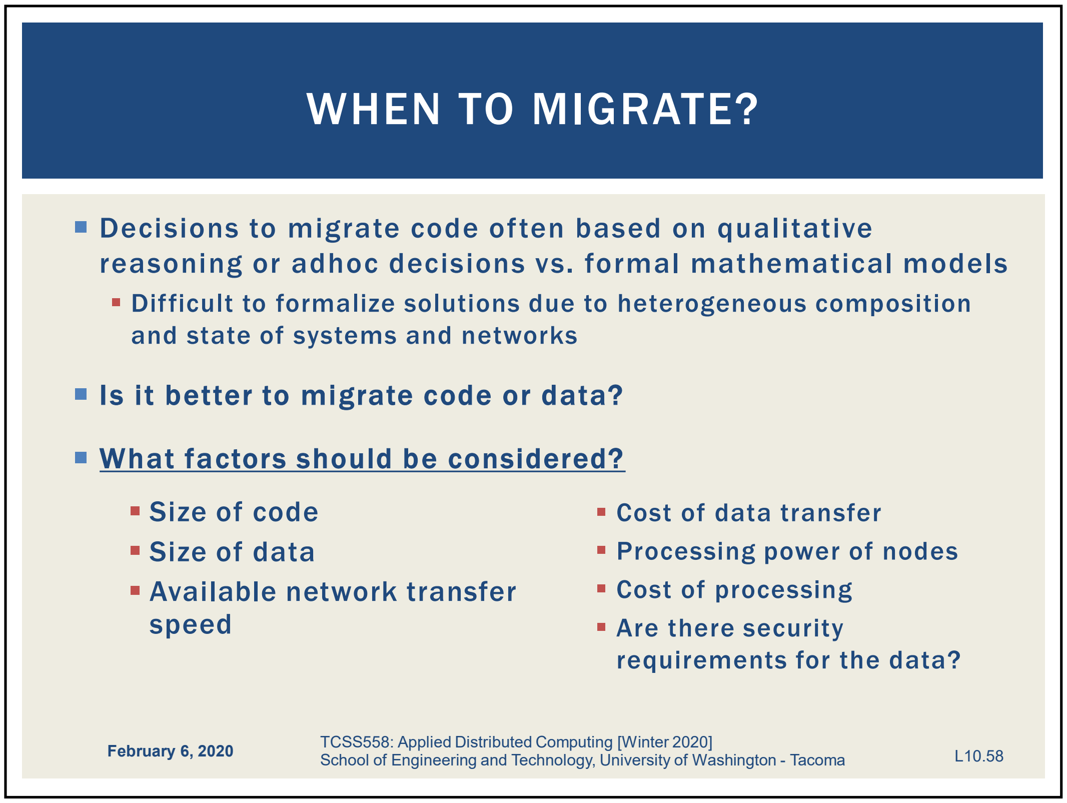


LOAD DISTRIBUTION ALGORITHMS

- Make decisions concerning allocation and redistribution of tasks across machines
- Provide resource management for compute intensive systems
- Often CPU centric
  - Algorithms should also account for other resources
  - Network capacity may be larger bottleneck that CPU capacity

February 6, 2020	TCSS558: Applied Distributed Computing [Winter 2020] School of Engineering and Technology, University of Washington - Tacoma	L10.57

57



WHEN TO MIGRATE?

- Decisions to migrate code often based on qualitative reasoning or adhoc decisions vs. formal mathematical models
  - Difficult to formalize solutions due to heterogeneous composition and state of systems and networks
- Is it better to migrate code or data?
- <u>What factors should be considered?</u>
  - Size of code
  - Size of data
  - Available network transfer speed
  - Cost of data transfer
  - Processing power of nodes
  - Cost of processing
  - Are there security requirements for the data?

February 6, 2020 TCSS558: Applied Distributed Computing [Winter 2020] School of Engineering and Technology, University of Washington - Tacoma L10.58

58

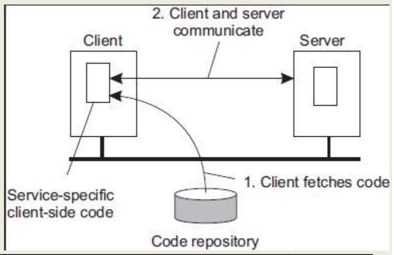
APPROACHES TO CODE MIGRATION

- Traditional clients
  - Client interacts with server using specific protocol
  - Tight coupling of client->server limits system flexibility
  - Difficult to change protocol when there are ***many*** clients
- Dynamic web clients
  - Web browser downloads client code immediately before use
  - New versions can readily be distributed

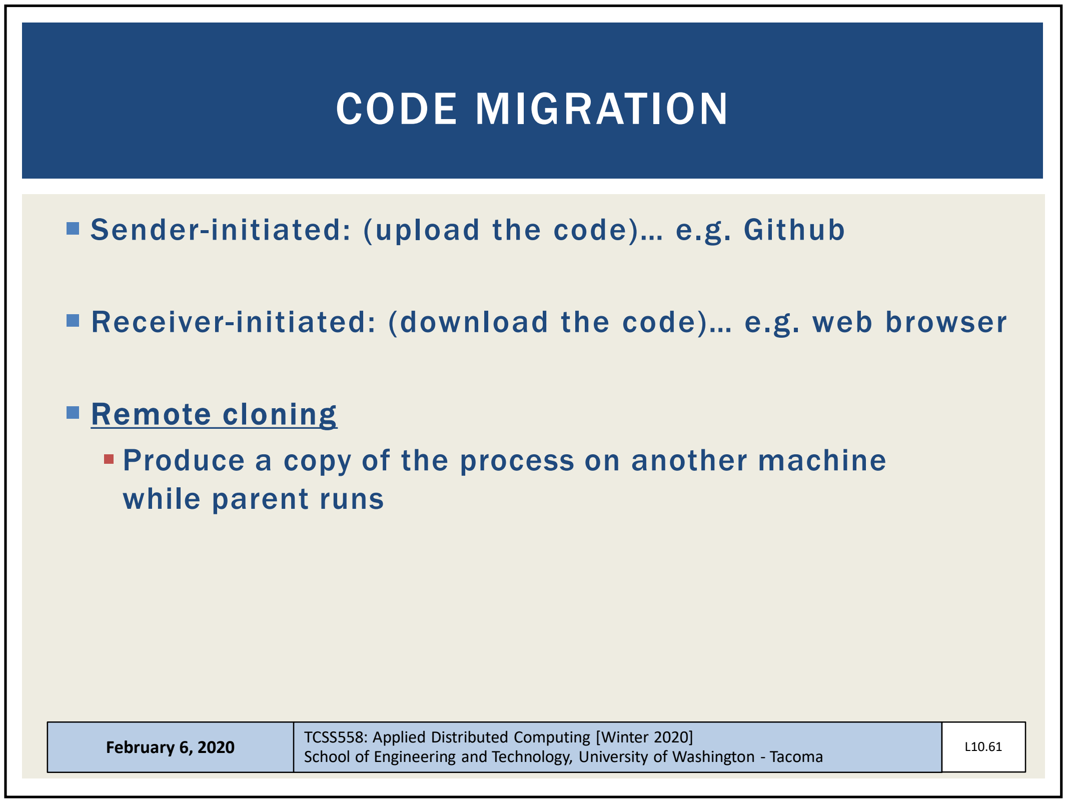
59

DYNAMIC WEB CLIENTS

- Advantages
  - Client code loaded in as necessary
  - Discarded when no longer needed
  - Can easily change the client/server protocol
- Disadvantages
  - Security: we have to trust the code
  - Downloading client requires network bandwidth & time


2. Client and server communicate
Client
Server
1. Client fetches code
Service-specific client-side code
Code repository

60

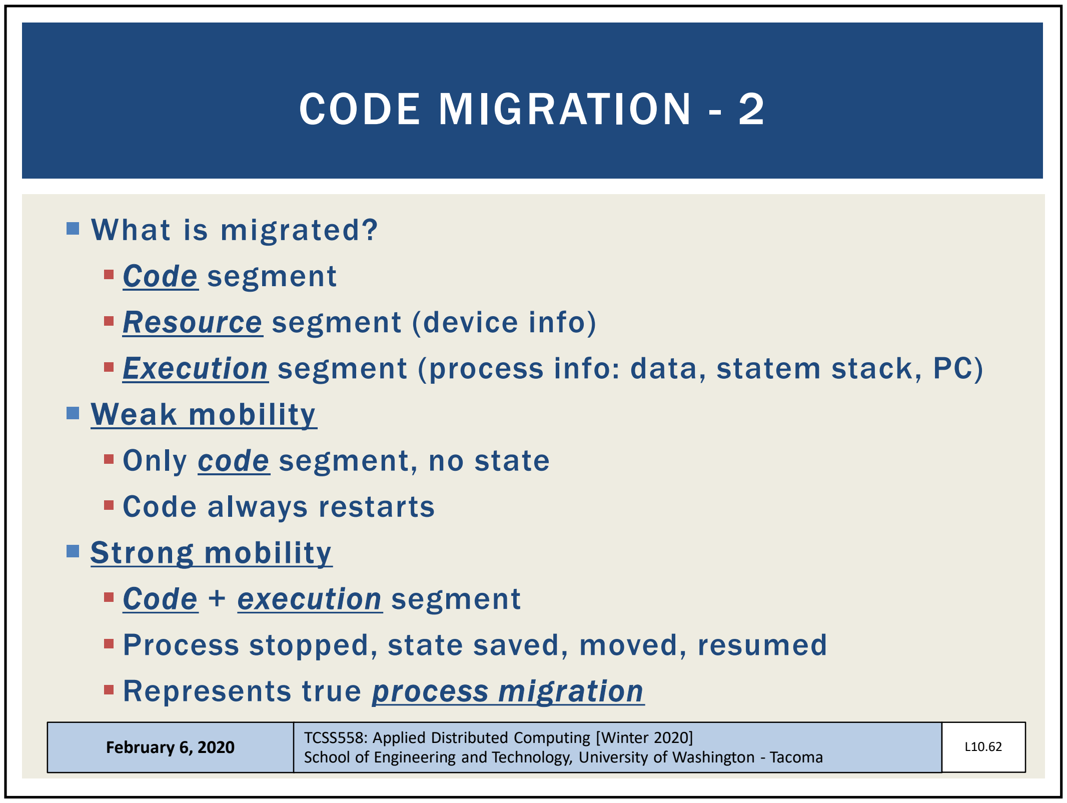


CODE MIGRATION

- Sender-initiated: (upload the code)… e.g. Github
- Receiver-initiated: (download the code)… e.g. web browser
- **Remote cloning**
  - Produce a copy of the process on another machine while parent runs

February 6, 2020	TCSS558: Applied Distributed Computing [Winter 2020] School of Engineering and Technology, University of Washington - Tacoma	L10.61

61



CODE MIGRATION - 2

- What is migrated?
  - ***Code*** segment
  - ***Resource*** segment (device info)
  - ***Execution*** segment (process info: data, statem stack, PC)
- **Weak mobility**
  - Only ***code*** segment, no state
  - Code always restarts
- **Strong mobility**
  - ***Code*** + ***execution*** segment
  - Process stopped, state saved, moved, resumed
  - Represents true ***process migration***

February 6, 2020	TCSS558: Applied Distributed Computing [Winter 2020] School of Engineering and Technology, University of Washington - Tacoma	L10.62

62

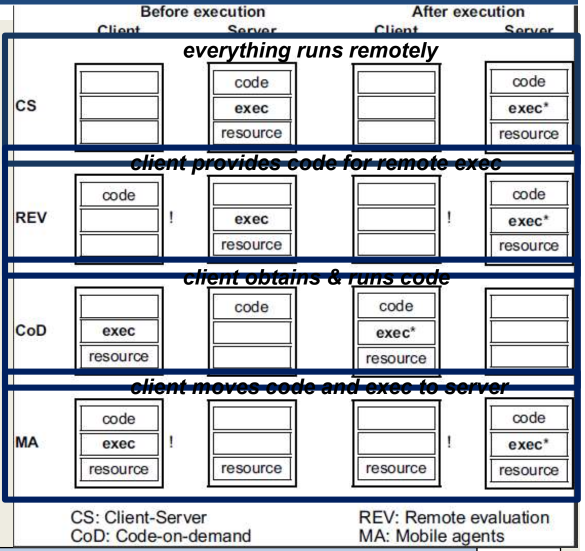
CODE MOBILITY TYPES

* indicates what is modified

- CS: Client-Server
- REV: Remote Evaluation
- CoD: Code-on-demand
- MA: Mobile agents

- Where does state get modified?

- State is stored in **exec**



63

MIGRATION OF HETEROGENEOUS SYSTEMS

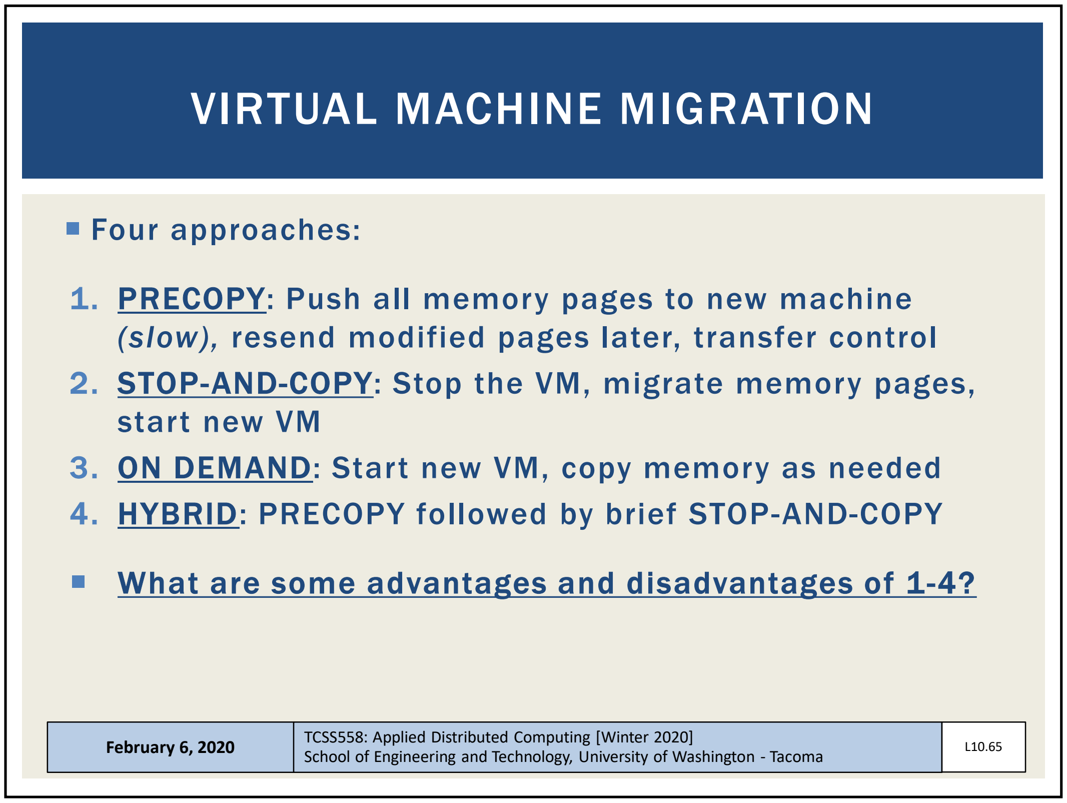
- Assumption: code will always work at new node
- Invalid if node architecture is different (*heterogeneous*)

- What approaches are available to migrate code across heterogeneous systems?

- Intermediate code
  - 1970s Pascal: generate machine-independent intermediate code
  - Programs could then run anywhere
  - **Today:** web languages: Javascript, Java

- VM Migration

64



VIRTUAL MACHINE MIGRATION

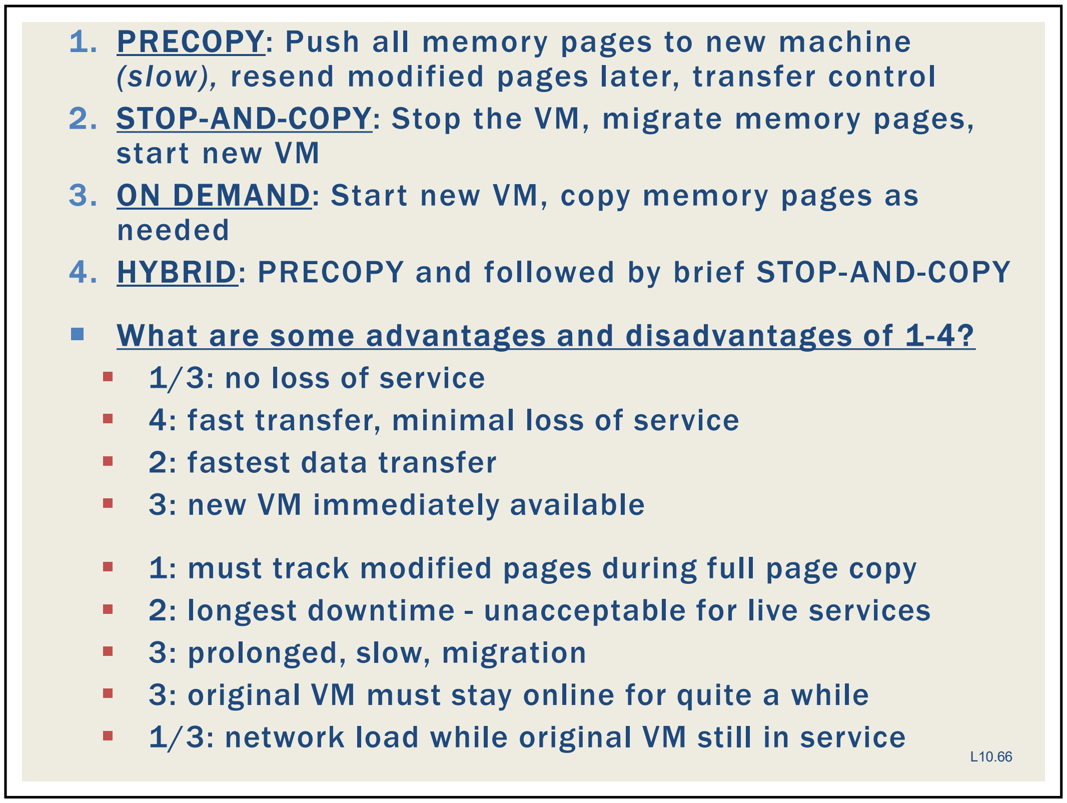
- Four approaches:

1. PRECOPY: Push all memory pages to new machine *(slow)*, resend modified pages later, transfer control
2. STOP-AND-COPY: Stop the VM, migrate memory pages, start new VM
3. ON DEMAND: Start new VM, copy memory as needed
4. HYBRID: PRECOPY followed by brief STOP-AND-COPY

- What are some advantages and disadvantages of 1-4?

February 6, 2020 | TCSS558: Applied Distributed Computing [Winter 2020] School of Engineering and Technology, University of Washington - Tacoma | L10.65

65

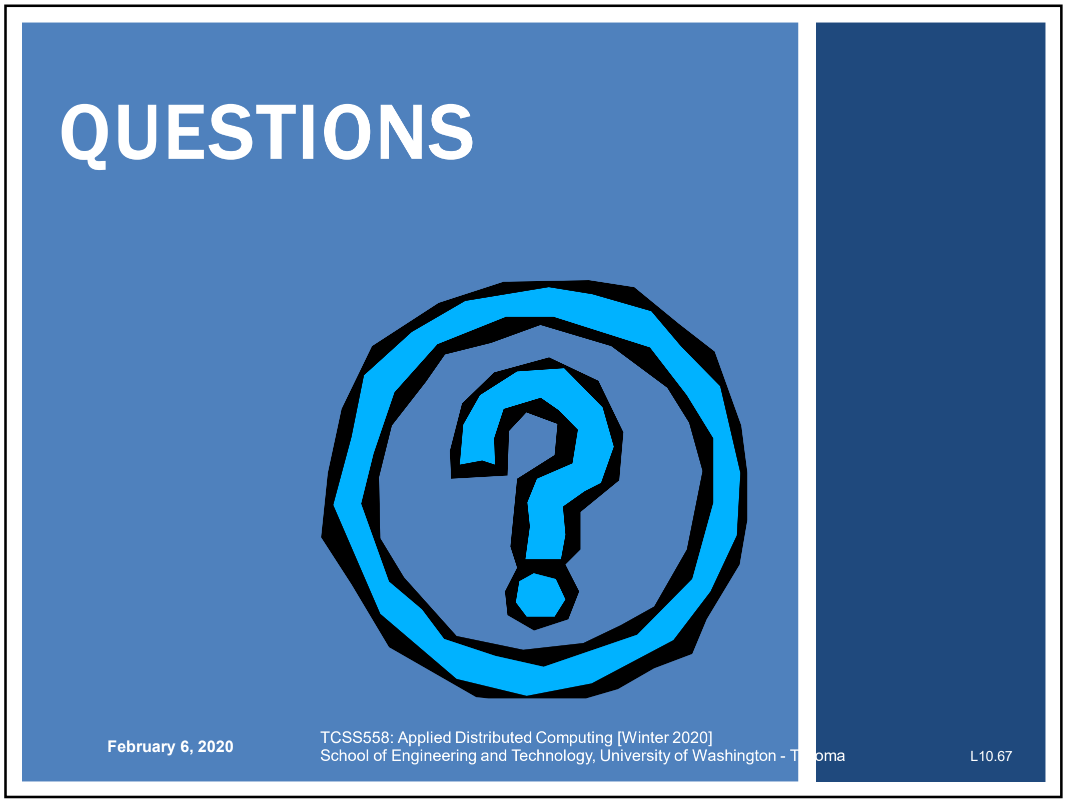


1. PRECOPY: Push all memory pages to new machine *(slow)*, resend modified pages later, transfer control
2. STOP-AND-COPY: Stop the VM, migrate memory pages, start new VM
3. ON DEMAND: Start new VM, copy memory pages as needed
4. HYBRID: PRECOPY and followed by brief STOP-AND-COPY

- What are some advantages and disadvantages of 1-4?
  - 1/3: no loss of service
  - 4: fast transfer, minimal loss of service
  - 2: fastest data transfer
  - 3: new VM immediately available
  - 1: must track modified pages during full page copy
  - 2: longest downtime - unacceptable for live services
  - 3: prolonged, slow, migration
  - 3: original VM must stay online for quite a while
  - 1/3: network load while original VM still in service

L10.66

66



QUESTIONS

February 6, 2020

TCSS558: Applied Distributed Computing [Winter 2020]
School of Engineering and Technology, University of Washington - Tacoma

L10.67

67