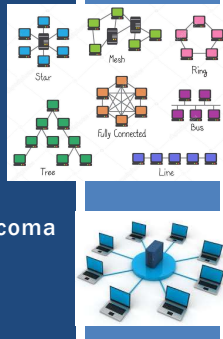


## TCSS 558: APPLIED DISTRIBUTED COMPUTING

### Chapter 3 - Processes

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



## OBJECTIVES

- Homework 1
- Midterm 2/13
- Feedback
- Chapter 3 Processes
  - 3.1 Threads – cont'd
  - 3.2 Virtualization
  - 3.3 Clients
  - 3.4 Servers

February 6, 2019

TCCS558: Applied Distributed Computing [Winter 2019]  
School of Engineering and Technology, University of Washington - Tacoma

L8.2

## FEEDBACK – 2/6

- How do you determine if it is more time efficient to generate a new thread for every request, or to create a thread pool (slide 24)?
- For servicing incoming client requests, having a pre-initialized thread pool should *generally* always be more time efficient than creating thread(s) on demand to service incoming requests
- Measure thread creation/destruction time – *using a thread pool should negate this latency*

February 6, 2019

TCCS558: Applied Distributed Computing [Winter 2019]  
School of Engineering and Technology, University of Washington - Tacoma

L8.3

## POSSIBLE CONCERNS W/ THREAD POOLS

- How large should the thread pool be? Larger pools require more memory and longer up-front initialization time.
- How much memory is required to maintain the thread pool? Can we afford the memory investment? – *if the pool is too large there may not be enough memory to process user requests*
- Issues recycling threads: should threads share and maintain open database connections? Or any other connection-oriented TCP sessions for reuse?
- How can a request's session context be decoupled from individual worker threads?

February 6, 2019

TCCS558: Applied Distributed Computing [Winter 2019]  
School of Engineering and Technology, University of Washington - Tacoma

L8.4

## FEEDBACK - 2

- I am still unclear on the difference between paravirtualization and virtualization in hardware.
- Is the difference the extent to which it occurs on the hardware?
- Based on textbook note 3.5:
- **GOAL:** run all user mode instructions directly on the CPU
- x86 instruction set has ~17 privileged user mode instructions
- Paravirtualization: Hypervisor provides efficient hypercalls to eliminate side effects of privileged instructions
- Guest OS uses drivers and modified kernel to call hypercalls
- **Paravirtualization is the software coordination between the hypervisor and guest (via hypercalls) used to improve performance**
- Performance O/H ~10%-50% depending on PV type + workload

February 6, 2019

TCCS558: Applied Distributed Computing [Winter 2019]  
School of Engineering and Technology, University of Washington - Tacoma

L8.5

## FEEDBACK - 3

- **Virtualization in hardware:**
- Specific technology to support virtualization built-in to hardware
- **CPUs**
- **VT-x** is codename for Intel's CPU Virtualization support (2005)
- Intel's technology for virtualization on the x86 platform
- Ten new instructions to permit entering and exiting a virtual execution mode where the guest OS perceives itself as running with full privilege (ring 0), but the host OS remains protected
- **PCI Bus**
- **SR-IOV** – single root input/output virtualization (2009)
  - Supports sharing PCI adapters (e.g. network cards) among many guests

February 6, 2019

TCCS558: Applied Distributed Computing [Winter 2019]  
School of Engineering and Technology, University of Washington - Tacoma

L8.6

## FEEDBACK - 4

- **I/O**
- **VT-d** – Intel's technology for directed I/O (2009)
  - Input/output memory management unit (IOMMU) that allows guest VMs to directly use peripheral devices: network card, accelerated graphics cards, and hard-drive controllers, through DMA (direct memory access) and interrupt remapping. Also called PCI passthrough
- **SSDs**
- **NVMe** – non-volatile memory express (2011)
  - Open logical device interface specification for accessing non-volatile storage media (e.g. SSDs) attached via a PCI Express (PCIe) bus
  - Next gen SSD interface that integrates virtualization enhancements
  - Various EC2 instances now support local NVMe SSDs:  
See: <https://www.ec2instances.info/?filter=NVMe>

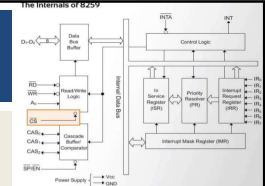
February 6, 2019

TCCS558: Applied Distributed Computing [Winter 2019]  
School of Engineering and Technology, University of Washington - Tacoma

L8.7

## FEEDBACK - 5

- **CPU Interrupts**
- **APICv (2013)**



- What is a Programmable interrupt controller (PIC) ?
- A modular device within CPUs used to combine several sources of interrupt onto one or more CPU lines, while allowing priority levels to be assigned to its interrupt outputs.
- When a PIC has multiple interrupt outputs to assert, it asserts them in priority order.
- APIC: is an advanced programmable interrupt controller which supersedes the original 8259 coupled w/ 8086 CPU

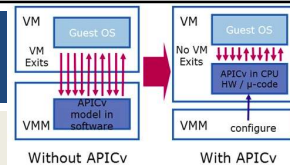
February 6, 2019

TCCS558: Applied Distributed Computing [Winter 2019]  
School of Engineering and Technology, University of Washington - Tacoma

L8.8

## FEEDBACK - 6

- **What is APICv?**
- Advanced programmable interrupt controller virtualization
- APICv: interrupt controller that includes support for targeting interrupt overhead reduction in virtualization environments
- **Without APICv:** Figure shows all virtualized activities relating to interrupts and the APIC to and from the guest OS go through the VMM
- **With APICv:** activities relating to interrupts are executed by the hardware (APICv), not in the VMM.
- Eliminates need to issue the "VM exit" command and reduces overhead providing increased I/O throughput for VMs.
- Available on CPUs in late 2013 into 2014

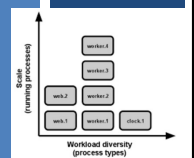


February 6, 2019

TCCS558: Applied Distributed Computing [Winter 2019]  
School of Engineering and Technology, University of Washington - Tacoma

L8.9

## CH. 3: PROCESSES



L8.10

## CH. 3.3: CLIENTS



L8.11

## TYPES OF CLIENTS

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

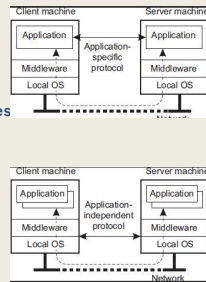
February 6, 2019

TCCS558: Applied Distributed Computing [Winter 2019]  
School of Engineering and Technology, University of Washington - Tacoma

L8.12

## 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 6, 2019

TCSS558: Applied Distributed Computing [Winter 2019]  
School of Engineering and Technology, University of Washington - Tacoma

L8.13

## 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

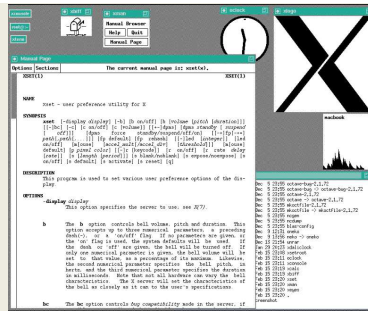
February 6, 2019

TCSS558: Applied Distributed Computing [Winter 2019]  
School of Engineering and Technology, University of Washington - Tacoma

L8.14

## X WINDOWS - 2

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



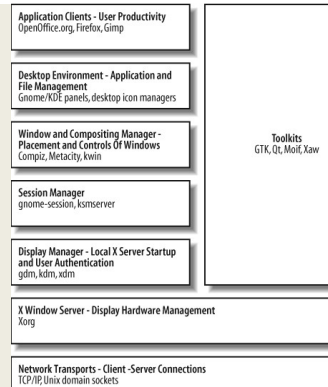
February 6, 2019

TCSS558: Applied Distributed Computing [Winter 2019]  
School of Engineering and Technology, University of Washington - Tacoma

L8.15

### 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



February 6, 2019

TCSS558: Applied Distributed Computing [Winter 2019]  
School of Engineering and Technology, University of Washington - Tacoma

L8.16

## 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 6, 2019

TCSS558: Applied Distributed Computing [Winter 2019]  
School of Engineering and Technology, University of Washington - Tacoma

L8.17

## 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
[ -x $HOME/.Xresources ] && xrdb $HOME/.Xresources
xsetroot -solid grey

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

February 6, 2019

TCSS558: Applied Distributed Computing [Winter 2019]  
School of Engineering and Technology, University of Washington - Tacoma

L8.18

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 6, 2019

TCCS558: Applied Distributed Computing [Winter 2019]  
School of Engineering and Technology, University of Washington - Tacoma

L8.19

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 6, 2019

TCCS558: Applied Distributed Computing [Winter 2019]  
School of Engineering and Technology, University of Washington - Tacoma

L8.20

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 6, 2019

TCCS558: Applied Distributed Computing [Winter 2019]  
School of Engineering and Technology, University of Washington - Tacoma

L8.21

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 6, 2019

TCCS558: Applied Distributed Computing [Winter 2019]  
School of Engineering and Technology, University of Washington - Tacoma

L8.22

REMOTE COMPUTER IN THE CLOUD

- EC2 instance with a GUI. . .!!!

February 6, 2019

TCCS558: Applied Distributed Computing [Winter 2019]  
School of Engineering and Technology, University of Washington - Tacoma

L8.23

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)<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 6, 2019

TCCS558: Applied Distributed Computing [Winter 2019]  
School of Engineering and Technology, University of Washington - Tacoma

L8.24

## 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 6, 2019

TCS558: Applied Distributed Computing [Winter 2019]  
School of Engineering and Technology, University of Washington - Tacoma

L8.25

## 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 lose application semantics
- Can transport any GUI this way
- THINC** hybrid approach
- Send video device driver commands over network
- More powerful than pixel based operations
- Less powerful compared to protocols such as X

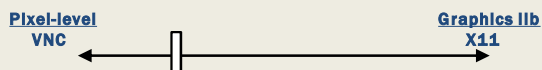
February 6, 2019

TCS558: Applied Distributed Computing [Winter 2019]  
School of Engineering and Technology, University of Washington - Tacoma

L8.26

## TRADEOFFS: ABSTRACTION OF REMOTE DISPLAY PROTOCOLS

- Tradeoff space: abstraction level of remote display protocols



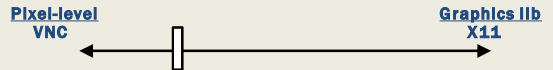
February 6, 2019

TCS558: Applied Distributed Computing [Winter 2019]  
School of Engineering and Technology, University of Washington - Tacoma

L8.27

## TRADEOFFS: ABSTRACTION OF REMOTE DISPLAY PROTOCOLS

- Tradeoff space: abstraction level of remote display protocols



- 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

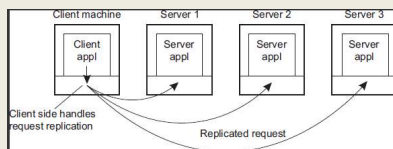
February 6, 2019

TCS558: Applied Distributed Computing [Winter 2019]  
School of Engineering and Technology, University of Washington - Tacoma

L8.28

## 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



February 6, 2019

TCS558: Applied Distributed Computing [Winter 2019]  
School of Engineering and Technology, University of Washington - Tacoma

L8.29

## 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 6, 2019

TCS558: Applied Distributed Computing [Winter 2019]  
School of Engineering and Technology, University of Washington - Tacoma

L8.30



## CH. 3.4: SERVERS

L8.31

## 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, 2019 TCCS558: Applied Distributed Computing [Winter 2019]  
School of Engineering and Technology, University of Washington - Tacoma L8.32

## 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, 2019 TCCS558: Applied Distributed Computing [Winter 2019]  
School of Engineering and Technology, University of Washington - Tacoma L8.33

## 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, 2019 TCCS558: Applied Distributed Computing [Winter 2019]  
School of Engineering and Technology, University of Washington - Tacoma L8.34

## COMMON PORTS

packetlife.net

TCP/UDP Port Numbers			
7 Echo	554 RTSP	2745 Eagle 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 GUPP	8118 Proxy
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 Eagle H
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 XMGT
88 Kerberos	989-990 FTP over SSL	4333 mSQL	9800 WebDAV
102 MS Exchange	993 IMAP4 over SSL	4444 Raster	9898 Dabber
110 POP3	995 POP3 over SSL	4664 Google Desktop	9988 NeoSoylark
113 ident	1025 Microsoft RPC	4672 Oracle	9999 Urchin
119 NNTP (Usenet)	1026-1029 Windows Messenger	4899 Radmin	10000 Webmin
123 NTP	1080 SOCKS Proxy	5000 UmiP	10000 BackupExec
135 Microsoft RPC	1080 RDP	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 BAST	5190 XMPP	14562 Supermicro

February 6, 2019 TCCS558: Applied Distributed Computing [Winter 2019]  
School of Engineering and Technology, University of Washington - Tacoma L8.36

## TYPES OF SERVERS

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

February 6, 2019 TCCS558: Applied Distributed Computing [Winter 2019]  
School of Engineering and Technology, University of Washington - Tacoma L8.36

## 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 6, 2019

TCCS558: Applied Distributed Computing [Winter 2019]  
School of Engineering and Technology, University of Washington - Tacoma

L8.37

## 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 6, 2019

TCCS558: Applied Distributed Computing [Winter 2019]  
School of Engineering and Technology, University of Washington - Tacoma

L8.38

## 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 6, 2019

TCCS558: Applied Distributed Computing [Winter 2019]  
School of Engineering and Technology, University of Washington - Tacoma

L8.39

## 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**
  - Losing 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, 2019

TCCS558: Applied Distributed Computing [Winter 2019]  
School of Engineering and Technology, University of Washington - Tacoma

L8.40

## 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 6, 2019

TCCS558: Applied Distributed Computing [Winter 2019]  
School of Engineering and Technology, University of Washington - Tacoma

L8.41

## 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 6, 2019

TCCS558: Applied Distributed Computing [Winter 2019]  
School of Engineering and Technology, University of Washington - Tacoma

L8.42



## OBJECT SERVERS

- 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**
  - 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 6, 2019

TCSS558: Applied Distributed Computing [Winter 2019]  
School of Engineering and Technology, University of Washington - Tacoma

L8.43

## 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 6, 2019

TCSS558: Applied Distributed Computing [Winter 2019]  
School of Engineering and Technology, University of Washington - Tacoma

L8.44

## 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

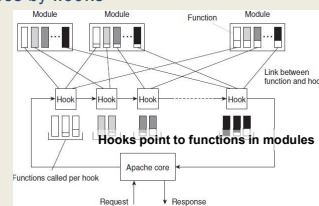
February 6, 2019

TCSS558: Applied Distributed Computing [Winter 2019]  
School of Engineering and Technology, University of Washington - Tacoma

L8.45

## 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



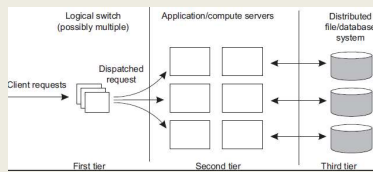
February 6, 2019

TCSS558: Applied Distributed Computing [Winter 2019]  
School of Engineering and Technology, University of Washington - Tacoma

L8.46

## 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 6, 2019

TCSS558: Applied Distributed Computing [Winter 2019]  
School of Engineering and Technology, University of Washington - Tacoma

L8.47

## 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 6, 2019

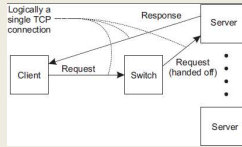
TCSS558: Applied Distributed Computing [Winter 2019]  
School of Engineering and Technology, University of Washington - Tacoma

L8.48



## 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 6, 2019

TCCS558: Applied Distributed Computing [Winter 2019]  
School of Engineering and Technology, University of Washington - Tacoma

L8.49

## 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 6, 2019

TCCS558: Applied Distributed Computing [Winter 2019]  
School of Engineering and Technology, University of Washington - Tacoma

L8.50

## 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 6, 2019

TCCS558: Applied Distributed Computing [Winter 2019]  
School of Engineering and Technology, University of Washington - Tacoma

L8.51

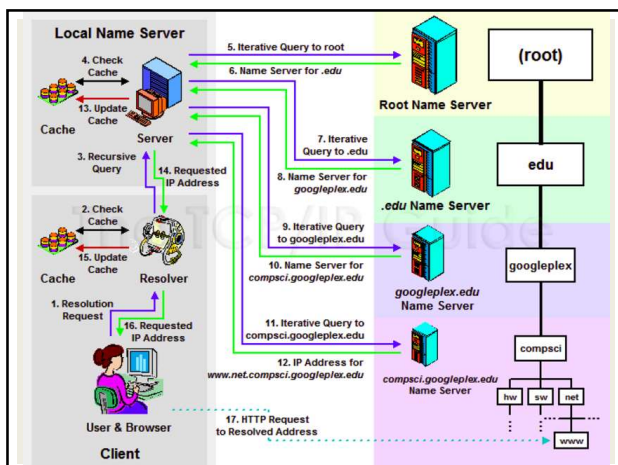
## DNS LOOKUP

- 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 6, 2019

TCCS558: Applied Distributed Computing [Winter 2019]  
School of Engineering and Technology, University of Washington - Tacoma

L8.52



## 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, 2019

TCCS558: Applied Distributed Computing [Winter 2019]  
School of Engineering and Technology, University of Washington - Tacoma

L8.54

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 6, 2019

TCSS558: Applied Distributed Computing [Winter 2019]  
School of Engineering and Technology, University of Washington - Tacoma

L8.55

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 6, 2019

TCSS558: Applied Distributed Computing [Winter 2019]  
School of Engineering and Technology, University of Washington - Tacoma

L8.56

QUESTIONS




February 6, 2019

TCSS558: Applied Distributed Computing [Winter 2019]  
School of Engineering and Technology, University of Washington - Tacoma

L8.57

EXTRA SLIDES



58