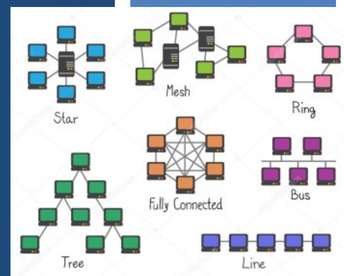
TCSS 558:
APPLIED DISTRIBUTED COMPUTING



Processes:
Clients & Servers

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

1

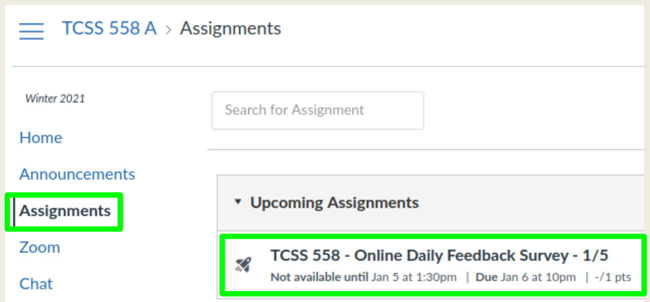
OBJECTIVES – 2/2

- **Questions from 1/31**
- Assignment 0: Cloud Computing Infrastructure Tutorial
- Assignment 1: Key/Value Store
  - Java Maven project template files posted
- Midterm Thursday February 9
  - 2nd hour - Tuesday February 7 – practice midterm questions
- Chapter 3: Processes
  - Chapter 3.3: Clients
  - Chapter 3.4: Servers

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



February 2, 2023	TCSS558: Applied Distributed Computing [Winter 2023] School of Engineering and Technology, University of Washington - Tacoma	L10.3

3

SURVEY QUESTIONS

- Survey has two questions:
- Be sure to add your questions about the previous class to the ***second question***
- 1st question: After today's class, comment on any new concepts that you learned about?
  - *Have been getting questions here…*
- 2nd question: After today's class, what point(s) remain least clear to you?
  - **>> Please add questions HERE**

February 2, 2023	TCSS558: Applied Distributed Computing [Winter 2023] School of Engineering and Technology, University of Washington - Tacoma	L10.4

4

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:

1	2	3	4	5	6	7	8	9	10
Mostly Review To Me				Equal New and Review					Mostly New to Me

Question 2 0.5 pts

Please rate the pace of today's class:

1	2	3	4	5	6	7	8	9	10
Slow				Just Right					Fast

February 2, 2023 TCSS558: Applied Distributed Computing [Winter 2023]
School of Engineering and Technology, University of Washington - Tacoma L10.5

5

MATERIAL / PACE

- Please classify your perspective on material covered in today's class (27 respondents):
- 1-mostly review, 5-equal new/review, 10-mostly new
- **Average – 6.02 (↓ - *previous 6.25*)**

- Please rate the pace of today's class:
- 1-slow, 5-just right, 10-fast
- **Average – 5.67 (↑ - *previous 5.62*)**

February 2, 2023 TCSS558: Applied Distributed Computing [Winter 2023]
School of Engineering and Technology, University of Washington - Tacoma L10.6

6

FEEDBACK FROM 1/31

- ***When single thread(ed server) blocks for I/O, doesn't it mean it handles I/O after other client requests are completed?***
- With single-threaded server:
- Main loop of server: receives request, examines it, carries it out to completion before processing next one
- While waiting for disk: server is idle and does not process other requests. Requests from other clients cannot be handled.
  - Requests may be queued or ignored (lost)
- If server is running on a dedicated machine, the CPU will be idle while the server waits for the disk.
- Net result: many fewer requests processed per unit time
  - Lower throughput, higher turnaround time

FEEDBACK - 2

- ***Isn't this the same as the finite state machine performing I/O asynchronously?***
- FSM: disk I/O operations are asynchronous
- They occur in parallel to request processing
- When the disk I/O completes, the server thread is interrupt from whatever it is doing, so that it can process the results of the disk I/O operations

FEEDBACK - 3

- ***Or does it mean the single threaded server does not handle I/O at all even after all other requests are fulfilled?***
- Single threaded server does perform I/O when needed
- When a request involves I/O, the server blocks its only thread and waits for the I/O to complete
- This can result in considerable idle time

9

THREAD-LEVEL PARALLELISM EXAMPLE

- Metric – measures degree of parallelism realized by running system, by calculating average utilization:

$$TLP = \frac{\sum_{i=1}^{N} i \cdot c_i}{1 - c_0}$$

- C_i – fraction of time that i threads are executed simultaneously
- N – maximum threads that can execute at any one time
- C_0 – is the fraction of time that 0 threads are executed – *CPU is idle…*

- ***Idle threads (blocked for I/O) are not executing !***

10

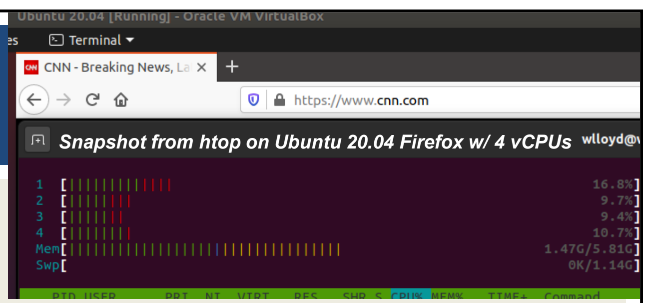
TLP EXAMPLE- 2

- Image a 4-core CPU, need to know:
  - fraction of time just 1 thread runs – load average <= 1.0
  - fraction of time 2 threads run – load average 2.0
  - fraction of time 3 threads run – load average 3.0
  - fraction of time 4 threads run – load average 4.0
- Divide by 1 minus fraction of time CPU is idle

- **Example:**
- 4-virtual CPU Ubuntu 20.04 Virtual Machine
- Firefox browser pointed at news website

11

CALCULATE TLP:



Assume htop's CPU bars report 2-second utilization averages for the computer:

- CPU 1: 16.8%
- CPU 2: 9.7%
- CPU 3: 9.4%
- CPU 4: 10.7%
- CPU idle: 53.4%

$$\frac{(1 \cdot .168)+(2 \cdot .097)+(3 \cdot .094)+(4 \cdot .107)}{1 - .534}$$

$$\frac{(.168)+(.194)+(.282)+(.428)}{.466}$$

$$\frac{1.072}{.466} = 2.3 \text{ estimated TLP}$$

$$TLP = \frac{\sum_{i=1}^{N} i \cdot c_i}{1 - c_0}$$

12

INFER TLP FROM HTOP (?)

- While this approach approximates TLP of a computer over a time-interval, it may not accurately characterize TLP of a program
- Running the application on a minimal idle machine with no other users may help to estimate TLP
- Inside a Docker container the htop CPU bars report host-level CPU utilization, and not container (application) level
- A tool is needed to isolate only the activity of an application for estimating TLP
- Container level metrics available from the cgroup virtual file system can identify CPU user and CPU kernel time for a container

- It is not clear if the CPU utilization bars in htop are averages or instantaneous statuses
- If they are time quantum averages of CPU utilization, the update interval can be adjusted in htop:
- **htop –d** <update interval>
- Bars are color coded: **kernel** **user** **I/O-wait** **SoftIRQ** **IRQ** **nice**

February 2, 2023	TCSS558: Applied Distributed Computing [Winter 2023] School of Engineering and Technology, University of Washington - Tacoma	L10.13

13

OBJECTIVES – 2/2

- Questions from 1/31
- **Assignment 0: Cloud Computing Infrastructure Tutorial**
- Assignment 1: Key/Value Store
  - Java Maven project template files posted
- Midterm Thursday February 9
  - 2nd hour - Tuesday February 7 – practice midterm questions
- Chapter 3: Processes
  - Chapter 3.3: Clients
  - Chapter 3.4: Servers

February 2, 2023	TCSS558: Applied Distributed Computing [Winter 2023] School of Engineering and Technology, University of Washington - Tacoma	L10.14

14

ASSIGNMENT 0

- ***Preparing for Assignment 0:***
  - Establish AWS Account
    - Standard account
      - Complete AWS Cloud Credits Survey and provide AWS account ID
      - Credits will be automatically loaded by Amazon into accounts
- Tasks:
  - Task 1 - Establish local Linux/Ubuntu environment
  - Task 2 –AWS account setup, obtain user credentials
  - Task 3 – Intro to: Amazon EC2 & Docker: create Dockerfile for Apache Tomcat
  - Task 4 – Create Dockerfile for haproxy
  - Task 5 – Working with Docker-Machine
  - Task 6 - Config 3 multiple server configs to load balance requests for RESTful Fibonacci web service
  - Task 7 – Test configs and submit results

15

TESTING CONNECTIVITY TO SERVER

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

16

17

OBJECTIVES – 2/2

- Questions from 1/31
- Assignment 0: Cloud Computing Infrastructure Tutorial
- **Assignment 1: Key/Value Store**
  - **Java Maven project template files posted**
- Midterm Thursday February 9
  - 2nd hour - Tuesday February 7 – practice midterm questions
- Chapter 3: Processes
  - Chapter 3.3: Clients
  - Chapter 3.4: Servers

February 2, 2023	TCSS558: Applied Distributed Computing [Winter 2023] School of Engineering and Technology, University of Washington - Tacoma	L10.18

18

ASSIGNMENT 1

- Multi-protocol TCP/UDP/RMI Key Value Store
- Implement a “GenericNode” where the application can be launched to assume the role of a client or server for a Key/Value Store data store
- Recommended in Java (11)
- Client node program interacts with server node to put, get, delete, or list items in a key/value store
- Multi-threaded or single-threaded server

OBJECTIVES – 2/2

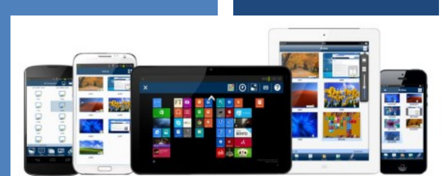
- Questions from 1/31
- Assignment 0: Cloud Computing Infrastructure Tutorial
- Assignment 1: Key/Value Store
  - Java Maven project template files posted
- **Midterm Thursday February 9**
  - **2nd hour - Tuesday February 7 – practice midterm questions**
- Chapter 3: Processes
  - Chapter 3.3: Clients
  - Chapter 3.4: Servers

OBJECTIVES – 2/2

- Questions from 1/31
- Assignment 0: Cloud Computing Infrastructure Tutorial
- Assignment 1: Key/Value Store
  - Java Maven project template files posted
- **Midterm Thursday February 9**
  - **2nd hour - Tuesday February 7 – practice midterm questions**
- Chapter 3: Processes
  - **Chapter 3.3: Clients**
  - Chapter 3.4: Servers

February 2, 2023	TCSS558: Applied Distributed Computing [Winter 2023] School of Engineering and Technology, University of Washington - Tacoma	L10.21

21



CH. 3.3: CLIENTS

L10.22

22

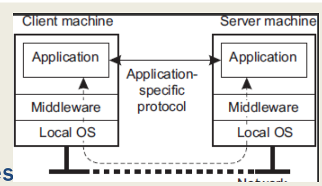
TYPES OF CLIENTS

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

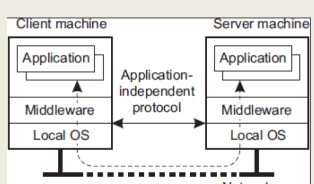
23

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



24

X WINDOWS

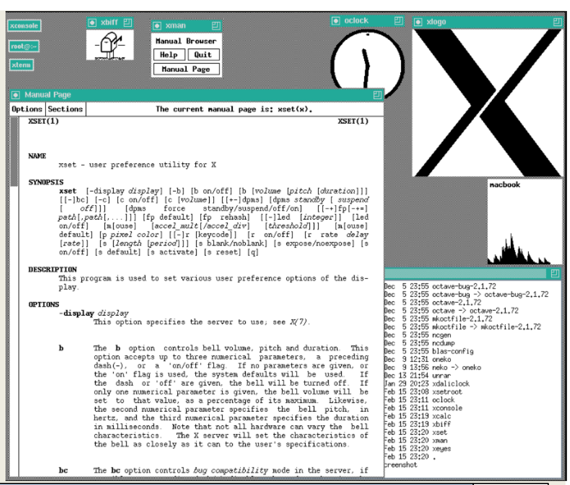
- Layered architecture to transport UI over network
- Remote desktop functionality for Linux/Unix systems
- X kernel acts as a server
  - Provides the X protocol: application level protocol
  - Xlib instances (client applications) exchange data and events with X kernels (servers)
  - Clients and servers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

February 2, 2023	TCSS558: Applied Distributed Computing [Winter 2023] School of Engineering and Technology, University of Washington - Tacoma	L10.25

25

X WINDOWS - 2

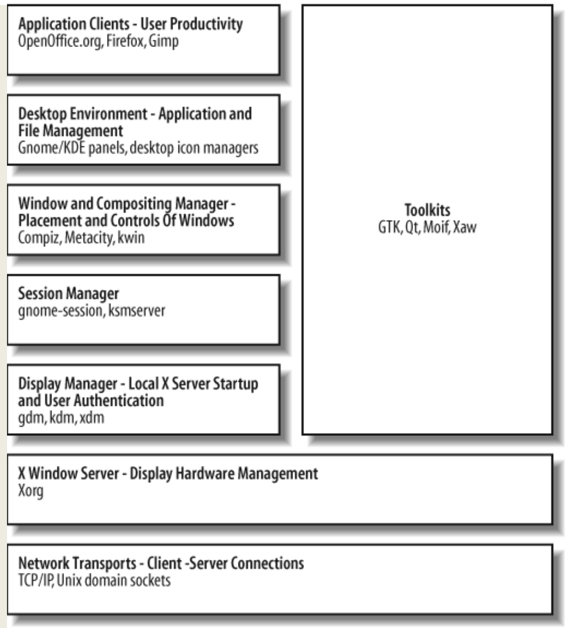
- Window manager:
  - Application running atop of X-windows which provides flair
  - Many variants
  - Without X windows is quite bland



February 2, 2023	TCSS558: Applied Distributed Computing [Winter 2023] School of Engineering and Technology, University of Washington - Tacoma	L10.26

26

- **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 2, 2023 | TCSS558: Applied Distributed Computing [Winter 2023] School of Engineering and Technology, University of Washington - Tacoma | L10.27

27

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 2, 2023 | TCSS558: Applied Distributed Computing [Winter 2023] School of Engineering and Technology, University of Washington - Tacoma | L10.28

28

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

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


EXAMPLE: VNC SERVER – UBUNTU 18.04

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

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

VNC SERVER - UBUNTU 20.04 - GNOME

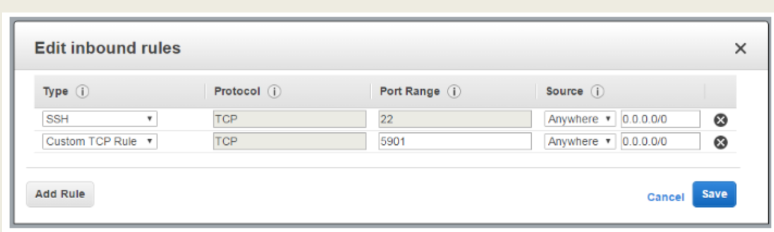
```
# install vnc server
sudo apt install tigervnc-standalone-server
Sudo apt install ubuntu-gnome-desktop
vncserver :1              # creates a config file
vncserver -kill :1        # stop server
vi ~/.vnc/xstartup        # edit config file

#!/bin/sh
# Start Gnome 3 Desktop
[ -x /etc/vnc/xstartup ] && exec /etc/vnc/xstartup
[ -r $HOME/.Xresources ] && xrdb $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
```


31

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:



32

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

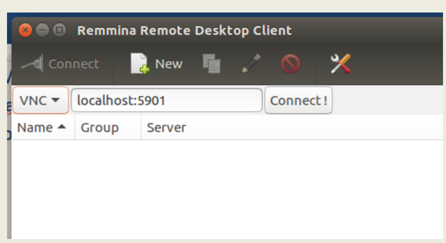
```
ssh -i <ssh-keyfile> -L 5901:127.0.0.1:5901 -N
-f -l <username> <EC2-instance ip_address>
```

- For example:

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

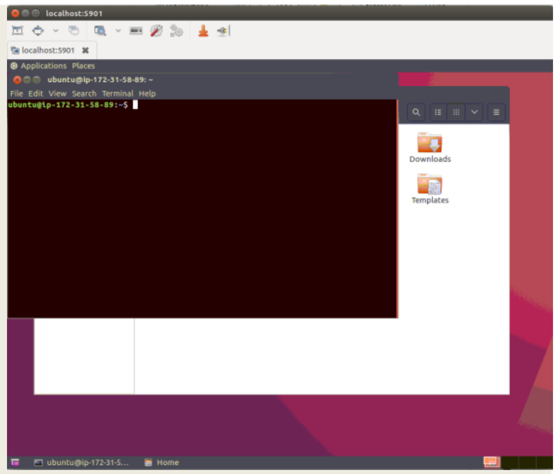

EXAMPLE: VNC CLIENT - 2

- 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"



REMOTE COMPUTER IN THE CLOUD

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



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

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 adverselt 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 2, 2023	TCSS558: Applied Distributed Computing [Winter 2023] School of Engineering and Technology, University of Washington - Tacoma	L10.37

37

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

- **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 2, 2023	TCSS558: Applied Distributed Computing [Winter 2023] School of Engineering and Technology, University of Washington - Tacoma	L10.38

38

TRADEOFFS: ABSTRACTION OF REMOTE DISPLAY PROTOCOLS

- Tradeoff space: abstraction level of remote display protocols

TRADEOFFS: ABSTRACTION OF REMOTE DISPLAY PROTOCOLS

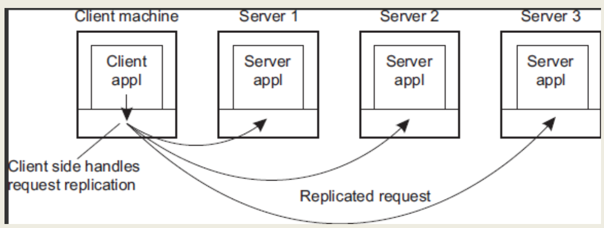
- Tradeoff space: abstraction level of remote display protocols

- Generic – no app context
- Graphics data
- Higher network bandwidth
- Fewer colors
- Utilize graphics compression
- Thin client

- Application context is available
- UI data/operations
- Lower network bandwidth
- More colors
- Thick client

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



41

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

42

WE WILL RETURN AT 2:44PM

43

OBJECTIVES – 2/2

- Questions from 1/31
- Assignment 0: Cloud Computing Infrastructure Tutorial
- Assignment 1: Key/Value Store
  - Java Maven project template files posted
- **Midterm Thursday February 9**
  - **2^{nd} hour - Tuesday February 7 – practice midterm questions**
- Chapter 3: Processes
  - Chapter 3.3: Clients
  - **Chapter 3.4: Servers**

February 2, 2023	TCSS558: Applied Distributed Computing [Winter 2023] School of Engineering and Technology, University of Washington - Tacoma	L10.44

44

CH. 3.4: SERVERS

L10.45

45

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 2, 2023	TCSS558: Applied Distributed Computing [Winter 2023] School of Engineering and Technology, University of Washington - Tacoma	L10.46

46

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

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

Common Ports

packetlife.net

TCP/UDP Port Numbers

Port	Service	Port	Service	Port	Service	Port	Service
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

49

TYPES OF SERVERS

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

50

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)

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`

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

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

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*

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

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

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?**

EJB – ENTERPRISE JAVA BEANS

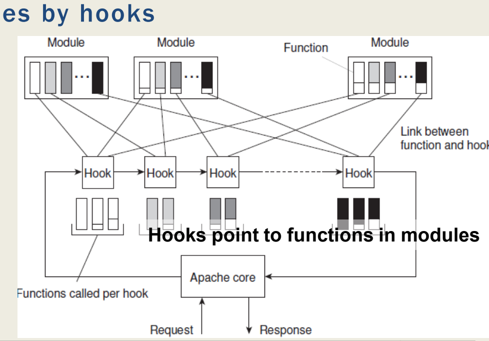
- EJB- specialized Java object hosted by a EJB web container
- 4 types: stateless, stateful, entity, and message-driven beans
- Provides "middleware" standard (framework) for implementing back-ends of enterprise applications
- EJB web application containers integrate support for:
  - Transaction processing
  - 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 2, 2023	TCSS558: Applied Distributed Computing [Winter 2023] School of Engineering and Technology, University of Washington - Tacoma	L10.59

59

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

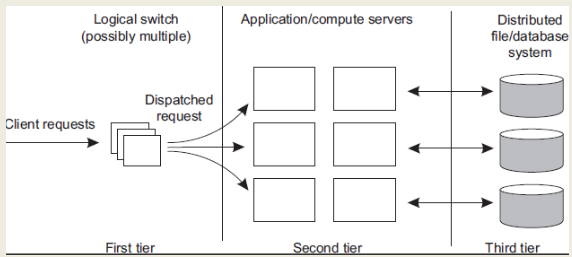


February 2, 2023	TCSS558: Applied Distributed Computing [Winter 2023] School of Engineering and Technology, University of Washington - Tacoma	L10.60

60

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



61

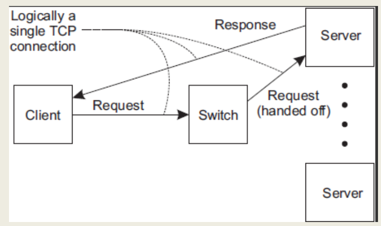
LAN REQUEST DISPATCHING

- 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 of connection to a selected server

62

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



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?

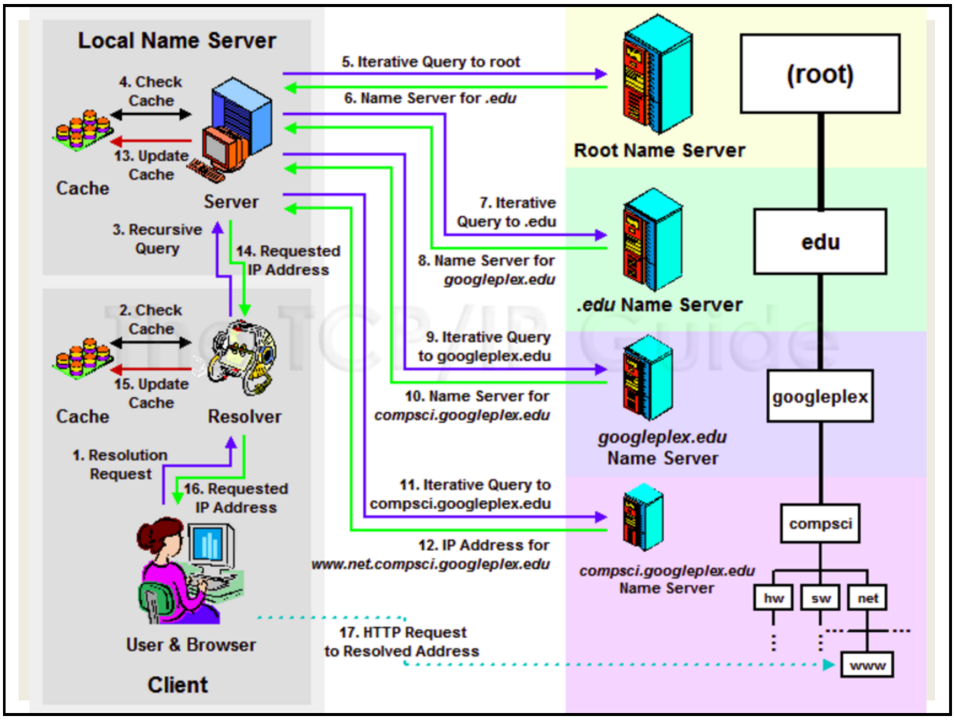
WAN REQUEST DISPATCHING

- 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>
```


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*



67

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

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 2, 2023	TCSS558: Applied Distributed Computing [Winter 2023] School of Engineering and Technology, University of Washington - Tacoma	L10.69

69

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 server in WA: ~3.63x

<u>WA client</u>: local-google 22.458ms to VA-google 81.637ms

Latency to ping WA server in VA: ~48.7x

<u>VA client</u>: 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 2, 2023	TCSS558: Applied Distributed Computing [Winter 2023] School of Engineering and Technology, University of Washington - Tacoma	L10.70

70

QUESTIONS

71