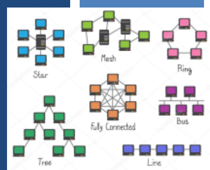
TCSS 558: APPLIED DISTRIBUTED COMPUTING

Processes: Clients & Servers

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



1

OBJECTIVES – 2/7

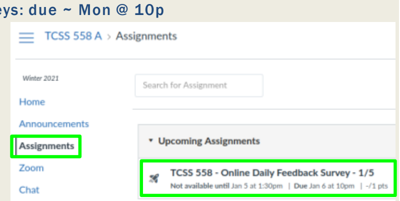
- Questions from 2/2
- Assignment 0: Cloud Computing Infrastructure Tutorial
- Assignment 1: Key/Value Store
  - Java Maven project template files posted
- Midterm Thursday February 9
- Chapter 3: Processes
  - Chapter 3.4: Servers
- Midterm Thursday February 9
  - 2nd hour – practice midterm questions

February 7, 2023 | TCSS558: Applied Distributed Computing [Winter 2023] School of Engineering and Technology, University of Washington - Tacoma | L11.2

2

ONLINE DAILY FEEDBACK SURVEY

- Daily Feedback Quiz in Canvas – Available After Each Class
- Extra credit available for completing surveys ***ON TIME***
- Tuesday surveys: due by ~ Wed @ 10p
- Thursday surveys: due ~ Mon @ 10p



February 7, 2023 | TCSS558: Applied Distributed Computing [Winter 2023] School of Engineering and Technology, University of Washington - Tacoma | L11.3

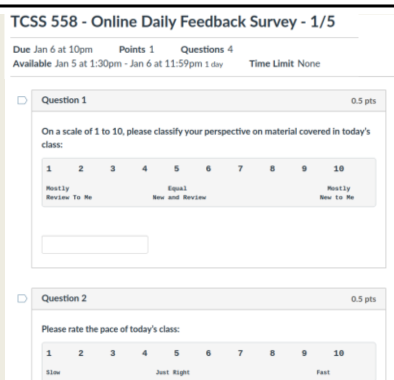
3

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 7, 2023 | TCSS558: Applied Distributed Computing [Winter 2023] School of Engineering and Technology, University of Washington - Tacoma | L11.4

4



February 7, 2023 | TCSS558: Applied Distributed Computing [Winter 2023] School of Engineering and Technology, University of Washington - Tacoma | L11.5

5

MATERIAL / PACE

- Please classify your perspective on material covered in today's class (34 respondents):
- 1-mostly review, 5-equal new/review, 10-mostly new
- **Average – 6.66** (↓ - *previous 6.02*)
- Please rate the pace of today's class:
- 1-slow, 5-just right, 10-fast
- **Average – 5.85** (↑ - *previous 5.67*)

February 7, 2023 | TCSS558: Applied Distributed Computing [Winter 2023] School of Engineering and Technology, University of Washington - Tacoma | L11.6

6

FEEDBACK FROM 2/2

7

OBJECTIVES – 2/7

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

8

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

9

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

10

OBJECTIVES – 2/7

- Questions from 2/2
- 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 3.4: Servers
- Midterm Thursday February 9
  - 2nd hour – practice midterm questions

11

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

12

OBJECTIVES – 2/7

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

13

MIDTERM

- **TCSS 558 B**
- In class in MLG 311
- Designed to take less than 2 hours
- Combination of multiple choice, true/false, fill in the blank, short answer questions
- Inclusive of content through chapter 3.4 – Servers
- 5 page of notes – double-sided (submit w/ test)
- No note sharing
- No smart phones, laptop, books
- No wi-fi enabled devices
- **TCSS 558 C**
- Details via sent Canvas announcement

14

OBJECTIVES – 2/7

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

15

16

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

17

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, then immediately wait for next incoming request
- Multithreaded servers are concurrent servers
  - E.g. Apache Tomcat
- ***Alternative to threads***: fork new process for each incoming request

18

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

19

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

20

TYPES OF SERVERS

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

21

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)

22

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`

23

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

24

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

25

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*

26

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

27

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

28

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?**

29

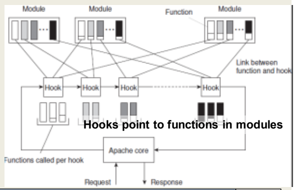
EJB – ENTERPRISE JAVA BEANS

- 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

30

APACHE WEB SERVER

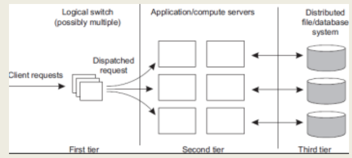
- Highly configurable, extensible, platform independent
- Supports TCP HTTP protocol communication
- Uses hooks – placeholders for group of functions
- Requests processed in phases by hooks
- Many hooks:
  - Translate a URL
  - Write info to log
  - Check client ID
  - Check access rights
- Hooks processed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



31

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



32

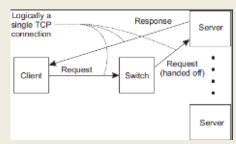
LAN REQUEST DISPATCHING

- 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

33

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



34

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?

35

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>
```

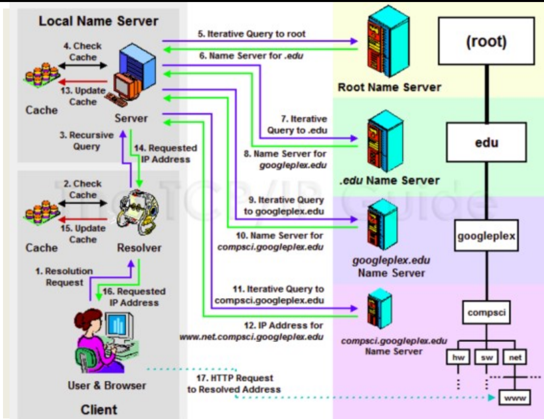

36

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 7, 2023 | TCSS558: Applied Distributed Computing [Winter 2023] School of Engineering and Technology, University of Washington - Tacoma | L11.37

37



38

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 7, 2023 | TCSS558: Applied Distributed Computing [Winter 2023] School of Engineering and Technology, University of Washington - Tacoma | L11.39

39

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 7, 2023 | TCSS558: Applied Distributed Computing [Winter 2023] School of Engineering and Technology, University of Washington - Tacoma | L11.40

40

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
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 7, 2023 | TCSS558: Applied Distributed Computing [Winter 2023] School of Engineering and Technology, University of Washington - Tacoma | L11.41

41

WE WILL RETURN AT 2:40PM

42

OBJECTIVES – 2/7

- Questions from 2/2
- 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 3.4: Servers
- Midterm Thursday February 9
  - **2nd hour – practice midterm questions**

February 7, 2023 | TCSS558: Applied Distributed Computing [Winter 2023] School of Engineering and Technology, University of Washington - Tacoma | L11.43

43

PRACTICE MIDTERM

- Envisioned as a class activity
- Form groups of 1-3 in class or via Zoom breakout rooms
- Each group works on 2 assigned questions
- Group submits PDF solution by end of day today
- Instructor shares solutions provided from the class submission via Canvas by Wednesday morning
  - Subject to updates for late submissions
- Outcome:
  All students have access to solutions for review and practice
- Follow link:
  https://tinyurl.com/24fckac7

February 7, 2023 | TCSS558: Applied Distributed Computing [Winter 2023] School of Engineering and Technology, University of Washington - Tacoma | L11.44

44

QUESTIONS

February 7, 2023 | TCSS558: Applied Distributed Computing [Winter 2023] School of Engineering and Technology, University of Washington - Tacoma | L11.45

45