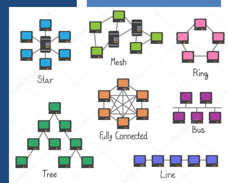
TCSS 558: APPLIED DISTRIBUTED COMPUTING

Types of Distributed Systems

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



OBJECTIVES – 1/12

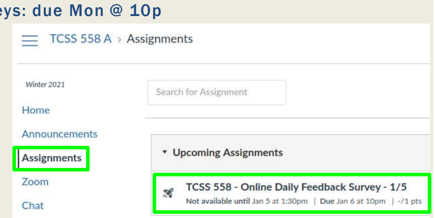
- Questions from 1/7
- Activity: Design goals of distributed systems
- Chapter 1.3 – Types of distributed systems
  - HPC, Cluster, Grid, Cloud
  - Distributed information systems
    - Transactions
    - Application Integration: Shared files, DBs, RPC, RMI, Message-oriented middleware
  - Pervasive Systems
    - Ubiquitous computing systems
    - Mobile systems
    - Sensor networks

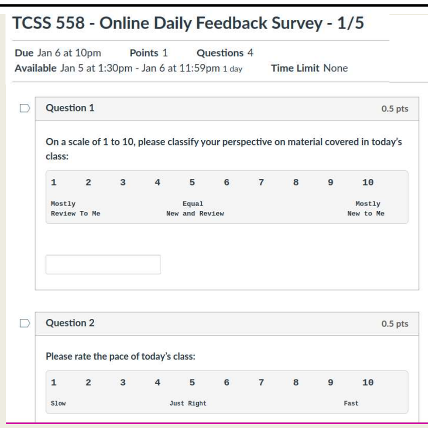
TCSS 558 OFFICE HOURS – WINTER 2021

- Fridays 11:30a - 12:30p via Zoom
  - ZOOM link shared weekly via Canvas announcements
- Tuesdays 3:30p after class
- Thursdays 3:30p after class
  - Same ZOOM link as class
- By email appointment: wlloyd@uw.edu
  - Zoom link sent by email

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



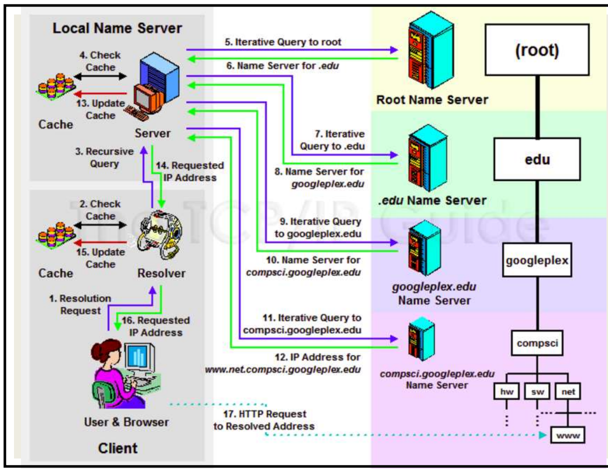


MATERIAL / PACE

- Please classify your perspective on material covered in today's class (23 respondents):
- 1-mostly review, 5-equal new/review, 10-mostly new
- **Average – 6.92** (↓ - *previous 7.48*)
- Please rate the pace of today's class:
- 1-slow, 5-just right, 10-fast
- **Average – 5.46** (↓ - *previous 5.83*)

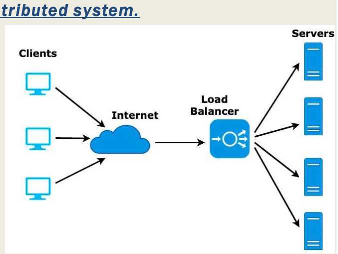
FEEDBACK FROM 1/7

- ***I'm still a bit confused on some approaches to scaling such as DNS and WWW, how they could increase in scale in terms of distributed systems.***
- DNS is covered in Chapter 3.4 on Servers
- The idea is that there is a hierarchy of servers that resolve the translation of a hostname to a numerical address and visa-versa

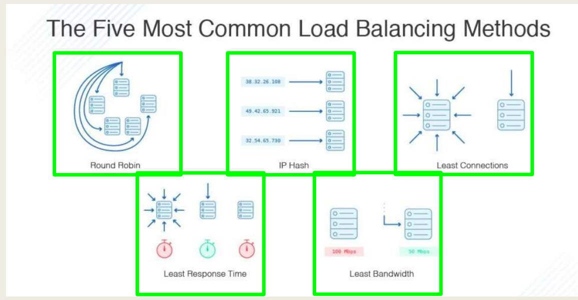


FEEDBACK - 2

- ***One concept we went over was Load Balancer (or proxy server), its unclear to me how Load Balancers distribute user requests to nodes of a distributed system.***
- Client request to the server's IP/Hostname routes the load balancer
- Load balancer decides where to send the client's request based on capacity (how busy?), or location of user's data



METHODS LOAD BALANCERS USE TO ROUTE REQUESTS



FEEDBACK - 3

- ***I wonder if parallel computing is just one kind of distributed computing - that is a means to distribute requests to different machines or systems to enhance the computing efficiency***
- ***>>> Distributed computing is just one type of parallel computing***
- When accomplishing parallel computing with distributed computing, the question is: ***how fast is the interconnect?***
- Map-Reduce: distribute tasks to nodes that operate over different data so they can run in parallel with no synchronization
  - *This is called embarrassingly parallel*
  - *Not applicable for some tasks*
  - There is no shared memory
  - Many tasks run in isolation at the same time
- Parallel computing also occurs with shared memory systems:
  - On the same computer
  - On a distributed system using special libraries to share memory and high speed inter-connects (e.g. networks)
  - On a super computers with interconnected memory

FEEDBACK - 4

- ***I think I still get confused about the openness part***
- Design software components of the distributed so that they are easily reusable in other systems
- Software should be:
- **Interoperable**: work with other components
- **Portable**: capable of being extracted out of the application and reused
- **Extensible**: easy to add new features/capabilities over time

SURVEY LINKS

AT:
http://faculty.washington.edu/wlloyd/courses/tcss558/announcements.html

TCSS 558:
Applied Distributed Computing
W UNIVERSITY of WASHINGTON | TACOMA

[ANNOUNCEMENTS] | Syllabus | Grading | Schedule | Assignments | Home |

Course Announcments

1. Please check the SCHEDULE page for information related to the posting and due dates of the a
2. Please complete the online course demographics survey: [HERE]
3. Please complete the AWS Cloud Credits survey: [HERE]

January 12, 2021 | TCSS558: Applied Distributed Computing [Winter 2021] School of Engineering and Technology, University of Washington - Tacoma | L3.13

OBJECTIVES – 1/12

- Questions from 1/7
- Activity: Design goals of distributed systems
- Chapter 1.3 – Types of distributed systems
  - HPC, Cluster, Grid, Cloud
  - Distributed information systems
    - Transactions
    - Application Integration: Shared files, DBs, RPC, RMI, Message-oriented middleware
  - Pervasive Systems
    - Ubiquitous computing systems
    - Mobile systems
    - Sensor networks

January 12, 2021 | TCSS558: Applied Distributed Computing [Winter 2021] School of Engineering and Technology, University of Washington - Tacoma | L3.14

CLASS ACTIVITY 1

- We will form groups of ~2-3 and enter breakout rooms
- Each group will complete a Google Doc worksheet
- Add names to Google Doc as they appear in Canvas
- Once completed, **one person** submits a PDF of the Google Doc to Canvas
- Instructor will score all group members based on the uploaded PDF file
- To get started:
  - Log into your UW Google Account
  - Link to shared Google Drive
  - Follow link:

    https://tinyurl.com/y5nmmro9

October 7, 2020 | TCSS562: Software Engineering for Cloud Computing [Fall 2020] School of Engineering and Technology, University of Washington - Tacoma | L3.15

OBJECTIVES – 1/12

- Questions from 1/7
- Activity: Design goals of distributed systems
- Chapter 1.3 – Types of distributed systems
  - HPC, Cluster, Grid, Cloud
  - Distributed information systems
    - Transactions
    - Application Integration: Shared files, DBs, RPC, RMI, Message-oriented middleware
  - Pervasive Systems
    - Ubiquitous computing systems
    - Mobile systems
    - Sensor networks

January 12, 2021 | TCSS558: Applied Distributed Computing [Winter 2021] School of Engineering and Technology, University of Washington - Tacoma | L3.16

WE WILL RETURN AT 2:35PM

OBJECTIVES – 1/12

- Questions from 1/7
- Activity: Design goals of distributed systems
- Chapter 1.3 – Types of distributed systems
  - HPC, Cluster, Grid, Cloud
  - Distributed information systems
    - Transactions
    - Application Integration: Shared files, DBs, RPC, RMI, Message-oriented middleware
  - Pervasive Systems
    - Ubiquitous computing systems
    - Mobile systems
    - Sensor networks

January 12, 2021 | TCSS558: Applied Distributed Computing [Winter 2021] School of Engineering and Technology, University of Washington - Tacoma | L3.18

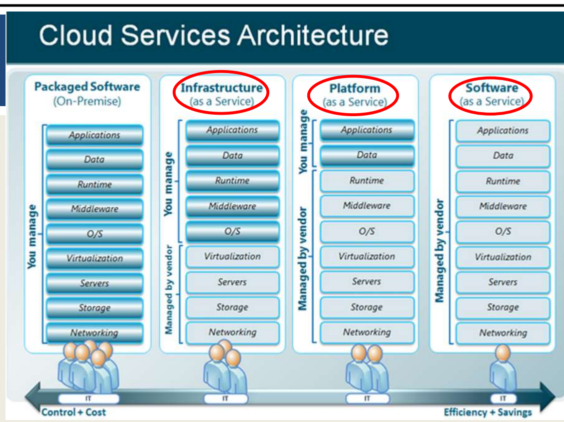
OBJECTIVES – 1/12

- Questions from 1/7
- Activity: Design goals of distributed systems
- Chapter 1.3 – Types of distributed systems
  - HPC, Cluster, Grid, Cloud
  - Distributed information systems
    - Transactions
    - Application Integration: Shared files, DBs, RPC, RMI, Message-oriented middleware
  - Pervasive Systems
    - Ubiquitous computing systems
    - Mobile systems
    - Sensor networks

TECHNOLOGY INNOVATIONS LEADING TO CLOUD COMPUTING

- Super computers
  - Huge multiprocessor system which shares RAM
  - Technically "not distributed"
  - Hardware all in one location
- High performance distributed computing
  - Cluster computing
  - Grid computing
  - Cloud computing
  - Virtualization
  - Others

Cloud Services Architecture



PUBLIC CLOUD COMPUTING

- Offers computing, storage, communication at ¢ per hour
- No premium to scale:

```
        1000 computers   @    1 hour
=        1 computer      @ 1000 hours
```

- Illusion of infinite scalability to cloud user
- As many computers as you can afford
- Leading examples:
  Amazon Web Services, Google App Engine, Microsoft Azure
- Amazon runs its own e-commerce on AWS!
- Billing models are becoming increasingly granular
  - By the minute, second, tenth of a second
  - Example: AWS Lambda $0.0000002 per request
    $0.000000208 to rent 128MB / 100-ms

PUBLIC CLOUD COMPUTING - 2

m4.large ec2 virtual machine:
2 vCPU cores, 8 GB RAM, Intel Xeon E5-2666 v3
10¢ an hour, 24 hrs/day,
30 days/month → $72.00/month
on-demand EC2 instance

AWS Lambda Function-as-a-Service (FaaS) w/o free tier:
2 vCPU cores, 3GB RAM, Intel Xeon E5-2666 v3 (maybe?)
as 2,592,000 x 1-sec service calls
24 hrs/day, 30 days/month:
$130.14 (8GB = $347.04)

PAAS SERVICES IMPLEMENTATION

- PaaS services often built atop of IaaS
  - Amazon RDS, Heroku, Amazon Elasticache
- Scalability
  - VM resources can support fluctuations in demand
- Dependability.
  - PaaS services built on highly available IaaS resources

OBJECTIVES – 1/12

- **Questions from 1/7**
- Activity: Design goals of distributed systems
- Chapter 1.3 – Types of distributed systems
  - HPC, Cluster, Grid, Cloud
  - Distributed information systems
    - Transactions
    - Application Integration: Shared files, DBs, RPC, RMI, Message-oriented middleware
  - Pervasive Systems
    - Ubiquitous computing systems
    - Mobile systems
    - Sensor networks

DISTRIBUTED INFORMATION SYSTEMS

- **Enterprise-wide** integrated applications
  - Organizations confronted with too many applications
  - Interoperability among applications was difficult
  - Led to many middleware-based solutions
- Key concepts
  - Component based architectures - database components, processing components
  - **Distributed transaction** – Client wraps requests together, sends as single aggregated request
  - Atomic: ***all*** or ***none*** of the individual requests should be executed
- Different systems define different ***action*** primitives
  - Components of the atomic transaction
  - Examples: send, receive, forward, READ, WRITE, etc.

DISTRIBUTED INFORMATION SYSTEMS - 2

- Transaction primitives

Primitive	Description
BEGIN_TRANSACTION	Mark the start of a transaction
END_TRANSACTION	Terminate the transaction and try to commit
ABORT_TRANSACTION	Kill the transaction and restore the old values
READ	Read data from a file, a table, or otherwise
WRITE	Write data to a file, a table, or otherwise

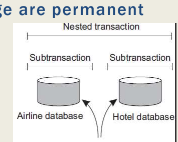
- Transactions are **all-or-nothing**
  - All operations are executed
  - None are executed

OBJECTIVES – 1/12

- **Questions from 1/7**
- Activity: Design goals of distributed systems
- Chapter 1.3 – Types of distributed systems
  - HPC, Cluster, Grid, Cloud
  - Distributed information systems
    - Transactions
    - Application Integration: Shared files, DBs, RPC, RMI, Message-oriented middleware
  - Pervasive Systems
    - Ubiquitous computing systems
    - Mobile systems
    - Sensor networks

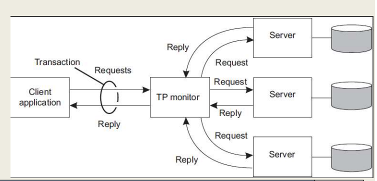
TRANSACTIONS: ACID PROPERTIES

- **A**tomic: The transaction occurs indivisibly
- **C**onsistent: Transaction does not create variant states across nodes during slow updates (e.g. system variants)
  - Replicas remain constant until all updated
  - Two phase commit: data pushed first, then the commit
- **I**solated: Transactions do not interfere with each other
- **D**urable: Once a transaction commits, change are permanent
- Nested transaction: transaction constructed with many sub-transactions
- Follows a logical division of work
- Must support "rollback" of sub-transactions



TRANSACTION PROCESSING MONITOR

- Allow an application to access multiple DBs via a transactional programming model
- **TP monitor**: coordinates commitment of sub-transactions using a distributed commit protocol (Ch. 8)
- Save application complexity from having to coordinate



OBJECTIVES – 1/12

- **Questions from 1/7**
- Activity: Design goals of distributed systems
- Chapter 1.3 – Types of distributed systems
  - HPC, Cluster, Grid, Cloud
  - Distributed information systems
    - Transactions
    - Application Integration: Shared files, DBs, RPC, RMI, Message-oriented middleware
  - Pervasive Systems
    - Ubiquitous computing systems
    - Mobile systems
    - Sensor networks

ENTERPRISE APPLICATION INTEGRATION

- Support application components direct communication with each other, not via databases
- **Communication mechanisms**:
- **Remote procedure call** (RPC)
  - Local procedure call packaged as a message and sent to server
  - Supports distribution of function call processing
- **Remote method invocation** (RMI)
  - Operates on objects instead of functions
- RPC and RMI – led to tight coupling
- Client and server endpoints must be up and running
- Interfaces coupled to specific languages and not ***interoperable***
- This led to evolution of: **Message-oriented middleware** (MOM)

MESSAGE-ORIENTED MIDDLEWARE

- Publish and subscribe systems:
  - Rabbit MQ, Apache Kafka, AWS SQS/SNS
- Reduces tight coupling of RPC/RMI
- Applications indicate interest for specific type(s) of messages by sending requests to logical contact points
- Communication middleware delivers messages to subscribing applications

CHALLENGES WITH VARIOUS APPLICATION INTEGRATION METHODS

- Integration via shared data files and transfers
  - Shared data files (e.g. XML)
  - Leads to file management challenges (concurrent updates, etc.)
- Shared database
  - Centralized DB, transactions to coordinate changes among users
  - Common data schema required – can be challenging to derive
  - For many reads and updates, shared DB becomes bottleneck (***limited scalability)***
- Remote procedure call – app A executes on and against app B data. App A lacks direct access to app B data.
- Messaging middleware - ensures nodes temporarily offline later on, can receive messages

OBJECTIVES – 1/12

- **Questions from 1/7**
- Activity: Design goals of distributed systems
- Chapter 1.3 – Types of distributed systems
  - HPC, Cluster, Grid, Cloud
  - Distributed information systems
    - Transactions
    - Application Integration: Shared files, DBs, RPC, RMI, Message-oriented middleware
  - Pervasive Systems
    - Ubiquitous computing systems
    - Mobile systems
    - Sensor networks

PERVASIVE SYSTEMS

- Existing everywhere, widely adopted…
- Combine current network technologies, wireless computing, voice recognition, internet capabilities and AI to create an environment where connectivity of devices is embedded, unobtrusive, and always available
- Many sensors infer various aspects of a user's behavior
  - Myriad of actuators to collect information, provide feedback
- **TYPES OF PERVASIVE SYSTEMS:**
  - Ubiquitous computing systems
  - Mobile systems
  - Sensor networks

OBJECTIVES – 1/12

- **Questions from 1/7**
- Activity: Design goals of distributed systems
- Chapter 1.3 – Types of distributed systems
  - HPC, Cluster, Grid, Cloud
  - Distributed information systems
    - Transactions
    - Application Integration: Shared files, DBs, RPC, RMI, Message-oriented middleware
  - Pervasive Systems
    - Ubiquitous computing systems
    - Mobile systems
    - Sensor networks

PERVASIVE SYSTEM TYPE: UBIQUITOUS COMPUTING SYSTEMS

- Pervasive and continuously present
- Goal: embed processors everywhere (day-to-day objects) enabling them to communicate information
- Requirements for a ubiquitous computing system:
  - **Distribution** – devices are networked, distributed, and accessible transparently
  - **Interaction** – unobtrusive (low-key) between users and devices
  - **Context awareness** – optimizes interaction
  - **Autonomy** – devices operate autonomously, self-managed
  - **Intelligence** – system can handle wide range of dynamic actions and interactions

UBIQUITOUS COMPUTING SYSTEM EXAMPLES

- Apple Watch
- Amazon Echo Speaker
- Amazon EchoDot (single speaker design)
- Fitbit
- Electronic Toll Systems
- Smart Traffic Lights
- Self Driving Cars
- Home Automation

UBIQUITOUS COMPUTING SYSTEM EXAMPLE

- **Domestic ubiquitous computing environment example:**
- Interconnect lighting and environmental controls with personal biometric monitors woven into clothing so that illumination and heating conditions in a room might be modulated, continuously and imperceptibly
- IoT technology helps enable ubiquitous computing

OBJECTIVES – 1/12

- **Questions from 1/7**
- Activity: Design goals of distributed systems
- Chapter 1.3 – Types of distributed systems
  - HPC, Cluster, Grid, Cloud
  - Distributed information systems
    - Transactions
    - Application Integration: Shared files, DBs, RPC, RMI, Message-oriented middleware
  - Pervasive Systems
    - Ubiquitous computing systems
    - Mobile systems
    - Sensor networks

PERVASIVE SYSTEM TYPE: MOBILE SYSTEMS

- Emphasis on mobile devices, e.g. smartphones, tablet computers
- New devices: remote controls, pagers, active badges, car equipment, various GPS-enabled devices,
- Devices move: ***where is the device?***
- Changing location: leverage **m**obile **a**dhoc **net**work (MANET)
- MANET is an ad hoc network that can change locations and configure itself on the fly. MANETs are mobile, they use wireless connections to connect to various networks.
- VANET (**V**ehicular **A**d Hoc **Net**work), is a type of MANET that allows vehicles to communicate with roadside equipment.

OBJECTIVES – 1/12

- **Questions from 1/7**
- Activity: Design goals of distributed systems
- Chapter 1.3 – Types of distributed systems
  - HPC, Cluster, Grid, Cloud
  - Distributed information systems
    - Transactions
    - Application Integration: Shared files, DBs, RPC, RMI, Message-oriented middleware
  - Pervasive Systems
    - Ubiquitous computing systems
    - Mobile systems
    - Sensor networks

PERVASIVE SYSTEM TYPE: SENSOR NETWORKS

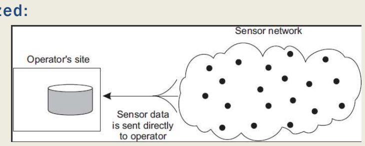
- Tens, to hundreds, to thousands of small nodes
- Simple: small memory/compute/communication capacity
- Wireless, battery powered (or battery-less)
- Limited: restricted communication, constrained power
- Equipped with sensing devices
- Some can act as actuators (control systems)
  - Example: enable sprinklers upon fire detection
- Sensor nodes organized in neighborhoods
- Scope of communication:
  - Node – neighborhood – system-wide

PERVASIVE SYSTEM TYPE: SENSOR NETWORKS - 2

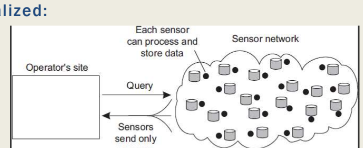
- Collaborate to process sensor data in app-specific manner
- Provide mix of data collection and processing
- **Nodes may implement a distributed database**
- Database organization: centralized to decentralized
- In network processing: forward query to all sensor nodes along a tree to aggregate results and propagate to root
- Is aggregation simply data collection?
- Are all nodes homogeneous?
- Are all network links homogeneous?
- How do we setup a tree when nodes have heterogeneous power and network connection quality?

CENTRALIZED VS. DECENTRALIZED DATA STORAGE

- Centralized:



- Decentralized:



WHO AGGREGATES AND STORES DATA?

- Consider the **tradeoff space** for:
  - sensor network data storage and processing

Centralized ⟵ ⟶ **Decentralized**

Centralized:
- Single point-of-failure
- No node coordination
- No node processing or storage
- "Dumb" nodes
- Less expensive node
- More network traffic

Decentralized:
- Nodes require high compute power
- "Smart" nodes
- Expensive nodes
- Less network traffic

SENSOR NETWORKS - 3

- What are some unique requirements for sensor networks middleware?
  - Sensor networks may consist of different types of nodes with different functions
  - Nodes may often be in suspended state to save power
    - Duty cycles (1 to 30%), strict energy budgets
  - Synchronize communication with duty cycles
  - How do we manage membership when devices are offline?

TYPES OF DISTRIBUTED SYSTEMS

- HPC, Cluster, Grid, Cloud
- Distributed information systems
  - Transactions
  - Application Integration: Shared files, DBs, RPC, RMI, Message-oriented middleware
- Pervasive Systems
  - Ubiquitous computing systems
  - Mobile systems
  - Sensor networks

QUESTIONS