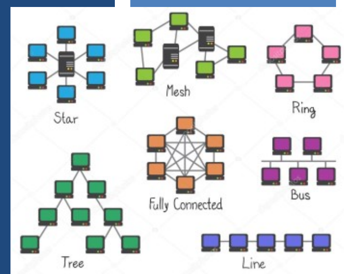
TCSS 558:
APPLIED DISTRIBUTED COMPUTING



Types of Distributed Systems

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

1

OBJECTIVES – 1/11

- Questions from 1/9
- 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 11, 2024	TCSS558: Applied Distributed Computing [Winter 2024] School of Engineering and Technology, University of Washington - Tacoma	L3.2

2

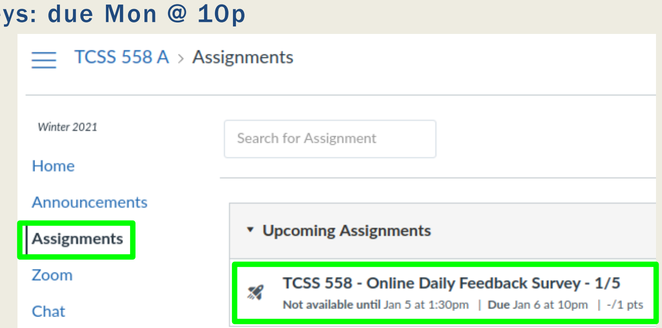
TCSS 558 OFFICE HOURS – WINTER 2024

- 26 of 32 survey responses so far – top 3 votes
  - #1: Tuesday afternoon before class (66.7%)
  - #2: Thursday afternoon before class (58.3%)
  - #3: Prefer office hours online via Zoom (41.7%)
  - Of note: Prefer office hours in-person (20.8%)
- Office hour - Tuesdays before class
  - ~2:30 – 3:30p CP 229 and Zoom
- Additional hours will be added as needed
  - In particular, we will try to add extra hours on Thursdays before class, or on Fridays via Zoom at peak times
- Also by email appointment: wlloyd@uw.edu

3

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



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

5

MATERIAL / PACE

- Please classify your perspective on material covered in today's class (33 respondents):
- 1-mostly review, 5-equal new/review, 10-mostly new
- **Average – 6.60 (↓ - *previous 6.85*)**

- Please rate the pace of today's class:
- 1-slow, 5-just right, 10-fast
- **Average – 5.16 (↓ - *previous 5.50*)**

6

FEEDBACK FROM 1/9

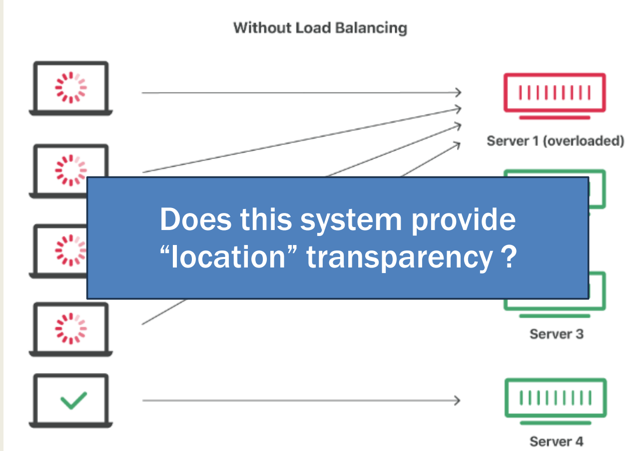
- ***I plan to read the chapter before the next class and hope that between the textbook and the lectures, things will start to make more sense.***
  - Great !
  - The focus is on Chapter 1 from the 3rd edition

- ***Load balancer / load balancing***
  - Network traffic (client requests) are distributed across multiple servers to enable the application to scale **horizontally**
  - Improve application responsiveness, availability

January 11, 2024	TCSS558: Applied Distributed Computing [Winter 2024] School of Engineering and Technology, University of Washington - Tacoma	L3.7

7

LOAD BALANCING



Does this system provide "location" transparency ?

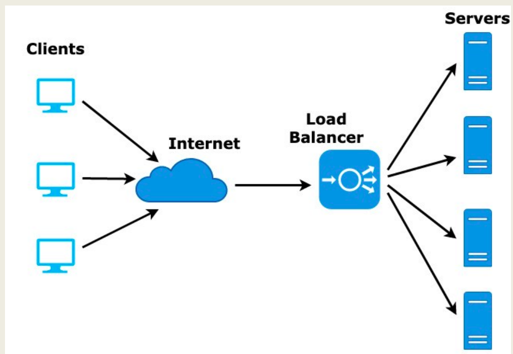
image credit: https://www.cloudflare.com/learning/performance/what-is-load-balancing/

January 11, 2024	TCSS558: Applied Distributed Computing [Winter 2024] School of Engineering and Technology, University of Washington - Tacoma	L3.8

8

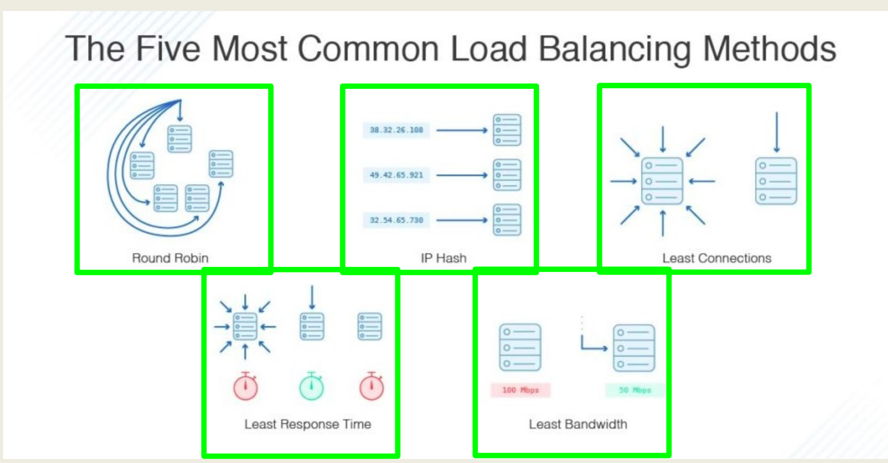
LOAD BALANCING - 2

- How do Load Balancers distribute user requests to nodes of a distributed system?
- Client request to the host routes to the load balancer
- Load balancer decides where to send the client's request based on capacity (how busy?), or location of user's data



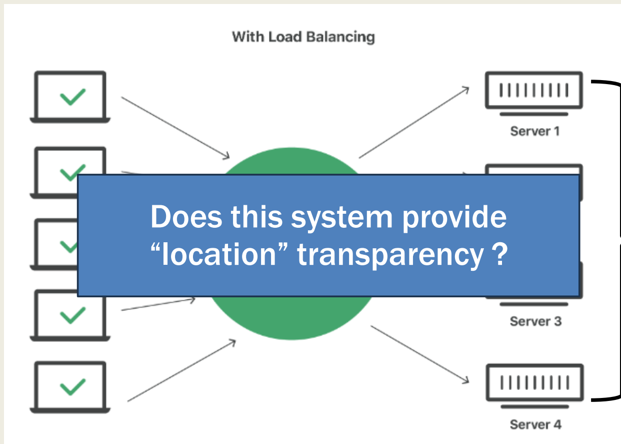
9

METHODS LOAD BALANCERS USE TO ROUTE REQUESTS



10

LOAD BALANCING - 3



Does this system provide “location” transparency ?

← == what type of scaling?

image credit: https://www.cloudflare.com/learning/performance/what-is-load-balancing/

FEEDBACK - 2

- ***Fault tolerance***
  - The ability of a system to continue functioning despite failures
  - Every computer will eventually fail
  - Redundancy is required (duplicate/extra servers)

FAULT TOLERANCE

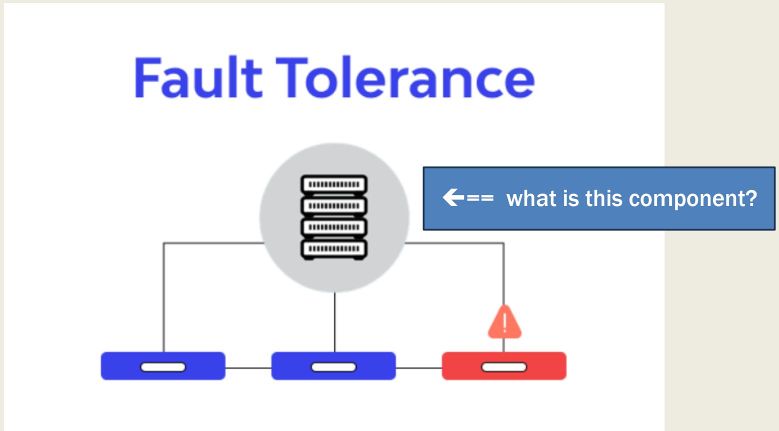
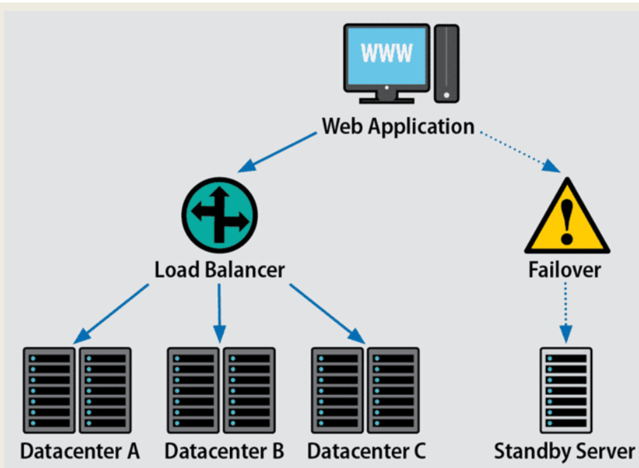


Image credit: https://www.wallarm.com/what/what-is-fault-tolerance

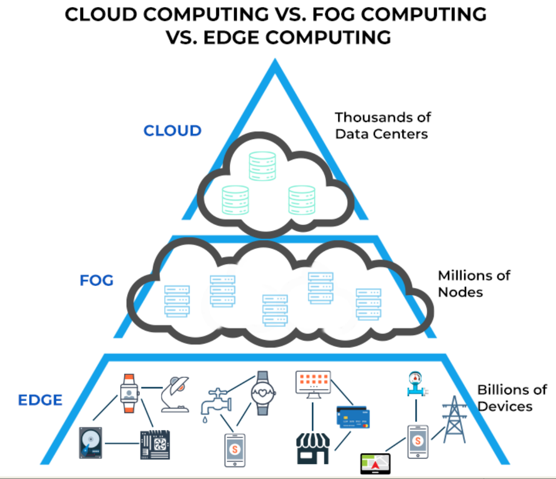
FAULT TOLERANCE - 2



EDGE / FOG / CLOUD

- In distributed systems nodes can be anywhere at the edge, in the fog, or in the cloud…

Cloud: remote servers
Fog: local servers
Edge: clients and/or mini-servers (Internet of things – IOT)



15

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]**

16

OBJECTIVES – 1/11

- Questions from 1/9
- **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

CLASS ACTIVITY 1

- We will form groups of ~2-3
  - Remote students will use Canvas breakout rooms
- Each group will complete a MS Word Doc worksheet
- Add names to top of worksheet as they appear in Canvas
- Once completed, **one person** submits a PDF of the Word Doc to Canvas
- Grader will score all group members based on the uploaded PDF file
- To get started:
  - Log into Canvas, TCSS 558 A
  - Find worksheet under Class Activity 1

WE WILL RETURN AT 5:00PM

19

OBJECTIVES – 1/11

- Questions from 1/9
- 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 11, 2024	TCSS558: Applied Distributed Computing [Winter 2024] School of Engineering and Technology, University of Washington - Tacoma	L3.20

20

OBJECTIVES – 1/11

- Questions from 1/9
- 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

21

TECHNOLOGY INNOVATIONS LEADING TO CLOUD COMPUTING

- Super computers
  - Huge multiprocessor systems with shared memory/RAM
  - Technically “not distributed”
  - Hardware all in one location
  - Initially expensive with proprietary designs
  - Traditionally supported HPC – High Performance Computing scientific applications
    - Weather forecasting
    - Molecular dynamics simulation
    - Protein modeling
  - Cost millions of dollars
  - Large systems consume MWs of electricity

22

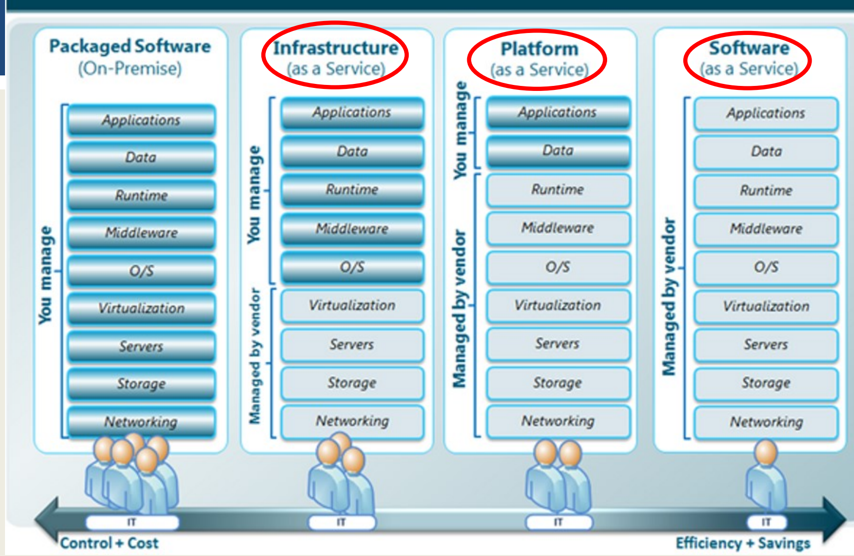
INNOVATIONS LEADING TO CLOUD COMPUTING - 2

- **Cluster computing**
  - Group of interconnected homogeneous servers
  - Design emphasizes redundancy as server components are easily interchanged to keep overall system running
- **Grid computing**
  - Distributed heterogeneous servers organized into logical pools of loosely coupled resources connected by the internet
  - geographically dispersed
  - middleware software supports workload distribution and coordination functions
- **Virtualization**
- **Others**

January 11, 2024	TCSS558: Applied Distributed Computing [Winter 2024] School of Engineering and Technology, University of Washington - Tacoma	L3.23

23

Cloud Services Architecture



January 11, 2024	TCSS 558: Applied Distributed Computing [Winter 2024] School of Engineering and Technology, University of Washington - Tacoma	L3.24

24

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 hour, minute, second, now millisecond
  - Example: AWS Lambda $0.0000002 per request (call)
    $0.0000000021 to rent 128MB / 1-ms

PUBLIC CLOUD COMPUTING - 2

- IaaS vs FaaS 1-month cost comparison - 30.4167 days/month

<u>**c4.large ec2 virtual machine (Infrastructure-as-a-Service - IaaS):**</u>
2 vCPU cores, 3.75 GB RAM, Intel Xeon E5-2666 v3
10¢ an hour, 24 hrs/day, billed by the second (60 sec min)
Cost → $73.00/month on-demand EC2 instance

<u>**AWS Lambda (Function-as-a-Service – FaaS serverless):**</u>
2 vCPU cores, 3GB RAM, Intel Xeon E5-2666 v3 (maybe?)
as 2,628,000 x 1-sec service calls
$0.00001667 GB/sec, billed by the millisecond (no min)
<u>**What is the cost ???**</u>

$131.43 (1.8x)

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

January 11, 2024 | TCSS 558: Applied Distributed Computing [Winter 2024] School of Engineering and Technology, University of Washington - Tacoma | L3.27

27

OBJECTIVES – 1/11

- Questions from 1/9
- 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 11, 2024 | TCSS558: Applied Distributed Computing [Winter 2024] School of Engineering and Technology, University of Washington - Tacoma | L3.28

28

DISTRIBUTED INFORMATION SYSTEMS

- **Enterprise-wide** integrated applications (example: UW Workday)
  - 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

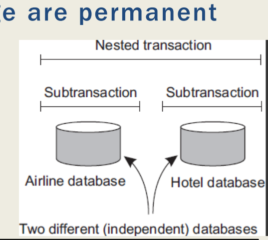
OBJECTIVES – 1/11

- Questions from 1/9
- 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

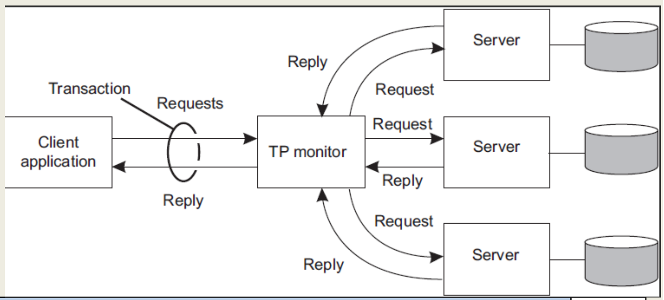
- Atomic: The transaction occurs indivisibly
- Consistent: Transaction does not create variant states across nodes during slow updates (e.g. system variants)
  - Replicas remain constant until all updated
  - Two phase commit: data pushed first, then the commit
- Isolated: Transactions do not interfere with each other
- Durable: 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)
  - Saves application complexity from having to coordinate distributed transactions



33

OBJECTIVES – 1/11

- Questions from 1/9
- 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

34

ENTERPRISE APPLICATION INTEGRATION

- Dist. Info systems 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 invocations** (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
- 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 – 1/11

- Questions from 1/9
- 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 – 1/11

- Questions from 1/9
- 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

41

UBIQUITOUS COMPUTING DEVICES EXAMPLES

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

42

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 – 1/11

- Questions from 1/9
- 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 mobile adhoc network (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 (Vehicular Ad Hoc Network), is a type of MANET that allows vehicles to communicate with roadside equipment.

January 11, 2024	TCSS558: Applied Distributed Computing [Winter 2024] School of Engineering and Technology, University of Washington - Tacoma	L3.45

45

OBJECTIVES – 1/11

- Questions from 1/9
- 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 11, 2024	TCSS558: Applied Distributed Computing [Winter 2024] School of Engineering and Technology, University of Washington - Tacoma	L3.46

46

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

January 11, 2024	TCSS558: Applied Distributed Computing [Winter 2024] School of Engineering and Technology, University of Washington - Tacoma	L3.47

47

PERVASIVE SYSTEM TYPE:
SENSOR NETWORKS - 2

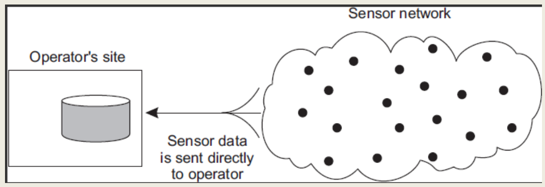
- Collaborate to process sensor data in app-specific manner
- Provide mix of data collection and processing
- **Nodes may implement a distributed database**
- Database organization: centralized to decentralized
- In network processing: forward query to all sensor nodes along a tree to aggregate results 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?

January 11, 2024	TCSS558: Applied Distributed Computing [Winter 2024] School of Engineering and Technology, University of Washington - Tacoma	L3.48

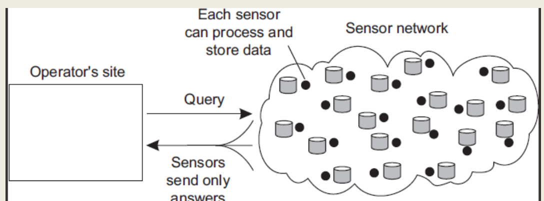
48

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**

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

- 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?

January 11, 2024	TCSS558: Applied Distributed Computing [Winter 2024] School of Engineering and Technology, University of Washington - Tacoma	L3.51

51

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 11, 2024	TCSS558: Applied Distributed Computing [Winter 2024] School of Engineering and Technology, University of Washington - Tacoma	L3.52

52

QUESTIONS

January 11, 2024

TCSS558: Applied Distributed Computing [Winter 2024]
School of Engineering and Technology, University of Washington - Tacoma

L3.53

53