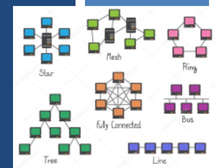
January 12, 2023 TCSS558: Applied Distributed Computing [Winter 2023] School of Engineering and Technology, University of Washington - Tacoma L4.1

1

TCSS 558: APPLIED DISTRIBUTED COMPUTING

Distributed Systems: Types and Architectures - II

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



2

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
- Coming Soon - - PREVIEW:
  - Task 0 - Establish local Linux/Ubuntu environment
  - Task 1 –AWS account setup, obtain user credentials
  - Task 2 – Intro to: Amazon EC2 & Docker: create Dockerfile for Apache Tomcat
  - Task 3 – Create Dockerfile for haproxy
  - Task 4 – Working with Docker-Machine
  - Task 5 – Submit: Results of testing alternate server configs

January 12, 2023 TCSS558: Applied Distributed Computing [Winter 2023] School of Engineering and Technology, University of Washington - Tacoma L4.3

3

OBJECTIVES – 1/12

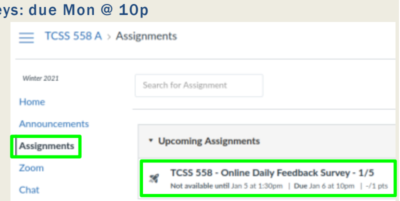
- **Questions from 1/10**
- Message Oriented Middleware
- Chapter 1.3 – Types of distributed systems
  - Pervasive Systems: Ubiquitous, Mobile, Sensor networks
- Chapter 2: Distributed System Architectures:
  - Chapter 2.1 – Architectural Styles
  - Layered
  - Object-based
    - Service oriented architecture (SOA)
  - Resource-centered architectures
    - Representational state transfer (REST)
  - Event-based
    - Publish and subscribe (Rich Site Summary RSS feeds)

January 12, 2023 TCSS558: Applied Distributed Computing [Winter 2023] School of Engineering and Technology, University of Washington - Tacoma L4.4

4

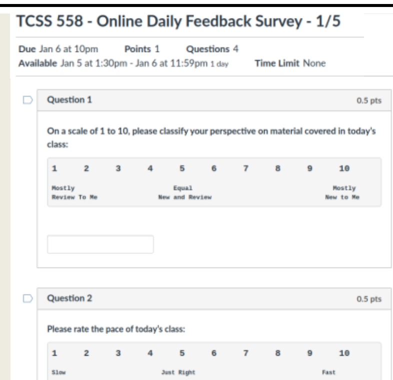
ONLINE DAILY FEEDBACK SURVEY

- 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



January 12, 2023 TCSS558: Applied Distributed Computing [Winter 2023] School of Engineering and Technology, University of Washington - Tacoma L4.5

5



January 12, 2023 TCSS558: Applied Distributed Computing [Winter 2023] School of Engineering and Technology, University of Washington - Tacoma L4.6

6

MATERIAL / PACE

- Please classify your perspective on material covered in today's class (31 respondents):
- 1-mostly review, 5-equal new/review, 10-mostly new
- **Average – 5.94** (↓ - *previous 6.12*)

- Please rate the pace of today's class:
- 1-slow, 5-just right, 10-fast
- **Average – 5.30** (↓ - *previous 5.43*)

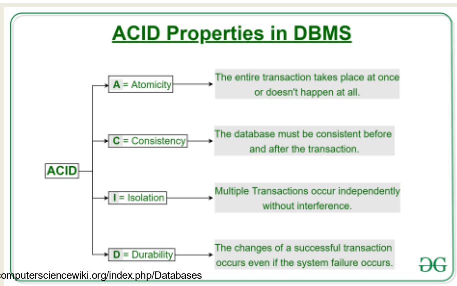
7

FEEDBACK FROM 1/10

- ***What are some examples of ACID properties?***
- ACID properties are the 4 key properties of ACID transactions
- ACID transactions guarantee that each data write (C), read (R), or modification (U/D) are:
- **Atomic**- all or no statements of a transaction (C R U D) are executed
- **Consistent**- data (copies) will be consistent. No data corruption
- **Isolated**- transactions from users don't interfere w/ each other
- **Durable**- in the event of system failure, data updates are saved
- ACID transactions have highest possible data reliability and integrity
- Without ACID transactions, on power loss, some data may be saved, but other data not – this results in an inconsistent state that is difficult and time consuming to recover from

8

ACID PROPERTIES



Credit:
https://computersciencewiki.org/index.php/Databases

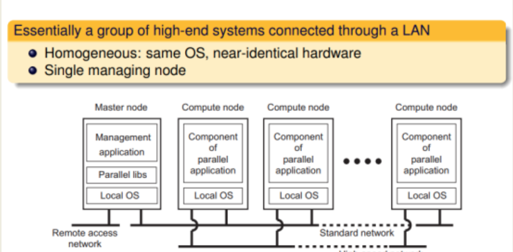
9

FEEDBACK - 2

- ***What are the specific differences between Cluster computing and Grid computing?***

10

CLUSTER COMPUTING

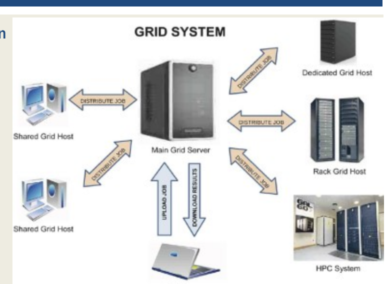


Credit:
Van Steen, M., 2017. Distributed systems.

11

GRID COMPUTING

- Is the aggregation of nodes from everywhere
- Nodes are
  - Heterogeneous
  - Dispersed across several organizations
  - Span a wide-area network (such as the Internet)



Credit:
https://jimsgnblog.blogspot.com/2017/10/grid-computing.html

12

FEEDBACK - 3

- ***What's the clear difference between openness and (distribution) transparency?***
- ***Openness***
- Support for interoperability, portability, extensibility
- Often supported via well-defined interfaces
- ***Distribution transparency***
- Hide from users the fact that a system is actually distributed
- Goal enables users to use a distributed system without specialized knowledge or coupling to specific nodes
- Different types of transparency are considered
- Total distribution transparency is unrealistic and likely unwanted

13

FEEDBACK - 4

- ***What's the difference between consistency and concurrency?***
- **Consistency**
- Every node (replica) has the same view of data at a given point in time irrespective of whichever client has updated the data
- **Concurrency**
- Multiple operations or requests processed simultaneously
- If two concurrent transactions act on the same data, one transaction must go before another
  - One transaction will be blocked, and must wait for the other to finish

14

DISTRIBUTED INFORMATION SYSTEMS

- ***What are some examples of Distributed Information Systems, their key components, and distributed transactions?***
- ‘Distributed Information System’ refers to a set of distributed applications that shares data
- Early on organizations had many networked applications hosted via the Internet
- A networked application ran on a server to make it available to remote clients
- Clients combined requests for different applications, sent the requests, collected responses, and presented a coherent result to the user
- Integration was solely in the client
- Next step was to enable direct application-to-application communication, leading to enterprise application integration

15

DISTRIBUTED INFORMATION SYSTEMS - 2

- The evolution of ***distributed information systems*** is supported by an ever-evolving array of communication technologies and methods
  - Notion of transactions and ACID transactions
  - Transaction Processing Monitor – central coordinator responsible for coordinating the execution of a transaction across a distributed system
  - Remote Procedure Call (RPC)
  - Distributed Objects (CORBA, Java, .NET remoting)
  - Message Oriented Middleware (MOM)

16

OBJECTIVES – 1/12

- Questions from 1/10
- **Message Oriented Middleware**
- Chapter 1.3 – Types of distributed systems
  - Pervasive Systems: Ubiquitous, Mobile, Sensor networks
- Chapter 2: Distributed System Architectures:
  - Chapter 2.1 – Architectural Styles
  - Layered
  - Object-based
    - Service oriented architecture (SOA)
  - Resource-centered architectures
    - Representational state transfer (REST)
  - Event-based
    - Publish and subscribe (Rich Site Summary RSS feeds)

17

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

18

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

January 12, 2023 | TCSS558: Applied Distributed Computing [Winter 2023] School of Engineering and Technology, University of Washington - Tacoma | L4.19

19

OBJECTIVES – 1/12

- Questions from 1/10
- Message Oriented Middleware
- Chapter 1.3 – Types of distributed systems
  - **Pervasive Systems: Ubiquitous, Mobile, Sensor networks**
- Chapter 2: Distributed System Architectures:
  - Chapter 2.1 – Architectural Styles
  - Layered
  - Object-based
    - Service oriented architecture (SOA)
  - Resource-centered architectures
    - Representational state transfer (REST)
  - Event-based
    - Publish and subscribe (Rich Site Summary RSS feeds)

January 12, 2023 | TCSS558: Applied Distributed Computing [Winter 2023] School of Engineering and Technology, University of Washington - Tacoma | L4.20

20

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

January 12, 2023 | TCSS558: Applied Distributed Computing [Winter 2023] School of Engineering and Technology, University of Washington - Tacoma | L4.21

21

OBJECTIVES – 1/12

- Questions from 1/10
- Message Oriented Middleware
- Chapter 1.3 – Types of distributed systems
  - **Pervasive Systems: Ubiquitous Mobile, Sensor networks**
- Chapter 2: Distributed System Architectures:
  - Chapter 2.1 – Architectural Styles
  - Layered
  - Object-based
    - Service oriented architecture (SOA)
  - Resource-centered architectures
    - Representational state transfer (REST)
  - Event-based
    - Publish and subscribe (Rich Site Summary RSS feeds)

January 12, 2023 | TCSS558: Applied Distributed Computing [Winter 2023] School of Engineering and Technology, University of Washington - Tacoma | L4.22

22

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

January 12, 2023 | TCSS558: Applied Distributed Computing [Winter 2023] School of Engineering and Technology, University of Washington - Tacoma | L4.23

23

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

January 12, 2023 | TCSS558: Applied Distributed Computing [Winter 2023] School of Engineering and Technology, University of Washington - Tacoma | L4.24

24

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/cooling control for a room might be modulated, continuously and imperceptibly
- IoT technology helps enable ubiquitous computing

25

OBJECTIVES – 1/12

- Questions from 1/10
- Message Oriented Middleware
- Chapter 1.3 – Types of distributed systems
  - **Pervasive Systems: Ubiquitous, Mobile, Sensor networks**
- Chapter 2: Distributed System Architectures:
  - Chapter 2.1 – Architectural Styles
  - Layered
  - Object-based
    - Service oriented architecture (SOA)
  - Resource-centered architectures
    - Representational state transfer (REST)
  - Event-based
    - Publish and subscribe (Rich Site Summary RSS feeds)

26

PERVASIVE SYSTEM TYPE: MOBILE SYSTEMS

- Emphasis on mobile devices, e.g. smartphones, tablet computers
- 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 consisting of mobile devices. The network is continuously self-configuring. Devices use wireless connections to constitute the network.
  - Key points: self configurating, no permanent infrastructure
- VANET (Vehicular Ad Hoc Network), is a type of MANET that allows vehicles to communicate with roadside equipment.

27

OTHER MANETS

- SPAN – Smart phone ad-hoc network
  - Peer-to-peer networks leveraging bluetooth and wifi available from smart phones without relying on cellular networks, wireless access points, or traditional network infrastructure
- iMANET – Internet based mobile ad-hoc network
  - Ad hoc networks that consists of both mobile devices and Internet-gateway nodes which allows the network to access the Internet and hence span beyond a local ad hoc network

28

OBJECTIVES – 1/12

- Questions from 1/10
- Message Oriented Middleware
- Chapter 1.3 – Types of distributed systems
  - **Pervasive Systems: Ubiquitous, Mobile, Sensor networks**
- Chapter 2: Distributed System Architectures:
  - Chapter 2.1 – Architectural Styles
  - Layered
  - Object-based
    - Service oriented architecture (SOA)
  - Resource-centered architectures
    - Representational state transfer (REST)
  - Event-based
    - Publish and subscribe (Rich Site Summary RSS feeds)

29

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

30

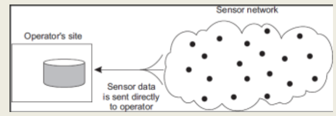
PERVASIVE SYSTEM TYPE: SENSOR NETWORKS - 2

- 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?

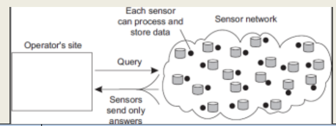
31

CENTRALIZED VS. DECENTRALIZED DATA STORAGE

- Centralized:



- Decentralized:



32

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
- Central server can experience intense network traffic

- Nodes require high compute power
- "Smart" nodes
- Expensive nodes
- network traffic is distributed

33

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?

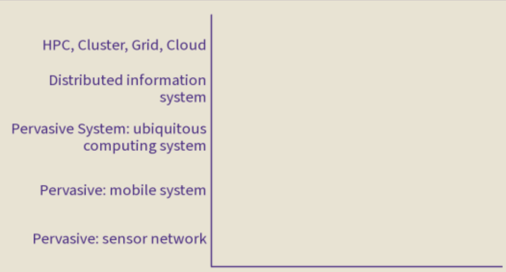
34

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

35

Identify the type of distributed system: E-commerce website (e.g. eBay, Amazon)



Start the presentation to see live content. For screen share software, share the entire screen. Get help at pollev.com/app

36

Identify the type of distributed system: Assisted living home monitoring system for elderly

- HPC, Cluster, Grid, Cloud
- Distributed information system
- Pervasive system: ubiquitous computing system
- Pervasive system: mobile system
- Pervasive system: sensor network

Start the presentation to see live content. For screen share software, share the entire screen. Get help at pollev.com/app

37

Identify the type of distributed system: Seismic monitoring network - warning system for earthquakes

- HPC, Cluster, Grid, Cloud
- Distributed information system
- Pervasive system: ubiquitous computing system
- Pervasive system: mobile system
- Pervasive system: sensor network

Start the presentation to see live content. For screen share software, share the entire screen. Get help at pollev.com/app

38

EXAMPLES OF DISTRIBUTED SYSTEMS

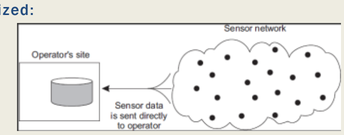
- **Classify the following types of distributed systems:**
- Web search engine
- Assisted living home monitoring system for elderly
- Ecommerce websites: e.g. eBay, Amazon
- Wikipedia: online encyclopedia
- Amazon Elastic Compute Cloud (EC2)
- Massively multiplayer online games (MMO)
- Seismic monitoring network: warning system for earthquakes
- Worldwide Large Hadron Collider (LHC) Computing Grid
- Hospital health informatics and records system
- Canvas: web-based learning environment
- Modern automobile with self-driving features

January 12, 2023 | TCSS558: Applied Distributed Computing [Winter 2023] School of Engineering and Technology, University of Washington - Tacoma | L4.39

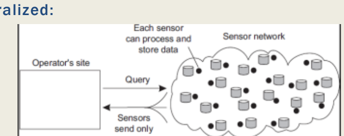
39

WHAT ARE SOME TRADEOFFS FOR CENTRALIZED VS. DECENTRALIZED DATA STORAGE? EXAMPLE: SENSOR NETWORKS

- Centralized:



- Decentralized:



January 12, 2023 | TCSS558: Applied Distributed Computing [Winter 2023] School of Engineering and Technology, University of Washington - Tacoma | L4.40

40

WE WILL RETURN AT 2:40PM

41

OBJECTIVES – 1/12

- Questions from 1/10
- Message Oriented Middleware
- Chapter 1.3 – Types of distributed systems
  - Pervasive Systems: Ubiquitous, Mobile, Sensor networks
- **Chapter 2: Distributed System Architectures:**
  - Chapter 2.1 – Architectural Styles
  - Layered
  - Object-based
    - Service oriented architecture (SOA)
  - Resource-centered architectures
    - Representational state transfer (REST)
  - Event-based
    - Publish and subscribe (Rich Site Summary RSS feeds)

January 12, 2023 | TCSS558: Applied Distributed Computing [Winter 2023] School of Engineering and Technology, University of Washington - Tacoma | L4.42

42

DISTRIBUTED SYSTEM ARCHITECTURES

- Provides logical organization of a distributed system into software **components**
- **Logical**: How system is perceived, modeled
  - *The OO/component abstractions*
  - *The "idealists" view of the system*
- **Physical** – how it really exists
  - The "realist" view of the system
- Middleware
  - Helps separate application from platforms
  - Helps organize and assemble distributed components
  - Helps components communicate
  - Enables system to be extended
  - Supports replication within the distributed system
  - Provides "realization" of the architecture

43

CENTRALIZED VS. DECENTRALIZED DISTRIBUTED SYSTEM ARCHITECTURE

- Tradeoff space: degree of distribution of the system

Fully Centralized ← hybrid → **Decentralized**

Fully Centralized	Decentralized
Single point-of-failure	Multiple failure points
No nodes: vertical scaling	Nodes: horizontal scaling
Always consistent	Eventually consistent
Less available (fewer 9s)	More available (more 9s)
Immediate updates	Rolling updates
No data partitions	Data partitioned or replicated

44

ARCHITECTURAL BUILDING BLOCKS

- **COMPONENT**: modular unit with well-defined, required, and provided **interfaces** that is replaceable within its environment
- Components can be replaced while system is running
- Interfaces must remain the same
- Preserving interfaces enables interoperability
- **CONNECTOR**: enables flow of ***control*** and ***data*** between components
- Distributed system architectures are conceived using components and connectors

45

OBJECTIVES – 1/12

- Questions from 1/10
- Message Oriented Middleware
- Chapter 1.3 – Types of distributed systems
  - Pervasive Systems: Ubiquitous, Mobile, Sensor networks
- Chapter 2: Distributed System Architectures:
  - **Chapter 2.1 – Architectural Styles**
  - Layered
  - Object-based
    - Service oriented architecture (SOA)
  - Resource-centered architectures
    - Representational state transfer (REST)
  - Event-based
    - Publish and subscribe (Rich Site Summary RSS feeds)

46

ARCHITECTURAL STYLES

- Layered
- Object-based
  - Service oriented architecture (SOA)
- Resource-centered architectures
  - Representational state transfer (REST)
- Event-based
  - Publish and subscribe (Rich Site Summary RSS feeds)

47

OBJECTIVES – 1/12

- Questions from 1/10
- Message Oriented Middleware
- Chapter 1.3 – Types of distributed systems
  - Pervasive Systems: Ubiquitous, Mobile, Sensor networks
- Chapter 2: Distributed System Architectures:
  - Chapter 2.1 – Architectural Styles
  - **Layered**
  - Object-based
    - Service oriented architecture (SOA)
  - Resource-centered architectures
    - Representational state transfer (REST)
  - Event-based
    - Publish and subscribe (Rich Site Summary RSS feeds)

48

DISTRIBUTED SYSTEM GOALS TO CONSIDER

- **Consider how architectural style may impact:**
- Availability
- Accessibility
- Responsiveness
- Scalability
- Openness
- Distribution transparency
- Supporting resource sharing
- Other factors…

49

LAYERED ARCHITECTURES

- Components organized in layers
- Component at layer L_j downcalls to lower-level components at layer L_i (where i < j)
- Calls go down
- Exceptional cases may produce upcalls

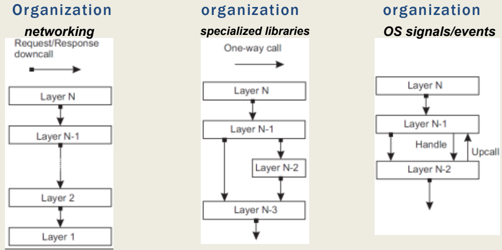
50

LAYERED ARCHITECTURES - 2

Pure-layered Organization — ***networking***

Mixed-layered organization — ***specialized libraries***

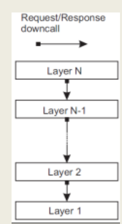
Layered w/ upcalls organization — ***OS signals/events***



51

LAYERED ARCHITECTURES - 3

- Consider an architecture with 5 layers
- Does a client interacting with "Layer 5" of the distributed system need to be concerned with details regarding the implementation of lower layers *(layers 1, 2, 3, 4)* ?



52

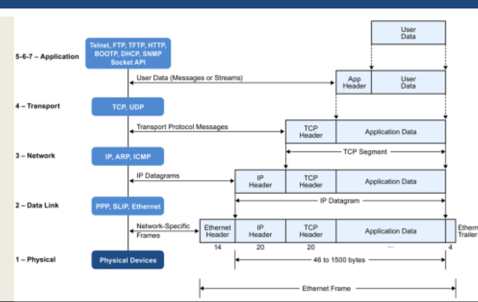
COMMUNICATION-PROTOCOL STACKS

- Example: pure-layered organization
- Each layer offers an interface specifying functions of the layer
- Communication protocol: rules used for nodes to communicate
- Layer provides a **service**
- **Interface** makes service available
- **Protocol** implements communication for a layer

- **New services can be built atop of existing layers to reuse lower level implementation(s)**
  - Abstractions make it easier to reuse existing layers which already implement communication basics

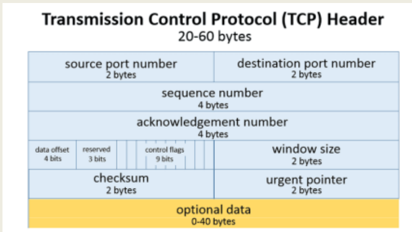
53

HOW A NETWORK PACKET IS BUILT



54

TCP HEADER

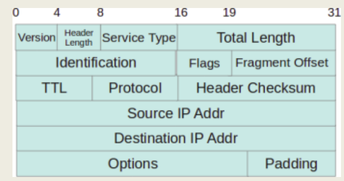


January 12, 2023 | TCSS558: Applied Distributed Computing [Winter 2023] School of Engineering and Technology, University of Washington - Tacoma | L4.55

55

IP HEADER

- Source / Destination IP Addr
- IPv4: 32bits / 4 bytes
- IPv6: 128bits / 16 bytes



January 12, 2023 | TCSS558: Applied Distributed Computing [Winter 2023] School of Engineering and Technology, University of Washington - Tacoma | L4.56

56

TRANSMISSION CONTROL PROTOCOL (TCP)

- TCP (layer 4) provides easy to use API
- API supports:
  - setup, tear down of connection(s)
  - sending and receiving of messages
- TCP preserves ordering of transferred data
- TCP detects and corrects lost data

- But TCP is "protocol" agnostic
  - A protocol is a language of messages exchanged to enable communication
  - Application layer communication is programming language agnostic
  - Code can be written in many programming languages to "speak" the "language" of a custom protocol known as an ***APPLICATION PROTOCOL***

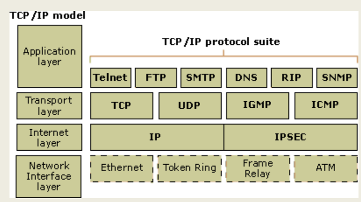
- What should the application protocol say?

January 12, 2023 | TCSS558: Applied Distributed Computing [Winter 2023] School of Engineering and Technology, University of Washington - Tacoma | L4.57

57

COMMON APPLICATION LAYER PROTOCOLS

- Telnet, FTP, TFTP, HTTP, DHCP, DNS, NTP, POP, RTP, SMTP, Telnet, RPC, LDAP

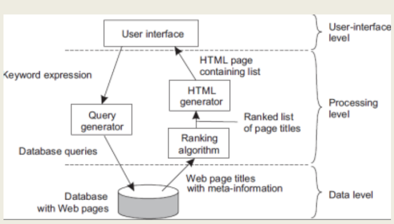


January 12, 2023 | TCSS558: Applied Distributed Computing [Winter 2023] School of Engineering and Technology, University of Washington - Tacoma | L4.58

58

APPLICATION LAYERING

- Distributed application example: Internet search engine



January 12, 2023 | TCSS558: Applied Distributed Computing [Winter 2023] School of Engineering and Technology, University of Washington - Tacoma | L4.59

59

APPLICATION LAYERING

- Three logical layers of distributed applications
  - The data level
  - Application interface level
  - The processing level

January 12, 2023 | TCSS558: Applied Distributed Computing [Winter 2023] School of Engineering and Technology, University of Washington - Tacoma | L4.60

60

APPLICATION LAYERING

- Three logical layers of distributed applications
  - The data level (M)
  - Application interface level (V)
  - The processing level (C)
- Model view controller architecture – distributed systems
  - Model – database - handles data persistence
  - View – user interface - also includes APIs
  - Controller – middleware / business logic

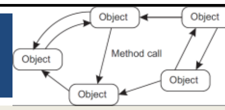
OBJECTIVES – 1/12

- Questions from 1/10
- Message Oriented Middleware
- Chapter 1.3 – Types of distributed systems
  - Pervasive Systems: Ubiquitous, Mobile, Sensor networks
- Chapter 2: Distributed System Architectures:
  - Chapter 2.1 – Architectural Styles
  - Layered
  - **Object-based**
    - **Service oriented architecture (SOA)**
  - Resource-centered architectures
    - Representational state transfer (REST)
  - Event-based
    - Publish and subscribe (Rich Site Summary RSS feeds)

OBJECT-BASED ARCHITECTURES

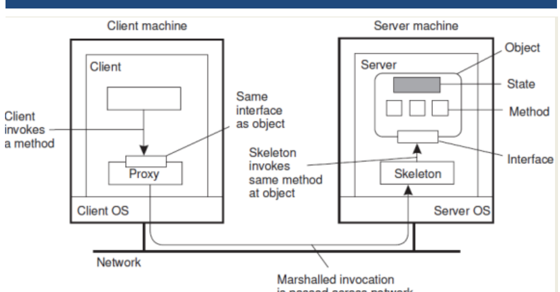
- Enables loose and flexible component organization
- Objects == components
- Enable distributed node interaction via function calls over the network
- Began with C - Remote Procedure Calls (RPC)
  - Straightforward: package up function inputs, send over network, transfer results back
  - Language independent
  - In contrast to web services, RPC calls originally were more intimate in nature
  - Procedures more "coupled", not as independent
  - The goal was not to decouple and widgetize everything

OBJECT-BASED ARCHITECTURES - 2



- Distributed objects Java- Remote Method Invocation (RMI)
  - Adds object orientation concepts to remote function calls
  - Clients bind to proxy objects
  - Proxy provide an object interface which transfers method invocation over the network to the remote host
- How do we replicate objects?
  - Object marshalling – serialize data, stream it over network
  - Unmarshalling- create an object from the stream
  - Unmarshall local object copies on the remote host
  - JSON, XML are some possible data formats

DISTRIBUTED OBJECTS



DISTRIBUTED OBJECTS - 2

- A counterintuitive feature is that state is not distributed
- Each "remote object" maintains its own state
- Remote objects may not be replicated
- Objects may be "mobile" and move around from node to node
  - Common for data objects
- For distributed (remote) objects consider
  - Pass by value
  - Pass by reference .... *(does this make sense?)*

SERVICE ORIENTED ARCHITECTURE

- Services provide always-on encapsulated functions over the internet/web
- Leverage redundant cloud computing infrastructure
- Services may:
  - Aggregate multiple languages, libraries, operating systems
  - Include (wrap) legacy code
- Many software components may be involved in the implementation
  - Application server(s), relational database(s), key-value stores, in memory-cache, queue/messaging services

67

SERVICE ORIENTED ARCHITECTURE - 2

- Are more easily developed independently and shared vs. systems with distributed object architectures
- Less coupling
- An error while invoking a distributed object may crash the system
- An error calling a service (e.g. mismatching the interface) generally does not result in a system crash

68

OBJECTIVES – 1/12

- Questions from 1/10
- Message Oriented Middleware
- Chapter 1.3 – Types of distributed systems
  - Pervasive Systems: Ubiquitous, Mobile, Sensor networks
- Chapter 2: Distributed System Architectures:
  - Chapter 2.1 – Architectural Styles
  - Layered
  - Object-based
    - Service oriented architecture (SOA)
  - **Resource-centered architectures**
    - **Representational state transfer (REST)**
  - Event-based
    - Publish and subscribe (Rich Site Summary RSS feeds)

69

RESOURCE BASED ARCHITECTURES

- Motivation:
  - Increasing number of services available online
  - Each with specific protocol(s), methods of interfacing
  - Connecting services w/ different TCP/IP protocols → integration nightmare
    - Need for specialized client for each service that speaks the application protocol "language"...
- Need standardization of interfaces
  - Make services/components more pluggable
  - Easier to adopt and integrate
  - Common architecture

70

REST SERVICES

- Representational State Transfer (REST)
- Built on HTTP
- Four key characteristics:
  1. Resources identified through single naming scheme
  2. Services offer the same interface
     - Four operations: GET PUT POST DELETE
  3. Messages to/from a service are fully described
  4. After execution server forgets about client
     - Stateless execution

71

HYPERTEXT TRANSPORT PROTOCOL (HTTP)

- An ASCII-based request/reply protocol for transferring information on the web
- HTTP request includes:
  - request method (GET, POST, etc.)
  - Uniform Resource Identifier (URI)
  - HTTP protocol version understood by the client
  - headers—extra info regarding transfer request
- HTTP response from server
  - Protocol version & status code →
  - Response headers
  - Response body

HTTP status codes:
2xx — *all is well*
3xx — *resource moved*
4xx — *access problem*
5xx — *server error*

72

REST-FUL OPERATIONS

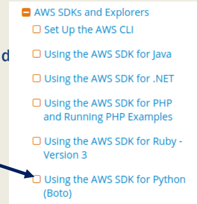
Operation	Description	
PUT	Create a new resource	(C)reate
GET	Retrieve state of a resource in some format	(R)ead
POST	Modify a resource by transferring a new state	(U)pdate
DELETE	Delete a resource	(D)elete

- Resources often implemented as objects in OO languages
- REST is weak for tracking state
- Generic REST interfaces enable ubiquitous *"so many"* clients

73

EXAMPLE: AMAZON S3

- Amazon S3 offers a REST-based interface
- Requires signing HTTP authorization header or passing authentication parameters in the URL query string
- REST: GET/PUT/POST/DELETE
- SOAP: 16 operations, moving toward deprecation
- Python boto ~50 operations (SDK for Python)
- SDKs for other languages



74

REST - 2

- Defacto web services protocol
- Requests made to a URI – uniform resource identifier
- Supersedes SOAP – Simple Object Access Protocol
- Access and manipulate web resources with a predefined set of stateless operations (known as web services)
- Responses most often in JSON, also HTML, ASCII text, XML, no real limits as long as text-based
- curl – generic command-line REST client: https://curl.haxx.se/

75

```
// WSDL Service Definition
<?xml version="1.0" encoding="UTF-8"?>
<definitions  name ="DayOfWeek"
  targetNamespace="http://www.roguewave.com/soapworx/examples/DayOfWeek.wsdl"
  xmlns:tns="http://www.roguewave.com/soapworx/examples/DayOfWeek.wsdl"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns="http://schemas.xmlsoap.org/wsdl/">
  <message name="DayOfWeekInput">
    <part name="date" type="xsd:date"/>
  </message>
  <message name="DayOfWeekResponse">
    <part name="dayOfWeek" type="xsd:string"/>
  </message>
  <portType name="DayOfWeekPortType">
    <operation name="GetDayOfWeek">
      <input message="tns:DayOfWeekInput"/>
      <output message="tns:DayOfWeekResponse"/>
    </operation>
  </portType>
  <binding name="DayOfWeekBinding" type="tns:DayOfWeekPortType">
    <soap:binding style="document"
      transport="http://schemas.xmlsoap.org/soap/http"/>
    <operation name="GetDayOfWeek">
      <soap:operation soapAction="getdayofweek"/>
      <input>
        <soap:body use="encoded"
          namespace="http://www.roguewave.com/soapworx/examples"
          encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
      </input>
      <output>
        <soap:body use="encoded"
        namespace="http://www.roguewave.com/soapworx/examples"
        encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
      </output>
    </operation>
  </binding>
  <service name="DayOfWeekService" >
    <documentation>
      Returns the day-of-week name for a given date
    </documentation>
    <port name="DayOfWeekPort" binding="tns:DayOfWeekBinding">
      <soap:address location="http://localhost:8090/dayofweek/DayOfWeek"/>
    </port>
   </service>
</definitions>
```


76

```
// REST/JSON
// Request climate data for Washington

{
 "parameter": [
  {
    "name": "latitude",
    "value":47.2529
  },
  {
    "name": "longitude",
    "value":-122.4443
  }
  ]
}
```


77

OBJECTIVES – 1/12

- Questions from 1/10
- Message Oriented Middleware
- Chapter 1.3 – Types of distributed systems
  - Pervasive Systems: Ubiquitous, Mobile, Sensor networks
- Chapter 2: Distributed System Architectures:
  - Chapter 2.1 – Architectural Styles
  - Layered
  - Object-based
    - Service oriented architecture (SOA)
  - Resource-centered architectures
    - Representational state transfer (REST)
  - **Event-based**
    - **Publish and subscribe (Rich Site Summary RSS feeds)**

78

PUBLISH-SUBSCRIBE ARCHITECTURES

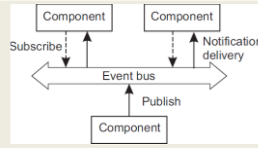
- Enables separation between processing and coordination
- Types of coordination: ***temporal and referential coupling:***

	Temporally coupled (at the same time)	Temporally decoupled (at different times)
Referentially coupled (*dependent on name*)	**Direct** Explicit synchronous service call	**Mailbox** Asynchronous by name (address)
Referentially decoupled (*name not required*)	**Event-based** Event notices published to shared bus, w/o addressing	**Shared data space** Processes write tuples to a shared data space

Publish and subscribe architectures

79

PUBLISH-SUBSCRIBE ARCHITECTURES - 2

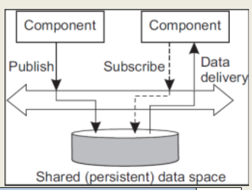


- **Event-based coordination**
- Processes do not know about each other explicitly
- **Processes:**
  - **Publish:** a notification describing an event
  - **Subscribe:** to receive notification of specific kinds of events
- Assumes subscriber is presently up (*temporally coupled*)
- Subscribers must actively **MONITOR** event bus

80

PUBLISH SUBSCRIBE ARCHITECTURES - 3

- **Shared data space**
- Full decoupling (name and time)
- Processes publish “tuples” to shared dataspace (publish)
- Processes provide search pattern to find tuples (subscribe)
- When tuples are added, subscribers are notified of matches
- **Key characteristic:** Processes have no explicit reference to each other



81

PUBLISH SUBSCRIBE ARCHITECTURES - 4

- Subscriber describes events interested in
- Complex descriptions are intensive to evaluate and fulfil
- **Middleware will:**
- Publish matching notification and data to subscribers
  - Common if middleware lacks storage
- Publish only matching notification
  - Common if middleware provides storage facility
  - Client must explicitly fetch data on their own
- Publish and subscribe systems are generally scalable
- **What would reduce the scalability of a publish-and-subscribe system?**

82

QUESTIONS

83