IS 651: Distributed Systems

Jianwu Wang

https://userpages.umbc.edu/~jianwu/

Spring 2021

About instructor

Jianwu Wang (John-woo)

Currently

- Assistant Professor of Data Science

Previously

- Research Scientist, UCSD

Research Area

- Data Science, Big Data, Distributed Computing (including Service Computing)

Contact: jianwu@umbc.edu

Office hour: 5:00-6:00 pm Monday & Thursday or by appointment

Introduce Yourself

- Basic info: name, where you are from, which year in the program
- Career goal
- Background/experiences in distributed systems
- What do you hope to learn from this course?
- Experiences with online learning and related tools: WebEx, Blackboard Collaborate, Panopto, Lockdown Browser, Slack, Piazza, etc.
- A fun thing you did over the winter break 😀

Online Teaching/Learning

- Challenges
    - Engagement between students and instructor, and between students
    - Students in different time zones
- Solutions (details in next slides)
    - Push for more interaction (need your help and participation to achieve it)
    - Synchronous lecturing with recordings

Main Tools for Instruction and Interaction

- WebEx: for synchronous lecturing, discussion and office hour
- Piazza: for questions and answers
- Slack: for quick messages and team communication

Links are at Piazza.

Course Syllabus and Schedule

- Course website: https://userpages.umbc.edu/~jianwu/is651/651.syll.s21.html
- Homework/exercises are subject to change
- Current slides are the ones used in previous semester for your reference
    - They will be updated after each lecture and should have Spring 2021 on it

Synchronous communication

- Lecture time (4:30-7:00 Friday) via WebEx
  - Please mute yourself by default
  - Write in chat if you have any questions. You can unmute yourself to talk more about your questions.
  - You can stay after lecture for quick discussions
- Office hour (5-6 Monday and Thursday) via WebEx
  - Make appointment ahead of time on the google spreadsheet

Asynchronous communication

- Piazza (http://piazza.com/umbc/spring2021/is651/home)
  - You can use it through either website or **smartphone app**
  - Folder for each chapter
  - Send to instructor on private issues
    - Better informed and faster responses than emails
- Slack (https://is-651-umbc.slack.com)
  - You can use it through either website, **desktop app** or **smartphone app**
  - You can create your team's own channels for discussions
- Instructor will try to reply within 24 hours
  - Ask your questions early, not right before the deadline

Online Instruction Information

- More at this Piazza post: https://piazza.com/class/kjy8thbsiiih4?cid=6

What is this course about?

- Overview of the concepts, systems and techniques of distributed systems
  - Basic concepts and principles of distributed systems, which are useful in many real-world applications/projects
  - Lectures/readings, discussions, case studies, extensive hands-on exercises/homework
  - By the end of this course, you will have a good technical understanding of many distributed system related technologies
- NOT a programming class
  - We will use several languages (XML, JavaScript, XQuery, etc.), only the basics for exercises/homework
  - No real programming. Only need to understand programs and make some changes
- More technical than MIS, more application-oriented than CS

Why this course is important?

- People are using distributed systems everyday
  - Web sites: Gmail, Facebook, …
  - Distributed databases: MySQL Cluster, Hbase, …
  - Distributed file systems: NFS, AFS, …
  - Distributed scientific software: MPI, OpenMP, …
- Knowing the knowledge might help your future career
  - One of most actively evolving topic: Cloud, Big Data, Mobile, GPU, …
  - Design/implement a new distributed system
  - Running data analytics on a distributed system
  - Managing a distributed database
  - …

Grading

- Participation: 4 points (4%)
  - Exercise/homework presentation is an important part of participation
  - Each student has two chances to present his/her exercise/homework
  - Use Presenter 3 column only if Presenter 1 & 2 are filled for all exercises/homework
- 11 team-based exercises: 11 x 2 =22 points (22%)
  - Due: Thursday before the next lecture
- 7 team-based homework: 7 x 5 = 35 points (35%)
  - Due: Thursday before the next lecture
- 1 team-based case study: 9 points (9%) grading rubric
  - Select a topic and find a related paper or project interesting to your group
  - One bonus point if you are able to demonstrate your work
- 2 *non-comprehensive* exams: 15+15 = 30 points (30%)

Assignment Submission

- You can work with your teammates on exercise/homework assignment. But each student still needs to have your own implementation (including URL)
- You can submit your exercise/homework as many times as you need
- Submission after deadline (End of Thursday) will be penalized:
    - -10% for each additional day you used
- Upload your source code files for your submission
- Very similar homework/exercise submissions from different teams will be investigated and reported

Academic Integrity

- Very similar homework/exercise submissions from different teams will be investigated and reported
- We plan to use **Respondus Lockdown Browser (RLDB)** for exams to prevent plagiarism
- You have much higher chance to fail because of plagiarism than not learning well in class

How to Study Well?

- Good participation in class: question, presentation, etc.
- Good group collaboration for exercise, homework, and case study
- Start exercise and homework early
- Read the requirements carefully
- Read book chapters, especially before exams
- If needed, ask for help early
    - If you are experiencing any problems that affect your performance in this class, please contact instructor immediately
- Study hard from the beginning of the semester

Prerequisite Knowledge

- Programming with systems analysis and design
- Networking
- Databases
- Linux

IS 651: Distributed Systems Chapter 1: Distributed Systems Introduction

Jianwu Wang

https://userpages.umbc.edu/~jianwu/

Spring 2021

Learning Objectives

- After learning chapter 1, you should be able to
  - Understand the basics of distributed systems and service oriented architecture (SOA)
  - Understand the layered architecture of network
  - Can access and navigate linux server (gl.umbc.edu)

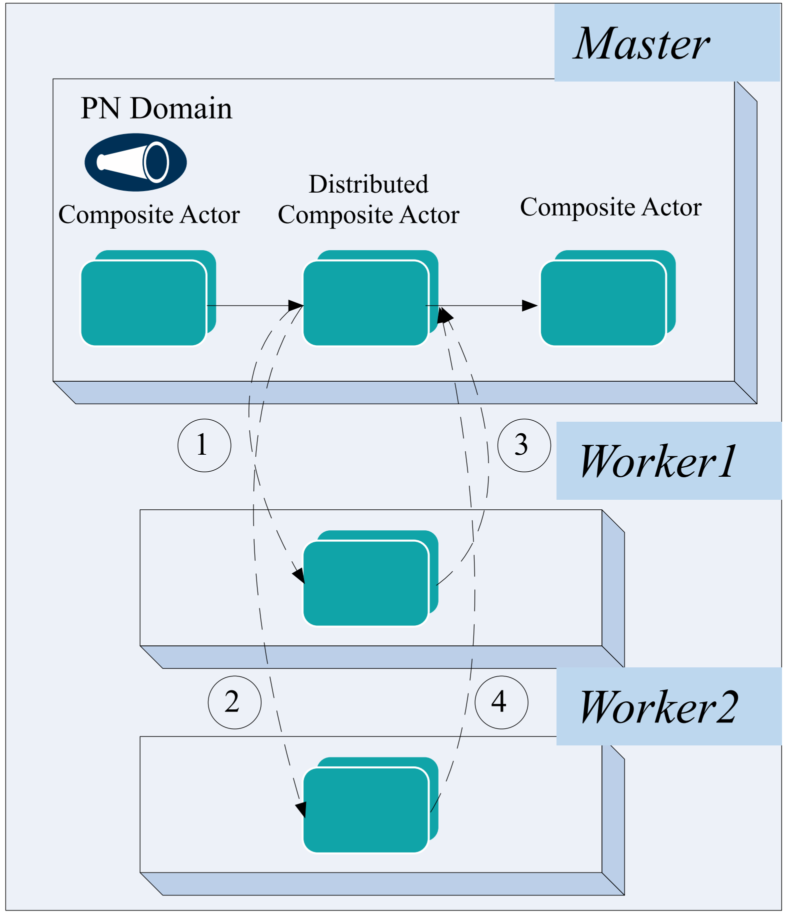
Lecture Components

- Understand the basics of distributed systems and service oriented architecture (SOA)
- Understand the layered architecture of network

Distributed Systems

- A distributed system consists of **multiple autonomous** computers that communicate through a computer **network**. The computers interact with each other in order to achieve a **common goal**.

- Definition: A distributed system is a collection of independent computers that appears to its users as a single coherent system.



Service-Oriented Architecture (SOA)

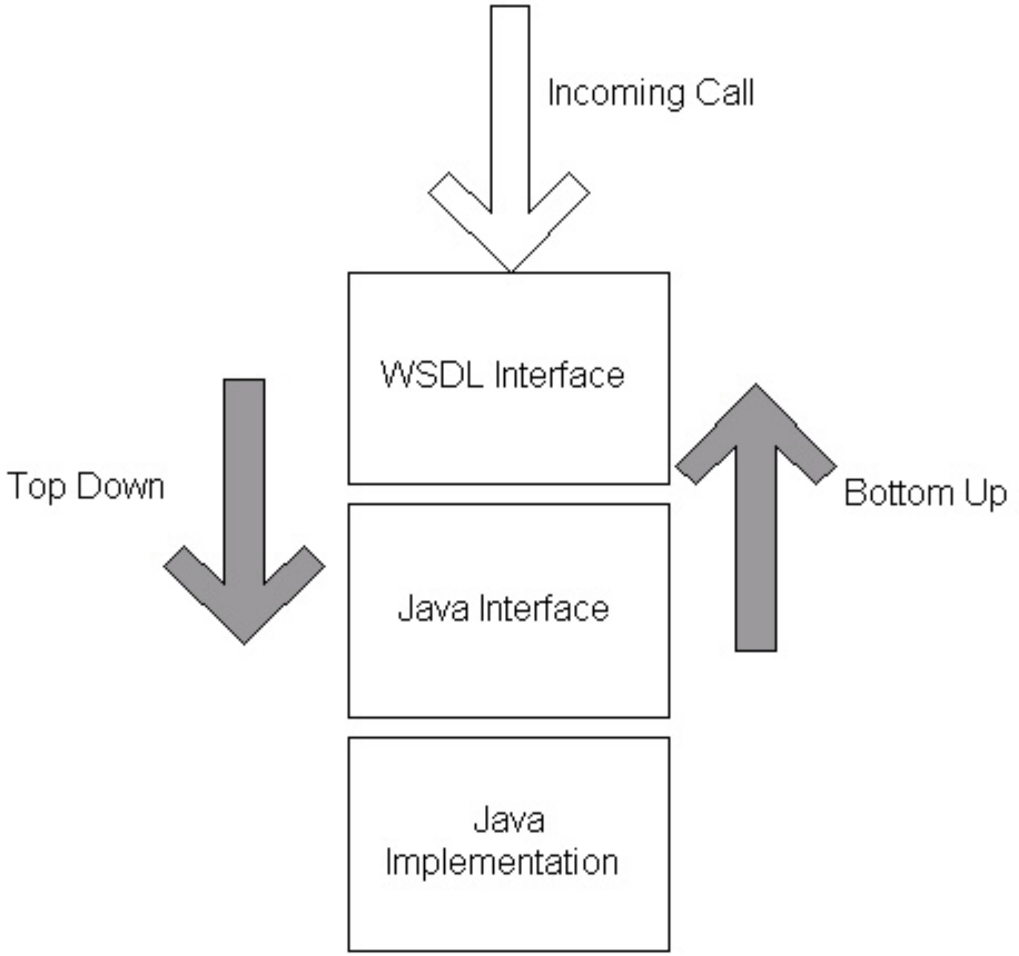
- SOA is an evolution of distributed computing
- SOA defines how two computing entities, such as programs, interact in such a way as to enable one entity to perform a unit of work **on behalf of** another entity
- Service interactions are defined using a description language. Each interaction is **self-contained** and **loosely coupled**, so that each interaction is independent of any other interaction
- This is an **abstract** architectural concept and no specific technology is assumed

SOA is very popular in real world

- Examples:
  - REST service
  - Amazon Web Services
  - Cloud computing follows XaaS architecture
  - Microservices
  - Service oriented manufacturing
  - MEAN (MongoDB, Express.js, AngularJS, and Node.js) stack for web application
  - …

SOA Concepts

- Top-down
- Bottom-up
- Loosely-coupled
- Interface, not application



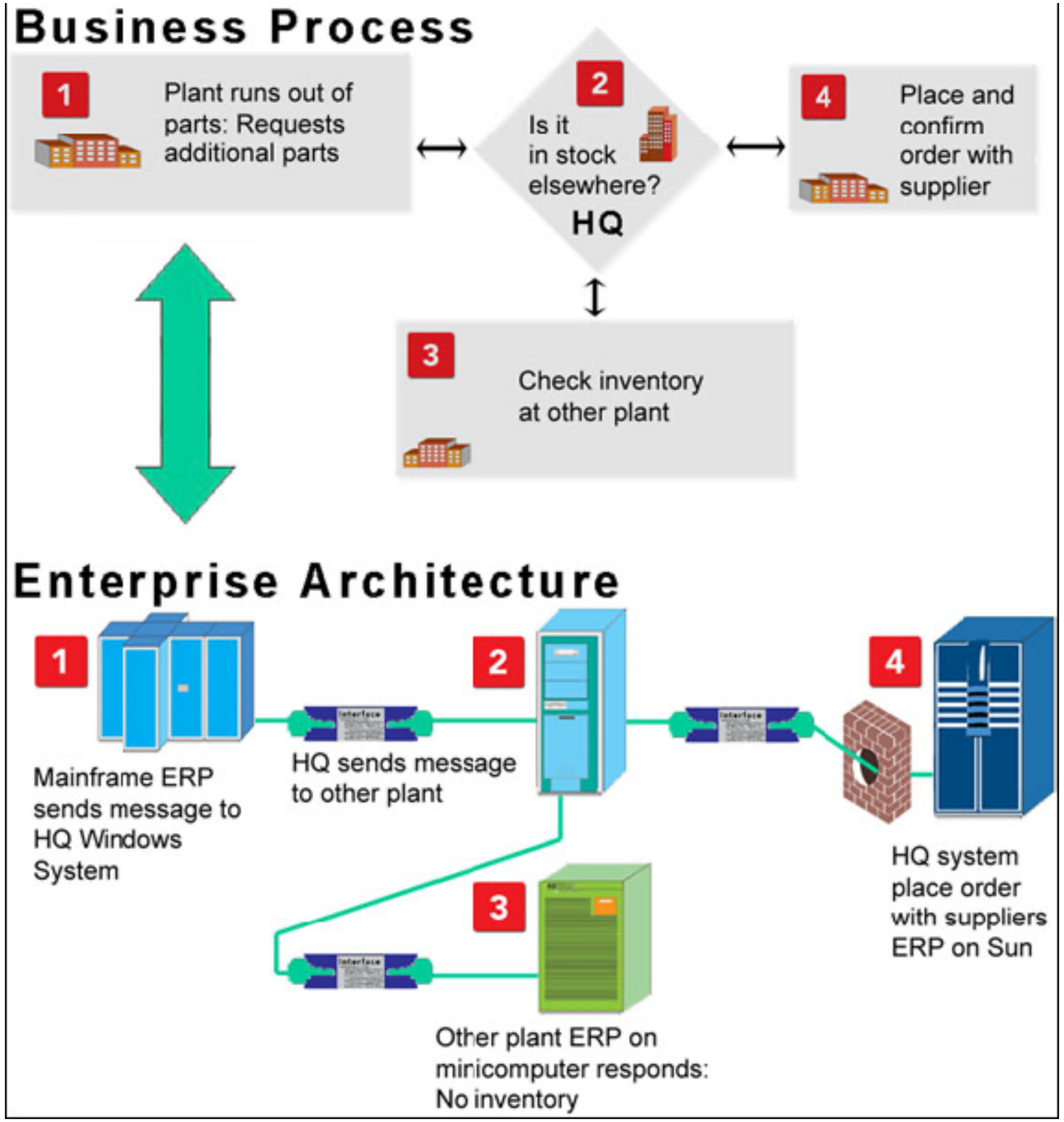
XML Web Service

- XML Web Services are the current technology most associated with SOA
- A Web Service (XML Web Service) is a unit of code that can be activated using **HTTP** requests. Stated another way, a Web Service is an application component (in any language) that can be remotely callable using standard Internet Protocols such as HTTP and XML.
- Web Services came into existence to deliver distributed computing over the **Internet**.
- A major advantage of the Web services architecture is, it allows programs written in different languages on different platforms to communicate with each other in a standards-based way.

Traditional B2B Scenario

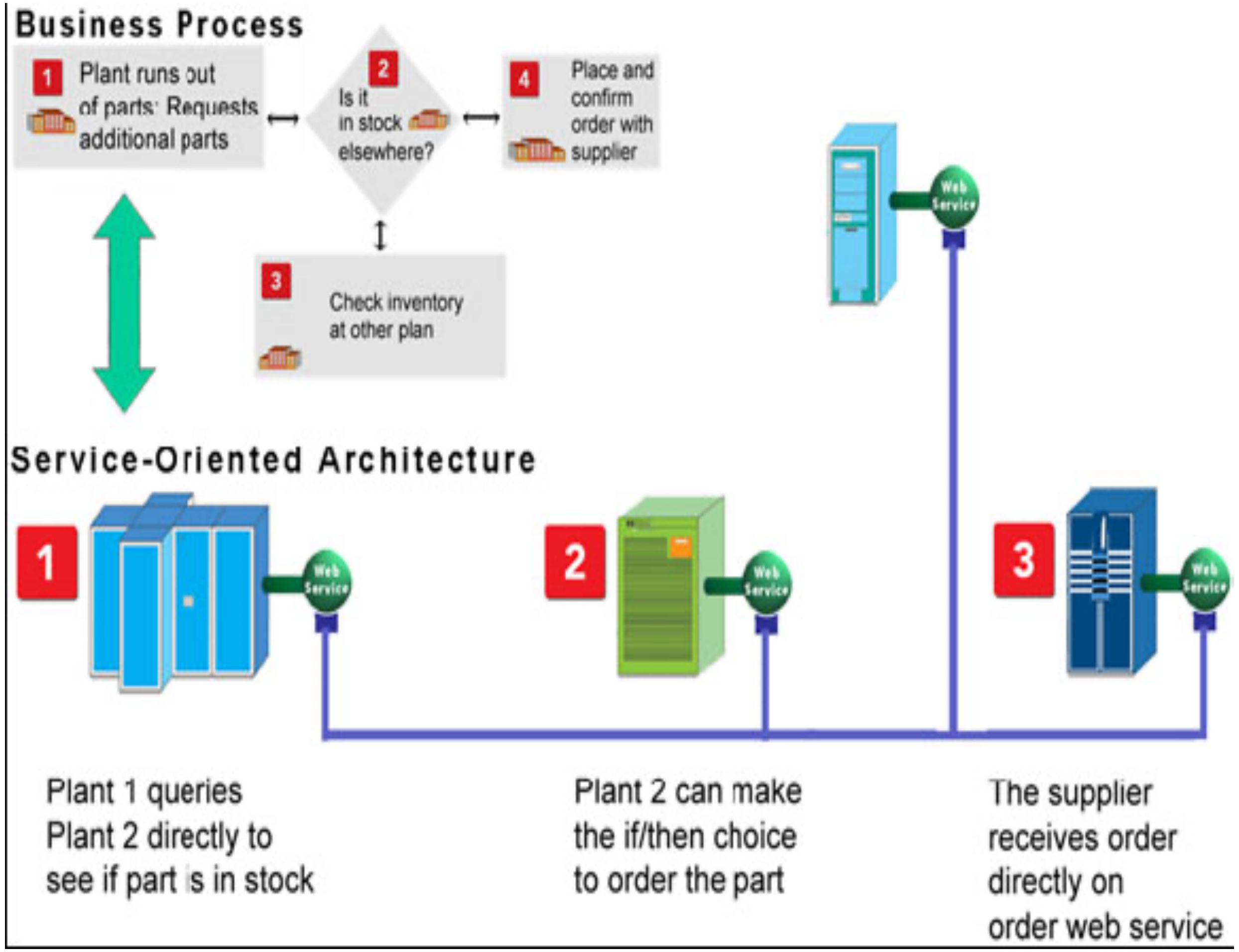
- To support the company's inventory query and supply ordering with no web services, the enterprise architecture requires that four systems be connected using three proprietary interfaces
- Across organization communication

From http://www.javaworld.com/javaworld/jw-11-2005/jw-1128-soa.html


Business Process
1
Plant runs out of parts: Requests additional parts
2
Is it in stock elsewhere?
HQ
4
Place and confirm order with supplier
3
Check inventory at other plant
Enterprise Architecture
1
Mainframe ERP sends message to HQ Windows System
HQ sends message to other plant
2
4
HQ system place order with suppliers ERP on Sun
3
Other plant ERP on minicomputer responds: No inventory

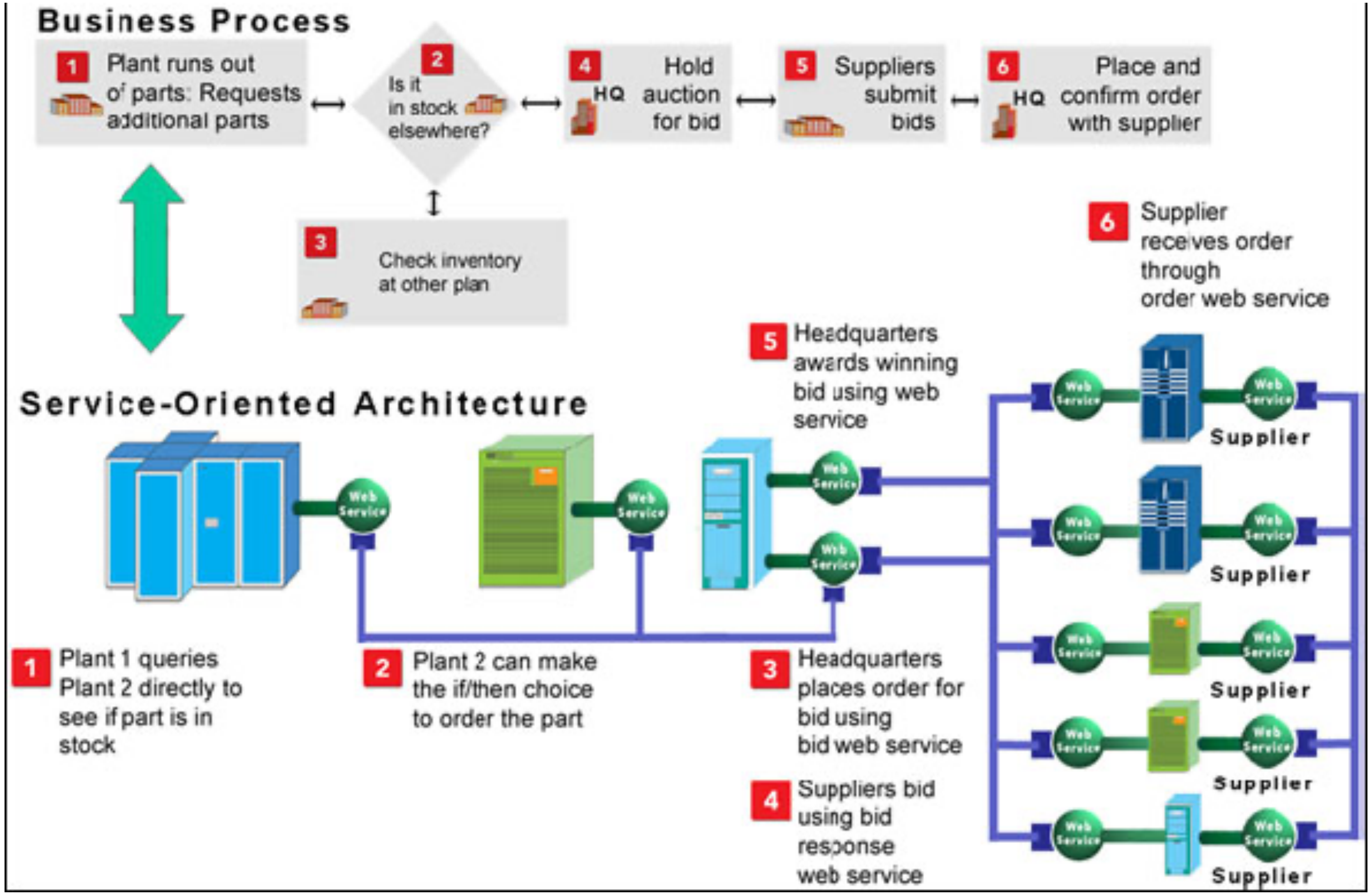
B2B Scenario with SOA

- SOA opens up a number of new possibilities for conducting B2B commerce without significant reworking of the underlying systems
- In addition to eliminating the proprietary interfaces, the SOA makes it easily possible for the first plant to check directly with the second plant and place orders **without** going through the HQ computer


Business Process
1
Plant runs out of parts: Requests additional parts
2
Is it in stock elsewhere?
4
Place and confirm order with supplier
3
Check inventory at other plan
Web Service
Service-Oriented Architecture
1
2
3
Web Service
Web Service
Web Service
Plant 1 queries Plant 2 directly to see if part is in stock
Plant 2 can make the if/then choice to order the part
The supplier receives order directly on order web service

B2B Scenario with More SOA

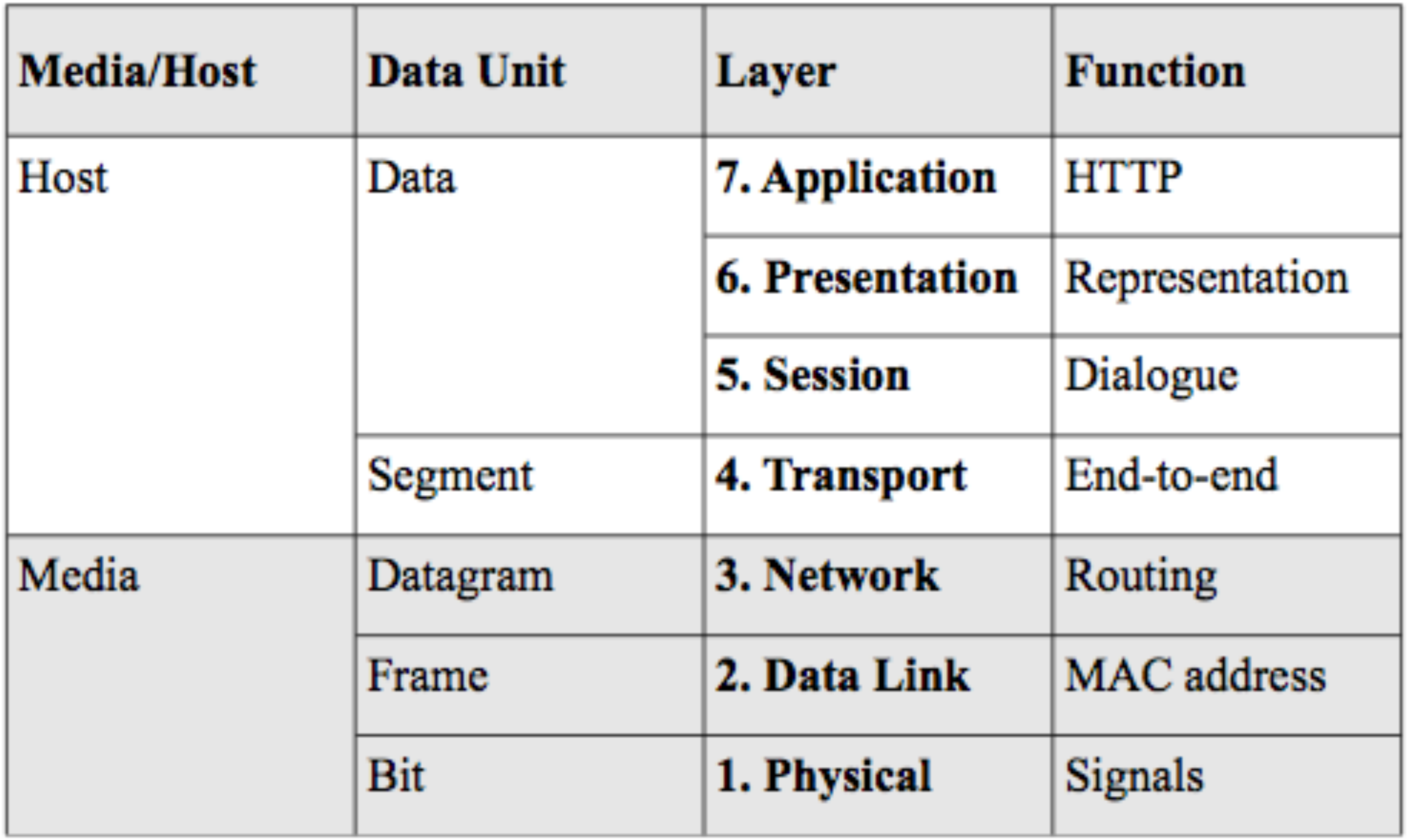
- The manufacturer now wants to have an electronic competitive bidding system for its orders
- The suppliers who want to bid on the opportunity to win business from the manufacturer can connect to the bidding system through a Web service


Business Process
1 Plant runs out of parts: Requests additional parts
2 Is it in stock elsewhere?
3 Check inventory at other plan
4 HQ Hold auction for bid
5 Suppliers submit bids
6 HQ Place and confirm order with supplier
Service-Oriented Architecture
Web Service
1 Plant 1 queries Plant 2 directly to see if part is in stock
2 Plant 2 can make the if/then choice to order the part
3 Headquarters places order for bid using bid web service
4 Suppliers bid using bid response web service
5 Headquarters awards winning bid using web service
6 Supplier receives order through order web service
Supplier
Supplier
Supplier
Supplier
Supplier

Lecture Components

- Understand the basics of distributed systems and service oriented architecture (SOA)
- Understand the layered architecture of network

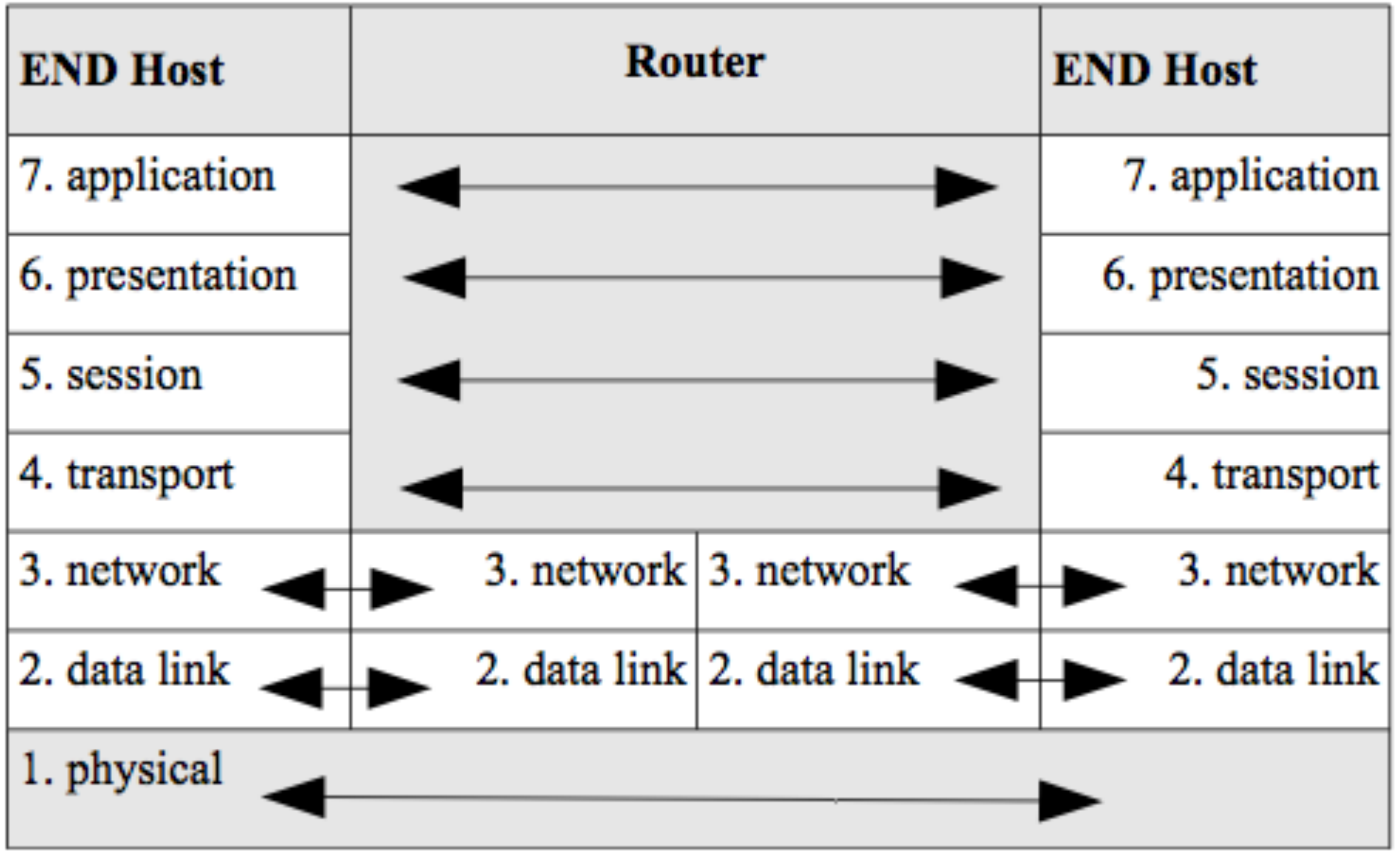
Networking - OSI Seven Layer Model



Media/Host	Data Unit	Layer	Function
Host	Data	**7. Application**	HTTP
		6. Presentation	Representation
		5. Session	Dialogue
	Segment	**4. Transport**	End-to-end
Media	Datagram	**3. Network**	Routing
	Frame	**2. Data Link**	MAC address
	Bit	**1. Physical**	Signals

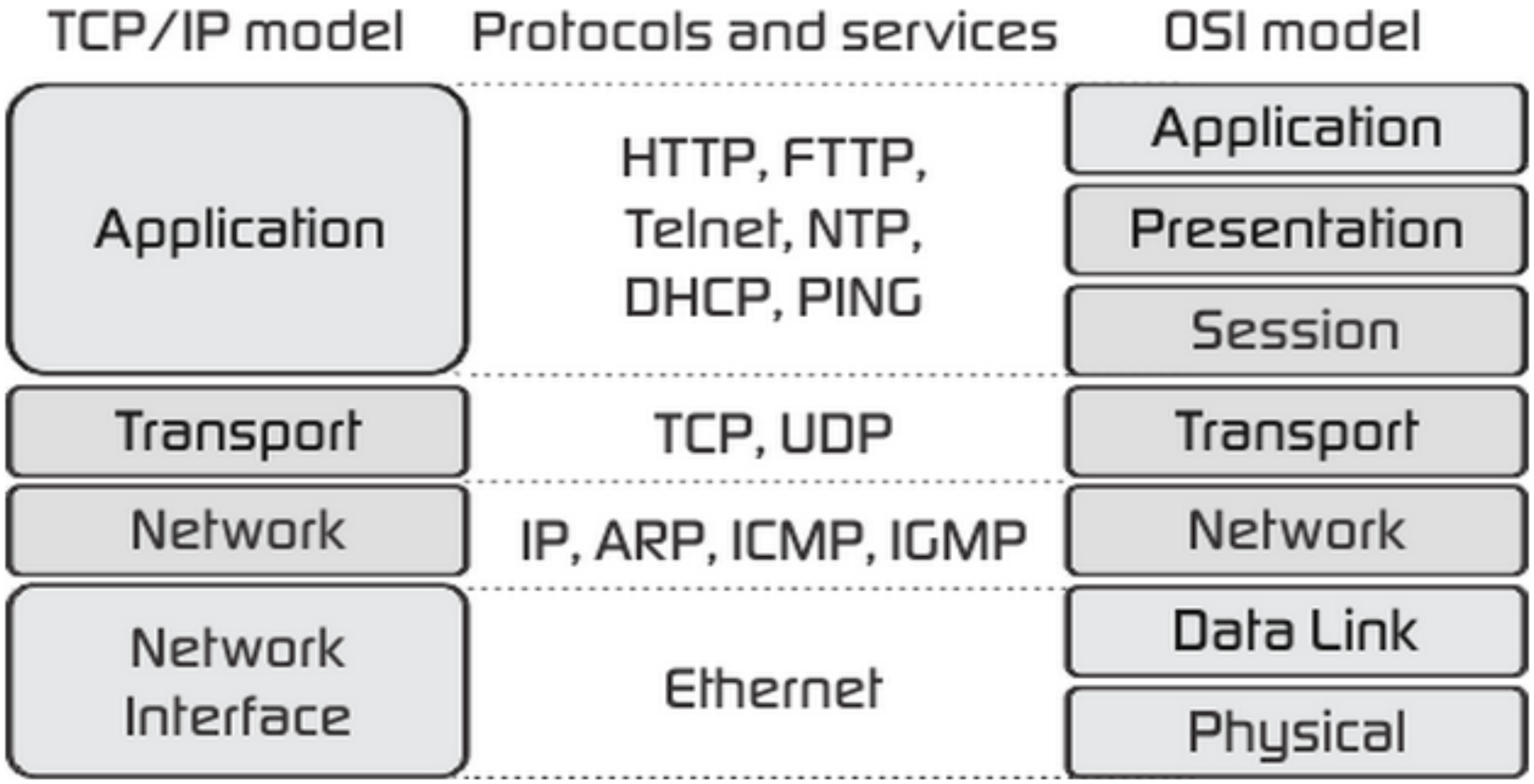
Networking - Peer-to-Peer (P2P) Communication

- Peer-to-peer communications through protocol data unit (PDU) headers
- The data unit of one layer is part of data unit of its underlying layer



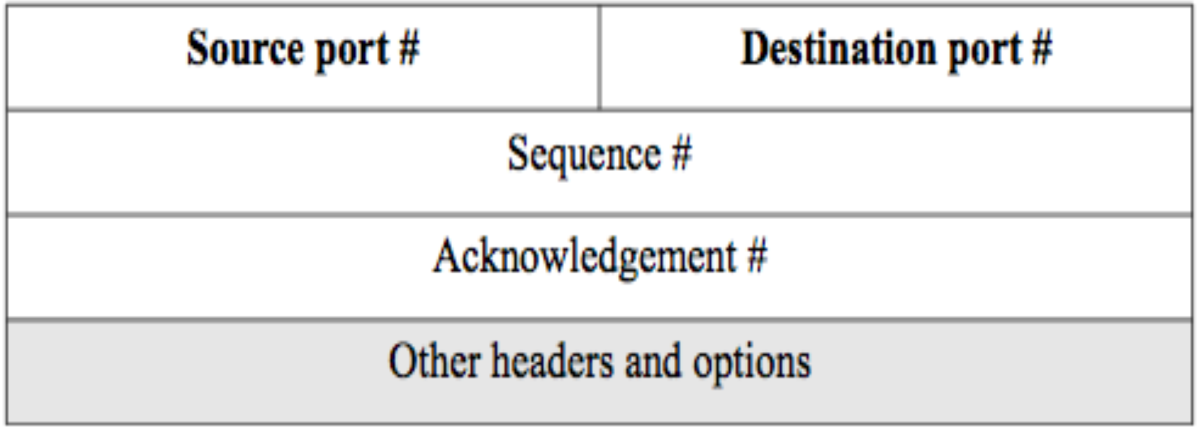
END Host	Router		END Host
7. application			7. application
6. presentation			6. presentation
5. session			5. session
4. transport			4. transport
3. network	3. network	3. network	3. network
2. data link	2. data link	2. data link	2. data link
1. physical			

Networking - TCP/IP



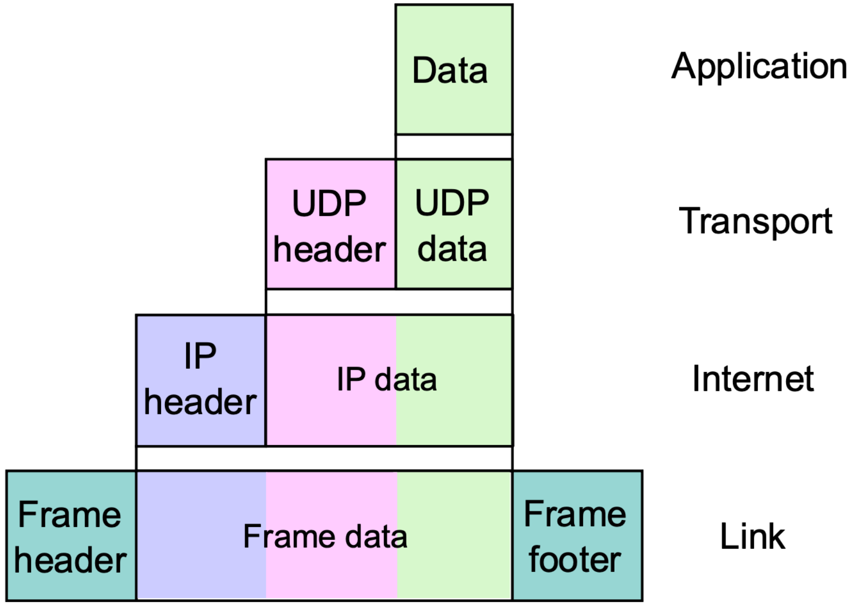
TCP/IP model	Protocols and services	OSI model
Application	HTTP, FTTP, Telnet, NTP, DHCP, PING	Application
		Presentation
		Session
Transport	TCP, UDP	Transport
Network	IP, ARP, ICMP, IGMP	Network
Network Interface	Ethernet	Data Link
		Physical

Networking - TCP/IP Headers



Source port #	Destination port #
Sequence #	
Acknowledgement #	
Other headers and options	

TCP Header



TCP/IP header encapsulation, from Wiki

Distributed Systems Topics

- **Architecture**: how distributed systems are put together to provide a single system abstraction from many separate parts
- **Fault-tolerance**: how distributed systems can continue to provide service when some parts have failed to provide availability until they are recovered
- **Consistency**: how distributed systems can maintain logical coherency when data is distributed
- **Scalability**: how a distributed system can grow to meet demand in an efficient and effective way
- **Performance**: how to optimize response time when components are distributed
- **Security**: how to ensure data integrity and confidentiality is a distributed environment

Eight Fallacies for Distributed Systems

- The network is reliable
- Latency is zero
- Bandwidth is infinite
- The network is secure
- Topology does not change
- There is one administrator
- Transport cost is zero
- The network is homogeneous

Unix/Linux Command-line

- Use of an SSH client to login to a (linux) server
  - Mac: Terminal command line
  - Windows: Putty, Bitvise SSH Client
- Use of an SCP client for file transfer between your machine to a server
  - Mac: Terminal command line, FileZilla, cyberduck
  - Windows: FileZilla, cyberduck, Bitvise SSH Client, WinSCP
- Use of a small number of unix commands
  - ls, mkdir, cd, pwd, man, more, cat, wget, …
- Edit a text file on a unix machine
  - vi, nano, emacs, …
- Navigation of your student account on gl.umbc.edu

Tool Demonstration

- SSH
- Unix commands: vi, pwd, etc.
- FileZilla

To-Do List Before the Next Class

- Introduce yourself on Piazza (self-introduction folder), which will help find teams (Part of Exercise 1)
  - Name
  - Time zone
  - Which year in the program
  - Background/experiences in distributed systems and programming
- Form teams as soon as possible
  - Post your team info on Piazza (team-introduction folder): team name, team members
  - Each team will have 1-3 members
- Work on exercise 1 with your teammates
  - The same team will work together on exercise, homework and case study
  - Submit exercise 1 by the end of Thursday
- Exercise/homework notes
  - Even exercises and homework are team based. The submission is still **individual submission** because many are associated with individual accounts
  - Any time a homework asks you to put information into a file with a .txt extension, it must be a plain text file. Never use a word processor format
  - Most submissions include uploading text files you worked on and urls on how to visit them on gl machine

Explanation of Chapter 1 References and Exercise 1

- https://userpages.umbc.edu/~jianwu/is651/651.ref.s21.html#ch1