

Design and Engineering of Computer Systems

Lecture 1: Introduction to Computer Systems

Mythili Vutukuru

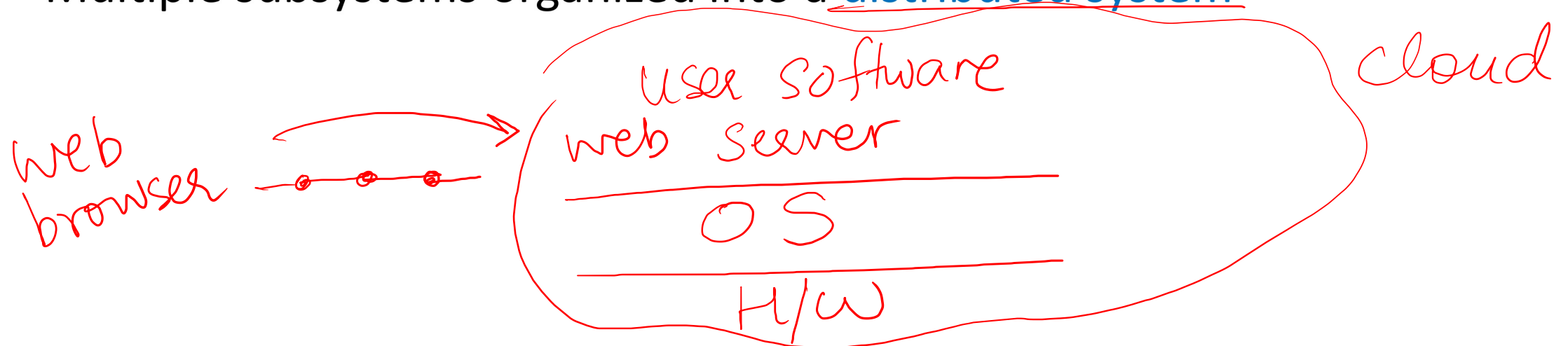
IIT Bombay

Overview of the course

- **Computer system:** A set of computers (or computing elements) that provide a specific functionality to a set of users
- Examples:
 - E-commerce website ✓
 - Train reservation system ✓
 - Video conferencing system ✓
 - Video streaming service ✓
 - Online banking system ✓
- This course will help you understand how to design and build such systems
 - Understand the various components and subsystems
 - Understand how all components work together end-to-end

Components of a computer system

- User Software: web servers and clients (browser), applications, databases, video players, mobile applications, ..
- System software: operating system (OS), networking software, ...
- Hardware: CPU, memory, disk, networking hardware (routers, switches, interconnects)
- Multiple subsystems organized into a distributed system



Real life computer systems are complex

- Real life systems are complex:
 - Multiple interacting components and sub-systems
 - Each component independently developed, but have to work together for a common purpose
 - Prone to failures, bugs, crashes
- But still, we expect:
 - The system always does what it is supposed to do ... (functional correctness)
 - Quickly, efficiently, for a large number of users, lots of data ... (performance)
 - Even when it is overloaded or when failures occur ... (reliability)
- Functional requirements and non-functional requirements

Example: e-commerce website

- Functional requirements:

- Search for products, buy, pay, ship
- Get recommendations based on past purchases
- Store user shipping, billing, order information
- Interface for sellers to sell their products

- Non-functional requirements:

- Support millions of transactions per second (throughput)
- Return responses quickly, within a few seconds (latency)
- Be able to increase capacity easily, quickly during special “sale” (scalability)
- Be available even when some machines fail or malfunction (fault tolerance)
- Do not charge for a product and then crash before confirmation (atomicity)
- Show same shopping cart and account details across devices (consistency)

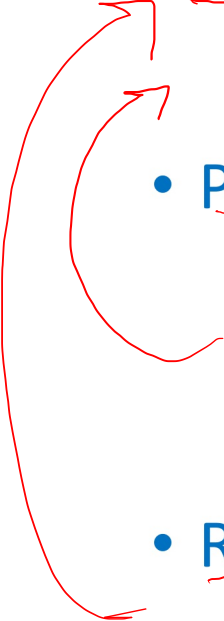
More examples of requirements from computer systems

- Video streaming service: recommend good movies, store many movies and retrieve efficiently, stream movies without delays, handle billing and charging correctly, ramp up performance when large number of users are watching a popular video
- Online banking service: quick and efficient access of all account services, safety / security / integrity of account data, do not lose any money even if there are any software/hardware failures!
- Instant messaging: quick communications across individual and group channels, support large number of users, support to send large multimedia files efficiently, communicate even when users are on different networks

More examples of requirements from computer systems

- Social media networking: ability to share and view text/images/videos from connections quickly in real-time, handle large volumes of messages from large number of users, recommend suitable content to display from the large number of posts from friends, recommend new connections
- Online storage of files, documents: ability to store/retrieve files from multiple computers, handle lots of data from large number of users, keep multiple copies of data in sync, ability to share with trusted contacts, do not lose any data!
- Internet: give me any information/video I want right away!

How are computer systems built?

- Design the system to satisfy the functional requirements
 - Build various software/hardware sub-systems and put them together
 - Rely on and reuse existing software/hardware building blocks
 - Performance engineering
 - Measure performance of system: throughput (transactions per second), response time, capacity (amount of data, number of users)
 - Tune the system to satisfy any unmet performance goals
 - Scalability: does performance improve with increase in resources?
 - Reliability engineering
 - Enhance system components to be robust under failures
 - Provide consistency, atomicity, integrity of data
 - Iterate over the process till satisfied
- 

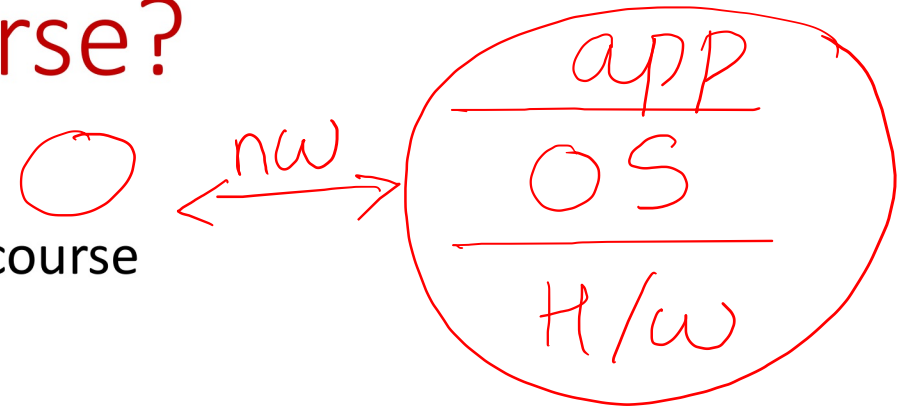
What will you learn in this course?

OS
H/w


- **Week 1:** Introduction to the course
 - What are computer systems?
 - Design principles common across multiple systems
- **Week 1:** Introduction to computer hardware: CPU, memory, I/O devices
 - High level concepts required for designing computer systems
 - Low level implementation details will be covered in a course on Computer Organization and Architecture
- **Weeks 2, 3, 4:** Operating Systems (OS)
 - Operating system is a basic building block on which any user application or computer system runs
 - Understanding system software essential to building user software
 - Virtual machines, containers

What will you learn in this course?

- **Week 5: Networking**
 - Basics of computer networks that are relevant to the course
 - Overview of network security
- **Week 6: End-to-end application design**
 - How to put together a user application from various components
 - Sample designs of real-life computer systems and applications
- **Week 7: Performance measurement and tuning**
 - Performance metrics, measurement, tuning
 - How to make systems more efficient (use given resources well) and scalable (improve performance when more resources given)
- **Week 8: Reliability and fault tolerance**
 - Principles of distributed system for tolerating failures, consistency, atomicity
- **Final goal**: be able to understand the workings of real-life computer systems end-to-end



Pre-requisites

- Introductory course in computer programming
 - Aptitude and interest in computer systems
- This course overlaps with multiple other courses in computer science: operating systems, computer networks, computer organization and architecture, virtualization, cloud computing, distributed systems 
 - No need to have done these courses before, we will cover all topics from basics
 - If you have done some of these courses already, you can still benefit from getting an end-to-end complete picture of computer systems in this course

Summary

- This lecture:
 - Examples of computer systems in real life ✓
 - Functional and non-functional requirements: examples ✓
 - Course outline ✓
- Come up with more examples of computer systems that you interact with every day. What is your expectation from such systems? Do the systems meet your requirements well?