Course Code	CSE630
Department	CSE
Course Name	Graduate Systems (GS)
Credits	4
Course Offered to	UG/PG
Whether the course is to be counted	
If the course is to be counted towards other B.Tech programs(For Ex if a course with CSE no. satisfies the requirement of 32 credits of B.Tech ECE program that students have to do in last 4 semesters, then the drop down answer should be ECE)	
Course Description	This course is an advanced form of computer systems. The objective is to refresh the foundations of computer architecture, operating systems, and computer networks, and then learn the advanced concepts to build real-life high-performance, scalable, reliable, and fault-tolerant distributed systems.

Pre-requisites			**Anti-requisites**
Pre-requisite (Mandatory)	Pre-requisite (Desirable)	Pre-requisite(other)	
			CSE231, CSE232

*Please insert more rows if required

Post Conditions*(For suggestions on verbs please refer the second sheet)			
CO1	**CO2**	**CO3**	**CO4**
Students will be able to explain the basic concepts of operating systems and computer networks	Students will be able to understand the principles of design an end-to-end distributed system application	Students will be able to determine the reasons for the performance bottlenecks and suggest solutions.	Students will be able to design real-life applications, benchmark their performance.

Weekly Lecture Plan

Week Number	Lecture Topic	COs Met	Assignment/Labs/Tutorial
1	**Principles of computer system design:** Modularity, Abstraction, Layering, Virtualization, Hierarchy, Indirection, Parallelism, Caching, Fixed sizing, Indexing, Separation of state from computation, Replication. **Process:** system calls (fork, exec, exit, wait),shell, kernel mode of execution, hardware trap	C01	**Tut 1:** Virtualbox and docker installation **Assignment 1:** related to Linux shell in C
2	**Threads:** Race condition, critical section, locks, hardware atomic instructions, Multicore scheduling: design choices	CO1	**Tut 2:** Multithreading using pthreads
3	**Virtualization:** Virtual machines (trap-and-emulate, paravirtualization, full virtualisation), Containers (namespaces, cgroups); **Memory virtualization:** Address translation, MMU, TLB, Extended page tables (EPT), Shadow page tables, design decisions	CO1, CO3	**Tut 3:** Performance analysis using "perf" tool
4, 5, 6	**Introduction to computer networks:** The end-to-end argument, Internet routing: IP addressing, routing protocols, SDN concept; **Transport protocol:** TCP, UDP, SCTP, Bandwidth delay product, congestion control, end-to-end delays **Application layer protocols:** HTTP, data serialization formats (e.g., json, protobuf, flatbuffers), RPC (gRPC)	CO1, CO2	**Tut 4:** Socket programming **Tut 5:** Network debugging tools **Tut 6:** Packet capture with Wireshark/tshark/tcpdump **Assignment 2:** Related to socket programming and performance analysis
7, 8	**Multi-threaded server design:** synchronization, atomicity, deadlocks, producer-consumer), event-driven multi-threaded server (e.g., nginx); **Multi-tier application design:** Storage options (RDBMS, NoSQL), API design (REST, RPC, publish-subscribe); **End-to-end application design:** Use cases such as e-commerce, video streaming systems, social networking, instant messaging, etc.	CO1, CO2	**Tut 7:** Kubernetes (with service mesh) installation (minikube) **Tut 8:** Interprocess communication using FIFO **Project:** Starts in Week 7
9	**Deployment of computer systems:** cloud deployment, cloud orchestration, network architecture (firewall, DMZ); **Performance measurement and tuning:** performance metrics, load testing, performance profiling	CO2, CO3, CO4	**Project milestone 1:** (a) Demo on initial system setup and configuration (b) Report on suggested additional features
10, 11	**Optimizations:** caching in computer systems; **Scalability:** multicore speedup, load balancer design, building efficient systems using the given resources; **Reliability and fault-tolerance:** replication and consistency, atomicity	CO2, CO3, CO4	
12, 13	**Distributed transactions:** sharding, consistent hashing, 2-phase commit; **Putting all pieces together:** Case studies of distributed systems design	CO2, CO3, CO4	**Project milestone 2:** (a) Final project demo (b) Final report

*Please insert more rows if required

Weekly Lab Plan

Week Number	Laboratory Exercise	COs Met	Platform (Hardware/Software)
Course does not have a lab component.			

*Please insert more rows if required

Assessment Plan

Type of Evaluation	% Contribution in Grade
Project	**35**
Programming assignments	**20**
Mid-sem	**15**
End-sem	**30**

*Please insert more row for other type of

Resource Material

Type	Title
Books	1. Operating Systems: Three Easy Pieces, by Remzi H. Arpaci-Dusseau and Andrea C. Arpaci-Dusseau 2. Computer Networking: a Top Down Approach (8th edition), by Jim Kurose and Keith Ross 3. Principles of Distributed Computing, by Roger Wattenhofer 4. Computer Systems: A Programmer's Perspective, by Randal E. Bryant and David R. O'Hallaron 5. Cloud Native Networking Deep-Dive, by Chander Govindarajan, Priyanka Naik
Web references	1. Virtual Machine Monitors: Current Technology and Future Trends 2. Virtual Machine Monitors, by OSTEP 3. Containers and Cgroups 4. Linux namespaces 5. kvm: the Linux Virtual Machine Monitor 6. Distributed systems reading list https://github.com/theanalyst/awesome-distributed-systems 7. Relevant research papers for real-world use cases
Homework/Tutorials/Assignments	**Tutorials:** 0: Linux familiarity: using the shell, creating a VM, compiling C program --> recall SP refresher & OOPD 1: Virtualbox and docker installation 2: Multithreading using pthreads 3. Performance analysis using "perf" tool to monitor hardware performance monitoring units (PMUs) 4: Socket programming 5: Network debugging tools 6. Packet capture with Wireshark/tshark/tcpdump 7: Kubernetes and service mesh installation (minikube) 8: Interprocess communication using FIFO **Smaple assignments:** 1: Write a simple Linux shell in C 2: Performance analysis of Multi-threaded & Event-driven TCP server
Project	**Sample examples:** Real-life use case code is given. Students will deploy it over docker platform, implement few additional features, analyze performance, enable scale and reliability. **Example real-life applications:** Video streaming service, Social media networking, Online storage of files and documents, Online banking, E-commerce service, Swarm service.

References:

1. An Open-Source Benchmark Suite for Microservices and Their Hardware-Software Implications for Cloud & Edge Systems, ASPLOS 2019
2. Starter code-base available here