Operating Systems

Lecture 4

Time Sharing Operating Systems

Ms. Minal Rajwar

Introduction to Time Sharing Systems

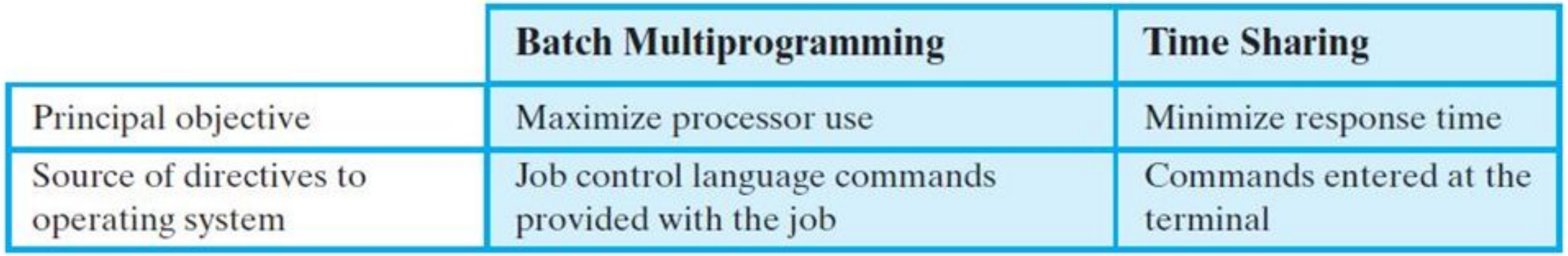
- **Time-Sharing Systems**
- In a time-sharing system, multiple users simultaneously access the system through terminals, with the OS interleaving the execution of each user program in a short burst or quantum of computation.

- Thus, if there are n users actively requesting service at one time, each user will only see on the average 1/n of the effective computer capacity, not counting OS overhead.

- However, given the relatively slow human reaction time, the response time on a properly designed system should be similar to that on a dedicated computer. Both batch processing and time sharing use multiprogramming.

- The key differences are listed below

Introduction to Time Sharing Systems



	Batch Multiprogramming	Time Sharing
Principal objective	Maximize processor use	Minimize response time
Source of directives to operating system	Job control language commands provided with the job	Commands entered at the terminal

- One of the first time-sharing operating systems to be developed was the Compatible Time-Sharing System (CTSS), developed at MIT by a group known as Project MAC.

- The system was first developed for the IBM 709 in 1961 and later transferred to an IBM 7094.

Introduction to Time Sharing Systems

- The system ran on a computer with 32,000 36-bit words of main memory, with the resident monitor consuming 5000 of that. When control was to be assigned to an interactive user, the user's program and data were loaded into the remaining 27,000 words of main memory.

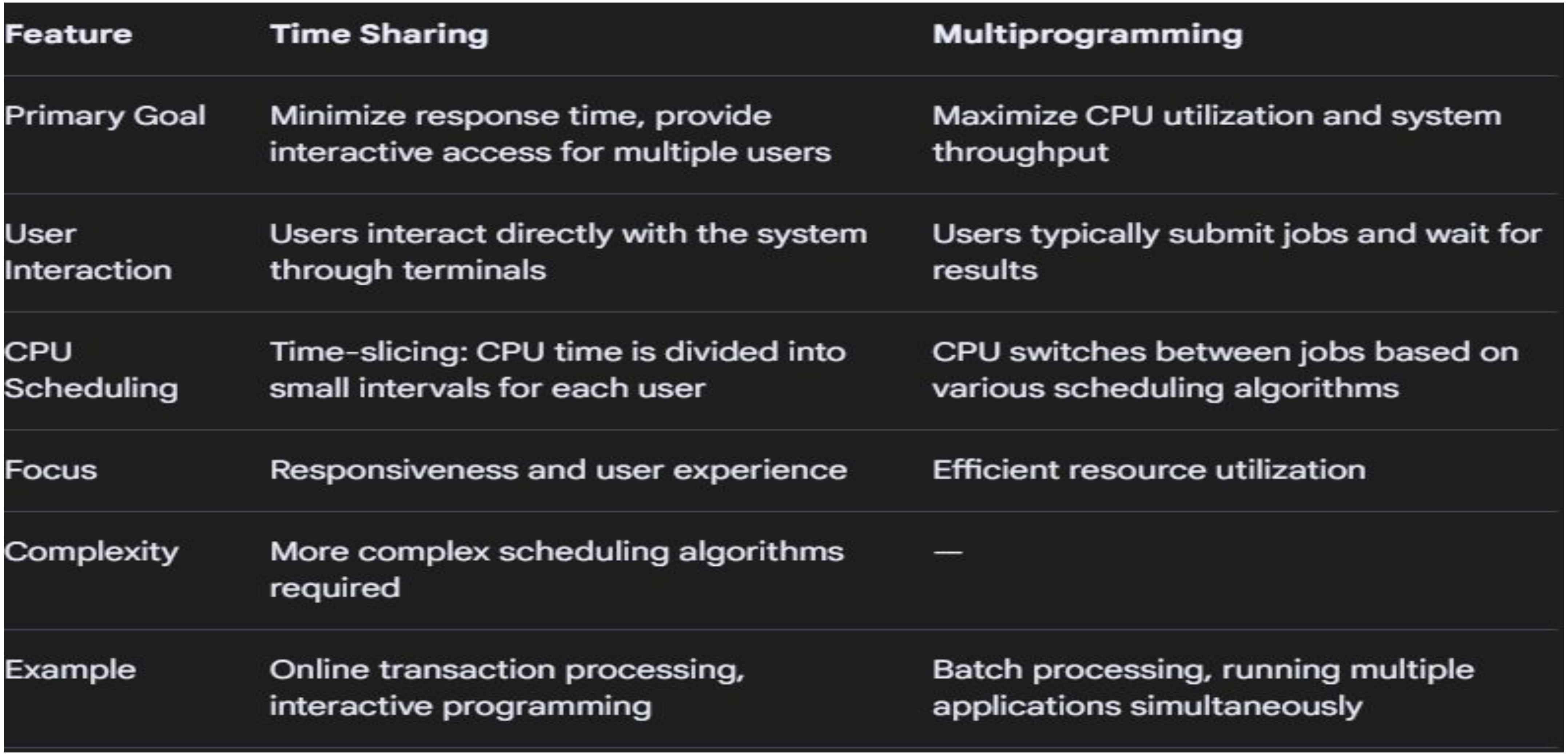
- A program was always loaded to start at the location of the 5000th word; this simplified both the monitor and memory management. A system clock generated interrupts at a rate of approximately one every 0.2 seconds. At each clock interrupt, the OS regained control and could assign the processor to another user. This technique is known as time slicing.

- Thus, at regular time intervals, the current user would be preempted and another user loaded in. To preserve the old user program status for later resumption, the old user programs and data were written out to disk before the new user programs and data were read in. Subsequently, the old user program code and data were restored in main memory when that program was next given a turn.

Need for Time Sharing

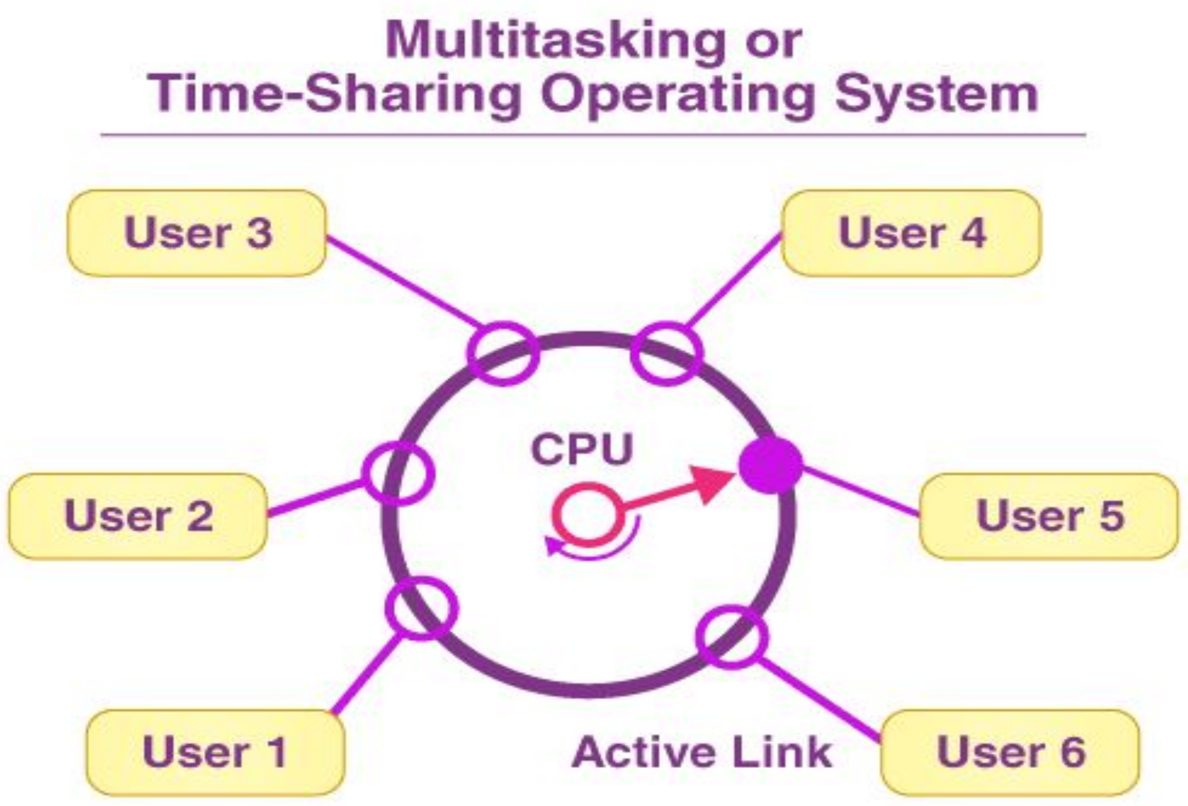
- Batch and multiprogramming systems lacked **interactivity**.
- Users had to wait long for output.
- Time-sharing was introduced in the 1960s to reduce waiting time and **enable interaction**.

Time Sharing vs. Multiprogramming



Feature	Time Sharing	Multiprogramming
Primary Goal	Minimize response time, provide interactive access for multiple users	Maximize CPU utilization and system throughput
User Interaction	Users interact directly with the system through terminals	Users typically submit jobs and wait for results
CPU Scheduling	Time-slicing: CPU time is divided into small intervals for each user	CPU switches between jobs based on various scheduling algorithms
Focus	Responsiveness and user experience	Efficient resource utilization
Complexity	More complex scheduling algorithms required	—
Example	Online transaction processing, interactive programming	Batch processing, running multiple applications simultaneously

Time Sharing System Architecture



- Each user connects via terminal.
- Jobs go to a **ready queue**
- Scheduler selects jobs based on **Round Robin**
- Output sent back to the respective terminal

Advantages of Time Sharing

- **Interactive**: Users get quick feedback.
- **Fair**: Every process gets CPU time
- **Efficient**: High CPU utilization
- Supports **multi-user systems**

Limitations / Challenges

- Too frequent **context switching** reduces efficiency
- Requires **careful tuning** of time quantum
- Short quantum → high overhead
- Long quantum → low responsiveness

Real-World Examples

- UNIX/Linux terminals
- **Windows/macOS** multitasking
- **Mainframes** in the 1970s with 100+ terminals
- **Cloud computing**: Containers, VMs share CPU via time-slicing

Quiz & Discussion

- What is a time-sharing system? Give one real-world use case of time-sharing.

- Why do we need CPU scheduling?