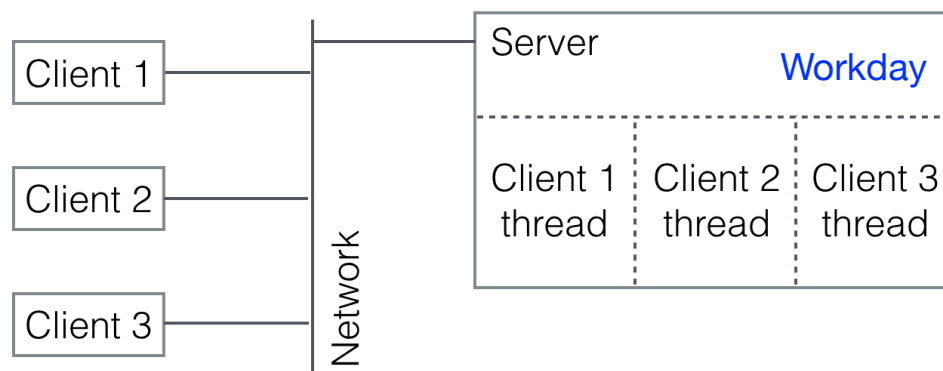


Concurrent Programming (I)

Overview

- A **concurrent program** is a program designed to have *two or more* execution contexts. Such a program is said to be **multithreaded**, since more than once execution context can be *active simultaneously*.
- Although the programs used in this course so far are all single-threaded, multithread programming is widely used in daily life.
- Do you know any examples of concurrent programming?
 - One typical example is the networks. Take Workday Student for instance, what we use on a web browser is the front end of Workday system. We call it the client side. There is a database system on the other side of the network managing all kinds of information, such as students' information, faculty's information, courses' information, and etc. We call the database system the server side, which is the back end of the Workday system.
 - Every time a student logs into Workday, a new thread is created for the student. So that the server side must be able to handle multithreads.
 - Please note that all these threads share the same database systems. A database consists of a number of tables. A table is actually a piece of memory. It is very possible that several users want to access the same piece of memory at the same time: read the information or modify the information stored at that piece of memory (the remaining seats of a course).
 - How to make those users can access to the same piece of memory successfully and get the correct information. These make concurrent programming challenging but interesting.



- Two fundamental problems:
 - Race condition
 - Deadlock