CISC 3120

C20: Asynchronous I/O, Threads, and Networking

Hui Chen

Department of Computer & Information Science

CUNY Brooklyn College

Outline

- Recap
  - I/O: network & File
  - Application: GUI
  - Problem?
- Duplex communications
- Nonblocking I/O
- Asynchronous I/O
- Threads and concurrency

Communication Services

- Simplex
- Half duplex
- Full duplex

How do we achieve full-duplex?

- If underlying communication channel is full duplex, how do we take advantage of it?
- Two programs (processes) at each side
  - One does receiving
  - One does transmitting
- One program (process), two threads at each side
  - One does receiving
  - One does transmitting

Process and Thread

- Processes and threads exist to support multitasking
- Process
  - A program in execution and associated data structures (e.g., a process control block)
  - A process may have one or more threads of execution

Process and Thread: Comparison

	Process	Thread
Address space	A process usually has its own address space (implication: two processes cannot access each other's variables)	Multiple threads of a process shares the same address space (implications: they can access the process's variables)
States and Controls	Processes usually have larger set of states and supporting data structure	Multiple threads of a process shares share the process states and other resources, in addition to memory
Interfacing	Inter-process communication (IPC)	Share the process memory
Context-switch	Generally slower than threads	Generally faster than processes when switching between different processes

Multithreading in Java

- Two approaches
  - Implementing the Runnable interface
  - Extending the Thread class
- Prefer to implementing the Runnable interface
- A few related classes
  - Timer, TimerTask, Runnable, Thread, …

Concurrency in JavaFX

- Service
- Task

Multithreading: Text-based App Example

- One thread deals with InputStream of a Socket
- One thread deals with OutputStream of the Socket

Multithreading: JavaFX App Example

- One thread deals with listening, accepting and InputStream of a Socket
- An EventHandler deals with OutputStream of the Socket

Questions

- Channel
  - Simplex, half-duplex, full-duplex
- Process and threads
- Application examples
- Application
  - Client/server, peer-to-peer, hybrid

Types of I/O

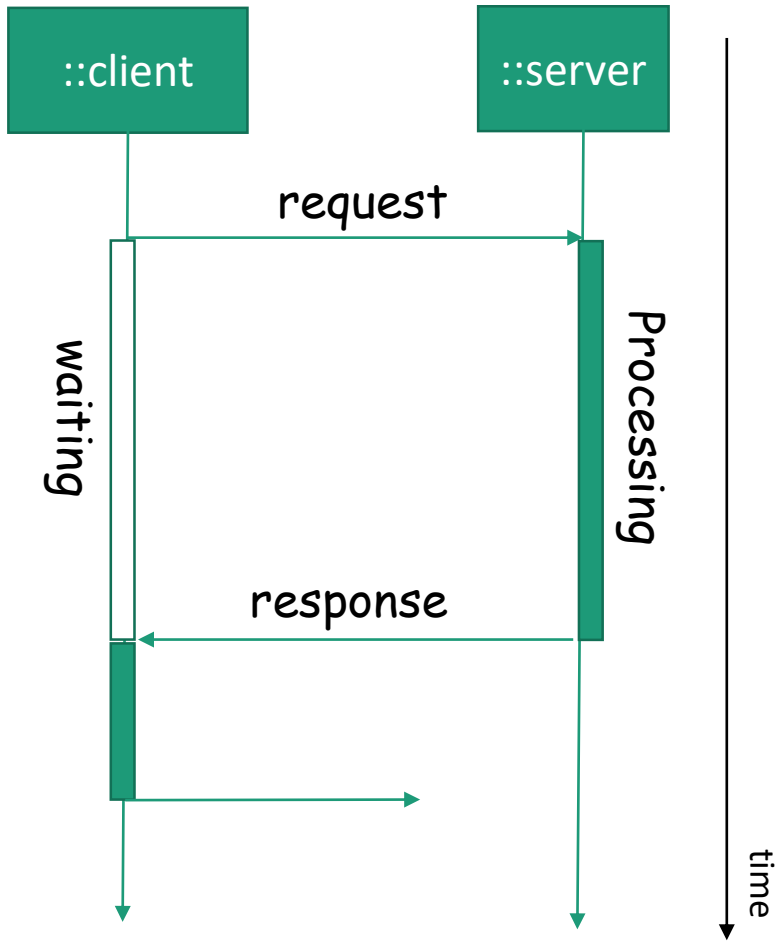
- Blocking I/O vs. Non-blocking I/O
  - Blocking: thread does not progress until I/O completes or fails (wait indefinitely for I/O)
  - Non-blocking: thread progresses only wait a timeout for I/O
- Synchronous I/O vs. Asynchronous I/O
  - Synchronous I/O: thread waits for I/O to complete or fail
  - Asynchronous I/O: thread is notified by I/O completion, progress, or failure

Blocking and Non-blocking I/O

- Use scenario
  - Blocking
    - I/O is reliable (we know the data will arrive or a failure)
  - Non-blocking
    - I/O is unreliable (there is no guarantee the data will arrive at all, or before a deadline)

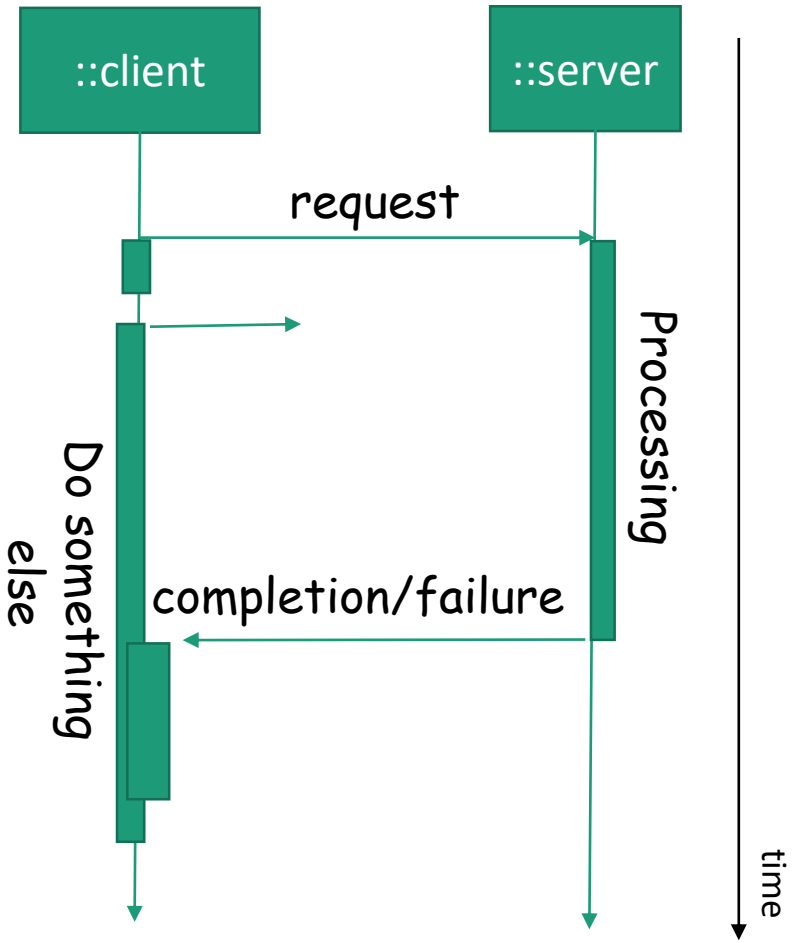
Synchronous I/O

- A process/thread must wait for I/O to complete/fail
  - Examples
    - read
    - write
    - connect/accept
- Application scenario
  - Example
    - Click UI, wait for data, refresh UI
  - What if the user wishes the app to do something else while waiting?



Asynchronous I/O

- A process/thread immediately returns upon issuing a request for I/O
- The process/thread can poll I/O status or being notified failure/completion
- Application scenario
  - Example
    - Click UI, do something else (continue event loop), refresh UI when being notified
  - the user can make the application more responsive



Non-blocking I/O in Java

- Java I/O streams are blocking
  - In java.io package
  - Since JDK 1.0
- Java Channel interface provides non-blocking I/O
  - Channel in java.nio package
  - Since JDK 1.4
  - Often used with Java Selector
  - Often referred to as a part of Java NIO

Asynchronous I/O in Java

- AsynchronousChannel interface
  - AsynchronousChannel in java.nio package
  - Since JDK 1.7
  - Often referred to as a part of Java NIO.2

I/O Types: Discussion

- Blocking, non-blocking, synchronous, and asynchronous I/O
  - A non-blocking I/O is not necessarily asynchronous
  - Synchronous I/O is typically blocking
  - Asynchronous I/O is typically non-blocking
- Concurrency and I/O
  - With concurrency (e.g., Runnable, Threads, Task, Service …) support and the Observer pattern, we may emulate non-blocking or asynchronous I/O
  - With added complexity to the application

Questions

- Types of I/O
  - Blocking vs. Non-blocking
    - A non-blocking I/O is not necessarily asynchronous
  - Synchronous vs. Asynchronous
    - Synchronous I/O is typically blocking
    - Asynchronous I/O is typically non-blocking

I/O and Serialized Objects

- Object streams
  - File I/O
  - Network I/O
- Object serialization

Assignments

- Practice assignments
- Project 4
- Review
  - Review #4 and Take-Home Test 4