I/O Models

Modern Operating Systems, by Andrew Tanenbaum — Chap. 5.2

Unix Network Programming, 3rd Edition, by Richard Stevens — Chap. 6

Example: UNIX Virtual File System (VFS)

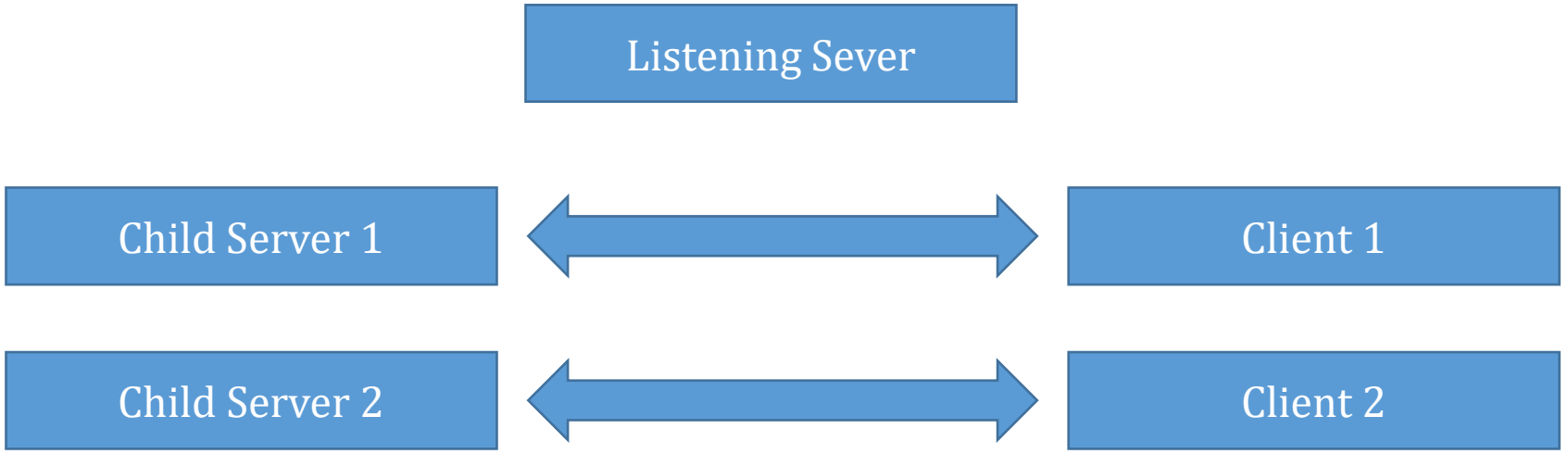
- Standard set of operations
  - open, read, write, seek, fctl, close
- Blocking read/write
  - Invoke read() system library function / system call and wait for it to finish
  - When control returns, you can assume the read is complete
- Note: Buffered I/O writes are a case where the abstract model is "leaky"
  - When control returns, you assume the write is complete
  - But in reality, your write is in a buffer… not completely on file yet

Types of Concurrency

- True Concurrency (e.g. processor)
  - Multi-processor machines – multiple threads executing simultaneously
  - Uni-processor machines with kernel threads
    - Even though multiple threads execute simultaneously, programmer has no control over context switching, so must ASSUME threads execute simultaneously
- Apparent Concurrency
  - Single process multiplexes among multiple clients
  - User level threads
  - Programmer has control over when context switches occur

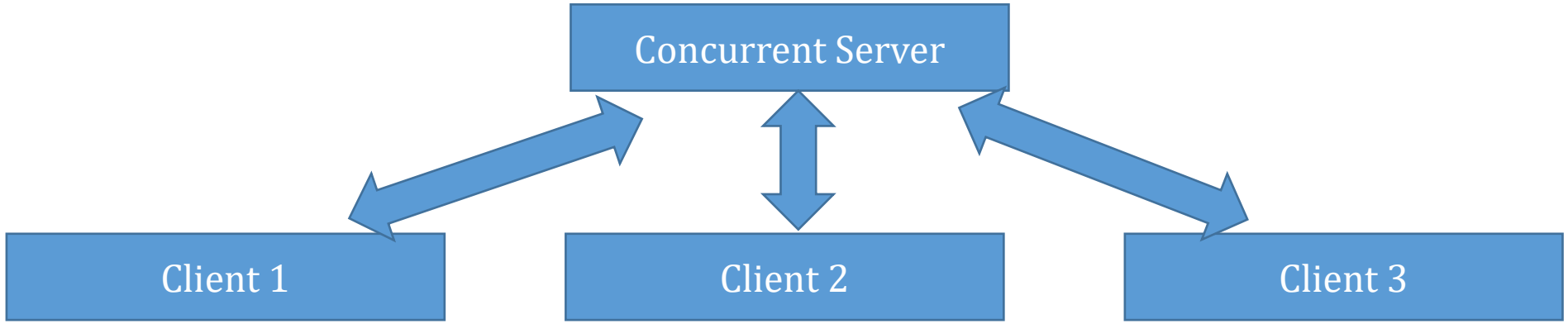
Example of "True Concurrency"

- Web server - "Listening" daemon for new requests from clients
- When a request is received, Listening daemon forks a child server
  - Child server satisfies new client request(s) and exits when done



Example of "Apparent Concurrency"

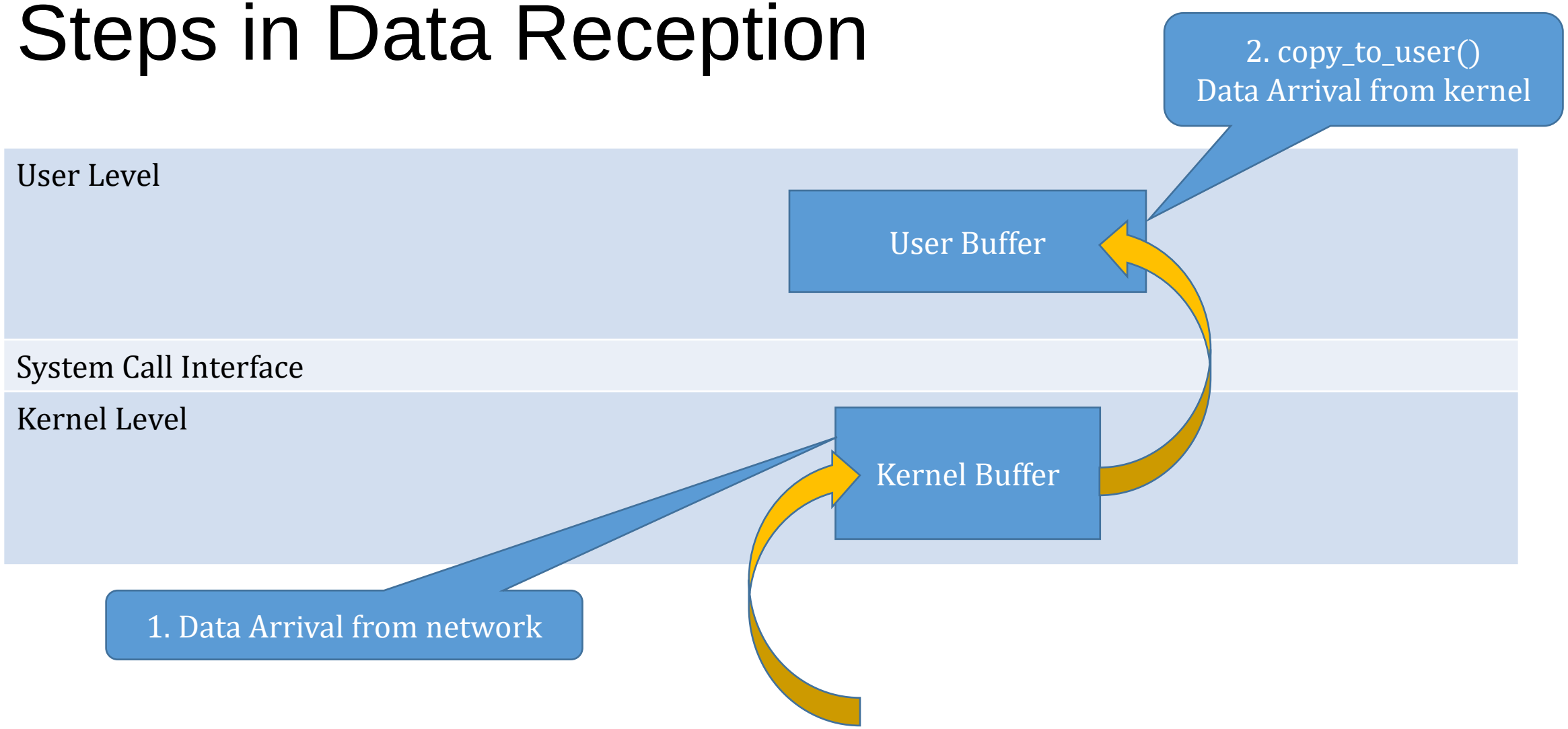
- Concurrent Server Daemon creates client request queue
- Satisfies each client request in the queue sequentially



I/O Models

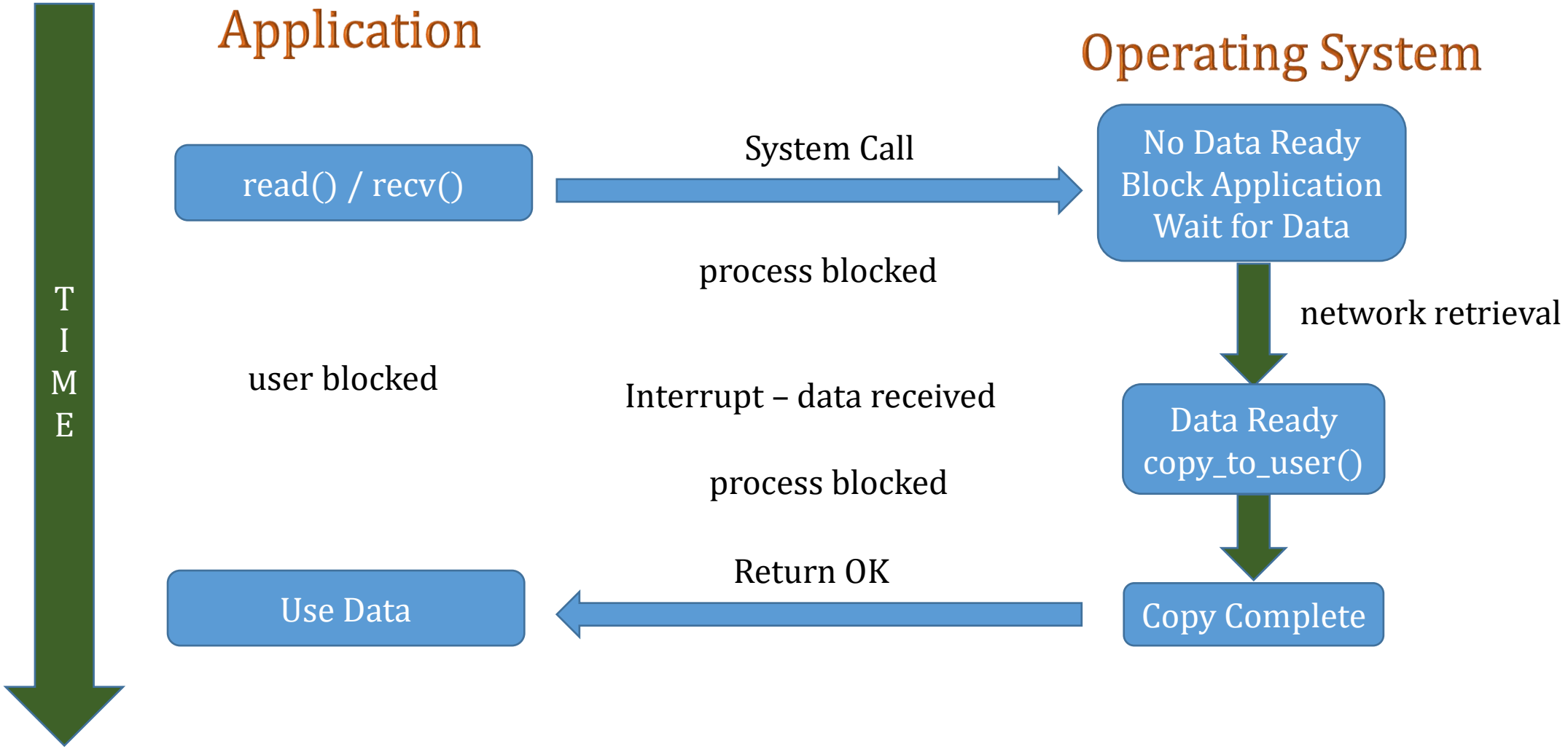
- Blocking I/O
- Non-Blocking I/O
- Signal Driven I/O
- Asynchronous I/O
- I/O Multiplexing – select()

Steps in Data Reception

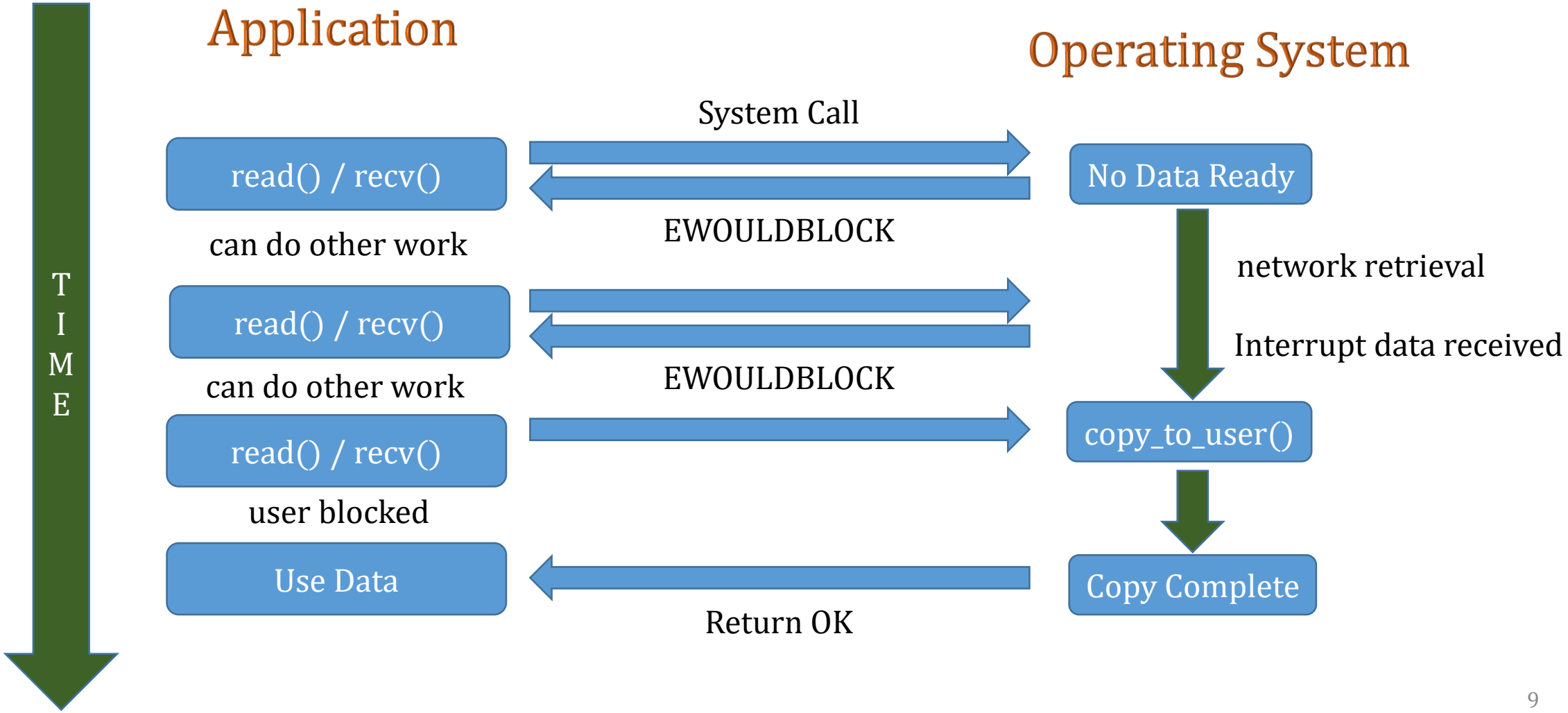


Overheads: Context Switching, Data Copying

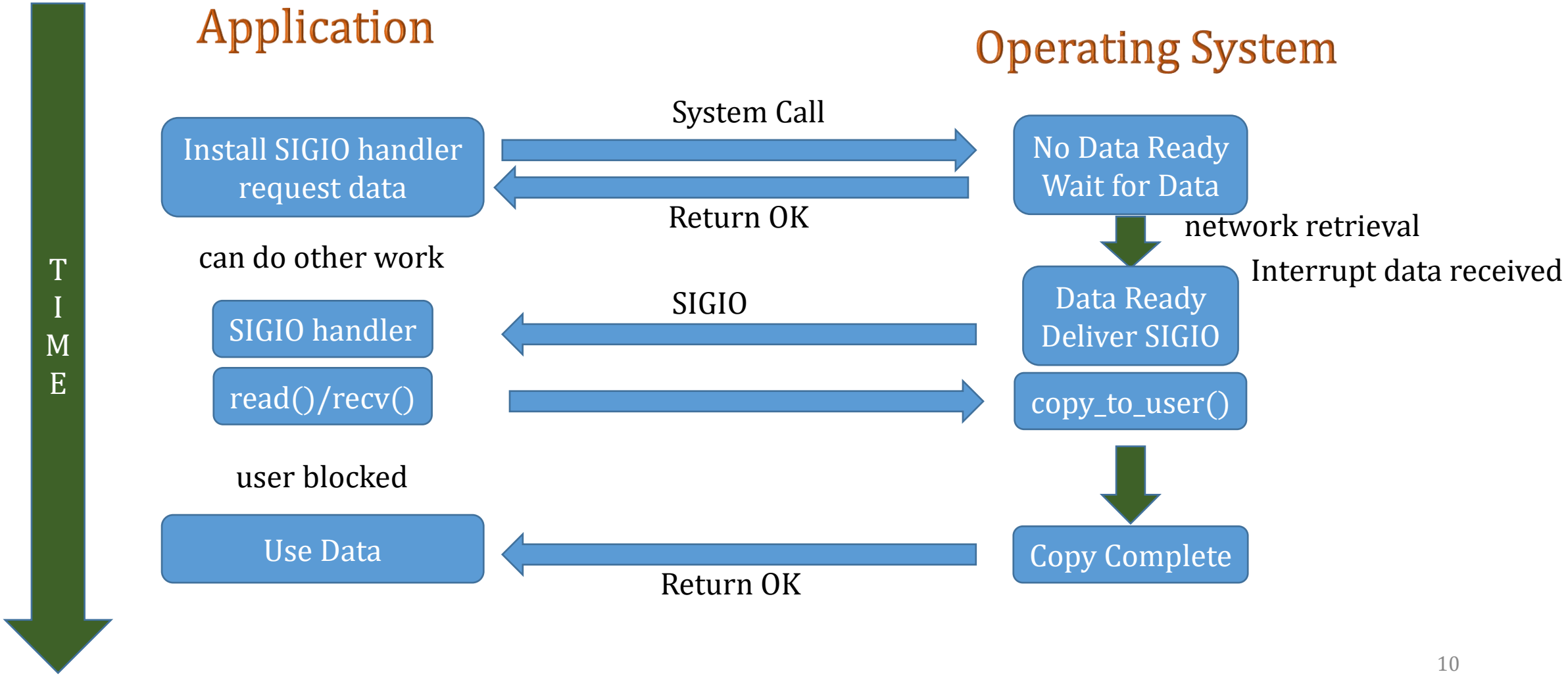
Blocking I/O Model



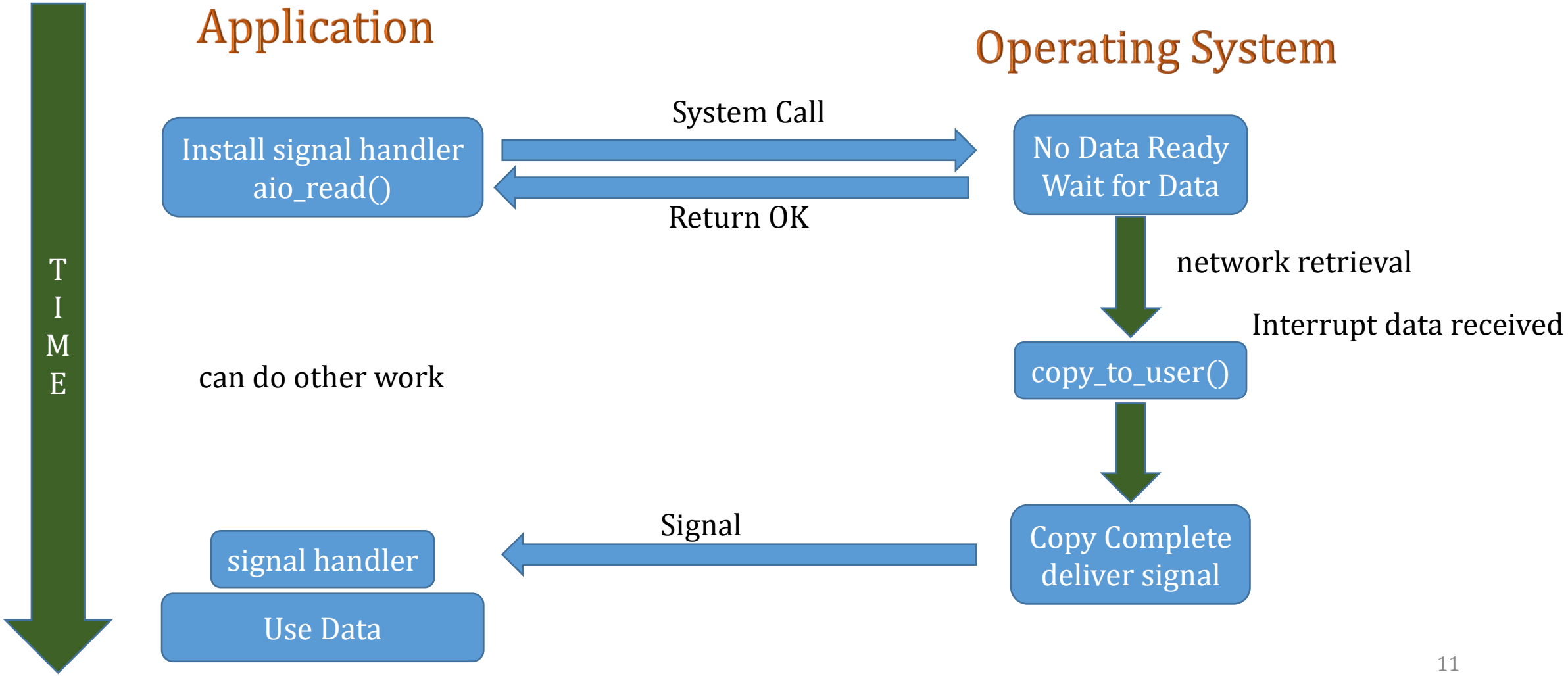
Non-Blocking I/O Model (polling)



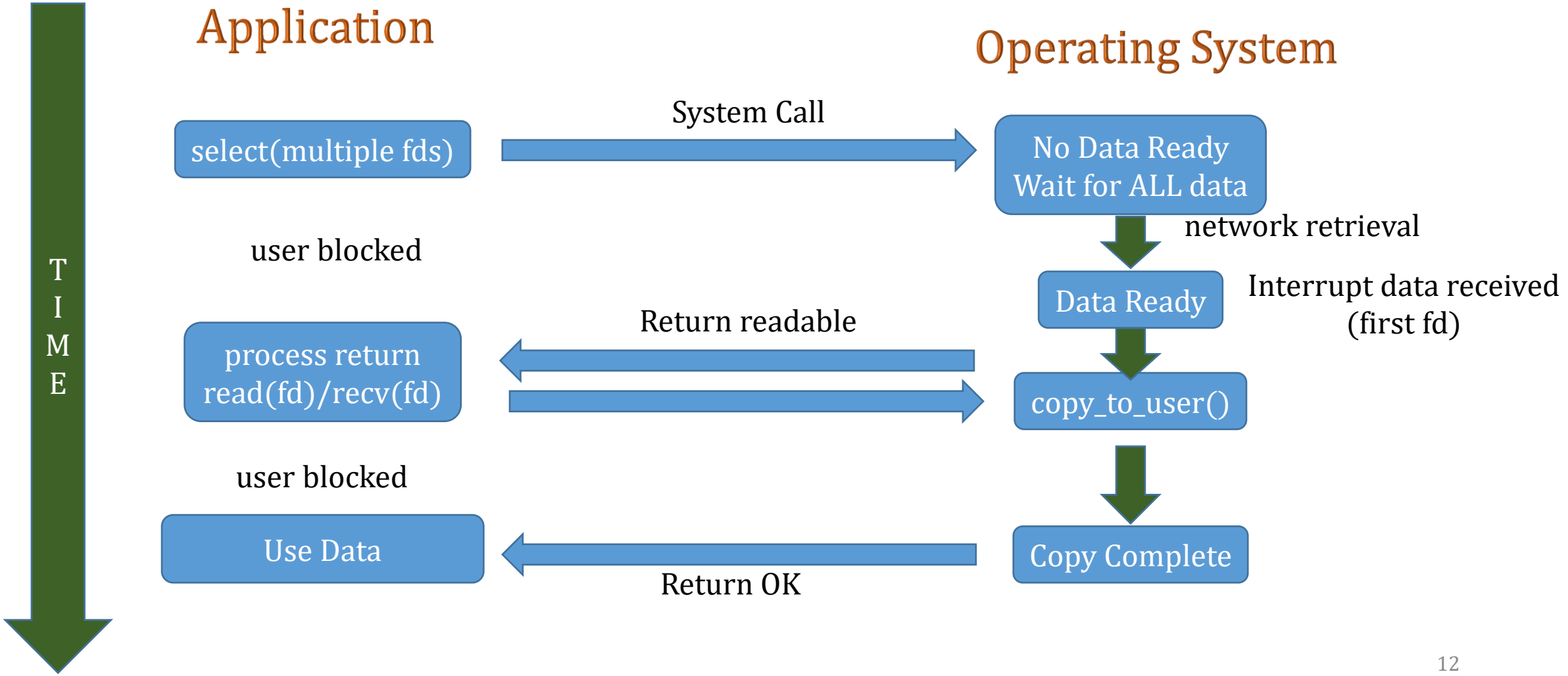
Signal Driven I/O Model



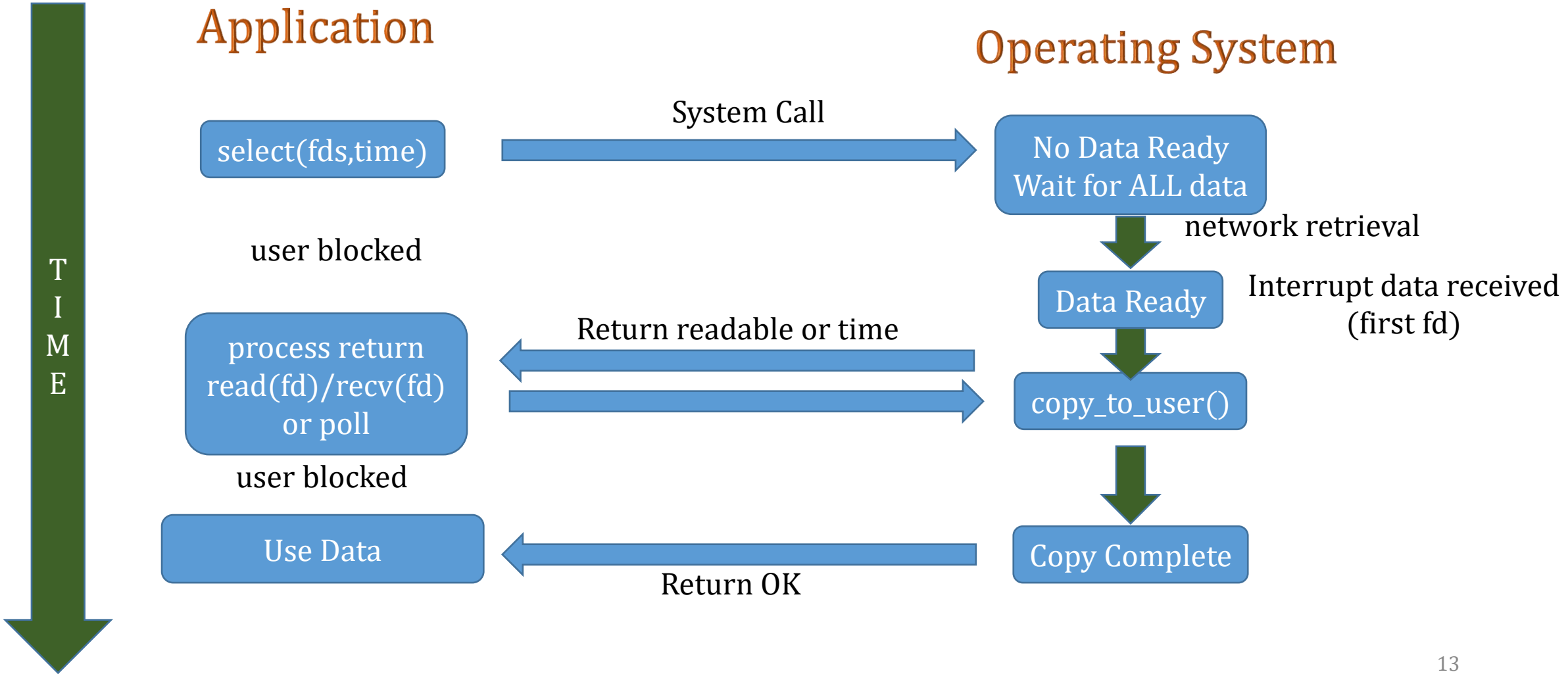
Asynchronous I/O Model



I/O Multiplexing



I/O Multiplexing (with wait time)



I/O Multiplexing Example

- Concurrent Server Daemon creates client request queue
- Satisfies each client request as soon as data is available