

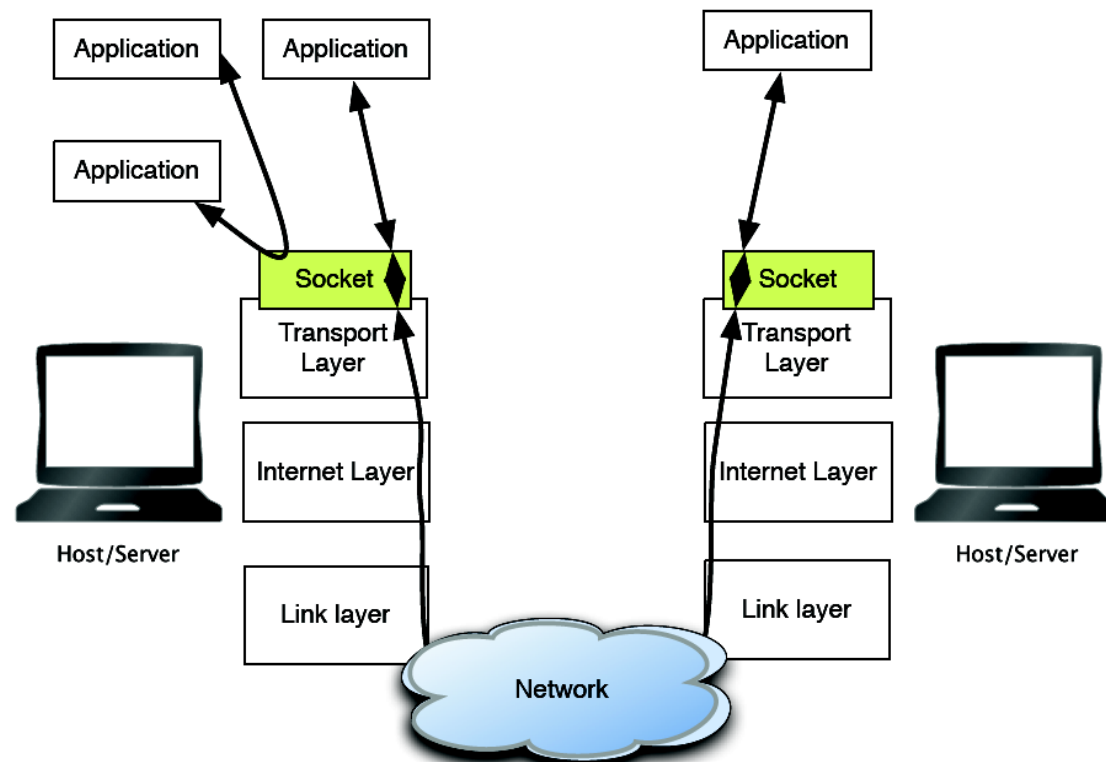
# Network Programming (Sockets)

**Redes e Serviços**

**Licenciatura em Engenharia Informática  
DETI-UA**

# Sockets (1)

- Inter-process communication mechanism
  - Either local or remote processes
- Provide an abstraction for processes to exchanging information
  - Follows a client/server paradigm.



# Sockets (2)

- A Socket is identified by
  - ♦ Family: AF\_INET (IPv4), AF\_INET6 (IPv6) and many other less common.
    - Defines the address structure.
    - Defines also the communications layer (e.g. IP version).
  - ♦ Type: Determines what transport protocol is used.
    - UDP – Connectionless (SOCK\_DGRAM).
    - TCP – Connection oriented (SOCK\_STREAM).
    - RAW – Direct access to a layer of the stack (SOCK\_RAW).
      - Allows to send and receive crafted packets.
      - e.g. the ping command (ICMP packets).
  - ♦ Address: local address(IP or path)
    - Also remote address if connection oriented
  - ♦ Port: Local port 0-65535
    - Also remote port if connection oriented
- Restriction
  - ♦ 1 socket per Address, per Port, per Protocol, per Family, per Host



# Sockets (3)

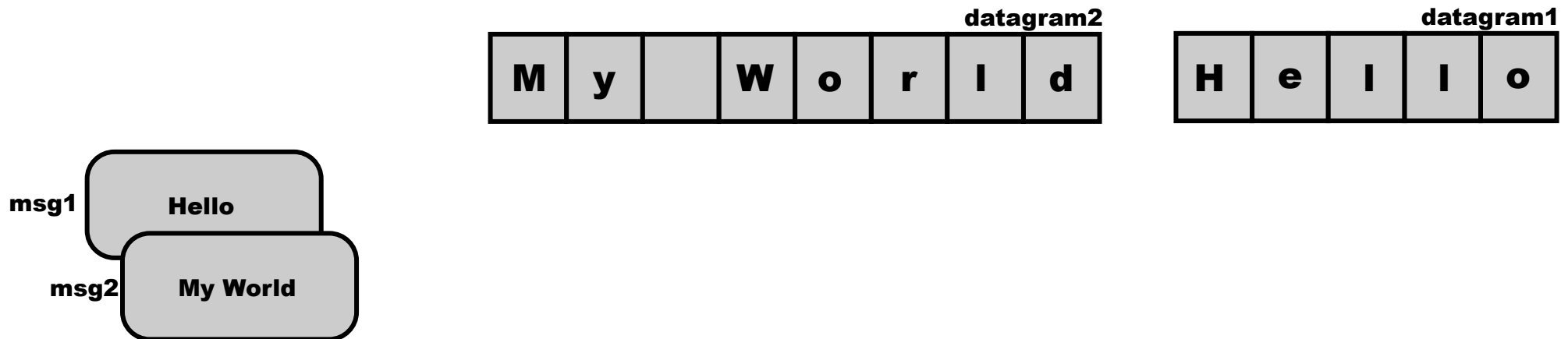
- AF\_INET/AF\_INET6 families
  - Allows communication between processes on any IP/IPv6 enabled machine.
  - Endpoints can be on local or remote machines
    - ➔ 127.0.0.1 or ::1 for the localhost
- A Socket must be “Bound” to a local IP/PORT
  - Sockets can be bound to a specific address or to any address
    - ➔ e.g. 192.168.0.1 (only listens in this address)
    - ➔ e.g. 0.0.0.0 (listens in all active addresses and broadcast)
  - bind() method can be used to associate a Socket to a local IP/Port.



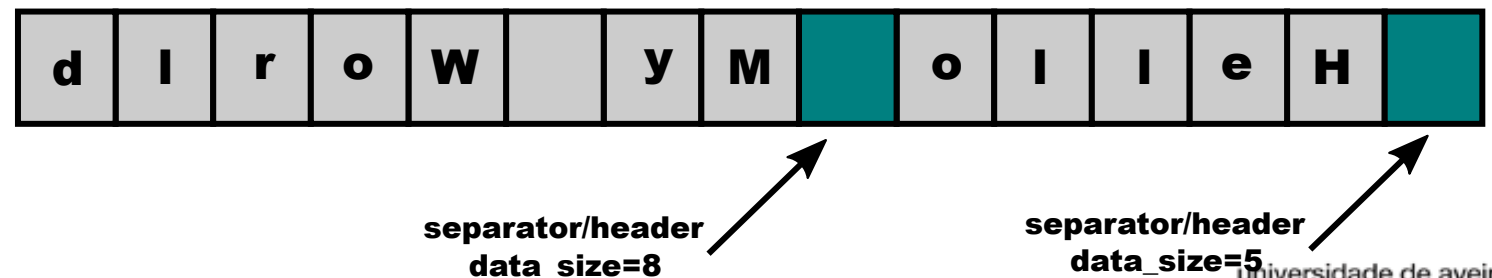
# Byte Stream vs. Datagrams

- TCP needs application-level message separators (headers).
  - Must contain size information of each “independent” data chunk in the bytestream.

## Datagrams (Connection-Less)



## Byte Stream (Connection-Oriented)

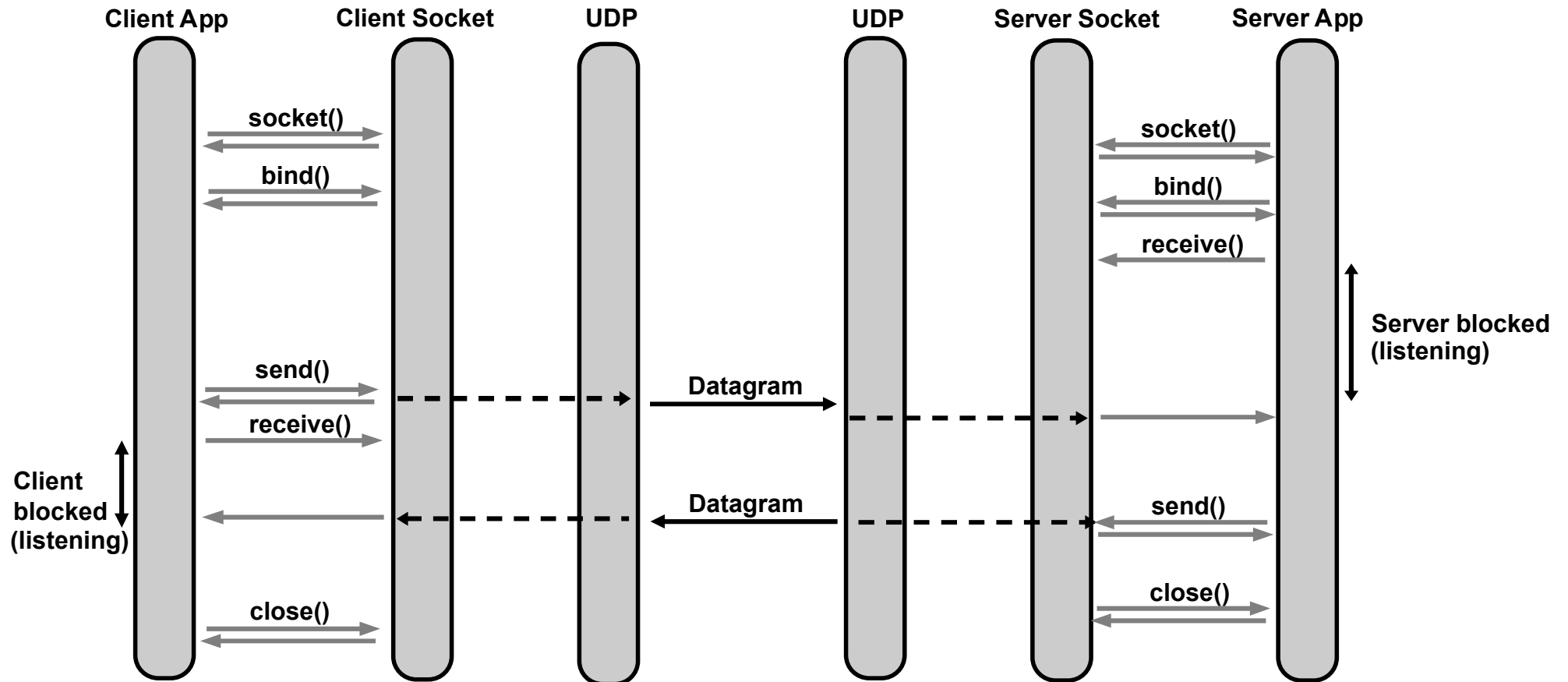


# Socket IO / Blocking

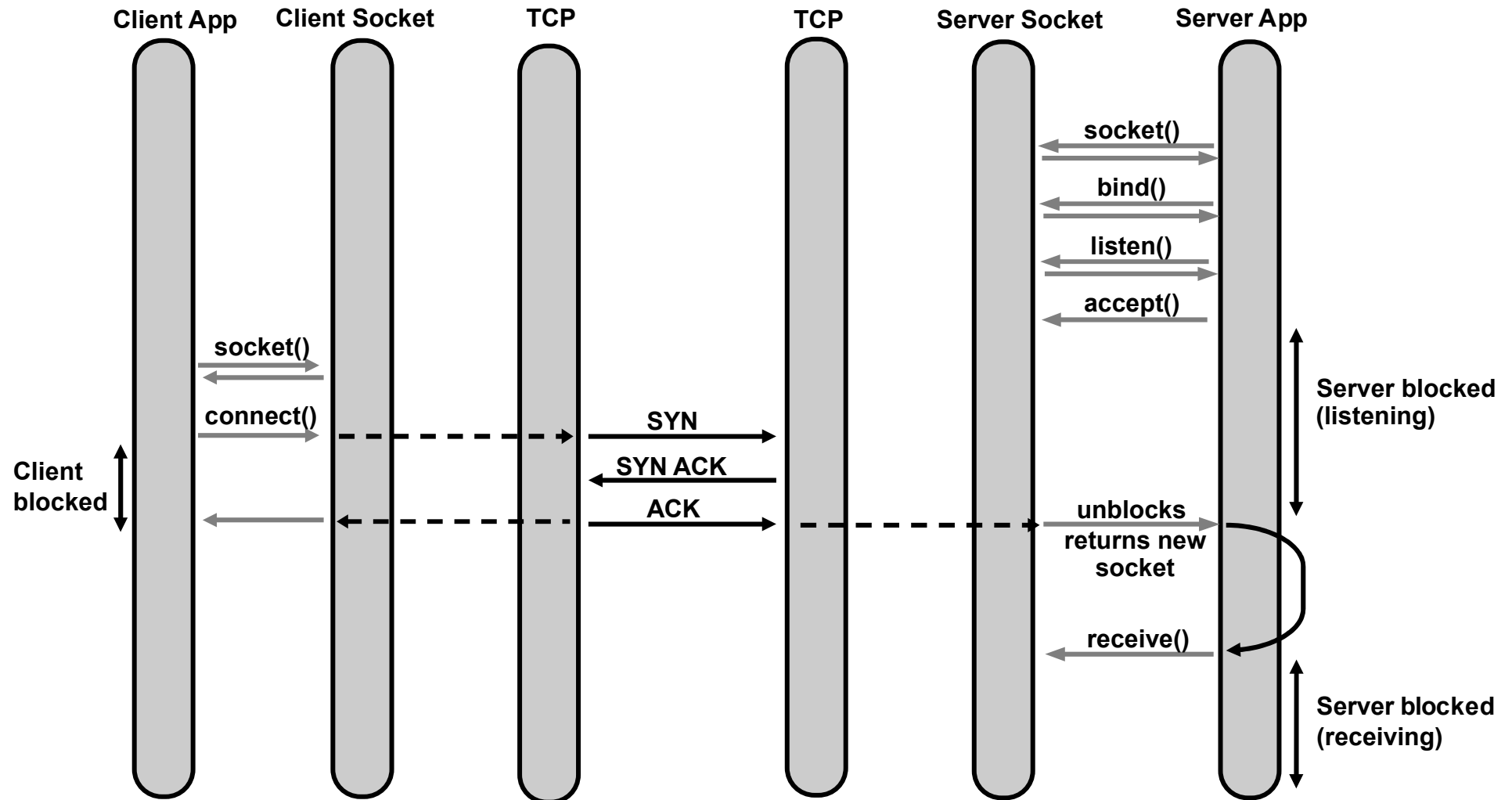
- Socket Operations are Blocking
  - ♦ They block until:
    - Packet is fully sent,
    - Client is accepted,
    - Packet is received,
    - Etc...
  - ♦ Can be set to non-blocking.
    - Program flow must take that in consideration.



# Connection-Less

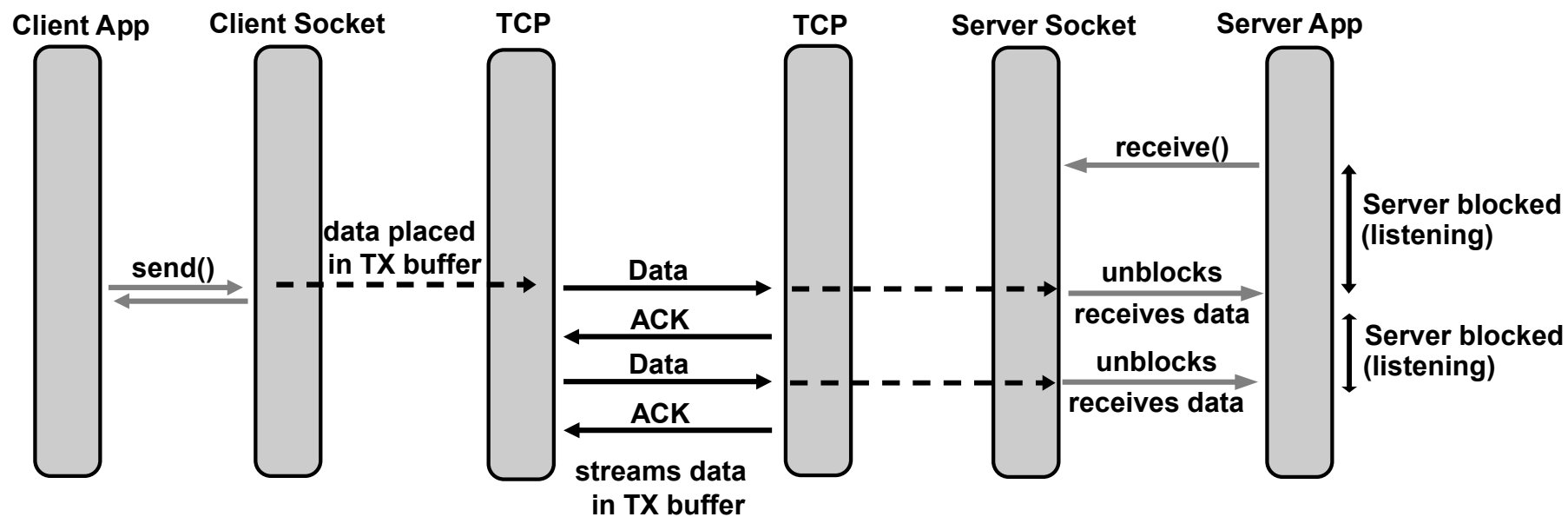


# Connection-Oriented (1)

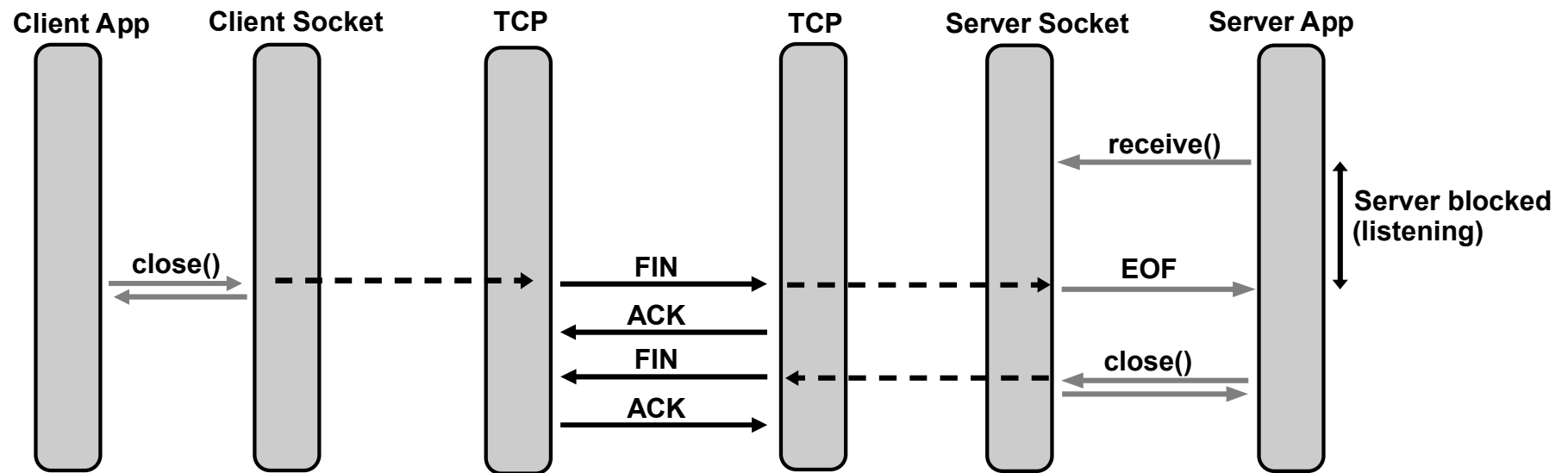




# Connection-Oriented (2)



# Connection-Oriented (3)



# Non-Blocking IO

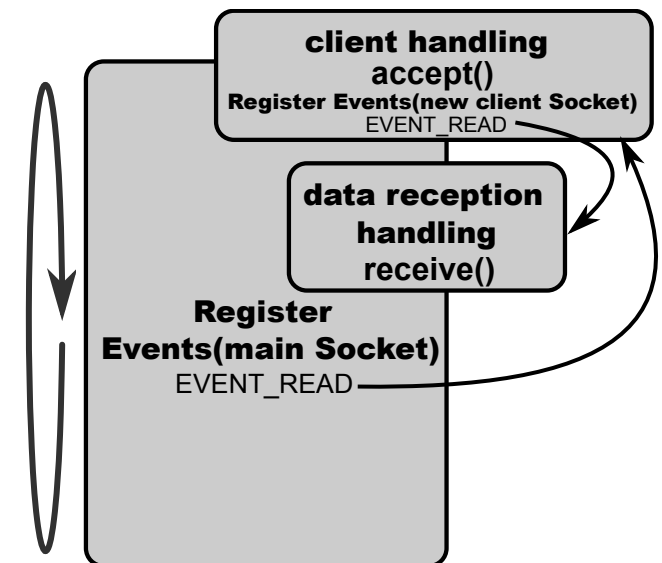
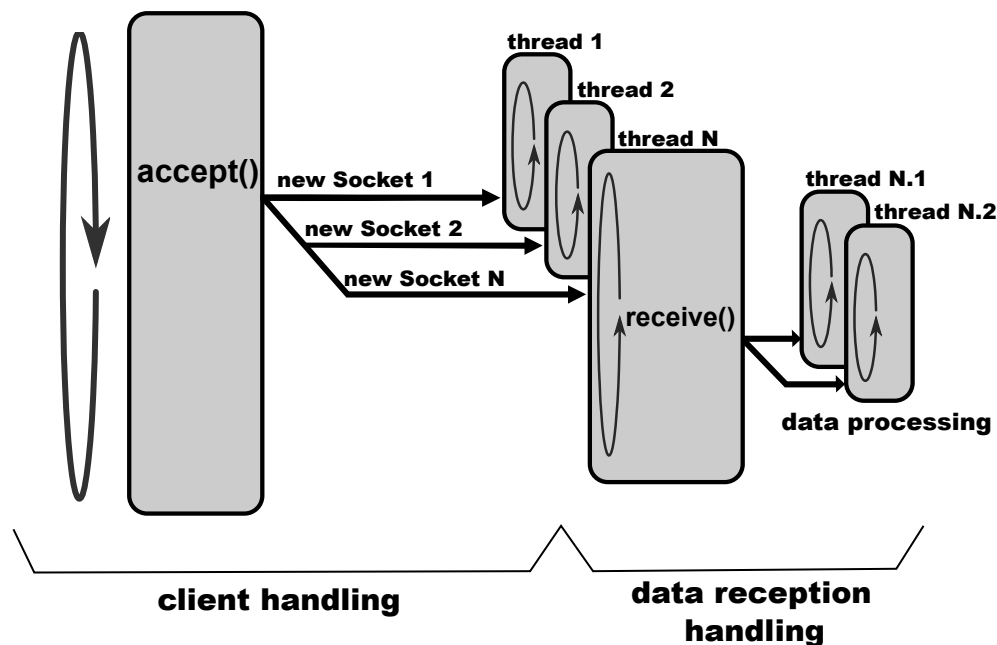
- Solutions for Socket Operations Blocking

- Threads

- Multiple parallel process can be used to process simultaneous connections.
- Most solutions used (and still use) IO operations with multiple threads.

- Selector

- Socket is set to non-blocking.
- Actions are performed upon the detection of predefined socket events (e.g., EVENT\_READ – data available to read).



# Socket Timeouts

- A socket can be in one of three modes:
  - ♦ Blocking,
    - Default state.
  - ♦ Non-blocking,
  - ♦ or Timeout.
- In blocking mode, operations block until complete or the system returns an error (such as connection timed out).
- In non-blocking mode, operations fail if they cannot be completed immediately.
  - ♦ Selects can be used to know when and whether a socket is available for reading or writing.
- In timeout mode, operations fail if they cannot be completed within the timeout specified for the socket (they raise a timeout exception) or if the system returns an error.



# Data Format



# Textual vs. Binary Structure

- Textual

- Pure text (format based on CSV, TSV, newline, ...), HTML, JSON, XML.
- Larger messages and higher processing times.
  - Higher Bandwidth, CPU and Memory requirements.
  - Constrains utilization in high performance applications.

- Binary Structure

- Defined by the protocol stack (definition of formats and methodologies).
- Faster at all levels.
- Little/Big Endian concerns.
  - Must depend on platform and/or be defined by the protocol stack.

```
{"msg_id":21654,  
"values":[12, 45, 109]  
}
```

Message data has **42 bytes**

VS.

Structure format

```
uint16 msg_id  
uint8 num_values  
uint8 values[]
```

Message data

has **6 bytes**

```
0x5496  
0x03  
0x0C 0x2D 0x6D
```

Big Endian



# Network/Host Formats

- Different computers architectures/OS use different byte orderings internally for their multibyte integer.
  - ♦ **htonl(i) , htons(i)**
    - ➔ 32-bit or 16-bit integer from host format to network format (Big-endian).
  - ♦ **ntohl(i) , ntohs(i)**
    - ➔ 32-bit or 16-bit integer from network format to host format.

