Application Requirements and Networks

Redes de Comunicações 1

Licenciatura em Engenharia de Computadores e Informática

Applications

- Elastics: use the bandwith that is available (eg: file transfer, e-mail, …)
- Inelastics: need a minimum bandwidth (eg: voice, video, multimedia…)

Voice

- Inelastic
- Runs through UDP protocol
  - No guarantees of delivery and in order
- Can tolerate losses
- Does not tolerate delays, delays variation, and low bandwidths

Voice call

- No changes 
- 10% packet loss 
- 30% packet loss 
- Limited bandwidth to 1.5 KB 
- Limited bandwidth to 1 KB 

Videoconference

- **Inelastic**
- Runs through Real-time protocol, through UDP protocol
  - No guarantees of delivery and in order
- Cannot tolerate losses or low bandwidths
- Can tolerate delays or delays variation if buffering is applied

Videoconference

- 1% packet loss

- 5% packet loss

Recorded audio

- Inelastic
- Can run through TCP or UDP
- Can tolerate losses
- Does not tolerate delays, delays variation, and low bandwidths

Recorded audio

- Music with bitrate of 128 Kbps
  - No changes 
- 5% packet loss 
- 15% packet loss 
- Delay of 200 ms 
- Limited bandwidth to 10 KB 

File Transfer

- Elastic
- Rus through TCP, with guarateed delivery in order

- File size 66.1 Mbits
- Time transfer 2 mins, with bandwidth 900 KB/s

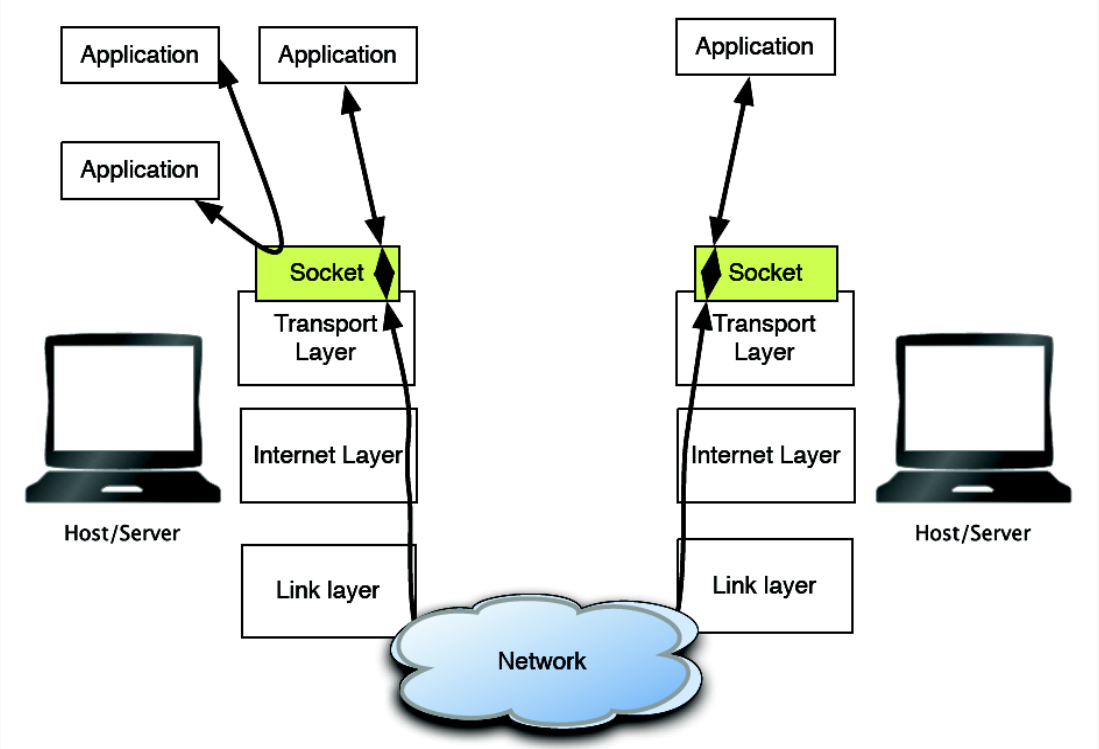
File transfer

Delay (ms)	%Losses	%Duplic	Bandwidth (B/s)	Transmission time (min)
50	-----	-----	150 000	12
100	----	-----	50 000	36
-----	5	-----	150 000	12
-----	15	-----	1 500	Long time
-----	-----	10	900 000	2
-----	-----	50	790 000	5

How to connect Machines?
Sockets and Network Programming

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)

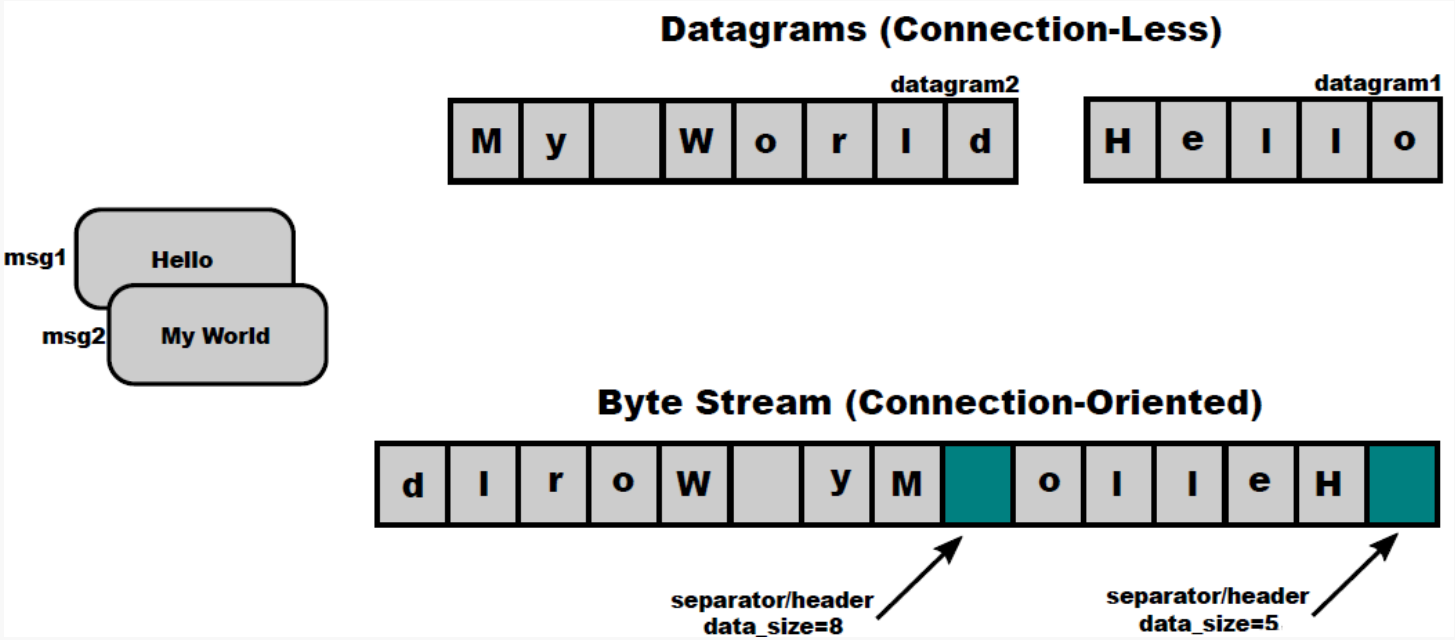
- **A Socket is identified by**
  - Family: AF_INET (IPv4), AF_INET6 (IPv6) and many other less common.
    - Defines the address structure.
    - Defines also the communications layer (e.g. IP version).
  - Type: Determines what transport protocol is used.
    - UDP – Connectionless (SOCK_DGRAM).
    - TCP – Connection oriented (SOCK_STREAM).
    - RAW – Direct access to a layer of the stack (SOCK_RAW).
      - build different protocols, ping command, etc.
  - 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

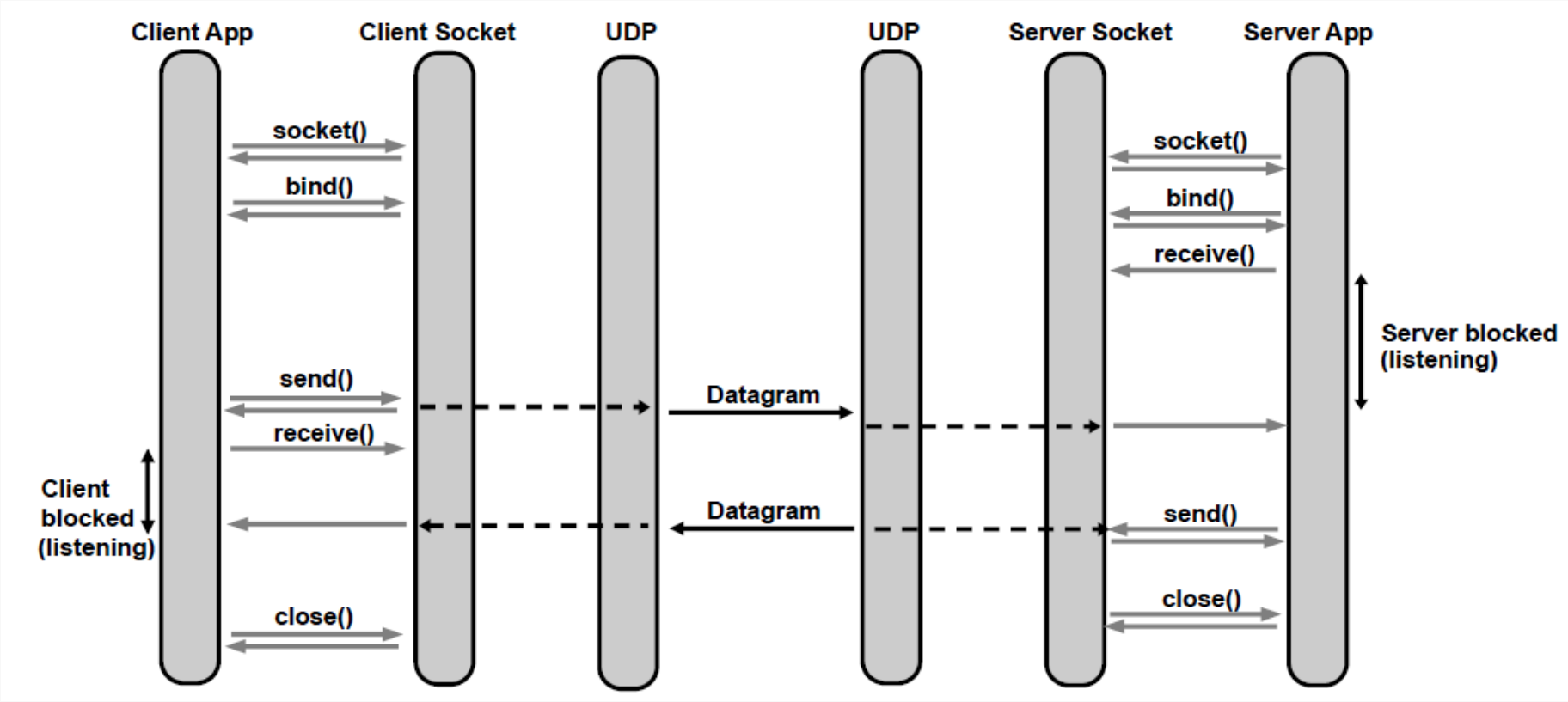
- **TCP needs application-level message separators (headers).**
  - Must contain size information of each “independent” data chunk in the bytestream.



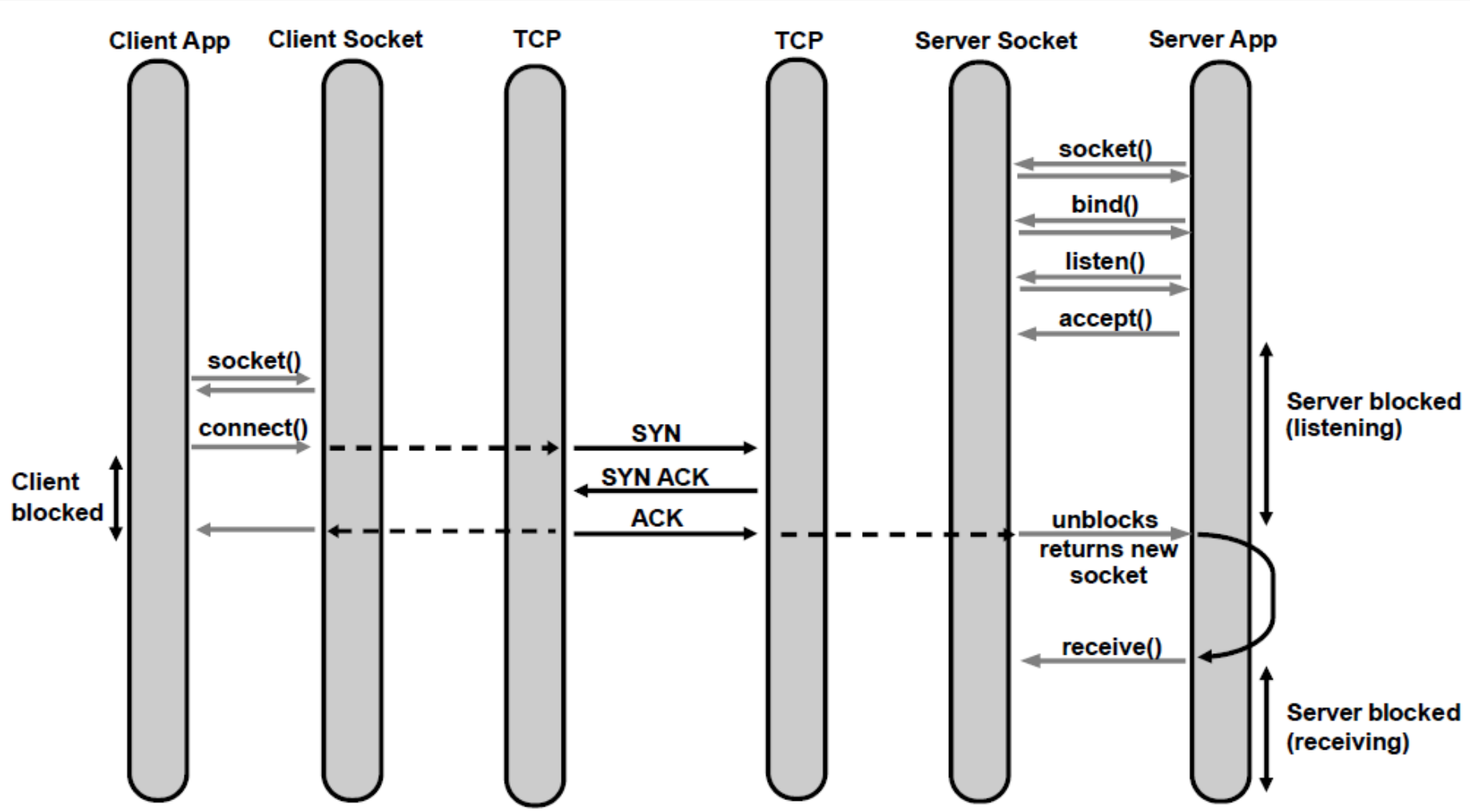
Socket IO / Blocking

- **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.

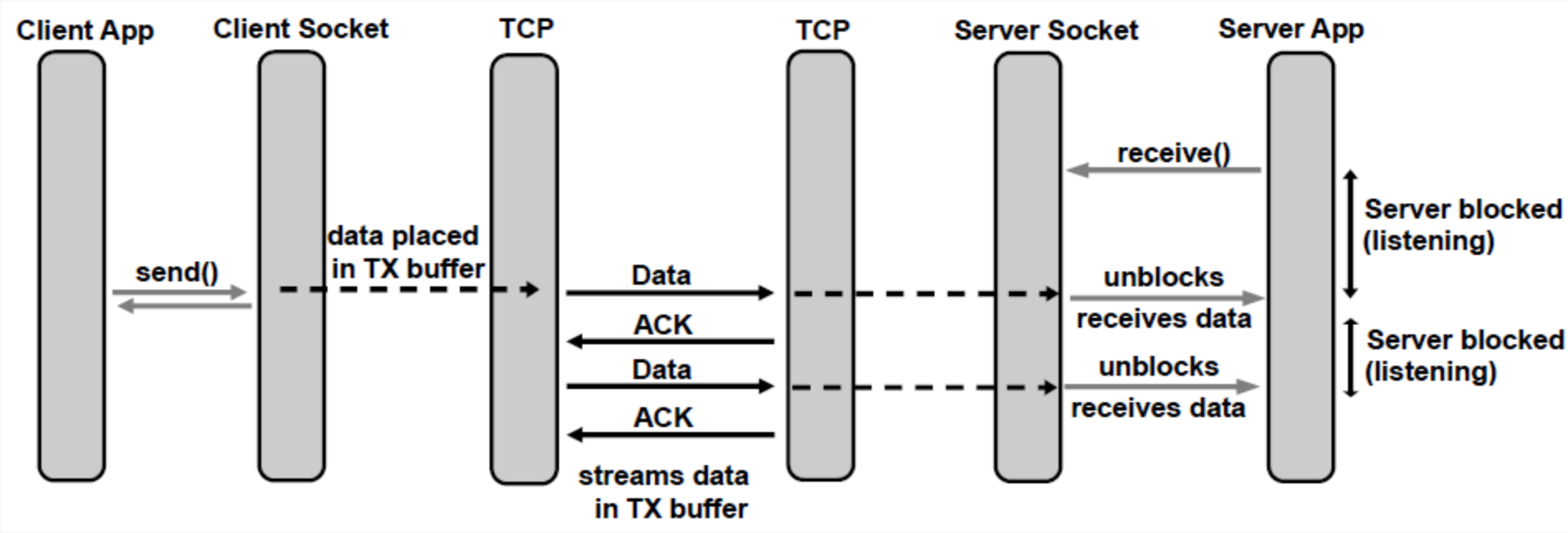
Connection-Less

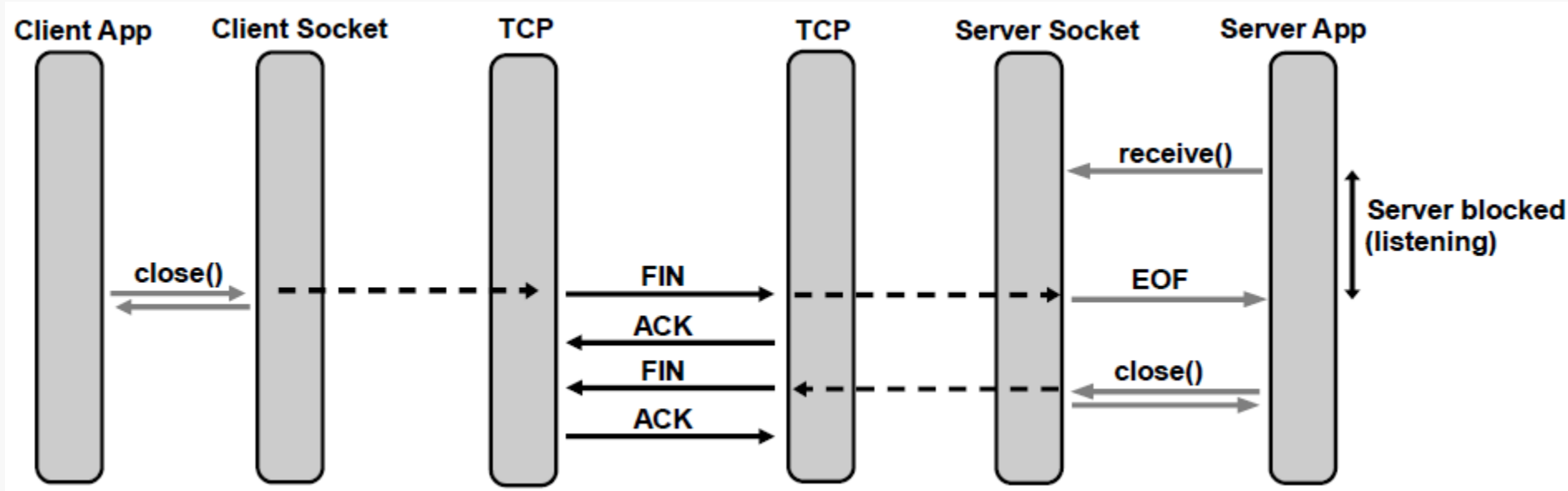


Connection-Oriented (1)



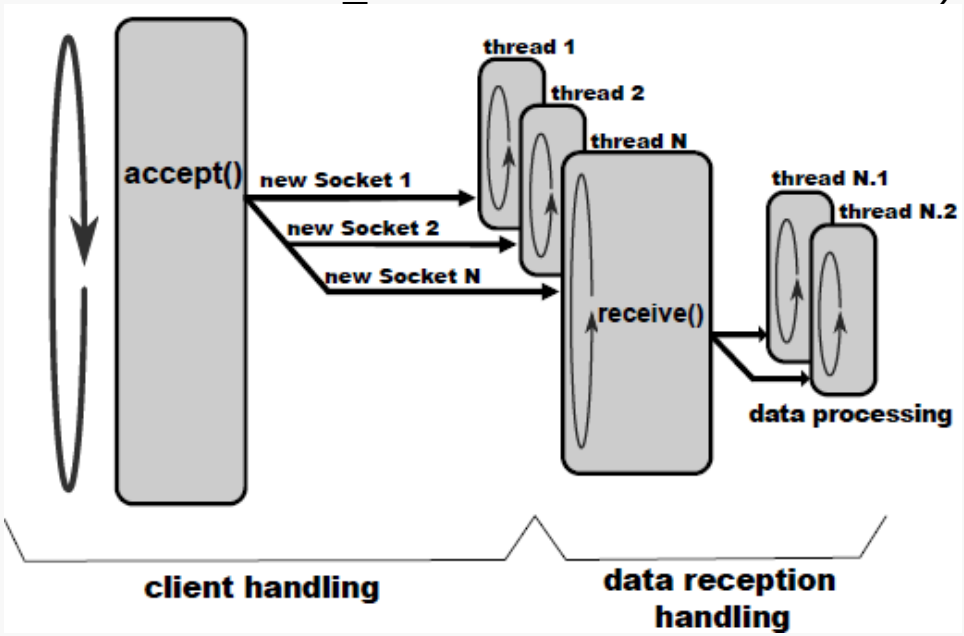
Connection-Oriented (2)

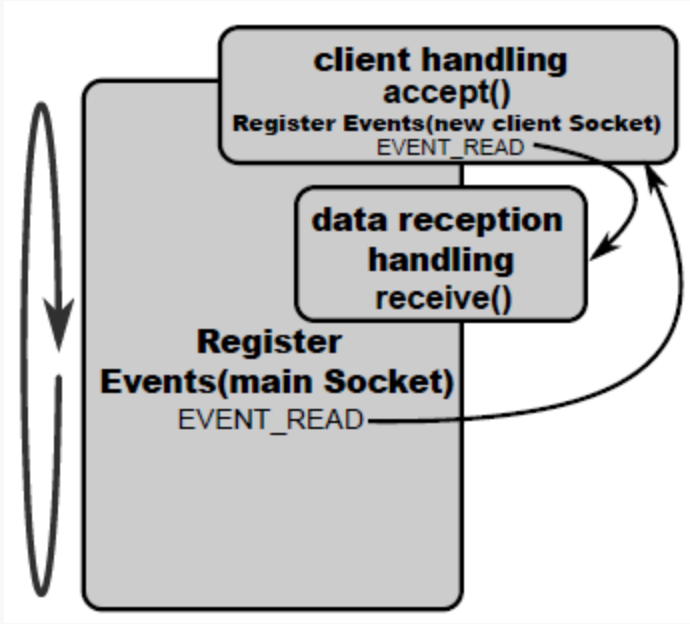




Non-Blocking IO

- **Solutions for Socket Operations Blocking**
  - Threads
    - Multiple parallel processes 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).





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.

Message data has **42 bytes**

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

vs.

Structure format

- **uint16 msg_id**
- **uint8 num_values**
- **uint8 values[]**

Message data has **6 bytes**

- **0x5496**
- **0x03**
- **0x0C 0x2D 0x6D**