Introduction to Sockets Programming

javed shaikh

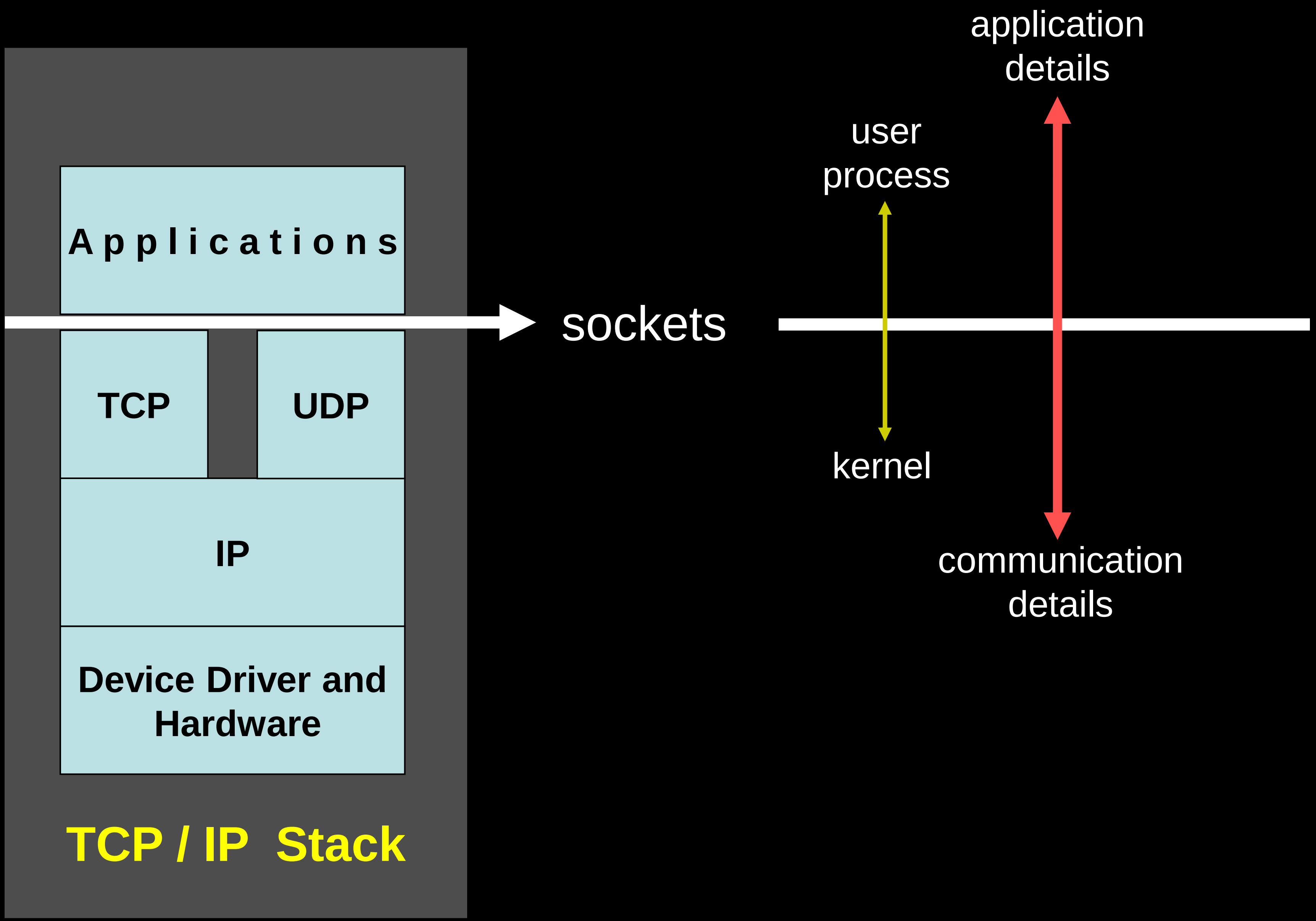
/references

- Unix Network Programming, The Sockets Networking API, Volume 1 (3rd Ed.)
  - W. Richard Stevens, Bill Frenner, Andrew M. Rudoff

/history

- Sockets API released in 1983 with 4.2 BSD (Berkeley Software Distribution) system
- Many UNIX systems use BSD networking code including the sockets
- Others have written their own networking code
- Linux does not belong to Berkeley-derived classification

/stack


application details
user process
Applications
sockets
TCP
UDP
kernel
IP
communication details
Device Driver and Hardware
TCP / IP Stack

/definition

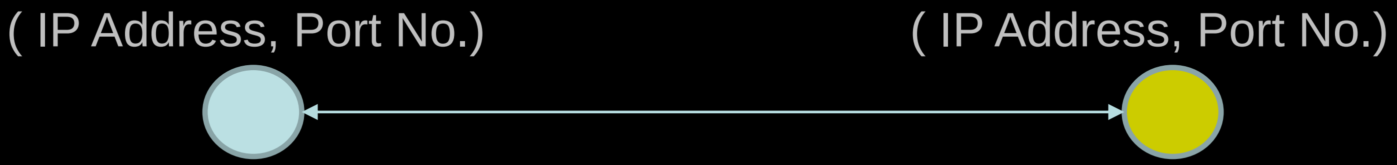
- A socket is a software endpoint that establishes bidirectional communication between a server and client program

- Socket is identified on the Internet by the host's IP address and port number to which it is bound

/port numbers

- Well known ports: *0 – 1023*
- Registered ports: *1024 – 49151*
- Dynamic or private ports: 49152 – 65535
- Internet Assigned Numbers Authority (IANA) maintains list of port number assignments
- RFC 3232
- http://www.iana.org

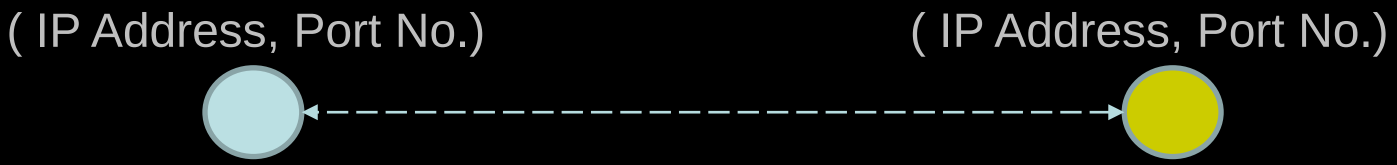
/communication



(IP Address, Port No.)

(IP Address, Port No.)

Connection oriented

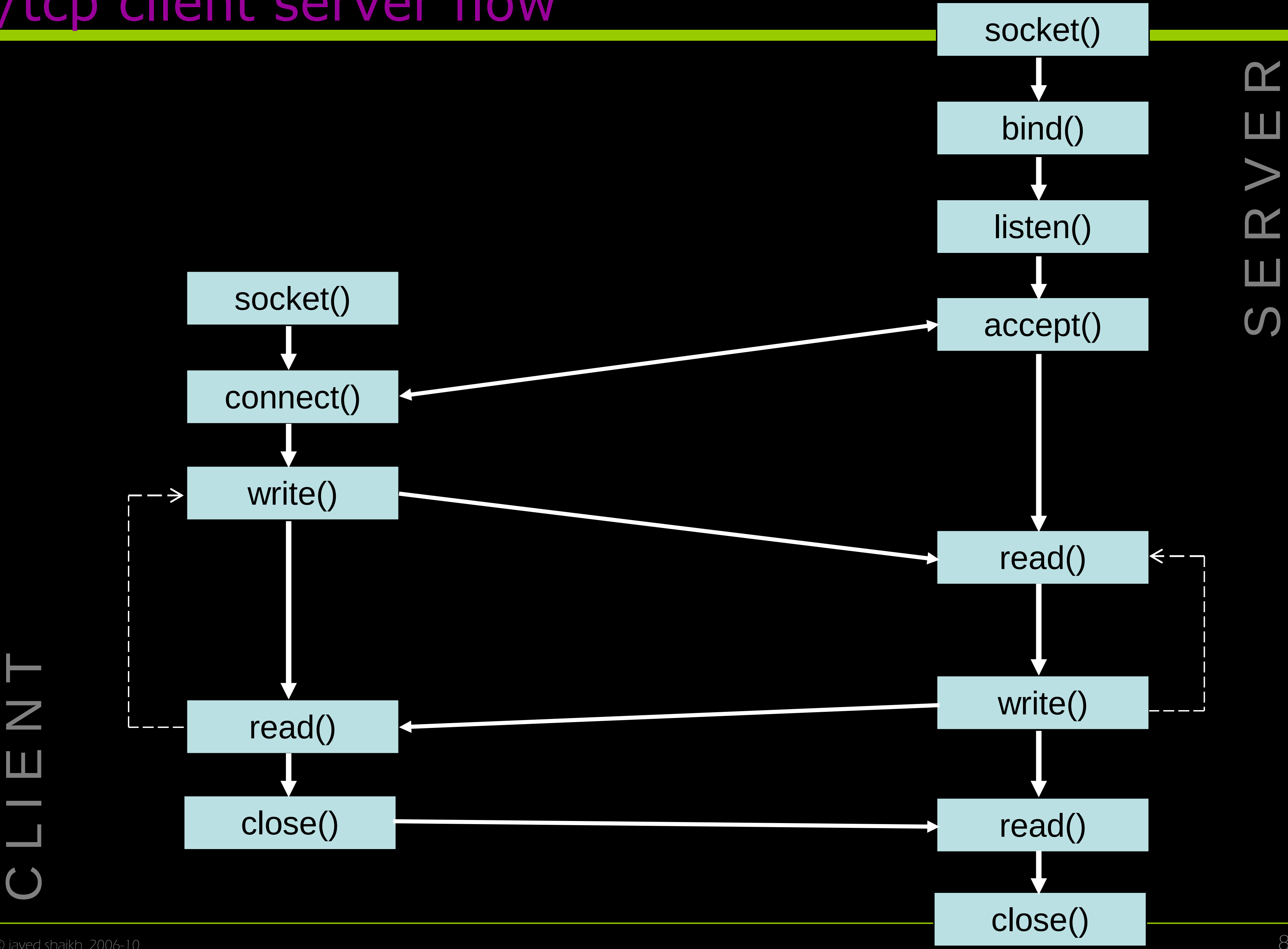


(IP Address, Port No.)

(IP Address, Port No.)

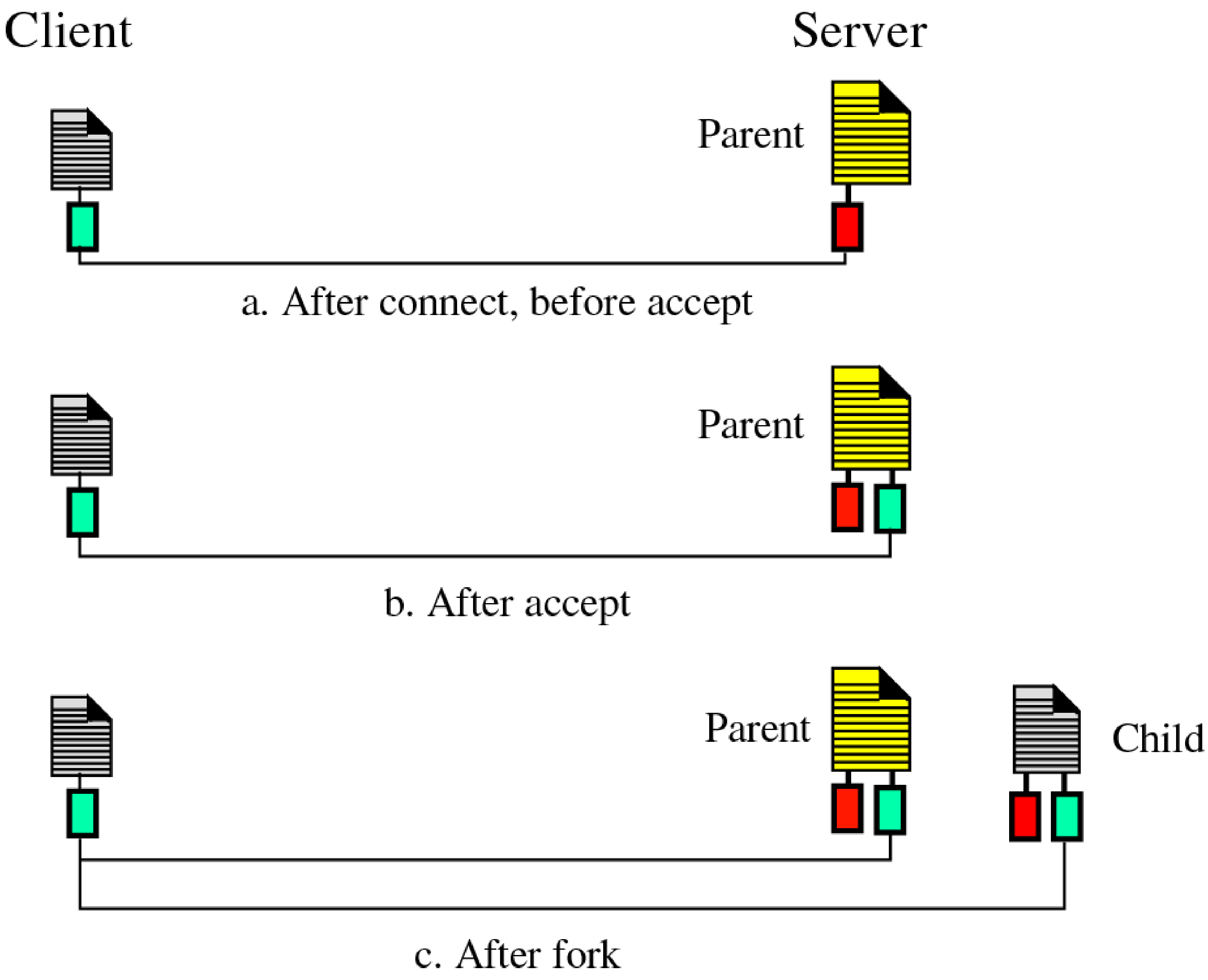
Connection-less

/tcp client server flow

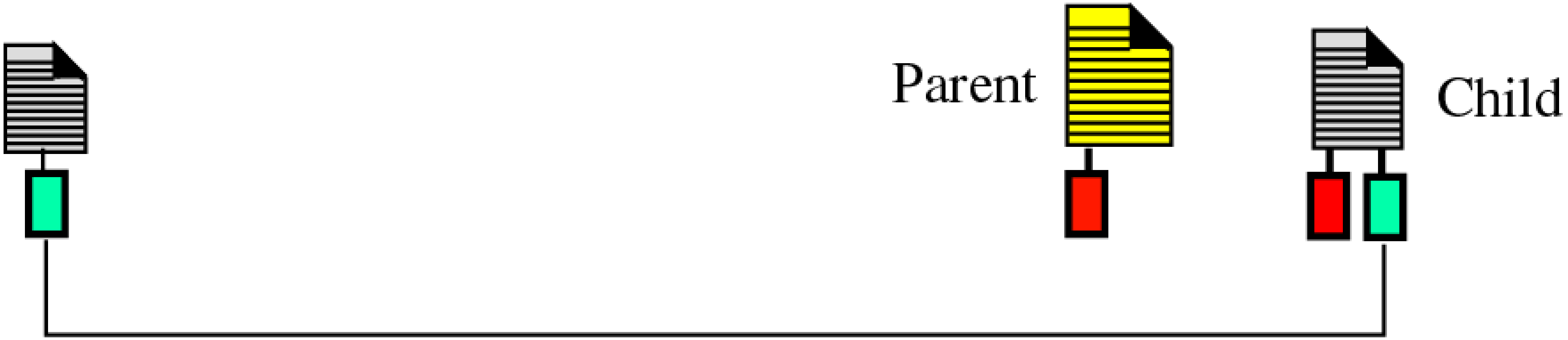


© javed shaikh, 2006-10

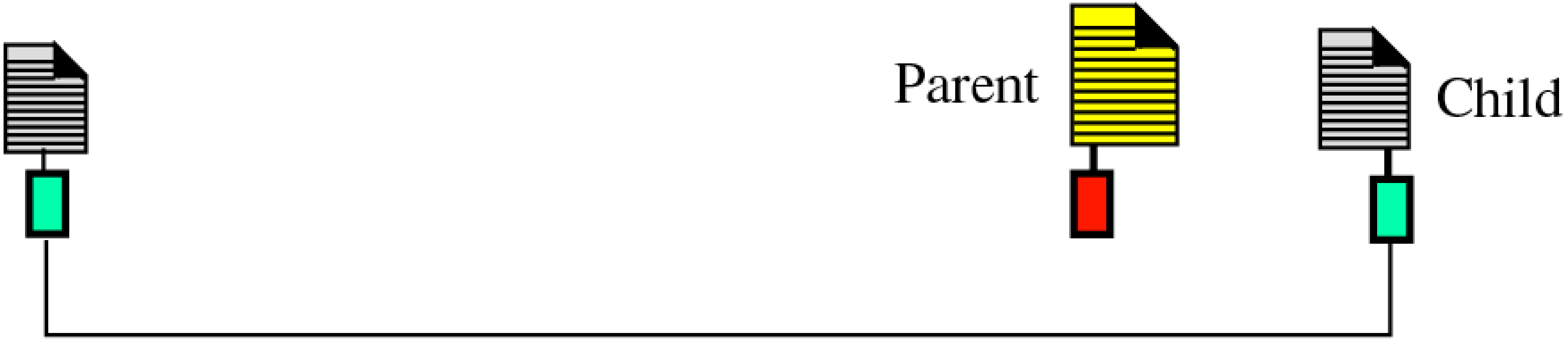
/tcp/port management



/tcp/port management

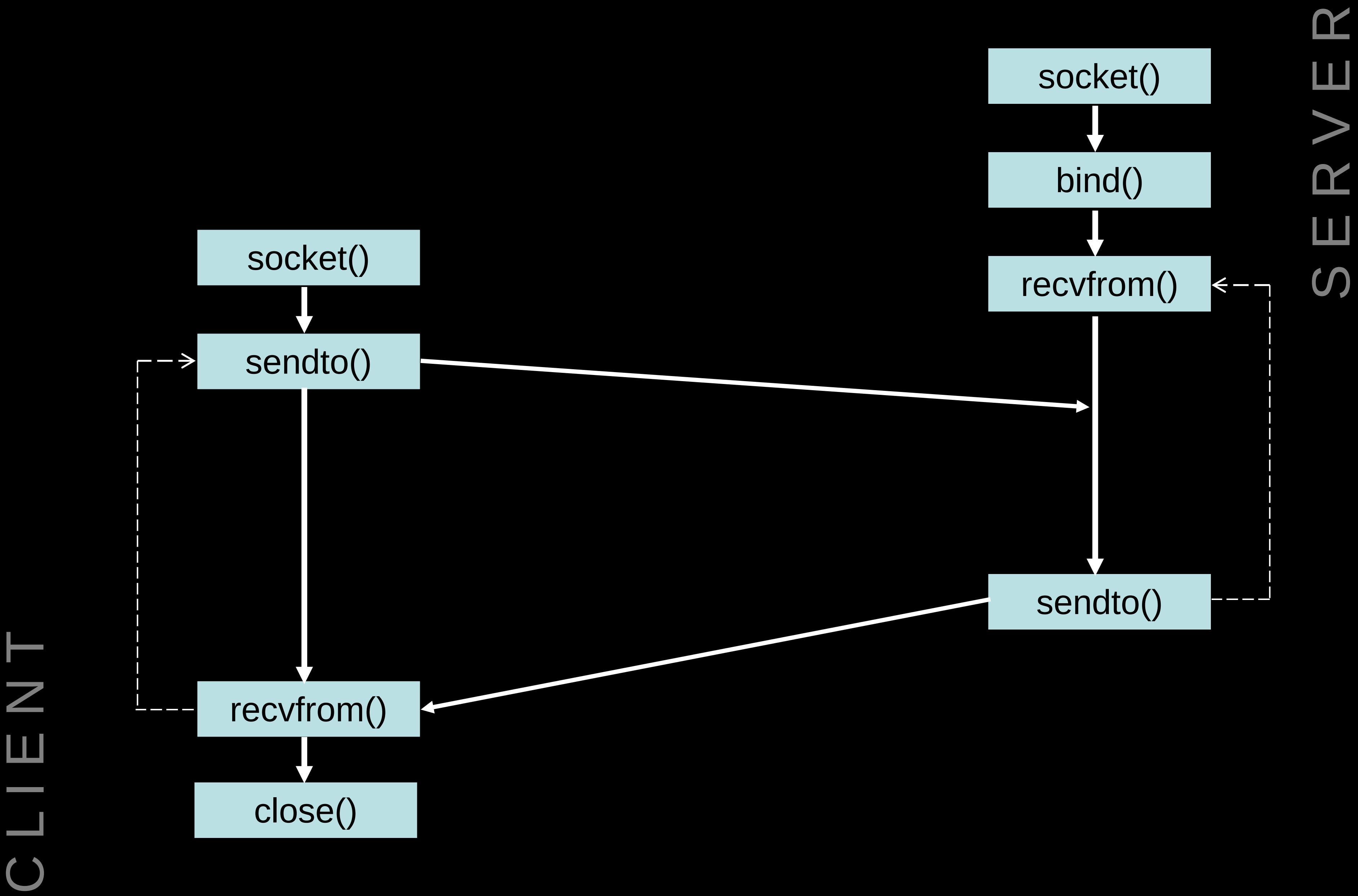


d. After parent closes ephemeral port



e. After child closes well-known port

/udp client server flow



/data types

Datatype	Description	Header
int8_t	signed 8-bit int	<sys/types.h>
uint8_t	unsigned 8-bit int	
int16_t	.	
uint16_t	.	
int32_t	.	
uint32_t	.	
sa_family_t socklen_t	Addr family & length of socket address struct	<sys/socket.h>
in_addr_t	IPv4 addr, uint32_t	<netinet/in.h>
in_port_t	TCP or UDP port, uint16t	

/address struct

Internet Address

```
struct in_addr {

 in_addr_t s_addr;                /* 32 bit IPv4 address */
                                  /* network byte ordered */
};
```

Socket Address

```
struct sockaddr_in {

 uint8_t sin_len;                 /* length of structure */
 sa_family_t sin_family;          /* AF_INET */

 in_port_t sin_port;              /* 16-bit TCP or UDP port
                                        number */

 struct in_addr sin_addr;         /* 32 bit IPv4 address */
 char sin_zero[8];                /* unused */
};
```

/byte manipulation functions

```
void bzero(void *dest, size_t nbytes);

void bcopy(const void *src, void *dst,
                                  size_t nbytes);

int bcmp(const void *ptr1, const void *ptr2,
                                 size_t nbytes);

int *memset(void *dest, int c, size_t len);

int *memcpy(void *dest, const void *src,
                              size_t nbytes);

int memcmp(const void *ptr1, const void *ptr2,
                                 size_t nbytes);
```

/socket creation

- int socket (int *NAMESPACE*, int *STYLE*,
  int *PROTOCOL*)
  - NAMESPACE
    - PF_LOCAL, PF_UNIX
    - PF_INET
    - PF_INET6
    - …
  - STYLE
    - SOCK_STREAM
    - SOCK_DGRAM
    - SOCK_RAW
  - PROTOCOL
- Return values
  - EPROTONOSUPPORT
  - EACCES
  - …

/assigning a name

- int bind *(int sockfd, const struct sockaddr *my_addr, socklen_t addrlen)*
  - sockfd
    - Valid socket descriptor created by socket()
  - my_addr
    - Pointer to socket address
  - Addrlen
    - Size of my_addr
- Return values
  - EACCES
  - EADDRINUSE
  - EBADF
  - EINVAL
  - ENOTSOCK

/wait for connections

- int listen *(int sockfd, int backlog)*
  - sockfd
    - Valid socket descriptor created by socket() and successfully named
  - backlog
    - Queue length for completely established sockets waiting to be accepted
    - Default for linux is 128 (/proc/sys/net/core/somaxconn)
- Return values
  - EADDRINUSE
  - EBADF
  - ENOTSOCK
  - EOPNOTSUPP

/accept connections

- int accept *(int sockfd, struct sockaddr *addr, socklen_t addrlen)*
  - sockfd
    - Valid socket descriptor created by socket(), successfully named and made ready for connections
  - addr
    - On successful return, contains remote socket information
- Return values
  - On success, returns new socket descriptor
  - EAGAIN / EWOULDBLOCK
  - EBADF
  - ECONNABORTED
  - EINTR

/to initiate connection

- int connect *(int sockfd, const struct sockaddr *serv_addr, socklen_t addrlen)*
  - sockfd
    - Valid socket descriptor and (optional) successfully named
  - serv_addr
    - Contains IP addr, port number information of the remote socket
- Return values
  - On success, returns new socket descriptor
  - ECONNREFUSED
  - EISCONN
  - ENETUNREACH
  - ETIMEDOUT

/to send data

- ssize_t send *(int s, const void *buf,*
  size_t len, int flags)
  - flags
    - MSG_DONTROUTE
    - MSG_DONTWAIT
    - MSG_OOB

- ssize_t sendto
  *(int s, const void *buf, size_t len, int flags,*
  *const struct sockaddr *to, socklen_t tolen)*
- Return values
  - On success, returns length of data sent
  - ECONNRESET
  - ENOMEM
  - ENOTCONN
  - EOPNOTSUPP

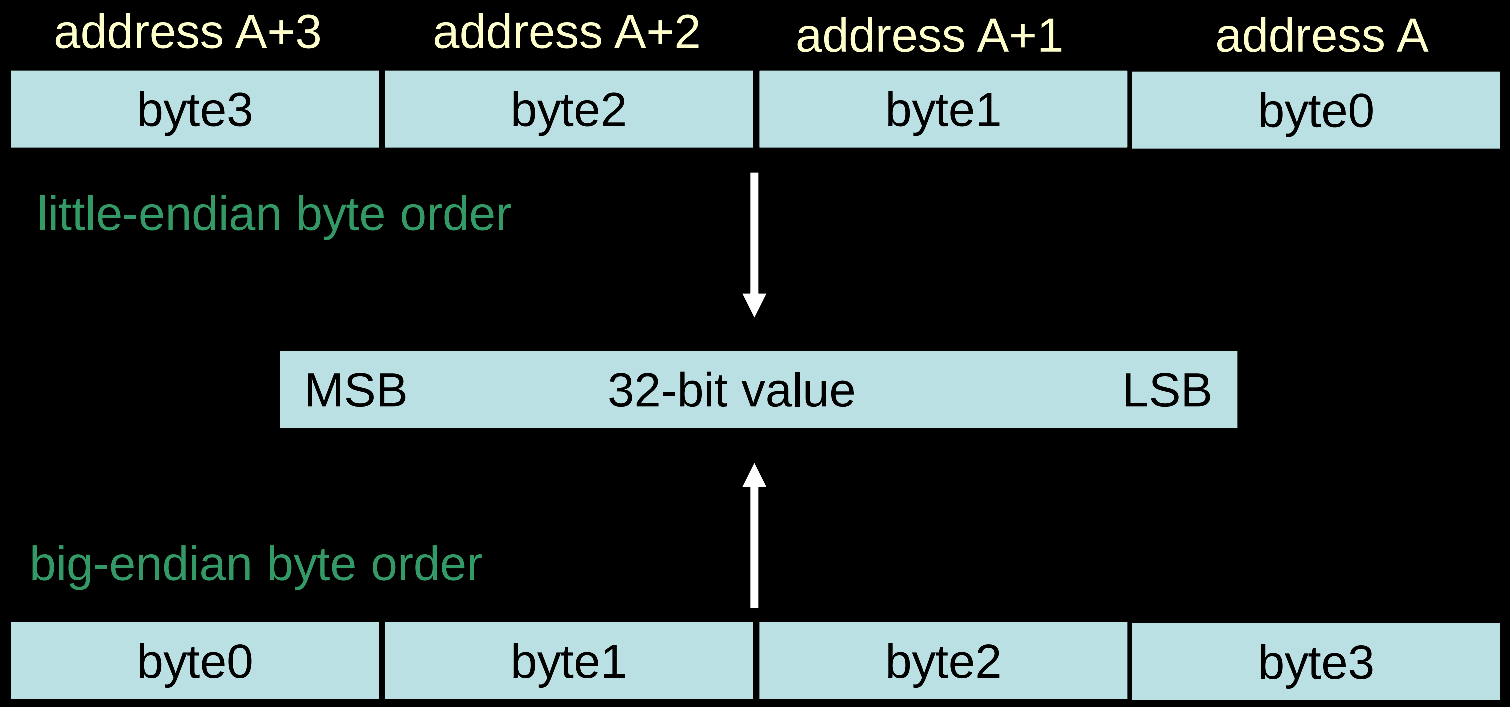
/to receive data

- ssize_t recv *(int s, void *buf,*
  size_t len, int flags)
  - flags
    - MSG_PEEK
    - MSG_DONTWAIT
    - MSG_OOB

- ssize_t recvfrom
  *(int s, const void *buf, size_t len, int flags,*
  *struct sockaddr *from, socklen_t from_len)*
- Return values
  - On success, returns length of data received
  - EAGAIN
  - EBADF
  - ECONNREFUSED
  - ENOTCONN

/byte ordering functions

- There are two ways to store a 32-bit integer:



address A+3 address A+2 address A+1 address A

byte3 byte2 byte1 byte0

little-endian byte order

MSB 32-bit value LSB

big-endian byte order

byte0 byte1 byte2 byte3

- The terms little endian and big endian indicate which end of the multibyte value, the little end or the big end, is stored at the starting address of the value

/byte ordering functions

```
uint16_t   htons(uint16_t host16bitvalue);

uint32_t   htonl(uint32_t host32bitvalue);

uint16_t   ntohs(uint16_t net16bitvalue);

uint32_t   ntohl(uint32_t net16bitvalue);
```

- In systems with big-endian byte ordering, the above 4 functions are usually defined as null macros

Sample code

/assignments

- Write a C program to determine host byte order
- Sample socket programs:
  - Study
  - Change server port numbers, compile and execute
  - Add error checks for each socket library function

Thank You