Socket Option

Introduction

- Ways to get and set the socket option that affect a socket
  - ***getsockopt , setsockopt***

getsockopt and **setsockopt** function

- **These functions apply only to sockets.**

#include <sys/socket.h>
int getsockopt(int ***sockfd***, int ***level***, int ***optname***, void ****optval***, socklent_t ****optlen***);
int setsockopt(int ***sockfd***, int ***level*** , int ***optname***, const void ****optval***, socklent_t ***optlen***);

•***sockfd*** => Open socket descriptor
•***Level*** => Code in the system to interpret the option (generic, ipv4, ipv6, TCP)
•***Optval*** => Pointer to a variable from which the new value of the option is fetched by ***setsockopt,*** or into which the current value of the option is stored by ***getsockopt.***
•***Optlen*** => The size of the option variable.

getsockopt and **setsockopt** function

Level	Optname	**getsockopt**	**setsockopt**
SOL_SOCKET	SO_SNDBUF	O	O
	SO_RCVBUF	O	O
	SO_REUSEADDR	O	O
	SO_KEEPALIVE	O	O
	SO_BROADCAST	O	O
	SO_DONTROUTE	O	O
	SO_OOBINLINE	O	O
	SO_ERROR	O	X
	SO_TYPE	O	X

getsockopt and **setsockopt** function

- There are two basic types of options:
  - Binary options enable or disable a certain feature (flags).
  - Options that fetch and return specific values that we can either set or examine.

socket state

- Some socket options have timing considerations about when to set or fetch the option versus the state of the socket.
- The following socket options are inherited by a connected TCP socket from the listening socket: SO_DEBUG, SO_DONTROUTE, SO_KEEPALIVE, SO_LINGER, SO_OOBINLINE, SO_RCVBUF, and SO_SNDBUF.
- This is important with TCP because the connected socket is not returned to a server by accept until the three-way handshake is completed by the TCP layer.
- We must set that option for the listening socket to ensure that one of these socket options is set for the connected socket when the three-way handshake completes.

Generic socket option

- **SO_BROADCAST**
  - Enable or disable the ability of the process to send broadcast messages. (**only datagram socket**: Ethernet, token ring..).
  - You cannot broadcast on a point-to-point link or any connection-based transport protocol such as SCTP or TCP.
- **SO_DEBUG**
  - This option is supported only by TCP.
  - When enabled for a TCP socket, the kernel keeps track of detailed information about all the packets sent or received by TCP for the socket.

Generic socket option

- **SO_DONTROUTE**
  - Outgoing packets are to bypass the normal routing mechanisms of the underlying protocol.
  - The destination must be on a directly-connected network, and messages are directed to the appropriate network interface according to the destination address.
  - This option is often used by routing daemons (e.g., routed and gated) to bypass the routing table and force a packet to be sent out a particular interface.
- **SO_ERROR**
  - When error occurs on a socket, the protocol module in a Berkeley-derived kernel sets a variable named **so_error for that socket.**
  - **Process can obtain the value of so_error by fetching the SO_ERROR socket option**

SO_KEEPALIVE

- **SO_KEEPALIVE**
  - When the keep-alive option is set for a TCP socket and no data has been exchanged across the socket in either direction for two hours, TCP automatically sends a keep-alive probe to the peer.
- Peer response
  - ACK (everything OK)
  - RST: Tells the local TCP that the peer host has crashed and rebooted. The socket's pending error is set to ECONNRESET and the socket is closed.
  - No response: Berkeley-derived TCPs send 8 additional probes, 75 seconds apart, trying to elicit a response. TCP will give up if there is no response within 11 minutes and 15 seconds after sending the first probe. The socket’s pending error is set to ETIMEDOUT and the socket is closed

SO_LINGER

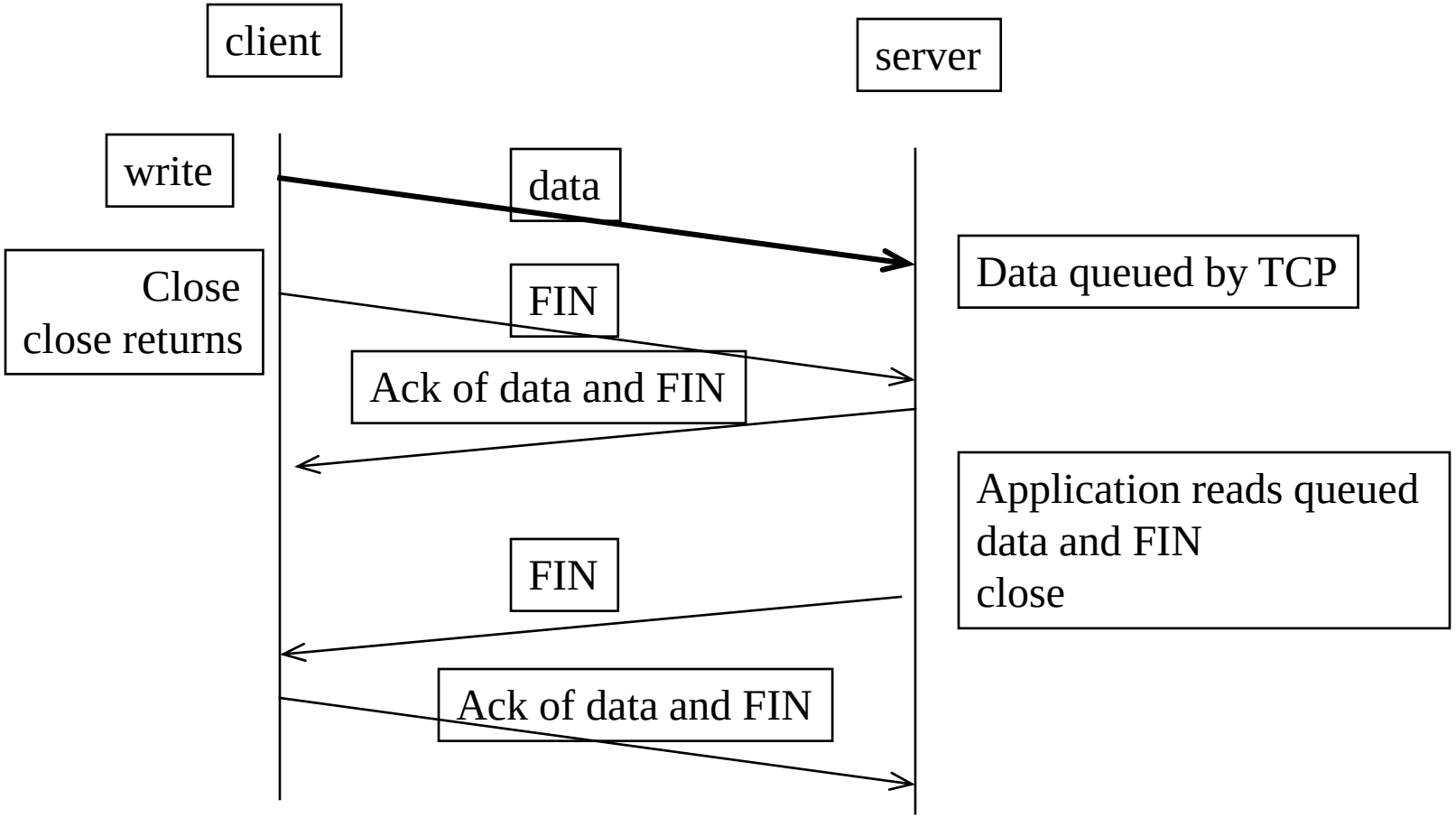
- Specify how the ***close*** function operates for a connection-oriented protocol (default: close returns immediately)
  - struct linger{

    int l_onoff; /* 0 = off, nonzero = on */

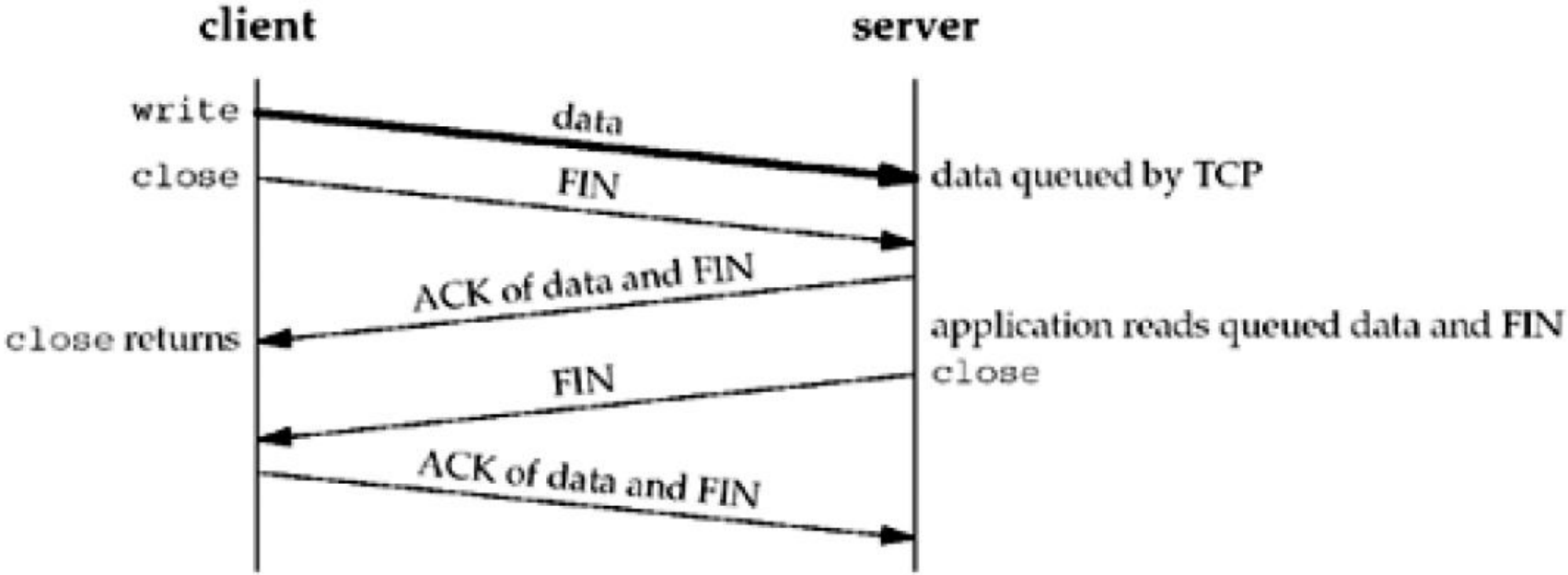
    int l_linger; /*linger time : second*/

    };
- If l_onoff is 0, the option is turned off.
- ***l_onoff*** = nonzero and ***l_linger*** is 0: TCP aborts the connection when it is closed, and discards any remaining data in the send buffer.
- ***l_onoff*** = nonzero and ***l_linger*** is nonzero: Process wait until the remaining data is sent or until linger time expires.
  - If the socket has been set nonblocking it will not wait for the ***close*** to complete, even if linger time is nonzero.

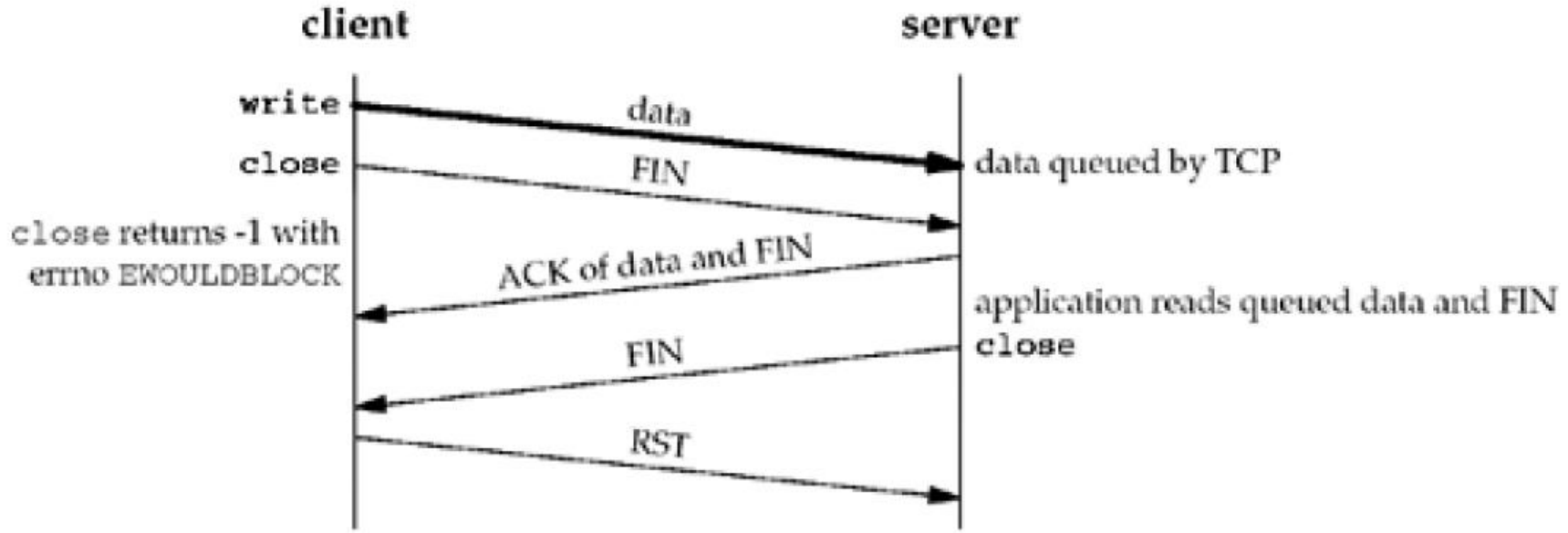
SO_LINGER (Default operation of close: it returns immediately)



SO_LINGER (close with SO_LINGER socket option set and l_linger a positive value)



SO_LINGER (close with SO_LINGER socket option set and l_linger a small positive value)



- A way to know that the peer application has read the data
  - use an application-level ack or application ACK
  - **client**

```
char ack;
Write(sockfd, data, nbytes); // data from client to server
n=Read(sockfd, &ack, 1); // wait for application-level ack
```

  - **server**

```
nbytes=Read(sockfd, buff, sizeof(buff)); //data from client
//server verifies it received the correct amount of data from
// the client
Write(sockfd, “”, 1);//server’s ACK back to client
```

SO_RCVBUF , SO_SNDBUF

- The receive buffers are used by TCP, UDP, and SCTP to hold received data until it is read by the application.
- With TCP, the available room in the socket receive buffer limits the window that TCP can advertise to the other end
- Default send-buffer, and receive-buffer size.
  - Default TCP send and receive buffer size :
    - 4096 bytes (Older Berkeley-derived implementations)
    - 8192-61440 bytes (newer systems)
  - Default UDP buffer size: 9000bytes, 40000 bytes
- SO_RCVBUF option must be set before the connection is established.
- TCP socket buffer size should be at least three times the MSSs.

SO_RCVLOWAT , SO_SNDLOWAT

- Every socket has a receive low-water mark and send low-water mark. (used by select function)
- **Receive low-water mark:**
  - the amount of data that must be in the socket receive buffer for select to return “readable”.
  - Default receive low-water mark : 1 for TCP and UDP
- **Send low-water mark:**
  - the amount of available space that must exist in the socket send buffer for select to return “writable”
  - Default send low-water mark : 2048 for TCP
  - UDP send buffer never change because dose not keep a copy of send datagram.

SO_RCVTIMEO, SO_SNDTIMEO

- These two socket options allow us to place a timeout on socket receives and sends.
- Specify the timeouts in seconds and microseconds.
- We disable a timeout by setting its value to 0 seconds and 0 microseconds.
- Both timeouts are disabled by default.

SO_REUSEADDR, SO_REUSEPORT

- Allow a listening server to start and bind its well known port even if previously established connection exist that use this port as their local port.
- Allow multiple instance of the same server to be started on the same port, as long as each instance binds a different local IP address.
- Allow a single process to bind the same port to multiple sockets, as long as each bind specifies a different local IP address.
- Allow completely duplicate bindings: multicasting

SO_TYPE

- This option returns the socket type.
- The integer value returned is a value such as SOCK_STREAM or SOCK_DGRAM.
- This option is typically used by a process that inherits a socket when it is started.

SO_USELOOPBACK

- This option applies only to sockets in the routing domain(AF_ROUTE).
- This option defaults to ON for sockets.
- When this option is enabled, the socket receives a copy of everything sent on the socket.

IPv4 socket option

- These socket options are processed by IPv4 and have a level of ***IPPROTO_IP.***
- IP_HDRINCL => If this option is set for a raw IP socket, we must build our IP header for all the datagrams that we send on the raw socket.
  - Normally, the kernel builds the IP header for datagrams sent on a raw socket, but there are some applications that build their own IP header to override values that IP would place into certain header fields.

IPv4 socket option

- IP_OPTIONS=>Setting this option allows us to set IP options in the IPv4 header.
- This requires intimate knowledge of the format of the IP options in the IP header.
- IP_RECVDSTADDR=>This socket option causes the destination IP address of a received UDP datagram to be returned as ancillary data by ***recvmsg***.

IP_RECVIF

- This socket option causes the index of the interface on which a UDP datagram is received to be returned as ancillary data by recvmsg.

IP_TOS

- Set the type-of-service(TOS) field in IP header for a TCP, UDP, or SCTP socket.
- If we call *getsockopt* for this option, the current value that would be placed into the TOS (type of service) field in the IP header is returned.

IP_TTL

- With this option, we can set and fetch the default TTL that the system will use for unicast packets sent on a given socket.
- BSD uses the default of 64 for both TCP and UDP sockets.
- There is no way to obtain the value from a received datagram

TCP socket option

- There are two socket options for TCP.
- We specify the level as IPPROTO_TCP.

TCP_KEEPALIVE

- It specifies the idle time in second for the connection before TCP starts sending the keepalive probe.
- Default 2 hours
- This option is effective only when the SO_KEEPALIVE socket option is enabled.

TCP_MAXRT

- It specifies the amount of time in seconds before a connection is broken once TCP starts retransmitting data.
  - 0: use default
  - -1:retransmit forever
  - positive value: rounded up to next transmission time

TCP_MAXSEG

- This socket option allows us to fetch or set the MSS for a TCP connection.
- The value returned is the maximum amount of data that our TCP will send to the other end;
- It is the MSS announced by the other end with its SYN, unless our TCP chooses to use a smaller value than the peer's announced MSS.

TCP_NODELAY

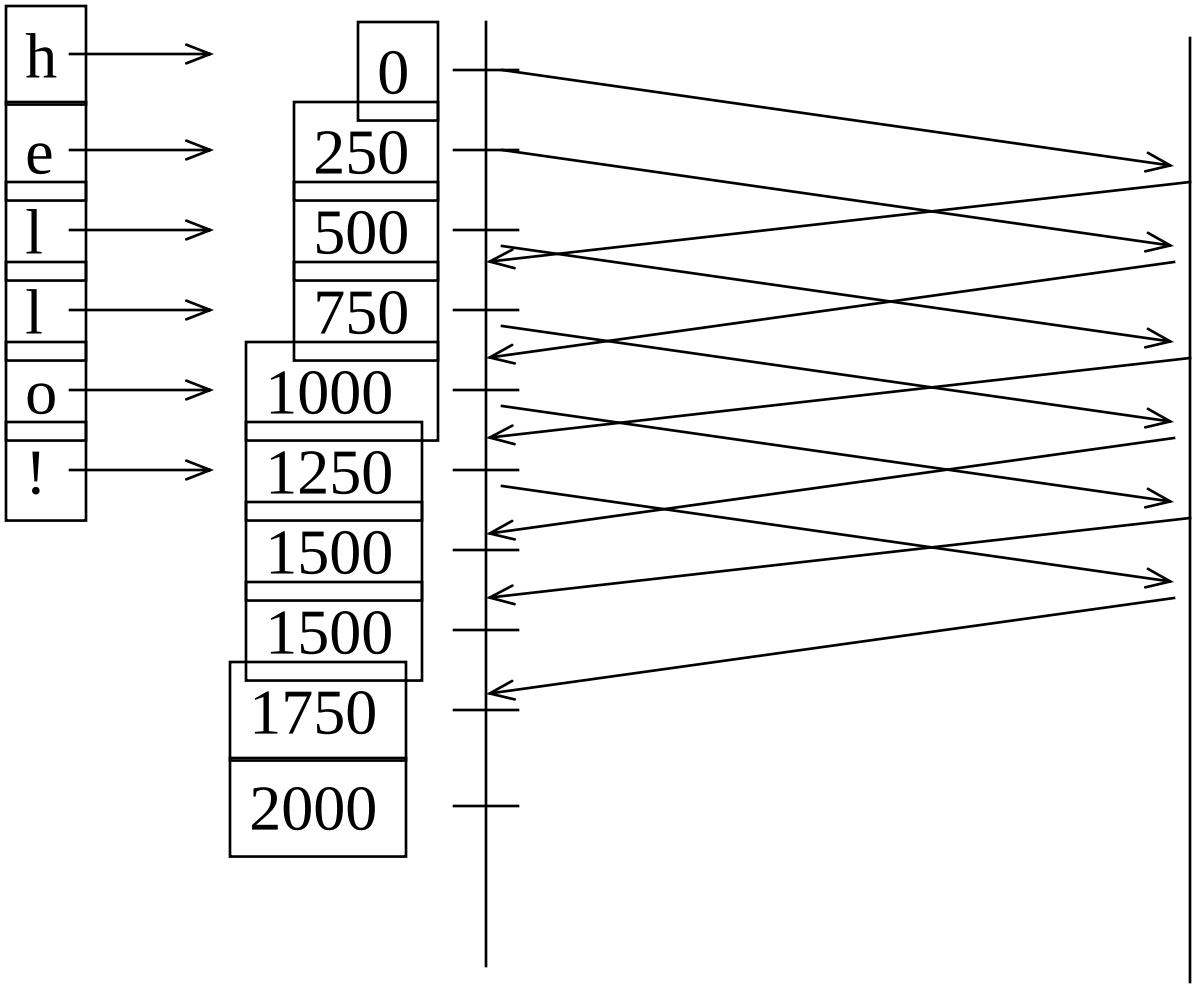
- This option disables TCP's ***Nagle algorithm***.

  (default this algorithm enabled)
- Purpose of the ***Nagle algorithm***.
  - The algorithm states that if a given connection has outstanding data, then no small packets will be sent on the connection in response to a user write operation until the existing data is acknowledged.
  - The definition of a "small" packet is any packet smaller than the MSS.

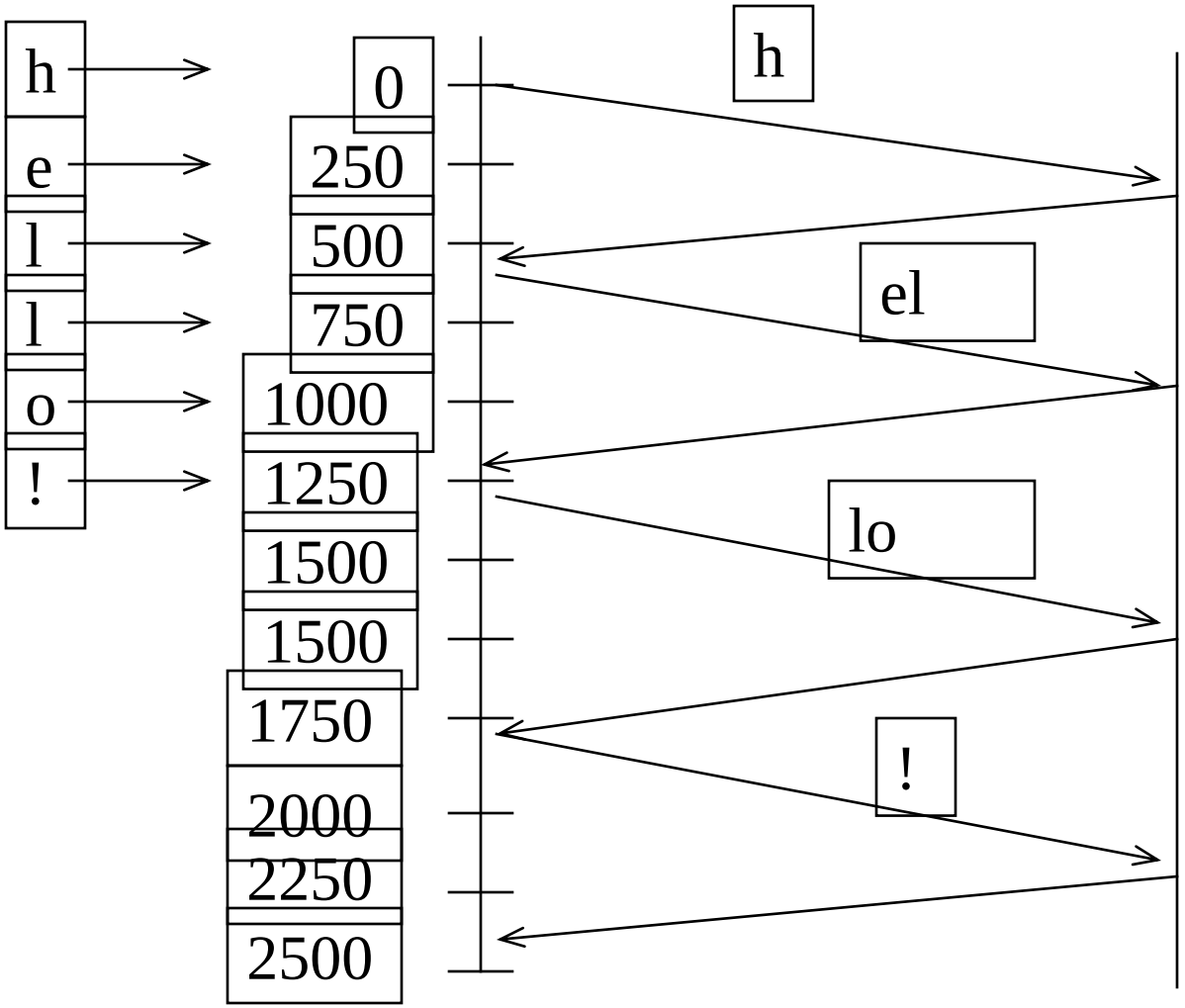
Nagle algorithm

- Default enabled.
- Reduce the number of small packet on the WAN.
- If given connection has outstanding data , then no small packet data will be sent on connection until the existing data is acknowledged.

Nagle algorithm disabled



Nagle algorithm enabled



Thank you