TCP out-of-band signaling explained

Frank Rehberger
2019

About Frank

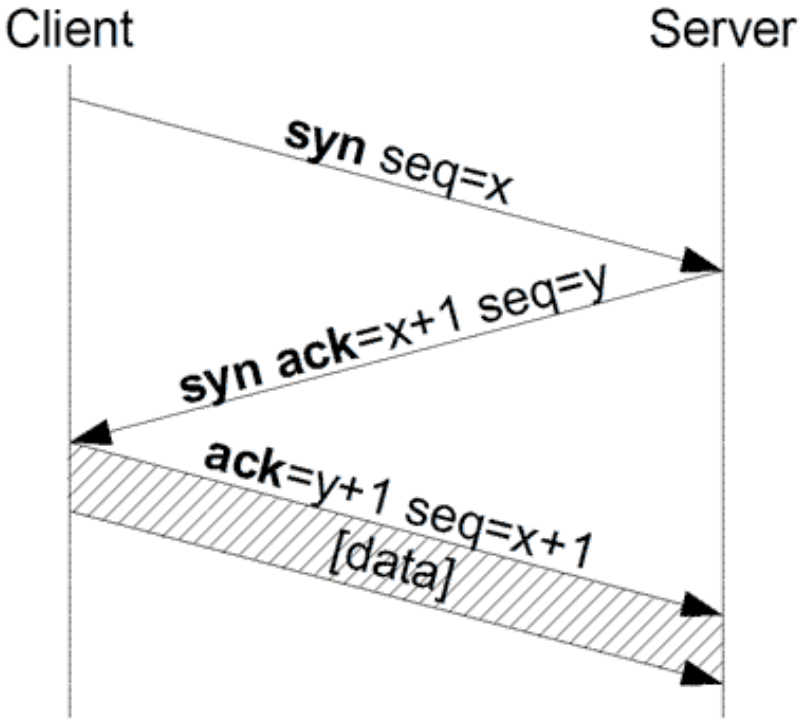
Freelancer in area of

- Network Security
- Embedded Security
- Mobile Apps
- System Engineering
- Testing

TCP Protocol

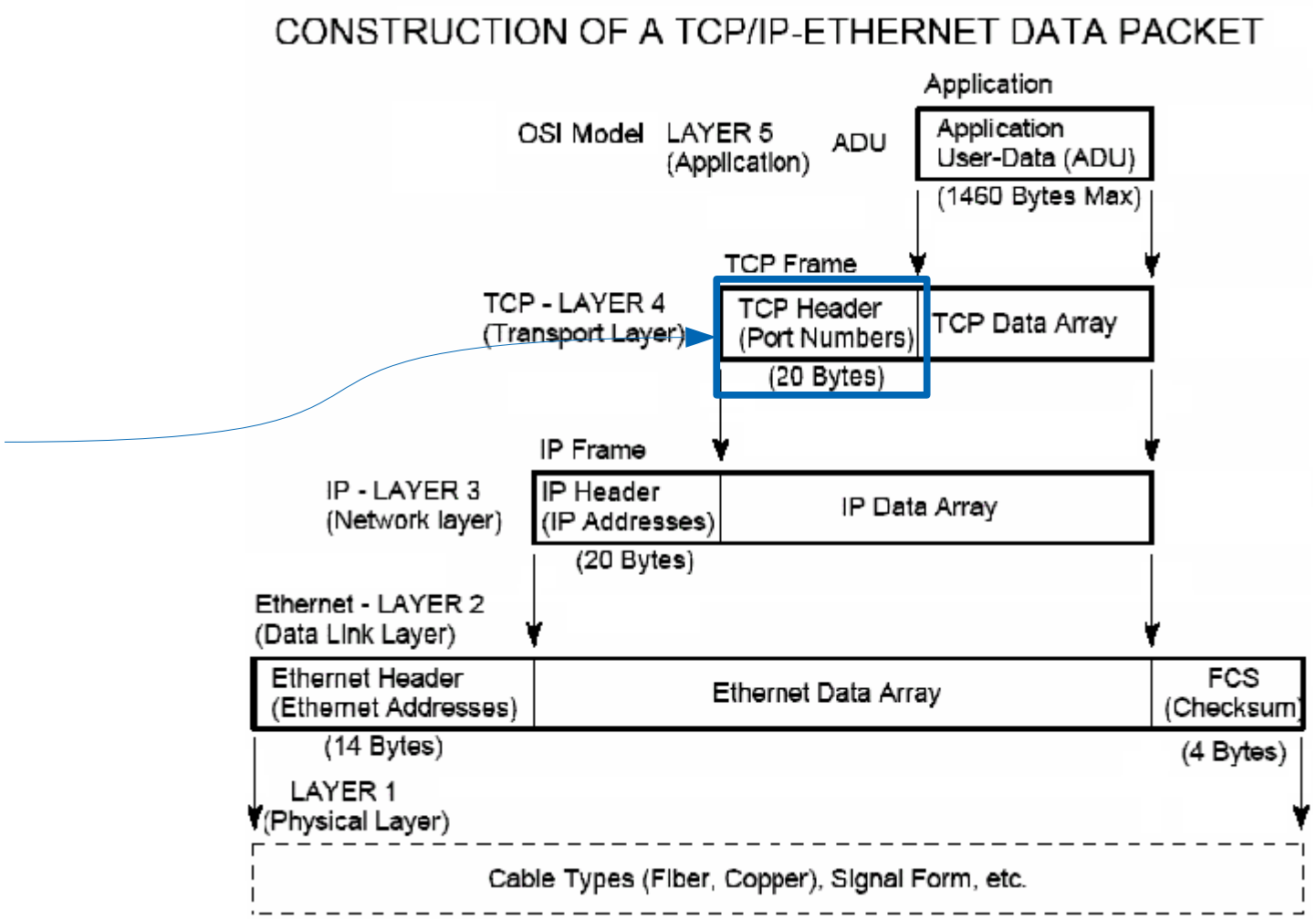
- RFC 793 (1981)

TCP 3-way handshake

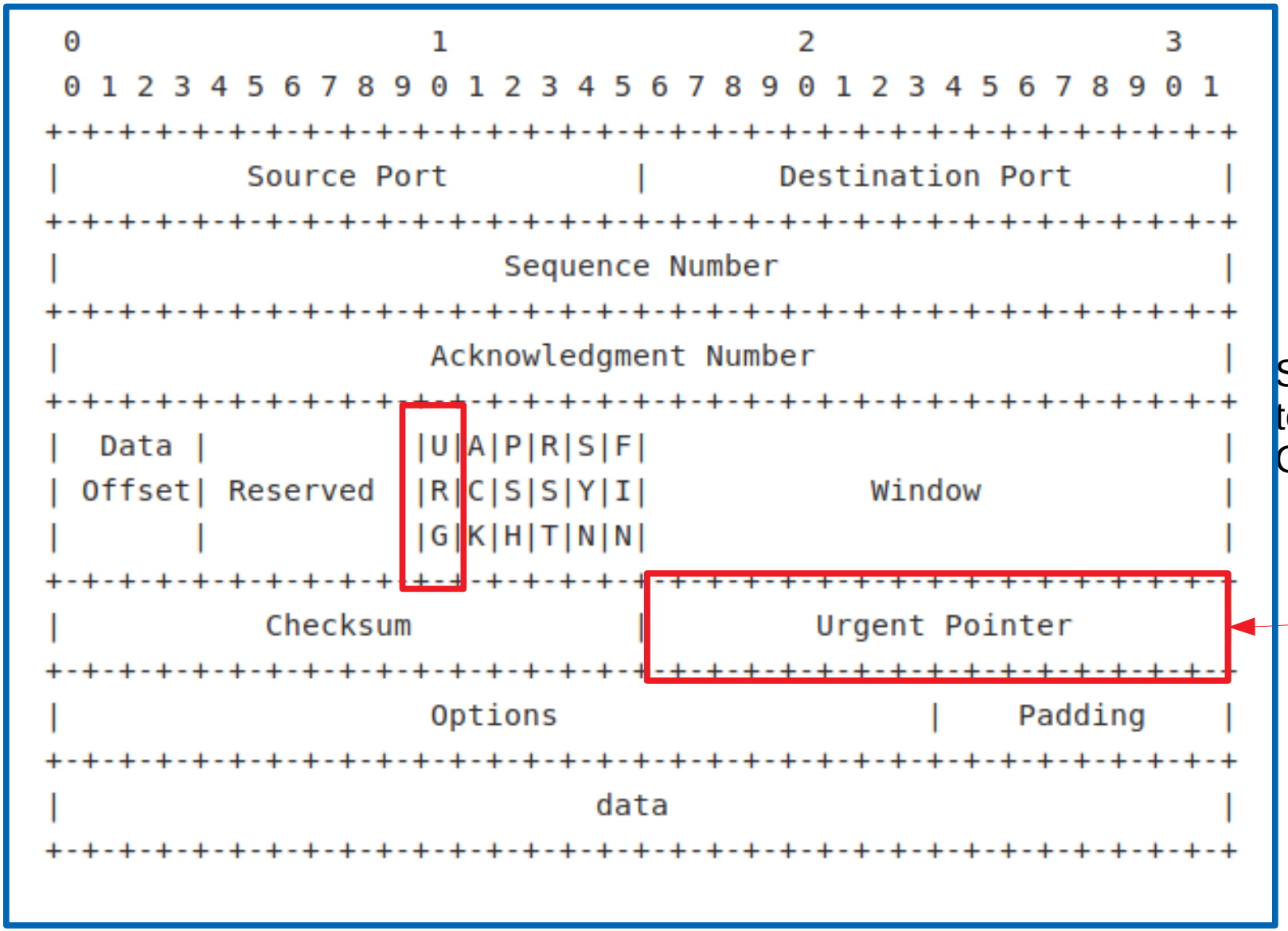


TCP/IP model	Protocols and services
Application	HTTP, FTTP, Telnet, NTP, DHCP, PING
Transport	TCP, UDP
Network	IP, ARP, ICMP, IGMP
Network Interface	Ethernet

TCP Encapsulation & Frames



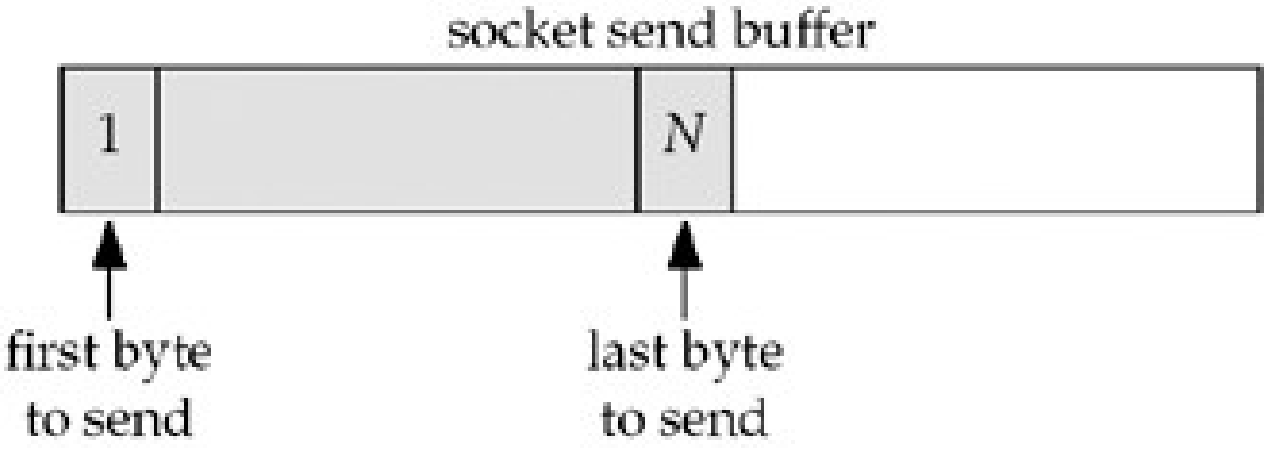
TCP Header



Seq-Number-Offset to first byte behind OOB byte

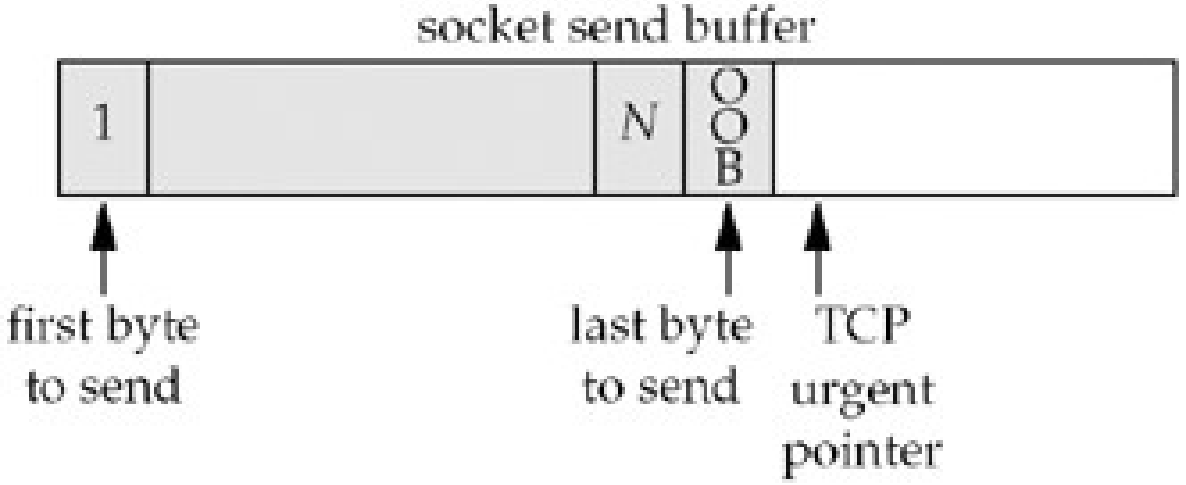
Socket send buffer containing data to send.

```
send(fd, buffer, N, 0);
```



Socket send buffer after one byte of out-of-band data is written by application

```
send(fd, "a", 1, MSG_OOB);
```



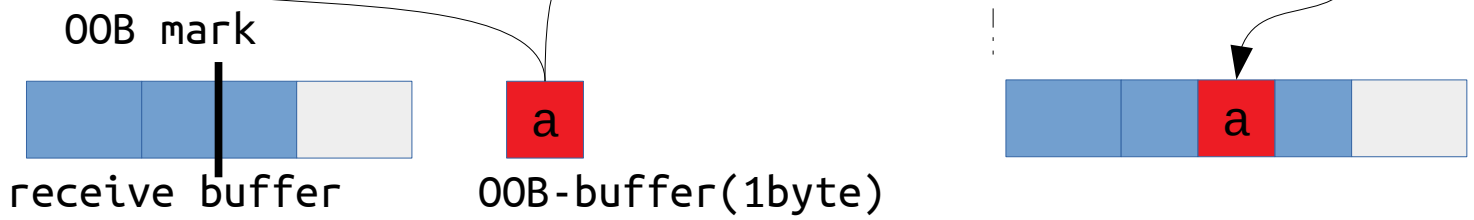
TCP Urgent Mode and Socket OOB Data(not SO_OOBINLINE)

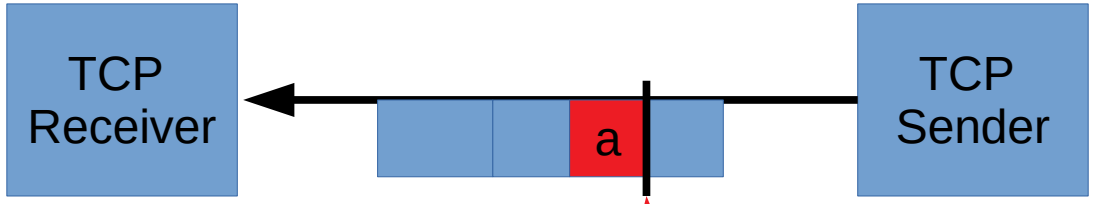
```
poll(fds)
..
if (…revent & POLLPRI)
  recv(fd,… ,MSG_OOB);
```

```
sigHandler() {
   recv(fd,… ,MSG_OOB);
}
```

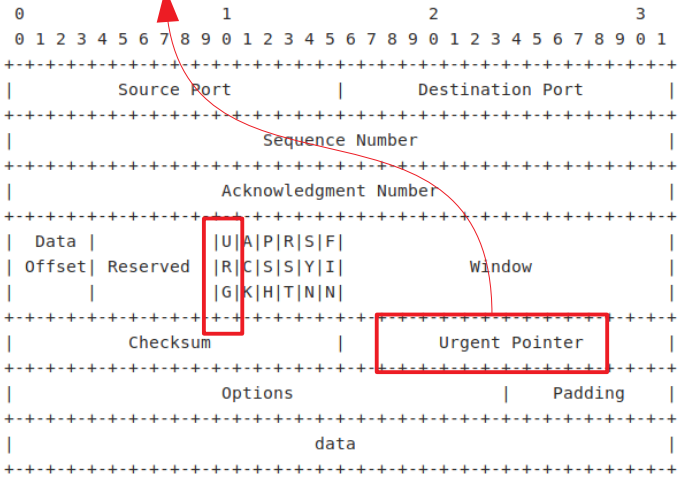
```
write(fd,…);
write(fd,…);
send(fd, “a”, MSG_OOB)
write(fd,…);
```

Socket API





TCP OOB concepts
- Notification
  - SIGURG signal
  - POLLPRI ready
- Position of OOB byte
- Value of OOB byte



Select Socket API

>> Vector **exceptfds** awaiting OOB-data events

///// C

```
#include <unistd.h>

int select(int nfds, fd_set *readfds,
   fd_set *writefds,fd_set *exceptfds,
   struct timeval *timeout);
```

///// Python (example)

```
readfds=[conn]; writefds=[]; exceptfs=[conn]
readable, writable, exceptional =
    select.select(readfds, writefds, exceptfds,
    timeout)
```

Poll Socket API

The POLL API was intended to replace the Select API

```
///// C
#include <poll.h>
int poll(struct pollfd *fds, nfds_t nfds, int
timeout);

// cases
if (fds[i].revents & POLLPRI) {...
```

Newer API no longer refers to “exception” But to PRIO-data

```
///// Python (example)
poller = select.poll()
poller.register(conn, select.POLLIN |
  select.POLLPRI | select.POLLHUP | select.POLLERR )
…
for fd, flag in poller.poll():
   if flag & socket.POLLPRI: …
```

Issues

- Two conflicting interpretations of RFC 793 (where the concept is introduced).
  - For example: The implementation of OOB data in the Berkeley Software Distribution (BSD) does not conform to the Host Requirements specified in RFC 1122.
- Limited to 1 byte.
- May be overwritten by subsequent OOB data.

Issues (cont)

- An "exception" signaling OOB data may occur even if the next read doesn't contain the OOB data (the network stack on the sender may flag any already queued data, so the other side will know there's OOB ASAP). This is often handled by entering a "drain" loop where you discard data until the actual OOB data is available.
- You don't need to handle it at the receiving end even if you are sending it - OOB data is transparently ignored in all circumstances unless you actively go about receiving it.

  !!! NULL exception-set and don’t define POLLPRI, otherwise unhandled OOB data will cause high IO-load.

Use Cases for TCP OOB

Signaling to peer

- Congestion (receiver shall enter “drain” loop)
- Abort large data transfers (used by FTP)

OOB-event being transfered/acknowledged to peer even if TCP window-size 0

TCP OOB – Client (Python)

```
#!/usr/bin/python3
import socket
import sys
import time

server_addr = ('localhost', 8884)
print('\nconnecting to {}'.format(server_addr), file=sys.stderr)
server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
sock.connect(server_addr)

sock.send('01234'.encode())
time.sleep(3)

sock.send('a'.encode(), socket.MSG_OOB)
time.sleep(3)

sock.send('56789'.encode())
sock.close()
```

TCP OOB – Server (Python)

```
#!/usr/bin/python3

import socket
import select

conn = accept_connection('localhost', 8884)

poller = select.poll()
poller.register(conn, select.POLLIN | select.POLLPRI | select.POLLHUP |
select.POLLERR )

while connected(conn):
    for fd, flag in poller.poll():
        if flag & select.POLLIN:
            data = conn.recv(1024)
            if data: handle(data)
            else: peer_closed_connection(conn)
        elif flag & select.POLLPRI:
            data = conn.recv(1, socket.MSG_OOB)
            if data: handle_oob_signal(data)
            else: peer_closed_connection(conn)
        elif flag & (select.POLLERR | select.POLLHUP):
           handle_io_failure(conn)
```

Wake up on OOB-data

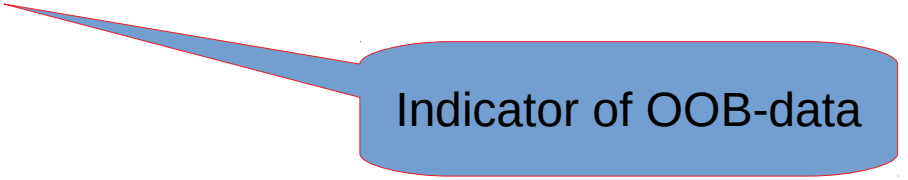
Even if POLLRI is set, at first read the regular data from Recv-buffer (drain)

Connection-Close is not POLLERR, but POLLIN and empty input!!

Did reach the POLLRI marker in stream, now read the OOB data.

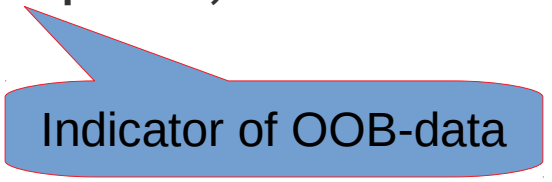
Common Misinterpretation of Select Exceptions

```
inputs=[conn]
outputs=[]

while inputs:

    # Wait for at least one of the sockets to be ready for processing

    readable, writable, exceptional = select.select(inputs, outputs, inputs)

    …
    for s in readable:
      data = s.recv(1024)
      if data: handle(data)
      elif: handle_peer_close(s) /* no data available although readable */

    …
    for s in exceptional:
      # many implementations panic here, closing the connection
```



Better: Ignore Select Exceptions at all

```
inputs=[conn]
outputs=[]
oob_exceptions=[] # empty list, ignore OOB data events at all

while inputs:

    # Wait for at least one of the sockets to be ready for processing

    readable, writable, exceptions_never =
             select.select(inputs, outputs, oob_exceptions)

    …
    for s in readable:
      data = s.recv(1024)
      if data: handle(data)
      elif: handle_peer_close(s) /* no data available although readable */

    …
    for s in exceptions_never:
      # never reached
```



Known Attack

WinNuke

OOB message causing blue screen of death on Microsoft Windows 95, Microsoft Windows NT

Reference

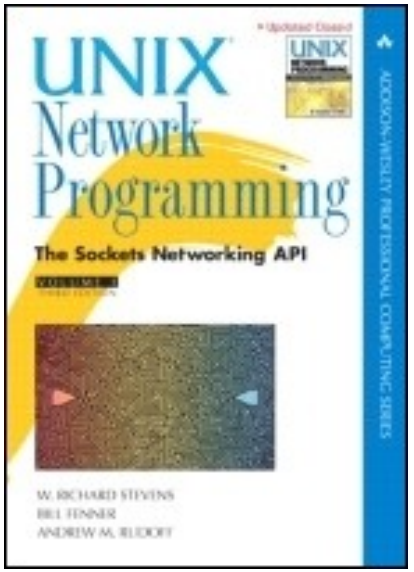
UNIX® Network Programming Volume 1, Third Edition: The Sockets Networking API

By W. Richard Stevens, Bill Fenner, Andrew M. Rudoff

Publisher : Addison Wesley

Pub Date : November 21, 2003

ISBN : 0-13-141155-1



Thank you

Frank Rehberger
info@frehberg.com
https://frehberg.com
#nosegv