Go SCTP!

by

Olivier Van Acker

Who am I?

Software engineer from London

Birkbeck
UNIVERSITY OF LONDON

Student

Broadcasting industry

zeebox
The best thing to happen to TV since TV

BBC

VPRO

What is this talk about?

Shiny new + Shiny new = Double Fun

Two new technologies

Exploring

Stream Control Transmission Protocol

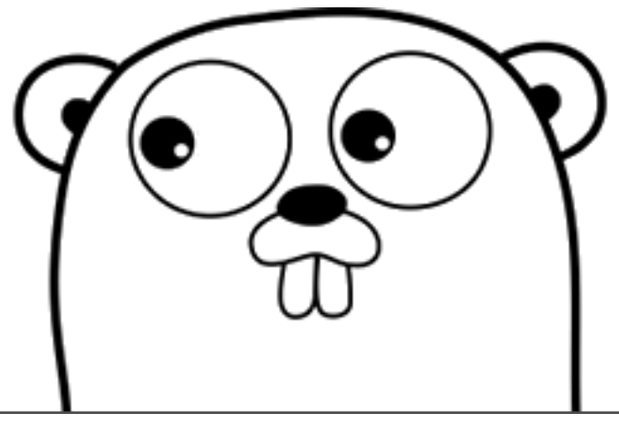
&

Go

Why FreeBSD?

Overview

SCTP & Go



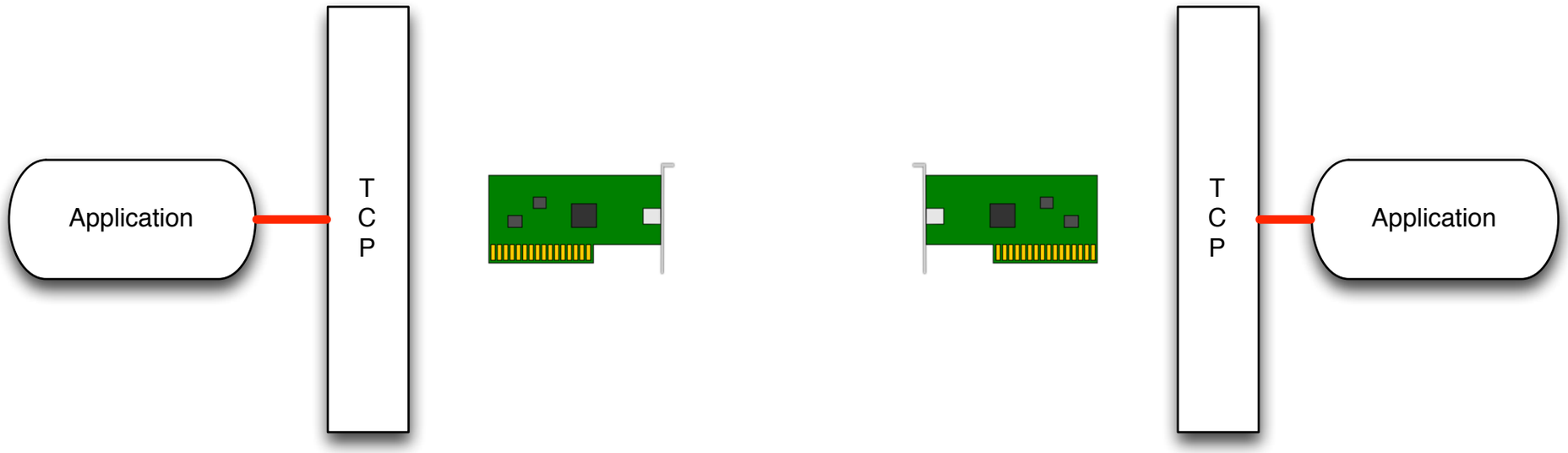
Birds eye view SCTP

Main characteristics

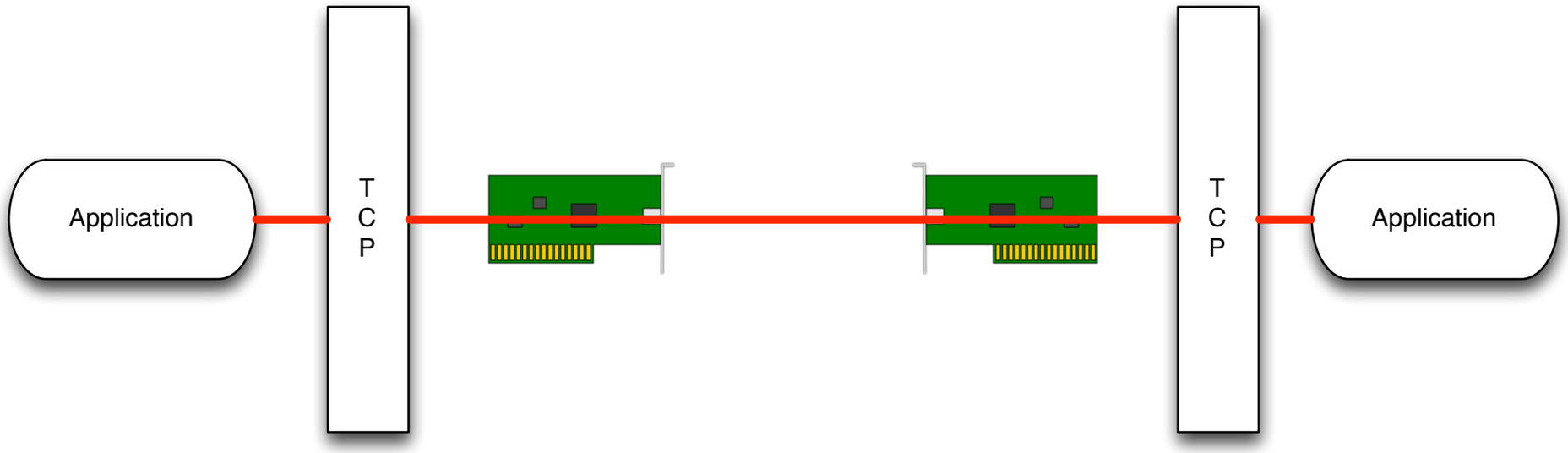
- Reliable transport protocol on top of IP
- Message based
- Multi-stream

Associations

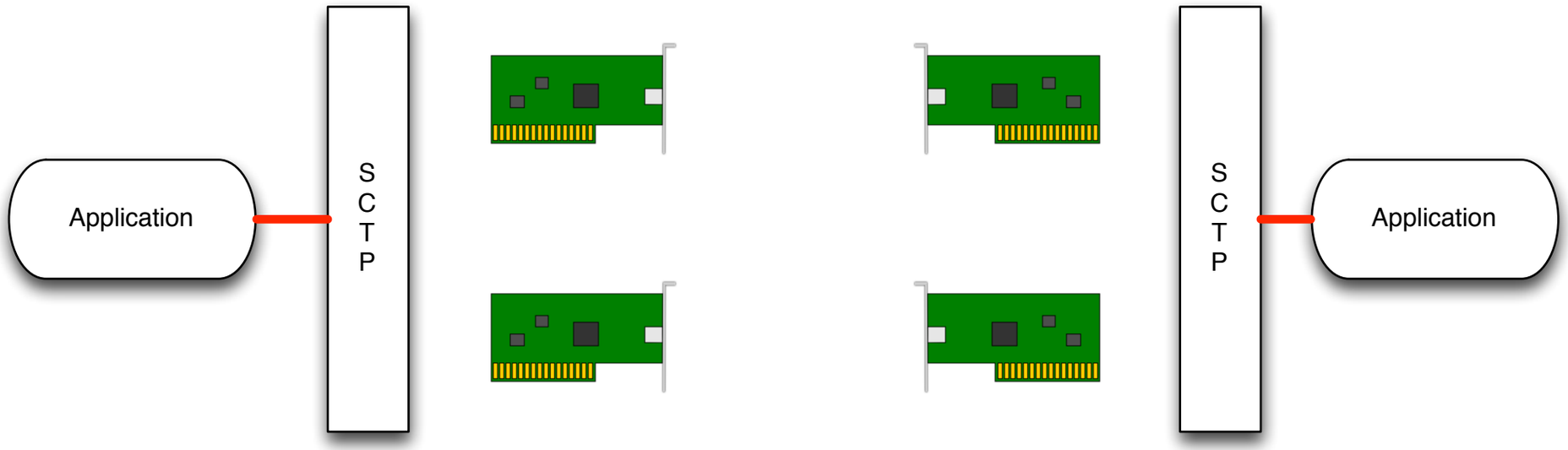
TCP connection



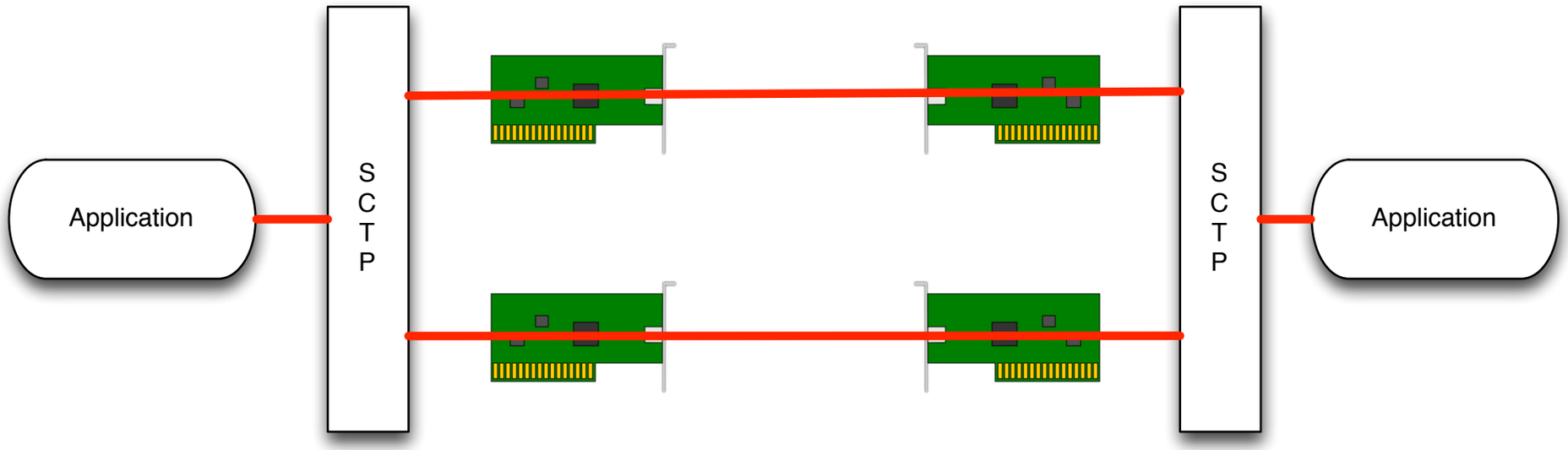
TCP connection



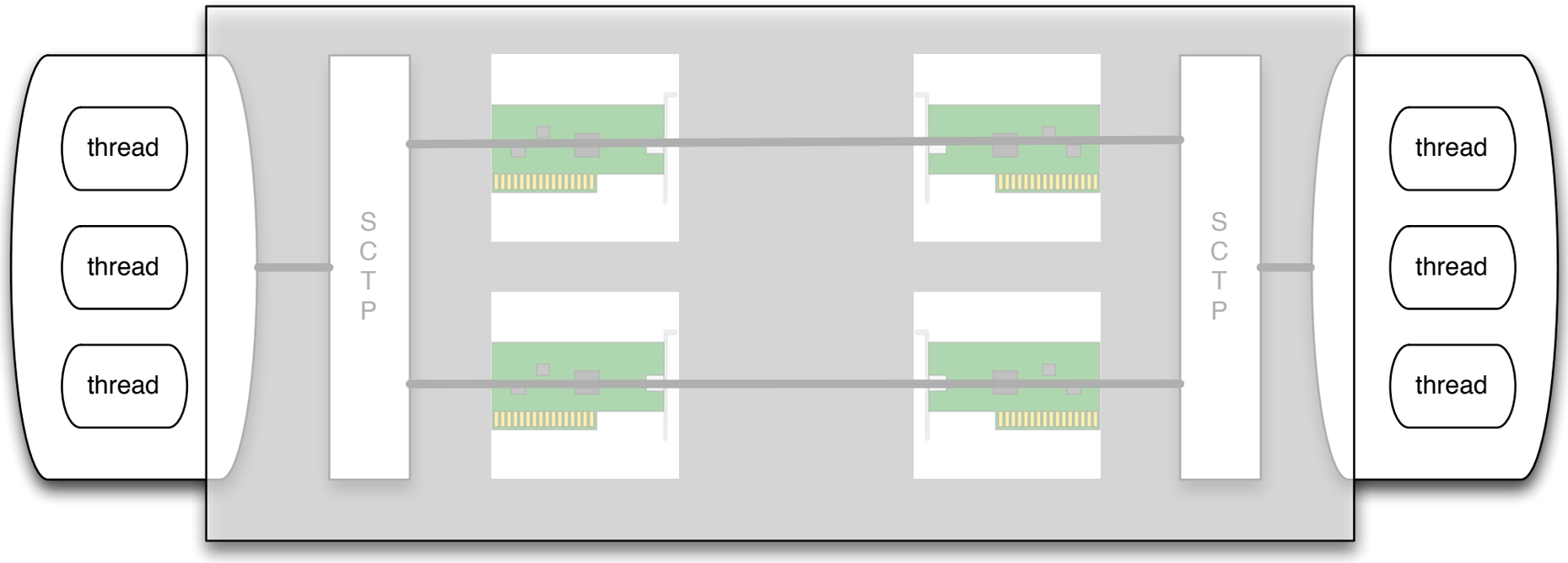
SCTP association



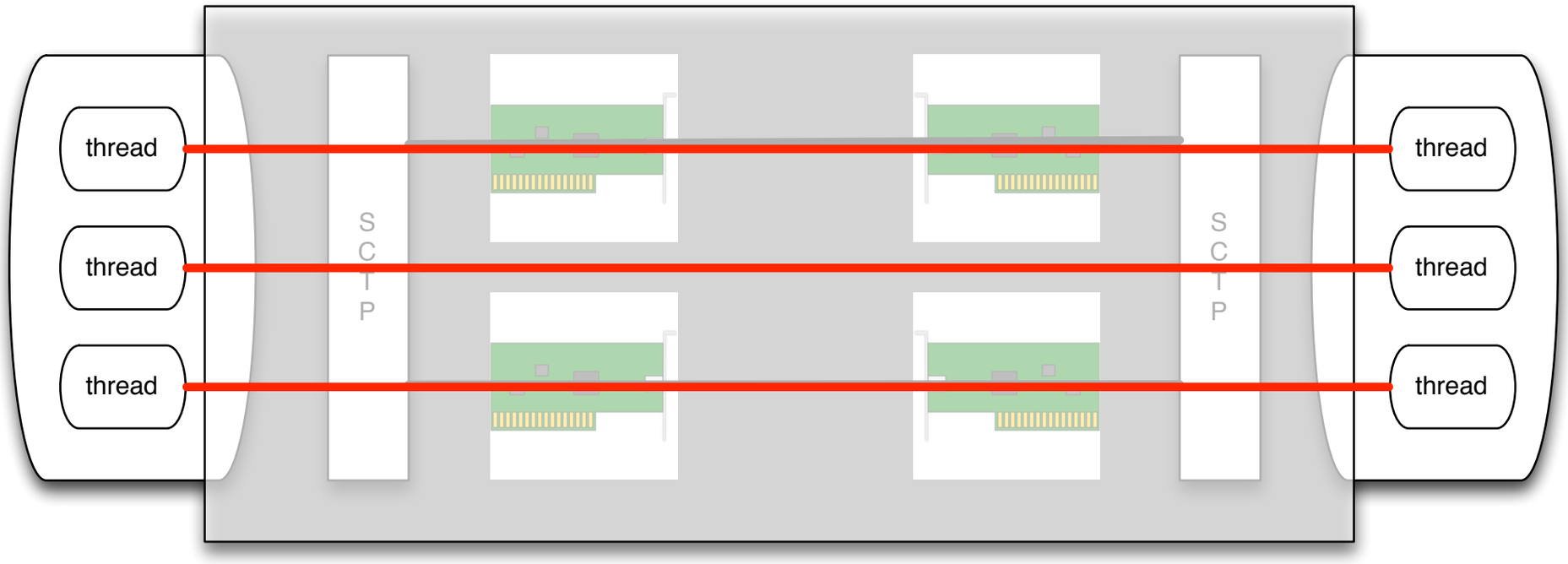
Multi-homing



Another abstraction

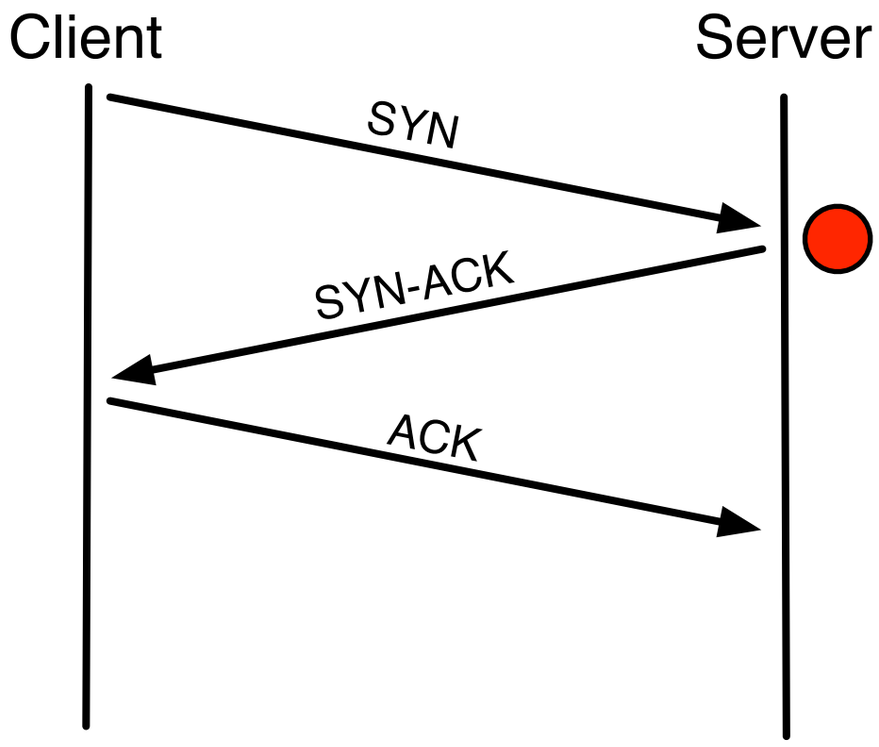


Streams

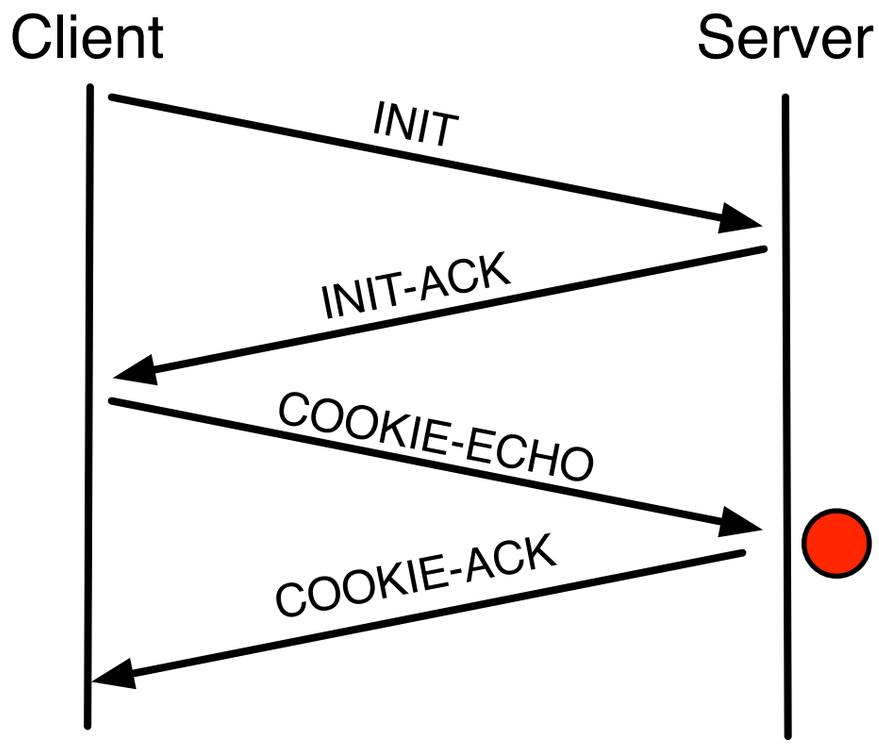


Messages

Connection



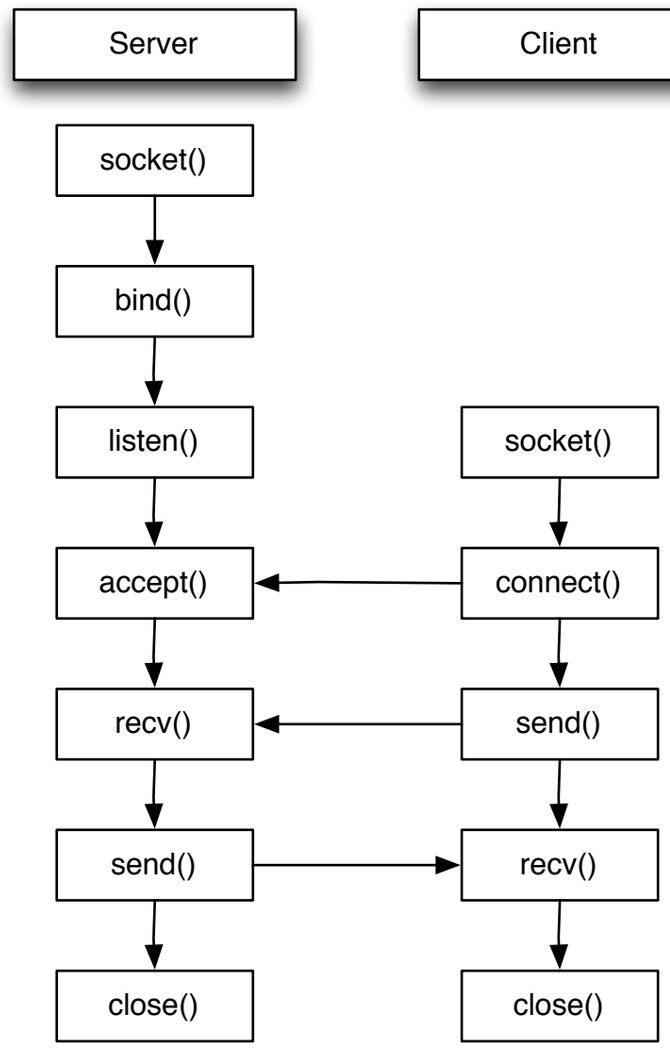
TCP 3 way handshake



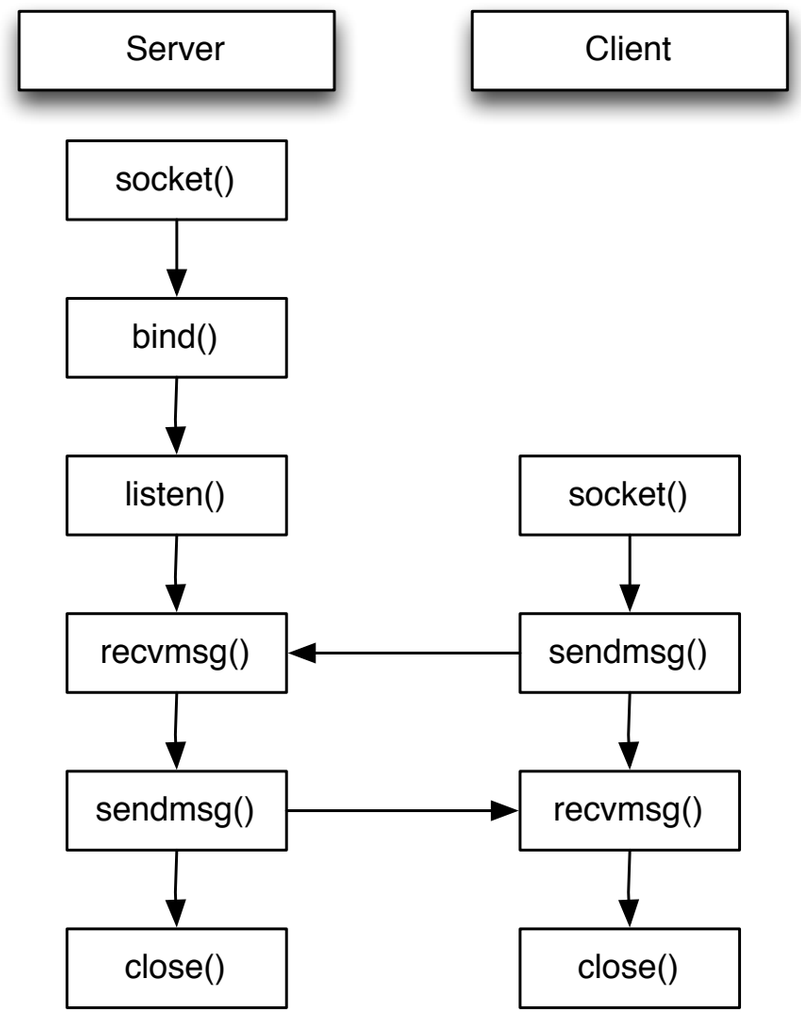
SCPT 4 way handshake

Two styles

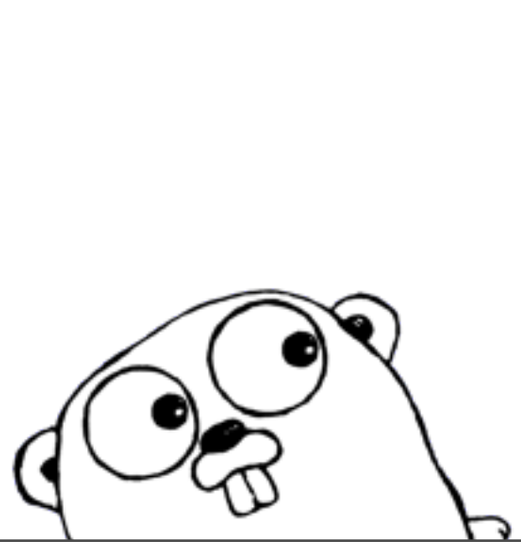
One to one



One to many



Birds eye view Go



Example

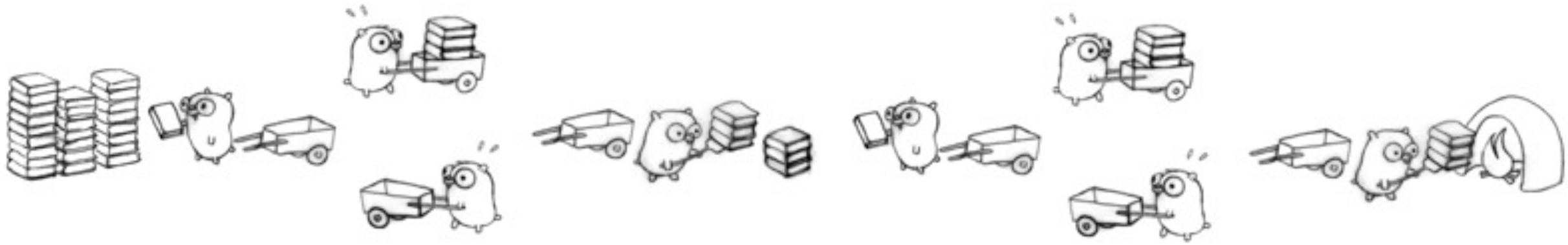
```go
func main() {
  m := image.NewRGBA(image.Rect(0, 0, 400, 400))
  blue := color.RGBA{0, 0, 0, 255}
  draw.Draw(m, m.Bounds(), &image.Uniform{blue}, image.ZP, draw.Src)
  drawFern(m, 400, 400, 10000)
  f, _ := os.OpenFile("fern.png", os.O_CREATE | os.O_WRONLY, 0666)
  png.Encode(f, m); err != nil
}

func drawFern(m *image.RGBA, x float32, y float32, steps int) {
  if steps != 0 {
    x, y = transform(x, y)
    drawPoint(m, x, y)
    drawFern(m, x, y, steps - 1)
  }
}

func drawPoint(m *image.RGBA, x float32, y float32) {
  b := m.Bounds()
  height := float32(b.Max.Y)
  width := float32(b.Max.X )
  scale := float32(height / 11)
  y = (height - 25) - (scale * y)
  x = (width / 2) + (scale * x)
  m.Set(int(x), int(y), color.RGBA{0, 255, 0, 255})
}

func transform(x float32, y float32) (float32, float32) {
  rnd := rand.Intn(101)
  switch {
    case rnd == 1: x, y = transformPoint(x,y, 0.0, 0.0, 0.0, 0.16, 0.0)
    case rnd <= 7: x, y = transformPoint(x,y, 0.2, -0.26, 0.23, 0.22, 0.0)
    case rnd <= 14: x, y = transformPoint(x,y, -0.15, 0.28, 0.26, 0.24, 0.44)
    case rnd <= 100: x, y = transformPoint(x,y, 0.85, 0.04, -0.04, 0.85, 1.6)
  }
  return x, y
}

func transformPoint(x, y, a, b, c ,d , s float32) (float32, float32) {
  return ((a * x) + (b * y)), ((c * x) + (d * y) + s)
}
```

C syntax

```
01 func transform(x float32, y float32) (float32, float32) {
02   rnd := rand.Intn(101)
03   switch {
04     case rnd == 1: x, y = transformPoint(x,y, 0.0, 0.0, 0.0, 0.16, 0.0)
05     case rnd <= 7: x, y = transformPoint(x,y, 0.2, -0.26, 0.23, 0.22, 0.0)
06     case rnd <= 14: x, y = transformPoint(x,y, -0.15, 0.28, 0.26, 0.24, 0.44)
07     case rnd <= 100: x, y = transformPoint(x,y, 0.85, 0.04, -0.04, 0.85, 1.6)
08   }
09   return x, y
10 }
```

Polymorphism

```go
package main

import (
  "fmt"
)

type animal interface {
  Talk()
}

type Cat

func (c Cat) Talk() {
  fmt.Println("Meeow")
}

func main() {
  var c Cat
  c.Talk()
  a := animal(c)
  a.Talk()
}
```



Goroutines

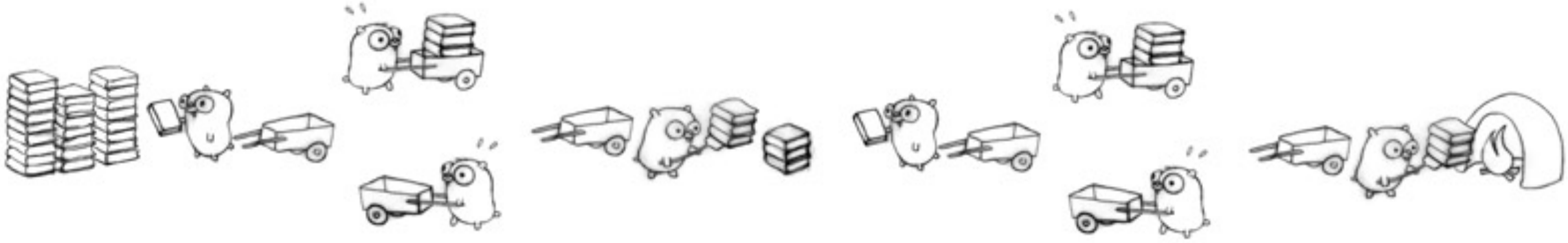
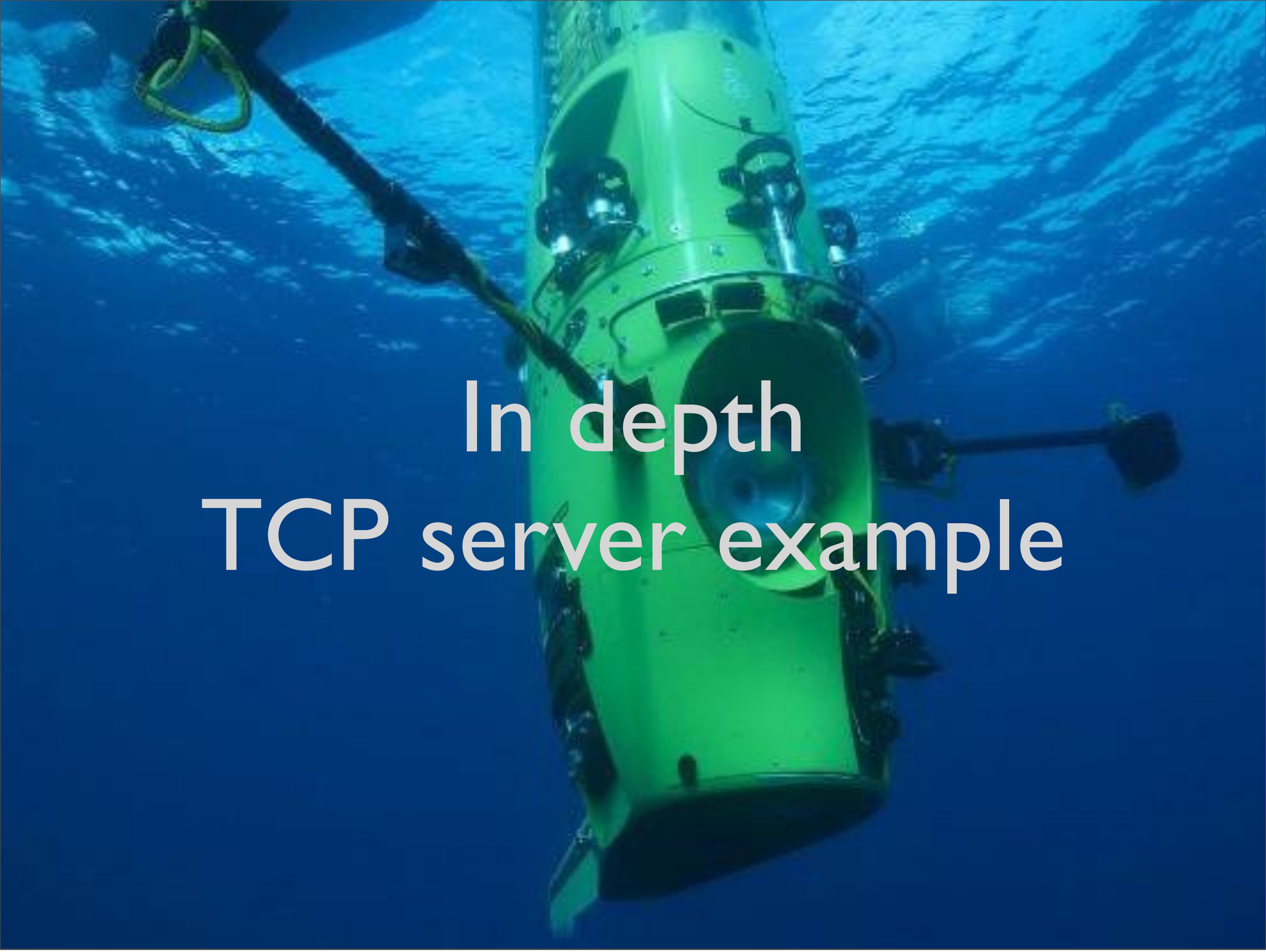


Image source: http://concur.rspace.googlecode.com/hg/talk/concur.html#slide-24

```
01    for {
02      c, err := netlisten.Accept()
03      if err != nil {
04        log.Printf("Error accepting: %s", err)
05        continue
06      }
07      go handleClient(c, Message)
08    }
```

How does Go make a system call?

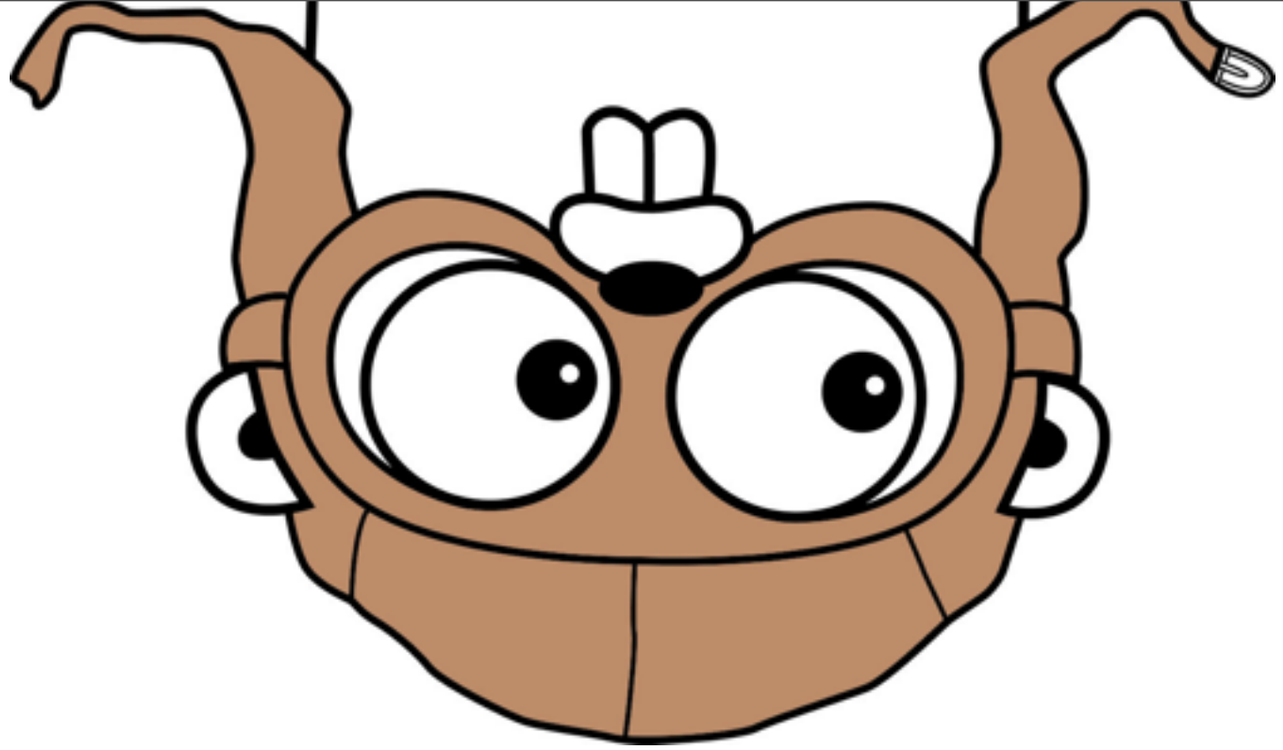


In depth

TCP server example

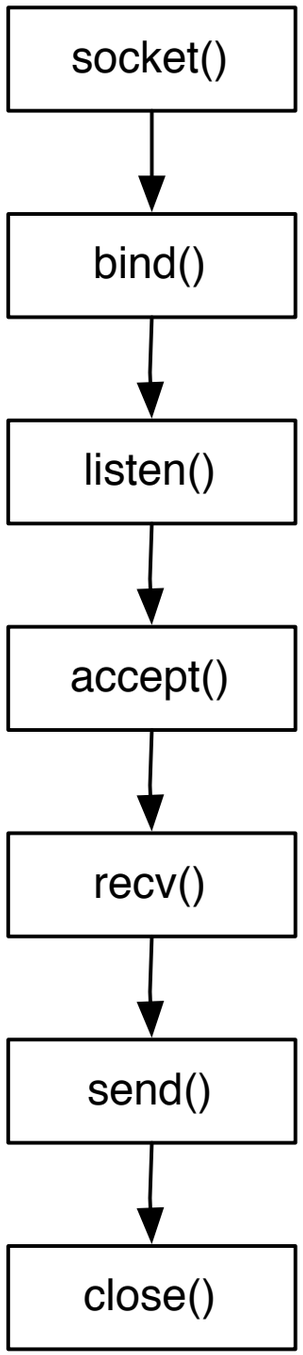
Echo server

```go
func echoServer(Address *string, Message *string) {
  var netlisten net.Listener
  var err error

  netlisten, err = net.Listen("tcp", *settings.Address)

  if err != nil {
    log.Printf("Error listening: %s", err)
    os.Exit(-1)
  }
  defer netlisten.Close()

  for {
    c, err := netlisten.Accept()
    if err != nil {
      log.Printf("Error accepting: %s", err)
      continue
    }
    go handleClient(c, Message)
  }
}
```

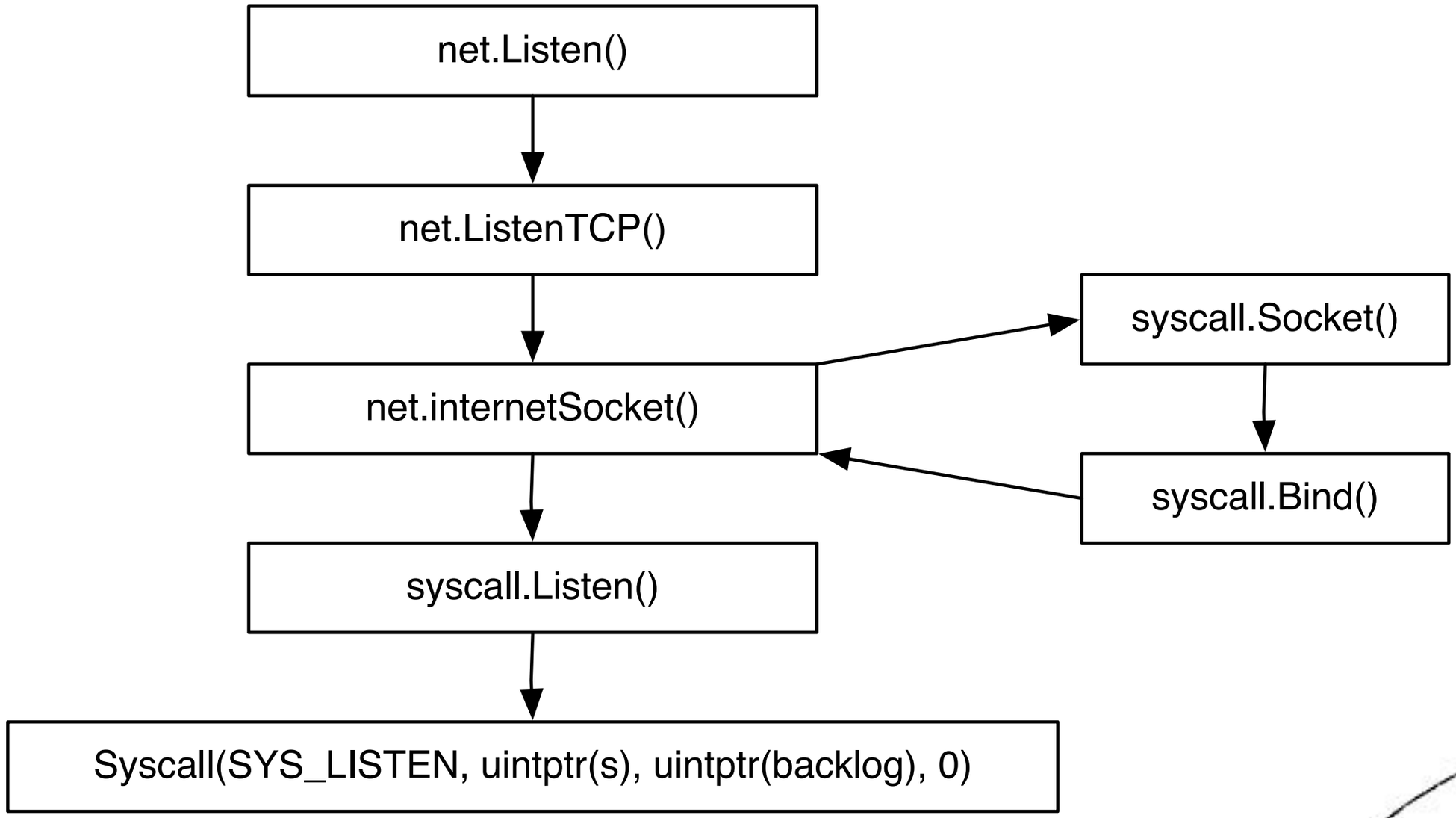


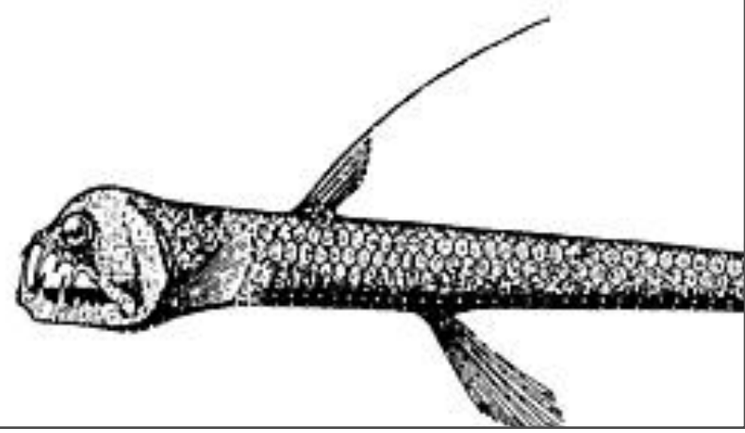
Listen

Reminder



To the bottom





Handcrafted assembly

```
 1 TEXT  ·Syscall(SB),7,$0
 2   CALL  runtime·entersyscall(SB)
 3   MOVQ  16(SP), DI
 4   MOVQ  24(SP), SI
 5   MOVQ  32(SP), DX
 6   MOVQ  $0, R10
 7   MOVQ  $0, R8
 8   MOVQ  $0, R9
 9   MOVQ  8(SP), AX // syscall entry
10   SYSCALL
11   JCC ok
12   MOVQ  $-1, 40(SP) // r1
13   MOVQ  $0, 48(SP)  // r2
14   MOVQ  AX, 56(SP)  // errno
15   CALL  runtime·exitsyscall(SB)
16   RET
17 ok:
18   MOVQ  AX, 40(SP)  // r1
19   MOVQ  DX, 48(SP)  // r2
20   MOVQ  $0, 56(SP)  // errno
21   CALL  runtime·exitsyscall(SB)
22   RET
```

Auto generated code

SCTP API

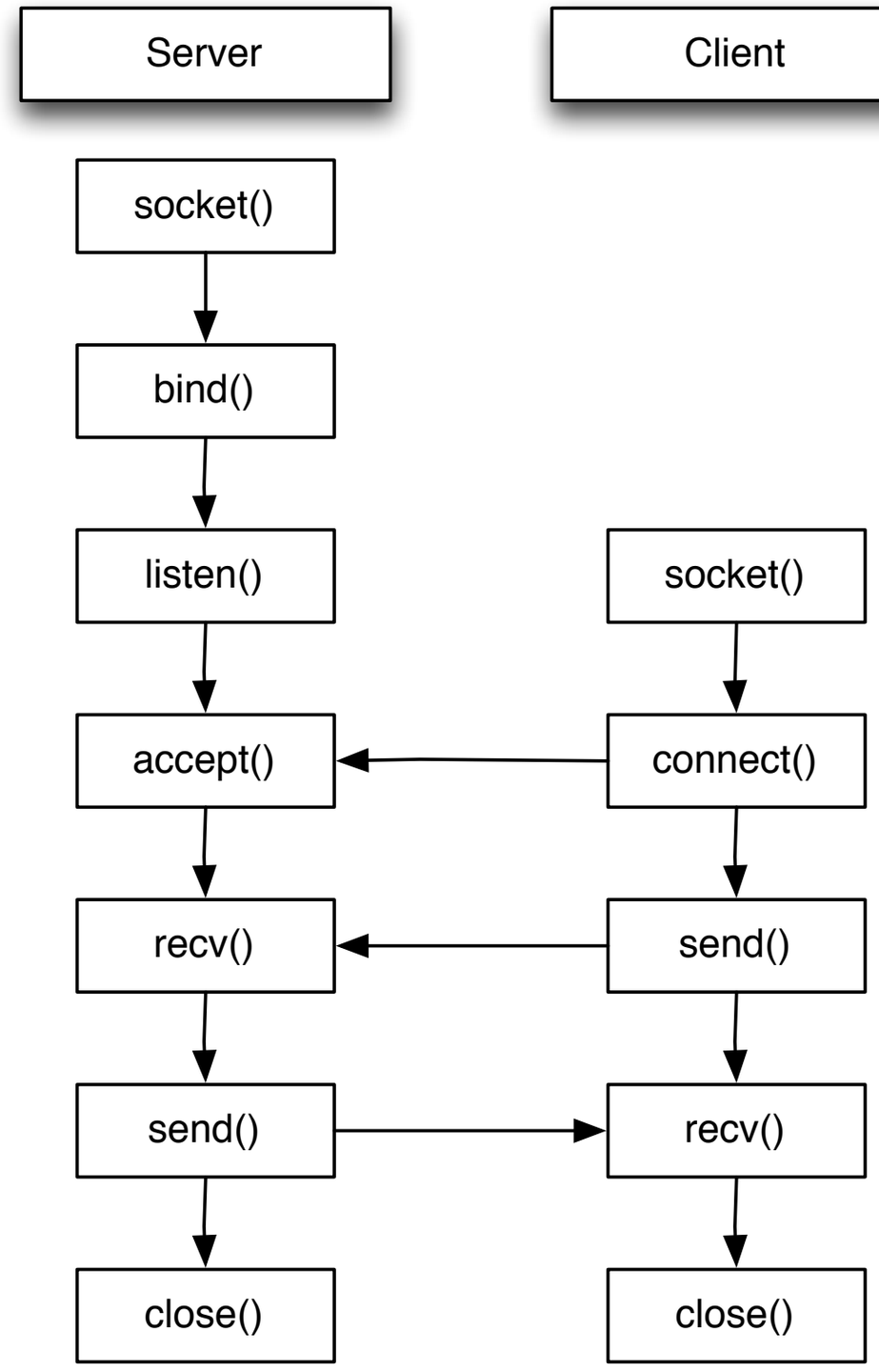
RFC 6458

one to one

(almost)

no difference with TCP

One to one

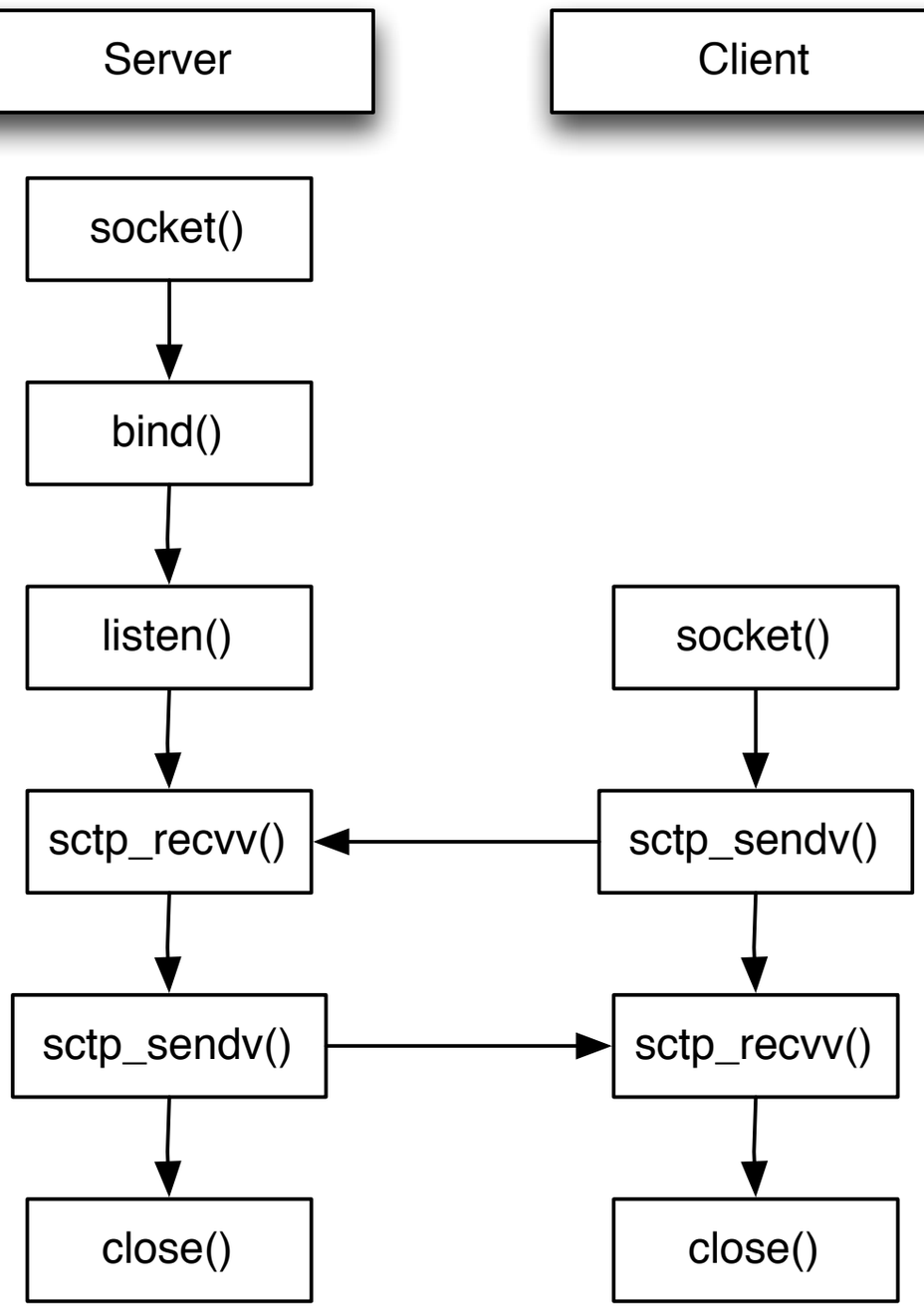

Server
Client
socket()
bind()
listen()
socket()
accept()
connect()
recv()
send()
send()
recv()
close()
close()

Code Change

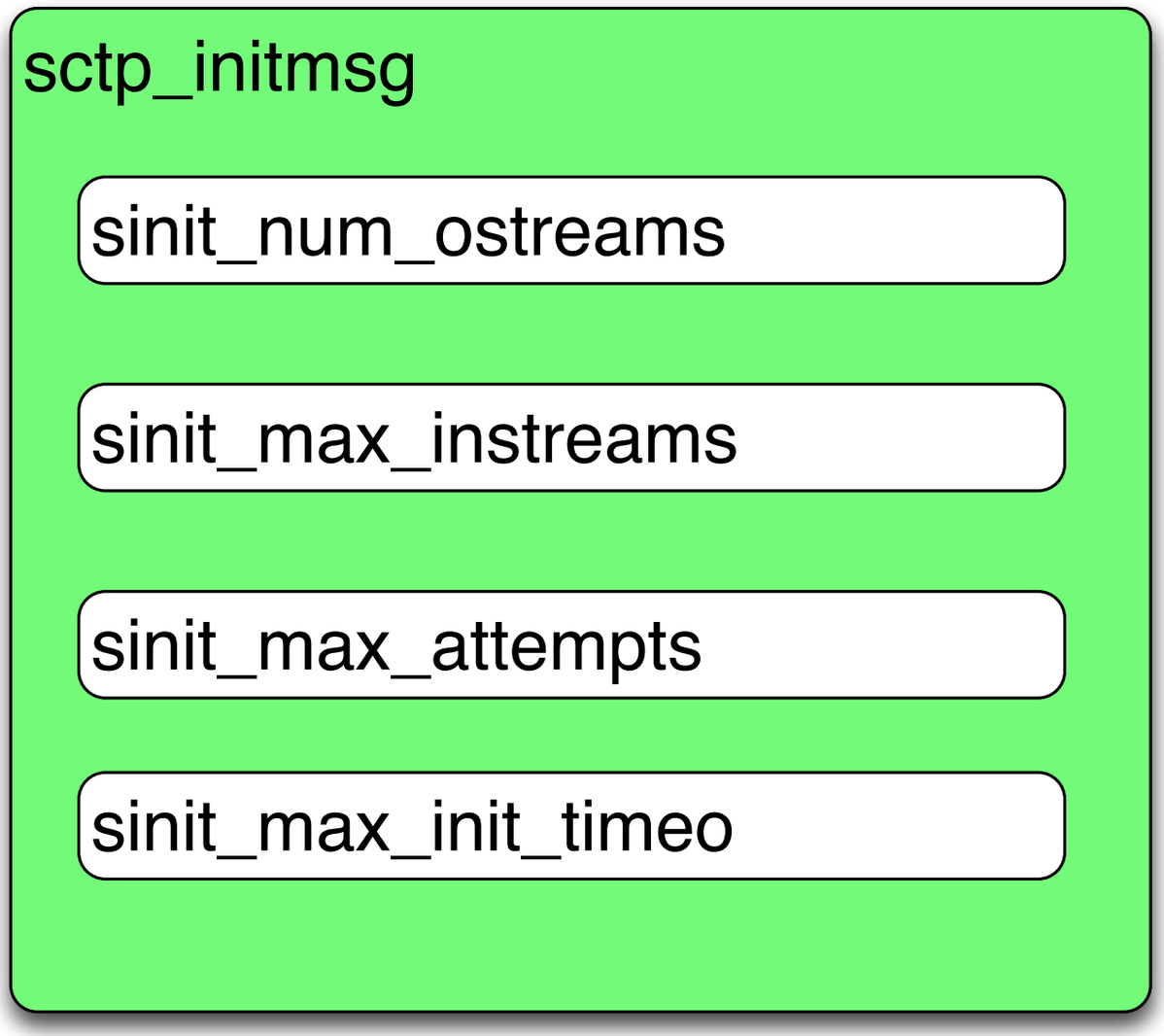
```
if(settings.sctp) {
  sd = socket(PF_INET, SOCK_STREAM, IPPROTO_SCTP);
} else {
  sd = socket(PF_INET, SOCK_STREAM, IPPROTO_TCP);
}
```

One to many

One to many


Server
Client
socket()
bind()
listen()
socket()
sctp_recvv()
sctp_sendv()
sctp_sendv()
sctp_recvv()
close()
close()

Initiation structure



API Calls

Function	Description
sctp_sendv()	Send a message
sctp_recvv()	Receive a message
sctp_bindx()	Bind to a subset of addresses
sctp_connectx()	Connect to a peer via multiple addresses
sctp_peeloff()	Separate an association into a separate socket
sctp_getpaddrs()	Return all peer addresses
sctp_freeoaddrs()	Frees all resources
sctp_getladdrs()	Return all local addresses
sctp_freeladdrs()	Frees all local addresses

Wrappers

sctp_sendv() = sendmsg()

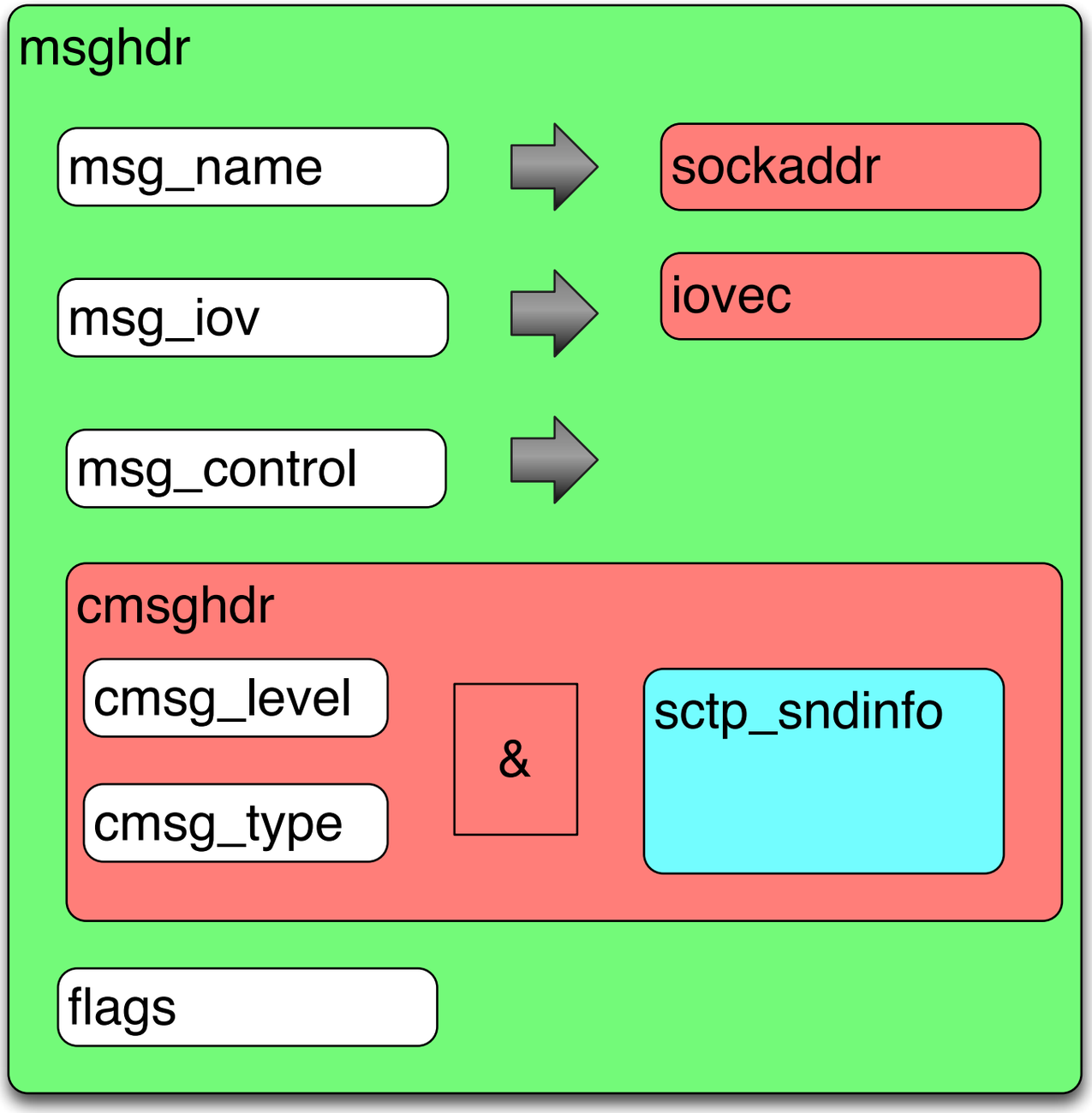
&

sctp_recvv() = recvmsg()

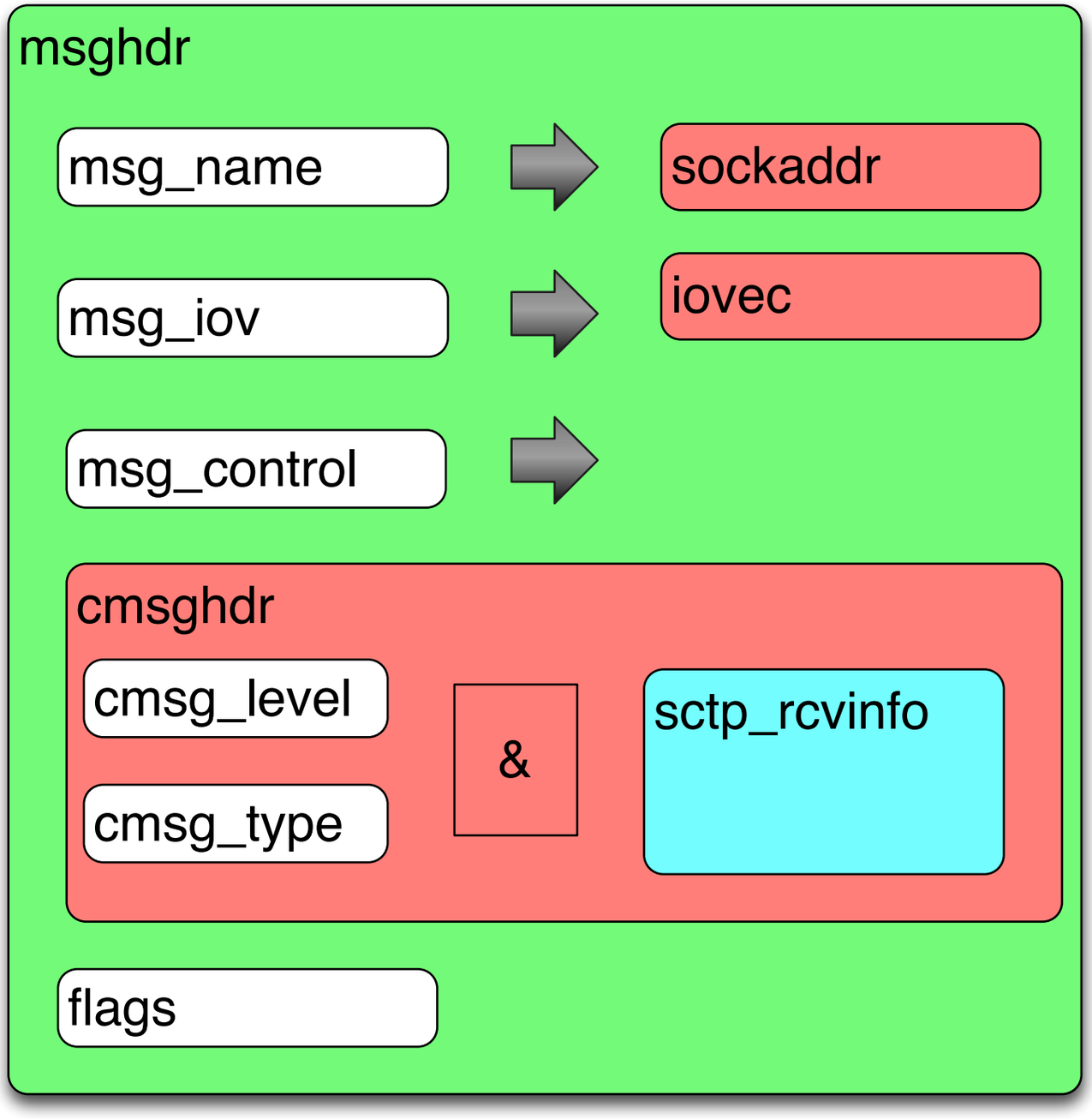
Ancillary data

```
ssize_t
sendmsg(int s, const struct msghdr *msg, int flags);

ssize_t
recvmsg(int s, struct msghdr *msg, int flags);
```

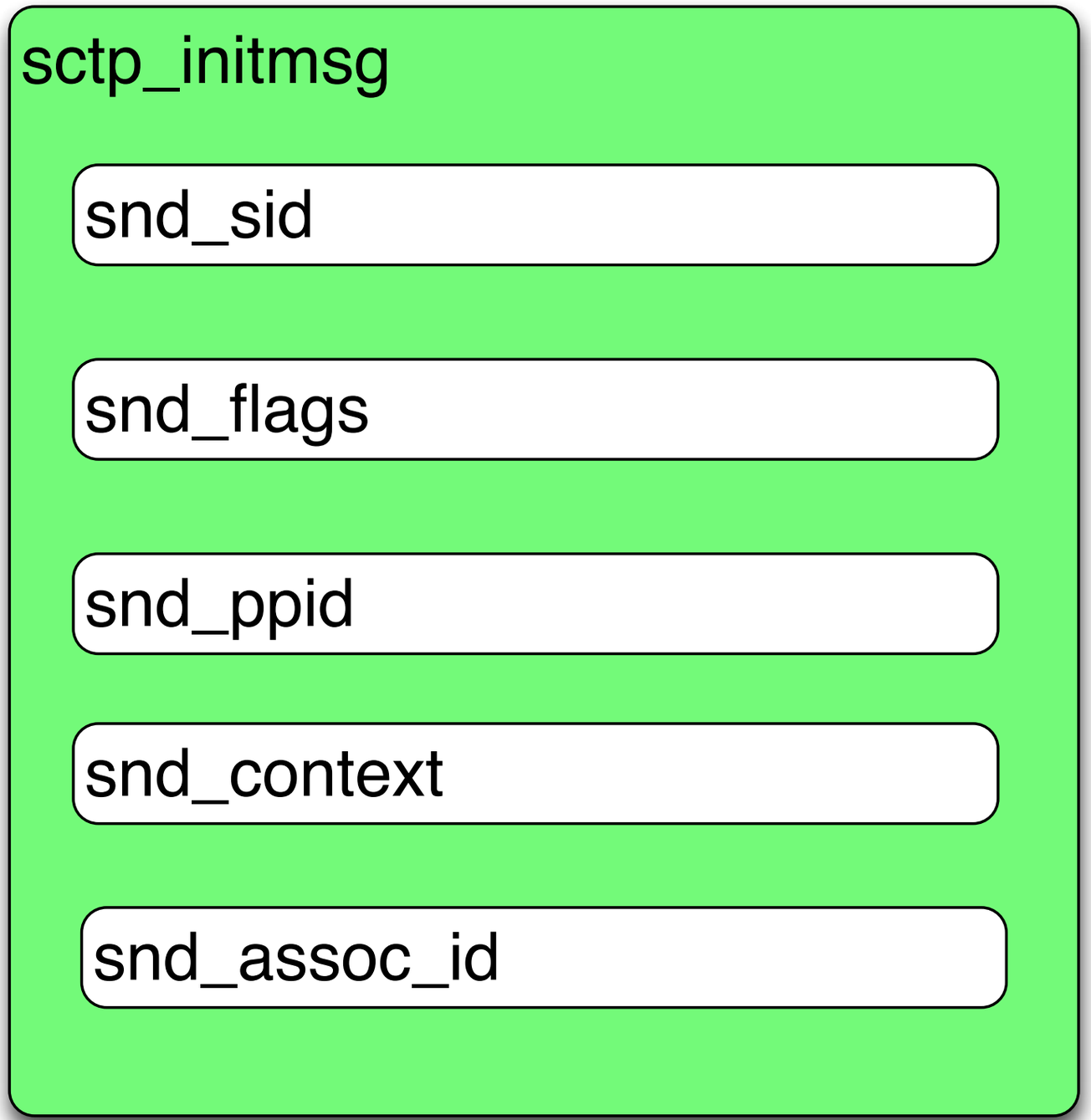
Structure for sending the message



Structure for receiving message



Send information structure



How to do this in Go?

Version 1

```
hg clone -u release https://code.google.com/p/go
```

Add structures

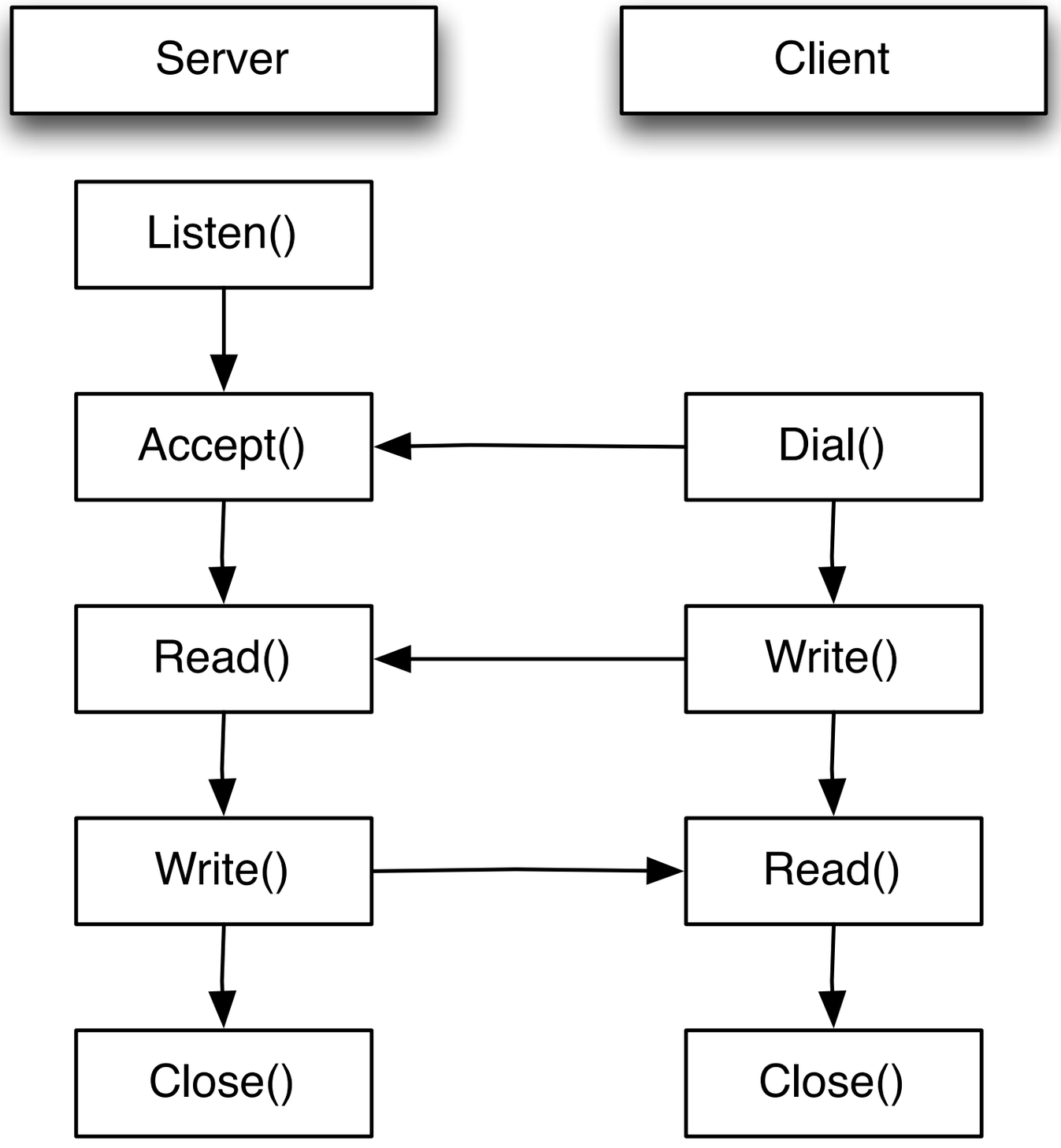
syscall/types_freebsd.go

```
#include <netinet/sctp.h>

 type SCTPInitMsg C.struct_sctp_initmsg
 type SCTPSndInfo C.struct_sctp_sndinfo
 type SCTPRcvInfo C.struct_sctp_rcvinfo

const (
   SizeofSCTPSndInfo = C.sizeof_struct_sctp_sndinfo
   SizeofSCTPRcvInfo = C.sizeof_struct_sctp_rcvinfo
   SizeofSCTPInitMsg = C.sizeof_struct_sctp_initmsg
)
```

syscall/ztypes_freebsd.go

```
type SCTPSndInfo struct {
    Sid      uint16
    Flags    uint16
    Ppid     uint32
    Context  uint32
    Assoc_id uint32
}
```

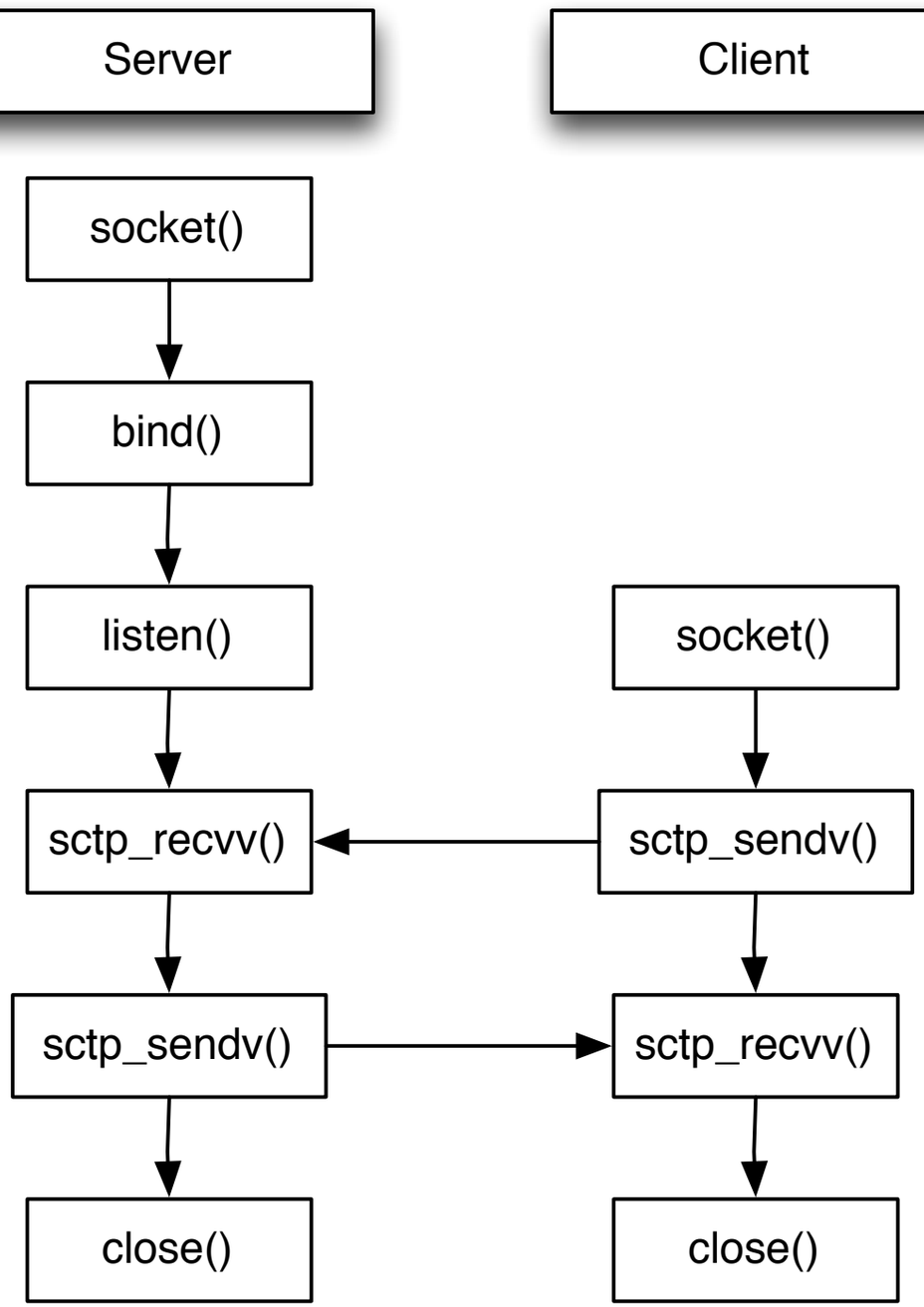
One to one


Server
Client
Listen()
Accept()
Dial()
Read()
Write()
Write()
Read()
Close()
Close()

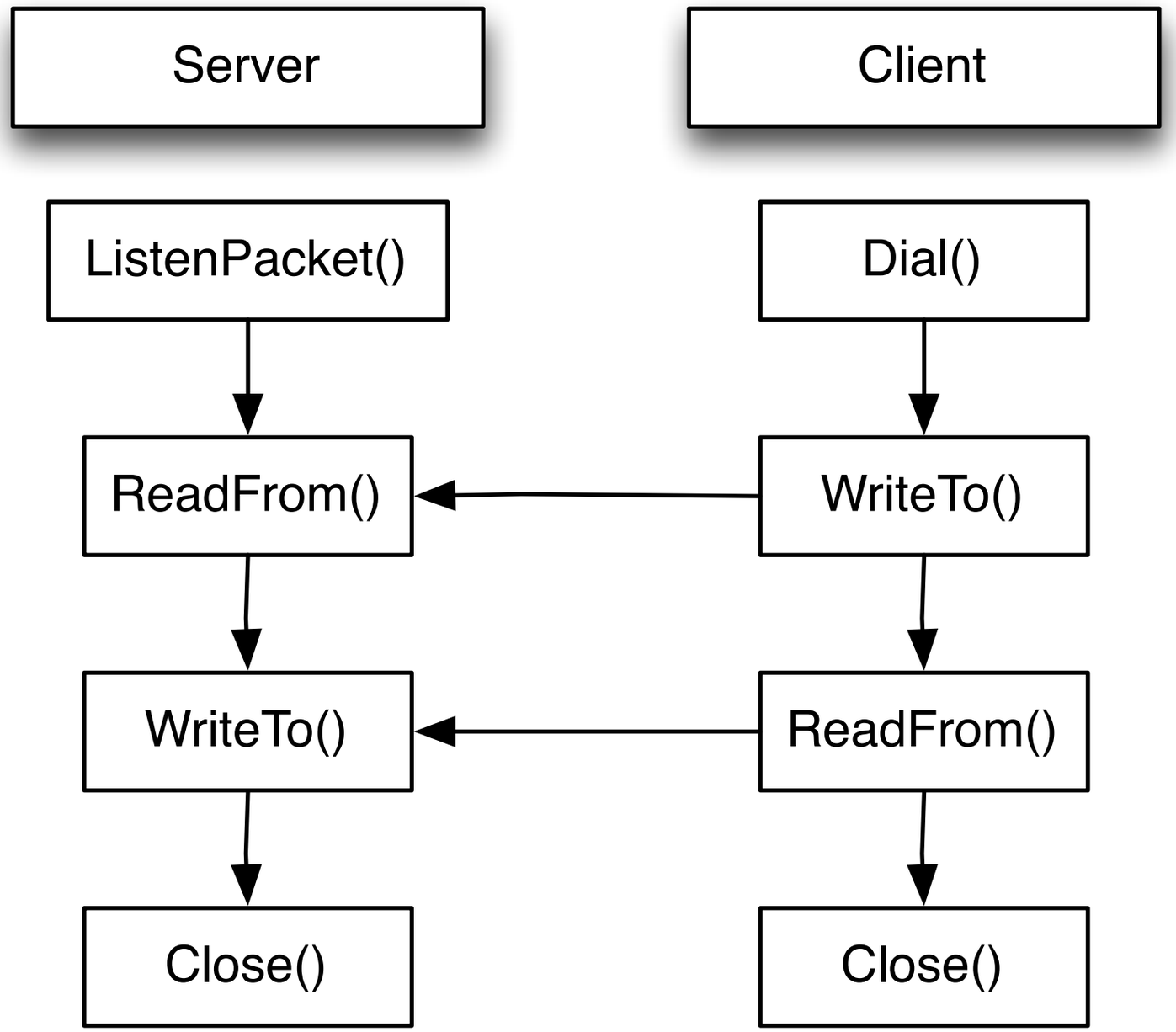
net.Conn implementation

```
type Conn interface {
    Read(b []byte) (n int, err error)
    Write(b []byte) (n int, err error)
    Close() error
    LocalAddr() Addr
    RemoteAddr() Addr
    SetDeadline(t time.Time) error
    SetReadDeadline(t time.Time) error
    SetWriteDeadline(t time.Time) error
}
```

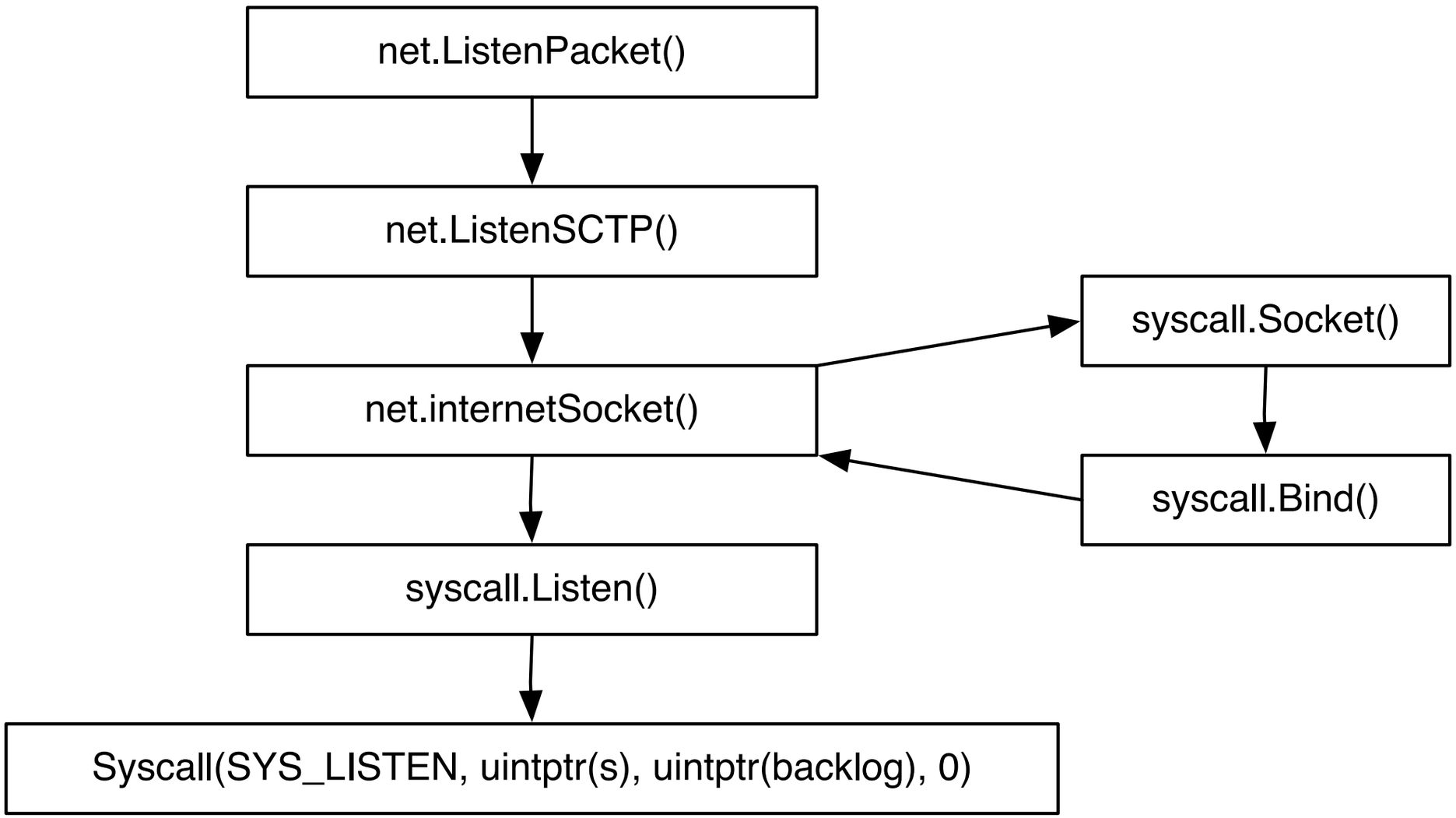
One to many


Server
Client
socket()
bind()
listen()
socket()
sctp_recvv()
sctp_sendv()
sctp_sendv()
sctp_recvv()
close()
close()

One to many


Server
Client
ListenPacket()
Dial()
ReadFrom()
WriteTo()
WriteTo()
ReadFrom()
Close()
Close()

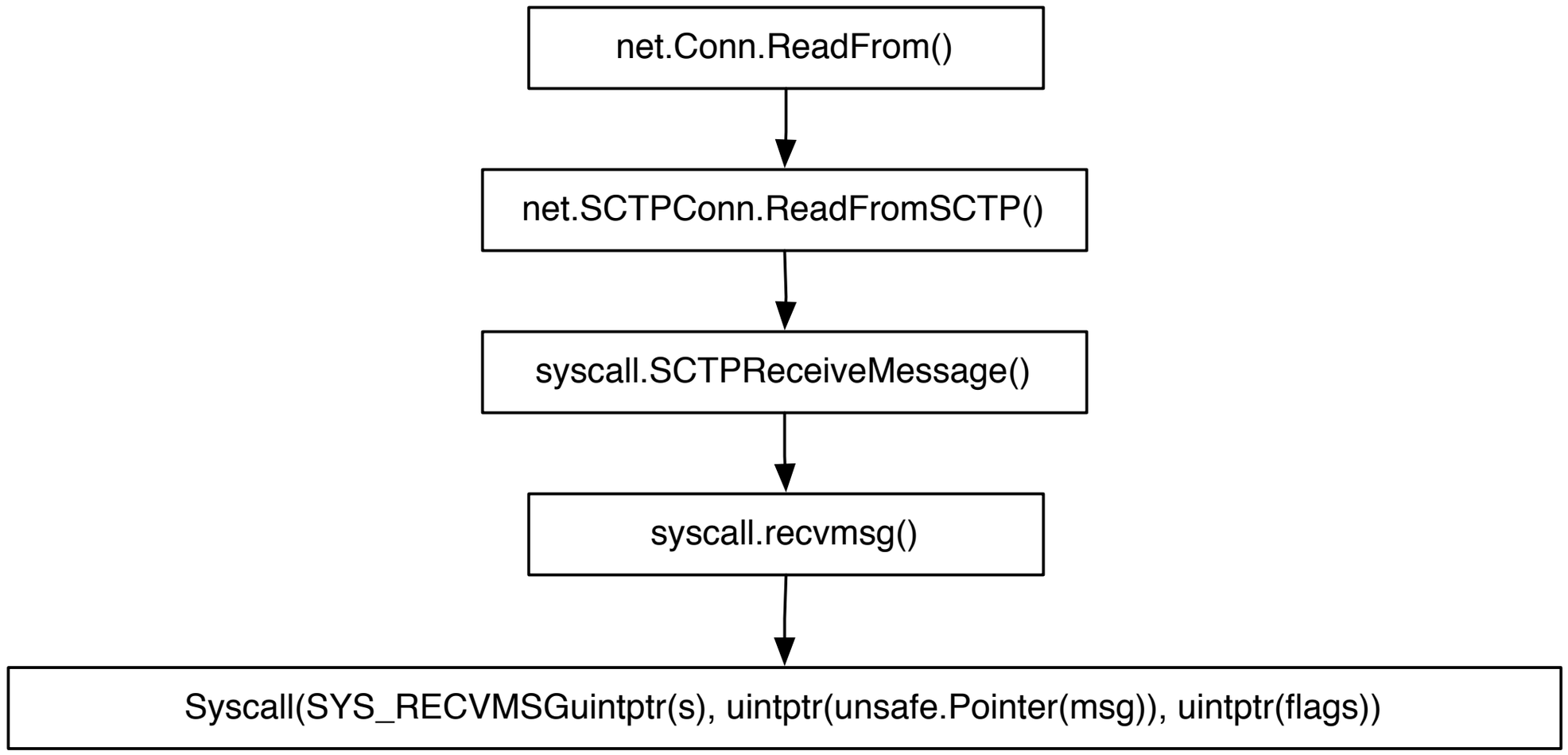
Setting up


net.ListenPacket()
net.ListenSCTP()
net.internetSocket()
syscall.Socket()
syscall.Bind()
syscall.Listen()
Syscall(SYS_LISTEN, uintptr(s), uintptr(backlog), 0)

PacketConn Interface

```
type PacketConn interface {
    ReadFrom(b []byte) (n int, addr Addr, err error)
    WriteTo(b []byte, addr Addr) (n int, err error)
    Close() error
    LocalAddr() Addr
    SetDeadline(t time.Time) error
    SetReadDeadline(t time.Time) error
    SetWriteDeadline(t time.Time) error
}
```

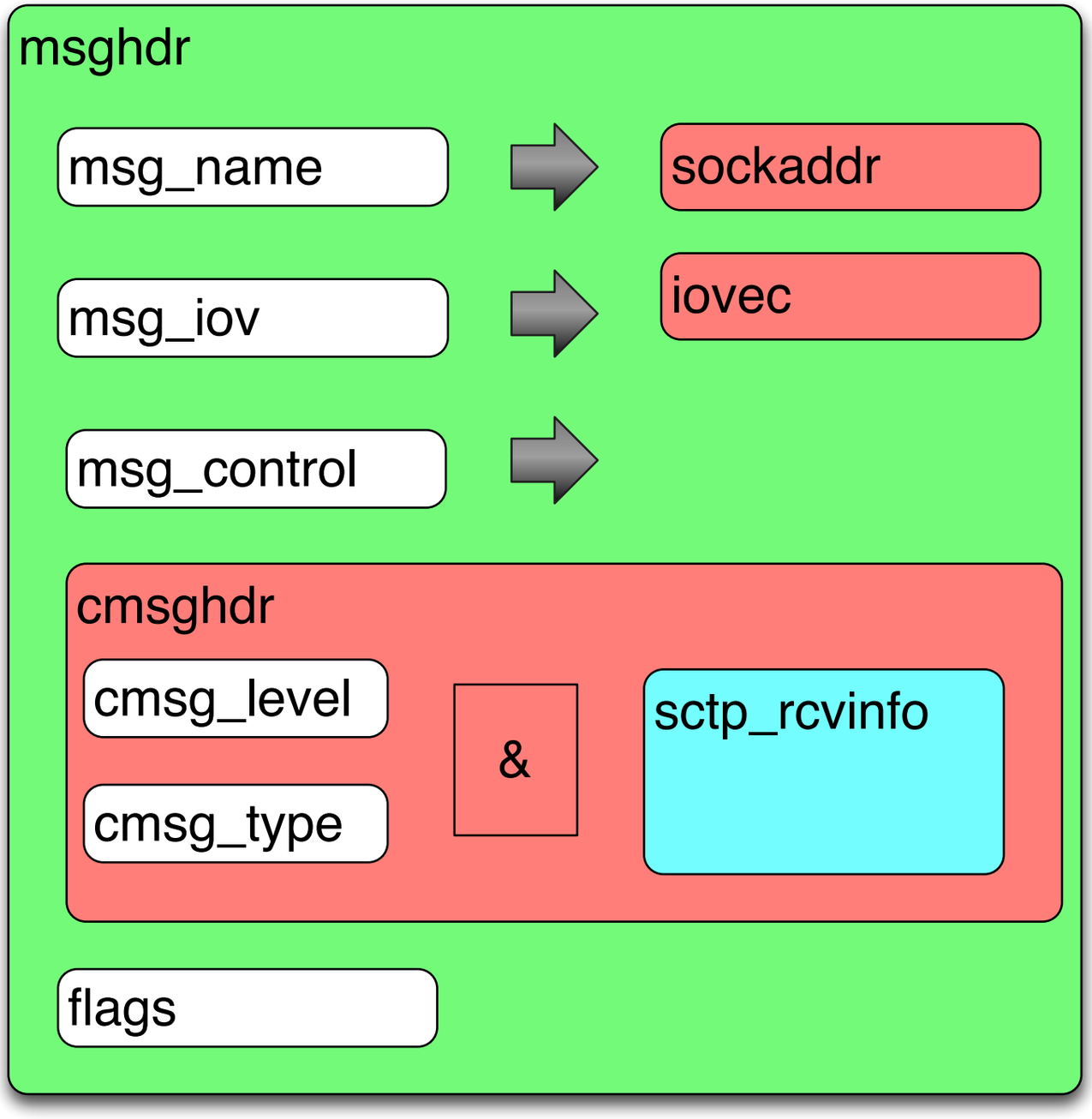
ReadFrom


net.Conn.ReadFrom()
net.SCTPConn.ReadFromSCTP()
syscall.SCTPReceiveMessage()
syscall.recvmsg()
Syscall(SYS_RECVMSGuintptr(s), uintptr(unsafe.Pointer(msg)), uintptr(flags))

SCTPReceiveMessage

```
func SCTPReceiveMessage(fd int, p []byte)
     (n int, from Sockaddr, rinfo *SCTPRcvInfo, flags int, err error) {

  // Message header
  var msg Msghdr
  var rsa RawSockaddrAny
  msg.Name = (*byte)(unsafe.Pointer(&rsa))
  msg.Namelen = uint32(SizeofSockaddrAny)
  // Create struct for message
  var iov Iovec
  if len(p) > 0 {
    iov.Base = (*byte)(unsafe.Pointer(&p[0]))
    iov.SetLen(len(p))
  }
  msg.Iov = &iov
  msg.Iovlen = 1

  // Message control header
  var cmsg *Cmsghdr

  controlBuffer := make([]byte, SizeofCmsghdr + SizeofSCTPRcvInfo)
  data := (controlBuffer[cmsgAlignOf(SizeofCmsghdr):])
  rinfo = (*SCTPRcvInfo)(unsafe.Pointer(&data[0]))
  cmsg = (*Cmsghdr)(unsafe.Pointer(&controlBuffer[0]))
  cmsg.Level = IPPROTO_SCTP
  cmsg.Type = SCTP_SNDRCV
  msg.Control = (*byte)(unsafe.Pointer(&controlBuffer[0]))
  msg.SetControllen(len(controlBuffer))

  flags = 0;
  n, err = recvmsg(fd, &msg, flags)

  if err != nil {
    return 0, nil, nil, 0, err
  }
  from, err = anyToSockaddr(&rsa)
  return
}
```

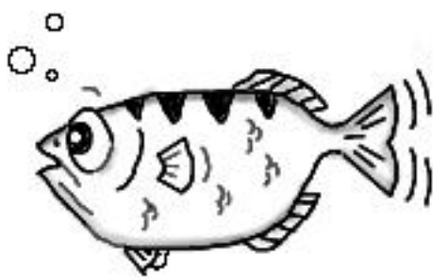
Structure for receiving message



Message header

```
1    var msg Msghdr
2    var rsa RawSockaddrAny
3    msg.Name = (*byte)(unsafe.Pointer(&rsa))
4    msg.Namelen = uint32(SizeofSockaddrAny)
5
6    var cmsg *Cmsghdr
7    controlBuffer := make([]byte, SizeofCmsghdr + SizeofSCTPRcvInfo)
8    data := (controlBuffer[cmsgAlignOf(SizeofCmsghdr):])
9    rinfo = (*SCTPRcvInfo)(unsafe.Pointer(&data[0]))
10   cmsg = (*Cmsghdr)(unsafe.Pointer(&controlBuffer[0]))
11   cmsg.Level = IPPROTO_SCTP
12   cmsg.Type = SCTP_SNDRCV
13   msg.Control = (*byte)(unsafe.Pointer(&controlBuffer[0]))
14   msg.SetControllen(len(controlBuffer))
```

Demo debug session



More...

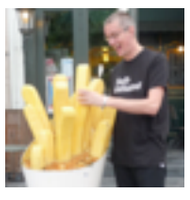
http://cyberroadie.wordpress.com

@cyberroadie

https://github.com/cyberroadie

https://bitbucket.org/cyberroadie/go-sctp

cyberroadie@gmail.com



http://www.cyberroadie.org