

Go SCTP!
by
Olivier Van Acker

Who am I?

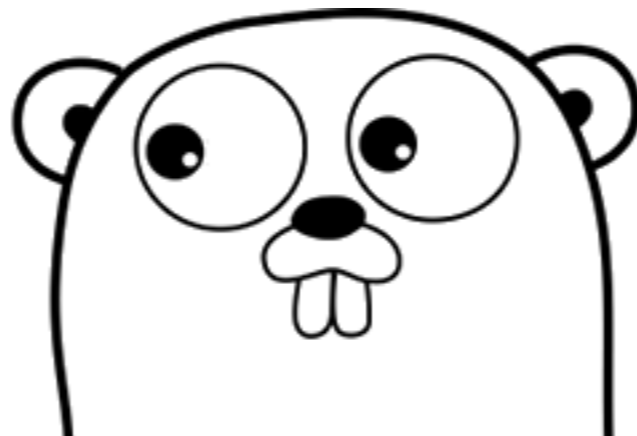


Why FreeBSD?



Overview

SCTP & Go

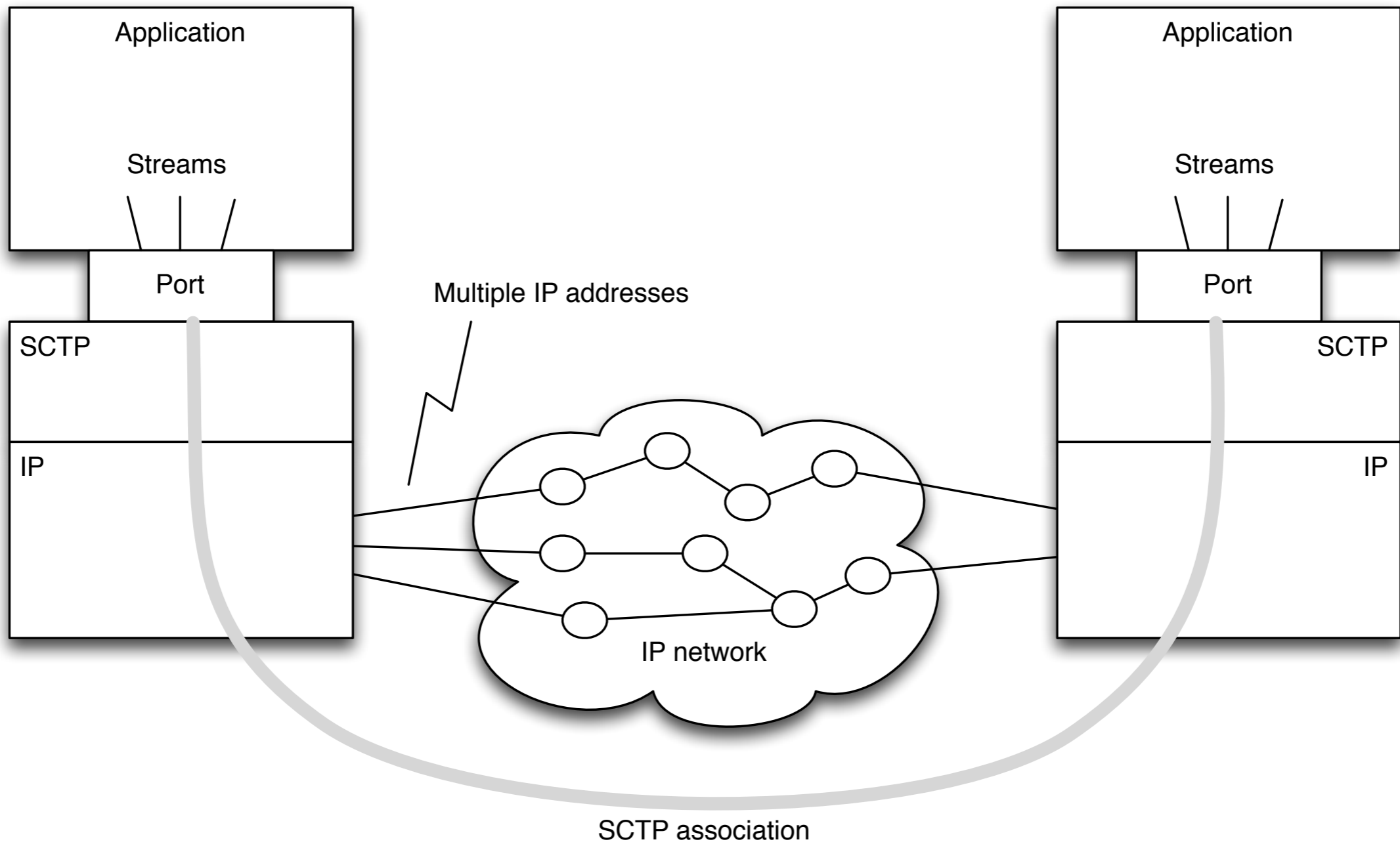


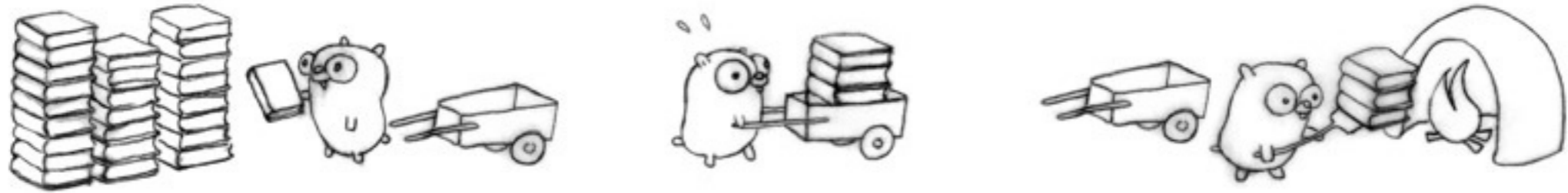
Birds eye view SCTP



Associations

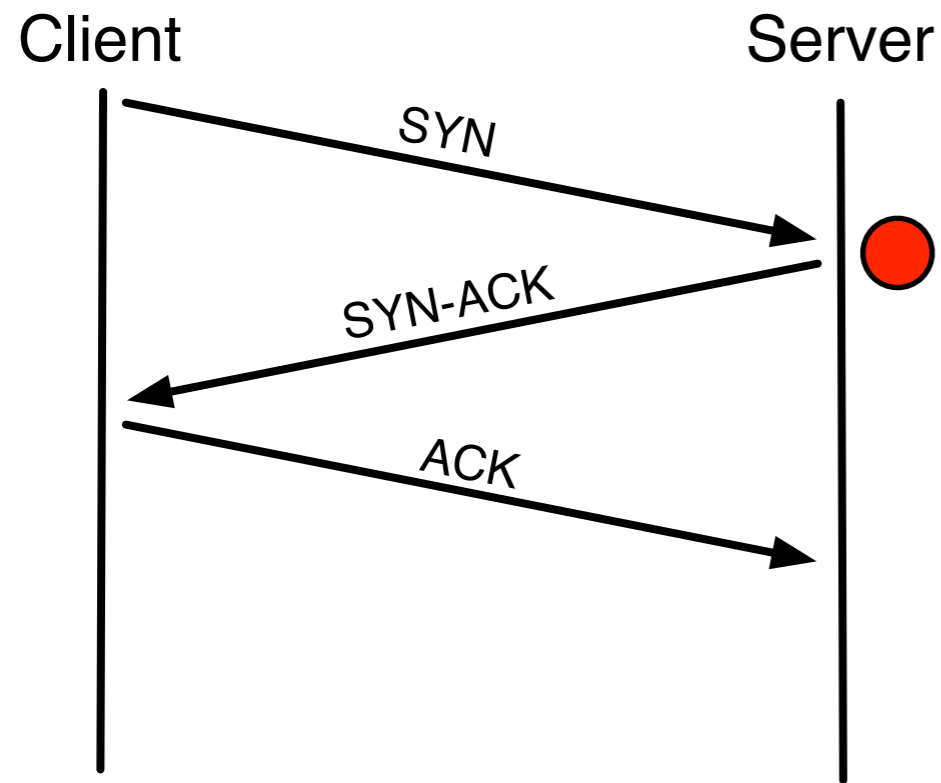




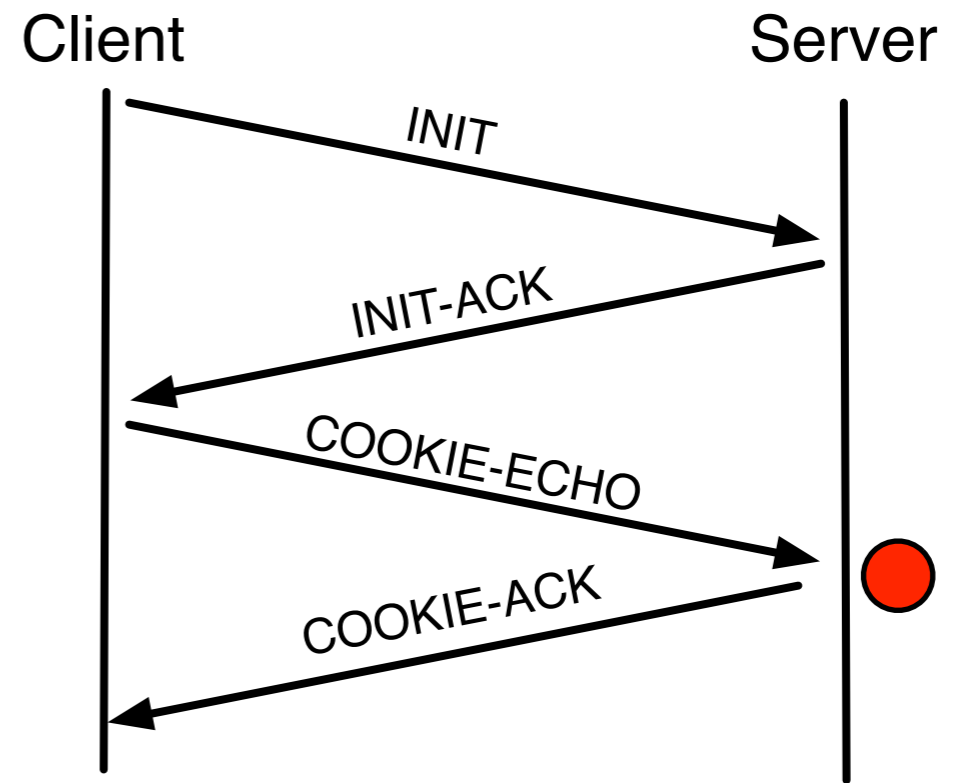


Messages

Connection



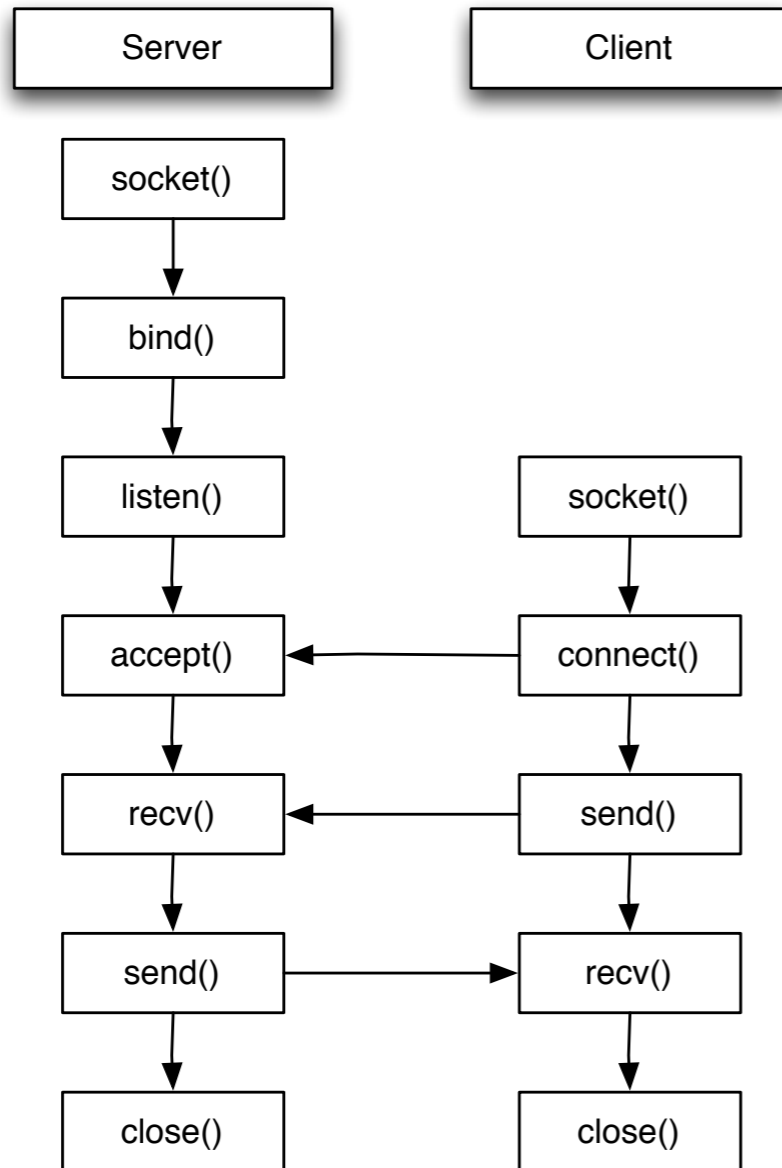
TCP 3 way handshake



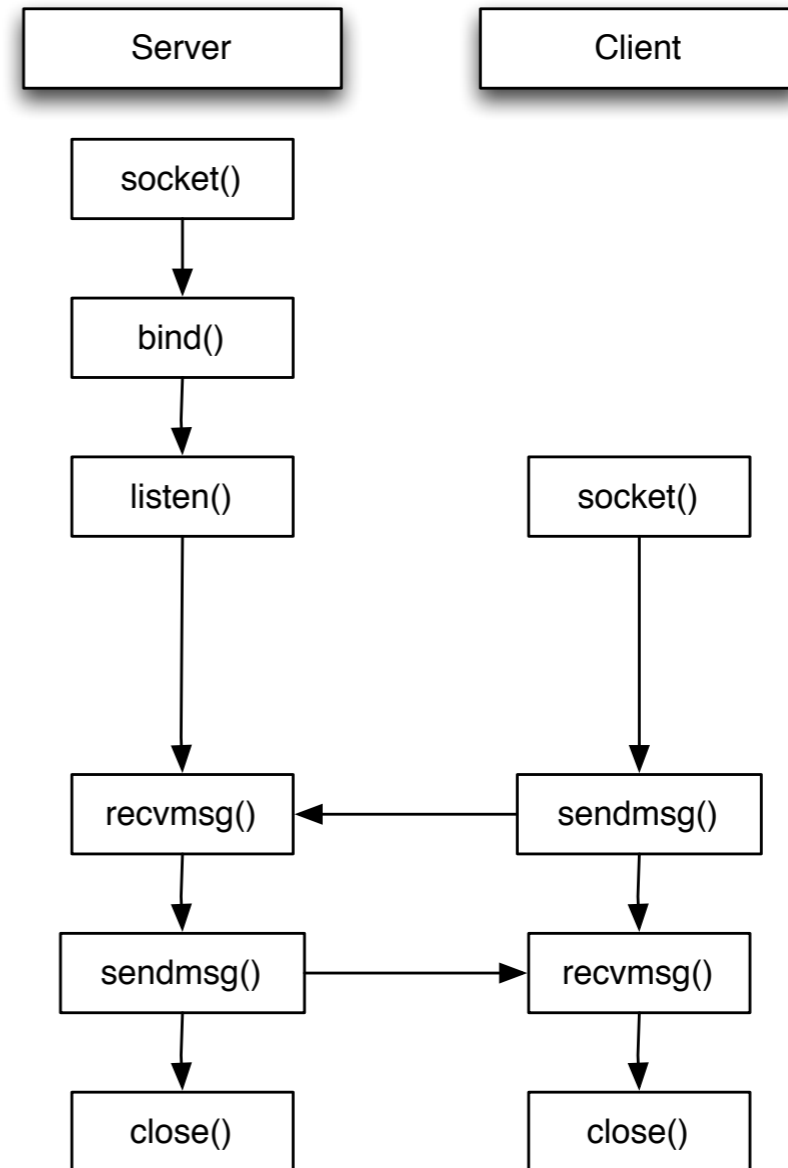
SCPT 4 way handshake

Socket API

TCP



SCTP



Birds eye view Go



Data

```
var number int  
var first_name, last_name string  
var p *[]int = new([]int)  
var v []int = make([]int, 100)
```

Functions

```
func hello(name string, count int) (greeting string, err error) {  
    if count = 0 {  
        return nil, errors.New("Cannot say hello zero times")  
    }  
    greeting = "Hello" + name, nil  
    return  
}
```

Functions

```
greeting, err := hello("paard", 0)
if err != nil {
    println("error!")
} else {
    println(greeting)
}
```

Functions

```
greeting, _ := hello("paard", 1)
```

Structures

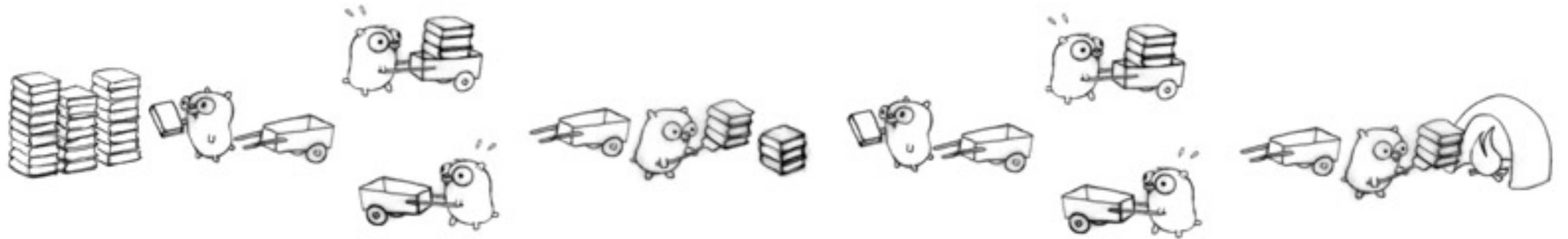
```
type Person struct {  
    name string  
    age int  
}
```


Structures

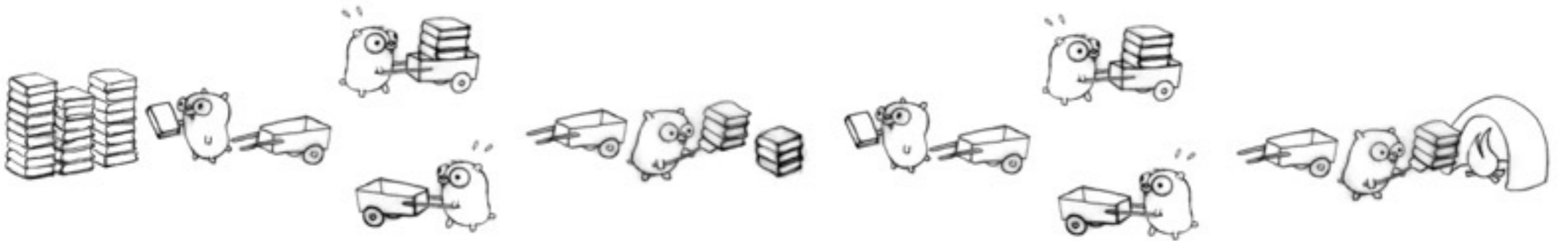
```
func (p Person) SayHello(name string) {  
    return "Hello " + name ", my name is " + p.name  
}
```

Interfaces

```
type animal interface {  
    Talk()  
}  
  
type Cat  
  
func (c Cat) Talk() {  
    fmt.Println("Meow")  
}  
  
func main() {  
    var c Cat  
    c.Talk()  
    a := animal(c)    // Cast from cat to animal  
    a.Talk()  
}
```

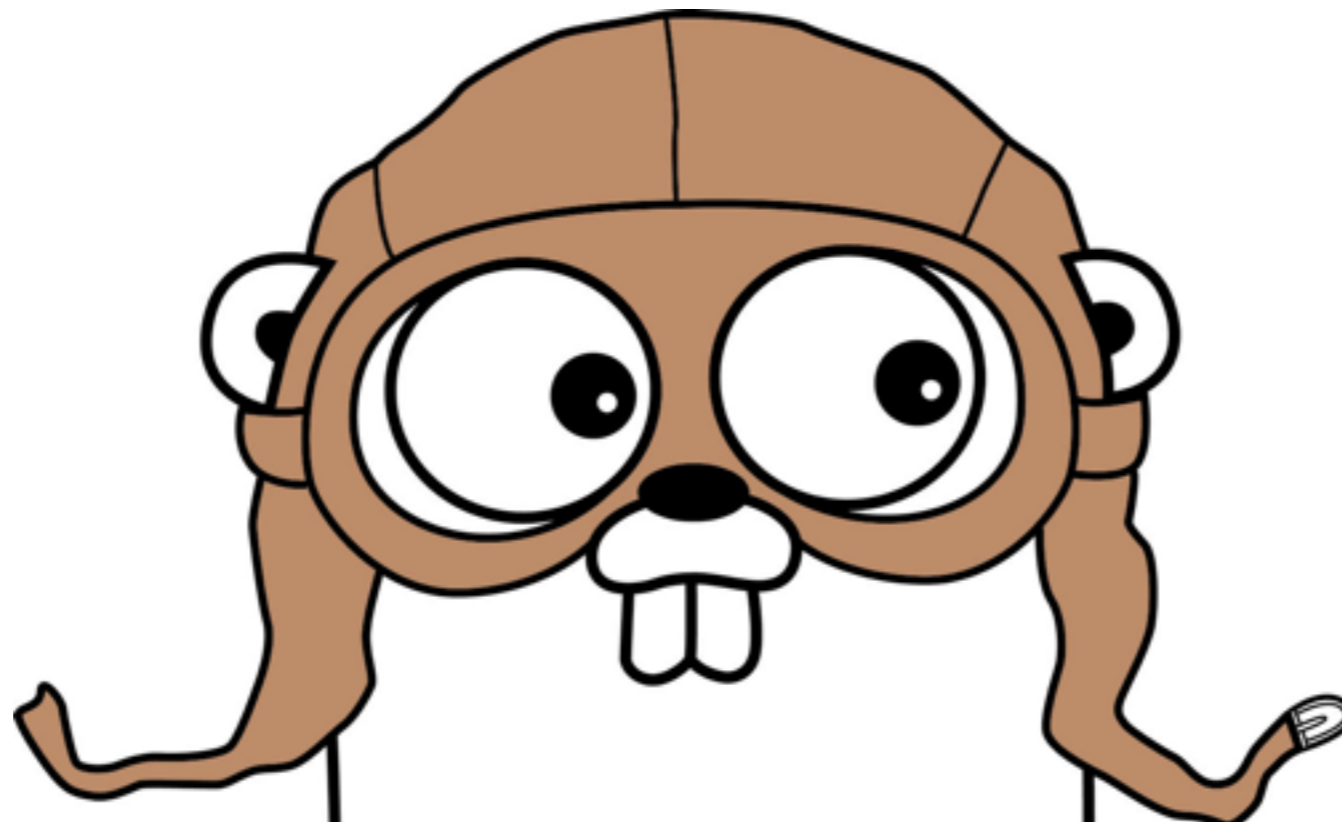


Much more



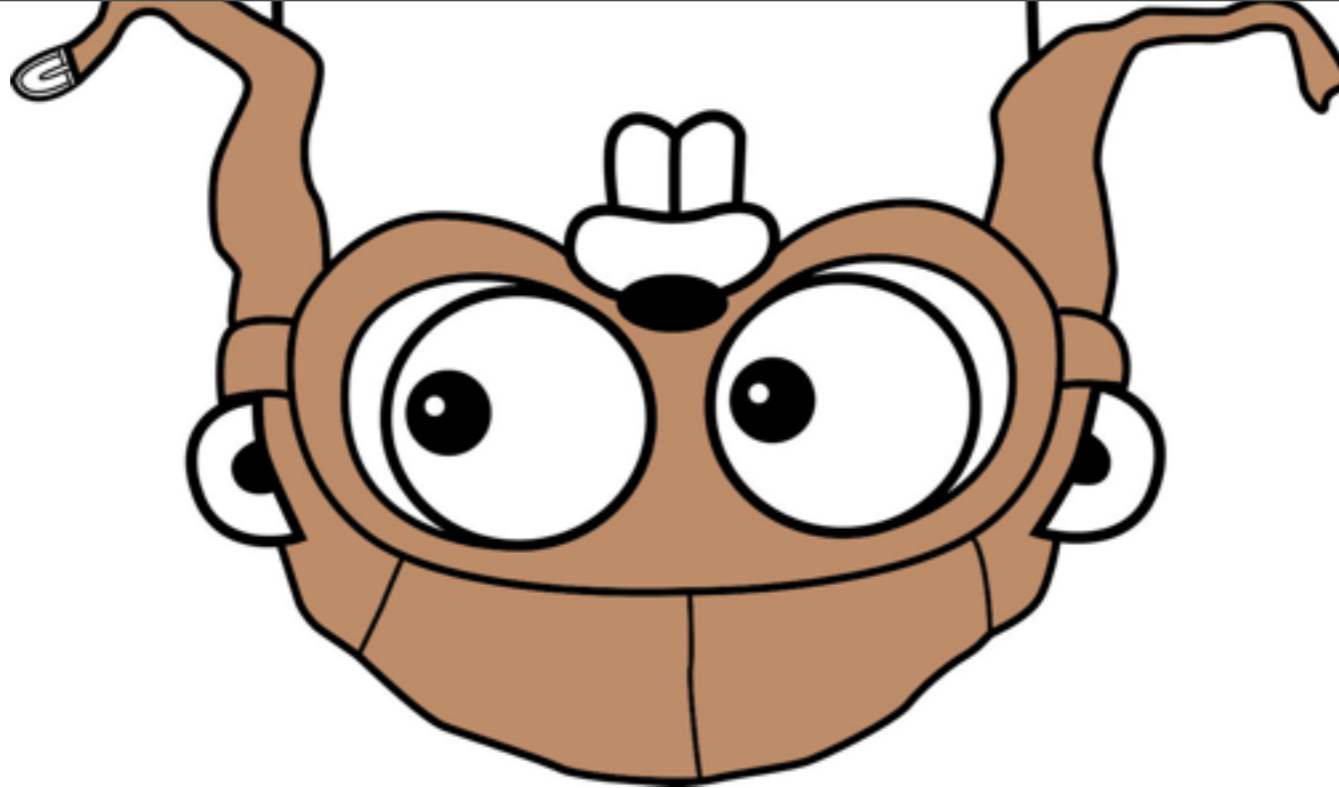
Go Network Library

Client



```
package main
import "net"

func main() {
    conn, err := net.Dial("tcp", "localhost:1234")
    if err != nil {
        return
    }
    defer conn.Close()
    conn.Write([]byte("Hello world!"))
}
```



Server

```
package main
import "net"

func main() {
    listen, err := net.Listen("tcp", "localhost:1234")
    if err != nil {
        return
    }
    buffer := make([]byte, 1024)
    for {
        conn, err := listen.Accept()
        if err != nil {
            continue
        }
        conn.Read(buffer)
        println(string(buffer))
    }
}
```


Extending the Go networking library with SCTP

Typical Server

```
package main
import "net"

func main() {
    conn, _ := net.ListenPacket("sctp", "localhost:4242")
    defer conn.Close()
    message := make([]byte, 1024)
    conn.ReadFrom(message)
    print(string(message))
}
```

SCTP Specific

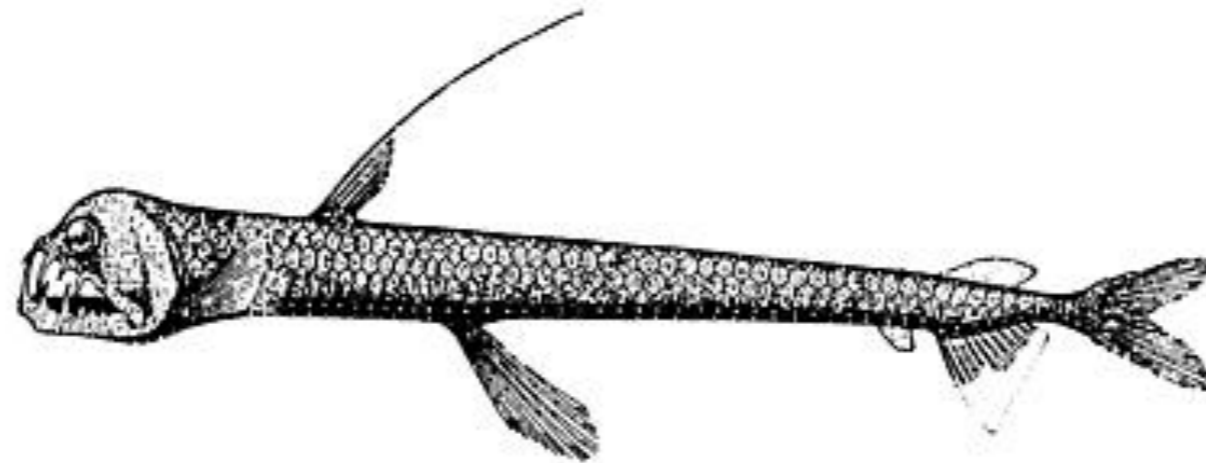
```
(*SCTPConn).ReadFromSCTP(message *string)  
    (sid int, ssn int, ppid int,  
     aid int, addr SCTPAddr, err error)
```

Typical Server

```
package main
import (
    "net"
    "strconv"
)

func main() {
    addr, _ := net.ResolveSCTPAddr("sctp", "localhost:4242")
    conn, _ := net.ListenSCTP("sctp", addr)
    defer conn.Close()
    for {
        message := make([]byte, 1024)
        _, _, stream, _ := conn.ReadFromSCTP(message)
        println("stream " +
            strconv.Itoa(int(stream)) +
            ": " + string(message))
    }
}
```

Bottom up



Handcrafted assembly

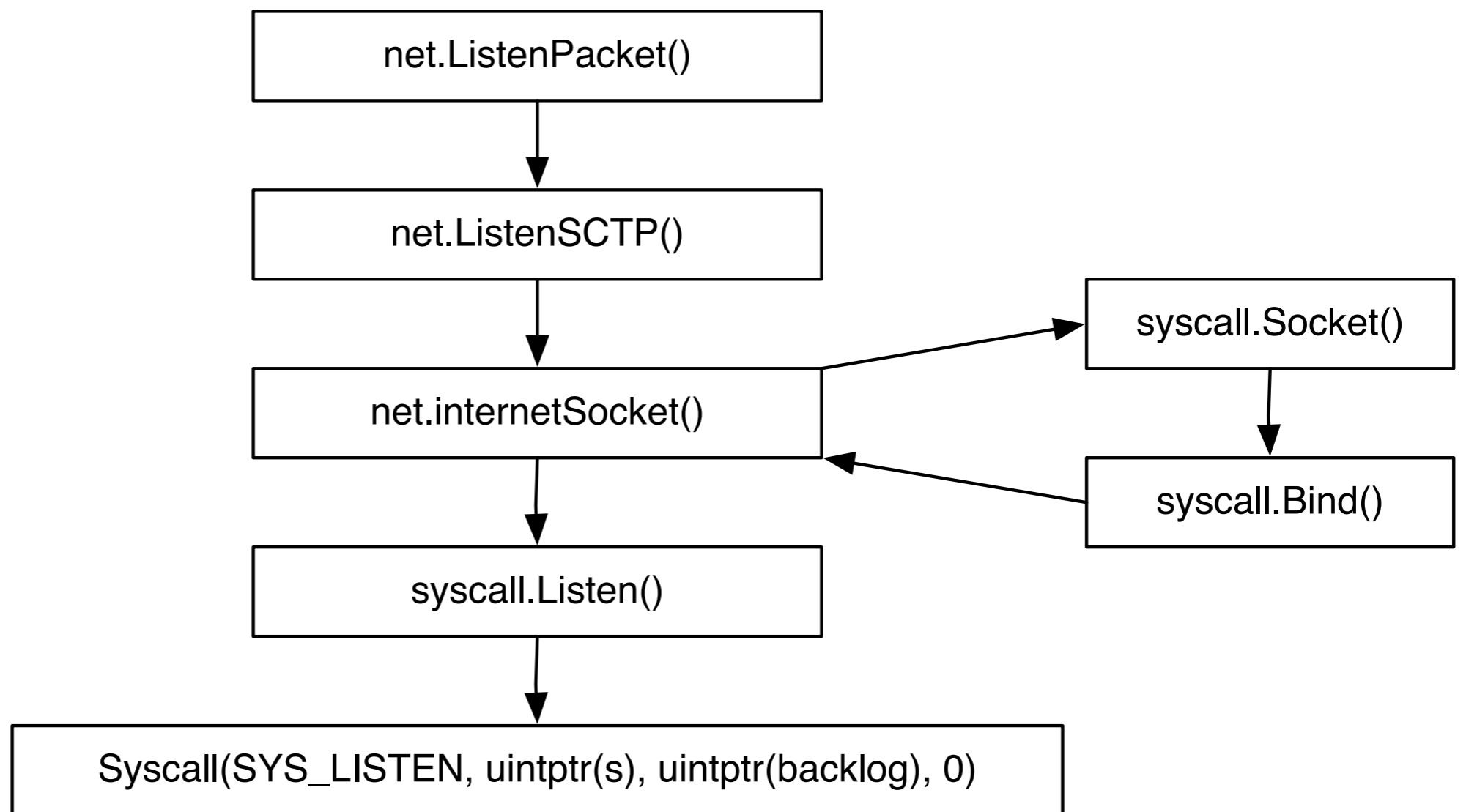
```
TEXT    ·Syscall(SB),7,$0
        CALL    runtime·entersyscall(SB)
        MOVQ    16(SP), DI
        MOVQ    24(SP), SI
        MOVQ    32(SP), DX
        MOVQ    $0, R10
        MOVQ    $0, R8
        MOVQ    $0, R9
        MOVQ    8(SP), AX // syscall entry
        SYSCALL
        JCC    ok
        MOVQ    $-1, 40(SP) // r1
        MOVQ    $0, 48(SP) // r2
        MOVQ    AX, 56(SP) // errno
        CALL    runtime·exitsyscall(SB)
        RET
ok:
        MOVQ    AX, 40(SP) // r1
        MOVQ    DX, 48(SP) // r2
        MOVQ    $0, 56(SP) // errno
        CALL    runtime·exitsyscall(SB)
        RET
```

Auto generated functions

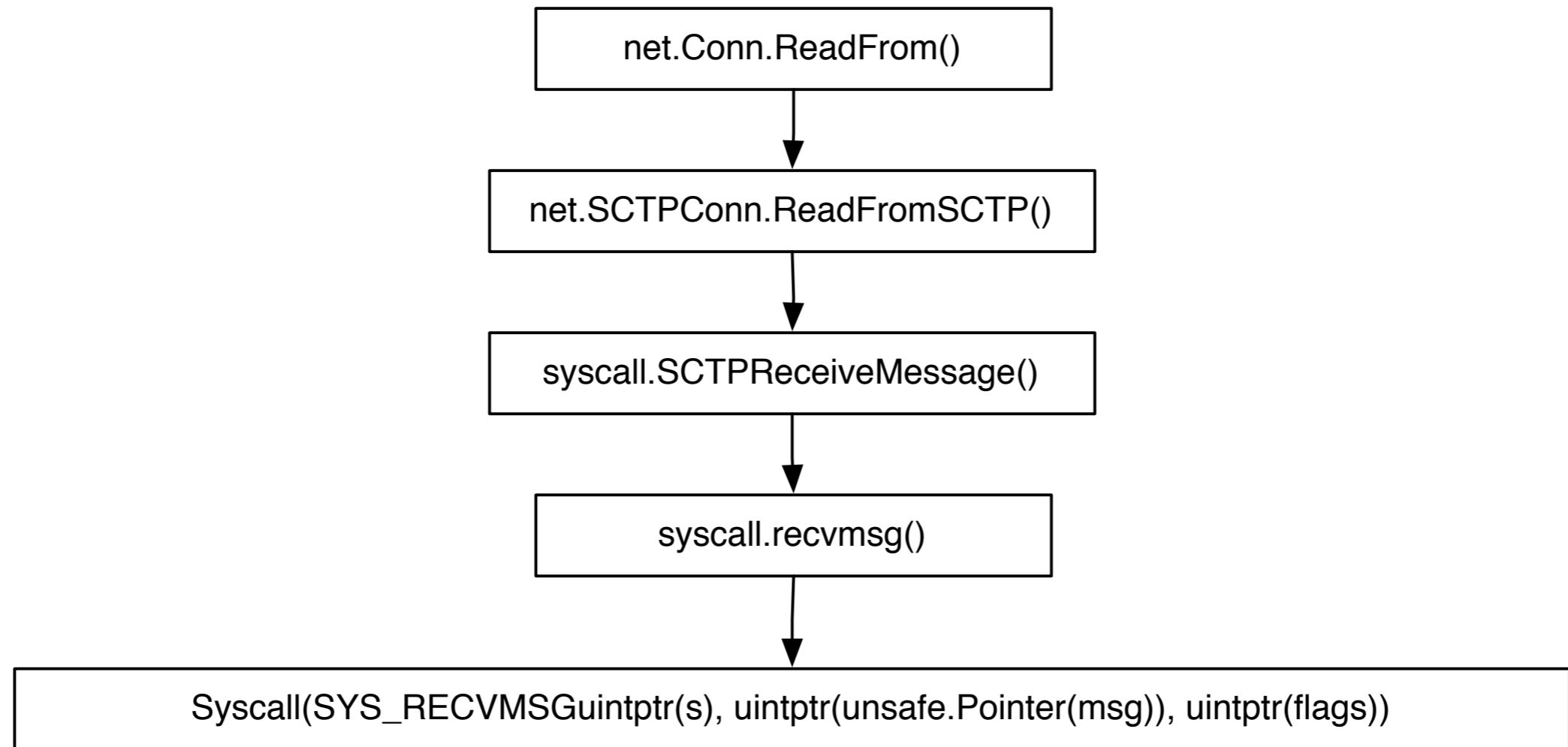
```
func sendmsg(s int, msg *MsgHdr, flags int) (err error) {  
    _, _, e1 := Syscall(SYS_SENDMSG, uintptr(s),  
                        uintptr(unsafe.Pointer(msg)), uintptr(flags))  
    if e1 != 0 {  
        err = e1  
    }  
    return  
}
```

**Add structures for
ancillary data**

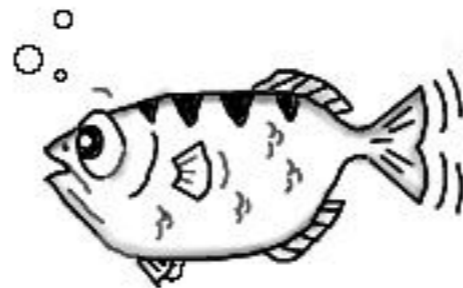
Setting up

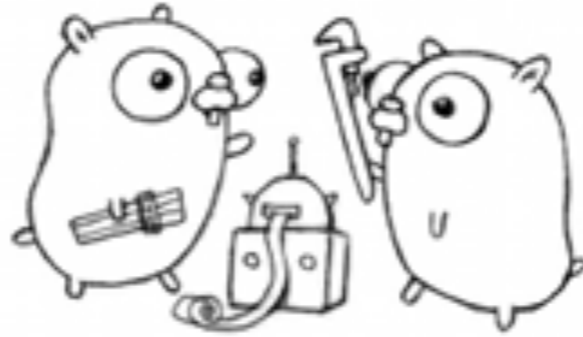


ReadFrom



Demo debug session





Future work

More...



<http://cyberroadie.wordpress.com>



@cyberroadie



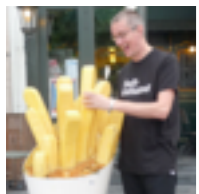
<https://github.com/cyberroadie>



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



olivier@robotmotel.com



<http://www.cyberroadie.org>

Permission granted to us pictures by Rob Pike (Gopher) and Elsevier (Association overview)