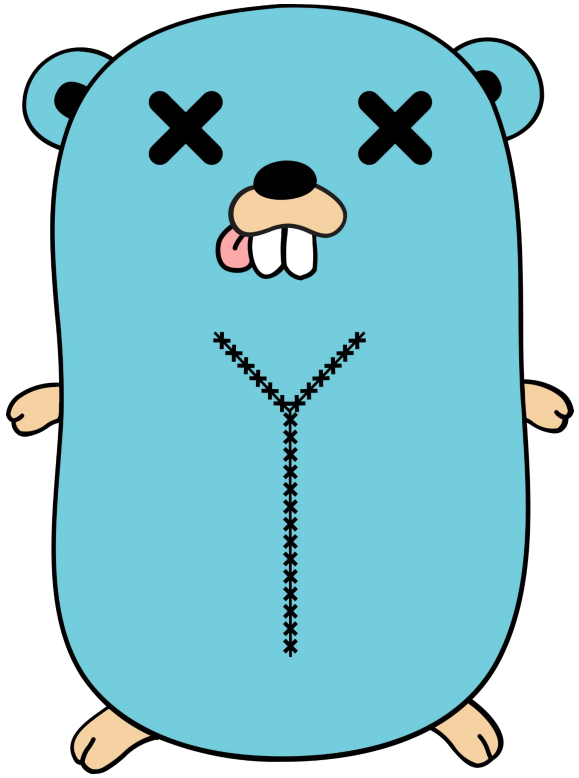
JESÚS ESPINO

Software Engineer @ Mattermost

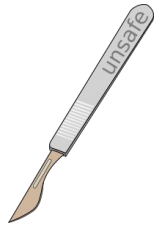
Dissecting a Channel in Go

INTRODUCTION



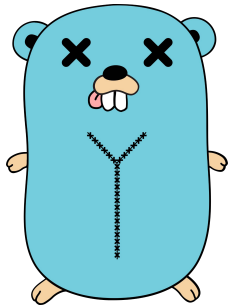
- Channels are one of the most commonly used built-in structures in go.
- We understand how to use them, but not necessarily how they work.
- We are going to analyze how they work under the hood.
- We are going to do it through an experimental approach.
- After this talk you will understand better how channels are shaped in memory and what are the implications of that.

CLASS MATERIALS



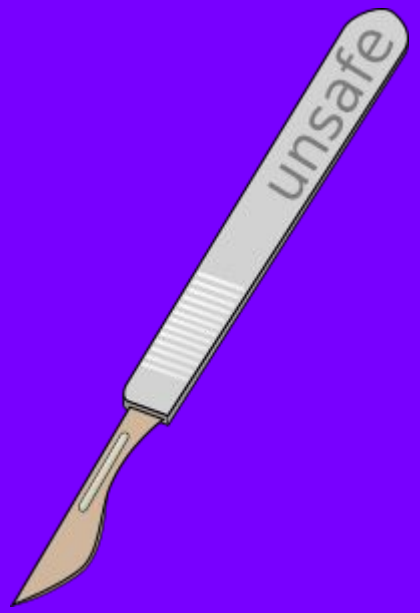
- The scalpel

- The microscope



- The subject

The scalpel

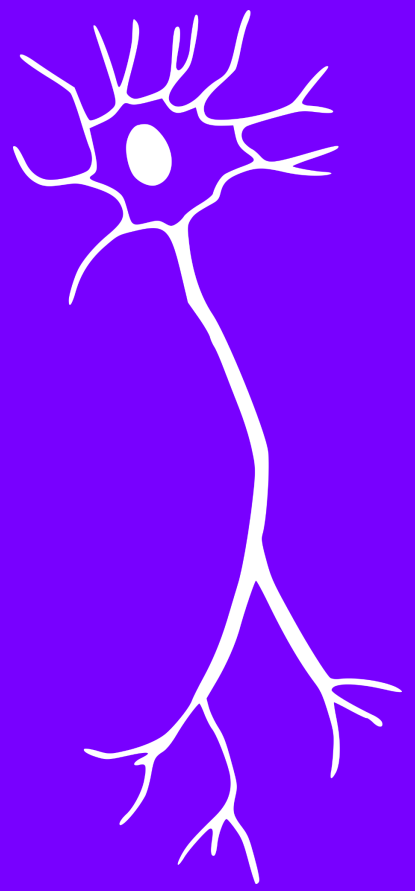



```
func Scalpel(channel *(chan int32)) *channelStruct {
    cs := unsafe.Pointer(*(*uintptr)(unsafe.Pointer(channel)))
    return (*channelStruct)(cs)
}
```

The microscope

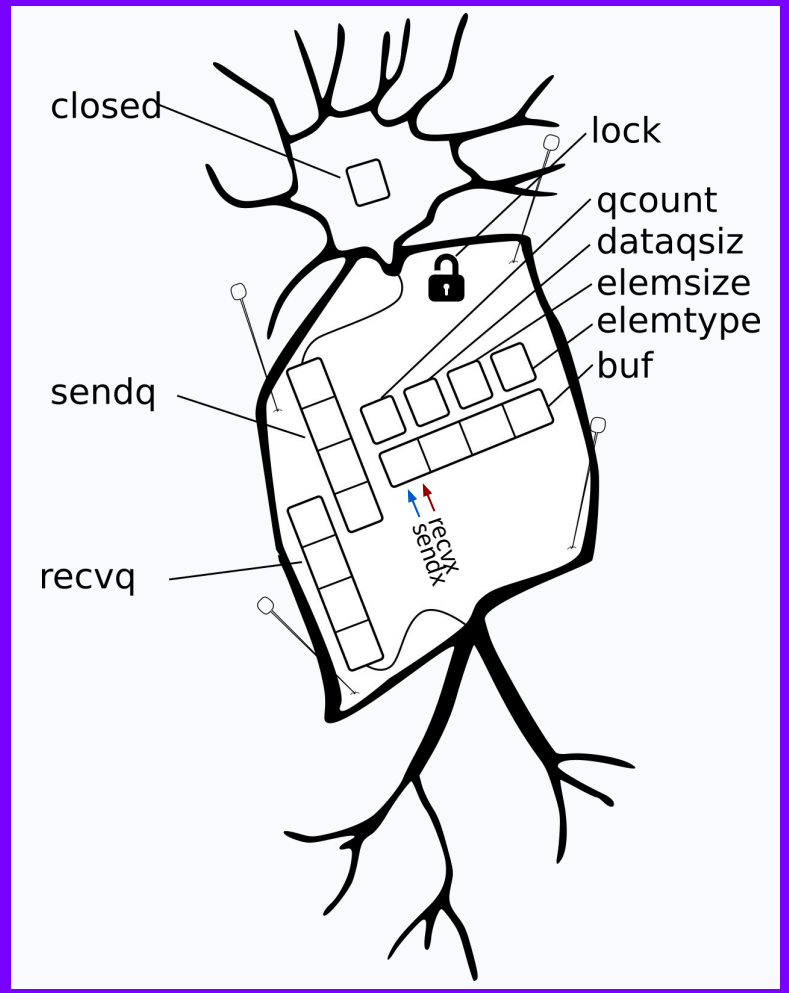
```
func Microscope(cs *channelStruct) {
    fmt.Printf("Total data in queue: %d\n", cs.qcount)
    fmt.Printf("Size of the queue: %d\n", cs.dataqsiz)
    fmt.Printf("Buffer address: %p\n", cs.buf)
    fmt.Printf("Element size: %d\n", cs.elemsize)
    fmt.Printf("Queued elements: %v\n", *cs.buf)
    fmt.Printf("Closed: %d\n", cs.closed)
    fmt.Printf("Element Type Address: %d\n", cs.elemtype)
    fmt.Printf("Send Index: %d\n", cs.sendx)
    fmt.Printf("Receive Index: %d\n", cs.recvx)
    fmt.Printf("Receive Wait list first address: 0x%x\n", cs.recvq.first)
    fmt.Printf("Receive Wait list last address: 0x%x\n", cs.recvq.last)
    fmt.Printf("Send Wait list first address: 0x%x\n", cs.sendq.first)
    fmt.Printf("Send Wait list last address: 0x%x\n", cs.sendq.last)
    fmt.Println("-------------------------------")
}
```

- The subject



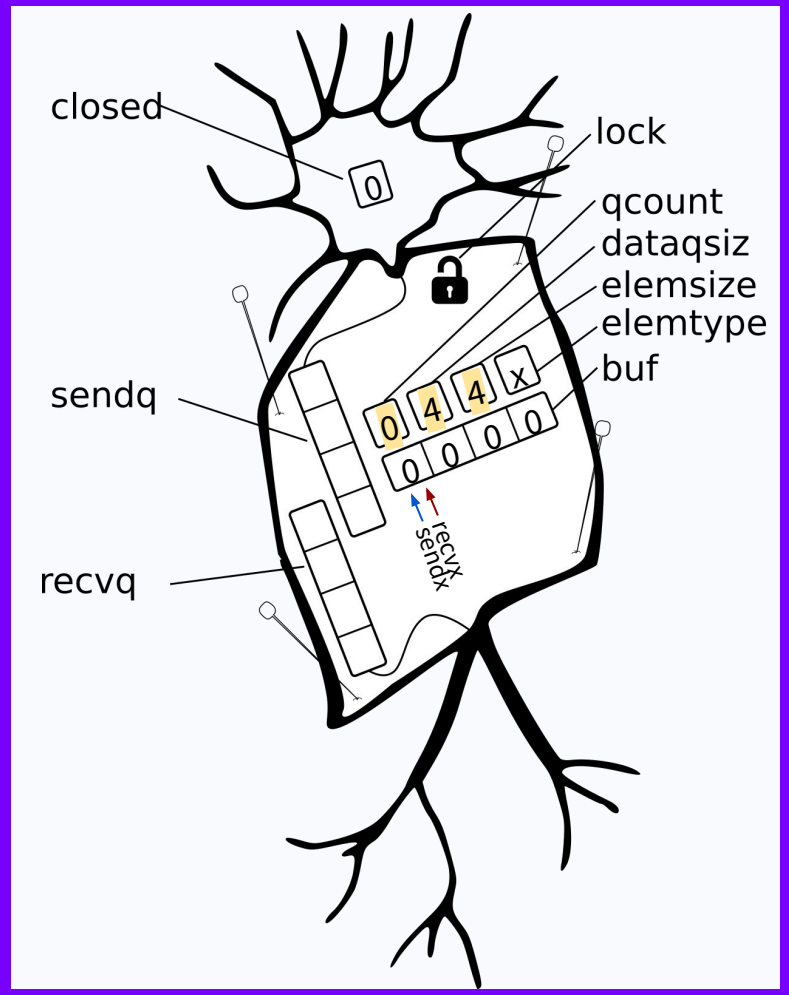
```
c := make(chan int32, 4)
```

Inside the subject



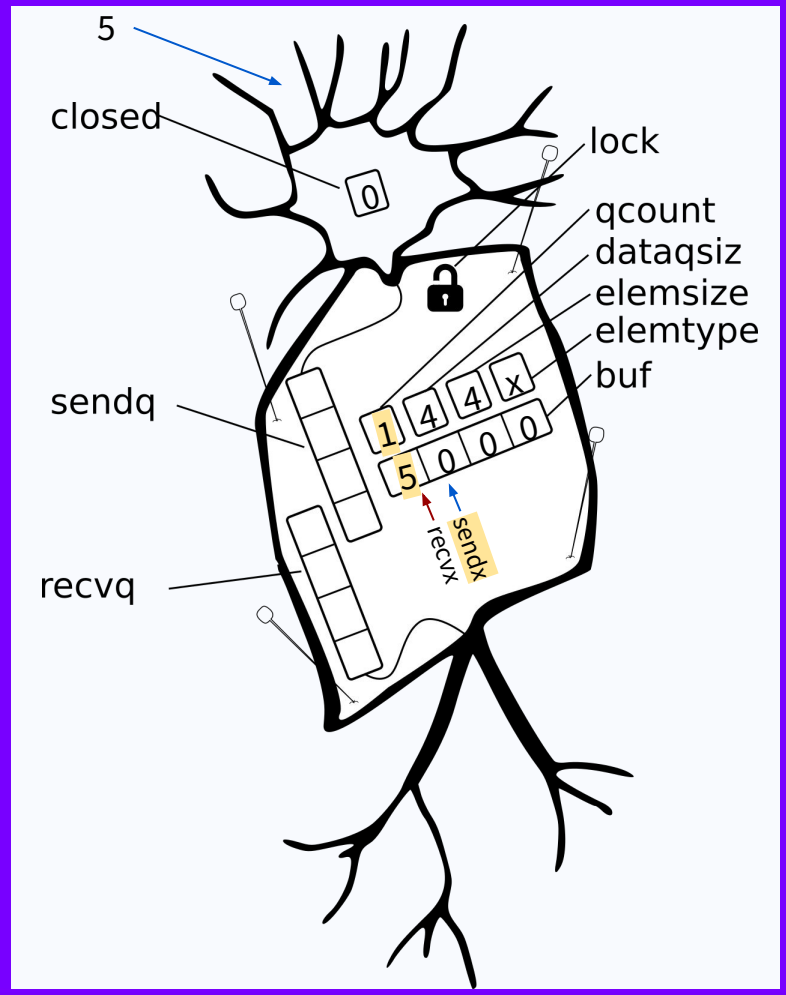

```
type waitq struct {
    first uintptr
    last  uintptr
}

type channelStruct struct {
    qcount   uint
    dataqsiz uint
    buf      *[4]int32
    elemsize uint16
    closed   uint32
    elemtype uintptr
    sendx    uint
    recvx    uint
    recvq    waitq
    sendq    waitq
    lock uintptr
}
```

Channel creation



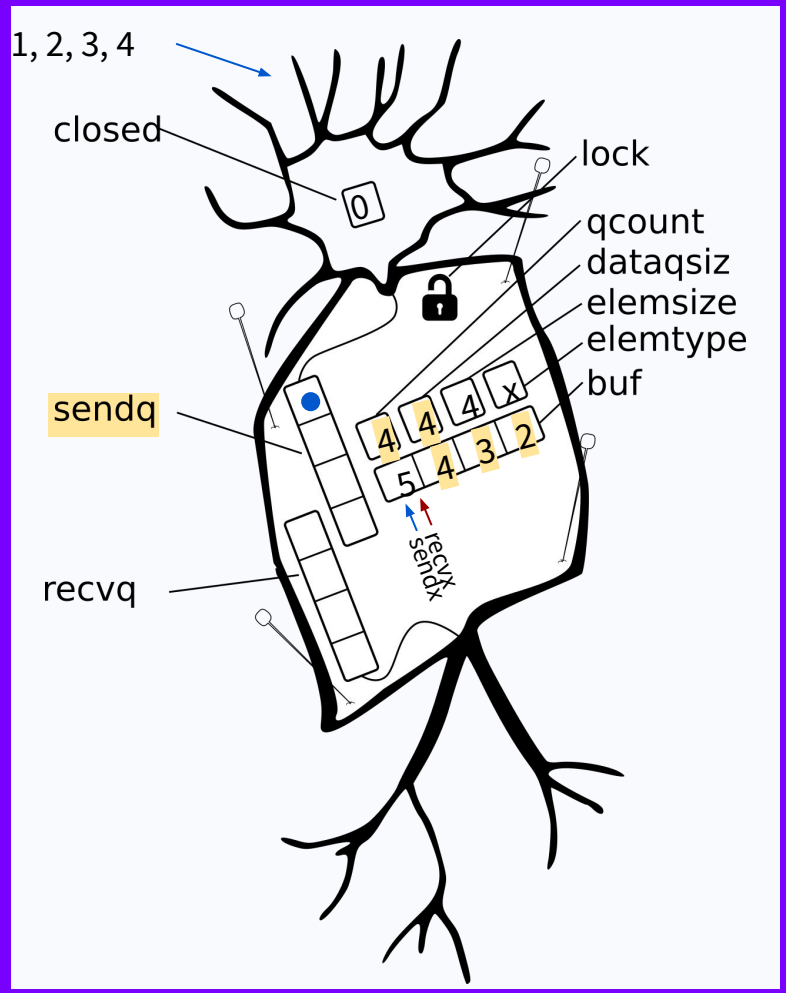

```
c := make(chan int32, 4)
cs = Scalpel(&c)
Microscope(cs)
--------------
Total data in queue: 0
Size of the queue: 4
Buffer address: 0xc000130060
Element size: 4
Queued elements: [0 0 0 0]
Closed: 0
Element Type Address: 4870720
Send Index: 0
Receive Index: 0
Receive Wait list first address: 0x0
Receive Wait list last address: 0x0
Send Wait list first address: 0x0
Send Wait list last address: 0x0
```

Insert into the channel



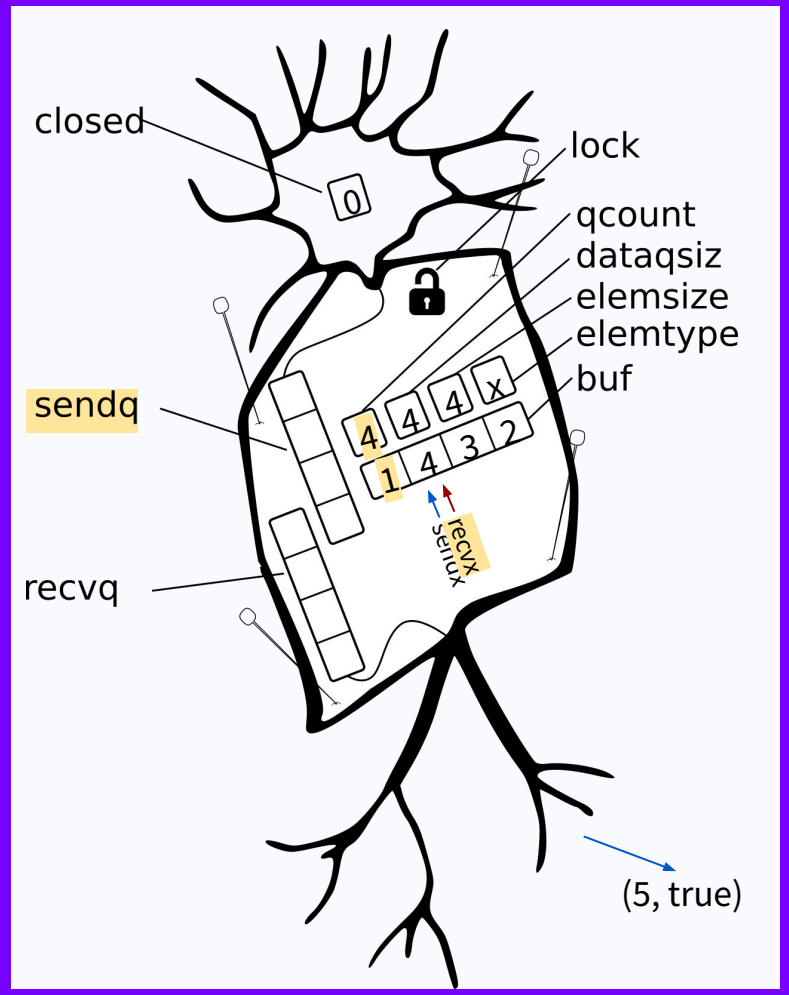

```
c <- 5
Microscope(cs)
--------------
Total data in queue: 1
Size of the queue: 4
Buffer address: 0xc000130060
Element size: 4
Queued elements: [5 0 0 0]
Closed: 0
Element Type Address: 4870720
Send Index: 1
Receive Index: 0
Receive Wait list first address: 0x0
Receive Wait list last address: 0x0
Send Wait list first address: 0x0
Send Wait list last address: 0x0
```

Fill the channel buffer



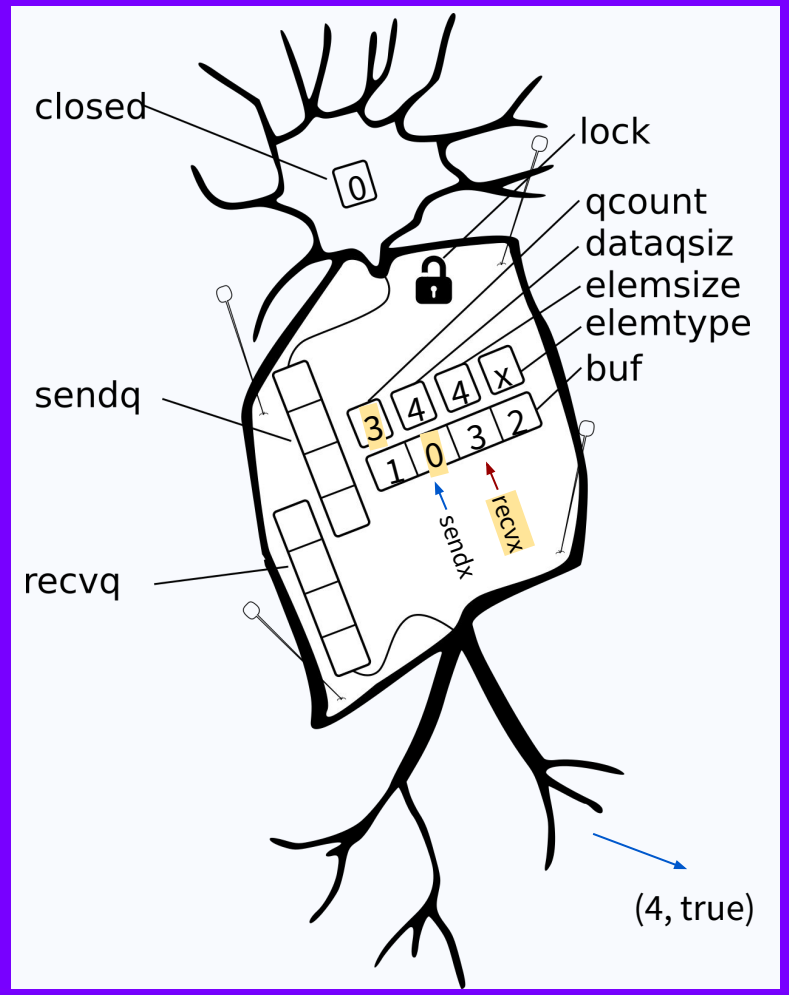

```
c <- 4
c <- 3
c <- 2
c <- 1
Microscope(cs)
--------------
Total data in queue: 4
Size of the queue: 4
Buffer address: 0xc000130060
Element size: 4
Queued elements: [5 4 3 2]
Closed: 0
Element Type Address: 4870720
Send Index: 0
Receive Index: 0
Receive Wait list first address: 0x0
Receive Wait list last address: 0x0
Send Wait list first address: 0xc000028060
Send Wait list last address: 0xc000028060
```

Read from the channel



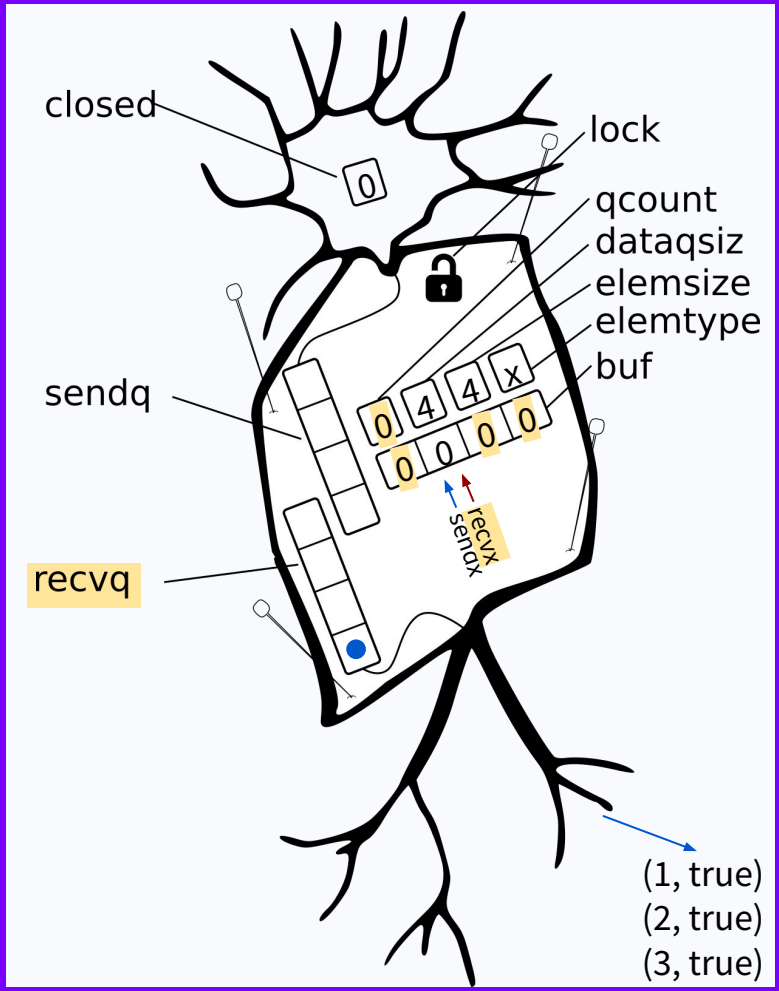

```
<-c
Microscope(cs)
--------------
Total data in queue: 4
Size of the queue: 4
Buffer address: 0xc000130060
Element size: 4
Queued elements: [1 4 3 2]
Closed: 0
Element Type Address: 4870720
Send Index: 1
Receive Index: 1
Receive Wait list first address: 0x0
Receive Wait list last address: 0x0
Send Wait list first address: 0x0
Send Wait list last address: 0x0
```

More reading



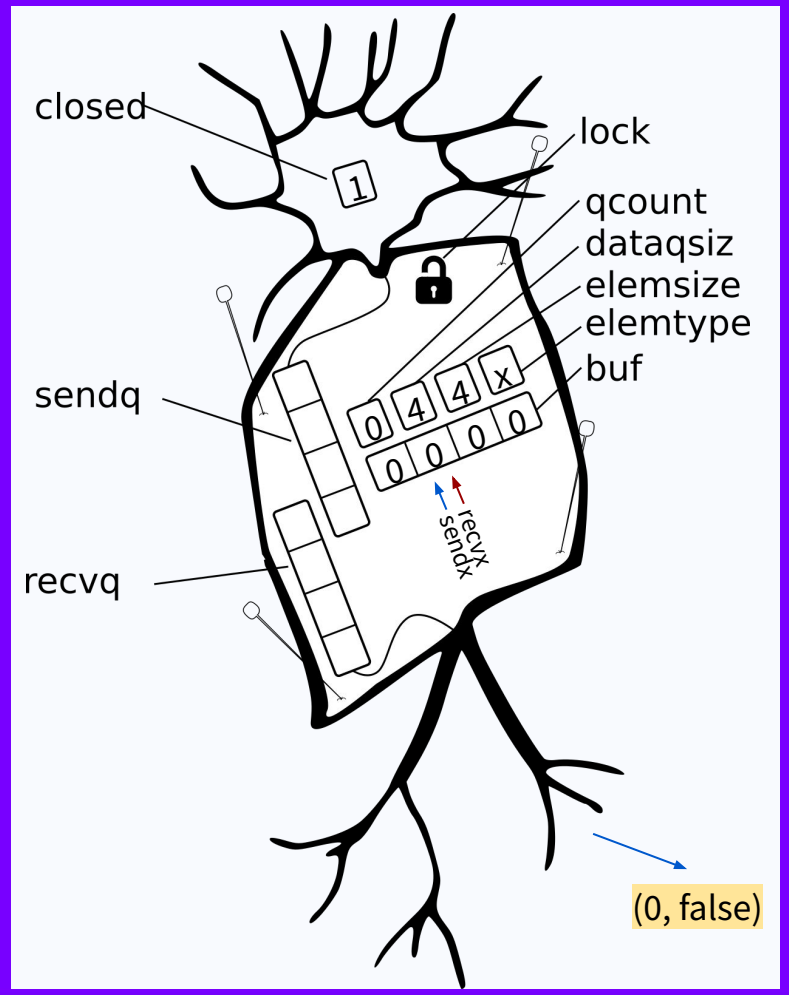

```
<-c
Microscope(cs)
--------------
Total data in queue: 3
Size of the queue: 4
Buffer address: 0xc000130060
Element size: 4
Queued elements: [1 0 3 2]
Closed: 0
Element Type Address: 4870720
Send Index: 1
Receive Index: 2
Receive Wait list first address: 0x0
Receive Wait list last address: 0x0
Send Wait list first address: 0x0
Send Wait list last address: 0x0
```

Wait for data




```
<-c
<-c
<-c
<-c
Microscope(cs)
--------------
Total data in queue: 0
Size of the queue: 4
Buffer address: 0xc000130060
Element size: 4
Queued elements: [0 0 0 0]
Closed: 0
Element Type Address: 4870720
Send Index: 1
Receive Index: 1
Receive Wait list first address: 0xc000194000
Receive Wait list last address: 0xc000194000
Send Wait list first address: 0x0
Send Wait list last address: 0x0
```

Close channel




```
close(c)
Microscope(cs)
--------------
Total data in queue: 0
Size of the queue: 4
Buffer address: 0xc000130060
Element size: 4
Queued elements: [0 0 0 0]
Closed: 1
Element Type Address: 4870720
Send Index: 1
Receive Index: 1
Receive Wait list first address: 0x0
Receive Wait list last address: 0x0
Send Wait list first address: 0x0
Send Wait list last address: 0x0
```

References

- The channel go code: `src/runtime/chan.go`
- My code: http://github.com/jespino/dissecting-go

Thank You

jespinog

jespino

jesus-espino