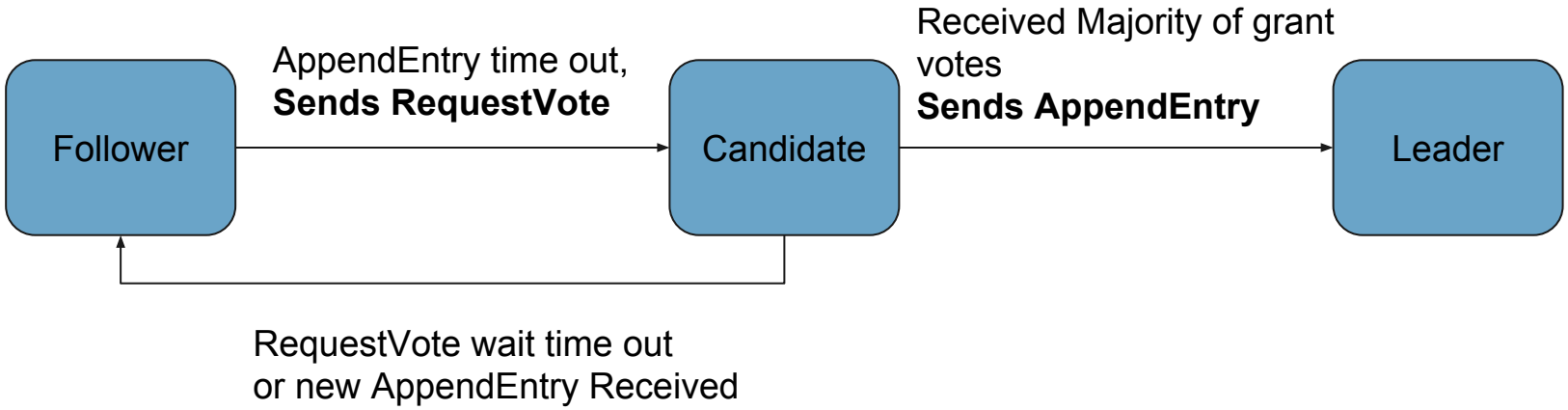
Lab 1 (Leader Election)

Wonsup Yoon

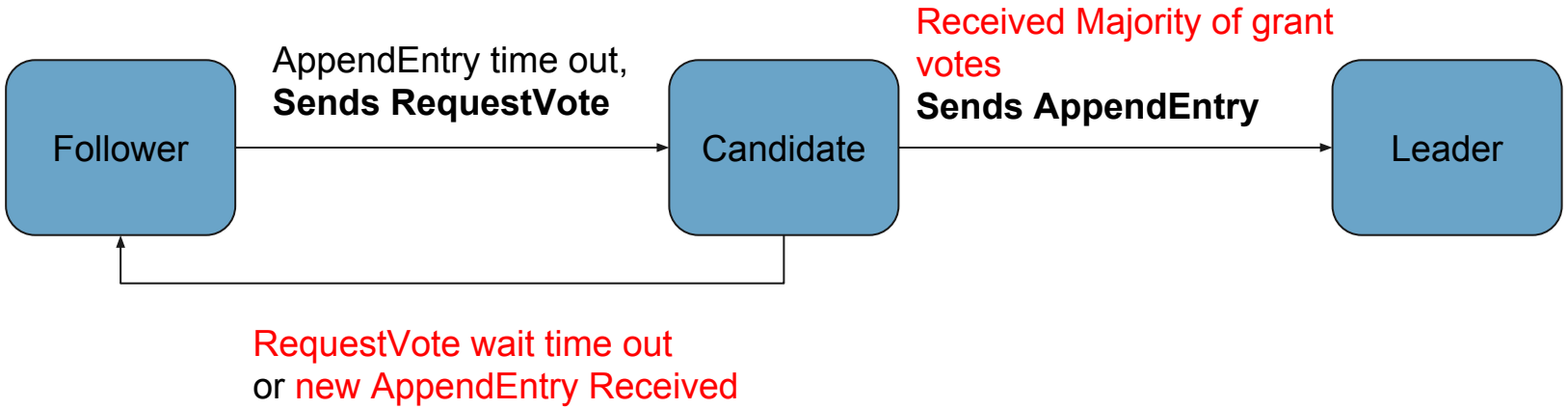
Go

- Developed by Google
  - Robert Griesemer (V8 Engine)
  - Rob Pike (Unix, Plan 9)
  - Ken Thompson (C, Unix, Plan 9)
- Go is a statically typed, compiled language in the tradition of C, with memory safety, garbage collection, structural typing, and **concurrency**
- Used by
  - Docker
  - Kubernetes
  - Ethereum
  - Hyperledger

Leader Election in Raft (Simplified Version)



Leader Election in Raft (Simplified Version)



Can happen concurrently!!

Concurrency Programming

- Multiple events can happen concurrently.
- Go introduced new concepts (compared to C language)
  - Goroutine
  - Channel
- **Debugging is very hard**

Goroutine

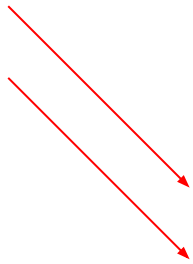
- **Lightweight thread**
- New goroutine will execute concurrently with the calling one.

- Execute a function in go: function()
- Execute a function as goroutine: **go** function()

```go
package main

import "fmt"

func f(from string) {
    for i := 0; i < 3; i++ {
        fmt.Println(from, ":", i)
    }
}

func main() {

   f("direct")

   go f("goroutine")

   go func(msg string) {
        fmt.Println(msg)
    }("going")

   fmt.Scanln()
    fmt.Println("done")
}
```

```
$ go run goroutines.go
direct : 0
direct : 1
direct : 2
goroutine : 0
going
goroutine : 1
goroutine : 2
<enter>
done
```

```go
package main

import "fmt"

func f(from string) {
    for i := 0; i < 3; i++ {
        fmt.Println(from, ":", i)
    }
}

func main() {

   f("direct")

   go f("goroutine")

   go func(msg string) {
        fmt.Println(msg)
    }("going")

   fmt.Scanln()
    fmt.Println("done")
}
```



```
$ go run goroutines.go
direct : 0
direct : 1
direct : 2
goroutine : 0
going
goroutine : 1
goroutine : 2
<enter>
done
```

Data Race

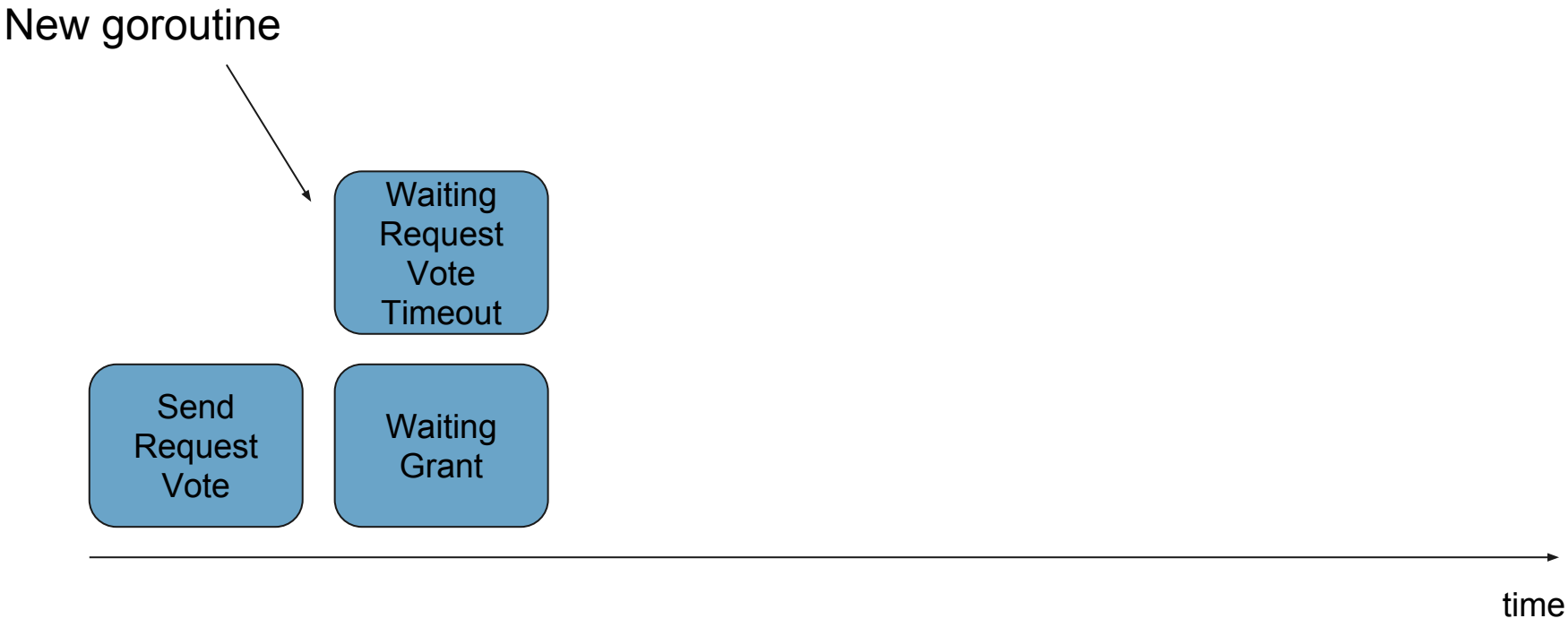
- In our project all goroutine share same ***state***. => Data Race
- Without considering data race, your program can behave unexpected!

Data Race

- In our project all goroutine share same ***state***. => Data Race
- Without considering data race, your program can behave unexpected!

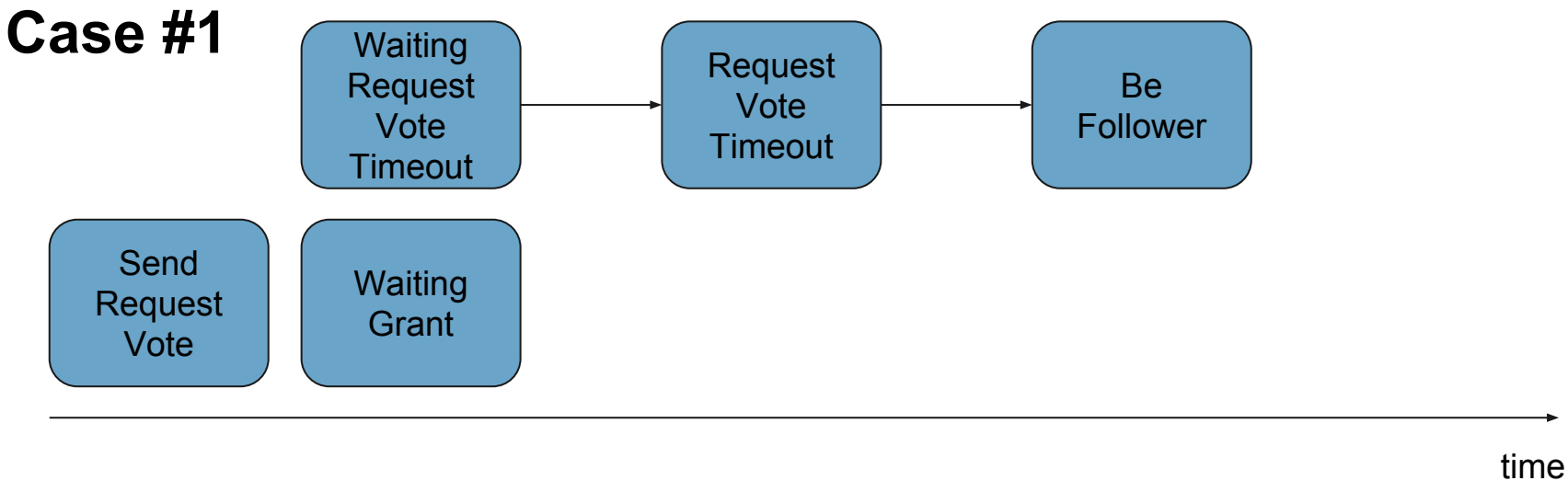
Data Race

- In our project all goroutine share same ***state***. => Data Race
- Without considering data race, your program can behave unexpected!



Data Race

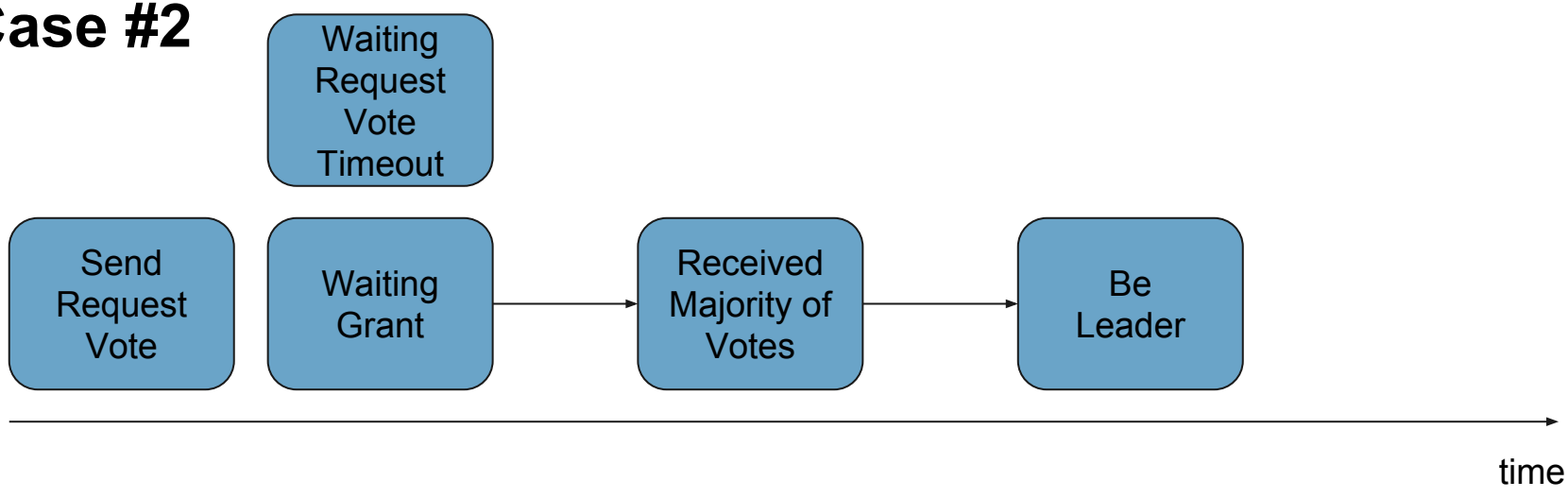
- In our project all goroutine share same ***state***. => Data Race
- Without considering data race, your program can behave unexpected!



Data Race

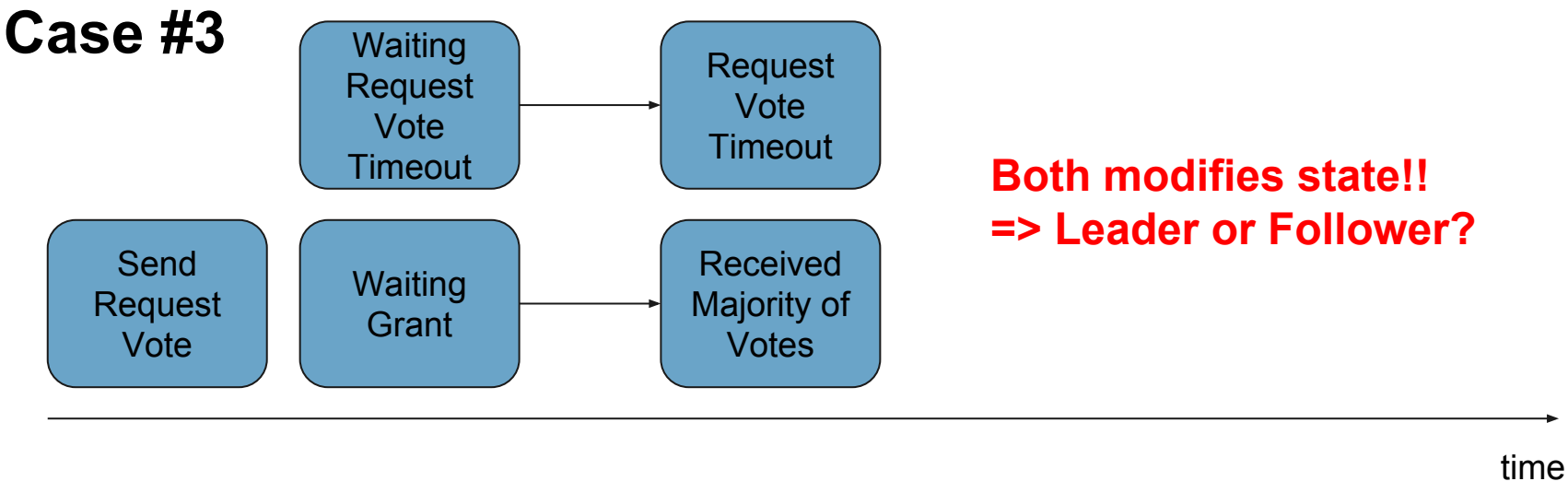
- In our project all goroutine share same ***state***. => Data Race
- Without considering data race, your program can behave unexpected!

Case #2



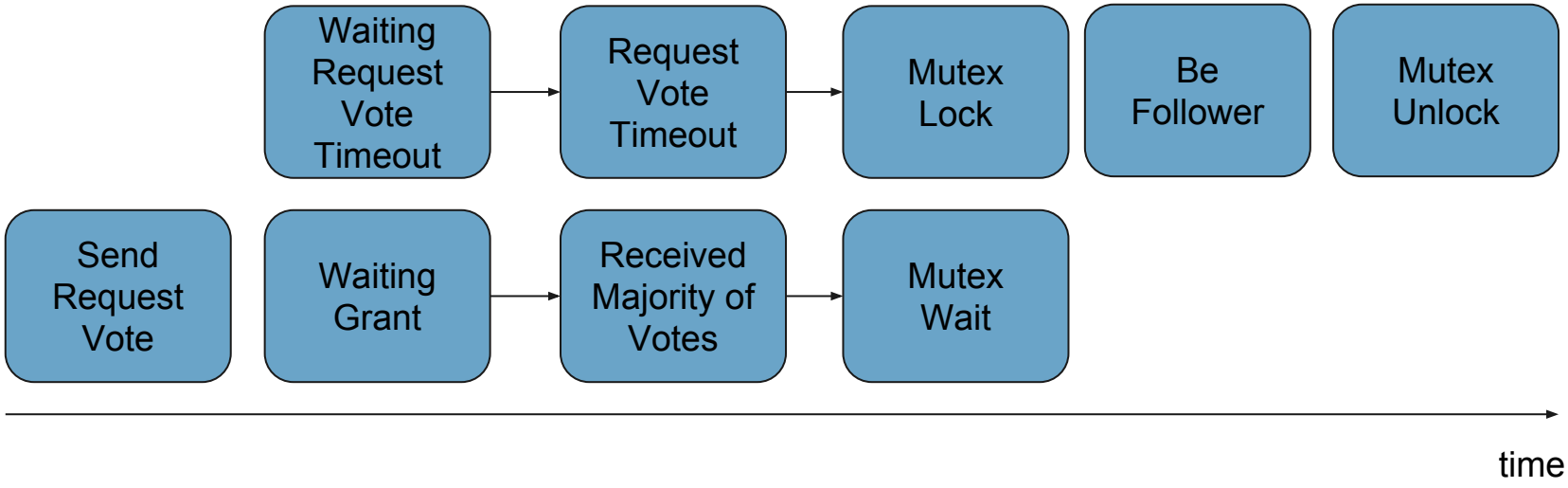
Data Race

- In our project all goroutine share same ***state***. => Data Race
- Without considering data race, your program can behave unexpected!



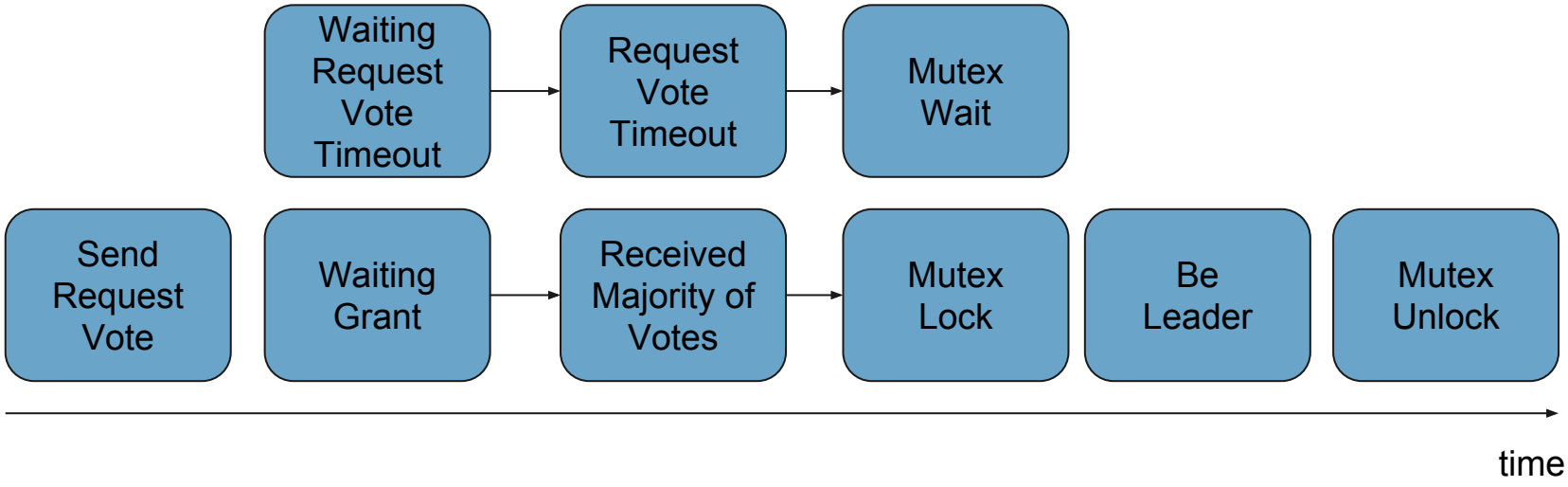
Mutex

- Our project a ***mutex***
- You can use this for the data race



Mutex

- Our project a ***mutex***
- You can use this for the data race



Deadlock

- Acquire mutex in every goroutine start.
- It can lead to ***Deadlock***

Deadlock

- Acquire mutex in every goroutine start.
- It can lead to ***Deadlock***

- Mutex acquired
- Waiting timer

- Waiting mutex unlock

Slack

- Please visit our slack channel #raft for any questions or discussions

Alert!

COPY = F