IT大咖说
知识分享平台

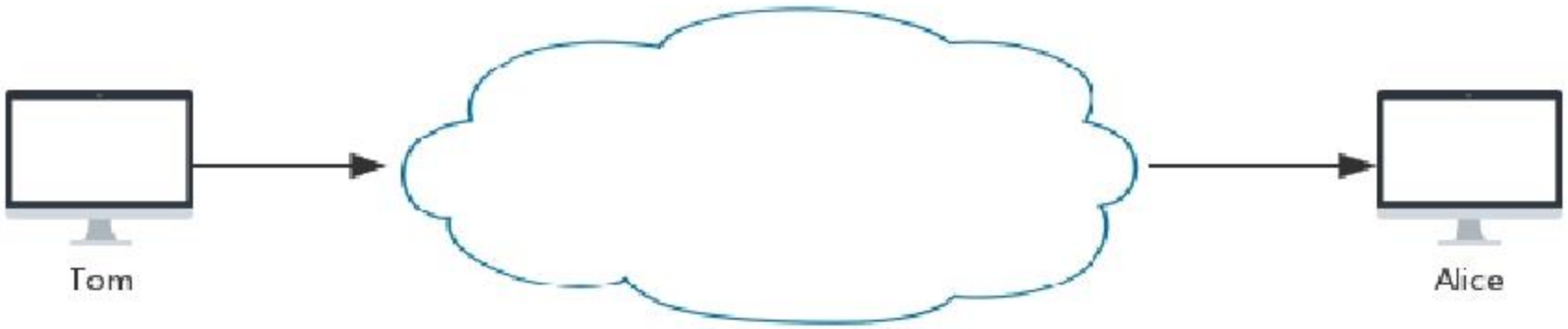
美图　长连接消息通道

架构平台 任勇全

meitu

美图　长连接消息通道

- Tom 如何将消息发给Alice



bifrost

什什么是bifrost

IT大咖说 知识分享平台

- 北北欧神话彩虹桥
- 适配多种业务场景
- 保证消息可靠传输

meitu

适配不不同的消息模型

IT大咖说 知识分享平台

直播IM与传统IM是否具有相同的消息模型？

meitu

传统IM

- 一一对一一单聊
- 多对多群聊
- 群不不会过大大

直播IM

IT大咖说 知识分享平台

- 百万人人群嗨的聊天室
- 有些消息更更重要（比比如礼物）？

meitu

什什么是bifrost

IT大咖说 知识分享平台

- 北北欧神话彩虹桥
- 适配多种业务场景
- 保证消息可靠传输

meitu

美图　长连接消息通道

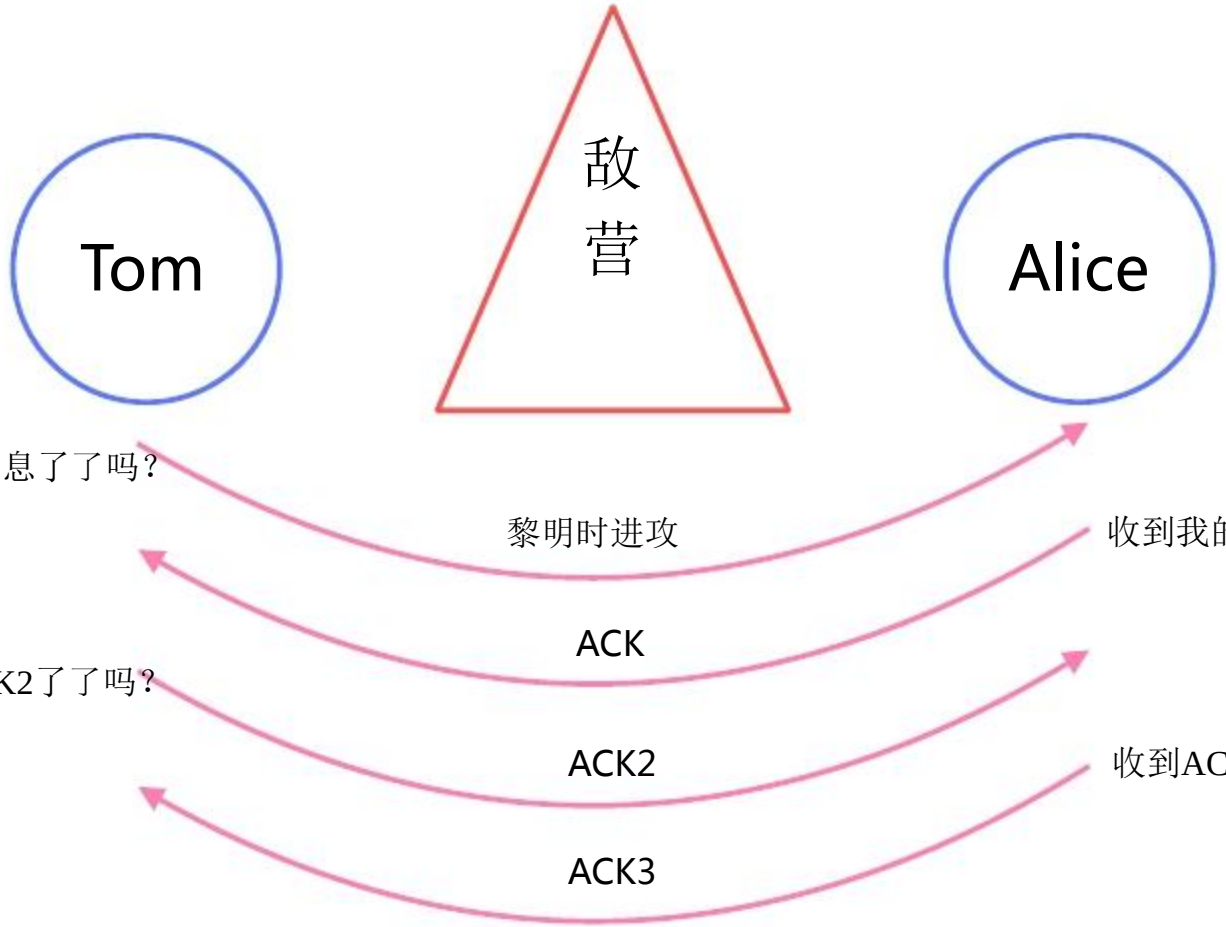
两军问题

IT大咖说 知识分享平台

Tom和Alice约定在同一一时刻发起进攻，单独进攻的话会攻击失败

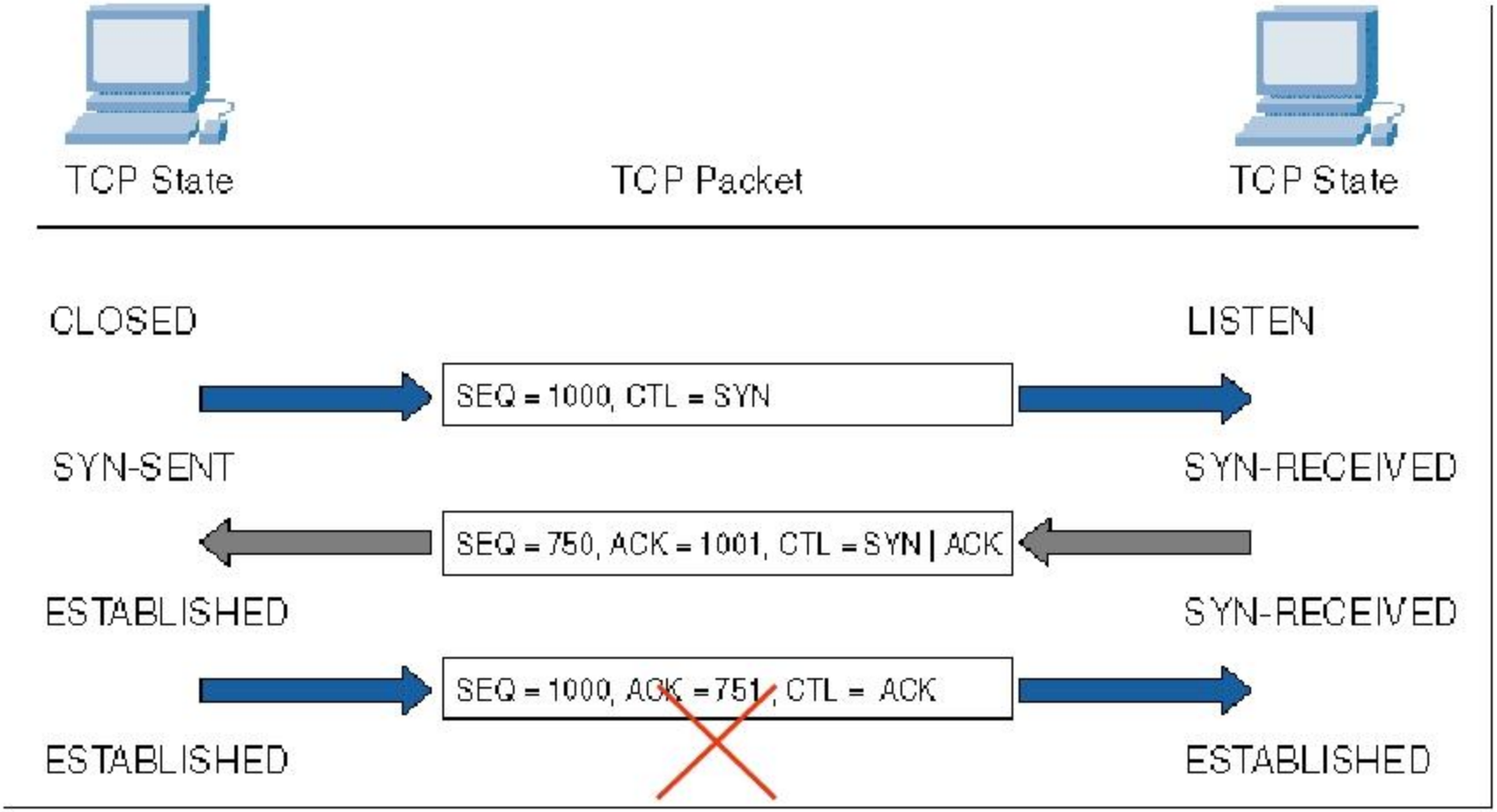
meitu

两军问题-永远无无法达成一一致



-
- 永远无无法达成一一致
-

TCP 状态总会一一致吗？



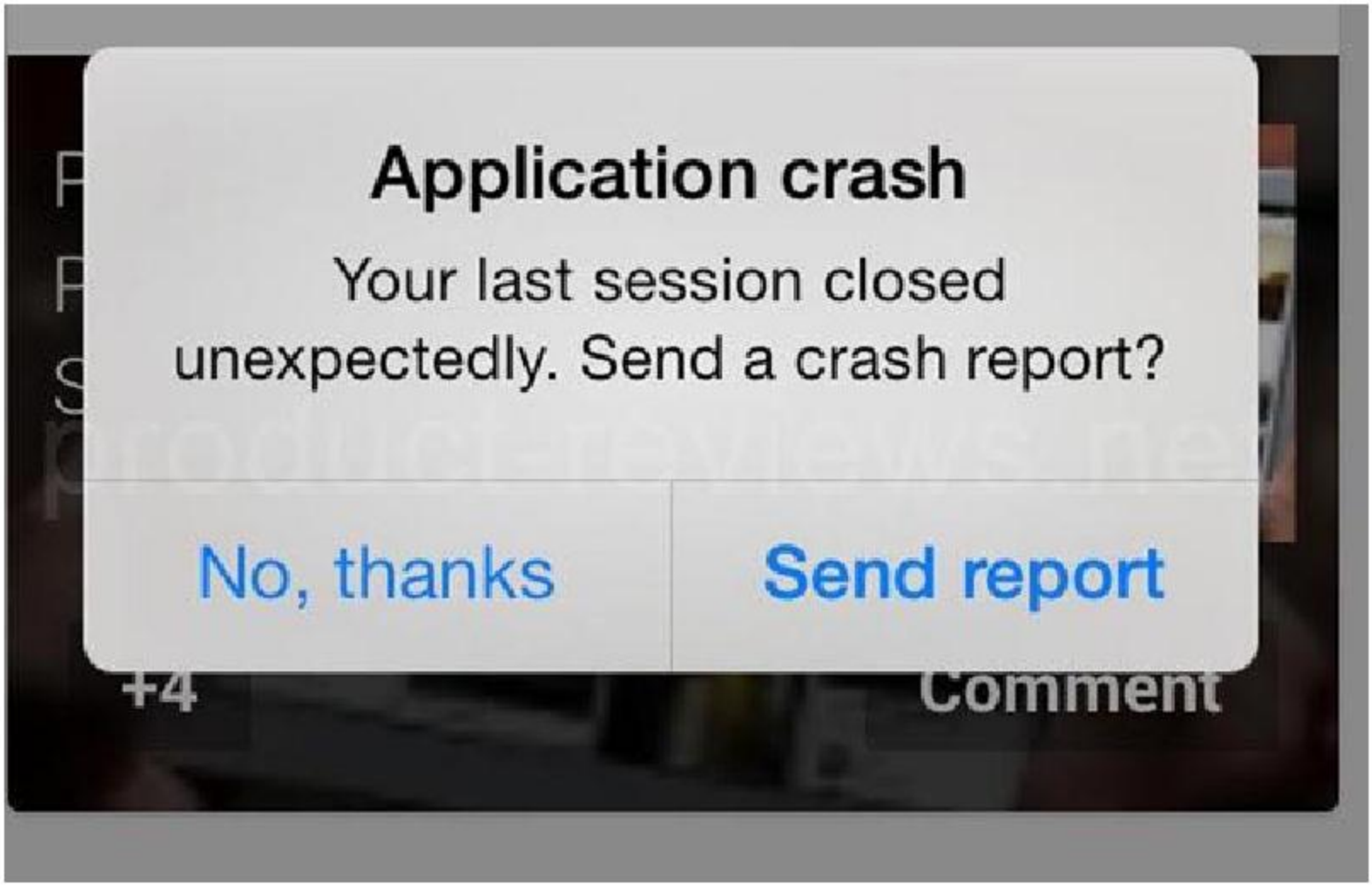
应用用层可靠

IT大咖说 知识分享平台

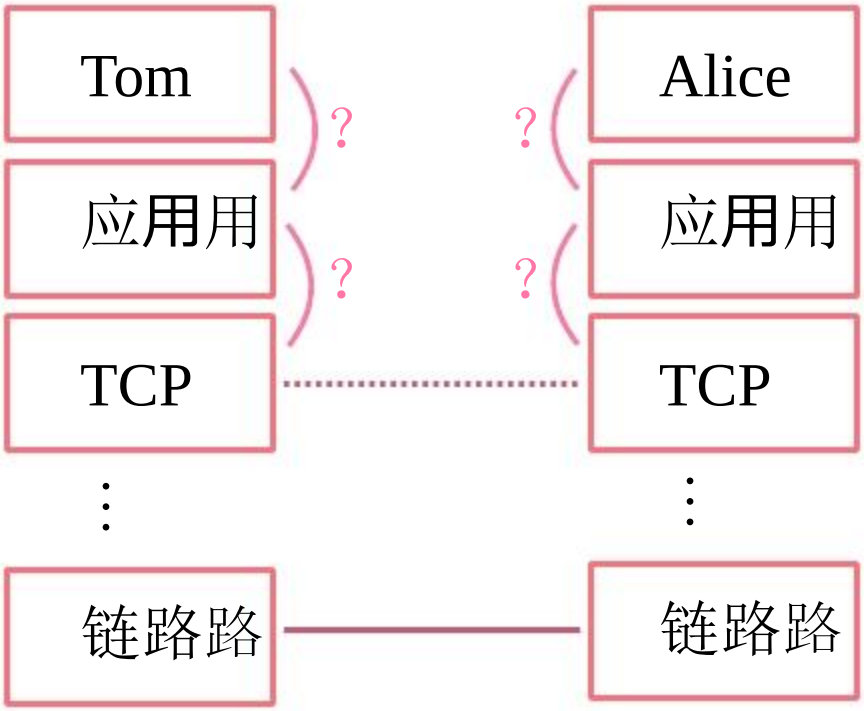
TCP保证传输层可靠还不不够吗？

meitu

应用用层可靠

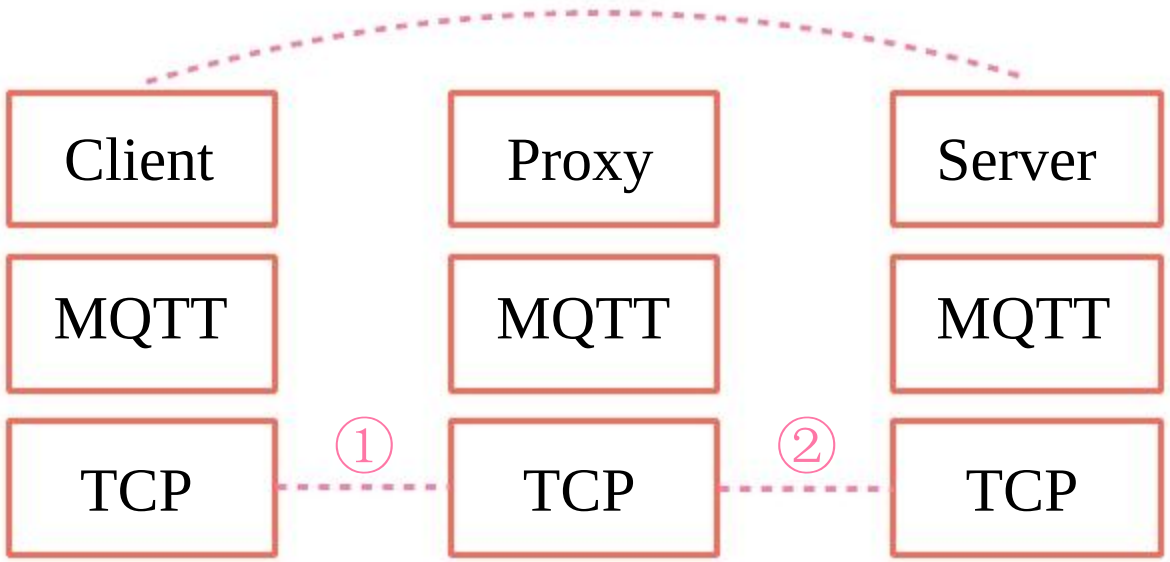

Application crash
Your last session closed unexpectedly. Send a crash report?
No, thanks
Send report
+4
Comment

应用用层可靠



应用用层可靠

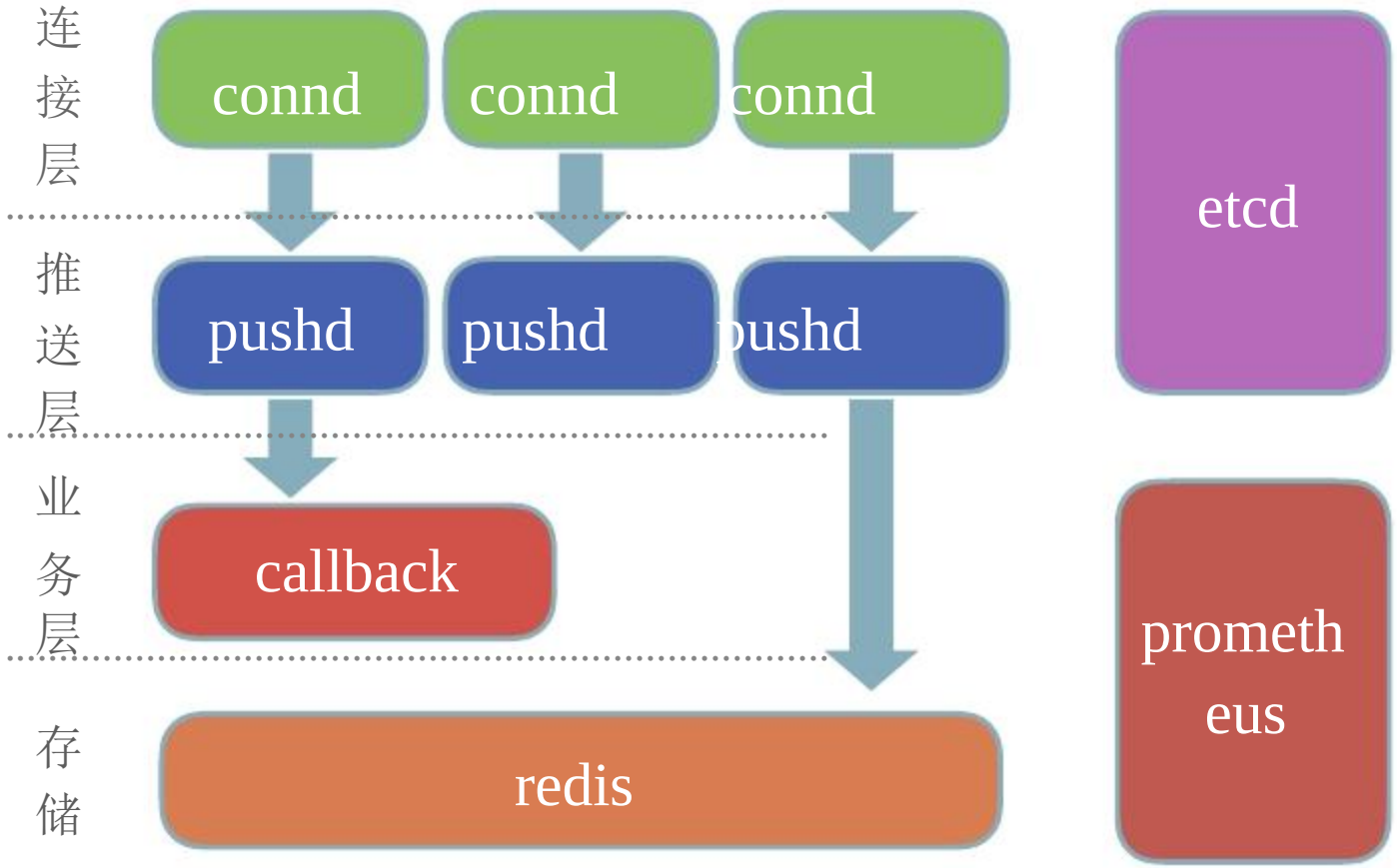
应用用层可靠



MQTT 协议

- 物联网网传输标准
- 二二进制协议，消息格式精简，适合移动设备
- Pub/Sub，适合多种消息传输模型
- 依赖TCP，并支支持应用用层可靠传输

bifrost 架构


连接层
connd
connd
connd
推送层
pushd
pushd
pushd
业务层
callback
存储
redis
etcd
prometh
eus

美图　长连接消息通道

- Tom 如何将消息发给Alice

连接管理理的挑战

- 千万级用用户连接，并可能会不不断增　长
- 维护连接状态

线程模型

并发连接的难题—线程内存占用用

- ulimit -s(8192KB)
- PTHREAD_STACK_MIN(16384)
- pthread_attr_setstacksize

并发连接的难题—线程内存占用用

- man pthread_create

  On Linux/x86-32, the default stack size for a new thread is 2 megabytes. Under the NPTL threading implementation, if the RLIMIT_STACK soft resource limit at the time the program started has any value other than "unlimited", then it determines the default stack size of new threads.

并发连接的难题—线程内存占用用

- man pthread_setstacksize

  pthread_attr_setstacksize() can fail with the following error:

  EINVAL The stack size is less than PTHREAD_STACK_MIN (16384) bytes

并发连接的难题—线程上下文文切换

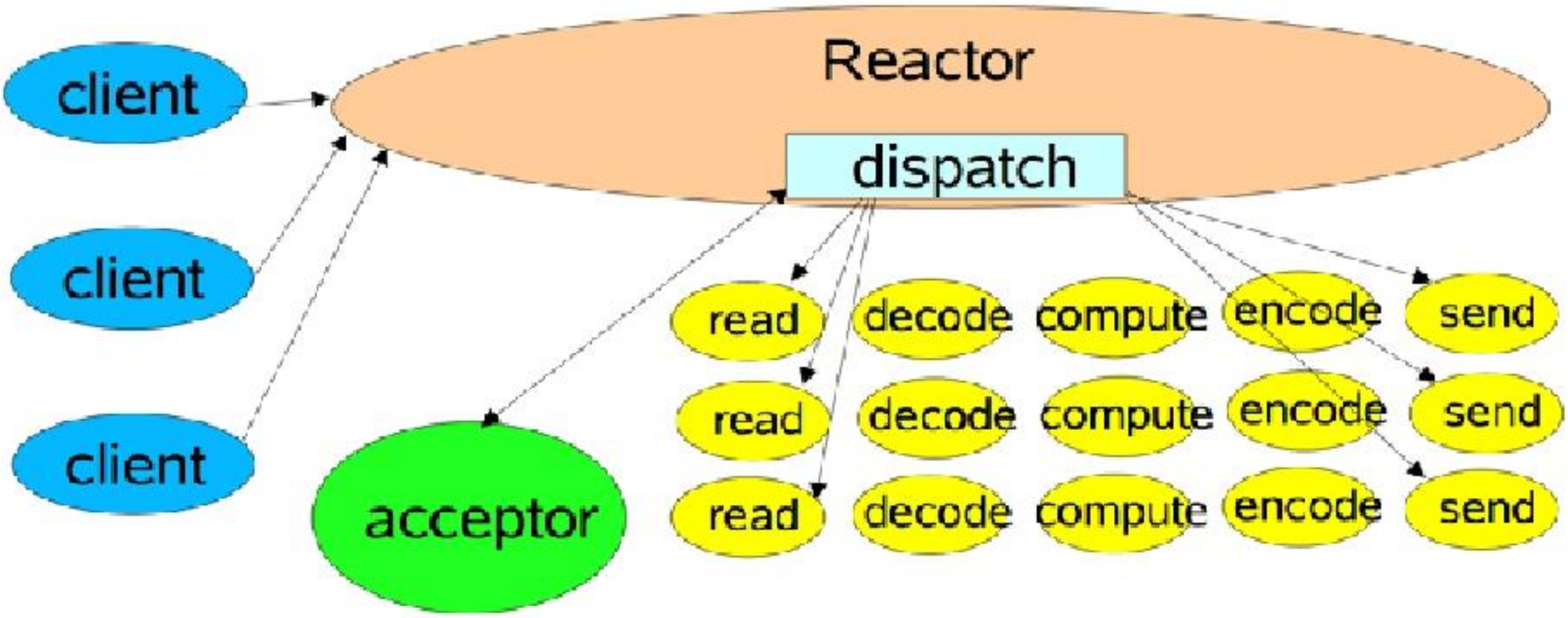
IT大咖说 知识分享平台

The #1 cause of context switches is having more active threads than you have processors. As the ratio of active threads to processors increases, the number of context switches also increases - linearly if you're lucky, but often exponentially.

meitu

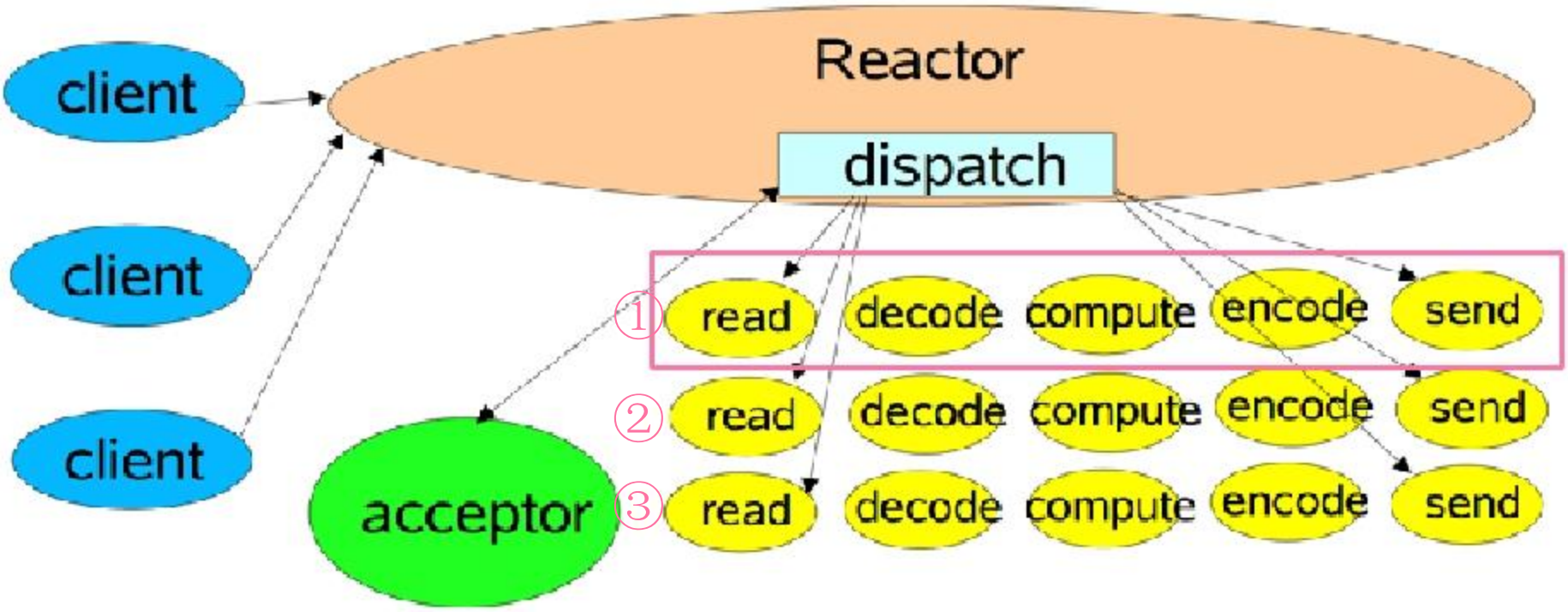
事件模型

并发连接的难题—select, poll, epoll



调度模型

Golang的并发—抽象执行行行路路径



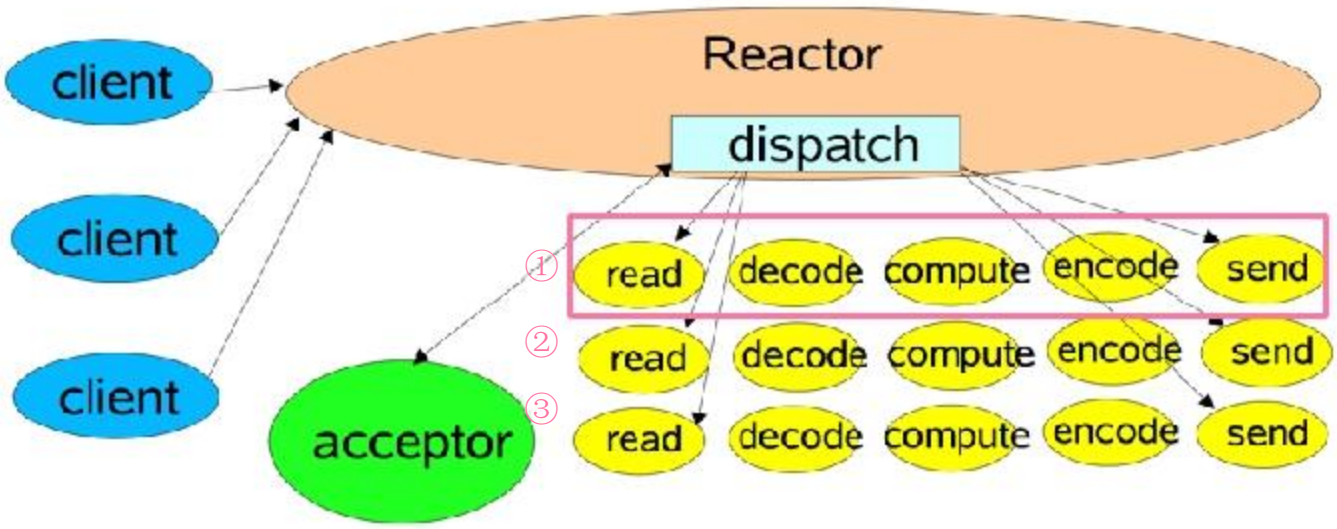
Golang 的并发—goroutine 调度时机

网网络IO

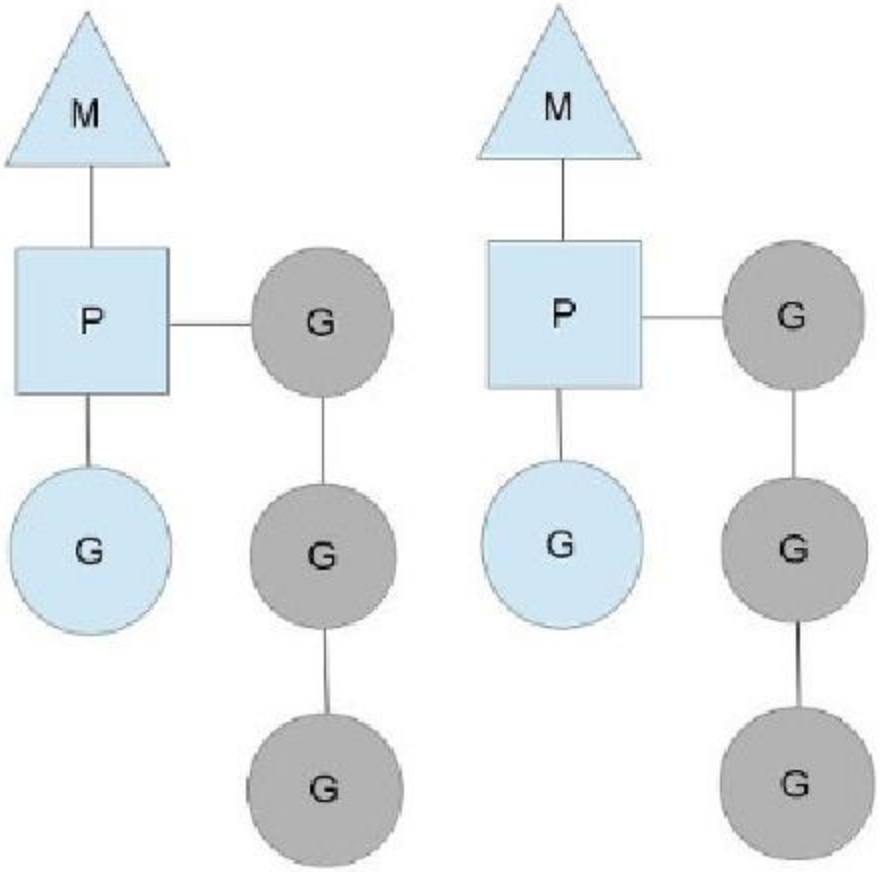
系统调用用

channel读写

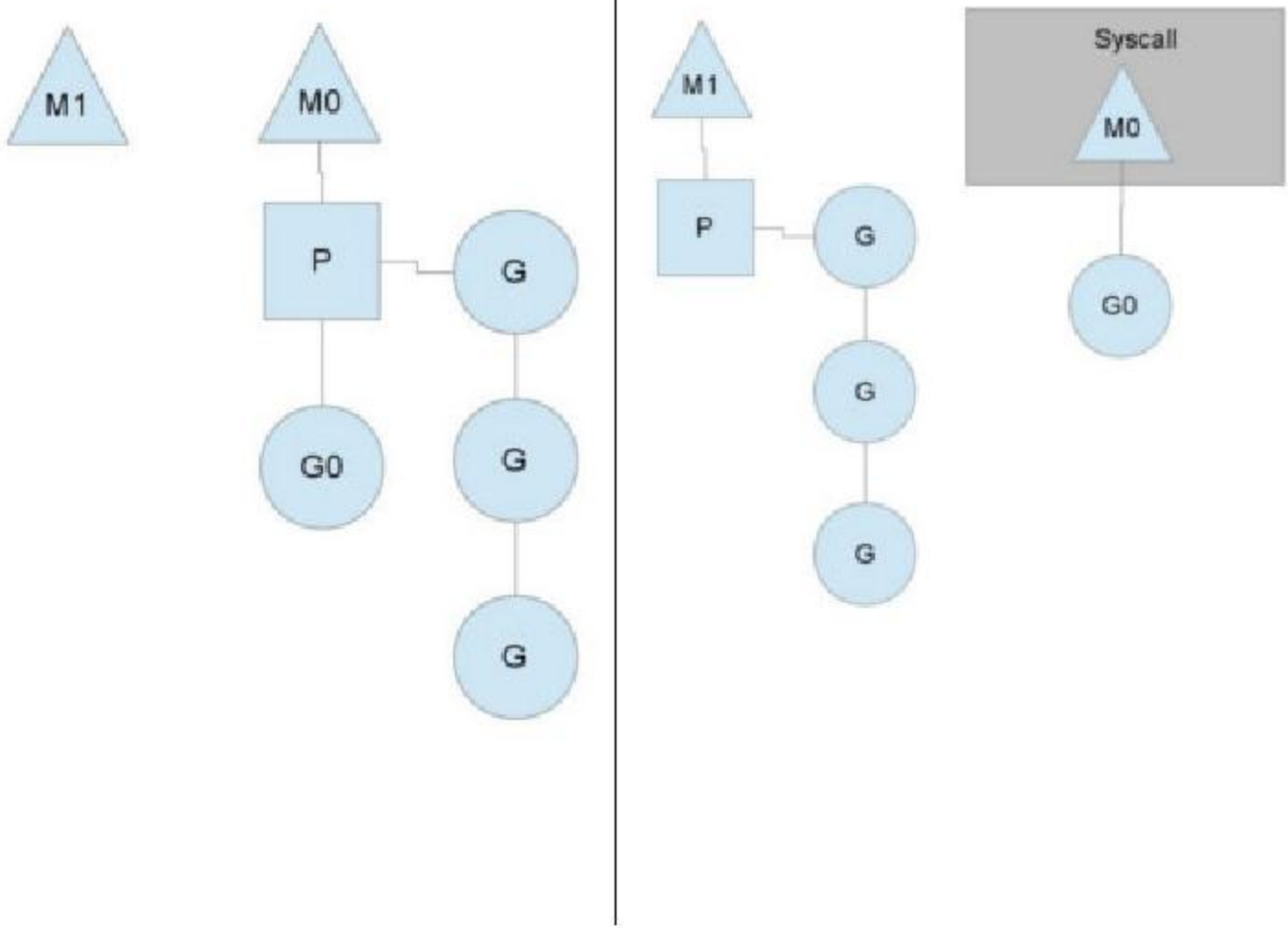
抢占



Golang的并发—goroutine 调度



Golang的并发—goroutine 调度—syscall



Golang的并发—goroutine

IT大咖说 知识分享平台

- 用用户态线程，调度更更轻量量
- 动态Stack大大小小，最小小2KB，Stack无无溢出

meitu

连接、状态与路路由

IT大咖说 知识分享平台

连接是两端互相拥有对方方的识别信息及关联数据

meitu

连接、状态与路路由

	STATE	LOCAL ADDRESS	LOCAL PORT	REMOTE ADDRESS	REMOTE PORT
Connection 1					
Connection 2					
Connection 3					
Connection n					

连接、状态与路路由

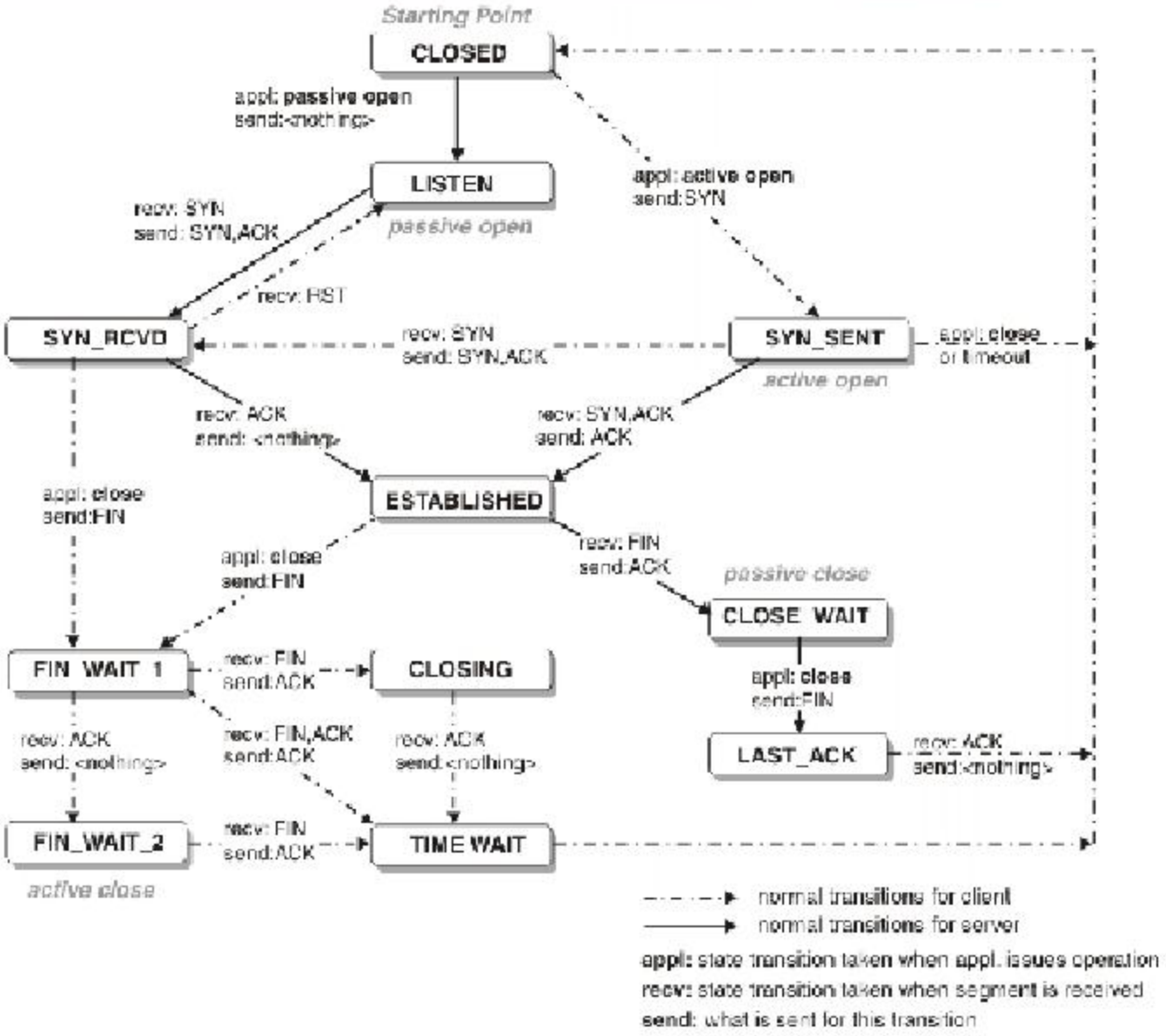
IT大咖说 知识分享平台

In information technology and computer science, a program is described as stateful if it is designed to remember preceding events or user interactions; the remembered information is called the state of the system.

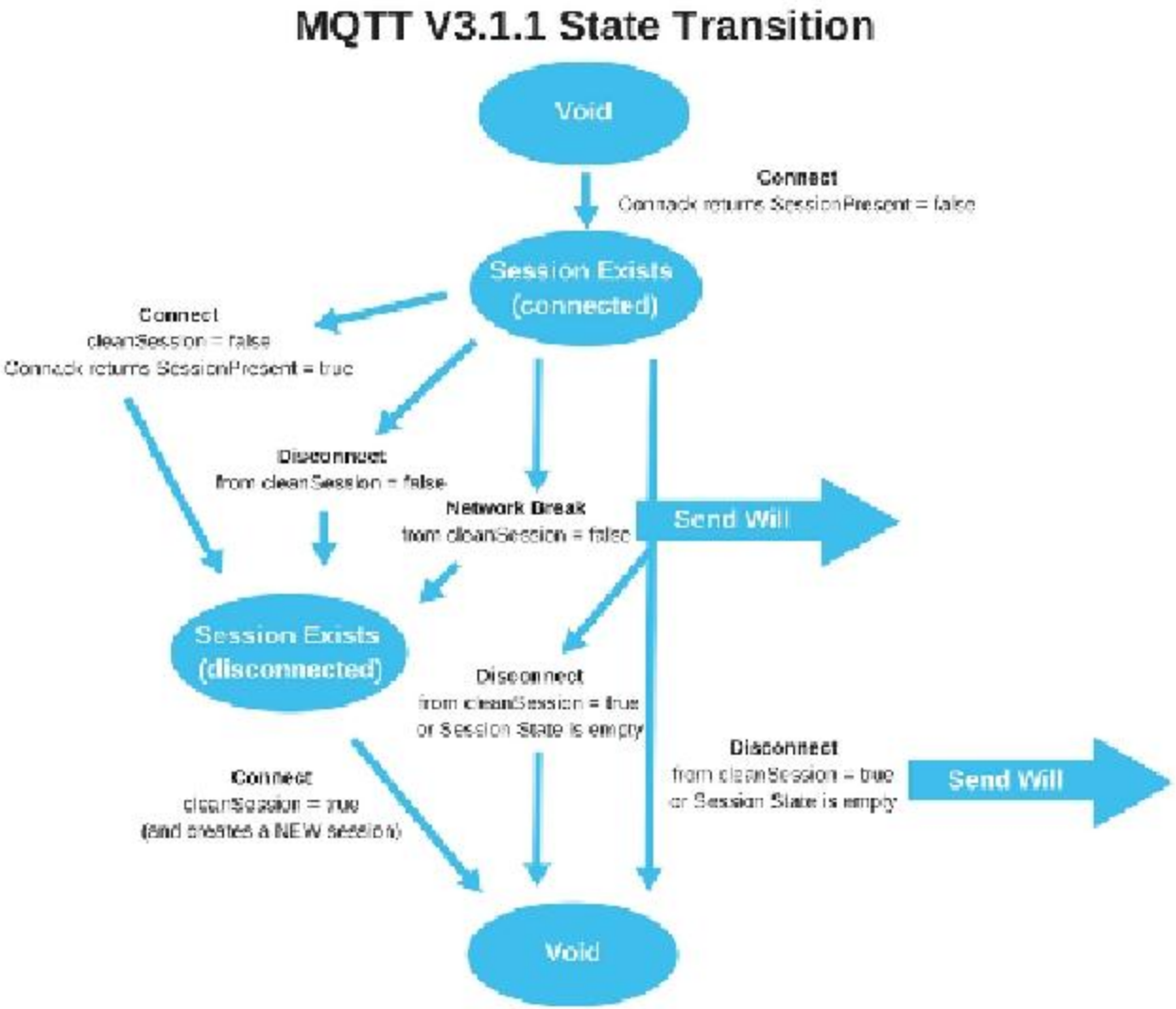
—Wikipedia

meitu

连接状态与路路由


Starting Point
CLOSED
appl: passive open
send:<nothing>
LISTEN
passive open
appl: active open
send:SYN
recv: SYN
send: SYN,ACK
recv: RST
SYN_RCVD
recv: SYN
send: SYN,ACK
SYN_SENT
appl: close
or timeout
active open
recv: ACK
send: <nothing>
recv: SYN,ACK
send: ACK
appl: close
send:FIN
ESTABLISHED
appl: close
send:FIN
recv: FIN
send:ACK
passive close
CLOSE_WAIT
FIN_WAIT_1
recv: FIN
send:ACK
CLOSING
appl: close
send:FIN
recv: ACK
send: <nothing>
recv: FIN,ACK
send:ACK
recv: ACK
send:<nothing>
LAST_ACK
recv: ACK
send:<nothing>
FIN_WAIT_2
recv: FIN
send:ACK
TIME WAIT
active close
normal transitions for client
normal transitions for server
appl: state transition taken when appl. issues operation
recv: state transition taken when segment is received
send: what is sent for this transition

连接状态与路路由


MQTT V3.1.1 State Transition
Void
Connect
Connack returns SessionPresent = false
Session Exists
(connected)
Connect
cleanSession = false
Connack returns SessionPresent = true
Disconnect
from cleanSession = false
Network Break
from cleanSession = false
Send Will
Session Exists
(disconnected)
Disconnect
from cleanSession = true
or Session State is empty
Disconnect
from cleanSession = true
or Session State is empty
Send Will
Connect
cleanSession = true
(and creates a NEW session)
Void

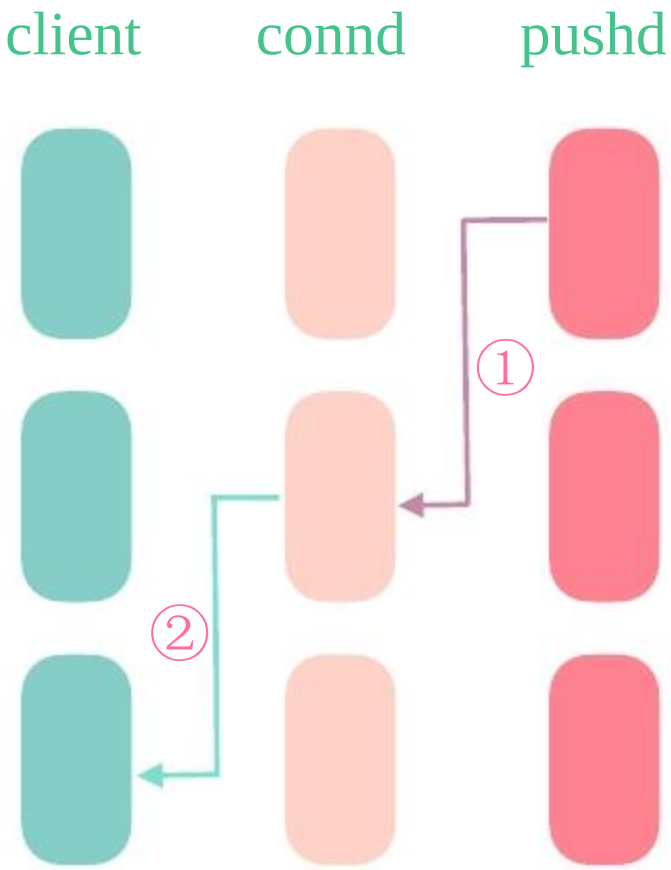
连接状态与路路由

IT大咖说 知识分享平台

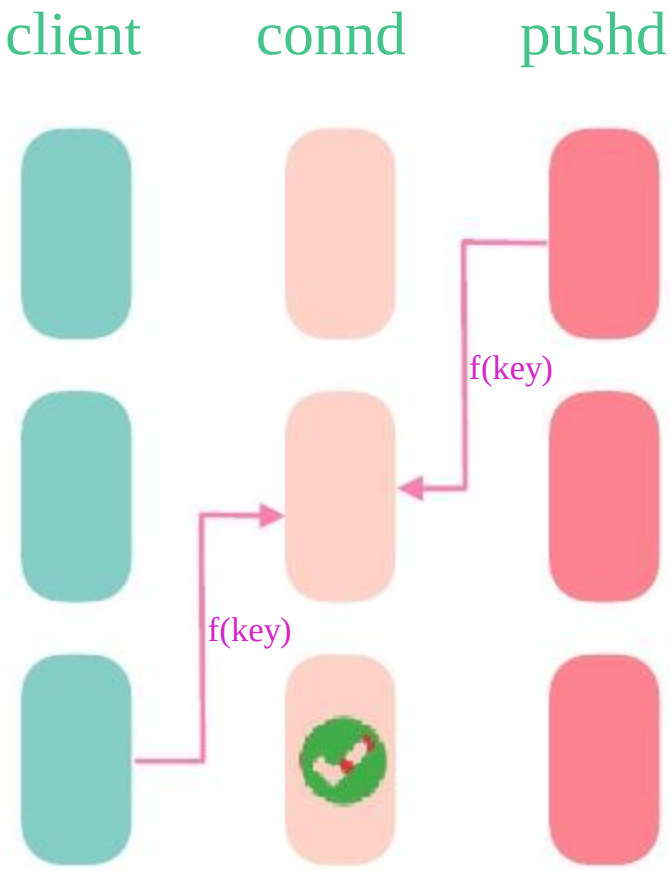
Bifrost 基于状态信息进行行行消息路路由

meitu

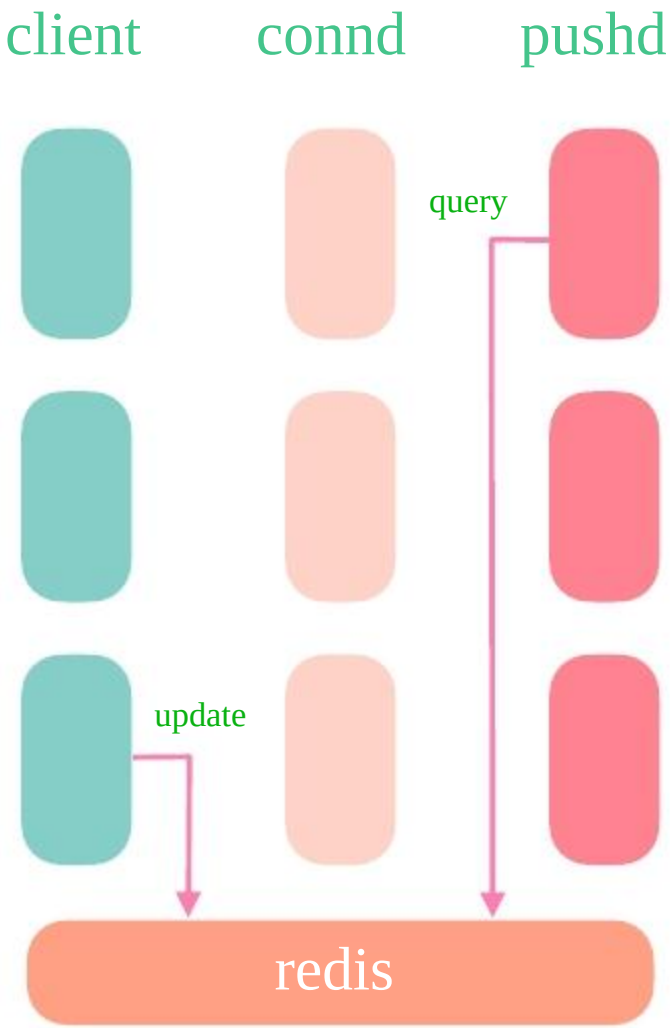
连接状态与路路由



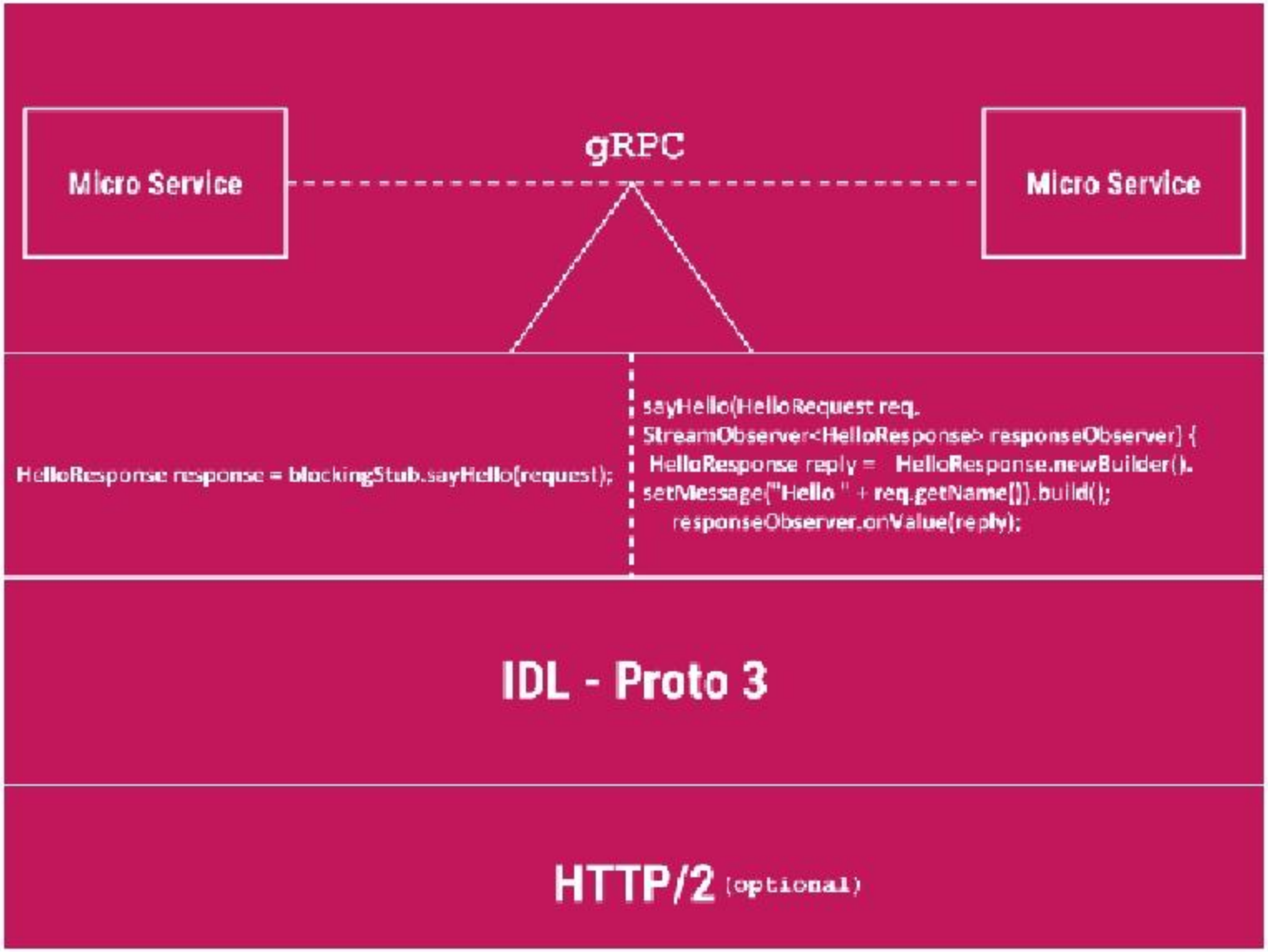
连接状态与路路由—算法约定



连接状态与路路由—外部存储

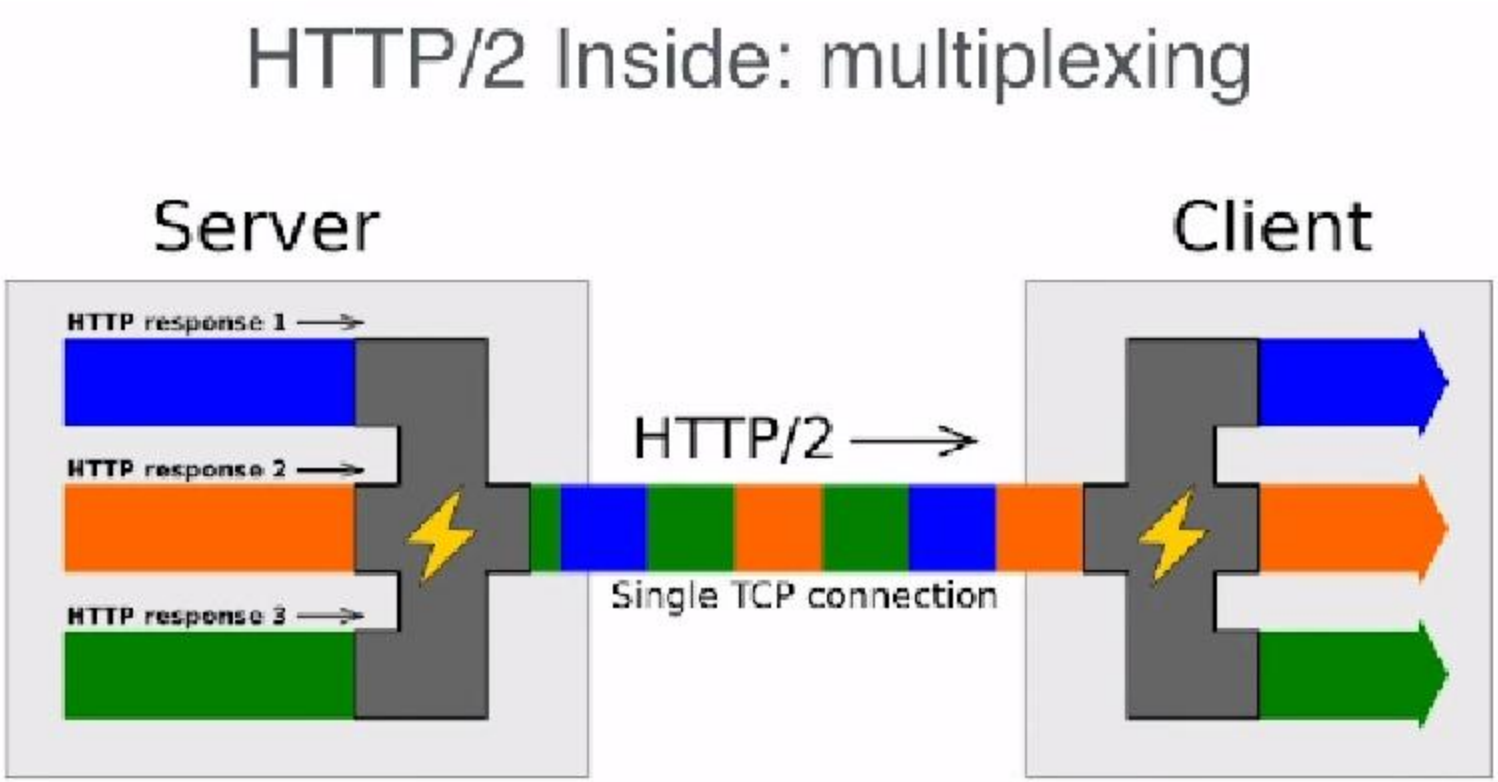


分层间通信


Micro Service
gRPC
Micro Service
HelloResponse response = blockingStub.sayHello(request);
sayHello(HelloRequest req,
StreamObserver<HelloResponse> responseObserver) {
HelloResponse reply = HelloResponse.newBuilder().
setMessage("Hello " + req.getName()).build();
responseObserver.onValue(reply);
IDL - Proto 3
HTTP/2 (optional)

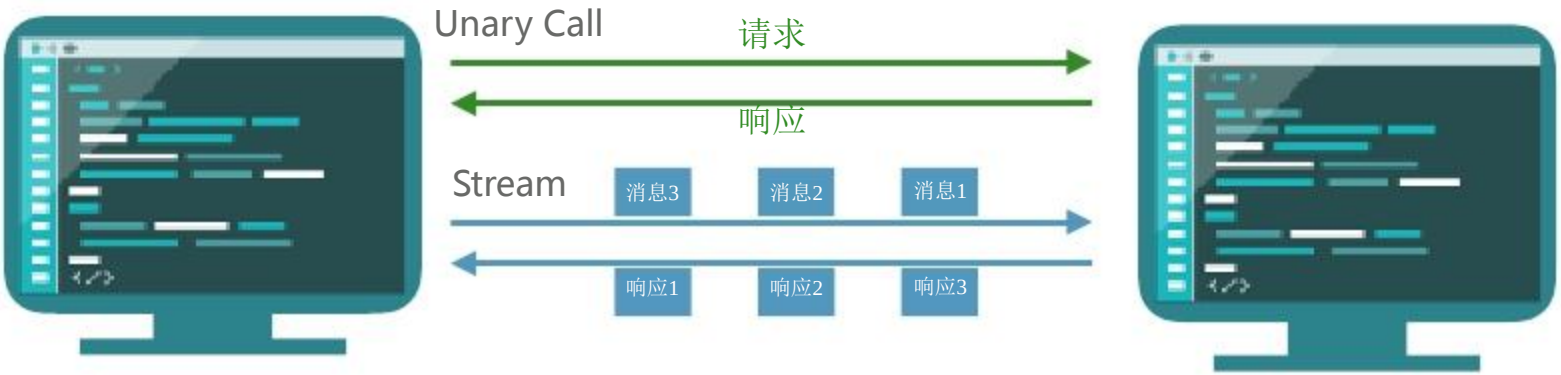
分层间通信—http/2 多路路复用用

IT大咖说 知识分享平台

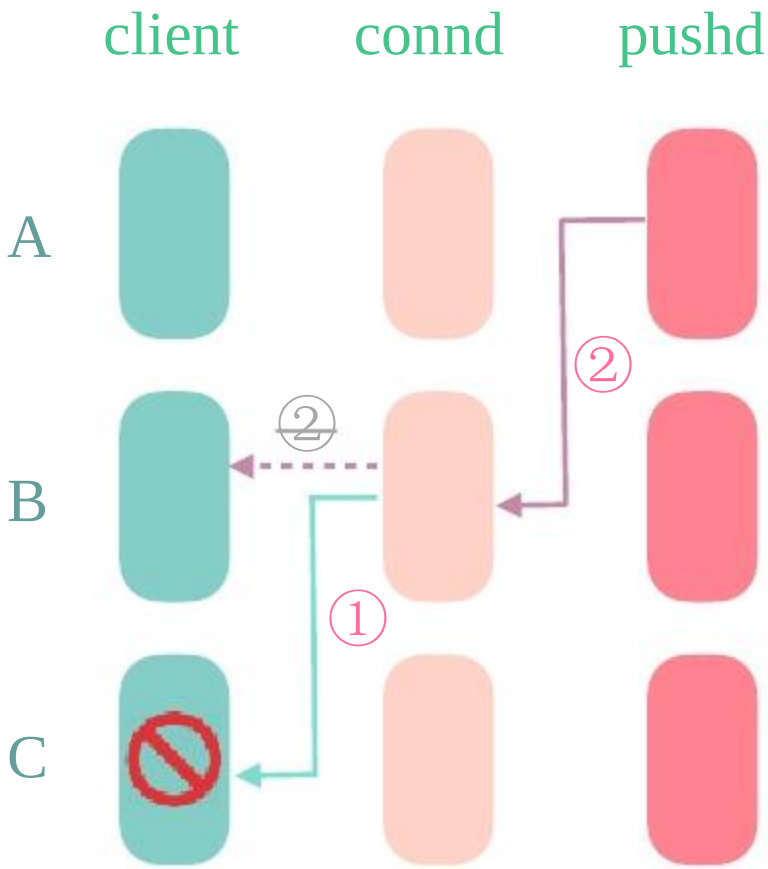


meitu

Stream or Unary call ？

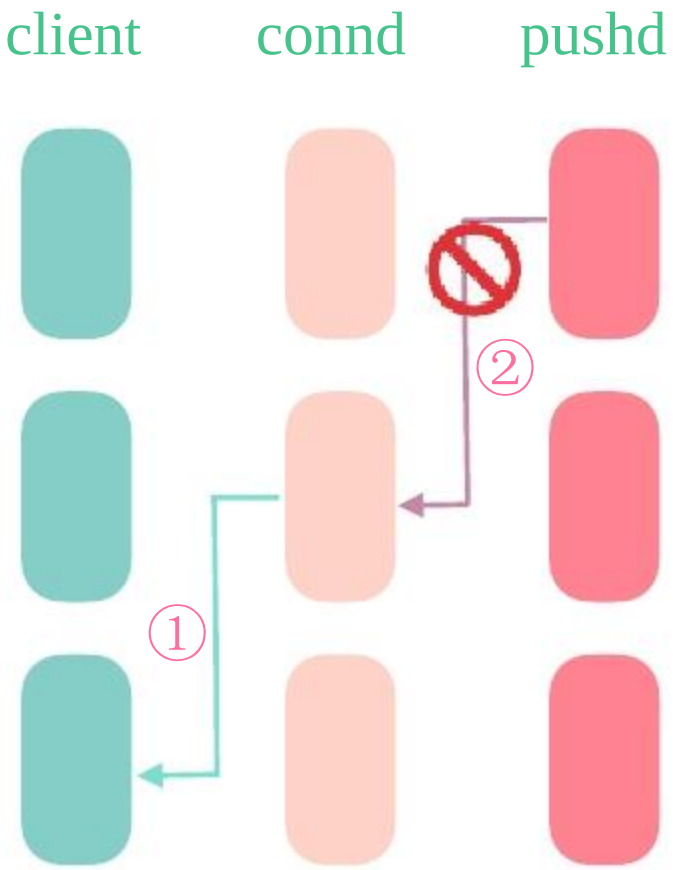


Stream的难题—Head of line blocking



Stream的难题—消息可靠

IT大咖说 知识分享平台



meitu

分层间通信

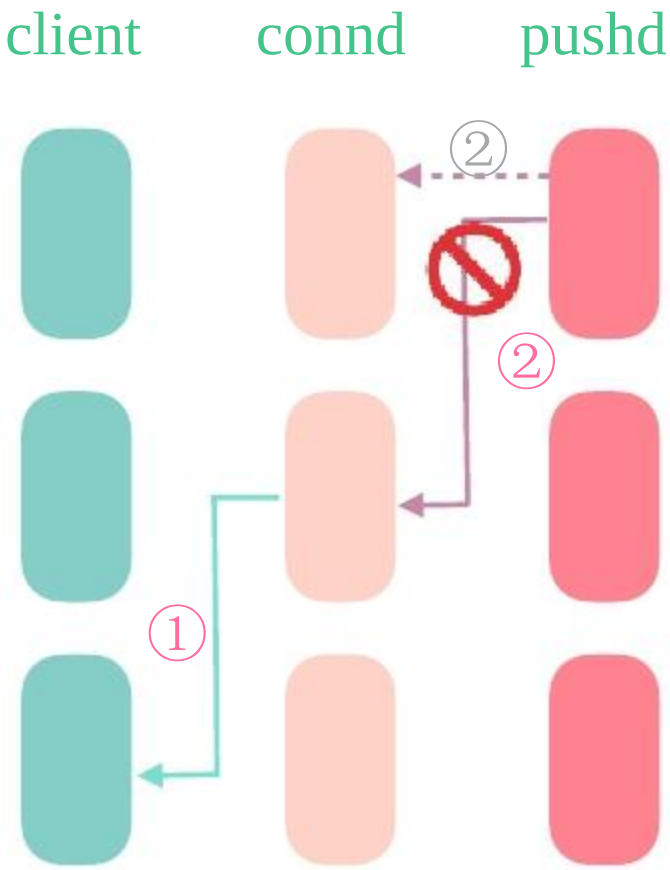
IT大咖说 知识分享平台

- Stream
  - 较高高的传输性能
  - 单Stream容易易造成HOL blocking
  - 消息可靠性实现困难

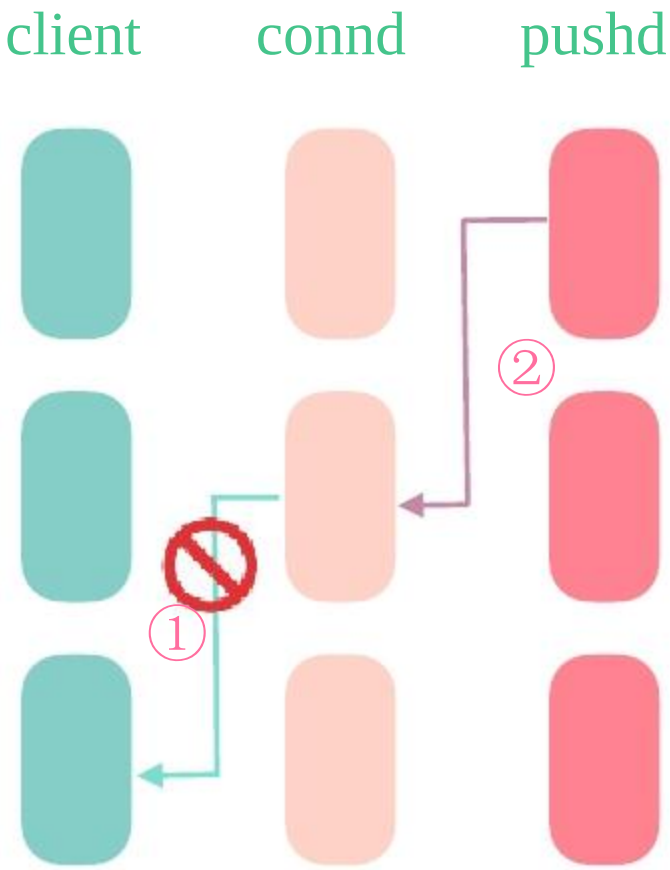
- Unary call
  - 性能比比Stream差，但够用用
  - 无无HOL blocking
  - 重试或故障转移保证消息可靠

meitu

可靠性保证—pushd 到 connd

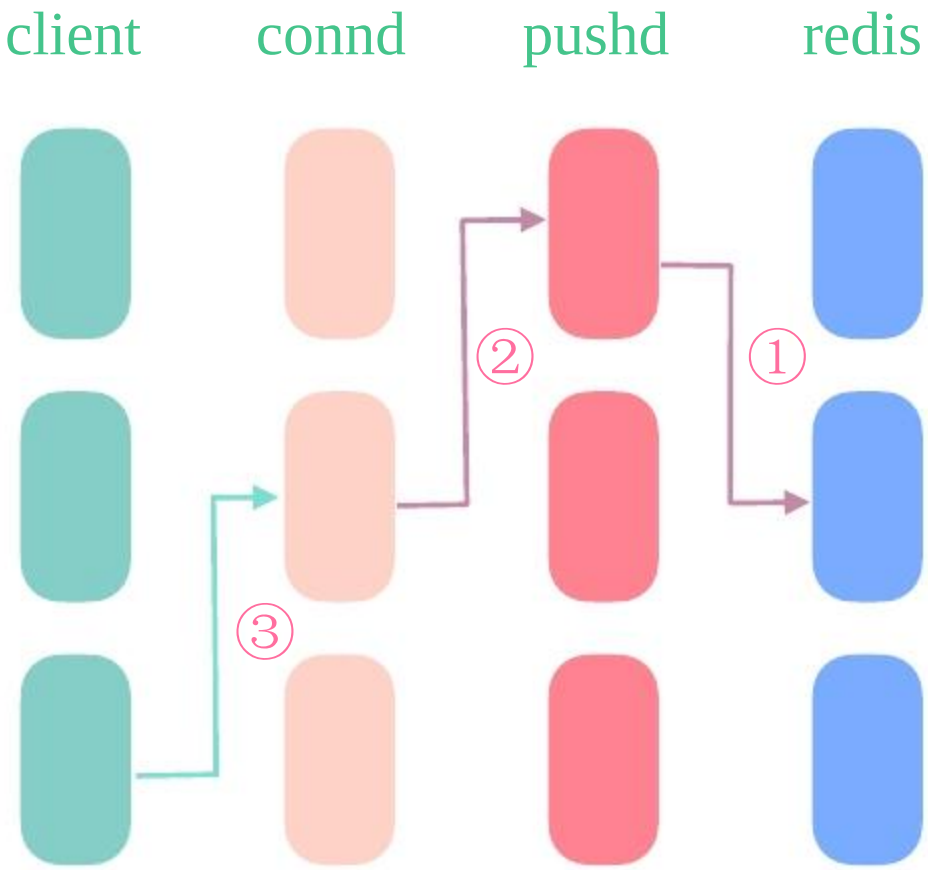


可靠性保证—connd 到 client



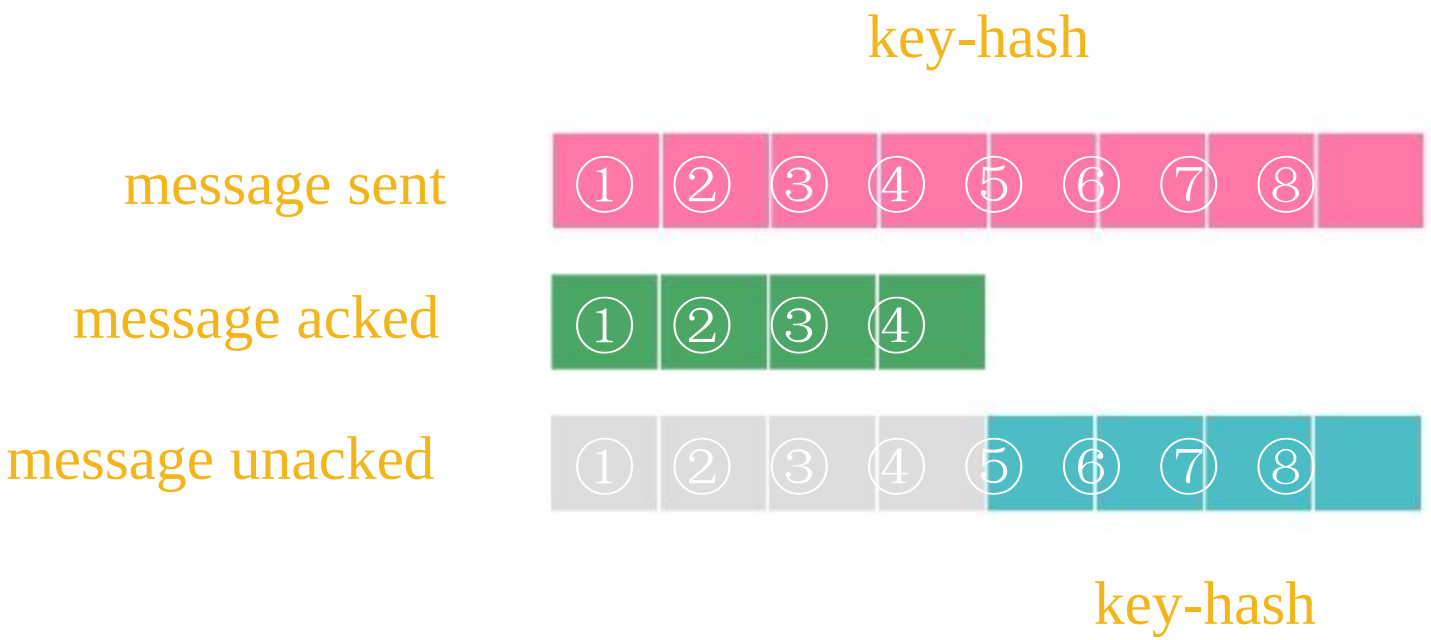
可靠性保证—持久化到redis

IT大咖说 知识分享平台



meitu

可靠性保证—持久化到redis



可靠性保证—持久化到redis

IT大咖说 知识分享平台

- 原子子操作
  - multi-exec
  - lua脚本

- Key-hash 查询性能
  - hgetall
  - hscan
  - 降级为广广播，避免查询

meitu

合适的语言言

为什什么是Go？

Golang—简单

IT大咖说 知识分享平台

595 (17%)	simplicity	
543 (15%)	easy	
523 (15%)	concurrency	
495 (14%)	simple	
454 (13%)	fast	
293 (8%)	syntax	
287 (8%)	standard library	
286 (8%)	tooling	
270 (8%)	static	
266 (7%)	performance	
235 (7%)	speed	
202 (6%)	interfaces	
184 (5%)	channels	
183 (5%)	community	
180 (5%)	good	
177 (5%)	compilation	
177 (5%)	goroutines	
167 (5%)	binary	
156 (4%)	great	
148 (4%)	tools	
146 (4%)	compiled	
137 (4%)	compile	
127 (4%)	type	
124 (3%)	small	
118 (3%)	c	
114 (3%)	gofmt	
114 (3%)	libraries	
88 (2%)	clean	
87 (2%)	easy to learn	
82 (2%)	deployment	
78 (2%)	memory	
78 (2%)	strong	
76 (2%)	concise	
76 (2%)	single binary	
73 (2%)	low	
73 (2%)	static typing	
71 (2%)	build	
68 (2%)	easy to read	
63 (2%)	fast compilation	
56 (2%)	simple syntax	
55 (2%)	type system	
54 (2%)	simple language	
51 (1%)	easy concurrency	
47 (1%)	static binaries	
46 (1%)	go fmt	
45 (1%)	fast compile	
43 (1%)	small language	
41 (1%)	error handling	
39 (1%)	concurrency model	
39 (1%)	go routines	
38 (1%)	easy to use	
38 (1%)	statically typed	
36 (1%)	cross platform	
35 (1%)	concurrency primitives	
35 (1%)	goroutines channels	
33 (1%)	easy to write	
27 (1%)	great standard library	
23 (1%)	ease of use	
940 (26%)	No response	

meitu

Simplicity

Number of keywords is an approximate measure of complexity

C (K&R)	K&R	32
C++	1991	48
Java	3rd edition	50
C#	2010	77
C++0x	2010	72+11*
JavaScript	ECMA-262	26+16*
Python	2.7	31
Pascal	ISO	35
Modula-2	1980	40
Oberon	1990	32
Go	2010	25

Hello, world 2.0

Serving http://localhost:8080/world:

```
package main

import (
    "fmt"
    "http"
)

func handler(c *http.Conn, r *http.Request) {
    fmt.Fprintf(c, "Hello, %s.", r.URL.Path[1:])
}

func main() {
    http.ListenAndServe(":8080",
                        http.HandlerFunc(handler))
}
```

Golang—并发

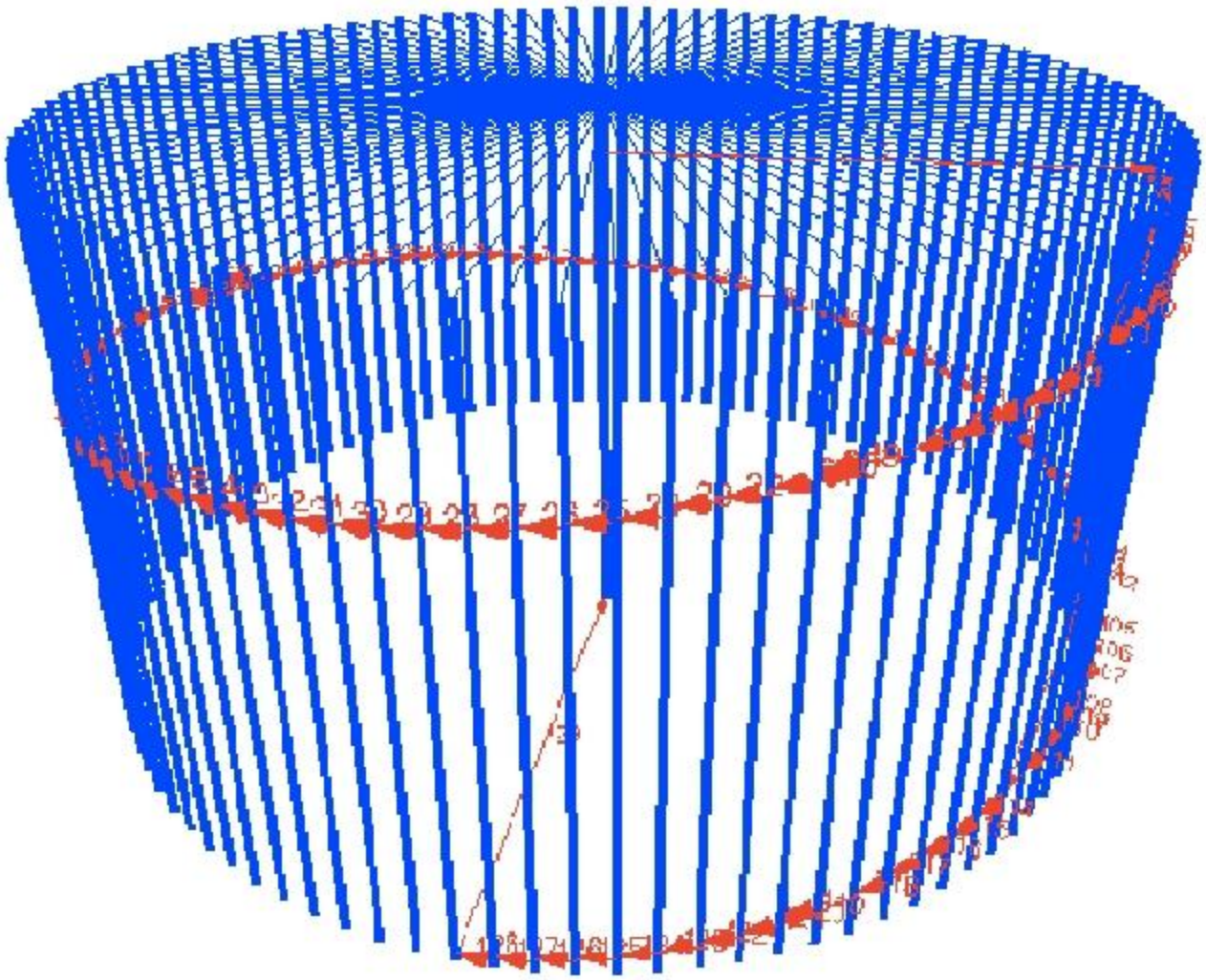
IT大咖说 知识分享平台

```go
package main

import "time"

func main() {
    var Ball int
    table := make(chan int)
    go player(table)
    go player(table)

    table <- Ball
    time.Sleep(1 * time.Second)
    <-table
}

func player(table chan int) {
    for {
        ball := <-table
        ball++
        time.Sleep(100 * time.Millisecond)
        table <- ball
    }
}
```

meitu

Golang—并发

Golang—并发



Golang—gc

美图　长连接消息通道

meitu