

# Distributed Hash Table: Fair and Free

PPCA 2021 大作业 分布式哈希表

---

林超凡

August 4, 2021

ACM Class 2020

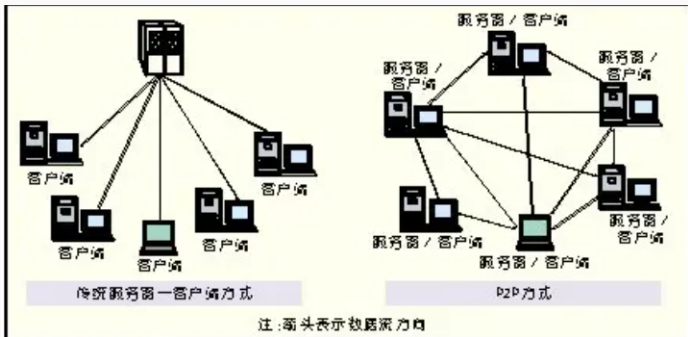


# Introduction

---

# 客户端/服务器与 P2P 网络

- C/S 架构
- 对等网络 (Peer to Peer Networking)



RPC 是什么？ Remote Procedure Call 一个客户端想要调用另一个客户端的方法

```
func (this *ReceiverType) FindValue(args *FindValueArg, reply *FindValueRet) error {  
    *reply = this.Node.FindValue(args.Key, &args.Hash)  
    this.Node.kBucketUpdate(args.Sender)  
    return nil  
}
```

流程 (假设 A 想调用 B 的方法):

- A.Diag(B.ip)
- A.Call(方法名, 参数, 返回地址)...
- B 收到 Call, 执行对应方法

# 可能遇到的问题

也许...

- 某一客户端电源线被踢掉 (ForceQuit)
- 网络通信繁忙
- 恶意的网络攻击

# Chord Protocol

---

# 最朴素的分布式想法

每个客户端采用同样的哈希函数: 一致性哈希 (比如 SHA-1)

每个客户端维护一部分数据——分块

接入网络时候, 原有客户端将一部分数据转移给新节点; 退出网络时, 将自己的数据转移给其它节点.



# 最朴素的分布式想法

A: {data1, data2, data3}

# 最朴素的分布式想法

A: {data1, data2}

B: {data3}

# 最朴素的分布式想法

A: {data1}

B: {data3}

C: {data2}

# 最朴素的分布式想法

B: {data1, data3}

C: {data2}

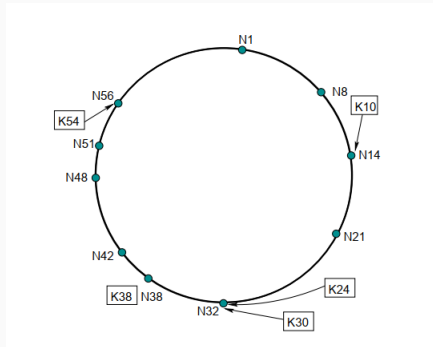
规定一些客户端间共同遵守的规则, 我们称为**协议 (protocol)**

- 哪些数据归谁管?
- 怎样快速地查找数据?
- 节点强制退出怎样做到尽量不丢失数据?

在哈希表中, 我们可以将数据简单地视为 Key-Value 对研究  
同时每个客户端我们下面称作一个节点 (Node)

# Chord 环

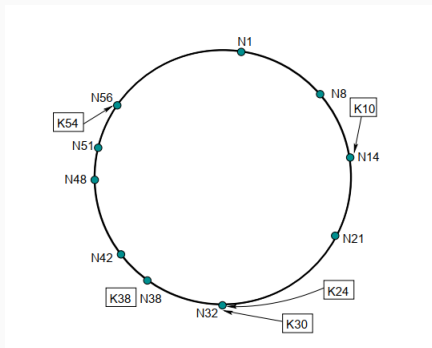
任何哈希函数都有它的值域。对于 SHA-1,  $[0, 2^{160})$   
能否给每个节点也分配一个哈希值?  $\text{hash}=\text{SHA1}(\text{IP})$



这样数据对和节点都有哈希值 (称作 id, identifier)  
节点将环分成好几段, 规定每个节点维护自己上面 (逆时针走) 那段值域的数据

# 高效地查找:Finger Table

每个节点都有唯一的前驱 (predecessor)、后继 (successor)

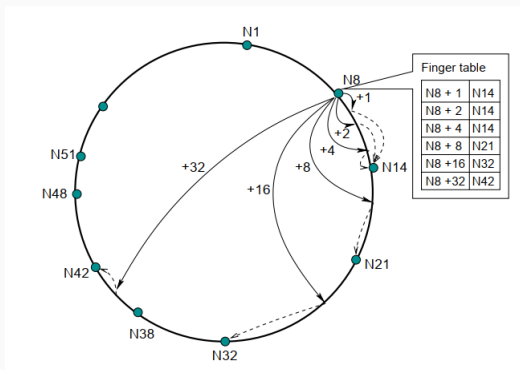


查找: 对于给定的 key 值, 找到其后面第一个节点, 即 `findSuccessor`

# 高效地查找:Finger Table

不止于维护后继: Finger Table

对于每个点 (哈希值为  $id$ ), 维护  $id + 1, id + 2, \dots, id + 2^M$  的后继 Node

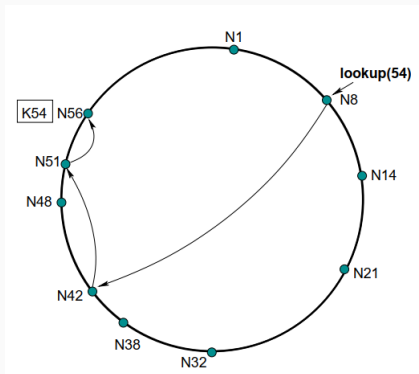




# 高效地查找:Finger Table

$n.findSuccessor(key)$ :  $\log N$

- 如果 key 值在自己与后继之间，返回后继
- 否则，给 finger 表里离 key 值最近的且为其前继的节点发送 findSuccessor RPC
- $n'.findSuccessor$



# Implementation

---

C < go < C++

- 为并发而生 go routine
- 简洁且与 C 高度相似的语法
- 零值初始化以及自动内存回收

虽然...

```
func (this *NodeType) Join(ip string) bool  
{
```

## DHT Interface

```
NewNode(port int) dhtNode
```

```
Run()
```

```
Create()
```

```
Join(addr string) bool
```

```
Quit()
```

```
ForceQuit()
```

```
Ping(addr string) bool
```

```
Put(key string, value string) bool
```

```
Get(key string) (bool, string)
```

```
Delete(key string) bool //Kademlia 不实现此方法
```

PubNode: 实现目标 interface

Receiver: 实现 rpc 服务

Node: 实现 chord 协议中的节点

rpc 设计相关...

- 使用 go 的 net 包, 每个节点开一个 Server 和 Client
- 使用 net 包中的 Dial/Call 实现 rpc
  - \* 为防止因网络繁忙而阻塞, 每次呼叫 3 次
- 将需要远程调用的方法封装为 rpc Methods

```
err = client.Call( serviceMethod: "ReceiverType.FindSuccessor", &this.Addr.Id, &this.succList[0])
client.Close()
```

## Node 设计相关...

- **FixFinger** 单开线程并行维护，每次修复一位，然后将位置加 1 准备对下个位置 Fix
- **FindSuccessor** 按照论文编写，跳 Finger 表，且要求 Ping 的通，若 Finger 表全部失效则访问后继
- **ForceQuit** 每个节点备份自己后继的数据，同时维护后继列表，当维护后继列表过程中发现后继失效，将备份数据合并到后继列表第一个有效节点。



# 遇到的问题

并行中的死锁问题: [github.com/sasha-s/go-deadlock](https://github.com/sasha-s/go-deadlock)

因为维护周期导致的Wrong Answer: 缩短维护周期

环境导致的socket:too many open files等问题: 使用虚拟机, 修改ulimits 和 config

## Log

- Week1: 爽摸
- Week2: chord + tengu 下载、上传部分
- Week3: kademia
- Week4: 调试, tengu music player

## TO DO

- ☒ 基本的 Go 语法
- ☒ OOP 练习: LinkList
- ☒ 读懂测试代码 (道阻且长...)
- ☒ net 入门, 实现简单多 server RPC
- ☒ DHT框架设计 (粗略√)
- ☒ debug.go & utils.go
- ☒ node.go
- ☒ rpc.go
- ☒ ForceQuit
- ☒ BT App
- ☒ Magnet Support & Test
- ☒ Kademia Learning
- ☒ Kademia Implement
- ☒ Music Player

## 一个测试工程师走进一个酒吧...

Environment Setting:

UpdateInterval = 50ms

RemoteTryTime = 3

RemoteTryInterval = 25ms

Final print:

Basic test passed with fail rate 0.0000

Force quit test passed with fail rate 0.0000

Quit & Stabilize test passed with fail rate 0.0000

dht@ubuntu:~/Desktop/DHT-2021\$

## Kademlia & Tengu

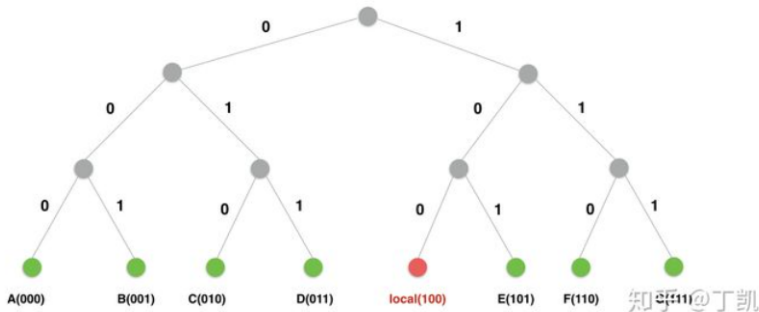
---

Chord 中的距离:  $dis(a, b) = (a - b + 2^{160}) \bmod 2^{160}$

Kademlia 中的距离:  $dis(a, b) = a \otimes b$

# Kademlia Protocol

<https://zhuanlan.zhihu.com/p/38425656>



是一个支持上传/下载的文件分享系统  
也可以用来在 PPCA 摸鱼时听歌

```
Tengu beta 1.0
Type "help" for more information.
help
Tengu is a File Sharing System based on DHT

Tengu Commands:

    upload -fp=<file-path> -sp=<seed-path> -fn=<fileName>           #to upload a file
    download -fp=<file-path> -sp=<seed-path> -sn=<seedFileName> -mg=<magnet> #to download a file
    music-upload -mp=<music-path> -sp=<seed-path> -so=<song-name> -al=<album> #to upload a song
    music-play -al=<album> -so=<song-name> -sp=<seed-path>           #to play a song.
    help                                                         #show help
    quit                                                         #quit from tengu

Tengu Environment Setting:

    Default Torrent Path: torrent/
    Default Upload Path: upload/
    Default Download Path: download/
    Default Music Path: music/

Hints:

    You can omit "-fn" "-so" arguments to upload the whole directory.
```

BitTorrent 协议, 一种基于 TCP/IP 协议的 P2P 文件传输协议

上传者生成一个 种子 (.torrent)

拿到种子便可以为之索引去 下载

```
d8:announce0:4:infod6:lengthi11804287e4:name24:Melty  
_Land_Nightmare.mp312:piece lengthi1048576e6:pieces  
480:b42f29a12ce6ebb8c72cefbb4abf7bed774f936194ac820bd3806739...
```



## bith: BitTorrent Info Hash

```
Tengu beta 1.0
Type "help" for more information.
download -mg=magnet:?xt=urn:bti:h:7f171f3e38f804b13323d34c268f6d4c5f3ec066
Magnet to Torrent Success! saved to: torrent/file.torrent
Torrent Resolved Finish:

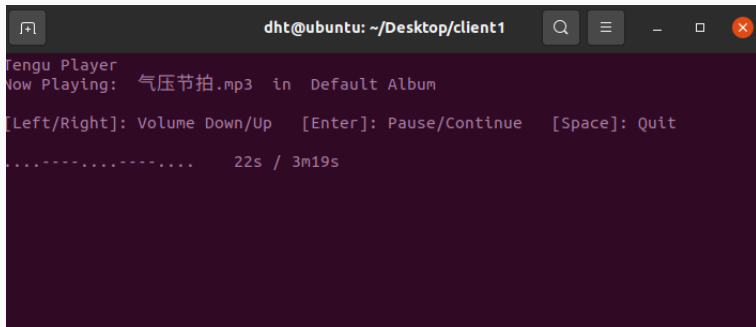
* FileName: PH.pdf Size: 61971632 bytes

Piece #60 Download Finish. (1.67%)
Piece #23 Download Finish. (3.33%)
Piece #24 Download Finish. (5.00%)
Piece #1 Download Finish. (6.67%)
Piece #25 Download Finish. (8.33%)
Piece #2 Download Finish. (10.00%)
Piece #26 Download Finish. (11.67%)
Piece #3 Download Finish. (13.33%)
Piece #27 Download Finish. (15.00%)
Piece #28 Download Finish. (16.67%)
Piece #4 Download Finish. (18.33%)
Piece #5 Download Finish. (20.00%)
Piece #29 Download Finish. (21.67%)
Piece #6 Download Finish. (23.33%)
Piece #30 Download Finish. (25.00%)
Piece #7 Download Finish. (26.67%)
Piece #31 Download Finish. (28.33%)
```

# Tengu Music Player

歌单的存储基于共享文件系统

解析 mp3 格式文件: [github.com/faiface/beep](https://github.com/faiface/beep)



```
dht@ubuntu: ~/Desktop/client1
Tengu Player
Now Playing: 气压节拍.mp3 in Default Album

[Left/Right]: Volume Down/Up  [Enter]: Pause/Continue  [Space]: Quit

.....----- 22s / 3m19s
```

## Reference

---

- [dht.pdf](#) from @xmhuangzhen
- [Golang Net](https://pkg.go.dev/net) (<https://pkg.go.dev/net>)
- [Chord: A Scalable Peer-to-peer Lookup Protocol for Internet Applications](#)
- [Kademlia: A Peer-to-Peer Information System Based on the XOR Metric](#)
- [Building a BitTorrent client from the ground up in Go](#)
- [Kademlia 算法学习](#) (<https://shuwoom.com/?p=813>)

Thank You for Listening!