p2p缓存系统

基于Golang的Aop设计模式

龚浩华
QQ 29185807 月牙寂

背景

- Web缓存（类似CDN技术）
  - 网页、图片
  - 普通下载
  - 普通视频
- P2P缓存
  - 下载（bt等）
  - 视频（qvod、百度影音等）

背景

- P2P缓存好处
  - 一次获取，多次利用
  - 减少局域网出网流量
  - 提升用户体验

背景

- P2P缓存服务器（基于c++开发）
  - 代码量大
  - 协议数量多
  - 耦合性高
  - 潜在bug多

重构 or 重新推倒？

背景

现实世界是怎么样的

分布式、并发

职能化、松散化

自组织、智能化

程序框架是否也可以这样？

OOP

- C++对象代码运行
  - 有属性（成员变量）、有行为（成员函数）
    没有可以独立执行的机制
  - 有属性（成员变量）、有行为（成员函数）
    还有独立执行的活动（有自己独立的线程）
  - 有属性（成员变量）、有行为（成员函数）
    借助其他线程运行

AOP

AOP （agent-oriented programming）

Agent：智能体、职能代理。源于分布式人工智能（DAI）

1、自主的、智能的

2、具有社会性（与环境通信）

3、反应能力，理解环境并对环境刺激做出适应的反应

4、主动性，不是简单的反应，而是有目的的反应

5、一般agent处在分布式网络中，行为具有局部效应和全局效应

golang

1、对象：　类
2、属性：成员变量
3、行为：成员函数
4、自主性、并发：协程
5、职能化、松散耦合：interface，函数变量
6、通信：chan

golang

常规分布式的缺点

1、缺乏全局状态知识

分布式系统中的节点只能访问自身的状态，无法获取到全局的状态

2、缺乏全局时间

分布式系统中的节点无法做到时间的完全一致性，会导致一些行为的顺序不确定

3、非确定

普通程序输入则得到固定的输出。分布式系统则存在很多差异。

golang

基于golang的分布式程序（单进程）

1、针对　缺乏全局状态知识

　　全局状态是可以获取到的

2、针对 缺乏全局时间

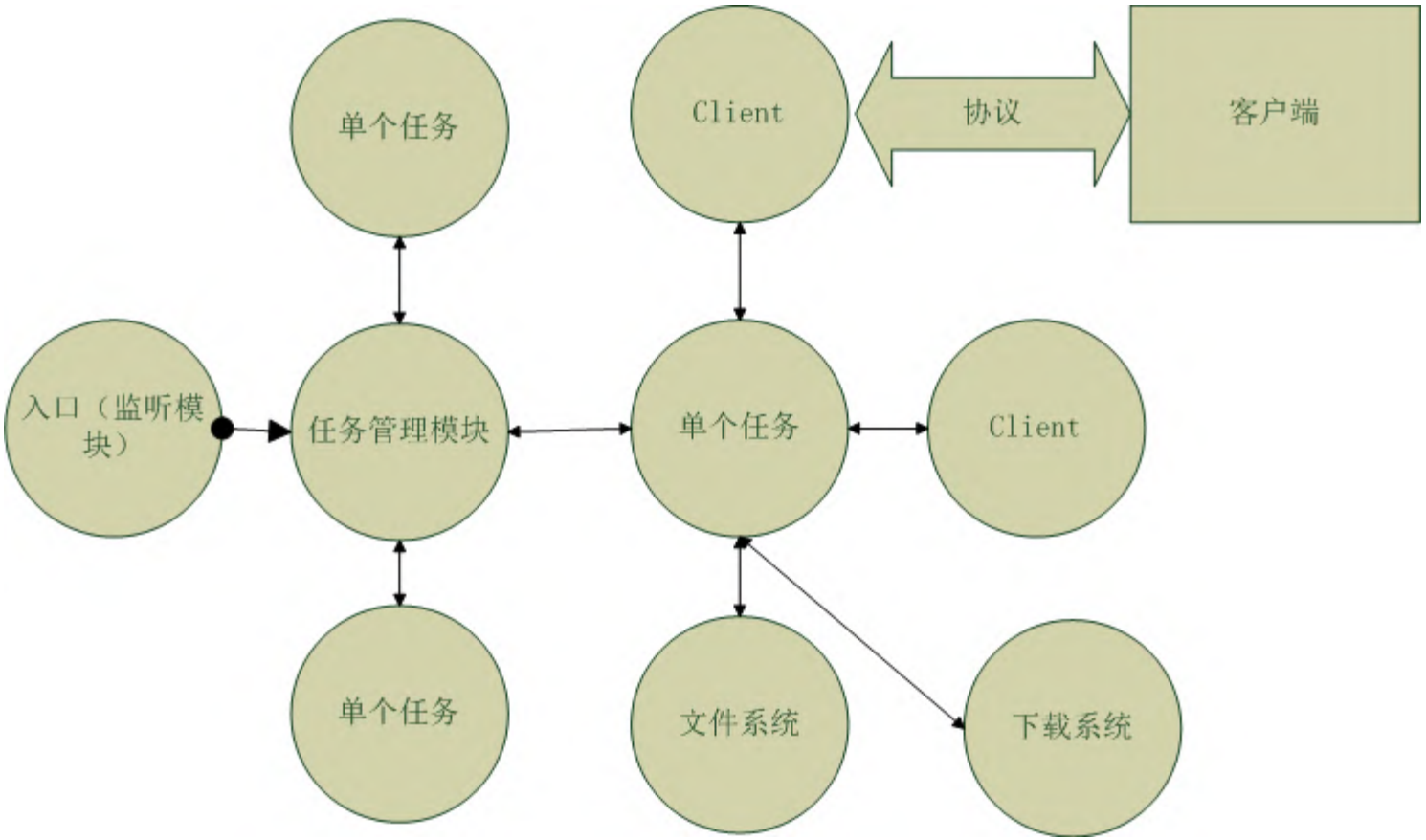
　　全局时间是一致的

3、非确定

　　仍然存在不确定性

现实世界的设计模式直接可以拿来借鉴

P2P缓存框架



P2P缓存框架

1、入口监听模块
常驻
功能监听识别连接

2、任务管理模块
常驻
管理任务、分流client

P2P缓存框架

3、任务模块

文件缓存度：是否需要下载

文件热点程度：是否热点

文件下载状态：sleep、down、限速

client数量、存活

自身存在的必要检查：如超时无client连接

P2P缓存框架

4、文件模块

具体文件的存储，另外再加上内存缓存系统

5、client

对应于与客户端的一个连接，通过协议进行通信。通过任务模块再来进行文件模块的读写，将数据发送给客户端

6、下载模块

如何识别热点？群体智能

群体智能

蝗虫行为

1、跟随前面的蝗虫

2、和周围的蝗虫步伐保持一致

3、与后面的蝗虫保持距离

博得三规则

1、避免（碰撞）：避免碰撞到其他个体

2、定向：按照最接近自己的个体的平均方向前进

3、吸引力（凝聚）：向最接近自己的个体平均位置移动

群体智能

典型群体智能算法有：
1、蚂蚁寻找食物，蚁群算法

2、鸟群寻找食物，粒子群算法

3、鱼群算法

简单规则 + 正反馈、负反馈

群体智能

常规热点算法

1、收集所有的任务列表

2、进行排序

3、同步的问题

群体智能

一个网吧有n个位置，每个人都可以去网吧，但规定，每次只能预约1小时，如果时间到了，则可以续费继续一个小时，如果不需要则放弃位置。

1、m < n时，则m个人都可以获取到上网位置

2、当m>n时，则有n个人可以获取到上网位置，剩余的人，则在等待

群体智能

1、 每个人，都会定期检查自己的时间，是不是到了1个小时，如果是，则判定自己是否需要继续使用，则优先续约1小时，如果不需要则放弃位置。等待的人中，则随机获取到这个位置。

2、每个人，都会定期检查自己的时间，是不是到了1个小时，如果是，则放弃位置。如果还需要继续使用则到等待队列中，不需要使用，则直接退出。

群体热点算法

热点任务优先下载
限制下载任务数量

1、每个任务定时尝试下载（是否有下载空缺）

2、当某个任务有新的连接数连接的时候尝试下载（是否有下载空缺）

3、当任务正在下载的时候，如果下载速度慢，或无下载速度的时候，sleep，让出下载

群体热点算法

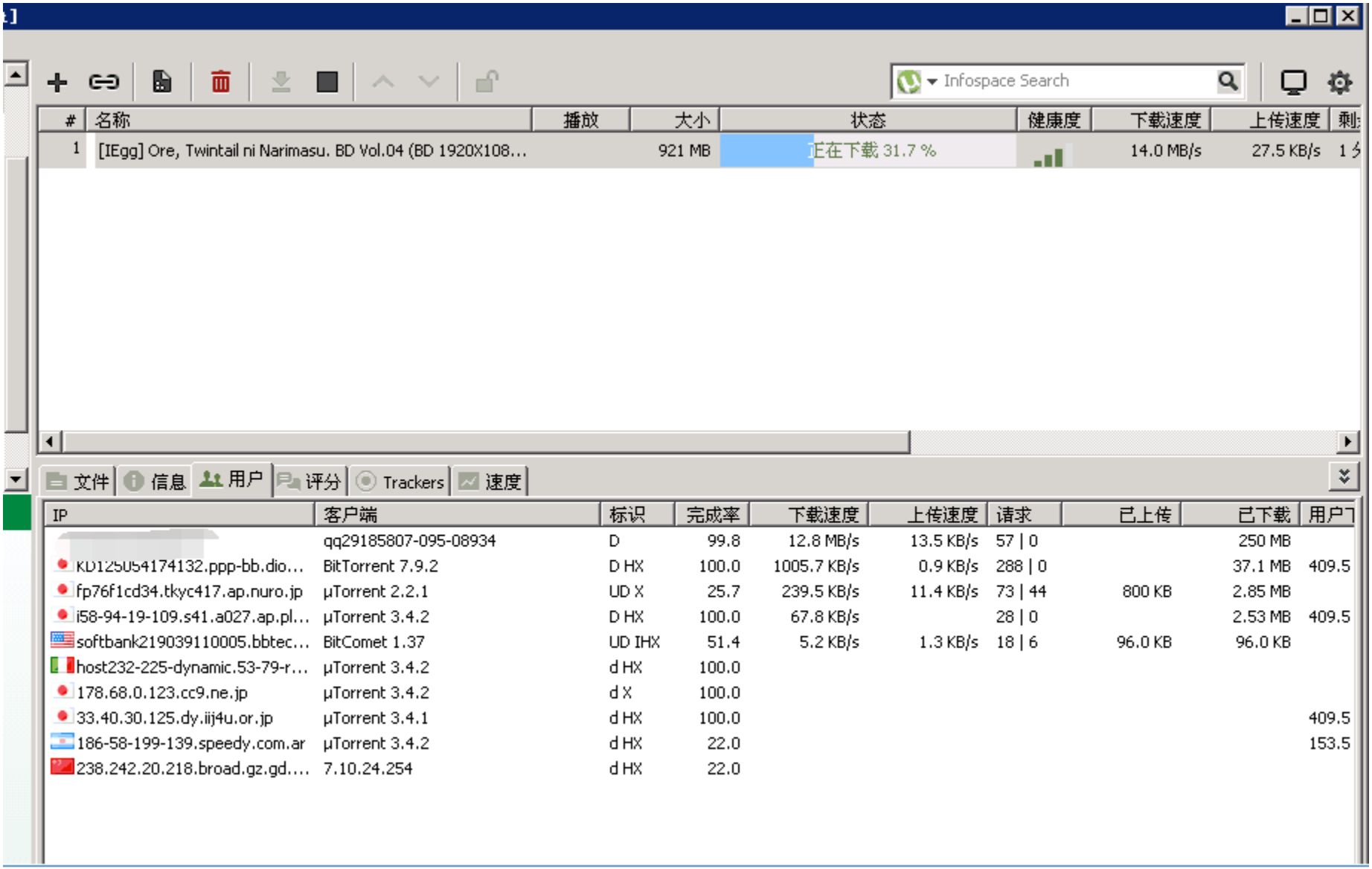
热点任务优先下载

限制下载任务数量

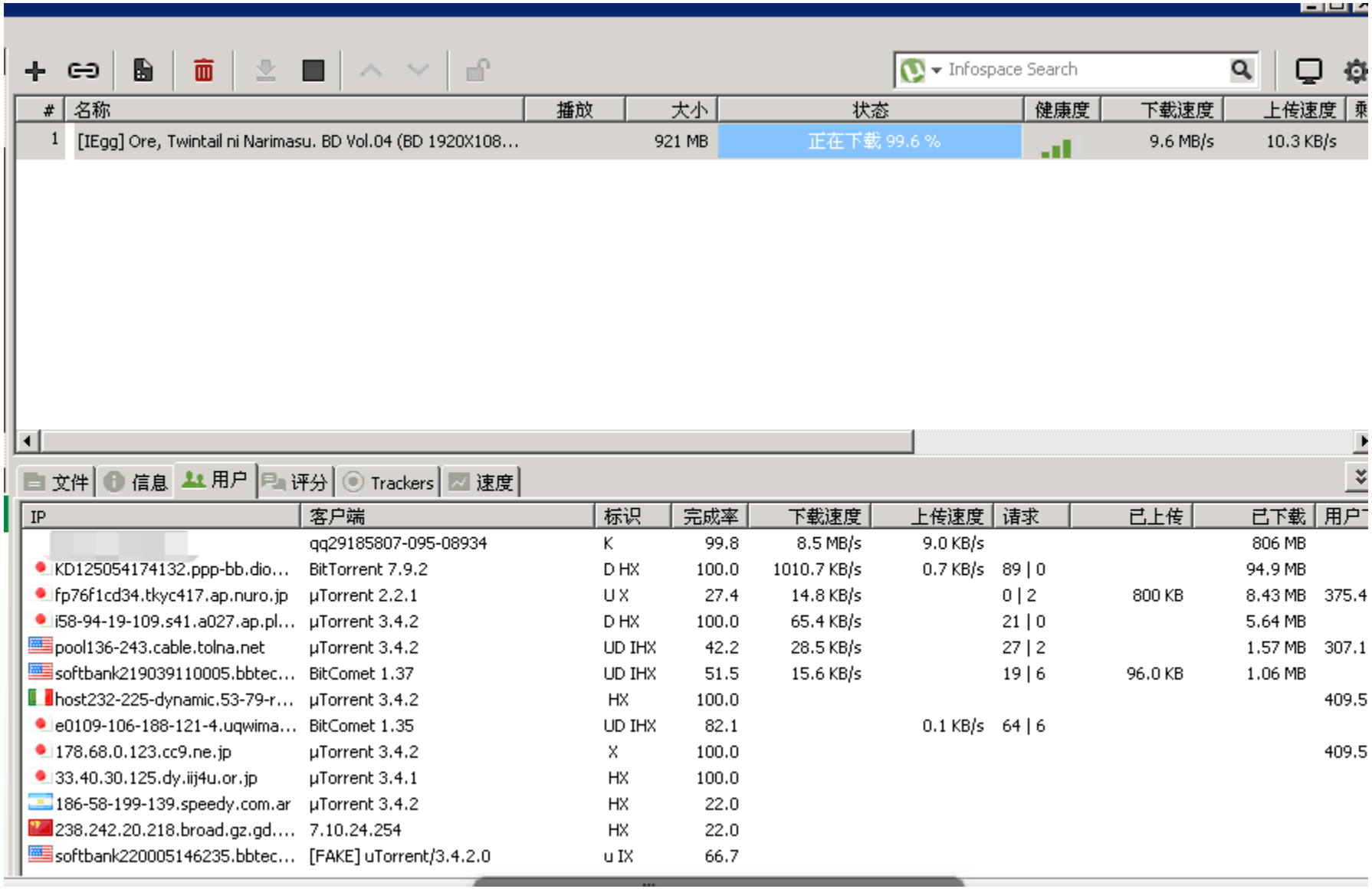
1、全局收集任务，根据任务连接数排名，在前n的任务给分发下载时间片。

2、任务定时更新自己的时间片

效果


Infospace Search
名称 播放 大小 状态 健康度 下载速度 上传速度
1 [IEgg] Ore, Twintail ni Narimasu. BD Vol.04 (BD 1920X108... 921 MB 正在下载 31.7 % 14.0 MB/s 27.5 KB/s
文件 信息 用户 评分 Trackers 速度
IP 客户端 标识 完成率 下载速度 上传速度 请求 已上传 已下载 用户
qq29185807-095-08934 D 99.8 12.8 MB/s 13.5 KB/s 57 | 0 250 MB
KD125054174132.ppp-bb.dio... BitTorrent 7.9.2 D HX 100.0 1005.7 KB/s 0.9 KB/s 288 | 0 37.1 MB 409.5
fp76f1cd34.tkyc417.ap.nuro.jp µTorrent 2.2.1 UD X 25.7 239.5 KB/s 11.4 KB/s 73 | 44 800 KB 2.85 MB
i58-94-19-109.s41.a027.ap.pl... µTorrent 3.4.2 D HX 100.0 67.8 KB/s 28 | 0 2.53 MB 409.5
softbank219039110005.bbtec... BitComet 1.37 UD IHX 51.4 5.2 KB/s 1.3 KB/s 18 | 6 96.0 KB 96.0 KB
host232-225-dynamic.53-79-r... µTorrent 3.4.2 d HX 100.0
178.68.0.123.cc9.ne.jp µTorrent 3.4.2 d X 100.0
33.40.30.125.dy.iij4u.or.jp µTorrent 3.4.1 d HX 100.0 409.5
186-58-199-139.speedy.com.ar µTorrent 3.4.2 d HX 22.0 153.5
238.242.20.218.broad.gz.gd.... 7.10.24.254 d HX 22.0

效果


Infospace Search
名称 播放 大小 状态 健康度 下载速度 上传速度
1 [IEgg] Ore, Twintail ni Narimasu. BD Vol.04 (BD 1920X108... 921 MB 正在下载 99.6 % 9.6 MB/s 10.3 KB/s
文件 信息 用户 评分 Trackers 速度
IP 客户端 标识 完成率 下载速度 上传速度 请求 已上传 已下载 用户
qq29185807-095-08934 K 99.8 8.5 MB/s 9.0 KB/s 806 MB
KD125054174132.ppp-bb.dio... BitTorrent 7.9.2 D HX 100.0 1010.7 KB/s 0.7 KB/s 89 | 0 94.9 MB
fp76f1cd34.tkyc417.ap.nuro.jp µTorrent 2.2.1 U X 27.4 14.8 KB/s 0 | 2 800 KB 8.43 MB 375.4
i58-94-19-109.s41.a027.ap.pl... µTorrent 3.4.2 D HX 100.0 65.4 KB/s 21 | 0 5.64 MB
pool136-243.cable.tolna.net µTorrent 3.4.2 UD IHX 42.2 28.5 KB/s 27 | 2 1.57 MB 307.1
softbank219039110005.bbtec... BitComet 1.37 UD IHX 51.5 15.6 KB/s 19 | 6 96.0 KB 1.06 MB
host232-225-dynamic.53-79-r... µTorrent 3.4.2 HX 100.0 409.5
e0109-106-188-121-4.uqwima... BitComet 1.35 UD IHX 82.1 0.1 KB/s 64 | 6
178.68.0.123.cc9.ne.jp µTorrent 3.4.2 X 100.0 409.5
33.40.30.125.dy.iij4u.or.jp µTorrent 3.4.1 HX 100.0
186-58-199-139.speedy.com.ar µTorrent 3.4.2 HX 22.0
238.242.20.218.broad.gz.gd.... 7.10.24.254 HX 22.0
softbank220005146235.bbtec... [FAKE] uTorrent/3.4.2.0 u IX 66.7

Golang总结

1、全新的设计模式
代码少、逻辑直观简单

2、代码维护简单
松散耦合

3、快速开发

4、性能高

Golang一些经验

1、程序雪崩与GC问题

当对象数量过多的时候，GC扫描所有对象，造成卡顿时间过长。

如果这个时候协程数量过多，还在不断的生成的话。就会造成一个雪崩。

程序卡死，而且很难杀死。

很期待go1.5，解决GC问题

Golang一些经验

2、共享内存与chan通信

在大的模块之间定义接口的时候。

1、可以用函数对象。（接口只有一个的时候）

2、可以通过chan通信

3、可以通过interface定义接口（多个接口）

在模块内，可以考虑用读写锁访问变量或用原子变量

Golang一些经验

3、注意定时器tick

```
tick := time.Tick(1 * time.Second)

for {
    select {
        case <-tick:
            ...
        case <- xxxx:
        ...
        case <- xxx2:
        ....
    }
}
```

```
func sendTime(now int64, c interface{}) {
    // Non-blocking send of time on c.
    // Used in NewTimer, it cannot block anyway (buffer).
    // Used in NewTicker, dropping sends on the floor is
    // the desired behavior when the reader gets behind,
    // because the sends are periodic.
    select {
    case c.(chan Time) <- Now():
    default:
    }
}
```

Golang一些经验

4、注意匿名函数（要传参数）

```
for i := 0; i < 100; i++ {
    go func() {
        fmt.Println(i)
    }()
}
```

```
for i := 0; i < 100; i++ {
    go func(j int) {
        fmt.Println(j)
    }(i)
}
```

资料

《失控》
《人工智能：计算agent基础》
《生物启发计算》
《完美的群体：如何掌控群体智慧的力量》
《生物多智能体自主服务计算及其应用》
《分布式算法导论》

http://www.swarma.org/complex/models/caintro.htm 细胞自动机
http://www.swarma.org/thesis/ 复杂系统、人工智能