

# <iOS开发>之CocoaAsyncSocket使用



o惜乐o (/u/a589d2b37b9c) [+ 关注](#)  
2017.06.07 18:27\* 字数 1832 阅读 1666 评论 27 喜欢 29 阅读 1666 评论 27 喜欢 29  
(/u/a589d2b37b9c)

本文介绍了CocoaAsyncSocket库中GCDAsyncSocket类的使用、粘包处理以及时间延迟测试。

## 一.CocoaAsyncSocket介绍

CocoaAsyncSocket中主要包含两个类:

1.GCDAsyncSocket.

用GCD搭建的基于TCP/IP协议的socket网络库  
GCDAsyncSocket is a TCP/IP socket networking library built atop Grand Central Dispatch. -- 引自CocoaAsyncSocket.

2.GCDAsyncUdpSocket.

用GCD搭建的基于UDP/IP协议的socket网络库.  
GCDAsyncUdpSocket is a UDP/IP socket networking library built atop Grand Central Dispatch...-- 引自CocoaAsyncSocket.

## 二.下载CocoaAsyncSocket

- 首先,需要到这里 (<https://link.jianshu.com?t=https://github.com/robbiehanson/CocoaAsyncSocket>)下载CocoaAsyncSocket.
- 下载后可以看到文件所在位置.



文件路径

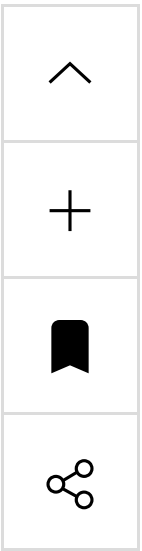
- 这里只要拷贝以下两个文件到项目中.



TCP开发使用的文件

## 三.客户端

因为,大部分项目已经有服务端socket,所以,先讲解客户端创建过程.



步骤:

1.继承 GCDAsyncSocketDelegate 协议.

2.声明属性

```
// 客户端socket
@property (strong, nonatomic) GCDAsyncSocket *clientSocket;
```

3.创建socket并指定代理对象为self,代理队列必须为主队列.

```
self.clientSocket = [[GCDAsyncSocket alloc] initWithDelegate:self delegateQueue:dispatch_get_main_queue()];
```

4.连接指定主机的对应端口.

```
NSError *error = nil;
self.connected = [self.clientSocket connectToHost:self.addressTF.text onPort:[self.portTF.text integerValue] viaInterface:nil withTimeout:-1 error:&error];
```

5.成功连接主机对应端口号.

```
- (void)socket:(GCDAsyncSocket *)sock didConnectToHost:(NSString *)host port:(uint16_t)port
{
    //    NSLog(@"连接主机对应端口%@", sock);
    [self showMessageWithStr:@"链接成功"];
    [self showMessageWithStr:[NSString stringWithFormat:@"服务器IP: %@-----端口: %d", host,port]];

    // 连接成功开启定时器
    [self addTimer];
    // 连接后,可读取服务端的数据
    [self.clientSocket readDataWithTimeout:- 1 tag:0];
    self.connected = YES;
}
```

注意:

\*The host parameter will be an IP address, not a DNS name. -- \*引自  
GCDAsyncSocket  
连接的主机为IP地址,并非DNS名称.

6.发送数据给服务端

```
// 发送数据
- (IBAction)sendMessageAction:(id)sender
{
    NSData *data = [self.messageTextField.text dataUsingEncoding:NSUTF8StringEncoding];

    // withTimeout -1 : 无穷大,一直等
    // tag : 消息标记
    [self.clientSocket writeData:data withTimeout:- 1 tag:0];
}
```

注意:

发送数据主要通过 - (void)writeData:(NSData \*)data withTimeout:(NSTimeInterval)timeout tag:(long)tag 写入数据的.

7.读取服务端数据

```
/**
  读取数据

  @param sock 客户端socket
  @param data 读取到的数据
  @param tag 本次读取的标记
  */
- (void)socket:(GCDAsyncSocket *)sock didReadData:(NSData *)data withTag:(long)tag
{
    NSString *text = [[NSString alloc] initWithData:data encoding:NSUTF8StringEncoding];
    [self showMessageWithStr:text];
    // 读取到服务端数据值后,能再次读取
    [self.clientSocket readDataWithTimeout:- 1 tag:0];
}
```

注意:

有的人写好代码,而且第一次能够读取到数据,之后,再也接收不到数据.那是因为,在读取到数据的代理方法中,需要再次调用 [self.clientSocket readDataWithTimeout:- 1 tag:0]; 方法,框架本身就是这么设计的.

8.客户端socket断开连接.

```
/**
  客户端socket断开

  @param sock 客户端socket
  @param err 错误描述
  */
- (void)socketDidDisconnect:(GCDAsyncSocket *)sock withError:(NSError *)err
{
    [self showMessageWithStr:@"断开连接"];
    self.clientSocket.delegate = nil;
    self.clientSocket = nil;
    self.connected = NO;
    [self.connectTimer invalidate];
}
```

注意:

socect断开连接时,需要清空代理和客户端本身的socket.

```
self.clientSocket.delegate = nil;
self.clientSocket = nil;
```

9.建立心跳连接.

```
// 计时器
@property (nonatomic, strong) NSTimer *connectTimer;

// 添加定时器
- (void)addTimer
{
    // 长连接定时器
    self.connectTimer = [NSTimer scheduledTimerWithTimeInterval:5.0 target:self selector:@selector(longConnectToSocket) userInfo:nil repeats:YES];
    // 把定时器添加到当前运行循环,并且调为通用模式
    [[NSRunLoop currentRunLoop] addTimer:self.connectTimer forMode:NSRunLoopCommonModes];
}

// 心跳连接
- (void)longConnectToSocket
{
    // 发送固定格式的数据,指令@"longConnect"
    float version = [[UIDevice currentDevice] systemVersion].floatValue;
    NSString *longConnect = [NSString stringWithFormat:@"123%f",version];

    NSData *data = [longConnect dataUsingEncoding:NSUTF8StringEncoding];

    [self.clientSocket writeData:data withTimeout:- 1 tag:0];
}
```

注意:  
心跳连接中发送给服务端的数据只是作为测试代码,根据你们公司需求,或者和后台商定好心跳包的数据以及发送心跳的时间间隔.因为这个项目的服务端socket也是我写的,所以,我自定义心跳包协议.客户端发送心跳包,服务端也需要有对应的心跳检测,以此检测客户端是否在线.

## 四.服务端

### 步骤:

1.继承 GCDAsyncSocketDelegate 协议.

2.声明属性

```
// 服务端socket(开放端口,监听客户端socket的连接)
@property (strong, nonatomic) GCDAsyncSocket *serverSocket;
```

3.创建socket并指定代理对象为self,代理队列必须为主队列.

```
// 初始化服务端socket
self.serverSocket = [[GCDAsyncSocket alloc] initWithDelegate:self delegateQueue:dispatch_get_main_queue()];
```

4.开放服务端的指定端口.

```
BOOL result = [self.serverSocket acceptOnPort:[self.portF.text integerValue] error:&error];
```

5.连接上新的客户端socket

```
// 连接上新的客户端socket
- (void)socket:(GCDAsyncSocket *)sock didAcceptNewSocket:(nonnull GCDAsyncSocket *)newSocket
{
    // 保存客户端的socket
    [self.clientSockets addObject:newSocket];
    // 添加定时器
    [self addTimer];

    [self showMessageWithStr:@"链接成功"];
    [self showMessageWithStr:[NSString stringWithFormat:@"客户端的地址: %@ ----- 端口: %d", newSocket.connectedHost, newSocket.connectedPort]];

    [newSocket readDataWithTimeout:- 1 tag:0];
}
```

6.发送数据给客户端

```
// socket是保存的客户端socket, 表示给这个客户端socket发送消息
- (IBAction)sendMessage:(id)sender
{
    if(self.clientSockets == nil) return;
    NSData *data = [self.messageTextField.text dataUsingEncoding:NSUTF8StringEncoding];
    // withTimeout -1 : 无穷大,一直等
    // tag : 消息标记
    [self.clientSockets enumerateObjectsUsingBlock:^(id _Nonnull obj, NSUInteger idx, BOOL * _Nonnull stop) {
        [obj writeData:data withTimeout:-1 tag:0];
    }];
}
```

7.读取客户端的数据

```

/**
 读取客户端发送的数据

@param sock 客户端的Socket
@param data 客户端发送的数据
@param tag 当前读取的标记
*/
- (void)socket:(GCDAsyncSocket *)sock didReadData:(NSData *)data withTag:(long)tag
{
    NSString *text = [[NSString alloc] initWithData:data encoding:NSUTF8StringEncoding];
    [self showMessageWithStr:text];

    // 第一次读取到的数据直接添加
    if (self.clientPhoneTimeDicts.count == 0)
    {
        [self.clientPhoneTimeDicts setObject:[self getCurrentTime] forKey:text];
    }
    else
    {
        // 键相同,直接覆盖,值改变
        [self.clientPhoneTimeDicts enumerateKeysAndObjectsUsingBlock:^(id _Nonnull key, id _Nonnull obj, BOOL * _Nonnull stop) {
            [self.clientPhoneTimeDicts setObject:[self getCurrentTime] forKey:text];
        }];
    }

    [sock readDataWithTimeout:- 1 tag:0];
}

```

### 8.建立检测心跳连接.

```

// 检测心跳计时器
@property (nonatomic, strong) NSTimer *checkTimer;

// 添加计时器
- (void)addTimer
{
    // 长连接定时器
    self.checkTimer = [NSTimer scheduledTimerWithTimeInterval:10.0 target:self selector:@selector(checkLongConnect) userInfo:nil repeats:YES];
    // 把定时器添加到当前运行循环,并且调为通用模式
    [[NSRunLoop currentRunLoop] addTimer:self.checkTimer forMode:NSRunLoopCommonModes];
}

// 检测心跳
- (void)checkLongConnect
{
    [self.clientPhoneTimeDicts enumerateKeysAndObjectsUsingBlock:^(id _Nonnull key, id _Nonnull obj, BOOL * _Nonnull stop) {
        // 获取当前时间
        NSString *currentTimeStr = [self getCurrentTime];
        // 延迟超过10秒判断断开
        if ([currentTimeStr doubleValue] - [obj doubleValue] > 10.0)
        {
            [self showMessageWithStr:[NSString stringWithFormat:@"%@"已经断开,连接时差%f",key,[currentTimeStr doubleValue] - [obj doubleValue]]];
            [self showMessageWithStr:[NSString stringWithFormat:@"移除%@",key]];
            [self.clientPhoneTimeDicts removeObjectForKey:key];
        }
        else
        {
            [self showMessageWithStr:[NSString stringWithFormat:@"%@"处于连接状态,连接时差%f",key,[currentTimeStr doubleValue] - [obj doubleValue]]];
        }
    }];
}

```

心跳检测方法只提供部分思路:

- 1.懒加载一个 可变字典 ,字典的 键 作为 客户端的标识 .如: 客户端标识 为 13123456789 .
- 2.在 - (void)socket:(GCDAsyncSocket \*)sock didReadData:(NSData \*)data withTag:(long)tag 方法中,将 读取到的数据 或者 数据中的部分字符串 作为键.字典的 值 为 系统当前时间 .服务端第一次读取数据时,字典中没有数据,所以,直接添加到可变字典中,之后每次读取数据时,

都用字典的 setObject: forKey: 方法添加字典,若 存储的键相同 ,即 客户端标识相同 ,键会被覆盖,再使用 系统的当前时间 作为值.

3.在 - (void)checkLongConnect 中,获取此时的 当前时间 ,遍历字典,将每个键的值和当前时间进行比较即可.判断的延迟时间可以写8秒.时间自定.之后,再根据自己的需求进行后续处理.

## 五.数据粘包处理.

1.粘包情况.

例如:包数据为: abcd .

| 接收类型 | 第1次接收  | 第2次接收    | 第3次接收        |
|------|--------|----------|--------------|
| 完整型  | abcd   | abcdabcd | abcdabcdabcd |
| 多余型  | abcdab | cdabcdab | cdabcdabcdab |
| 不完整型 | ab     | cda      | bcdabc       |

2.粘包解决思路.

- 思路1:  
发送方将数据包加上 包头 和 包尾 ,包头、包体以及包尾 用 字典 形式包装成 json字符串 ,接收方,通过解析获取 json字符串 中的包体,便可进行进一步处理.  
例如:

```
{
// head:包头,body:包体,end:包尾
NSMutableDictionary *dict = @{
    @"head" : @"phoneNum",
    @"body" : @(13133334444),
    @"end" : @(11)};
}
```

- 思路2:  
添加前缀.和包内容拼接成同一个字符串.

例如:当发送数据是 13133334444 ,如果出现粘包情况只属于 完整型 :

13133334444  
131333344441313334444  
1313333444413133344441313334444 ...

可以将 ab 作为前缀.则接收到的数据出现的粘包情况:

ab13133334444  
ab13133334444ab13133334444  
ab13133334444ab13133334444ab13133334444 ...

使用 componentsSeparatedByString: 方法,以ab为分隔符,将每个包内容存入数组中,再取对应数组中的数据操作即可.

- 思路3:  
如果最终要得到的数据的长度是个 固定长度 ,用一个 字符串 作为 缓冲池 ,每次收到数据,都用 字符串 拼接对应数据,每当 字符串的长度 和 固定长度 相同时,便得到一个 完整数据 ,处理完这个数据并 清空字符串 ,再进行下一轮的 字符拼接 .

例如:处理上面的 不完整型 .创建一个 长度是4 的 tempData 字符串 作为 数据缓冲池. 第1次 收到数据,数据是: ab , tempData 拼接上 ab , tempData 中只能再存储2个字符, 第2次 收到数据,将 数据长度 和 2 进行比较,第2次的数据是: cda ,截取前两位字符,即 cd , tempData 继续

拼接 cd ,此时, tempData 为 abcd ,就是我们想要的数据,我们可以处理这个数据,处理之后并清空 tempData ,将第2次 收到数据 的 剩余数据 ,即 cda 中的 a ,再与 tempData 拼接.之后,再进行类似操作.

- 核心代码

```
/**
 处理数据粘包

@param readData 读取到的数据
*/
- (void)dealStickPackageWithData:(NSString *)readData
{
    // 缓冲池还需要存储的数据个数
    NSInteger tempCount;

    if (readData.length > 0)
    {
        // 还差tempLength个数填满缓冲池
        tempCount = 4 - self.tempData.length;
        if (readData.length <= tempCount)
        {
            self.tempData = [self.tempData stringByAppendingString:readData];

            if (self.tempData.length == 4)
            {
                [self.mutArr addObject:self.tempData];
                self.tempData = @"";
            }
        }
        else
        {
            // 下一次的数据个数比要填满缓冲池的数据个数多,一定能拼接成完整数据,剩余的继续
            self.tempData = [self.tempData stringByAppendingString:[readData substringToIndex:tempCount]];
            [self.mutArr addObject:self.tempData];
            self.tempData = @"";

            // 多余的再执行一次方法
            [self dealStickPackageWithData:[readData substringFromIndex:tempCount
            ]];
        }
    }
}
```

- 调用

```
// 存储处理后的每次返回数据
@property (nonatomic, strong) NSMutableArray *mutArr;
// 数据缓冲池
@property (nonatomic, copy) NSString *tempData;

/** 第四次测试 -- 混合型**/
self.mutArr = nil;
/*
第1次 : abc
第2次 : da
第3次 : bcdabcd
第4次 : abcdabcd
第5次 : abcdabcdab
*/
// 数组中的数据代表每次接收的数据
NSArray *testArr4 = [NSArray arrayWithObjects:@"abc",@"da",@"bcdabcd",@"abcdabcd",@"abcdabcdab", nil];
self.tempData = @"";
for (NSInteger i = 0; i < testArr4.count; i++)
{
    [self dealStickPackageWithData:testArr4[i]];
}
NSLog(@"testArr4 = %@",self.mutArr);
```

- 结果:

```
2017-06-09 00:49:12.932976+0800 StickPackageDealDemo[10063:3430118] testArr4 = (  
    abcd,  
    abcd,  
    abcd,  
    abcd,  
    abcd,  
    abcd,  
    abcd,  
    abcd  
)
```

数据粘包处理Demo在文末.

## 六.测试.

### 1.测试配置.

测试时,两端需要处于同一WiFi下.客户端中的IP地址为服务端的IP地址,具体信息进入Wifi设置中查看.



IP和端口描述

### 2.测试所需环境.

将客户端程序安装在每个客户端,让一台服务端测试机和一台客户端测试机连接mac并运行,这两台测试机可以看到打印结果,所有由服务端发送到客户端的数据,通过客户端再回传给服务端,在服务端看打印结果.



当年的图



3.进行延迟差测试.  
延迟差 即服务端发送数据到 第一台客户端 和服务端发送数据到 最后一台客户端 的时间差.根据服务端发送数据给不同数量的客户端进行测试.而且,发送数据时,是随机发送.

延迟差测试结果:



延迟差测试

由图所知,延迟差在200毫秒以内的比例基本保持在99%以上.所以符合开发需求(延迟在200毫秒以内).

4.单次信息收发测试.  
让服务端给每个客户端随机发送 200 次数据.并计算服务端发送数据到某一客户端,完整的一次收发时间 情况.

单次信息收发测试结果:



单次信息收发测试

由图所知,一次收发时间基本在95%以上,这个时间会根据网络状态和数据包大小波动.不过,可以直观看到数据从服务端到客户端的时间.

## CSDN

iOS开发之CocoaAsyncSocket使用 ([https://link.jianshu.com?t=http://blog.csdn.net/cherish\\_joy/article/details/73790382](https://link.jianshu.com?t=http://blog.csdn.net/cherish_joy/article/details/73790382))

## 个人博客

iOS开发之CocoaAsyncSocket使用 (<https://link.jianshu.com?t=https://cherishjoyby.github.io/2017/06/07/iOS%E5%BC%80%E5%8F%91%E4%B9%8BCocoaAsyncSocket%E4%BD%BF%E7%94%A8/>)

## GitHub

数据粘包处理Demo (<https://link.jianshu.com?t=https://github.com/CherishJoyBy/BYStickPackageDealDemo>)  
CocoaAsyncSocket客户端Demo (<https://link.jianshu.com?t=https://github.com/CherishJoyBy/BYSocketClientDemo>)  
CocoaAsyncSocket服务端Demo (<https://link.jianshu.com?t=https://github.com/CherishJoyBy/BYSocketServerDemo>)  
CocoaAsyncSocket客户端Demo(含粘包解决和测试) (<https://link.jianshu.com?t=https://github.com/CherishJoyBy/BYSocketClientTestDemo>)  
CocoaAsyncSocket服务端Demo(含粘包解决和测试) (<https://link.jianshu.com?t=https://github.com/CherishJoyBy/BYSocketServerTestDemo>)



o惜乐o (/u/a589d2b37b9c)

写了 23713 字，被 55 人关注，获得了 105 个喜欢

(/u/a589d2b37b9c)写了 23713 字，被 55 人关注，获得了 105 个喜欢

+ 关注

路漫漫其修远兮,吾将上下而求索. GitHub:<https://github.com/CherishJoyBy> 个人博客:<https://cherishJoy...>

喜欢 | 29








更多分享

(<http://cwb.assets.jianshu.io/notes/images/1282443>




写下你的评论...

27条评论

只看作者

[按喜欢排序](#)   [按时间正序](#)   [按时间倒序](#)



十一岁的加重 (/u/1856923ecb16)


2楼 · 2017.06.16 15:23

(/u/1856923ecb16)

大爱，收藏

赞     回复


o惜乐o (/u/a589d2b37b9c):    @十一岁的加重 (/users/1856923ecb16) 😊


2017.06.16 15:31     回复


十一岁的加重 (/u/1856923ecb16):    @o惜乐o (/users/a589d2b37b9c) 秒回

2017.06.16 15:32     回复

o惜乐o (/u/a589d2b37b9c):    @十一岁的加重 (/users/1856923ecb16) 给力

2017.06.16 15:34     回复

 添加新评论



BlackRainism (/u/75ddb37cd80c)


3楼 · 2017.07.16 20:25

(/u/75ddb37cd80c)


下载了demo，运行，IP和端口设置感觉都没问题，为什么俩边都收不到消息呐。。。有什么需要注意的？

赞     回复

o惜乐o (/u/a589d2b37b9c):    @BlackRainism (/users/75ddb37cd80c) 如果IP和端口没问题，SB中的端口也要指定，还有客户端指定要连接的服务器端IP,正常是不会有问题的,你再检查检查。

2017.07.16 20:45     回复

Cc\_87ac (/u/f245fa7e0d92):    我这边ip地址和端口设置也没问题 但是我发送了一个消息服务端接收不到

2017.08.17 10:05     回复

o惜乐o (/u/a589d2b37b9c): @Cc\_87ac (/users/f245fa7e0d92) 确定两个端都在同一网络下吗

2017.08.18 11:51 回复

添加新评论 | 还有2条评论, 展开查看



云飞君 (/u/892ca17b9a9c)

4楼 · 2017.09.18 15:56

(/u/892ca17b9a9c)  
调通啦~非常感谢~~

赞 回复

o惜乐o (/u/a589d2b37b9c): @云飞君 (/users/892ca17b9a9c) 能帮上忙就好

2017.10.26 10:56 回复

jack\_ (/u/2e59341d7af0): @云飞君 (/users/892ca17b9a9c) 你好, 可以请教你怎么调通的吗? 打扰了

2017.11.02 18:27 回复

添加新评论



miku酱啦 (/u/dcf72dbfbf4e)

5楼 · 2017.11.01 16:37

(/u/dcf72dbfbf4e)  
感谢博主分享,但是为GCDAsyncSocket对象设置队列那里,不是主队列也是可以的,如想弹幕这样的业务频繁收包的话 会卡顿UI的~

赞 回复

o惜乐o (/u/a589d2b37b9c): @miku酱啦 (/users/dcf72dbfbf4e) 指定为主队列, 是因为当时我查看该方法的实现部分, 发现在内部已经开启子线处理事务, 我也测试过提供一个并发队列给它, 但是出现警告表示让不要那么做, 具体警告忘记了, 可能对应的sdk包版本也不同了, 很久没做这块, 记不清了, 你可以仔细看下方法内部实现, 希望能帮到你

2017.11.01 18:01 回复

jack\_ (/u/2e59341d7af0): 你好, 请问你可以调通吗? 想请教你, 可以吗?

2017.11.02 19:33 回复

miku酱啦 (/u/dcf72dbfbf4e): @jack\_ (/users/2e59341d7af0) 我没有自己配服务端...我是原来仿制斗鱼时用斗鱼的弹幕服务器做的这块,你可以上斗鱼的开放平台上找,用它的服务器调.它的服务端肯定没问题

2017.11.03 08:50 回复

添加新评论



jack\_ (/u/2e59341d7af0)

6楼 · 2017.11.02 17:32

(/u/2e59341d7af0)  
你好, 这边IP和端口设置正确, 却一直连不通?

赞 回复

jack\_ (/u/2e59341d7af0): 方便请教一下吗? 谢谢!

2017.11.02 17:34 回复

o惜乐o (/u/a589d2b37b9c): @jack\_ (/users/2e59341d7af0) 有什么提示吗

2017.11.02 17:46 回复

jack\_ (/u/2e59341d7af0): @o惜乐o (/users/a589d2b37b9c) Error Domain=NSPOSIXErrorDomain Code=0 "Undefined error: 0" UserInfo={NSLocalizedDescription=Undefined error: 0, NSLocalizedFailureReason=Error in connect() function}








2017.11.02 17:57 回复

添加新评论

还有6条评论， 展开查看

被以下专题收入，发现更多相似内容

+ 收入我的专题

-  程序员 (/c/NEt52a?utm\_source=desktop&utm\_medium=notes-included-collection)
-  iOS Dev... (/c/3233d1a249ca?utm\_source=desktop&utm\_medium=notes-included-collection)
-  iOS Dev... (/c/ee25d429d275?utm\_source=desktop&utm\_medium=notes-included-collection)
-  iOS笔记 (/c/995f94182f16?utm\_source=desktop&utm\_medium=notes-included-collection)
-  IM & Ne... (/c/9a50ae9a4568?utm\_source=desktop&utm\_medium=notes-included-collection)
-  Socket通信 (/c/3364638a1bea?utm\_source=desktop&utm\_medium=notes-included-collection)
-  iOS开发 (/c/da553370c834?utm\_source=desktop&utm\_medium=notes-included-collection)

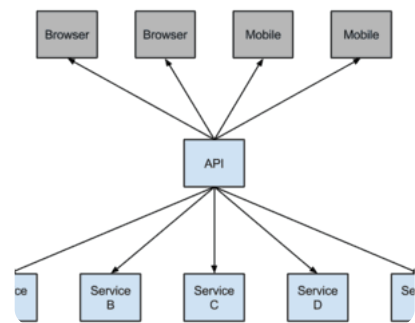
展开更多

## CocoaAsyncSocket使用 (/p/03c92cfe9d33?utm\_campaign=maleskin...

转载:http://www.cocoachina.com/ios/20170615/19529.html 参考:http://www.jb51.net/article/83941.htm

x董色 (/u/c71ea24e5e2d?utm\_campaign=maleskine&utm\_content=user&utm\_medium=seo\_notes&utm\_source=recommendation)

(/p/46fd0faecac1?



utm\_campaign=maleskine&utm\_content=note&utm\_medium=seo\_notes&utm\_source=recommendation)  
**Spring Cloud (/p/46fd0faecac1?utm\_campaign=maleskine&utm\_cont...**

Spring Cloud为开发人员提供了快速构建分布式系统中一些常见模式的工具（例如配置管理，服务发现，断路器，智能路由，微代理，控制总线）。分布式系统的协调导致了样板模式,使用Spring Cloud开发人员可以

卡卡罗2017 (/u/d90908cb0d85?utm\_campaign=maleskine&utm\_content=user&utm\_medium=seo\_notes&utm\_source=recommendation)

## iOS高级开发面试题精选总结 (/p/1c728702a7e0?utm\_campaign=malesk...

1、OC中创建线程的方法是什么？如果指定在主线程中执行代码？如何延时执行代码。【难度系数★★】 1) 创建线程的方法 NSThread NSOperationQueue和NSOperation GCD 2) 主线程中执行代码 [self

木旁\_G\_ShareT (/u/647cf5501f60?utm\_campaign=maleskine&utm\_content=user&utm\_medium=seo\_notes&utm\_source=recommendation)

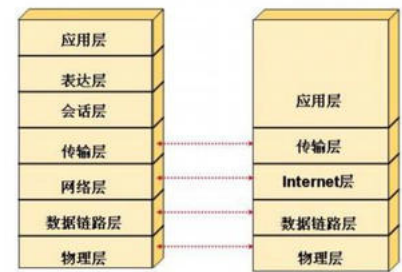


iOS开发网络篇—Socket编程 (/p/e5dbe603a676?utm\_campaign=males...

转自http://www.mamicode.com/info-detail-877996.html 一、网络各个协议：TCP/IP、SOCKET、HTTP等 网络七层由下往上分别为物理层、数据链路层、网络层、传输层、会话层、表示层和应用层。其中物理

 啾啾以啾啾 (/u/e33ec901288c?utm\_campaign=maleskine&utm\_content=user&utm\_medium=seo\_notes&utm\_source=recommendation)

(/p/f78491e27c74?



utm\_campaign=maleskine&utm\_content=note&utm\_medium=seo\_notes&utm\_source=recommendation)

网络编程: TCP/IP、HTTP/HTTPS、Socket和CocoaAsyncSocket使用 (转...

网络七层媒体层: 网络工程师所研究的对象 主机层: 用户所面向和关心的内容传输层 【TCP \ UDP】会话层表示层应用层 【HTTP \ FTP \ SMTP \ DNS】 拓展: 通信的基石套接字【Socket】 socket: 网络通信过程中端

 情书和海 (/u/8e614a37f109?utm\_campaign=maleskine&utm\_content=user&utm\_medium=seo\_notes&utm\_source=recommendation)

沈从文的理想国 (/p/e028a45e0354?utm\_campaign=maleskine&utm\_c...

“我行过许多地方的桥，看过许多次的云，喝过许多种类的酒，却只爱过一个正当最好年龄的人。”很多人因这句情话知道沈从文，因《边城》认识沈从文，我也未出其二。未读《边城》之前，我未曾在别人的作品中看

 木槿初尘 (/u/73d8e136a144?utm\_campaign=maleskine&utm\_content=user&utm\_medium=seo\_notes&utm\_source=recommendation)

(/p/17ee3c26250a?



utm\_campaign=maleskine&utm\_content=note&utm\_medium=seo\_notes&utm\_source=recommendation)

着我汉家衣裳，兴我礼仪之邦！ (/p/17ee3c26250a?utm\_campaign=mal...

热爱汉服的人越来越多，十月底至十一月初的西塘汉服文化周去了很多汉服爱好者，看到了很多汉服小姐姐和汉服小哥哥，小编我也去玩了两天，风景很美，汉服很美，人更美。放几张返图给大家瞅瞅。 P1是来自

 P22121728 (/u/3b9b3259b963?utm\_campaign=maleskine&utm\_content=user&utm\_medium=seo\_notes&utm\_source=recommendation)

(/p/f71cc7eb6f36?



utm\_campaign=maleskine&utm\_content=note&utm\_medium=seo\_notes&utm\_source=recommendation)

凡是过往 皆为序章 (/p/f71cc7eb6f36?utm\_campaign=maleskine&utm\_c...

把时间放在成长上，把时间放在真爱上，把时间放在有价值的事情上。 001 好久不见，你好吗？ 不知不觉的发现自己很久很久没有更新文章了。 仿佛一个世纪..... 这是一种很可怕的感觉。我都在干什么呢！ 在7月初

 陌上纤尘520 (/u/5c1adb130dc5?utm\_campaign=maleskine&utm\_content=user&utm\_medium=seo\_notes&utm\_source=recommendation)

(/p/a4440e8a2247?



utm\_campaign=maleskine&utm\_content=note&utm\_medium=seo\_notes&utm\_source=recommendation)

别样风情 (/p/a4440e8a2247?utm\_campaign=maleskine&utm\_content...

(/p/e2611fbca54e?



utm\_campaign=maleskine&utm\_content=note&utm\_medium=seo\_notes&utm\_source=recommendation)

**一个人的影响力不在仅仅取决于他的成就和贡献，更多的是取决于他跟这个...**

这几年有一个很有趣的现象，就是一出现像明星结婚或者出轨之类的都会被炒的火热，基本只要一出现全民都会去关注，于是有人就跑出来批判，认为这样的现象在一定程度上带坏了社会的风气，加之在去年老板在

 国玺同学 (/u/51ca9784b569?utm\_campaign=maleskine&utm\_content=user&utm\_medium=seo\_notes&utm\_source=recommendation)