Multipeer connectivity

----------by 张诚

专业名词

* Session objects (MCSession)：提供设备间的通信服务，如果创建了一个session，可以邀请其他的设备加入网络中，或者被其他设备邀请加入。
* Advertiser objects (MCNearbyServiceAdvertiser)：表示设备是可以被邀请加入特定的session.
* Advertiser assistant objects (MCAdvertiserAssistant)：功能与MCNearbyServiceAdvertiser类似，只是额外提供了接受邀请时的标准交互借口。
* Browser objects (MCNearbyServiceBrowser)：搜索附近的特定session类型的设备。
* Browser view controller objects (MCBrowserViewController)：提供标准的接口，让用户能够选择附近的设备加入一个session。
* Peer IDs (MCPeerID)： uniquely identify an app running on a device to nearby peers.

* 什么是Multipeer connectivity
* Multipeer connectivity是一个使附近设备通过Wi-Fi网络、P2P Wi-Fi以及蓝牙个人局域网进行通信的框架。互相链接的节点可以安全地传递信息、流或是其他文件资源，而不用通过网络服务。

广播与发现

* 通信的第一步是让大家互相知道彼此，我们通过广播(Advertising)和发现(discovering)服务来实现。
*
* 广播作为服务器搜索附近的节点，而节点同时也去搜索附近的广播。在许多情况下，客户端同时广播并发现同一个服务，这将导致一些混乱，尤其是在client-server模式中。
*
* 所以，每一个服务都应有一个类型（标示符），它是由ASCII字母、数字和“-”组成的短文本串，最多15个字符。通常，一个服务的名字应该由应用程序的名字开始，后边跟“-”和一个独特的描述符号。(作者认为这和 com.apple.*标示符很像)，就像下边：
* **static** NSString * **const** XXServiceType = @”zhangcheng";
* 一个节点有一个唯一标示MCPeerID对象，使用展示名称进行初始化，它可能是用户指定的昵称，或是单纯的设备名称。
* MCPeerID *localPeerID = [[MCPeerID alloc] initWithDisplayName: [[UIDevice currentDevice] name]];

使用前准备

```
需要添加的头文件
#import <MultipeerConnectivity/MultipeerConnectivity.h>
需要添加的协议
<MCSessionDelegate,MCNearbyServiceBrowserDelegate,MCBrowserViewControl
lerDelegate>
需要的指针
 //表示设备包含发现设备和建立会话阶段所需的各种属性
  MCPeerID*lockPeerID;
 //对象是最重要的，因为它代表目前的对等点（这个程序将运行的设备）将创建的会话
   MCSession*_session1;
     //广播服务
  MCAdvertiserAssistant* advertiser;
     //发现服务
  MCBrowserViewController* _broswerController;
     //显示数据
  UITextView*_textView;
static NSString*const xxServiceType=@"xx-service";
```

初始化广播和发现服务

```
//建立一个标示，使用设备名称
 lockPeerID=[[MCPeerID alloc]initWithDisplayName:[[UIDevice currentDevice] name]];
   //开始广播
 _session1 = [[MCSession alloc]initWithPeer:lockPeerID];
    _session1.delegate = self ;
    advertiser = [[MCAdvertiserAssistant alloc]initWithServiceType:xxServiceType discoveryInfo:nil session:_session1];
    [advertiser start];
    //开始发现
 MCNearbyServiceBrowser*browser=[[MCNearbyServiceBrowser alloc]initWithPeer:lockPeerID serviceType:xxServiceType];
    browser.delegate=self;
    [browser startBrowsingForPeers];
```

发现的相关代理

```
#pragma mark MCNearbyServiceBrowserDelegate
-(void)browser:(MCNearbyServiceBrowser *)browser foundPeer:(MCPeerID
*)peerID withDiscoveryInfo:(NSDictionary *)info
{//收到的是对方的名字
  NSLog(@"foundPeer~%@~~serviceType~
%@",peerID.displayName,browser.serviceType);
     //发送端和接收端要保持一致
  static NSString*const xxServiceType=@"xx-service";
      _broswerController = [[MCBrowserViewController
alloc]initWithServiceType:xxServiceType session:_session1];
     _broswerController.delegate = self ;
     [self presentViewController:_broswerController animated:YES
completion:^{
        // [browser stopBrowsingForPeers];
     }];
}
-(void)browser:(MCNearbyServiceBrowser *)browser lostPeer:(MCPeerID
*)peerID
{
     //对方断开连接诶
  NSLog(@"lostPeer~~%@",peerID.displayName);
}
```

MCBrowserViewController代理

```
#pragma mark MCBrowserViewControllerDelegate
-(void)browserViewControllerWasCancelled:(MCBrowserViewController *)browserViewController
{
    NSLog(@"cancel");
    [_broswerController dismissViewControllerAnimated:YES completion:nil];
}

-(void)browserViewControllerDidFinish:(MCBrowserViewController *)browserViewController
{
    [_broswerController dismissViewControllerAnimated:YES completion:nil];
    NSLog(@"Finish");
}
```

消息

* 消息分三种类型
* 1、消息数据（包括文本、图像以及可以转换为NSData对象的任何其他数据）
* 2、流数据
* 3、资源数据
* 传输这样的数据两种模式
* 1、可靠传输模式（类似TCP）
* 2、不可靠传输模式（类似UDP）

MCSession收取消息

```
//收取消息使用的是session的代理方法
-(void)session:(MCSession *)session didReceiveData:(NSData *)data fromPeer:(MCPeerID
*)peerID
{
    NSString*message=[[NSString alloc]initWithData:data encoding:NSUTF8StringEncoding ];
    NSLog(@"message~~%@",message);
//由于是辅线程中接收的消息所以要回到主线程中进行操作，否则崩溃
    dispatch_async(dispatch_get_main_queue(), ^{
          _textView.text=[NSString stringWithFormat:@"%@\n%@:说
%@",_textView.text,message,peerID.displayName];
    });
}
//连接的状态
-(void)session:(MCSession *)session peer:(MCPeerID *)peerID didChangeState:
(MCSessionState)state
{
//2为已经连接

  NSLog(@"didChangeState\n displayName~%@\n%ld",peerID.displayName,state);
     [self.dataArray addObject:session];
     if (state==2) {

        [self dismissViewControllerAnimated:YES completion:^{
```

MCSession发送消息1

```
//第一种发送消息
-(void)buttonClick{
    MCSession*session=[self.dataArray firstObject];
    NSString*message=@"hello world";
    NSData*data=[message dataUsingEncoding:NSUTF8StringEncoding];

    NSError*error;
消息MCSessionSendDataReliable 可靠
  MCSessionSendDataUnreliable 不可靠
//消息判断发送是否成功
BOOL is= [session sendData:data toPeers:session.connectedPeers
withMode:MCSessionSendDataReliable error:&error];
    if (!is) {
        NSLog(@"error~~%@",error);
    }else{

        _textView.text=[NSString stringWithFormat:@"%@\n我说:
%@",_textView.text,message];
    }
}
```

MCSession发送资源2（未验证）

```
//第二种发送资源
  NSString*fileName=@"文件名";
     NSString *filePath = [[NSHomeDirectory() stringByAppendingString:@"/Documents/"]
stringByAppendingPathComponent:fileName];
     //对方接收的新文件名
  NSString *modifiedName = [NSString stringWithFormat:@"%@_%@",
session.myPeerID.displayName,fileName ];
     NSURL *resourceURL = [NSURL fileURLWithPath:filePath];

     dispatch_async(dispatch_get_main_queue(), ^{
         NSProgress *progress = [session sendResourceAtURL:resourceURL withName:modifiedName
toPeer:[[session connectedPeers]firstObject]withCompletionHandler:^(NSError *error) {
             if (error) {
                 NSLog(@"Error: %@", [error localizedDescription]);
             }}];
//建立观察者，观察值的辩护
[progress addObserver:self
                   forKeyPath:@"fractionCompleted"
                      options:NSKeyValueObservingOptionNew
                      context:nil];

     });
```

MCSession发送资源3（未验证）

```
KVC观察资源传送情况
-(void)observeValueForKeyPath:(NSString *)keyPath ofObject:(id)object change:(NSDictionary *)change
context:(void *)context{
    //进度
 float Value=  [(NSProgress *)object fractionCompleted]*100;
    //更新进度百分比
  //.....
  }
//开始接收一个资源的传递
-(void)session:(MCSession *)session didStartReceivingResourceWithName:(NSString *)resourceName
fromPeer:(MCPeerID *)peerID withProgress:(NSProgress *)progress
{
    dispatch_async(dispatch_get_main_queue(), ^{
        [progress addObserver:self
                   forKeyPath:@"fractionCompleted"
                      options:NSKeyValueObservingOptionNew
                      context:nil];
    });
}
//接收完成
-(void)session:(MCSession *)session didFinishReceivingResourceWithName:(NSString *)resourceName
fromPeer:(MCPeerID *)peerID atURL:(NSURL *)localURL withError:(NSError *)error
{
    //resourceName文件名称
    //localURL 文件打开路径


}
```

MCSession发送资源3（未验证）

* NSOutputStream *outputStream = [session startStreamWithName:name toPeer:peer]; stream.delegate = self; [stream scheduleInRunLoop:[NSRunLoop mainRunLoop]
* [stream open];
* - (**void**)session:(MCSession *)session didReceiveStream: (NSInputStream *)stream
* withName:(NSString *)streamName fromPeer:(MCPeerID *)peerID { stream.delegate = self; [stream scheduleInRunLoop: [NSRunLoop mainRunLoop]
* [stream open]; }
* 输入和输出的streams必须安排好并打开,然后才能使用它们。一旦这样做,streams就可以被读出和写入。