



Praktikum – iOS-Entwicklung

Wintersemester 2018/19

Prof. Dr. Linnhoff-Popien

Markus Friedrich, Christoph Roch

Erinnerung: Präsentation der Zwischenergebnisse

- Wann: Mittwoch 12.12.18 in der Vorlesungsstunde (14:15 – 15:45 Uhr)
- Wo: Vorlesungsraum (Raum 151)
- Was: 5 minütige Präsentation
 - Appidee
 - Aktueller Stand
 - Eventuell teilfertige App zum Vorführen

Multipeer Connectivity

Einführung

Überblick

- Seit iOS 7
- Basierend auf dem Bonjour Protokoll (Automatische Erkennung von Netzwerkdiensten).
- MPC ermöglicht es Services in der Nähe zu erkennen und zu nutzen.
- Dazu wird Wi-Fi, Peer-to-Peer Wi-Fi und Bluetooth genutzt.

Phasen

Discovery Phase:

- Benutzung des **Service Browsers**...
 - ...um Peers für einen Service zu finden
 - ..um Peers zu einer Session einzuladen.
- Benutzung des **Advertisers**...
 - ...um Peers zu signalisieren, dass man einen bestimmten Service anbietet.
 - ...um über Einladungen anderer Peers informiert zu werden.

Session Phase:

- Direkte Kommunikation zwischen Peers einer Session.

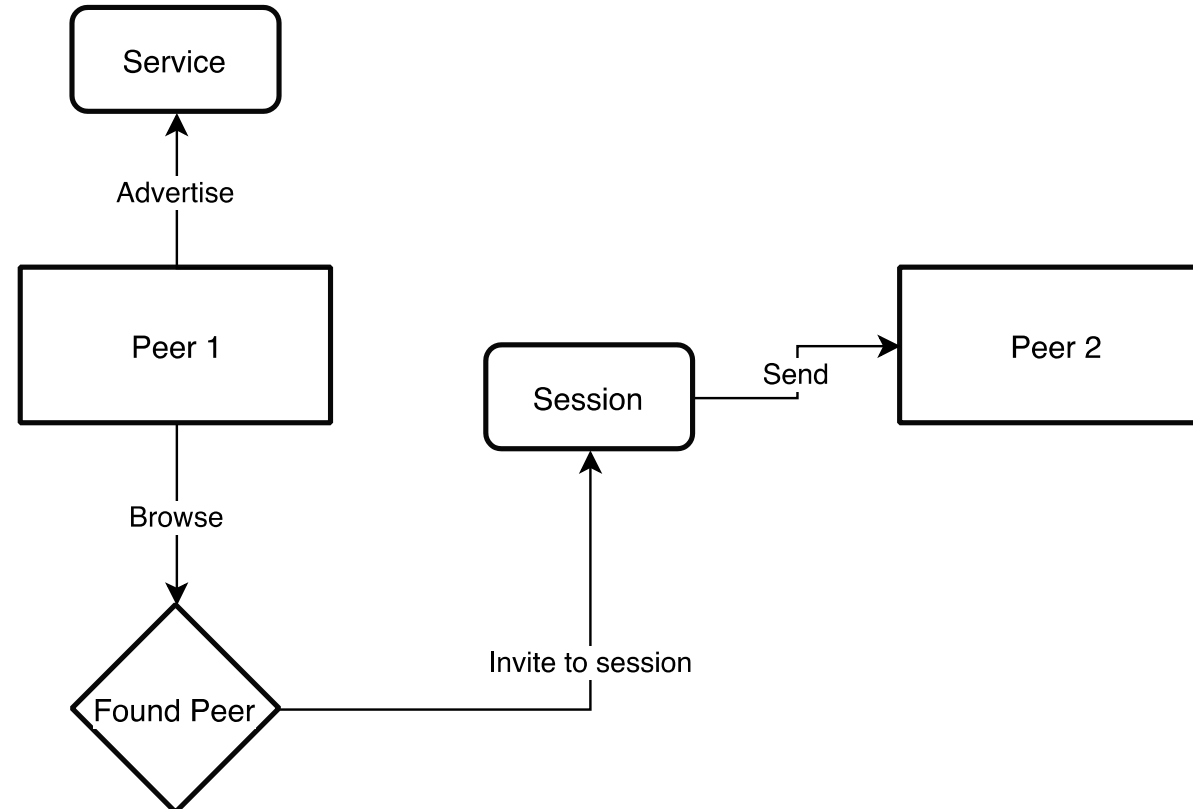
Sessions

- Peers kommunizieren über **Session-Objekte (MCSession)**
- Eine App erzeugt eine Session wenn ein Peer eine Einladung akzeptiert...
- ...oder wenn sie selbst eine Einladung erhält
- Session-Objekte halten eine Liste von Peer-Ids (MCPeerId)

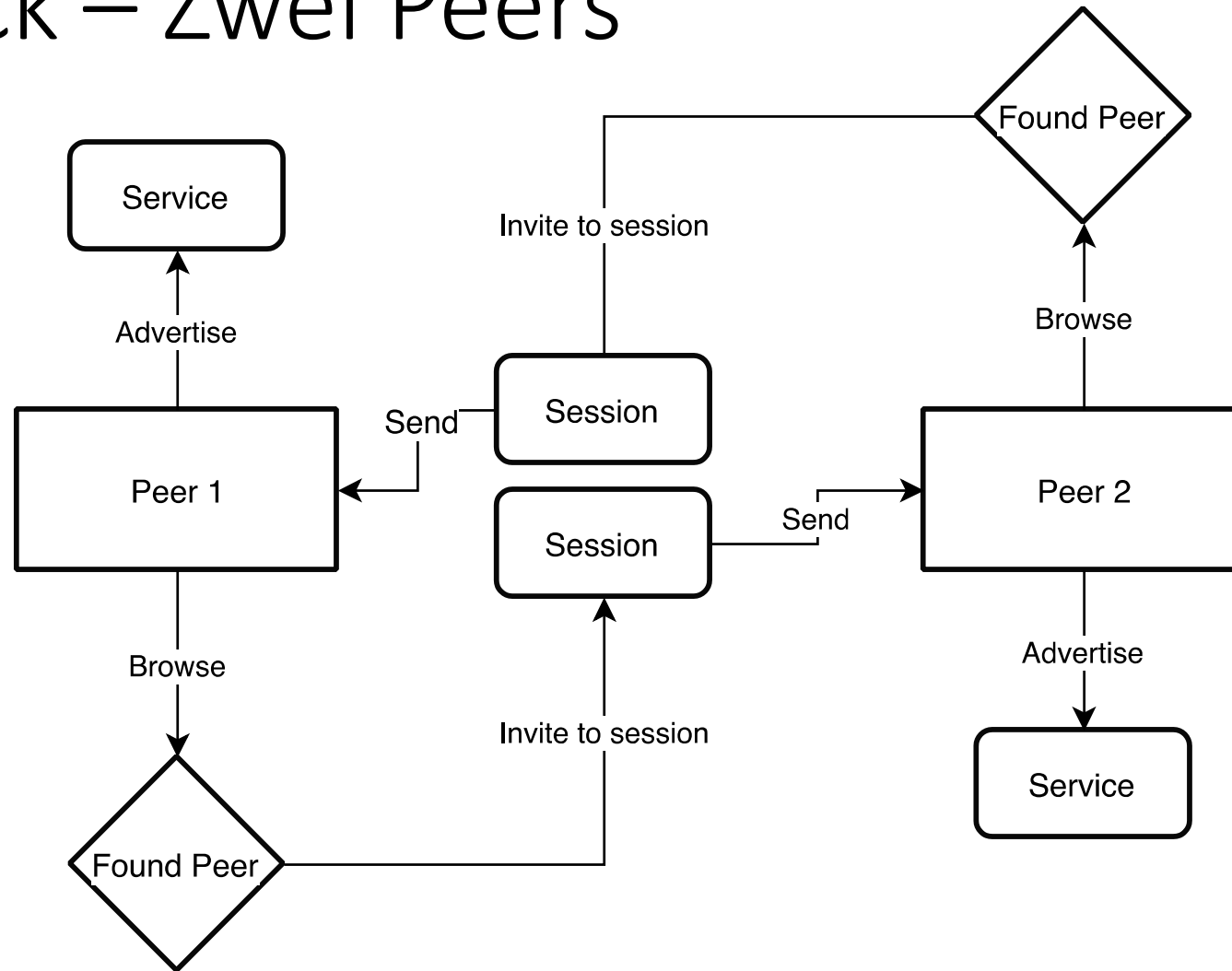
- Mögliche Kommunikation:
 - send(data, toPeers)
 - sendResource(at, toPeer)
 - startStream(toPeer)

- Verwendung von MCSessionDelegate um über hereinkommende Daten zu verwalten.
- Verwendete Verschlüsselung wird auch im MCSession Objekt konfiguriert.

Überblick



Überblick – Zwei Peers



Überblick

- Das Multipeer Connectivity Framework wird über die folgenden Klassen und Protokolle realisiert
- Kommunikation
 - **MCSession** und **MCSessionDelegate**
- Advertising von Services
 - **MCNearbyServiceAdvertiser** (peer, service type, discovery info) und **MCNearbyServiceAdvertiserDelegate** (wird aufgerufen, wenn eine Einladung erhalten wird)
- Browsing nach Services
 - **MCNearbyServiceBrowser** (peer, service type => auch für das Einladen eines Peers) und **MCNearbyServiceBrowserDelegate** (wird aufgerufen, wenn ein Peer gefunden wurde)

Advertising Services

- Advertising von Services geschieht über den `MCMNearbyServiceAdvertiser` und `MCMNearbyServiceAdvertiserDelegate` Protokoll

```
import MultipeerConnectivity

class MyServiceAdvertiser: MCMNearbyServiceAdvertiserDelegate {

    private let serviceAdvertiser : MCMNearbyServiceAdvertiser

    override init(){
        self.serviceAdvertiser = MCMNearbyServiceAdvertiser(...)
        self.serviceAdvertiser.delegate = self
        self.serviceAdvertiser.startAdvertisingPeer()

        // ...
    }
```

Service Advertiser Delegate

- Das MCNearbyServiceAdvertiserDelegate Protokoll verschiedene Methoden vor

```
func advertiser(advertiser, didReceiveInvitationFromPeer:)  
// Wird aufgerufen, bei Einladung zu einer Session  
  
func advertiser(advertiser, didNotStartAdvertisingPeer:)  
// Wird aufgerufen, wenn Advertisement fehlschlägt
```

Nach Services suchen

- Nach Services kann über ein MCNearbyServiceBrowser Objekt gescannt werden
- Außerdem muss unsere App das MCNearbyServiceBrowserDelegate Protokoll implementieren

```
class MyServiceScanner: MCNearbyServiceBrowserDelegate {  
  
    private let serviceBrowser : MCNearbyServiceBrowser  
  
    override init(){  
        self.serviceBrowser = MCNearbyServiceBrowser(myPeerId, „MyServiceType“)  
        self.serviceBrowser.delegate = self  
        self.serviceBrowser.startBrowsingForPeers()  
  
        // ...  
    }  
}
```

Service Browser Delegate

- Methoden des MCNearbyServiceBrowserDelegate Protokolls:

```
...  
func browser(browser, foundPeer: MCPeerID,...) // Wird aufgerufen wenn  
Peer gefunden wird => Invite  
  
func browser(browser, lostPeer: MCPeerID) // Wird aufgerufen wenn Peer  
verloren wurde
```

Verwendung

- App kann gleichzeitig Services advertisen und nach Services browsen
- Dazu z.B. eine eigene Klasse „ServiceManager“ implementieren

```
class MyServiceManager : MCNearbyServiceAdvertiserDelegate, MCNearbyServiceBrowserDelegate {  
  
    private let myPeerId = MCPeerID(displayName: UIDevice.current.name)  
    private let serviceAdvertiser : MCNearbyServiceAdvertiser  
    private let serviceBrowser : MCNearbyServiceBrowser  
  
    override init() {  
        self.serviceAdvertiser = MCNearbyServiceAdvertiser(...)  
        self.serviceBrowser = MCNearbyServiceBrowser(myPeerId, „MyServiceType“)  
  
        super.init()  
        self.serviceAdvertiser.delegate = self  
        self.serviceAdvertiser.startAdvertisingPeer()  
        self.serviceBrowser.delegate = self  
        self.serviceBrowser.startBrowsingForPeers()  
    }  
}
```

Verwendung

- Wichtig: wenn Objekt zerstört wird muss Advertising und Browsing gestoppt werden

```
class MyServiceManager : MCNearbyServiceAdvertiserDelegate, MCNearbyServiceBrowserDelegate {  
  
    ...  
  
    deinit {  
        self.serviceAdvertiser.stopAdvertisingPeer()  
        self.serviceBrowser.stopBrowsingForPeers()  
    }  
  
}
```

Versenden und Akzeptieren von Einladungen

```
class MyServiceManager : MCNearbyServiceAdvertiserDelegate, MCNearbyServiceBrowserDelegate, MCSessionDelegate {  
  
    // ...  
  
    // Anlegen eines lazy initiierten Session Objektes...  
    lazy var session : MCSession = {let session = MCSession(...)  
                                    session.delegate = self  
                                    return session}()  
  
    // ... und implementieren der MCSessionDelegate Methoden, wie z.B.  
    func session(_ session: MCSession, didReceive data: Data, fromPeer peerID: MCPeerID) {  
        NSLog("%@", "didReceiveData: \(data)")  
    }  
  
}
```


Alle entdeckten Peers einladen

```
class MyServiceManager : MCNearbyServiceAdvertiserDelegate, MCNearbyServiceBrowserDelegate, MCSessionDelegate {  
  
    // ...  
  
    func browser(_ browser: MCNearbyServiceBrowser, foundPeer peerID: MCPeerID, withDiscoveryInfo  
                info: [String : String]?) {  
        NSLog("%@", "foundPeer: \(peerID)")  
        NSLog("%@", "invitePeer: \(peerID)")  
        browser.invitePeer(peerID, to: self.session, withContext: nil, timeout: 10)  
    }  
}
```

Alle eingehenden Einladungen akzeptieren

```
class MyServiceManager : MCNearbyServiceAdvertiserDelegate, MCNearbyServiceBrowserDelegate, MCSessionDelegate {  
  
    // ...  
  
    func advertiser(_ advertiser: MCNearbyServiceAdvertiser, didReceiveInvitationFromPeer peerID:  
        MCPeerID, withContext context: Data?, invitationHandler: @escaping (Bool, MCSession?) -> Void) {  
        NSLog("%@", "didReceiveInvitationFromPeer \ \(peerID)")  
        invitationHandler(true, self.session)  
    }  
  
}
```

Weitere Informationen

- Ausführliches Tutorial mit Beispielcode:
 - <https://www.ralfebert.de/ios-examples/networking/multipeer-connectivity/>
- Apple Developer Sites:
 - <https://developer.apple.com/documentation/multipeerconnectivity-overview>