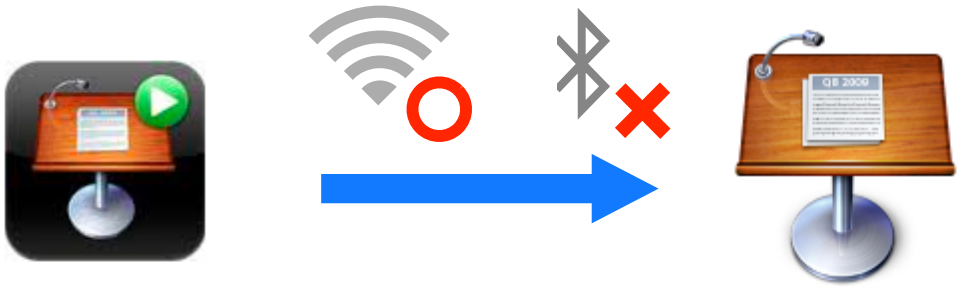
Bluetooth control without Picker

Yutaka Yasuda

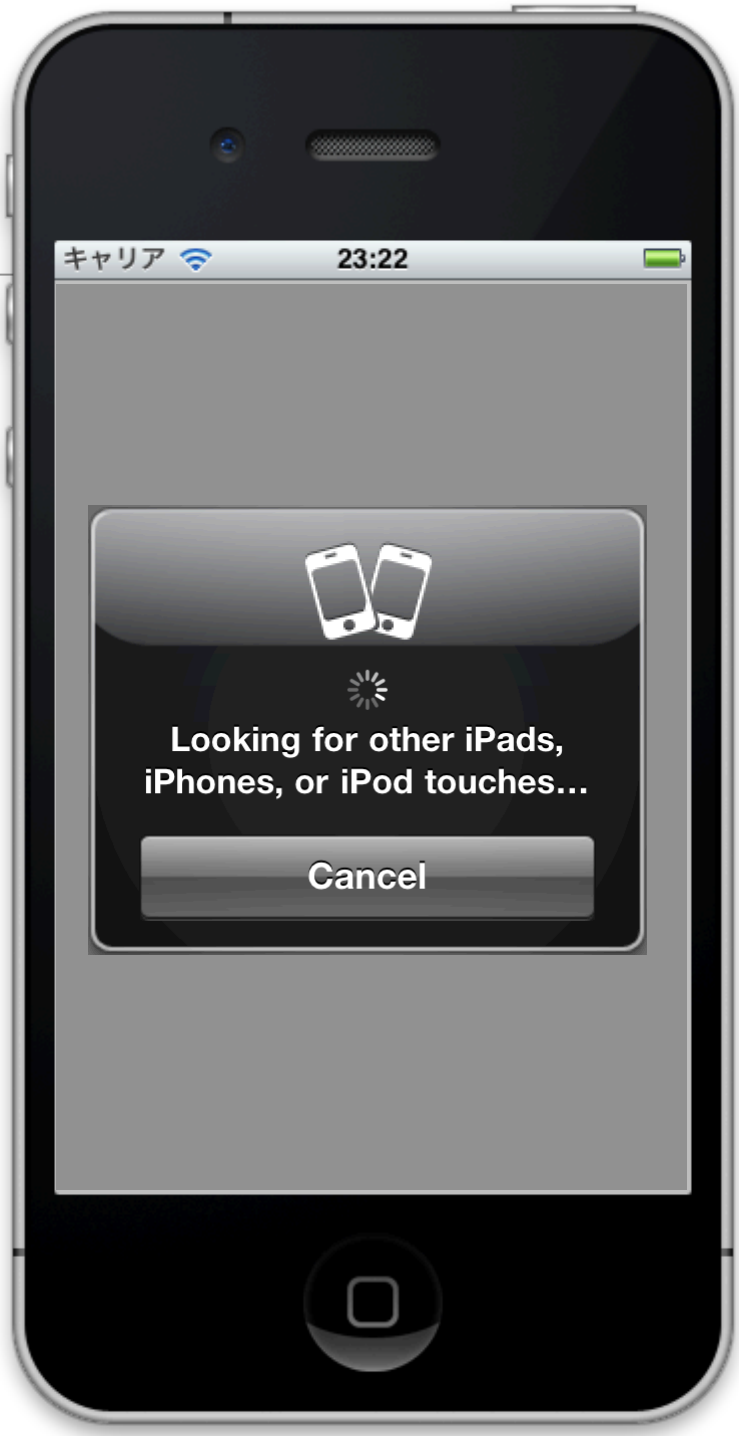
動機

- Bluetoothで何かしたい
- Bluetoothデバイスをコントロール
- Keynote を iPhone / Bluetoothでリモコン



PeerPiker

- GameKit に含まれている
- WiFi or Bluetooth で peer 接続するためのツール（GUIつきライブラリ）
- ちゃんと動くと良いんだけど
- ちゃんと動かない



GKRocket

- Sample source:
  http://developer.apple.com/library/ios/#samplecode/GKRocket/
- GKSession と GKVoiceChatService を使ったゲーム
- Picker を使わない
- それにしても遊び方がわかりません・・・・

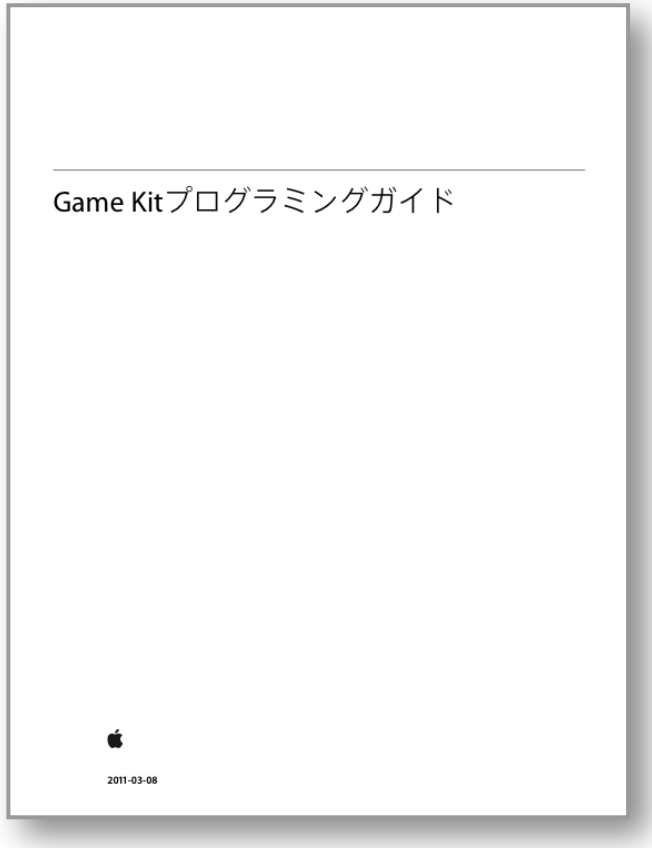
ドキュメント

- Game Kit プログラミングガイド

  http://developer.apple.com/jp/devcenter/ios/library/japanese.html

- 2011/3/8 で英語版と同じ

  （最近はほとんど全部同じ版）



接続形態

- peer to peer と Client server の二種類あり
- peer to per について
  詳細な記述あり
- client server では
  最大 16 接続

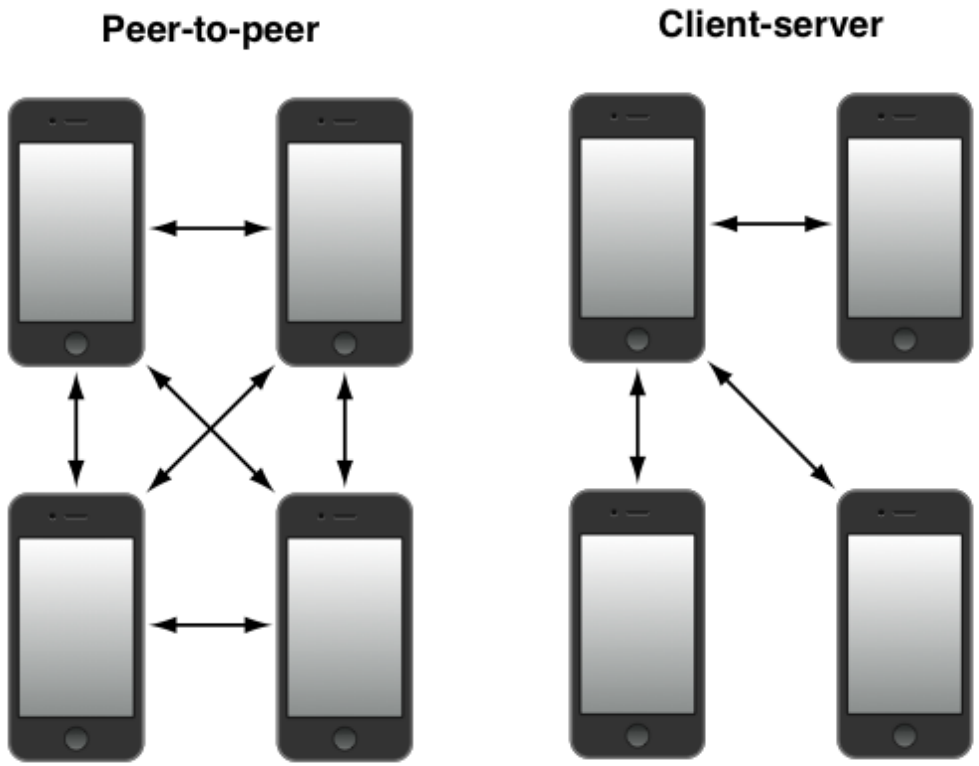


図 6-1 ネットワークトポロジー

Peer to Peer での接続管理

- 方法は二つ

  1. 独自のユーザインターフェイスを実装する

     （セッションで検出された他のユーザを表示する等の作業あり）

  2. GKPeerPickerController オブジェクトを使用する

     （2台のデバイス間にセッションを設定する標準UIつきライブラリ）

- いずれも接続できると GKSession オブジェクトができる

  これを利用して通信

Peer

- ネットワーク的には対等な接続をしているノード
- ソフトウェア的にはセッションオブジェクトと同義

  内部にデータの受信処理プログラムを登録
- Peer ID はシステムが一意に作成
- displayNameForPeer: メソッドで Peer のデバイス名を取得可能

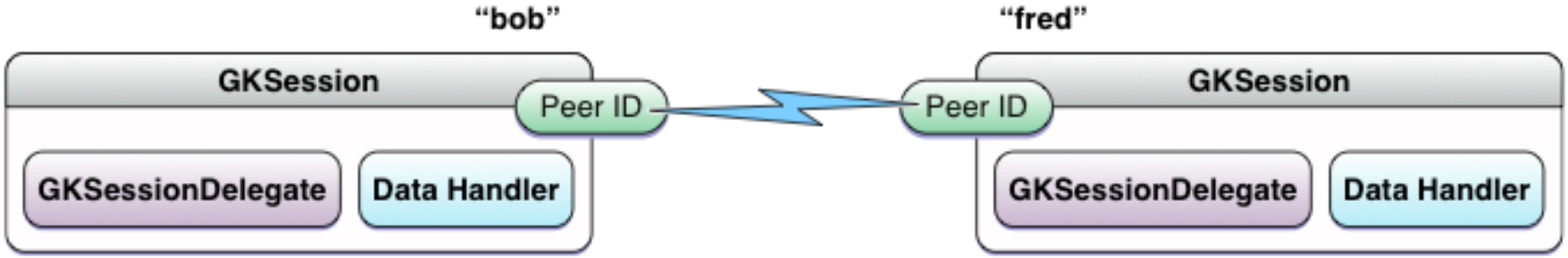


図 2 ほかのピアとのやり取りに使用されるピアID

Session ID

- 共通のネットワークに参加するノードが利用する一意な文字列

  “セッションIDは、登録済みの Bonjour サービスの短縮名でなければなりません”

  が、まあ実験なので適当にするのが良いかな

- ある初期化手続きで自動的にその session ID で広告される

  初期化によって検出も自動的に行われ、通知（callback）される

  検出されたノードに（これは意識的に）接続処理を行う

  接続が済めば、以後データの送受信ができる

典型的な手順

- session ID を決めて初期化

  自動的に session ID と peer ID がアドバタイズ(広告)される

  同時に検出も自動的に始まる
- 他ノードが（の）広告を検出すると callback によって通知
- 検出された相手に対して接続要求 (connect) を出す

  要求の到着も callback による通知
- 接続受理(accept)処理を行う

  これ以後データの送受信が可能
- 最後は disconnect して終了

データ送受信

- 送信処理はプログラムが任意のタイミングで行える
- データを受信するとセッションオブジェクトに登録した受信処理ハンドラが callback される
- 一度に交換するデータは 1KB 以下を推奨

  フラグメント（パケット分割）を避けよう

  最大でも 87KB まで
- 到達保証あり（再送あり）と保証無し（再送無し）を選択可

  送信時に決定可能（ほう）

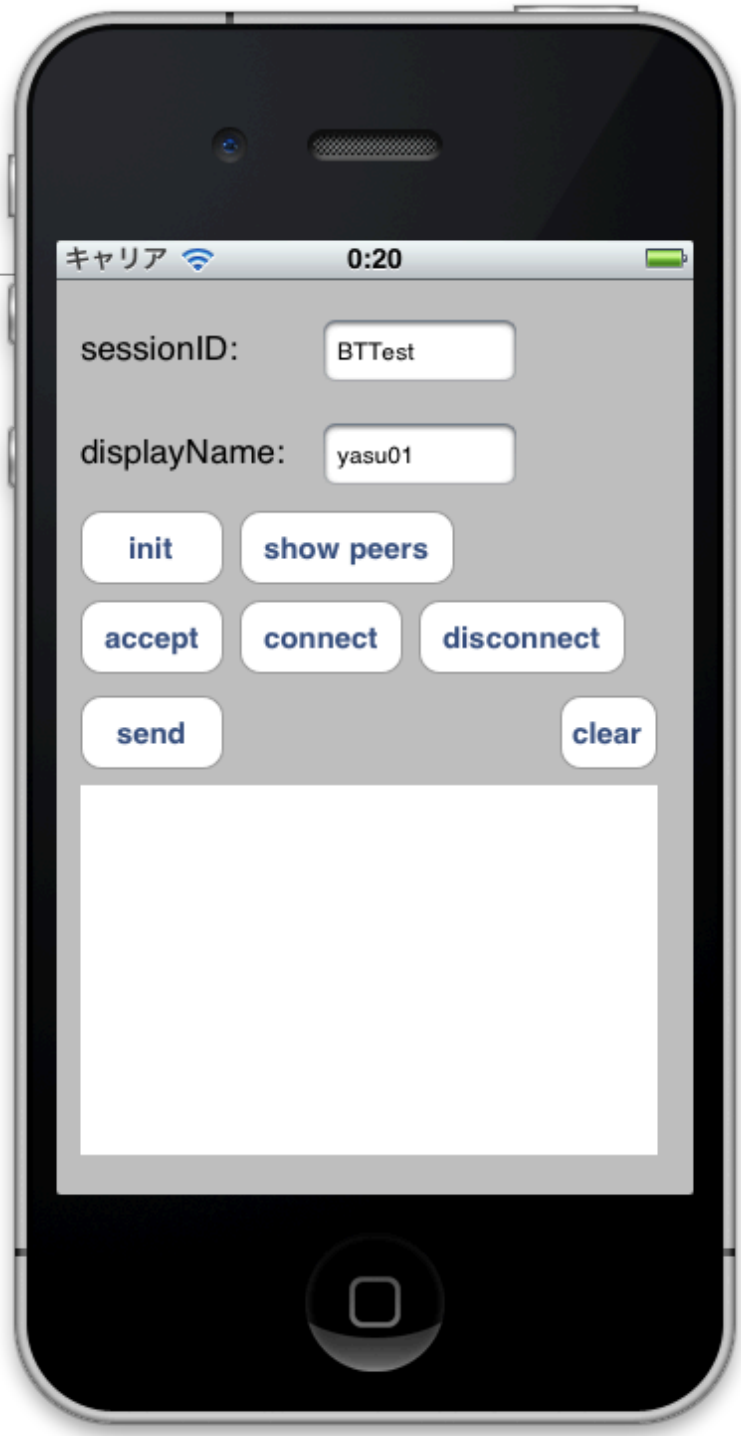
BTTestApp

- Bluetooth 接続を試すシンプルなアプリ

  GameKitなのでWiFi と共用

- Peer to Peer モードのみ対応
- Picker を使わない
- どんなときにどんなエラーが出るかといった追跡にも便利

注意：Bluetoothプログラムは
シミュレータでは動きません



初期化：

```
// GKsession を作り、ハンドラを設定し、広告をはじめる。これだけで準備は終わり。
session = [[GKSession alloc] initWithSessionID: [sessionID text]
                                   displayName: nil
                                   sessionMode: GKSessionModePeer];
session.delegate = self;  // いろいろ自分に定義
 // データ受信ハンドラを自分とする
[session setDataReceiveHandler:self withContext:nil];
session.available = YES;  // 広告開始
```

nil ならデバイス名が自動的に入る

明示的に文字列を与えればその名前で広告される

状態変化を通知する callback：

```
- (void)session: (GKSession *)l_session peer: (NSString *)peerID
didChangeState: (GKPeerConnectionState)state{
   switch (state) {
       case GKPeerStateConnected: // 接続された
       case GKPeerStateDisconnected: // 接続が切れた
       case GKPeerStateAvailable: // 広告された（availableな）ノードをみつけた
       case GKPeerStateUnavailable:  // 広告されていたノードが消えた
       case GKPeerStateConnecting: // 接続処理に入った
       default:
          break;
    }
}
```

接続要求を出す：

```
// available 状態のピアに接続要求を出す
// システムが見えているピア一覧を取得する一般的な手続き（指定された状態に限定）
peerList = [[NSMutableArray alloc]
       initWithArray:[session peersWithConnectionState: GKPeerStateAvailable]];

// 接続可能なピアに片っ端から接続申請
for ( NSString* aPeer in peerList) {
    [session connectToPeer: aPeer withTimeout: 1000];
}
```

受け入れ処理：

```
// 受け取った接続要求を受け入れる
// システムが見えているピア一覧を取得する一般的な手続き（指定された状態に限定）
peerList = [[NSMutableArray alloc]
    initWithArray:[session peersWithConnectionState: GKPeerStateConnecting]];

// 接続可能なピアに片っ端から接続申請
for ( NSString* aPeer in peerList) {
    [session acceptConnectionFromPeer: aPeer error: nil];
}
```

データの送信：

```
// システムが見えているピア一覧を取得する一般的な手続き（指定された状態に限定）
peerList = [[NSMutableArray alloc]
     initWithArray:[session peersWithConnectionState:GKPeerStateConnected]];
// 複数のピアに送る手続き
NSString *aStr = @"test-message";
[session sendData: [aStr dataUsingEncoding: NSASCIIStringEncoding]
            toPeers: peerList
       withDataMode: GKSendDataReliable
              error: nil];
```

Array をそのまま渡せる

信頼性あり通信

データの受信 callbak：

```
- (void)receiveData: (NSData *)data fromPeer: (NSString *)peer
                                    inSession: (GKSession *)l_session
                                      context: (void *)context {
  // 受信データの表示
  NSString* aStr = [[NSString alloc] initWithData: data
                                         encoding: NSASCIIStringEncoding];
  NSLog(@"receive #%@#", aStr);
```

セッションの切断：

```
// 接続中のピアを一発切断
[session disconnectFromAllPeers];

// 接続中の特定のピアだけ切断
[session disconnectPeerFromAllPeers];
```

接続要求の受信 (callback)：

```
- (void)session: (GKSession *) l_session
     didReceiveConnectionRequestFromPeer: (NSString *) peerID
```

これで要求を受信したら自動的に accept する、といった処理が可能

“すれ違い通信” への道だ！

peerID から displayName への変換：

```
[session displayNameForPeer: peerID]
```

とすることで、peerID （数字10桁ほど）から名前が得られる

SessionManager class of GKRocket

- setupSession
- destroySession
- willTerminate:
- willResume:

セッションの生成と消滅

- session: didReceiveConnectionRequestFromPeer:
- session: connectionWithPeerFailed: withError:
- session: didFailWithError:
- session: peer: didChangeState:

イベントからの callbak

- receiveData: fromPeer: inSession: context:

- displayNameForPeer:

雑用

legincebokumetsu App

- 事象が起きたときにボタンを押す
- その場で他のユーザもボタンを押せば、その数がカウントされる
- あとで記録を皆で共有する（ソーシャル化）

夢は広がる

みんながんばろう