Game Kitプログラミングガイド

2009-05-28

Apple Inc.
1 Infinite Loop
Cupertino, CA 95014
U.S.A.

アップルジャパン株式会社
〒163-1450 東京都新宿区西新宿
3 丁目20 番2 号
東京オペラシティタワー
http://www.apple.com/jp/

目次

図

はじめに

Game Kitフレームワークは、ゲームデベロッパ向けに、複数のiPhoneユーザを1つにつなげる機能を提供します。Game Kitには次のテクノロジーが含まれています。

- **ピアツーピア接続**。アプリケーションで複数のiPhone間にアドホックなBluetoothネットワークを作成できます。このネットワークはゲームを想定したものですが、アプリケーションのユーザ間のどのような種類のデータ交換にも利用できます。たとえば、アプリケーションでピアツーピア接続を使用して、電子名刺などのデータを共有できます。
- **ゲーム内ボイス**。アプリケーションで2台のiPhone間の音声通信を可能にします。ゲーム内ボイスは、アプリケーションを使用して2人のユーザ間に独自のネットワーク接続を作成します。

対象読者

この文書は、ユーザのiPhoneをBluetooth経由でほかのローカルデバイスに接続する機能や、ボイスチャット機能をアプリケーションに組み込もうとしているデベロッパを対象にしています。

この書類の構成

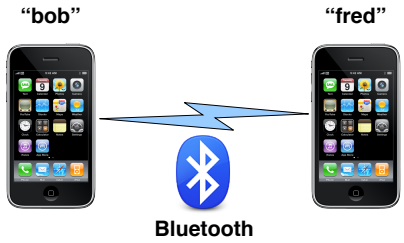
この文書は次の項目で構成されています。

- 「ピアツーピア接続」（9 ページ）では、Game Kitフレームワークに含まれる接続機能の概要を説明します。
- 「Peer Pickerによるピアの検索」（15 ページ）では、アプリケーションでPeer Pickerを使用して、iPhoneユーザをほかのユーザのiPhone上で実行されているアプリケーションのコピーに接続させる方法を説明します。
- 「セッションの使用」（17 ページ）では、Peer Pickerで設定したセッションをアプリケーションで使用する方法を説明します。
- 「ゲーム内ボイス」（19 ページ）では、Game Kitフレームワークで利用可能な音声テクノロジーの概要を紹介します。
- 「ボイスチャットの追加」（23 ページ）では、2台のiPhoneを接続するセッションを使用して、アプリケーションに音声通信を追加する方法を説明します。

ピアツーピア接続

GKSessionクラスを使うと、図 1に示すような、アドホックなBluetoothネットワークをアプリケーションで作成し、管理できます。複数のデバイス上で実行しているアプリケーションのコピーが互いを検出し、情報を交換することで、iPhone上にマルチプレーヤーゲームを作成するためのシンプルかつパワフルな手段を提供します。さらにセッションは、ユーザ同士が互いに連携できるような刺激的な新しい方法を、すべてのアプリケーションに提供します。

図 1　Bluetoothネットワーク



Bluetoothネットワークは、初期のiPhoneや第一世代のiPod touchではサポートされていません。また、iPhone Simulatorでもサポートされていません。

ピアツーピアアプリケーションを開発する場合、セッションで検出したほかのユーザに表示する独自のユーザインターフェイスを実装するか、GKPeerPickerControllerオブジェクトを使用して2台のiPhone間にセッションを設定する標準ユーザインターフェイスを提供できます。

デバイス間に一度ネットワークが確立された後は、GKSessionクラスはネットワーク上で送受信されるデータのフォーマットには関知しません。アプリケーションにとって最適なデータフォーマットを自由に設計できます。

注： この文書では、ピアツーピア接続クラス群で提供されるインフラストラクチャについて説明します。ネットワークゲームや、アプリケーションの設計および実装については取り上げません。

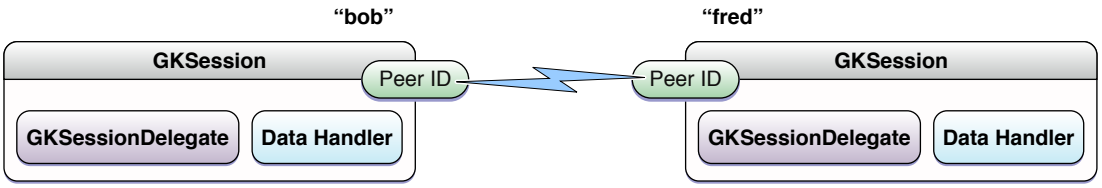
セッション

セッションは、作成されて互いを検出すると1つのネットワークに接続されます。アプリケーションは、接続済みのセッションを使用してほかのiPhoneにデータを送信します。アプリケーションはデリゲートを提供して接続要求を処理し、データハンドラでほかのiPhoneからアプリケーションへ送られるデータを受信します。

ピア

アドホックネットワークに接続したiPhoneを**ピア**と呼びます。ピアはアプリケーション内で実行しているセッションと同義です。各セッションは固有の**ピア識別子**文字列または`peerID`を作成して、ネットワーク上のほかのユーザに対して自分を識別させます。ネットワーク上のほかのピアとのやり取りはそれぞれのピアIDを通じて行われます。たとえばアプリケーションは、ほかのピアのIDを知っている場合、図 2に示すように、セッションの`displayNameForPeer:`メソッドを呼び出すことで、ユーザによる判読が可能なピア名を取得できます。

図 2　ほかのピアとのやり取りに使用されるピアID



ネットワーク上のほかのピアは、ローカルセッションにはさまざまな状態に見えます。ピアはネットワークに現れたり、消えたり、セッションに接続したり、セッションから切断したりします。アプリケーションでは、デリゲートの`session:peer:didChangeState:`メソッドを実装して、ピアの状態が変更された場合に通知されるようにします。

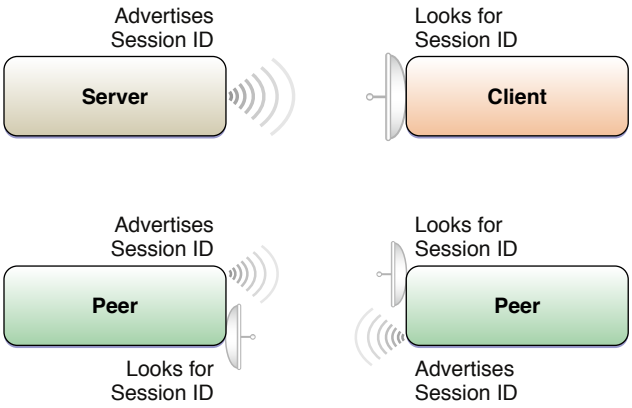
ほかのピアの検出

セッションはすべて、それぞれ独自のサービスタイプを実装しています。これは特定のゲームであったり、名刺交換のような機能であったりします。デベロッパには、サービスタイプに必要なものや、ピア間での交換に必要なデータを決定する責任があります。

セッションは、セッションの初期化時に設定される**セッションモード**に基づいてネットワーク上のほかのピアを検出します。アプリケーションでは、セッションを**サーバ**（ネットワーク上にサービスタイプをアドバタイズする）、**クライアント**（アドバタイズしているサーバを検索する）、または**ピア**（サーバのようにアドバタイズすると同時にクライアントのように検索もする）に設定できます。図 3にセッションモードを示します。

サーバは、自分のサービスタイプを**セッション識別子**文字列、または`sessionID`でアドバタイズします。クライアントは、一致するセッションIDを持つサーバのみを検索します。

図 3 サーバ、クライアント、およびピア



セッションIDは、Bonjourサービスの登録済み短縮名です。Bonjourサービスの詳細については、Bonjour Networkingを参照してください。セッション作成時にセッションIDを指定しなかった場合、セッションはアプリケーションのバンドル識別子を使用してIDを生成します。

接続を確立するには、少なくとも1台のiPhoneがサーバとしてアドバタイズを行い、別のiPhoneがそれを検索する必要があります。アプリケーションでは両方のモードになるためのコードを提供します。これを実装するには、アドバタイズと検索を同時に行うピアがもっとも柔軟な方法です。ただし、双方がアドバタイズと検索を行うため、セッションがほかのデバイスを検出するのにはより長い時間がかかります。

サーバの実装

サーバとして動作するアプリケーションは`initWithSessionID:displayName:sessionMode:`をセッションモード`GKSessionModeServer`、または`GKSessionModePeer`で呼び出してセッションを初期化します。アプリケーションでセッションを設定した後、セッションの`isAvailable`プロパティを`YES`に設定してサービスをアドバタイズします。

サーバは、クライアントが接続を要求すると通知を受け取ります。クライアントが接続要求を送信すると、デリゲートの`session:didReceiveConnectionRequestFromPeer:`メソッドが呼び出されます。デリゲートの典型的な動作は、`peerID`文字列を使用して、`displayNameForPeer:`を呼び出し、ユーザによる判読が可能な名前を取得します。次に、接続を受け入れるかどうかを決定するインターフェイスをユーザに表示します。

デリゲートは、セッションの`acceptConnectionFromPeer:error:`を呼び出して要求を受け入れるか、`denyConnectionFromPeer:`を呼び出して要求を拒否します。

接続が正常に作成されると、デリゲートの`session:peer:didChangeState:`メソッドが呼び出され、デリゲートに新しいピアが接続したことを知らせます。

サービスへの接続

クライアントとして動作するアプリケーションは`initWithSessionID:displayName:sessionMode:`をセッションモード`GKSessionModeServer`、または`GKSessionModePeer`で呼び出してセッションを初期化します。セッションの設定後、アプリケーションは、セッションの`isAvailable`プロパティ

をYESに設定してアドバタイズしているサーバを探そうとネットワークを検索します。セッションがGKSessionModePeerセッションモードに設定されると、前述のようにサーバとしてのアドバタイズも行います。

クライアントが利用可能なサーバを検出すると、デリゲートのsession:peer:didChangeState:メソッドが呼び出され、検出したサーバのpeerID文字列が提供されます。アプリケーションではdisplayNameForPeer:を呼び出して、ユーザ表示のためにユーザによる判読が可能な名前を取得できます。ユーザが接続するピアを選択すると、アプリケーションはセッションのconnectToPeer:withTimeout:メソッドを呼び出して接続要求を行います。

接続が正常に作成されると、デリゲートのsession:peer:didChangeState:メソッドが呼び出され、アプリケーションに新しいピアが接続したことを知らせます。

データ交換

セッションに接続したピアはほかの接続済みピアとデータを交換できます。アプリケーションは、sendDataToAllPeers:withDataMode:error:メソッドを呼び出すことですべての接続済みピアへ、またはsendData:toPeers:withDataMode:error:メソッドを呼び出すことで一部のピアへデータを送信できます。データは、NSDataオブジェクトにカプセル化されたメモリの任意のブロックです。アプリケーションは、データに使用したい任意のデータフォーマットを設計し、使用できます。独自のデータフォーマットはアプリケーションで自由に作成できます。最高のパフォーマンスを引き出すには、データオブジェクトのサイズを小さく（長さ1000バイト以下に）することをお勧めします。大きいメッセージ（95キロバイトまで）は、より小さい固まりに分割して送信先で組み立てなおす必要があるかもしれません。それにより遅延やオーバヘッドが増える可能性があります。

データを送信するには、送信先へのデータ到達に失敗した場合にセッションが再度データを送信する**確実な**方法か、送信を1度しか行わない**確実性の低い**方法かを選ぶことができます。確実性の低いメッセージは、データがリアルタイムに到着しなければほかのピアにとって有益とならない場合や、古いデータを再送するのではなく更新されたパケットを送信することのほうが重要な場合（たとえば推測航法情報）に適しています。

確実なメッセージは送信側が送信した順序とおりに参加者に受信されます。

ほかのピアから送信されたデータを受信するには、アプリケーションでオブジェクト上にreceiveData:fromPeer:inSession:context:メソッドを実装します。アプリケーションではsetDataReceiveHandler:withContext:メソッドを呼び出してこのオブジェクトをセッションに提供します。接続済みのピアからデータを受信すると、アプリケーションのメインスレッド上にデータハンドラが呼び出されます。

重要： ほかのピアから受信したデータはすべて*信頼できない*データとして取り扱う必要があります。ほかのピアから受信したデータは必ず検証し、セキュリティの脆弱性を避けるために慎重にコードを書くよう気をつけてください。詳細については、『*Secure Coding Guide*』を参照してください。

ピアの切断

アプリケーションでセッションを終了する準備ができたら、disconnectFromAllPeersメソッドを呼び出します。

アプリケーションは`disconnectPeerFromAllPeers:`メソッドを呼び出して特定のピアを切断できます。

ネットワークは本質的に不確かなものです。ピアが一定期間応答しなくなると、そのピアは自動的にセッションから切断されます。アプリケーションでは`disconnectTimeout`プロパティを変更して、セッションがほかのピアを切断するまでの時間を制御できます。

アプリケーションは、ほかのピアが切断したことをデリゲートの`session:peer:didChangeState:`メソッド内で検知できます。

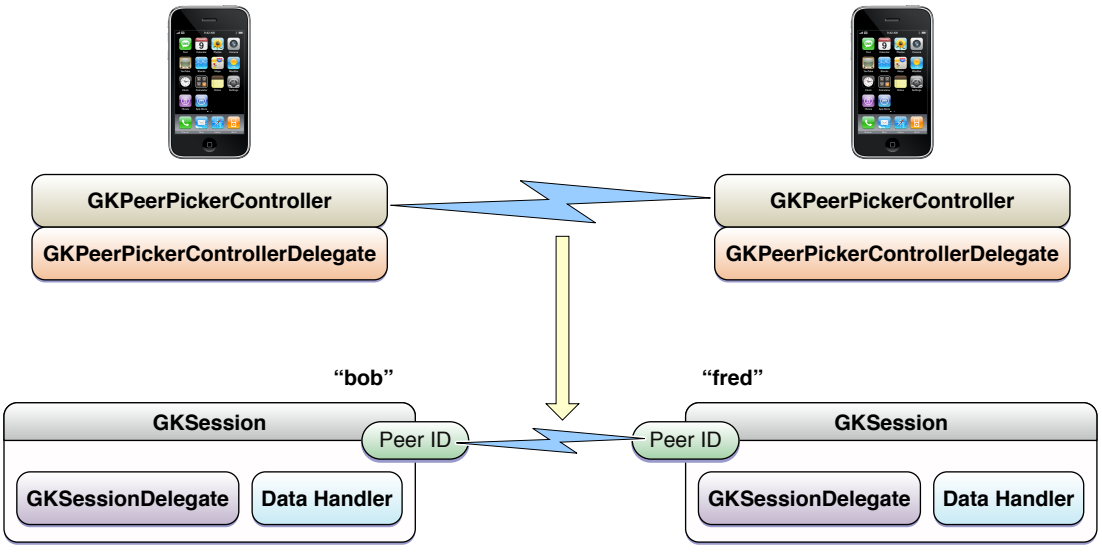
クリーンアップ

アプリケーションでセッションを廃棄する準備ができたら、アプリケーションはほかのピアから切断し、`isAvailable`フラグを`NO`に設定し、データハンドラおよびデリゲートを削除してセッションを解放します。

Peer Picker

`GSSession`のデリゲートを使用して独自のユーザインターフェイスを実装するよう選択できる一方で、Game Kitは検出および接続処理の標準的なユーザインターフェイスを提供しています。`GKPeerPickerController`オブジェクトはユーザインターフェイスを提供し、ほかのユーザのアクションに応答します。その結果、2つのピアを接続する`GKSession`が完全に設定された状態で作成されます。図 4にPeer Pickerがどのように動作するかを示します。

図 4　ネットワーク上の2つのピアを接続するセッションを作成するPeer Picker



Peer Pickerコントローラの設定

アプリケーションでPeer Pickerとのユーザインターフェイスとしてコントローラが呼び出すデリゲートを提供します。

Peer Pickerコントローラの`setConnectionTypesMask:`プロパティで、アプリケーションがユーザに対して提供する利用可能な接続メソッドのリストを設定します。iPhone OS 3.0では、Peer Pickerはローカルbluetoothネットワークとインターネットネットワークのどちらかを選択できます。アプリケーションで複数のネットワーク形式を含むようにマスクを設定すると、Peer Pickerコントローラは、ユーザがどちらのネットワークを使用するかを選択するための追加のダイアログを表示します。ユーザがネットワークを選択すると、コントローラはデリゲートの`peerPickerController:didSelectConnectionType:`メソッドを呼び出します。

重要： iPhone OS 3.0では、Peer Pickerはインターネット接続を設定しません。アプリケーションでインターネット接続を提供し、ユーザがインターネット接続を選択した場合、アプリケーションではPeer Pickerをキャンセルしてインターネット接続を設定するための独自のユーザインターフェイスを表示する必要があります。

Peer Pickerによって作成されたセッションをアプリケーションでカスタマイズする場合は、デリゲートの`peerPickerController:sessionForConnectionType:`メソッドを実装することができます。アプリケーションでこのメソッドを実装しない場合、Peer Pickerはアプリケーション用にデフォルトセッションを作成します。

Peer Pickerの表示

アプリケーションでPeer Pickerコントローラを設定すると、コントローラの`show`メソッドを呼び出してユーザインターフェイスを表示します。ユーザがほかのピアに接続すると、デリゲートの`peerPickerController:didConnectPeer:toSession:`メソッドが呼び出されます。アプリケーションはセッションの所有権を取得し、コントローラの`dismiss`メソッドを呼び出してダイアログを非表示にします。

ユーザが接続の試みをキャンセルすると、デリゲートの`peerPickerControllerDidCancel:`メソッドが呼び出されます。

Peer Pickerによるピアの検索

Peer Pickerは、Bluetooth経由で2人のユーザを接続するための標準ユーザインターフェイスを提供します。必要に応じてアプリケーションでは、インターネット接続かBluetooth接続かをユーザが選択できるようにPeer Pickerを設定できます。インターネット接続が選択された場合、アプリケーションはPeer Pickerのダイアログを閉じ、接続を完了させるための独自のユーザインターフェイスを表示する必要があります。

この章を読み終えたら「セッションの使用」（17 ページ）を読み、作成したセッションでアプリケーションが何をできるのか参照してください。

アプリケーションにPeer Pickerを追加するには、Peer Pickerコントローラのdelegateメソッドを保持する新しいクラスを作成します。次の手順に従います。

1. `GKPeerPickerController`オブジェクトを作成し、初期化します。

```
picker = [[GKPeerPickerController alloc] init];
```

2. デリゲートをアタッチします（メソッドはこの手順を進める中で定義します）。

```
picker.delegate = self;
```

3. 許可するネットワーク種別を設定します。

```
picker.connectionTypesMask = GKPeerPickerConnectionTypeNearby |
GKPeerPickerConnectionTypeOnline;
```

通常Peer Pickerのデフォルトは Bluetooth接続のみです。アプリケーションで接続種別マスクにインターネット（オンライン）接続を追加することもできます。アプリケーションでこれを行った場合、`peerPickerController:didSelectConnectionType:`メソッドも実装する必要があります。

4. 必要に応じて、インターネット接続が選択された場合にダイアログを終了させるために、`peerPickerController:didSelectConnectionType:`メソッドを実装します。

```
- (void)peerPickerController:(GKPeerPickerController *)picker
didSelectConnectionType:(GKPeerPickerConnectionType)type {
    if (type == GKPeerPickerConnectionTypeOnline) {
        picker.delegate = nil;
        [picker dismiss];
        [picker autorelease];
       // ここで独自のインターネットユーザインターフェイスを実装する
    }
}
```

5. デリゲートの`peerPickerController:sessionForConnectionType:`メソッドを実装します。

```
- (GKSession *)peerPickerController:(GKPeerPickerController *)picker
sessionForConnectionType:(GKPeerPickerConnectionType)type
{
```

```
    GKSession* session = [[GKSession alloc] initWithSessionID:myExampleSessionID
 displayName:myName sessionMode:GKSessionModePeer];
    [session autorelease];
    return session;
}
```

Peer Pickerコントローラの標準の動作を置き換えたい場合にのみ、アプリケーションでこれを実装する必要があります。

6. 設定済みセッションの所有権を受け取るためにデリゲートの`peerPickerController:didConnectPeer:toSession:`メソッドを実装します。

```
- (void)peerPickerController:(GKPeerPickerController *)picker
didConnectPeer:(NSString *)peerID toSession:(GKSession *) session {
// セッションの所有権を受け取るために保持しているプロパティを使用する
    self.gameSession = session;
// オブジェクトがセッションのデリゲートにもなると想定する
    session.delegate = self;
    [session setDataReceiveHandler: self withContext:nil];
// pickerを削除する
    picker.delegate = nil;
    [picker dismiss];
    [picker autorelease];
// ゲームを開始する
}
```

7. ユーザがPickerをキャンセルした場合に対処できるように、アプリケーションで`peerPickerControllerDidCancel:`メソッドも実装する必要があります。

```
- (void)peerPickerControllerDidCancel:(GKPeerPickerController *)picker
{
    picker.delegate = nil;
    // コントローラは自動的にダイアログを解除する
    [picker autorelease];
}
```

8. アプリケーションでダイアログを表示するためのコードを追加します。

```
[picker show];
```

セッションの使用

この章ではPeer Pickerによって設定されたGKSessionオブジェクトの使用方法を説明します。Peer Pickerの設定方法の詳細については、「Peer Pickerによるピアの検索」（15 ページ）を参照してください。

セッションは、ほかのピアに関する情報および接続済みピアから送信されるデータの2種類のデータを受信します。アプリケーションは、デリゲートを提供してほかのピアに関する情報を受信し、データハンドラを提供してほかのピアからの情報を受信します。

アプリケーション内でセッションを使用するには、最初に「Peer Pickerによるピアの検索」（15 ページ）の手順に従い、次にここから続けてください。

1. セッションデリゲートのsession:peer:didChangeState:メソッドを実装します。

   ほかのピアがセッションに対して状態を変化させると、セッションのデリゲートは通知を受けます。これらの状態のほとんどはPeer Pickerにより自動的に処理されます。アプリケーションで独自のユーザインターフェイスを実装している場合は、状態の変更をすべて処理する必要があります。とりあえずユーザがネットワークに対して接続または切断した場合にアプリケーションで対応しなければなりません。

```
- (void)session:(GKSession *)session peer:(NSString *)peerID
didChangeState:(GKPeerConnectionState)state
{
    switch (state)
    {
        case GKPeerStateConnected:
// ほかのピアのpeerIDを記録する
// ピアが接続したことをゲームに通知する
        break;
        case GKPeerStateDisconnected:
// ピアがいなくなったことをゲームに通知する
        break;
    }
}
```

2. ほかのピアにデータを送信します。

```
- (void) mySendDataToPeers: (NSData *) data
{
   [session sendDataToAllPeers:data withDataMode:GKSendDataReliable error:nil];
}
```

3. ほかのピアからデータを受け取ります。

```
- (void) receiveData:(NSData *)data fromPeer:(NSString *)peer inSession:
(GKSession *)session context:(void *)context
{
    // データのバイトを読み取り、アプリケーション固有のアクションを実行する
}
```

データはアプリケーションで即座に処理するか、または保持しておいて後でアプリケーション内で処理することができます。アプリケーションでは、このメソッド内での非常に長い計算は避けるべきです。

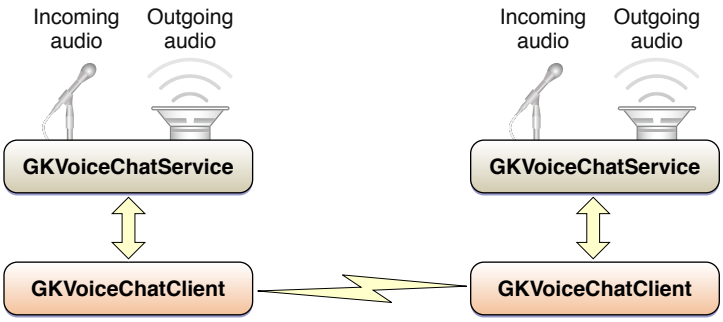
4. 接続を終了する準備ができたら、セッションをクリーンアップします。

```
[session disconnectFromAllPeers];
session.available = NO;
[session setDataReceiveHandler: nil withContext:nil];
session.delegate = nil;
[session release];
```

ゲーム内ボイス

図 1に示すように、GKVoiceChatServiceオブジェクトによって、アプリケーションにおいて2台のiPhone間のボイスチャット機能を簡単に作成できます。ボイスチャットサービスは、マイクをサンプリングし、ほかの参加者から受信した音声を再生します。ゲーム内ボイスは、GKVoiceChatClientプロトコルを実装したクライアントをアプリケーションが提供することを前提としています。クライアントの主な責任は、2人の参加者を接続し、ボイスチャットサービスが設定データを交換できるようにすることです。

図 1　ゲーム内ボイス



ボイスチャットの設定

参加者識別子

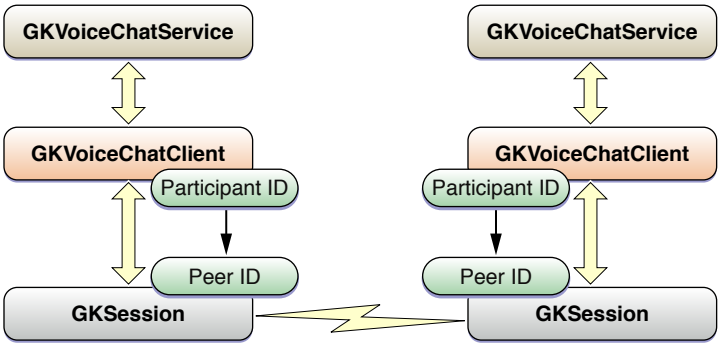
ボイスチャットのすべての参加者は、クライアントが提供する固有の**参加者識別子**文字列で識別されます。参加者識別子文字列の形式と意味はクライアントの決定に任されます。

ほかの参加者の検出

ボイスチャットサービスは、クライアントのネットワーク接続を使用して参加者間の設定データを交換し、二者間に直接の接続を作成します。ただし、ボイスチャットサービスはほかの参加者の参加者識別子を検出するためのメカニズムは提供しません。ほかのユーザの参加者識別子を提供し、これらの識別子をほかの参加者との接続へと変換するのはアプリケーションの責任です。

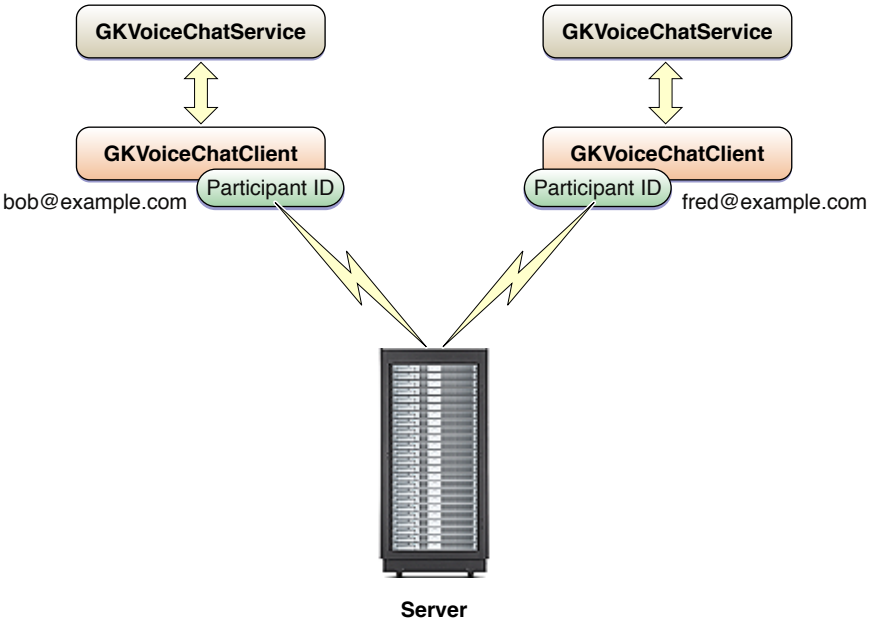
たとえば、GKSessionオブジェクトを通じてアプリケーションがすでにほかのデバイスに接続している場合（「ピアツーピア接続」　(9 ページ）を参照）、ネットワーク上のそれぞれのピアはすでにpeerID文字列により一意に識別されています。セッションではすでにほかの参加者のpeerID文字列を知っています。図 2に示すように、クライアントはそれぞれのピアのIDを参加者識別子として再利用し、セッションを使用してデータの送受信を行えます。

図 2 ピアツーピアベースの検出



2台のデバイスが互いを直接認識していない場合、アプリケーションでは2人の参加者が互いを検出し接続を行うためのほかのサービスが必要です。図 3では、サーバが電子メールアドレスで参加者をそれぞれ識別し、二者間のデータを送信できます。

図 3 サーバベースの検出



サーバの設計によって、クライアントに参加者識別子のリストを提供するか、ユーザがほかのユーザの参加者識別子（電子メールアドレス）を提供する必要があります。どちらの場合でも、サーバは2人のユーザ間のデータを転送する仲介者です。

ボイスチャットサービスが設定データをほかの参加者に送信する場合、クライアントの`voiceChatService:sendData:toParticipantID:`メソッドを呼び出します。クライアントはほかの参加者に確実かつ即座にデータを送信できる必要があります。ほかのクライアントがデータを受信すると、サービスの`receivedData:fromParticipantID:`メソッドを呼び出してサービスにデータを転送します。ボイスチャットサービスはこの接続を使用して、2人の参加者の間に独自のリアルタイムネットワーク接続を設定します。ボイスチャットサービスはクライアントの接続を、自身の接続を作成するためにのみ使用します。

リアルタイムデータ転送

場合によってはファイアウォールやNATベースのネットワークによって、ボイスチャットサービスが独自のネットワーク接続を確立するのが妨げられることがあります。アプリケーションはクライアント内にオプションのメソッドを実装して、参加者間のデータのリアルタイム転送を提供できます。クライアントで`voiceChatService:sendRealTimeData:toParticipantID:`メソッドを実装していると、ボイスチャットサービスが独自のリアルタイム接続を作成できない場合に、クライアントはフォールバックし、このメソッドを呼び出してデータを転送します。

チャットの開始

ボイスチャットを開始するには、参加者の1人がボイスチャットサービスの`startVoiceChatWithParticipantID:error:`メソッドをほかの参加者の`participantID`と共に呼び出します。サービスは前記のクライアントのネットワークを使用して新規のチャットを要求します。

サービスが接続要求を受信すると、クライアントの`voiceChatService:didReceiveInvitationFromParticipantID:callID:`メソッドが呼び出されて処理を行います。クライアントはサービスの`acceptCallID:error:`メソッドを呼び出してチャット要求を受け入れるか、`denyCallID:`を呼び出して要求を拒否します。接続を受け入れるかどうか、ユーザに確認したい場合もあります。

一度接続が確立されて受け入れられると、クライアントは自身の`voiceChatService:didStartWithParticipantID:`メソッドの呼び出しを受け取ります。

ほかの参加者からの切断

アプリケーションはサービスの`stopVoiceChatWithParticipantID:`メソッドを呼び出してボイスチャットを終了します。ほかのユーザがもう利用可能でないことを検出した場合もアプリケーションではチャットを停止する必要があります。

チャットの制御

参加者が接続すると、会話は自動的に2台のiPhone間で転送されます。アプリケーションでサービスの`microphoneMuted`プロパティを設定することでローカルのマイクをミュートしたり、サービスの`remoteParticipantVolume`プロパティを設定することでリモートの参加者のボリュームを調整することができます。

アプリケーションはまた、接続の両端の音量レベルを監視できるようにすることも可能です。たとえば、これを使用して、参加者が話しているときにユーザインターフェイスにインジケータを設定するといったことができます。ローカルユーザに対しては、アプリケーションは`inputMeteringEnabled`を`YES`に設定してメータを有効にし、`inputMeterLevel`プロパティを読み取ってマイクのデータを取得します。同様に、アプリケーションで`outputMeteringEnabled`を`YES`に設定し、`outputMeterLevel`プロパティを読み取ることでほかの参加者を監視することができます。アプリケーションのパフォーマンスを向上させるためには、測定を有効にするのは2人の参加者のメータレベルを読み取りたい時だけに限定するべきです。

ボイスチャットの追加

ボイスチャットはアプリケーションが提供するネットワーク接続の上に実装されます。次の例ではGKSessionオブジェクトを使用してクライアントにネットワークを提供しています。GKSessionオブジェクトの詳細については「ピアツーピア接続」（9 ページ）を参照してください。ボイスチャットを実装するには、次の手順を実行します。

1. 再生と録音を行うためのオーディオセッションを設定します。

```
AVAudioSession *audioSession = [AVAudioSession sharedInstance];
[audioSession setCategory:AVAudioSessionCategoryPlayAndRecord error:myErr];
[audioSession setActive:YES error:myErr];
```

2. クライアントのparticipantIDメソッドを実装します。

```
- (NSString *)participantID
{
    return session.peerID;
}
```

参加者識別子はクライアントを一意に識別する文字列です。セッションのpeerID文字列がすでにピアを一意に識別しているため、クライアントはそれを参加者識別子として再利用します。

3. クライアントのvoiceChatService:sendData:toParticipantID:メソッドを実装します。

```
- (void)voiceChatService:(GKVoiceChatService *)voiceChatService sendData:(NSData
 *)data toParticipantID:(NSString *)participantID
{
    [session sendData:data toPeer:[NSArray arrayWithObject:participantID]
withDataMode:GKSendDataReliable error:nil];
}
```

チャット中のほかの参加者にデータを送信する必要があると、サービスはクライアントを呼び出します。最も一般的には、こうしてほかの参加者との独自のリアルタイム接続を確立する場合です。GKSessionおよびGKVoiceChatServiceの両方がデータを保持するためにNSDataオブジェクトを使用するため、単純にそれをセッションに渡します。

同じネットワークが独自の情報の送信にも使用されている場合、ボイスチャットデータと自身のデータを区別するためにパケットの先頭に識別子を付加する必要が生じることもあります。

4. セッションの受信ハンドラを実装してボイスチャットサービスにデータを転送します。

```
- (void) receiveData:(NSData *)data fromPeer:(NSString *)peer inSession:
(GKSession *)session context:(void *)context;
{
[[GKVoiceChatService defaultVoiceChatService] receivedData:data
fromParticipantID:peer];
}
```

この関数はクライアントのvoiceChatService:sendData:toParticipantID:メソッドのミラーリングであり、セッションから受信したデータをボイスチャットサービスへ転送します。

5. クライアントをボイスチャットサービスに接続します。

```
MyChatClient *myClient = [[MyChatClient alloc] initWithSession:session];
[GKVoiceChatService defaultVoiceChatService].client = myClient;
```

6. ほかの参加者に接続します。

```
[[GKVoiceChatService defaultVoiceChatService]
startVoiceChatWithParticipantID:otherPeer error:nil];
```

アプリケーションでは接続処理の一部としてこれを自動で行うか、またはユーザにボイスチャットを個別に作成する機会を提供します。ボイスチャットの自動作成にもっとも適しているのは、セッションデリゲートの`session:peer:didChangeState:`メソッド内です。

7. オプションのクライアントメソッドを実装します。

アプリケーションでほかのユーザの確認にネットワーク接続を使わない場合、`GKVoiceChatClient`プロトコルの追加メソッドを実装する必要が生じることがあります。`GKVoiceChatClient`プロトコルは、ほかの参加者が接続を試みた場合や、状態を変化させた場合にクライアントに通知するための多くのメソッドを提供しています。

書類の改訂履歴

この表は「*Game Kit プログラミングガイド*」の改訂履歴です。

日付	メモ
2009-05-28	改訂し、概念を説明する資料をさらに追加しました。
2009-03-12	GameKitを使用してBluetooth上のローカルネットワークを実装する方法、および任意のネットワーク上のボイスチャットサービスを実装する方法について説明する新規文書。

書類の改訂履歴