Bluetooth Network for iOS, tvOS and Android

Introduction

This plugin provides a basic local network using Bluetooth Low Energy. It is built on top of the popular Bluetooth LE for iOS, tvOS and Android plugin also available in the Unity asset store.

That plugin is included in this plugin since it relies on it.

In order for client mode to function on an Android device the Android OS version must be at least 5.0 and the hardware also support peripheral mode. There are a number of devices that have the correct OS version, but not the hardware or drivers to support client mode.

If you start the Networking on an Android device that doesn't support client mode you will get an error in the Initialize error callback indicating as much. You can continue to use the network library in server mode if you like.

WARNING: The throughput of this network library is about 8 packets per second. It is not suited for games that require a higher throughput.

How It Works

Since this plugin uses Bluetooth to do the networking it works with the same principals as Bluetooth.

Bluetooth works by having centrals (the server) that watch for peripherals (the client). Those peripherals advertise that they are ready to be connected to and then the central connects to them.

Peripherals do not initiate any actions. They simply advertise that they can be connected to and then respond to commands from the central.

The one exception is that a central can subscribe to the characteristic (value) of a peripheral and from then on whenever the peripheral changes the value of that characteristic, the central is automatically notified with the new value.

The server of this networking plugin is a Bluetooth central. The clients are Bluetooth peripherals.

Internally the plugin sets up a single Bluetooth service and characteristic and uses that to perform all communication.

Version Changes

1.04

Updated initialization handling when bluetooth is off

1.03

Added support for the new Android permissions API in 2018.3

1.0

Initial Release

Setup Guide

Setting up this plugin involves installing the package and writing scripts to interact with the plugin. Follow these steps:

1. Import the package into your Unity project
2. Create a game object that you can attach a script to
3. Create a script that makes calls into the Networking object that you have obtained from a public variable set in the inspector or populated using GetComponent<Networking>();

Example Code

There is an example project that shows how to connect 2 devices together and send a text message that you type into the InputField.

Plugin Layout

The plugin has several parts.

Editor

This folder contains some scripts that are required for older versions of Unity.

Example

The example is detailed in the section above.

Plugins

This folder includes all the scripts and native code needed to build the plugin.

Support

For email support you can email support@shatalmic.com

Notes

All operations are asynchronous. This means you can't start a new operation until you have received the callback from the previous operation. If you do you will probably cancel the previous operation.

You can use timeouts and hope that you are done with the previous operation, but this is not as deterministic.

API Reference

Initialization Errors

When Initialize is called there are several errors that can occur. You will receive the error text as the parameter to the errorAction callback. Here is a list of those errors:

Bluetooth LE Not Enabled

Namespace

The Networking class lives in a namespace called Shatalmic. This is the name of the company that created this plugin.

Networking Methods

```
public void Initialize (Action<string> onError, Action<string> o
nStatusMessage)
```

This method is the first method you call to initialize the networking system. The onError and onStatusMessage callbacks will get called by the plugin when errors or status messages occur ANY TIME during the use of the plugin. This is due to the asynchronous nature of this plugin.

```
public void StartServer (string networkName, Action<NetworkDevic
e> onDeviceReady, Action<NetworkDevice> onDeviceDisconnected, Ac
tion<NetworkDevice, string, byte[]> onDeviceData)
```

This method is called to start the network server. The networkName is used to match up to other devices that will use the same name to connect together.

The callbacks all have the first parameter of a NetworkDevice. That object is used to know which device you are talking to when the callback is made.

The name of each callback method tells what type of event the callback is called for.

The onDeviceData event contains a string that is the Bluetooth characteristic. This is for future enhancement.

For now it is always the same characteristic. This method also has a byte array parameter which is the raw byte data the client is sending to the server.

```
public void StopServer (Action onFinishedStoppingServer)
```

This method is used to stop the networking server. After all clients have been disconnected from the onFinishedStoppingServer callback will be called.

```
public void WriteDevice (NetworkDevice device, byte[] bytes, Act
ion onWritten)
```

This method is used to write to a client from the server. The device parameter is the same as one returned in the StartServer, onDeviceReady callback. That is why those devices should be saved in some kind of list by the user of this plug.

The bytes parameter is the raw byte array to be send to the client.

After the bytes have been written the onWritten callback will be called. Do not try to write more data to the client until after this callback has been received.

```
public void StartClient (string networkName, string clientName,
Action onStartedAdvertising, Action<string, string, byte[]> onCh
aracteristicWritten)
```

This method is used to start a networking client that can be detected by a networking server.

The networkName parameter must match exactly the same name used when the server is started.

The clientName parameter is a name passed on to the server that can be used to display to end users.

The onStartedAdvertising callback is called when everything is set up and the client is waiting to be connected to a server.

```
public void StopClient (Action onStoppedAdvertising)
```

This method stops the client from advertising that it is available for connection. The onStoppedAdvertising callback will be called when that is completed.