Contents

Introduction .................................................................................................................................. 3
Background to 802.11 and Wi-Fi Profiles ...................................................................................... 4
Ad-Hoc Networking in the Intel Laptop Gaming TDK 2.0 .......................................................... 5
Getting Started with the Laptop Gaming TDK 2.0 ...................................................................... 5
Summary .................................................................................................................................... 7
References .................................................................................................................................... 8
About the Authors ....................................................................................................................... 8

Figures

1. Network Gaming with the Client-Server Model ............................................................... 3
2. Peer-to-peer Gaming Model ............................................................................................. 4
3. A sample Wi-Fi Profile: an Ad-Hoc network with SSID named COMPANY_PRODUCT1 .......................................................... 5
4. Instantiate an Ad-hoc Object ........................................................................................... 6
5. Example of Loading an Ad-Hoc Profile .......................................................................... 6
6. Start an Ad-Hoc Network ................................................................................................. 6
7. Example of Retrieving List of Users .............................................................................. 7

§
Introduction

People playing multi-player games usually do so over the Internet, with a remote host. This model is called the client-server model. In the case of multiple users, each user just connects to a common server, and the server forwards the packets to connected users. Figure 1 illustrates the client-server model.

![Diagram of Network Gaming with the Client-Server Model](image)

This client-server model suffers the following major drawbacks: a user cannot play games where there is no Internet infrastructure, or when the connection is too bad, or when the server is not available (either the server is down or refuses users because the maximum number of users is reached). Another drawback is that it limits the gamers from randomly announcing, discovering and joining a networked game.

Some consoles like the Nintendo Dshave* provided the capability for gamers to announce, discover and join each other’s games in a peer-to-peer (and in most cases in close proximity) fashion. Consoles have very little variability in their hardware design facilitating a standard and easy to connect interface. This is not the case with PCs where the wireless adapters could be developed by different OEMs and there might be interoperability issues between different OEM adapters, especially when it comes to peer-to-peer networking.

In this white paper we propose another approach to playing such games without the need of a game server and without an Internet infrastructure. This model is referred to as peer-to-peer gaming: multiple users within a certain range of each other can directly connect with each other using an Ad-Hoc network.

Ad-Hoc networking is a wireless technology that allows two or more mobile devices (with 802.11 a/b/g/n supported adapters) to connect together to form a private network. Ad-Hoc networking was first used for military applications such as battlefield communication, but today usage models include playing game in a coffee shop, playing game at an airport, exchanging information in a meeting room and so on. Figure 2 shows an Ad-Hoc network for three nearby laptops.
Background to 802.11 and Wi-Fi Profiles

802.11 is a protocol used for wireless communication. This protocol takes the Ethernet 802.3 and extends it to the wireless world. The 802.11 protocol consists of two components: the MAC sublayer (data link) and the physical layer. The 802.11 MAC layer determines how to access the medium and the 802.11 physical layer is responsible for details of transmission and receiving data. There are many 802.11 physical layers, among them are 802.11 frequency-hopping spread-spectrum, 802.11 direct-sequence spread-spectrum, 802.11a, 802.11b, 802.11g, and so on.

Each 802.11 network is referred as a Basic Service Set (BSS) in which each mobile station can communicate with each other. There are two types of 802.11 networks: independent network and infrastructure network. Independent network, or Ad-Hoc, is a network in which each laptop can communicate directly with each other, this type of network is used for short distance communication and is setup only for a short period of time. In an infrastructure network, each mobile station (laptop) communicates with a so-called access point. Thus, every communication channel has to go through access points. The access point also provides communication with a wired LAN if any. Infrastructure networks are more reliable than independent networks.

Each network is identified by a unique Service Set Identity (SSID). SSID is a character string defined by users. When two mobile stations connect to a network with the same SSID, they can communicate each other. Both independent network and infrastructure network have SSIDs.

A Wireless LAN Profile defines the type of connection and a set of attributes. One attribute of the Wireless LAN Profile is the SSID; another identifies the type BSS. Each mobile device uses the profile to establish the desired wireless connection. A typical Ad-Hoc wireless profile is shown in Figure 3.
Gaming Over Ad-Hoc Peer-to-Peer Networks

Figure 3. A sample Wi-Fi Profile: an Ad-Hoc network with SSID named COMPANY_PRODUCT1

```xml
<?xml version="1.0"?><WLANProfile xmlns="http://www.microsoft.com/networking/WLAN/profile/v1"><name>COMPANY_PRODUCT1-adhoc</name><SSIDConfig><SSID><hex>434F4D50414E595F50524F4455435431</hex><name>COMPANY_PRODUCT1</name></SSID><connectionType>IBSS</connectionType><MSM><security><authEncryption><authentication>open</authentication><encryption>none</encryption><useOneX>false</useOneX></authEncryption></security></MSM></SSIDConfig></WLANProfile>
```

**Note:** One attribute of the above profile is the SSID. The SSID is “COMPANY_PRODUCT1” and the corresponding hex string is 434F4D50414E595F50524F4455435431. Other attribute is the connection type, in this example it is IBSS, or Ad-Hoc network. Also, this network is configured as an open network with no protection key.

Ad-Hoc networks have the following typical characteristics: limited resources (each mobile device has a limited resource such as battery); distributed system (provide the ability for any node can join and leave the network at any time), poor physical security (anyone can send and receive signals in the air) and no fixed infrastructure (any device can move). Today, Ad-Hoc Networks are used in military and commercial applications.

Ad-Hoc Networking in the Intel Laptop Gaming TDK 2.0

The Intel Laptop Gaming Technology Development Kit [[1]] provides a set of APIs for network connectivity, power management, display information and processor information. Version 2.0 of this TDK adds Ad-Hoc networking APIs running on Microsoft Windows Vista* and Windows* XP. You can use these APIs to integrate with games and other applications to provide Ad-Hoc networking facilities. Source codes and sample codes are also included in the TDK.

The goal of the Laptop Gaming TDK is to provide a set of simple APIs that allows developers to manage their Ad-Hoc network and monitor peers in that Ad-Hoc network easily. Developers just need to use Microsoft Visual Studio* 2005 to compile and integrate these APIs with their applications.

Getting Started with the Laptop Gaming TDK 2.0

This section shows the most popular APIs in order to start an Ad-Hoc network and monitor peers.
To get started, include the appropriate header files and instantiate an object of type AdHoc.

**Figure 4. Instantiate an Ad-hoc Object**

```cpp
#include "AdHoc.h"

AdHoc AdHocApp;
```

Next, the application needs to load the corresponding Ad-Hoc wireless profile. A wireless profile includes information about configuration for the wireless card. The profile tells the 802.11 card to use an Ad-Hoc network and a specific SSID. An example SSID for a gaming application could be a combination of the game developer company name and the game title name. The company name differentiates each company with others. The game title differentiates each product in a company. That way, each application (game title) has a distinct SSID, avoiding collisions among different game titles. Even if collision does happen with similar SSID naming, the game could build some simple checks in place to ensure that the two game clients are compatible (i.e., they are the same versions of the game). The code for loading a corresponding Ad-Hoc wireless profile is shown in Figure 5.

**Figure 5. Example of Loading an Ad-Hoc Profile**

```cpp
............
int rc;
rc = AdHocApp.LoadProfile("COMPANY", "PRODUCT1");

if (rc == IMG_SUCCESS)
{
    // Continue your code
    ............
}
```

After the profile is loaded, the application can start the Ad-Hoc network by calling the method `StartAdHocNetwork`. This method in turn calls a Windows API to establish a Wi-Fi network giving an Ad-Hoc profile loaded.

**Figure 6. Start an Ad-Hoc Network**

```cpp
{
    ............
    int rc;
    rc = AdHocApp.StartAdHocNetwork("COMPANY", "PRODUCT1");
    if (rc == IMG_SUCCESS)
    {
        // Continue your code
        ............
    }
}
```

This method invokes two actions in sequence: first, the Ad-Hoc network is created by calling a Windows Native Wi-Fi API. Second, after the network is created, a component called Message Transport starts to exchanges messages with other peers in that Ad-Hoc...
network. The Message Transport component is responsible for monitoring and keeping track of peers in the same network. In version 2.0 of the TDK, there are four types of message being exchanged among peers:

- **Init message**: this message is broadcast when a host connects to the Ad-Hoc network.
- **Reply message**: upon receiving an init message, a node responds to the sender.
- **Keep-alive message**: broadcast to let others know that the node is still there.
- **Leave message**: this message is broadcast to let others know that the node is leaving the network.

The Message Transport component keeps track of peers with the above messaging system. At any time, the application can query the current list of peers by calling the method `GetListOfLocalUsers`.

**Figure 7. Example of Retrieving List of Users**

```c
{
    int rc = IMG_FAIL;
    int nNumPeers = 0;
    PeerData ListOfLocalUsers[MAX_NUM_PEER] = {0};

    rc = AdHocApp.GetListOfLocalUsers(nNumPeers,
                                         ListOfLocalUsers);

    if ((rc == IMG_SUCCESS) && (nNumPeers != 0))
    {
        for (int i=0; i<nNumPeers; i++)
        {
            // Retrieve data
            ........
        }
    }
}
```

The Ad-Hoc component returns to the application a list of peers: a peer name and its IP address. Knowing a peer address, the application now can start sending application specific messages to that peer (e.g., game invite message). After the receiver accepts the invitation, both sender and sender can exchange game data, thereby entering into a multiplayer networked mode.

Finally when a user wants to terminate the game, the `ShutdownNetwork` API method should be called. This method broadcasts the Leave message to other peers to notify that the node is about to exit, and then terminate the Ad-Hoc connection by calling into a Windows API.

**Summary**

The Intel Laptop Gaming TDK 2.0 provides a new set of simple APIs to helping game developers integrate their network games into Ad-Hoc networks easily. Developers can focus their effort on their game development rather than network management.

These APIs are easy to use and tested on Intel platforms with both Windows XP and Vista. Although these APIs are not yet tested on AMD laptops, they should work. Using these APIs, developers can setup a specific Ad-Hoc network and monitor users in that network.
Gaming Over Ad-Hoc Peer-to-Peer Networks

The design of the software is modular so that the architecture can be extended to include the use of the next generation of Intel wireless card where a laptop can be act as an Access Point and a Ad-Hoc network at the same time.

References


About the Authors

Loc Q Nguyen received his M.B.A. from the University of Dallas, Texas, and his Master's degree in Electrical Engineering from McGill University, Canada. He is currently an Application Engineer at Intel Corporation.

Satheesh G Subramanian is a Senior Applications Engineer in the Software Solutions Group at Intel Corporation currently working on graphics and gaming technology enabling and evangelism. Prior to this role, he had worked on developing proprietary VLSI CAD solutions for the Intel Microprocessor Teams. He has a Masters in Computer Science from the University of Illinois at Chicago, specializing in virtual reality and collaborative immersive environments in the CAVE™ virtual environments. He has presented in various technical conferences including SIGGRAPH and GDC. He is currently pursuing an MBA in entrepreneurship and entertainment management from the UCLA Anderson School of Management.