

# UEFI Development Exploration 49 – UEFI and Network 1

原创 luobing4365 Posted on 2020-03-22 21:47:04 Read 4.6k Collection 14 Likes 2

copyright

Category Column: UEFI Development Article Tags: UEFI Programming UEFI Network Low-level programming Low-level application development UEFI shell



UEFI Development This column includes this content

503 Subscribe 104 articles

Subscribe to

our column

(Please keep it-> Author: Luo Bing <https://blog.csdn.net/luobing4365> )

UEFI provides a very complete TCP/IP network protocol stack. Developers can even develop their own web servers on UEFI. Its network protocol stack is shown in Figure 1:

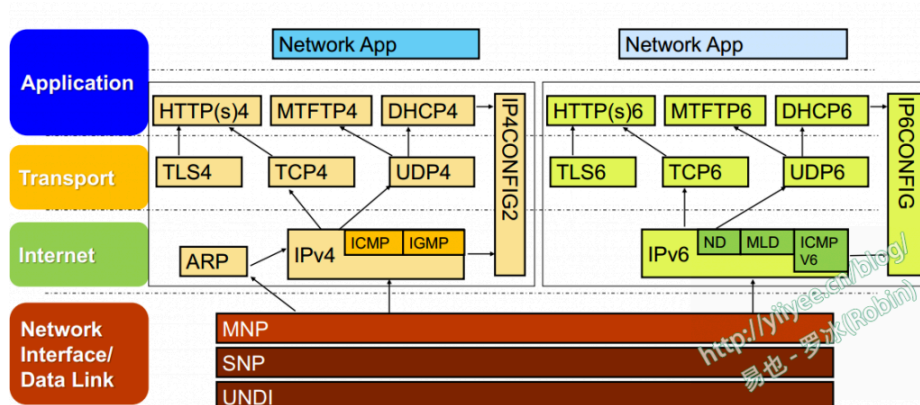


Figure 1 UEFI network protocol stack

## 1) Data Link Layer:

ARP (Address Resolution Protocol): Address translation protocol, converts IP address to physical MAC address;

MNP (Managed Network Protocol): Provides I/O operations for asynchronous data packets;

UNDI (Universal Network Device Interface): Universal network device interface;

SNP (Simple Network Protocol): Initializes and closes the network interface, hands over the network data frame to the network interface for transmission to the destination address, and receives the data frame from the network interface;

DPC (Deferred Procedure Call): Deferred procedure call, which is used to solve the TPL lock problem in the UEFI network stack;

## 2) Network layer:

IP (Internet Protocol): used for peer-to-peer data transmission between hosts;

## 3) Transport layer:

UDP (User Datagram Protocol): provides connectionless, unreliable datagram delivery service;

TCP (Transmission Control Protocol): a connection-oriented, reliable data transmission protocol;

## 4) Application layer:

MTFTP (Multicast Trivial File Transfer Protocol): Multicast Trivial File Transfer Protocol;

DHCP (Dynamic Host Configuration Protocol): Dynamic Host Configuration Protocol, providing services for discovering network boot servers;

PXE (PreBoot eXecution Environment): Pre-boot execution environment, used to discover network boot devices and download boot files;

iSCSI (Internet Small Computer System Interface): Network Small Computer System Interface, which sends the SCSI protocol originally used for the local machine through the TCP/IP network, is a storage technology based on the Internet and SCSI-3 protocol;

For the application to be developed, the transport layer protocols such as TCP and UDP, and the application layer protocols such as MTFTP and DHCP are mainly used. Before use, the network test environment needs to be configured first.

The network test environment we use can be divided into three types: one is the Nt32 simulation environment provided by Tianocore; the second is the real UEFI environment running on the computer; the third is to use a virtual machine to build it, the more common one is to use VirtualBox or Qemu.

## 1) Using the network in the Nt32 emulator

Before configuring the network, please make sure that the Nt32 simulator has been compiled. The specific compilation method is as follows:

```
C:\MyWorkspace> build -p Nt32Pkg\Nt32Pkg.dsc -a IA32
```

For network configuration under UEFI, there is an official reference document on Github at:

<https://github.com/tianocore/tianocore.github.io/wiki/Network-io>

According to the document, Nt32 network settings can be referred to "UEFI Network Stack for EDK Getting Started Guide". However, this article is a bit old, and some details are slightly different. Here is a brief description.

### 1) Download and install Winpcap .

**Winpcap is a professional software** for network packet capture . It is a free and public network access system. It can provide win32 applications with the ability to access the underlying network. In the simulator, it is equivalent to the driver of the network card. Download address:

<https://www.winpcap.org/default.htm>.

### 2) Download SnpNt32Io source code and compile

The code can be downloaded from github: <https://github.com/tianocore/edk2-NetNt32Io> . Create a folder NetNt32Io in the C drive and copy **the source code** into it.

Download Winpcap development package WpdPack from: <https://www.winpcap.org/devel.htm> . After downloading, unzip the compressed file of WpdPack and copy it to the C:\NetNt32Io directory.

Open the **Visual Studio** command line (the same command line as used to compile the UEFI code), enter the source code directory, and enter the following command:

```
C:\NetNt32Io> nmake TARGET=RELEASE
```

The directory Release\_IA32 will be automatically generated in the NetNt32Io folder. Copy SnpNt32Io.dll in this directory to the root directory of the UEFI emulator.

```
C:\NetNt32Io>copy /yc:\NetNt32IoRelease_IA32\SnpNt32Io.dll  
c:\MyWorkspace\Build\NT32\IA32\DEBUG_VS2015x86\IA32\
```

**3) Start the Nt32 simulator.** There are many ways to start it. I am more accustomed to directly clicking the SecMain.exe executable file in the directory to start the simulator.

### 4) Enter UEFI Shell and load the network protocol.

```
Shell> fs0:
```

```
FS0:\> load SnpNt32Dxe.efi MnpDxe.efi ArpDxe.efi Ip4Dxe.efi VlanConfigDxe.efi Udp4Dxe.efi Dhcp4Dxe.efi Mtftp4Dxe.efi Tcp4Dxe.efi
```

### 5) Configure the network card

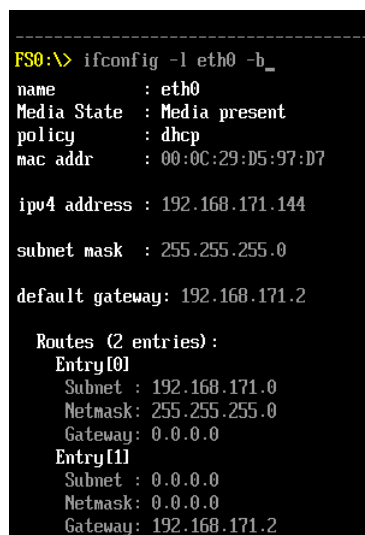
The environment I work in is dynamically assigned IP addresses through DHCP, using the ifconfig command to set:

```
FS0:\>ifconfig -s eth0 dhcp
```

You can also set a static IP address using the following command:

```
FS0:\>ifconfig -s eth0 static 192.168.1.188 255.255.255.0 192.168.1.1
```

That is to say, set a static IP address, subnet mask and gateway IP address. Of course, readers can also freely configure according to their own network conditions, including DNS address. The specific usage can be found through the command "ifconfig -? -b". Whether the IP address is successfully allocated can be checked through the command "ifconfig -l eth0".

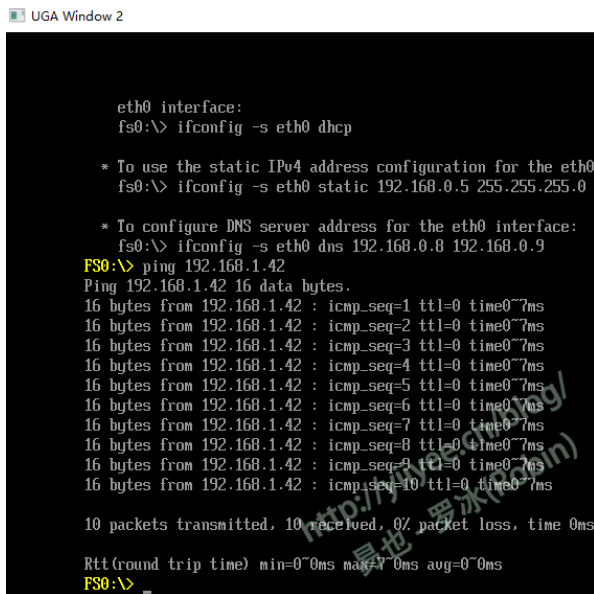


```
-----  
FS0:\> ifconfig -l eth0 -b_  
name           : eth0  
Media State    : Media present  
policy         : dhcp  
mac addr       : 00:0C:29:D5:97:D7  
  
ipv4 address   : 192.168.171.144  
subnet mask    : 255.255.255.0  
default gateway: 192.168.171.2  
  
Routes (2 entries):  
Entry[0]  
  Subnet : 192.168.171.0  
  Netmask: 255.255.255.0  
  Gateway: 0.0.0.0  
Entry[1]  
  Subnet : 0.0.0.0  
  Netmask: 0.0.0.0  
  Gateway: 192.168.171.2
```

Figure 2 View IP address configuration

### 6) Test network connection

The network connection can be tested through the ping command, as shown in the following screenshot:



```

UGA Window 2

eth0 interface:
fs0:\> ifconfig -s eth0 dhcp

* To use the static IPv4 address configuration for the eth0 interface:
fs0:\> ifconfig -s eth0 static 192.168.0.5 255.255.255.0

* To configure DNS server address for the eth0 interface:
fs0:\> ifconfig -s eth0 dns 192.168.0.8 192.168.0.9
FS0:\> ping 192.168.1.42
Ping 192.168.1.42 16 data bytes.
16 bytes from 192.168.1.42 : icmp_seq=1 ttl=0 time0.7ms
16 bytes from 192.168.1.42 : icmp_seq=2 ttl=0 time0.7ms
16 bytes from 192.168.1.42 : icmp_seq=3 ttl=0 time0.7ms
16 bytes from 192.168.1.42 : icmp_seq=4 ttl=0 time0.7ms
16 bytes from 192.168.1.42 : icmp_seq=5 ttl=0 time0.7ms
16 bytes from 192.168.1.42 : icmp_seq=6 ttl=0 time0.7ms
16 bytes from 192.168.1.42 : icmp_seq=7 ttl=0 time0.7ms
16 bytes from 192.168.1.42 : icmp_seq=8 ttl=0 time0.7ms
16 bytes from 192.168.1.42 : icmp_seq=9 ttl=0 time0.7ms
16 bytes from 192.168.1.42 : icmp_seq=10 ttl=0 time0.7ms

10 packets transmitted, 10 received, 0% packet loss, time 0ms

Rtt(round trip time) min=0.0ms max=0.7ms avg=0.0ms
FS0:\>

```

Figure 3 Checking the network connection in the Nt32 simulator

## 2 Using the network in a real UEFI environment

Using the network in a real UEFI environment mainly involves loading the network card driver and network protocol. I have done experiments on several machines. The BIOS in some environments may have problems and the configuration was not successful. The following experiments were completed on Intel's NUC6CAYH.

### 1) Download the network card driver under UEFI

You can download it from Intel's website at the following address:

<https://downloadcenter.intel.com/download/29137/Ethernet-Intel-Ethernet-Connections-Boot-Utility-Preboot-Images-and-EFI-Drivers>

I downloaded PREBOOT.exe, version 25.0. After installation (it is best not to install it on the C drive, it is better to install it directly on the desktop and delete it after use), there is a corresponding driver in the directory /APPS/EIF/EFIx64.

The driver is named in the form of EnnnnXm, where nnnn is the version number and m refers to different network card types. For example, E9112X3.EFI indicates a PCI-E Gigabit network port driver, and E7512X4.EFI indicates a 10Gbit/s network port driver.

The driver required by my experimental platform is E9112X3.EFI. I copied it to the UEFI boot disk and prepared for the next experiment.

### 2) Compile the x64 network protocol driver

EDK network protocol driver, the source code of Ipv4 is in MdeModulePkg, and the source code of Ipv6 is in NetworkPkg. We are currently mainly conducting experiments on Ipv4, so we need to compile MdeModulePkg. The operation of Ipv6 is similar, so I will not repeat it.

Open the Visual Studio command line and compile as follows:

```

C:\MyWorkspace> edksetup.bat;
C:\MyWorkspace> build -p MdeModulePkg\MdeModulePkg.dsc -a X64

```

After the compilation is complete, in the directory C:\MyWorkspace\Build\MdeModule\DEBUG\_VS2015x86\X64, copy the following drivers to the UEFI boot disk: SnpDxe.efi, MnpDxe.efi, ArpDxe.efi, Ip4Dxe.efi, VlanConfigDxe.efi, Udp4Dxe.efi, Dhcp4Dxe.efi, Mtftp4Dxe.efi, Tcp4Dxe.efi.

### 3) Load the network card driver and network protocol driver.

Use the UEFI boot disk to enter the UEFI shell test environment.

```

Shell>fs0:
fs0:>load E9112X3.EFI
fs0:> load SnpDxe.efi MnpDxe.efi ArpDxe.efi Ip4Dxe.efi VlanConfigDxe.efi Udp4Dxe.efi Dhcp4Dxe.efi Mtftp4Dxe.efi Tcp4Dxe.efi

```

### 4) Configure the network card.

Use ifconfig command to set:

```
fs0:\>ifconfig -s eth0 dhcp
```

This step is the same as step 5 of configuring the network for the Nt32 simulator in the previous section. For related commands, refer to the previous section.

### 5) Test the network card.

Test the network card through the Ping command, as shown in the following figure:

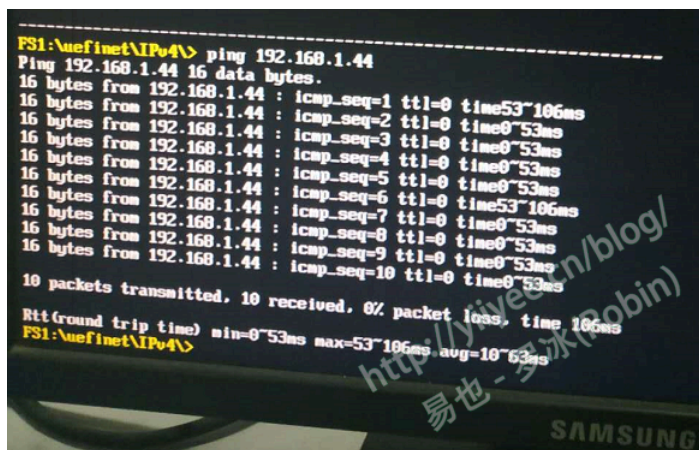


Figure 4 Checking network connection in a real UEFI environment

At this point, the network test environment has been set up, and you can use the UEFI network protocol stack to develop network applications.