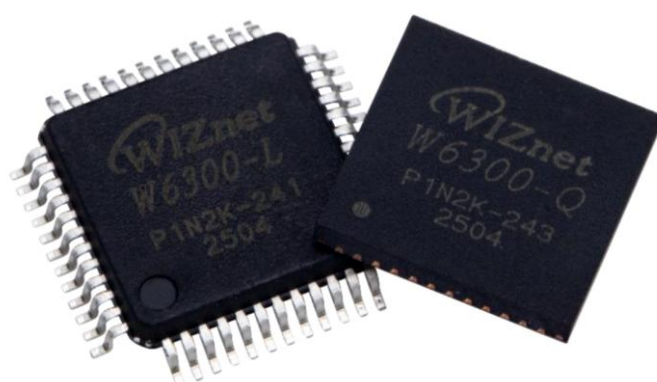


How to implement Auto-Configuration for W6300

Version 1.0.0



© 2025 WIZnet Co., Ltd. All Rights Reserved.

For more information, visit our website at <http://www.wiznet.io>

Table of Contents

1	Auto-Configuration Introduction	3
2	Stateless Auto-Configuration	3
2.1	Stateless Link-Local Address Auto-Configuration.....	3
2.1.1	Sudo Code	6
2.2	Stateless Global Address Auto-Configuration	7
2.2.1	Sudo Code	8
3	Stateful Auto-Configuration.....	9
3.1	Stateful Global Address Auto-Configuration(DHCPv6)	9
3.1.1	Sudo Code	10
4	Document History Information	11

1 Auto-Configuration Introduction

Host IP Address의 자동 생성은 DHCP와 같은 서버를 이용하여 주소를 획득하는 Stateful auto-configuration 방법과 Host 측에서 스스로 주소를 생성하는 Stateless auto-configuration 방법으로 분류된다. 서버를 사용하는 방법은 Host 측에서 DHCP 서버에 주소를 요청하면, 서버에서 할당 가능한 주소 중 하나를 Host 측에 할당하는 것이다. 따라서 서버는 대규모 데이터베이스를 갖추어야 하며, 엄격한 관리가 요구된다. 그에 반해 Stateless auto-configuration 방식은 Host 측에서 스스로 주소를 생성하는 방법으로 자신의 인터페이스 ID 정보와 Router로부터 획득한 Prefix 정보 또는 well-known prefix 정보를 이용하여 주소를 생성하는 방법이다. 따라서 Host가 자신의 IP Address에 대한 생성 및 할당을 책임진다.

2 Stateless Auto-Configuration

Stateless Auto-Configuration에는 Link-Local Address와 Global Address를 설정하는 방법이 있다.

2.1 Stateless Link-Local Address Auto-Configuration

링크-로컬 주소는 단일 링크 내에서만 사용가능하며, 경계 라우터가 외부망으로 전파되는 것을 차단한다. 물론, 라우터가 존재하지 않거나 DHCP 서버가 존재한다고 할지라도 사용 가능하다. 이와 같은 환경에서 생성된 링크-로컬 주소는 시간적인 사용 제약을 받지 않으며, 단일 링크 내에서는 언제든지 사용 가능하다. 이러한 링크-로컬 주소로는 라우터의 도움 없이 주소 자동 설정 방식에 의해 획득 될 수 있으며, 이웃 탐색 프로토콜 등에서 송수신 노드 주소 값으로 이용된다. 이와 같은 링크 로컬의 주소의 형식은 아래 그림과 같다.

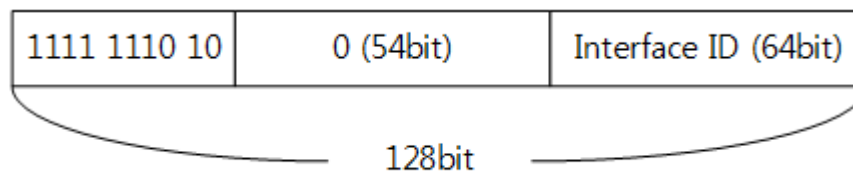


Figure 1 링크 로컬 주소의 구조

IPv6 주소의 앞 부분에 위치하는 비트 집합을 Prefix라 하며, 위 그림의 FE80::/10은 링크-로컬 용으로 미리 정해진 Prefix이다. 이 Prefix는 각 패킷이 전달될 수 있는 활용 범위를 단일 링크로 제한하기 위해 이용되고, 패킷이 속한 네트워크를 식별하기 위해서도 사용된다. 따라서 이 Prefix를 이용하면, 라우터로부터 Prefix에 대한 정보를 획득하지 않고서도 Host만이 자신의 링크 로컬 주소를 생성할 수 있는 것이다. 이와 같은 링크-로컬 주소는 시스템 부팅 시를 비롯하여 다음과 같은 경우에 설정된다.

- 시스템 시작 시, 네트워크 인터페이스를 초기화 하는 경우

- 일시적인 인터페이스 오류가 발생한 이후나 시스템 관리상 일시적으로 사용할 수 없도록 한 후에 인터페이스가 초기화된 경우
- 인터페이스가 처음으로 링크에 붙는 경우
- 시스템 관리상 인터페이스를 사용할 수 없게 한 이후에 인터페이스를 사용할 수 있게 하는 시스템 조작이 있는 경우

일반적인 링크-로컬 주소를 생성하는 방법을 살펴보면, 그림과 같이 well-known Prefix 정보에 자신의 인터페이스 ID 정보를 붙여서 생성한다. 이 때, Prefix와 인터페이스 ID 사이에는 0으로 채워지며, 만일 사용하고자 하는 ID 비트수가 Prefix 비트 수인 10bit를 제외한 나머지 118bit 이상일 경우에는 주소 생성이 불가능하다. 그러나, 일반적으로 인터페이스 ID는 64bit를 사용하도록 권고되고 있다.



Figure 2 Ethernet 환경에서의 인터페이스 ID 구성 방법

현재 가장 널리 사용되고 있는 Ethernet 환경을 기반으로 링크-로컬 주소 설정 방법을 살펴보고자 한다. 앞에서 설명한 것처럼 Prefix는 이미 정해져 있으므로, Ethernet 환경에서 링크-로컬 주소를 생성하기 위해서는 인터페이스 ID를 구성해야 한다. 인터페이스 ID를 구성하는 방법은 Figure 2에 제시된 방법을 따라 생성하면 되며, Ethernet 링크-계층 주소가 48bit의 “00:08:DC:17:FC:0F”이면 인터페이스 ID는 64bit의 “02:08:DC:FF:FE:17:FC:0F” 값을 얻을 수 있다. 여기서, “FF:FE”는 Ethernet 환경에서 IEEE EUI-64(Extended Unique Identifier) 형식을 따르도록 첨가한 것이며, 7번째 비트를 “1”로 변경한 것은 인터페이스 ID가 글로벌 범주를 가진다는 의미이다.

이렇게 만들어진 인터페이스 ID를 Figure 1의 인터페이스 ID 필드에 넣으면, 링크-로컬 주소가 완성된다. 이 링크-로컬 주소는 링크 내에서 주소의 유일성을 검증 받지 않았기 때문에 임시 주소라 불리며, Figure 3과 같은 중복주소 검출 과정(DAD)을 수행해야만 된다

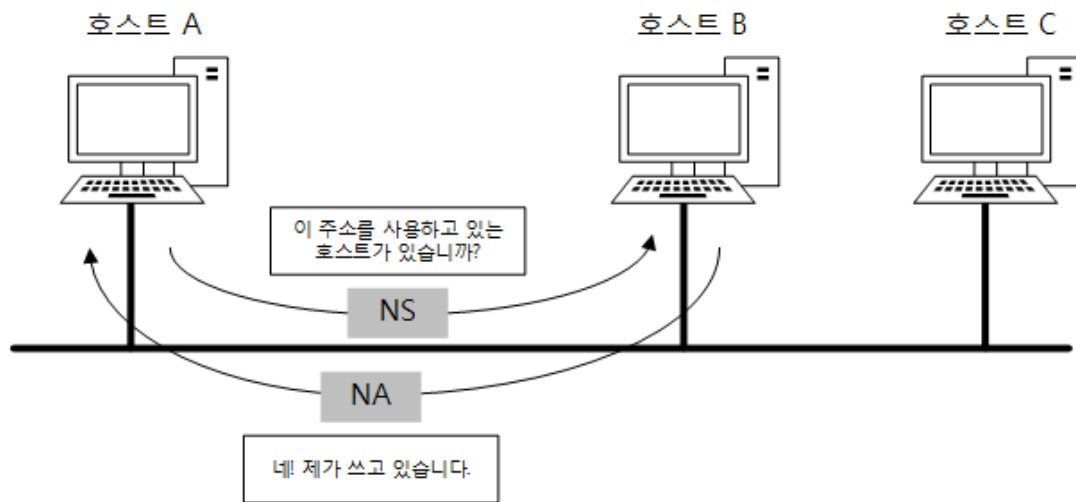


Figure 3 중복주소 검출 과정

DAD는 인터페이스에 주소를 할당하기 전에 충돌여부를 검사하는 과정이며, 링크-로컬 주소를 생성하기 위한 비상태형 주소설정 방식에서는 강제사항이다. 그림3은 이웃 탐색 프로토콜의 NS와 NA 메시지 형식을 이용한 DAD 방법을 나타낸다. Host A에서 임시 주소를 생성한 후 NS 패킷을 망으로 보내게 된다. 이때, 일정시간 동안 NA 패킷을 기다리다가 받지 못할 경우, 이 주소에 대한 유일성이 검증된 것으로 판단하여 인터페이스에 주소를 할당한다. 그러나 다른 Host로부터 NA 패킷을 받았다는 것은 이미 그 주소를 다른 Host가 사용하고 있다는 의미이므로 인터페이스에 그 주소를 할당할 수 없다. 이렇게 주소 할당이 실패하면, 상태형 주소 할당 방식을 이용해서 주소를 획득 해야 한다.

2.1.1 Sudo Code

TargetAddr은 Figure 2와 같이 만들어 입력한다.

```
uint8_t flags;

setSLDIP6R(TargetAddr); // Set IPv6 Link Local Address
setSLRTR(0x2000); // Set Timeout value

setSLIMR(TIMEOUT | DAD_NS); // Timeout(0x80)||DAD_NS(0x04) Interrupt Masking
setSLCR(DAD_NS); //SLCMD_DAD_NS

do
{
    flags = getSLIR();
}while(flags == 0);

setSLIRCLR(0xFF; // SLIR All Clear

if((flags&TIMEOUT)==TIMEOUT) // DAD Success !
{
    printf("DAD Success!!\r\n");
    setLLAR(TargetAddr);
}
else if((flags&DAD_RECV)==DAD_RECV) // DAD Fail !
{
    printf("DAD Failed!!\r\n");
}
```

2.2 Stateless Global Address Auto-Configuration

글로벌 주소를 획득하기 위해서는 Prefix가 이미 정해져 있는 링크-로컬 주소의 경우와는 달리 라우터로부터 Prefix와 같은 주소 설정에 필요한 정보를 획득하여야 한다.

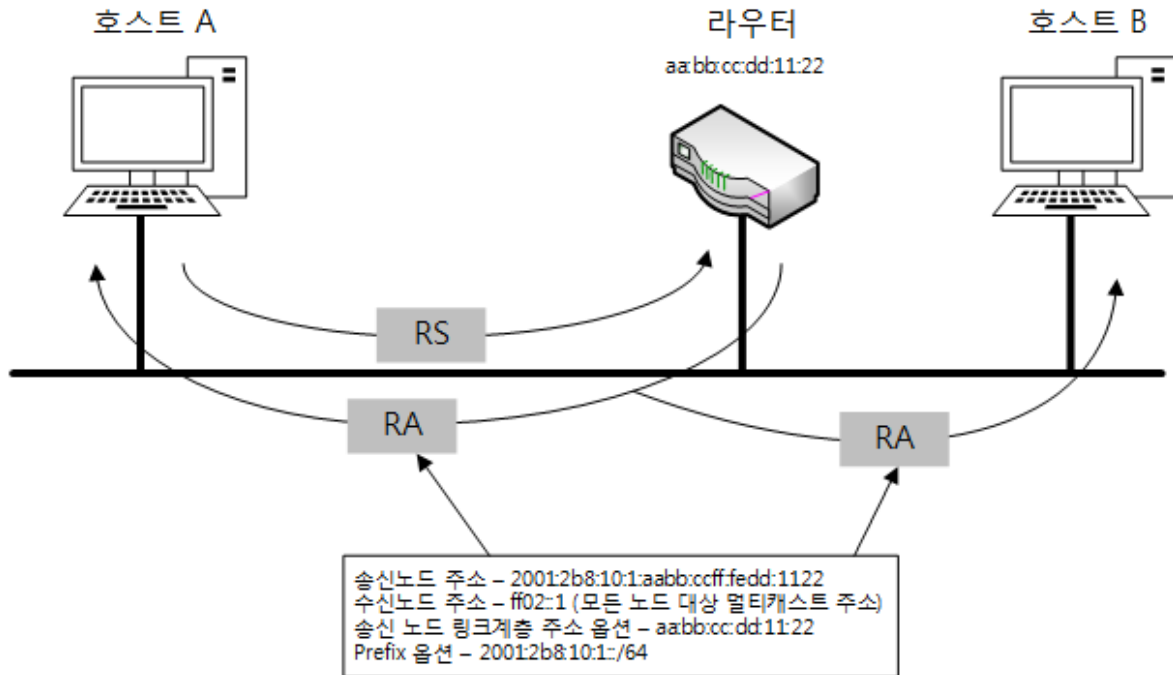


Figure 4 라우터로부터 주소 설정 정보 획득 과정

Figure 4에서 나타낸 것처럼, 라우터에게 이러한 정보를 포함한 RA 패킷을 보내도록 Host 쪽에서 RS 메시지를 보내거나, 라우터가 주기적으로 보내는 RA 패킷에서 획득할 수 있다. 이 RA 패킷에 포함되는 정보는 Prefix 정보, 유효 생존기간(valid lifetime), 선호 생존기간(preferred lifetime)과 각 Host의 NA 메시지들의 재전송 간격등이다. Host는 이렇게 얻어진 Prefix 정보와 자신의 인터페이스 ID를 이용하여 글로벌 주소를 설정한다. 글로벌 주소 생성 방식이 링크-로컬 주소의 생성의 경우와의 차이는 중복 주소 검출 과정을 수행하지 않는다는 것이다. 이는 링크-로컬 주소 생성시 사용된 인터페이스 ID와 동일한 64비트 인터페이스 ID를 사용하는 경우에만 해당된다.

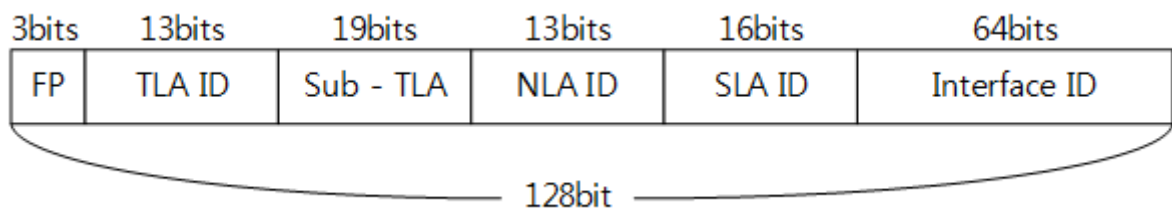


Figure 5 글로벌 주소 형식

주소생성 과정을 좀 더 자세히 살펴보면, Figure 4처럼 먼저 Host A에서 RS 패킷을 모든 라우터의 목적지로 보내면 라우터에서 주소 설정 정보를 포함한 RA패킷을 전송하게 된다. 이때, Ra 패킷의 목적지 주소는 모든 노드 대상의 멀티캐스트 주소인 FE02::1이다. 이는 RS 패킷을 보낸 호스트 A에 아직 정식 주소를 할당되지 못했기 때문에, Host A가 RA 패킷을 받을 수 있도록 모든 노드로 전송해야만 한다. Host A는 RA 메시지의 유효성 등을 검사한 후 자신의 인터페이스 ID와 수신한 Prefix 정보를 이용하여 Figure 5과 같은 주소 구조에 따라 글로벌 주소를 생성하게 된다. 주소 생성 후에도 Host A는 계속해서 RA메시지를 수신하여 자신의 Prefix 정보를 갱신하고 유효 시간 등을 점검해야 한다. 또한, 패킷 전송을 원활하게 수행하기 위해 이웃 노드들에 대한 정보를 보관하여 체계적으로 관리해야 한다.

2.2.1 Sudo Code

Stateless Link-Local Address Auto-Configuration인 DAD 이 후에 해야 한다.

```
DAD_run(LinklocalAddress);

setSLIMR(TIMEOUT&RA_RECV); // interrupt mask
setSLCR(AUTO_RS); //autoconfiguration RS command

do
{
    flags = getSLIR();
    printf(".");
}while(flags == 0);

setSLIRCLR(0xFF); // SLIR Clear

if((flags&Timeout)==Timeout)
{
    printf("Timeout!!\r\n");
}

else if((flags&RA_RECV)==RA_RECV)
{
    printf("Received RA!!\r\n");

    getPLR();          // prefix length
    getPFR();          // ICMPv6 option flag
    getVLTR();         // valid lifetime
    getPLTR();         // preferred lifetime
    getPAR(PREFIX_array); // prefix address
    getLLAR(LinklocalAddress); // link-local address
    //global front 7bit = prfix address
    for(i=0 ; i <8 ; i++) global_addr [i] = PREFIX_array[i];
    //global behind 7bit = link local address
    for(j=8 ; j<16; j++) global_addr[j] = LinklocalAddress [j];

    setGUAR(global_addr);
}
```


3 Stateful Auto-Configuration

Stateful Auto-Configuration에는 Stateless Link-Local Address Auto-Configuration을 한 후 DHCPv6 Server를 이용해 Global Address를 설정하는 방법이 있다.

3.1 Stateful Global Address Auto-Configuration(DHCPv6)

DHCPv6는 IPv6용 DHCP 프로토콜로서 Stateless Auto-Configuration에 대응되는 Stateful Auto-Configuration을 지원한다. DHCP는 IP주소, 라우팅 정보, 운영체제 설치정보, 디렉토리 서비스 정보 등을 소수의 DHCP 서버상에 관리를 집중화 시킴으로써 유지비용을 줄일 수 있도록 하는 메커니즘이다. 이 외에도 DHCP는 새로운 설정 파라미터들을 전달할 수 있도록 정의된 DHCP 옵션을 통해서 쉽게 확장될 수 있도록 설계되었다.

여기서는 DHCPv6의 몇 가지 특징 및 주소 획득 과정만 간단히 설명하도록 하겠다.

- DHCP는 IPv6 Stateless Auto-Configuration과 함께 사용 가능하다.
- DHCP는 보안상의 이유를 제외하고는 클라이언트의 네트워크 파라미터들을 수동으로 구성하지 않는다.
- DHCP는 정적인 설정, DHCP를 사용하지 않는 노드, 그리고 기존의 네트워크 프로토콜 과도 상호 공존할 수 있다.

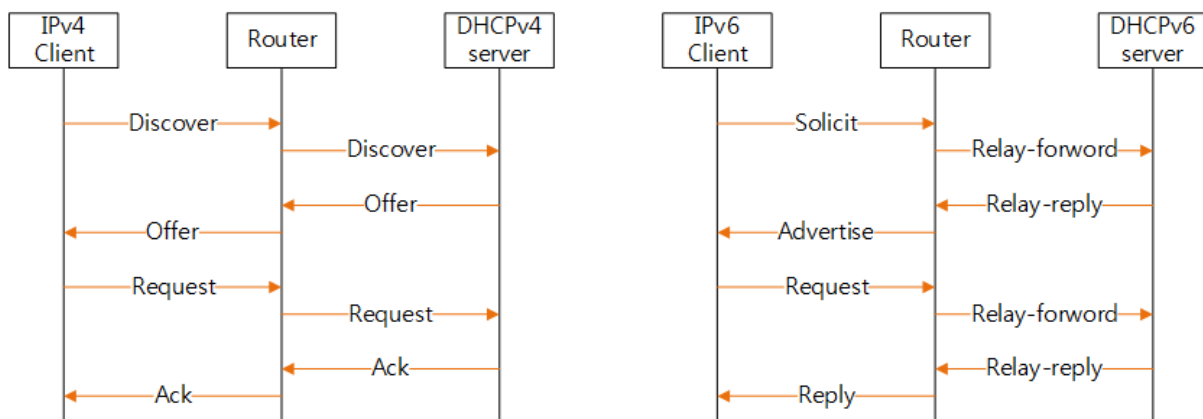


Figure 6 DHCP와 DHCPv6 과정

DHCPv6는 Discover Message 대신 Solicit Message를 보낸다. 하지만 message 내용은 비슷하다. DHCPv6 server가 있는지 찾는 것이다. 따라서 DHCP를 이해하고 있다면 DHCPv6도 어렵지 않게 이해 할 수 있다.

3.1.1 Sudo Code

Stateless Link-Local Address Auto-Configuration을 한 후에 해야 한다.

```
DAD_run(LinkLocalAddress);

DHCP_init(DHCP_SOCKET, test_buf);

while(1)
{
    switch(DHCP_run())
    {
        case DHCP_IP_ASSIGN:
        case DHCP_IP_CHANGED:
            check_flag = 1;
            if(check_flag)
            {
                check_flag = 0;
                close(TCP_SOCKET);
            }
            break;
        case DHCP_IP_LEASED:
            if(check_flag)
            {
                check_flag = 0;
            }

            // TO DO YOUR NETWORK APPs.
            loopback_tcps(TCP_SOCKET, TCP_PORT, test_buf, AF_INET);
            break;
        case DHCP_FAILED:
            my_dhcp_retry++;
            if(my_dhcp_retry > MY_MAX_DHCP_RETRY)
            {
                my_dhcp_retry = 0;
                DHCP_stop();    // if restart, recall DHCP_init()
            }
            break;
        default:
            break;
    }
}
```

4 Document History Information

Version	Date	Descriptions
Ver. 1.0.0	Dec.2025	Release

Copyright Notice

Copyright 2025 WIZnet Co., Ltd. All Rights Reserved.

Technical support : <https://maker.wiznet.io/forum>

Sales & Distribution: sales@wiznet.io

For more information, visit our website at <http://www.wiznet.io>