1. Source code

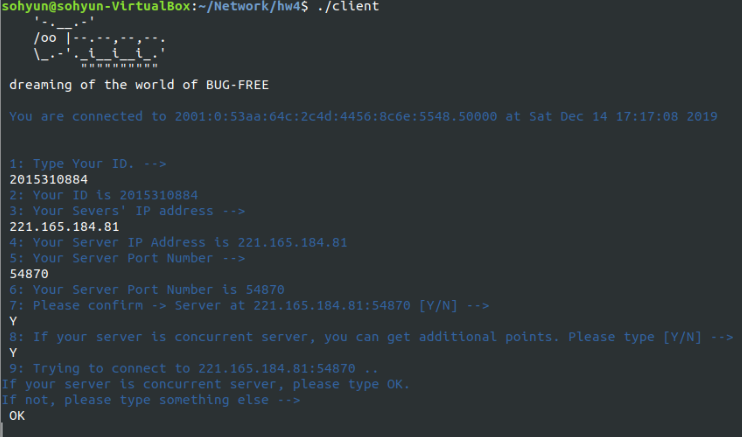
|  |
| --- |
| firstClient.c |
| ﻿#include <stdio.h>  #include <stdlib.h>  #include <string.h>  #include <unistd.h>  #include <arpa/inet.h>  #include <sys/socket.h>  #define BUF\_SIZE 2000  void error\_handling(char \*message);  int main()  {  int sock;  struct sockaddr\_in6 serv\_addr; //IPv6 socket 구조체  char message[BUF\_SIZE];    int str\_len;  char serv\_ipaddr[INET6\_ADDRSTRLEN];  int serv\_port;  //socket open. 서버로 연결할 소켓 생성  sock = socket(AF\_INET6, SOCK\_STREAM, 0);  if(sock == -1)  error\_handling("sock() error");    //IPv6용 구조체에 주소 정보 할당  memset(&serv\_addr, 0, sizeof(serv\_addr));  serv\_addr.sin6\_family = AF\_INET6;  serv\_addr.sin6\_flowinfo = 0;  char \*ip = "2001:0:53aa:64c:2c4d:4456:8c6e:5548";  inet\_pton(AF\_INET6, ip, &serv\_addr.sin6\_addr); //binary 형태의 주소로 변환  serv\_addr.sin6\_port = htons(50000);  //connect  if(connect(sock, (struct sockaddr\*)&serv\_addr, sizeof(serv\_addr)) == -1)  error\_handling("connect() error!");  //data for sending  char \*id = "2015310884\n";  char \*address = "221.165.184.81\n";  char \*port = "54870\n";  char \*yes = "Y\n";  char \*concurrent = "Y\n"; //Y //N  char \*connectYes = "OK\n"; //OK //YES  //처음에 빈 값 받기  char buf[BUF\_SIZE];  memset(buf, 0, BUF\_SIZE);  recv(sock, buf, BUF\_SIZE, 0);    //bug picture  memset(buf, 0, BUF\_SIZE);  recv(sock, buf, BUF\_SIZE, 0);  printf("%s %s", buf, id);  //id 보내기  send(sock, id, strlen(id), 0);  //response  memset(buf, 0, BUF\_SIZE);  recv(sock, buf, BUF\_SIZE, 0);  printf("%s", buf);  memset(buf, 0, BUF\_SIZE);  recv(sock, buf, BUF\_SIZE, 0);  printf("%s %s", buf, address);  //ip address 보내기  send(sock, address, strlen(address), 0);  //response  memset(buf, 0, BUF\_SIZE);  recv(sock, buf, BUF\_SIZE, 0);  printf("%s", buf);  memset(buf, 0, BUF\_SIZE);  recv(sock, buf, BUF\_SIZE, 0);  printf("%s %s", buf, port);  //port num 보내기  send(sock, port, strlen(port), 0);  //response  memset(buf, 0, BUF\_SIZE);  recv(sock, buf, BUF\_SIZE, 0);  printf("%s", buf);  memset(buf, 0, BUF\_SIZE);  recv(sock, buf, BUF\_SIZE, 0);  printf("%s %s", buf, yes);  //Y 보내기  send(sock, yes, strlen(yes), 0);  //response  memset(buf, 0, BUF\_SIZE);  recv(sock, buf, BUF\_SIZE, 0);  printf("%s", buf);  memset(buf, 0, BUF\_SIZE);  recv(sock, buf, BUF\_SIZE, 0);  printf("%s %s", buf, concurrent);  //concurrent OK 보내기  send(sock, concurrent, strlen(concurrent), 0);  //response  memset(buf, 0, BUF\_SIZE);  recv(sock, buf, BUF\_SIZE, 0);  printf("%s", buf);  memset(buf, 0, BUF\_SIZE);  str\_len = recv(sock, buf, BUF\_SIZE, 0);  printf("%s %s", buf, connectYes);  //connect ok 보내기  send(sock, connectYes, strlen(connectYes), 0);  //response  memset(buf, 0, BUF\_SIZE);  while(recv(sock, buf, BUF\_SIZE, 0) != 0)  {  printf("%s", buf);  memset(buf, 0, BUF\_SIZE);  }    //memset(buf, 0, BUF\_SIZE);  //recv(sock, buf, BUF\_SIZE, 0);  //printf("%s", buf);    close(sock);  return 0;  }  void error\_handling(char \*message)  {  fputs(message, stderr);  fputc('\n', stderr);  exit(1);  } |

|  |
| --- |
| secondServer.c |
| ﻿#include <stdio.h>  #include <stdlib.h>  #include <string.h>  #include <unistd.h>  #include <signal.h>  #include <sys/wait.h>  #include <arpa/inet.h>  #include <sys/socket.h>  #define BUF\_SIZE 20  void error\_handling(char \*message);  void remove\_zombie(int sig);  int main()  {  int serv\_sock, clnt\_sock;  struct sockaddr\_in serv\_adr, clnt\_adr;  pid\_t pid;  struct sigaction act;  int state;  socklen\_t clnt\_adr\_sz;  int str\_len;  char buf[BUF\_SIZE];  //signal handling 등록  act.sa\_handler = remove\_zombie;  sigemptyset(&act.sa\_mask);  act.sa\_flags = 0;  state = sigaction(SIGCHLD, &act, 0);  //시작 알림  printf("secondServer start!\n");  //socket 생성  serv\_sock = socket(PF\_INET, SOCK\_STREAM, 0);  memset(&serv\_adr, 0, sizeof(serv\_adr));  serv\_adr.sin\_family = AF\_INET;  serv\_adr.sin\_addr.s\_addr = htonl(INADDR\_ANY);  serv\_adr.sin\_port = htons(54870);  if(bind(serv\_sock, (struct sockaddr\*)&serv\_adr, sizeof(serv\_adr)) == -1)  error\_handling("bind() error");  if(listen(serv\_sock, 5) == -1)  error\_handling("listen() error");  while(1)  {  clnt\_adr\_sz = sizeof(clnt\_adr);  clnt\_sock = accept(serv\_sock, (struct sockaddr\*)&clnt\_adr, &clnt\_adr\_sz);  if(clnt\_sock==-1)  {  puts("no new client");  continue;  }  else  {  puts("new client connected!");  }    pid = fork();  if(pid == -1)  {  close(clnt\_sock);  puts("fork failed");  continue;  }  if(pid == 0) //자식 프로세스  {  close(serv\_sock);    //receive token  memset(buf, 0, BUF\_SIZE);  read(clnt\_sock, buf, BUF\_SIZE);    fputs(buf, stdout);  //send token to secondClient  int newSock;  struct sockaddr\_in newServ\_adr;    newSock = socket(PF\_INET, SOCK\_STREAM, 0);  if(newSock == -1)  error\_handling("newSocket error");  memset(&newServ\_adr, 0, sizeof(newServ\_adr));  newServ\_adr.sin\_family=AF\_INET;  newServ\_adr.sin\_addr.s\_addr=inet\_addr("192.168.0.18");  newServ\_adr.sin\_port=htons(54877);  //connect to new  if(connect(newSock, (struct sockaddr\*)&newServ\_adr, sizeof(newServ\_adr)) == -1)  error\_handling("connect() error!");  char buf2[BUF\_SIZE];  memset(buf2, 0, BUF\_SIZE);    for(int i=0; i<20; i++)  {  buf2[i] = buf[i];  }  buf2[20] = '\n';    send(newSock, buf2, strlen(buf2), 0);  puts("sended token");  fputs(buf2, stdout);  //close  close(newSock);    //memset(buf, 0, sizeof(buf));  //while( (str\_len=read(clnt\_sock, buf, BUF\_SIZE)) != 0 )  //{  // fputs(buf, stdout);  // memset(buf, 0, sizeof(buf));  //}    close(clnt\_sock);  puts("client disconnected!");  return 0;  }  else  {  close(clnt\_sock);  }  }  close(serv\_sock);  return 0;  }  void remove\_zombie(int sig)  {  pid\_t pid;  int status;  pid = waitpid(-1, &status, WNOHANG);  printf("removed pid: %d\n", pid);  }  void error\_handling(char \*message)  {  fputs(message, stderr);  fputc('\n', stderr);  exit(1);  } |

|  |
| --- |
| secondClient.c |
| ﻿#include <stdio.h>  #include <stdlib.h>  #include <string.h>  #include <unistd.h>  #include <signal.h>  #include <sys/wait.h>  #include <arpa/inet.h>  #include <sys/socket.h>  #define BUF\_SIZE 20  void error\_handling(char \*message);  void remove\_zombie(int sig);  int main()  {  int serv\_sock, clnt\_sock;  struct sockaddr\_in serv\_adr, clnt\_adr;  pid\_t pid;  struct sigaction act;  int state;  socklen\_t clnt\_adr\_sz;  int str\_len;  char buf[BUF\_SIZE];  //signal handling 등록  act.sa\_handler = remove\_zombie;  sigemptyset(&act.sa\_mask);  act.sa\_flags = 0;  state = sigaction(SIGCHLD, &act, 0);  //시작 알림  printf("start!\n");  //socket 생성  serv\_sock = socket(PF\_INET, SOCK\_STREAM, 0);  memset(&serv\_adr, 0, sizeof(serv\_adr));  serv\_adr.sin\_family = AF\_INET;  serv\_adr.sin\_addr.s\_addr = htonl(INADDR\_ANY);  serv\_adr.sin\_port = htons(54877);  if(bind(serv\_sock, (struct sockaddr\*)&serv\_adr, sizeof(serv\_adr)) == -1)  error\_handling("bind() error");  if(listen(serv\_sock, 5) == -1)  error\_handling("listen() error");  while(1)  {  clnt\_adr\_sz = sizeof(clnt\_adr);  clnt\_sock = accept(serv\_sock, (struct sockaddr\*)&clnt\_adr, &clnt\_adr\_sz);  if(clnt\_sock==-1)  {  puts("no new client");  continue;  }  else  {  puts("new client connected!");  }    pid = fork();  if(pid == -1)  {  close(clnt\_sock);  puts("fork failed");  continue;  }  if(pid == 0) //자식 프로세스  {  close(serv\_sock);    //receive token  memset(buf, 0, BUF\_SIZE);  //read(clnt\_sock, buf, BUF\_SIZE);  puts("hey5");  while(read(clnt\_sock, buf, BUF\_SIZE)!=0) ////  fputs(buf, stdout);  //send token to lab  int newSock;  struct sockaddr\_in6 newServ\_addr; //IPv6 socket 구조체  //connect to new server  newSock = socket(AF\_INET6, SOCK\_STREAM, 0);  if(newSock == -1)  error\_handling("sock() error");    //IPv6용 구조체에 주소 정보 할당  memset(&newServ\_addr, 0, sizeof(newServ\_addr));  newServ\_addr.sin6\_family = AF\_INET6;  newServ\_addr.sin6\_flowinfo = 0;  char \*labIp = "2001:0:53aa:64c:2c4d:4456:8c6e:5548";  inet\_pton(AF\_INET6, labIp, &newServ\_addr.sin6\_addr); //binary 형태의 주소로 변환  newServ\_addr.sin6\_port = htons(50000);    //send  char buf2[BUF\_SIZE];  memset(buf2, 0, BUF\_SIZE);  strcpy(buf2, buf);  send(newSock, buf2, BUF\_SIZE, 0);  puts("sended token");  fputs(buf2, stdout);  //close  close(newSock);    //memset(buf, 0, sizeof(buf));  //while( (str\_len=read(clnt\_sock, buf, BUF\_SIZE)) != 0 )  //{  // fputs(buf, stdout);  // memset(buf, 0, sizeof(buf));  //}    close(clnt\_sock);  puts("client disconnected!");  return 0;  }  else  {  close(clnt\_sock);  }  }  close(serv\_sock);  return 0;  }  void remove\_zombie(int sig)  {  pid\_t pid;  int status;  pid = waitpid(-1, &status, WNOHANG);  printf("removed pid: %d\n", pid);  }  void error\_handling(char \*message)  {  fputs(message, stderr);  fputc('\n', stderr);  exit(1);  } |

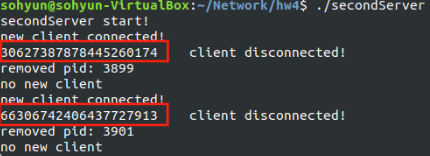
2. Screenshot

>First Stage



>20-byte string received from the server

-Multi Process를 사용하여Concurrent 하게 구현했기에, 2개의 token을 받았다.



3. Setup to connect to the Ipv6 server

|  |
| --- |
| ﻿ ﻿int sock;  struct sockaddr\_in6 serv\_addr; //IPv6 socket 구조체  //서버로 연결할 소켓 생성  sock = socket(AF\_INET6, SOCK\_STREAM, 0);  if(sock == -1)  error\_handling("sock() error");    //IPv6용 구조체에 주소 정보 할당  memset(&serv\_addr, 0, sizeof(serv\_addr));  serv\_addr.sin6\_family = AF\_INET6;  serv\_addr.sin6\_flowinfo = 0;  char \*ip = "2001:0:53aa:64c:2c4d:4456:8c6e:5548";  inet\_pton(AF\_INET6, ip, &serv\_addr.sin6\_addr); //binary 형태의 주소로 변환  serv\_addr.sin6\_port = htons(50000);  //connect  if(connect(sock, (struct sockaddr\*)&serv\_addr, sizeof(serv\_addr)) == -1)  error\_handling("connect() error!"); |

4. Screenshot of Successful conversation

-Token을 Second Stage Server'(Client)에 보내기까지는 했지만, 마지막 단계에서

연구실 Client로 Token을 보내는 데 실패했다. 그래서 Successful screenshot이 없다.

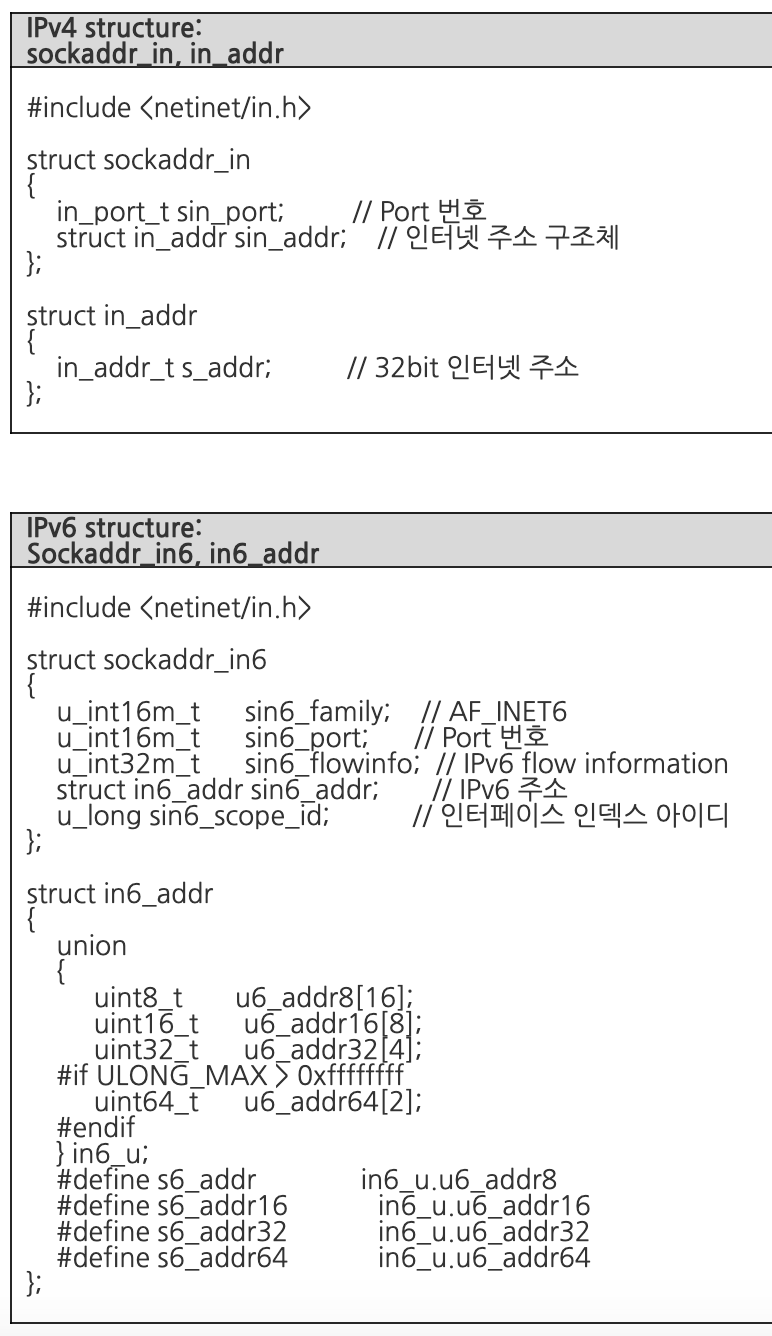
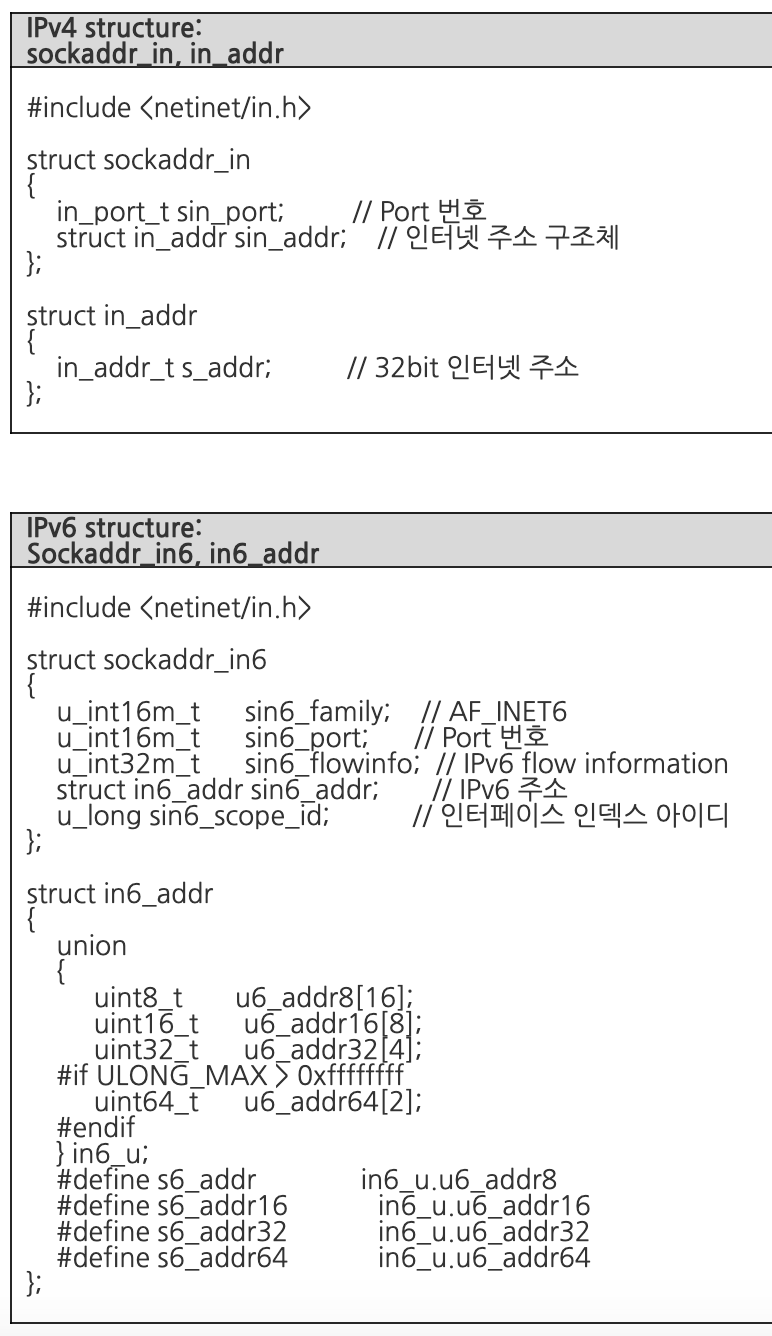
5. Unique Experience

(1) Miredo

-우분투를 껐다 킬 때마다, miredo를 remove해주고, 다시 install해줘야 한다는 것을 몰랐다. 그래서 Ipv6 서버에 connect 실패 원인을 계속 다른 데서 찾았었다.

(2) Ipv6

Ipv6통신을 위한 초기 설정에서 sockaddr\_in6 구조체를 사용해봄으로써 Ipv4에서 사용하는 sockaddr\_in 과의 차이점을 살펴볼 수 있었다.



(3) 학번 정확하게 보내기

﻿send(sock, id, strlen(id), 0);

sizeof(id)로 보내면 버퍼에 남아있는 값들도 같이 보내게 되므로, 정확하게 보낸 것이 아니게 된다.

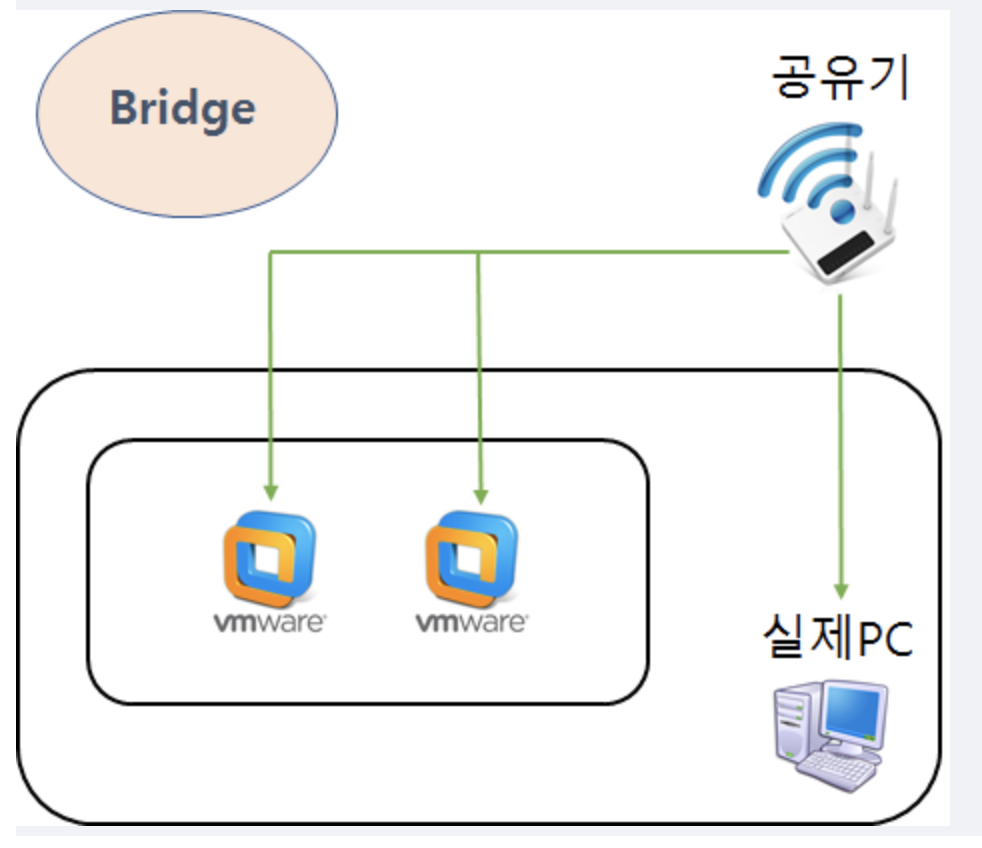
(4) 포트포워딩 설정하기

-virtualBox에서 돌리고 있는 나의 서버를 외부에서도 접속할 수 있도록 허용하기 위해서는 먼저, virtualBox의 네트워크 모드를 확인해보아야한다. Host-only, NAT, Bridged 모드가 있는데, 각각을 알아보았더니 다음과 같은 특징이 있었다.

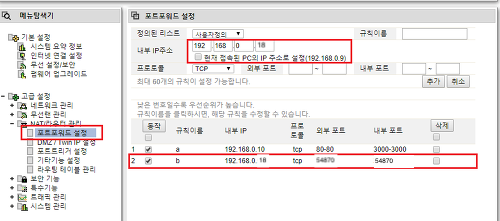
Host-only: VM들끼리만 통신한다. VM은 실제PC (호스트 컴퓨터)를 넘어서 통신할 수 없다. 인터넷 연결이 필요하지 않고, 소프트웨어 테스트를 위해 사용하는 경우 사용한다.

NAT : Virtual Adapter로 vmnet을 사용한다. 공유기가 실제 PC에게 IP를 준다. 그리고 실제 PC가 VM에게 IP를 주면서, VM이 자체적으로 내부 네트워크 대역 할당 및 자체 DHCP 서버를 띄워 통신한다.

Bridge : 공유기가 실제 PC에게 IP를 줄 뿐만 아니라, 공유기가 VM에게도 IP를 준다. 공유기는 VM도 별개의 실제 PC로 보기 때문에 개별적으로 IP를 할당 한다. 즉, VM은 실제 PC가 물리적 네트워크에서 동작하는 것처럼 동작한다.



-나의 virtualBox는 Bridge 모드로 설정돼있었다. 그래서 iptime공유기 설정에 들어가서 포트포워딩 설정에 규칙을 다음과 같은 방식으로 추가해주어야했다.



-이제 외부에서 내 virtualBox에서 돌아가는 서버에 접속 가능하게 되었다.

(5) Multi Process 구현하기 중 좀비 프로세스 처리하기

-프로세스에서 return하게 되면, 자동으로 자원 정리를 해줄 수 있는 함수를 등록하였다. 시그널 핸들링이라고도 하는데, 이 시그널 핸들링에 remove\_zombie 함수를 저장해두었다.

﻿void remove\_zombie(int sig)

{

pid\_t pid;

int status;

pid = waitpid(-1, &status, WNOHANG);

printf("removed pid: %d\n", pid);

}

-waitpid 함수는 인수로 주어진 pid 번호의 자식프로세스가 종료되거나, 시그널 함수를 호출하는 신호가 전달될 때까지 waitpid 호출한 영역에서 일시 중지 된다.