J. Internet Comput. Serv.
ISSN 1598-0170 (Print) / ISSN 2287-1136 (Online)
http://www.jics.or.kr

OMNeT++의 INET 프레임워크를 이용한 네트워크 시뮬레이션 가이드☆

A Guide to Network Simulation Using the INET Framework for OMNeT++

박 수 연[1] 김 문 성[1*]
Sooyeon Park Moonseong Kim

요 약

INET은 OMNeT++ 시뮬레이터의 모듈을 활용하여 유·무선 및 모바일 네트워크 등의 시뮬레이션을 위한 다양한 프로토콜, 에이전트, 기타 모델을 제공하는 프레임워크이다. INET의 모듈을 조합하여 네트워크를 구성하고, 그 모듈 간 메시지 송수신을 통해 네트워크 구성 요소 간의 상호작용을 구현할 수 있다. 연구자는 필요에 따라 C++ 언어를 사용해 새로운 모듈을 구현하거나 기존 모듈을 수정하여 네트워크 구성에 적용할 수 있다. 본 논문에서는 간단한 네트워크를 생성하여 데이터를 송수신하는 시뮬레이션을 통해 INET의 모듈을 활용한 네트워크 시뮬레이션 방법을 설명한다. 이를 통해 연구자가 INET 프레임워크의 다양한 모델을 활용하여 필요한 네트워크를 구성하고 그 성능을 실험하는 데 도움을 주고자 한다.

☞ 주제어 : OMNet++ (Objective Modular Network Testbed in C++), INET 프레임워크, 시뮬레이션

ABSTRACT

INET is a framework that utilizes the modules of the OMNeT++ simulator to provide various protocols, agents, and other models for simulating wired, wireless, and mobile networks. By combining the modules of INET, we can configure a network and implement interactions between network components through message transmission and reception among the modules. Researchers can also implement new modules or modify existing ones using the C++ language as needed and apply them to the network configuration. This paper explains the method of network simulation using INET modules by providing a simple network and simulating data transmission and reception. Through this, researchers can utilize the various models of the INET framework to configure the necessary network and experiment its performance.

☞ keyword : OMNet++ (Objective Modular Network Testbed in C++), INET Framework, Simulation

1. 서 론

네트워크 시뮬레이션은 통신 시스템의 설계 및 분석에 있어 주요한 도구로 자리 잡고 있다. 실제 환경에서 네트워크를 구축하고 테스트하는 과정은 많은 비용이 발생할 수 있으며, 다양한 변수를 고려해야 하기 때문에 매우 복잡하다. 이러한 문제를 해결하기 위해 연구자들은 네트워크 시뮬레이션을 활용하여 비용과 시간을 절약할 수 있을 뿐만 아니라, 다양한 시나리오를 테스트하여 네트워크 구성을 사전에 계획할 수 있다. 특히 네트워크의 복잡성과 다양성이 증가함에 따라, 초기 설계 단계에서 시뮬레이션을 통한 검증의 중요성이 커지고 있다.

INET 프레임워크(이하 INET)는 OMNeT++ 시뮬레이터에서 제공하는 모듈을 활용하여 네트워크 시뮬레이션을 위한 다양한 프로토콜, 에이전트 및 기타 모델을 제공한다. INET은 오픈 소스 통신 네트워크 시뮬레이션 패키지로, 연구자가 새로운 프로토콜을 설계하고 검증할 수 있는 유연하고 확장 가능한 프레임워크를 제공한다[1-3].

INET은 네트워크 프로토콜, 장치, 애플리케이션, 유틸리티, 환경 모델 및 네트워크 인터페이스를 지원하는 광범위한 라이브러리를 제공한다. 연구자는 이 라이브러리를 활용하여 다양한 네트워크 시나리오를 시뮬레이션할 수 있으며, 필요에 따라 C++ 언어를 사용해 기존 모듈을 수정하거나 새로운 프로토콜을 구현할 수 있다.

또한, INET은 특정 응용 분야에 특화된 여러 확장 모

1 Dept. of IT Convergence Software, Seoul Theological University, Bucheon, 14754, Korea.
* Corresponding author (moonseong@stu.ac.kr)
[Received 25 September 2024, Reviewed 26 September 2024, Accepted 2 October 2024]
☆ This work was supported by the Seoul Theological University Research Fund of 2024.

델을 통해 그 활용 범위가 넓어졌다. 예를 들어, Vines는 차량 네트워크 시뮬레이션을 위해 설계된 모델이며, CoRE4INET(Communication over Realtime Ethernet 4 INET)은 실시간 이더넷 통신을 위한 확장 모델이다. OverSim(Overlay Network Simulation)은 P2P(Peer-to-Peer) 네트워크 시뮬레이션을, LTE(Long Term Evolution) 모듈은 4G/5G 네트워크 등에 대한 시뮬레이션을 지원한다.

본 논문에서는 OMNeT++ 시뮬레이터의 INET에 대해 개략적으로 설명하고, 대표적인 INET 모델에 대해 소개한다. 이어서 간단한 예제를 통해 INET을 활용한 시뮬레이션 과정을 보여준다. 이를 통해 연구자들은 네트워크 시뮬레이션을 효과적으로 활용할 수 있으며, 다양한 네트워크 환경에서 발생할 수 있는 문제를 탐구하고 해결하는 데 유용한 도구로 사용할 수 있을 것이다.

본 논문은 다음과 같이 구성된다. 2장에서는 INET 프레임워크에 대해 소개하고, 3장에서는 INET 모델에 대해 설명한다. 그리고 4장에서는 INET 모델을 이용한 유선 네트워크 시뮬레이션 방법을 기술하며, 마지막으로 5장에서 결론을 맺는다.

2. INET 프레임워크 소개

INET 프레임워크는 OMNeT++ 시뮬레이션 환경에서 사용되는 오픈 소스 모델 라이브러리로, 다양한 프로토콜, 에이전트, 기타 모델을 모듈 형태로 제공한다. 이를 통해 새로운 프로토콜을 설계하고 검증할 수 있으며, 특히 새로운 시나리오를 탐색하거나 프로토콜 스택의 모든 계층에서 새로운 통신 프로토콜을 설계하고 검증하는 데 유용하다.

초기 INET은 표준 통신 프로토콜 라이브러리를 포함하여 IPv4, TCP, UDP와 같은 인터넷 프로토콜 스택을 지원하였으며, 현재는 유·무선, 모바일, 애드혹, 센서 네트워크, 링크 및 무선 물리 계층 프로토콜 등 다양한 응용 모델과 다양한 프로토콜 및 구성 요소들을 지원한다. 또한 노드 이동성, 시각화, 네트워크 에뮬레이션 등을 지원하며, 다른 시뮬레이션 프레임워크의 기반으로도 활용되고 있다[3].

INET 모듈은 OMNeT++에서 제공하는 기본 모듈(simple module)과 상기 기본 모듈들로 구성된 복합 모듈(compound module)로 이루어지며, 이러한 모듈들 간의 통신은 메시지 송수신을 통해 이루어진다. 상기 기본 모듈은 cSimpleModule을 사용하여 C++로 구현되며, 복합 모듈의 심플 모듈들을 서브 모듈들(submodules)이라고 부른다. INET의 주요 구성요소에는 에이전트와 네트워크 프로토콜이 포함되며, 이들을 활용해 다양한 네트워크 시뮬레이션을 구현할 수 있다.

참고로, INET 모듈에 대한 디렉토리 구조는 다음과 같다[4].

- src/inet/applications/ : 트래픽 생성기 및 애플리케이션 모델
- src/inet/transportLayer/ : 전송 계층 프로토콜
- src/inet/networklayer/ : 네트워크 계층 프로토콜 및 액세서리
- src/inet/linklayer/ : 링크 계층 프로토콜 및 액세서리
- src/inet/physicallayer/ : 물리 계층 모델
- src/inet/routing/ : 라우팅 프로토콜(인터넷 및 애드혹)
- src/inet/mobility/ : 이동성 모델
- src/inet/power/ : 에너지 소비 모델링
- src/inet/environment/ : 물리적 환경 모델
- src/inet/node/ : 미리 구성된 네트워크 노드 모델
- src/inet/visualizer/ : 시각화 구성 요소(2D 및 3D)
- src/inet/common/ : 기타 유틸리티 구성 요소

이러한 모듈들은 C++ 소스 코드로 작성되어 있으며, 시뮬레이션이 실행될 때 NED(Network Description)에 정의된 시뮬레이션 환경에서 각 모듈의 동작이 실행된다.

3. INET Model 구성요소

본 장에서는 INET 프레임워크의 모델 중 표 1과 같이 inet.applications, inet.transportlayer, inet.networklayer, inet.routing, inet.linklayer, inet.node 패키지에 있는 몇 개의 프로토콜과 4장 시뮬레이션을 위해 사용한 모듈에 관해 상술한다.

트래픽을 생성하는 애플리케이션 모듈은 inet.application 패키지에 있다. UDPBasicApp, TCPBasicClientApp, IPTrafGen, EtherAppClient, EtherAppSrver는 CBR(Constant Bit Rate)/VBR (Variable Bit Rate) 트래픽을 생성하는 모듈이다. 이 트래픽 모듈들은 packetInterval 또는 상수 및 exponential(1s)과 같은 동등한 파라미터 값에 따라 CBR 또는 VBR 트래픽을 생성할 수 있다.

파일 전송은 TcpSessionApp모듈을 이용하여 모델링 할 수 있다. TcpSessionApp은 단일 연결 TCP 애플리케이션으로 일정한 바이트 수를 전송 후 연결을 종료한다. 전송은 (시간, 바이트 수) 쌍으로 제어할 수 있으며, IPv4와 IPv6 모두와 호환된다. DHCP(Dynamic Host Configuration Protocol)는 DhcpServer와 DhcpClient 모듈을 이용하여 구현할 수 있다. DhcpClient는 DHCP 클라이언트 프로토콜

(표 1) INET 모듈 [5]
(Table 1) INET Module [5]

프로토콜	모듈
inet.applications	
CBR/VBR	UDPBasicApp, TCPBasicClientApp, IPTrafGen, EtherAppClient, EtherAppSrver
파일전송	TCPSessionApp
DHCP	DhcpServer, DhcpClient
비디오	UDPVideoStreamClient, UDPVideoStreamServer
inet.transportlayer	
TCP	Tcp
UDP	Udp
SCTP	Sctp
inet.networklayer	
IPv4	Ipv4, Ipv4RoutingTable
ARP	Arp
IPv6	Ipv6
inet.routing	
link-state routing	inet.networklayer.ted.LinkStateRouting
OSPF	Ospfv2
BGP	Bgp
RIP	Rip
inet.linklayer	
PPP	Ppp
CSMA/CA	CsmaCaMac
inet.node	
이더넷 링크	Eth10M, Eth100M, Eth100G

을 구현하고 DhcpServer는 DHCP 서버 프로토콜을 구현하는 모듈로 UDP를 필요로 한다. 단순한 비디오 스트리밍은 UDPVideoStreamClient와 UDPVideoStreamServer 모듈을 이용한다. UdpVideoStreamClient는 비디오 스트리밍 클라이언트 모듈로 비디오 스트리밍 요청을 UdpVideoStreamServer 모듈에 전송한다. UdpVideoStreamServer는 비디오 스트리밍 서버 모듈로 비디오 스트리밍 요청을 수신하면 스트리밍을 시작한다.

inet.transportlayer 패키지에 있는 Tcp모듈은 TCP(Transmission Control Protocol) 프로토콜을 구현한 모듈이고, Udp 모듈은 UDP(User Datagram Protocol) 프로토콜을 구현한 모듈로 Tcp모듈과 Udp모듈은 IPv4와 IPv6 모두 지원한다. Sctp 모듈은 SCTP(Stream Control Transmission Protocol) 프로토콜을 구현한 모듈이다.

IPv4(Internet Protocol v4) 구현은 inet.networklayer.ipv4 패키지의 Ipv4 및 Ipv4RoutingTable 모듈로 구성된다. IPv4는 IPv4 프로토콜을 구현한 모듈이고 RoutingTable 모듈은 라우팅 테이블을 저장한다. inet.networklayer.arp.ipv4 패키지의 Arp 모듈은 IPv4와 IEEE 802의 6바이트 MAC 주소를 위한 ARP(Address Resolution Protocol)를 구현한 모듈이다. IPv6(Internet Protocol v6) 프로토콜은 inet.networklayer.ipv6 패키지의 Ipv6모듈로 구현되어 있다.

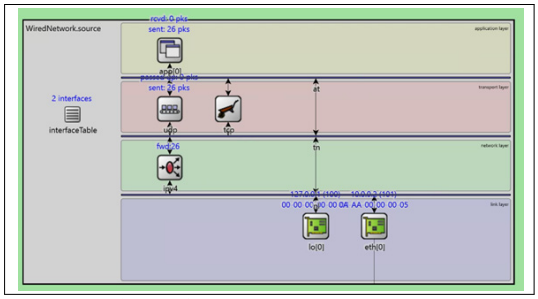
inet.networklayer.ted 패키지의 LinkStateRouting 모듈은 최소한의 링크 상태 라우팅 프로토콜을 구현한 모듈이다. 이 모듈은 기본적인 토폴로지 정보 외에도 현재 링크 사용 상태를 네트워크의 모든 호스트에게 플러딩을 통해 배포한다.

inet.routing 패키지의 Ospfv2 모듈은 OSPFv2(Open Shortest Path First) 라우팅 프로토콜을 구현한 모듈이고, Bgp 모듈은 BGPv4(Border Gateway Protocol) 라우팅 프로토콜을 구현한 모듈이다. Rip(Routing Information Protocol) 모듈은 RFC 2453 (RIPv2)과 RFC 2080 (RIPng)에 명시된 거리 벡터 라우팅을 구현한 모듈로 벨만-포드 알고리즘을 사용하여 최적 경로를 계산하고 라우터는 주기적으로 자신의 경로정보를 이웃 라우터에게 전송한다.

inet.linklayer 패키지에 Ppp 모듈은 PPP(Point-to-Point Protocol)를 구현한 모듈이다. PPP는 링크를 구성하고 유지 관리를 하는 복잡한 프로토콜로 Ppp 모듈은 캡슐화와 캡슐화 해제 및 큐잉을 수행한다.

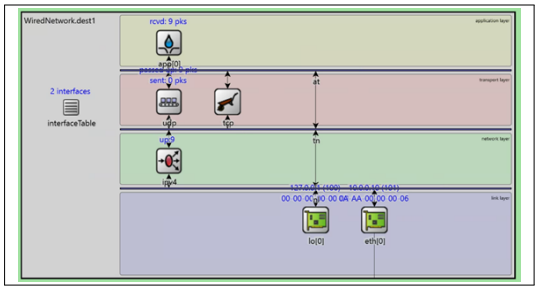
INET은 이더넷 네트워크를 시뮬레이션하기 위한 모듈 세트를 포함하고 있다. 이더넷 링크 Eth10M(10Mbps), 패스트 이더넷 Eth100M(100Mbps), 기가비트 이더넷 Eth100G(100Gbps) 모듈이 구현되어 있다. 또한 MAC 프로토콜을 구현한 CsmaCaMac 모듈이 있다. 이 모듈은 가상의 CSMA/CA(Carrier-Sense Multiple Access/Collision Avoidance) 기반의 MAC(Media Access Control) 프로토콜을 구현한 모듈로 선택적으로 확인 응답과 재시도 메커니즘을 포함하고 있다.

지금까지 패키지별 모듈들에 대해 간단히 기술하였다. 다음은 4장의 시뮬레이션을 이해하기 위해 사용된 모듈들에 대한 사전 정보를 전달한다. 먼저 StandardHost 모듈은 inet.node.inet 패키지에 있는 모듈로 SCTP, TCP, UDP 레이어를 가지고 있는 IPv4 호스트이다. 게이트 ethg[]를 이용하여 다른 노드와 이더넷 인터페이스를 통해 연결된다. 기본적으로 full-duplex 연결을 지원한다. 이더넷 인터페이스의 개수는 매개변수 numEthInterfaces를 이용하여 설정한다[6].



(그림 1) Qtenv에서 소스 호스트
(Figure 1) Source host in Qtenv

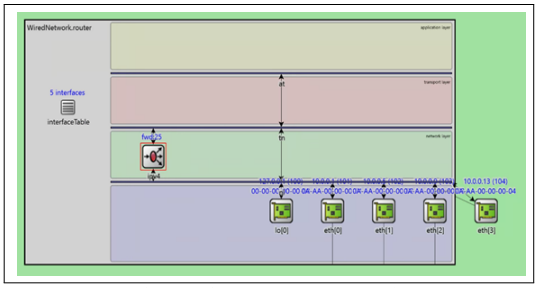
다음 장의 시뮬레이션에서는 1개의 소스 호스트와 3개의 목적지 호스트를 고려하였다. 그림 1은 Qtenv에서 StandardHost 인스턴스인 소스 호스트의 그래픽 표현이다. 소스 호스트의 application layer에는 1개의 Udp 어플리케이션이 있다. 이 어플리케이션이 패킷을 생성 후 transport layer에 있는 udp로 전송한다. Udp는 network layer에 있는 ipv4에게 패킷을 보내고, ipv4는 목적지로 가기 위한 이더넷 인터페이스인 link layer에 있는 eth[0]을 통해 패킷을 전송한다. 즉, 소스 호스트에는 1개의 이더넷 인터페이스가 있다.



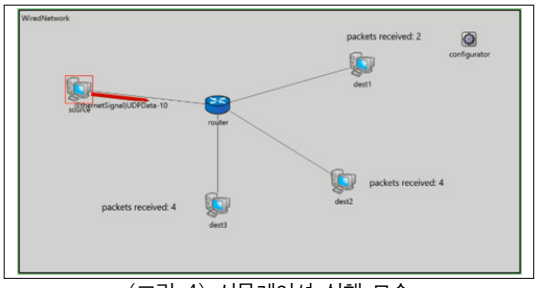
(그림 2) Qtenv에서 목적지 호스트
(Figure 2) Destination host in Qtenv

그림 2는 Qtenv에서 목적지 호스트의 그래픽 표현이다. 목적지 호스트는 1개의 udp 애플리케이션과 1개의 이더넷 인터페이스를 가지고 있다. 목적지 호스트는 link layer의 이더넷 인터페이스 eth[0]에서 수신한 패킷을 network layer의 ipv4와 transport layer의 udp를 통해 app에게 전달한다.

그림 3은 Qtenv에서 Router 인스턴스인 라우터의 그래픽 표현이다. Router 모듈은 inet.node.inet 패키지에 있는 모듈로 무선, 이더넷, PPP, 외부 인터페이스를 지원하는 IPv4 라우터 모듈이다. 게이트 ethg[]를 이용하여 다른 노



(그림 3) Qtenv에서 라우터
(Figure 3) Router in Qtenv



(그림 4) 시뮬레이션 실행 모습
(Figure 4) A Snapshot of Simulation Execution

드와 이더넷 인터페이스를 통해 연결된다. 기본적으로 full-duplex 연결을 지원한다. 이더넷 인터페이스의 개수는 매개변수 numEthInterfaces를 이용하여 설정한다. 그림 3의 라우터에는 link layer에 4개의 이더넷 인터페이스가 있다. 즉, 그림 2를 통해 라우터에 1개의 소스와 3개의 목적지 호스트가 라우터와 연결되어 있음을 알 수 있다. 이더넷 인터페이스로부터 수신한 패킷은 network layer의 ipv4로 전달되고, ipv4에서 목적지로 가기 위한 이더넷 인터페이스를 통해 패킷을 전달한다.

Eth10M 모듈은 inet.node.ethernet 패키지에 있는 모듈로 10 megabit/sec 이더넷 링크이다. 본 논문의 시뮬레이션에는 라우터와 4개의 호스트를 Eth10M 모듈을 이용하여 연결하였다.

UdpBasicApp 모듈은 inet.applications.udpapp 패키지에 있으며, 이는 일정한 간격으로 UDP 패킷을 전송하는 모듈로 IPv4와 IPv6 모두와 호환된다. 전송 간격을 파라미터 sendInterval에 일정한 값을 설정하거나 랜덤 값을 설정할 수 있다. UDP 패킷의 목적지는 destAddress매개변수에 1개 이상을 설정할 수 있다. 목적지가 2개 이상일 경우 패킷을 전송할 때마다 무작위로 하나의 목적지 주소가 선택된다. UdpBasiApp의 목적지 호스트는 UdpSink 또는

```
import inet.node.inet.Router;
import inet.node.ethernet.Eth10M;
import inet.node.inet.StandardHost;
import inet.networklayer.configurator.ipv4.Ipv4NetworkConfigurator;

network WiredNetwork
{
    parameters:
        @display("bgb=616,321");

        @figure[rcvdPkText0](type=indicatorText; pos=402,30;
                anchor=w; font=,8; textFormat="packets received: %g"; initialValue=0);
        @statistic[packetReceived0](source=dest1.app[0].packetReceived;
                record=figure(count); targetFigure=rcvdPkText0);

        @figure[rcvdPkText1](type=indicatorText; pos=430,210; anchor=w;
                font=,8; textFormat="packets received: %g"; initialValue=0);
        @statistic[packetReceived1](source=dest2.app[0].packetReceived;
                record=figure(count); targetFigure=rcvdPkText1);

        @figure[rcvdPkText2](type=indicatorText; pos=100,240; anchor=w;
                    font=,8; textFormat="packets received: %g"; initialValue=0);
        @statistic[packetReceived2](source=dest3.app[0].packetReceived;
                record=figure(count); targetFigure=rcvdPkText2);

    submodules:
        configurator: Ipv4NetworkConfigurator {
            @display("is=s;p=554,35");
        }
        router: Router {
            @display("p=244,116");
            numEthInterfaces = 4;
        }
        source: StandardHost {
            @display("p=73,97");
            numEthInterfaces = 1;
        }
        dest1: StandardHost {
            @display("p=422,66");
            numEthInterfaces = 1;
        }
        dest2: StandardHost {
            @display("p=399,210");
            numEthInterfaces = 1;
        }
        dest3: StandardHost {
            @display("p=244,239");
            numEthInterfaces = 1;
        }

    connections:
        source.ethg[0] <--> Eth10M <--> router.ethg[0];
        router.ethg[1] <--> Eth10M <--> dest2.ethg[0];
        router.ethg[2] <--> Eth10M <--> dest1.ethg[0];
        router.ethg[3] <--> Eth10M <--> dest3.ethg[0];
}
```

(그림 5) 시뮬레이션 네트워크 환경(WiredNetwork)의 NED 소스 코드

(Figure 5) NED Source Code for Simulation Network Environment (WiredNetwork)

UdpBasicApp, UdpEchoApp을 사용할 수 있다. 본 논문의 시뮬레이션의 소스 호스트는 UdpBasciApp이 udp 패킷을 생성하여 목적지 호스트에 전송한다.

UdpSink 모듈은 inet.applications.udpapp 패키지에 있는 모듈로 수신한 UDP 패킷을 처리 후 출력한다. 다음 장 시뮬레이션의 목적지 호스트는 UdpSink가 소스가 전송한 udp 패킷을 수신한다.

Ipv4NetworkConfigurator 모듈은 inet.networklayer.configurator.ipv4 패키지에 있는 모듈로 IPv4 네트워크에 IPv4 주소를 할당하고 정적 라우팅을 설정한다. 전역 모듈로 설정한 Ipv4NetworkConfigurator는 호스트의 IP주소, 정적 경로 등을 계산하지만 노드의 라우팅 테이블과 인터페이스를 실제로 구성하지는 않는다. 이러한 설정 과정을 통해 시뮬레이션을 실행하면 Ipv4NetworkConfigurator가 라우터와 4개의 호스트에 IP주소 및 경로를 설정한다.

4. INET를 이용한 이더넷 네트워크 시뮬레이션 방법

본 장에서는 INET 프레임워크의 사용을 이해하기 위해서 간단한 시뮬레이션 방법을 소개한다. 우선 NED 언어를 사용하여 시뮬레이션 환경인 네트워크 토폴로지를 구성하고, 시뮬레이션을 실행하기 위해 설정 파일(configuration file)인 omnetpp.ini 파일을 생성한다. 여기서 시뮬레이션과 관련된 파라미터를 ned 또는 ini 파일에서 다양하게 설정할 수 있다. 이와 관련된 내용은 잠시 후 예제 코드(그림 5 및 6)를 통해 설명할 것이다[7-8].

시뮬레이션 실행 모습은 그림 4와 같으며, 네트워크 환경은 4개의 노드와 한 개의 라우터로 구성된 유선 네트워크 환경에서의 실행되는 모습이다. source에서 목적지 노드들(dest1, dest2, dest3)에게 UDP데이터를 전송하며, 목적지 노드는 수신한 데이터의 수를 보여주고 있다.

그림 5는 NED 언어의 텍스트 편집기(Source)에서 WiredNetwork를 정의한(Line 6) 소스 코드이다. 이 코드에서는 submodules(Line 26)에 네트워크의 구성 요소로 4개의 StandardHost(Lines 34, 38, 42, 46)와 1개의 Router(Line 30), 1개의 IPv4NetworkConfigurator(Line 27)를 사용하여 네트워크를 정의하였다. 각 StandardHost(source, dest1, dest2, dest3)는 1개의 이더넷 인터페이스를 가지며(Lines 36, 40, 44, 48), Router(router)는 4개의 이더넷 인터페이스를 가진다(Line 32).

또한, connections(Line 51)에서는 4개의 StandardHost (source, dest1, dest2, dest3)와 Router(router)를 Eth10M 채널을 사용하여 이더넷 인터페이스로 연결하여 패킷 송수신이 가능하다(Lines 52-55).

parameters(Line 8)에 @figure와 @statistic을 사용하여 시뮬레이션 도중 dest1, dest2, dest3이 source로부터 수신한 패킷 수를 동적으로 보여줄 주 있도록 설정하였다(Lines 11-24).

```
1  [Config Wired]
2  network = WiredNetwork
3
4  *.source.numApps = 1
5  *.source.app[0].typename = "UdpBasicApp"
6  *.source.app[0].messageLength = 100B
7  *.source.app[0].sendInterval = exponential(12ms)
8  *.source.app[0].destAddresses = "dest1 dest2 dest3"
9  *.source.app[0].destPort = 5001
10 *.source.app[0].packetName = "UDPData"
11
12 *.dest*.numApps = 1
13 *.dest*.app[0].typename = "UdpSink"
14 *.dest*.app[0].localPort = 5001
```

(그림 6) omnetpp.ini 매개변수 설정
(Figure 6) Parameters in the omnetpp.ini

그림 6에서는 시뮬레이션 실행을 위한 ini 파일의 작성을 보여준다. 시뮬레이션 환경 설정 이름은 Wired이며(Line 1), 그림 4와 같은 네트워크 환경 WiredNetwork(Line 2)을 등록한다. 시뮬레이션에서 필요한 특정 매개변수의 설정은 다음과 같다.

source, dest1, dest2, dest3 호스트의 애플리케이션 수를 각각 1개로 설정한다(Lines 4, 12). 노드 source의UdpBasicApp(Line 5), 노드들(dest1, dest2, dest3)의 UdpSink(Line 13) 애플리케이션들을 선택한다. 출발 노드(source)의 목적지 노드들(dest1, dest2, dest3)의 지정은 Line 8과 같이 기재한다. 출발 노드의 목적지 노드에 대한 포트 5001을 설정하고(Line 9), 목적지 노드들의 로컬포트를 5001로 지정한다(Line 14). 전송되는 패킷의 이름은 UDPData(Line 10), 각 패킷의 길이는 100바이트(Line 6), 패킷 전송에 대한 분포는 지수분포를 따르도록 설정한다(Line 7).

5. 결 론

INET은 OMNeT++ 시뮬레이터의 다양한 모듈을 활용하여 네트워크 시뮬레이션을 위한 종합적인 라이브러리를 제공한다. INET 라이브러리에는 네트워크 프로토콜, 장치, 애플리케이션, 유틸리티, 환경 모델 및 네트워크 인터페이스 등이 포함되어 있어, 연구자가 복잡한 네트워크 환경을 시뮬레이션하는 데 필요한 다양한 도구를 제공한다.

연구자는 호스트, 라우터, 기타 필수 장치와 같은 INET 모듈들을 결합하여 전체 네트워크를 구성하고, 이러한 모듈 간의 상호작용을 통해 네트워크 시뮬레이션을 수행할 수 있다. 만약 시뮬레이션에 필요한 특정 모듈이 없다면, 연구자는 C++을 사용하여 새로운 INET 모듈을 직접 구현하거나 기존 모듈의 소스 코드를 수정하여 재사용할 수 있다.

본 연구에서는 INET의 대표적인 모듈들을 이용하여 유선 네트워크를 구성하고, 유니캐스트로 메시지를 전송하는 간단한 시뮬레이션 예제를 제시하였다. 향후에는 INET 프레임워크를 활용한 무선 및 이동 통신 네트워크 시뮬레이션 등을 소개할 예정이다.

참고문헌(Reference)

[1] Retrieved from https://omnetpp.org/download-items/INET.html (October 2024)

[2] Retrieved from https://github.com/inet-framework/inet (October 2024)

[3] Virdis and M. Kirsche, “Recent Advances in Network Simulation: The OMNeT++ Environment and Its Ecosystem,” p. 55, Switzerland: Springer, 2019. https://link.springer.com/book/10.1007/978-3-030-12842-5

[4] Retrieved from https://inet.omnetpp.org/docs/users-guide/ch-usage.html (October 2024)

[5] Retrieved from https://inet.omnetpp.org/Protocols.html (October 2024)

[6] Retrieved from https://doc.omnetpp.org/inet/api-current/neddoc/inet.node.inet.StandardHost.html (September 2024)

[7] Retrieved from https://doc.omnetpp.org/omnetpp/manual/ (October 2024)

[8] S. Park, “A Basic Guide to Network Simulation Using OMNeT++,” KSII, Journal of Internet Computing and Services (JICS), vol. 25, no. 4, pp. 1-6, August 2024. http://dx.doi.org/10.7472/jksii.2024.25.4.1

◐ 저 자 소 개 ◑

박 수 연(Sooyeon Park)

2004년 한국공학대학교 컴퓨터공학과 (공학사)
2007년 한국공학대학교 컴퓨터공학과 (공학석사)
2011년 한국공학대학교 컴퓨터공학과 (공학박사)
2011년~2015년 전자부품연구원 연구원
2024년~현재 서울신학대학교 IT융합소프트웨어학과 초빙교수
관심분야 : 지능형 모바일 센서 네트워킹, 머신러닝 및 인공지능, 시뮬레이션, 데이터베이스 등
E-mail : anisoo@stu.ac.kr

김 문 성(Moonseong Kim)

2002년 성균관대학교 일반대학원 수학과 (이학석사)
2023년 경상국립대학교 일반대학원 수학과 (박사수료)
2007년 성균관대학교 일반대학원 전기전자및컴퓨터공학부 (공학박사)
2007년~2009년 미국 미시간주립대학교 컴퓨터과학공학과 연구원
2009년~2018년 특허청 사무관 (서기관 대우)
2018년~현재 서울신학대학교 IT융합소프트웨어학과 교수 (행정사 / 기술거래사)
관심분야 : 모바일 센서 네트워크, 지능형 모바일 컴퓨팅, 머신러닝 및 인공지능, 정보보안, 지식재산권 등
E-mail : moonseong@stu.ac.kr