NS-3 Tutorial, Manual, Testing Documents

2011년 2월

경북대학교 통신프로토콜연구실

최 상일 (overcycos@gmail.com)

요 약

Ns-3는 ns-2와 같은 Network simulator이지만 ns-2와는 완전히 다르다. 이런 차이로 ns-2에서 동작되는 simulation이 ns-3에서는 동작하지 않을 수도 있지만, ns-3의 경우 C++을 사용함으로써 객체 지향적 성격을 가지게 되었고 ns-2에서는 구현되지 못했던 여러 부분들을 구현해 낼 수 있게 되었다. 본 문서에서는 ns-3를 접하는 사용자들을 위해 nsnam [1]에서 제공하는 세가지 문서(Tutorial, Manual, Testing)를 통해 알 수 있는 바를 간략하게 기술하고 있다.

목 차

1. 서론

Ns-3는 C++과 Python으로 구현된 문서이며, C++ 언어를 사용함에 따라 객체 지향의 성격을 가진다. Ns-3는 기존의 ns-2와는 다른 형태를 가지기 때문에 처음 접하는 사용자는 Simulation 제작에 어려움을 가질 수도 있다. 이를 위해 이 문서에서는 nsnam [1]에서 제공하는 Tutorial, Manual, Testing 문서의 내용에 대해 간략히 기술하도록 한다.

2. Ns-3 Tutorial 문서

Ns-3는 Otcl과 C++로 구현된 ns-2와는 달리 Python과 C++로 구현되어 있으며 ns-2의 후속 버전이 아닌 새로운 simulator로서 제작이 되었기 때문에 기존에 ns-2에서 사용되었던 모든 모델을 지원하지는 못하지만 ns-2에서 지원되지 않았던 Multiple interface, IP addressing 등을 지원할 수 있다.

Ns-3의 resource는 http://www.nsnam.org [1]에서 다운받을 수 있다. Ns-3는 Linux 혹은 Linux-like 환경에서만 동작이 가능하며 C++, Python 등 몇 가지 패키지를 설치해 놓아야 한다. 필요한 패키지는 위의 웹 페이지에서 확인 할 수 있다.

개념적으로 ns-3에는 Node, Application, Channel, NetDevice, Helper가 있다. 빈 Node에 특정 기능을 가진 application을 넣어 특별한 node를 만들고, NIC의 역할을 하는 NetDevice를 Node에 붙여준다. 그리고 통신에 사용될 Channel을 생성해 각 NetDevice에 연결하면 노드간에 패킷 전송이 가능하게 된다. 마지막으로 Helper는 각 개념의 생성시 필요한 복잡한 내부 설정에 관련된 코드를 묶어놓은 것으로 사용자는 이 Helper를 사용함으로써 위의 개념들을 간단한 몇 줄의 코드로 생성 및 설정을 할 수 있다. Ns-3의 sample코드 중 하나인 firse.cc(point-to-point link, 두 노드 사이에서 단일 패킷 되풀이)를 통해 위의 개념들을 살펴보겠다.

```
NodeContainer nodes;
nodes.Create (2);

PointToPointHelper pointToPoint;
pointToPoint.SetDeviceAttribute ("DataRate", StringValue ("5Mbps"));
pointToPoint.SetChannelAttribute ("Delay", StringValue ("2ms"));

NetDeviceContainer devices;
devices = pointToPoint.Install (nodes);
```

코드 1. node, channel, device 생성 및 설정

NodeContainer를 이용해 Node를 생성하고, PointToPointHelper를 이용해 Channel을 생성하며 채널의 값을 설정한다. 그리고 NetdeviceContainer로 NetDevice를 생성하고 Install 멤버 함수를 통해 node와 Netdevice를 연결.

```
InternetStackHelper stack;
stack.Install (nodes);
```

코드 2. Internet stack 생성 및 설정

InternetStackHelper를 이용하여 internet stack을 생성하고, node를 stack에 넣는다.

```
Ipv4AddressHelper address;
address.SetBase ("10.1.1.0", "255.255.255.0");

Ipv4InterfaceContainer interfaces = address.Assign (devices);
```

코드 3. Address 생성 및 설정

Ipv4AddressHelper를 이용해 Ipv4 주소를 생성하고, Gateway 및 Subnetmask를 설정하고 device에 해당 인터페이스를 등록한다.

```
Ipv4InterfaceContainer interfaces = address.Assign (devices);

UdpEchoServerHelper echoServer (9);

ApplicationContainer serverApps = echoServer.Install (nodes.Get (1));
serverApps.Start (Seconds (1.0));
serverApps.Stop (Seconds (10.0));

UdpEchoClientHelper echoClient (interfaces.GetAddress (1), 9);
echoClient.SetAttribute ("MaxPackets", UintegerValue (1));
echoClient.SetAttribute ("Interval", TimeValue (Seconds (1.)));
echoClient.SetAttribute ("PacketSize", UintegerValue (1024));

ApplicationContainer clientApps = echoClient.Install (nodes.Get (0));
clientApps.Start (Seconds (2.0));
clientApps.Stop (Seconds (10.0));
```

코드 4. Interface 설정 및 Client, Server 생성

이 후, 서버와 클라이언트를 만들고 port 번호 등을 설정하여 패킷을 보낸다.

Ns-3에서 simulation이 제대로 동작되었는지 확인하기 위해 tracing을 하여야 한다. 코드 내부에 cout<<endl 을 이용한 화면출력함수를 이용하여도 되지만, 내부에서 TraceHelper를 이용해 tracing 파일을 생성하는 것이 확인하기에 더 편리하다. 생성된 파일은 .pcap 확장자를 가지고 있으며 이 파일을 WireShark로 불러오면 전송된 패킷들의 정보를 확인 할 수 있다.

3. NS-3 Manual 문서

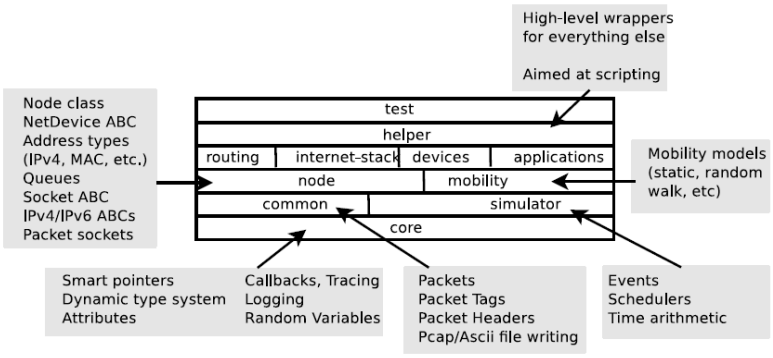

그림 1. Software organization of ns-3

Ns-3는 simulation core와 model이 C++로 구현되어 있는 네트워크 시뮬레이터로 이 장에서는 위의 그림에서 알아야 할 부분에 대해서 살펴보겠다.

3.1 Core

Core 부분에서 attribute, tracing, real time, distributed, packet, helper에 대해서 알아보겠다.

Attribute는 말 그대로 속성을 나타내며 ns-3의 class들은 각기 여러 개의 attribute를 가진다. 특별한 점은 AddAttribute를 이용해 class에 인위적으로 attribute를 추가할 수 있다는 것이다.

```
TypeId DropTailQueue::GetTypeId (void)
{
  static TypeId tid = TypeId ("ns3::DropTailQueue")
    .SetParent<Queue> ()
    .AddConstructor<DropTailQueue> ()
    .AddAttribute ("MaxPackets",
                   "The maximum number of packets accepted by this DropTailQueue.",
                   UintegerValue (100),
                   MakeUintegerAccessor (&DropTailQueue::m_maxPackets),
                   MakeUintegerChecker<uint32_t> ())
    ;

  return tid;
}
```

코드 5. AddAttribute 사용

위의 코드는 m_maxPackets에 접근 할 수 있는 MaxPackets라는 속성을 추가한 것으로 위에위부터 Attribute명, 설명, 기본값, 접근 할 변수, 오류 체크에 대한 내용이 있다.

Tracing은 사용자가 제안한 환경에서 시뮬레이션을 한 뒤, 성능의 차이를 확인하거나 오류를 찾기 위해서는 시뮬레이션 동안에 발생한 내용들을 사용자가 볼 수 있도록 해주는 것이다. Ns-3에서는 pcap tracing과 ascii tracing을 제공하고 있다.

Real time은 실제 네트워크 스택에 시뮬레이션을 구현하기 위해 각 장비들이 같은 시뮬레이션 clock을 가지도록 하드웨어 clock으로 고정시키는 것을 말한다. 초기에는 시뮬레이션 과정에서 지정된 시간이 초과되는 것을 특정범위 내에서 허가하는 BestEffort 모드와 초과시 시뮬레이션을 종료시키는 HardLimit모드가 있었지만, 현재에는 ns-3 tree에서 제외된 상태이다.

Distributed는 multiple processor에서 단일 시뮬레이션 프로그램이 동작되는 것을 말하는 것이다. 이를 위해서는 logical process간에 메시지 교환이 필요하고 이 교환에는 Message Passing Interface(MPI)가 있어야 한다. 현재 Distributed는 Point-to-Point Link에서만 동작이 가능하다.

Packet은 Byte buffer, Byte tags, Packet tags, metadata로 구성되어 있다. Byte buffer는 헤더와 트레일러의 정보를 가지고 있고, Byte tag는 packet byte buffer의 byte subset에 tag를 붙이기 위해 사용이 되며, Packet tag는 packet 자체의 tag를 위해 사용된다. 마지막으로 metadata는 byte buffer에 있는 헤더와 트레일러의 정보를 가져 context 없이 패킷의 헤더를 확인하기 편하게 출력하는데 사용된다.

Helper는 low-level API를 사용하여 구현을 하는데 코드가 너무 길고 어렵기 때문에 해당 API들을 모아 사용자가 손 쉽게 API를 사용할 수 있도록 도와주도록 ns-3에서 제공하는 것이다.

3.2 Node and NetDevices

Ns-3에서 node는 빈 공백과 같은 것으로 NetDevice, protocol, application을 추가하며 특정한 기능을 하게 된다. Ns-3에서는 다양한 net device와 framework를 지원한다.

Point-to-Point NetDevice를 통해 Point-to-Point link를 이용할 수 있으며, CSMA NetDevice를 통해 bus network link를 이용하고, Wifi NetDevice를 통해 802.11 기반의 infrastructure와 ad hoc network를 이용하고, Wimax NetDevice를 통해 802.16 기반의 network model을 이용한다. 또한 LTE Module을 통해 3GPP E-UTRAN infrastructure와 LTE(Long Term Evolution) 모델을 제공하고, UAN Framework를 통해 수중 네트워크를 사용할 수 있다.

3.3 Emulation

Ns-3는 testbed와 가상 환경의 통합을 위해 Emu NetDevice와 Tap NetDevice를 제공한다. Emu NetDevice를 통해 실제 네트워크에 Packet을 보낼 수 있고, Tap NetDevice를 통해 시뮬레이션 노드를 실제 네트워크 내의 호스트가 되도록 할 수 있다. 이 두 NetDevice를 통해 실제 네트워크와 가상 환경이 서로 데이터를 주고 받을 수 있다.

3.4 Sockets APIs

Ns-3에는 Native ns-3 API와 POSIX-like API 이렇게 두 가지 socket API가 있다. POSIX-like API는 아직 구현되지 않았지만 Native ns-3 API는 C++로 구현되었으며 여러 종류의 transport protocol을 제공할 수 있다.

ns-3에서는 buffer variant를 통해 application 단에서 유용한 정보를 packet에 encode 할 수 있다.

4. Ns-3 Testing and Validation 문서

Software testing은 에러를 찾기 위해 프로그램을 동작시키는 방법이다. 일반적으로 이 testing에서는 프로그램이 구현하려고 했던 프로그램과 일치하는지, target system에 적용가능한지, simulation이 나타내는 결과가 믿을 만 한지, 외부적 혹은 내부적 압박에 정상적 종료가 되어 사용자에게 피해가 가지 않는지, 현재 적용되어 있는 시스템에 비해 성능이 뛰어난지, 오랜 시간이 지나도 안정된 상태가 지속되는지 등에 대한 내용을 확인하는 것이다.

Ns-3에서는 test.py 라는 파일이 있는데, 이 파일은 모든 테스트를 구동하고 사용자가 읽을 수 있는 폼으로 결과를 모으는 기능을 수행한다.

```
Options:
  -h, --help            show this help message and exit
  -c KIND, --constrain=KIND
                        constrain the test-runner by kind of test
  -e EXAMPLE, --example=EXAMPLE
                        specify a single example to run
  -g, --grind           run the test suites and examples using valgrind
  -k, --kinds           print the kinds of tests available
  -l, --list            print the list of known tests
  -m, --multiple        report multiple failures from test suites and test
                        cases
  -n, --nowaf           do not run waf before starting testing
  -s TEST-SUITE, --suite=TEST-SUITE
                        specify a single test suite to run
  -v, --verbose         print progress and informational messages
  -w HTML-FILE, --web=HTML-FILE, --html=HTML-FILE
                        write detailed test results into HTML-FILE.html
  -r, --retain          retain all temporary files (which are normally
                        deleted)
  -t TEXT-FILE, --text=TEXT-FILE
                        write detailed test results into TEXT-FILE.txt
  -x XML-FILE, --xml=XML-FILE
                        write detailed test results into XML-FILE.xml
```

그림 2. test.py 옵션 값

5. 결 론

지금까지 본 문서에서는 ns-3를 처음 접하는 사용자를 위해 nsnam [1]에서 제공하는 세 가지 문서가 설명하는 바를 기록하였다. 본 문서가 기반한 세 가지 문서는 ns-3를 이용해 코드를 작성하는 부분에 대한 설명보다는 기본 개념과 제공 가능한 모델들에 대해 설명하고 있다. 이 문서를 통해 ns-3에 대한 넓은 바탕을 가진 후, 제공되는 sample 코드를 이용해 코드 분석을 한다면 훨씬 수월하게 ns-3에 대해 이해할 수 있으리라 생각된다.

참고 문헌

[1] The ns-3 network simulator, http://nsnam.org/