

NS-3 구성요소 및 동작과정

<https://youtu.be/UVQxHXztFro>

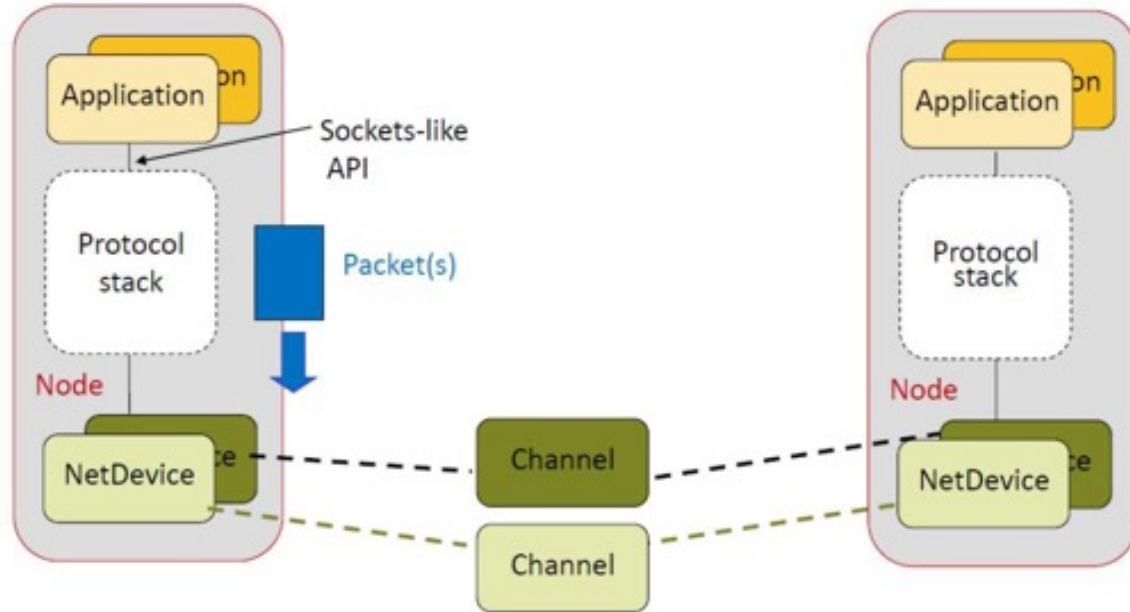
NS3 구성 요소

NS3 동작 과정

코드 분석

NS-3 구성요소

- **Node** : 네트워크에 연결하는 **컴퓨터 디바이스**
- **Application** : 노드 상에서 실행시킬 **프로그램**
- **NetDevice** : 다른 노드와 **통신할 수 있도록 하는 장치** (실제 컴퓨터의 NIC*에 해당)
- **Channel** : **통신 채널**
- **Helper** : 시뮬레이션 시 필요한 작업들을 쉽게 수행할 수 있도록 지원 (IP 주소 할당, 채널에 연결 등)



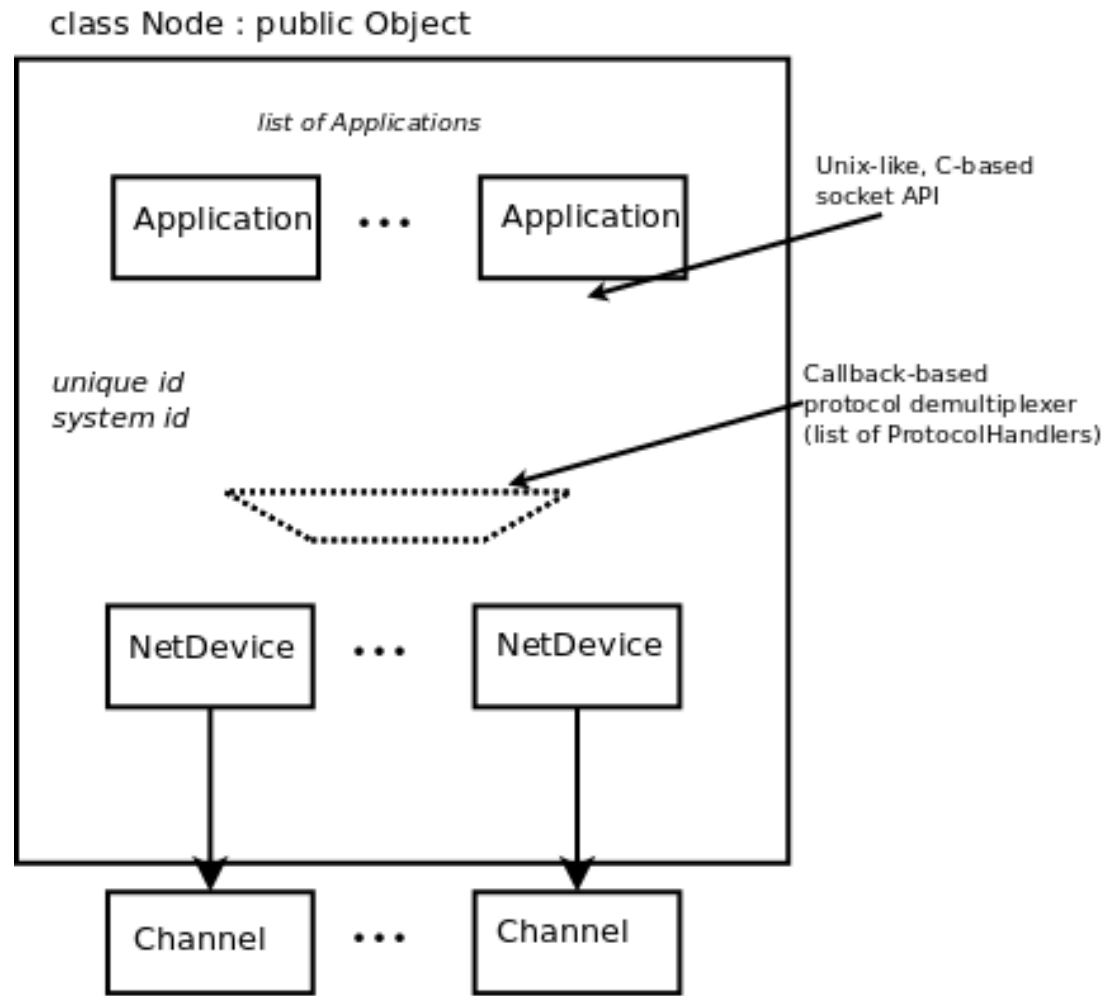
*Network Interface Controller (NIC) : 컴퓨터를 네트워크에 연결하여 통신하기 위해 사용하는 하드웨어 장치

NS3에서의 노드

• Class Node

- Id : 노드 식별자
- SystemId : 병렬 시뮬레이션에 사용되는 식별자 (MPI)
- NetDevice[] : 다른 노드와 통신할 수 있도록 하는 장치
- Application[] : 노드 상에서 실행시킬 응용 프로그램

```
class Node : Object
{
    uint32_t          m_id;
    uint32_t          m_sid;
    vector<NetDevice> m_devices;
    vector<Application> m_applications;
}
```



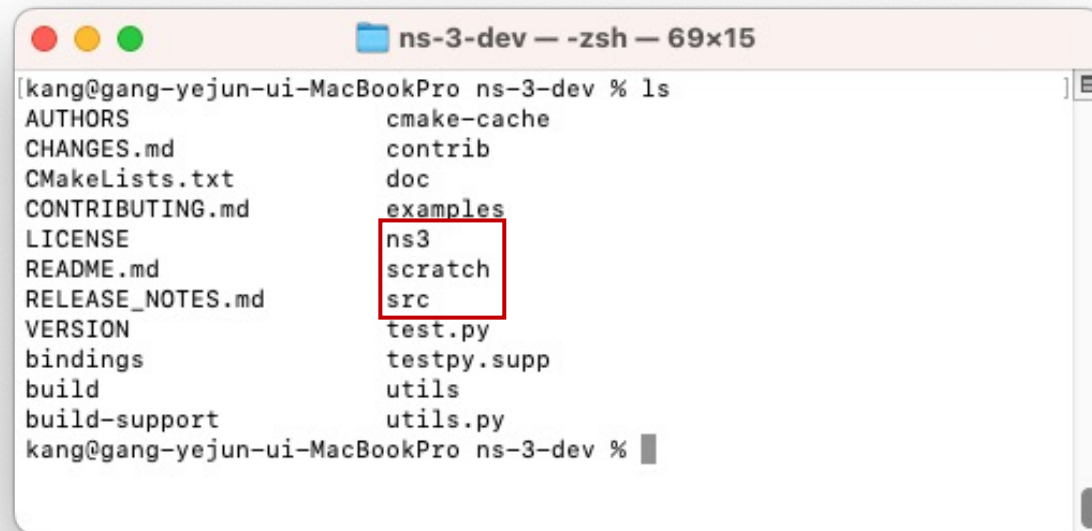
NS3 동작과정

- ns3 실행 파일

- ns3 빌드 및 실행 프로그램

- scratch 폴더

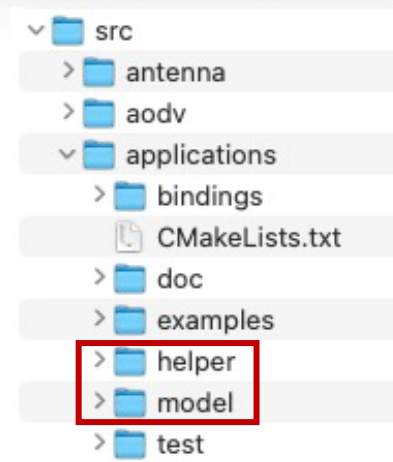
- c++로 작성된 **시나리오 파일**이 포함되어 있는 폴더
- ns3 실행 시 scratch 폴더 내에 시나리오 파일을 찾게 됨



```
ns-3-dev — -zsh — 69x15
[kang@gang-yejun-ui-MacBookPro ns-3-dev % ls
AUTHORS                cmake-cache
CHANGES.md            contrib
CMakeLists.txt         doc
CONTRIBUTING.md       examples
LICENSE                ns3
README.md              scratch
RELEASE_NOTES.md       src
VERSION                test.py
bindings               testpy.supp
build                  utils
build-support          utils.py
kang@gang-yejun-ui-MacBookPro ns-3-dev %
```

- src 폴더

- c++로 작성된 network, lte, point-to-point, wifi, **application** 등과 같은 모듈들이 있음
- ns3에서 제공하는 모듈 외 필요한 모듈이 있다면 해당 폴더에 추가하여 사용 가능
- applications/helper : 다른 클래스의 설정을 쉽게 하기 위한 클래스
- applications/model : **노드 상에서 실행시킬 프로그램 (ex. 합의 알고리즘)**



NS3 시나리오 파일 분석

- scratch/blockchain-simulator.cc

command line을 통해 옵션을 부여하여
노드의 개수 설정

```
./ns3 run "blockchain-simulator --numNodes=4"
```

ns3 실행

파일명

노드의 개수

시간을 표현하는 단위 설정

Logging 활성화 및 함수 호출

```
int main(int argc, char *argv[])
{
    int numNodes;

    CommandLine cmd;
    cmd.AddValue("numNodes", "Number of node", numNodes=4);
    cmd.Parse(argc, argv);

    NS_LOG_UNCOND("Hello PoW Simulator");

    Time::SetResolution(Time::NS);

    LogComponentEnable("PoW", LOG_LEVEL_INFO);
    startSimulator(numNodes);

    return 0;
}
```

시나리오 파일 분석

NodeContainer를 통해 **노드 생성**

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

다른 노드와 p2p 통신할 수 있도록
NetDevice 및 Channel 설치

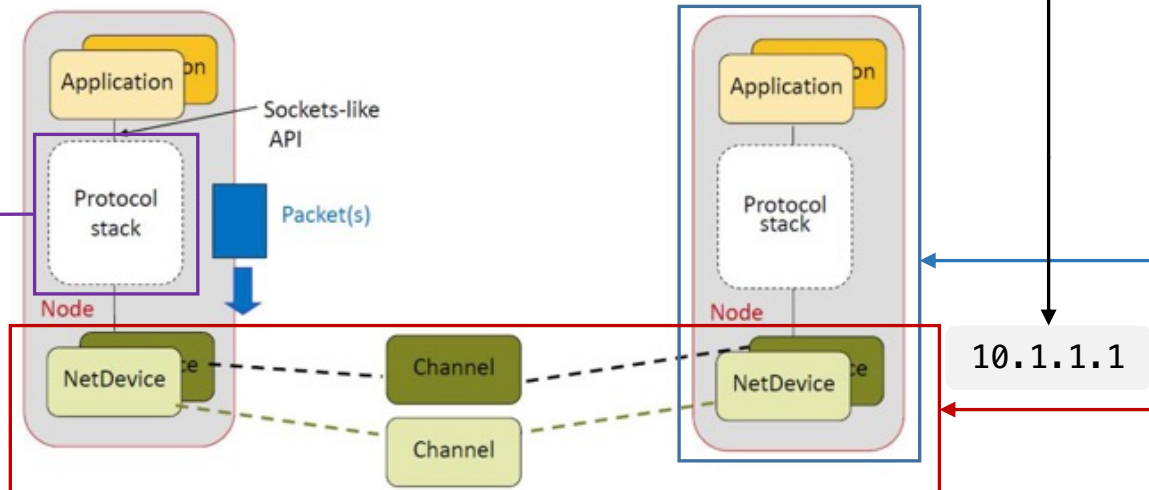
```
PointToPointHelper pointToPoint;  
pointToPoint.SetDeviceAttribute("DataRate", StringValue("3Mbps"));  
pointToPoint.SetChannelAttribute("Delay", StringValue("3ms"));  
  
Ptr<Node> p1 = nodes.Get(i);  
Ptr<Node> p2 = nodes.Get(j);  
NetDeviceContainer device = pointToPoint.Install(p1, p2);
```

NetDevice에 **IP 주소 할당**

```
Ipv4AddressHelper address;  
address.SetBase("10.1.1.0", "255.255.255.0");  
Ipv4InterfaceContainer interface;  
interface.Add(address.Assign(device.Get(0)));  
interface.Add(address.Assign(device.Get(1)));
```

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

Protocol stack 설치



applications/helper 파일 분석

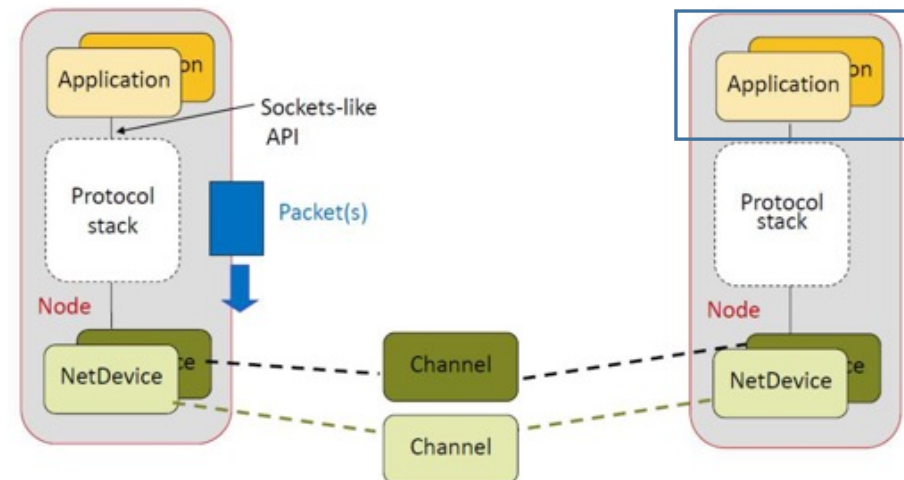
- src/applications/helper/network-helper.cc

노드에 **Application** 설치

```
NetworkHelper networkHelper(numNodes);  
ApplicationContainer nodeApp = networkHelper.Install (nodes);
```

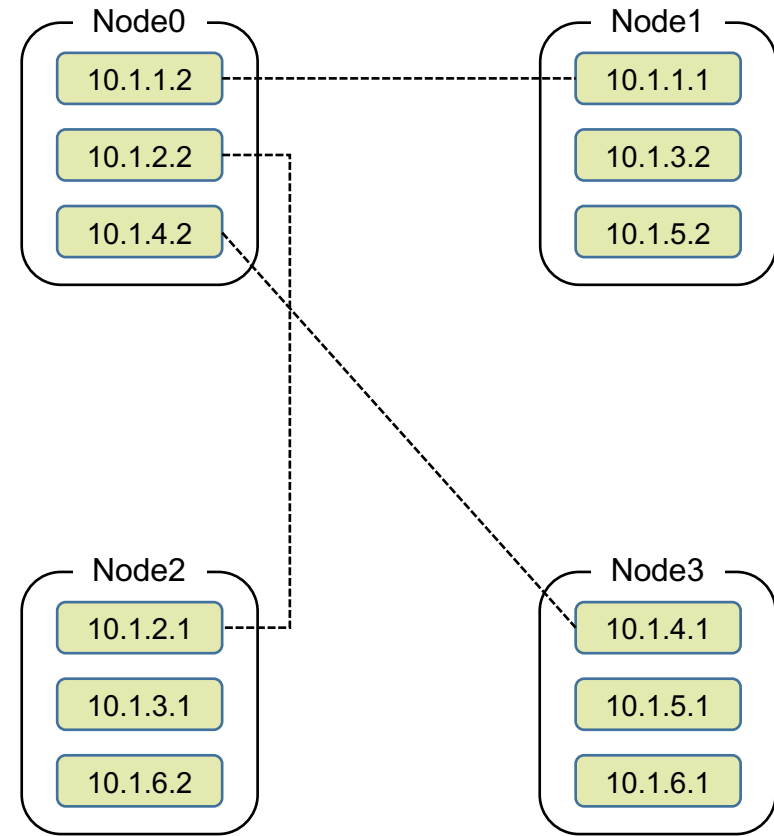
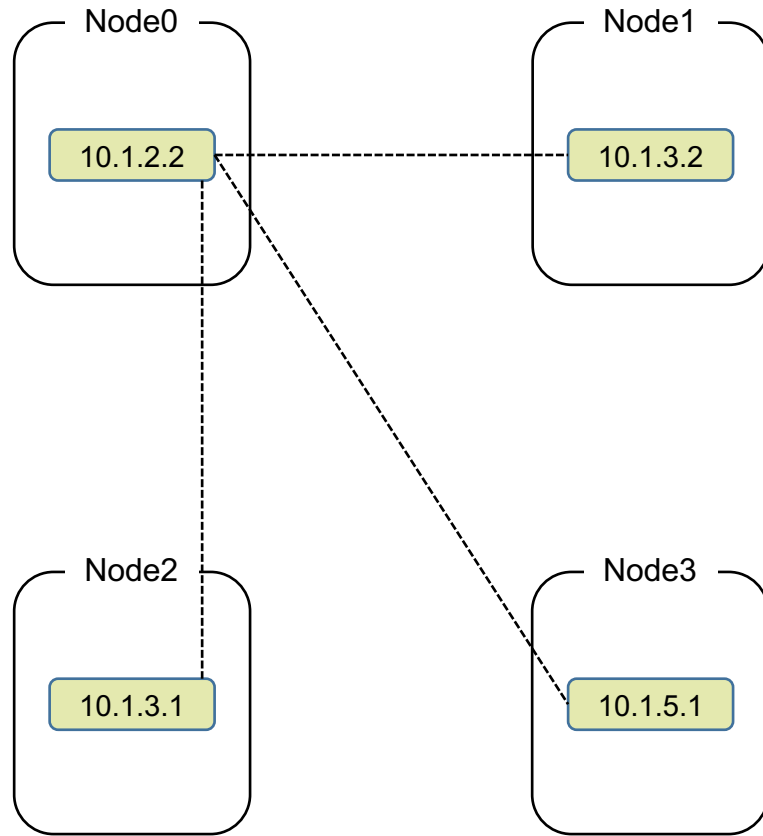
```
ApplicationContainer NetworkHelper::Install (NodeContainer c)  
{  
    ApplicationContainer apps;  
  
    for (NodeContainer::Iterator i = c.Begin (); i != c.End (); i++)  
    {  
        Ptr<PoW> app = m_factory.Create<PoW> ();  
  
        uint32_t nodeId = (*i)->GetId();  
  
        app -> m_id = nodeId;  
        app -> numNodes = m_nodeNo;  
        app -> m_peersAddresses = m_nodesConnectionsIps[nodeId];  
        (*i)->AddApplication (app);  
  
        apps.Add (app);  
    }  
    return apps;  
}
```

```
NetworkHelper::NetworkHelper(uint32_t totalNoNodes)  
{  
    m_factory.SetTypeId ("ns3::PoW");  
    m_nodeNo = totalNoNodes;  
}
```



실제 코드

실제 코드



Q & A