NS3 LTE Simülasyonu

Beycan Kahraman
504092502

30.04.2012

Sunum İçeriği

- NS3'teki LTE kütüphanesi
- NS3 Sınıfları ve Fonksiyonları
- NS3'te Basit Bir Uygulama
- Birkaç NS3 Sınıfı Daha…

NS3'teki LTE Kütüphanesi

- Yeni nesil ağ simülasyonları için NS3 kütüphanesi (http://www.nsnam.org/)
- Kütüphanenin dökümantasyonu (http://www.nsnam.org/docs/release/3.13/doxygen/index.html)
- LTE kütüphanesiyle ilgili detaylı bilgi (http://www.nsnam.org/docs/release/3.13/doxygen/index.html)

NS3'teki LTE Kütüphanesi

- Temel UE ve eNB tasarımları
- UE ve eNB için Radio Recource Control (RRC) işlemleri
- DL için Adaptive Modulation and Coding (AMC) şeması
- Data Radio Bearers (+QoS)
- Channel Quality Indicator (CQI)
- DL/UL için packet scheduling
- Kaynak bloklama seviyesinde fiziksel katman
- Outdoor E-UTRAN yayılım kaybı modeli

NS3'teki LTE Kütüphanesi

Fiziksel Katman

- DL işleminde kullanılan alt kanal için SINR hesaplanır
- Channel Quality Indicator (CQI) değerleri hesaplanır ve geri gönderilir
- Alınan paketler ilgili cihaza aktarılır

Yayılma Kaybı Modeli

- Yolkaybı: 128.1 + (37.6 * $\log_{10}$(uzaklık))
- Jakes Multipath model
- 10 dB Penetration Loss
- Shadowing: log-normal dist. (m: 0dB, std: 8dB)

NS3'teki LTE Kütüphanesi

İçerik ve Kısıtlamalar

- DL yönetimi Downlink Packet Scheduler tarafından yapılmıştır, paketler zamanlama kararlarının ardından gönderilir
- UL için Packet Scheduler henüz oluşturulmamış
- Kontrol mesajları ideal bir kanal üzerinden gönderiliyor

Gelecek

- Daha etkin bir Radio Resource Management (RRM) tasarımı
- UL Packet Scheduler, …

NS3 Sınıfları ve Fonksiyonları

LteHelper

- LteNetDevice objelerini üretmeyi ve yönetmeyi kolaylaştırır
- İçinde iki farklı cihaz çeşidi tanımlanmıştır:
  `DEVICE_TYPE_USER_EQUIPMENT` ve `DEVICE_TYPE_ENODEB`.

Sınıf

```
void AddMobility (Ptr<LtePhy> phy, Ptr<MobilityModel> m),
NetDeviceContainer Install (NodeContainer c, NetDeviceType type),
void RegisterUeToTheEnb (Ptr<UeNetDevice> ue, Ptr<EnbNetDevice>
    enb),
void EnableLogComponents (void),
void AddDownlinkChannelRealization (Ptr<MobilityModel>
    enbMobility, Ptr<MobilityModel> ueMobility, Ptr<LtePhy> phy);
```

fonksiyonlarıyla yönetilir.

NS3 Sınıfları ve Fonksiyonları

LogComponentEnable

- NS3 içinde istenen sınıflar için farklı seviyelerde loglama yapılmasını sağlar. Loglama türleri `LOG_NONE, LOG_ERROR, LOG_WARN, LOG_DEBUG, LOG_FUNCTION, LOG_LOGIC, LOG_ALL` olarak sıralanabilir.

NodeContainer

- NS3'te tanımlanan düğümleri tutmaya yarar. İçindeki vektörde bir iterator yardımıyla gezilebilir. Aşağıdaki fonksiyonlar yardımıyla kullanılır:

```
Ptr<Node> Get (uint32_t index) const;
void Create (uint32_t numberOfNodes);
void Add (Ptr<Node> node);
```

NS3 Sınıfları ve Fonksiyonları

NetDeviceContainer

- NS3'te üretilen cihazların tutulduğu sınıftır. Cihaz olarak bahsedilen, Linux'te device olarak kullandığımız yapıdır. Ağ aygıtlarının çıkışındaki bağlantı noktalarını temsil eder. İçindeki vektörde bir iterator yardımıyla gezilebilir.

```
Ptr<NetDevice> Get (uint32_t i) const;
void Add (Ptr<NetDevice> device);
```

NodeContainer

- Ipv4 adreslerini düğümlere atama işini kolaylaştırmak için hazırlanmış bir sınıftır. Kullanıcıya sağladığı fonksiyonlar:

```
void SetBase (Ipv4Address network, Ipv4Mask mask,
    Ipv4Address base = "0.0.0.1");
Ipv4InterfaceContainer Assign (const NetDeviceContainer &c);
```

NS3 Sınıfları ve Fonksiyonları

InternetStackHelper

- Düğümlere IP/TCP/UDP fonksiyonelliğini sağlamak için hazırlanmış kolaylaştırıcı bir sınıftır. Kullanıcılara sağladığı fonksiyonlar:

```
void SetRoutingHelper (const Ipv4RoutingHelper &routing);
void Install (Ptr<Node> node) const;
void SetTcp (std::string tid);
void SetIpv4StackInstall (bool enable);
```

Ipv4InterfaceContainer

- Ipv4 adresi atanmış cihazları tutmak için oluşturulmuştur. İçindeki vektörde bir iterator yardımıyla gezilebilir.

```
void Add (Ptr<Ipv4> ipv4, uint32_t interface);
std::pair<Ptr<Ipv4>, uint32_t> Get (uint32_t i) const;
Ipv4Address GetAddress (uint32_t i, uint32_t j = 0) const;
```

NS3 Sınıfları ve Fonksiyonları

EnbLteDevice

- NetDevice sınıfından türetilmiş LteNetDevice sınıfından türemiştir. Genel NS3 cihazalarına ek olarak aşağıdaki fonksiyonlar eklenmiştir:

```
void SetUeManager (Ptr<UeManager> m);
void SetMacEntity (Ptr<EnbMacEntity> m);
void SendIdealPdcchMessage (void);
```

ConstantPositionMobilityModel

- Düğümlerin başlangıçtaki pozisyonlarını korumaları için üretilmiştir. Diğerleri: `RandomWaypoint, GaussMarkov, ...`

```
Vector DoGetPosition (void) const;
void DoSetPosition (const Vector &position);
Vector DoGetVelocity (void) const;
```

NS3 Sınıfları ve Fonksiyonları

UdpServerHelper

- UDP paketleri oluşturan basit sunucu uygulamaları oluşturmaya yarar. Aşağıdaki fonksiyonlar yardımıyla kullanılır:

```
void SetAttribute (std::string n, const AttributeValue &v);
ApplicationContainer Install (NodeContainer c);
Ptr<UdpServer> GetServer (void);
```

UdpClientHelper

- Yukarıdaki sınıfın istemci versiyonu. Benzer fonksiyonlar içeriyor.

NS3 Sınıfları ve Fonksiyonları

ApplicationContainer

- Üretilen uygulamaların referanslarını içindeki bir vektörde tutar. Aşağıdaki fonksiyonlar yardımıyla kullanılır:

```
void Add (Ptr<Application> application);
Ptr<Application> Get (uint32_t i) const;
void Start (Time start);
void Stop (Time stop);
```

IpcsClassifierRecord

- Bu sınıfı WiMAX için oluşturulmuş olan kütüphaneden kullanıyor. Ip paketleri için sınıflandırma kayıtları tutar. Tutulan kayıt isimlerini `srcAddress, srcMask, dstAddress, dstMask, srcportLow, srcPortHigh, dstPortLow, dstportHigh, L4 protocol, priority` şeklinde sıralayabiliriz.

NS3 Sınıfları ve Fonksiyonları

RadioBearerInstance

- UE ile eNB arasındaki akışların atandığı LTE radyo bearer'ları temsil eder. İçinde iki farklı yön tanımlanmıştır: `DIRACTION_TYPE_UL` ve `DIRECTION_TYPE_DL`. Üç farklı bearer tipi tanımlanmıştır: `TYPE_SRB1, TYPE_SRB2, TYPE_DRB`. İlki düşük öncelikli sinyalleşmede, ikincisi yüksek öncelikli sinyalleşmede, sonuncusu ise veri iletişiminde kullanılır. Aşağıdaki fonksiyonlar yardımıyla kullanılır:

```
void SetBearerDirection (BearerDirection direction);
void SetBearerType (BearerType type);
void SetQosParameters (Ptr<BearerQosParameters> qosPar);
void SetRlcEntity (Ptr<RlcEntity> rlc);
void SetIpcsClassifierRecord (IpcsClassifierRecord* c);
```

RLC: Radio Link Control

NS3'te Basit Bir Uygulama

- Şu aşamada basit bir LTE uygulamasını inceleyebiliriz.

Kütüphaneler

```
#include "ns3/core-module.h"
#include "ns3/network-module.h"
#include "ns3/applications-module.h"
#include "ns3/mobility-module.h"
#include "ns3/config-store-module.h"
#include "ns3/internet-module.h"
#include "ns3/lte-module.h"
#include <iostream>
#include "ns3/global-route-manager.h"
NS_LOG_COMPONENT_DEFINE ("lte-itu-blg609e");
using namespace ns3;
```

NS3'te Basit Bir Uygulama

```
int nbUE = 3;
LteHelper lte;

//lte.EnableLogComponents ();
LogComponentEnable("UdpClient", LOG_LEVEL_INFO);
LogComponentEnable("UdpServer", LOG_LEVEL_INFO);

// CREATE NODE CONTAINER AND CREATE LTE NODES
NodeContainer ueNodes;
NodeContainer enbNodes;
ueNodes.Create(nbUE);
enbNodes.Create(1);
```

NS3'te Basit Bir Uygulama

```
// CREATE DEVICE CONTAINER, INSTALL DEVICE TO NODE
NetDeviceContainer ueDevs, enbDevs;
ueDevs = lte.Install(ueNodes,
    LteHelper::DEVICE_TYPE_USER_EQUIPMENT);
enbDevs = lte.Install(enbNodes, LteHelper::DEVICE_TYPE_ENODEB);

// INSTALL INTERNET STACKS
InternetStackHelper stack;
stack.Install(ueNodes);
stack.Install(enbNodes);
Ipv4AddressHelper address;
address.SetBase("10.1.1.0", "255.255.255.0");
Ipv4InterfaceContainer UEinterfaces = address.Assign(ueDevs);
Ipv4InterfaceContainer ENBinterface = address.Assign(enbDevs);
```

NS3'te Basit Bir Uygulama

```
// MANAGE LTE NET DEVICES
Ptr<EnbNetDevice> enb;
enb = enbDevs.Get(0)->GetObject<EnbNetDevice> ();

std::vector<Ptr<UeNetDevice>> ue(nbUE);
for (int i = 0; i < nbUE; i++)
{
    ue.at(i) = ueDevs.Get(i)->GetObject<UeNetDevice> ();
    lte.RegisterUeToTheEnb(ue.at(i), enb);
}
```

NS3'te Basit Bir Uygulama

```
// CONFIGURE DL and UL SUB CHANNELS

// Define a list of sub channels for the downlink
std::vector<int> dlSubChannels;
for (int i = 0; i < 25; i++) {
    dlSubChannels.push_back(i);
}

// Define a list of sub channels for the uplink
std::vector<int> ulSubChannels;
for (int i = 50; i < 100; i++) {
    ulSubChannels.push_back(i);
}
```

NS3'te Basit Bir Uygulama

```
enb->GetPhy()->SetDownlinkSubChannels(dlSubChannels);
enb->GetPhy()->SetUplinkSubChannels(ulSubChannels);

for (int i = 0; i < nbUE; i++) {
    ue.at(i)->GetPhy()->SetDownlinkSubChannels(dlSubCh…);
    ue.at(i)->GetPhy()->SetUplinkSubChannels(ulSubCh…);
}

// CONFIGURE MOBILITY
Ptr<ConstantPositionMobilityModel> enbMobility =
    CreateObject<ConstantPositionMobilityModel> ();
enbMobility->SetPosition(Vector(0.0, 0.0, 0.0));
lte.AddMobility(enb->GetPhy(), enbMobility);
```

NS3'te Basit Bir Uygulama

```
for (int i = 0; i < nbUE; i++)
{
    Ptr<ConstantVelocityMobilityModel> ueMobility =
        CreateObject<ConstantVelocityMobilityModel> ();
    ueMobility->SetPosition(Vector(30.0, 0.0, 0.0));
    ueMobility->SetVelocity(Vector(100.0, 100.0, 100.0));

    lte.AddMobility(ue.at(i)->GetPhy(), ueMobility);

    lte.AddDownlinkChannelRealization(enbMobility,
        ueMobility, ue.at(i)->GetPhy());
}
```

NS3'te Basit Bir Uygulama

```
// CONGIFURE CLIENT SERVER APPLICATION
UdpServerHelper udpServer;
ApplicationContainer serverApp;
UdpClientHelper udpClient;
ApplicationContainer clientApp;

udpServer = UdpServerHelper(100);
serverApp = udpServer.Install(ueNodes.Get(0));
serverApp.Start(Seconds(0.02));
serverApp.Stop(Seconds(2));
```

NS3'te Basit Bir Uygulama

```
udpClient = UdpClientHelper(UEinterfaces.GetAddress(0), 100);
udpClient.SetAttribute("MaxPackets", UintegerValue(1200));
udpClient.SetAttribute("Interval", TimeValue(Seconds(0.12)));
udpClient.SetAttribute("PacketSize", UintegerValue(800));
clientApp = udpClient.Install(enbNodes.Get(0));
clientApp.Start(Seconds(0.01));
clientApp.Stop(Seconds(2));

//CREATE RADIO BEARER
Ptr<RadioBearerInstance> bearer = ;
    CreateObject<RadioBearerInstance>(); // RBI
bearer->SetBearerDirection(RBI::DIRECTION_TYPE_DL);
bearer->SetBearerType(RBI::BEARER_TYPE_DRB);
```

NS3'te Basit Bir Uygulama

```
IpcsClassifierRecord *ipcs = new
IpcsClassifierRecord( UEinterfaces.GetAddress(0),
    "255.255.255.0", ENBinterface.GetAddress(0),
    "255.255.255.0", 100, 100, 0, 10000, 17, 1);
bearer->SetIpcsClassifierRecord(ipcs);

enb->GetRrcEntity()->AddDownlinkNgbrBearer(bearer);

Simulator::Stop (Seconds (5.0));
Simulator::Run ();
Simulator::Destroy ();
```

NS3'te Basit Bir Uygulama

```
Waf: Entering directory `~/workspace/ns-3-dev/build'
Waf: Leaving directory `~/workspace/ns-3-dev/build'
'build' finished successfully (0.826s)
Starting simulation.....
TraceDelay TX 800 bytes to 10.1.1.1 Uid: 10 Time: 0.01
TraceDelay TX 800 bytes to 10.1.1.1 Uid: 133 Time: 0.13
TraceDelay: RX 788 bytes from 10.1.1.4 Sequence Number: 1
    Uid: 133 TXtime: +129999999.0ns RXtime:
    +131000000.0ns Delay: +1000001.0ns
…
TraceDelay TX 800 bytes to 10.1.1.1 Uid: 1933 Time: 1.93
TraceDelay: RX 788 bytes from 10.1.1.4 Sequence Number: 16
    Uid: 1933 TXtime: +1929999984.0ns RXtime:
    +1931000000.0ns Delay: +1000016.0ns
Done.
```

Birkaç NS3 Sınıfı Daha

AmcModule

- 3GPP TSG-RAN WG1 - R1-081483 standardında verilen Adaptive Modulation and Coding Scheme'nin gerçeklenmiş halidir. Bu sınıf

```
int GetMcsFromCqi (int cqi)
int GetTbSizeFromMcs (int mcs)
double GetSpectralEfficiencyFromCqi (int cqi)
vector<int> CreateCqiFeedbacks (vector<double> sinr)
MCS: Modulation and Coding Scheme
CQI: Channel Quality Indication
```

fonksiyonları yardımıyla yönetilir.

Birkaç NS3 Sınıfı Daha

BearerQosParameters

- Bearer'lara atanan servis kalitesi parametreleri bu kısımda ayarlanmaktadır. İçinde iki çeşit BearerQosType tanımlanmıştır: `BEARER_TYPE_GBR` ve `BEARER_TYPE_NGBR`.

```
BearerQosParameters(int qci, bool apec, bool apev, gbr, mbr);
void SetBearerQosType (BearerQosType QosType);      double
BearerQosType GetBearerQosType (void) const;
void SetMaxDelay (double targetDelay);
double GetMaxDelay (void) const;
qci: the Qos Class Identifier
apec: the Allocation and Retention Priority (ARP) of pre-
      emption capability
apev: the ARP of pre-emption vulnerability
gbr: the maximum bit rate to guarantee
mbr: the minimum bit rate to guarantee
```

Birkaç NS3 Sınıfı Daha

ChannelRealization

- Kanaldaki kayıpları yöneten kendi içinde bütün bir yayılma modelidir. Kullanıcıya sağladığı fonksiyonlar:

```
void SetJakesFadingLossModel (Ptr<JakesFadingLossModel> l);
void SetPathLossModel (Ptr<PathLossModel> l);
void SetShadowingLossModel (Ptr<ShadowingLossModel> l);
void SetPenetrationLossModel (Ptr<PenetrationLossModel> l);
Bu 4 Set fonksiyonu için Get fonksiyonları
```

olarak listelenebilir.

Sunum Sonu

- Dinlediğiniz için teşekkürler, SORULARINIZ?