

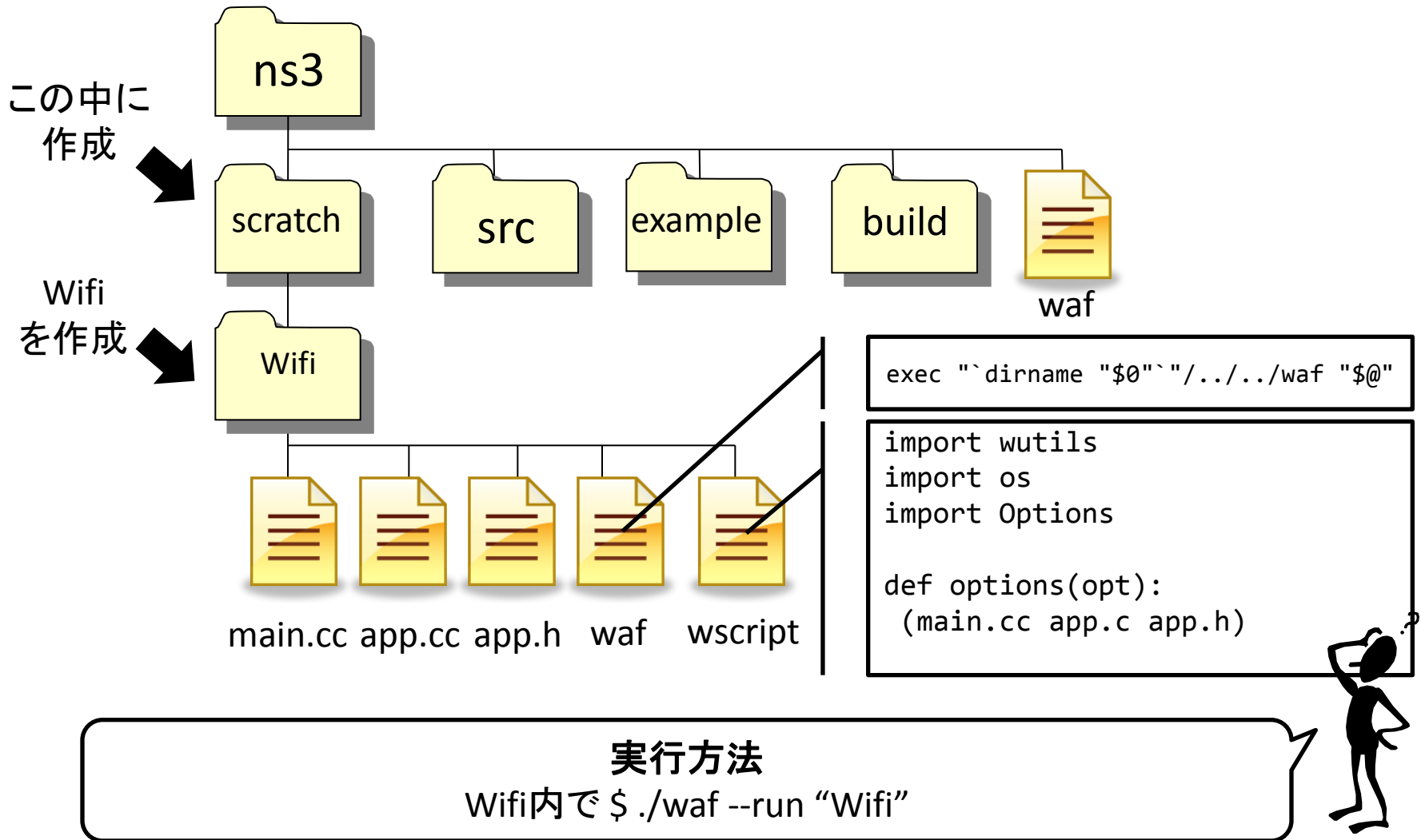
ns-3シミュレーションの実行方法

静岡大学 情報学研究科

杉山佑介

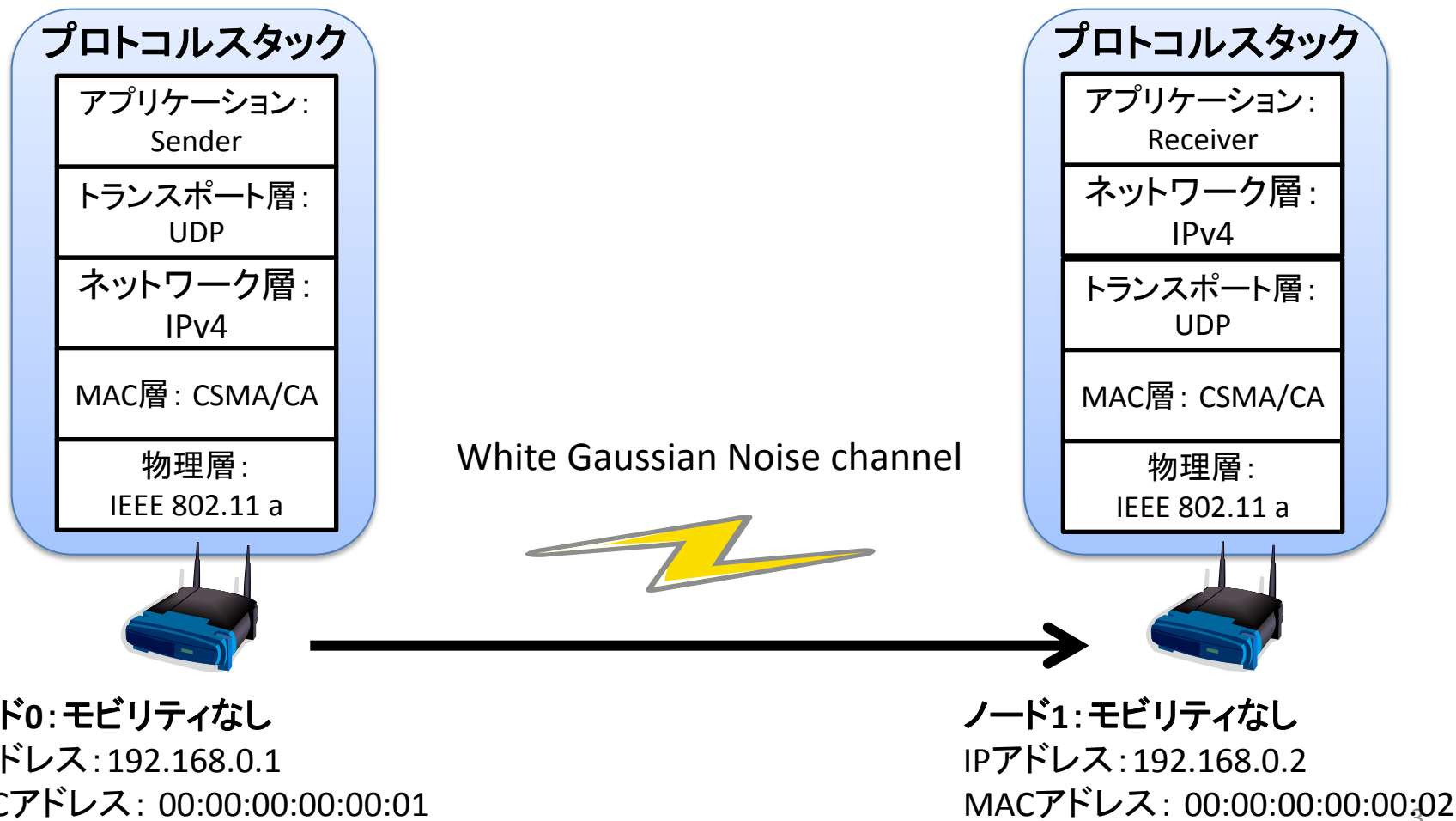
作成日: 2014/07/22

プログラムの実装・実行方法 ~作業用ディレクトリの作成~



Wifiシミュレーションプログラムの実行

- ノード0からノード1間の距離を変化させて、ノード0からノード1へパケットが届いた割合を求めるシミュレーション



プログラムの実装・実行方法

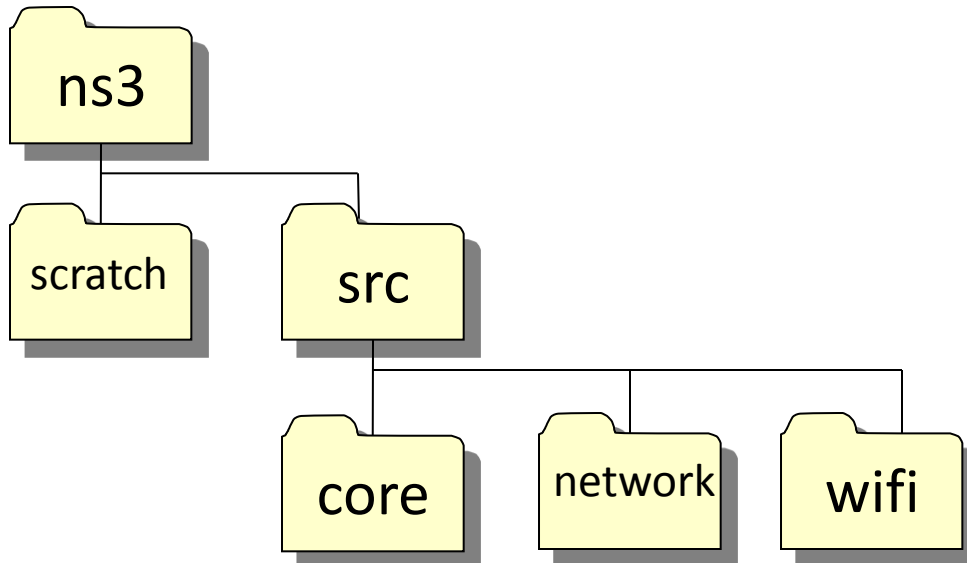
～main関数～

- include
- 引数処理
- ノード定義
 - Wifi(物理層, MAC層)接続
 - ネットワーク層・トランスポート層接続
 - モビリティ接続
 - アプリケーション層接続
- Dataコレクション
 - 送信回数カウント
 - 受信回数カウント
 - SQLiteのフォーマットで書き込み
- シミュレーションの実行と終了

プログラムの実装・実行方法

~main関数:include~

```
#include "ns3/core-module.h"  
#include "ns3/network-module.h"  
#include "ns3/mobility-module.h"  
#include "ns3/wifi-module.h"  
#include "ns3/internet-module.h"  
#include "ns3/stats-module.h"
```



モジュールのinclude

- core-module
 - ・オリジナルの型関係
 - ・オリジナルのポインタ関係
 - ・イベント構造関係
- network-module
 - ・パケットの構造関係
 - ・ソケット関係
- mobility-module
 - ・ノードの動き関係
- wifi-module
 - ・IEEE 802.11の無線関係
- internet-module
 - ・ipv4, ipv6関係
 - ・UDP, TCP関係
- stats-module
 - ・DataCollection関係
 - ・gnuplot, sqlite関係



プログラムの実装・実行方法

~main関数: 引数処理~

```
double distance = 90;  
CommandLine cmd;  
cmd.AddValue ("distance", "Distance apart to place nodes (in meters).", distance);  
cmd.Parse (argc, argv);
```

ヘルプの表示

```
$ ./waf --run "WifiFD --help"
```

デフォルト値のまま実行

```
$ ./waf --run "WifiFD"
```

distanceを変えて実行

```
$ ./waf --run "WifiFD --distance=80"
```

```
$ ./waf --run "WifiFD --distance=100"
```

ヘルプの表示結果

Program Arguments:

```
--distance: Distance apart to place  
nodes (in meters). [90]
```



プログラムの実装・実行方法

～main関数：ノード定義～

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



プログラムの実装・実行方法

～main関数：Wifi接続部～

```
WifiHelper wifi = WifiHelper::Default ();  
NqosWifiMacHelper wifiMac = NqosWifiMacHelper::Default ();  
wifiMac.SetType ("ns3::AdhocWifiMac");  
YansWifiPhyHelper wifiPhy = YansWifiPhyHelper::Default ();  
YansWifiChannelHelper wifiChannel = YansWifiChannelHelper::Default ();  
wifiPhy.SetChannel (wifiChannel.Create ());  
NetDeviceContainer nodeDevices = wifi.Install (wifiPhy, wifiMac, nodes);
```



White Gaussian Noise channel



プログラムの実装・実行方法

～main関数：ネットワーク・トランスポート層接続～

```
InternetStackHelper internet;  
internet.Install (nodes);  
Ipv4AddressHelper ipAddrs;  
ipAddrs.SetBase (“192.168.0.0”, “255.255.255.0”);  
ipAddrs.Assign (nodeDevices);
```

プロトコルスタック

トランスポート層:
UDP

ネットワーク層:
IPv4

CSMA/CA

IEEE 802.11 a



ノード0

IPアドレス: 192.168.0.1

MACアドレス: 00:00:00:00:00:01

White Gaussian Noise channel



プロトコルスタック

トランスポート層:
UDP

ネットワーク層:
IPv4

CSMA/CA

IEEE 802.11 a



ノード1

IPアドレス: 192.168.0.2

MACアドレス: 00:00:00:00:00:02

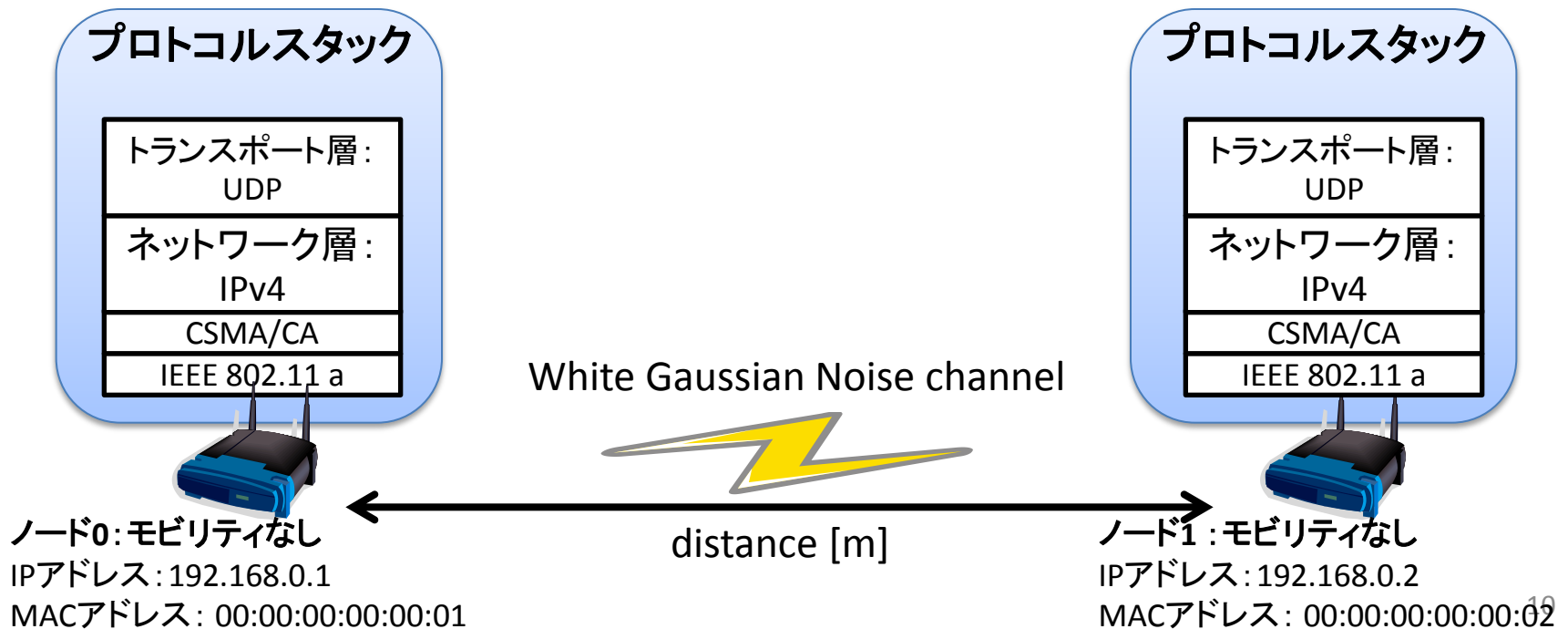
プログラムの実装・実行方法

~main関数:モビリティ接続~

```

| MobilityHelper mobility;
| Ptr<ListPositionAllocator> positionAlloc = CreateObject<ListPositionAllocator>();
| positionAlloc->Add (Vector (0.0, 0.0, 0.0));
| positionAlloc->Add (Vector (0.0, distance, 0.0));
| mobility.SetPositionAllocator (positionAlloc);
| mobility.Install (nodes);

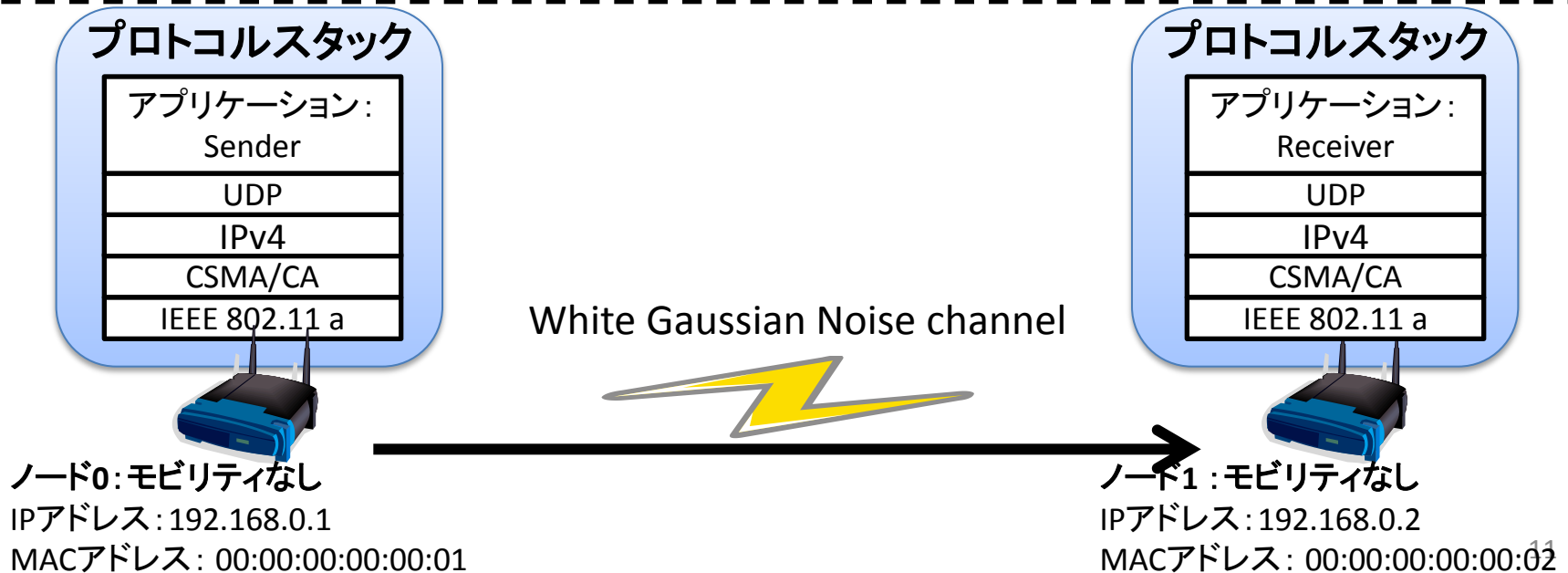
```



プログラムの実装・実行方法

~main関数:アプリケーション層接続~

```
Ptr<Node> source = NodeList::GetNode (0);  
Ptr<Node> sink   = NodeList::GetNode (1);  
Ptr<Sender> sender      = CreateObject<Sender>();  
Ptr<Receiver> receiver = CreateObject<Receiver>();  
source ->AddApplication (sender);  
sink   ->AddApplication (receiver);  
sender  ->SetStartTime (Seconds (0));  
receiver->SetStartTime (Seconds (0));  
Config::Set ("/NodeList/0/ApplicationList/*/$Sender/Destination", Ipv4AddressValue  
("192.168.0.2"));
```



プログラムの実装・実行方法

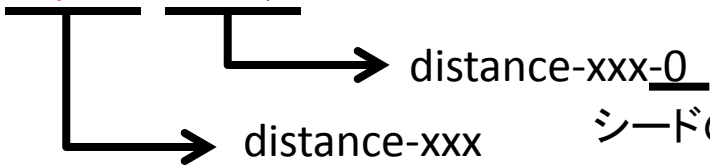
～main関数: Dataコレクション～

```
DataCollector data;
```

```
data.DescribeRun (experiment, strategy, input, runID);
```

distance-xxx-0
シードの値

distance-xxx



DataCollector::
data

DataCollectorとは

- ・データを集める役割を持つ

DataCollectorで集めるデータとは

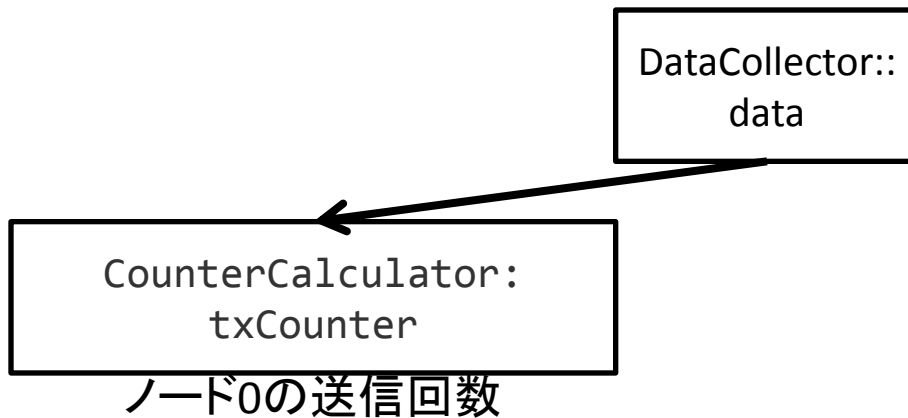
- ・CounterCalculator: 回数のデータ(送信や受信の回数)
- ・TimeMinMaxAvgTotalCalculator: 時間のデータ(遅延の時間)
- ・PacketSizeMinMaxAvgTotalCalculator: パケットのデータ(パケットのサイズ等)



プログラムの実装・実行方法

~main関数：送信回数カウント~

```
Ptr<CounterCalculator<uint32_t> > txCounter =  
CreateObject<CounterCalculator<uint32_t> >();  
txCounter->SetKey ("tx");  
txCounter->SetContext ("node[0]");  
Config::Connect ("/NodeList/0/DeviceList/*/$ns3::WifiNetDevice/Phy/PhyTxBegin",  
MakeBoundCallback (&SenderTx, txCounter));  
data.AddDataCalculator (txCounter);
```



ns-3ではノード番号とデバイス番号を指定して、関数のトレースが可能

“/NodeList/0/DeviceList/*/\$ns3::WifiNetDevice/Phy/PhyTxBegin”の説明

ノード番号

デバイス番号(*は全て)

物理層の関数

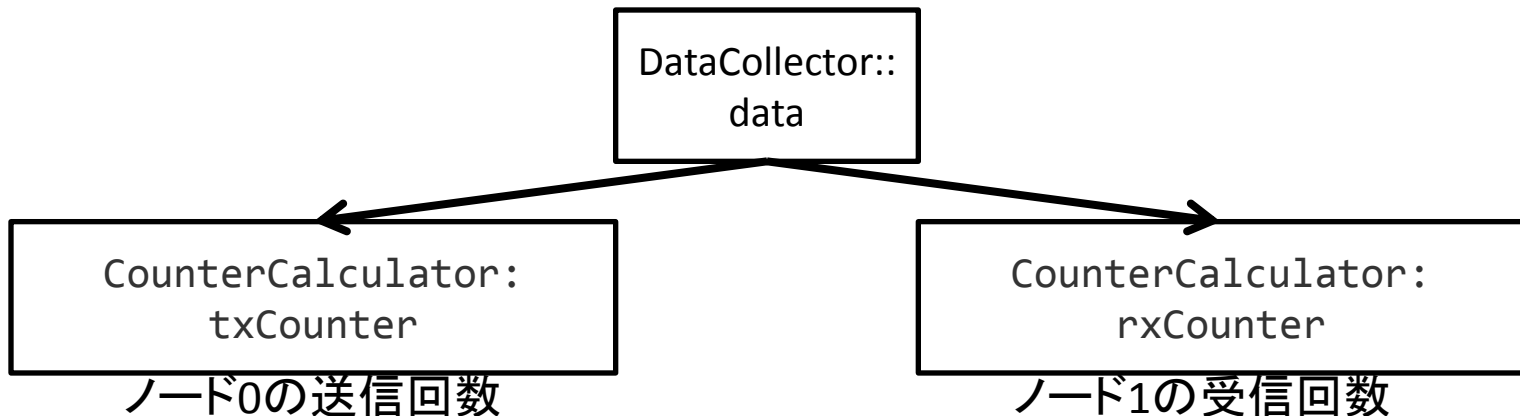
この関数が呼ばれたらSenderTxも呼ばれる



プログラムの実装・実行方法

~main関数：受信回数カウント~

```
Ptr<CounterCalculator<uint32_t> > rxCounter =  
CreateObject<CounterCalculator<uint32_t> >();  
rxCounter->SetKey (“rx”);  
rxCounter->SetContext (“node[1]”);  
Config::Connect (“/NodeList/1/DeviceList/*/$ns3::WifiNetDevice/Phy/PhyRxEnd”,  
MakeBoundCallback (&ReceiverRx, rxCounter));  
data.AddDataCalculator (rxCounter);
```



プログラムの実装・実行方法

~main関数:SQLiteのフォーマットで書き込み~

```
Ptr<DataOutputInterface> output = 0;  
output = CreateObject<SqliteDataOutput>();  
output->Output (data);
```

シミュレーション結果の算出方法

1. distanceを変えてシミュレーション (`$./waf --run "WifiFD --distance=xxx "`)
2. data.dbに結果が溜まる
3. SQLのクエリを発行して結果を作成

クエリ発行の例

```
CMD="select exp.input, avg((rx.value * 100) / tx.value)  
from Singletons rx, Singletons tx, Experiments exp  
where rx.run = tx.run AND tx.run = exp.run AND  
      rx.variable='rx' AND tx.variable='tx'  
group by exp.input ¥  
order by abs(exp.input) ASC;"  
sqlite3 -noheader data.db "$CMD" > distance.data  
sed -i.bak "s/|/ /" distance.data  
rm distance.data.bak
```



```
data.data  
25 0.0  
50 5.0  
75 11.4  
...  
175 100.0
```



プログラムの実装・実行方法

~main関数:シミュレーションの実行と終了~

```
 Simulator::Run ();
```

※ 注意 この間でSQLiteのフォーマットで書き込みをする

```
 Simulator::Destroy ();
```

シミュレーションの実行

- Simulator::Run ()

メモリの解放

- Simulator::Destroy ()

