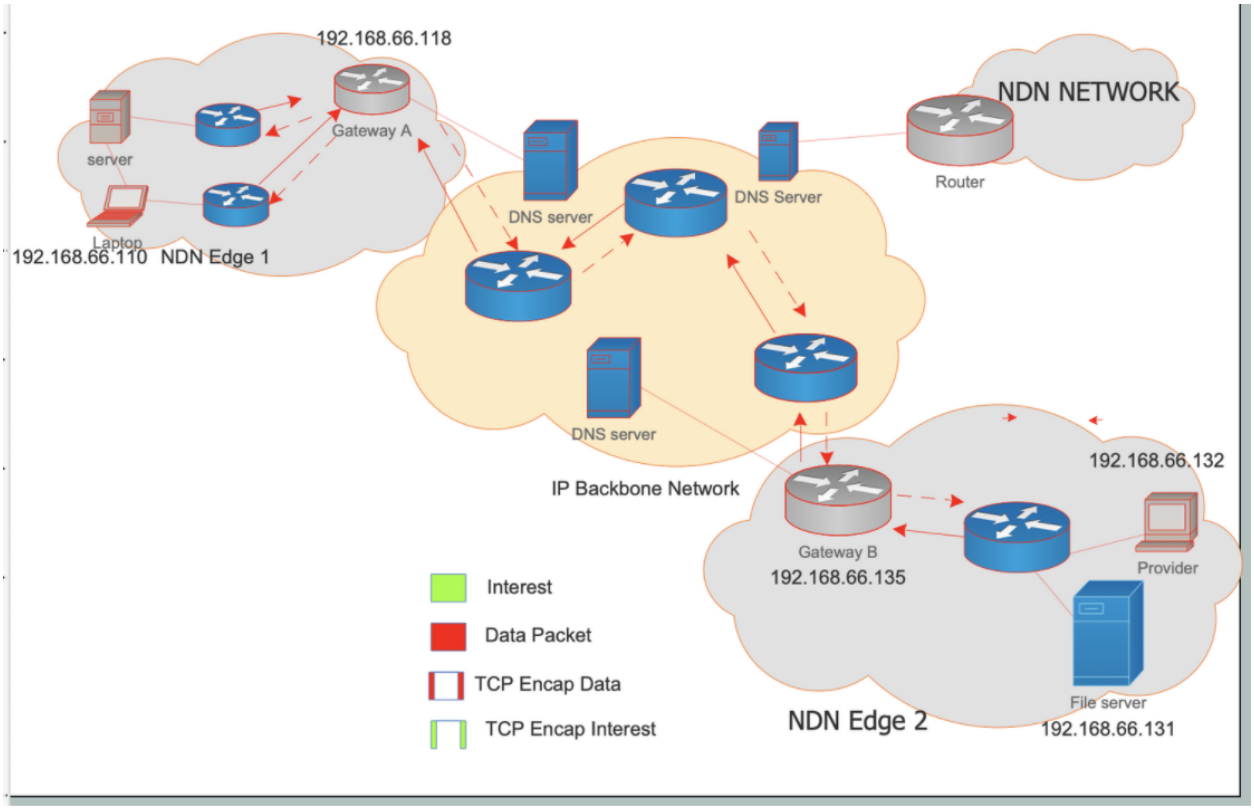
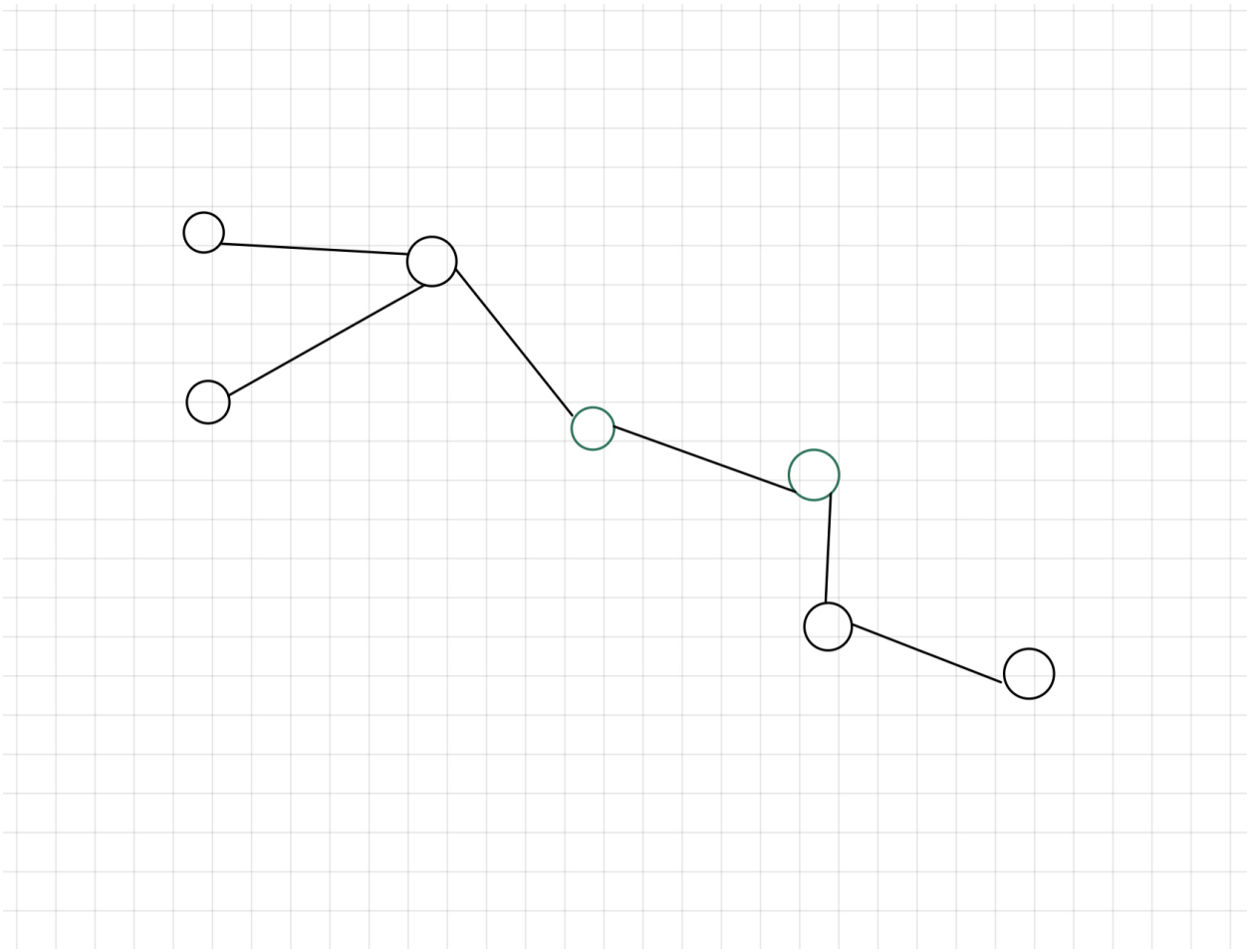
Build Topologies.

Created	@July 11, 2022 9:10 PM
Type	Task 🔨
Status	
Epic	
Sprint	
Priority	P1 🔥
Tasks	
Timeline	
Product Manager	
Engineers	





Sample

```
# topo-6-node.txt

#   /------\                                                /------\
#   | Src1 |<--+                                          +-->| Dst1 |
#   \------/    \                                        /    \------/
#              \                                      /
#                +-->/------\   "bottleneck"  /------\<-+
#                    | Rtr1 |<===============>| Rtr2 |
#                +-->\------/                 \------/<-+
#               /                                      \
#   /------\   /                                        \    /------\
#   | Src2 |<--+                                          +-->| Dst2 |
#   \------/                                                   \------/

router

# node  comment     yPos    xPos
Src1   NA        1       3
Src2   NA        3       3
Rtr1   NA        2       5
Rtr2   NA        2       7
Dst1   NA        1       9
Dst2   NA        3       9
Dst3   NA       6       9

link

# srcNode   dstNode     bandwidth   metric  delay   queue
Src1        Rtr1        10Mbps      1        10ms    20
Src2        Rtr1        10Mbps      1        10ms    20
Rtr1        Rtr2        1Mbps       1        10ms    20
```

```
Dst1        Rtr2        10Mbps      1        10ms    20
Dst2        Rtr2        10Mbps      1        10ms    20
```

Topologies

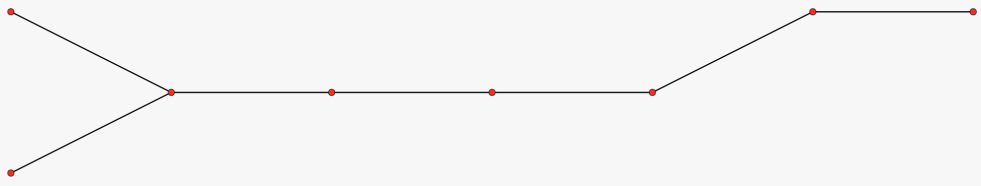
```
#ndn-ip-ndn.txt

#   /------\
#   | Src1 |<--+
#   \------/    \
#              \
#                +-->/------\ "bottleneck"./--ip---\         /--ip--\               /------\<--+
#                    | Rtr1 |<============>| Rtr2 |<=======>|  Rtr3 |<=============>|  Rtr4 |
#                +-->\------/              \------/          \-----/                \------/<--+
#               /                                                                            \
#   /------\    /                                                                              \      /----\        /-----\
#   | Src2 |<--+                                                                                +--->| Dst1 |+---->| Dst2 |
#   \------/                                                                                          \----/        \-----/
#
#

# node  comment     yPos    xPos
Src1   NA          1       3
Src2   NA          3       3
Rtr1   NA          2       5
Rtr2   NA          3       7
Rtr3   NA          2       9
Rtr4   NA          2       11
Dst1   NA          3       13
Dst2   NA          3       15


link

# srcNode   dstNode     bandwidth   metric  delay   queue
Src1        Rtr1        10Mbps      1        10ms    20
Src2        Rtr1        10Mbps      1        10ms    20
Rtr1        Rtr2        1Mbps       1        10ms    20
Rtr2        Rtr3        1Mbps       1        10ms    20
Rtr3        Rtr4        1Mbps       1        10ms    20
Rtr4        Dst1        10Mbps      1        10ms    20
Dst1        Dst2        10Mbps      1        10ms    20
```



读取txt的拓扑文件

```
#ns3/scratch/helloworld.cc

..
int
main(int argc, char* argv[])
```

```
{
  CommandLine cmd;
  cmd.Parse(argc, argv);

  AnnotatedTopologyReader topologyReader("", 25);
  topologyReader.SetFileName("src/ndnSIM/examples/topologies/ndn-ip-ndn.txt");
  topologyReader.Read();

...
```

主函数

8:39 Jul Mon

```
// ndn-congestion-topo-plugin.cpp

#include "ns3/core-module.h"
#include "ns3/network-module.h"
#include "ns3/ndnSIM-module.h"

namespace ns3 {

int
main(int argc, char* argv[])
{
  CommandLine cmd;
  cmd.Parse(argc, argv);

//载入 拓扑文件
  AnnotatedTopologyReader topologyReader("", 25);
  topologyReader.SetFileName("src/ndnSIM/examples/topologies/ndn-ip-ndn.txt");
  topologyReader.Read();

   // Getting containers for the consumer/producer
  Ptr<Node> consumer1 = Names::Find<Node>("Src1");
  Ptr<Node> consumer2 = Names::Find<Node>("Src2");
  Ptr<Node> producer1 = Names::Find<Node>("Dst1");
  Ptr<Node> producer2 = Names::Find<Node>("Dst2");
  Ptr<Node> Iprouter1 = Names::Find<Node>("Rtr2");
  Ptr<Node> Iprouter2 = Names::Find<Node>("Rtr3");
  Ptr<Node> iGate1 = Names::Find<Node>("Rtr1");
  Ptr<Node> iGate2 = Names::Find<Node>("Rtr4");



  // Install NDN stack on ndn node
  ndn::StackHelper ndnHelper;
  //ndnHelper.SetOldContentStore("ns3::ndn::cs::Lru", "MaxSize", "10000");
  ndnHelper.Install(counsmer1);
  ndnHelper.Install(counsmer2);
  ndnHelper.Install(producer1);
  ndnHelper.Install(producer2);
  ndnHelper.Install(iGate1);
  ndnHelper.Install(iGate2);

  //install ipv4 stack on ip node
  Ipv4StackHelper ipv4Helper;
  ipve4Helper.Install(Iprouter1);
  ipve4Helper.Install(Iprouter2);


  // Choosing forwarding strategy
  ndn::StrategyChoiceHelper::InstallAll("/prefix", "/localhost/nfd/strategy/best-route");

  // Installing global routing interface on all nodes
  ndn::GlobalRoutingHelper ndnGlobalRoutingHelper;
  ndnGlobalRoutingHelper.InstallAll();


  ndn::AppHelper consumerHelper("ns3::ndn::ConsumerCbr");
  consumerHelper.SetAttribute("Frequency", StringValue("100")); // 100 interests a second

  // on the first consumer node install a Consumer application
  // that will express interests in /dst1 namespace
  consumerHelper.SetPrefix("/dst1");
```

```
  consumerHelper.Install(consumer1);

  // on the second consumer node install a Consumer application
  // that will express interests in /dst2 namespace
  consumerHelper.SetPrefix("/dst2");
  consumerHelper.Install(consumer2);

  ndn::AppHelper producerHelper("ns3::ndn::Producer");
  producerHelper.SetAttribute("PayloadSize", StringValue("1024"));

  // Register /dst1 prefix with global routing controller and
  // install producer that will satisfy Interests in /dst1 namespace
  ndnGlobalRoutingHelper.AddOrigins("/dst1", producer1);
  producerHelper.SetPrefix("/dst1");
  producerHelper.Install(producer1);

  // Register /dst2 prefix with global routing controller and
  // install producer that will satisfy Interests in /dst2 namespace
  ndnGlobalRoutingHelper.AddOrigins("/dst2", producer2);
  producerHelper.SetPrefix("/dst2");
  producerHelper.Install(producer2);

  // Calculate and install FIBs
  ndn::GlobalRoutingHelper::CalculateRoutes();

  Simulator::Stop(Seconds(20.0));

  Simulator::Run();
  Simulator::Destroy();

  return 0;
}

} // namespace ns3

int
main(int argc, char* argv[])
{
  return ns3::main(argc, argv);
}
```

22:45 11 Jul

```
// ndn-congestion-topo-plugin.cpp

#include "ns3/core-module.h"
#include "ns3/network-module.h"
#include "ns3/ndnSIM-module.h"
#include "ns3/internet-stack-helper.h"

namespace ns3 {

int
main(int argc, char* argv[])
{
  CommandLine cmd;
  cmd.Parse(argc, argv);

  AnnotatedTopologyReader topologyReader("", 25);
  topologyReader.SetFileName("src/ndnSIM/examples/topologies/ndn-ip-ndn.txt");
  topologyReader.Read();

  //Install NDN stack on all nodes
  //ndn::StackHelper ndnHelper;
  //ndnHelper.SetOldContentStore("ns3::ndn::cs::Lru", "MaxSize", "10000");
  //ndnHelper.InstallAll();

  // Choosing forwarding strategy
  //ndn::StrategyChoiceHelper::InstallAll("/prefix", "/localhost/nfd/strategy/best-route");

  // Installing global routing interface on all nodes
  ndn::GlobalRoutingHelper ndnGlobalRoutingHelper;
  //ndnGlobalRoutingHelper.InstallAll();
```

```
 // Getting containers for the consumer/producer
Ptr<Node> consumer1 = Names::Find<Node>("Src1");
Ptr<Node> consumer2 = Names::Find<Node>("Src2");
Ptr<Node> producer1 = Names::Find<Node>("Dst1");
Ptr<Node> producer2 = Names::Find<Node>("Dst2");
Ptr<Node> Iprouter1 = Names::Find<Node>("Rtr2");
Ptr<Node> Iprouter2 = Names::Find<Node>("Rtr3");
Ptr<Node> iGate1 = Names::Find<Node>("Rtr1");
Ptr<Node> iGate2 = Names::Find<Node>("Rtr4");




// Install NDN stack on ndn node
ndn::StackHelper ndnHelper;
//ndnHelper.InstallAll();
//ndnHelper.SetOldContentStore("ns3::ndn::cs::Lru", "MaxSize", "10000");
ndnHelper.Install(consumer1);
ndnHelper.Install(consumer2);
ndnHelper.Install(producer1);
ndnHelper.Install(producer2);
ndnHelper.Install(iGate1);
ndnHelper.Install(iGate2);

//install ipv4 stack on ip nodes
ns3::InternetStackHelper stack;
stack.Install(Iprouter1);
stack.Install(Iprouter2);
ndnHelper.Install(Iprouter1);
ndnHelper.Install(Iprouter2);

//ndnGlobalRoutingHelper.InstallAll();

//Installing global routing interface on all nodes
ndnGlobalRoutingHelper.Install(consumer1);
ndnGlobalRoutingHelper.Install(consumer2);
ndnGlobalRoutingHelper.Install(producer1);
ndnGlobalRoutingHelper.Install(producer2);
ndnGlobalRoutingHelper.Install(iGate1);
ndnGlobalRoutingHelper.Install(iGate2);







//ipv4 address
//Ipv4AddressHelper address;



// Getting containers for the consumer/producer
//Ptr<Node> consumer1 = Names::Find<Node>("Src1");
//Ptr<Node> consumer2 = Names::Find<Node>("Src2");

//Ptr<Node> producer1 = Names::Find<Node>("Dst1");
//Ptr<Node> producer2 = Names::Find<Node>("Dst2");

ndn::AppHelper consumerHelper("ns3::ndn::ConsumerCbr");
consumerHelper.SetAttribute("Frequency", StringValue("100")); // 100 interests a second

// on the first consumer node install a Consumer application
// that will express interests in /dst1 namespace
consumerHelper.SetPrefix("/dst1");
consumerHelper.Install(consumer1);

// on the second consumer node install a Consumer application
// that will express interests in /dst2 namespace
consumerHelper.SetPrefix("/dst2");
consumerHelper.Install(consumer2);

ndn::AppHelper producerHelper("ns3::ndn::Producer");
producerHelper.SetAttribute("PayloadSize", StringValue("1024"));

// Register /dst1 prefix with global routing controller and
// install producer that will satisfy Interests in /dst1 namespace
ndnGlobalRoutingHelper.AddOrigins("/dst1", producer1);
producerHelper.SetPrefix("/dst1");
```

```
  producerHelper.Install(producer1);

  // Register /dst2 prefix with global routing controller and
  // install producer that will satisfy Interests in /dst2 namespace
  ndnGlobalRoutingHelper.AddOrigins("/dst2", producer2);
  producerHelper.SetPrefix("/dst2");
  producerHelper.Install(producer2);

  // Calculate and install FIBs
  ndn::GlobalRoutingHelper::CalculateRoutes();

  Simulator::Stop(Seconds(20.0));

  Simulator::Run();
  Simulator::Destroy();

  return 0;
}

} // namespace ns3

int
main(int argc, char* argv[])
{
  return ns3::main(argc, argv);
}
```

```
// ndn-congestion-topo-plugin.cpp

#include "ns3/core-module.h"
#include "ns3/network-module.h"
#include "ns3/ndnSIM-module.h"
#include "ns3/internet-stack-helper.h"
#include "ns3/ipv4-address-helper.h"
#include "ns3/ipv4-interface-address.h"
#include "ns3/csma-helper.h"
#include "ns3/ipv4-address.h"



namespace ns3 {

int
main(int argc, char* argv[])
{
  CommandLine cmd;
  cmd.Parse(argc, argv);

  AnnotatedTopologyReader topologyReader("", 25);
  topologyReader.SetFileName("src/ndnSIM/examples/topologies/ndn-ip-ndn.txt");
  topologyReader.Read();

  //Install NDN stack on all nodes
  //ndn::StackHelper ndnHelper;
  //ndnHelper.SetOldContentStore("ns3::ndn::cs::Lru", "MaxSize", "10000");
  //ndnHelper.InstallAll();

  // Choosing forwarding strategy
  //ndn::StrategyChoiceHelper::InstallAll("/prefix", "/localhost/nfd/strategy/best-route");

  // Installing global routing interface on all nodes
  //ndn::GlobalRoutingHelper ndnGlobalRoutingHelper;
  //ndnGlobalRoutingHelper.InstallAll();



   // Getting containers for the consumer/producer
  Ptr<Node> consumer1 = Names::Find<Node>("Src1");
  Ptr<Node> consumer2 = Names::Find<Node>("Src2");
  Ptr<Node> producer1 = Names::Find<Node>("Dst1");
  Ptr<Node> producer2 = Names::Find<Node>("Dst2");
  Ptr<Node> Iprouter1 = Names::Find<Node>("Rtr2");
  Ptr<Node> Iprouter2 = Names::Find<Node>("Rtr3");
  Ptr<Node> iGate1 = Names::Find<Node>("Rtr1");
  Ptr<Node> iGate2 = Names::Find<Node>("Rtr4");
```

```
// Install NDN stack on ndn node
ndn::StackHelper ndnHelper;
//ndnHelper.InstallAll();
//ndnHelper.SetOldContentStore("ns3::ndn::cs::Lru", "MaxSize", "10000");
ndnHelper.Install(consumer1);
ndnHelper.Install(consumer2);
ndnHelper.Install(producer1);
ndnHelper.Install(producer2);
ndnHelper.Install(iGate1);
ndnHelper.Install(iGate2);



//install ipv4 stack on ip nodes
ns3::InternetStackHelper stack;
stack.Install(Iprouter1);
stack.Install(Iprouter2);
stack.Install(iGate1);
stack.Install(iGate2);

NodeContainer net (Iprouter1, iGate1);
ns3::CsmaHelper csma;
NetDeviceContainer ndc = csma.Install (net);

NodeContainer net2 (Iprouter2, iGate2);
ns3::CsmaHelper csma2;
NetDeviceContainer ndc2 = csma2.Install (net2);



//ndnGlobalRoutingHelper.InstallAll();

//Installing global routing interface on all nodes
//ndnGlobalRoutingHelper.Install(consumer1);
//ndnGlobalRoutingHelper.Install(consumer2);
//ndnGlobalRoutingHelper.Install(producer1);
//ndnGlobalRoutingHelper.Install(producer2);
//ndnGlobalRoutingHelper.Install(iGate1);
//ndnGlobalRoutingHelper.Install(iGate2);







//ipv4 address
ns3::Ipv4AddressHelper ipv4Address;
ipv4Address.SetBase (Ipv4Address ("192.168.1.0"), Ipv4Mask("/24"));
Ipv4InterfaceContainer ic = ipv4Address.Assign (ndc);

//ipv4 address
ns3::Ipv4AddressHelper ipv4Address2;
ipv4Address2.SetBase (Ipv4Address ("192.168.114.0"), Ipv4Mask("/24"));
Ipv4InterfaceContainer ic2 = ipv4Address2.Assign (ndc2);







// Getting containers for the consumer/producer
//Ptr<Node> consumer1 = Names::Find<Node>("Src1");
//Ptr<Node> consumer2 = Names::Find<Node>("Src2");

//Ptr<Node> producer1 = Names::Find<Node>("Dst1");
//Ptr<Node> producer2 = Names::Find<Node>("Dst2");

ndn::AppHelper consumerHelper("ns3::ndn::ConsumerCbr");
consumerHelper.SetAttribute("Frequency", StringValue("100")); // 100 interests a second

// on the first consumer node install a Consumer application
// that will express interests in /dst1 namespace
consumerHelper.SetPrefix("/dst1");
consumerHelper.Install(consumer1);
```

```
  // on the second consumer node install a Consumer application
  // that will express interests in /dst2 namespace
  consumerHelper.SetPrefix("/dst2");
  consumerHelper.Install(consumer2);

  ndn::AppHelper producerHelper("ns3::ndn::Producer");
  producerHelper.SetAttribute("PayloadSize", StringValue("1024"));

  // Register /dst1 prefix with global routing controller and
  // install producer that will satisfy Interests in /dst1 namespace
  //ndnGlobalRoutingHelper.AddOrigins("/dst1", producer1);
  producerHelper.SetPrefix("/dst1");
  producerHelper.Install(producer1);

  // Register /dst2 prefix with global routing controller and
  // install producer that will satisfy Interests in /dst2 namespace
  //ndnGlobalRoutingHelper.AddOrigins("/dst2", producer2);
  producerHelper.SetPrefix("/dst2");
  producerHelper.Install(producer2);

  // Calculate and install FIBs
  //ndn::GlobalRoutingHelper::CalculateRoutes();

  Simulator::Stop(Seconds(20.0));

  Simulator::Run();
  Simulator::Destroy();

  return 0;
}

} // namespace ns3

int
main(int argc, char* argv[])
{
  return ns3::main(argc, argv);
}
```