SPTM - Dizajn i implementacija TOR modela u NS-3 simulatoru

Generated by Doxygen 1.9.8

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

Chapter 4

Namespace Documentation

4.1 ns3 Namespace Reference

Functions

- NS_LOG_COMPONENT_DEFINE ("UdpEchoClientApplication")
- NS_OBJECT_ENSURE_REGISTERED (UdpEchoClient)
- NS_LOG_COMPONENT_DEFINE ("UdpEchoServerApplication")
- NS_OBJECT_ENSURE_REGISTERED (UdpEchoServer)

4.1.1 Function Documentation

4.1.1.1 NS_LOG_COMPONENT_DEFINE() `[1/2]`

```
ns3::NS_LOG_COMPONENT_DEFINE (
             "UdpEchoClientApplication"   )
```

4.1.1.2 NS_LOG_COMPONENT_DEFINE() `[2/2]`

```
ns3::NS_LOG_COMPONENT_DEFINE (
             "UdpEchoServerApplication"   )
```

4.1.1.3 NS_OBJECT_ENSURE_REGISTERED() `[1/2]`

```
ns3::NS_OBJECT_ENSURE_REGISTERED (
             UdpEchoClient   )
```

4.1.1.4 NS_OBJECT_ENSURE_REGISTERED() `[2/2]`

```
ns3::NS_OBJECT_ENSURE_REGISTERED (
             UdpEchoServer   )
```

Chapter 5

Class Documentation

5.1 PacketTrace Struct Reference

Public Attributes

- double sendTime
- std::string path

5.1.1 Member Data Documentation

5.1.1.1 path

```
std::string PacketTrace::path
```

5.1.1.2 sendTime

```
double PacketTrace::sendTime
```

The documentation for this struct was generated from the following file:

- TOR.cc

Chapter 6

File Documentation

6.1 TOR.cc File Reference

This code will allow transfer of packets, udp-echo-client.cc located in src/applications/model allows sending of packets with string passed as data, while udp-echo-server.cc in the same directory allows the packets to be received.

```
#include "ns3/core-module.h"
#include "ns3/network-module.h"
#include "ns3/internet-module.h"
#include "ns3/point-to-point-module.h"
#include "ns3/applications-module.h"
#include "ns3/ipv4-global-routing-helper.h"
#include "ns3/netanim-module.h"
#include "ns3/mobility-module.h"
#include <iostream>
#include <string>
#include <vector>
#include <fstream>
```

Classes

- struct PacketTrace

Functions

- NS_LOG_COMPONENT_DEFINE ("SimpleTOR")
- static void SentPacket (Ptr< const Packet > p)

  This is a function which handles the sending of packets.
- static void ReceivedPacket (Ptr< const Packet > p)

  This is a function which handles the process of receiving packets.
- void Ratio ()

  This function allows the tor network statistic to be printed out, which prints all relevant data about the network simulation.
- int main (int argc, char ∗argv[ ])

  This is the main function which gets called once the simulation starts.

Variables

- std::map< uint32_t, PacketTrace > packetTracker
- std::vector< std::string > nodeNames = {"Client", "Entry", "Relay1", "Relay2", "Relay3", "Exit", "Destination"}

  Here, we are declaring node names for all nodes in the network topology.
- static Time g_firstPacketTime = Seconds(0.0)

  This is the variable which gets assigned a value of the time when the first packet was transmitted.
- static Time g_lastPacketTime = Seconds(0.0)

  This is the variable which gets assigned a value of the time when the last packet was transmitted.
- static bool g_firstPacket = true

  This is a boolean value for the first packet which can be true or false.
- static std::map< uint32_t, double > PacketStartTimes
- static double totalDelay = 0.0

  This is a variable, type double, which means it stores decimal values, and it stores the value of the total delay of the network.
- static int packetCount = 0
- uint32_t m_bytes_sent = 0

  This is an unsigned variable type, which means that it cannot have negative values, and the range for the numbers it can have is from 0 to 2^32. This variable stores the amount of sent bytes.
- uint32_t m_bytes_received = 0

  This variable stores the amount of received bytes.
- uint32_t m_packets_sent = 0

  This variable stores the amount of sent packets.
- uint32_t m_packets_received = 0

  This variable stores the amount of received packets.
- double m_time = 0
- std::map< uint32_t, double > m_delayTable

6.1.1 Detailed Description

This code will allow transfer of packets, udp-echo-client.cc located in src/applications/model allows sending of packets with string passed as data, while udp-echo-server.cc in the same directory allows the packets to be received.

6.1.2 Function Documentation

6.1.2.1 main()

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

This is the main function which gets called once the simulation starts.

If output.txt exists in the directory where the simulation is started, it will be replaced with output.txt with no content. std::ofstream output_file("output.txt"); // Replace output_txt if it was created before.

Simulation time is assigned to this variable.

This variable stores the maximum amount of packets.

This allows the user to specify parameters which could be changed while running the simulation, such as: ./ns3 run scratch/TOR.cc – -maxPackets=5 -simulationTime=30

Node container is specified.

7 nodes are being created.

Data rate parameter is being assigned to the point-to-point link.

Delay parameter is being assigned to the point-to-point link.

This holds a collection of network devices, such as wifi or ethernet devices.

This holds IPv4 addresses of network devices.

This installs a network stack on nodes.

This assigns IPv4 addresses to network devices.

This creates a P2P link between nodes 0 and 1.

This specifies the initialisation time of the server.

This specifies the time when the server stops responding.

This creates and allocates a mobility model and installs it for the nodes.

This will create an xml file which can later be viewed in NetAnim which provides network visualisation.

This sets the maximum amounts of packets per trace file.

This specifies node descriptions.

Apart from this, we choose node colors.

Network tracing is enabled here.

This allows the simulation to start.

This frees up resources which were allocated to the simulation while it was running.

If 0 is returned, that means that code execution was successfull.

6.1.2.2 NS_LOG_COMPONENT_DEFINE()

```
NS_LOG_COMPONENT_DEFINE (
              "SimpleTOR"  )
```

6.1.2.3 Ratio()

```
void Ratio ( )
```

This function allows the tor network statistic to be printed out, which prints all relevant data about the network simulation.

6.1.2.4 ReceivedPacket()

```
static void ReceivedPacket (
            Ptr< const Packet > p )  [static]
```

This is a function which handles the process of receiving packets.

This will allow an output file to be created, also, data can be appended to this file because of std::ios::app which is included. If we omit that, we would keep creating the file without appending data to it, which wouldn't be a solution because we need to track the data for every packet so that we can eventually use gnuplot to plot the captured data, not just the final data for the final packet which was sent.

This variable stores the time when the packet was received.

This variable stores the time when the packet was sent.

This variable stores the packet delay which is calculated by subtracting start time from the end time.

This represents the time when the packet was received.

This variable stores the throughput value, which is calculated by multiplying received bytes by 8 and dividing that by the duration. The value for the throughput is displayed in bits per second.

Output file gets created with contents of:

- Duration;
- Sent packets;
- Received packets;
- Throughput in bits per second;
- Packet delay.

This allows the output file to be closed after writing.

This message prints the time when the packet was received.

6.1.2.5 SentPacket()

```
static void SentPacket (
            Ptr< const Packet > p )  [static]
```

This is a function which handles the sending of packets.

Parameters

p	This is the packet that is being sent.

The number of sent bytes gets increased, it adds the size of the packet that is being sent to the current amount of sent bytes.

The variable which tracks the number of sent packets increases.

This if statement checks if the first packet is being sent, and if it is, the variable which is declared for storing the time of the first sent packets actually gets that value assigned to itself.

The start time of each sent packet gets extracted. This applies to every packet and the Uid of that packet gets extracted so that the message which prints at what time the specific packet got sent.

This message prints when a certain packet got sent.

6.1.3 Variable Documentation

6.1.3.1 g_firstPacket

```
bool g_firstPacket = true  [static]
```

This is a boolean value for the first packet which can be true or false.

6.1.3.2 g_firstPacketTime

```
Time g_firstPacketTime = Seconds(0.0)  [static]
```

This is the variable which gets assigned a value of the time when the first packet was transmitted.

6.1.3.3 g_lastPacketTime

```
Time g_lastPacketTime = Seconds(0.0)  [static]
```

This is the variable which gets assigned a value of the time when the last packet was transmitted.

6.1.3.4 m_bytes_received

```
uint32_t m_bytes_received = 0
```

This variable stores the amount of received bytes.

6.1.3.5 m_bytes_sent

```
uint32_t m_bytes_sent = 0
```

This is an unsigned variable type, which means that it cannot have negative values, and the range for the numbers it can have is from 0 to 2^32. This variable stores the amount of sent bytes.

6.1.3.6 m_delayTable

```
std::map<uint32_t, double> m_delayTable
```

6.1.3.7 m_packets_received

```
uint32_t m_packets_received = 0
```

This variable stores the amount of received packets.

6.1.3.8 m_packets_sent

```
uint32_t m_packets_sent = 0
```

This variable stores the amount of sent packets.

6.1.3.9 m_time

```
double m_time = 0
```

6.1.3.10 nodeNames

```
std::vector<std::string> nodeNames = {"Client", "Entry", "Relay1", "Relay2", "Relay3", "Exit",
"Destination"}
```

Here, we are declaring node names for all nodes in the network topology.

6.1.3.11 packetCount

```
int packetCount = 0  [static]
```

@btief This is a variable, type int, and it stores packet count.

6.1.3.12 PacketStartTimes

```
std::map<uint32_t, double> PacketStartTimes  [static]
```

6.1.3.13 packetTracker

```
std::map<uint32_t, PacketTrace> packetTracker
```

6.1.3.14 totalDelay

```
double totalDelay = 0.0  [static]
```

This is a variable, type double, which means it stores decimal values, and it stores the value of the total delay of the network.

6.2 udp-echo-client.cc File Reference

This file allows the sending of packets in TOR.cc which is located in the scratch folder.

```
#include "udp-echo-client.h"
#include "ns3/inet-socket-address.h"
#include "ns3/inet6-socket-address.h"
#include "ns3/ipv4-address.h"
#include "ns3/ipv6-address.h"
#include "ns3/log.h"
#include "ns3/nstime.h"
#include "ns3/packet.h"
#include "ns3/simulator.h"
#include "ns3/socket-factory.h"
#include "ns3/socket.h"
#include "ns3/trace-source-accessor.h"
#include "ns3/uinteger.h"
```

Namespaces

- namespace ns3

Functions

- ns3::NS_LOG_COMPONENT_DEFINE ("UdpEchoClientApplication")
- ns3::NS_OBJECT_ENSURE_REGISTERED (UdpEchoClient)

6.2.1 Detailed Description

This file allows the sending of packets in TOR.cc which is located in the scratch folder.

6.3 udp-echo-server.cc File Reference

This file enables the process of receiving packets in TOR.cc file located in the ../../scratch folder.

```
#include "udp-echo-server.h"
#include "ns3/address-utils.h"
#include "ns3/inet-socket-address.h"
#include "ns3/inet6-socket-address.h"
#include "ns3/ipv4-address.h"
#include "ns3/ipv6-address.h"
#include "ns3/log.h"
#include "ns3/nstime.h"
#include "ns3/packet.h"
#include "ns3/simulator.h"
#include "ns3/socket-factory.h"
#include "ns3/socket.h"
#include "ns3/udp-socket.h"
#include "ns3/uinteger.h"
```

Namespaces

- namespace ns3

Functions

- ns3::NS_LOG_COMPONENT_DEFINE ("UdpEchoServerApplication")
- ns3::NS_OBJECT_ENSURE_REGISTERED (UdpEchoServer)

6.3.1 Detailed Description

This file enables the process of receiving packets in TOR.cc file located in the ../../scratch folder.

Index