My Project

Generated by Doxygen 1.8.4

Sat May 3 2014 21:31:22

Contents

1 Hierarchical Index **1**

1.1 Class Hierarchy . . . 1

2 Class Index **3**

2.1 Class List . . . 3

3 File Index **5**

3.1 File List . . . 5

4 Class Documentation **7**

4.1 Address Class Reference . . . 7

4.1.1 Detailed Description . . . 8

4.1.2 Member Function Documentation . . . 8

4.1.2.1 convertHWAddrToColonFormat . . . 8

4.1.2.2 isSame . . . 8

4.1.2.3 isSameMACAddr . . . 8

4.1.2.4 isSet . . . 8

4.1.2.5 setHostname . . . 8

4.1.2.6 setHWAddr . . . 8

4.1.2.7 setHWAddrFromColonFormat . . . 8

4.1.2.8 setPort . . . 9

4.2 adv Struct Reference . . . 9

4.3 cShared Struct Reference . . . 9

4.3.1 Detailed Description . . . 9

4.4 LossyReceivingPort Class Reference . . . 9

4.4.1 Detailed Description . . . 10

4.4.2 Constructor & Destructor Documentation . . . 10

4.4.2.1 LossyReceivingPort . . . 10

4.4.3 Member Function Documentation . . . 10

4.4.3.1 receivePacket . . . 10

4.5 mySendingPort Class Reference . . . 10

4.5.1 Member Function Documentation . . . 11

4.5.1.1 timerHandler . . . 11
4.6 mySendingPort2 Class Reference . . . 11
4.6.1 Member Function Documentation . . . 11
4.6.1.1 timerHandler . . . 11
4.7 Packet Class Reference . . . 12
4.7.1 Detailed Description . . . 12
4.7.2 Constructor & Destructor Documentation . . . 12
4.7.2.1 Packet . . . 12
4.7.3 Member Function Documentation . . . 13
4.7.3.1 accessHeader . . . 13
4.7.3.2 extractHeader . . . 13
4.7.3.3 fillPayload . . . 13
4.7.3.4 getBufferSize . . . 13
4.7.3.5 getHeaderSize . . . 13
4.7.3.6 getPayload . . . 13
4.7.3.7 getPayloadSize . . . 13
4.7.3.8 makePacket . . . 13
4.7.3.9 setPayloadSize . . . 13
4.7.4 Member Data Documentation . . . 14
4.7.4.1 DEFAULT_PAYLOAD_SIZE . . . 14
4.8 PacketHdr Class Reference . . . 14
4.8.1 Detailed Description . . . 14
4.8.2 Member Function Documentation . . . 15
4.8.2.1 accessInfo . . . 15
4.8.2.2 getIntegerInfo . . . 15
4.8.2.3 getOctet . . . 15
4.8.2.4 getShortIntegerInfo . . . 15
4.8.2.5 getSize . . . 15
4.8.2.6 init . . . 15
4.8.2.7 setHeaderSize . . . 15
4.8.2.8 setIntegerInfo . . . 15
4.8.2.9 setOctet . . . 15
4.8.2.10 setShortIntegerInfo . . . 15
4.9 Port Class Reference . . . 16
4.9.1 Detailed Description . . . 17
4.9.2 Member Function Documentation . . . 17
4.9.2.1 decodeSockAddress . . . 17
4.9.2.2 init . . . 17
4.9.2.3 setSockAddress . . . 17
4.10 ReceivingPort Class Reference . . . 18

4.10.1 Detailed Description . . . 18
4.10.2 Constructor & Destructor Documentation . . . 18
4.10.2.1 ReceivingPort . . . 18
4.10.3 Member Function Documentation . . . 18
4.10.3.1 init . . . 18
4.10.3.2 receivePacket . . . 19
4.10.4 Member Data Documentation . . . 19
4.10.4.1 pkt_ . . . 19
4.11 res Struct Reference . . . 19
4.12 SendingPort Class Reference . . . 19
4.12.1 Detailed Description . . . 20
4.12.2 Member Function Documentation . . . 20
4.12.2.1 init . . . 20
4.12.2.2 sendPacket . . . 21
4.12.2.3 setBroadcast . . . 21
4.12.2.4 setBroadcastOff . . . 21
4.12.2.5 timerHandler . . . 21
4.12.3 Member Data Documentation . . . 21
4.12.3.1 bcastflag_ . . . 21
4.12.3.2 sendingbuf_ . . . 21
4.12.3.3 timer_ . . . 21
4.13 TxTimer Class Reference . . . 21
4.13.1 Detailed Description . . . 22
4.13.2 Member Function Documentation . . . 22
4.13.2.1 startTimer . . . 22
4.13.2.2 stopTimer . . . 22
4.13.2.3 timerProc . . . 22
4.13.3 Member Data Documentation . . . 23
4.13.3.1 port_ . . . 23
4.13.3.2 tdelay_ . . . 23
4.13.3.3 tid_ . . . 23

5 File Documentation **25**

5.1 common.h File Reference . . . 25
5.1.1 Detailed Description . . . 26
5.2 host.cpp File Reference . . . 26
5.2.1 Detailed Description . . . 27
5.3 newport.h File Reference . . . 27
5.3.1 Detailed Description . . . 27
5.4 newport2.h File Reference . . . 27

5.4.1 Detailed Description . . . 27
5.5 router.cpp File Reference . . . 27
5.5.1 Detailed Description . . . 29
5.5.2 Macro Definition Documentation . . . 29
5.5.2.1 lossPercent . . . 29
5.5.2.2 prtTimeToExpire . . . 29
5.5.2.3 rtTimeToExpire . . . 29
5.5.2.4 sleepDelay . . . 29
5.5.2.5 timerWrap . . . 29
5.5.3 Function Documentation . . . 29
5.5.3.1 AddRoutingTableEntry . . . 29
5.5.3.2 CheckPendingRequestTableExpired . . . 29
5.5.3.3 CheckRoutingTableEntryExpired . . . 30
5.5.3.4 CreateConnectionsList . . . 30
5.5.3.5 DeletePendingRequestTableEntry . . . 30
5.5.3.6 DeleteRoutingTableEntry . . . 30
5.5.3.7 Display2DVector . . . 30
5.5.3.8 NodeRecProc . . . 30
5.5.3.9 SearchConnectionsTable . . . 30

Index **32**

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

- Address . . . 7
- adv . . . 9
- cShared . . . 9
- Packet . . . 12
- PacketHdr . . . 14
- Port . . . 16
  - ReceivingPort . . . 18
    - LossyReceivingPort . . . 9
  - SendingPort . . . 19
    - mySendingPort . . . 10
    - mySendingPort2 . . . 11
- res . . . 19
- TxTimer . . . 21

Chapter 2

Class Index

2.1 Class List

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

Address
: An address class . . . 7

adv . . . 9

cShared . . . 9

LossyReceivingPort
: A receiving port simulating link loss and delay . . . 9

mySendingPort . . . 10

mySendingPort2 . . . 11

Packet
: A Packet class . . . 12

PacketHdr
: A Packet Header class . . . 14

Port
: Port class abstacts functions of communication interfaces . . . 16

ReceivingPort
: A Receiving Port Class . . . 18

res . . . 19

SendingPort
: SendingPort is an subclass of Port for sending purpose . . . 19

TxTimer
: A timer to schedule a later event/transmission occured in a port . . . 21

Chapter 3

File Index

3.1 File List

Here is a list of all documented files with brief descriptions:

common.h
: Header file for common.cpp 25

host.cpp
: Implementation of host 26

newport.h
: Inherited class from SendingPort to incorporate acknowledgement and timer functionality . . . 27

newport2.h
: Edited newport.h to avoid acknowledgements 27

router.cpp
: Implementation of router 27

Chapter 4

Class Documentation

4.1 Address Class Reference

An address class.

`#include <common.h>`

Public Member Functions

- Address ()
  Constructor.
- Address (const char *hostname, short port)
  Alternative construcor with parameters.
- bool isSet ()
- void setPort (const short port)
- short getPort ()
  get the port #
- void setHostname (const char *hostname)
  set the hostname
- char * getHostname ()
  get the hostname string pointer
- unsigned char * getHWAddr ()
  get the MAC address
- void setHWAddr (unsigned char *hwaddr)
- void setHWAddrFromColonFormat (const char *colon_seperated_macaddr)
- char * convertHWAddrToColonFormat ()
- Address * clone ()
  function to clone this address
- bool isSame (Address *addr)
- bool isSameMACAddr (Address *addr)

Protected Attributes

- char hostname_ [MAX_HOSTNAME_LENGTH]
  both hostname and ipaddress format (10.0.0.1) could be given as a string
- short port_
  port number for UDP or TCP (Transport layer)
- char * ipaddr_

optional use... ignore....

- unsigned char macaddr_ [MAC_ADDR_LENGTH]

  optional use for Ethernet Socket

4.1.1 Detailed Description

An address class.

Address Class is to handle addresses of unix/linux sockets. For normal sockets, the address used will be a combination of IP address and port. In Socket Programming, IP address itself is usually not enough to distinguish an connection, port # is also needed. For PF_PACKET sockets, the address used is MAC address (HW address). So, we also put macaddr_ as a member variable.

4.1.2 Member Function Documentation

4.1.2.1 char * Address::convertHWAddrToColonFormat ( )

Convert HW Address to Colon Seperated format

4.1.2.2 bool Address::isSame (Address * *addr*) [inline]

Compare whether the two normal "name+port" address is same or not

4.1.2.3 bool Address::isSameMACAddr (Address * *addr*)

Compare if two mac address is same or not. use memcmp to compare each byte.

4.1.2.4 bool Address::isSet ( ) [inline]

Check if an address has already been set or remain uninitialized

4.1.2.5 void Address::setHostname (const char * *hostname*) [inline]

set the hostname

use strcpy function to duplicate a string

4.1.2.6 void Address::setHWAddr (unsigned char * *hwaddr*)

copy mac address

4.1.2.7 void Address::setHWAddrFromColonFormat (const char * *colonmac*)

Function to convert the input MAC address string to bytes. First, check the MAC address is valid

- there are at least 12 Hex characters
- there are no other charcter except colon

4.1.2.8 void Address::setPort (const short *port*) [inline]

set the port # of the Address

The documentation for this class was generated from the following files:

- common.h
- common.cpp

4.2 adv Struct Reference

Public Attributes

- SendingPort * **my_adv_port**

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

- host.cpp

4.3 cShared Struct Reference

Public Attributes

- short **receivingPortNum**
- LossyReceivingPort * **fwdRecvPort**
- mySendingPort * **fwdSendPort**

4.3.1 Detailed Description

Argument to be send to <NodeRecProc>(void *arg)

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

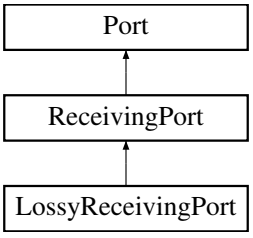
- router.cpp

4.4 LossyReceivingPort Class Reference

A receiving port simulating link loss and delay.

`#include <common.h>`

Inheritance diagram for LossyReceivingPort:



Public Member Functions

- LossyReceivingPort (float lossyratio)
- Packet * receivePacket ()

Protected Attributes

- float loss_ratio_

  how probable a packet will get dropped in receiving
- int secdelay_

  how large is the link propagation delay.

Additional Inherited Members

4.4.1 Detailed Description

A receiving port simulating link loss and delay.

A receiving port which would random drop packets and delay packet reception in 1 second.

4.4.2 Constructor & Destructor Documentation

4.4.2.1 LossyReceivingPort::LossyReceivingPort (float *lossyratio*)

Constructor with parameter (drop probability p) Using a fixed link delay: 1 second

4.4.3 Member Function Documentation

4.4.3.1 Packet * LossyReceivingPort::receivePacket ( )

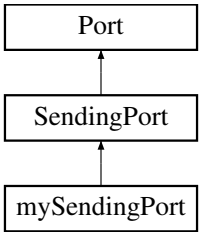
Simulate link delay of 1 seconds. Drop packets with a propabilty equal to loss_ratio

The documentation for this class was generated from the following files:

- common.h
- common.cpp

4.5 mySendingPort Class Reference

Inheritance diagram for mySendingPort:



Public Member Functions

- void **setACKflag** (bool flag)
- bool **isACKed** ()
- void timerHandler ()

Public Attributes

- Packet * **lastPkt_**

Additional Inherited Members

4.5.1 Member Function Documentation

4.5.1.1 void mySendingPort::timerHandler ( ) [inline], [virtual]

TimerHandler is called when the TxTimer expires. This function is virtual. So another child class has to be derived from this base class to use the timer.

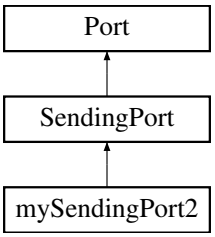
Implements SendingPort.

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

- newport.h

4.6 mySendingPort2 Class Reference

Inheritance diagram for mySendingPort2:



Public Member Functions

- void timerHandler ()

Additional Inherited Members

4.6.1 Member Function Documentation

4.6.1.1 void mySendingPort2::timerHandler ( ) [inline], [virtual]

TimerHandler is called when the TxTimer expires. This function is virtual. So another child class has to be derived from this base class to use the timer.

Implements SendingPort.

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

- newport2.h

4.7 Packet Class Reference

A Packet class.

`#include <common.h>`

Public Member Functions

- Packet (int buffer_length)
- int fillPayload (int size, char *inputstream)
- char * getPayload ()
- void setPayloadSize (int size)
- int getBufferSize ()
- int getPayloadSize ()
- int getHeaderSize ()
- PacketHdr * accessHeader ()
- void extractHeader (char *streambuf)
- int makePacket (char *streambuf)

Static Public Attributes

- static const int DEFAULT_PAYLOAD_SIZE = 512

Protected Attributes

- int size_

  Packet length in Bytes.
- int length_

  Maximum allocated Size of Payload buffer. it is no less than the size_.
- char * payload_

  Payload Pointer.
- PacketHdr * header_

  Header Pointer;.

4.7.1 Detailed Description

A Packet class.

Packet is an entity which contains a set of ordered information bits. This packet object only carries a pointer to payload information, not any socket address information.

4.7.2 Constructor & Destructor Documentation

4.7.2.1 Packet::Packet (int *buffer_length*)

Alternate Packet Constructor: Specify a large buffer.... This is useful to define a packet receiving buffer.

4.7.3 Member Function Documentation

4.7.3.1 PacketHdr* Packet::accessHeader ( ) [inline]

Get the packet header

4.7.3.2 void Packet::extractHeader (char * *streambuf*)

Extract packet header from the incoming stream

4.7.3.3 int Packet::fillPayload (int *size*, char * *inputstream*)

A function to fill payload. user can specify the content of payload for applications like audio/video playback...

4.7.3.4 int Packet::getBufferSize ( ) [inline]

get the size of packet buffer where payload is stored.

4.7.3.5 int Packet::getHeaderSize ( ) [inline]

get the size of the packet header

4.7.3.6 char* Packet::getPayload ( ) [inline]

get a pointer to the payload

Returns

a char pointer

4.7.3.7 int Packet::getPayloadSize ( ) [inline]

get the size of the packet

4.7.3.8 int Packet::makePacket (char * *streambuf*)

Assemble header and payload and headersize in a single stream

4.7.3.9 void Packet::setPayloadSize (int *size*)

set packet size. As packet already has a default buffer, there are two ways to determine the payload

- set size only, let the payload be as it is –> SetPayloadSize
    1. if necessary, adjust the payload buffer size.
- set size and also fill the payload with speficic data (e.g. for Audio and Video Applications...)

See Also

fillPayload

4.7.4 Member Data Documentation

4.7.4.1 const int Packet::DEFAULT_PAYLOAD_SIZE = 512 [static]

Default payload size is 512 Bytes

The documentation for this class was generated from the following files:

- common.h
- common.cpp

4.8 PacketHdr Class Reference

A Packet Header class.

`#include <common.h>`

Public Member Functions

- void init ()
- short getShortIntegerInfo (int position)
- int getIntegerInfo (int position)
- unsigned char * accessInfo ()
- int getSize ()
- void setIntegerInfo (int a, int position)
- void setShortIntegerInfo (short b, int position)
- void setOctet (unsigned char c, int position)
- void setHeaderSize (int len)
- unsigned char getOctet (int position)

Protected Attributes

- unsigned char * info_

  pointer to the header's content
- int length_

  length of the header

4.8.1 Detailed Description

A Packet Header class.

Packethdr is an entity which represents the packet header portion of a packet. Important protocol information fields are stored in the header.

Based on the functions provided in PacketHdr. You can put three datatype in a header by calling those functions:

- 16-bit integer.
- 32-bit integer.
- a 8-bit character (ASCII code).

info_ is defined "unsigned char" because reading a number byte by byte dose not allow any mistakes in type conversion involving the HSB. And both 16-bit and 32-bit data are large enough for any sequence number or other signaling appearing in a short test, so we do not need functions for 64-bit number.

4.8.2 Member Function Documentation

4.8.2.1 unsigned char* PacketHdr::accessInfo ( ) [inline]

Get a pointer to the actual information header .

4.8.2.2 int PacketHdr::getIntegerInfo (int *position*)

read the information filed at "postion" as a 32-bit integer

4.8.2.3 unsigned char PacketHdr::getOctet (int *position*) [inline]

get an octet

4.8.2.4 short PacketHdr::getShortIntegerInfo (int *position*)

read the information filed at "postion" as a short integer

4.8.2.5 int PacketHdr::getSize ( ) [inline]

get the length(size) of the header

4.8.2.6 void PacketHdr::init ( ) [inline]

Clear all information fileds as empty

4.8.2.7 void PacketHdr::setHeaderSize (int *len*) [inline]

Set the header size

4.8.2.8 void PacketHdr::setIntegerInfo (int *a,* int *position*)

set a 4-byte(32-bit) information field with an integer

4.8.2.9 void PacketHdr::setOctet (unsigned char *c,* int *position*) [inline]

set one octet as a desired character An octet in computer networking is an eight bit quantity

4.8.2.10 void PacketHdr::setShortIntegerInfo (short *b,* int *position*)

set a 2-byte(16-bit) information field with an short integer

The documentation for this class was generated from the following files:

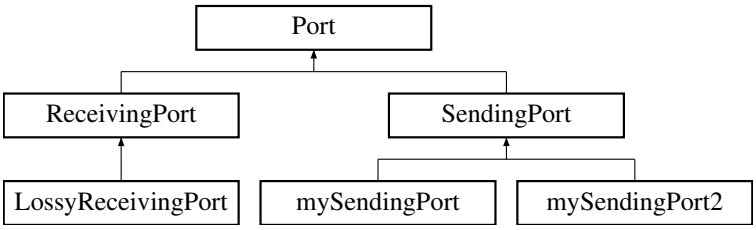
- common.h
- common.cpp

4.9 Port Class Reference

Port class abstacts functions of communication interfaces.

`#include <common.h>`

Inheritance diagram for Port:



Public Member Functions

- Port ()

  Constructor.
- virtual ~Port ()

  Deconstructor.
- virtual void init ()=0
- void setAddress (Address *addr)

  set the port's own address
- void setRemoteAddress (Address *daddr)

  set the address of the port at the other end of communication link
- Address * getRemoteAddr ()

  get the address of the port at the other end of communication link
- void closePort ()

  close the port

Protected Member Functions

- struct sockaddr * setSockAddress (Address *addr, struct sockaddr_in *address)

  cast an Address to socket address format
- void decodeSockAddress (Address *addr, struct sockaddr_in *address)

  cast a socket address to normal address format
- void setHostname (const char *hostname)

  set hostname of local address
- void setPort (const short port)

  set port of local address
- void setRemoteHostname (const char *hostname)

  set hostname of remote address
- void setRemotePort (const short port)

  set port no of a remote address
- int getSock ()

  get the socket file descriptor

Protected Attributes

- Address myaddr_

  The default address of mine.
- Address itsaddr_

  The default address of the other end of communication link.
- struct sockaddr_in mySockAddress_

  IN UDP. this will be INADDR_ANY, not really my IP addr.
- struct sockaddr_in dstSockAddress_

  for every outgoing packet of a connection, the destination address.
- int sockfd_

  socket file descriptor

4.9.1 Detailed Description

Port class abstacts functions of communication interfaces.

Port is an abstract class for the interface to send/receive a packet, whether UDP, TCP Socket, or IP raw socket. The common funcitons for a port are defiend here:

- init
- setSocketAddress

  Port is also associated with a pair of address. One Port send to one Address only, no matter the address is unicast or broadcast.

4.9.2 Member Function Documentation

4.9.2.1 void Port::decodeSockAddress (Address * *addr,* struct sockaddr_in * *address*) [protected]

cast a socket address to normal address format

Function to interpreate hostname and port from the SocketAddress

4.9.2.2 virtual void Port::init ( ) [pure virtual]

Function to initialize the port

Implemented in ReceivingPort, and SendingPort.

4.9.2.3 struct sockaddr * Port::setSockAddress (Address * *addr,* struct sockaddr_in * *address*) [protected]

cast an Address to socket address format

Fill sockaddr_in 'address' structure with information taken from 'addr' and return it cast to a 'struct sockaddr'. It handles following situations:

- if hostname is given as empty "", then INADDR_ANY is used in return
- if an IP address is given, then address could be set directly
- if a hostname is given, call gethostbyname() to find the ip address of the hostname from DNS

The documentation for this class was generated from the following files:

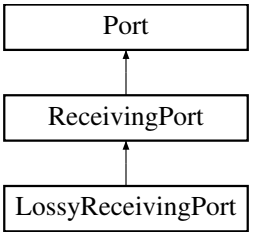
- common.h
- common.cpp

4.10 ReceivingPort Class Reference

A Receiving Port Class.

`#include <common.h>`

Inheritance diagram for ReceivingPort:



Public Member Functions

- ReceivingPort ()
- void init ()
- Packet * receivePacket ()

  The main receive function of receiving port to receive a single packet.

Protected Attributes

- Packet * pkt_
- char * tmpBuffer_

  temporary buffer for packets

Additional Inherited Members

4.10.1 Detailed Description

A Receiving Port Class.

ReceivingPort is an abstract class for the interface to receive a packet. The main function for the receiving port:

- initialize
- Receive Packet

4.10.2 Constructor & Destructor Documentation

4.10.2.1 ReceivingPort::ReceivingPort ( )

Constructor

4.10.3 Member Function Documentation

4.10.3.1 void ReceivingPort::init ( ) [virtual]

Init Funciton to initialize a socket port.

The port binds to its own address, generate a UDP socket.

Bind to local address is one important task in init() Here source address of node itself (myaddr_) does not really be used by bind function of port. The program use INADDR_ANY as the address filled in address parameters of bind(). So, we need an empty hostname with the port number.

When the UDP port is recieving, we need to create a default buffer to store received packet. The data in buffer will be copy to the corresponding flow once the packet's sender address is checked.

Implements Port.

4.10.3.2 Packet * ReceivingPort::receivePacket ( )

The main receive function of receiving port to receive a single packet.

The main receive function of a receiving port. First,check addresses. Then call recvfrom() to get a packet. after this, pkt_ variable stores information of the packet and tmpSockAddr stores the sender information. then, recast tmpSockAddr to itsaddr_. As the socket is not set as non-blocking, the recvfrom() call blocks usually, but if the main program use select() to do synchronized I/O Multiplexing, this call will not block.

The design structure allows a future implementation improvement.

packet size is the maximum allowed packet above UDP or buffer size

4.10.4 Member Data Documentation

4.10.4.1 Packet* ReceivingPort::pkt_ [protected]

This pointer points to the packet. This packet is just received.

The documentation for this class was generated from the following files:

- common.h
- common.cpp

4.11 res Struct Reference

Public Attributes

- LossyReceivingPort * **my_res_port**
- mySendingPort * **my_req_port**

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

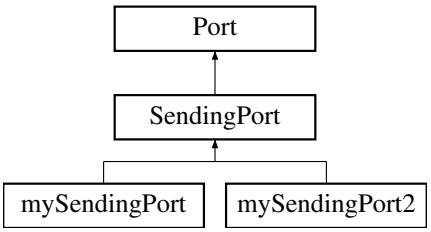
- host.cpp

4.12 SendingPort Class Reference

SendingPort is an subclass of Port for sending purpose.

`#include <common.h>`

Inheritance diagram for SendingPort:



Public Member Functions

- SendingPort (char *hostname, short port)
  Another constructor with local address given.
- virtual ~SendingPort ()
  Deconstructor.
- void init ()
- void sendPacket (Packet *pkt)
- void setBroadcast ()
- void setBroadcastOff ()
- virtual void timerHandler ()=0

Public Attributes

- TxTimer timer_

Protected Attributes

- int bcastflag_
  this flag indicates the port ought to broadcast or unicast a packet.
- char * sendingbuf_

Additional Inherited Members

4.12.1 Detailed Description

SendingPort is an subclass of Port for sending purpose.

SendingPort is an subclass of Port for sending purpose

4.12.2 Member Function Documentation

4.12.2.1 void SendingPort::init ( ) [virtual]

Function to initialize the port

Init Funciton to initialize a socket port. Init will do

- create socket
- bind socket

Notes: Bind to local address is one important task in init() Here source address of node itself (myaddr_) does not really be used by bind function of port. The program use INADDR_ANY as the address filled in address parameters of bind(). So, we need an empty hostname with the port number.

Implements Port.

4.12.2.2 void SendingPort::sendPacket (Packet * *pkt*)

Function to send a packet. The default socket file descriptor will always be used for send() only a sockfd used by the port.

The main send function of UDP Socket Sending Port.

call sendto()

4.12.2.3 void SendingPort::setBroadcast ( ) [inline]

toggle broadcast option on

4.12.2.4 void SendingPort::setBroadcastOff ( ) [inline]

toggle broadcast option off

4.12.2.5 virtual void SendingPort::timerHandler ( ) [pure virtual]

TimerHandler is called when the TxTimer expires. This function is virtual. So another child class has to be derived from this base class to use the timer.

Implemented in mySendingPort, and mySendingPort2.

4.12.3 Member Data Documentation

4.12.3.1 int SendingPort::bcastflag_ [protected]

this flag indicates the port ought to broadcast or unicast a packet.

If broadcast, an broadcast IP address (192.168.255.255, etc) need to be supplied

4.12.3.2 char* SendingPort::sendingbuf_ [protected]

Sending buffer

4.12.3.3 TxTimer SendingPort::timer_

The timer used to schedule future events in a sending port. When this timer expires, the timerHandler will be called.

The documentation for this class was generated from the following files:

- common.h
- common.cpp

4.13 TxTimer Class Reference

A timer to schedule a later event/transmission occured in a port.

`#include <common.h>`

Public Member Functions

- TxTimer (SendingPort *txport)

  TxTimer class.
- void startTimer (float delay)
- void stopTimer ()

Static Public Member Functions

- static void * timerProc (void *arg)

Protected Attributes

- SendingPort * port_
- struct timespec tdelay_
- pthread_t tid_

Friends

- class **SendingPort**

4.13.1 Detailed Description

A timer to schedule a later event/transmission occured in a port.

```
The timer is associated with a SendingPort object.
when the timer expires, the SendingPort::timerHandler() will be called.
```

The intenal design of this class is a little tricky. Usually, LinuxThreads does not support a thread function as a member function of C++ class. I designed timerProc as a static function, and give the class pointer as the 4th argument of the pthread_create

4.13.2 Member Function Documentation

4.13.2.1 void TxTimer::startTimer (float *delay*)

Function to start a timer which will expire after a certain delay

Parameters

delay,:	the timing delay in seconds.

4.13.2.2 void TxTimer::stopTimer ( )

Function to stop a timer

4.13.2.3 void * TxTimer::timerProc (void * *arg*) [static]

Function to create a seperate thread for this timer it will call timerHandler() function of the port_

4.13.3 Member Data Documentation

4.13.3.1 SendingPort* TxTimer::port_ [protected]

port the timer belongs to

4.13.3.2 struct timespec TxTimer::tdelay_ [protected]

delay variable used by nanosleep()

4.13.3.3 pthread_t TxTimer::tid_ [protected]

thread id variable

The documentation for this class was generated from the following files:

- common.h
- common.cpp

Chapter 5

File Documentation

5.1 common.h File Reference

Header file for common.cpp.

```
#include <string.h>
#include <sys/socket.h>
#include <sys/types.h>
#include <unistd.h>
#include <netinet/in.h>
#include <netdb.h>
#include <arpa/inet.h>
#include <cstdlib>
#include <ctime>
```

Classes

- class PacketHdr

  A Packet Header class.
- class Packet

  A Packet class.
- class Address

  An address class.
- class Port

  Port class abstacts functions of communication interfaces.
- class TxTimer

  A timer to schedule a later event/transmission occured in a port.
- class SendingPort

  SendingPort is an subclass of Port for sending purpose.
- class ReceivingPort

  A Receiving Port Class.
- class LossyReceivingPort

  A receiving port simulating link loss and delay.

Macros

- #define **MAX_HOSTNAME_LENGTH** 256
- #define **MAC_ADDR_LENGTH** 6

- #define **MAX_HEADER_SIZE** 256
- #define **DEFAULT_SEND_PORT** 3000
- #define **DEFAULT_RECV_PORT** 4000
- #define **MAXBUFLENGTH** 10000
- #define **MTU_SIZE** 1500
- #define **INADDR_NONE** (-1)

5.1.1 Detailed Description

Header file for common.cpp.

5.2 host.cpp File Reference

Implementation of host.

```
#include "common.h"
#include "newport.h"
#include <iostream>
#include "math.h"
#include <fstream>
#include <stdlib.h>
#include <string>
#include <vector>
#include "newport2.h"
#include <sstream>
#include <algorithm>
#include <cstdlib>
```

Classes

- struct adv
- struct res

Macros

- #define **advertisementInterval** 10
- #define **lossPercent** 0.05

Functions

- void * **advertisement** (void *args)
- void * **receivedata** (void *args)
- int **main** (int argc, const char *argv[])

Variables

- std::vector< int > **content**
- int **host_id** = 1

5.2.1 Detailed Description

Implementation of host.

5.3 newport.h File Reference

Inherited class from SendingPort to incorporate acknowledgement and timer functionality.

```
#include "common.h"
#include <iostream>
```

Classes

- class mySendingPort

5.3.1 Detailed Description

Inherited class from SendingPort to incorporate acknowledgement and timer functionality.

5.4 newport2.h File Reference

Edited newport.h to avoid acknowledgements.

```
#include "common.h"
#include <iostream>
```

Classes

- class mySendingPort2

5.4.1 Detailed Description

Edited newport.h to avoid acknowledgements.

5.5 router.cpp File Reference

Implementation of router.

```
#include <iostream>
#include "common.h"
#include "newport.h"
#include <vector>
#include <list>
```

Classes

- struct cShared

Macros

- #define rtTimeToExpire 30
- #define prtTimeToExpire 30
- #define timerWrap 6000
- #define sleepDelay 1
- #define lossPercent 0.2

Functions

- void Display2DVector (vector< vector< int > > nameOfVector)

  Displays 2dimensional vector/ table.
- void CreateConnectionsList (int argc, char *argv[])

  Creates a list of mapping between receiving, destination and sending ports.
- int SearchConnectionsTable (int receivingPortNum)

  Finds mapping between received port number and destination port number.
- void AddRoutingTableEntry (int contentId, int recPortNum, int numHops)

  Adds a new routing table entry or edits the timer on an already existing entry.
- void UpdateRoutingTableEntryTTL ()

  If the timer wraps around the maximum value this functions resets it.
- void DeleteRoutingTableEntry (int row)

  Delete a certain routing table entry.
- void CheckRoutingTableEntryExpired (int currentTime)

  Check if any entries in the Routing table have expired.
- int getReceivingPort (int requestedContentId)

  Returns Receiving port of the connection for a given Content ID.
- int getNumberHops (int requestedContentId)

  Returns Number of Hops to a given Content ID as per the routing table.
- bool contentIdExists (int requestedContentId)

  Checks if a specific Content ID already exists in the routing table.
- void UpdatePendingRequestTable (int requestedContentId, int requestingHostId, int receivingPort)

  Make a new entry in the Pending request Table or update an already existing entry.
- void UpdatePendingRequestTableTTL ()

  Update the timer in Pending Request table if timer wraps around.
- void DeletePendingRequestTableEntry (int requestedContentId, int requestingHostId)

  Delete a certain pending request table entry using unique combination of content ID and Host ID.
- void CheckPendingRequestTableExpired (int currentTime)

  Check if any entries in the Pending Request table have expired.
- int SearchPendingRequestTable (int contentId, int hostId)

  Return Destination port number associated with a given unique combination of content ID and host ID.
- void ExpiryTimer ()

  Keeps track of which entries in the Routing and Pending request tables need to be deleted.
- void * NodeRecProc (void *arg)

  Receiver thread function which is constantly listening on the receiving port for a given connection.
- void StartNodeThread (pthread_t *thread, vector< int > &ports)

  Sets up the receiving and sending port numbers for a given connection and calls the thread function.
- int main (int argc, char *argv[])

  Main function calls the separate threads for each receiving port.

Variables

- static int globalTimer =0

  Global timer - Clock for the Routing and Pending Request Tables.
- static vector< vector< int > > connectionsList

  Connections table- 2D vector mapping destination address to sending port+"receiving port"(interface)
- static vector< vector< int > > routingTable

  Routing table- content id + receiving port + #hops + time to expire.
- static vector< vector< int > > pendingRequestTable

  Pending request table- Requested id + host id + destination port + time to expire.

5.5.1 Detailed Description

Implementation of router.

5.5.2 Macro Definition Documentation

5.5.2.1 #define lossPercent 0.2

Loss percentage for Lossy receiving port

5.5.2.2 #define prtTimeToExpire 30

Time to expire for Pending Request Table in seconds

5.5.2.3 #define rtTimeToExpire 30

Time to expire for Routing Table in seconds

5.5.2.4 #define sleepDelay 1

Delay to check and update timers in Routing and Pending Request tables

5.5.2.5 #define timerWrap 6000

Time before Timer wraps around

5.5.3 Function Documentation

5.5.3.1 void AddRoutingTableEntry (int *contentId*, int *recPortNum*, int *numHops*)

Adds a new routing table entry or edits the timer on an already existing entry.

If a new advertisement packet is received a new entry is made in the Routing table. If it is an update for an already existing entry the timer is updated.

5.5.3.2 void CheckPendingRequestTableExpired (int *currentTime*)

Check if any entries in the Pending Request table have expired.

If any entries in the Pending Request Table have expired they are deleted

5.5.3.3 void CheckRoutingTableEntryExpired (int *currentTime*)

Check if any entries in the Routing table have expired.

If any entries in the Routing Table have expired they are deleted

5.5.3.4 void CreateConnectionsList (int *argc,* char * *argv[]*)

Creates a list of mapping between receiving, destination and sending ports.

A connection between 2 devices comprises 4 ports. This table maintains the corresponding port numbers for receiving, destination and sending ports that are used by these 2 devices. The port numbers are calculated by a formula.

Parameters

argc	Number of devices connected to + 1
argv[]	Source and the list of connected devices

5.5.3.5 void DeletePendingRequestTableEntry (int *requestedContentId,* int *requestingHostId*)

Delete a certain pending request table entry using unique combination of content ID and Host ID.

This function is essential in case a certain pending request times out.

5.5.3.6 void DeleteRoutingTableEntry (int *row*)

Delete a certain routing table entry.

Parameters

row	Row to be deleted

5.5.3.7 void Display2DVector (vector< vector< int > > *nameOfVector*)

Displays 2dimensional vector/ table.

Parameters

nameOfVector	Vector to be printed

5.5.3.8 void* NodeRecProc (void * *arg*)

Receiver thread function which is constantly listening on the receiving port for a given connection.

Checks whether the packet is of type request, response or advertisement and accordingly services it

5.5.3.9 int SearchConnectionsTable (int *receivingPortNum*)

Finds mapping between received port number and destination port number.

For a given connection between 2 devices, this function returns the corresponding destination number given a specific receiving port it listens on.

Parameters

receivingPort-Num	Receiving Port Number

Index

accessHeader
    Packet, 13
accessInfo
    PacketHdr, 15
AddRoutingTableEntry
    router.cpp, 29
Address, 7
    convertHWAddrToColonFormat, 8
    isSame, 8
    isSameMACAddr, 8
    isSet, 8
    setHWAddr, 8
    setHWAddrFromColonFormat, 8
    setHostname, 8
    setPort, 8
adv, 9

bcastflag_
    SendingPort, 21

cShared, 9
CheckPendingRequestTableExpired
    router.cpp, 29
CheckRoutingTableEntryExpired
    router.cpp, 29
common.h, 25
convertHWAddrToColonFormat
    Address, 8
CreateConnectionsList
    router.cpp, 30

DEFAULT_PAYLOAD_SIZE
    Packet, 14
decodeSockAddress
    Port, 17
DeletePendingRequestTableEntry
    router.cpp, 30
DeleteRoutingTableEntry
    router.cpp, 30
Display2DVector
    router.cpp, 30

extractHeader
    Packet, 13

fillPayload
    Packet, 13

getBufferSize
    Packet, 13
getHeaderSize
    Packet, 13
getIntegerInfo
    PacketHdr, 15
getOctet
    PacketHdr, 15
getPayload
    Packet, 13
getPayloadSize
    Packet, 13
getShortIntegerInfo
    PacketHdr, 15
getSize
    PacketHdr, 15

host.cpp, 26

init
    PacketHdr, 15
    Port, 17
    ReceivingPort, 18
    SendingPort, 20
isSame
    Address, 8
isSameMACAddr
    Address, 8
isSet
    Address, 8

lossPercent
    router.cpp, 29
LossyReceivingPort, 9
    LossyReceivingPort, 10
    LossyReceivingPort, 10
    receivePacket, 10

makePacket
    Packet, 13
mySendingPort, 10
    timerHandler, 11
mySendingPort2, 11
    timerHandler, 11

newport.h, 27
newport2.h, 27
NodeRecProc
    router.cpp, 30

Packet, 12
    accessHeader, 13
    DEFAULT_PAYLOAD_SIZE, 14
    extractHeader, 13

fillPayload, 13
getBufferSize, 13
getHeaderSize, 13
getPayload, 13
getPayloadSize, 13
makePacket, 13
Packet, 12
setPayloadSize, 13
PacketHdr, 14
accessInfo, 15
getIntegerInfo, 15
getOctet, 15
getShortIntegerInfo, 15
getSize, 15
init, 15
setHeaderSize, 15
setIntegerInfo, 15
setOctet, 15
setShortIntegerInfo, 15
pkt_
ReceivingPort, 19
Port, 16
decodeSockAddress, 17
init, 17
setSockAddress, 17
port_
TxTimer, 23
prtTimeToExpire
router.cpp, 29

receivePacket
LossyReceivingPort, 10
ReceivingPort, 19
ReceivingPort, 18
init, 18
pkt_, 19
receivePacket, 19
ReceivingPort, 18
ReceivingPort, 18
res, 19
router.cpp, 27
AddRoutingTableEntry, 29
CheckPendingRequestTableExpired, 29
CheckRoutingTableEntryExpired, 29
CreateConnectionsList, 30
DeletePendingRequestTableEntry, 30
DeleteRoutingTableEntry, 30
Display2DVector, 30
lossPercent, 29
NodeRecProc, 30
prtTimeToExpire, 29
rtTimeToExpire, 29
SearchConnectionsTable, 30
sleepDelay, 29
timerWrap, 29
rtTimeToExpire
router.cpp, 29

SearchConnectionsTable
router.cpp, 30
sendPacket
SendingPort, 20
SendingPort, 19
bcastflag_, 21
init, 20
sendPacket, 20
sendingbuf_, 21
setBroadcast, 21
setBroadcastOff, 21
timer_, 21
timerHandler, 21
sendingbuf_
SendingPort, 21
setBroadcast
SendingPort, 21
setBroadcastOff
SendingPort, 21
setHWAddr
Address, 8
setHWAddrFromColonFormat
Address, 8
setHeaderSize
PacketHdr, 15
setHostname
Address, 8
setIntegerInfo
PacketHdr, 15
setOctet
PacketHdr, 15
setPayloadSize
Packet, 13
setPort
Address, 8
setShortIntegerInfo
PacketHdr, 15
setSockAddress
Port, 17
sleepDelay
router.cpp, 29
startTimer
TxTimer, 22
stopTimer
TxTimer, 22

tdelay_
TxTimer, 23
tid_
TxTimer, 23
timer_
SendingPort, 21
timerHandler
mySendingPort, 11
mySendingPort2, 11
SendingPort, 21
timerProc
TxTimer, 22
timerWrap
router.cpp, 29

TxTimer, 21
port_, 23
startTimer, 22
stopTimer, 22
tdelay_, 23
tid_, 23
timerProc, 22