Network simulator

Generated by Doxygen 1.9.5

1 readme	1
1.1 Requirements with Qt Creator	 1
1.2 Requirements without Qt Creator	 1
1.2.1 A compiler	 1
1.2.2 Qt	 1
1.2.3 cmake	 2
1.2.4 make	 2
1.3 Building the gui without Qt Creator	 2
1.4 Building with Qt Creator	 2
1.5 Source content	 2
1.5.1 MainWindow	 2
1.5.2 Nodeltem	 2
1.5.3 PacketItem	 2
1.5.4 Resources.qrc	 2
1.5.5 main.cpp	 2
2 Hierarchical Index	3
2.1 Class Hierarchy	
2.1 Class fileratchy	 3
3 Class Index	5
3.1 Class List	 5
	_
4 File Index	7
4.1 File List	 7
5 Class Documentation	9
5.1 Client Class Reference	 9
5.1.1 Member Function Documentation	 9
5.1.1.1 getName()	 9
5.1.1.2 print()	 10
5.1.1.3 receive()	 10
5.1.1.4 send() [1/2]	 10
5.1.1.5 send() [2/2]	 10
5.2 MainWindow Class Reference	 10
5.2.1 Detailed Description	 11
5.2.2 Constructor & Destructor Documentation	 11
5.2.2.1 MainWindow()	 11
5.2.3 Member Function Documentation	 12
5.2.3.1 addClient()	 12
5.2.3.2 addLine()	 12
5.2.3.3 addRouter()	 12
5.2.3.4 addServer()	 13
5.2.3.5 dropPacket()	 13
5.2.3.6 getNode()	 13

5.2.3.7 getNodeByName()	14
5.2.3.8 sendPacket()	14
5.3 Node Class Reference	14
5.4 NodeCaller Class Reference	15
5.4.1 Detailed Description	16
5.4.2 Member Function Documentation	16
5.4.2.1 call_print	16
5.4.2.2 call_print_routing_table	17
5.4.2.3 change_max_capacity	17
5.4.2.4 getId()	17
5.4.2.5 send_packet	17
5.5 Nodeltem Class Reference	18
5.5.1 Detailed Description	19
5.5.2 Constructor & Destructor Documentation	19
5.5.2.1 Nodeltem()	19
5.5.3 Member Function Documentation	20
5.5.3.1 addConnection()	20
5.5.3.2 addName()	20
5.5.3.3 advance()	20
5.5.3.4 changeSpeed()	20
5.5.3.5 contextMenuEvent()	21
5.5.3.6 getNameItem()	21
5.5.3.7 itemChange()	21
5.5.3.8 moveTextItem()	22
5.5.3.9 operator<<()	22
5.5.3.10 position()	22
5.5.3.11 setColor()	22
5.5.3.12 updateConnections()	23
5.6 Packet Class Reference	23
5.7 PacketItem Class Reference	23
5.7.1 Detailed Description	24
5.7.2 Constructor & Destructor Documentation	24
5.7.2.1 PacketItem()	24
5.7.3 Member Function Documentation	24
5.7.3.1 advance()	25
5.7.3.2 changeSpeed()	25
5.7.3.3 setSpeed()	25
5.8 Router Class Reference	25
5.8.1 Member Function Documentation	26
5.8.1.1 getName()	26
5.8.1.2 print()	26
5.8.1.3 receive()	27

5.8.1.4 send() [1/2]	27
5.8.1.5 send() [2/2]	27
5.9 Server Class Reference	27
5.9.1 Member Function Documentation	28
5.9.1.1 getName()	28
5.9.1.2 print()	28
5.9.1.3 receive()	28
5.9.1.4 send() [1/2]	28
5.9.1.5 send() [2/2]	28
5.10 WaitThread Class Reference	29
6 File Documentation	31
6.1 mainwindow.h	31
6.2 nodeitem.h	32
6.3 nodes.h	35
6.4 packet.h	37
6.5 packetitem.h	38
Index	41

Chapter 1

readme

#TODOS The waiting until calling receive in send() and in packet.h needs to either be done in a separate thread or with a signal through QTimer or something so that the gui and animations don't get blocked.

To create a handler that calls send() when and where appropriate.

1.1 Requirements with Qt Creator

There shouldn't be any other requirements if you use Qt Creator.

On macOS you could use homebrew to install Qt Creator. Run brew install --cask qt-creator

On linux just use your package manager of choice.

1.2 Requirements without Qt Creator

1.2.1 A compiler

On windows you should probably just install MinGW. Using other compilers through WSL 2 is probably possible as

On macOS, you could use clang. Install Xcode and its command line tools.

1.2.2 Qt

This application was built with Qt version 6.4.0

On windows go to https://www.qt.io/download and find an installer if you just want a binary. You probably want the open source version. Alternatively, you can compile from source. Instructions here: httpsecinderic://doc.qt.io/qt-6/build-sources.html

On macOS, you could use homebrew to install qt. Run brew install qt

On linux just use your package manager of choice.

2 readme

1.2.3 cmake

On windows, follow instructions from https://cmake.org/install/

On macOS, you could use homebrew to install cmake. Run brew install cmake

On linux just use your package manager of choice.

1.2.4 make

1.3 Building the gui without Qt Creator

Run cmake on this directory and make. If you don't mind where build files and the application binary are placed run $\column{ \column{ \c$

If you want build files and the application binary somewhere else, navigate to your desired directory and run $\c make $PATH_TO_GUI$, where $$PATH_TO_GUI$ is the path to this directory. $\c make$

Now you should have a runnable binary

1.4 Building with Qt Creator

Open the directory in Qt Creator, build and run.

1.5 Source content

1.5.1 MainWindow

MainWindow contains the source code for the main window of the program. mainwindow.ui holds the layout of the window.

1.5.2 Nodeltem

Is a visual representation of nodes and their connections.

1.5.3 PacketItem

Describes the animation item that moves when packets are sent between nodes.

1.5.4 Resources.qrc

Is a container that holds and links resources external to the source code.

1.5.5 main.cpp

Entry point of the program.

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

ode	
Client	9
Router	25
Server	27
acket	23
GraphicsPolygonItem	
Nodeltem	
PacketItem	23
MainWindow	
MainWindow	10
Object	
NodeCaller	15
Thread	
WaitThread	29

4 Hierarchical Index

Chapter 3

Class Index

3.1 Class List

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

Client	9
MainWindow	
The MainWindow class The main window of the program	10
Node	14
NodeCaller	
The NodeCaller class An object for interacting between nodeitems and mainwindow	15
Nodeltem	
The Nodeltem class A grapchical item which represents a node	18
Packet	23
PacketItem	
The PacketItem class Is the graphical item which moves in the animation of packages	23
Router	25
Server	27
WaitThread	29

6 Class Index

Chapter 4

File Index

4.1 File List

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

mainwindow.h		 														 							
nodeitem.h	 	 														 							
nodes.h	 	 														 							
packet.h	 	 														 							
packetitem.h		 														 							

8 File Index

Chapter 5

Class Documentation

5.1 Client Class Reference

Inheritance diagram for Client:



Public Member Functions

- Client (unsigned short id, unsigned int max_capacity, unsigned short connection_id, std::pair< unsigned int, unsigned short > connection_cost, std::string name, Nodeltem *nodeitem)
- void **create_packet** (unsigned short sender_id, unsigned short receiver_id, std::string content, unsigned int flag)
- void send ()
- void send (Packet *packet)
- void receive (Packet *packet)
- void print ()
- std::string getName () const

Additional Inherited Members

5.1.1 Member Function Documentation

5.1.1.1 getName()

```
std::string Client::getName ( ) const [inline], [virtual]
```

Implements Node.

5.1.1.2 print()

```
void Client::print ( ) [virtual]
```

Implements Node.

5.1.1.3 receive()

Implements Node.

5.1.1.4 send() [1/2]

```
void Client::send ( ) [virtual]
```

Implements Node.

5.1.1.5 send() [2/2]

Implements Node.

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

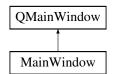
- · nodes.h
- · client.cpp

5.2 MainWindow Class Reference

The MainWindow class The main window of the program.

```
#include <mainwindow.h>
```

Inheritance diagram for MainWindow:



Public Member Functions

 MainWindow (std::map< unsigned short, Node * > *map1, std::map< std::string, unsigned short > *map2, QWidget *parent=nullptr)

MainWindow Consructor for the MainWindow class.

void addServer (unsigned short id, std::string name, float x, float y)

addServer Creates a graphical server nodeitem and adds it to the scene.

• void addRouter (unsigned short id, std::string name, float x, float y)

addRouter Creates a graphical router noditem and adds it to the scene.

void addClient (unsigned short id, std::string name, float x, float y)

addClient Creates a graphical client nodeitem and adds it to the scene.

void addLine (Nodeltem *node1, Nodeltem *node2)

addLine Adds a graphical line between two nodeitems.

void sendPacket (Nodeltem *node1, Nodeltem *node2)

sendPacket Animates a moving packet between two nodeitems

void dropPacket (NodeItem *node)

dropPacket Animates a dropped packet by changing the color of the node to black.

Nodeltem * getNode (unsigned short id)

getNode Getter for the graphical nodeitem.

Nodeltem * getNodeByName (std::string name)

getNodeByName Getter for the graphical nodeitem.

5.2.1 Detailed Description

The MainWindow class The main window of the program.

5.2.2 Constructor & Destructor Documentation

5.2.2.1 MainWindow()

```
MainWindow::MainWindow (
    std::map< unsigned short, Node * > * map1,
    std::map< std::string, unsigned short > * map2,
    QWidget * parent = nullptr )
```

MainWindow Consructor for the MainWindow class.

Parameters

map1	A map with id keys and Node values
map2	A DNS map
parent	An optional pointer to a parent QObject

5.2.3 Member Function Documentation

5.2.3.1 addClient()

```
void MainWindow::addClient (
          unsigned short id,
          std::string name,
          float x,
          float y )
```

addClient Creates a graphical client nodeitem and adds it to the scene.

Parameters

id	The id of the client node the created nodeitem corresponds to.
name	The name of the client.
Х	The x coordinate of the created nodeitem.
У	The y coordinate of the created nodeitem.

5.2.3.2 addLine()

addLine Adds a graphical line between two nodeitems.

Parameters

node1	The first nodeitem to be connected.
node2	The second nodeitem to be connected.

5.2.3.3 addRouter()

```
void MainWindow::addRouter (
          unsigned short id,
          std::string name,
          float x,
          float y )
```

addRouter Creates a graphical router noditem and adds it to the scene.

Parameters

id	The id of the router node the created nodeitem corresponds to.
name	The name of the router
Х	The x coordinate of the created nodeitem.
У	The y coordinate of the created nodeitem.

5.2.3.4 addServer()

```
void MainWindow::addServer (
         unsigned short id,
         std::string name,
         float x,
         float y )
```

addServer Creates a graphical server nodeitem and adds it to the scene.

Parameters

id	The id of the server node the created nodeitem corresponds to.
name	The name of the server
Х	The x coordinate of the created nodeitem.
У	The y coordinate of the created nodeitem.

5.2.3.5 dropPacket()

dropPacket Animates a dropped packet by changing the color of the node to black.

Parameters

node	The node which drops a packet.
------	--------------------------------

5.2.3.6 getNode()

```
\begin{tabular}{ll} {\tt NodeItem} & {\tt MainWindow::getNode} & ( \\ & {\tt unsigned} & {\tt short} & id \end{tabular} )
```

getNode Getter for the graphical nodeitem.

Parameters

```
id The id of the nodeitem.
```

Returns

A pointer to the nodeitem.

5.2.3.7 getNodeByName()

getNodeByName Getter for the graphical nodeitem.

Parameters

1	name	The name of the node	
---	------	----------------------	--

Returns

A pointer to the nodeitem.

5.2.3.8 sendPacket()

sendPacket Animates a moving packet between two nodeitems

Parameters

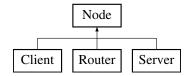
node1	The node the packet leaves from.
node2	The node the packet approaches.

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

- mainwindow.h
- · mainwindow.cpp

5.3 Node Class Reference

Inheritance diagram for Node:



Public Member Functions

- **Node** (unsigned short id, unsigned int max_capacity, unsigned short connection_id, std::pair< unsigned int, unsigned short > connection_cost, Nodeltem *nodeitem)
- virtual void send ()=0
- virtual void send (Packet *packet)=0
- virtual void receive (Packet *packet)=0
- virtual void **print** ()=0
- bool not_full_after_add (unsigned int added_size)
- void change max capacity (unsigned int new capacity)
- void change_current_capacity (unsigned int change)
- · unsigned short getId () const
- · unsigned int getCurrentCapacity () const
- · unsigned int getMaxCapacity () const
- size_t queue_packet_count () const
- unsigned short **getConnectionId** () const
- virtual std::string getName () const =0
- Nodeltem * getNodeltem ()

Protected Attributes

- unsigned short id_
- std::queue < Packet * > packets_
- · unsigned int max_capacity_
- · unsigned int current_capacity_
- unsigned short connection_id_
- std::pair< unsigned int, unsigned short > connection_cost_
- Nodeltem * nodeitem_

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

· nodes.h

5.4 NodeCaller Class Reference

The NodeCaller class An object for interacting between nodeitems and mainwindow.

#include <nodeitem.h>

Inheritance diagram for NodeCaller:



Public Slots

void change_max_capacity (unsigned short id)

change_max_capacity Signals mainwindow to change nodeitems max capacity.

void call_print (unsigned short id)

call_print Signals mainwindow to print out nodeitem's information.

void call_print_routing_table (unsigned short id)

call_print_routing_table Signals mainwindow to call print_routing_table()

void send_packet (unsigned short id)

send_packet Signals the mainwindow to call the send() method of class Node.

Signals

· void print_called ()

print_called A request to call the print() method of the Node class was made.

void change_max_capacity_called ()

change_max_capacity_called A request to call the change_max_capacity() method of the Node class was made.

void print_routing_table_called ()

print_routing_table_called A request was made to call the print_routing_table() method of the Router class.

void send_packet_called ()

send_packet_called A request was made to call the send() method of the Node class.

Public Member Functions

• unsigned short getId ()

getId A getter for the id of the nodeitem signalling to mainwindow.

5.4.1 Detailed Description

The NodeCaller class An object for interacting between nodeitems and mainwindow.

5.4.2 Member Function Documentation

5.4.2.1 call_print

```
void NodeCaller::call_print (
          unsigned short id ) [inline], [slot]
```

call_print Signals mainwindow to print out nodeitem's information.

Parameters

id The id of the calling nodeitem.

5.4.2.2 call_print_routing_table

```
void NodeCaller::call_print_routing_table (
          unsigned short id ) [inline], [slot]
```

call_print_routing_table Signals mainwindow to call print_routing_table()

Parameters

id The id of the calling router.

5.4.2.3 change_max_capacity

```
void NodeCaller::change_max_capacity (
          unsigned short id ) [inline], [slot]
```

change_max_capacity Signals mainwindow to change nodeitems max capacity.

Parameters

id The id of the calling nodeitem.

5.4.2.4 getId()

```
unsigned short NodeCaller::getId ( ) [inline]
```

getId A getter for the id of the nodeitem signalling to mainwindow.

Returns

id of the nodeitem.

5.4.2.5 send_packet

```
void NodeCaller::send_packet (  \mbox{unsigned short } id \mbox{ (inline], [slot]}
```

send_packet Signals the mainwindow to call the send() method of class Node.

Parameters

id The id of the calling node.

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

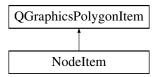
· nodeitem.h

5.5 Nodeltem Class Reference

The Nodeltem class A grapchical item which represents a node.

```
#include <nodeitem.h>
```

Inheritance diagram for Nodeltem:



Public Member Functions

NodeItem (unsigned short id, std::string name, double x, double y, NodeCaller *caller, QGraphicsItem *parent=nullptr)

Nodeltem Constructor of nodeitem.

QVariant itemChange (GraphicsItemChange change, const QVariant &value)

itemChange Describes how this nodeitem should react to a change in position.

- void addConnection (std::pair < Nodeltem *, Nodeltem * > connection, QGraphicsLineItem *line)
 addConnection Adds a new connection to static collections.
- void updateConnections ()

updateConnections Updates the graphical lines between each connected nodeitem.

• QPointF position ()

position Returns the centre of this nodeitem.

void addName (std::string name)

addName Adds a graphical textitem to this nodeitem.

• void moveTextItem (qreal x, qreal y)

moveTextItem Moves this nodeitems textitem by (x,y)

QGraphicsSimpleTextItem * getNameItem ()

 ${\it getName Item\ A\ getter\ for\ the\ graphical\ textitem\ underneath\ each\ node item.}$

void resetTicks ()

resetTicks Reset ticks_. Used for calculating how long the animation ought to be.

void setColor (QBrush brush)

setColor Changes the color of this Nodeltem.

void dropPacket ()

dropPacket Setter for isDroppingPacket_. Makes this Nodeltem change its color to black.

void changeSpeed (double multiplier)

changeSpeed Multiplies the animationSpeed_

• void setSpeed ()

setSpeed Setter for animationSpeed_

void operator<< (Nodeltem *other)

operator << Animates a packetitem between this and other nodeitems.

void contextMenuEvent (QGraphicsSceneContextMenuEvent *event)

contextMenuEvent Opens a contextmenu and emits a signal to MainWindow if an action was chosen.

- · std::string getName () const
- · unsigned short getID () const

Protected Member Functions

• void advance (int phase)

advance Animates the nodeitem. Describes what the nodeitem ought to do in each tick of the timer.

5.5.1 Detailed Description

The Nodeltem class A grapchical item which represents a node.

5.5.2 Constructor & Destructor Documentation

5.5.2.1 Nodeltem()

```
NodeItem::NodeItem (
    unsigned short id,
    std::string name,
    double x,
    double y,
    NodeCaller * caller,
    QGraphicsItem * parent = nullptr ) [inline]
```

Nodeltem Constructor of nodeitem.

Parameters

id	The id of the nodeitem.	
name The name of the nodeitem.		
X	The x coordinate of the top left corner of the nodeitem.	
У	y The y coordinate of the top left corner of the nodeitem.	
caller	A QObject for signalling between nodeitems and mainwindow.	
parent	An optional pointer to a parent QObject.	

5.5.3 Member Function Documentation

5.5.3.1 addConnection()

```
void NodeItem::addConnection (
    std::pair< NodeItem *, NodeItem * > connection,
    QGraphicsLineItem * line ) [inline]
```

addConnection Adds a new connection to static collections.

Parameters

connection	The connection to be added.
line	A pointer to a lineitem.

5.5.3.2 addName()

addName Adds a graphical textitem to this nodeitem.

Parameters

5.5.3.3 advance()

advance Animates the nodeitem. Describes what the nodeitem ought to do in each tick of the timer.

Parameters

phase Describes whether the function was called before or after the change occu	rred.
---	-------

5.5.3.4 changeSpeed()

```
void NodeItem::changeSpeed (
```

```
double multiplier ) [inline]
```

changeSpeed Multiplies the animationSpeed_

Parameters

multiplier

5.5.3.5 contextMenuEvent()

contextMenuEvent Opens a contextmenu and emits a signal to MainWindow if an action was chosen.

Parameters

5.5.3.6 getNameItem()

```
QGraphicsSimpleTextItem * NodeItem::getNameItem ( ) [inline]
```

getNameItem A getter for the graphical textitem underneath each nodeitem.

Returns

A pointer to the textitem.

5.5.3.7 itemChange()

itemChange Describes how this nodeitem should react to a change in position.

Parameters

change	The type of change which occurred.
value	The value (magnitude) of the change.

Returns

Changed item.

5.5.3.8 moveTextItem()

```
void NodeItem::moveTextItem ( \label{eq:qreal} \mbox{qreal } x, \mbox{qreal } y \; ) \quad [\mbox{inline}]
```

moveTextItem Moves this nodeitems textitem by (x,y)

Parameters

X	The change in x.
У	The change in y.

5.5.3.9 operator << ()

operator << Animates a packetitem between this and other nodeitems.

Parameters

other	The other nodeitem.
-------	---------------------

5.5.3.10 position()

```
QPointF NodeItem::position ( ) [inline] position Returns the centre of this nodeitem.
```

Returns

Returns the centre coordinate of this nodeitem.

5.5.3.11 setColor()

setColor Changes the color of this Nodeltem.

5.6 Packet Class Reference 23

Parameters

brush The wanted color.

5.5.3.12 updateConnections()

```
void NodeItem::updateConnections ( ) [inline]
```

updateConnections Updates the graphical lines between each connected nodeitem.

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

· nodeitem.h

5.6 Packet Class Reference

Public Member Functions

- · Packet (unsigned int size, unsigned short sender_id, unsigned short receiver_id, std::string content)
- unsigned short getSenderId () const
- unsigned short getReceiverId () const
- bool isDebug () const
- · bool isCheat () const
- · unsigned int getSize () const
- time_t getTimeSent () const
- void setSenderId (unsigned short new_id)
- void setReceiverId (unsigned short new id)
- void setDebug ()
- void setCheat ()
- std::string & getContent ()
- void setSize (unsigned int new_size)
- std::string getContent_print () const
- void wait (unsigned int bandwidth, unsigned short latency)
- void print ()

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

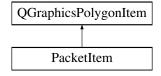
· packet.h

5.7 PacketItem Class Reference

The PacketItem class Is the graphical item which moves in the animation of packages.

#include <packetitem.h>

Inheritance diagram for PacketItem:



Public Member Functions

• PacketItem (QPointF start, QPointF end, QGraphicsScene *scene)

PacketItem Constructor of the animation item.

· void resetTicks ()

resetTicks Resets the animation counter to 0.

void setSpeed (greal magnitude)

setSpeed Sets the speed of the packetitem. Depends on the distance that needs to be covered.

void changeSpeed (double multiplier)

changeSpeed Changes the animation speed of the packetitem.

Protected Member Functions

• void advance (int phase)

advance Describes how the packetitem should act in each tick of the animation.

5.7.1 Detailed Description

The PacketItem class Is the graphical item which moves in the animation of packages.

5.7.2 Constructor & Destructor Documentation

5.7.2.1 PacketItem()

PacketItem Constructor of the animation item.

Parameters

	start	The start position of the packetitem.
		·
	end	The end position of the packetitem.
	scene	The scene in which the packetitem appears in.

5.7.3 Member Function Documentation

5.8 Router Class Reference 25

5.7.3.1 advance()

```
void PacketItem::advance (
          int phase ) [inline], [protected]
```

advance Describes how the packetitem should act in each tick of the animation.

Parameters

phase	Describes whether the change has happened or not.
-------	---

5.7.3.2 changeSpeed()

changeSpeed Changes the animation speed of the packetitem.

Parameters

multiplier

5.7.3.3 setSpeed()

setSpeed Sets the speed of the packetitem. Depends on the distance that needs to be covered.

Parameters

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

· packetitem.h

5.8 Router Class Reference

Inheritance diagram for Router:



Public Member Functions

- **Router** (unsigned short id, unsigned int max_capacity, unsigned short connection_id, std::pair< unsigned int, unsigned short > connection_cost, Nodeltem *nodeitem)
- void create_routing_table (const std::map< unsigned short, Router * > &routers)
- std::map< unsigned short, std::pair< unsigned int, unsigned short > > & getNeighbors ()
- · void print routing table () const
- void print_connections () const
- void addNeighbor (unsigned short id, const std::pair< unsigned int, unsigned short > &pair)
- void send ()
- void send (Packet *packet)
- void receive (Packet *packet)
- void send_to_end_node (Packet *packet)
- void print ()
- std::string getName () const

Additional Inherited Members

5.8.1 Member Function Documentation

5.8.1.1 getName()

```
std::string Router::getName ( ) const [inline], [virtual]
```

Implements Node.

5.8.1.2 print()

```
void Router::print ( ) [virtual]
```

Implements Node.

5.9 Server Class Reference 27

5.8.1.3 receive()

Implements Node.

5.8.1.4 send() [1/2]

```
void Router::send ( ) [virtual]
```

Implements Node.

5.8.1.5 send() [2/2]

Implements Node.

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

- · nodes.h
- · router.cpp

5.9 Server Class Reference

Inheritance diagram for Server:



Public Member Functions

- Server (unsigned short id, unsigned int max_capacity, unsigned short connection_id, std::pair< unsigned int, unsigned short > connection_cost, std::string name, std::string content_type, unsigned int content_size, Nodeltem *nodeitem)
- void send ()
- void send (Packet *packet)
- void receive (Packet *packet)
- void print ()
- void add (Packet *packet)
- std::string getContentType () const
- unsigned int getContentSize () const
- std::string getName () const

Additional Inherited Members

5.9.1 Member Function Documentation

```
5.9.1.1 getName()
std::string Server::getName ( ) const [inline], [virtual]
Implements Node.
5.9.1.2 print()
void Server::print ( ) [virtual]
Implements Node.
5.9.1.3 receive()
void Server::receive (
             Packet * packet ) [virtual]
Implements Node.
5.9.1.4 send() [1/2]
void Server::send ( ) [virtual]
Implements Node.
5.9.1.5 send() [2/2]
void Server::send (
            Packet * packet ) [virtual]
```

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

· nodes.h

Implements Node.

server.cpp

5.10 WaitThread Class Reference

Inheritance diagram for WaitThread:



Public Member Functions

• WaitThread (Node *node, Packet *packet, unsigned int bandwidth, unsigned short latency, QObject *parent=nullptr)

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

· nodes.h

Chapter 6

File Documentation

6.1 mainwindow.h

```
1 #ifndef MAINWINDOW_H
2 #define MAINWINDOW_H
4 #include <QMainWindow>
5 #include <QPainter>
6 #include <QGraphicsView>
7 #include <QTimer>
10 #include <map>
11
12 #include "nodeitem.h"
13 #include "nodes.h"
15 QT_BEGIN_NAMESPACE
16 namespace Ui { class MainWindow; }
17 QT_END_NAMESPACE
18
19
24 class MainWindow : public QMainWindow
25 {
26
       Q_OBJECT // necessary for slots and signals
28 public:
       MainWindow(std::map<unsigned short, Node*> *map1, std::map<std::string, unsigned short> *map2,
37
       QWidget *parent = nullptr);
38
       ~MainWindow();
39
49
       void addServer(unsigned short id, std::string name, float x, float y);
50
60
       void addRouter(unsigned short id, std::string name, float x, float y);
       void addClient(unsigned short id, std::string name, float x, float y);
80
       void addLine(NodeItem *node1, NodeItem *node2);
81
89
       void sendPacket(NodeItem *node1, NodeItem *node2);
90
       void dropPacket(NodeItem *node);
98
106
       NodeItem* getNode(unsigned short id);
107
        NodeItem* getNodeByName(std::string name);
115
116
117
118 private slots:
119
125
        void on_actionLoad_triggered();
126
127
        void on_actionClient_triggered();
128
129
        void on_actionRouter_triggered();
130
        void on_actionServer_triggered();
131
        void on_actionAdd_a_connection_triggered();
132
137
        void on actionCreate a packet triggered();
138
        void on_actionSend_packets_triggered();
```

```
144
149
        void sendPackages();
150
155
        void send_package();
156
161
        void change_node_max_capacity();
162
167
        void print_node();
168
173
        void print_routing_table();
174
179
        void on_actionChange_animation_speed_triggered();
180
181 private:
182
        Ui::MainWindow *ui;
183
        QGraphicsScene *scene;
184
189
        std::map<unsigned short, NodeItem *> nodeitems;
190
195
        QTimer *timer;
196
201
        NodeCaller *caller_;
202
2.07
        std::map<unsigned short, Client*> client_nodes;
212
        std::map<unsigned short, Router*> router_nodes;
213
214
        std::map<unsigned short, Node*> *main_map; // all nodes in simulation <node_id, Node>
215
        std::map<std::string, unsigned short> *DNS;
216
217
227
        void addLink(Router* first, unsigned int bandwidth, unsigned short latency, Router* second);
228
234
        bool readLinks();
235
241
        std::pair<float, float> nextPos();
242
248
        bool readNodes();
254
        void create_routing_tables();
255
260
        void create_lineitems();
261 };
2.62
263
265 #endif // MAINWINDOW_H
```

6.2 nodeitem.h

```
1 #ifndef NODEITEM_H
2 #define NODEITEM_H
4 #include <QGraphicsSimpleTextItem>
5 #include <QGraphicsScene>
7 #include <QMenu>
8 #include <QGraphicsSceneContextMenuEvent>
8 #include <QInputDialog>
10 #include "packetitem.h"
16 class NodeCaller : public QObject {
        Q_OBJECT // necessary for signals and slots
17
18
19 public:
       unsigned short getId() { return id_; }
27 public slots:
       void change_max_capacity(unsigned short id)
3.3
34
35
            id_{-} = id;
36
            emit change_max_capacity_called();
37
38
44
        void call_print(unsigned short id)
45
            id_{-} = id;
46
            emit print_called();
48
49
55
        void call_print_routing_table(unsigned short id)
56
57
            id = id;
            emit print_routing_table_called();
```

6.2 nodeitem.h

```
59
       }
60
66
       void send_packet (unsigned short id)
67
           id = id;
68
69
           emit send_packet_called();
70
71
72 signals:
73
78
       void print_called();
79
84
      void change max capacity called();
85
90
       void print_routing_table_called();
91
96
       void send_packet_called();
97
98 private:
103
       unsigned short id_;
104 };
105
106
111 class NodeItem : public QGraphicsPolygonItem
112 {
113 public:
125
        {\tt NodeItem} ({\tt unsigned \ short \ id, \ std::string \ name, \ double \ x, \ double \ y, \ NodeCaller \ \star caller, \ QGraphicsItem} \\
       *parent = nullptr) : QGraphicsPolygonItem(parent),
126
            caller_(caller), name_(name), id_(id)
127
            position_ = QPointF(0,0);
offset_ = QPointF(x+12,y+12);
128
129
130
            setFlag(QGraphicsItem::ItemIsMovable);
131
            \verb|setFlag(QGraphicsItem::ItemSendsScenePositionChanges);|\\
132
            QBrush brush(Qt::white);
            color = brush;
133
134
            isDroppingPacket_ = false;
135
            ticks_ = 0;
136
        }
137
146
        QVariant itemChange(GraphicsItemChange change, const QVariant &value)
147
            if (change == ItemPositionChange && scene()) {
148
149
                QPointF newPos = value.toPointF();
                position_ = newPos;
150
151
                updateConnections();
152
153
            return QGraphicsItem::itemChange(change, value);
        }
154
155
163
        void addConnection(std::pair<NodeItem *, NodeItem *> connection, QGraphicsLineItem* line)
164
165
            lines_.push_back(line);
166
            connections_.push_back(connection);
167
168
169
175
        void updateConnections()
176
177
            int i = 0;
178
            for (auto connection : connections_)
179
180
                lines_[i]->setLine(QLineF(connection.first->position(),
                                           connection.second->position()));
181
182
                i++;
183
            }
184
185
192
        QPointF position() { return position_ + offset_; }
193
200
        void addName(std::string name)
201
202
            nameItem_ = new QGraphicsSimpleTextItem(QString::fromStdString(name), this);
            203
204
205
206
            nameItem_->setPos(newPos);
207
208
216
        void moveTextItem(qreal x, qreal y)
217
218
            nameItem_->moveBy(x,y);
219
220
227
        QGraphicsSimpleTextItem* getNameItem() { return nameItem_; }
228
233
        void resetTicks() { ticks = 0; }
```

```
234
241
        void setColor(QBrush brush)
242
243
            color = brush;
2.44
245
250
        void dropPacket()
251
252
             isDroppingPacket_ = true;
253
254
261
        void changeSpeed(double multiplier)
262
263
             if (multiplier > 0) animationSpeed_ = multiplier;
264
265
        void setSpeed()
270
271
272
            animationSpeed_ = 1;
273
274
281
        void operator«(NodeItem *other)
282
            new PacketItem(this->position(), other->position(), this->scene());
283
284
285
286
293
        void contextMenuEvent (QGraphicsSceneContextMenuEvent *event)
294
295
             OMenu *menu = new OMenu();
296
            menu->setAttribute(Ot::WA DeleteOnClose);
297
            menu->clear();
298
            menu->addAction("Send a packet");
299
            menu->addAction("Change maximum capacity");
            menu->addAction("Print information");
300
301
            if (this->name_ == std::to_string(this->id_))
                 menu->addAction("Print routing table");
302
303
            QAction *a = menu->exec(event->screenPos());
304
             if (a == nullptr) return;
305
             else if (a->text().toStdString() == "Send a packet") {
306
                 caller_->send_packet(id_);
307
            else if (a->text().toStdString() == "Change maximum capacity") {
308
309
                 caller_->change_max_capacity(id_);
310
311
             else if (a->text().toStdString() == "Print information") {
312
                 caller_->call_print(id_);
313
             else if (a->text().toStdString() == "Print routing table") {
314
315
                caller_->call_print_routing_table(id_);
316
317
318
        std::string getName() const { return name_; }
unsigned short getID() const { return id_; }
319
320
321
322 protected:
329
        void advance(int phase)
330
331
            if (!phase) return;
            if (this->isDroppingPacket_) this->setBrush(Qt::black);
332
333
            else this->setBrush(color);
334
335
            ticks_ += animationSpeed_;
336
             if (ticks_ > 30)
337
                 isDroppingPacket_ = false;
338
                 ticks_ = 0;
            }
339
340
        }
341
342
343 private:
348
        inline static std::vector<QGraphicsLineItem *> lines_;
349
354
        inline static std::vector<std::pair<NodeItem *, NodeItem *> connections ;
355
360
        inline static double animationSpeed_;
361
366
        bool isDroppingPacket_;
367
372
        double ticks ;
373
374
379
        unsigned short id_;
380
385
        std::string name_;
386
```

6.3 nodes.h 35

```
391
        QBrush color;
392
397
        QPointF offset_;
398
403
        OPointF position ;
404
410
        QGraphicsSimpleTextItem* nameItem_;
411
416
        NodeCaller *caller_;
417 };
418
419 #endif // NODEITEM_H
```

6.3 nodes.h

```
1 #ifndef NODES_HPP
2 #define NODES_HPP
4 #include "packet.h"
5 #include "nodeitem.h"
7 #include <string>
8 #include <map>
9 #include <queue>
10 #include <iostream>
11
13 #include <QThread>
14
1.5
16
17
18 extern std::map<std::string, unsigned short> DNS;
19
20 class Node {
2.1
22 protected:
23
       unsigned short id_;
       std::queue<Packet*> packets_;
unsigned int max_capacity_; // KILObytes
24
25
       unsigned int current_capacity_; // KILObytes
unsigned short connection_id_; // for routers end node id, end nodes router id
26
27
28
        std::pair<unsigned int, unsigned short> connection_cost_;
29
       NodeItem *nodeitem_;
30
32
33 public:
34
        Node (unsigned short id, unsigned int max_capacity, unsigned short connection_id, std::pair<unsigned
        int, unsigned short> connection_cost, NodeItem *nodeitem)
        : id_(id), max_capacity_(max_capacity), connection_id_(connection_id), current_capacity_(0), connection_cost_(connection_cost), nodeitem_(nodeitem) {}
35
36
37
        virtual void send() = 0; // pop from queue, call receiver receive()
       virtual void send(Packet* packet) = 0; // cheat version
virtual void receive(Packet* packet) = 0; // different for different nodes
38
39
40
        virtual void print() = 0;
41
42
43
        virtual ~Node() = default;
44
45
        // getters, setters etc.
46
        // COMPARING HAPPENS IN KILOBYTES
48
        bool not_full_after_add(unsigned int added_size) { // arg = KB
49
           return current_capacity_ + added_size <= max_capacity_;</pre>
50
51
52
53
        void change_max_capacity(unsigned int new_capacity) { // kilobytes
54
            max_capacity_ = new_capacity;
55
56
57
        void change_current_capacity(unsigned int change) {
58
            current_capacity_ += change;
59
60
        unsigned short getId() const { return id_; }
62
        unsigned int getCurrentCapacity() const { return current_capacity_; }
63
        unsigned int getMaxCapacity() const { return max_capacity_; }
        size_t queue_packet_count() const { return packets_.size(); }
64
        unsigned short getConnectionId() const { return connection_id_; }
65
```

```
virtual std::string getName() const = 0;
       NodeItem* getNodeItem() { return nodeitem_; }
68
69
70 };
71
72 class WaitThread : public QThread
73
       Q_OBJECT
74
75
       void run() override
76
           unsigned int transmission_delay = size_ \star 8 / bandwidth_; // KB -> Kb
77
78
79
            if (transmission_delay < 1) {</pre>
                double t_d = size_ * 8 / bandwidth_; // same as above, seconds t_d *= 1000000; // seconds to microseconds
80
81
82
                transmission_delay = t_d; // to unsigned int
83
                QThread::usleep(transmission_delay); // sleep microseconds
           } else {
84
85
                QThread::sleep(transmission_delay); // sleep seconds
            }
87
88
           QThread::usleep(1000 * latency_); // sleep microseconds
89
           node_->receive(packet_);
90
            return;
91
92
93 public:
94
       WaitThread(Node* node, Packet* packet, unsigned int bandwidth, unsigned short latency, QObject*
       parent = nullptr) : QThread(parent), node_(node), packet_(packet), bandwidth_(bandwidth),
       latency_(latency) { size_ = packet->getSize(); }
95
96
97 private:
98
       unsigned int size_;
99
       unsigned int bandwidth_;
100
        unsigned short latency_;
101
        Node *node ;
        Packet *packet_;
102
103 };
104
105
106 extern std::map<unsigned short, Node*> main_map; // all nodes in simulation <node_id, Node>
107
108 class Router : public Node {
109
110 private:
111
        std::map<unsigned short, unsigned short> routing_table_; // <receiver_id, where_to_send_id>
112
        std::map<unsigned short, std::pair<unsigned int, unsigned short» neighbors_; // <neighbor_id,
       <bandwidth, latency»KILObits and ms</pre>
113
114
115 public:
116
        Router(unsigned short id, unsigned int max_capacity, unsigned short connection_id,
       std::pair<unsigned int, unsigned short> connection_cost, NodeItem *nodeitem)
117
            Node(id, max_capacity, connection_id, connection_cost, nodeitem) {}
118
119
        void create_routing_table(const std::map<unsigned short, Router*>& routers);
120
121
        std::map<unsigned short, std::pair<unsigned int, unsigned short»& getNeighbors() { return
       neighbors_; }
122
        void print_routing_table() const;
123
124
        void print_connections() const;
125
126
        void addNeighbor(unsigned short id, const std::pair<unsigned int, unsigned short>& pair) {
       neighbors_.insert({id, pair}); }
127
128
        void send(); // normal version
void send(Packet* packet); // cheat version
129
130
        void receive(Packet* packet);
131
132
        void send_to_end_node(Packet* packet);
133
        void print();
134
135
136
        std::string getName() const { return std::to_string(this->getId()); }
137
138
139 };
140
141 class Client : public Node {
142
143 private:
144
        std::string name_;
145
146 public:
147
        Client (unsigned short id, unsigned int max capacity, unsigned short connection id,
```

6.4 packet.h 37

```
std::pair<unsigned int, unsigned short> connection_cost, std::string name, NodeItem *nodeitem)
148
             : name_(name), Node(id, max_capacity, connection_id, connection_cost, nodeitem) {}
149
150
        void create_packet(unsigned short sender_id, unsigned short receiver_id, std::string content,
151
                                                          unsigned int flag);
152
153
        void send(); // normal version
154
         void send(Packet* packet); // cheat version
155
         void receive(Packet* packet);
156
        void print();
157
158
         std::string getName() const { return name_; }
159
160 };
161
162 class Server : public Node {
163
164 private:
165
        std::string name_;
        std::string content_type_; // .mp4 for Youtube, .jpg for Instagram etc.
unsigned int content_size_; // small for pictures, large for videos, KILObytes
166
167
168
169 public:
        Server(unsigned short id, unsigned int max_capacity, unsigned short connection_id,
std::pair<unsigned int, unsigned short> connection_cost, std::string name, std::string content_type,
170
171
       unsigned int content_size, NodeItem *nodeitem)
172
             name_(name), content_type_(content_type), content_size_(content_size), Node(id, max_capacity,
       connection_id, connection_cost, nodeitem) {}
173
174
        void send(); // normal version
        void send(Packet* packet); // cheat version
175
176
        void receive(Packet* packet);
177
        void print();
178
179
        void add(Packet* packet) { packets_.push(packet); }
180
181
182
        std::string getContentType() const { return content_type_; }
183
        unsigned int getContentSize() const { return content_size_; }
184
185
        std::string getName() const { return name_; }
186
187 };
188
189 #endif
```

6.4 packet.h

```
1 #ifndef PACKET H
2 #define PACKET H
4 #include <string>
5 #include <time.h>
6 #include <unistd.h>
7 #include <iostream>
10 class Packet {
11
        unsigned int size_; // !!!CHANGED!!! KILObytes
13
       unsigned short sender_id_;
14
       unsigned short receiver_id_;
15
16
        std::string content_;
        time_t time_sent_;
       bool debug_; // print useful data when moving bool cheat_; // this will bypass the queue
18
19
2.0
21
        public:
22
23
        Packet (unsigned int size, unsigned short sender_id, unsigned short receiver_id, std::string content)
2.4
        size_(size), sender_id_(sender_id), receiver_id_(receiver_id), content_(content), debug_(false),
        cheat_(false) {
25
            time_sent_ = time(NULL);
26
28
        ~Packet() = default;
29
30
        // getters, setters etc.
        unsigned short getSenderId() const { return sender_id_; }
31
        unsigned short getReceiverId() const { return receiver_id_; }
bool isDebug() const { return debug_; }
32
```

```
34
         bool isCheat() const { return cheat_; }
         unsigned int getSize() const { return size_;
35
36
         time_t getTimeSent() const { return time_sent_; }
37
         void setSenderId(unsigned short new_id) { sender_id_ = new_id; }
void setReceiverId(unsigned short new_id) { receiver_id_ = new_id; }
38
39
         void setDebug() { debug_ = true; }
void setCheat() { cheat_ = true; }
40
41
42
43
         std::string& getContent() { return content_; }
44
         void setSize(unsigned int new_size) { size_ = new_size; }
45
46
47
         std::string getContent_print() const { return content_; }
48
49
         void wait(unsigned int bandwidth, unsigned short latency) { // BW: Mb/s, latency: ms, size_ KB
50
               // wait transfer
              unsigned int transmission_delay = size_ * 8 / bandwidth; // KB -> Kb
51
52
               if (transmission_delay < 1) {</pre>
                    double t_d = size_ * 8 / bandwidth; // same as above, seconds t_d *= 1000000; // seconds to microseconds transmission_delay = t_d; // to unsigned int
55
56
                    {\tt usleep\,(transmission\_delay)\,;}\ //\ {\tt sleep\,\,microseconds}
57
58
               } else {
                    sleep(transmission_delay); // sleep seconds
60
61
62
               usleep(1000 * latency); // sleep microseconds
63
64
65
         void print() {
             std::cout « "Packet from " « sender_id_ « " to " « receiver_id_

« "\nContent: " « content_ « ", size: " « size_ « "KB"

« "\nDebug packet: " « debug_

« "\nTime elapsed since creation: " « time(NULL) - time_sent_ « std::endl;
68
69
70
71 }
73 };
74
7.5
76 #endif
```

6.5 packetitem.h

```
1 #ifndef PACKETITEM_H
2 #define PACKETITEM H
4 #include <OGraphicsScene>
5 #include <QGraphicsItem>
6 #include <QtMath>
12 class PacketItem : public QGraphicsPolygonItem
13 {
14
15 public:
24
        PacketItem(QPointF start, QPointF end, QGraphicsScene *scene) : QGraphicsPolygonItem()
25
26
            qreal x = 0, y = 0;
27
            OPolygonF Triangle;
28
            Triangle.append(QPointF(x,y));
30
            Triangle.append(QPointF(x-10,y+13));
31
            Triangle.append(QPointF(x+10,y+13));
32
            Triangle.append(QPointF(x,y));
33
            QBrush redBrush(Qt::yellow);
34
35
            QPen blackpen(Qt::black);
36
37
            this->setPolygon(Triangle);
38
            this->setPen(blackpen);
39
            this->setBrush (redBrush);
40
            this->setZValue(1);
41
42
            scene->addItem(this);
43
44
            QPointF vector = end - start;
            qreal dot = QPointF::dotProduct(vector, QPointF(0, vector.ry()));
qreal mvector = qSqrt(qPow(vector.rx(), 2) + qPow(vector.ry(), 2));
45
46
47
            qreal angle;
```

6.5 packetitem.h

```
if (vector.rx() >= 0)
50
                angle = qRadiansToDegrees(qAcos(-(dot/(mvector*vector.ry()))));
51
                angle = qRadiansToDegrees(qAcos((dot/(mvector*vector.ry())))) - 180;
52
5.3
54
           this->setRotation(angle);
           this->setSpeed(mvector);
55
56
           this->setPos(start + (vector/3));
57
       }
58
       void resetTicks() { ticks = 0; }
63
64
70
       void setSpeed(greal magnitude)
71
72
           speed = animationSpeed * magnitude / 75;
73
74
       void changeSpeed(double multiplier)
80
81
           if (multiplier > 0) animationSpeed = multiplier;
83
84
85 protected:
91
       void advance(int phase)
92
93
            if (!phase) return;
94
           this->setPos(mapToParent(0,-(speed)));
95
           ticks += animationSpeed;
           if (ticks > 30) {
    this->scene()->removeItem(this);
96
97
98
               delete this;
99
100
        }
101
102 private:
107 qrea
        qreal angle;
108
113
        qreal speed;
114
119
        double ticks;
120
125
        inline static double animationSpeed;
126 };
127
128 #endif // PACKETITEM_H
```

Index

addClient	Nodeltem, 21
MainWindow, 12	
addConnection	MainWindow, 10
Nodeltem, 20	addClient, 12
addLine	addLine, 12
MainWindow, 12	addRouter, 12
addName	addServer, 13
Nodeltem, 20	dropPacket, 13
addRouter	getNode, 13
MainWindow, 12	getNodeByName, 14
addServer	MainWindow, 11
MainWindow, 13	sendPacket, 14
advance	moveTextItem
Nodeltem, 20	Nodeltem, 22
PacketItem, 24	,
i acketitetti, 24	Node, 14
call_print	NodeCaller, 15
NodeCaller, 16	call_print, 16
call print routing table	call_print_routing_table, 17
NodeCaller, 17	change max capacity, 17
	getld, 17
change_max_capacity	send_packet, 17
NodeCaller, 17	Nodeltem, 18
changeSpeed	addConnection, 20
Nodeltem, 20	,
PacketItem, 25	addName, 20
Client, 9	advance, 20
getName, 9	changeSpeed, 20
print, 9	contextMenuEvent, 21
receive, 10	getNameItem, 21
send, 10	itemChange, 21
contextMenuEvent	moveTextItem, 22
Nodeltem, 21	Nodeltem, 19
	operator<<, 22
dropPacket	position, 22
MainWindow, 13	setColor, 22
	updateConnections, 23
getld	
NodeCaller, 17	operator<<
getName	Nodeltem, 22
Client, 9	
Router, 26	Packet, 23
Server, 28	PacketItem, 23
getNameItem	advance, 24
Nodeltem, 21	changeSpeed, 25
getNode	PacketItem, 24
MainWindow, 13	setSpeed, 25
getNodeByName	position
MainWindow, 14	Nodeltem, 22
	print
itemChange	Client 9

42 INDEX

```
Router, 26
    Server, 28
receive
    Client, 10
    Router, 26
    Server, 28
Router, 25
    getName, 26
    print, 26
    receive, 26
    send, 27
send
    Client, 10
    Router, 27
    Server, 28
send_packet
    NodeCaller, 17
sendPacket
    MainWindow, 14
Server, 27
    getName, 28
    print, 28
    receive, 28
    send, 28
setColor
    Nodeltem, 22
setSpeed
    PacketItem, 25
updateConnections
    Nodeltem, 23
WaitThread, 29
```