

Network simulator

Generated by Doxygen 1.9.5



<b>1 readme</b>	<b>1</b>
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 NodeItem	2
1.5.3 PacketItem	2
1.5.4 Resources.qrc	2
1.5.5 main.cpp	2
<b>2 Hierarchical Index</b>	<b>3</b>
2.1 Class Hierarchy	3
<b>3 Class Index</b>	<b>5</b>
3.1 Class List	5
<b>4 File Index</b>	<b>7</b>
4.1 File List	7
<b>5 Class Documentation</b>	<b>9</b>
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 getNameltem()	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
<b>6 File Documentation</b>	<b>31</b>
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
<b>Index</b>	<b>41</b>



# 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 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: <https://doc.qt.io/qt-6/build-sources.html>

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 well.

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: <https://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.

### 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 `\cmake . \make`

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

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:

Node . . . . .	14
Client . . . . .	9
Router . . . . .	25
Server . . . . .	27
Packet . . . . .	23
QGraphicsPolygonItem	
NodeItem . . . . .	18
PacketItem . . . . .	23
QMainWindow	
MainWindow . . . . .	10
QObject	
NodeCaller . . . . .	15
QThread	
WaitThread . . . . .	29



## Chapter 3

# Class Index

### 3.1 Class List

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

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



## Chapter 4

# File Index

### 4.1 File List

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

<a href="#">mainwindow.h</a>	.....	??
<a href="#">nodeitem.h</a>	.....	??
<a href="#">nodes.h</a>	.....	??
<a href="#">packet.h</a>	.....	??
<a href="#">packetitem.h</a>	.....	??

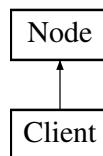


## Chapter 5

# 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, [NodeItem](#) \*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()

```
void Client::receive (
    Packet * packet ) [virtual]
```

Implements [Node](#).

#### 5.1.1.4 send() [1/2]

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

Implements [Node](#).

#### 5.1.1.5 send() [2/2]

```
void Client::send (
    Packet * packet ) [virtual]
```

Implements [Node](#).

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

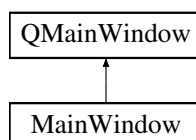
- nodes.h
- client.cpp

## 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 Constructor 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 nodeitem 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](#) ([NodeItem](#) \*node1, [NodeItem](#) \*node2)  
*addLine Adds a graphical line between two nodeitems.*
- void [sendPacket](#) ([NodeItem](#) \*node1, [NodeItem](#) \*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.*
- [NodeItem](#) \* [getNode](#) (unsigned short id)  
*getNode Getter for the graphical nodeitem.*
- [NodeItem](#) \* [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](#) Constructor for the [MainWindow](#) class.

#### Parameters

<i>map1</i>	A map with id keys and <a href="#">Node</a> values
<i>map2</i>	A DNS map
<i>parent</i>	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

<i>id</i>	The id of the client node the created nodeitem corresponds to.
<i>name</i>	The name of the client.
<i>x</i>	The x coordinate of the created nodeitem.
<i>y</i>	The y coordinate of the created nodeitem.

### 5.2.3.2 addLine()

```
void MainWindow::addLine (
    NodeItem * node1,
    NodeItem * node2 )
```

addLine Adds a graphical line between two nodeitems.

#### Parameters

<i>node1</i>	The first nodeitem to be connected.
<i>node2</i>	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 nodeitem and adds it to the scene.

## Parameters

<i>id</i>	The id of the router node the created nodeitem corresponds to.
<i>name</i>	The name of the router
<i>x</i>	The x coordinate of the created nodeitem.
<i>y</i>	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

<i>id</i>	The id of the server node the created nodeitem corresponds to.
<i>name</i>	The name of the server
<i>x</i>	The x coordinate of the created nodeitem.
<i>y</i>	The y coordinate of the created nodeitem.

**5.2.3.5 dropPacket()**

```
void MainWindow::dropPacket (
    NodeItem * node )
```

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

## Parameters

<i>node</i>	The node which drops a packet.
-------------	--------------------------------

**5.2.3.6 getNode()**

```
NodeItem * MainWindow::getNode (
    unsigned short id )
```

getNode Getter for the graphical nodeitem.

**Parameters**

<i>id</i>	The id of the nodeitem.
-----------	-------------------------

**Returns**

A pointer to the nodeitem.

**5.2.3.7 getNodeByName()**

```
NodeItem * MainWindow::getNodeByName (
    std::string name )
```

getNodeByName Getter for the graphical nodeitem.

**Parameters**

<i>name</i>	The name of the node.
-------------	-----------------------

**Returns**

A pointer to the nodeitem.

**5.2.3.8 sendPacket()**

```
void MainWindow::sendPacket (
    NodeItem * node1,
    NodeItem * node2 )
```

sendPacket Animates a moving packet between two nodeitems

**Parameters**

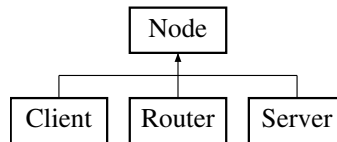
<i>node1</i>	The node the packet leaves from.
<i>node2</i>	The node the packet approaches.

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, [NodeItem](#) \*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
- [NodeItem](#) \* **getNodeItem** ()

## 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\_**
- [NodeItem](#) \* **nodeitem\_**

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

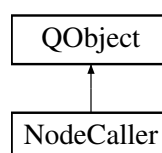
- nodes.h

## 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

<i>id</i>	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

<i>id</i>	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

<i>id</i>	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 (
    unsigned short id ) [inline], [slot]
```

send\_packet Signals the mainwindow to call the send() method of class [Node](#).

## Parameters

<i>id</i>	The id of the calling node.
-----------	-----------------------------

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

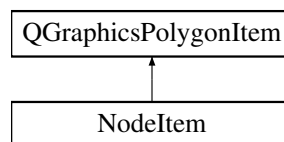
- nodeitem.h

## 5.5 NodeItem Class Reference

The [NodeItem](#) class A graphical item which represents a node.

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

Inheritance diagram for NodeItem:



### Public Member Functions

- [NodeItem](#) (unsigned short id, std::string name, double x, double y, [NodeCaller](#) \*caller, QGraphicsItem \*parent=nullptr)  
*NodeItem 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< [NodeItem](#) \*, [NodeItem](#) \* > 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 \* [getNameltem](#) ()  
*getNameltem A getter for the graphical textitem underneath each nodeitem.*
- 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 NodeItem.*
- void [dropPacket](#) ()  
*dropPacket Setter for isDroppingPacket\_. Makes this NodeItem change its color to black.*
- void [changeSpeed](#) (double multiplier)



- changeSpeed Multiplies the animationSpeed\_*
- void **setSpeed** ()  
*setSpeed Setter for animationSpeed\_*
- void **operator<<** ([NodeItem](#) \*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 [NodeItem](#) class A graphical item which represents a node.

### 5.5.2 Constructor & Destructor Documentation

#### 5.5.2.1 NodeItem()

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

[NodeItem](#) Constructor of nodeitem.

#### Parameters

<i>id</i>	The id of the nodeitem.
<i>name</i>	The name of the nodeitem.
<i>x</i>	The x coordinate of the top left corner of the nodeitem.
<i>y</i>	The y coordinate of the top left corner of the nodeitem.
<i>caller</i>	A QObject for signalling between nodeitems and mainwindow.
<i>parent</i>	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

<i>connection</i>	The connection to be added.
<i>line</i>	A pointer to a lineitem.

### 5.5.3.2 addName()

```
void NodeItem::addName (
    std::string name ) [inline]
```

addName Adds a graphical textitem to this nodeitem.

#### Parameters

<i>name</i>	The desired name.
-------------	-------------------

### 5.5.3.3 advance()

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

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

#### Parameters

<i>phase</i>	Describes whether the function was called before or after the change occurred.
--------------	--

### 5.5.3.4 changeSpeed()

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

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

changeSpeed Multiplies the animationSpeed\_

#### Parameters

<i>multiplier</i>	
-------------------	--

### 5.5.3.5 contextMenuEvent()

```
void NodeItem::contextMenuEvent (
    QGraphicsSceneContextMenuEvent * event ) [inline]
```

contextMenuEvent Opens a contextmenu and emits a signal to [MainWindow](#) if an action was chosen.

#### Parameters

<i>event</i>	Right click pressed on nodeitem.
--------------	----------------------------------

### 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()

```
QVariant NodeItem::itemChange (
    GraphicsItemChange change,
    const QVariant & value ) [inline]
```

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

#### Parameters

<i>change</i>	The type of change which occurred.
<i>value</i>	The value (magnitude) of the change.

**Returns**

Changed item.

**5.5.3.8 moveTextItem()**

```
void NodeItem::moveTextItem (
    qreal x,
    qreal y ) [inline]
```

moveTextItem Moves this nodeitems textitem by (x,y)

**Parameters**

<i>x</i>	The change in x.
<i>y</i>	The change in y.

**5.5.3.9 operator<<()**

```
void NodeItem::operator<< (
    NodeItem * other ) [inline]
```

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

**Parameters**

<i>other</i>	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()**

```
void NodeItem::setColor (
    QBrush brush ) [inline]
```

setColor Changes the color of this [NodeItem](#).

## Parameters

<i>brush</i>	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:

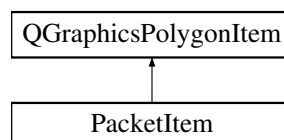
- packet.h

## 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** (qreal 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::PacketItem (
    QPointF start,
    QPointF end,
    QGraphicsScene * scene ) [inline]
```

[PacketItem](#) Constructor of the animation item.

#### Parameters

<i>start</i>	The start position of the packetitem.
<i>end</i>	The end position of the packetitem.
<i>scene</i>	The scene in which the packetitem appears in.

### 5.7.3 Member Function Documentation

### 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

<i>phase</i>	Describes whether the change has happened or not.
--------------	---

### 5.7.3.2 changeSpeed()

```
void PacketItem::changeSpeed (
    double multiplier ) [inline]
```

changeSpeed Changes the animation speed of the packetitem.

#### Parameters

<i>multiplier</i>	
-------------------	--

### 5.7.3.3 setSpeed()

```
void PacketItem::setSpeed (
    qreal magnitude ) [inline]
```

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

#### Parameters

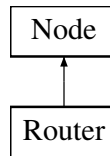
<i>magnitude</i>	The distance moved.
------------------	---------------------

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, [NodeItem](#) \*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](#).



### 5.8.1.3 receive()

```
void Router::receive (
    Packet * packet ) [virtual]
```

Implements [Node](#).

### 5.8.1.4 send() [1/2]

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

Implements [Node](#).

### 5.8.1.5 send() [2/2]

```
void Router::send (
    Packet * packet ) [virtual]
```

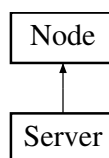
Implements [Node](#).

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, [NodeItem](#) \*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]
```

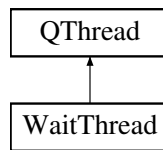
Implements [Node](#).

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

- nodes.h
- 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
3
4 #include <QMainWindow>
5 #include <QPainter>
6 #include <QGraphicsView>
7 #include <QTimer>
8
9
10 #include <map>
11
12 #include "nodeitem.h"
13 #include "nodes.h"
14
15 QT_BEGIN_NAMESPACE
16 namespace Ui { class MainWindow; }
17 QT_END_NAMESPACE
18
19
20 class MainWindow : public QMainWindow
21 {
22     Q_OBJECT // necessary for slots and signals
23
24 public:
25     MainWindow(std::map<unsigned short, Node*> *map1, std::map<std::string, unsigned short> *map2,
26               QWidget *parent = nullptr);
27     ~MainWindow();
28
29     void addServer(unsigned short id, std::string name, float x, float y);
30
31     void addRouter(unsigned short id, std::string name, float x, float y);
32
33     void addClient(unsigned short id, std::string name, float x, float y);
34
35     void addLine(NodeItem *node1, NodeItem *node2);
36
37     void sendPacket(NodeItem *node1, NodeItem *node2);
38
39     void dropPacket(NodeItem *node);
40
41     NodeItem* getNode(unsigned short id);
42
43     NodeItem* getNodeByName(std::string name);
44
45 private slots:
46     void on_actionLoad_triggered();
47
48     void on_actionClient_triggered();
49     void on_actionRouter_triggered();
50     void on_actionServer_triggered();
51     void on_actionAdd_a_connection_triggered();
52
53     void on_actionCreate_a_packet_triggered();
54
55     void on_actionSend_packets_triggered();
56 }
```

```

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
207     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();
249
254     void create_routing_tables();
255
260     void create_lineitems();
261 };
262
263
264
265 #endif // MAINWINDOW_H

```

## 6.2 nodeitem.h

```

1 #ifndef NODEITEM_H
2 #define NODEITEM_H
3
4 #include <QGraphicsSimpleTextItem>
5 #include <QGraphicsScene>
6 #include <QMenu>
7 #include <QGraphicsSceneContextMenuEvent>
8 #include <QInputDialog>
9
10 #include "packetitem.h"
11
12
16 class NodeCaller : public QObject {
17     Q_OBJECT // necessary for signals and slots
18
19 public:
25     unsigned short getId() { return id_; }
26
27 public slots:
33     void change_max_capacity(unsigned short id)
34     {
35         id_ = id;
36         emit change_max_capacity_called();
37     }
38
44     void call_print(unsigned short id)
45     {
46         id_ = id;
47         emit print_called();
48     }
49
55     void call_print_routing_table(unsigned short id)
56     {
57         id_ = id;
58         emit print_routing_table_called();

```

```

59     }
60
61     void send_packet(unsigned short id)
62     {
63         id_ = id;
64         emit send_packet_called();
65     }
66
67 signals:
68
69     void print_called();
70
71     void change_max_capacity_called();
72
73     void print_routing_table_called();
74
75     void send_packet_called();
76
77 private:
78     unsigned short id_;
79 };
80
81 class NodeItem : public QGraphicsPolygonItem
82 {
83 public:
84     NodeItem(unsigned short id, std::string name, double x, double y, NodeCaller *caller, QGraphicsItem
85 *parent = nullptr) : QGraphicsPolygonItem(parent),
86 caller_(caller), name_(name), id_(id)
87 {
88     position_ = QPointF(0,0);
89     offset_ = QPointF(x+12,y+12);
90     setFlag(QGraphicsItem::ItemIsMovable);
91     setFlag(QGraphicsItem::ItemSendsScenePositionChanges);
92     QBrush brush(Qt::white);
93     color = brush;
94     isDroppingPacket_ = false;
95     ticks_ = 0;
96 }
97
98 QVariant itemChange(GraphicsItemChange change, const QVariant &value)
99 {
100     if (change == ItemPositionChange && scene()) {
101         QPointF newPos = value.toPointF();
102         position_ = newPos;
103         updateConnections();
104     }
105     return QGraphicsItem::itemChange(change, value);
106 }
107
108 void addConnection(std::pair<NodeItem *, NodeItem *> connection, QGraphicsLineItem* line)
109 {
110     lines_.push_back(line);
111     connections_.push_back(connection);
112 }
113
114 void updateConnections()
115 {
116     int i = 0;
117     for (auto connection : connections_)
118     {
119         lines_[i]->setLine(QLineF(connection.first->position(),
120 connection.second->position()));
121         i++;
122     }
123 }
124
125 QPointF position() { return position_ + offset_; }
126
127 void addName(std::string name)
128 {
129     nameItem_ = new QGraphicsSimpleTextItem(QString::fromStdString(name), this);
130     QPointF center = this->boundingRect().center();
131     QPointF newPos(this->mapToScene(center).x() - (nameItem_->boundingRect().width() / 2),
132 this->mapToScene(center).y() - (nameItem_->boundingRect().height() / 2));
133     nameItem_->setPos(newPos);
134 }
135
136 void moveTextItem(qreal x, qreal y)
137 {
138     nameItem_->moveBy(x,y);
139 }
140
141 QGraphicsSimpleTextItem* getNameItem() { return nameItem_; }
142
143 void resetTicks() { ticks_ = 0; }

```

```

234
241 void setColor(QBrush brush)
242 {
243     color = brush;
244 }
245
250 void dropPacket()
251 {
252     isDroppingPacket_ = true;
253 }
254
261 void changeSpeed(double multiplier)
262 {
263     if (multiplier > 0) animationSpeed_ = multiplier;
264 }
265
270 void setSpeed()
271 {
272     animationSpeed_ = 1;
273 }
274
281 void operator«(NodeItem *other)
282 {
283     new PacketItem(this->position(), other->position(), this->scene());
284 }
285
286
293 void contextMenuEvent(QGraphicsSceneContextMenuEvent *event)
294 {
295     QMenu *menu = new QMenu();
296     menu->setAttribute(Qt::WA_DeleteOnClose);
297     menu->clear();
298     menu->addAction("Send a packet");
299     menu->addAction("Change maximum capacity");
300     menu->addAction("Print information");
301     if (this->name_ == std::to_string(this->id_))
302         menu->addAction("Print routing table");
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     }
308     else if (a->text().toStdString() == "Change maximum capacity") {
309         caller_>change_max_capacity(id_);
310     }
311     else if (a->text().toStdString() == "Print information") {
312         caller_>call_print(id_);
313     }
314     else if (a->text().toStdString() == "Print routing table") {
315         caller_>call_print_routing_table(id_);
316     }
317 }
318
319 std::string getName() const { return name_; }
320 unsigned short getID() const { return id_; }
321
322 protected:
323 void advance(int phase)
324 {
325     if (!phase) return;
326     if (this->isDroppingPacket_) this->setBrush(Qt::black);
327     else this->setBrush(color);
328
329     ticks_ += animationSpeed_;
330     if (ticks_ > 30) {
331         isDroppingPacket_ = false;
332         ticks_ = 0;
333     }
334 }
335
336
337 private:
338     inline static std::vector<QGraphicsLineItem *> lines_;
339
340     inline static std::vector<std::pair<NodeItem *, NodeItem *>> connections_;
341
342     inline static double animationSpeed_;
343
344     bool isDroppingPacket_;
345
346     double ticks_;
347
348     unsigned short id_;
349
350     std::string name_;
351
352

```



```

391     QBrush color;
392
397     QPointF offset_;
398
403     QPointF 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
3
4  #include "packet.h"
5  #include "nodeitem.h"
6
7  #include <string>
8  #include <map>
9  #include <queue>
10 #include <iostream>
11
12
13 #include <QThread>
14
15
16
17
18 extern std::map<std::string, unsigned short> DNS;
19
20 class Node {
21
22 protected:
23     unsigned short id_;
24     std::queue<Packet*> packets_;
25     unsigned int max_capacity_; // KILobytes
26     unsigned int current_capacity_; // KILobytes
27     unsigned short connection_id_; // for routers end node id, end nodes router id
28     std::pair<unsigned int, unsigned short> connection_cost_;
29     NodeItem *nodeitem_;
30
31
32
33 public:
34     Node(unsigned short id, unsigned int max_capacity, unsigned short connection_id, std::pair<unsigned
35         int, unsigned short> connection_cost, NodeItem *nodeitem)
36         : id_(id), max_capacity_(max_capacity), connection_id_(connection_id), current_capacity_(0),
37         connection_cost_(connection_cost), nodeitem_(nodeitem) {}
38
39     virtual void send() = 0; // pop from queue, call receiver receive()
40     virtual void send(Packet* packet) = 0; // cheat version
41     virtual void receive(Packet* packet) = 0; // different for different nodes
42
43     virtual void print() = 0;
44
45     virtual ~Node() = default;
46
47     // getters, setters etc.
48
49     // COMPARING HAPPENS IN KILOBYTES
50     bool not_full_after_add(unsigned int added_size) { // arg = KB
51         return current_capacity_ + added_size <= max_capacity_;
52     }
53
54     void change_max_capacity(unsigned int new_capacity) { // kilobytes
55         max_capacity_ = new_capacity;
56     }
57
58     void change_current_capacity(unsigned int change) {
59         current_capacity_ += change;
60     }
61
62     unsigned short getId() const { return id_; }
63     unsigned int getCurrentCapacity() const { return current_capacity_; }
64     unsigned int getMaxCapacity() const { return max_capacity_; }
65     size_t queue_packet_count() const { return packets_.size(); }
66     unsigned short getConnectionId() const { return connection_id_; }

```

```

67     virtual std::string getName() const = 0;
68     NodeItem* getNodeItem() { return nodeitem_; }
69
70 };
71
72 class WaitThread : public QThread
73 {
74     Q_OBJECT
75     void run() override
76     {
77         unsigned int transmission_delay = size_ * 8 / bandwidth_; // KB -> Kb
78
79         if (transmission_delay < 1) {
80             double t_d = size_ * 8 / bandwidth_; // same as above, seconds
81             t_d *= 1000000; // seconds to microseconds
82             transmission_delay = t_d; // to unsigned int
83             QThread::usleep(transmission_delay); // sleep microseconds
84         } else {
85             QThread::sleep(transmission_delay); // sleep seconds
86         }
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_;
102     Packet *packet_;
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>>KIBits and ms
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)
: Node(id, max_capacity, connection_id, connection_cost, nodeitem) {}
117
118     void create_routing_table(const std::map<unsigned short, Router*>& routers);
119
120     std::map<unsigned short, std::pair<unsigned int, unsigned short>> getNeighbors() { return
neighbors_; }
121
122     void print_routing_table() const;
123     void print_connections() const;
124
125     void addNeighbor(unsigned short id, const std::pair<unsigned int, unsigned short>& pair) {
neighbors_.insert({id, pair}); }
126
127     void send(); // normal version
128     void send(Packet* packet); // cheat version
129     void receive(Packet* packet);
130
131     void send_to_end_node(Packet* packet);
132
133     void print();
134
135     std::string getName() const { return std::to_string(this->getId()); }
136
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,

```

```

        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_;
166     std::string content_type_; // .mp4 for Youtube, .jpg for Instagram etc.
167     unsigned int content_size_; // small for pictures, large for videos, KILobytes
168
169 public:
170     Server(unsigned short id, unsigned int max_capacity, unsigned short connection_id,
171            std::pair<unsigned int, unsigned short> connection_cost, std::string name, std::string content_type,
172            unsigned int content_size, NodeItem *nodeitem)
173         : name_(name), content_type_(content_type), content_size_(content_size), Node(id, max_capacity,
174            connection_id, connection_cost, nodeitem) {}
175
176     void send(); // normal version
177     void send(Packet* packet); // cheat version
178     void receive(Packet* packet);
179     void print();
180
181     void add(Packet* packet) { packets_.push(packet); }
182
183     std::string getContentType() const { return content_type_; }
184     unsigned int getContentSize() const { return content_size_; }
185
186     std::string getName() const { return name_; }
187 };
188
189 #endif

```

## 6.4 packet.h

```

1 #ifndef PACKET_H
2 #define PACKET_H
3
4 #include <string>
5 #include <time.h>
6 #include <unistd.h>
7 #include <iostream>
8
9
10 class Packet {
11
12 private:
13     unsigned int size_; // !!!CHANGED!!! KILobytes
14     unsigned short sender_id_;
15     unsigned short receiver_id_;
16     std::string content_;
17     time_t time_sent_;
18     bool debug_; // print useful data when moving
19     bool cheat_; // this will bypass the queue
20
21 public:
22
23     Packet(unsigned int size, unsigned short sender_id, unsigned short receiver_id, std::string content)
24         : size_(size), sender_id_(sender_id), receiver_id_(receiver_id), content_(content), debug_(false),
25         cheat_(false) {
26         time_sent_ = time(NULL);
27     }
28
29     ~Packet() = default;
30
31     // getters, setters etc.
32     unsigned short getSenderId() const { return sender_id_; }
33     unsigned short getReceiverId() const { return receiver_id_; }
34     bool isDebug() const { return debug_; }

```

```

34     bool isCheat() const { return cheat_; }
35     unsigned int getSize() const { return size_; }
36     time_t getTimeSent() const { return time_sent_; }
37
38     void setSenderId(unsigned short new_id) { sender_id_ = new_id; }
39     void setReceiverId(unsigned short new_id) { receiver_id_ = new_id; }
40     void setDebug() { debug_ = true; }
41     void setCheat() { cheat_ = true; }
42
43
44     std::string& getContent() { return content_; }
45     void setSize(unsigned int new_size) { size_ = new_size; }
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
51         unsigned int transmission_delay = size_ * 8 / bandwidth; // KB -> Kb
52
53         if (transmission_delay < 1) {
54             double t_d = size_ * 8 / bandwidth; // same as above, seconds
55             t_d *= 1000000; // seconds to microseconds
56             transmission_delay = t_d; // to unsigned int
57             usleep(transmission_delay); // sleep microseconds
58         } else {
59             sleep(transmission_delay); // sleep seconds
60         }
61
62         usleep(1000 * latency); // sleep microseconds
63
64     }
65
66     void print() {
67         std::cout << "Packet from " << sender_id_ << " to " << receiver_id_
68         << "\nContent: " << content_ << ", size: " << size_ << "KB"
69         << "\nDebug packet: " << debug_
70         << "\nTime elapsed since creation: " << time(NULL) - time_sent_ << std::endl;
71     }
72
73 };
74
75
76 #endif
77

```

## 6.5 packetitem.h

```

1  #ifndef PACKETITEM_H
2  #define PACKETITEM_H
3
4  #include <QGraphicsScene>
5  #include <QGraphicsItem>
6  #include <QtMath>
7
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
28         QPolygonF Triangle;
29         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
34         QBrush redBrush(Qt::yellow);
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;
45         qreal dot = QPointF::dotProduct(vector, QPointF(0, vector.ry()));
46         qreal mvector = qSqrt(qPow(vector.rx(), 2) + qPow(vector.ry(), 2));
47
48         qreal angle;

```

```
49         if (vector.rx() >= 0)
50             angle = qRadiansToDegrees(qAcos(-(dot/(mvector*vector.ry()))));
51         else
52             angle = qRadiansToDegrees(qAcos((dot/(mvector*vector.ry())))) - 180;
53
54         this->setRotation(angle);
55         this->setSpeed(mvector);
56         this->setPos(start + (vector/3));
57     }
58
59     void resetTicks() { ticks = 0; }
60
61     void setSpeed(qreal magnitude)
62     {
63         speed = animationSpeed * magnitude / 75;
64     }
65
66     void changeSpeed(double multiplier)
67     {
68         if (multiplier > 0) animationSpeed = multiplier;
69     }
70
71 protected:
72     void advance(int phase)
73     {
74         if (!phase) return;
75         this->setPos(mapToParent(0,-(speed)));
76         ticks += animationSpeed;
77         if (ticks > 30) {
78             this->scene()->removeItem(this);
79             delete this;
80         }
81     }
82
83 private:
84     qreal angle;
85
86     qreal speed;
87
88     double ticks;
89
90     inline static double animationSpeed;
91 };
92
93 #endif // PACKETITEM_H
```



# Index

- addClient
  - MainWindow, [12](#)
- addConnection
  - NodelItem, [20](#)
- addLine
  - MainWindow, [12](#)
- addName
  - NodelItem, [20](#)
- addRouter
  - MainWindow, [12](#)
- addServer
  - MainWindow, [13](#)
- advance
  - NodelItem, [20](#)
  - PacketItem, [24](#)
- call\_print
  - NodeCaller, [16](#)
- call\_print\_routing\_table
  - NodeCaller, [17](#)
- change\_max\_capacity
  - NodeCaller, [17](#)
- changeSpeed
  - NodelItem, [20](#)
  - PacketItem, [25](#)
- Client, [9](#)
  - getName, [9](#)
  - print, [9](#)
  - receive, [10](#)
  - send, [10](#)
- contextMenuEvent
  - NodelItem, [21](#)
- dropPacket
  - MainWindow, [13](#)
- getId
  - NodeCaller, [17](#)
- getName
  - Client, [9](#)
  - Router, [26](#)
  - Server, [28](#)
- getNamelItem
  - NodelItem, [21](#)
- getNode
  - MainWindow, [13](#)
- getNodeByName
  - MainWindow, [14](#)
- itemChange
  - NodelItem, [21](#)
- MainWindow, [10](#)
- addClient, [12](#)
- addLine, [12](#)
- addRouter, [12](#)
- addServer, [13](#)
- dropPacket, [13](#)
- getNode, [13](#)
- getNodeByName, [14](#)
- MainWindow, [11](#)
- sendPacket, [14](#)
- moveTextItem
  - NodelItem, [22](#)
- Node, [14](#)
- NodeCaller, [15](#)
  - call\_print, [16](#)
  - call\_print\_routing\_table, [17](#)
  - change\_max\_capacity, [17](#)
  - getId, [17](#)
  - send\_packet, [17](#)
- NodelItem, [18](#)
  - addConnection, [20](#)
  - addName, [20](#)
  - advance, [20](#)
  - changeSpeed, [20](#)
  - contextMenuEvent, [21](#)
  - getNamelItem, [21](#)
  - itemChange, [21](#)
  - moveTextItem, [22](#)
  - NodelItem, [19](#)
  - operator<<, [22](#)
  - position, [22](#)
  - setColor, [22](#)
  - updateConnections, [23](#)
- operator<<
  - NodelItem, [22](#)
- Packet, [23](#)
- PacketItem, [23](#)
  - advance, [24](#)
  - changeSpeed, [25](#)
  - PacketItem, [24](#)
  - setSpeed, [25](#)
- position
  - NodelItem, [22](#)
- print
  - Client, [9](#)

- 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
  - NodelItem, [22](#)
- setSpeed
  - PacketItem, [25](#)
- updateConnections
  - NodelItem, [23](#)
- WaitThread, [29](#)