8 Appendix

8.1 antGraph

antGraph is a side product of the technology used to create NTS. The internal structure used to represent the NTS network within the ntsMaster program being based on a graph; a study of graphs was undertaken. It was quickly seen that the NTS network had too many proprietary behaviors compared with a standard graph for it to be worth sub-classing from a graph structure. An underlying graph structure was therefore dropped for a specialized structure, though based on the graph design, as described in the ntsMaster section.

The set of original classes for the purpose of using an underlying graph structure having been designed and implemented, these were updated to create small graph framework. The next few paragraphs will quickly describe these classes, which are included in the NTS distribution for completeness.

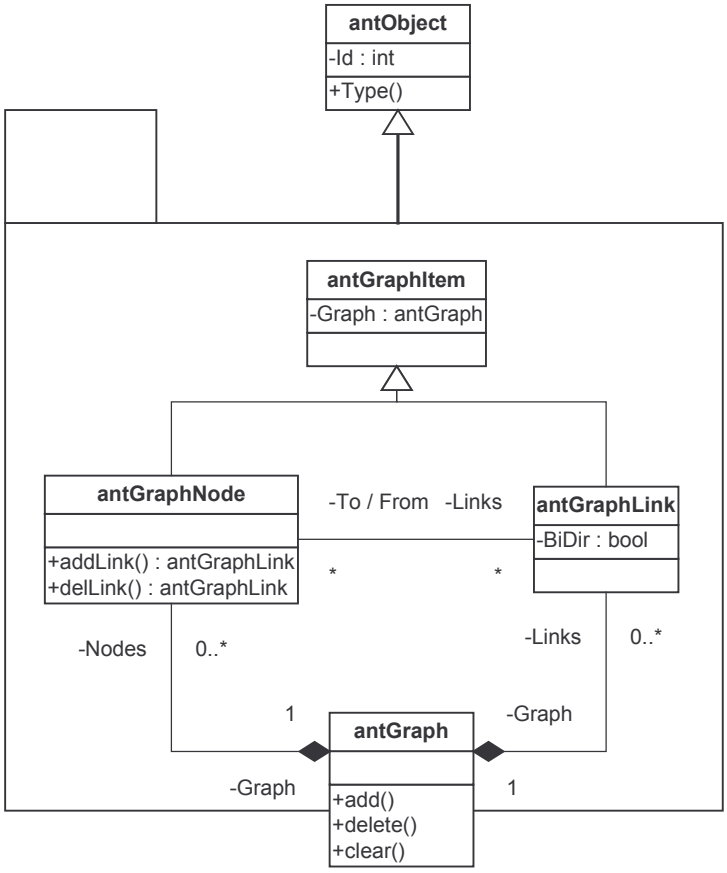

Figure 33: antGraph Package

The *antObject* inherits the Qt *QObject*, so that the whole structure can use signals and slots (for more information on signals and slots refer section 2.3 Qt). As in the ntsNetwork framework, all instantiation of the structure should be done via the *antGraph* facade. Any modification affecting the structure of the graph should also be done via this class.

Standard applications of graphs are supported: adding a node, adding links in between nodes (these have direction, or can be bi-directional), removing a node (with link removal cascading), and removing a single link. The option can be set whether all links are bi-directional (this converts all previously created links to bi-directional ones).

To use this structure to add more functionality to the nodes or links, one should create a map with the nodes or links returning a class with the appropriate extra functionality. Another approach would be to subclass. If only limited extra functionality is needed one can re-implement the "init" function, which is run at instantiation (this technique is used in Qt Designer). A full sub-classing of all classes can also be implemented for more complex additional functionalities, with the new classes calling their parent's functions when needed. When this is the case keep in mind that, it is the re-implementation of the *antGraph* class, which should handle all instantiation if you want to keep the graph stable.

8.2 UML Diagrams

Follows some UML Diagrams of some of the main packages of NTS

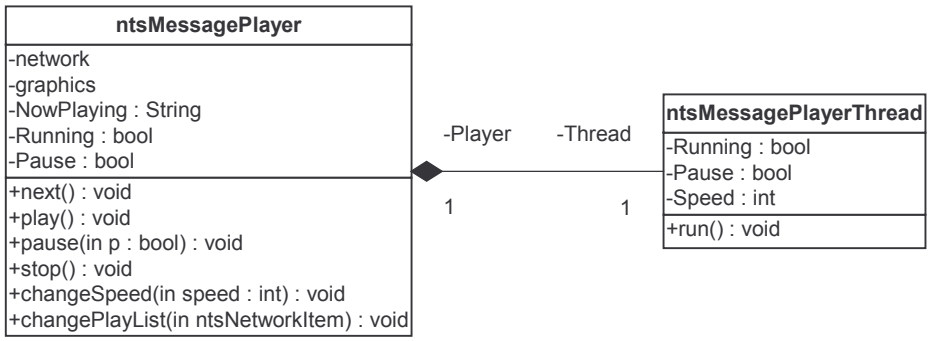

Figure 34: ntsPlayer

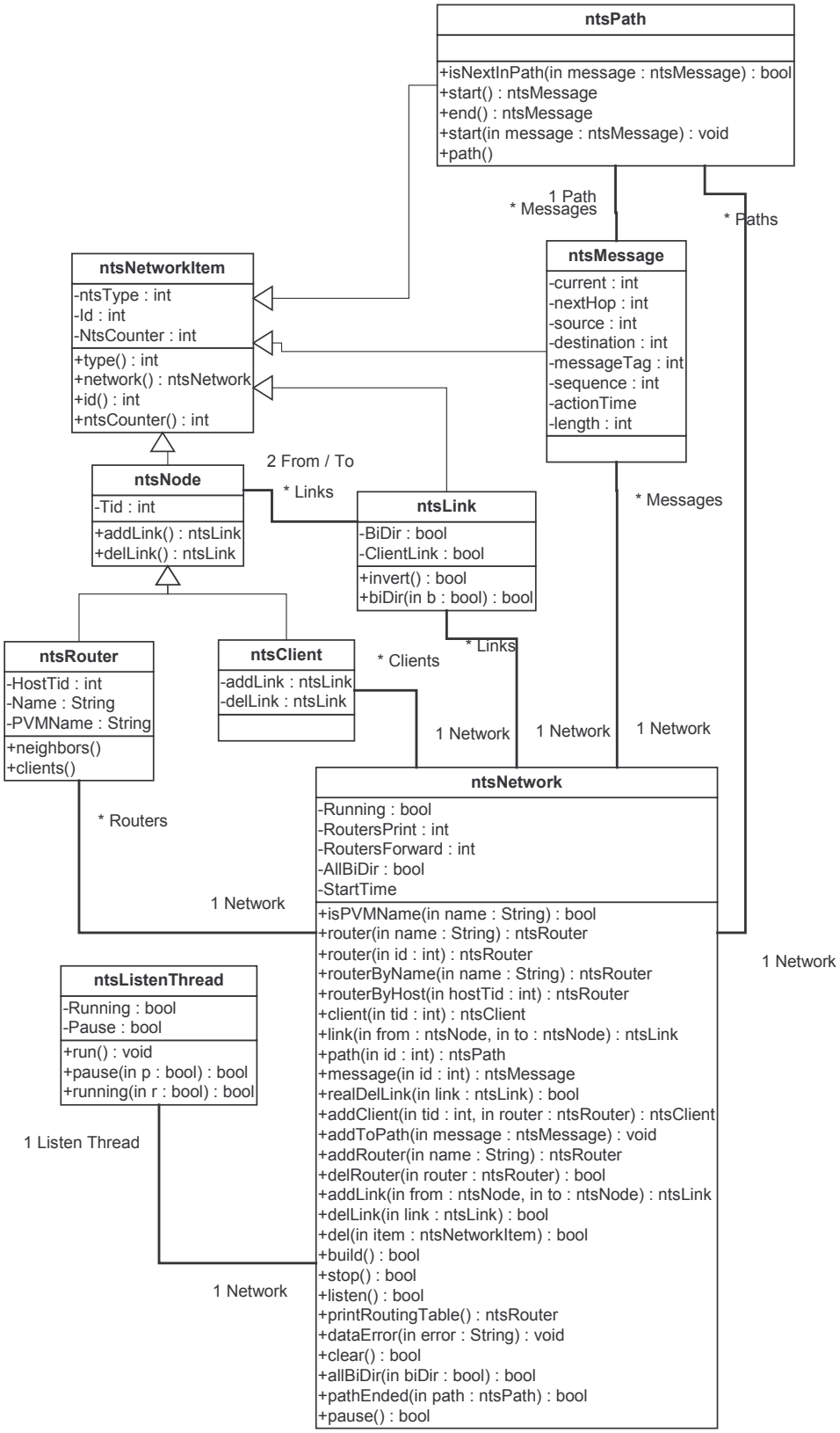

Figure 35: ntsNetwork Package

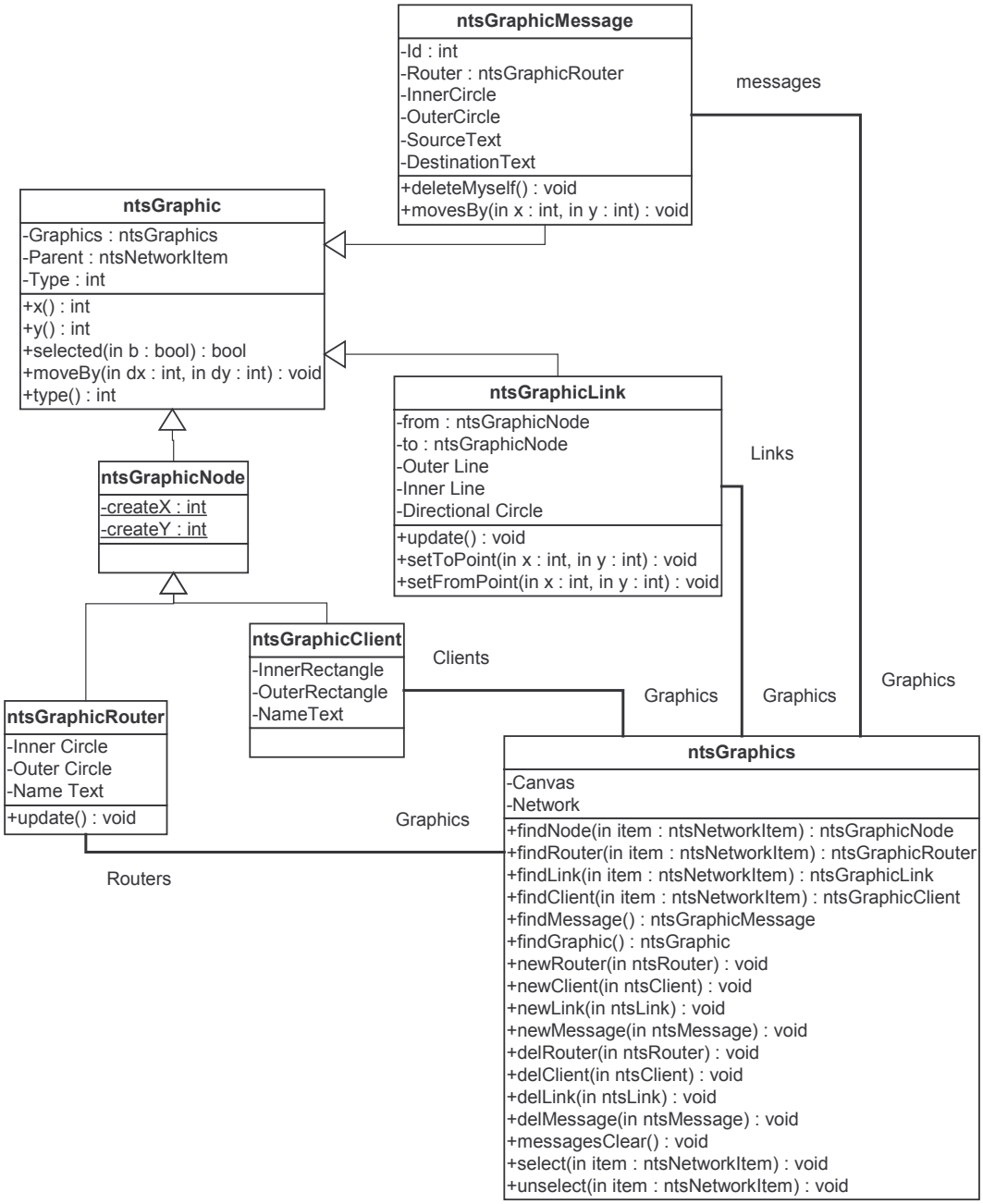

Figure 36: ntsGraphics Package

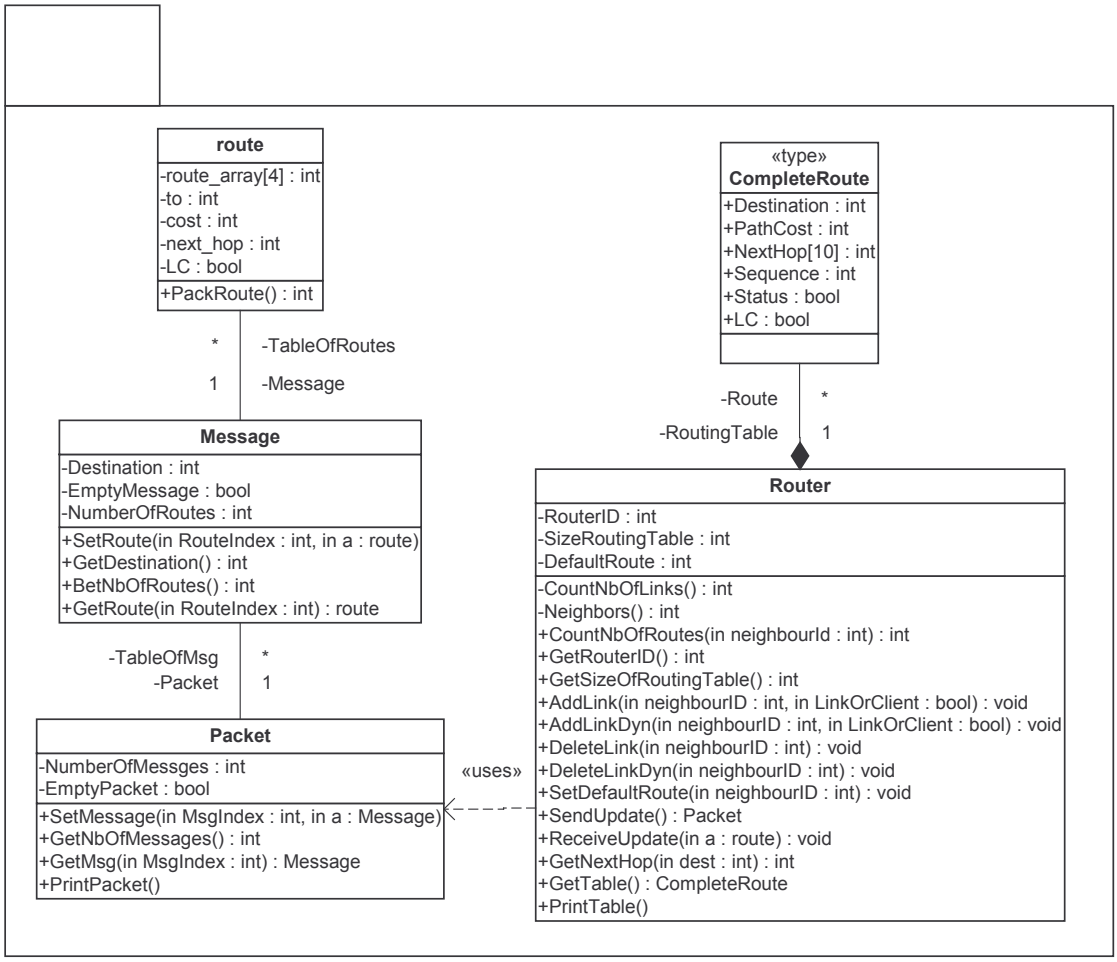

Figure 37: ntsRouter, Router Package