

Utilities in the GMLPS Lightwave Agile Switching Simulator (GLASS)

Version: Draft 1.0

TABLE OF CONTENTS

| | |
|--|-----------|
| TABLE OF TABLES | II |
| 1 INTRODUCTION | 1 |
| 2 THE CLASS GOV.NIST.ANTD.SSF.UTIL.NETUTIL..... | 1 |
| 2.1 THE NODES..... | 1 |
| 2.2 THE LINKS..... | 2 |
| 2.3 SIMULATION CONTROL..... | 2 |
| 2.4 TOPOLOGY CONVERSION | 3 |
| 3 THE CLASS GOV.NIST.ANTD.OPTICAL.UTIL.NETUTIL..... | 3 |
| 4 THE CLASS GOV.NIST.ANTD.OPTICAL.UTIL.NODEUTIL..... | 3 |
| 4.1 OPTICAL NETWORK INTERFACE CARDS (ONICs) | 3 |
| 4.2 THE LINKS | 4 |
| 5 THE CLASS GOV.NIST.ANTD.OPTICAL.UTIL.LINKUTIL..... | 5 |
| 6 THE CLASS GOV.NIST.ANTD.OPTICAL.UTIL.PATHUTIL | 6 |
| 6.1 CREATION OF CONNECTIONS AND ROUTES | 6 |
| 6.2 GETTING INFORMATIONS..... | 7 |
| 7 THE CLASS GOV.NIST.ANTD.MERLIN.UTIL.ALGOUTIL | 9 |
| 7.1 EXECUTING ALGORITHMS..... | 9 |
| 7.2 GETTING STATISTICS..... | 9 |
| 8 THE CLASS GOV.NIST.ANTD.MERLIN.UTIL.BACKUPUTIL | 10 |
| 8.1 BACKUP LINK | 11 |
| 8.2 BACKUP ROUTE | 12 |
| 8.3 OTHER TOOLS..... | 12 |
| 9 THE CLASS GOV.NIST.ANTD.MERLIN.UTIL.CONNECTIONUTIL..... | 14 |

| | | |
|-----------|--|-----------|
| 9.1 | CREATING/DELETING CONNECTIONS | 14 |
| 9.2 | SETTING THE SWITCHES | 14 |
| 9.3 | CONFIGURING THE CONNECTIONS..... | 15 |
| 9.4 | MORE TOOLS | 16 |
| 10 | THE CLASS GOV.NIST.ANTD.MERLIN.PROTOCOL.UTIL.LINKSTATETABLE | 17 |
| 11 | THE PACKAGE GOV.NIST.ANTD.MERLIN.ALGORITHM.ROUTING.UTIL | 17 |
| 11.1 | THE CLASS GRAPH..... | 17 |
| 11.2 | THE CLASS VERTEX | 17 |
| 11.3 | THE CLASS EDGE | 17 |

TABLE OF TABLES

| | | |
|----------|---|----|
| Table 1 | The nodes in the Net..... | 1 |
| Table 2 | The links in the Net | 2 |
| Table 3: | Manipulation of the running simulation | 2 |
| Table 4 | ONICs in node..... | 3 |
| Table 5 | Links attached to a node | 4 |
| Table 6 | Nodes and interfaces attached to a link | 5 |
| Table 7 | Finding a connection | 7 |
| Table 8 | Information about connections..... | 8 |
| Table 9 | Running algorithms for one connection | 9 |
| Table 10 | Algorithm statistics..... | 10 |
| Table 11 | Backup link tools | 11 |
| Table 12 | Working with the switches..... | 14 |
| Table 13 | Using connections | 15 |

1 INTRODUCTION

This document presents some useful classes, which provides an easy access to the information located in the GLASS framework.

2 THE CLASS `GOV.NIST.ANTD.SSF.UTIL.NETUTIL`

This class contains methods that provide access to the elements of a Glass instance.

All the methods located in this class are **static**.

2.1 THE NODES

Multiple functions allows the user to retrieve the nodes depending on the type wanted:

Table 1 The nodes in the Net

| Method | Return |
|---|--|
| <i>public static Vector getOXCs(Glass net)</i> | A Vector of OXC in the net. |
| <i>public static Vector getOXCEdgeRouters(Glass net)</i> | A Vector of OXCEdgeRouter in the net. |
| <i>public static Vector getLSRs(Glass net)</i> | A Vector of LSR in the net. |
| <code>public static Vector getRouters(Glass net)</code> | A Vector of Router in the net. |
| <i>public static Vector getHosts(Glass net)</i> | A Vector of Host in the net. |
| <i>public static Vector getNonOpticalNodes(Glass net)</i> | Non optical nodes (Router and Host). |
| <i>public static Vector getNodes(Glass net)</i> | All the nodes of the net. |
| <code>public static Host getNode(int id, Glass net)</code> | The node with the given id or null. |
| <code>public static Host getNodeOfIP (int ip, Glass net)</code> | The node with the given IP address. The IP address is located in the NIC (Network interface card). |
| <i>public static EventManager getEventManager(Glass net)</i> | The EventManager for the given net. |

The class EventManager is a special node in the GLASS framework. This node is used to handle the scripted events, and is also used by the GLASS-TSC (Topology and Simulation Creator) to interact with the simulation.

2.2 THE LINKS

Table 2 The links in the Net

| Method | Return |
|---|---|
| <i>public static Vector getOpticalLinks(Glass net)</i> | A Vector of optical links in the net. |
| <i>public static Vector getNonOpticalLinks(Glass net)</i> | A Vector of non-optical links in the net. |
| <i>public static Vector getLinks(Glass net)</i> | A Vector of all the links in the net. |
| <i>public static link getLink(int id, Glass net)</i> | The link with the given id or null. |

Note: In the SSF framework, the link does not have an id. The GLASS framework requires an id in all optical links. Here fore GLASS also allows the link to have an id but it is not mandatory. So to retrieve a link that does not have an id, use getLinks (Glass net) or use another method (depending if you know the node or not). This method returns a list of all links and the developer has to determine the correct id on his own.

2.3 SIMULATION CONTROL

The following methods are used by the GLASS-TSC to control the simulation run.

Table 3: Manipulation of the running simulation

| Method | Action |
|---|---|
| <i>public static void stopSimulationProgress(Glass net)</i> | Pauses the simulation by blocking the EventManager (so that the discrete event simulation is in a pause mode). |
| <i>public static void ResumeSimulationProgress(Glass net)</i> | Resumes the simulation of the given net after it has been paused. |
| <i>public static void setSimulationSpeed (Glass net, int slpTime)</i> | control the simulation speed. The slpTime is a value between 0 (for no delay) and 100 (to stop the simulation). |

| | |
|---|--|
| <i>public static long getSimulationTime (Glass net)</i> | Returns the current simulation time in simulation ticks. To convert the ticks into real time, divide it by the static attribute “Glass.frequency”. |
|---|--|

2.4 TOPOLOGY CONVERSION

Sometimes it is useful to have the topology information in a table. NetUtil provide a method that returns a matrix.

public static int[][] getTopologyMatrix (Glass net) returns the topology in a matrix of ids. Each line of the result is as follow: [Node1_ID; Node2_ID; Link_ID]. Note that it only includes the optical links.

3 THE CLASS GOV.NIST.ANTD.OPTICAL.UTIL.NETUTIL

This class has been **deprecated**. Instead use the class gov.nist.antd.ssf.util.NetUtil.

4 THE CLASS GOV.NIST.ANTD.OPTICAL.UTIL.NODEUTIL

This class contains methods to facilitate the access of information that are contained in a node.

4.1 OPTICAL NETWORK INTERFACE CARDS (ONICS)

The following methods helps to retrieve one or all the optical interfaces of a node.

Table 4 ONICs in node

| Method | Return |
|--|--|
| <i>public static Vector getONICs (Host node)</i> | All the optical interfaces of a node. |
| <i>public static ONIC getONIC(Host node, int onicID)</i> | The ONIC of the given id in the given host or null if there is no such ONIC in the given host. |
| <i>public static ONIC getONICfromIP(Host node, int onicIP)</i> | The ONIC of the given ip in the given host or null if there is no such ONIC in the given host. |

| | |
|--|--|
| <i>public static ONIC getONICofPort(Host node, int port)</i> | The ONIC that contains the given port in the given host or null if there is no such port in the given host (see Note 2). |
| <i>public static ONIC getONIC(Host node, Fiber fiber)</i> | The ONIC of the given host that is attached to the given fiber or null if there is no ONIC attached to the fiber. |
| <i>public static ONIC getONIC(Host node, Lambda lambda)</i> | The ONIC of the given host that is attached to the given lambda or null if there is no ONIC attached to the lambda. |

Note1: a public HashTable (called interfaceAdresses) is available in the host where the values are the network interfaces of the node (NIC and ONIC).

Note2: The port is used to identify a fiber in an ONIC. As opposed to the fiber id, which is unique in an ONIC, the fiber port id is unique in a node.

4.2 THE LINKS

The NetUtil class also provides easy ways to access the links connected to a given node.

Table 5 Links attached to a node

| Method | Return |
|---|--|
| <i>public static Vector getOpticalLinks(ExtRouter node1, ExtRouter node2)</i> | A Vector of optical links that connect the 2 given nodes (must be LSR, OXCEdgeRouter or OXC). |
| <i>public static Vector getOpticalLinks(ExtRouter forNode)</i> | A Vector of links connected to the given node. |
| <i>public static OpticalLink getLink(Host node, int onicID)</i> | The optical link connected to the ONIC represented by its id in the given node or null if no such ONIC exists. |

| | |
|--|---|
| <i>public static Fiber getFiber (Host node, int onicID, int fiberID)</i> | The fiber (described by the ONIC-id and the fiber-id) in the given node or null if no such fiber. |
| <i>public static Lambda getLambda (Host node, int onicID, int fiberID, int lambdaID)</i> | The given lambda (described by the ONIC id, fiber id and lambda id). |

Note1: The id of an interface (NIC and ONIC) is unique inside a node.

Note2: The id of a fiber is unique in an ONIC.

Note3: The id of a lambda is unique in a fiber.

5 THE CLASS GOV.NIST.ANTD.OPTICAL.UTIL.LINKUTIL

This class contains easy access for components attached to a given link.

Table 6 Nodes and interfaces attached to a link

| Method | Return |
|--|--|
| <i>public static int[] getNodeIDs (_link link)</i> | An array of the node ids that are attached to the given link. |
| <i>public static ONIC getONIC(OpticalLink link, int nodeID)</i> | The ONIC that attach the given link to the node, represented by its id, or null if the node does not exist or if the link does not connect the node. |
| <i>public static boolean areSameAttachedNodes (OpticalLink link1, OpticalLink link2)</i> | True if the links connect the same nodes. |

Note: The class is SSF.Net._link (subclass of SSF.Net.link), which would forbid the use of instance of the class SSF.Net.link, but the Glass framework is creating the SSF.Net._link object for regular links in lieu of link SSF.Net.link Objects. This means the user can give any link that has been created during the simulation.

6 THE CLASS GOV.NIST.ANTD.OPTICAL.UTIL.PATHUTIL

This class is useful when working with connections, routing and path over optical networks.

6.1 CREATION OF CONNECTIONS AND ROUTES

createRouteObject:

public static OpticalConnection createRouteObject (ExtRouter source, ExtRouter dest, QualityOfService qos) has been deprecated because it creates a OpticalConnection object and not a route. Use the method createOpticalConnection instead.

CreateOpticalConnection:

public static OpticalConnection createOpticalConnection(ExtRouter source, ExtRouter dest, QualityOfService qos) creates a connection object between the source and the destination with the given quality of service. This method does not compute any algorithm but the return value is required as an input for the routing and wavelength algorithm.

CreateRouteFromNodes:

public static void createRouteFromNodes(int[] nodesId, OpticalConnection oRoute) creates the structure for a route inside the given optical connection. This is because a route is composed of a Vector of gov.nist.antd.optical.util.PtPBundle and that GLASS facilitates the implementations of are only dealing with IDs. If an algorithm computes a route using a list of node ids, then a call to this method will generate the correct internal data.

public static void createRouteFromNodes(Vector nodes, OpticalConnection oRoute) creates the internal data for the given optical connection using the Vector of nodes that compose the route.

If the Vector of nodes does not create a route from the source to the destination of the given optical connection, an `IllegalDataException` will be thrown.

CreateRouteFromLinks:

public static void createRouteFromLinks(int[] linksId, OpticalConnection oRoute) creates the data inside of the given optical connection (oRoute) using the list of link ids that represents the route from the source to the destination of the connection.

public static void createRouteFromLinks(Vector links, OpticalConnection oRoute) creates the data inside of the given optical connection (oRoute) using the list of links that represent the route from the source to the destination of the connection.

To remove a connection, one way is to call the following method:

public static boolean deleteConnection(OpticalConnection oRoute) removes the given optical connection from the PathContainer that stores it.

6.2 GETTING INFORMATIONS

Table 7 Finding a connection

| Method | Return |
|--|--|
| <i>public static OpticalConnection getRoute(Glass net, int id)</i> | Has been deprecated , use getConnection. |
| <i>public static OpticalConnection getConnection (Glass net, int id)</i> | The connection that has this id or null if no such connection is available. |
| <i>public static OpticalPath[] getAllPaths(OpticalLink link, PathContainer container)</i> | An array of all the OpticalPath where the given link is used in the given PathContainer. The array can be of size 0 if the link is never used by any connection. |
| <i>public static OpticalConnection[] getAllRoutes (OpticalLink link, PathContainer container)</i> | Has been deprecated . Use getAllConnections instead. |
| <i>public static OpticalConnection[] getAllConnections (OpticalLink link, PathContainer container)</i> | Returns an array of all the OpticalConnection where the given link is used in the given PathContainer. The array can be of size 0 if the link is never used by any connection. |
| <i>public static OpticalConnection getRouteOfLambda (Glass net, Lambda lambda)</i> | Has been deprecated , use getConnectionOfLambda. |
| <i>public static OpticalConnection getConnectionOfLambda (Glass net, Lambda lambda)</i> | The connection that is using the given lambda or null if no connection is using it. |
| <i>returns public static PathContainer[] getPathContainers(Glass net)</i> | All the path containers of the given network. |

Note: The class PathContainer is used by an algorithm to store the computed routes. This is the location where protocols can retrieve available connections.

Table 8 Information about connections

| Method | Return |
|--|---|
| <i>public static int getAvailableId(Glass net)</i> | an id that is not used by any connection. This is useful because the connection id is unique in the all system. |
| <i>public static boolean isRouteIdUsed(Glass net, int id)</i> | Has been deprecated , use <code>isConnectionIdUsed</code> . |
| <i>public static boolean isConnectionIdUsed(Glass net, int id)</i> | True if the given id is already used by another connection. |
| <i>public static boolean contains(OpticalConnection route, OpticalLink link)</i> | True if the given connection is using the link in its path. |
| <i>public static boolean contains(OpticalConnection route, Lambda lambda)</i> | True if the given connection is using the lambda in its path. |
| <i>public static int[] getSenderInformations(OpticalConnection oRoute)</i> | The port id's used by this connection on the sender side (see Note 1). |
| <i>public static int[] getReceiverInformations(OpticalConnection oRoute)</i> | The port id used by this connection on the receivers side |
| <i>public static String getReceiverProtocol(OpticalConnection route)</i> | The name of the protocol that is attached to this connection. |

Note1: The port id is very important because it represents the information a protocol needs to have in order to send data through the optical network using the OXCSwitch. There is one port per OpticalChannel.

public static OpticalChannelSegment getSegment (Glass net, Lambda lambda) returns the OpticalChannelSegment that is attached to this lambda.

Note2: An OpticalChannelSegment is attached to a lambda and shows that this lambda is used in an OpticalPath.

7 THE CLASS GOV.NIST.ANTD.MERLIN.UTIL.ALGOUTIL

This class provides tools for algorithms like executing and computing statistics.

7.1 EXECUTING ALGORITHMS

Table 9 Running algorithms for one connection

| Method | Return |
|--|---|
| <i>public static OpticalConnection executeRouting (OpticalConnection route)</i> | Executes the routing algorithm specified in the given connection. It also reset all the previous routes that may have been computed before in the connection. |
| <i>public static OpticalConnection executeWavelengthAssignment (OpticalConnection route)</i> | Executes the wavelength algorithm specified in the given connection. The given route must contain at least one possible route in order to complete the OpticalPath. |
| <i>public static OpticalConnection executeRWA (OpticalConnection route)</i> | executes the Routing and Wavelength Assignment algorithm (RWA). The RWA execute the routing and wavelength in one step. |

7.2 GETTING STATISTICS

public static Vector getRoutesUsingAlgo (String algoName, Glass net) returns the Vector of connections that is using the given algorithm in the given network.

public static int getNoLambdas (Glass net) returns the total number of lambdas in the net.

public static double[] getStatistic (String algoName, Glass net) returns the statistics of the given algorithm. The values are dependent on the type of the algorithm.

Table 10 Algorithm statistics

| Method | Return |
|--|---|
| <i>public static double[] getStatisticsFromRouting (Vector routes)</i> | The statistics of the given set of connections using the routing algorithm. The result is average number of hops, blocking probability, average delay, average distance, and throughput. |
| <i>public static double[] getStatisticsFromWavelength (Vector routes)</i> | The statistics of the given set of connections related to their wavelength algorithm. The result is wavelength utilization and the blocking probability. |
| <i>public static double[] getStatisticsFromRWA (Vector routes)</i> | The statistics of the given set of connections related to the RWA. The result is the average number of hops, the blocking probability, average delay, average distance, throughput, and wavelength utilization. |

8 THE CLASS GOV.NIST.ANTD.MERLIN.UTIL.BACKUPUTIL

This class gives some basic implementation for some functionality that can be useful for developers of backup/restoration protocols. The user has to understand that these methods may not be executed under all the possible scenarios. It is primary to give the user an idea on how to use and access the framework an all the methods may have not been tested yet.

8.1 BACKUP LINK

The following set of methods is used for a backup link algorithm:

Table 11 Backup link tools

| Method | Return |
|--|---|
| <i>public static OpticalConnection getBackupOfLink (int linkID, QualityOfService qos, Glass net)</i> | Tries to create a backup connection around the given link (that may has failed), by using the given Quality of Service. The link is represented by its id so the network must also be specified. If the backup is working the return value is the backup connection. Also see Note 1. |
| <i>public static OpticalConnection getBackupOfLink (OpticalLink link, QualityOfService qos)</i> | Tries to create a backup connection around the given link (that may have failed), using the given Quality of Service. If the backup is working the return value is the backup connection. |
| <i>public static OpticalConnection getBackupOfLink (OpticalConnection route, QualityOfService qos)</i> | Creates a backup of the given connection using the given QualityOfService. If the backup is working the return value is the backup connection otherwise null. |
| <i>public static void restoreLink (OpticalConnection route, OpticalLink link, Vector backup)</i> | Updates the route associated to the given connection by changing the link by the backup (the Vector contains the list of links that should replace the link). |
| <i>public static Vector[] restoreLink (Vector bundles, OpticalLink link, Vector backup)</i> | Looks into the Vector of bundles (that represents a route) and replace the given link by the given backup (the Vector contains the list of links that should replace the link). This manipulation may result in the splitting of |

| | |
|--|--|
| | the bundles. That's why the result is an array of new bundles. |
|--|--|

Note1: The QualityOfService also contains the algorithms to use.

Note2: All this methods also attached the backup route to the working route. The user has to make sure that the working route will be attached to the backup route (not done in every methods).

8.2 BACKUP ROUTE

If a protocol needs to create backup routes, then the following methods can be used:

public static OpticalConnection getBackupOfRoute (int routeID, QualityOfService qos, Glass net) creates a backup of the given connection (specified by its id in the given net) using the given QualityOfService. If the backup is working the return value is the backup connection.

public static OpticalConnection getBackupOfRoute (OpticalConnection route, QualityOfService qos) creates a backup of the given route using the given QualityOfService. If the backup is working the return value is the backup connection.

8.3 OTHER TOOLS

static OpticalConnection runAlgorithms (OpticalConnection oRoute) is not public and is primary used by the backup methods shown before. It calls the different algorithms of the given connection.

public static void filterRoute (OpticalConnection route) is used to remove redundancy in the route of the given connection.

For example, if the route is defined by the link list 3-5-7-7-8-9, we can remove the 7 because this means the message will go in both ways of the link. The result will then give a new link list 3-5-8-9 as shown is the following figure:

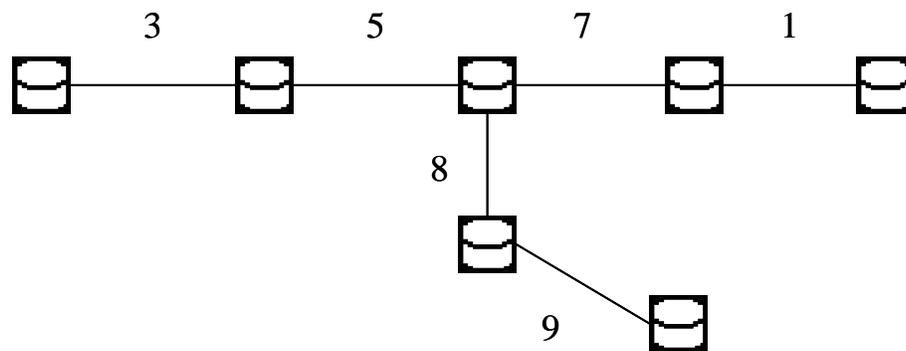


Figure 1 Illustration for the method filterRoute()

This method considers that there is only one link per bundle.

public static void replaceSegment(OpticalConnection route, OpticalConnection repRoute) is not implemented yet. It only checks the compatibility of the two given connections.

private static void checkRouteCompatibility (OpticalConnection route, OpticalConnection repRoute) checks if the connections have the same bandwidth.

public static void mergeRoutes (OpticalConnection route1, OpticalConnection route2) merges two routes without merging their path but use the current path to get information. Example, the route1 has a route defined as follow: 1-2-3-4-5-6 and route2 as 3-8-9-10. The result on the connection 1 would be 1-2-3-8-9-10-6 as shown below:

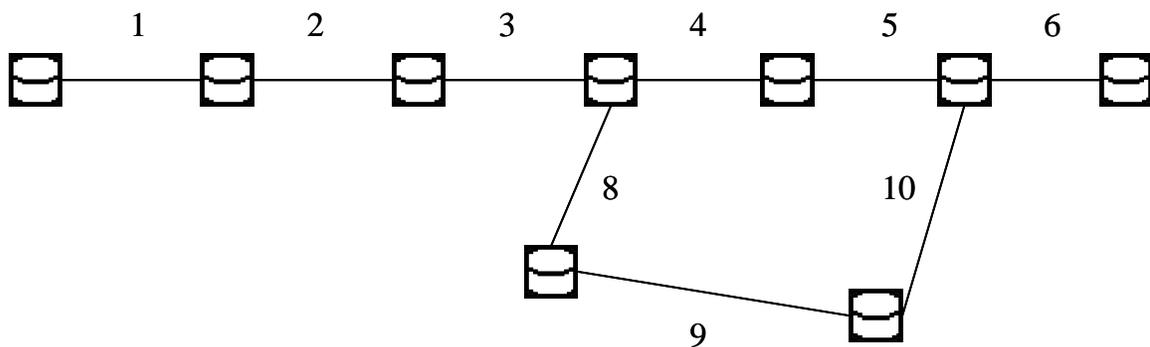


Figure 2 Illustration for the method mergeRoutes()

9 THE CLASS GOV.NIST.ANTD.MERLIN.UTIL.CONNECTIONUTIL

In this class you can find more tools about the connections.

9.1 CREATING/DELETING CONNECTIONS

public static OpticalConnection createConnection(ExtRouter source, ExtRouter dest, QualityOfService qos, boolean autoConnect) creates a connection from the source to destination node by using the information passed within the qos parameter. The boolean attribute autoConnect is used to specify if the OXC switch at the intermediate nodes must be set-up. For example if specify true if the user just needs a connection ready to use it. Otherwise if a signaling protocol is used to setup the switches then autoConnect will be false.

public static boolean deleteConnection(OpticalConnection oRoute) unregisters the potential owner of this connection and remove the connection from the path container.

9.2 SETTING THE SWITCHES

Table 12 Working with the switches

| Method | Return |
|---|---|
| <i>public static synchronized boolean connectSwitches(OpticalConnection oRoute)</i> | Tries to setup the switches along the connection path. In addition to this it also looks at the available add and drop ports at the source and destination. If the setup of the switches is done, it returns true. If an error occurred, the return value is false. |
| <i>public static synchronized boolean connectSwitches(OpticalConnection oRoute, int[] addPort)</i> | is similar to the previous method but the user specifies the list of add ports that should be used at the source node. The drop ports at the destination node are determined according to the availability. |
| <i>public static synchronized boolean connectSwitches(OpticalConnection oRoute, int[]</i> | tries to setup the switches of the given connection and uses the given add and drop |

| | |
|---|--|
| <i>addPort, int[] dropPort)</i> | ports at the source and destination nodes. If the set-up is done, it returns true otherwise false. |
| <i>public static synchronized boolean disconnectSwitches (OpticalConnection oRoute)</i> | Disconnects the switches (of the OXCSwitch) of the given connection to free the resources. The lambdas of the path will still be used. |

9.3 CONFIGURING THE CONNECTIONS

Table 13 Using connections

| Method | Return |
|--|--|
| <i>public static int[] register (ProtocolSession session, OpticalConnection oRoute)</i> | Tries to register the given protocol to the given route. At the destination, the receiver must be the same protocol. See the next method for more details. If the registration succeeds the return value is the add ports that are used by the connection, otherwise it returns null (see Note 1). |
| <i>public static int[] register (ProtocolSession srcSession, String destProtocolName, OpticalConnection oRoute)</i> | Tries to connect the given protocol to the given connection. The protocol on the receiver side is given so that two protocols with different names can still communicate (see Note 1). |
| <i>public static boolean unregister(ProtocolSession session, OpticalConnection oRoute)</i> | Tries to unregister the protocol that owns the given connection. If the protocol is not the owner of the connection then the returned value is false, otherwise true. This method does not reset the switches of the route. |

Note1: There are many reasons why the registration can fail:

- The connection is not configured (the switches not done, including the add and drop port), or no path is available,
- The path is failed and there is no backup or the backup is also failed,
- The connection is already used by another protocol,
- Some data are missing (like OXCSwitch at the source or destination).

If the registration is failed the returned value is null otherwise it is the list of add port that can be used by the protocol to send data over this connection.

9.4 MORE TOOLS

public static OpticalConnection findConnection(ExtRouter source ,ExtRouter dest ,QualityOfService qos) is looking at the path containers to find a connection that match the given parameter. If a route has the same source, same destination and the quality of service is compatible then the returned value is the found connection. Otherwise null.

public static boolean isCompatible(OpticalConnection route, ExtRouter source, ExtRouter dest ,QualityOfService qos) returns true if the given route has the same source, destination and if the quality of service is also compatible. The compatibility of the quality of service is just defined by the same algorithms. Also the given connection must not be used for protection purpose and must be available.

public static int[] getSenderInformations(OpticalConnection oRoute) returns the list of add ports used at the source for the given route. If the route is not connected to add port or if the path is failed, then the returned value is null. This method is used to test if a protocol can send information through the given route and to get the information about the port.

public static int[] getReceiverInformations(OpticalConnection oRoute) is equivalent to the previous one but on the receiver side. If the path is failed or the connection is not connected to drop ports then the returned value is null otherwise it represents the ports attached to the receiving protocol.

Note: All the methods in the class ConnectionUtil require that the developer is using the connections structure of the framework. If the connections created by an algorithm are not put in a PathContainer and the structure does not follow the one that is provided by GLASS, these methods will not work properly.

10 THE CLASS `GOV.NIST.ANTD.MERLIN.PROTOCOL.UTIL.LINKSTATE``TABLE`

This class provides a basic table to store link state information. This table can be used to store information about the failure of lambdas, fibers and links.

11 THE PACKAGE `GOV.NIST.ANTD.MERLIN.ALGORITHM.ROUTING.UTIL`

This package contains some basic classes to create a graph out of the network. This graph is used in the routing algorithms to decide which way to use.

11.1 THE CLASS `GRAPH`

This class is the main class that creates a graph out of the net given in the constructor.

The nodes (class `Vertex`) of the graph do only have representants of nodes using a subclass of `ExtRouter`, which includes the `LSR`, `OXCEdgeRouter`, and `OXC`. This is because the algorithms have primary been developed to compute routes over an optical network. The arcs (class `Edge`) represent the optical links between the nodes. To check if an edge must be created between node A and node B, the graph checks that there is at least one data lambda in one fiber that would allow a communication between A and B. It is possible to have more than one edge between two nodes if there are multiple links between these two nodes.

11.2 THE CLASS `VERTEX`

As mentioned before, this class is attached to a node (subclass of `ExtRouter`) and also contains an attribute `dist` that is used to represent the distance from a specific node another one. The meaning of this distance depends one the algorithm (number of hops, delay, physical distance...).

11.3 THE CLASS `EDGE`

This class is attached to a link and indicates that there is a lambda available between the two attached nodes. In the edge, the attribute `cost` is a link, which allows that algorithm to get any information out of the link (distance, delay, bandwidth...).

Note: These components can be easily extended to create a customized graph depended on different criteria. For example, the class `gov.nist.antd.merlin.protocol.protectionlink.LinkGraph` is creating a graph where there is no checking about the presence of data lambdas in the fibers.