# Robotics Service Bus - Bug #671
## rsb-java does not shutdown properly if a Spread connection fails

10/21/2011 09:22 PM - J. Moringen

| | | | |
|---|---|---|---|
| **Status:** | In Progress | **Start date:** | 10/21/2011 |
| **Priority:** | Normal | **Due date:** | |
| **Assignee:** | S. Wrede | **% Done:** | 20% |
| **Category:** | Java | **Estimated time:** | 0.00 hour |
| **Target version:** | | | |

**Description**

When trying to deactivate participants after a Spread connection has failed, something along the lines of the following happens:

```
spread.SpreadException: write(): java.net.SocketException: Broken pipe
    at spread.SpreadConnection.multicast(SpreadConnection.java:1934)
    at spread.SpreadConnection.disconnect(SpreadConnection.java:961)
    at rsb.transport.spread.SpreadWrapper.deactivate(SpreadWrapper.java:309)
    at rsb.transport.spread.SpreadPort.deactivate(SpreadPort.java:376)
    at rsb.transport.Router.deactivate(Router.java:143)
    at rsb.Listener$ListenerStateActive.deactivate(Listener.java:54)
    at rsb.Listener.deactivate(Listener.java:123)
    at rsb.patterns.Method$MethodStateActive.deactivate(Method.java:61)
    at rsb.patterns.Method.deactivate(Method.java:160)
    at rsb.patterns.Server$ServerStateActive.deactivate(Server.java:52)
    at rsb.patterns.Server.deactivate(Server.java:116)
    at client.main(client.java:138)
```

There seem to be two related issues:
- Since client code cannot handle this situation properly, the exception should be handled (probably by just ignoring it at rsb.transport.Router.deactivate(Router.java:143)) within RSB
- After the exception is thrown, some threads remain (not the main thread, since it terminates with the above exception) and continue to produce this kind of output:

```
Oct 21, 2011 9:09:24 PM rsb.transport.spread.ReceiverTask run
WARNING: Caught a SpreadException while trying to receive a message: Connection closed while reading header
Oct 21, 2011 9:09:24 PM rsb.transport.spread.ReceiverTask run
WARNING: Caught a SpreadException while trying to receive a message: Connection closed while reading header
Oct 21, 2011 9:09:24 PM rsb.transport.spread.ReceiverTask run
```

These could belong to participants associated to other methods of the remote server which do not get deactivated because of the exception.

Another instance of the problem can be seen here (search for "TimeoutException" in the output).

**Related issues:**

| | | |
|---|---|---|
| Blocked by Robotics Service Bus - Feature # 674: Implement ErrorHandler subsy... | **New** | **10/21/2011** |

### Associated revisions

**Revision 4dac83ba - 10/21/2011 10:52 PM - J. Moringen**

Deactivate server objects properly in java/{client,server}.java

refs #671 (since deactivation triggers that bug)

- java/client.java: deactivate the server object, even if an exception
  is thrown
- java/server.java: similar

**Revision 9b10f0c3 - 10/21/2011 11:38 PM - S. Wrede**

stop receiver task when connection is lost, refs #671

## History

**#1 - 10/21/2011 09:23 PM - J. Moringen**

*- Subject changed from rsb-java does not shutdown properly if the Spread fails to rsb-java does not shutdown properly if a Spread connection fails*

**#2 - 10/21/2011 09:26 PM - J. Moringen**

*- Description updated*

**#3 - 10/21/2011 09:57 PM - S. Wrede**

*- Status changed from New to In Progress*
*- Assignee set to S. Wrede*
*- Target version set to 0.5*

**#4 - 10/21/2011 11:54 PM - S. Wrede**

*- Target version changed from 0.5 to rsb-0.7*
*- % Done changed from 0 to 20*

The observed problem is a general one occuring whenever a ReceiverTask is instantiated and the SpreadConnection breaks during a call to its receive method. The currently implemented behavior was to wait a short time and retry the call. Hower, as ReceiverTask object cannot solve this situation on its own this behavior is not reasonable which is why I changed it to print a single warning and interrupt the receiver thread.

That said, it does not fully solve the situation as client threads may still deadlock in a blocking call and are currently not notified or aware of the underlying error. Therefore, we need to implement either an error handling strategy based on callbacks or explicit methods on our top-level objects indicating the failure state of a participant. However, this needs to be considered further. Hence, I propose to consider this for a next release. See #674.

**#5 - 10/22/2011 12:03 AM - J. Moringen**

Actually, the bug report was concerned with failing calls to *.deactivate and how these prevent the remainder of the shutdown from proceeding correctly.

I agree that failing connections have to be handled in more general situations as well, but these two cases (in the context of a call to deactivate vs. during normal operation) should be treated differently, in my opinion.

**#6 - 10/22/2011 12:09 AM - S. Wrede**

Is there an easy testcase to reproduce this?

**#7 - 10/22/2011 12:22 AM - J. Moringen**

The behavior described in the report can be reproduced using the integration test:

1. Obtain a workingcopy
2. Modify java/client.java to call a non-existent method instead of the terminate method
3. run the test with python test/integrationtest.py -s PATHTOSPREAD -t rpc java

This will cause the server (and maybe also the client) to run into a timeout and then the shit hits the fan ;)

**#8 - 11/18/2011 02:54 PM - S. Wrede**

*- Target version changed from rsb-0.7 to 0.6*

**#9 - 02/27/2012 03:30 PM - J. Wienke**

*- Target version deleted (0.6)*

**#10 - 09/02/2013 06:40 PM - J. Wienke**

I just tried to reproduce this in had partial success by modifying the listener example to sleep some time that I can use to kill spread. I still receive an exception at the deactivate call, but cannot see the thread issue anymore.

To me the following questions needs to be addressed: should deactivate always be able to succeed despite any errors or should it report errors? I think we need to decide on this apart from any general error handling strategy.