Java RMI not closing socket after lease expiration

Asked 16 years, 2 months ago Modified 11 years, 2 months ago Viewed 4k times

1

My RMI enabled application seems to be leaking sockets. I have a Java application providing a service over RMI. It is using the Java SE 1.6 RMI implementation running on Linux. The problem I am observing is that if a client obtains a reference to my Remote object, using the Registry, and then the connection is severed abruptly (power loss, cable unplugged, etc...), the server keeps the socket open. I would expect the RMI implementation to clean up after the client's lease expires, but that is not happening. On the server, my Remote object's `unreferenced()` method is called when the lease expires, but the socket remains visible in netstat in the "ESTABLISHED" state indefinitely.

Since we are not able to force the clients into any specific behavior, after several days we are hitting the default limit, 1024 in our Linux distro, for open file descriptors, causing the server to become unable to open any new sockets or files. I thought about TCP keepalives, but since RMI is abstracting away the network layer, I don't have access to the actual server socket after the connection has been established.

Is there any way to force the RMI layer to cleanup sockets tied to client connections with expired leases?

Update: The solution I used is similar to the chosen answer, but uses a different approach. I used a custom socket factory, and then wrapped the ServerSocket instance returned by createServerSocket() in a transparent wrapper that passed all methods through, except for accept(). In the accpet method, keepalives are enabled before the socket is returned.

```
public class KeepaliveSocketWrapper extends ServerSocket
{
    private ServerSocket _delegate = null;
    public KeepaliveSocketWrapper(ServerSocket delegate)
    {
        this._delegate = delegate;
    }

    public Socket accept() throws IOException
    {
        Socket s = _delegate.accept();
        s.setKeepAlive(true);
        return s;
    }
    .
    .
    .
}
```

java sockets rmi

Share

Improve this question

Follow

edited Oct 29, 2008 at 18:03

asked Oct 22, 2008 at 19:48

Peter

13 ● 1 ● 4

Good solution; you should have entered it as an answer so I could vote it up! – erickson Oct 29, 2008 at 18:07

And why have you preferred it to MySocketFactory? What's the advantage for your project? – Vladimir Dyuzhev Nov 6, 2008 at 12:06

You are conflating connection pooling with DGC leases. They aren't the same thing. RMI's TCP connections are closed by the client after a configurable idle timeout which is normally 15 seconds. DGC leases are consudersbly longer, in minutes, but they only affect DGC, not TCP connections. And RMI enables TCP keepalive at both ends by default. So the situation you describe is difficult to understand. Are you sure it is an RMI connection that's leaking? – user207421 Mar 26, 2015 at 17:55

1 Answer

Sorted by: Highest score (default)

0

```
public class MySocketFactorySettingKeepAlive ... {
  // overwrite createSocket methods, call super.createSocket, add keepAlive
}

Socket.setSocketImplFactory(youFactoryDemandingKeepAlive);
```

and then just use RMI.

The trick, if I remember right, is to manage to set the factory before the system opens first socket -- Java disallows to overwrite the factory; in worst case, use reflection to overwrite it :))) (kidding; would be a hack)

Also, you may need to allow specific operation in SecurityManager.

Share

Improve this answer

Follow

edited Oct 23, 2008 at 0:32

answered Oct 22, 2008 at 23:47

Vladimir Dyuzhev

18.3k ● 10 ● 52 ● 62

You can use some system property to specify the factory classname (forgot the name). – ddimitrov Oct 23, 2008 at 0:14

cannot find that property; only SSL and LDAP variants are known to Google – Vladimir Dyuzhev Oct 23, 2008 at 0:36

@ddimtrov No you can't. There is no such property. If you have counter-evidence, provide it. But there are both client and server socket factories per remote object in RMI. – user207421
Sep 29, 2013 at 9:55