Before regression:
svn checkout -r1480731 https://svn.apache.org/repos/asf/activemq/trunk
activemq-broker/src/main/java/org/apache/activemq/broker/region/Queue.java

```
@Override
public void afterCommit() throws Exception {
        LinkedList<Transaction> orderedWork = null;
        // use existing object to sync orderIndexUpdates that can be reassigned
        synchronized (sendLock) {
                if (transaction == orderIndexUpdates.peek()) {
                        orderedWork = orderIndexUpdates;
                        orderIndexUpdates = new LinkedList<Transaction>();

                        // talking all the ordered work means that earlier
                        // and later threads do nothing.
                        // this avoids contention/race on the sendLock that
                        // guards the actual work.
                }
        }
        // do the ordered work
        if (orderedWork != null) {
                sendLock.lockInterruptibly();
                try {
                        for (Transaction tx : orderedWork) {
                                sendSyncs.get(tx).processSend();
                        }
                } finally {
                sendLock.unlock();
        }
        for (Transaction tx : orderedWork) {
                sendSyncs.get(tx).processSent();
        }
        sendSyncs.remove(transaction);
}
}
```

Regressed version:
svn checkout -r1482789 https://svn.apache.org/repos/asf/activemq/trunk
activemq-broker/src/main/java/org/apache/activemq/broker/region/Queue.java

```
@Override
public void afterCommit() throws Exception {
        LinkedList<Transaction> orderedWork = new LinkedList<Transaction>();;
        // use existing object to sync orderIndexUpdates that can be reassigned
        synchronized (sendLock) {
                Transaction next = orderIndexUpdates.peek();
                while( next!=null && next.isCommitted() ) {
                        orderedWork.addLast(orderIndexUpdates.removeFirst());
                        next = orderIndexUpdates.peek();
                }
        }
        // do the ordered work
        if (!orderedWork.isEmpty()) {
                sendLock.lockInterruptibly();
                try {
                        for (Transaction tx : orderedWork) {
                                sendSyncs.get(tx).processSend();
                                sendSyncs.remove(tx);
                        }
                } finally {
                sendLock.unlock();
        }
}
}
```

```
//failed test:
/activemq-ra/target
List : [org.apache.activemq.ra.JmsXARollback2CxTransactionTest.txt,
org.apache.activemq.ra.JmsXAQueueTransactionTest.txt]
/activemq-stomp/target
List : [org.apache.activemq.transport.stomp.StompNIOSSLTest.txt,
org.apache.activemq.transport.stomp.StompNIOTest.txt,
org.apache.activemq.transport.stomp.StompSslAuthTest.txt,
org.apache.activemq.transport.stomp.StompSslTest.txt,
org.apache.activemq.transport.stomp.StompTest.txt]

/activemq-broker/target
List : [org.apache.activemq.JmsQueueTransactionTest.txt]

//More details :
        Classes StompNIOSSLTest, StompNIOTest, StompSslAuthTest, StompSslTest  extends
           StompTest

// failed test in StompTest.java :
        testTransactionCommit:
                ligne 842: assertNotNull("Should have received a message", message);
        testTransactionRollback:
                ligne 876: assertNotNull(message);
        message = null ???
        TextMessage message = (TextMessage)consumer.receive(10000);
        MessageConsumer consumer = session.createConsumer(queue);
        protected ActiveMQQueue queue;
        ActiveMQQueue extends Queue
```

corrected version:
svn checkout -r1482794 https://svn.apache.org/repos/asf/activemq/trunk
/activemq-broker/src/main/java/org/apache/activemq/broker/region/Queue.java

```
@Override
public void afterCommit() throws Exception {
        LinkedList<Transaction> orderedWork = new LinkedList<Transaction>();;
        // use existing object to sync orderIndexUpdates that can be reassigned
        synchronized (sendLock) {
                Transaction next = orderIndexUpdates.peek();
                while( next!=null && next.isCommitted() ) {
                        orderedWork.addLast(orderIndexUpdates.removeFirst());
                        next = orderIndexUpdates.peek();
                }
        }
        // do the ordered work
        if (!orderedWork.isEmpty()) {
                ArrayList<SendSync> syncs = new ArrayList<SendSync>(orderedWork.size());;
                sendLock.lockInterruptibly();
                try {
                        for (Transaction tx : orderedWork) {
                                SendSync sync = sendSyncs.get(tx);
                                sync.processSend();
                                syncs.add(sync);
                                sendSyncs.remove(tx);
                        }
                } finally {
                sendLock.unlock();
        }
        for (SendSync sync : syncs) {
                sync.processSent();
        }
}
}
```

```
//succeeded tests :
/activemq-ra/target
List : [org.apache.activemq.ra.JmsXARollback2CxTransactionTest.txt,
org.apache.activemq.ra.JmsXAQueueTransactionTest.txt]

/activemq-stomp/target
List : [org.apache.activemq.transport.stomp.StompNIOSSLTest.txt,
org.apache.activemq.transport.stomp.StompNIOTest.txt,
org.apache.activemq.transport.stomp.StompSslAuthTest.txt,
org.apache.activemq.transport.stomp.StompSslTest.txt,
org.apache.activemq.transport.stomp.StompTest.txt]

/activemq-broker/target
List : [org.apache.activemq.JmsQueueTransactionTest.txt]
```