Item 231
git_comments:

1. This is called from the initTransactions method in the producer as the first order of business. It finds the coordinator and then gets a PID.
2. Send AddOffsetsToTxnRequest
3. Send AddPartitionsRequest
4. find coordinator
5. get pid.

git_commits:

1. **summary:** KAFKA-5260; Producer should not send AbortTxn unless transaction has actually begun
   message: KAFKA-5260; Producer should not send AbortTxn unless transaction has actually begun Keep track of when a transaction has begun by setting a flag, `transactionStarted` when a successfull `AddPartitionsToTxnResponse` or `AddOffsetsToTxnResponse` had been received. If an `AbortTxnRequest` about to be sent and `transactionStarted` is false, don't send the request and transition the state to `READY` Author: Damian Guy <damian.guy@gmail.com> Reviewers: Apurva Mehta <apurva@confluent.io>, Guozhang Wang <wangguoz@gmail.com>, Jason Gustafson <jason@confluent.io> Closes #3126 from dguy/kafka-5260 (cherry picked from commit a8794d8a5d18bb4eaafceac1ef675243af945862) Signed-off-by: Jason Gustafson <jason@confluent.io>

github_issues:

github_issues_comments:

github_pulls:

1. **title:** KAFKA-5260: Producer should not send AbortTxn unless transaction has actually begun
   body: Keep track of when a transaction has begun by setting a flag, `transactionStarted` when a successfull `AddPartitionsToTxnResponse` or `AddOffsetsToTxnResponse` had been received. If an `AbortTxnRequest` about to be sent and `transactionStarted` is false, don't send the request and transition the state to `READY`
2. **title:** KAFKA-5260: Producer should not send AbortTxn unless transaction has actually begun
   body: Keep track of when a transaction has begun by setting a flag, `transactionStarted` when a successfull `AddPartitionsToTxnResponse` or `AddOffsetsToTxnResponse` had been received. If an `AbortTxnRequest` about to be sent and `transactionStarted` is false, don't send the request and transition the state to `READY`
3. **title:** KAFKA-5260: Producer should not send AbortTxn unless transaction has actually begun
   body: Keep track of when a transaction has begun by setting a flag, `transactionStarted` when a successfull `AddPartitionsToTxnResponse` or `AddOffsetsToTxnResponse` had been received. If an `AbortTxnRequest` about to be sent and `transactionStarted` is false, don't send the request and transition the state to `READY`
4. **title:** KAFKA-5260: Producer should not send AbortTxn unless transaction has actually begun
   body: Keep track of when a transaction has begun by setting a flag, `transactionStarted` when a successfull `AddPartitionsToTxnResponse` or `AddOffsetsToTxnResponse` had been received. If an `AbortTxnRequest` about to be sent and `transactionStarted` is false, don't send the request and transition the state to `READY`
5. **title:** KAFKA-5260: Producer should not send AbortTxn unless transaction has actually begun
   body: Keep track of when a transaction has begun by setting a flag, `transactionStarted` when a successfull `AddPartitionsToTxnResponse` or `AddOffsetsToTxnResponse` had been received. If an `AbortTxnRequest` about to be sent and `transactionStarted` is false, don't send the request and transition the state to `READY`
6. **title:** KAFKA-5260: Producer should not send AbortTxn unless transaction has actually begun
   body: Keep track of when a transaction has begun by setting a flag, `transactionStarted` when a successfull `AddPartitionsToTxnResponse` or `AddOffsetsToTxnResponse` had been received. If an `AbortTxnRequest` about to be sent and `transactionStarted` is false, don't send the request and transition the state to `READY`
   label: code-design

7. **title:** KAFKA-5260: Producer should not send AbortTxn unless transaction has actually begun
   body: Keep track of when a transaction has begun by setting a flag, `transactionStarted` when a successfull `AddPartitionsToTxnResponse` or `AddOffsetsToTxnResponse` had been received. If an `AbortTxnRequest` about to be sent and `transactionStarted` is false, don't send the request and transition the state to `READY`
8. **title:** KAFKA-5260: Producer should not send AbortTxn unless transaction has actually begun
   body: Keep track of when a transaction has begun by setting a flag, `transactionStarted` when a successfull `AddPartitionsToTxnResponse` or `AddOffsetsToTxnResponse` had been received. If an `AbortTxnRequest` about to be sent and `transactionStarted` is false, don't send the request and transition the state to `READY`
9. **title:** KAFKA-5260: Producer should not send AbortTxn unless transaction has actually begun
   body: Keep track of when a transaction has begun by setting a flag, `transactionStarted` when a successfull `AddPartitionsToTxnResponse` or `AddOffsetsToTxnResponse` had been received. If an `AbortTxnRequest` about to be sent and `transactionStarted` is false, don't send the request and transition the state to `READY`
10. **title:** KAFKA-5260: Producer should not send AbortTxn unless transaction has actually begun
    body: Keep track of when a transaction has begun by setting a flag, `transactionStarted` when a successfull `AddPartitionsToTxnResponse` or `AddOffsetsToTxnResponse` had been received. If an `AbortTxnRequest` about to be sent and `transactionStarted` is false, don't send the request and transition the state to `READY`
    label: code-design
11. **title:** KAFKA-5260: Producer should not send AbortTxn unless transaction has actually begun
    body: Keep track of when a transaction has begun by setting a flag, `transactionStarted` when a successfull `AddPartitionsToTxnResponse` or `AddOffsetsToTxnResponse` had been received. If an `AbortTxnRequest` about to be sent and `transactionStarted` is false, don't send the request and transition the state to `READY`
12. **title:** KAFKA-5260: Producer should not send AbortTxn unless transaction has actually begun
    body: Keep track of when a transaction has begun by setting a flag, `transactionStarted` when a successfull `AddPartitionsToTxnResponse` or `AddOffsetsToTxnResponse` had been received. If an `AbortTxnRequest` about to be sent and `transactionStarted` is false, don't send the request and transition the state to `READY`
13. **title:** KAFKA-5260: Producer should not send AbortTxn unless transaction has actually begun
    body: Keep track of when a transaction has begun by setting a flag, `transactionStarted` when a successfull `AddPartitionsToTxnResponse` or `AddOffsetsToTxnResponse` had been received. If an `AbortTxnRequest` about to be sent and `transactionStarted` is false, don't send the request and transition the state to `READY`
14. **title:** KAFKA-5260: Producer should not send AbortTxn unless transaction has actually begun
    body: Keep track of when a transaction has begun by setting a flag, `transactionStarted` when a successfull `AddPartitionsToTxnResponse` or `AddOffsetsToTxnResponse` had been received. If an `AbortTxnRequest` about to be sent and `transactionStarted` is false, don't send the request and transition the state to `READY`
15. **title:** KAFKA-5260: Producer should not send AbortTxn unless transaction has actually begun
    body: Keep track of when a transaction has begun by setting a flag, `transactionStarted` when a successfull `AddPartitionsToTxnResponse` or `AddOffsetsToTxnResponse` had been received. If an `AbortTxnRequest` about to be sent and `transactionStarted` is false, don't send the request and transition the state to `READY`
16. **title:** KAFKA-5260: Producer should not send AbortTxn unless transaction has actually begun
    body: Keep track of when a transaction has begun by setting a flag, `transactionStarted` when a successfull `AddPartitionsToTxnResponse` or `AddOffsetsToTxnResponse` had been received. If an `AbortTxnRequest` about to be sent and `transactionStarted` is false, don't send the request and transition the state to `READY`
17. **title:** KAFKA-5260: Producer should not send AbortTxn unless transaction has actually begun
    body: Keep track of when a transaction has begun by setting a flag, `transactionStarted` when a successfull `AddPartitionsToTxnResponse` or `AddOffsetsToTxnResponse` had been received. If an `AbortTxnRequest` about to be sent and `transactionStarted` is false, don't send the request and transition the state to `READY`
18. **title:** KAFKA-5260: Producer should not send AbortTxn unless transaction has actually begun
    body: Keep track of when a transaction has begun by setting a flag, `transactionStarted` when a successfull `AddPartitionsToTxnResponse` or `AddOffsetsToTxnResponse` had been received. If an

`AbortTxnRequest` about to be sent and `transactionStarted` is false, don't send the request and transition the state to `READY`
19. **title:** KAFKA-5260: Producer should not send AbortTxn unless transaction has actually begun
    body: Keep track of when a transaction has begun by setting a flag, `transactionStarted` when a successfull `AddPartitionsToTxnResponse` or `AddOffsetsToTxnResponse` had been received. If an `AbortTxnRequest` about to be sent and `transactionStarted` is false, don't send the request and transition the state to `READY`
20. **title:** KAFKA-5260: Producer should not send AbortTxn unless transaction has actually begun
    body: Keep track of when a transaction has begun by setting a flag, `transactionStarted` when a successfull `AddPartitionsToTxnResponse` or `AddOffsetsToTxnResponse` had been received. If an `AbortTxnRequest` about to be sent and `transactionStarted` is false, don't send the request and transition the state to `READY`
21. **title:** KAFKA-5260: Producer should not send AbortTxn unless transaction has actually begun
    body: Keep track of when a transaction has begun by setting a flag, `transactionStarted` when a successfull `AddPartitionsToTxnResponse` or `AddOffsetsToTxnResponse` had been received. If an `AbortTxnRequest` about to be sent and `transactionStarted` is false, don't send the request and transition the state to `READY`
22. **title:** KAFKA-5260: Producer should not send AbortTxn unless transaction has actually begun
    body: Keep track of when a transaction has begun by setting a flag, `transactionStarted` when a successfull `AddPartitionsToTxnResponse` or `AddOffsetsToTxnResponse` had been received. If an `AbortTxnRequest` about to be sent and `transactionStarted` is false, don't send the request and transition the state to `READY`
    label: code-design
23. **title:** KAFKA-5260: Producer should not send AbortTxn unless transaction has actually begun
    body: Keep track of when a transaction has begun by setting a flag, `transactionStarted` when a successfull `AddPartitionsToTxnResponse` or `AddOffsetsToTxnResponse` had been received. If an `AbortTxnRequest` about to be sent and `transactionStarted` is false, don't send the request and transition the state to `READY`

github_pulls_comments:

1. @apurvam @hachikuji - This seems to work, though i'm not entirely sure it is correct!
2. Refer to this link for build results (access rights to CI server needed): https://builds.apache.org/job/kafka-pr-jdk7-scala2.11/4304/ Test FAILed (JDK 7 and Scala 2.11).
3. Refer to this link for build results (access rights to CI server needed): https://builds.apache.org/job/kafka-pr-jdk8-scala2.12/4291/ Test FAILed (JDK 8 and Scala 2.12).
4. Refer to this link for build results (access rights to CI server needed): https://builds.apache.org/job/kafka-pr-jdk7-scala2.11/4306/ Test PASSed (JDK 7 and Scala 2.11).
5. Refer to this link for build results (access rights to CI server needed): https://builds.apache.org/job/kafka-pr-jdk8-scala2.12/4293/ Test PASSed (JDK 8 and Scala 2.12).
6. **body:** @apurvam i did start adding another state as you've suggested, though it seemed to be getting pretty messy due to having to check if the currentState is either `STARTED_TRANSACTION` or `IN_TRANSACTION` in multiple places. I'll have another look, but it wasn't really filling me with joy hence the boolean
   label: code-design
7. @apurvam i don't think this works by adding another state. The state transitions are happening on the Producers thread, yet we don't know we have successfully completed a transactional request until we get an `AddPartitionsToTxnResponse` or an `AddOffsetsToPartitionResponse`, which is happening on the sender thread. In the meantime the current state may have transitioned to subsequent states _before_ we get these responses. So when it comes to the time when we need to decide if we should send the `EndTxnRequest` we don't have any idea if the transaction was previously in the `STARTING_TRANSACTION` state. I thought adding a `previousState` might help, but no. As there may have been multiple transitions
8. Refer to this link for build results (access rights to CI server needed): https://builds.apache.org/job/kafka-pr-jdk7-scala2.11/4385/ Test PASSed (JDK 7 and Scala 2.11).
9. Refer to this link for build results (access rights to CI server needed): https://builds.apache.org/job/kafka-pr-jdk8-scala2.12/4371/ Test PASSed (JDK 8 and Scala 2.12).
10. **body:** @dguy I started typing a really long response, but deleted it. I guess the key point is that if we never transitioned from `STARTING_TRANSACTION` to `IN_TRANSACTION` at the time of commit

or abort, it means that either there are no `send` or `sendOffset` calls attempted, or they have failed. In the former, the additional state will do what we want: if we are committing or aborting in this state, we know there is nothing to do. HOwever, if the `AddPartitions` or `AddOffsets` RPCs failed fatally, or have never been successfully acknowledged, we would --correctly-- be in an error state, and hence have no record of whether we ever got out of `STARTING_TRANSACTION` to `IN_TRANSACTION`. So either we hack the transitions to the error state, or just maintain the separate variable. I think the separate variable is preferable.
label: code-design

11. Refer to this link for build results (access rights to CI server needed): https://builds.apache.org/job/kafka-pr-jdk7-scala2.11/4444/ Test PASSed (JDK 7 and Scala 2.11).
12. Refer to this link for build results (access rights to CI server needed): https://builds.apache.org/job/kafka-pr-jdk8-scala2.12/4430/ Test PASSed (JDK 8 and Scala 2.12).
13. Actually, it seems that currently `FATAL_ERROR` and `ABORTABLE_ERROR` states are not distinguished. I'm wondering if we can just use `FATAL_ERROR` state and in that state do not send abortTxn?
14. Refer to this link for build results (access rights to CI server needed): https://builds.apache.org/job/kafka-pr-jdk7-scala2.11/4479/ Test PASSed (JDK 7 and Scala 2.11).
15. Refer to this link for build results (access rights to CI server needed): https://builds.apache.org/job/kafka-pr-jdk8-scala2.12/4465/ Test PASSed (JDK 8 and Scala 2.12).
16. @guozhangwang The difference between `FATAL_ERROR` and `ABORTABLE_ERROR` is that the former state will throw an exception back at the user even if they try to call `abortTransaction()`. The only possible recovery is closing the producer (see related issue https://issues.apache.org/jira/browse/KAFKA-5342). In some cases, we may reach an abortable error before any partition has been successfully added to the transaction (this is unfortunately the consequence of not having a BeginTxn request). We want the user to call `abortTransaction()` in this case, but we want to avoid sending EndTxn since this would cause an invalid transaction state. I've been thinking about the current patch for a while and finally it LGTM. The subtle thing is the fact that we'll keep retrying AddPartition failures as long as we get retriable errors. If we receive an abortable error, we'll stop retrying and the EndTxn should reach the head of the queue. Luckily we made the AddPartitions request have all or nothing semantics, so any failure would mean we haven't started the transaction (assuming there were no previous successes).

github_pulls_reviews:

1. shall we also assert that the result is successful?
2. Should we also add a test case where there is a real authorization failure and yet the transaction can be aborted?
3. By authorization failure do you mean topic authorization?
4. Was this required to get your authorization integration test to work? If so, why?
5. **body:** Oops, that was me debugging. Will remove it
label: code-design

jira_issues:

1. **summary:** Producer should not send AbortTxn unless transaction has actually begun
description: When there is an authorization error in AddOffsets or AddPartitions, the producer will raise an authorization exception. When that happens, the user should abort the transaction. The problem is that in an authorization error, the coordinator will not have transitioned to a new state, so if it suddenly receives an AbortTxnRequest, that request will fail with an InvalidTxnState, which will be propagated to the error. The suggested solution is to keep track locally when we are certain that no transaction has been officially begun and to skip sending the AbortTxnRequest in that case.
2. **summary:** Producer should not send AbortTxn unless transaction has actually begun
description: When there is an authorization error in AddOffsets or AddPartitions, the producer will raise an authorization exception. When that happens, the user should abort the transaction. The problem is that in an authorization error, the coordinator will not have transitioned to a new state, so if it suddenly receives an AbortTxnRequest, that request will fail with an InvalidTxnState, which will be propagated to the error. The suggested solution is to keep track locally when we are certain that no transaction has been officially begun and to skip sending the AbortTxnRequest in that case.
3. **summary:** Producer should not send AbortTxn unless transaction has actually begun

description: When there is an authorization error in AddOffsets or AddPartitions, the producer will raise an authorization exception. When that happens, the user should abort the transaction. The problem is that in an authorization error, the coordinator will not have transitioned to a new state, so if it suddenly receives an AbortTxnRequest, that request will fail with an InvalidTxnState, which will be propagated to the error. The suggested solution is to keep track locally when we are certain that no transaction has been officially begun and to skip sending the AbortTxnRequest in that case.

jira_issues_comments:

1. GitHub user dguy opened a pull request: https://github.com/apache/kafka/pull/3126 KAFKA-5260: Producer should not send AbortTxn unless transaction has actually begun Keep track of when a transaction has begun by setting a flag, `transactionStarted` when a successfull `AddPartitionsToTxnResponse` or `AddOffsetsToTxnResponse` had been received. If an `AbortTxnRequest` about to be sent and `transactionStarted` is false, don't send the request and transition the state to `READY` You can merge this pull request into a Git repository by running: $ git pull https://github.com/dguy/kafka kafka-5260 Alternatively you can review and apply these changes as the patch at: https://github.com/apache/kafka/pull/3126.patch To close this pull request, make a commit to your master/trunk branch with (at least) the following in the commit message: This closes #3126 ---- commit e8e86e764ddc24d193bcaf0c2ba440a92d0a2534 Author: Damian Guy <damian.guy@gmail.com> Date: 2017-05-23T11:33:23Z don't send AbortTxn unless a transaction is Ongoing ----
2. Github user asfgit closed the pull request at: https://github.com/apache/kafka/pull/3126
3. Issue resolved by pull request 3126 [https://github.com/apache/kafka/pull/3126]