Item 3
git_comments:

1. dummy verb so we can use MS.droppedMessagesMap

git_commits:

1. **summary:** Log when messages are dropped due to cross_node_timeout (CASSANDRA-9793)
   message: Log when messages are dropped due to cross_node_timeout (CASSANDRA-9793) Patch by Stefania, reviewed by brandonwilliams for CASSANDRA-9793

github_issues:

github_issues_comments:

github_pulls:

github_pulls_comments:

github_pulls_reviews:

jira_issues:

1. **summary:** Log when messages are dropped due to cross_node_timeout
   description: When a node has clock skew and cross node timeouts are enabled, there's no indication that the messages were dropped due to the cross timeout, just that messages were dropped. This can errantly lead you down a path of troubleshooting a load shedding situation when really you just have clock drift on one node. This is also not simple to troubleshoot, since you have to determine that this node will answer requests, but other nodes won't answer requests from it. If the problem goes away on a reboot (and the machine does one-shot time sync, not continuous) it becomes even harder to detect because you're left with a weird piece of evidence such as "it's fine after a reboot, but comes back in about X days every time." It would help tremendously if there were a log message indicating how many messages (don't need them broken down by type) were eagerly dropped due to the cross node timeout.

jira_issues_comments:

1. **body:** [~brandon.williams] when we receive a message in IncomingTcpConnection.receiveMessage, if cross_node_timeout is enabled we change the timestamp as follows: {code} long timestamp = System.currentTimeMillis(); // make sure to readInt, even if cross_node_to is not enabled int partial = input.readInt(); if (DatabaseDescriptor.hasCrossNodeTimeout()) timestamp = (timestamp & 0xFFFFFFFF00000000L) | (((partial & 0xFFFFFFFFL) << 2) >> 2); {code} The timestamp becomes the constructionTime in MessageDeliveryTask, where we drop the message: {code} if (MessagingService.DROPPABLE_VERBS.contains(verb) && System.currentTimeMillis() > constructionTime + message.getTimeout()) { MessagingService.instance().incrementDroppedMessages(verb); return; } {code} So if cross_node_timeout is set, technically we always drop due to the cross_node_timeout. Are you proposing to add one more timestamp in MessageDeliveryTask, a real construction time that is never influenced by cross_node_timeout, and to drop depending on either timestamp indicating when it is the cross node timeout timestamp that caused the messages to be dropped? Or do you simply want add the information on whether cross node timeout is enabled to the existing log message in MessagingService: {code} private void logDroppedMessages() { boolean logTpstats = false; for (Map.Entry<Verb, DroppedMessageMetrics> entry : droppedMessages.entrySet()) { int dropped = (int) entry.getValue().dropped.count(); Verb verb = entry.getKey(); int recent = dropped - lastDroppedInternal.get(verb); if (recent > 0) { logTpstats = true; - logger.info("{} {} messages dropped in last {}ms", - new Object[] {recent, verb, LOG_DROPPED_INTERVAL_IN_MS}); + logger.info("{} {} messages dropped in last {} ms, cross node timeout is {}", + recent, + verb, + LOG_DROPPED_INTERVAL_IN_MS, + DatabaseDescriptor.hasCrossNodeTimeout() ? "set" : "not set"); lastDroppedInternal.put(verb, dropped); } } [...] {code}
   label: code-design

2. I want the former solution, the latter is easily derived from the yaml. bq. Are you proposing to add one more timestamp in MessageDeliveryTask, a real construction time that is never influenced by cross_node_timeout It seems that a boolean of whether we modified the timestamp or not would be enough.
3. You're quite right a boolean should be enough, resuming development.
4. This is ready for review, CI is pending: http://cassci.datastax.com/view/Dev/view/stef1927/job/stef1927-9793-2.0-testall/lastBuild/ http://cassci.datastax.com/view/Dev/view/stef1927/job/stef1927-9793-2.0-dtest/lastBuild/
5. [~brandon.williams] are you happy to review or should I find someone else?
6. I will do it.
7. Can you rebase for 2.2? I'm getting a lot of conflicts there.
8. Also it looks like we won't log tpstats on drop anymore, unless I'm mistaken.
9. Attached [2.1|https://github.com/stef1927/cassandra/commits/9793-2.1] and [2.2|https://github.com/stef1927/cassandra/commits/9793-2.2] patches and verified that the 2.2 patch applies to trunk. We still [log tcpstats on drop|https://github.com/stef1927/cassandra/commit/a00754d4dddb47bc4a4865131f282eb34fe8680b#diff-af09288f448c37a525e831ee90ea49f9L851] in 2.2.
10. Committed, thanks!
11. The logging level for dropped messages was set to ERROR by this patch. I've pushed a follow-up commit to lower it to INFO as {{4d1b8b418609bd2c431a0ea905a667ddaaa50bbe}}. (The error-level logging was also causing a lot of dtests to fail.)