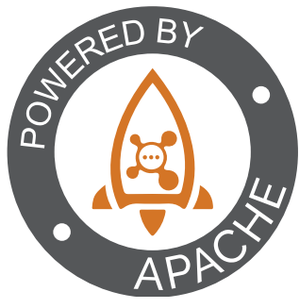


Message Filter and Retry In RocketMQ

费红健
erikfei1288@gmail.com

2019-01-24
Hangzhou

About me

- Middleware Developer/Architect
- 微信公号：“艾瑞克的技术江湖”
- 简书：https://www.jianshu.com/u/81e2cd2747f1

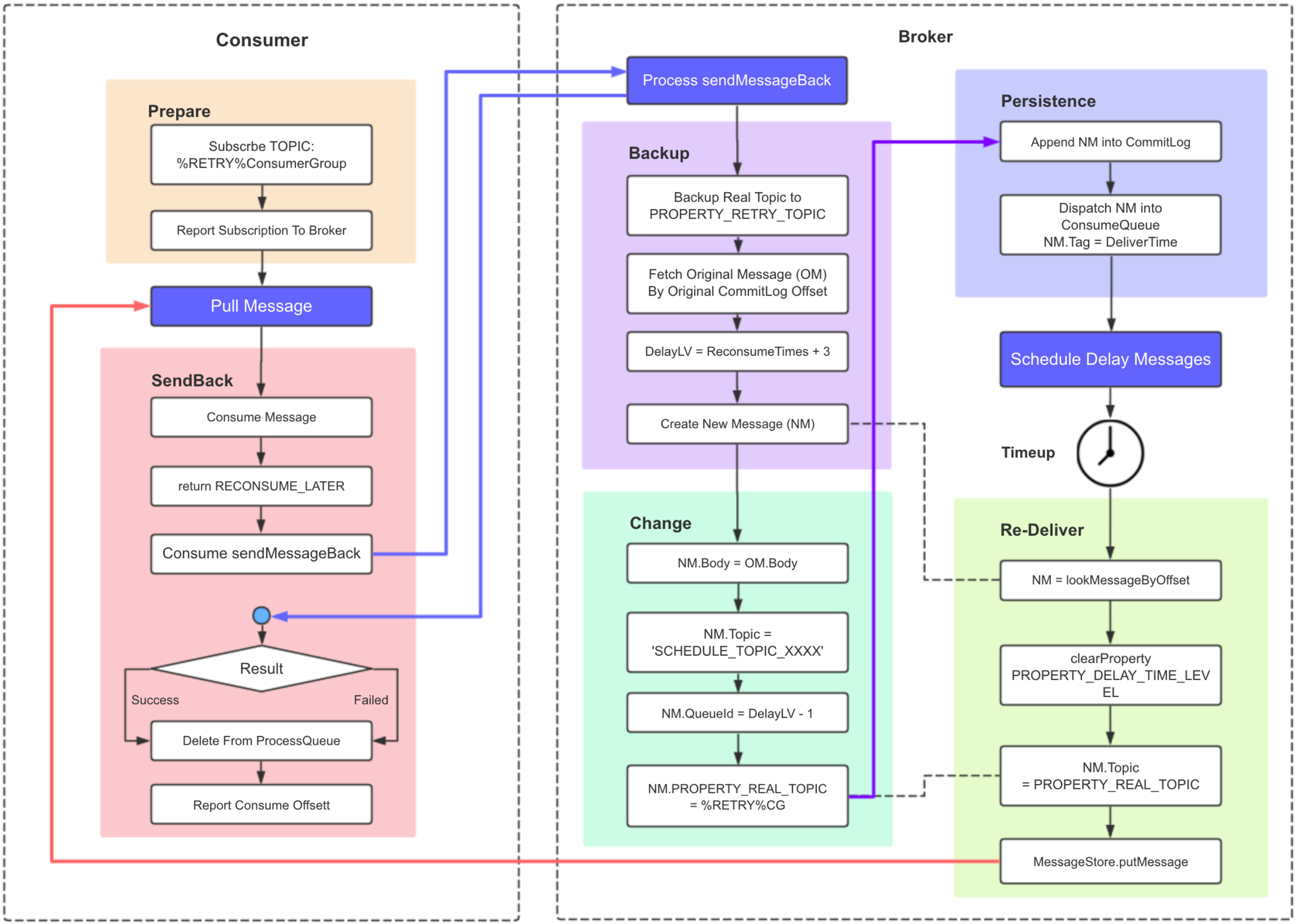
Agenda

- Overview
- Message Retry
- Message Filter

Message Retry

FAQ

1. What is going to happen when consumer listener returns “RECONSUME_LATER”?


Consumer
Prepare
Subscrbe TOPIC:
%RETRY%ConsumerGroup
Report Subscription To Broker
Pull Message
SendBack
Consume Message
return RECONSUME_LATER
Consume sendMessageBack
Result
Success
Failed
Delete From ProcessQueue
Report Consume Offsett
Broker
Process sendMessageBack
Backup
Backup Real Topic to
PROPERTY_RETRY_TOPIC
Fetch Original Message (OM)
By Original CommitLog Offset
DelayLV = ReconsumeTimes + 3
Create New Message (NM)
Change
NM.Body = OM.Body
NM.Topic =
'SCHEDULE_TOPIC_XXXX'
NM.QueueId = DelayLV - 1
NM.PROPERTY_REAL_TOPIC
= %RETRY%CG
Persistence
Append NM into CommitLog
Dispatch NM into
ConsumeQueue
NM.Tag = DeliverTime
Schedule Delay Messages
Timeup
Re-Deliver
NM = lookMessageByOffset
clearProperty
PROPERTY_DELAY_TIME_LEV
EL
NM.Topic
= PROPERTY_REAL_TOPIC
MessageStore.putMessage

Message Retry Code Time

```
private void copySubscription() throws MQClientException {
    try {
        Map<String, String> sub = this.defaultMQPushConsumer.getSubscription();
        if (sub != null) {
            for (final Map.Entry<String, String> entry : sub.entrySet()) {
                final String topic = entry.getKey();
                final String subString = entry.getValue();
                SubscriptionData subscriptionData = FilterAPI.buildSubscriptionData(this.defaultMQPushConsumer.ge
                    topic, subString);
                this.rebalanceImpl.getSubscriptionInner().put(topic, subscriptionData);
            }
        }

        if (null == this.messageListenerInner) {
            this.messageListenerInner = this.defaultMQPushConsumer.getMessageListener();
        }

        switch (this.defaultMQPushConsumer.getMessageModel()) {
            case BROADCASTING:
                break;
            case CLUSTERING:
                final String retryTopic = MixAll.getRetryTopic(this.defaultMQPushConsumer.getConsumerGroup());
                SubscriptionData subscriptionData = FilterAPI.buildSubscriptionData(this.defaultMQPushConsumer.ge
                    retryTopic, SubscriptionData.SUB_ALL);
                this.rebalanceImpl.getSubscriptionInner().put(retryTopic, subscriptionData);
                break;
            default:
                break;
        }
    } catch (Exception e) {
        throw new MQClientException("subscription exception", e);
    }
}
```

Message Retry Code Time

```
switch (this.defaultMQPushConsumer.getMessageModel()) {
    case BROADCASTING:
        for (int i = ackIndex + 1; i < consumeRequest.getMsgs().size(); i++) {
            MessageExt msg = consumeRequest.getMsgs().get(i);
            log.warn("BROADCASTING, the message consume failed, drop it, {}", msg.toString());
        }
        break;
    case CLUSTERING:
        List<MessageExt> msgBackFailed = new ArrayList<MessageExt>(consumeRequest.getMsgs().size());
        for (int i = ackIndex + 1; i < consumeRequest.getMsgs().size(); i++) {
            MessageExt msg = consumeRequest.getMsgs().get(i);
            boolean result = this.sendMessageBack(msg, context);
            if (!result) {
                msg.setReconsumeTimes(msg.getReconsumeTimes() + 1);
                msgBackFailed.add(msg);
            }
        }

        if (!msgBackFailed.isEmpty()) {
            consumeRequest.getMsgs().removeAll(msgBackFailed);

            this.submitConsumeRequestLater(msgBackFailed, consumeRequest.getProcessQueue(), consumeRequest.getMes
        }
        break;
    default:
        break;
}
```

Message Retry Code Time

```
final String retryTopic = msgExt.getProperty(MessageConst.PROPERTY_RETRY_TOPIC);
if (null == retryTopic) {
    MessageAccessor.putProperty(msgExt, MessageConst.PROPERTY_RETRY_TOPIC, msgExt.getTopic());
}
msgExt.setWaitStoreMsgOK(false);

int delayLevel = requestHeader.getDelayLevel();

int maxReconsumeTimes = subscriptionGroupConfig.getRetryMaxTimes();
if (request.getVersion() >= MQVersion.Version.V3_4_9.ordinal()) {
    maxReconsumeTimes = requestHeader.getMaxReconsumeTimes();
}

if (msgExt.getReconsumeTimes() >= maxReconsumeTimes
    || delayLevel < 0) {
    newTopic = MixAll.getDLQTopic(requestHeader.getGroup());
    queueIdInt = Math.abs(this.random.nextInt() % 99999999) % DLQ_NUMS_PER_GROUP;

    topicConfig = this.brokerController.getTopicConfigManager().createTopicInSendMessageBackMethod(newTopic,
        DLQ_NUMS_PER_GROUP,
        PermName.PERM_WRITE, topicSysFlag: 0
    );
    if (null == topicConfig) {
        response.setCode(ResponseCode.SYSTEM_ERROR);
        response.setRemark("topic[" + newTopic + "] not exist");
        return response;
    }
} else {
    if (0 == delayLevel) {
        delayLevel = 3 + msgExt.getReconsumeTimes();
    }

    msgExt.setDelayTimeLevel(delayLevel);
```

DLQ

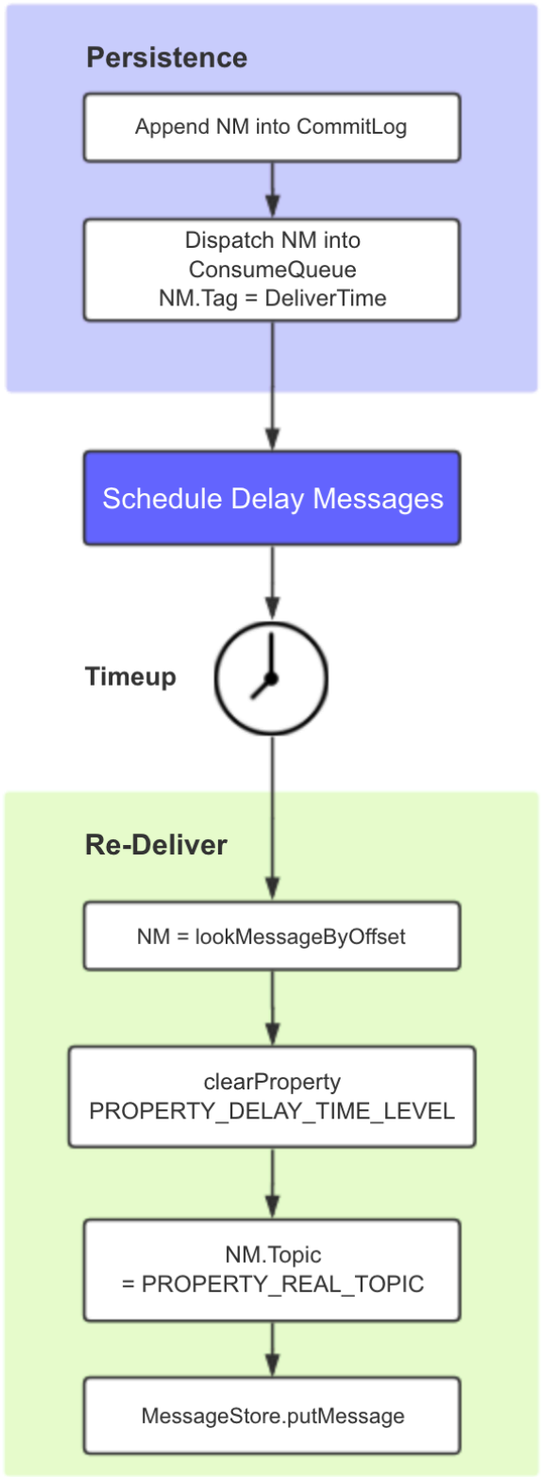
Message Retry Code Time

```
final int tranType = MessageSysFlag.getTransactionValue(msg.getSysFlag());
if (tranType == MessageSysFlag.TRANSACTION_NOT_TYPE
    || tranType == MessageSysFlag.TRANSACTION_COMMIT_TYPE) {
    // Delay Delivery
    if (msg.getDelayTimeLevel() > 0) {
        if (msg.getDelayTimeLevel() > this.defaultMessageStore.getScheduleMessageService().getMaxDelayLevel()) {
            msg.setDelayTimeLevel(this.defaultMessageStore.getScheduleMessageService().getMaxDelayLevel());
        }

        topic = ScheduleMessageService.SCHEDULE_TOPIC;
        queueId = ScheduleMessageService.delayLevel2QueueId(msg.getDelayTimeLevel());
        System.out.println("delay level :" + msg.getDelayTimeLevel());

        // Backup real topic, queueId
        MessageAccessor.putProperty(msg, MessageConst.PROPERTY_REAL_TOPIC, msg.getTopic());
        MessageAccessor.putProperty(msg, MessageConst.PROPERTY_REAL_QUEUE_ID, String.valueOf(msg.getQueueId()));
        msg.setPropertiesString(MessageDecoder.messageProperties2String(msg.getProperties()));

        msg.setTopic(topic);
        msg.setQueueId(queueId);
    }
}
```

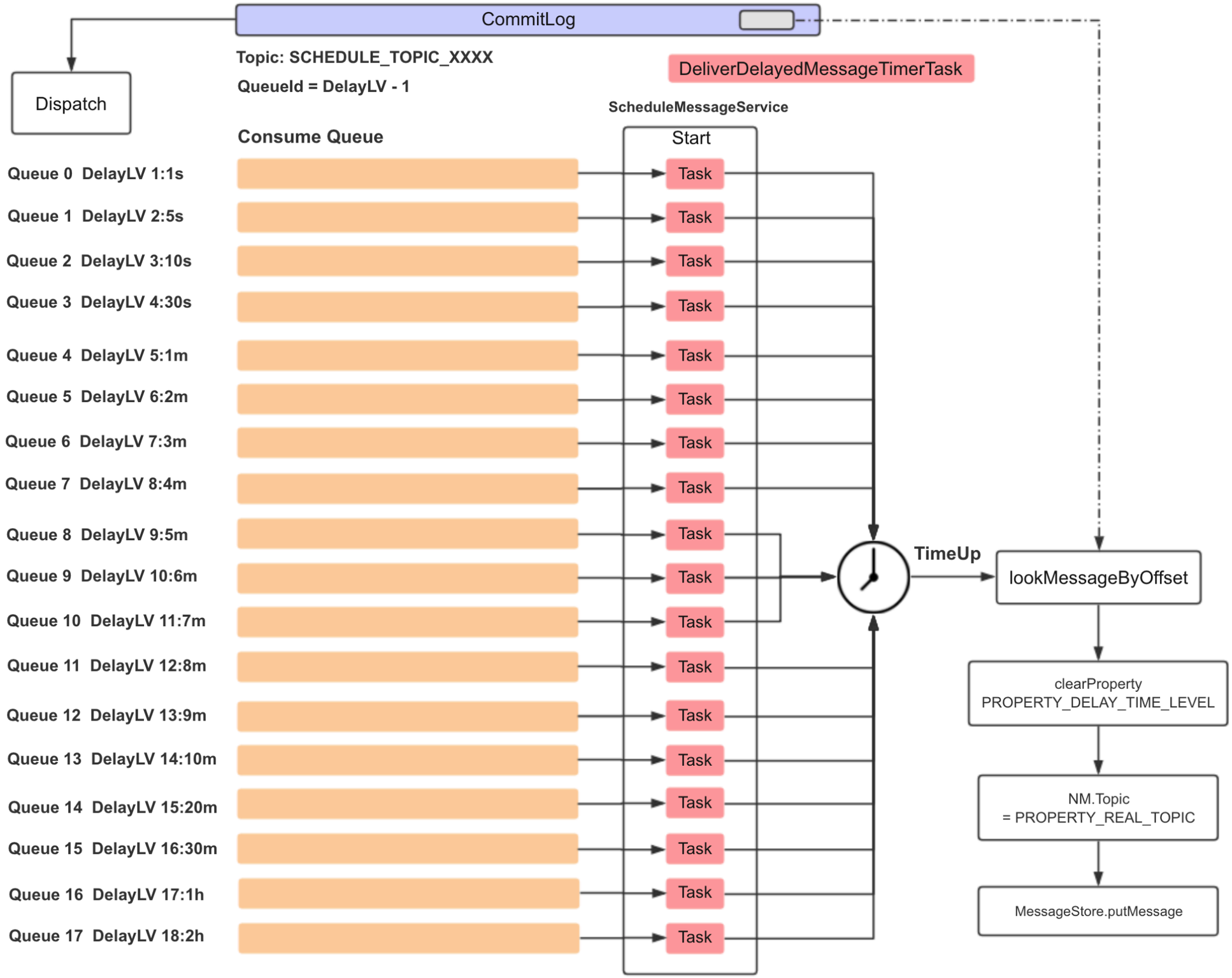
Message Retry Code Time

```
// Timing message processing
{
    String t = propertiesMap.get(MessageConst.PROPERTY_DELAY_TIME_LEVEL);
    if (ScheduleMessageService.SCHEDULE_TOPIC.equals(topic) && t != null) {
        int delayLevel = Integer.parseInt(t);

        if (delayLevel > this.defaultMessageStore.getScheduleMessageService().getMaxDelayLevel()) {
            delayLevel = this.defaultMessageStore.getScheduleMessageService().getMaxDelayLevel();
        }

        if (delayLevel > 0) {
            tagsCode = this.defaultMessageStore.getScheduleMessageService().computeDeliverTimestamp(delayLevel,
                storeTimestamp);
        }
    }
}
```

Overview



Detail



Message Retry Code Time

```
eric at EricdeMacBook-Pro in ~/store/config
$ ll
total 72
-rw-r--r--  1 eric  staff    27B Jan 23 16:00 consumerFilter.json
-rw-r--r--  1 eric  staff    27B Jan 23 16:00 consumerFilter.json.bak
-rw-r--r--  1 eric  staff   103B Jan 23 16:00 consumerOffset.json
-rw-r--r--  1 eric  staff   103B Jan 23 16:00 consumerOffset.json.bak
-rw-r--r--  1 eric  staff    60B Jan 23 16:00 delayOffset.json
-rw-r--r--  1 eric  staff    55B Jan 23 16:00 delayOffset.json.bak
-rw-r--r--  1 eric  staff   2.4K Jan 23 15:59 subscriptionGroup.json
-rw-r--r--  1 eric  staff   1.8K Jan 23 16:00 topics.json
-rw-r--r--  1 eric  staff   1.6K Jan 23 16:00 topics.json.bak
```

Message Retry Code Time

```java
public void start() {

    for (Map.Entry<Integer, Long> entry : this.delayLevelTable.entrySet()) {
        Integer level = entry.getKey();
        Long timeDelay = entry.getValue();
        Long offset = this.offsetTable.get(level);
        if (null == offset) {
            offset = 0L;
        }

        if (timeDelay != null) {
            this.timer.schedule(new DeliverDelayedMessageTimerTask(level, offset), FIRST_DELAY_TIME);
        }
    }

    this.timer.scheduleAtFixedRate(() -> {
            try {
                ScheduleMessageService.this.persist();
            } catch (Throwable e) {
                log.error("scheduleAtFixedRate flush exception", e);
            }
    }, delay: 10000, this.defaultMessageStore.getMessageStoreConfig().getFlushDelayOffsetInterval());
}
```

Message Retry Code Time

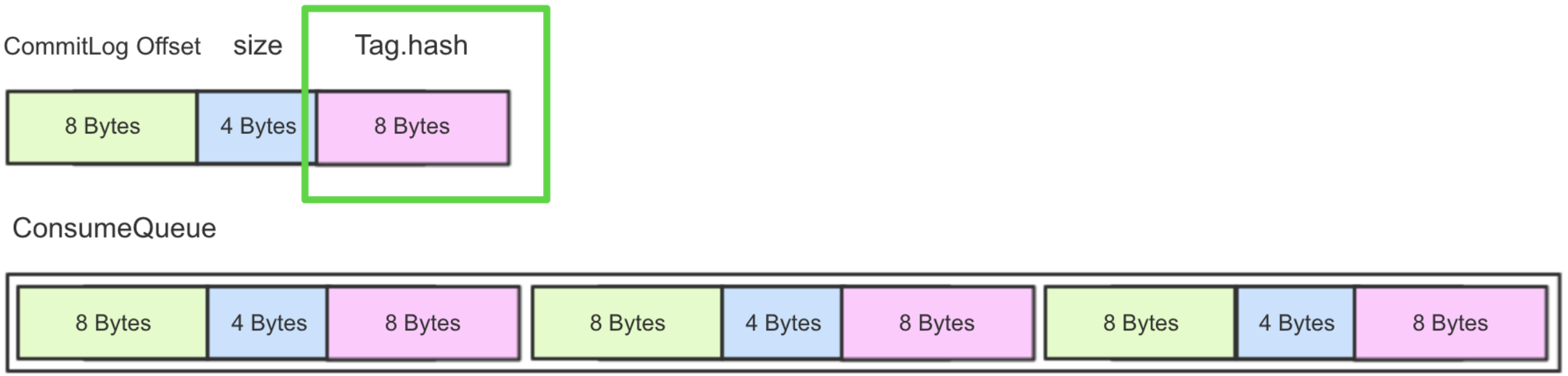
```
if (countdown <= 0) {
    MessageExt msgExt =
        ScheduleMessageService.this.defaultMessageStore.lookMessageByOffset(
            offsetPy, sizePy);

    if (msgExt != null) {    Need to re-deliver now
        try {
            MessageExtBrokerInner msgInner = this.messageTimeup(msgExt);
            PutMessageResult putMessageResult =
                ScheduleMessageService.this.defaultMessageStore
                    .putMessage(msgInner);

            if (putMessageResult != null
                && putMessageResult.getPutMessageStatus() == PutMessageStatus.PUT_OK) {
                continue;
            } else {
                // XXX: warn and notify me
                log.error(
                    "ScheduleMessageService, a message time up, but reput it failed, topic: {} msgId {}",
                    msgExt.getTopic(), msgExt.getMsgId());
                ScheduleMessageService.this.timer.schedule(
                    new DeliverDelayedMessageTimerTask(this.delayLevel,
                        nextOffset), DELAY_FOR_A_PERIOD);
                ScheduleMessageService.this.updateOffset(this.delayLevel,
                    nextOffset);
                return;
            }
        } catch (Exception e) {
```

Message Filter

FAQ

1. Where does RocketMQ filter messages? Broker or Consumer?

Message Filter
Consume Queue Unit



Message Filter Code Time

```java
public static SubscriptionData buildSubscriptionData(final String consumerGroup, String topic,
    String subString) throws Exception {
    SubscriptionData subscriptionData = new SubscriptionData();
    subscriptionData.setTopic(topic);
    subscriptionData.setSubString(subString);

    if (null == subString || subString.equals(SubscriptionData.SUB_ALL) || subString.length() == 0) {
        subscriptionData.setSubString(SubscriptionData.SUB_ALL);
    } else {
        String[] tags = subString.split( regex: "\\|\\|");
        if (tags.length > 0) {
            for (String tag : tags) {
                if (tag.length() > 0) {
                    String trimString = tag.trim();
                    if (trimString.length() > 0) {
                        subscriptionData.getTagsSet().add(trimString);
                        subscriptionData.getCodeSet().add(trimString.hashCode());
                    }
                }
            }
        } else {
            throw new Exception("subString split error");
        }
    }

    return subscriptionData;
}
```

Message Filter In Broker Code Time

```
if (messageFilter != null
    && !messageFilter.isMatchedByConsumeQueue(isTagsCodeLegal ? tagsCode : null, extRet ? cqExtUnit : null)) {
    if (getResult.getBufferTotalSize() == 0) {
        status = GetMessageStatus.NO_MATCHED_MESSAGE;
    }

    continue;
}
```

```
@Override
public boolean isMatchedByConsumeQueue(Long tagsCode, ConsumeQueueExt.CqExtUnit cqExtUnit) {
    if (null == subscriptionData) {
        return true;
    }

    if (subscriptionData.isClassFilterMode()) {
        return true;
    }

    // by tags code.
    if (ExpressionType.isTagType(subscriptionData.getExpressionType())) {

        if (tagsCode == null) {
            return true;
        }

        if (subscriptionData.getSubString().equals(SubscriptionData.SUB_ALL)) {
            return true;
        }

        return subscriptionData.getCodeSet().contains(tagsCode.intValue());
    } else {
```

Filter message by hashCode of tag.

Message Filter In Consumer Code Time

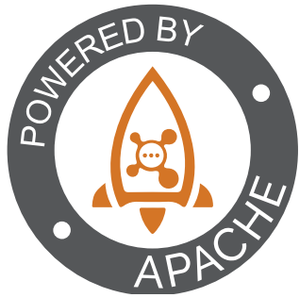
```
PullCallback pullCallback = new PullCallback() {
    @Override
    public void onSuccess(PullResult pullResult) {
        if (pullResult != null) {
            pullResult = DefaultMQPushConsumerImpl.this.pullAPIWrapper.processPullResult(pullRequest.
                subscriptionData);

            switch (pullResult.getPullStatus()) {
                case FOUND:
                    long prevRequestOffset = pullRequest.getNextOffset();
                    pullRequest.setNextOffset(pullResult.getNextBeginOffset());
                    long pullRT = System.currentTimeMillis() - beginTimestamp;
                    DefaultMQPushConsumerImpl.this.getConsumerStatsManager().incPullRT(pullRequest.ge
                        pullRequest.getMessageQueue().getTopic(), pullRT);

                    long firstMsgOffset = Long.MAX_VALUE;
                    if (pullResult.getMsgFoundList() == null || pullResult.getMsgFoundList().isEmpty(
                        DefaultMQPushConsumerImpl.this.executePullRequestImmediately(pullRequest);
                    } else {
                        firstMsgOffset = pullResult.getMsgFoundList().get(0).getQueueOffset();

                        DefaultMQPushConsumerImpl.this.getConsumerStatsManager().incPullTPS(pullReque
                            pullRequest.getMessageQueue().getTopic(), pullResult.getMsgFoundList().si

                        boolean dispatchToConsume = processQueue.putMessage(pullResult.getMsgFoundLis
                        DefaultMQPushConsumerImpl.this.consumeMessageService.submitConsumeRequest(
                            pullResult.getMsgFoundList(),
                            processQueue,
                            pullRequest.getMessageQueue(),
                            dispatchToConsume);
```

Do Filtering in consumer

Message Filter In Consumer Code Time

```
public PullResult processPullResult(final MessageQueue mq, final PullResult pullResult,
    final SubscriptionData subscriptionData) {
    PullResultExt pullResultExt = (PullResultExt) pullResult;

    this.updatePullFromWhichNode(mq, pullResultExt.getSuggestWhichBrokerId());
    if (PullStatus.FOUND == pullResult.getPullStatus()) {
        ByteBuffer byteBuffer = ByteBuffer.wrap(pullResultExt.getMessageBinary());
        List<MessageExt> msgList = MessageDecoder.decodes(byteBuffer);

        List<MessageExt> msgListFilterAgain = msgList;
        if (!subscriptionData.getTagsSet().isEmpty() && !subscriptionData.isClassFilterMode()) {
            msgListFilterAgain = new ArrayList<MessageExt>(msgList.size());
            for (MessageExt msg : msgList) {
                if (msg.getTags() != null) {
                    if (subscriptionData.getTagsSet().contains(msg.getTags())) {
                        msgListFilterAgain.add(msg);
                    }
                }
            }
        }
```

Filter message by tag set.



Thank you.