It encapsulates method to start,
stop and interrupt a thread

Runnable

The run() will be executed
in Thread.
It uses channel event
listener in enclosing class

ServiceThread

Thread

1   1

ResponseFuture

NettyRemotingAbstract   **NettyEventExecuter**

1   11

Encapsulate API to talk
to Server

ClientConfig   «uses»   MixAll

DatagramSocket

Bootstrap

RemotingClient   ChannelEventListener

MQProducer   MQProducerInner

DefaultMQProducer   **DefaultMQProducerImpl**   MQClientFactory   MQClientAPIImpl   **NettyRemotingClient**   ClientHousekeepingService   **ScheduledExecutorService**

1   1   1   1   1   1   1   1   1

TopAddressing

-topicPublishInfoTable   «uses»

1   1

TransactionMQProducer   1

MQClientManager

NettyRequestProcessor

BrokerController   ConsumerManager   ConsumerGroupInfo

1   1   1   1   *

1   1

1   «uses»

*   *

TopicPublishInfo   **TopicPublishInfo**   ClientRemotingProcessor

ChannelWrapper

ProducerManager

1

1

*

1

TransactionCheckListener

1

PullMessageService   PullRequest   NettyClientHandler   SimpleChannelInboundHandler

TransactionCheckListenerImpl

1   *

1

1

MessageQueue

ChannelDuplexHandler

<<uses>>

RebalanceService   NettyConnetManageHandler

1

<<uses>>   1

MessageToByteEncoder

NettyEncoder

LengthFieldBasedFrameDecoder

NettyDecoder

TopicConfigManager

SubscriptionGroupManager

SlaveSynchronize

DigestLogManager

ClientHousekeepingService

PullRequestHoldService

MessageToByteEncoder

NettyEncoder

«uses»

LengthFieldBasedFrameDecoder

NettyRequestProcessor

SendMessageProcessor

BrokerController

RemotingServer

NettyRemotingAbstract

NettyRemotingServer

SimpleChannelInboundHandler

«uses»

NettyDecoder

-storeHost

ExecutorService

ThreadPoolExecutor

NettyServerHandler

NettyConnetManageHandler

ConcurrentHashMap<String/* topic */, ConcurrentHashMap<Integer/* queueId */, ConsumeQueue>>

SocketAddress

MessageStore

DefaultMessageStore

ConsumeQueue

ServiceThread

AllocateMapedFileService

AllocateRequest

MapedFile

CleanCommitLogService

FlushConsumeQueueService

CleanConsumeQueueService

Total file size

IndexHeader.INDEX_HEADER_SIZE
+ (hashSlotNum * HASH_SLOT_SIZE)
+ (indexNum * INDEX_SIZE)

DispatchMessageService

IndexService

IndexFile

IndexHeader

ReputMessageService

StoreStatsService

ConfigManager

ScheduleMessageService

TransactionStateService

-tranRedoLog

-tranStateTable

CommitLog

MapedFileQueue

ReferenceResource

MessageStoreConfig

StoreCheckpoint

MapedFile

MappedByteBuffer

FlushCommitLogService

DirectByteBuffer

GroupCommitService

Write data to slave

HAService

AcceptSocketService

«uses»

HAConnection

WriteSocketService

Read request from slave

GroupTransferService

ReadSocketService

Guarantee message sender blocked
until its msg is synced to slave.

Sender waits on a countdown latch,
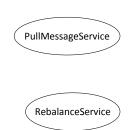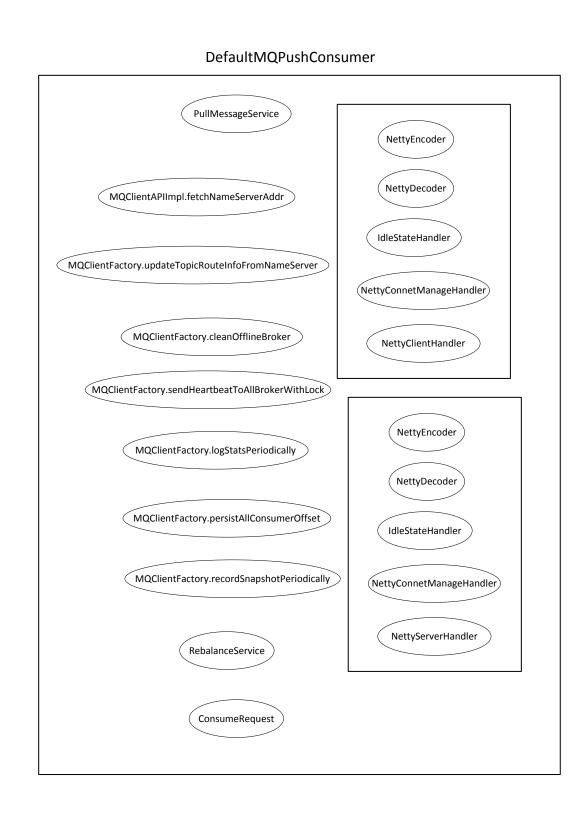it does the count down.

HAClient

NameServer

- DefaultRequestProcessor
- NettyRemotingServer.scanResponseTable
- NettyEventExecuter

  - NettyEncoder
  - NettyDecoder
  - IdleStateHandler
  - NettyConnetManageHandler
  - NettyServerHandler

Broker

- CleanConsumeQueueService
- IndexService
- FlushConsumeQueueService
- FlushRealTimeService
- GroupCommitRequest
- StoreStatsService
- ScheduleMessageService.persist
- ReputMessageService
- TransactionStateService
- AcceptSocketService
- GroupTransferService
- HAClient
- AllocateMapedFileService

  - NettyEncoder
  - NettyDecoder
  - IdleStateHandler
  - NettyConnetManageHandler
  - NettyServerHandler

  - NettyEncoder
  - NettyDecoder
  - IdleStateHandler
  - NettyConnetManageHandler
  - NettyClientHandler

- PullRequestHoldService
- ClientHousekeepingService.scanExceptionChannel
- ReadSocketService
- WriteSocketService
- DispatchMessageService

PullMessageService

RebalanceService

DefaultMQPushConsumer

- PullMessageService
- MQClientAPIImpl.fetchNameServerAddr
- MQClientFactory.updateTopicRouteInfoFromNameServer
- MQClientFactory.cleanOfflineBroker
- MQClientFactory.sendHeartbeatToAllBrokerWithLock
- MQClientFactory.logStatsPeriodically
- MQClientFactory.persistAllConsumerOffset
- MQClientFactory.recordSnapshotPeriodically
- RebalanceService
- ConsumeRequest

  - NettyEncoder
  - NettyDecoder
  - IdleStateHandler
  - NettyConnetManageHandler
  - NettyClientHandler

  - NettyEncoder
  - NettyDecoder
  - IdleStateHandler
  - NettyConnetManageHandler
  - NettyServerHandler

Filename is offset

GroupCommitService

requestsRead : ArrayList<GroupCommitRequest>

committedWhere

MapedFileQueue

MapedFile

wrotePostion

mappedByteBuffer

committedPosition

ConsumeQueue

CQStoreUnitSize = 20

CommitLog

GroupCommitRequest

DispatchMessageService

DispatchRequest

GroupCommitService

GroupCommitRequest

GroupTransferService

GroupCommitService.requestsWrite :
ArrayList<DispatchRequest>

DispatchMessageService.requestsWrite :
ArrayList<DispatchRequest>

GroupTransferService.requestsWrite :
ArrayList<DispatchRequest>

NettyRemotingAbstract

ChannelFutureListener

waitfor

Count down

ResponseFuture.countDownLatch

DefaultMessageStore

IndexService

IndexService.requestQueue:
LinkedBlockingQueue<Object[]>

WriteSocketService

ReadSocketService

HAConnection.slaveRequestOffset

slaveAckOff
set

...

...

slaveAckOff
set

HAConnection.slaveAckOffset

First diagram (Rebalance)

Participants: RebalanceService | MQClientFactory | DefaultMQPushConsumerImpl | RebalancePushImpl | AllocateMessageQueueAveragely | PullMessageService

- run
- doRebalance
- doRebalance
- doRebalance
- map(topic, SubscriptionData)
- getSubscriptionInner
- For each topic
- rebalanceByTopic
- Get mq set by topic
- findConsumerIdList
- Sort mq set and customer id list
- Allocate some mqs for the customer id
- allocate
- Remove unnessary process queues and add queues if needed
- updateProcessQueueTableInRebalance
- dispatchPullRequest
- executePullRequestImmediately
- executePullRequestImmediately
- truncateMessageQueueNotMyTopic

Second diagram (PullMessage)

Participants: PullMessageService | MQClientFactory | DefaultMQPushConsumerImpl | PullAPIWrapper | MQClientAPIImpl | NettyRemotingClient | NioSocketChannel

- run
- pullMessage
- selectConsumer
- pullMessage
- pullKernelImpl
- Async
- pullMessage
- pullMessageAsync
- invokeAsync
- invokeAsyncImpl
- new
- ResponseFuture
- Put responseFuture in responseTable, key:opaque
- writeAndFlush

Third diagram (Consume)

Participants: NettyClientHandler | NettyRemotingAbstract | ResponseFuture | InvokeCallback | PullCallback | PullAPIWrapper | MessageDecoder | ProcessQueue | ConsumeMessageConcurrentlyService | ThreadPoolExecutor | RemoteBrokerOffsetStore

- channelRead0
- processMessageReceived
- processMessageReceived
- Get responseFuture through cmd opaque
- processResponseCommand
- Implementation here
- getCallbackExecutor
- Implementation here
- If no call back executor, run in the same thread; otherwise in another thread
- executeInvokeCallback
- operationComplete
- onSuccess
- processPullResult
- Filter msg again
- decodes
- Msg List
- PullResultExt
- putMessage
- submitConsumeRequest
- Call listener's consumeMessage. Listener is passed in through DefaultMQPushConsumer.registerMessageListener. Then record stats data
- new
- ConsumeRequest
- submit
- run
- resetRetryTopic
- processConsumeResult
- Offset will be persisted to broker in MQClientFactory.persistAllConsumerOffset which is a schedule task
- updateOffset

```
NettyServerHandler  NettyRemotingAbstract  PullMessageProcessor  SubscriptionGroupManager  TopicConfigManager  FilterAPI  DefaultMessageStore  ConsumeQueue  DefaultMessageFilter  CommitLog  StoreStatsService  GetMessageResult                    NioSocketChannel
```

channelRead0

processMessageReceived

processRequestCommand

Run in a thread

processRequest

processRequest

findSubscriptionGroupConfig

selectTopicConfig

buildSubscriptionData

getMessage

findConsumeQueue

getIndexBuffer

isMessageMatched

getMessage

getGetMessageTransferedMsgCount

addMessage

new

ManyMessageTransfer

writeAndFlush

Port: 9876

NamesrvConfig

NamesrvStartup

«uses»

«uses»

NettyServerConfig

Map:
(topic, List<QueueData>)
(brokerName, BrokerData)
(clusterName, brokerName set)
(brokerAddr, BrokerLiveInfo)

«uses»

NamesrvController

1
1

1

RouteInfoManager

1

1

KVConfigManager

1

1

DefaultRequestProcessor

NettyRequestProcessor

1

*

1

RemotingServer

NettyRemotingServer

Executor

1 *

ThreadPoolExecutor

1

1

DefaultEventExecutorGroup

1

BrokerController | DefaultMessageStore | CleanConsumeQueueService | IndexService | FlushConsumeQueueService | CommitLog | StoreStatsService | ScheduleMessageService | ReputMessageService | TransactionStateService | HAService | NettyRemotingServer | BrokerOuterAPI | NettyRemotingClient | PullRequestHoldService | ClientHousekeepingService | DigestLogManager | AcceptSocketService | GroupTransferService | HAClient | HAConnection | ReadSocketService | WriteSocketService

start

start

start

start

start

start

start

start

start

start

start

createTempFile

start

start

start

start

start

start

start

start

start

start

start

start

start

registerBrokerAll

registerBrokerAll

registerBroker

Producer

Msg

Broker(Master)

Offset, size, timestamp, groupHashCode, MessageSysFlag.TransactionPreparedType

Offset, size, tagsCode(for trx prepare)

Offset, size, tagsCode

Msg

CommitLog

ConsumeQueue

redoLog:ConsumeQueue

tranStateTable:MapedFileQueue

MapedFile

B   TopicConfig, ConsumerOffset, DelayOffset, SubscriptionGroupConfig

NameServer

msg

Master Broker Info

Broker Info

Broker(Slave)

msg

offset, batch size

QeueeData, BrokerData(by topic)

offset, batch size

PushConsumer

NettyServerHandler | NettyRemotingAbstract | DefaultRequestProcessor | RemotingCommand | TopicConfigSerializeWrapper | RouteInfoManager | QueueData

channelRead0

processMessageReceived

Run in a thread

processRequestCommand

processRequest

registerBroker

createResponseCommand

getCustomHeader

decodeCommandCustomHeader

decode

Create tables to put
(clusterName, brokerNames) - clusterAddrTable
(brokerName, brokerData) - brokerAddrTable
(brokerId, brokerAddr) - BrokerData.brokerAddrs

clusterName : brokerName : brokerId : brokerAddr
1 : M   1 : M   1 : 1

registerBroker

new

BrokerLiveInfo

createAndUpdateQueueData

master id : 0
slave id > 0
Update for master only.
Upate topicQueueTable
for topic

equals

Save
(brokerAddr, BrokerLiveInfo).

For Slave:
set master addr and its
HAServer addr

Sequence diagram participants:
NamesrvStartup | KVConfigManager | Executors | RouteInfoManager | NettyEventExecuter

main

Load configuration

new → NamesrvController

initialize

load

new → NettyRemotingServer

newFixedThreadPool

registerProcessor

Thread name:
RemotingExecutorThread_...

defaultRequestProcessor =
(DefaultRequestProcessor, ThreadPoolExecuter)

new → DefaultRequestProcessor

registerDefaultProcessor

The following tasks
run periodically

scanNotActiveBroker

printAllPeriodically

Add shutdown hook

printAllPeriodically

Handlers seqence:

NettyEncoder
NettyDecoder
IdleStateHandler
NettyConnetManageHandler
NettyServerHandler

Thread name:
NettyServerWorkerThread_...

Thread number:
server threads in netty server
config

start

start

new → DefaultEventExecutorGroup

Run periodically

Remove timeout
response

start

scanResponseTable

non-block and
listen to "ACCEPT"

HAService | AcceptSocketService | ServerSocketChannel | RemotingUtil | GroupTransferService | HAClient | ReadSocketService | WriteSocketService | GroupCommitRequest | DefaultMessageStore | CommitLog | MapedFileQueue | MapedFile

start

beginAccept

open

openSelector

start

start

start

run

new

HAConnection

start

start

start

run

waitForRunning

doWaitTransfer

wakeupCustomer

run

processReadEvent

Get ack offset
from slave

If nextTransferFromWhere == -1
    if slaveRequestOffset == 0
        nextTransferFromWhere = last file offset;

    if slaveRequestOffset != 0
        nextTransferFromWhere = slaveRequestOffset

run

If last time sending is done
    if haven't sent heartbeat for a while
        send heartbeat
    else continue the sending

transferData

Construct transfer data with
configured batch size, send

getCommitLogData

getData

findMapedFileByOffset

selectMapedBuffer

transferData

Connect to master
and get max phy. offset

If it's master itself, this
function returns false,
the following work will
not be done.

run

connectMaster

reportSlaveMaxOffset

processReadEvent

dispatchReadRequest

appendToCommitLog

HAClient	AcceptSocketService	ReadSocketService	WriteSocketService
●	●	●	●
Connect Master	Wait for OP_ACCEPT(1s timeout)	Wait for OP_READ(1s timeout)	Wait for OP_WRITE(1s timeout)
Register OP_READ	◇ [Not all seleted keys are accessed]	Read slave ack offset	◇ [First time transmission]
Wait for OP_READ	◇ [OP_ACCEPT]	Update slaveAckOffset	◇ [Slave has no data]
dispatchReadRequest	Create HAConnection and start it	Update slaveRequestOffset if it < 0	Send from the last phy. file / Send from the slave requested offset
Report slave max offset	Clear seleted keys		◇ [Transimission not done last time]
			Continue the transimission
			Send from nextTransferFromWhere with configured batch size
			Update nextTransferFromWhere

TransactionMQProducer | DefaultMQProducerImpl | TransactionExecuterImpl | MQClientAPIImpl | NettyRemotingClient

sendMessageInTransaction

sendMessageInTransaction

send

Step 1: Send "Prepared"
msg to broker by putting
(PROPERTY_TRANSACTION_PREPARED, true)
property in msg

executeLocalTransactionBranch

Step2: get local trans state

endTransaction

endTransactionOneway

invokeOneway

invokeOnewayImpl

Participants (lifelines): DefaultMessageStore, CommitLog, MapedFileQueue, MessageStoreConfig, TransactionStateService, ConsumeQueue, ScheduleMessageService, MessageStoreConfig, IndexService, IndexHeader, MapedFile, SelectMapedBufferResult

- load
- isTempFileExist
- load
- load
- Load files under commitlog
- loadConsumeQueue
- ConcurrentHashMap<String/* topic */, ConcurrentHashMap<Integer/* queueId */, ConsumeQueue>>
- getStorePathConsumeQueue
- new
- ConsumeQueue
- Iterate topic and its queues
- putConsumeQueue
- load
- load
- For redo log from transaction/redolog
- load
- load
- load
- For state table from transaction\statetable
- load
- Map files under a specific directory (redo in this case) to DirectByteBuffer. Save in a map file list
- parseDelayLevel
- getStoreCheckpoint
- new
- StoreCheckpoint
- Load from store\index
- load
- new
- IndexFile
- From mapped byte buffer
- load
- load
- Check for each unit in mapfile: offset >=0 && size > 0. Will ignore the unit doesn't hold this condition and those after it. Set wrote & commited postion in the last file and delte the files after it.
- recover
- recoverConsumeQueue
- Recover redo log
- recover
- truncateDirtyFiles
- recover
- Last exit ok
- recoverNormally
- Last exit not ok
- recoverAbnormally
- last exit ok
- recoverStateTable
- recoverStateTableNormal
- recoverTopicQueueTable
- topic-queueid, minPhyOffset
- last exit not ok
- destroy
- recreateStateTable
- Step 1: Scan redo log and add "prepare" units to a set
- selectMapedBuffer
- release
- Step 2: Append "prepare" units to tranStateTable
- appendPreparedTransaction
- appendMessage

```
NettyServerHandler      NettyRemotingAbstract      SendMessageProcessor      DefaultMessageStore

channelRead0
                 processMessageReceived
                                          processRequestCommand

          Run in a thread
                                   processRequest
                                                             sendMessage

                                                     putMessage
```

Participants (lifelines, left to right):
DefaultMessageStore, CommitLog, MapedFileQueue, AllocateMapedFileService, MapedFile, DefaultAppendMessageCallback, MessageDecoder, AppendMessageResult, DispatchMessageService, GroupCommitService, HAService, GroupTransferService, ConsumeQueue, TransactionStateService, IndexService, StoreCheckpoint

Notes and messages:

- putMessage
- putMessage
- getLastMapedFile
- putRequestAndReturnMapedFile
- Put nextReq, nextNextReq in priority queue. Meanwile a thread running mmapOperation()
- getMapedFile
- setFirstCreateInQueue
- Get queueOffset by topic-queueId; If no entry, create one. Update queueOffset for transaction purpose
- If result is END_OF_FILE, create a new file and do appendMessage again
- appendMessage
- doAppend
- createMessageId
- STORETIMESTAMP + STOREHOSTADDRESS + OFFSET
- If not enough space, return END_OF_FILE; Otherwise put msg parts in buffer(mappedByteBuffer in MapedfFile)
- resetMsgStoreItemMemory
- getWroteBytes
- getStoreTimestamp
- new
- DispatchRequest
- putDispatchRequest
- putRequest
- new
- GroupCommitRequest
- If SYNC_FLUSH
- putRequest
- Not SYNC_FLUSH
- waitForFlush
- wakeup
- getHaService
- if SYNC_MASTER
- putRequest
- putRequest
- run
- Step1: send msg to ConsumeQueue; don't dispatch for TransactionPreparedType TransactionRollbackType
- doDispatch
- putMessagePostionInfo
- By topic, queue id
- findConsumeQueue
- putMessagePostionInfoWrapper
- putMessagePostionInfo
- Step2: Update transaction state table
- setLogicsMsgTimestamp
- Step3: Record redo log
- appendPreparedTransaction
- TransactionPreparedType
- updateTransactionState
- TransactionCommitType TransactionRollbackType
- putMessagePostionInfoWrapper
- putRequest
- run
- doWaitTransfer
- Check for every request if it's passed to slave. if it's not, set flag not ok. Wait up the senderwith the flag.

Participants (lifelines):
- DefaultMQProducer
- DefaultMQProducerImpl
- MQClientFactory
- MQClientAPIImpl
- NettyRemotingClient
- Bootstrap
- TopicPublishInfo
- NioSocketChannel

Messages / labels:

- send
- send
- makeSureStateOK
- checkMessage
- sendDefaultImpl
- tryToFindTopicPublishInfo
- new → TopicPublishInfo
- updateTopicRouteInfoFromNameServer
- getTopicRouteInfoFromNameServer
- invokeSync
- getAndCreateChannel
- getAndCreateNameserverChannel
- createChannel
- connect
- invokeSyncImpl
- updateTopicPublishInfo
- updateTopicSubscribeInfo
- writeAndFlush
- selectOneMessageQueue
- sendKernelImpl
- findBrokerAddressInPublish
- updateTopicRouteInfoFromNameServer
- findBrokerAddressInPublish
- sendMessage
- sendMessageSync
- invokeSync

Notes:

- Loop all the producers in producerTable.

  Topic publish info contains topic, mqs and sendWhichQueue

- Get brokers data and queues data

- GET_ROUTEINTO_BY_TOPIC_VALUE

- Loop all consumers in consumerTable.

  subscribeInfo is mq set

- Save (topic, TopicRouteData)

- 1. NioSocketChannel.writeAndFlush sends out the request
  2. A listener is added to handle the response.

- Get one in round robin. If last broker name is not null, choose one with a different broker name

- if can't find broker addr

- Only master

- Put producerGroup, topic, queue id in header

Start factory

DefaultMQProducerImpl | MQClientManager | MQClientFactory | DefaultMQProducer | MQClientAPIImpl | TopAddressing | NettyRemotingClient | NettyEventExecuter | PullMessageService | RebalanceService

start

checkConfig

If there's a factor associated with the client id, return it; otherwise create a new one, add it to a table and return it

getAndCreateMQClientFactory

registerProducer

Put (topicKey, topicPublishInfo) in table

getCreateTopicKey

udp port:
33333 + instanceName.hashCode % 10000

new

TopicPublishInfo

Get addresses from jmenv.tbsite.net:8080/rocketmq/nsaddr

start

makeSureInstanceNameIsOnly

new

DatagramSocket

fetchNameServerAddr

fetchNSAddr

If the addresses returned don't match existing, replace

updateNameServerAddressList

Handler sequence:

NettyEncoder,
NettyDecoder,
IdleStateHandler,
NettyConnetManageHandler,
NettyClientHandler

updateNameServerAddressList

Pick an event and pass to listener

start

start

start

start

run

startScheduledTask

start

Don't start factory

start

start

sendHeartbeatToAllBrokerWithLock

sendHeartbeatToAllBroker

sendHearbeat

Participants: BrokerController, TopicConfigManager, ConsumerOffsetManager, HAService, HAClient, ConsumerOffsetManager, BrokerOuterAPI, SlaveSynchronize

- initialize
- load
- load
- new → DefaultMessageStore
- load
- new → NettyRemotingServer
- registerProcessor
- new → SendMessageProcessor
- new → QueryMessageProcessor
- new → ClientManageProcessor
- registerProcessor
- registerDefaultProcessor
- persist
- recordPullTPS
- computePullTPS
- fetchNameServerAddr
- updateHaMasterAddress
- updateMasterAddress
- updateMasterAddress
- syncAll
- syncTopicConfig
- getAllTopicConfig
- syncConsumerOffset
- syncDelayOffset
- syncSubscriptionGroupConfig

Note (Create sendMessageExecutor):
Create sendMessageExecutor
pullMessageExecutor
adminBrokerExecutor

Note (Save the following in table):
Save the following in table
(SEND_MESSAGE_VALUE, SendMessageProcessor, sendMessageExecutor)
(CONSUMER_SEND_MSG_BACK_VALUE, SendMessageProcessor, sendMessageExecutor)
(PULL_MESSAGE_VALUE, PullMessageProcessor, pullMessageExecutor)
(QUERY_MESSAGE_VALUE, QueryMessageProcessor, pullMessageExecutor)
(VIEW_MESSAGE_BY_ID_VALUE, QueryMessageProcessor, pullMessageExecutor)
(HEART_BEAT_VALUE, ClientManageProcessor, adminBrokerExecutor)
(UNREGISTER_CLIENT_VALUE, ClientManageProcessor, adminBrokerExecutor)
(GET_CONSUMER_LIST_BY_GROUP_VALUE, ClientManageProcessor, adminBrokerExecutor)
(END_TRANSACTION_VALUE, EndTransactionProcessor, sendMessageExecutor)
(AdminBrokerProcessor, adminBrokerExecutor)

Note (Run the following tasks periodically):
Run the following
tasks periodically

Note (If name server is null):
If name server is null

Note (If it's slave):
If it's slave, get master's
HaAddress, and sync
information periodically

Note (key):
key: topic + TOPIC_GROUP_SEPARATOR + group, queue id
value: offset
During commit, the offset is updated. The table
is persisted periodically

Note (Use last 2 snapshots):
Use last 2 snapshots to
calculate pull tps

Format:
<length> <header length> <header data> <body data>

Use reflection to put
fields in customHeader
to extFields

NettyEncoder

RemotingCommand

RemotingSerializable

JSON

String

ByteBuf

RemotingUtil

encode

encodeHeader

buildHeader

makeCustomHeaderToNet

encode

toJson

toJSONString

getBytes

writeBytes

getBody

writeBytes

If exception happened

closeChannel