NSQ重塑之路

Youzan消息队列在微服务架构中的实践

李文

有赞技术专家

有赞

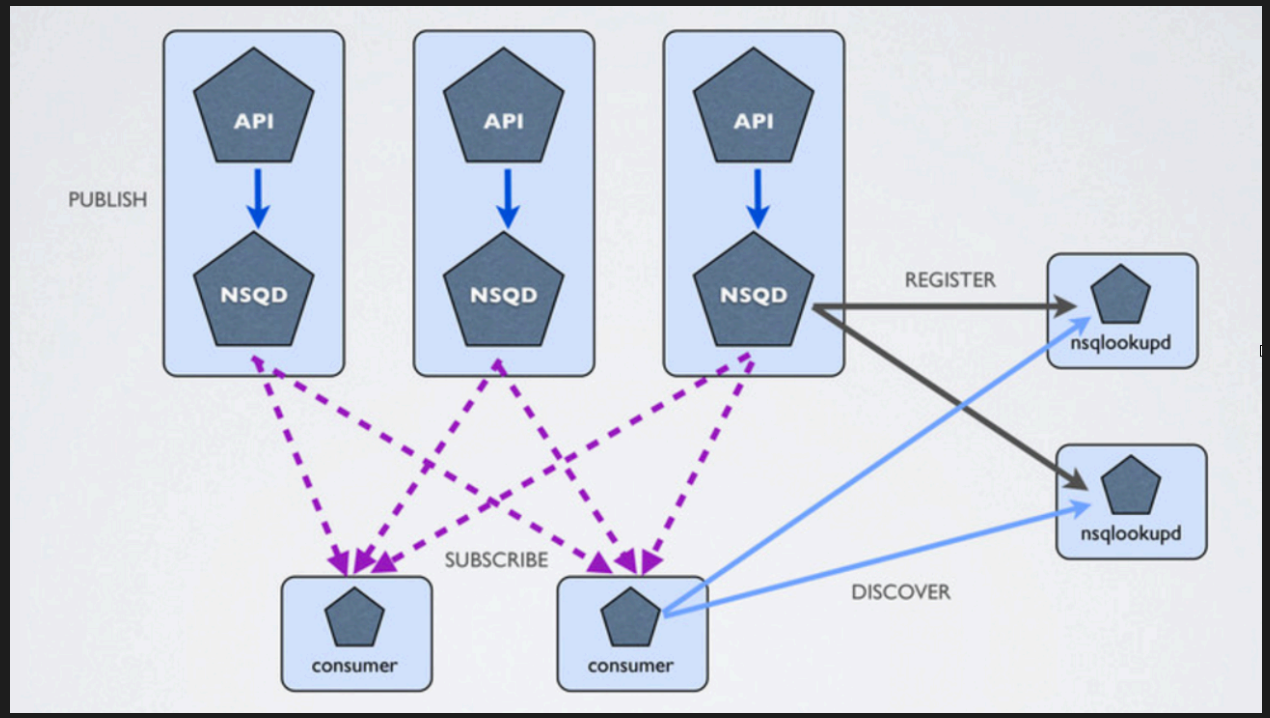
GopherChina 2017

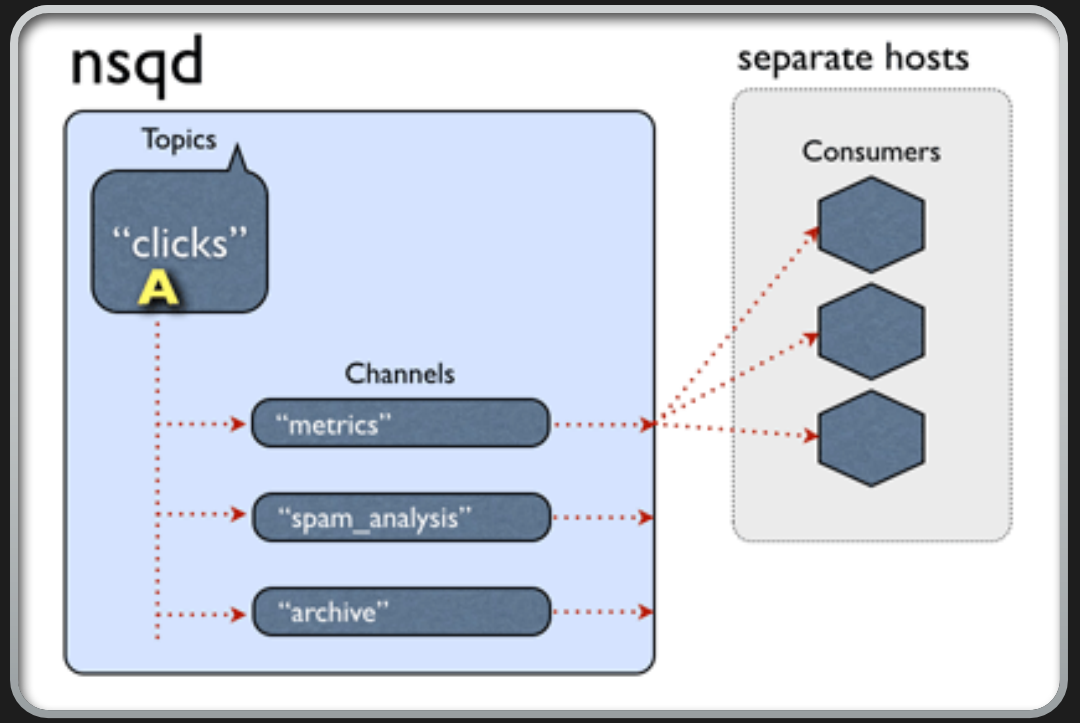
Agenda

- Original Architecture Overview
- Missing Features and the Demand of Youzan
- Redesigned Architecture
- Jepsen test in the new NSQ
- Compare with others
- Usage in Youzan

What is MQ?
What is NSQ?

Original Arch



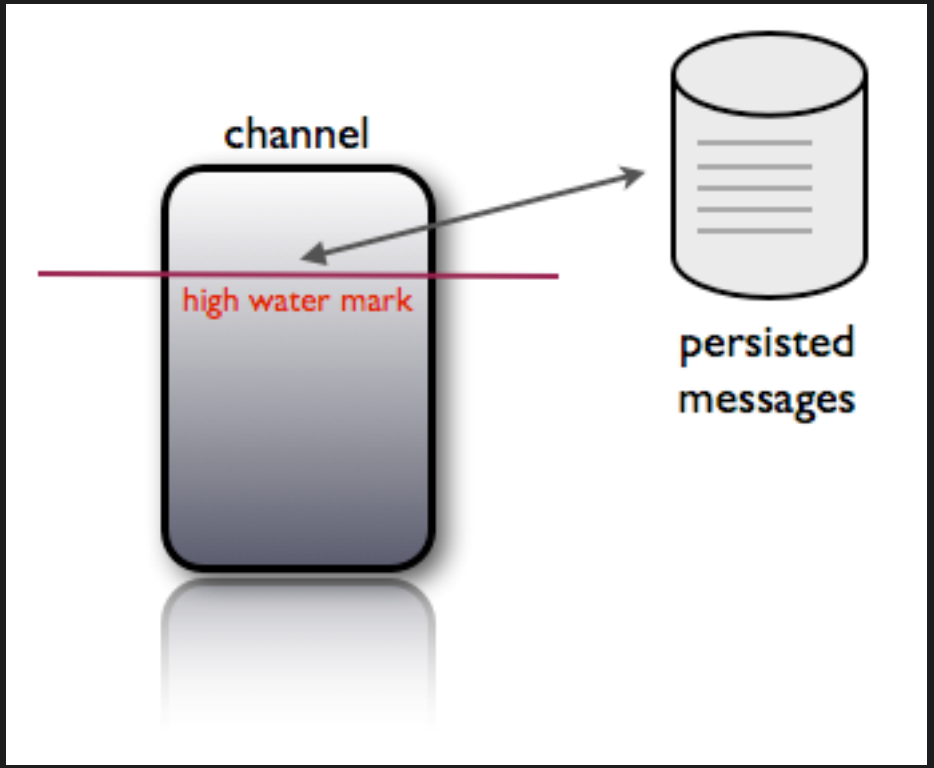


Missing Features

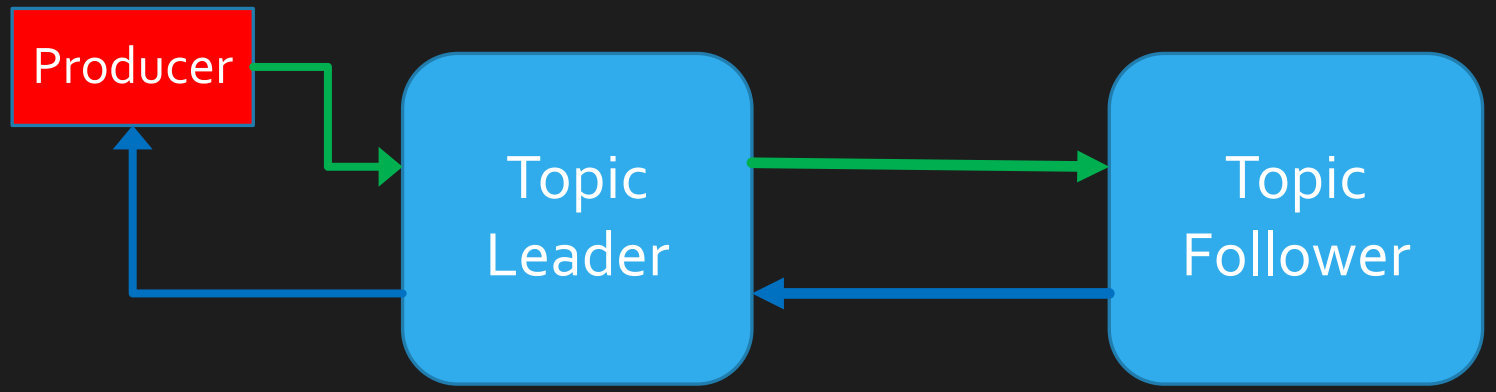
- Replication
- HA
- Auto-Balance
- Delivery in Order
- Tracing
- Consume History Messages

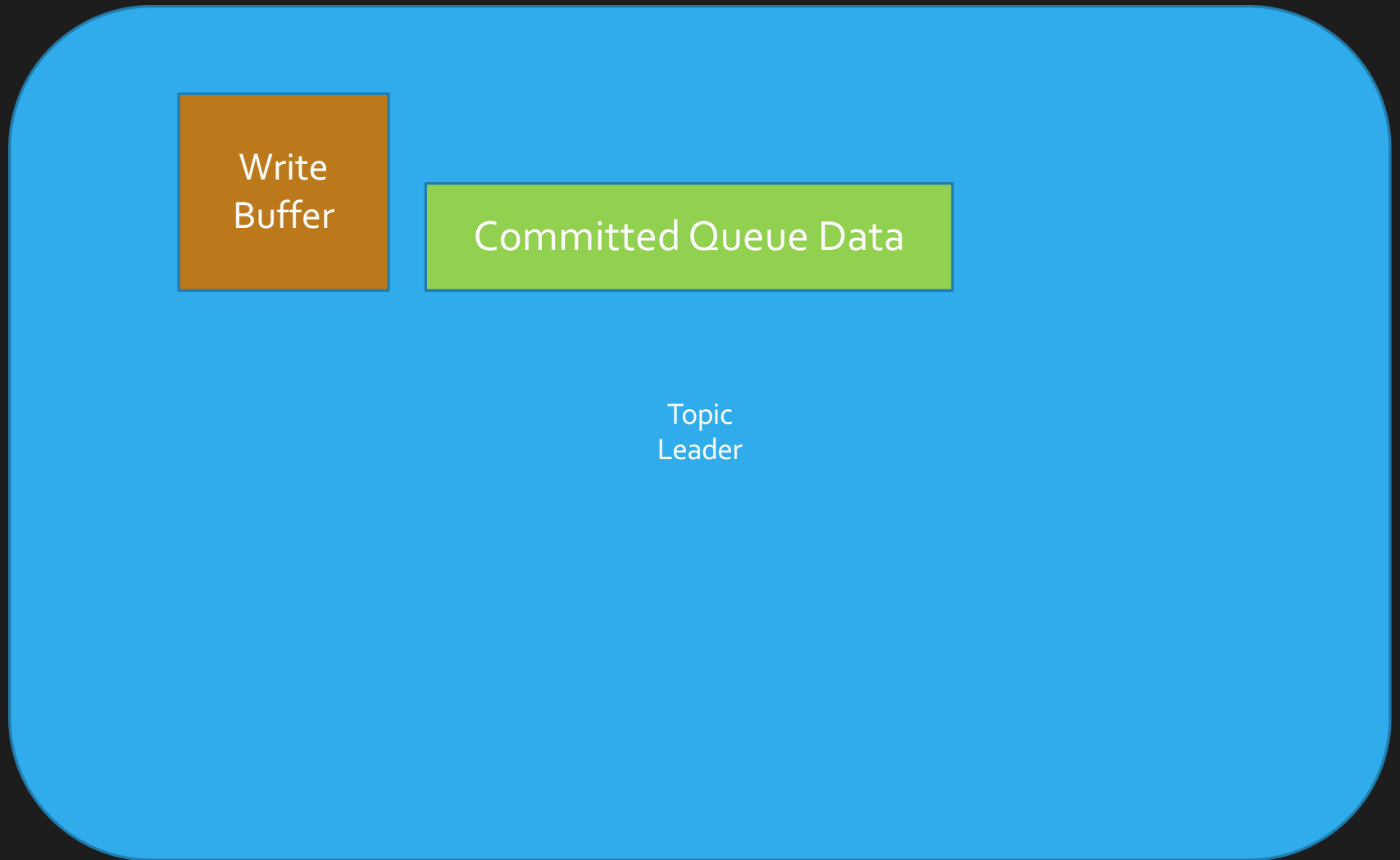
Redesigned Topic Queue

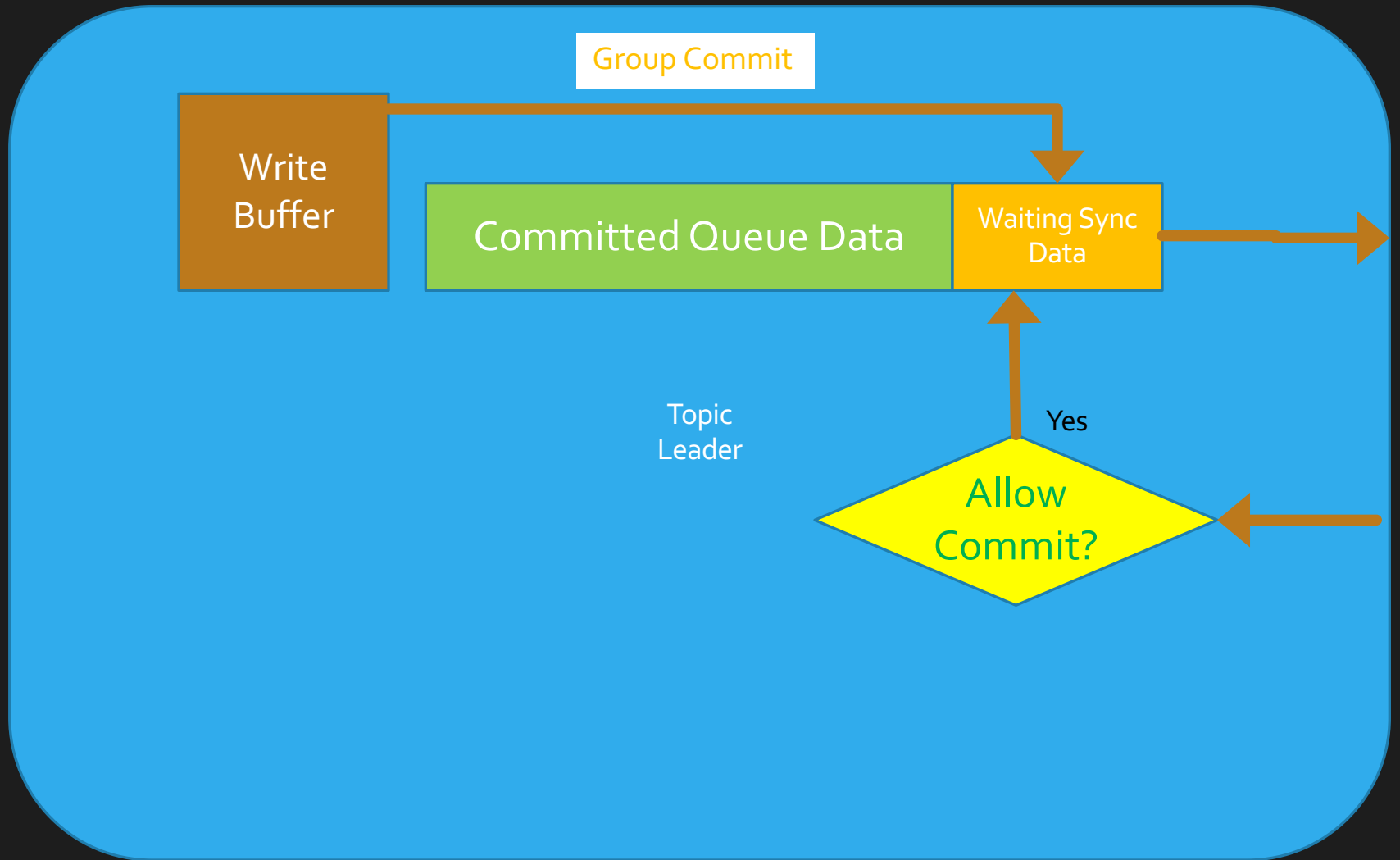
- Use go channel to store data
- Not searchable, Not Stable
- Redesigned: Use list of segment files

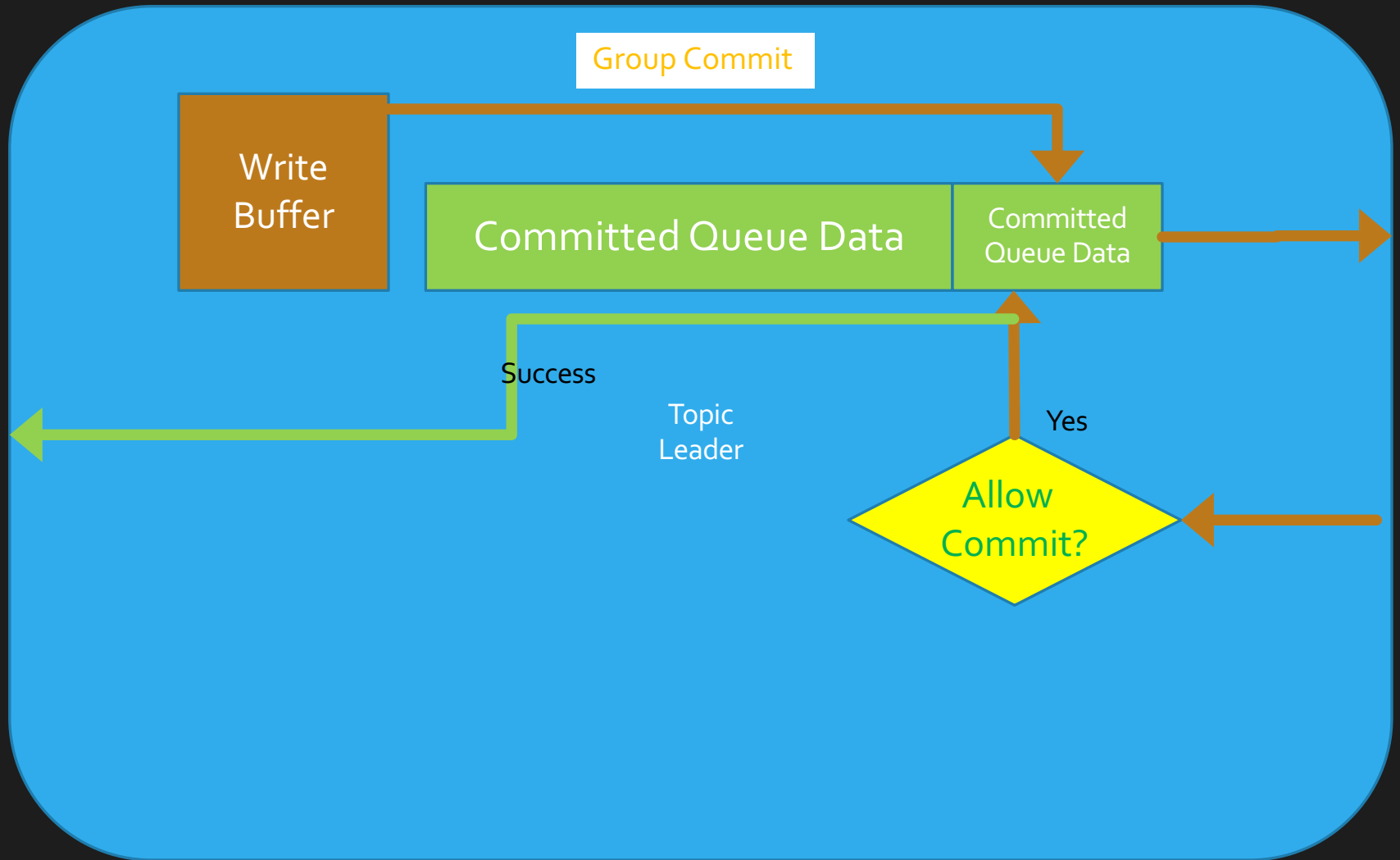


Topic Write Flow on NSQD




Write
Buffer
Committed Queue Data
Topic
Leader


Group Commit
Write Buffer
Committed Queue Data
Waiting Sync Data
Topic Leader
Yes
Allow Commit?


Group Commit
Write Buffer
Committed Queue Data
Committed Queue Data
Success
Topic Leader
Yes
Allow Commit?

Improve Write Performance

Group Commit In Go

```
info := &nsqd.PubInfo{
    Done:     make(chan struct{}),
    MsgBody:  msgBody,
    StartPub: time.Now(),
}
if clientTimer == nil {
    clientTimer = time.NewTimer(time.Second * 5)
} else {
    clientTimer.Reset(time.Second * 5)
}
select {
case topic.GetWaitChan() <- info:
default:
    select {
    case topic.GetWaitChan() <- info:
    case <-topic.QuitChan():
        nsqd.NsqLogger().Infof("topic %v put messages failed at exiting", topic.GetFullName())
        return nsqd.ErrExiting
    case <-clientTimer.C:
        nsqd.NsqLogger().Infof("topic %v put messages timeout ", topic.GetFullName())
        return ErrPubToWaitTimeout
    }
}
<-info.Done
return info.Err
```

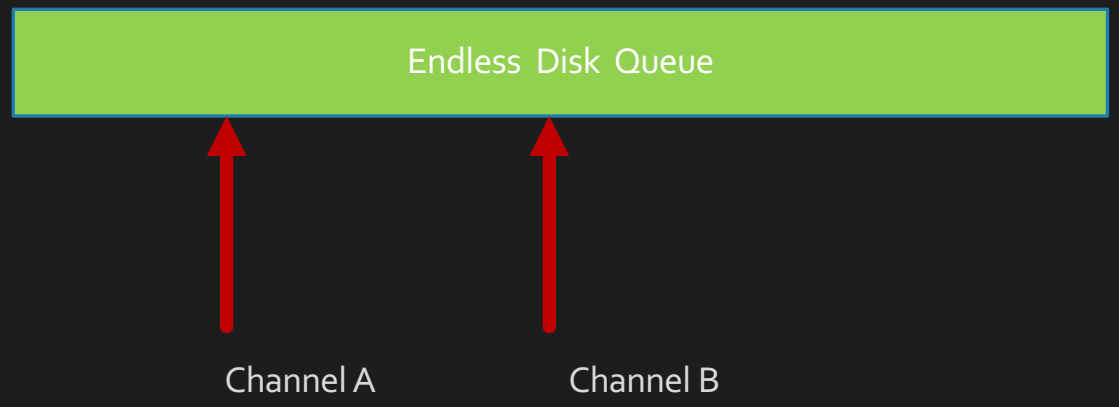
```
257     quitChan := topic.QuitChan()
258     infoChan := topic.GetWaitChan()
259     for {
260         select {
261         case <-quitChan:
262             return
263         case info := <-infoChan:
264             if info.MsgBody.Len() <= 0 {
265                 nsqd.NsqLogger().Logf("empty msg body")
266             }
267             messages = append(messages, nsqd.NewMessage(0, info.MsgBody.Bytes()))
268             pubInfoList = append(pubInfoList, info)
269             // TODO: avoid too much in a batch
270         default:
271             if len(pubInfoList) == 0 {
272                 select {
273                 case <-quitChan:
274                     return
275                 case info := <-infoChan:
276                     messages = append(messages, nsqd.NewMessage(0, info.MsgBody.Bytes()))
277                     pubInfoList = append(pubInfoList, info)
278                 }
279                 continue
280             }
281             var retErr error
282             if c.checkForMasterWrite(topicName, partition) {
283                 _, _, _, err := c.PutMessages(topic, messages)
284                 if err != nil {
285                     nsqd.NsqLogger().LogErrorf("topic %v put messages %v failed: %v", topic.GetFullName(), len(messages), err)
286                     retErr = err
287                 }
288             } else {
289                 topic.DisableForSlave()
290                 nsqd.NsqLogger().LogDebugf("should put to master: %v",
291                     topic.GetFullName())
292                 retErr = consistence.ErrNotTopicLeader.ToErrorType()
293             }
294             for _, info := range pubInfoList {
295                 info.Err = retErr
296                 close(info.Done)
297             }
298             pubInfoList = pubInfoList[:0]
299             messages = messages[:0]
300         }
301     }
```

Redesigned Consume Channel

- Copy all data from topic channel to consume channel
- Waste disk space, lots of data copy
- Redesigned: Use cursor only for consume

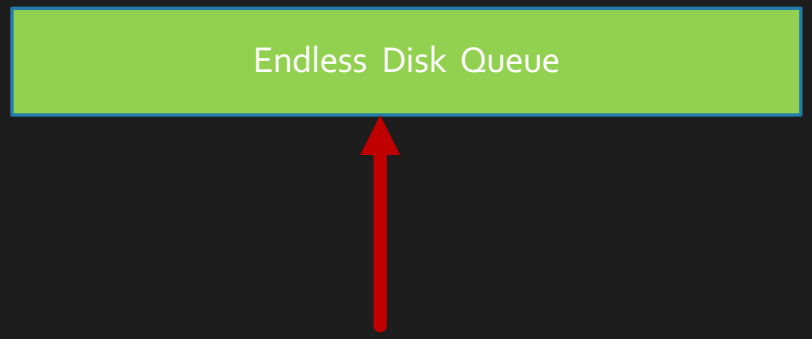
Consume Channel

- Each Channel Hold the Offset cursor Info
- All clients share same offset cursor info in the same channel

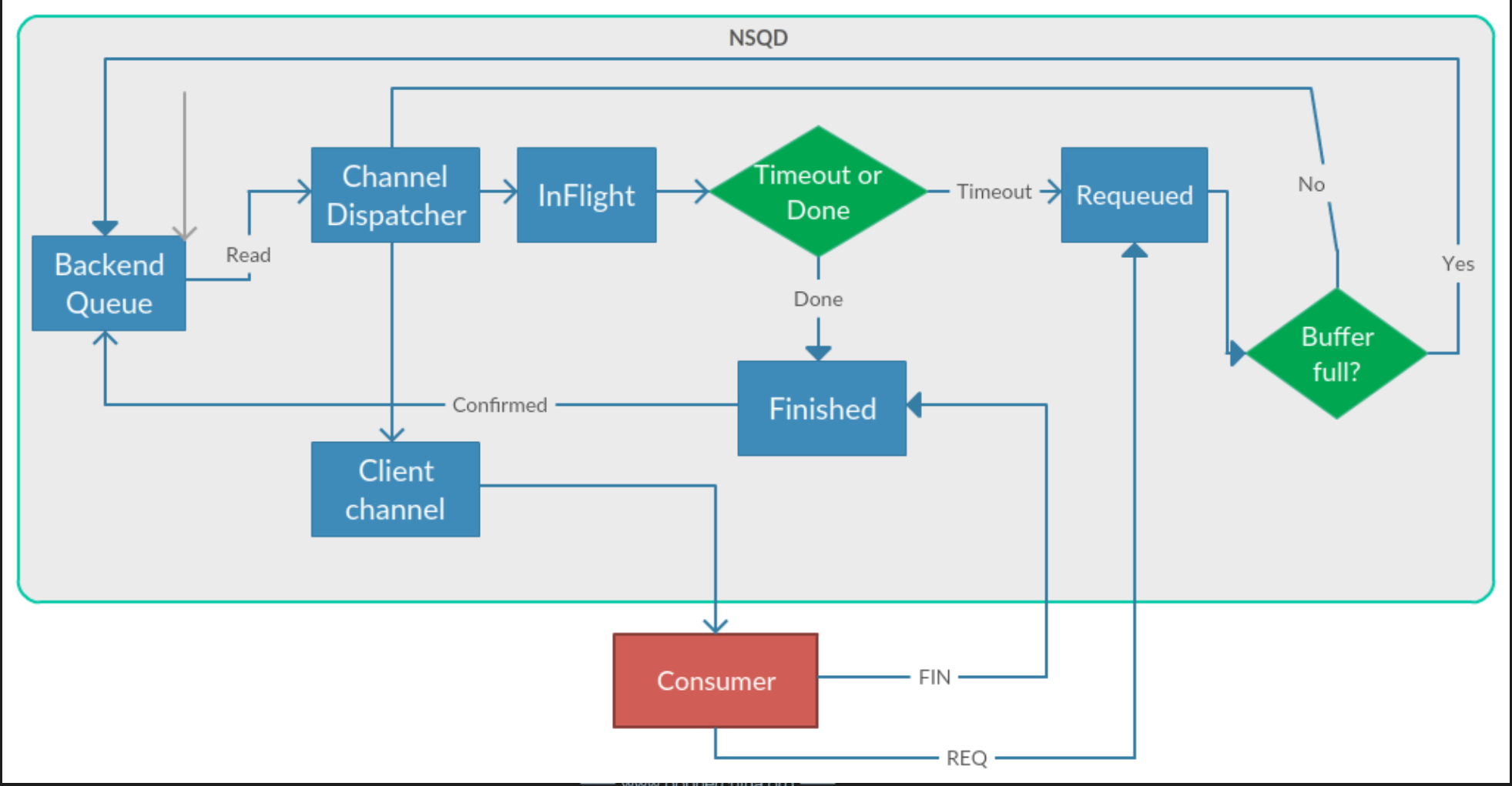


Consume in History

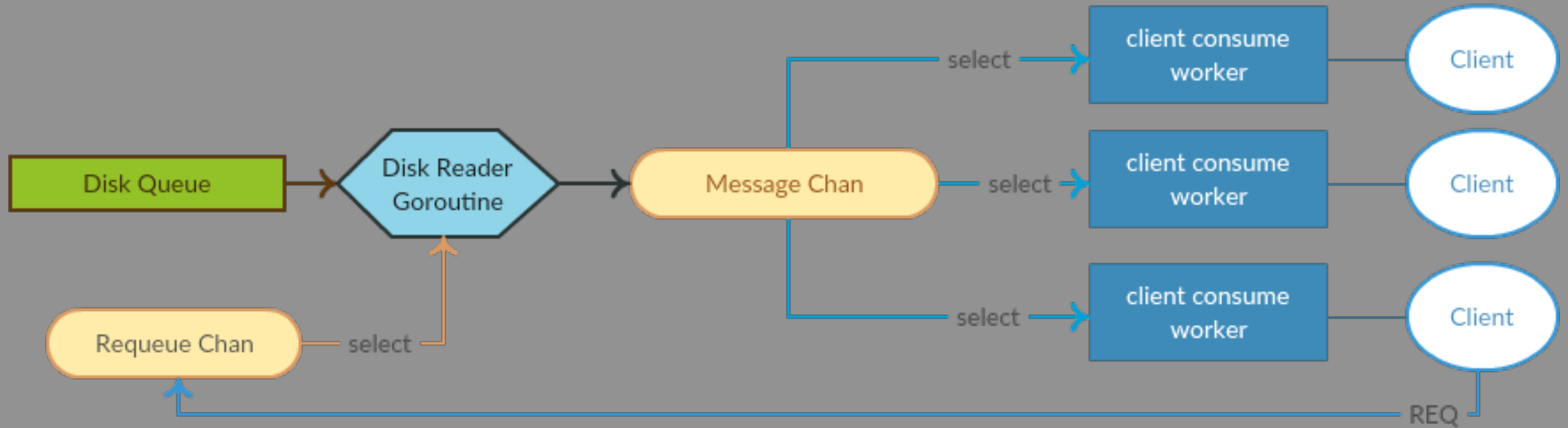
- Move consume cursor
- Support timestamp, disk queue offset



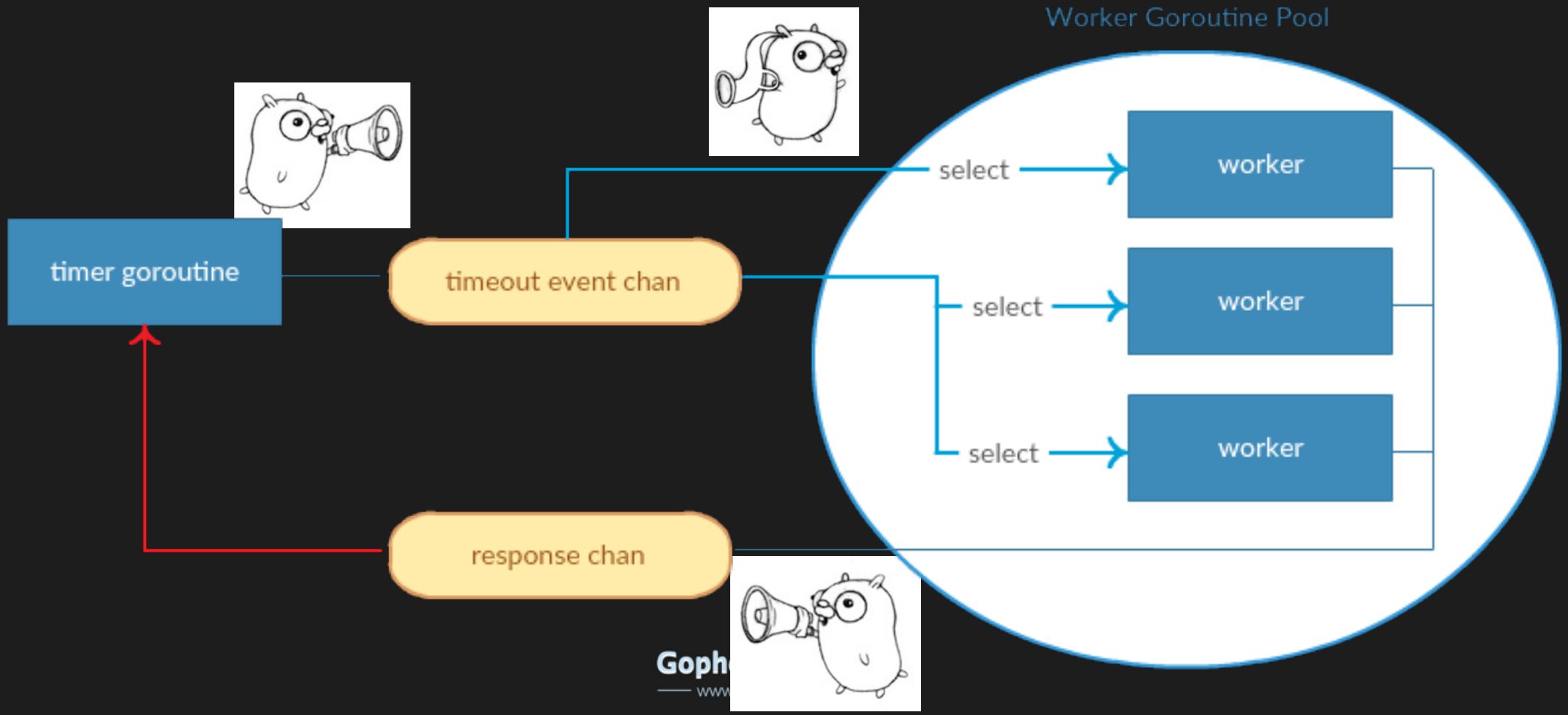
Message Sub State Machine



Dispatcher in Go

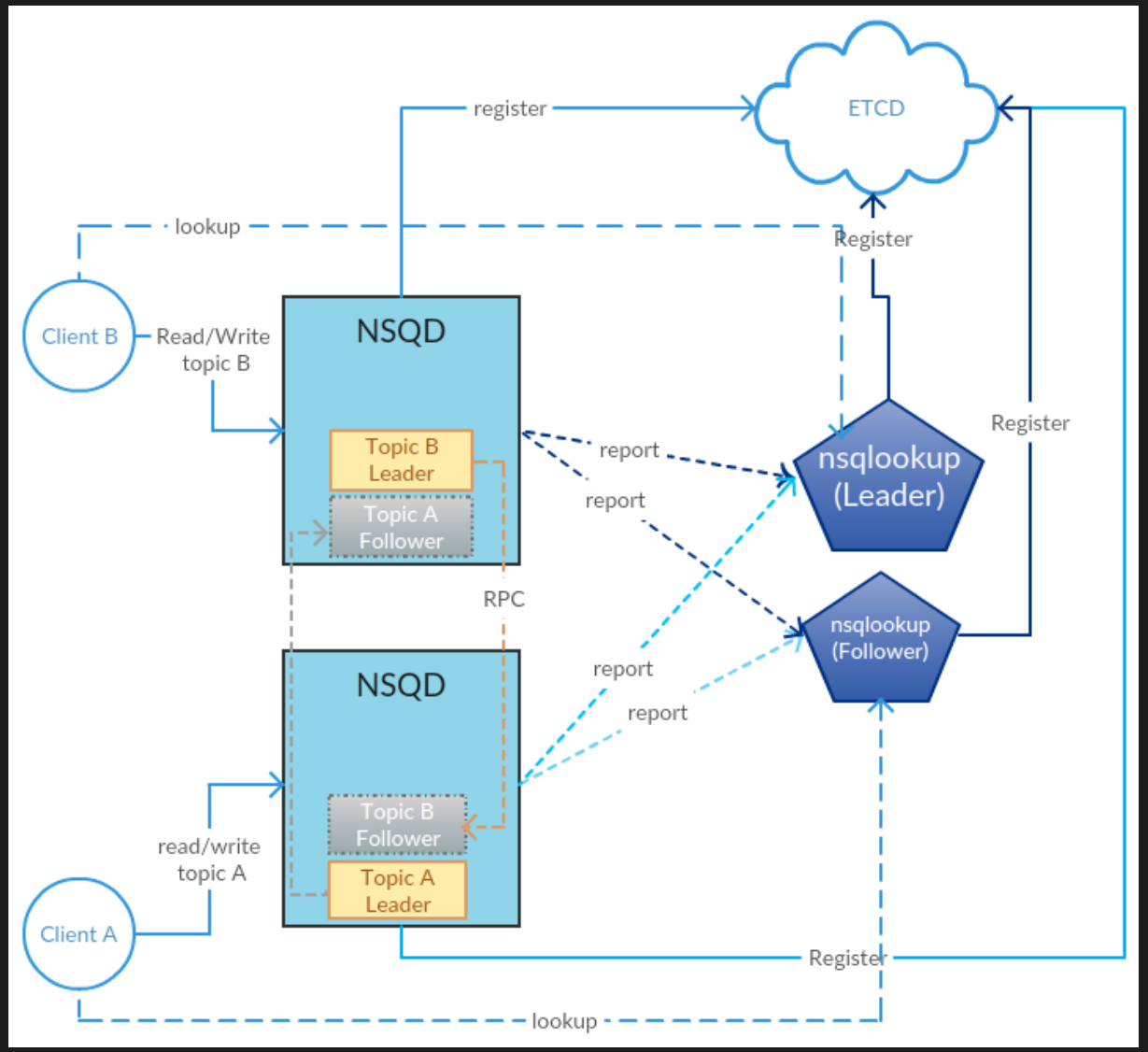


Optimize channel timeout in Go



Redesigned Replication & HA

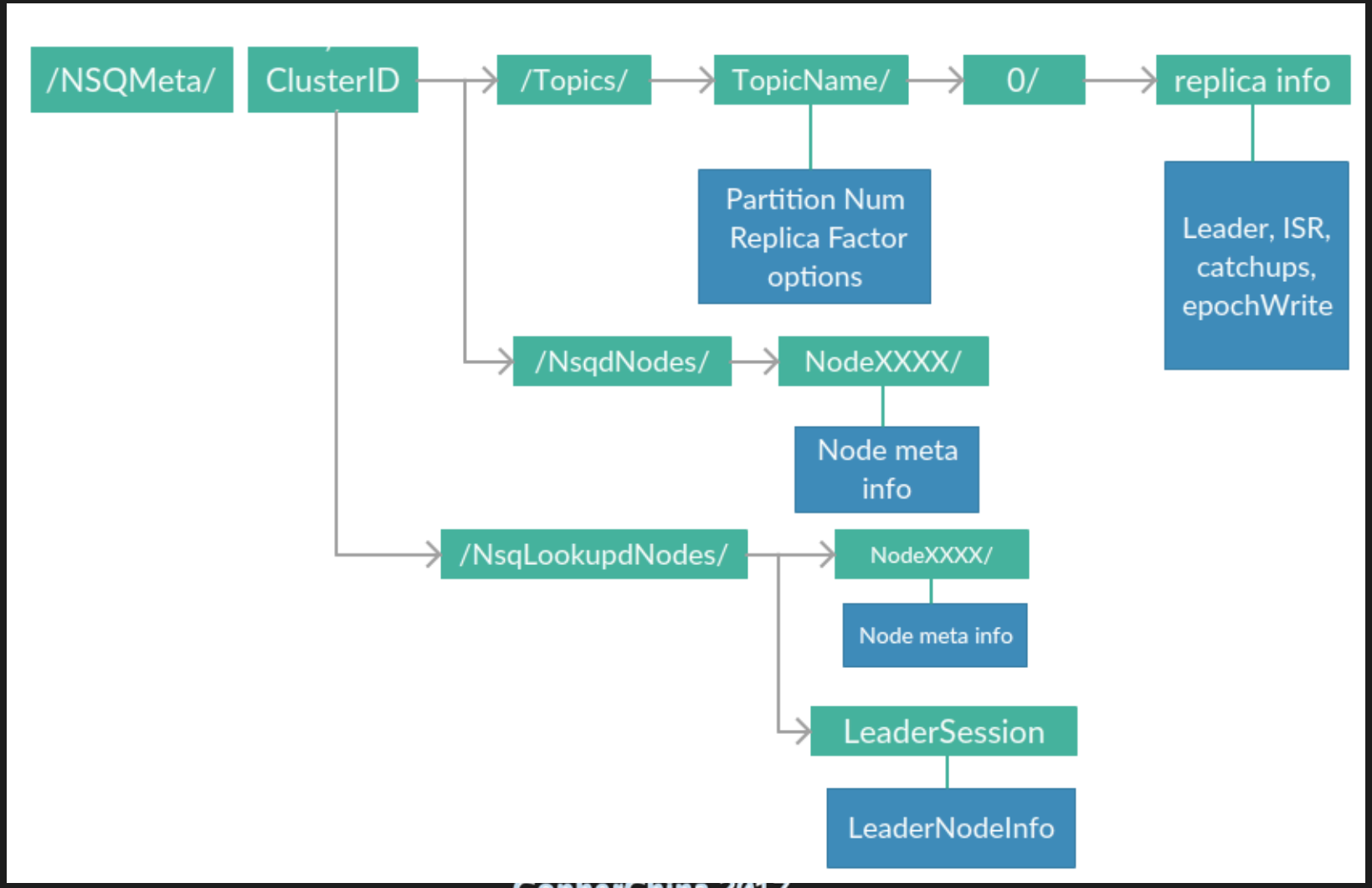
Redesigned Architecture



Architecture Detail

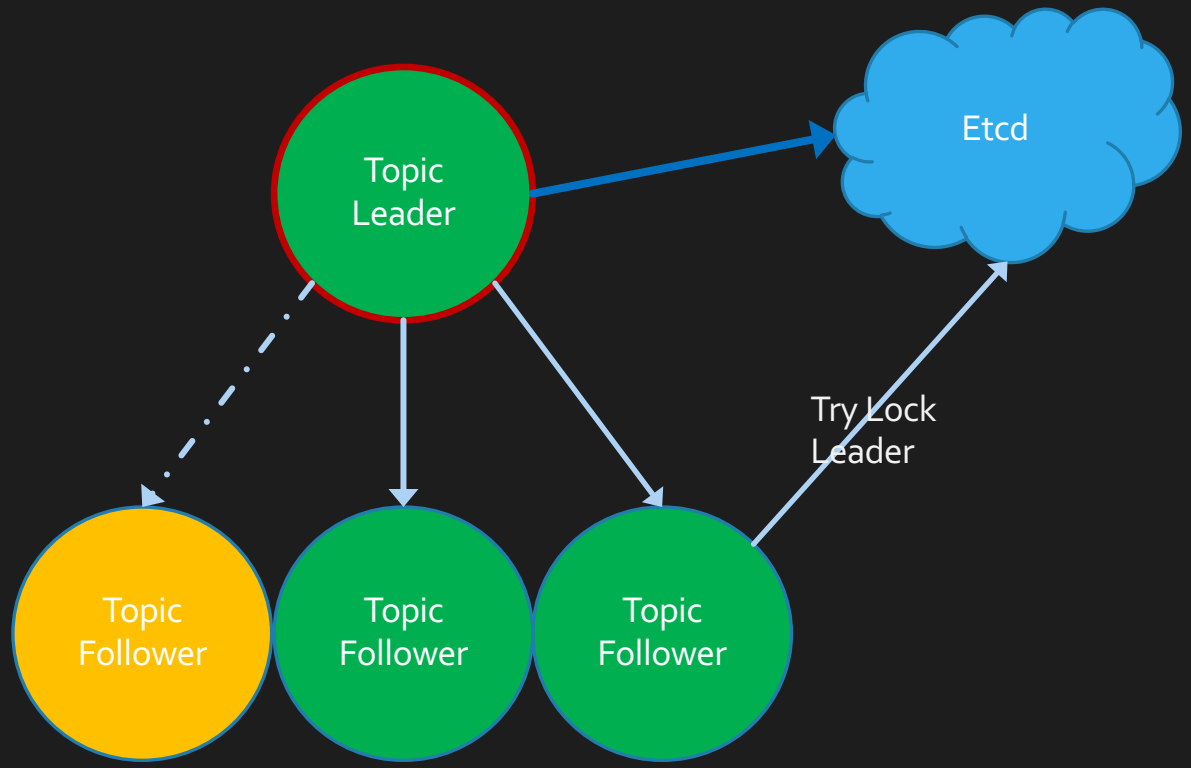
- Etcd: meta data service
- NSQLookup: lookup and placement service
- NSQD: topic data storage service

Meta in etcd


/NSQMeta/
ClusterID
/Topics/
TopicName/
0/
replica info
Partition Num
Replica Factor
options
Leader, ISR,
catchups,
epochWrite
/NsqdNodes/
NodeXXXX/
Node meta
info
/NsqLookupdNodes/
NodeXXXX/
Node meta info
LeaderSession
LeaderNodeInfo

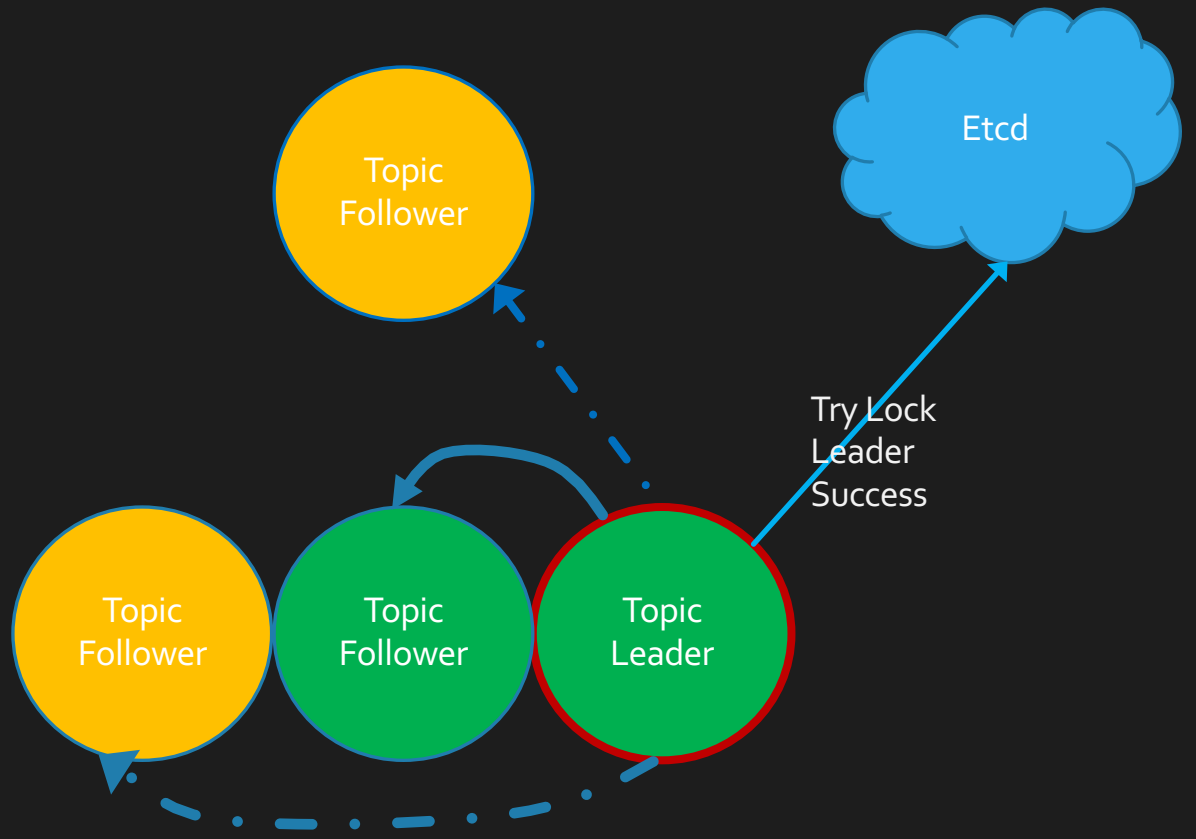
Replication and HA

- Green is ISR (In synced replicas)
- Yellow: Catching up node
- lookup will select the most newest ISR as new leader



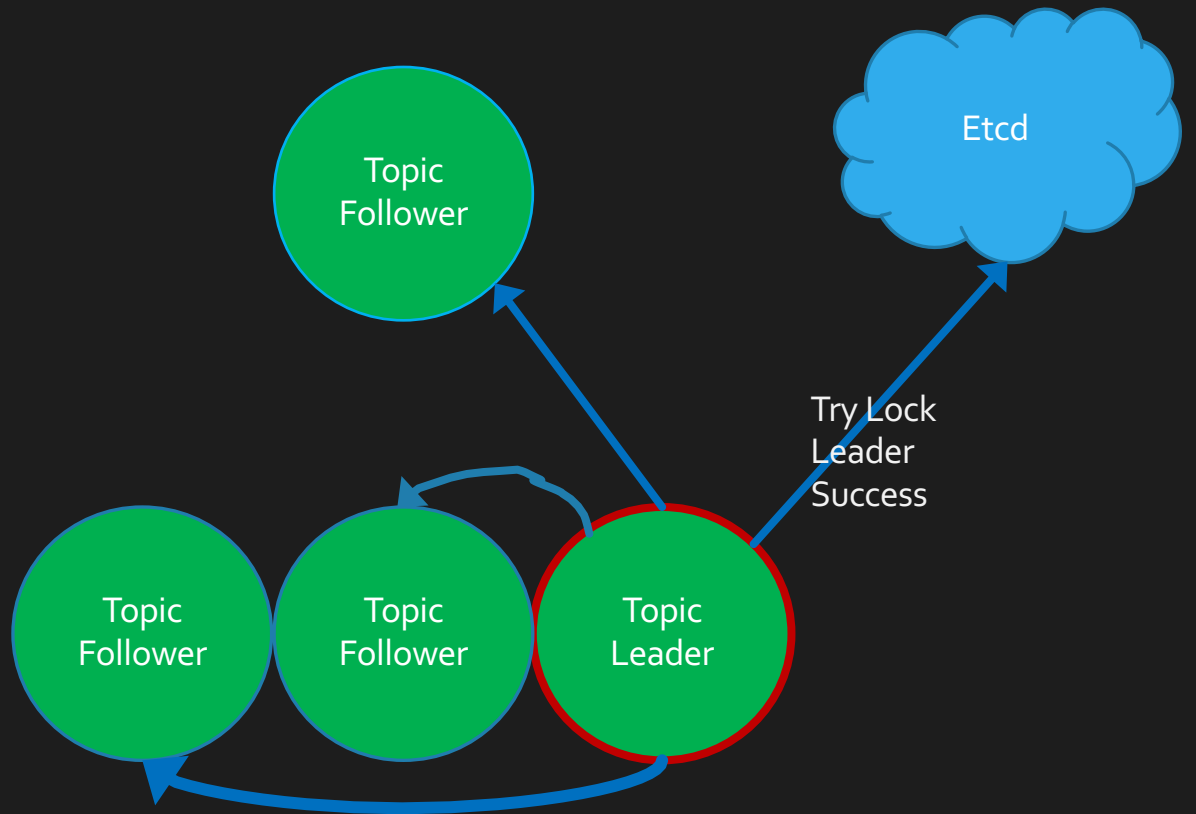
Replication and HA

- Green is ISR (In synced replicas)
- Yellow: Catching up node
- lookup will select the most newest ISR as new leader



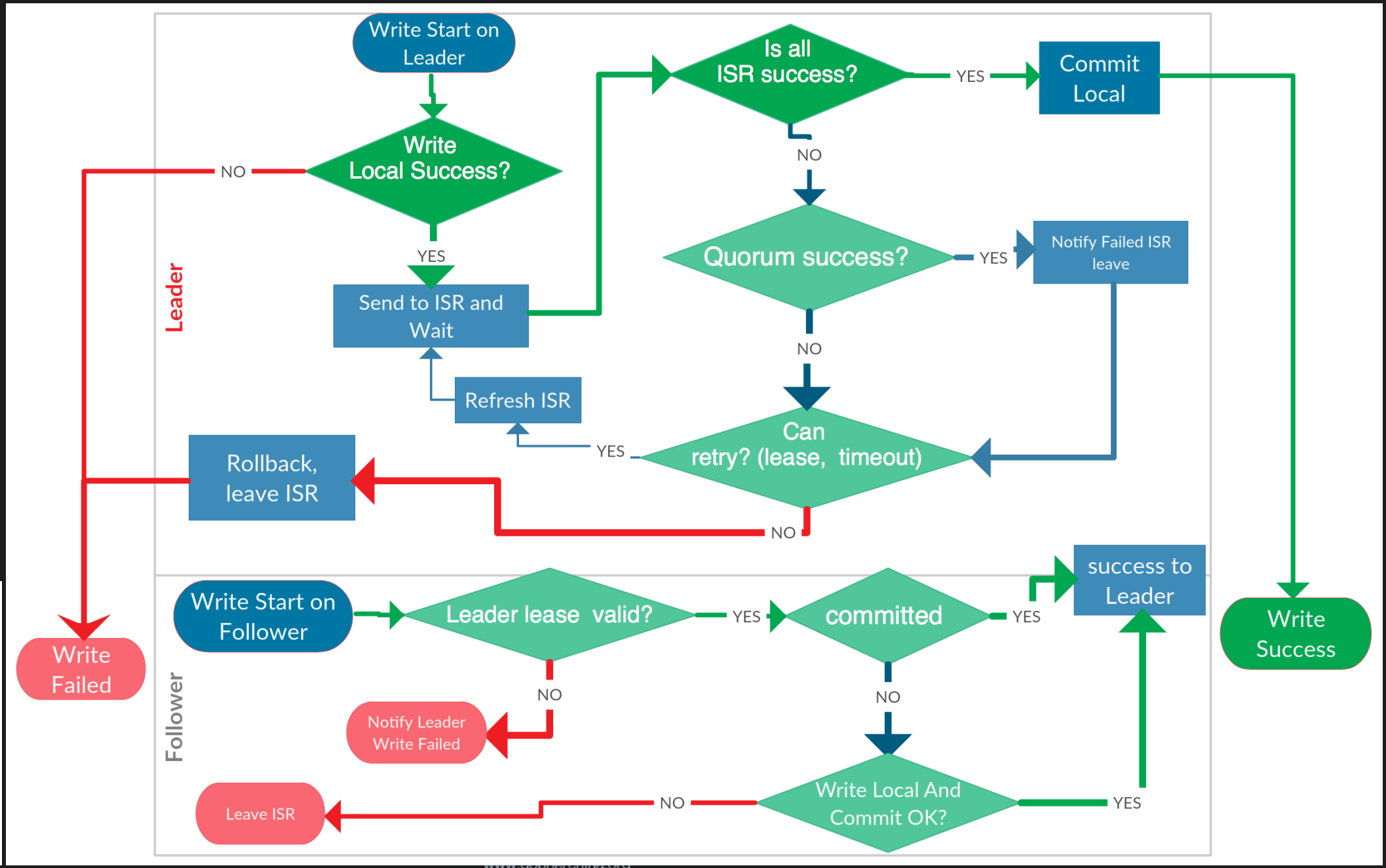
Replication and HA

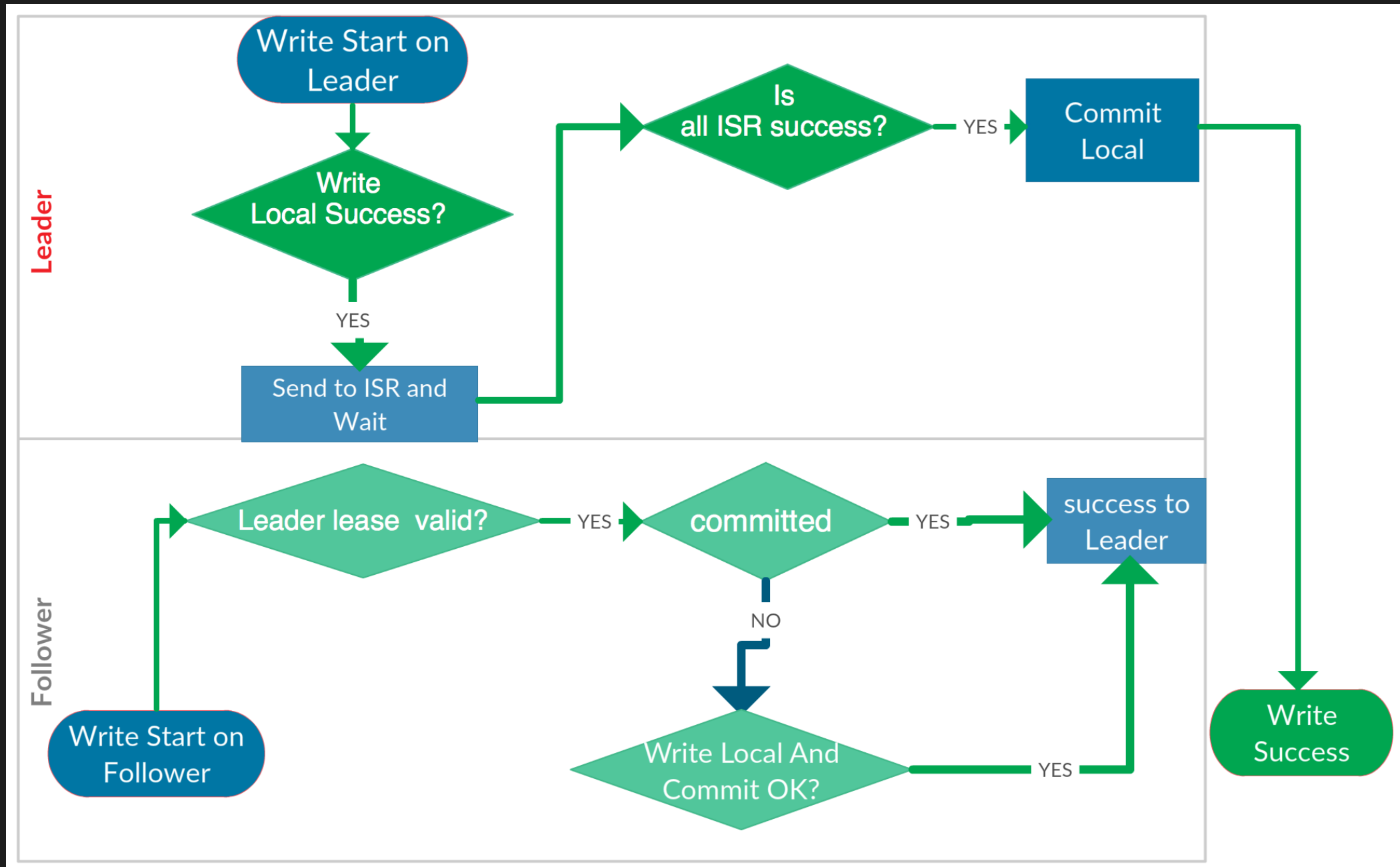
- Green is ISR (In synced replicas)
- Yellow: Catching up node
- lookup will select the most newest ISR as new leader

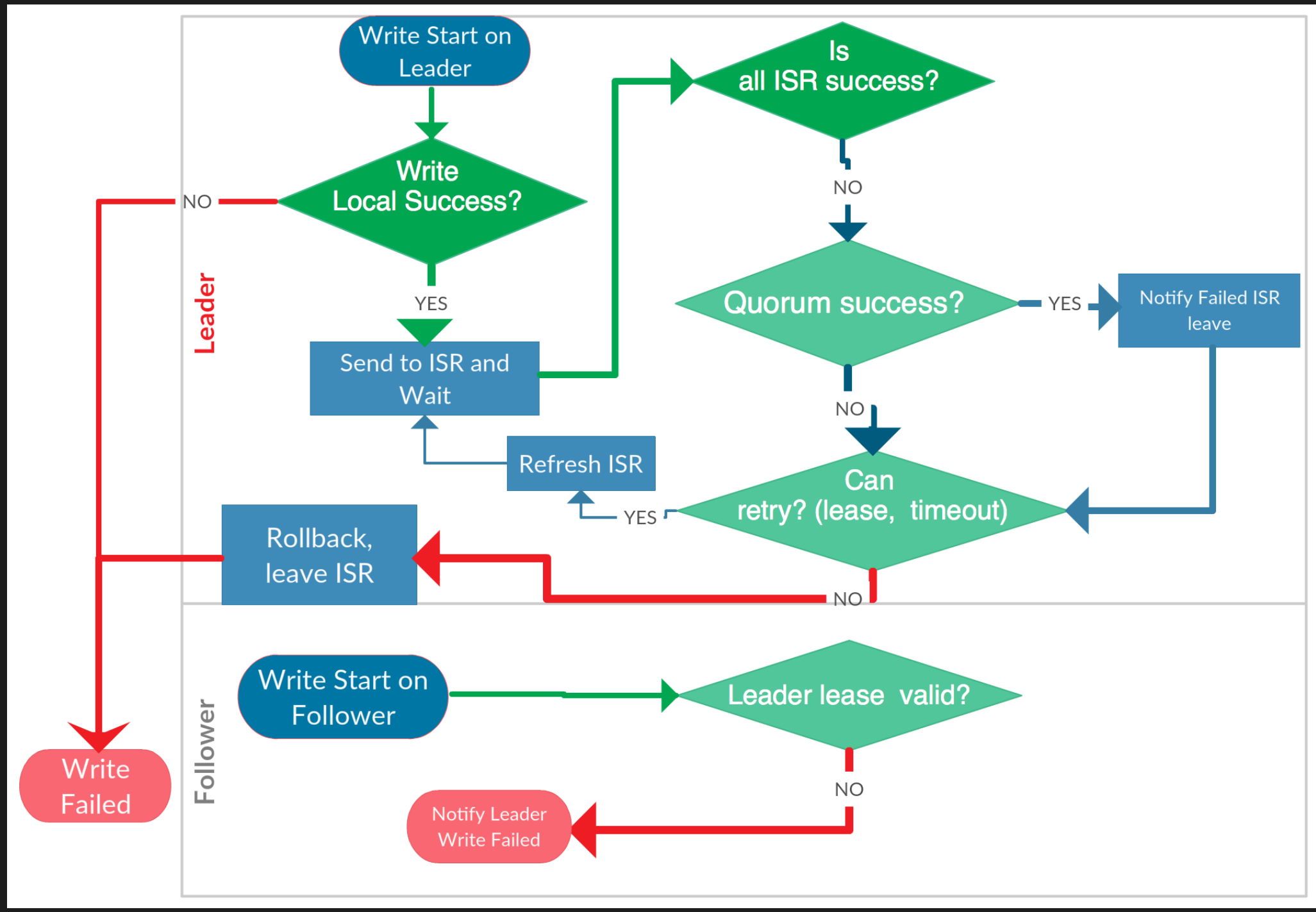


Dynamic ISR

- Avoid slow write due to failed replica
- Add/Remove replica for balance


Write Start on Leader
Write Local Success?
NO
YES
Send to ISR and Wait
Refresh ISR
Is all ISR success?
YES
Commit Local
NO
Quorum success?
YES
Notify Failed ISR leave
NO
Can retry? (lease, timeout)
YES
NO
Rollback, leave ISR
Leader
Write Failed
Write Start on Follower
Leader lease valid?
YES
committed
YES
success to Leader
NO
NO
Notify Leader Write Failed
Write Local And Commit OK?
YES
NO
Leave ISR
Follower
Write Success

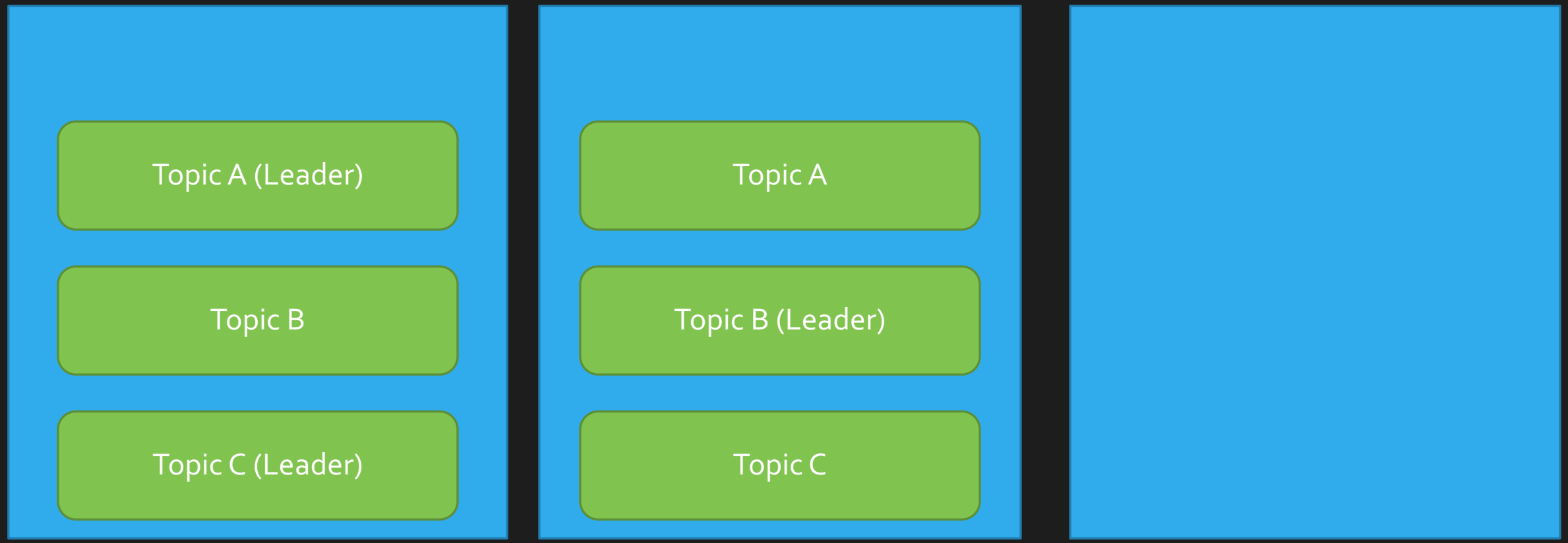

Leader
Write Start on Leader
Write Local Success?
YES
Send to ISR and Wait
Is all ISR success?
YES
Commit Local
Follower
Write Start on Follower
Leader lease valid?
YES
committed
YES
NO
Write Local And Commit OK?
YES
success to Leader
Write Success


Write Start on Leader
Write Local Success?
NO
YES
Leader
Send to ISR and Wait
Is all ISR success?
NO
Quorum success?
YES
Notify Failed ISR leave
NO
Refresh ISR
YES
Can retry? (lease, timeout)
Rollback, leave ISR
NO
Write Failed
Follower
Write Start on Follower
Leader lease valid?
NO
Notify Leader Write Failed

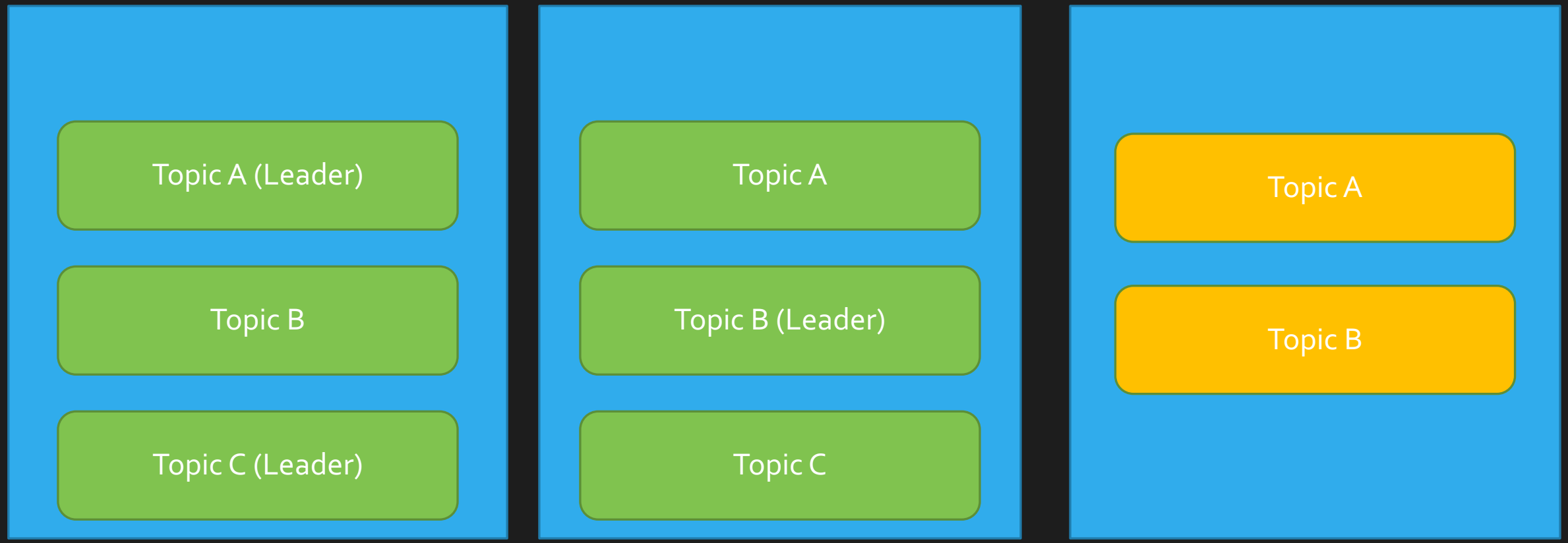
Auto Balance

- Load Factor: CPU load, Topic PUB size, Leaders
- Add/remove node balance
- Manual move data

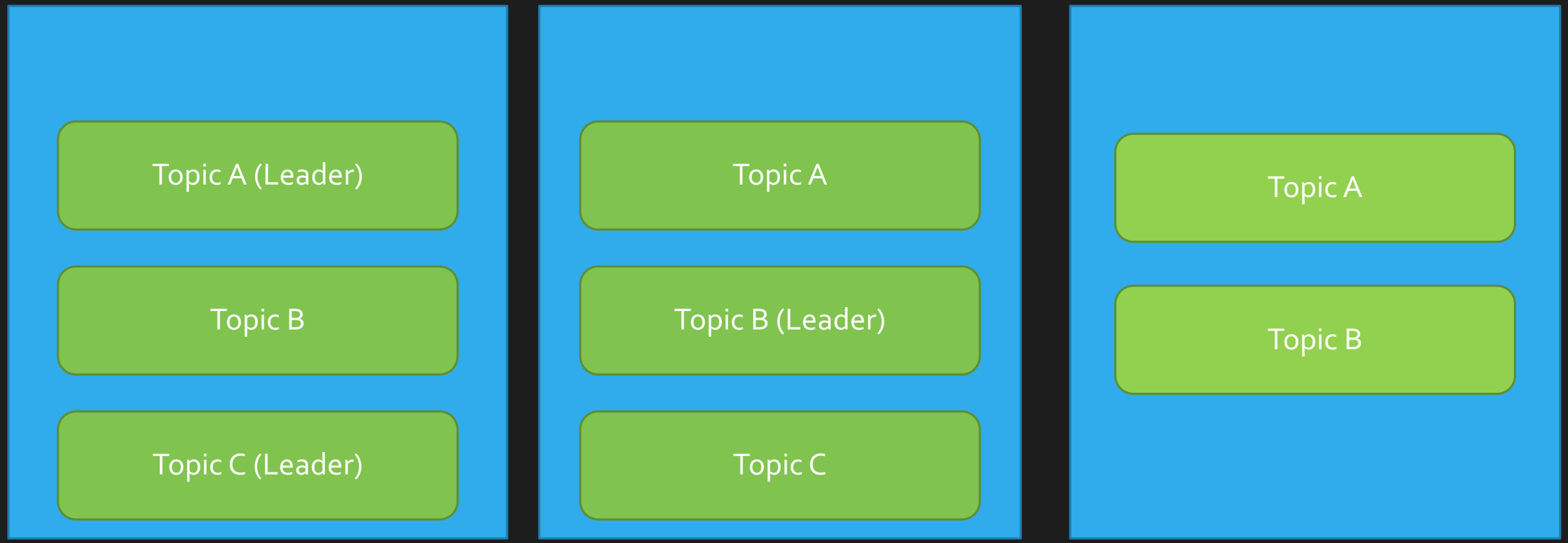
Auto Balance



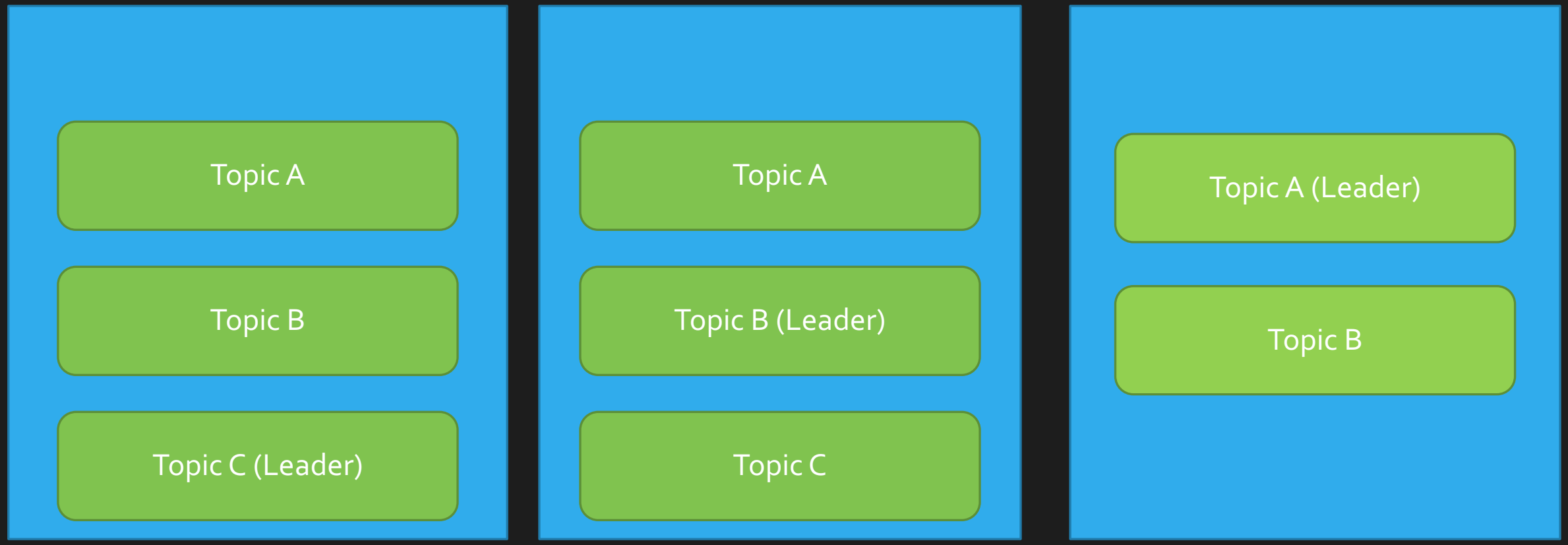
Auto Balance



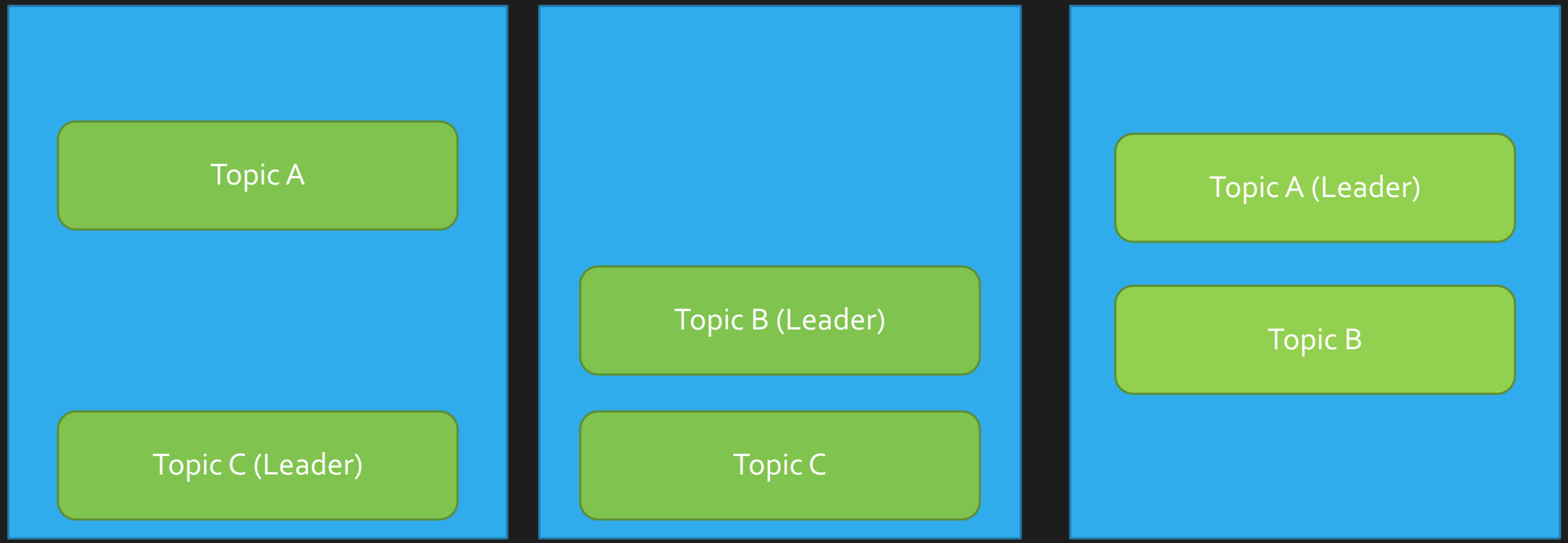
Auto Balance



Auto Balance



Auto Balance

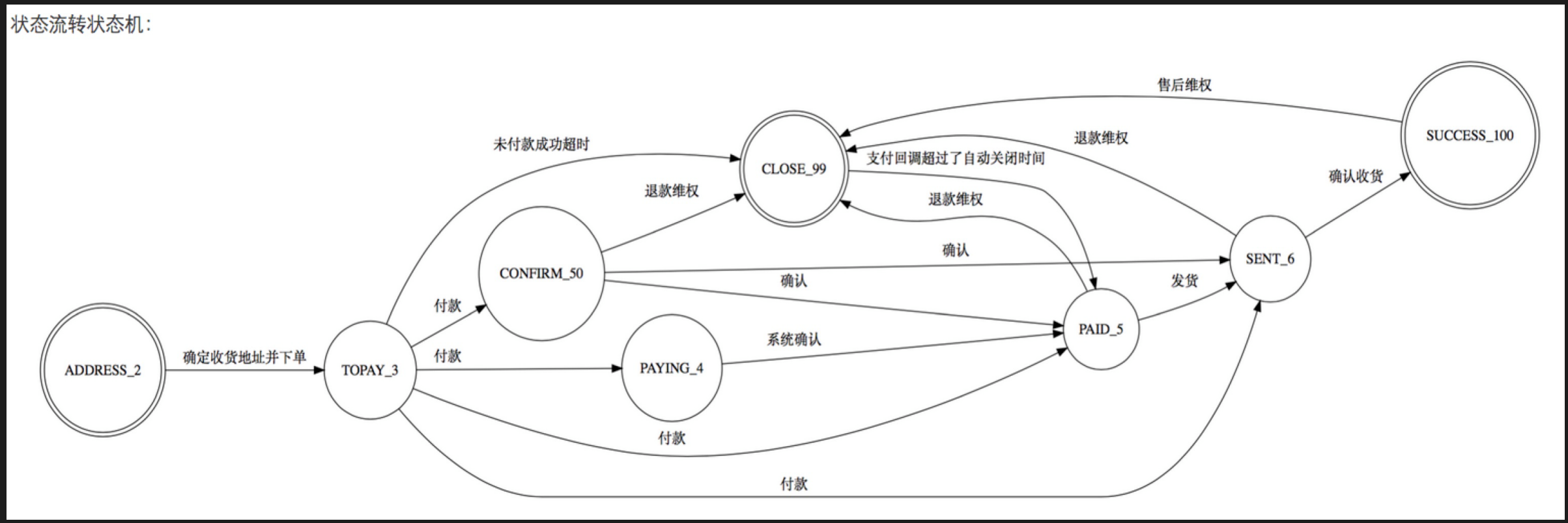


Smart Client

- Period refresh topic leader and partitions
- Retry on HA: query lookup quickly if failed on not leader
- Pub strategy: support round robin and sharding with primary key.

Delivery In Order

状态流转状态机：

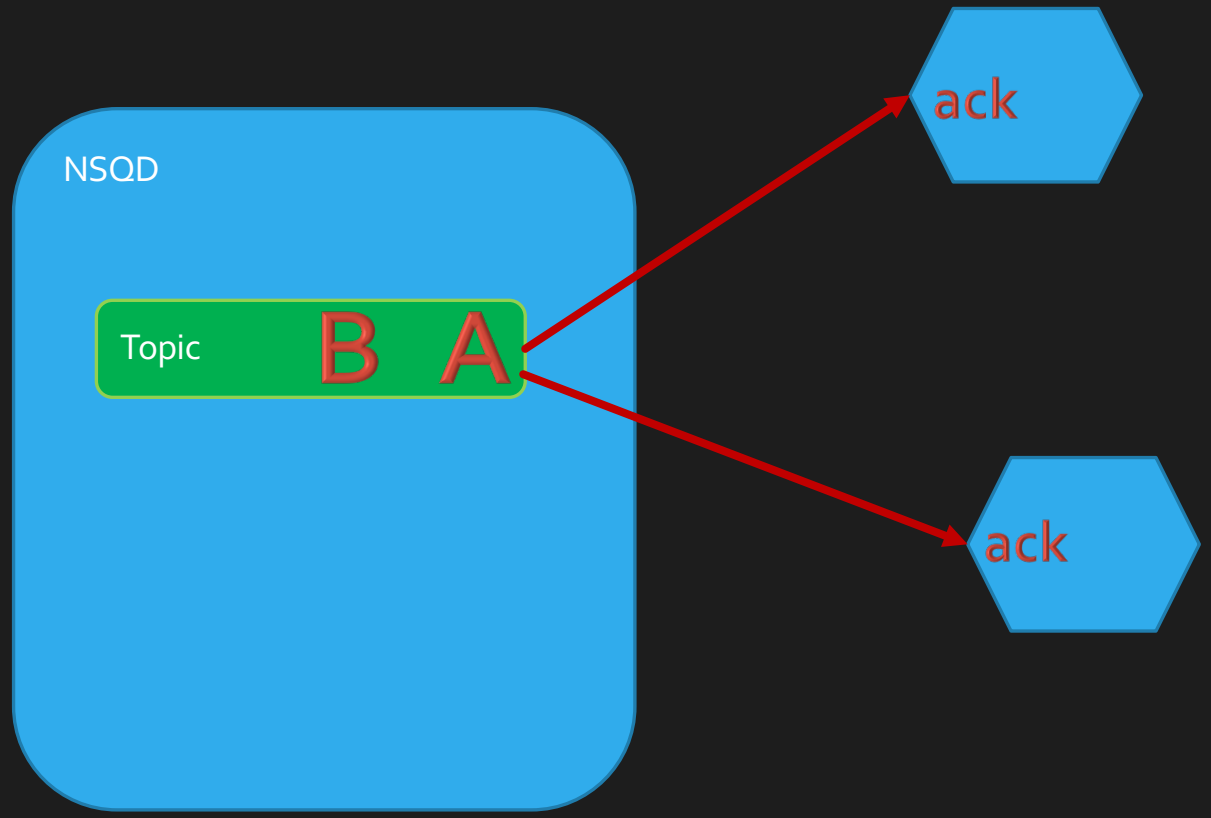


Delivery in Order

- Ordered in the same Partition
- PUB the same sharding key on the same node in single thread
- Deliver messages to the consumer client in order one by one

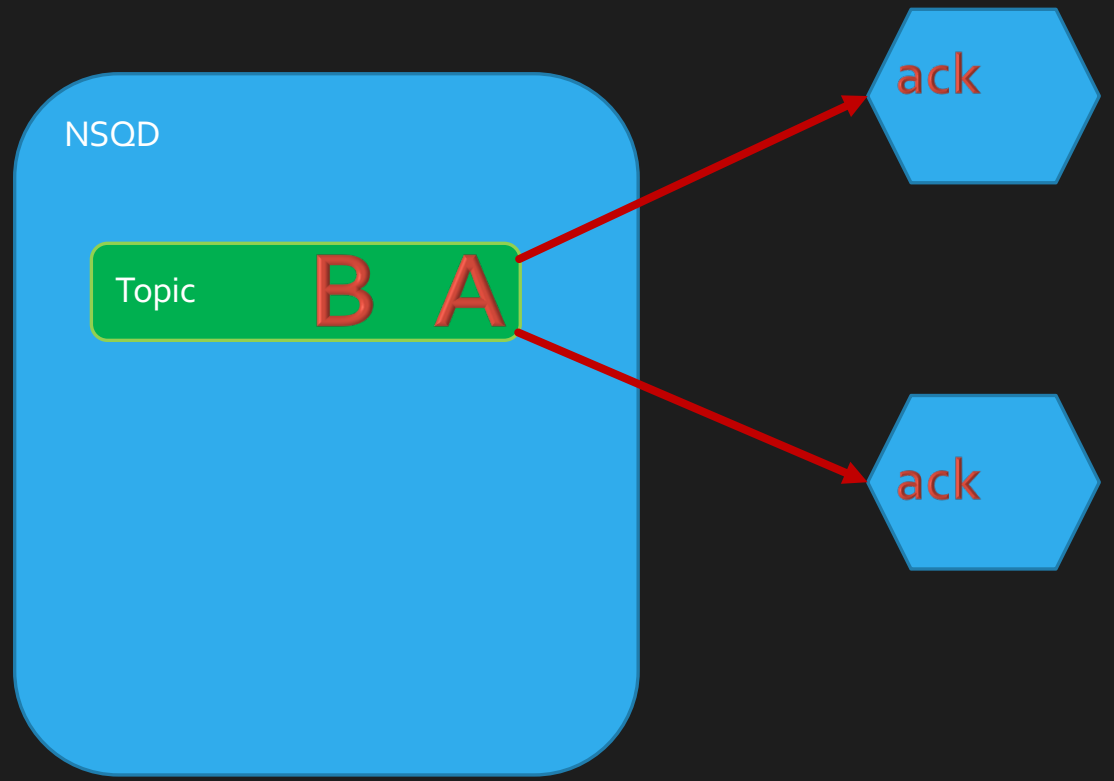
Delivery in Order

Out of order

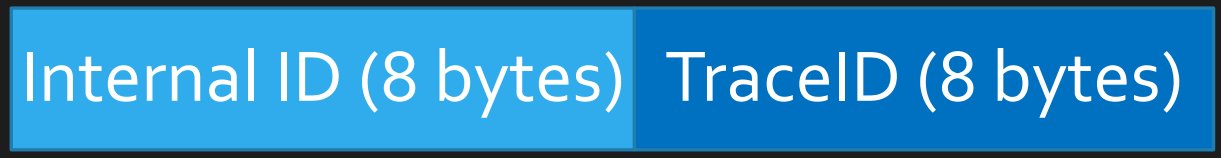


Delivery in Order

In order

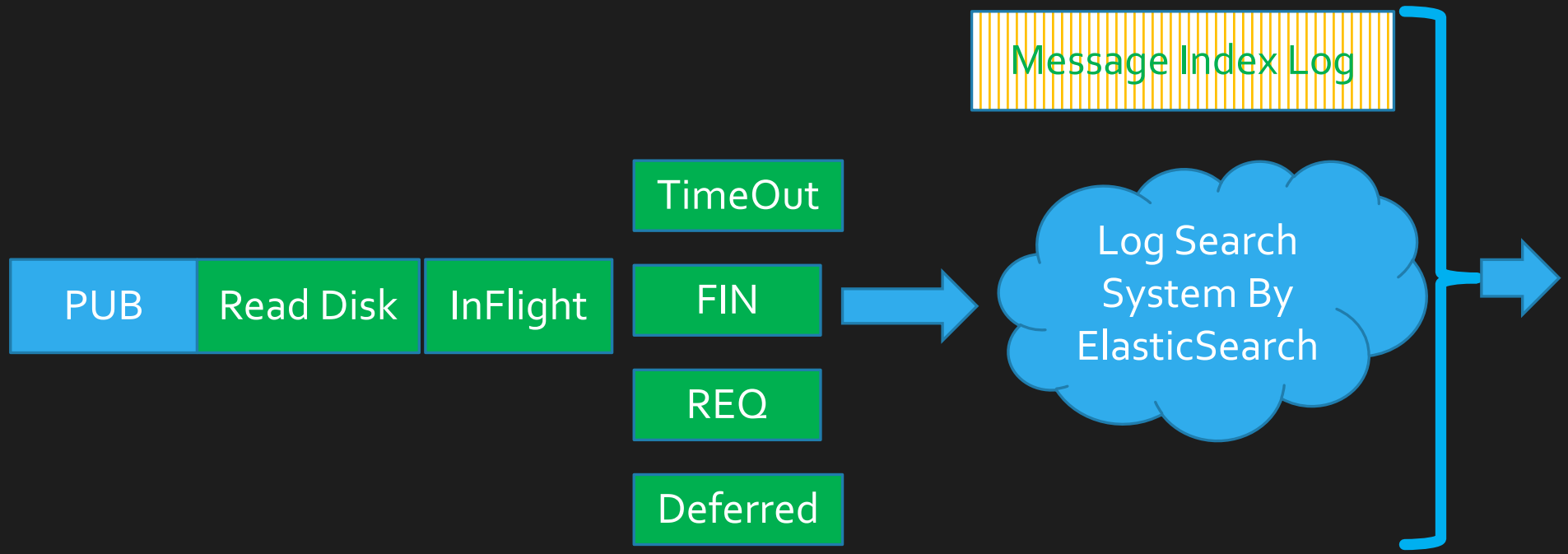


Tracing



Connect NSQ message with the business

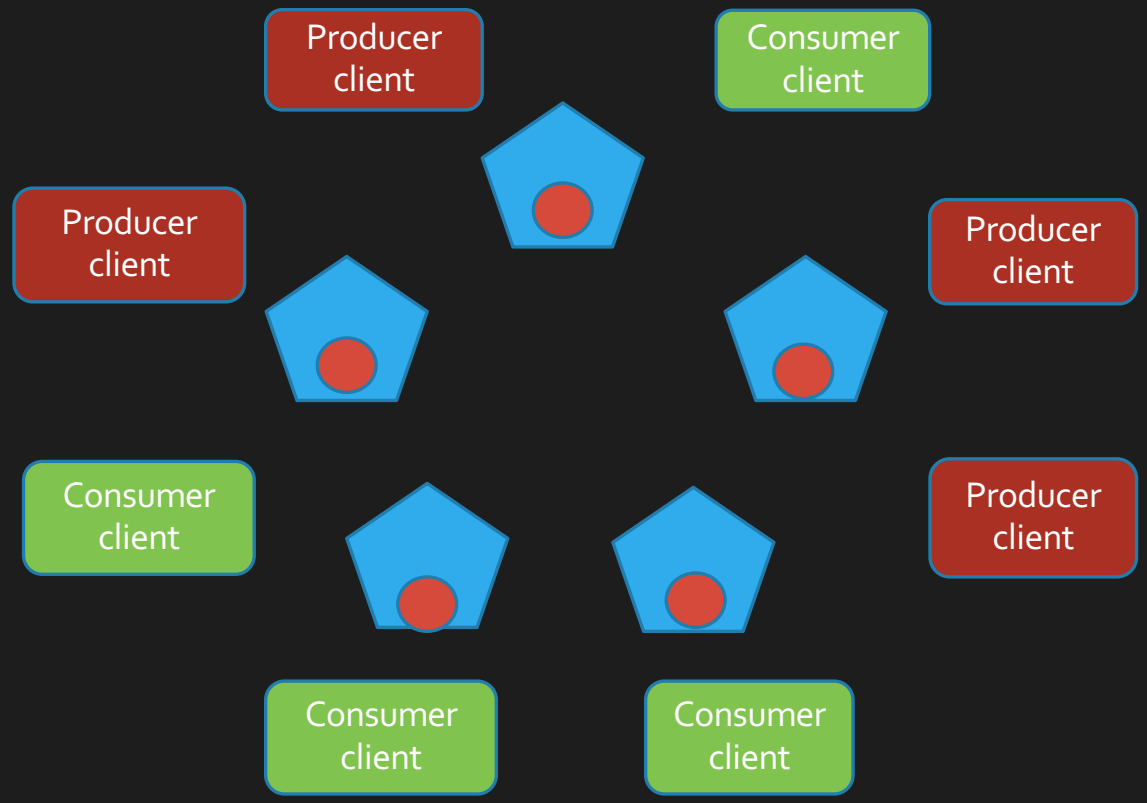
Tracing


Message Index Log
TimeOut
PUB
Read Disk
InFlight
FIN
REQ
Deferred
Log Search
System By
ElasticSearch

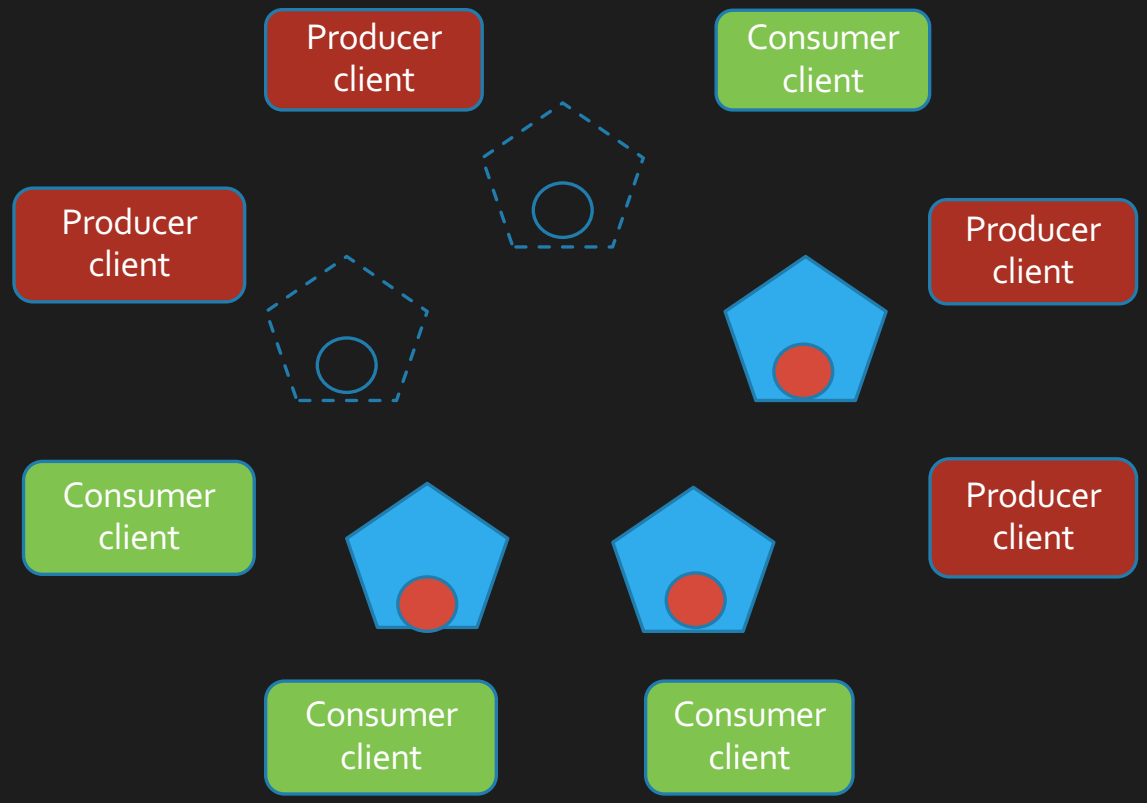
Jepsen test in new NSQ

- Jepsen: distributed system test tool
- How: make network partition (brain-split)

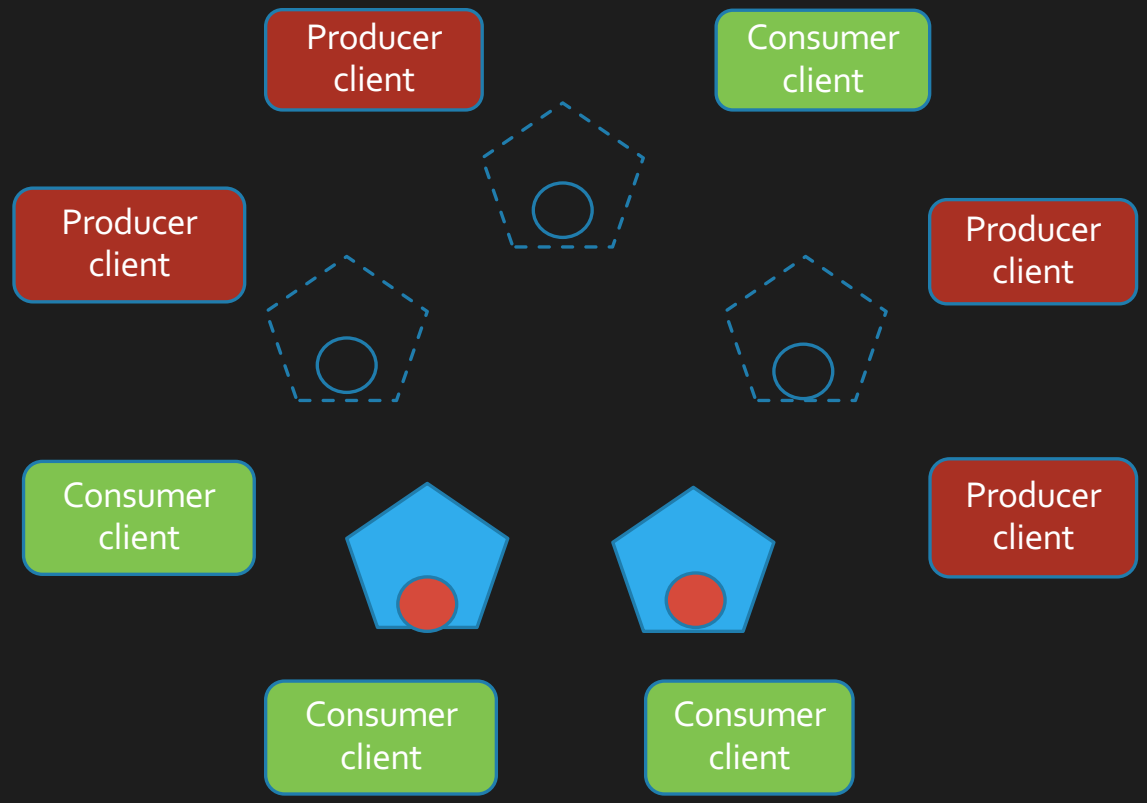
Jepsen test in new NSQ



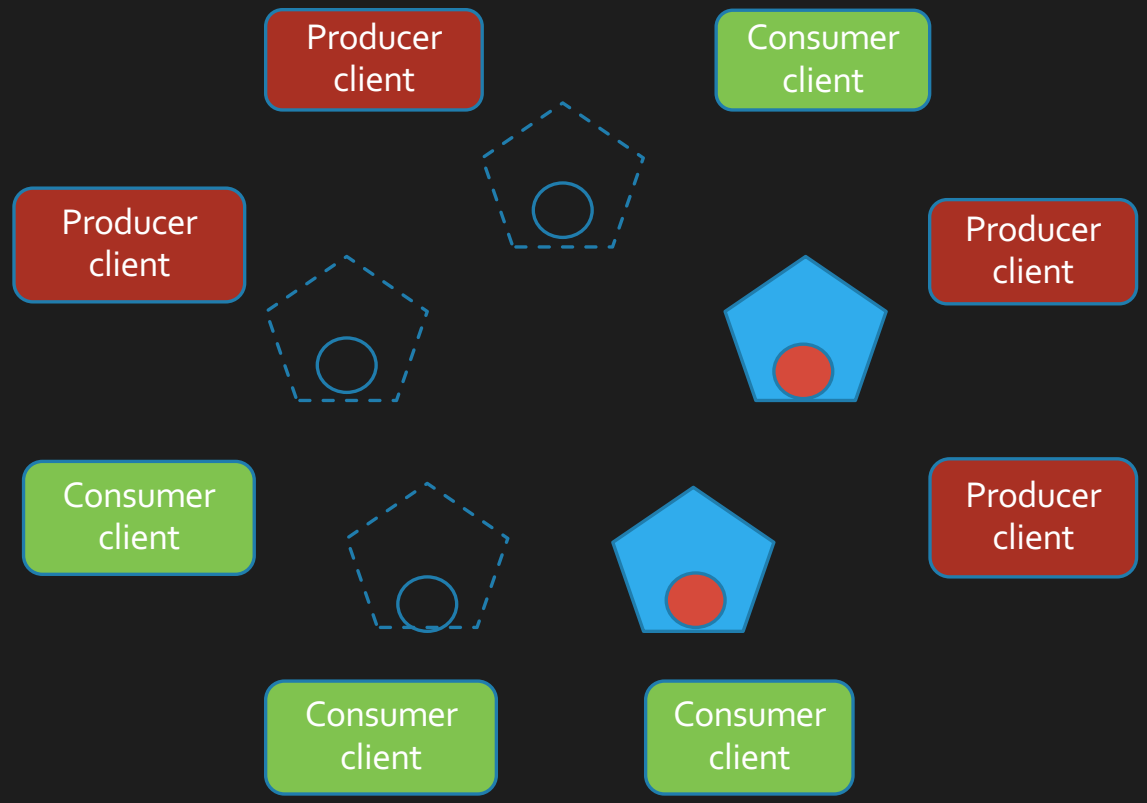
Jepsen test in new NSQ



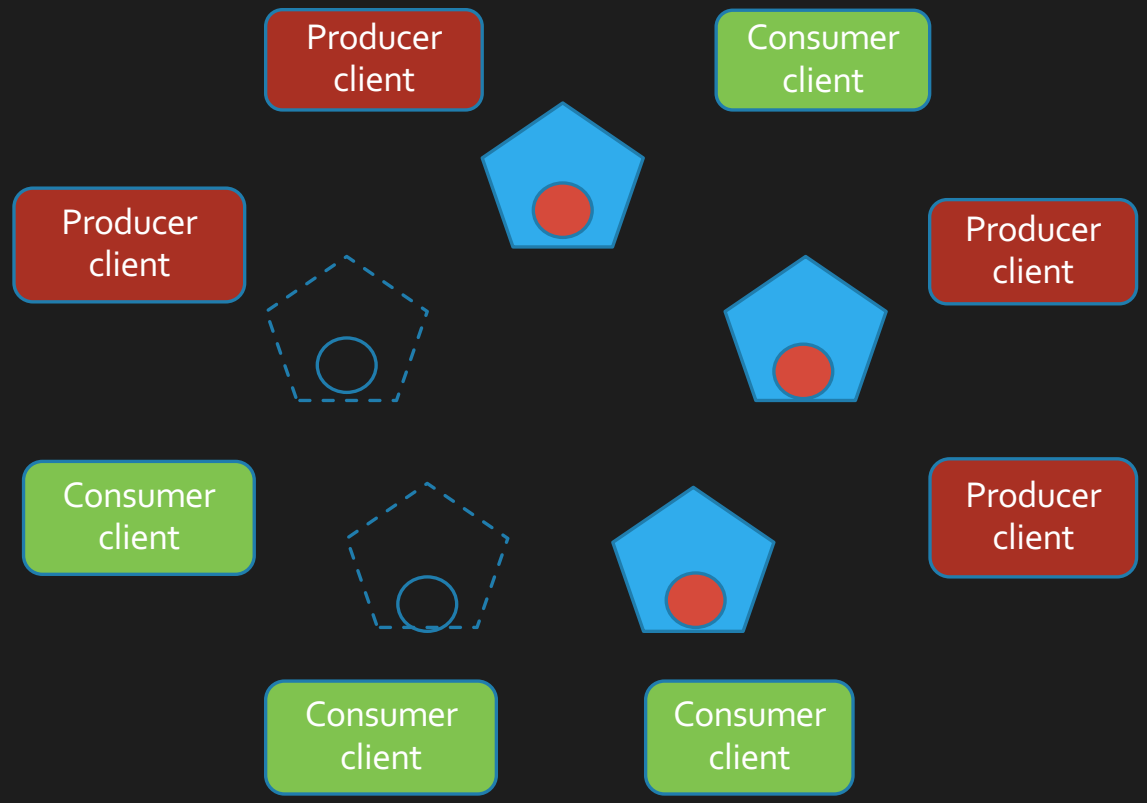
Jepsen test in new NSQ



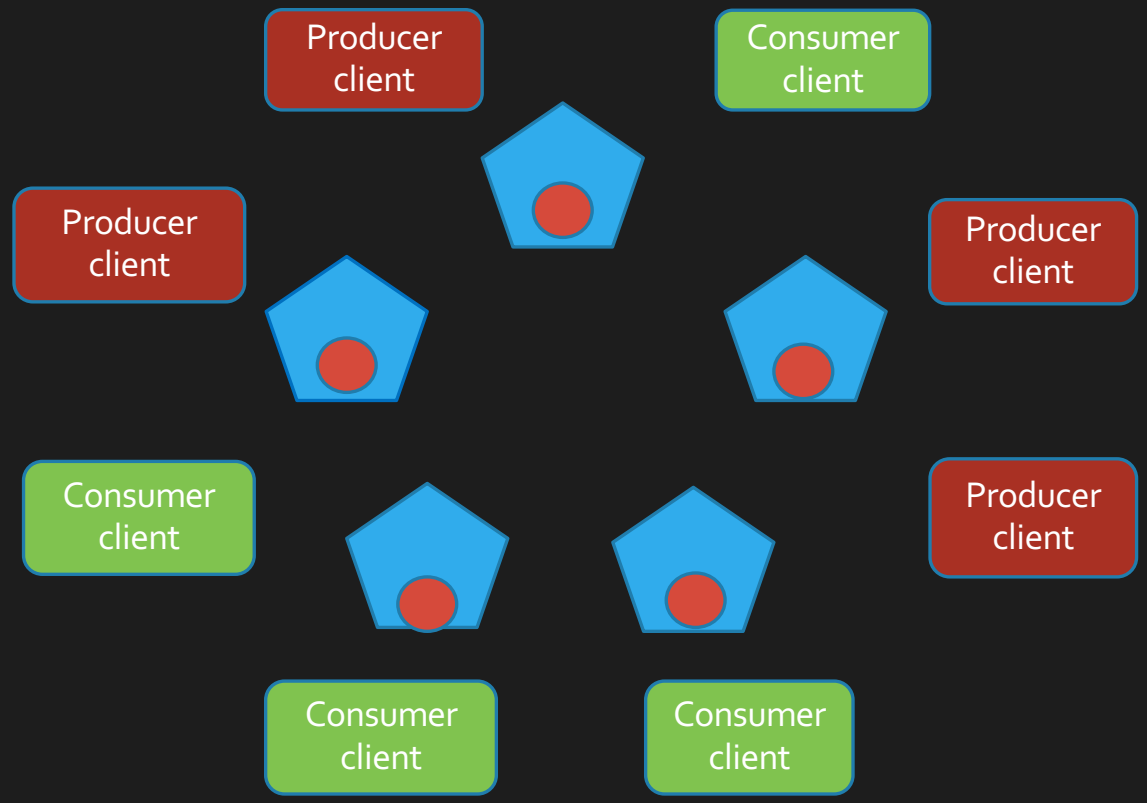
Jepsen test in new NSQ



Jepsen test in new NSQ



Jepsen test in new NSQ



Jepsen test in new NSQ

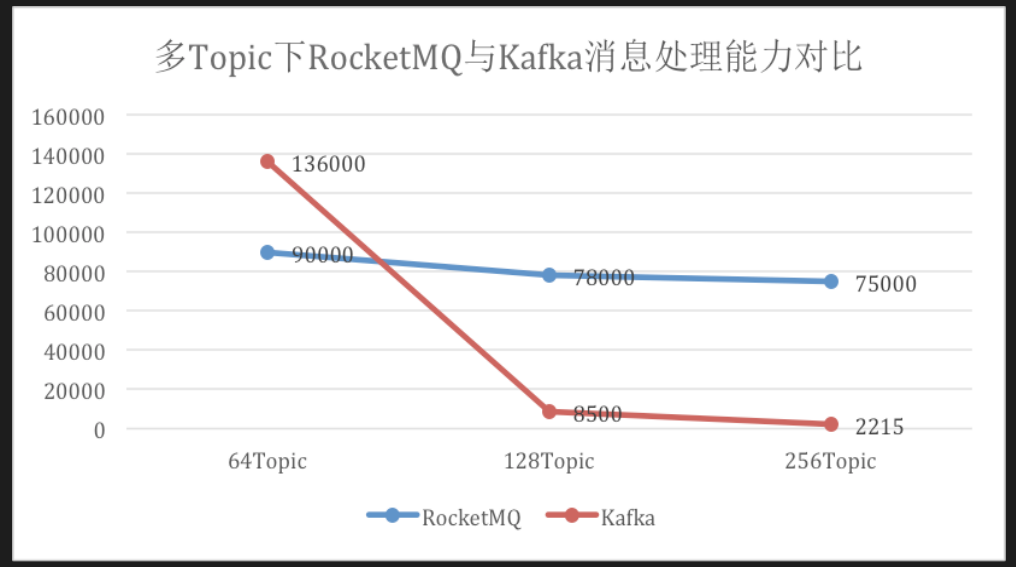
```
INFO [2017-03-17 12:44:38,032] main - jepsen.core {:latency {:valid? true},
 :total-queue
 {:valid? true,
  :lost #{},
  :recovered #{256 257 1728 255 254},
  :recovered-frac 5/2006,
  :unexpected-frac 0,
  :unexpected #{},
  :lost-frac 0,
  :duplicated-frac 71/2006,
  :ok-frac 1,
  :duplicated
  #{1976 1774 1909 1649 1722 1643 1913 1983 1781 1639 1648 1629 1997
    1912 1804 1721 1985 1778 1987 1987 1911 1996 1773 1773 1625 1627
    1995 1977 1989 1626 1624 1636 1800 1815 1650 1772 1998 1992 258
    1730 1807 1993 1680 1990 1994 1728 1910 1646 1986 1628 1816 1678
    1678 1679 1679 1799 1775 1789 254 1723 1975 1819 1637 1817 1640
    1818 1818 1818 1818 1818 1991}},
 :valid? true}


Everything looks good! ヽ(‘ー`)ノ
Report written to report/queue.txt

Ran 1 tests containing 1 assertions.
0 failures, 0 errors.
```

Performance and Data in Youzan

- 4 nodes test cluster (24-cores, 64GB Mem, HDD)
- 500k/s pub (50-bytes message, 32 partitions, 2 replicas)
- 900k/s pub (50-bytes, 4 partitions, 1 replica)
- 200k/s sub (50-bytes message, 4 partitions)
- 150 Billion messages processed until now in real production cluster



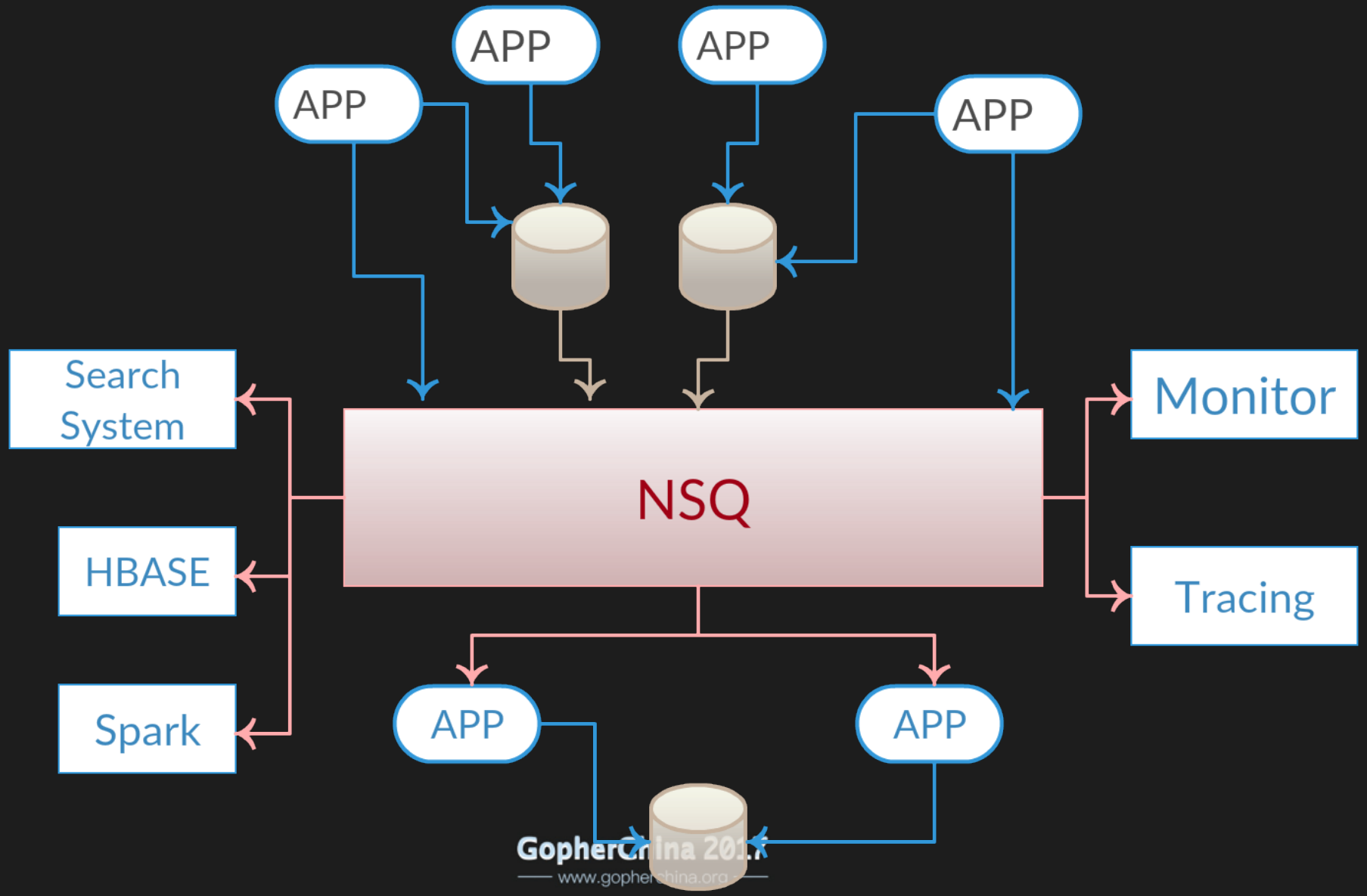
Compare

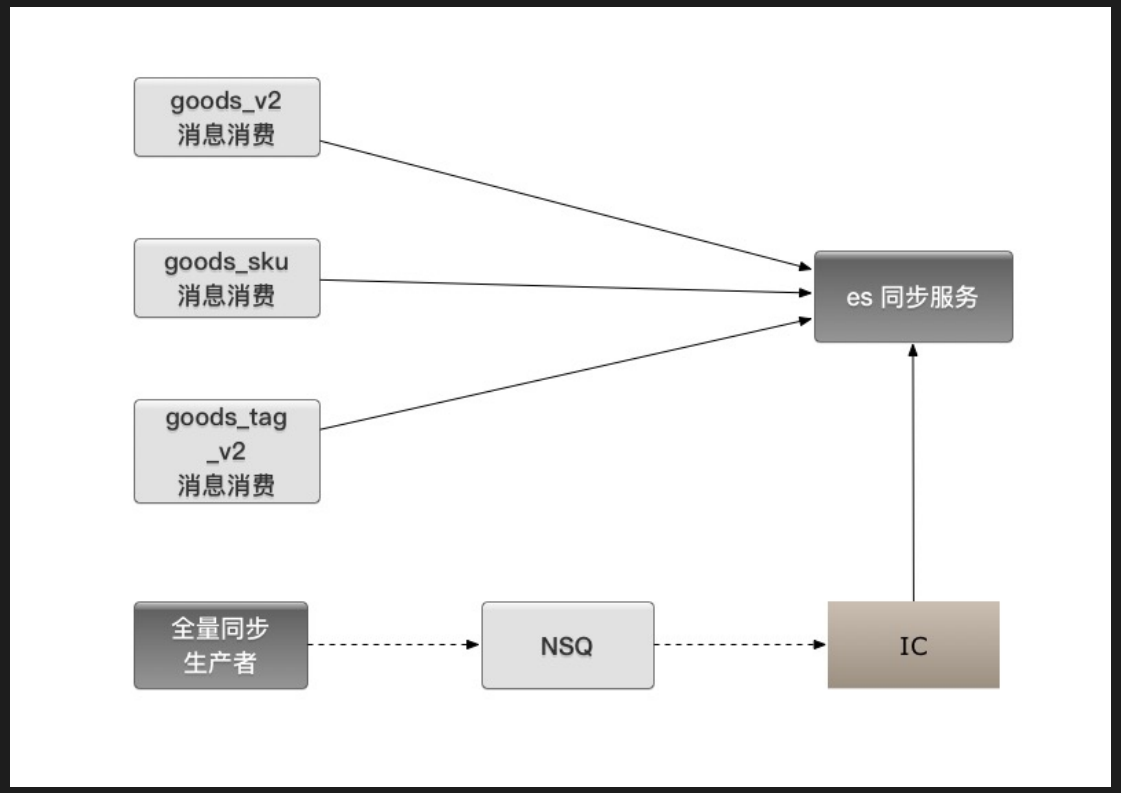
	开源生态	数据可靠性	多语言客户端	性能	灵活性	严格顺序	堆积能力	消息查询跟踪
Kafka	*****	**	**	***	****	*	****	*
RocketMQ	**	****	**	****	**	****	***	****
NSQ (redesigned)	***	****	***	****	****	*****	****	****

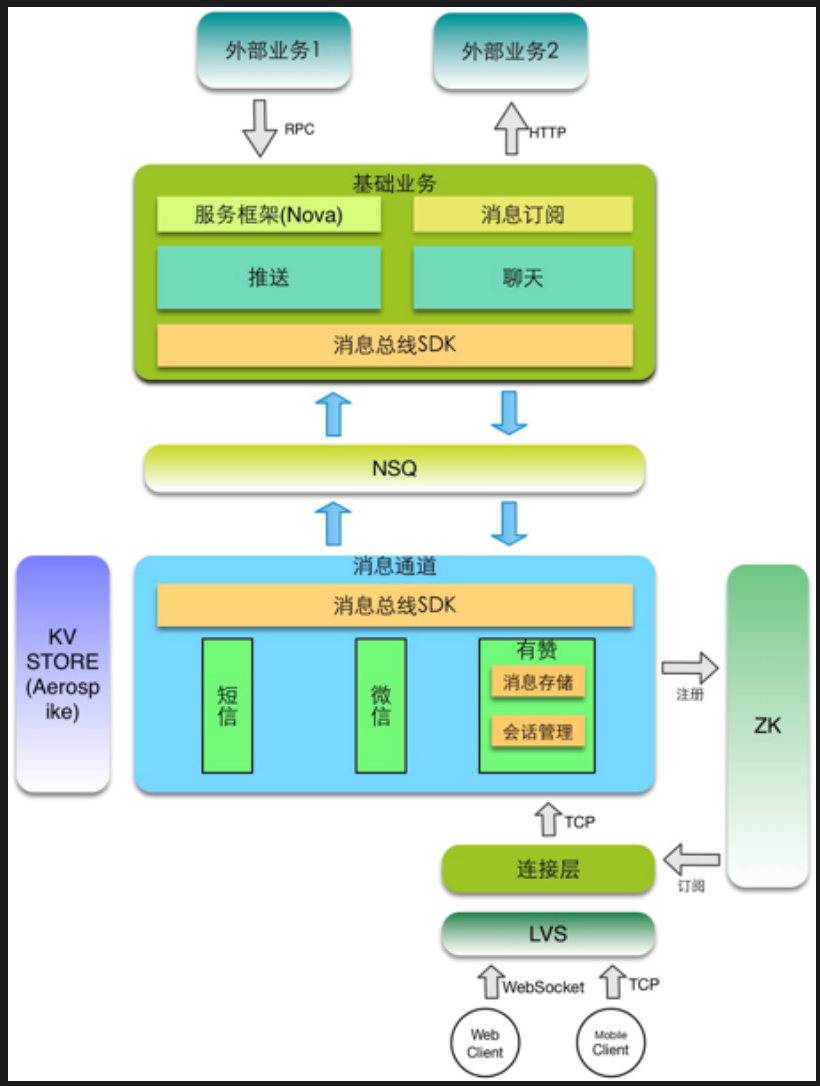
Migrate data in Youzan

- Consumer configure with both nsqlookup
- Producer configure with the new nsqlookup only
- Using Configure server to make it easy enough

Usage in Youzan




goods_v2
消息消费
goods_sku
消息消费
goods_tag
_v2
消息消费
es 同步服务
全量同步
生产者
NSQ
IC


外部业务1
外部业务2
RPC
HTTP
基础业务
服务框架(Nova)
消息订阅
推送
聊天
消息总线SDK
NSQ
消息通道
消息总线SDK
KV
STORE
(Aerosp
ike)
短
信
微
信
有赞
消息存储
会话管理
注册
ZK
TCP
连接层
订阅
LVS
WebSocket
TCP
Web
Client
Mobile
Client

NSQ in Youzan

- 发放奖励
- 销量实时统计
- 付款同步
- 发货同步
- 退款维权同步
- 状态同步到粉丝
- 异步缓存更新
- 排队下单
- ... .

Open source @Github

- Golang SDK: https://github.com/absolute8511/go-nsq
- Java SDK: https://github.com/youzan/nsqJavaSDK
- PHP SDK: https://github.com/youzan/php-nsq-client
- Spark connector: https://github.com/youzan/spark-nsq-consumer
- Flume connector: https://github.com/DoraALin/flume-nsq-sink
- NSQ Server: https://github.com/absolute8511/nsq
- More coming: https://github.com/youzan

Review the Redesign

- Channel queue -> disk file segments
- Consumer data copy -> consumer cursor
- No replication -> Replication & HA & Balance
- More : Consume in Order, Consume history, Tracing, Jepsen test.
- Most important: Keep protocol compatible

谢谢! Thanks

提问和答疑入口 Q&A