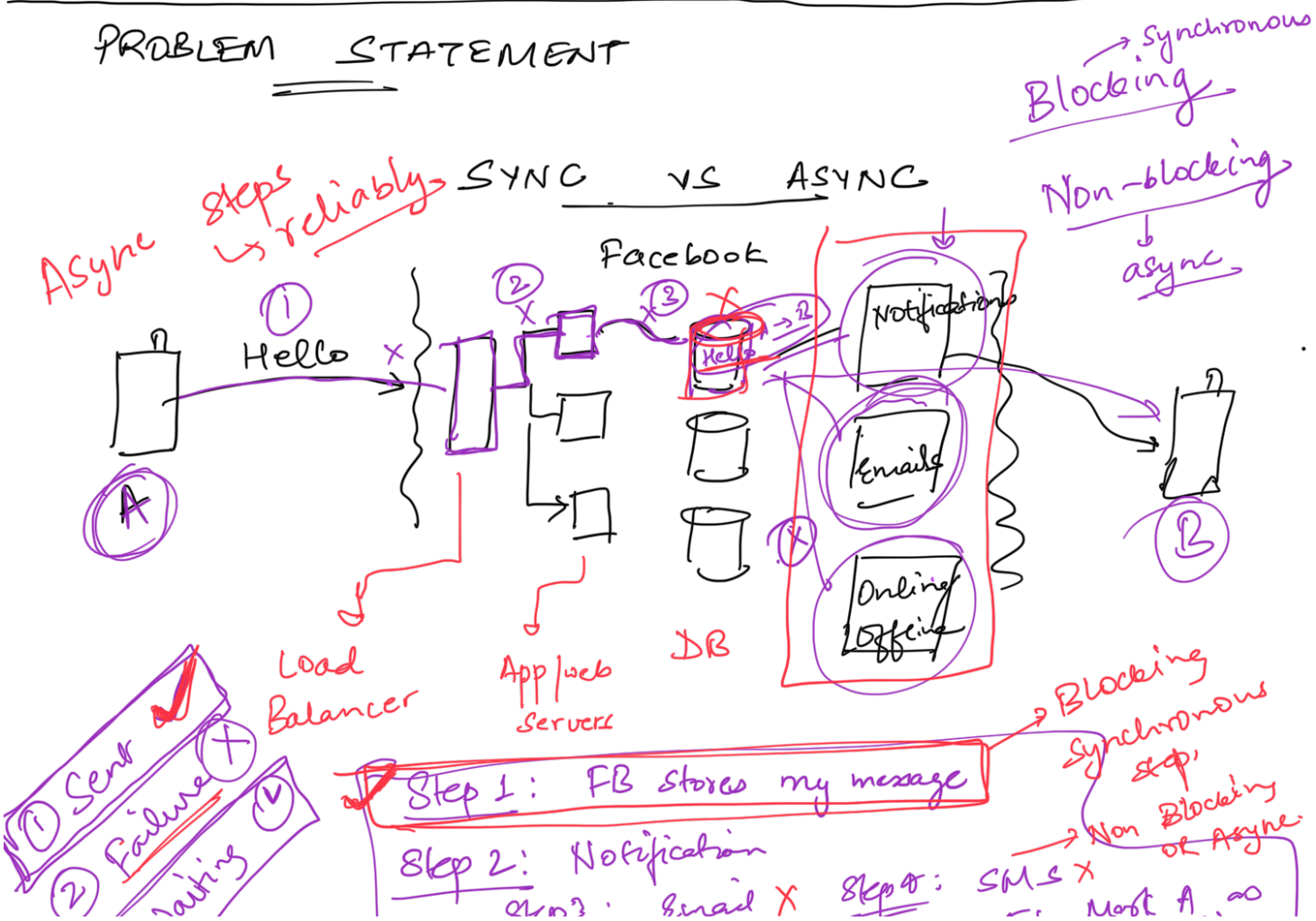
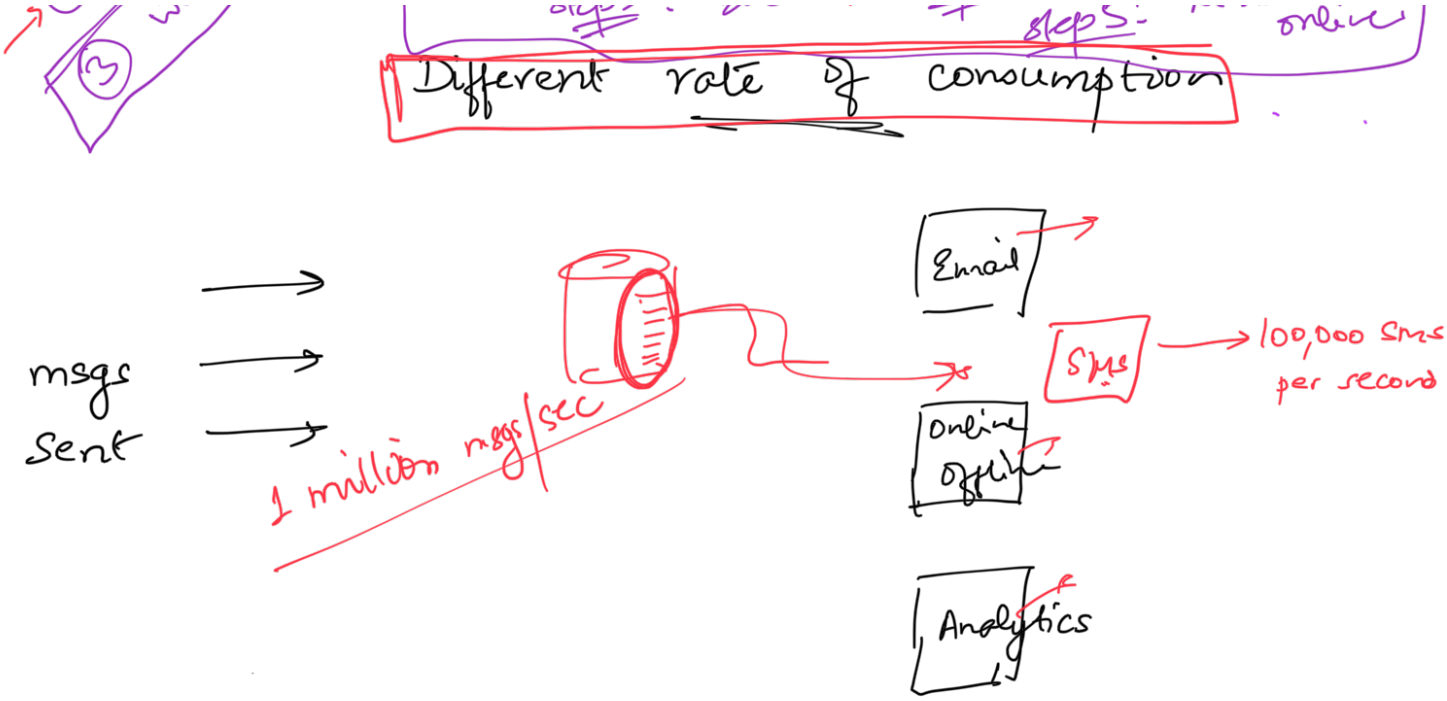
ARCHITECTURE OF KAFKA & ZOOKEEPER

1. INTRO
2. PROBLEM STATEMENT
3. STEP BY STEP :
   - RESILIENCY
   - SCALING
   - SCALING A TOPIC
4. STATE MANAGEMENT USING ZOOKEEPER / KRAFT

FAQ
- Q&A AT THE END
- SESSION IS LIVE.
- 2.5 HOUR SESSION

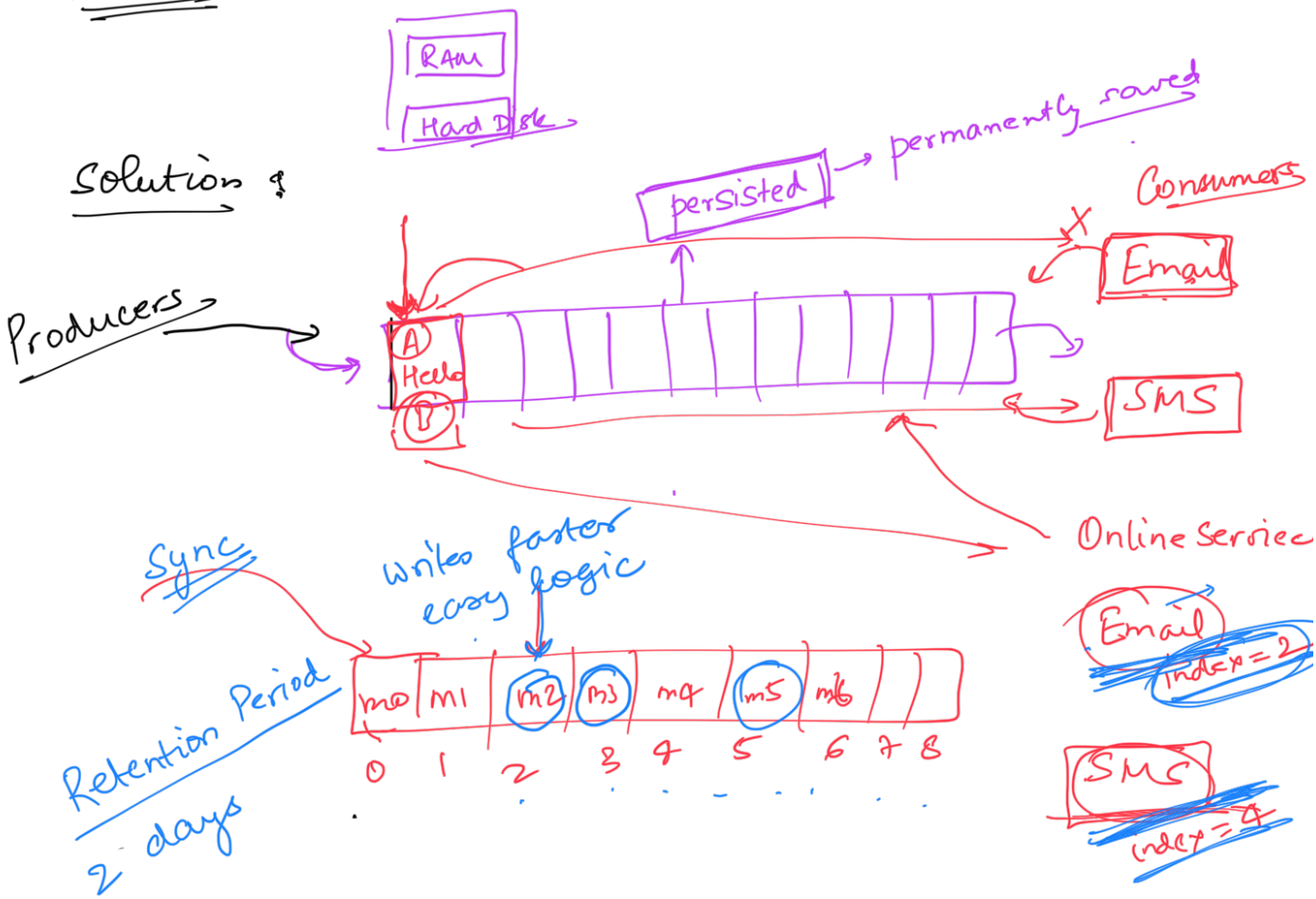
PROBLEM STATEMENT

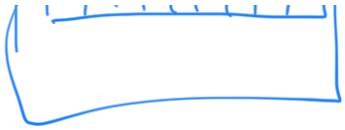




Problem :

Solution :



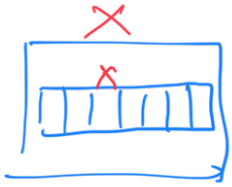


Example of messaging queues:

- Kafka ✓
- RabbitMQ ✓
- SQS ✓
- Azure Service Bus. ✓

IBM MQ.

Scaling a messaging queue



Issues ; ?

✓ ① - SPOF

② - Load / traffic + storage

FIFO Queue

① maintain order

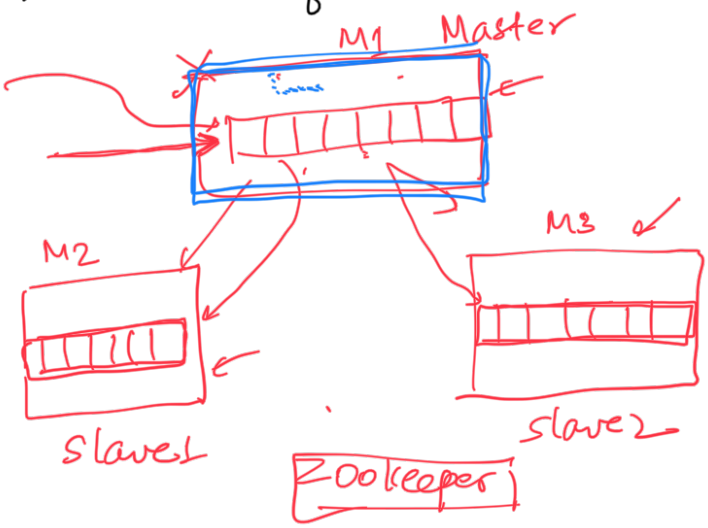
② Deletion oldest messages

Issue #1 : Machine failure ⇒ loss of data

SPOF ↑

① CAP Theorem (Scaler)

② Master Slave



Issue #2 ⇒ Single machine for write

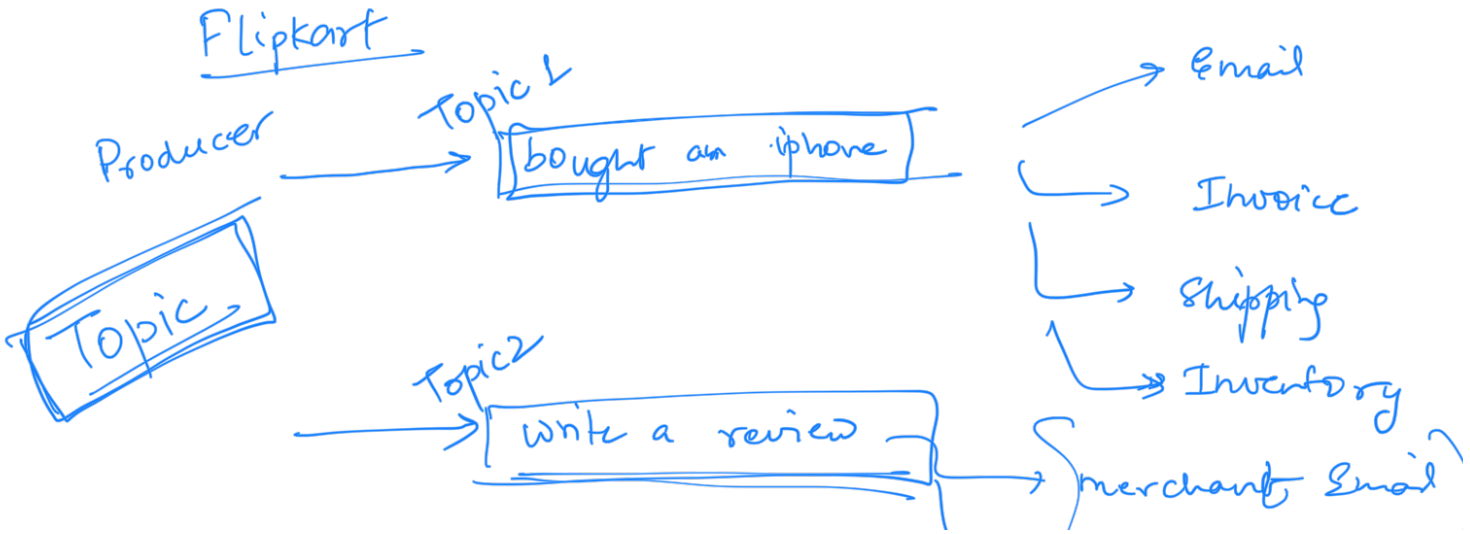
⇓

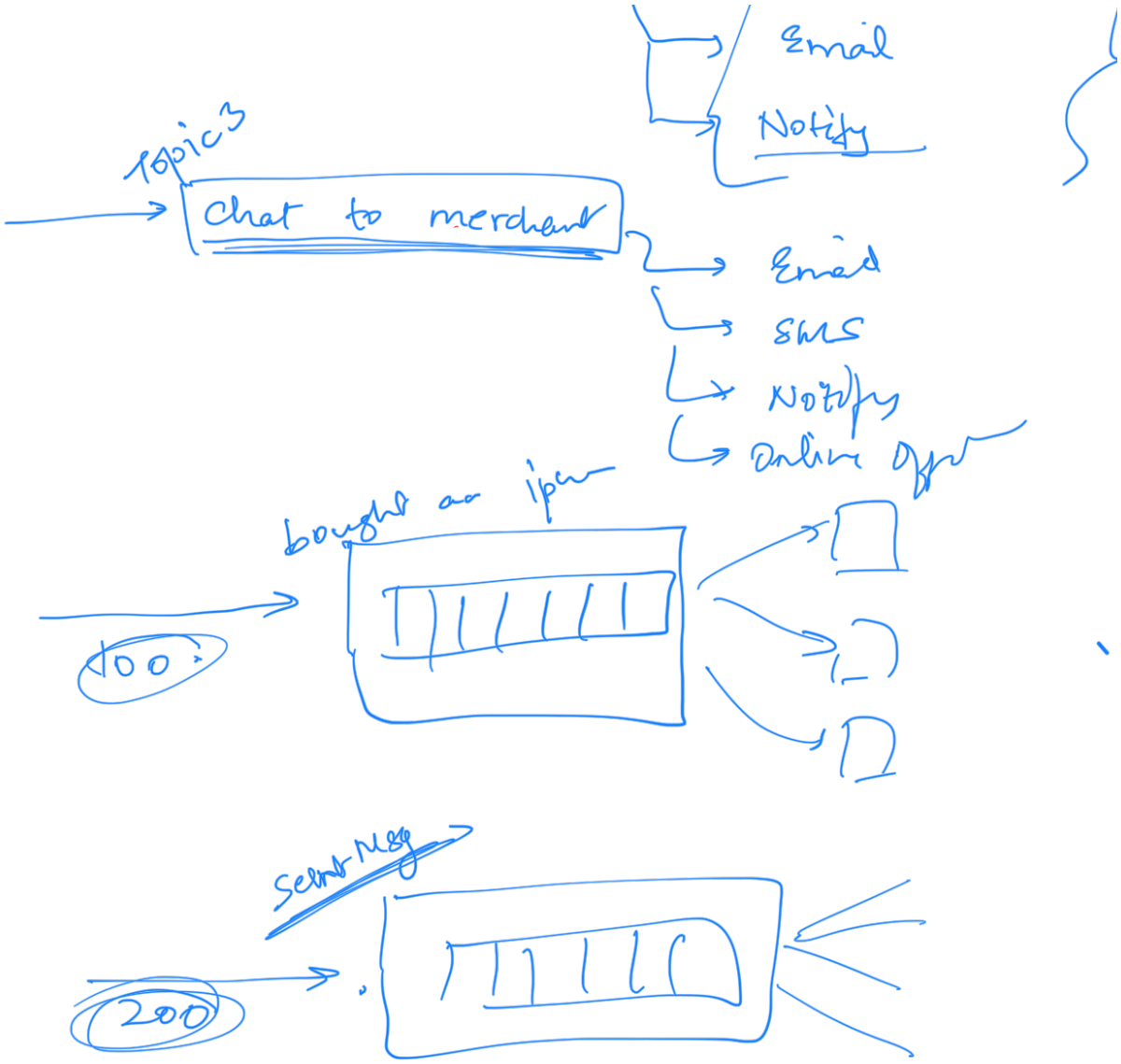
limited traffic capacity :(

Whatsapp ⟹ 30 B messages / day

⇒ ~0.5 million msgs / second

Solution #1: Can there be multiple producers?



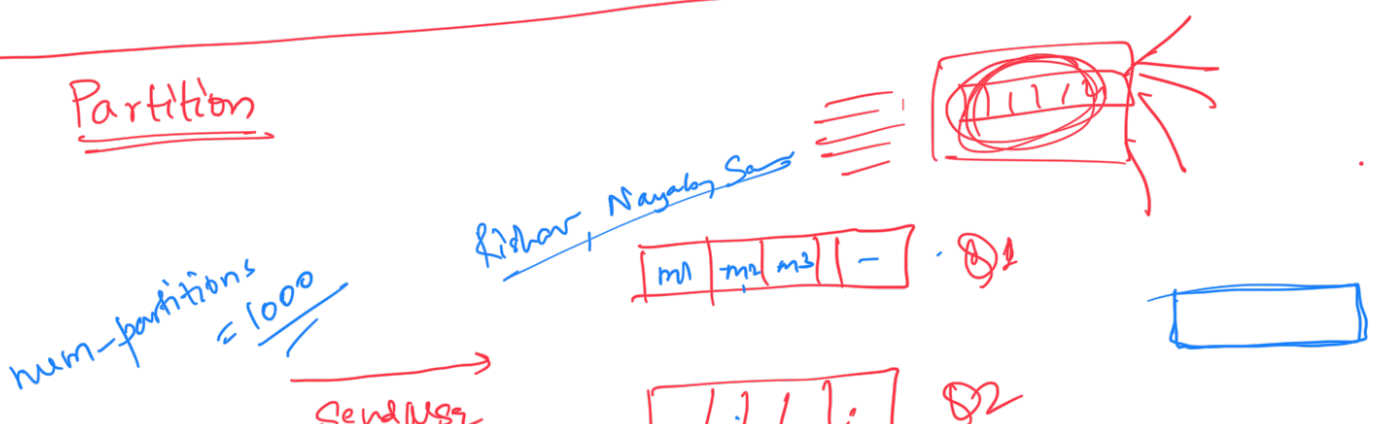


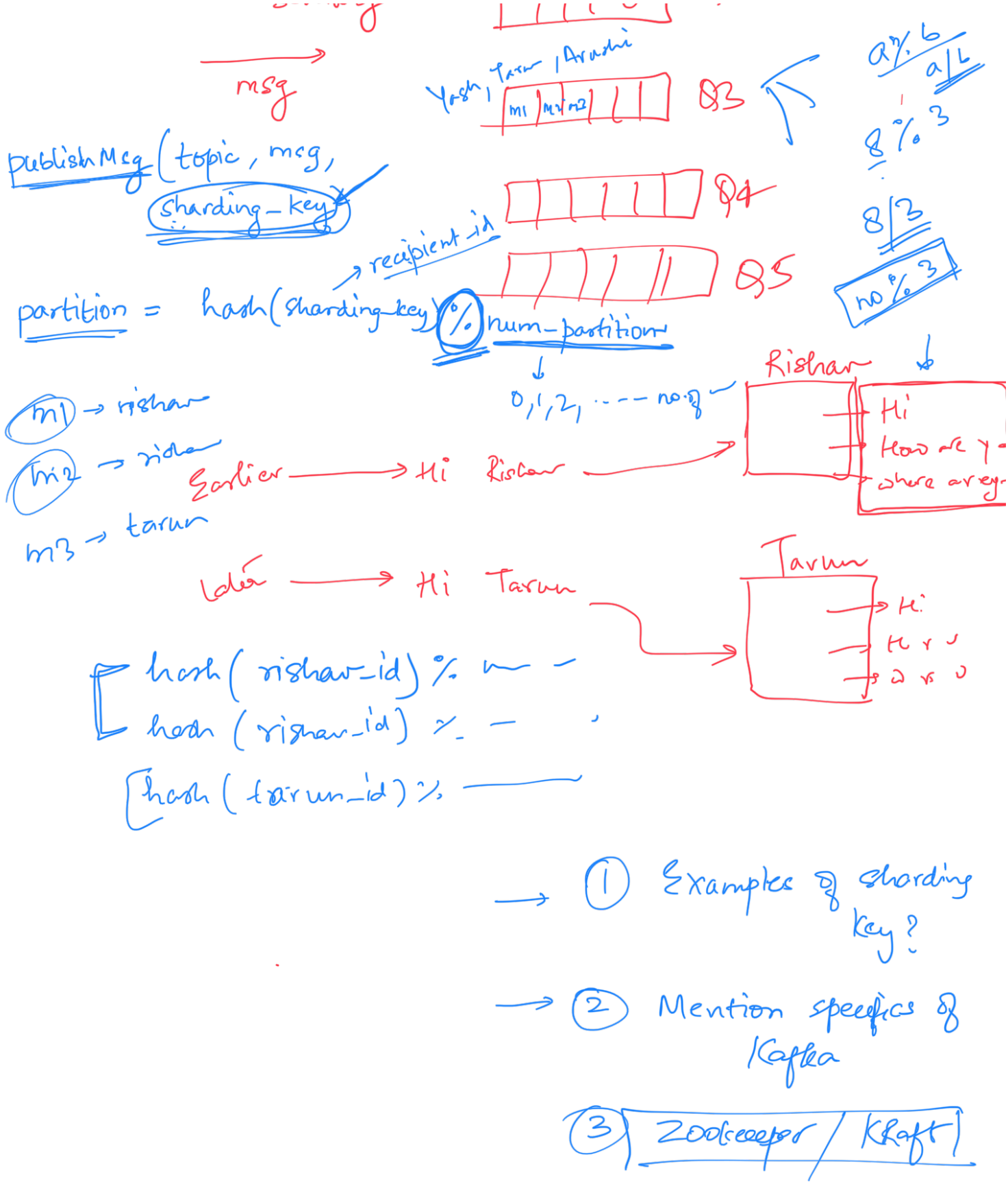
Producer

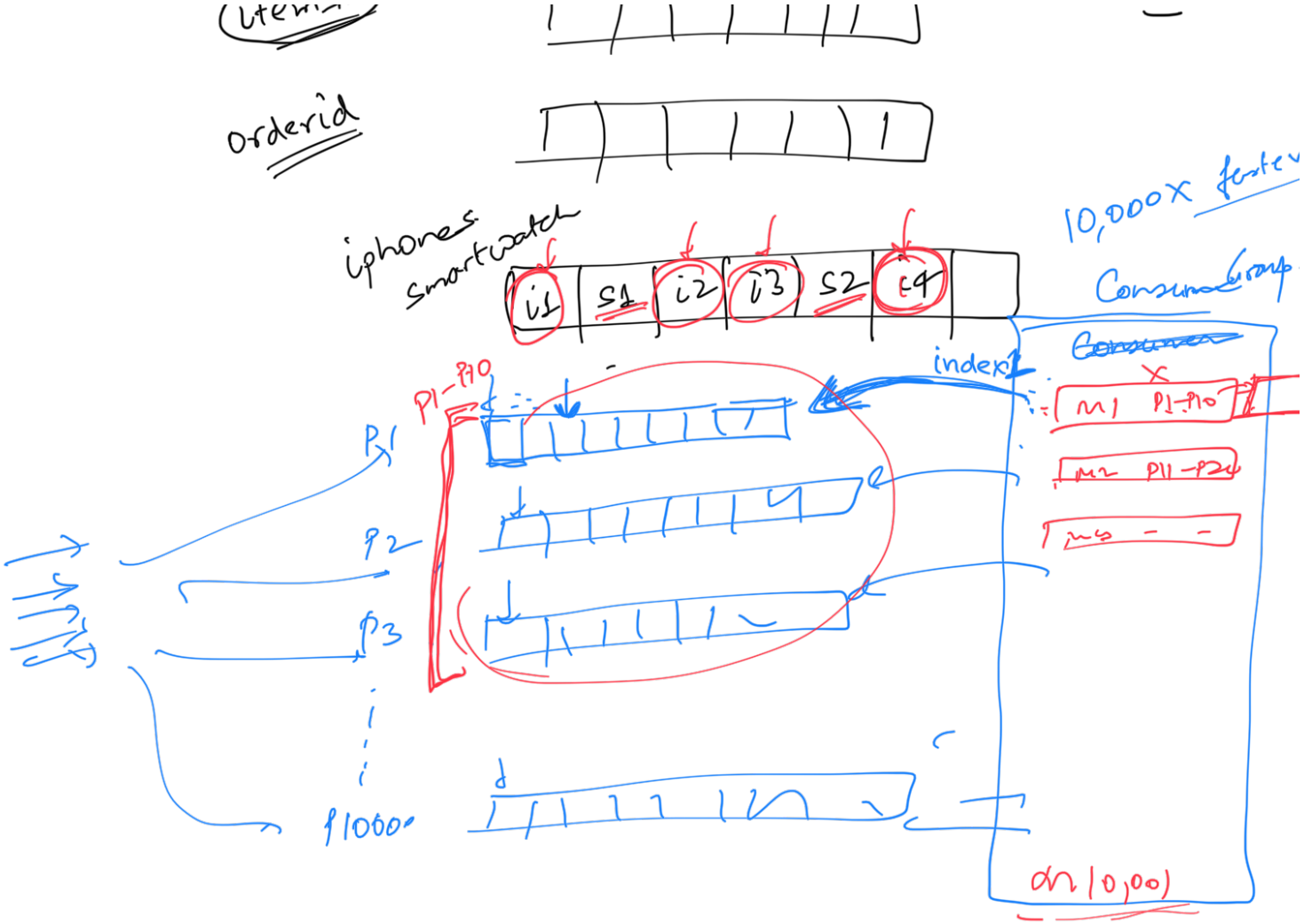
publishMsg(topic, msg, ...)

→ sendMsg() 1 million msgs/second

Partition




msg
publishMsg (topic, msg, sharding_key)
partition = hash(sharding_key) % num_partition
recipient_id
Q3
Q4
Q5
0, 1, 2, ... no.q
m1 → rishav
m2 → rishav
m3 → tarun
Earlier → Hi Rishav
Rishav
Hi
How are y
where are y
Later → Hi Tarun
Tarun
Hi
hash (rishav_id) % —
hash (rishav_id) % —
hash (tarun_id) % —
① Examples of sharding key?
② Mention specifics of Kafka
③ Zookeeper / Kraft
item Purchased (Topic)
Producer
P1
P2
P3
P4
P5
P6
Inventory

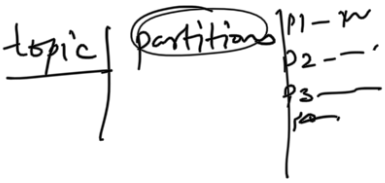


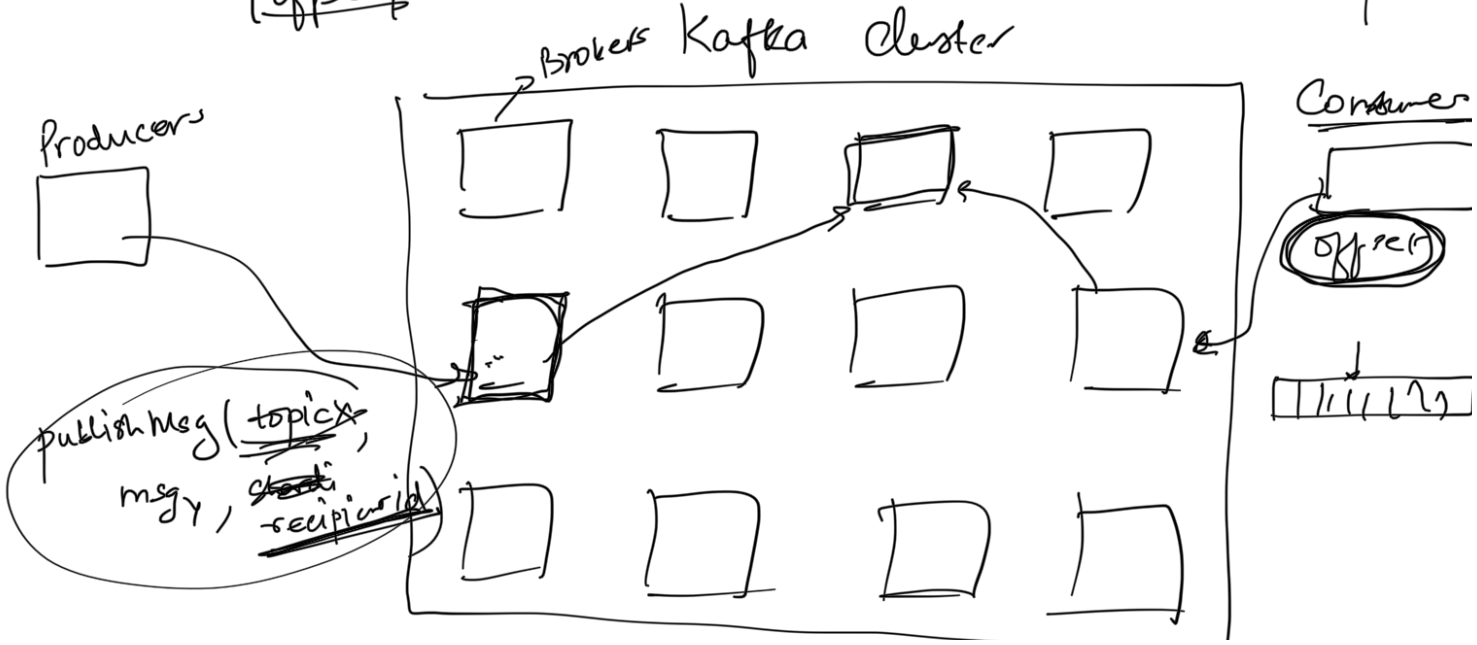
Broker → machine/server

Topic

Partition

Offset





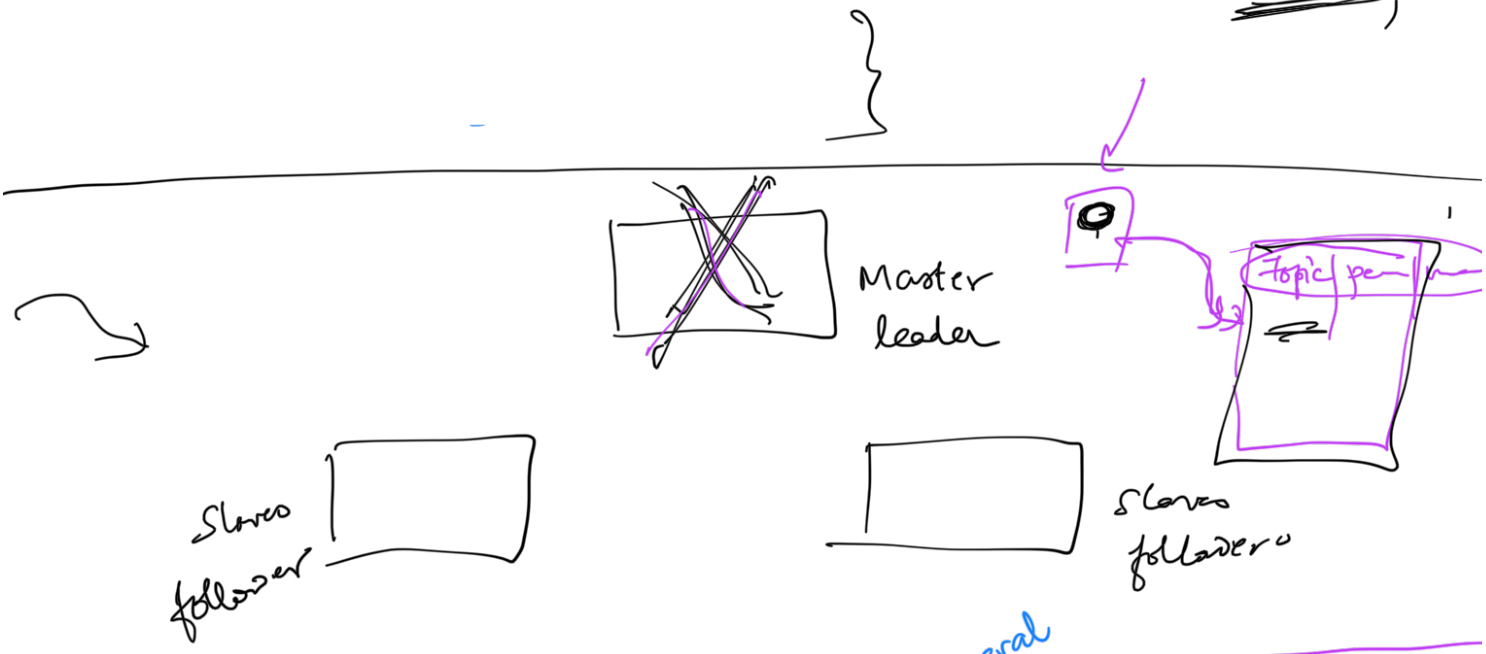
hash(recipientId) % numPartitions

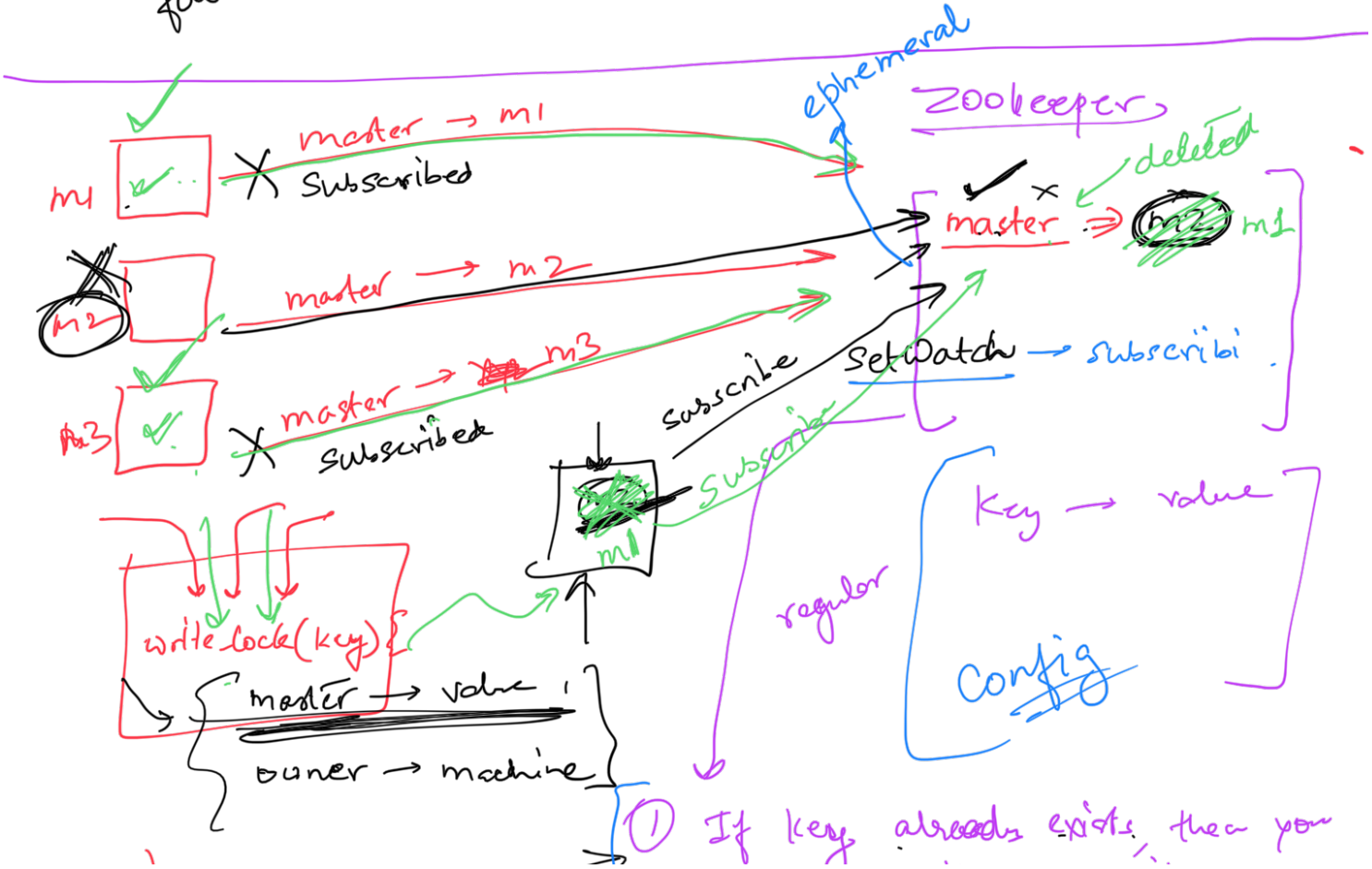
topic, partition_no

hash(...) {

return number;

}





}

- can't over-write values
2. Every entry has a owner
3. Every owner has to renew their ownership every few seconds