Kafka transactions and EOS

Vojtěch Juránek

Red Hat

May 24th 2024, Debezium F2F meeting, Brno

- Idempotent producers.
- Transactions
- Exactly once semantics.

Idempotent producer

- Each producer has assigned unique ID (PID) and epoch/generation.
- Each message has assigned sequence number that is incremented for every message sent.
- Broker keep maximum sequence number for given producer and partition (high watermark).
- Message is accepted by the broken only when the sequence number of the message for given epoch is high watermark + 1.

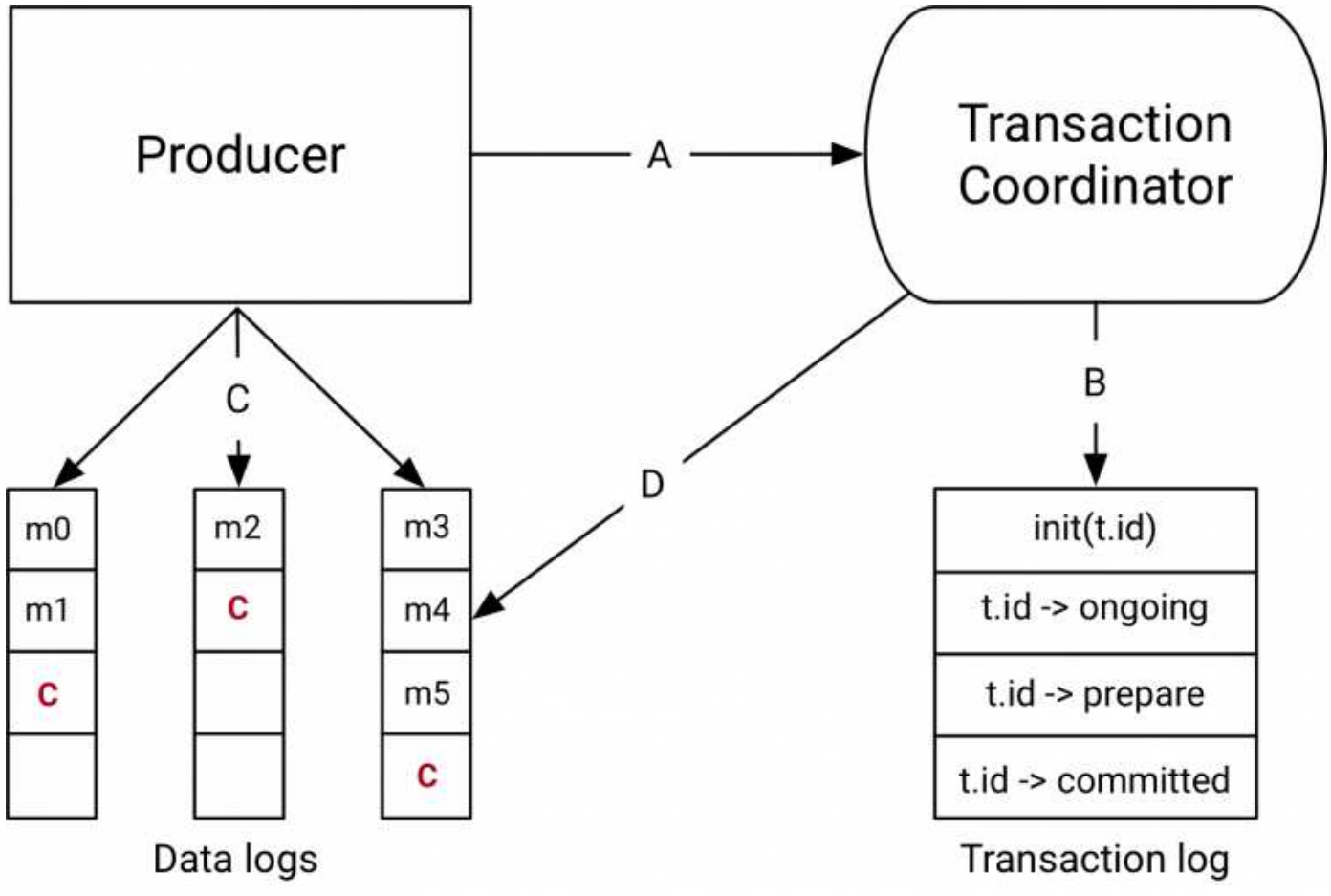
Kafka transactions

- Atomic writes across multiple Kafka topics and partitions.
- Atomic read-process-write cycles.
- Producer can run at most one ongoing transaction.
- Consumer delivers transactional messages to the application only if the transaction was committed.
- However, consumer is not guaranteed to be subscribed to all partitions that are part of the transaction.

Transaction coordinator

- Runs inside every Kafka broker.
- Takes care about transaction log - an internal Kafka topic, more specifically is a leader of selected partitions of transaction log.
- Transaction log stores only state of the transaction and associated metadata, not the records themselves.
- Owns subset of partitions in transaction log - for these partitions for which broker is the leader.

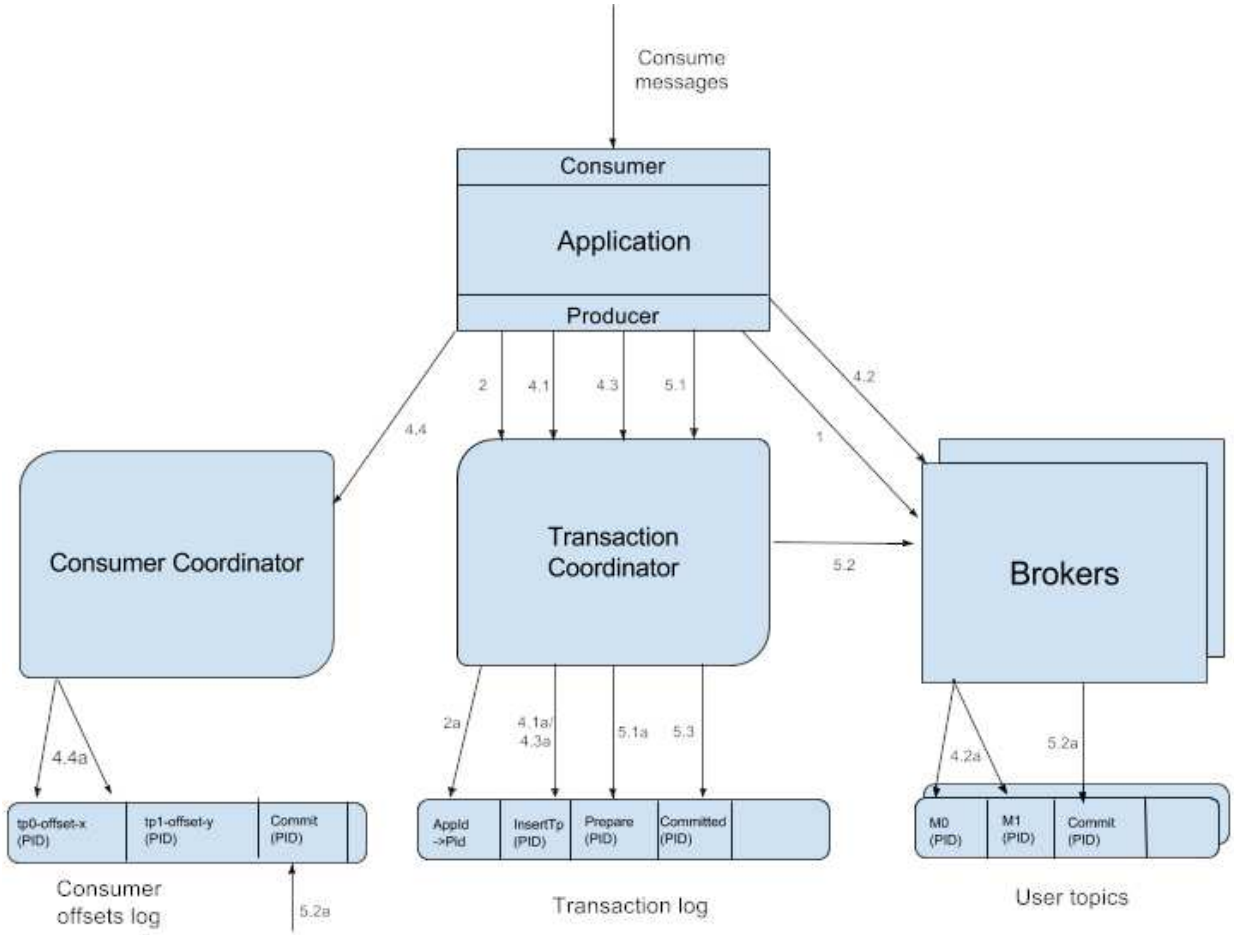
Transaction coordinator



Source: `https://www.confluent.io/blog/transactions-apache-kafka/`

Transaction flow

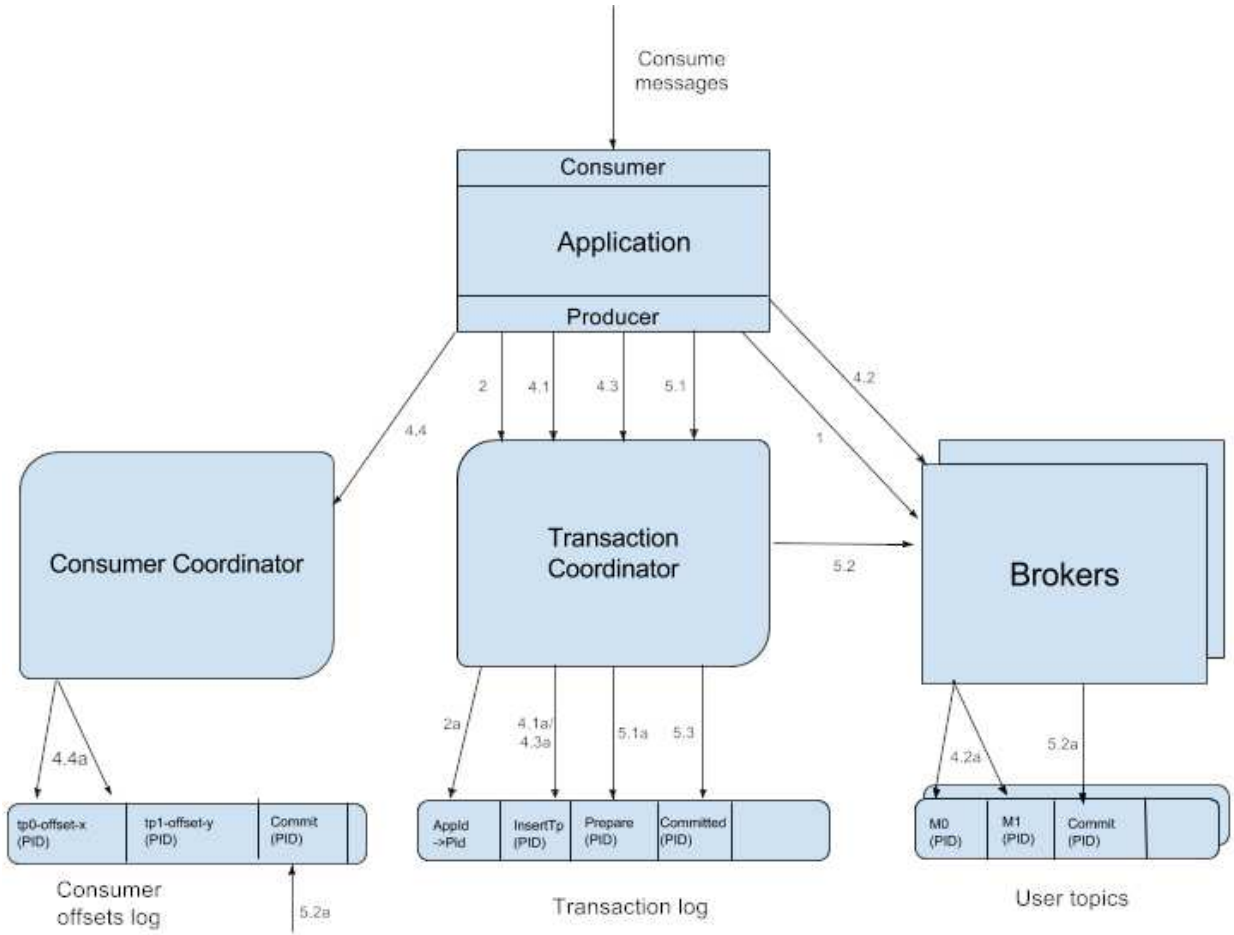
1. Producer finds transaction coordinator for its group.



Source: `https://cwiki.apache.org/confluence/display/KAFKA/KIP-98+-+Exactly+Once+Delivery+and+Transactional+Messaging`

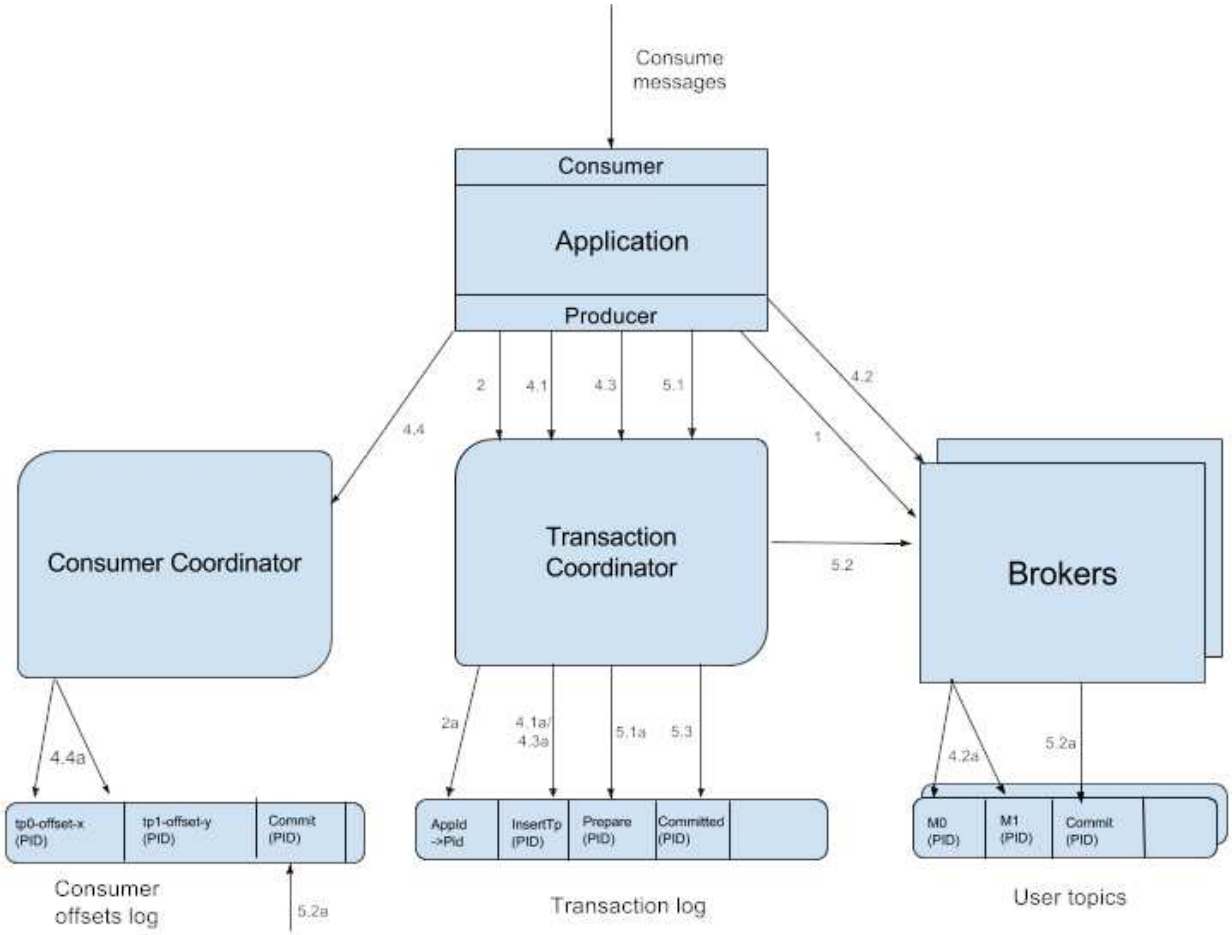
Transaction flow

2. Producer obtains producer ID (PID).
   - Based on it's `transactional.id` producer obtains PID from the coordinator.
   - Coordinator also bumps the epoch for given producer
   - and resolves exiting pending transaction from given producer



Transaction flow

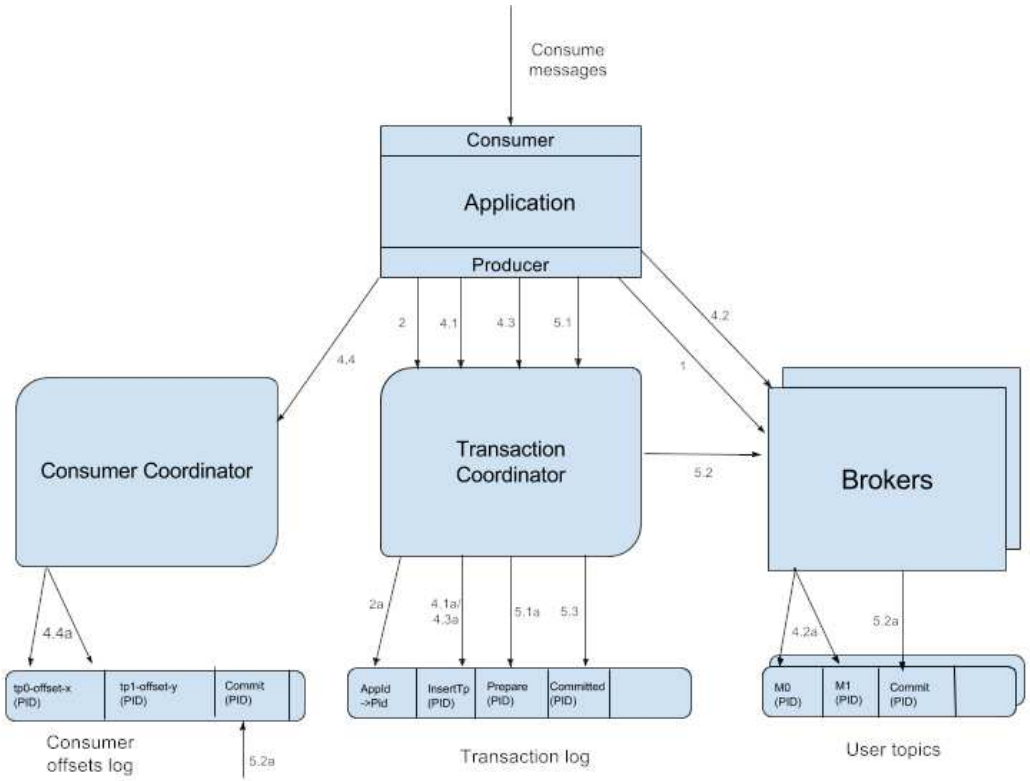
3. Producer starts the transaction (calls `beginTransaction()`).



Source: `https://cwiki.apache.org/confluence/display/KAFKA/KIP-98++Exactly+Once+Delivery+and+Transactional+Messaging`

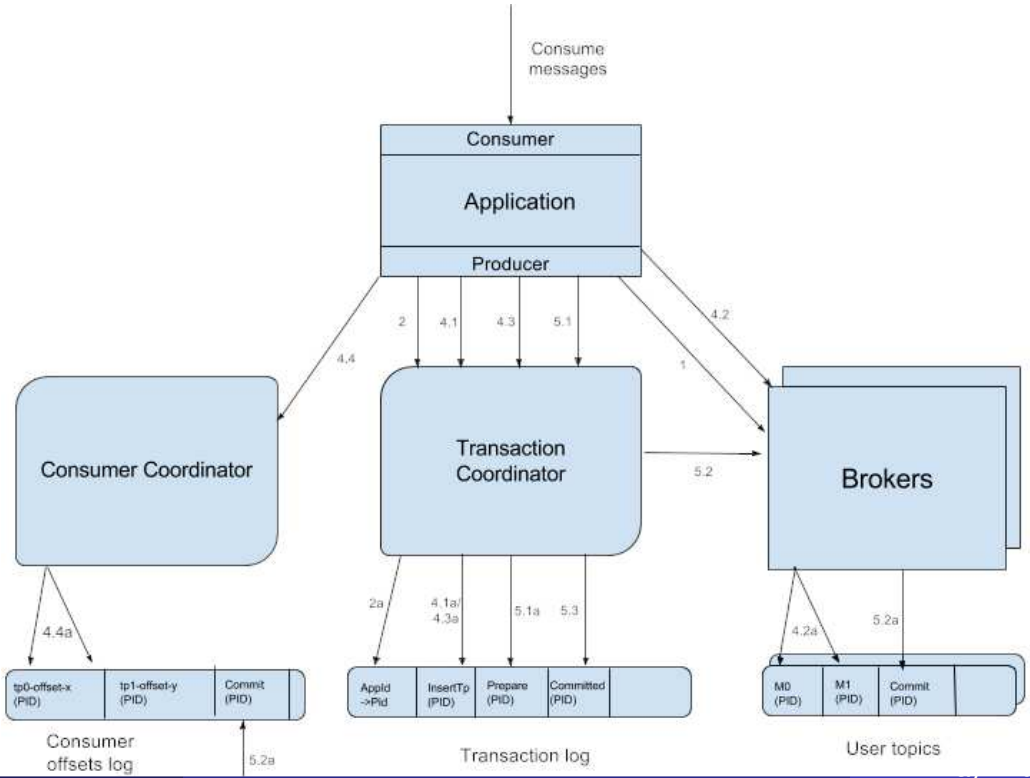
Transaction flow

4. Consume-transform-produce loop.
   - Producer requests adding partitions to TX request.
   - Coordinator starts TX timer.
   - Producer sends TX messages.
   - Producer sends TX coordinator request for adding offsets into TX (enables batching of consumer and produced messages).
   - Producer sends TX offset commit request to the consumer coordinator.



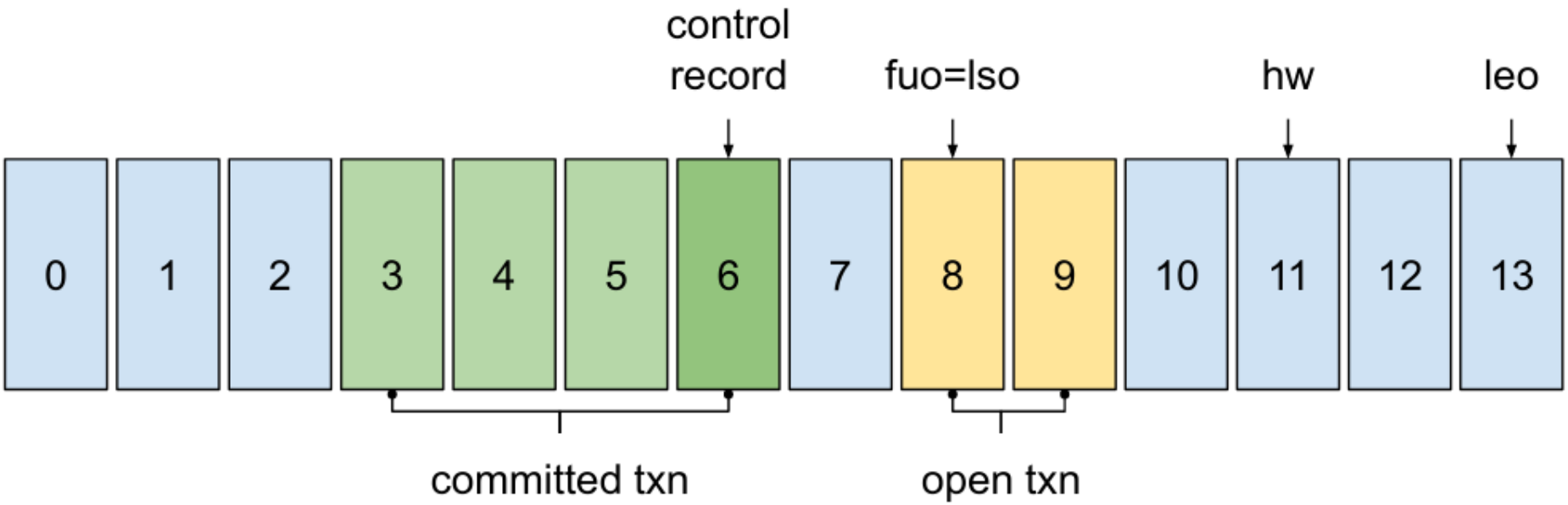
Transaction flow

5. Transaction is committed or aborted.
   - Coordinator writes prepare commit or prepare abort to TX log.
   - Coordinator sends commit to user logs.
   - Coordinator writes `commit` to TX log.
   - Coordinator sends TX marker to partition leaders of affected partitions.
   - Partitions leaders `commit` to their logs.
   - Coordinator writes `committed` to TX log.
   - Consumers delivers TX messages.



Offsets

- Last stable offset (LSO) - all lower offsets have been decided (committed or aborted).
- First unstable offset (FUO) - the earliest offset that is part of the ongoing transaction.
- High watermark (HW) - the offset of the last message that was successfully copied to all of the log's in-sync replicas.
- Log end offset (LEO) - the the highest offset of the partition.



Source: https://strimzi.io/blog/2023/05/03/kafka-transactions/

Related config options

Producer config:

- `enable.idempotence` - must be set to `true`.
- `transaction.timeout.ms` - maximum amout of time the coordinator will wait for transaction to be completed.
- `transactional.id` - has to be unique per producer.

Consumer config:

- `isolation.level = read_uncommitted | read_committed`

Broker config - has the sane defaults

- `transactional.id.timeout.ms`
- `max.transaction.timeout.ms`
- `transaction.state.log.replication.factor`
- `transaction.state.log.num.partitions`
- `transaction.state.log.min.isr`
- `transaction.state.log.segment.bytes`

KIP-618: Source connector EOS

- Wraps everything (sending records, committing offsets) into a transaction.
- Kafka TX framework also fences zombie producers.
- Source connector has to be able to resume from it's (external resource) offset position.

Source connector EOS related config options

Worker config:

- `exactly.once.source.support = disabled | preparing | enabled`

Consumer config:

- `exactly.once.support = requested | required`
- `transaction.boundary = poll | interval | connector`
- `offsets.storage.topic`
- `transaction.boundary.interval.ms`

Resource

- Kafka Idempotent Producer
- Kafka Transactional Messaging
- Kafka KIP-98: Exactly Once Delivery and Transactional Messaging
- Kafka KIP-618: Exactly-Once Support for Source Connectors
- Exactly Once Delivery and Transactional Messaging in Kafka (design docum
- Transactions in Apache Kafka (Confluent blog)
- Exactly-once semantics with Kafka transactions (Strimzi blog)

Questions?

Thank you!

Questions?