«Один раз в год сады цветут»

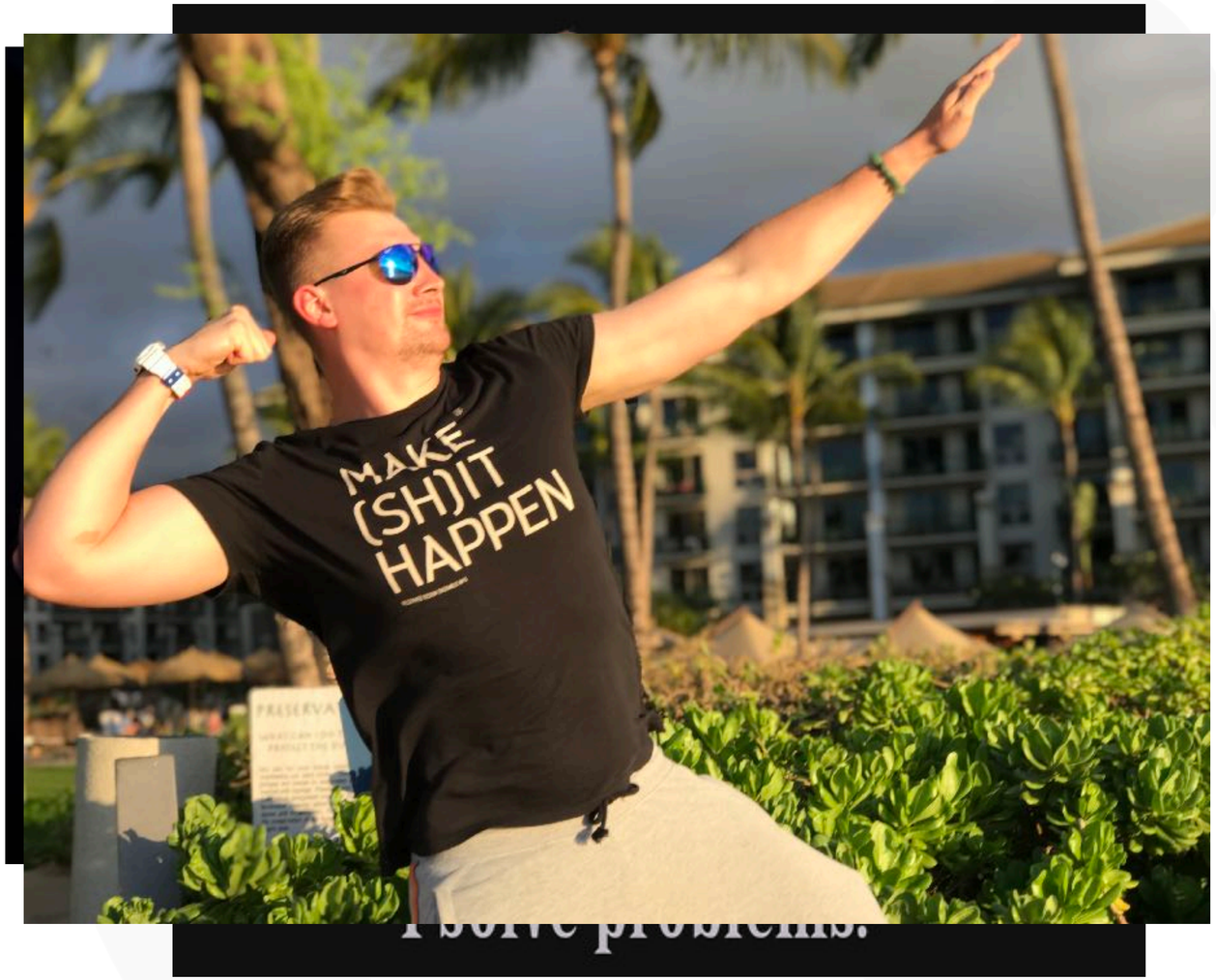
Разбор семантики exact!y-once в Apache Kafka

Ты кто такой? Давай...

confluent

Solutions Architect
Developer Advocate

@gamussa in internetz



Ну подпишись, плиз...

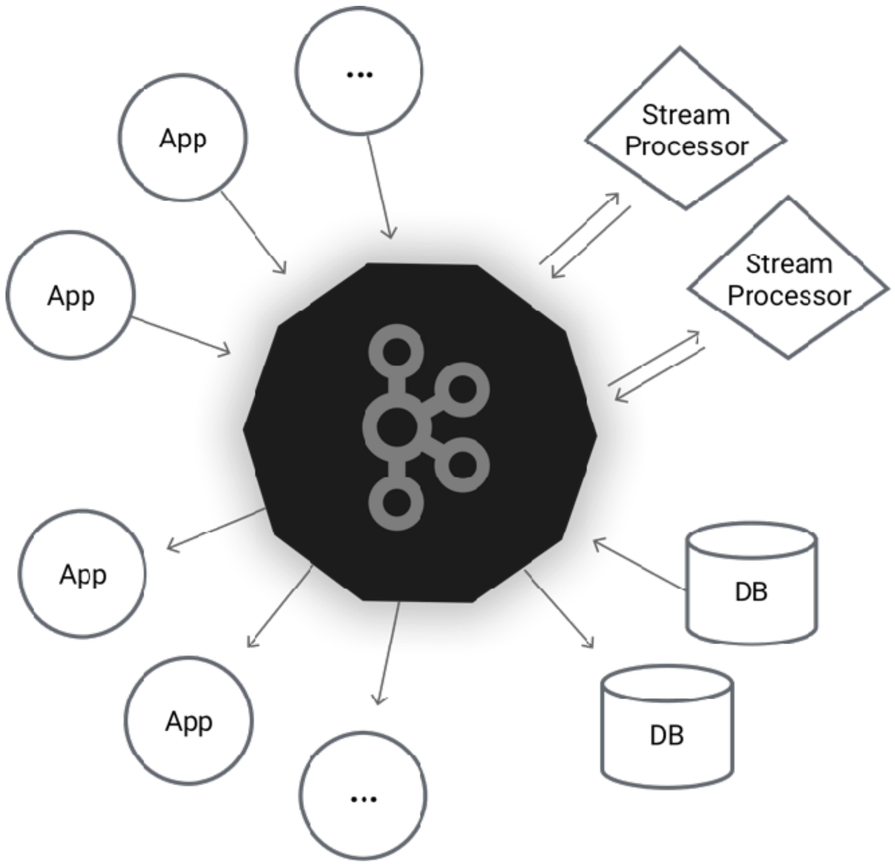
Тщеславие и Маркетинг ©

Разбор семантики Exact!y-once в Apache Kafka

- Про Кафку по-быстрому
- Ограничения старой семантики
- Углубляемся в Exact!y-once семантику
- Чему мы научились и будущие улучшения

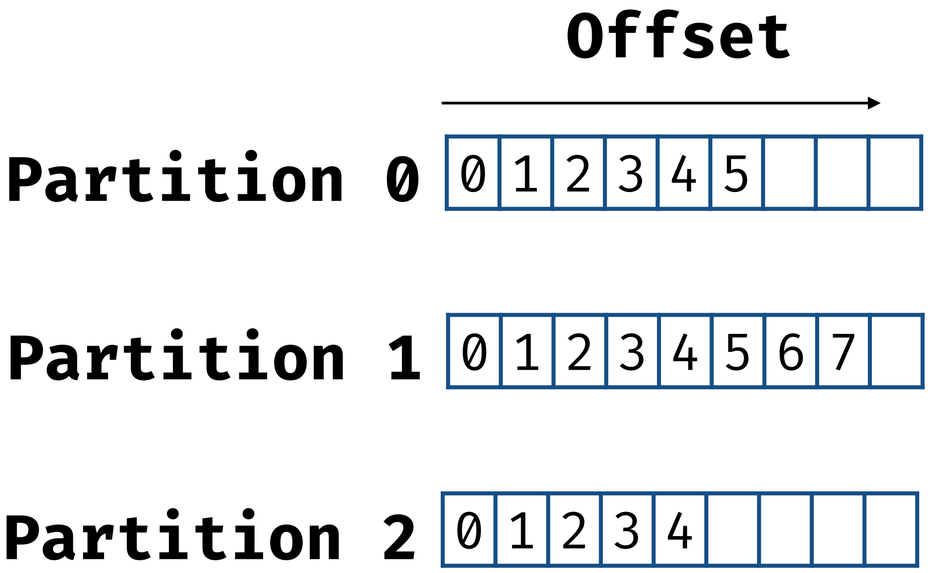
Про Кафку по-быстрому

- Kafka распределенная платформа обработки потоковой информации
  - publish
  - subscribe
  - process
  - store

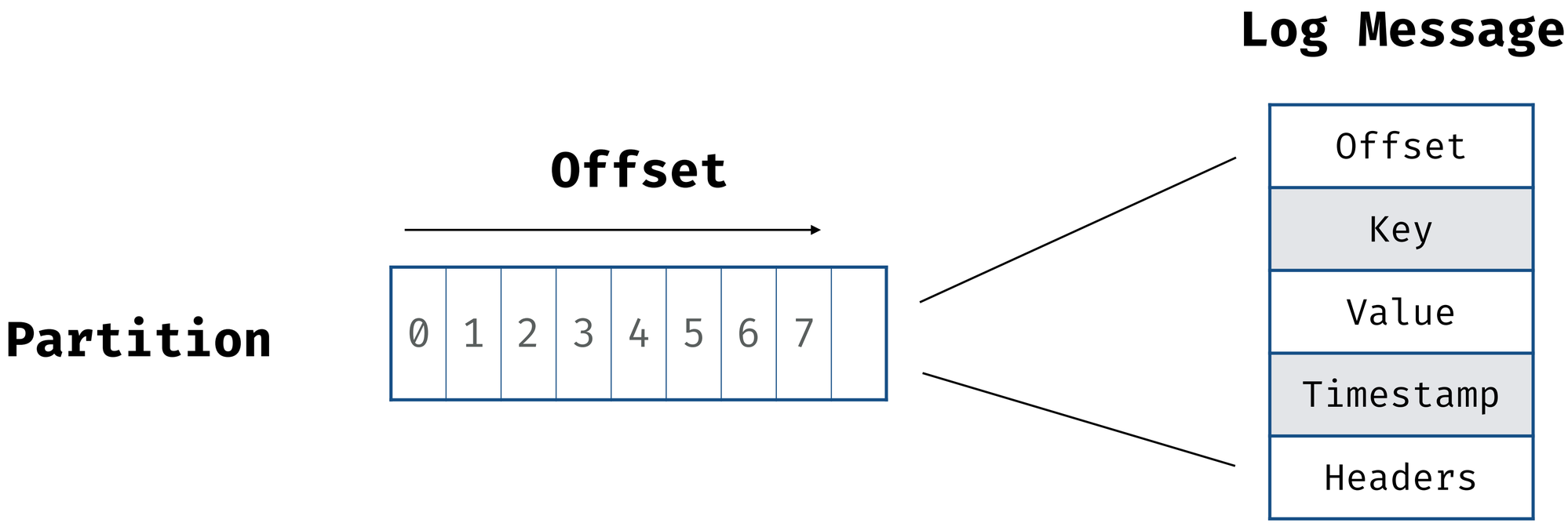


Про Кафку по-быстрому

- Kafka данные организованы в топиках
- Каждый топик состоит из партиций
- Партиция это реплицированный лог
- К сообщениям можно получать доступ по смещению (offset)

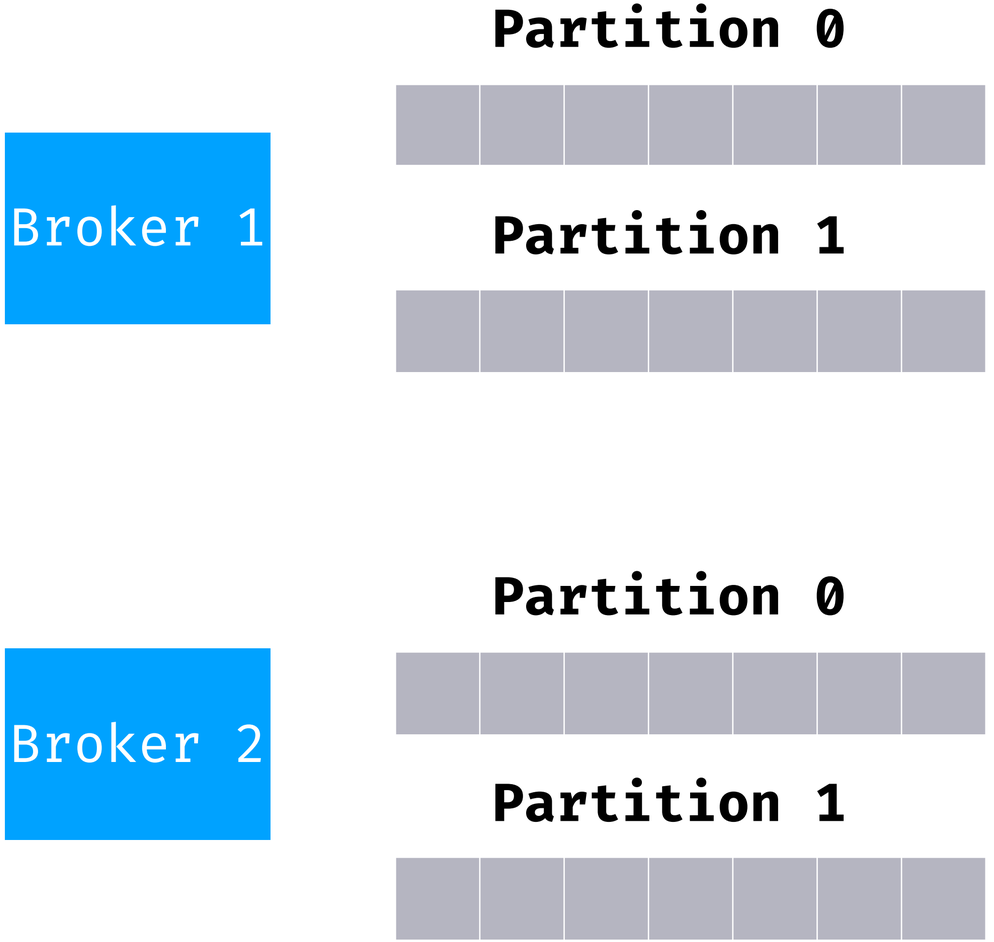


Про Кафку по-быстрому



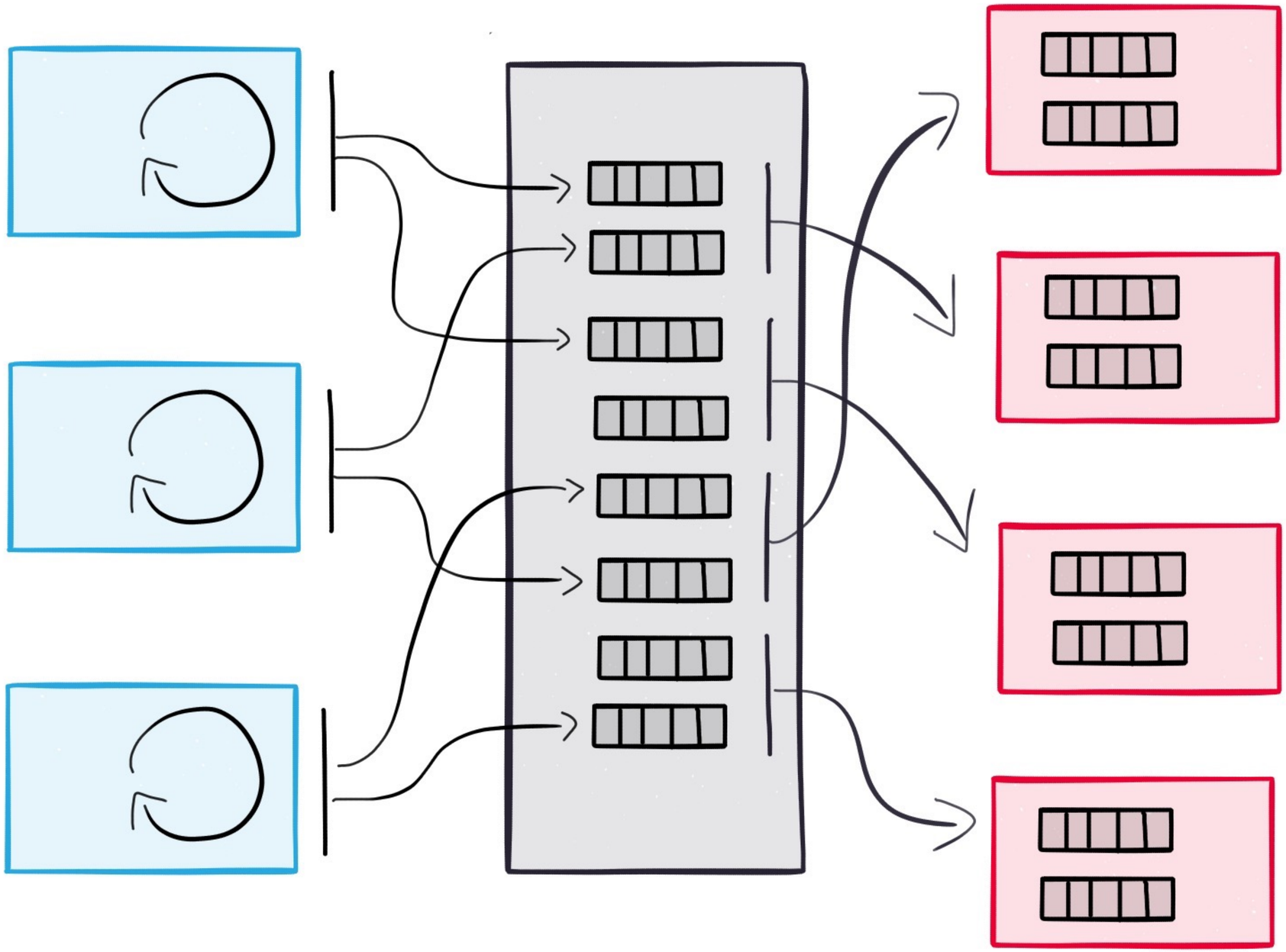
Про Кафку по-быстрому

- Партиции реплицируются
- Количество реплик настраивается
- После сбоя, реплики промотятся в лидеры

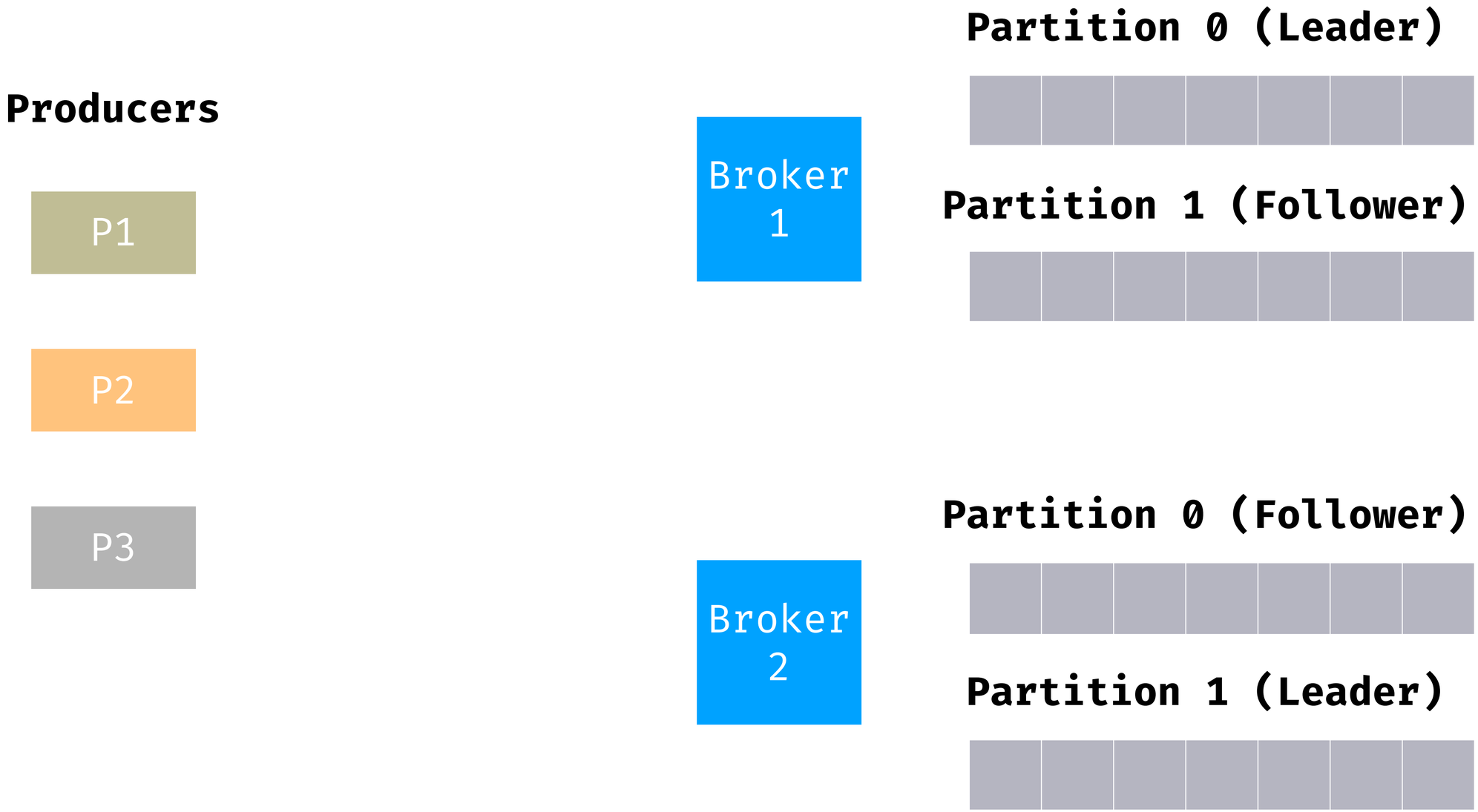


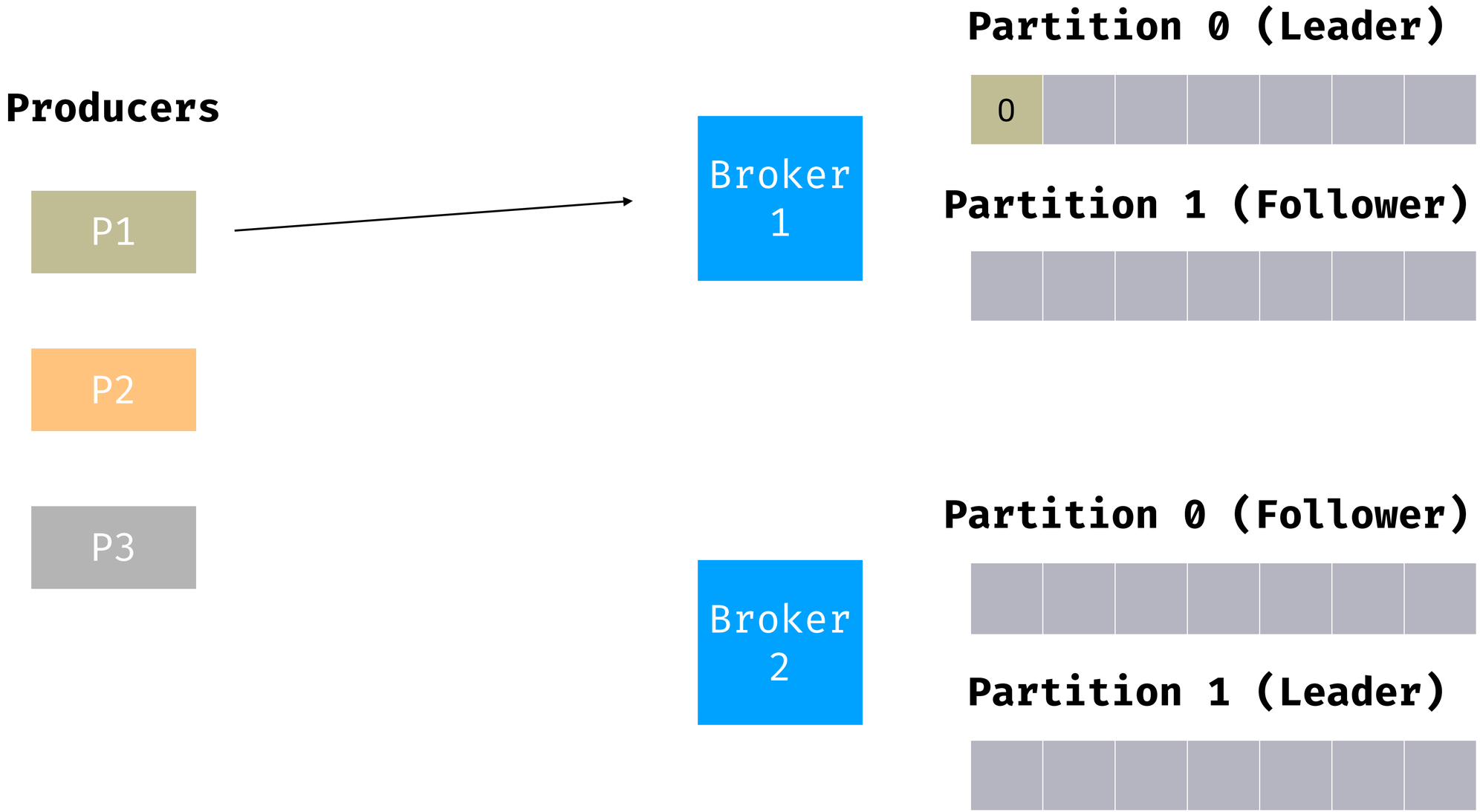
Про Кафку по-быстрому

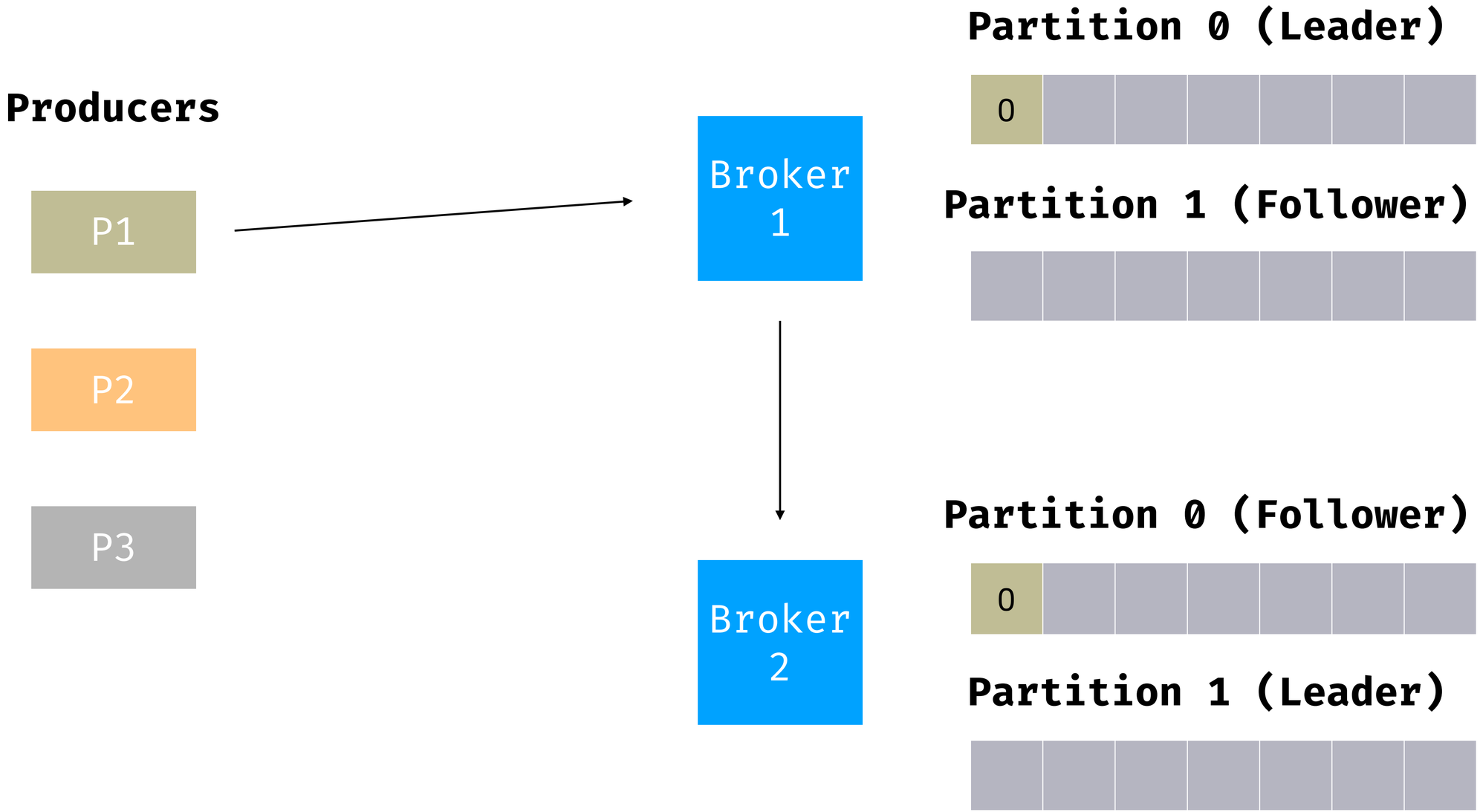
- Клиенты Кафки
  - Producers записывают данные в партиции
  - Consumers читают из партиции

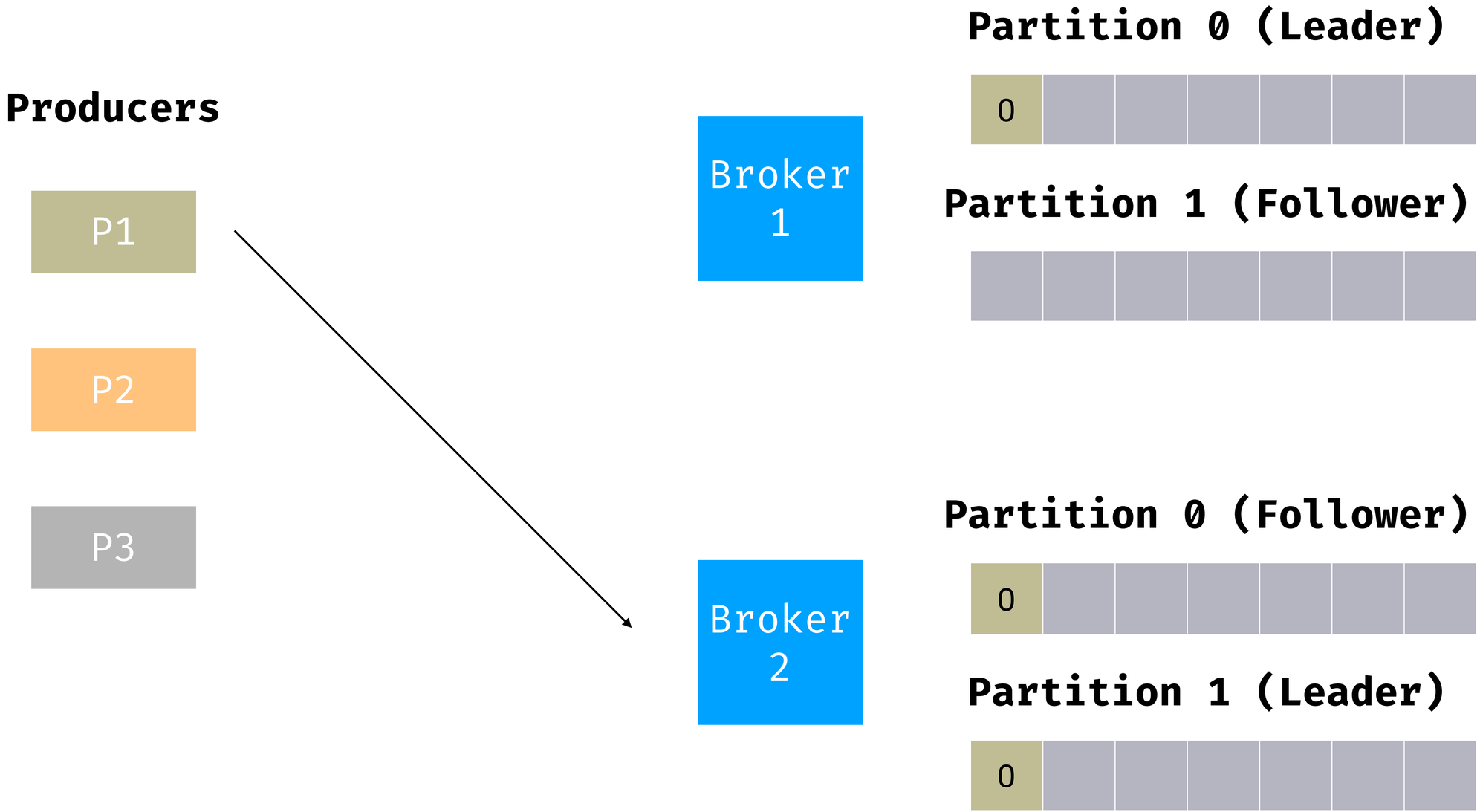


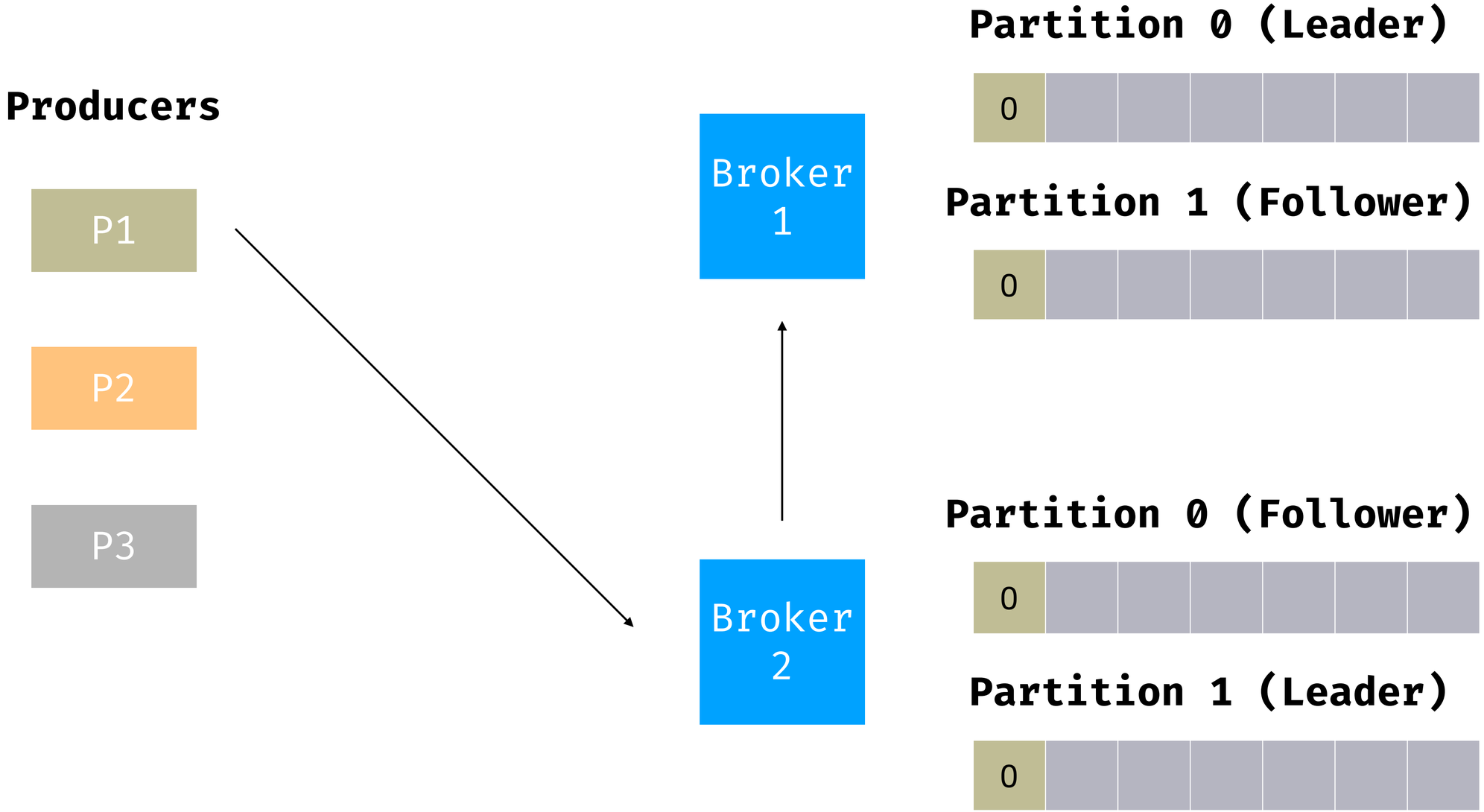
Kafka Producer

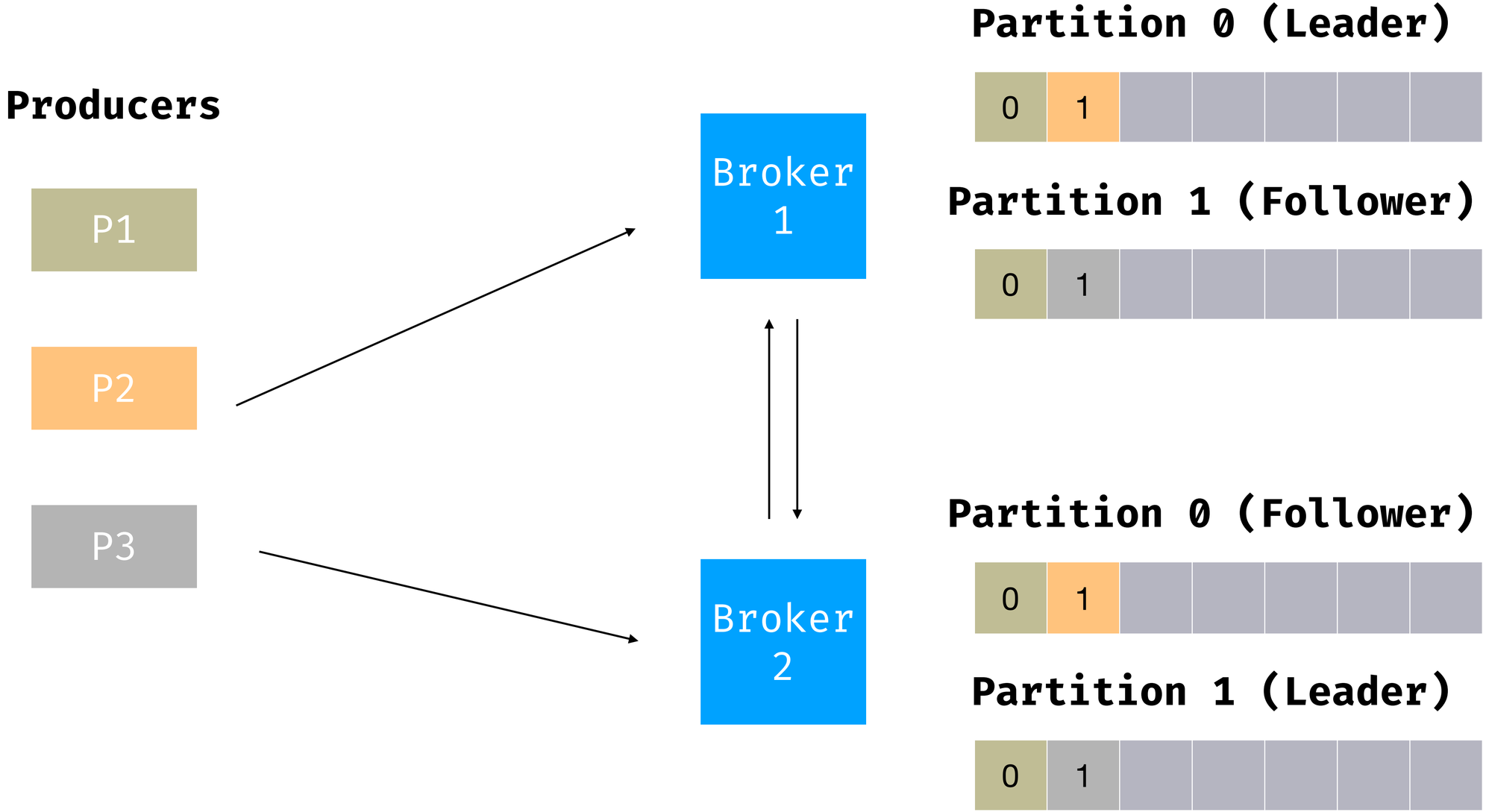

Producers
P1
P2
P3
Broker 1
Partition 0 (Leader)
Partition 1 (Follower)
Broker 2
Partition 0 (Follower)
Partition 1 (Leader)

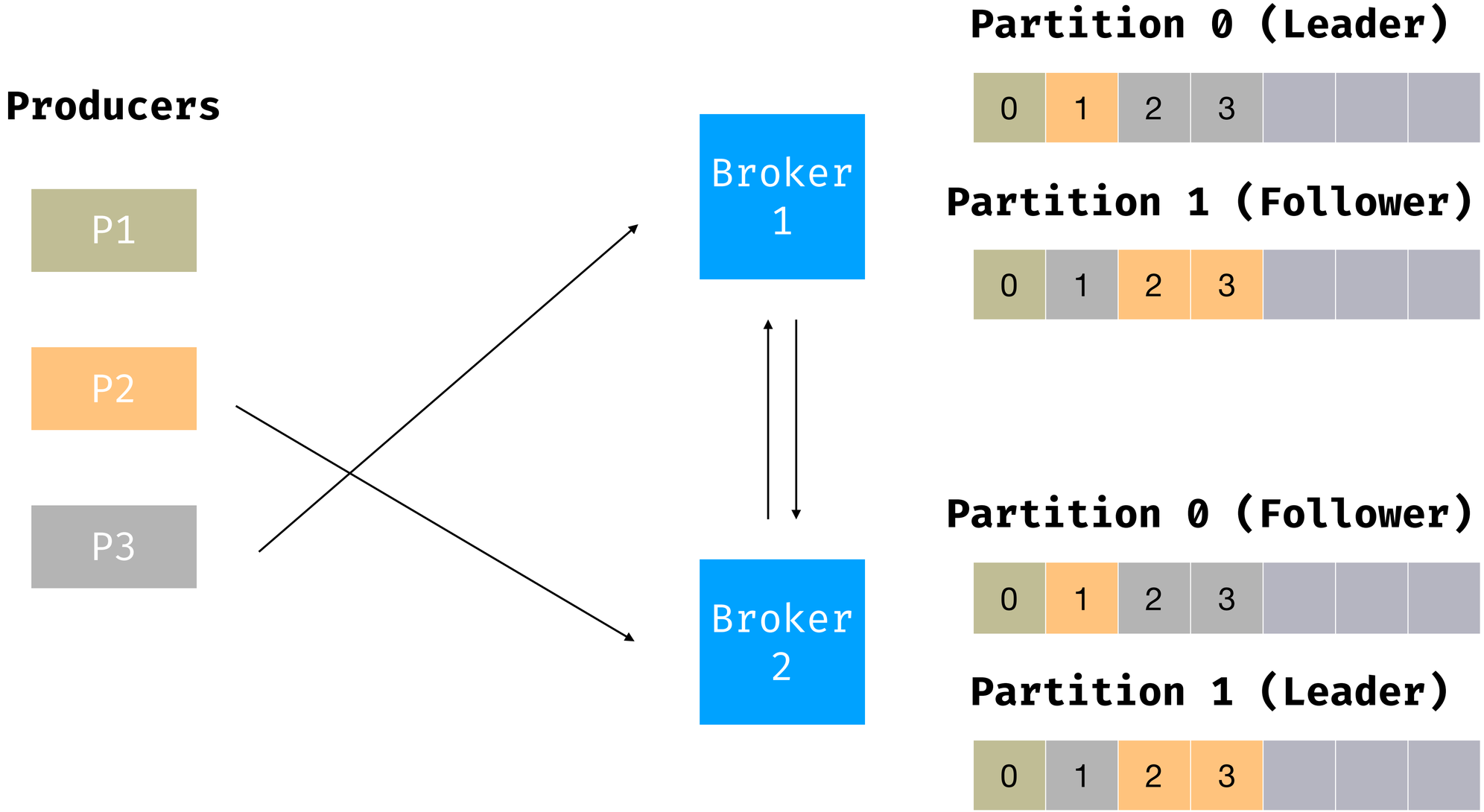

Producers
P1
P2
P3
Broker 1
Broker 2
Partition 0 (Leader)
0
Partition 1 (Follower)
Partition 0 (Follower)
Partition 1 (Leader)

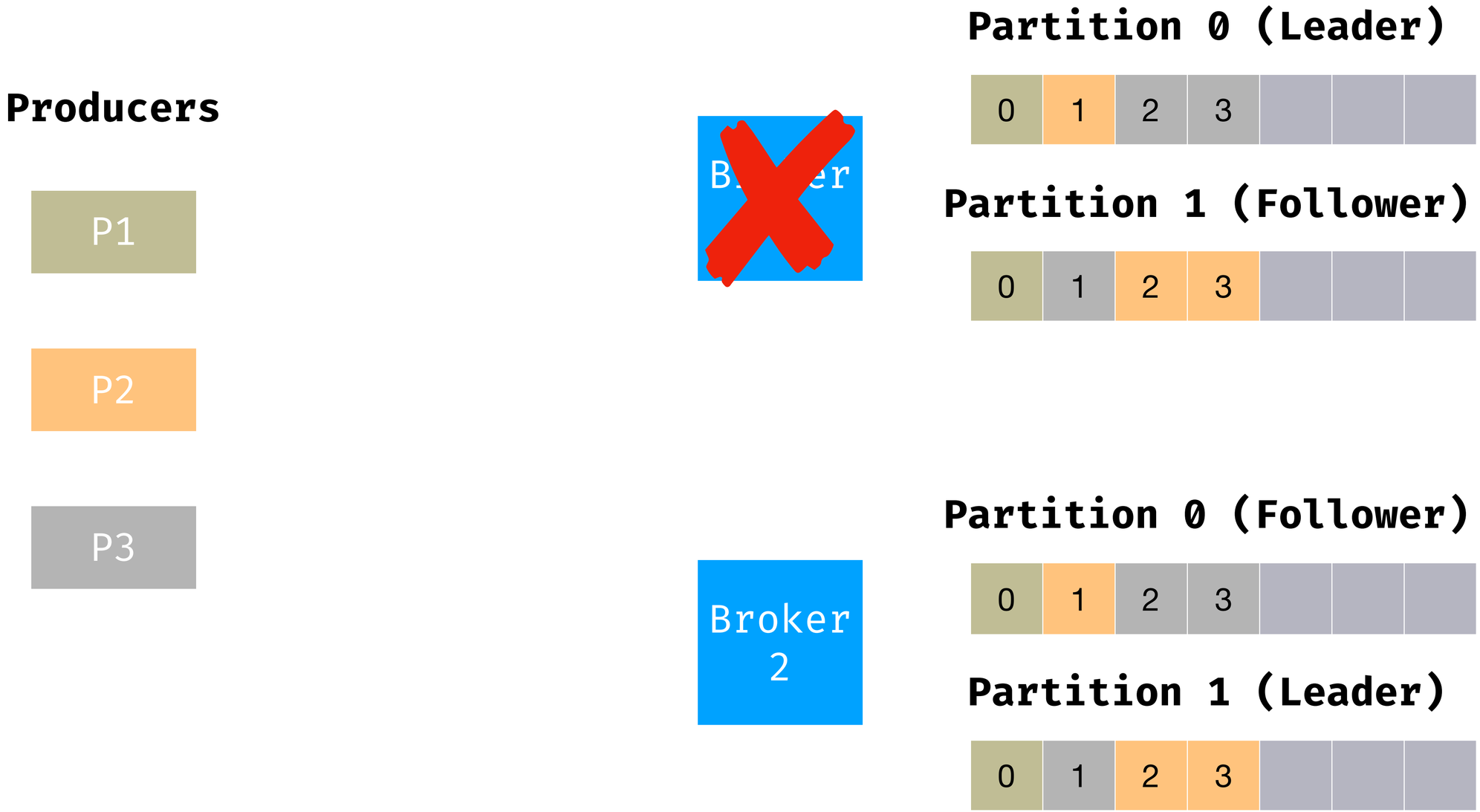

Producers
P1
P2
P3
Broker 1
Broker 2
Partition 0 (Leader)
0
Partition 1 (Follower)
Partition 0 (Follower)
0
Partition 1 (Leader)

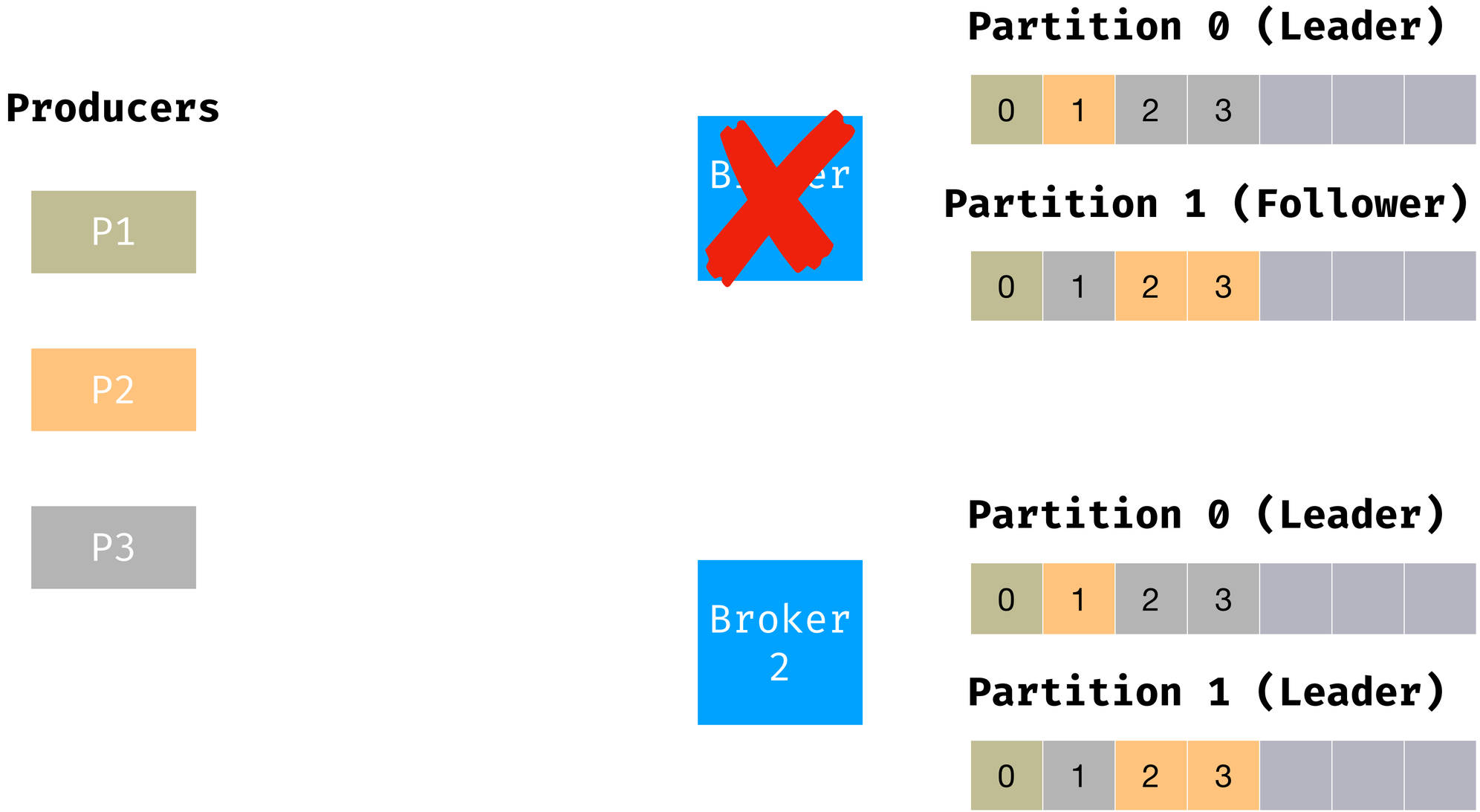

Producers
P1
P2
P3
Broker 1
Broker 2
Partition 0 (Leader)
0
Partition 1 (Follower)
Partition 0 (Follower)
0
Partition 1 (Leader)
0

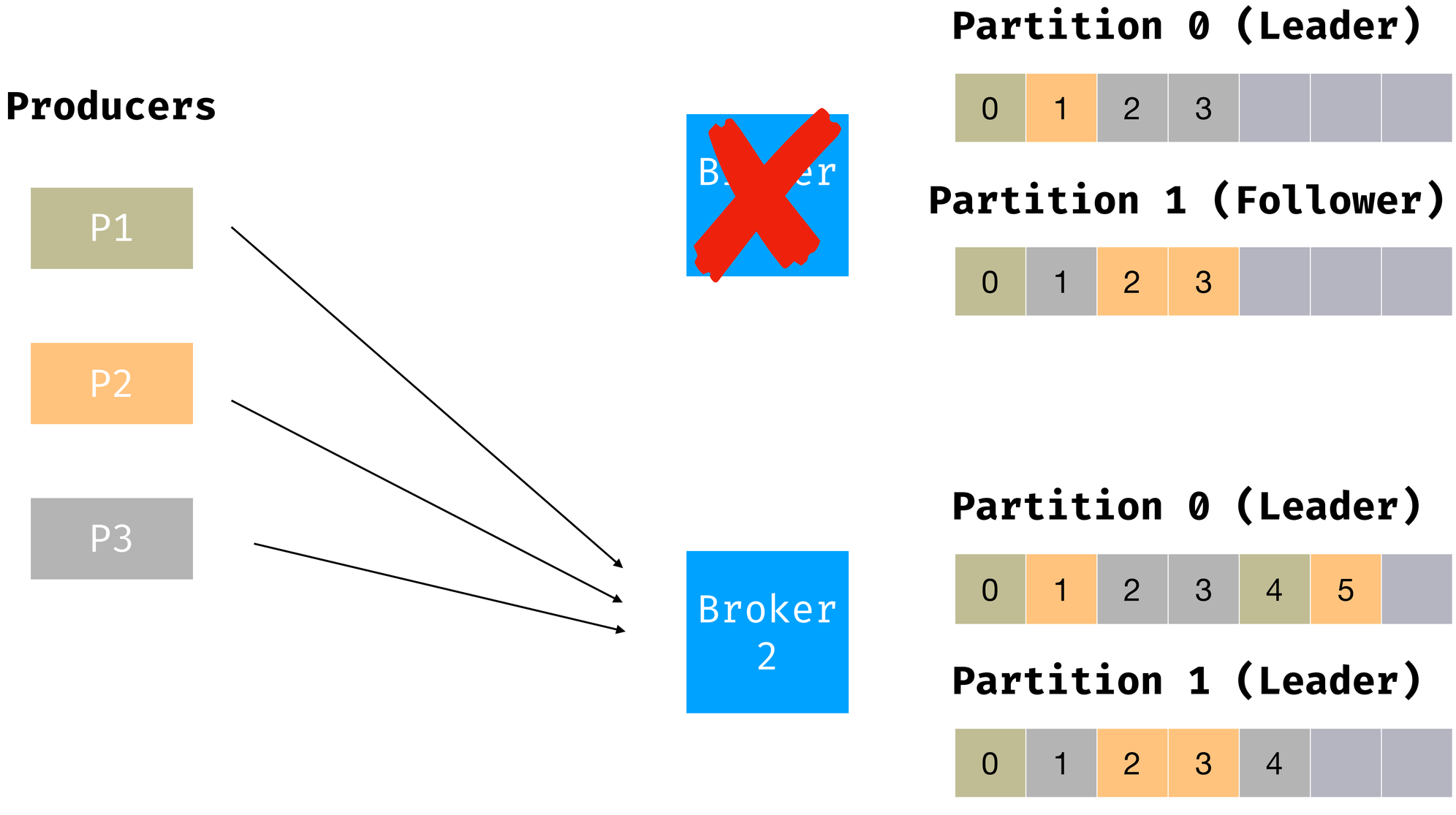

Producers
P1
P2
P3
Broker 1
Broker 2
Partition 0 (Leader)
0
Partition 1 (Follower)
0
Partition 0 (Follower)
0
Partition 1 (Leader)
0

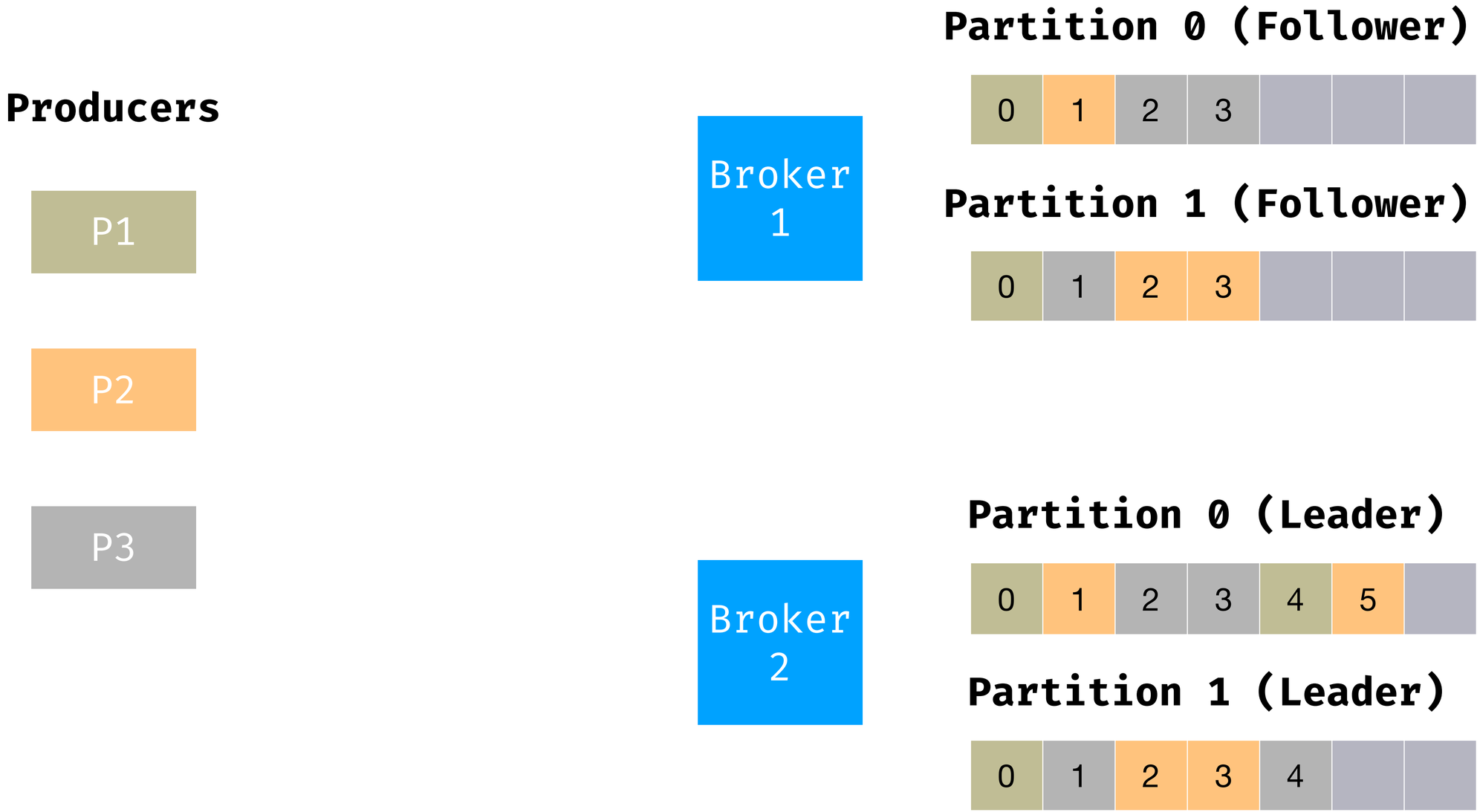

Producers
P1
P2
P3
Broker 1
Broker 2
Partition 0 (Leader)
0
1
Partition 1 (Follower)
0
1
Partition 0 (Follower)
0
1
Partition 1 (Leader)
0
1

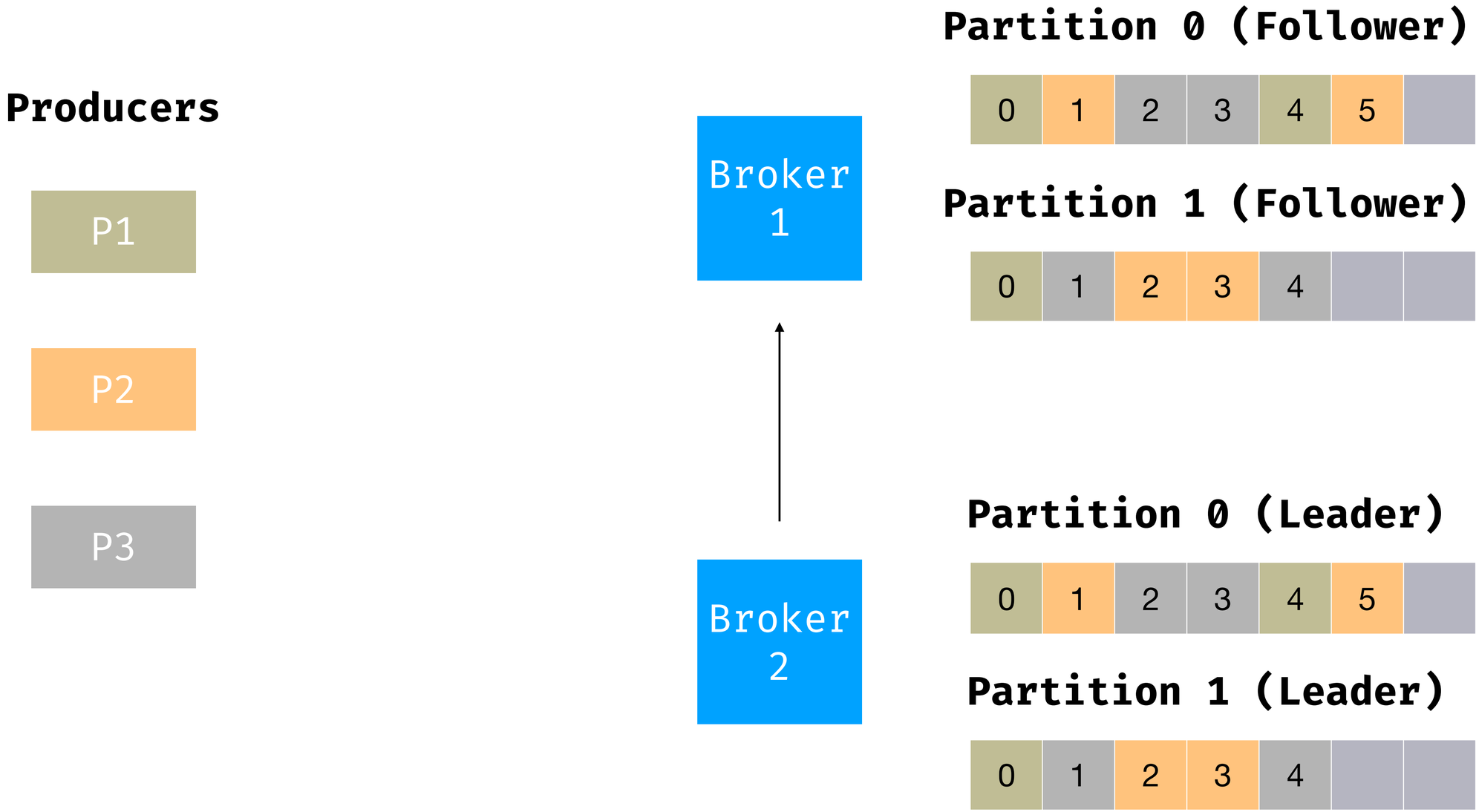

Producers
P1
P2
P3
Broker 1
Broker 2
Partition 0 (Leader)
0 1 2 3
Partition 1 (Follower)
0 1 2 3
Partition 0 (Follower)
0 1 2 3
Partition 1 (Leader)
0 1 2 3

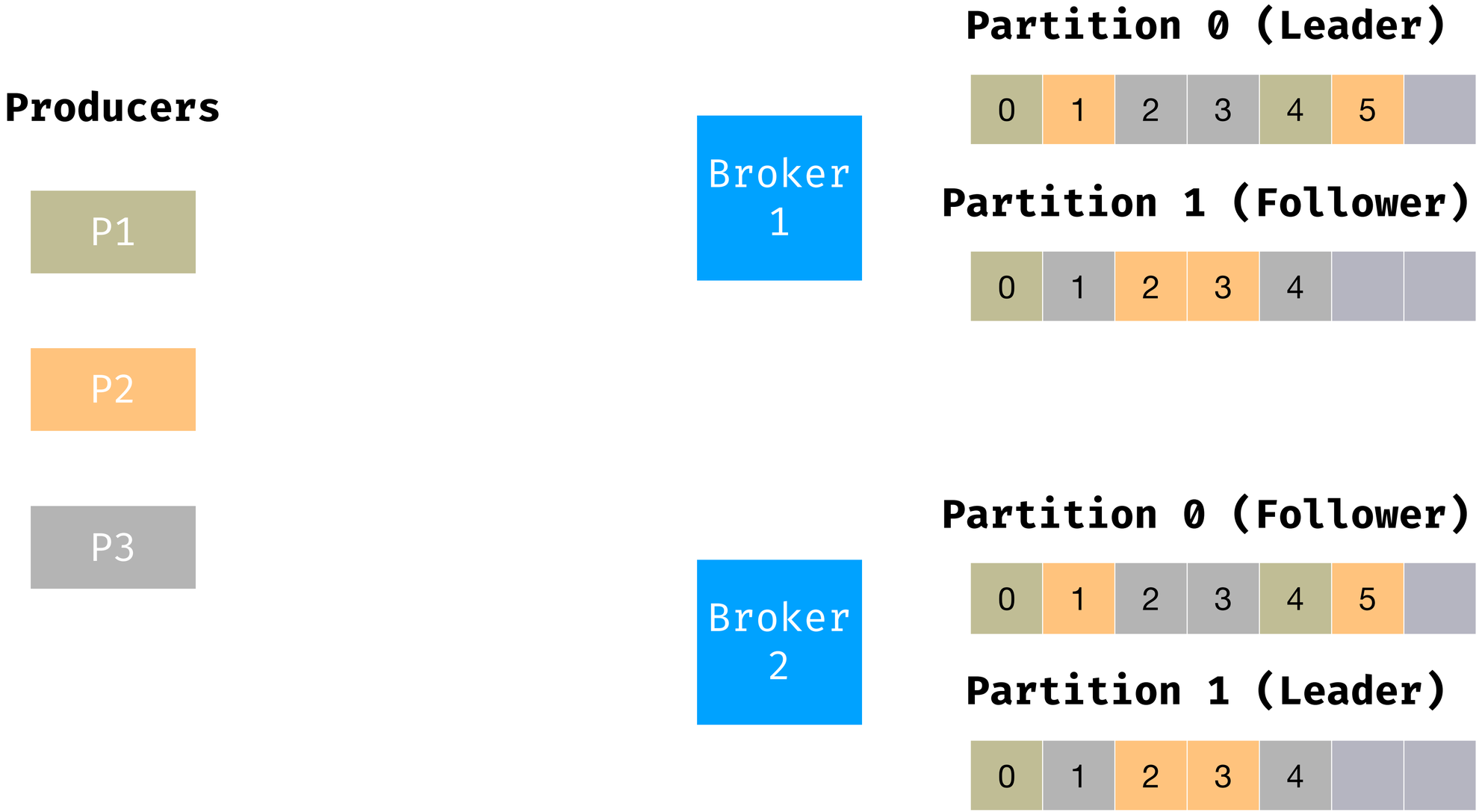

Producers
P1
P2
P3
Broker 2
Partition 0 (Leader)
0 1 2 3
Partition 1 (Follower)
0 1 2 3
Partition 0 (Follower)
0 1 2 3
Partition 1 (Leader)
0 1 2 3

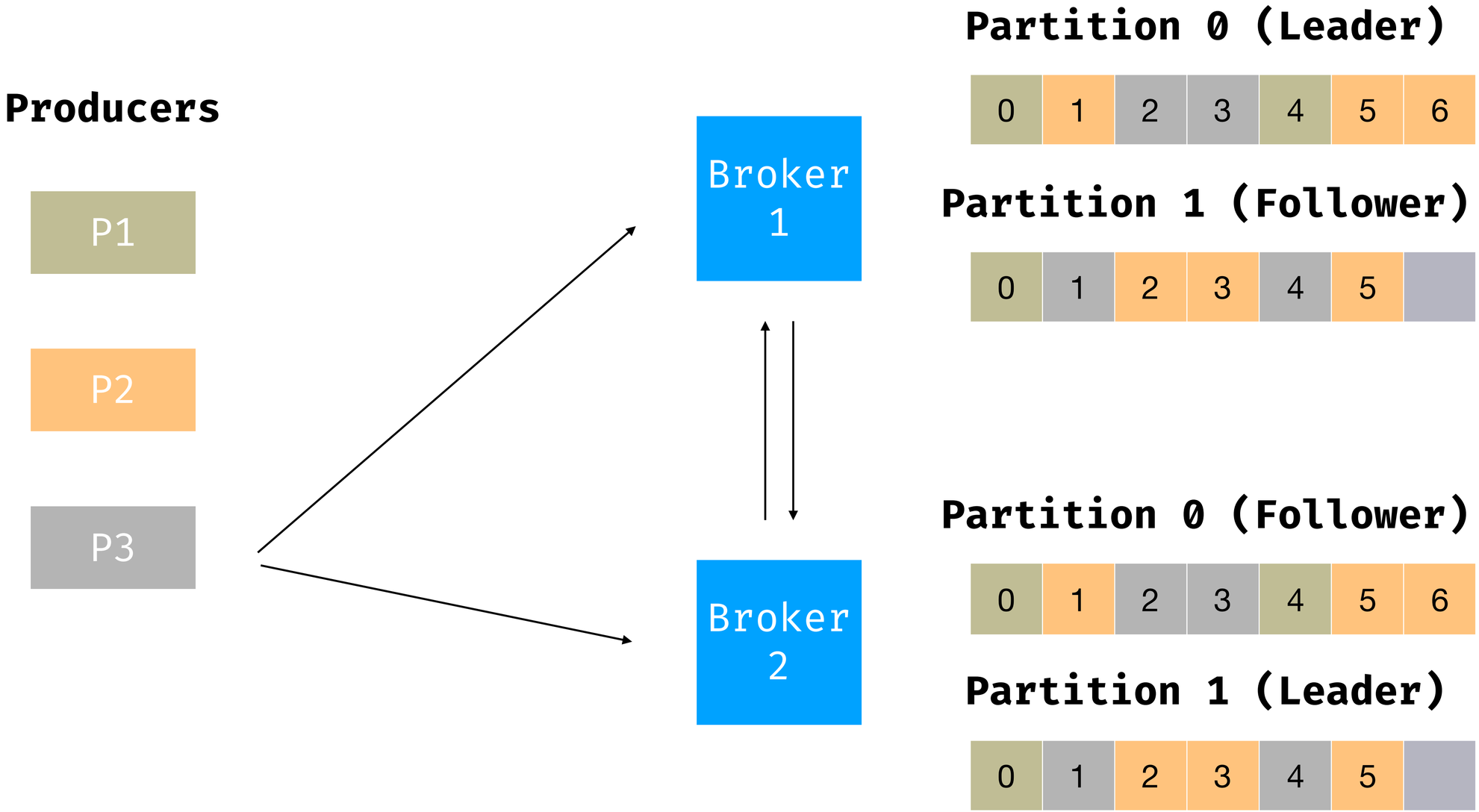

Producers
P1
P2
P3
Partition 0 (Leader)
0
1
2
3
Partition 1 (Follower)
0
1
2
3
Broker 2
Partition 0 (Leader)
0
1
2
3
Partition 1 (Leader)
0
1
2
3


Producers
P1
P2
P3
Broker 2
Partition 0 (Leader)
0 1 2 3
Partition 1 (Follower)
0 1 2 3
Partition 0 (Leader)
0 1 2 3 4 5
Partition 1 (Leader)
0 1 2 3 4


Producers
P1
P2
P3
Broker 1
Broker 2
Partition 0 (Follower)
0 1 2 3
Partition 1 (Follower)
0 1 2 3
Partition 0 (Leader)
0 1 2 3 4 5
Partition 1 (Leader)
0 1 2 3 4

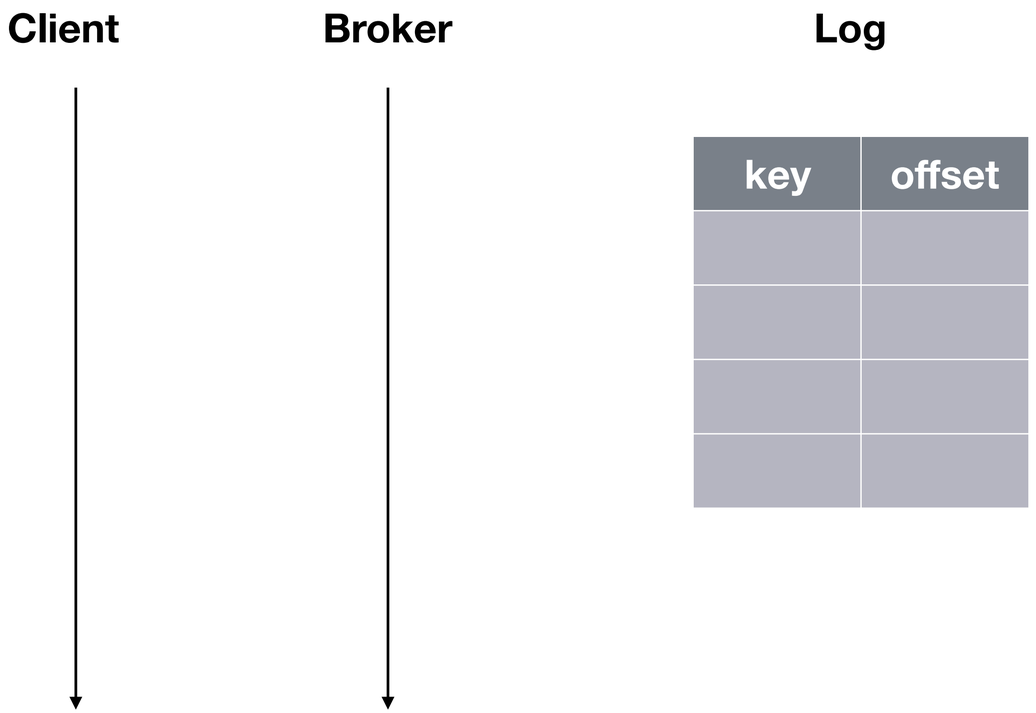

Producers
P1
P2
P3
Broker 1
Broker 2
Partition 0 (Follower)
0 1 2 3 4 5
Partition 1 (Follower)
0 1 2 3 4
Partition 0 (Leader)
0 1 2 3 4 5
Partition 1 (Leader)
0 1 2 3 4


Producers
P1
P2
P3
Broker 1
Partition 0 (Leader)
0 1 2 3 4 5
Partition 1 (Follower)
0 1 2 3 4
Broker 2
Partition 0 (Follower)
0 1 2 3 4 5
Partition 1 (Leader)
0 1 2 3 4


Producers
P1
P2
P3
Broker 1
Broker 2
Partition 0 (Leader)
0 1 2 3 4 5 6
Partition 1 (Follower)
0 1 2 3 4 5
Partition 0 (Follower)
0 1 2 3 4 5 6
Partition 1 (Leader)
0 1 2 3 4 5

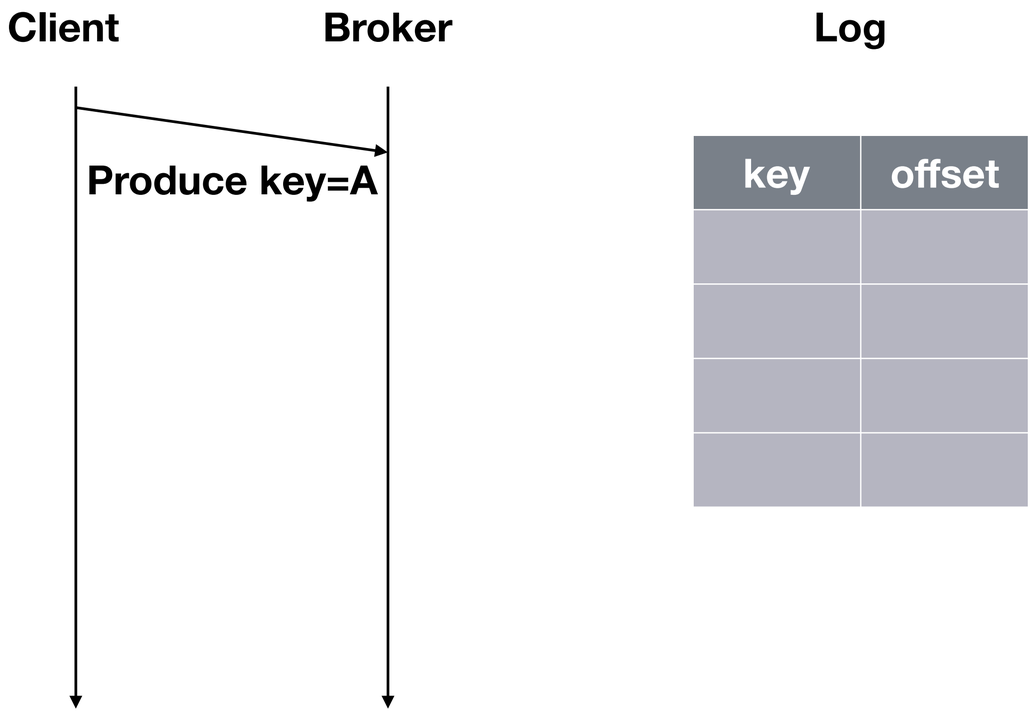
Протокол продюсера

- Для записи в партиции
- Ack означает успешный append



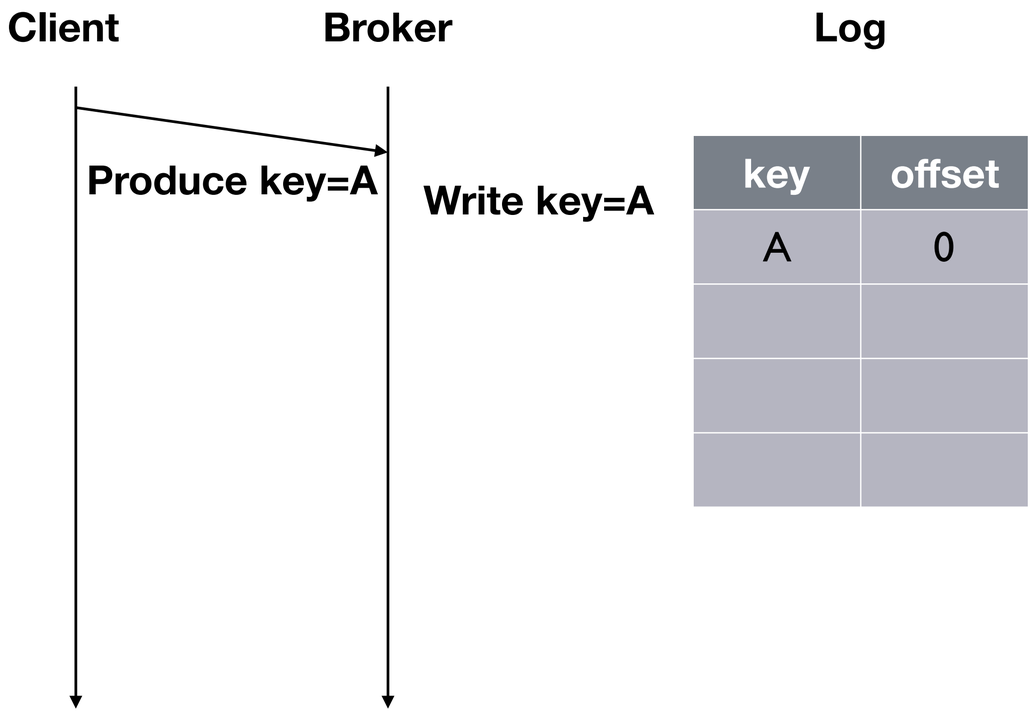
Протокол продюсера

- Для записи в партиции
- Ack означает успешный append



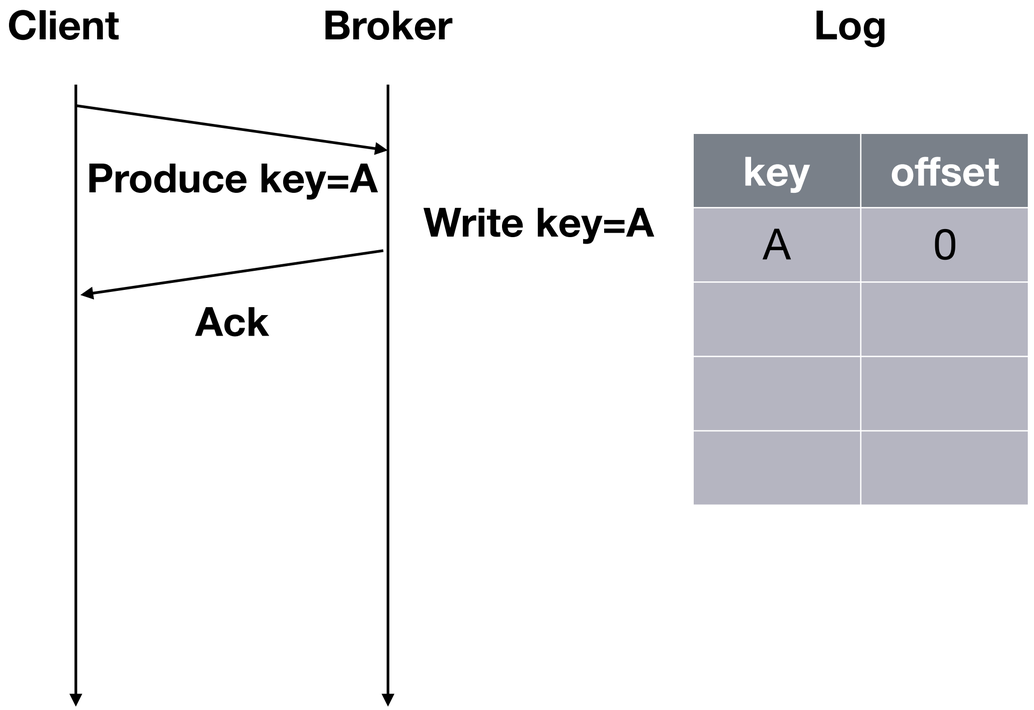
Протокол продюсера

- Для записи в партиции
- Ack означает успешный append



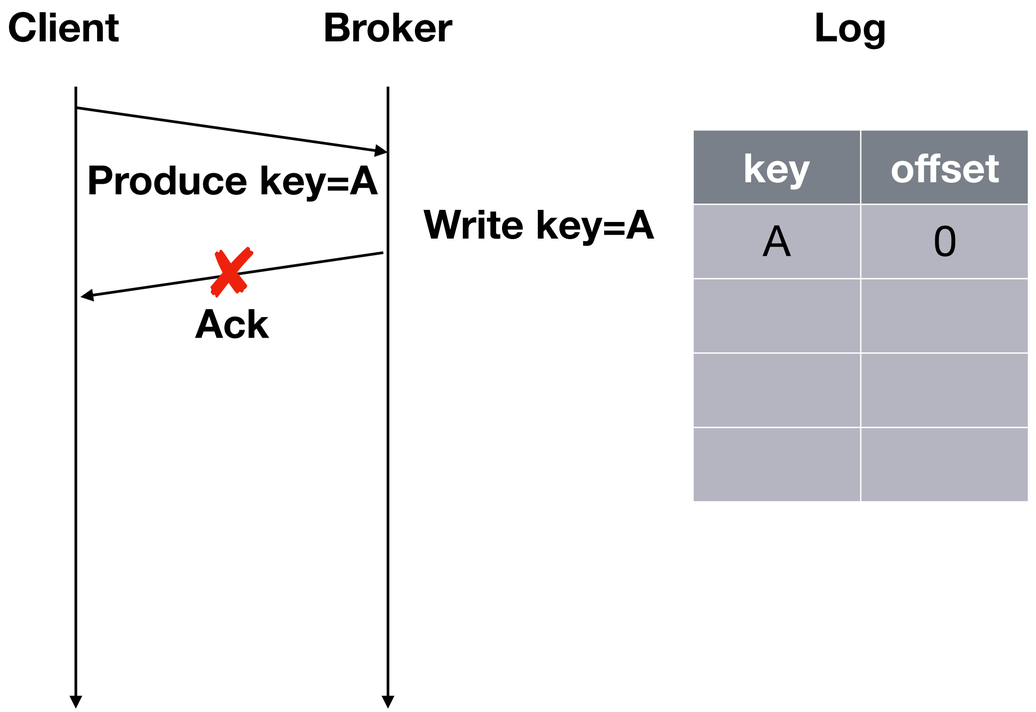
Протокол продюсера

- Для записи в партиции
- Ack означает успешный append



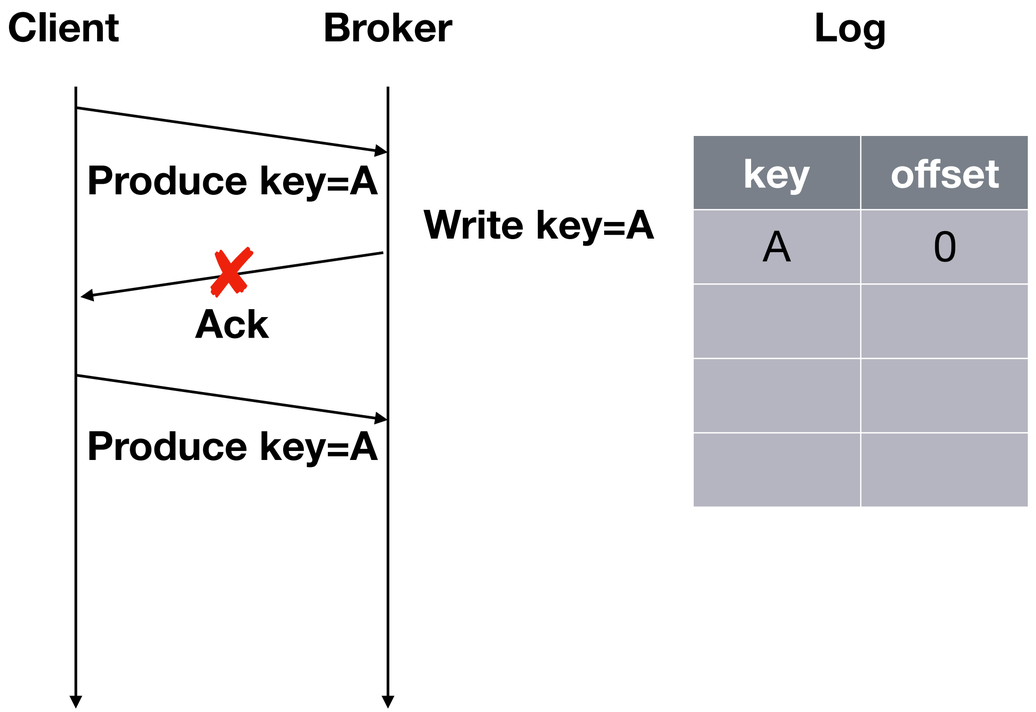
Протокол продюсера

- Для записи в партиции
- Ack означает успешный append



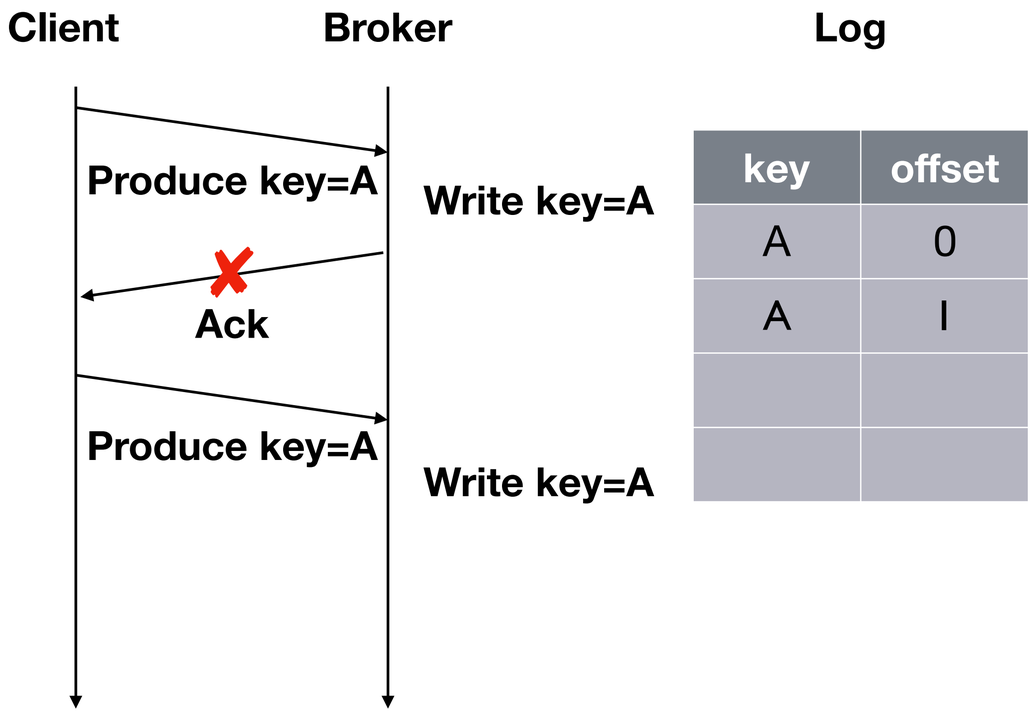
Протокол продюсера

- Для записи в партиции
- Ack означает успешный append



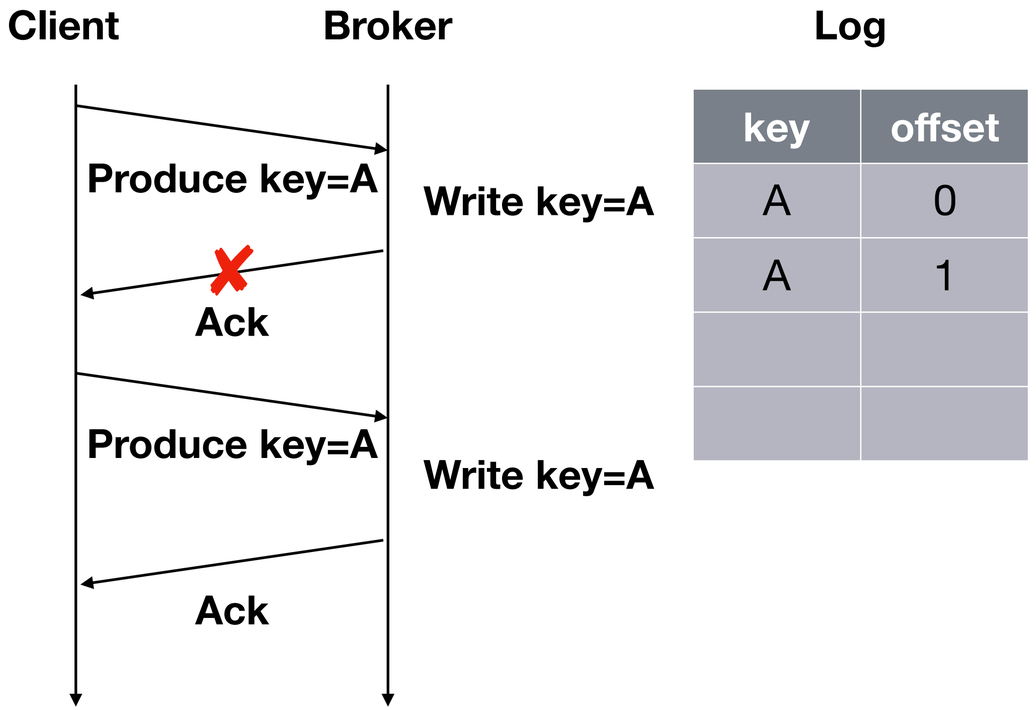
Протокол продюсера

- Для записи в партиции
- Ack означает успешный append

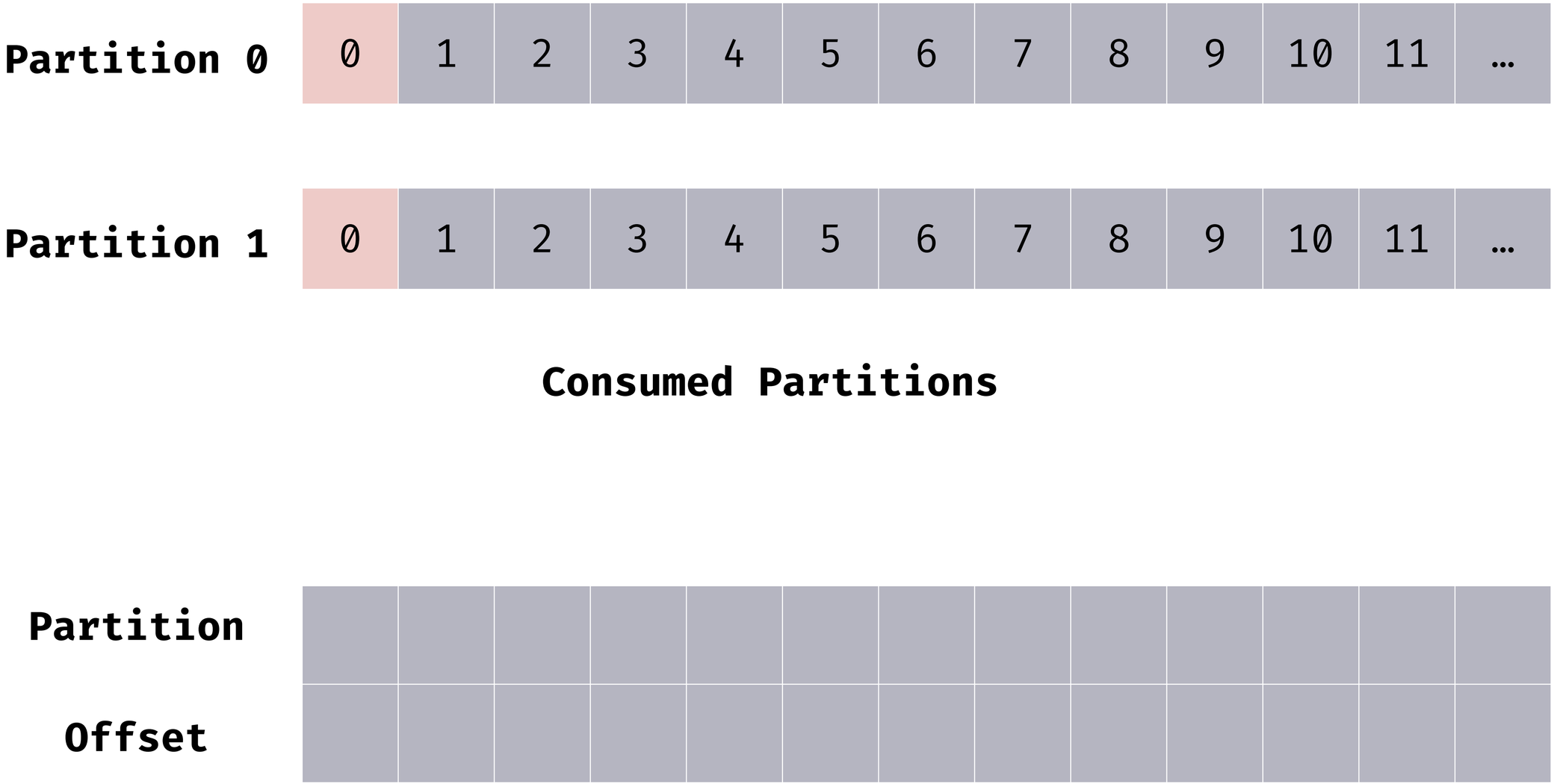


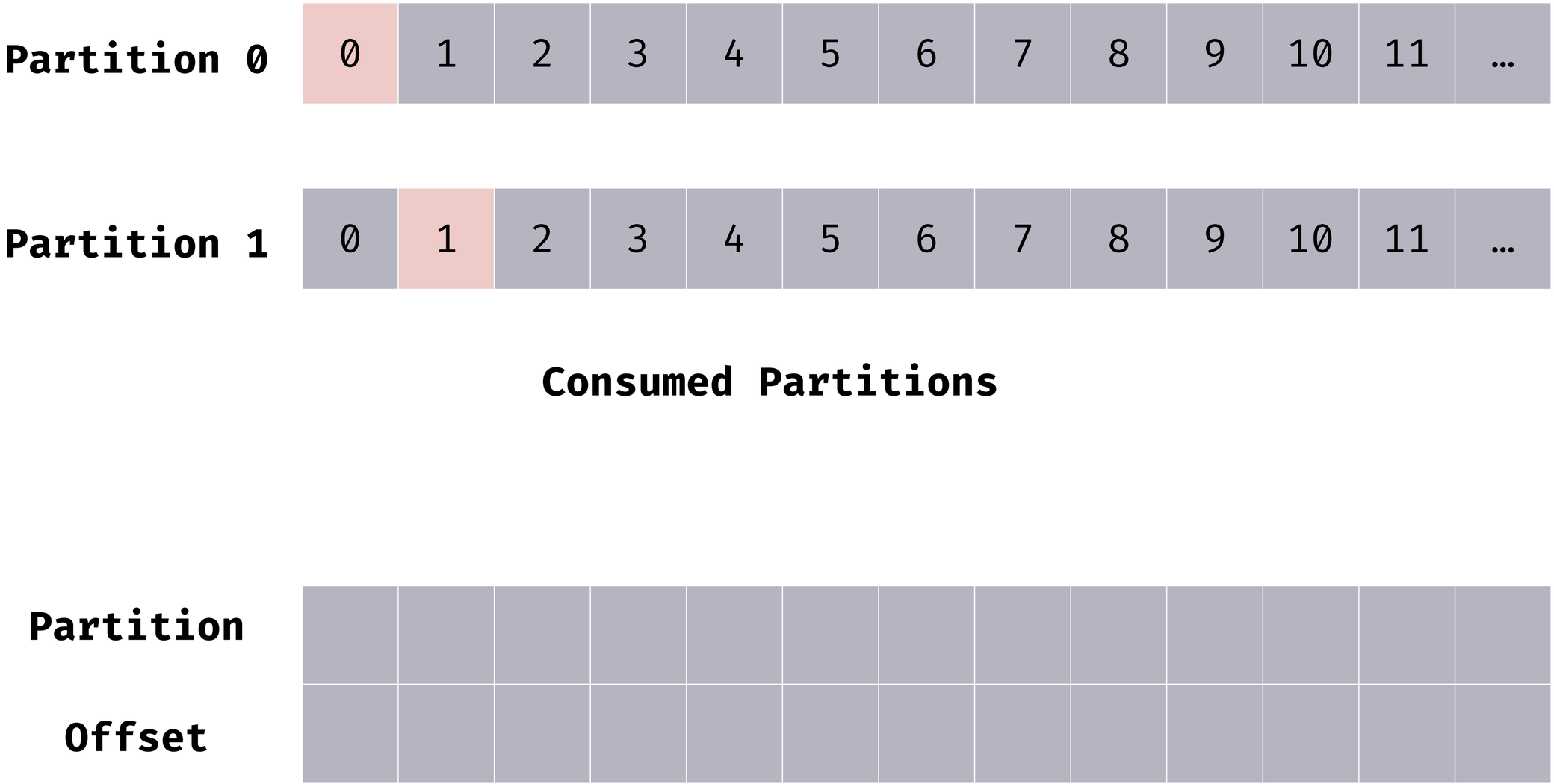
Протокол продюсера

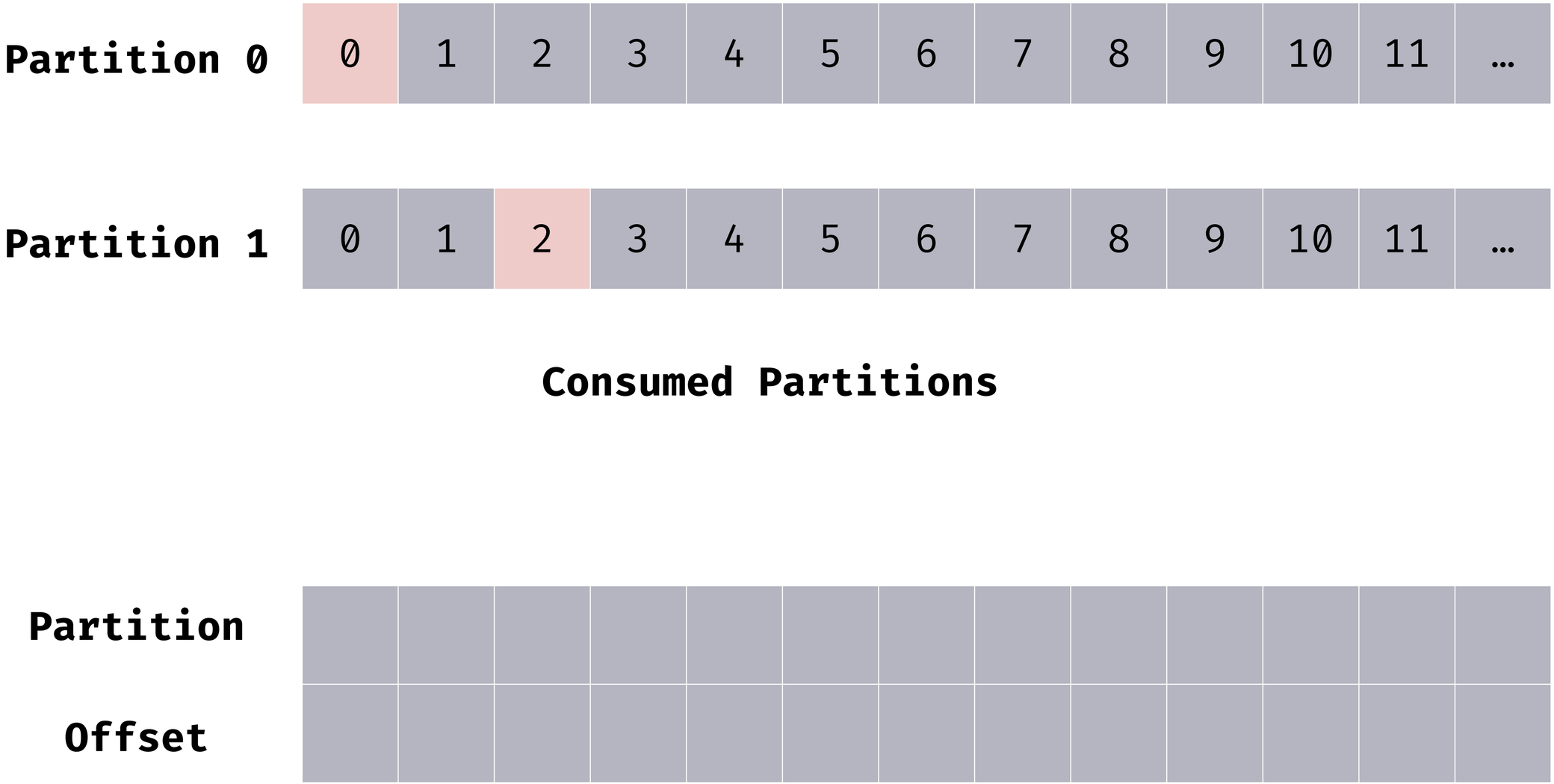
- Для записи в партиции
- Ack означает успешный append

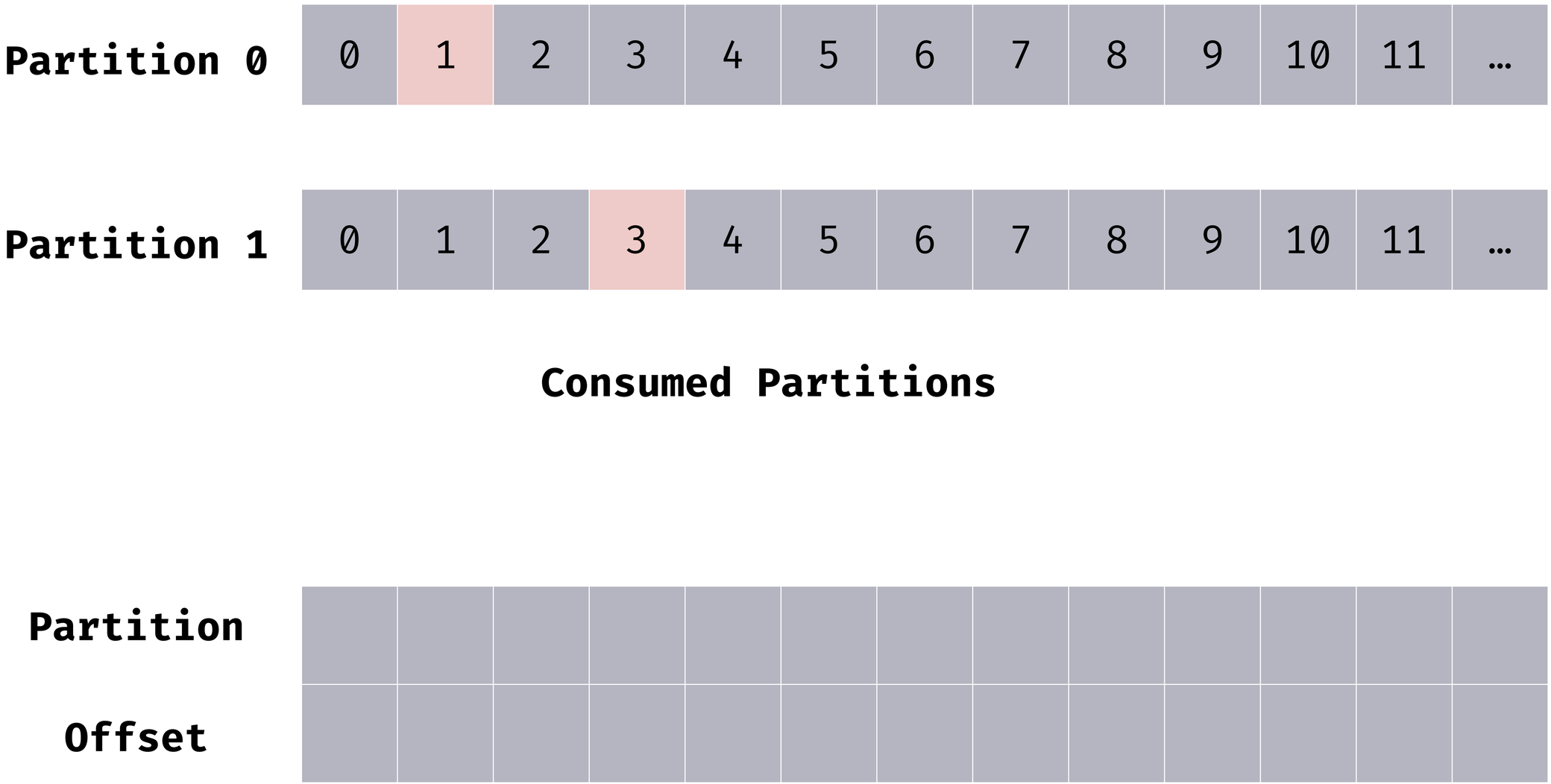


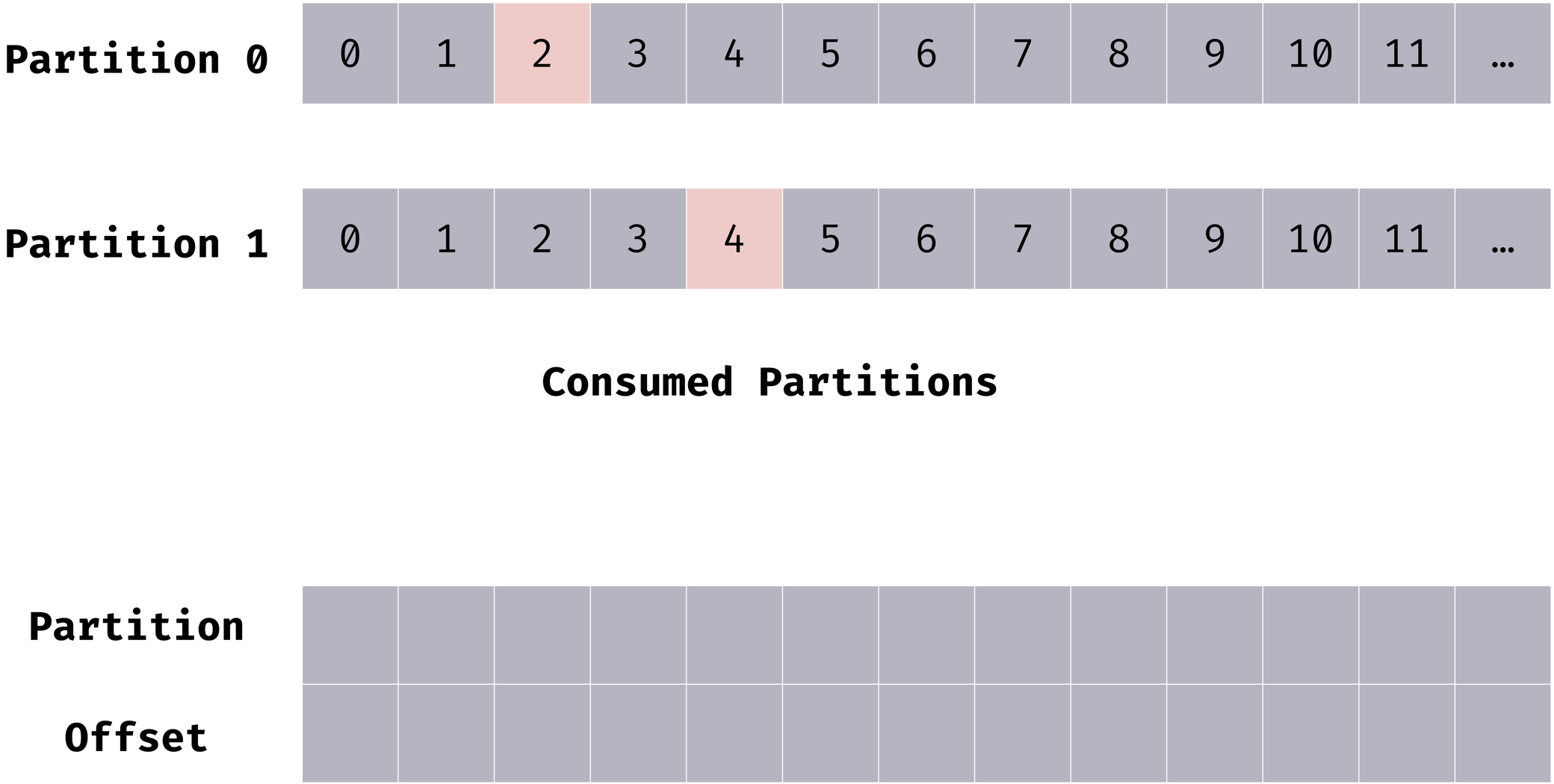
Kafka Consumer

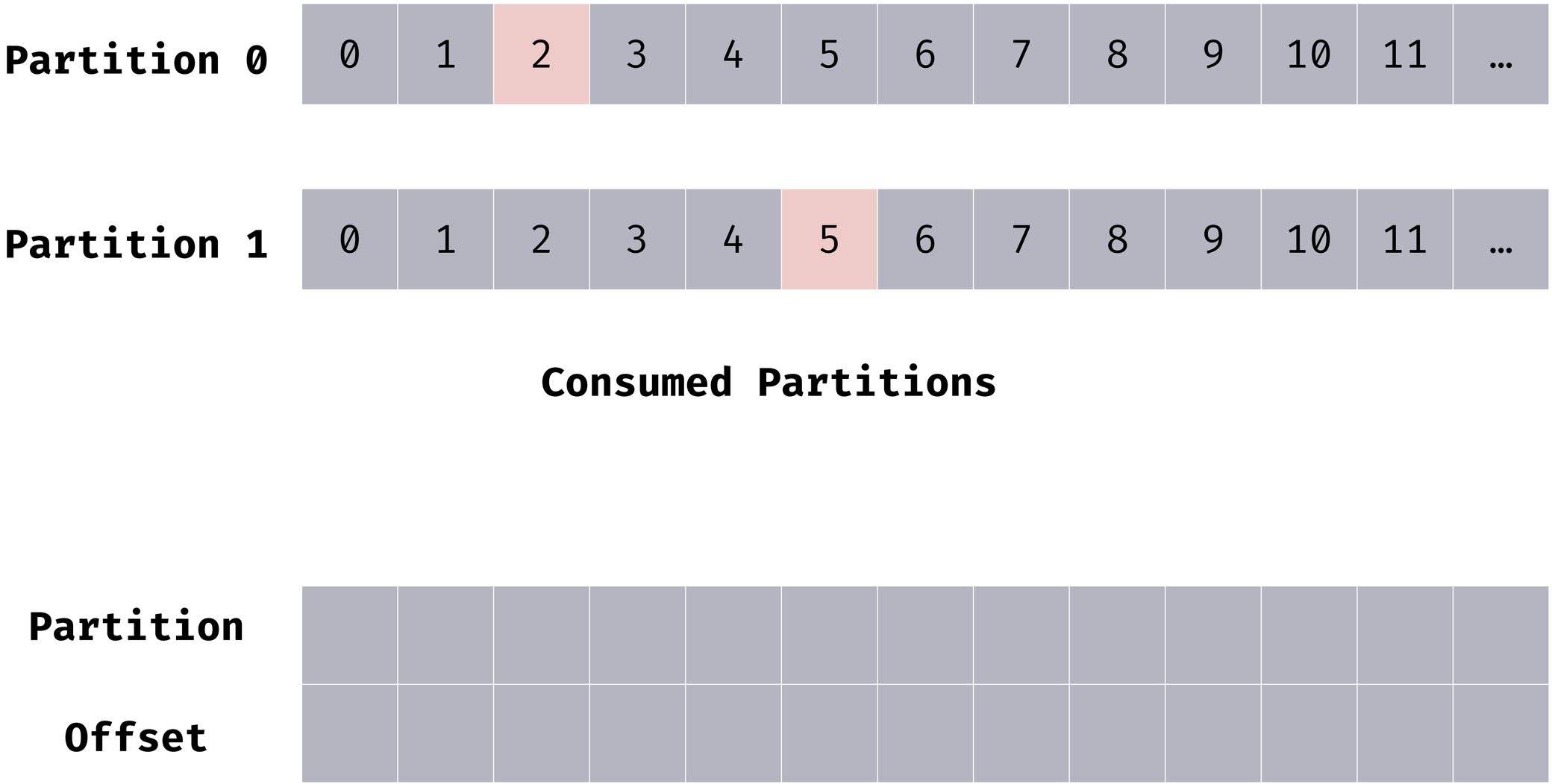

Partition 0
0 1 2 3 4 5 6 7 8 9 10 11 …
Partition 1
0 1 2 3 4 5 6 7 8 9 10 11 …
Consumed Partitions
Partition
Offset
Committed Offsets

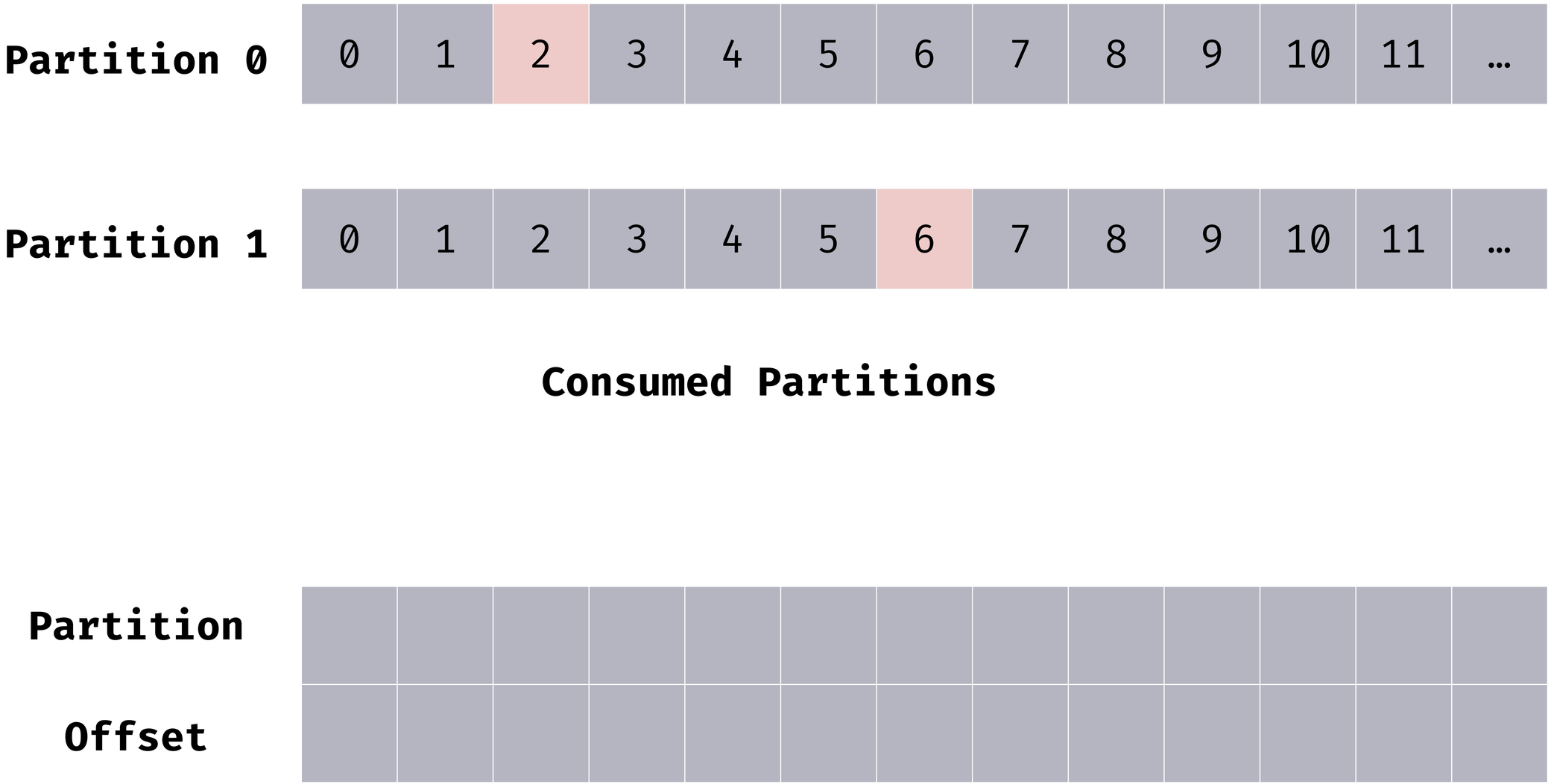

Partition 0
0 1 2 3 4 5 6 7 8 9 10 11 …
Partition 1
0 1 2 3 4 5 6 7 8 9 10 11 …
Consumed Partitions
Partition
Offset
Committed Offsets

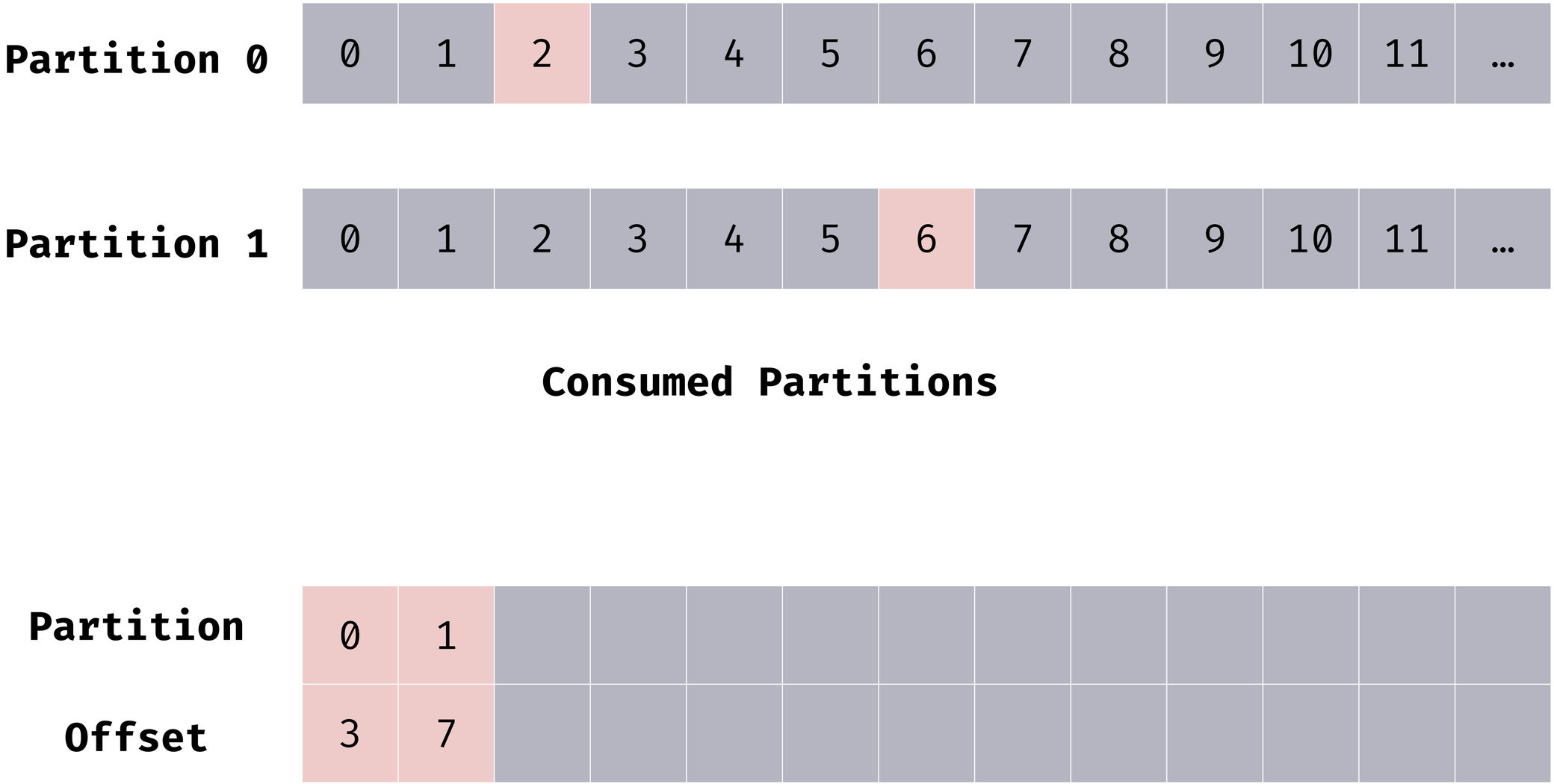

Partition 0
0 1 2 3 4 5 6 7 8 9 10 11 …
Partition 1
0 1 2 3 4 5 6 7 8 9 10 11 …
Consumed Partitions
Partition
Offset
Committed Offsets

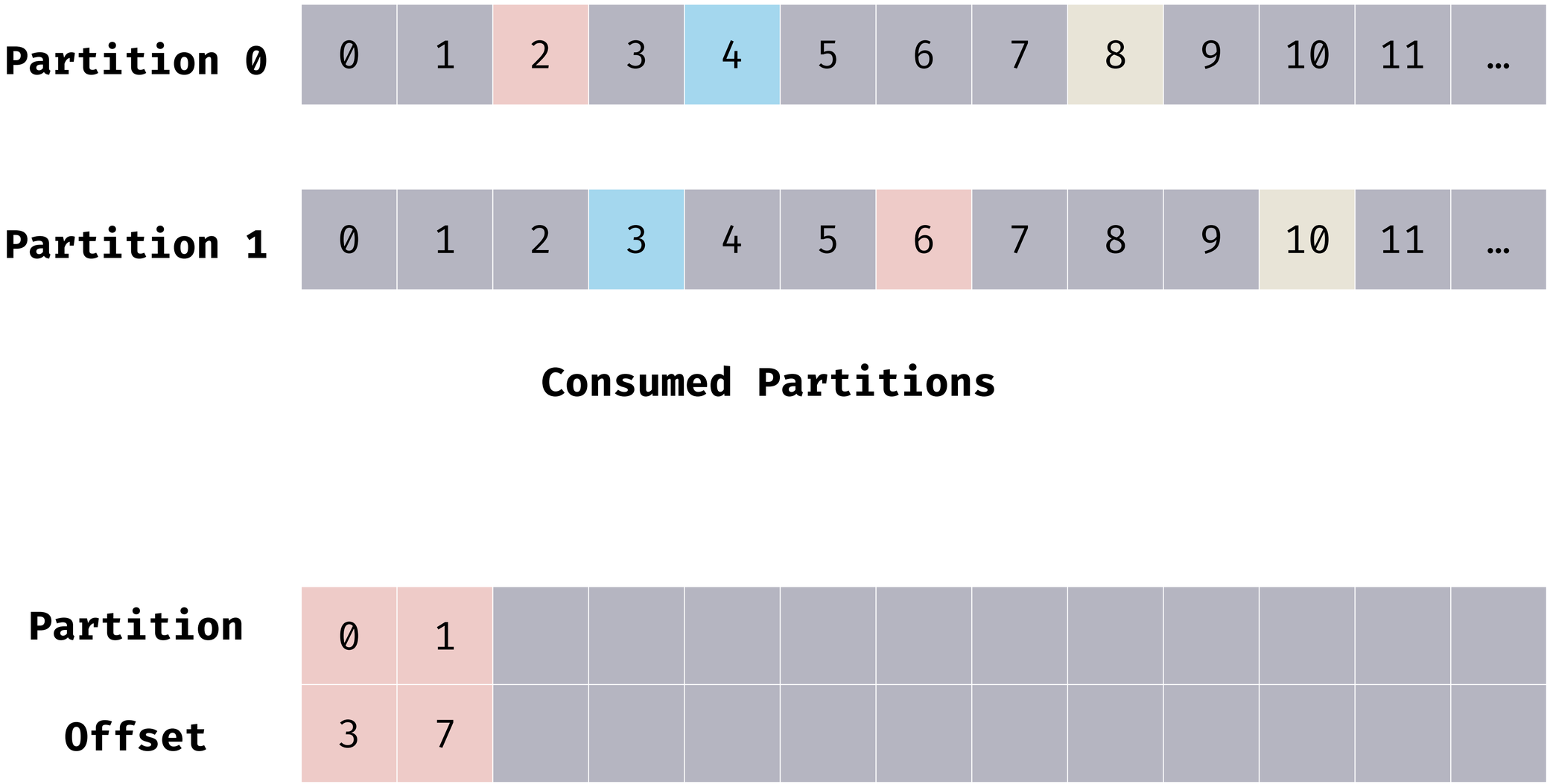

Partition 0
0
1
2
3
4
5
6
7
8
9
10
11
…
Partition 1
0
1
2
3
4
5
6
7
8
9
10
11
…
Consumed Partitions
Partition
Offset
Committed Offsets

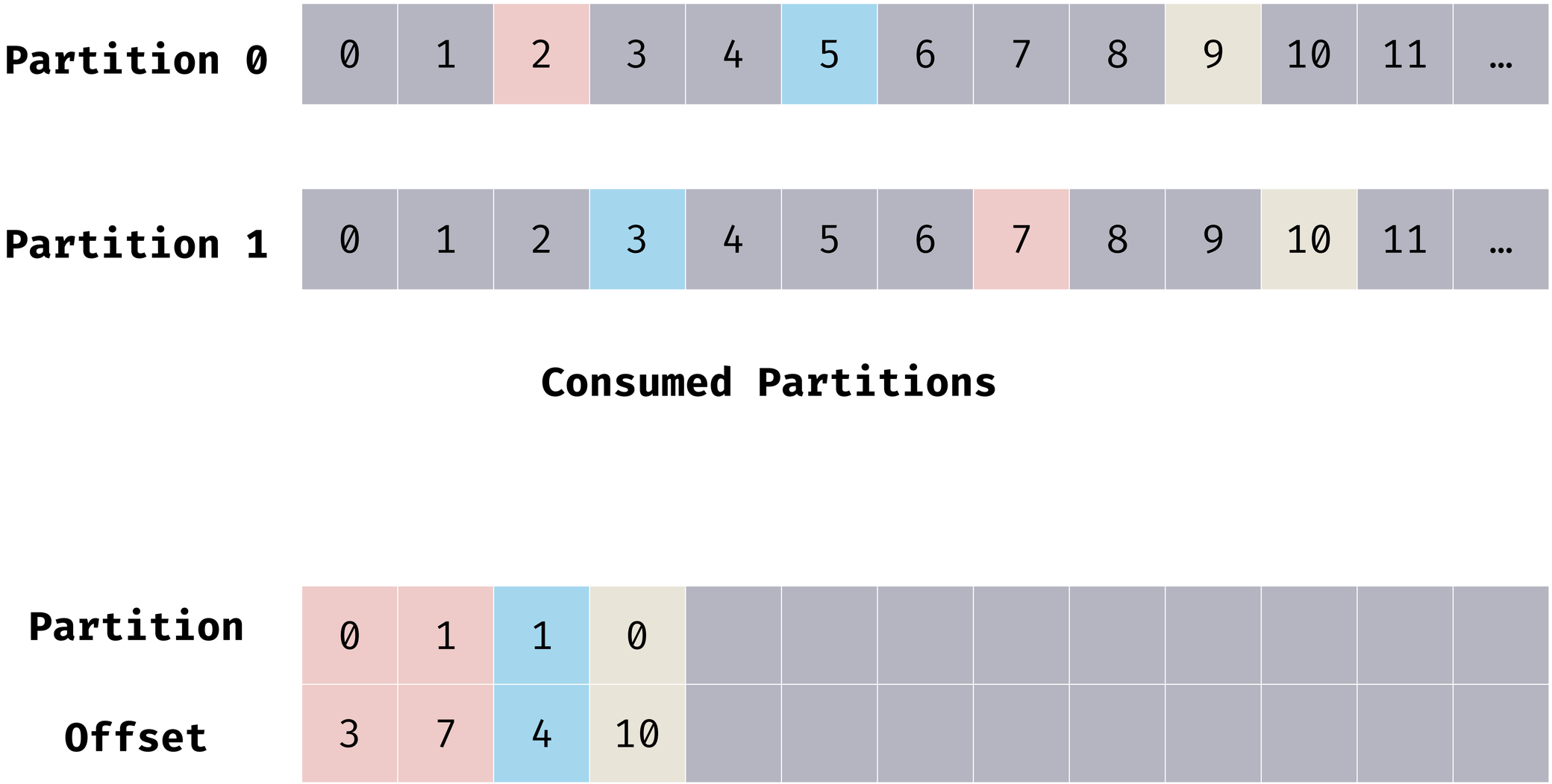

Partition 0
0 1 2 3 4 5 6 7 8 9 10 11 …
Partition 1
0 1 2 3 4 5 6 7 8 9 10 11 …
Consumed Partitions
Partition
Offset
Committed Offsets

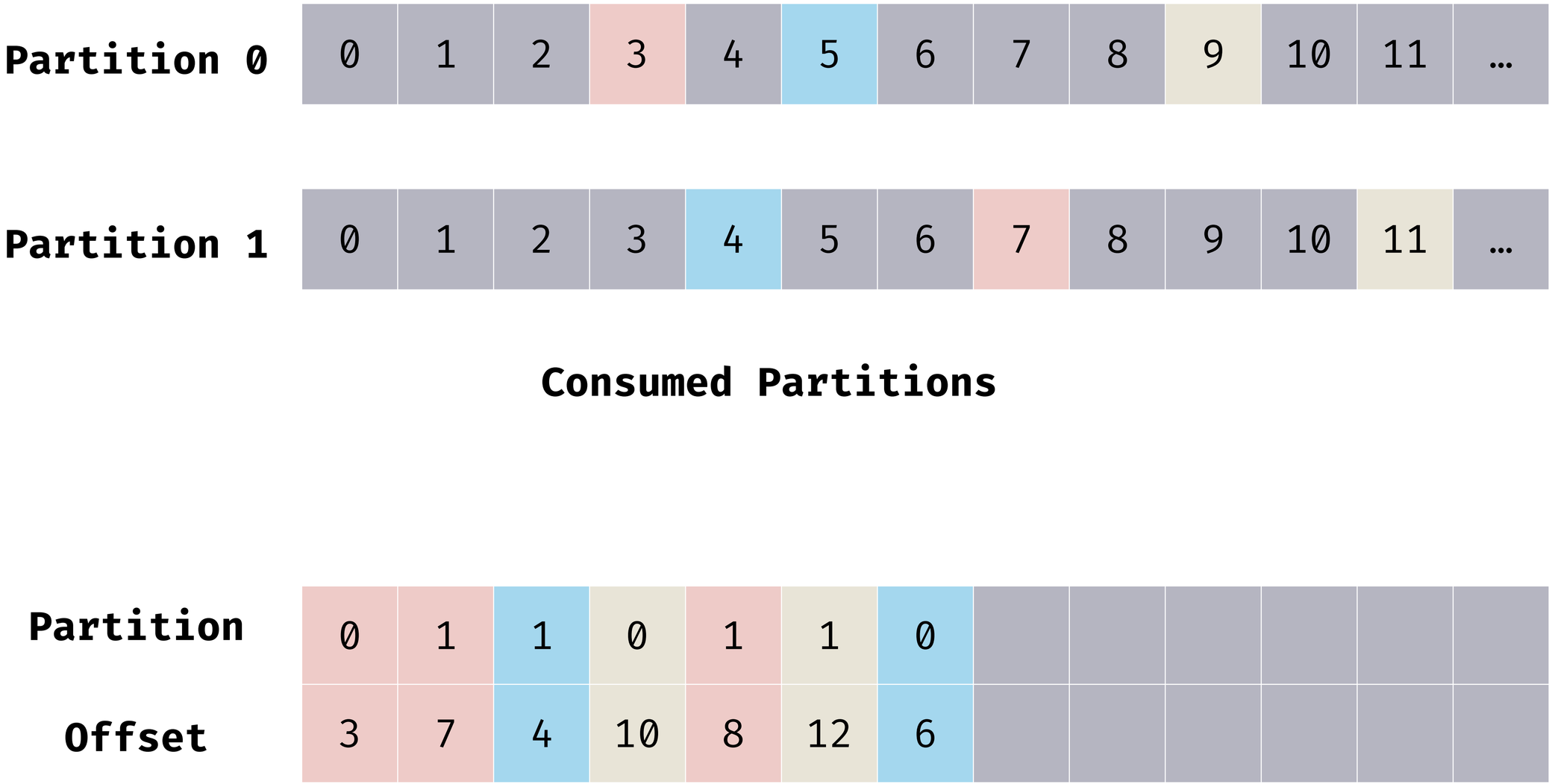

Partition 0
0 1 2 3 4 5 6 7 8 9 10 11 …
Partition 1
0 1 2 3 4 5 6 7 8 9 10 11 …
Consumed Partitions
Partition
Offset
Committed Offsets

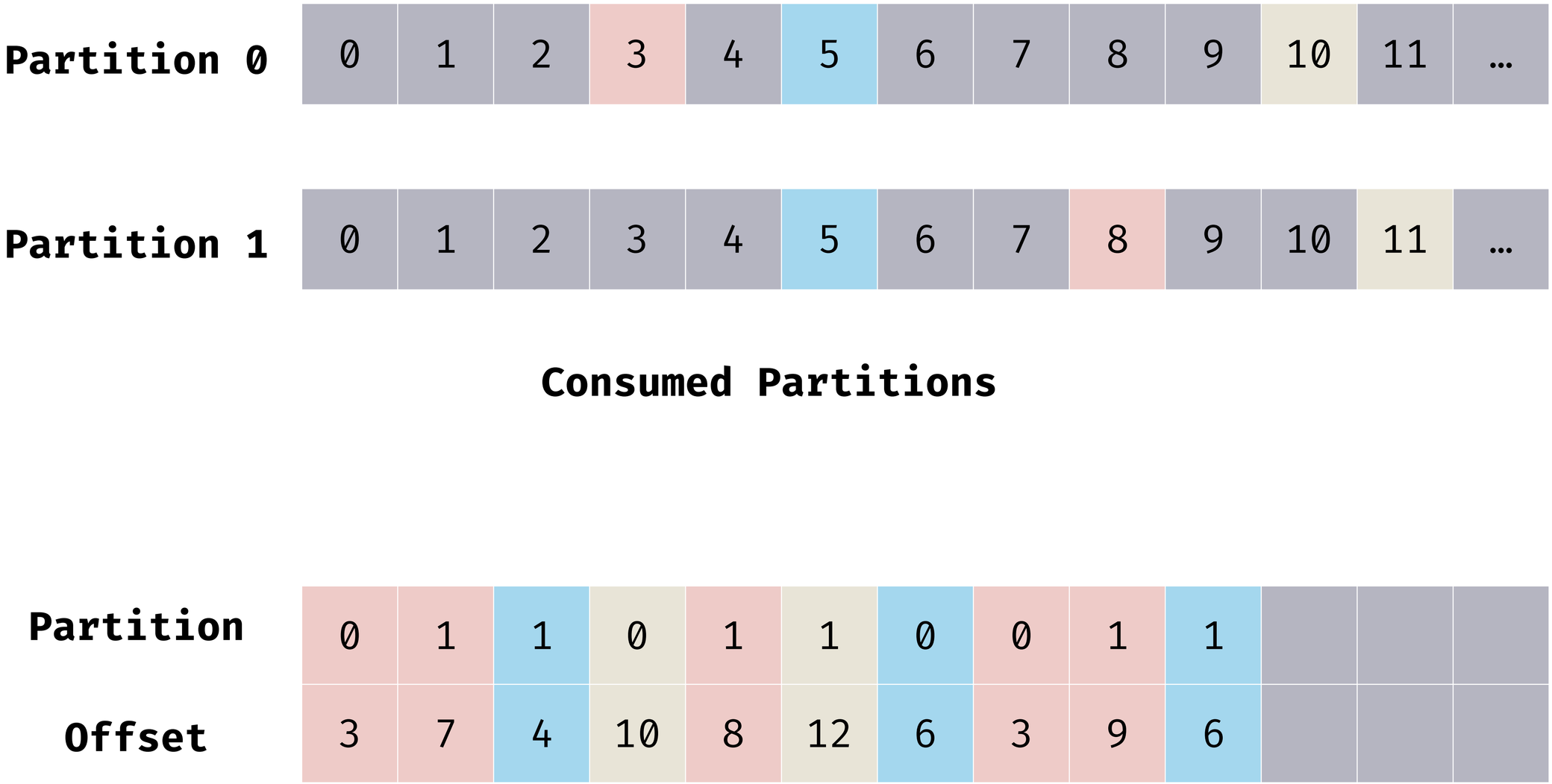

Partition 0
0 1 2 3 4 5 6 7 8 9 10 11 …
Partition 1
0 1 2 3 4 5 6 7 8 9 10 11 …
Consumed Partitions
Partition
Offset
Committed Offsets


Partition 0
0 1 2 3 4 5 6 7 8 9 10 11 …
Partition 1
0 1 2 3 4 5 6 7 8 9 10 11 …
Consumed Partitions
Partition
Offset
0 1
3 7
Committed Offsets


Partition 0
0 1 2 3 4 5 6 7 8 9 10 11 …
Partition 1
0 1 2 3 4 5 6 7 8 9 10 11 …
Consumed Partitions
Partition
Offset
0 1
3 7
Committed Offsets

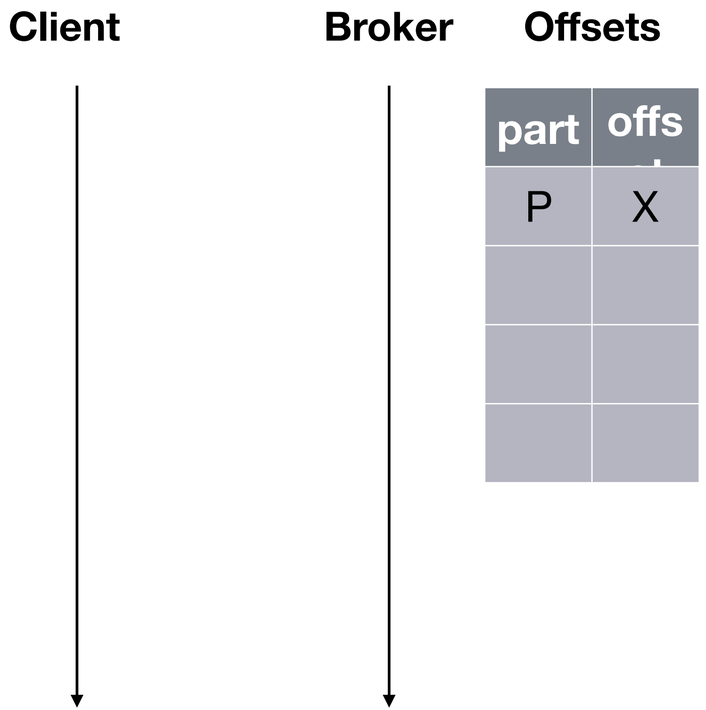

Partition 0
0 1 2 3 4 5 6 7 8 9 10 11 …
Partition 1
0 1 2 3 4 5 6 7 8 9 10 11 …
Consumed Partitions
Partition
Offset
0 1 1 0
3 7 4 10
Committed Offsets


Partition 0
0 1 2 3 4 5 6 7 8 9 10 11 …
Partition 1
0 1 2 3 4 5 6 7 8 9 10 11 …
Consumed Partitions
Partition
0 1 1 0 1 1 0
Offset
3 7 4 10 8 12 6
Committed Offsets


Partition 0
0 1 2 3 4 5 6 7 8 9 10 11 …
Partition 1
0 1 2 3 4 5 6 7 8 9 10 11 …
Consumed Partitions
Partition
0 1 1 0 1 1 0 0 1 1
Offset
3 7 4 10 8 12 6 3 9 6
Committed Offsets

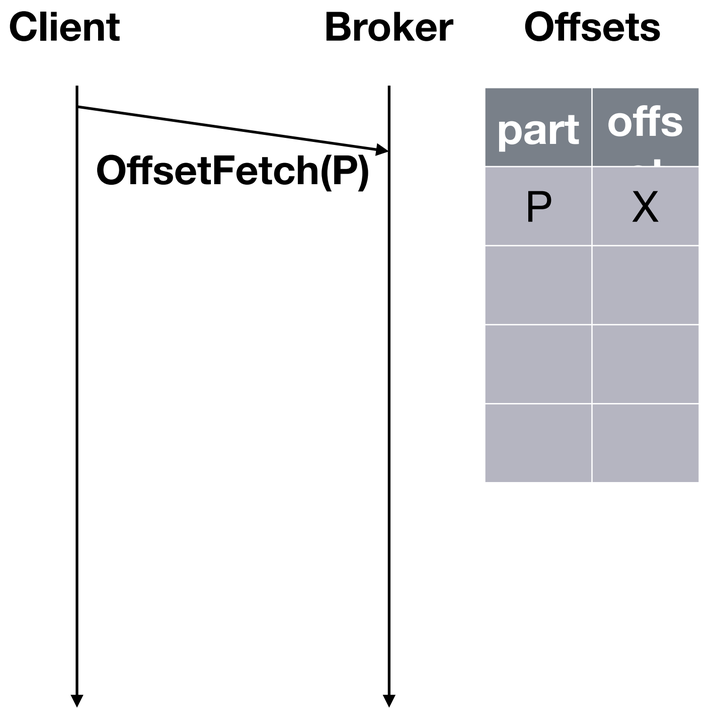
Протокол консьюмера

- Fetch API: взять диапазон записей по смещению (offset)
- OffsetCommit API: читать/записывать смещения (offset)



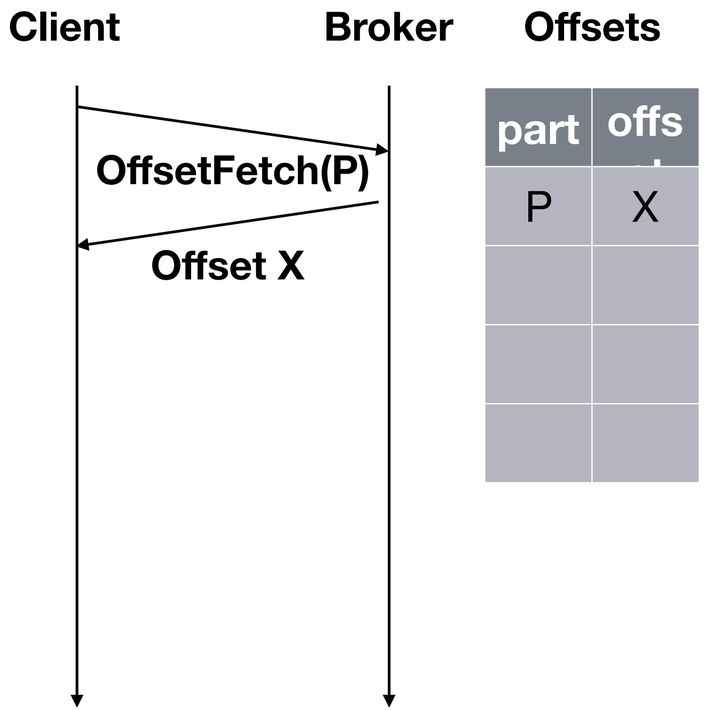
Протокол консьюмера

- Fetch API: взять диапазон записей по смещению (offset)
- OffsetCommit API: читать/записывать смещения (offset)



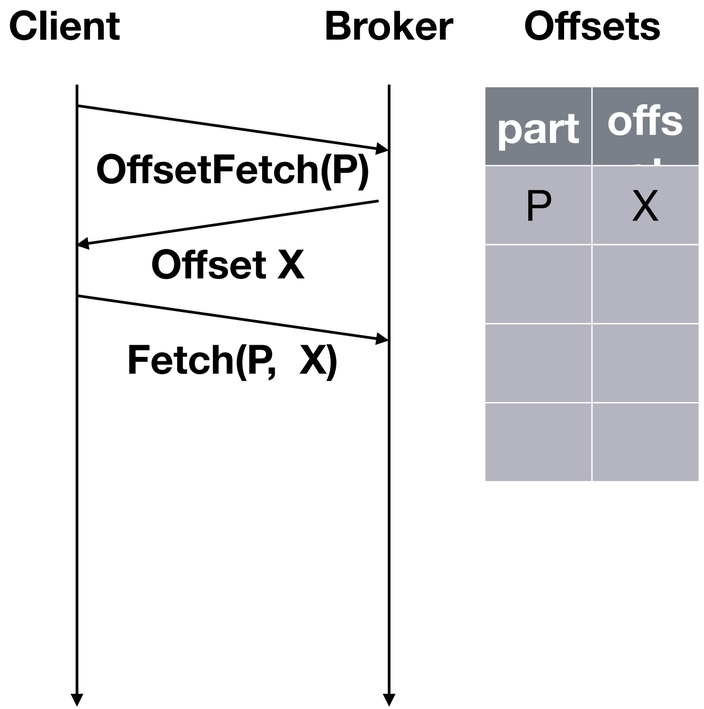
Протокол консьюмера

- Fetch API: взять диапазон записей по смещению (offset)
- OffsetCommit API: читать/записывать смещения (offset)



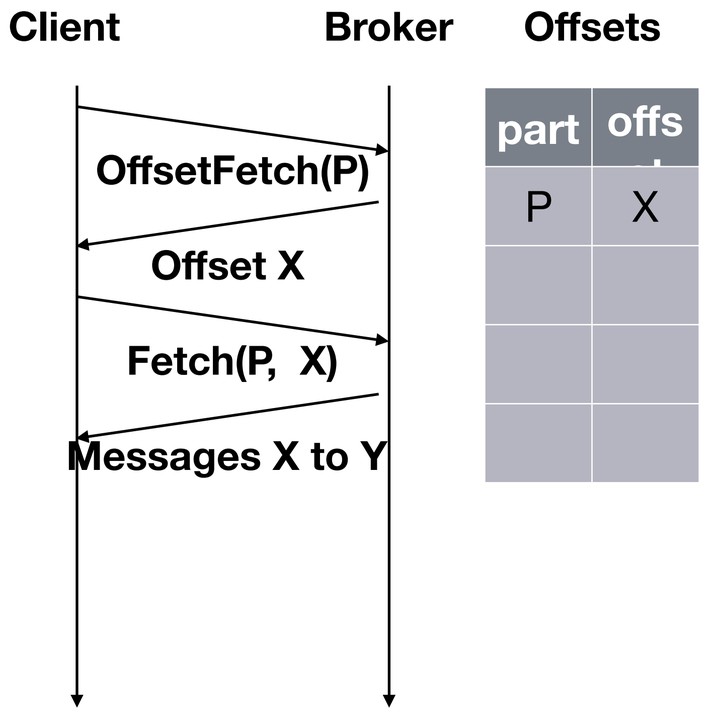
Протокол консьюмера

- Fetch API: взять диапазон записей по смещению (offset)
- OffsetCommit API: читать/записывать смещения (offset)



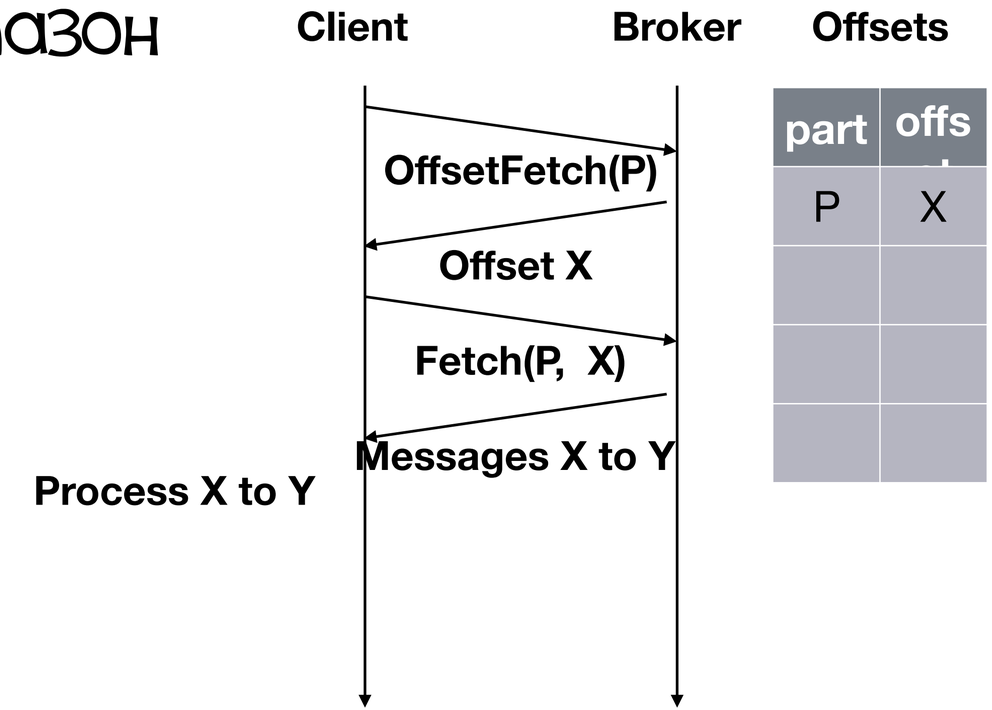
Протокол консьюмера

- Fetch API: взять диапазон записей по смещению (offset)
- OffsetCommit API: читать/записывать смещения (offset)



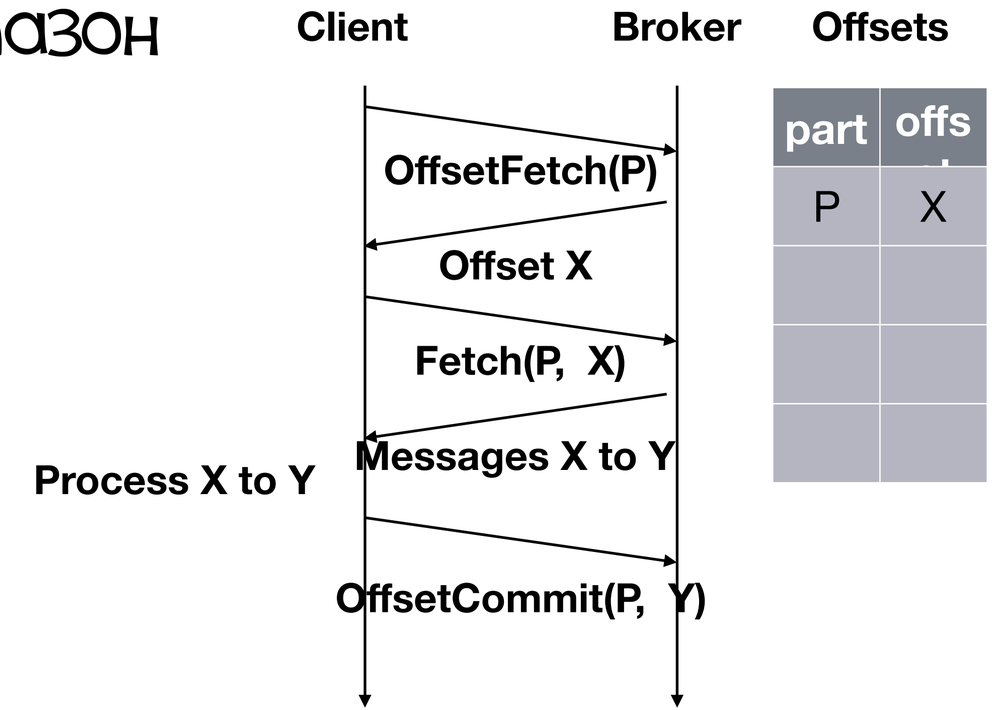
Протокол консьюмера

- Fetch API: взять диапазон записей по смещению (offset)
- OffsetCommit API: читать/записывать смещения (offset)



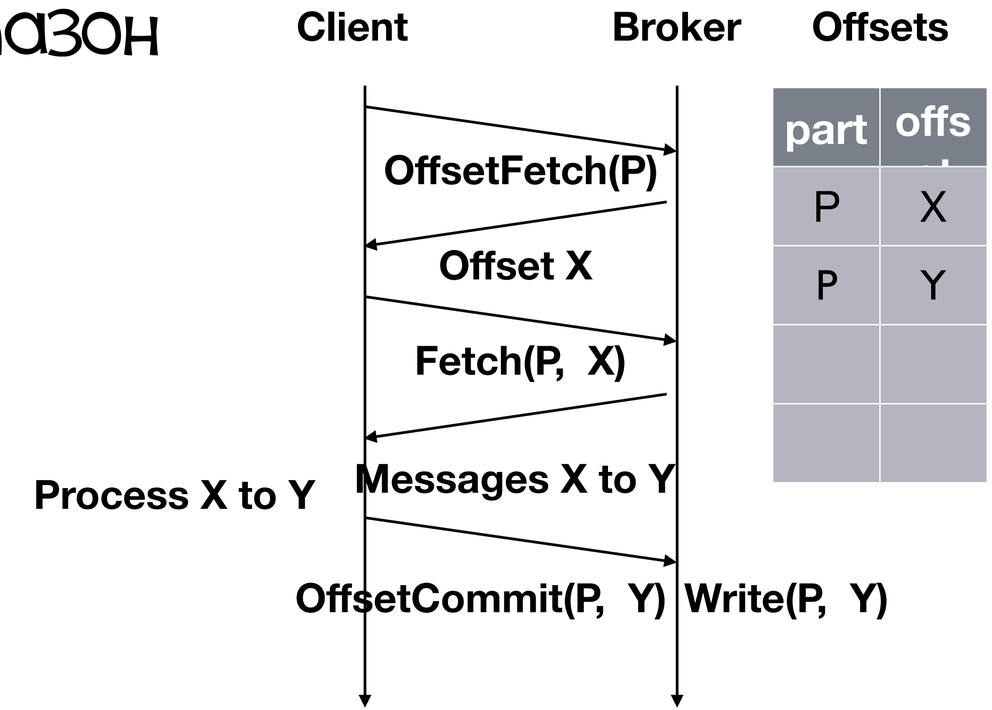
Протокол консьюмера

- Fetch API: взять диапазон записей по смещению (offset)
- OffsetCommit API: читать/записывать смещения (offset)



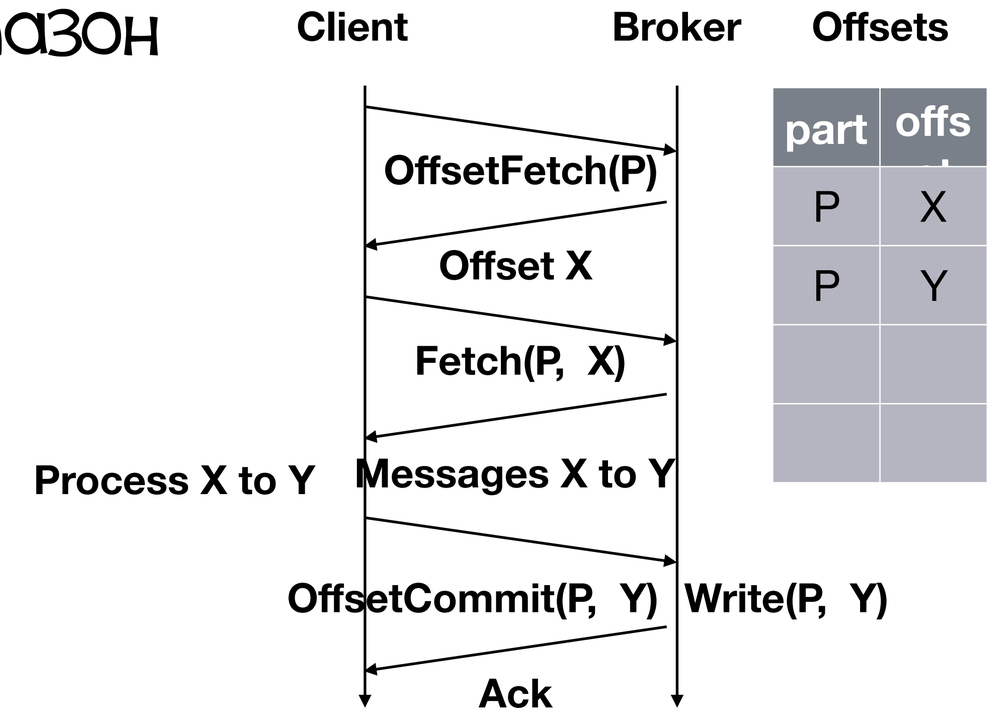
Протокол консьюмера

- Fetch API: взять диапазон записей по смещению (offset)
- OffsetCommit API: читать/записывать смещения (offset)



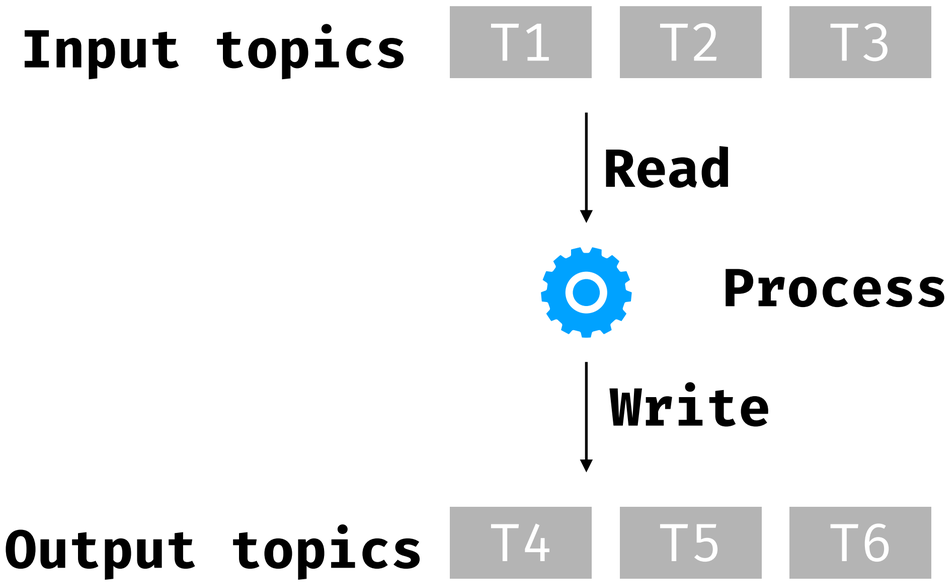
Протокол консьюмера

- Fetch API: взять диапазон записей по смещению (offset)
- OffsetCommit API: читать/записывать смещения (offset)



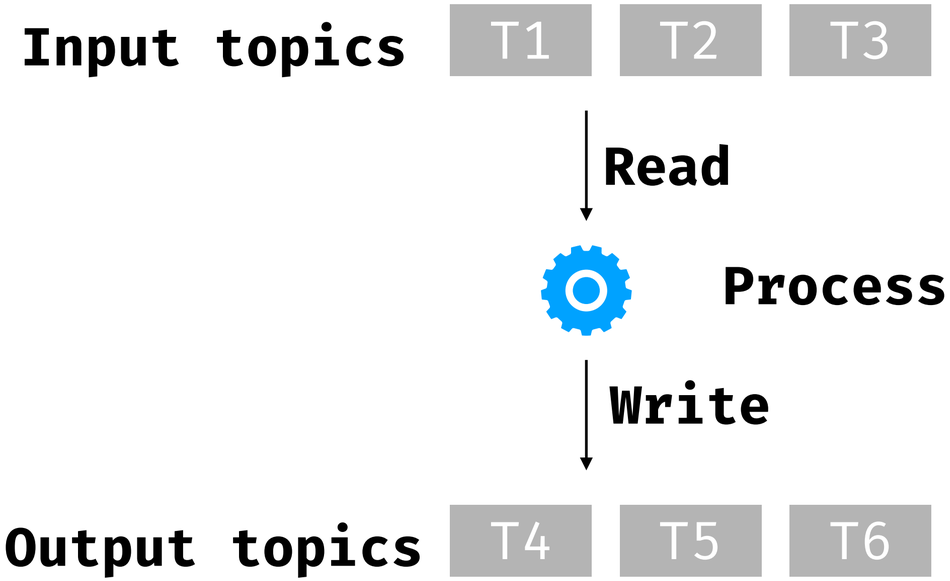
Модель обработки

- Прочитали из топика
- Что-то посчитали
- Записали результат в выходной топик



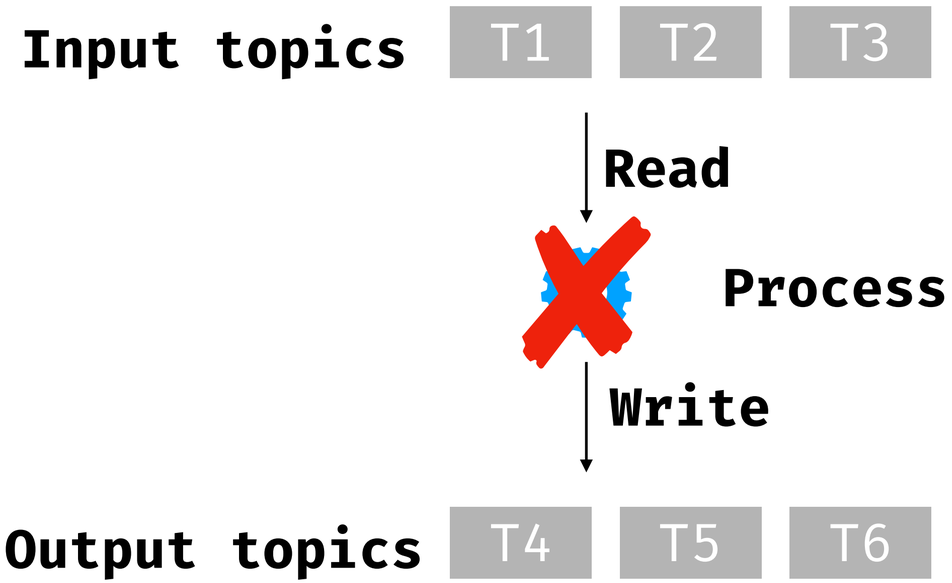
Модель обработки

- Виды сбоев
  - Все сломалось
  - Зомби



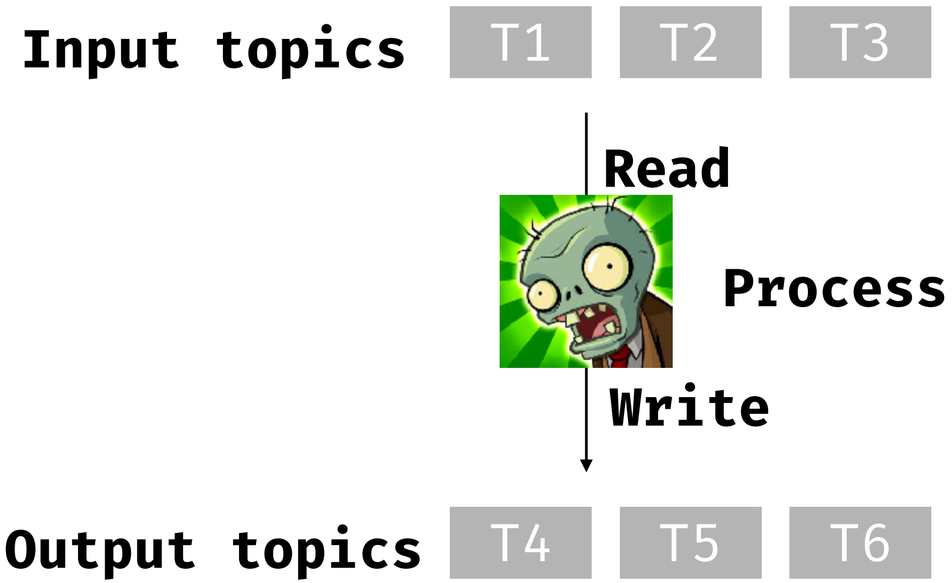
Модель обработки

- Виды сбоев
  - Все сломалось
  - Зомби



Модель обработки

- Виды сбоев
  - Все сломалось
  - Зомби



Семантики обработки сообщений

- Минимум один раз / At-least-once
- Максимум один раз / At-most-once: Вообще фигня какая-то!
- Точно один раз / Exact-once: То что вы на самом деле хотите

Семантика «Как минимум один раз»

```
c.subscribe(inputTopics)
while (true) {
  input = c.poll()
  output = process(input)
  p.send(output)
  c.commitOffsets()
}
```

Семантика «Как минимум один раз»

```
c.subscribe(inputTopics)
while (true) {
  input = c.poll()
  output = process(input)
  p.send(output)
  c.commitOffsets()
}
```

void subscribe(Collection<String> topics);

Семантика «Как минимум один раз»

```
c.subscribe(inputTopics)
while (true) {
  input = c.poll()
  output = process(input)
  p.send(output)
  c.commitOffsets()
}
```

ConsumerRecords<K, V> poll(long timeout);

Семантика «Как минимум один раз»

```
c.subscribe(inputTopics)
while (true) {
  input = c.poll()
  output = process(input)
  p.send(output)
  c.commitOffsets()
}
```

(Kafka Streams)

Семантика «Как минимум один раз»

```
c.subscribe(inputTopics)
while (true) {
  input = c.poll()
  output = process(input)
  p.send(output)
  c.commitOffsets()
}
```

Future<RecordMetadata> send(ProducerRecord<K, V> r);

Семантика «Как минимум один раз»

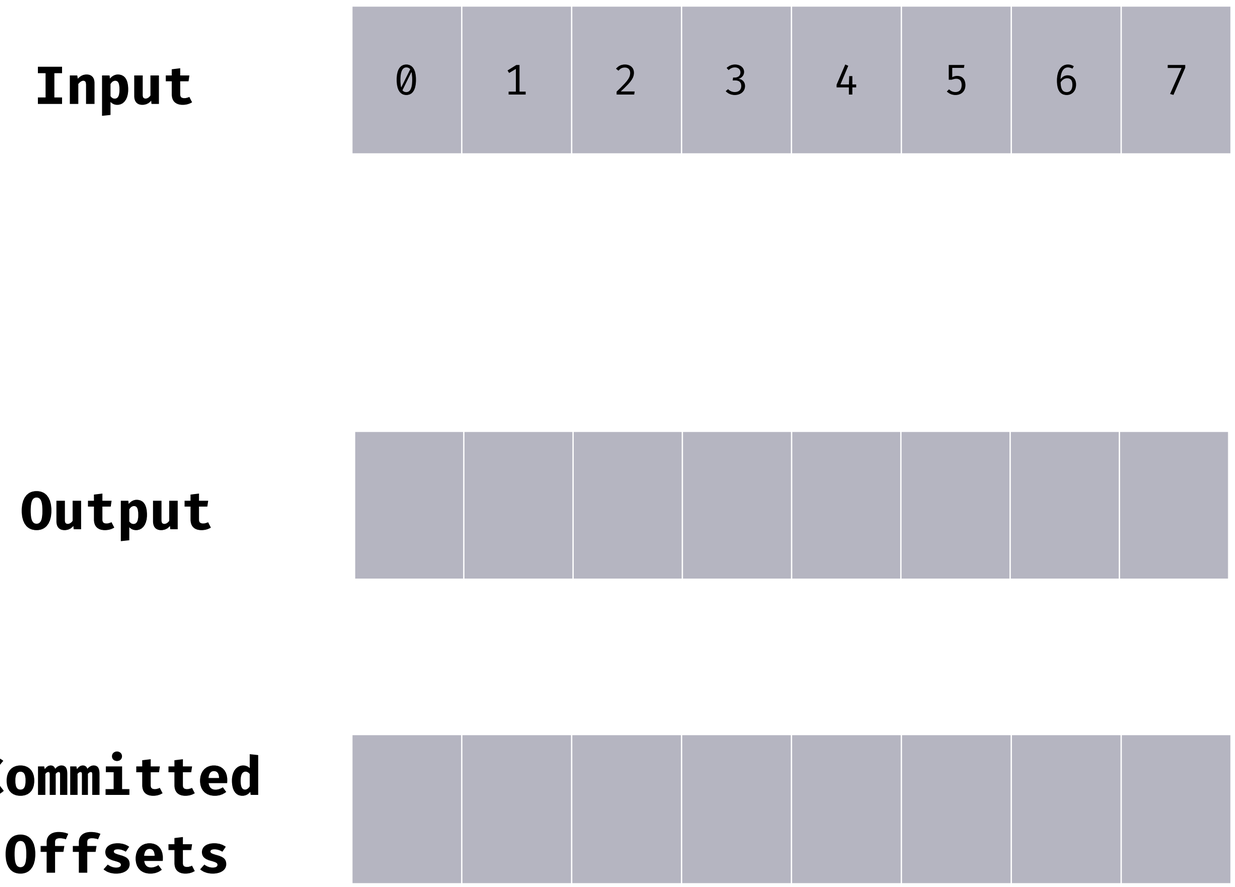
```
c.subscribe(inputTopics)
while (true) {
   input = c.poll()
   output = process(input)
   p.send(output)
   c.commitOffsets()
}
```

void commitSync();

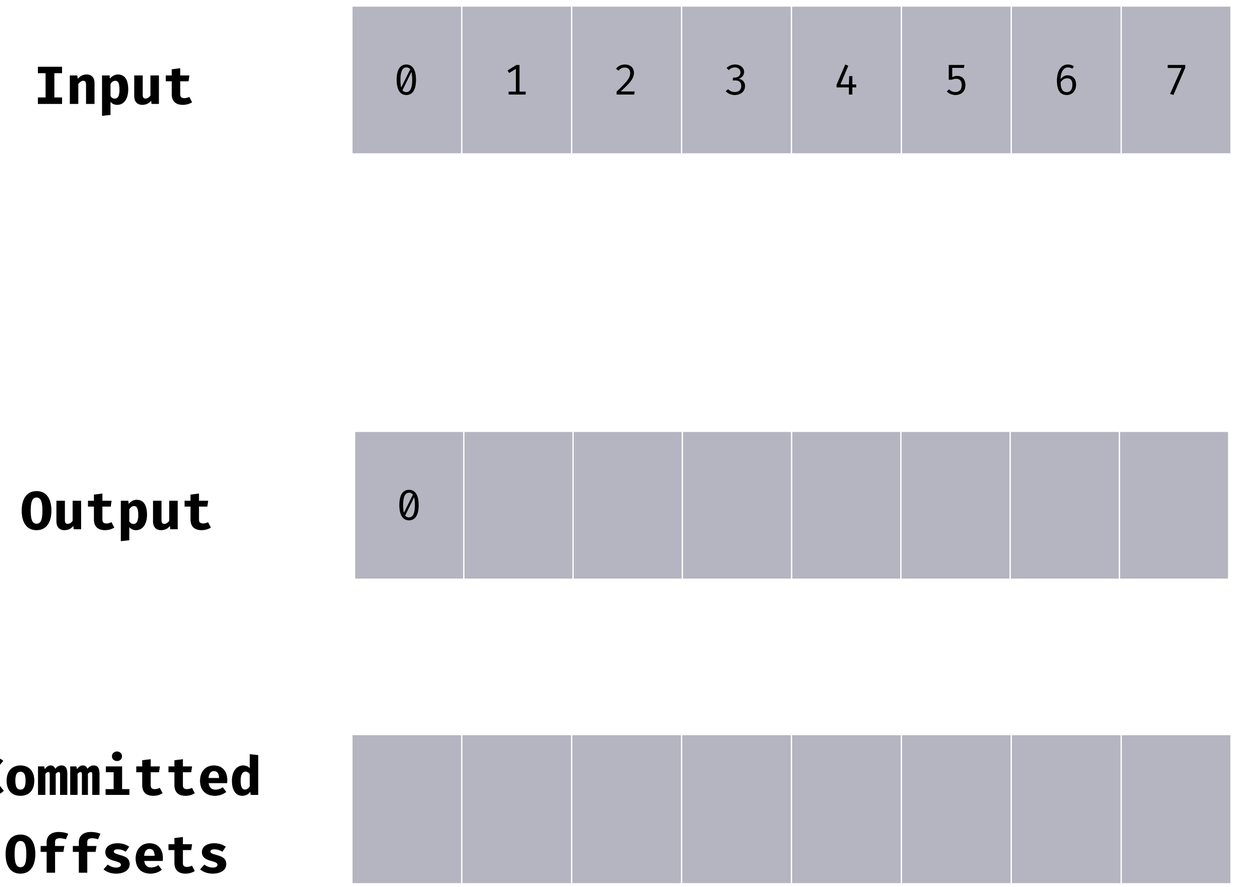
Семантика «Как минимум один раз»

```
while (true) {
  data = c.poll()
  p.send(data)
  c.commitOffsets()
}
```

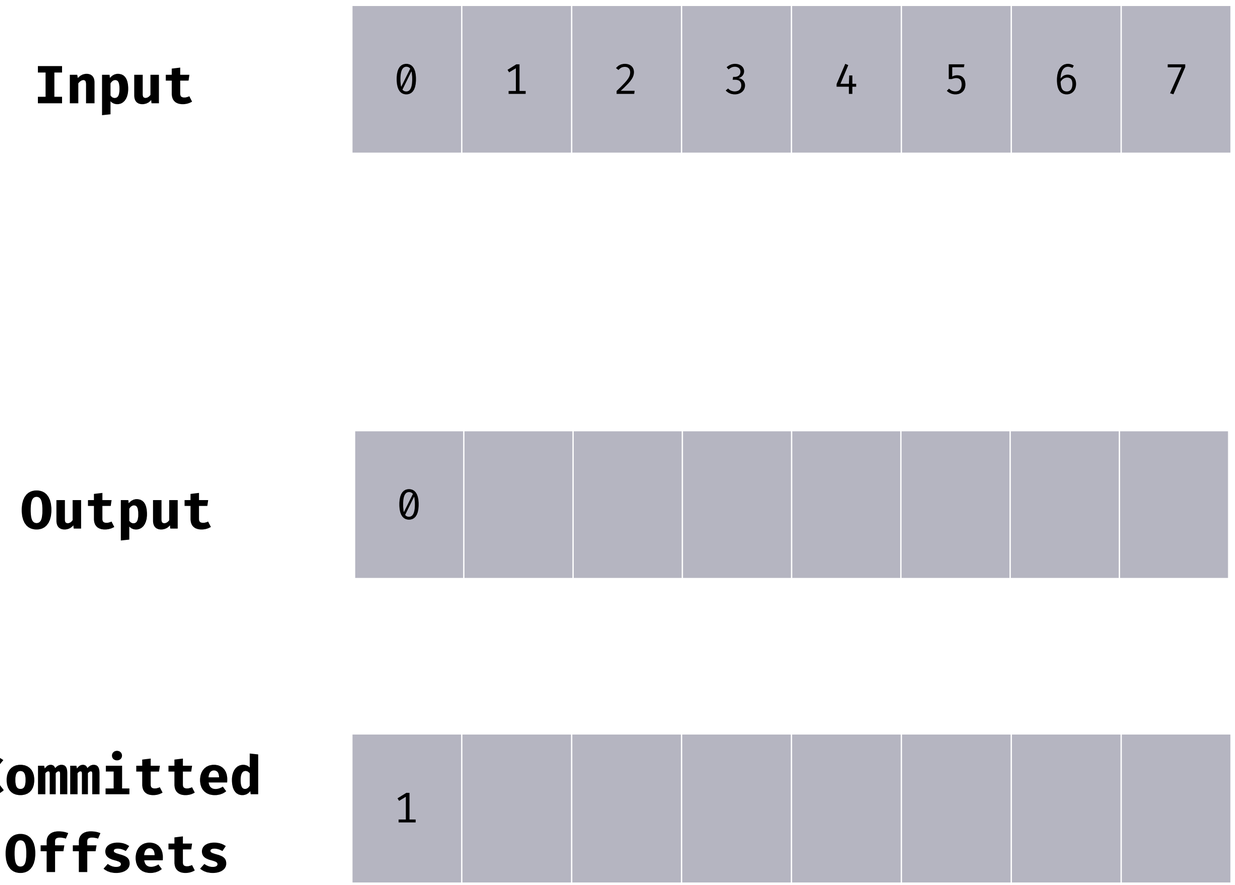
Например, когда просто надо скопировать ввод на вывод



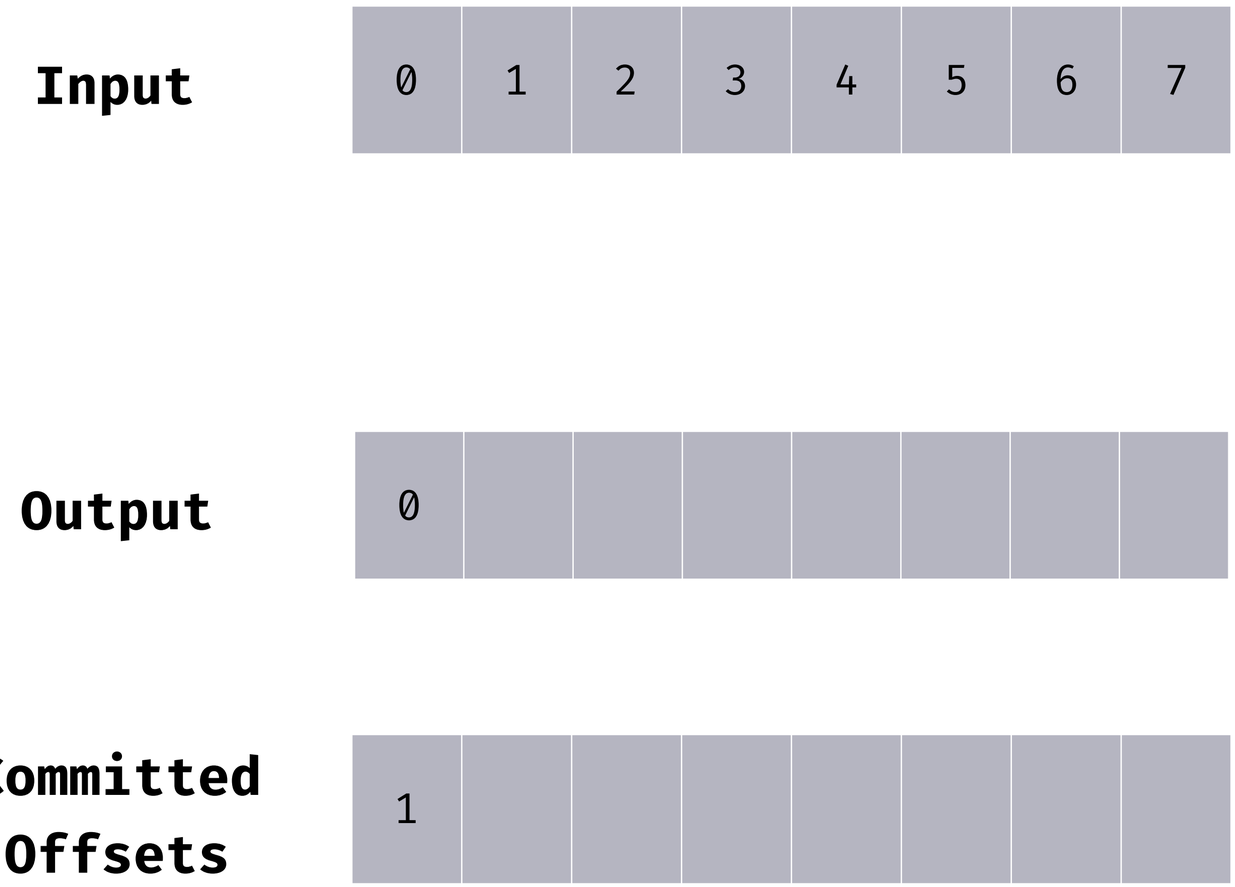

```
while (true) {
  data = c.poll()
  p.send(data)
  c.commitOffsets()
}
```



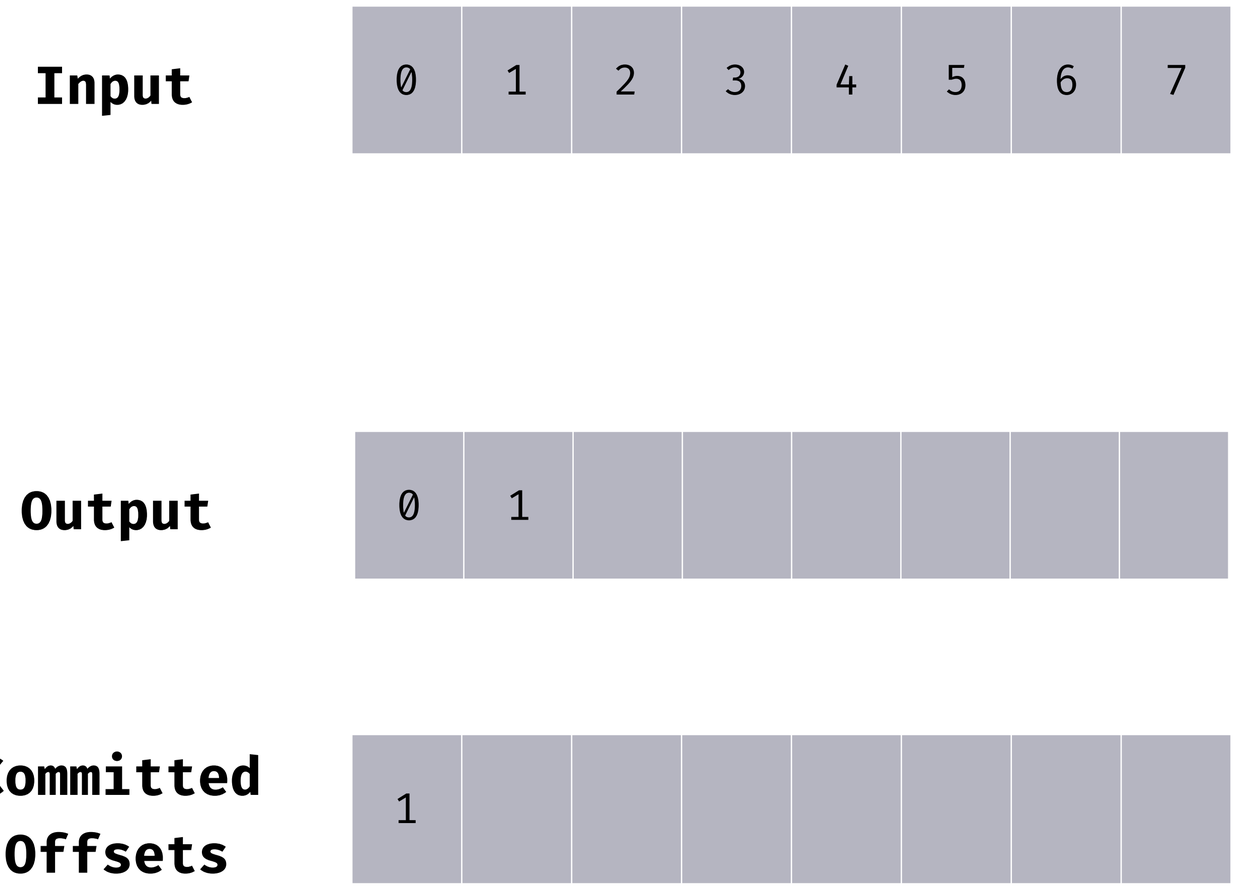

```
while (true) {
  data = c.poll()
  p.send(data)
  c.commitOffsets()
}
```



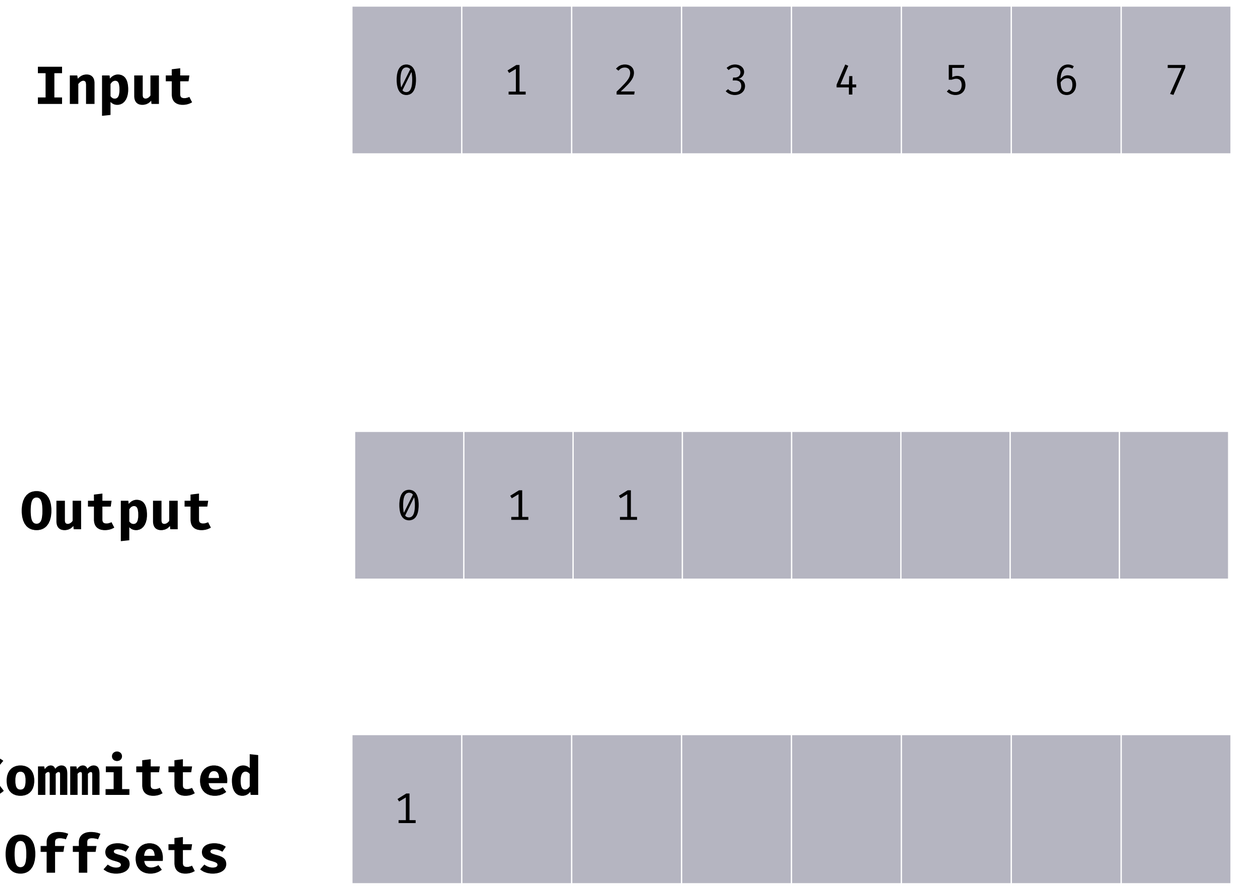

```
while (true) {
   data = c.poll()
   p.send(data)
   c.commitOffsets()
}
```




```
while (true) {
   data = c.poll()
   p.send(data)
   c.commitOffsets()
}
```

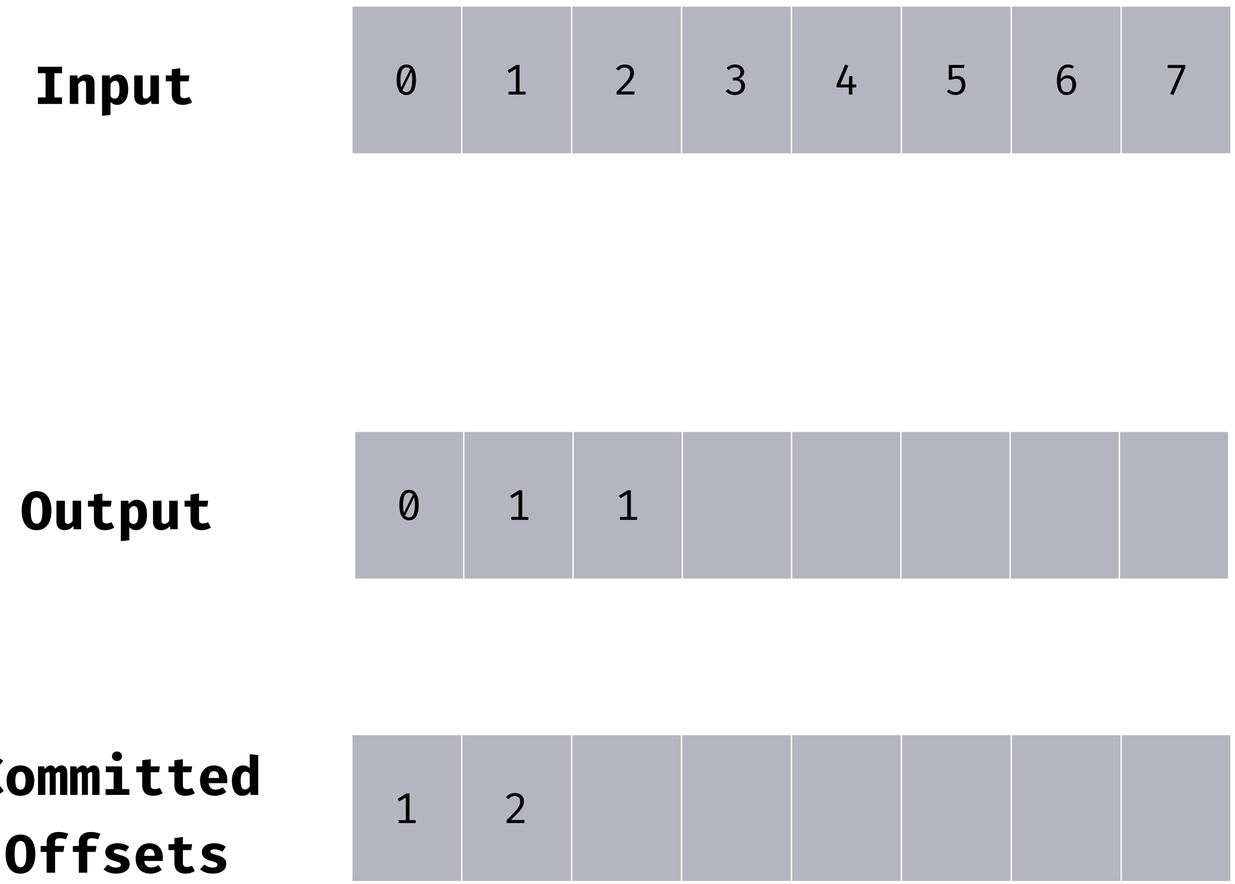



```
while (true) {
   data = c.poll()
   p.send(data)
   c.commitOffsets()
}
```

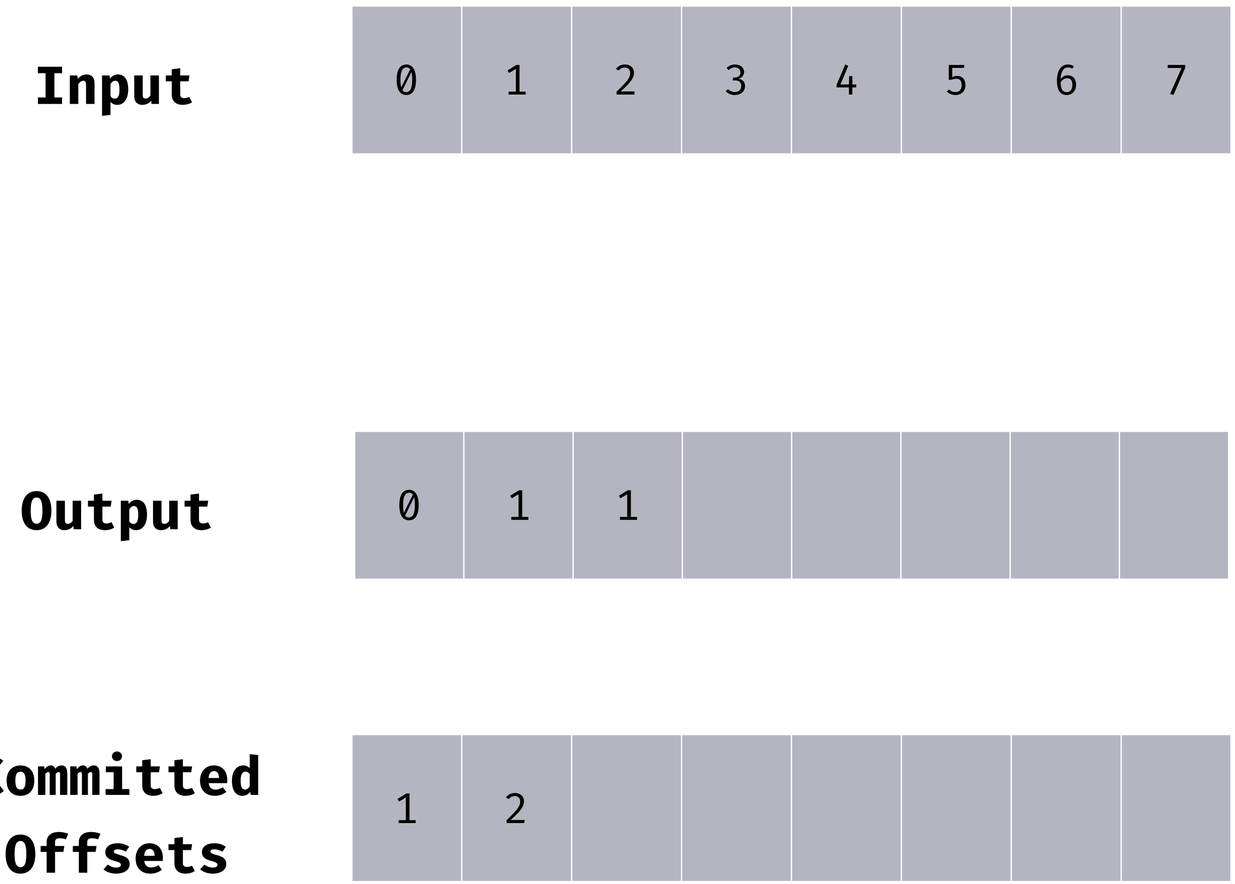



```
while (true) {
   data = c.poll()
   p.send(data)
   c.commitOffsets()
}
```

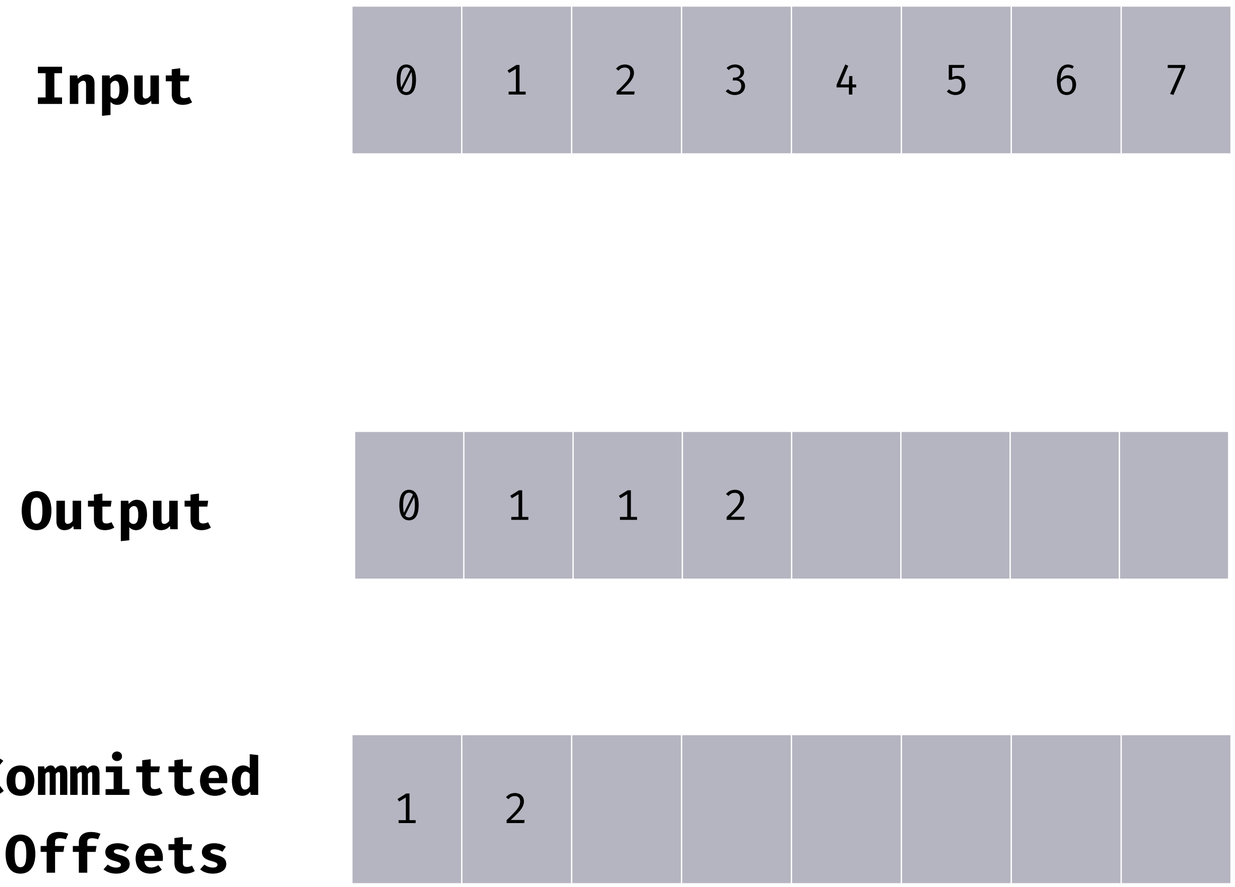
Ack lost, retry!



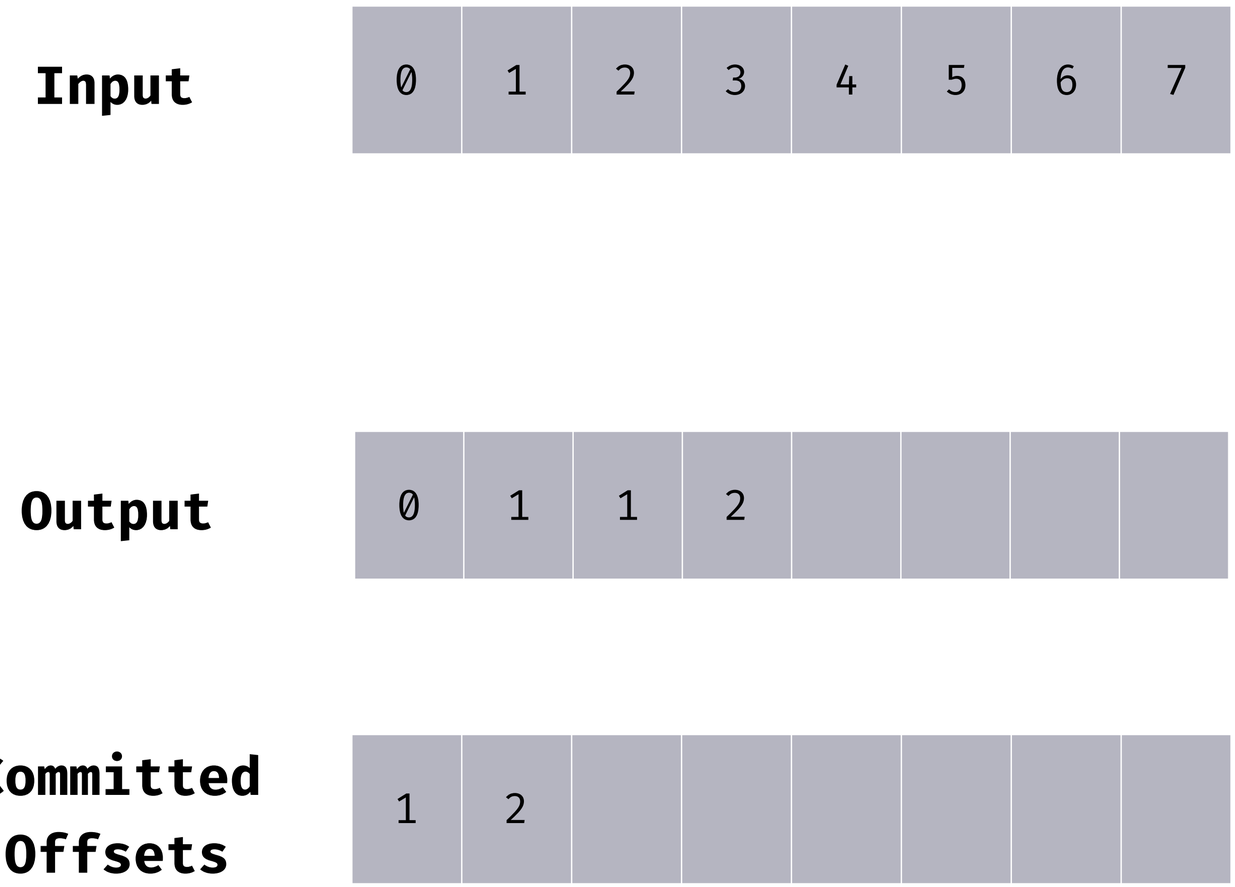

```
while (true) {
   data = c.poll()
   p.send(data)
   c.commitOffsets()
}
```




```
while (true) {
   data = c.poll()
   p.send(data)
   c.commitOffsets()
}
```



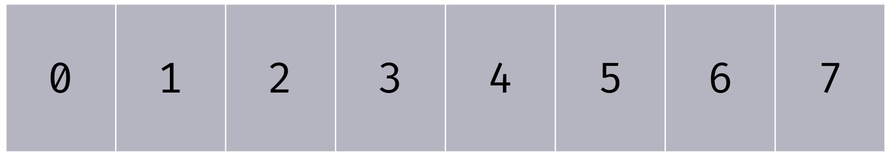

```
while (true) {
   data = c.poll()
   p.send(data)
   c.commitOffsets()
}
```



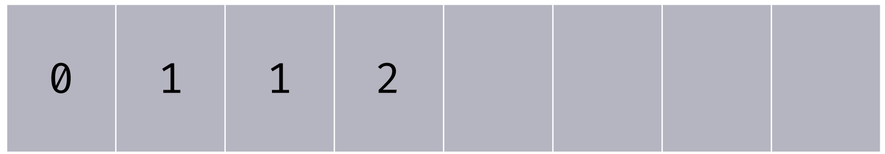

```
while (true) {
   data = c.poll()
   p.send(data)
   c.commitOffsets()
}
```

Process crash!

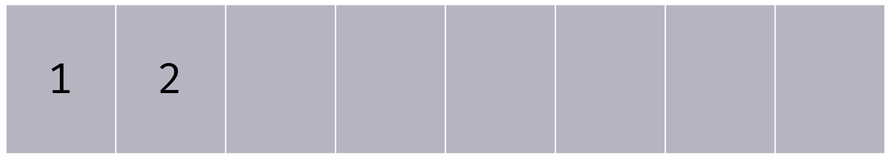
Input



Output

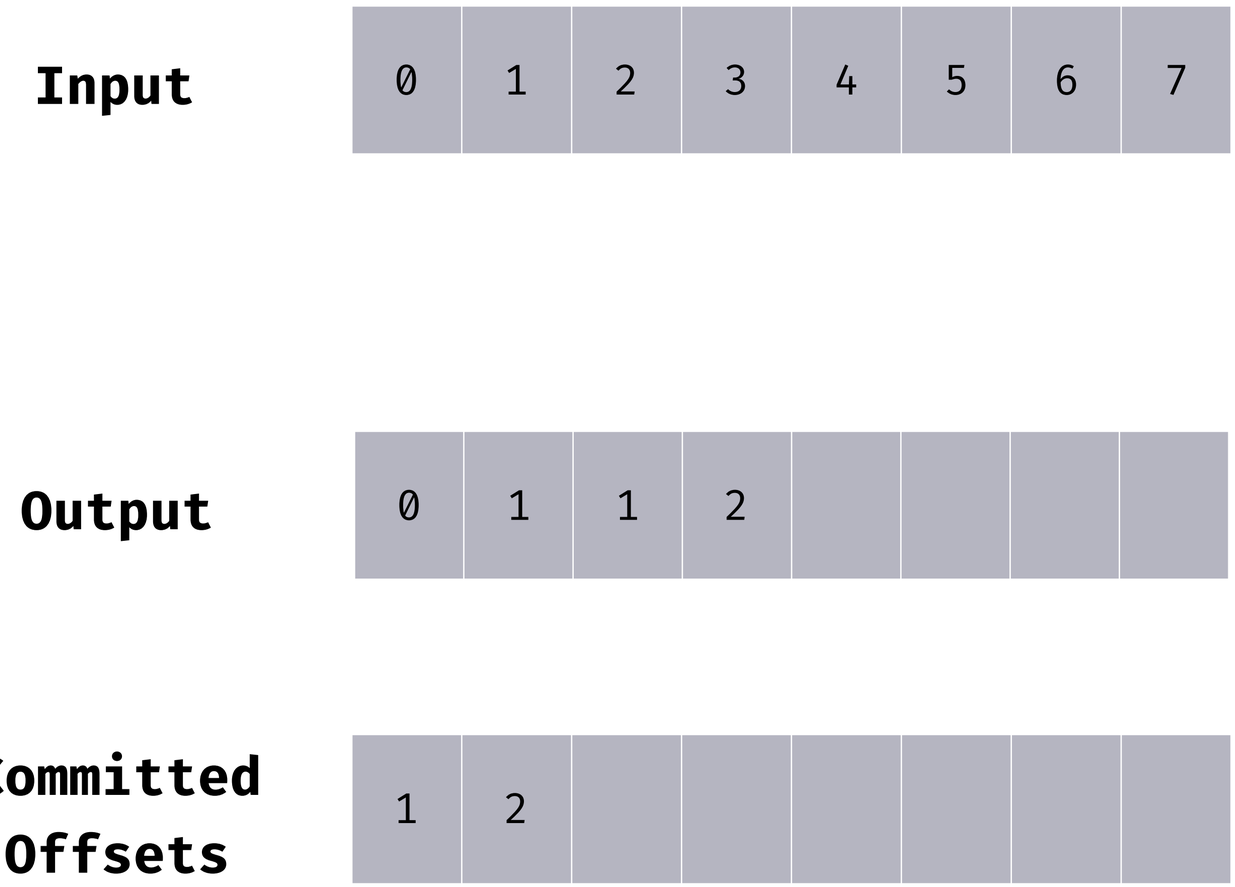


Committed Offsets

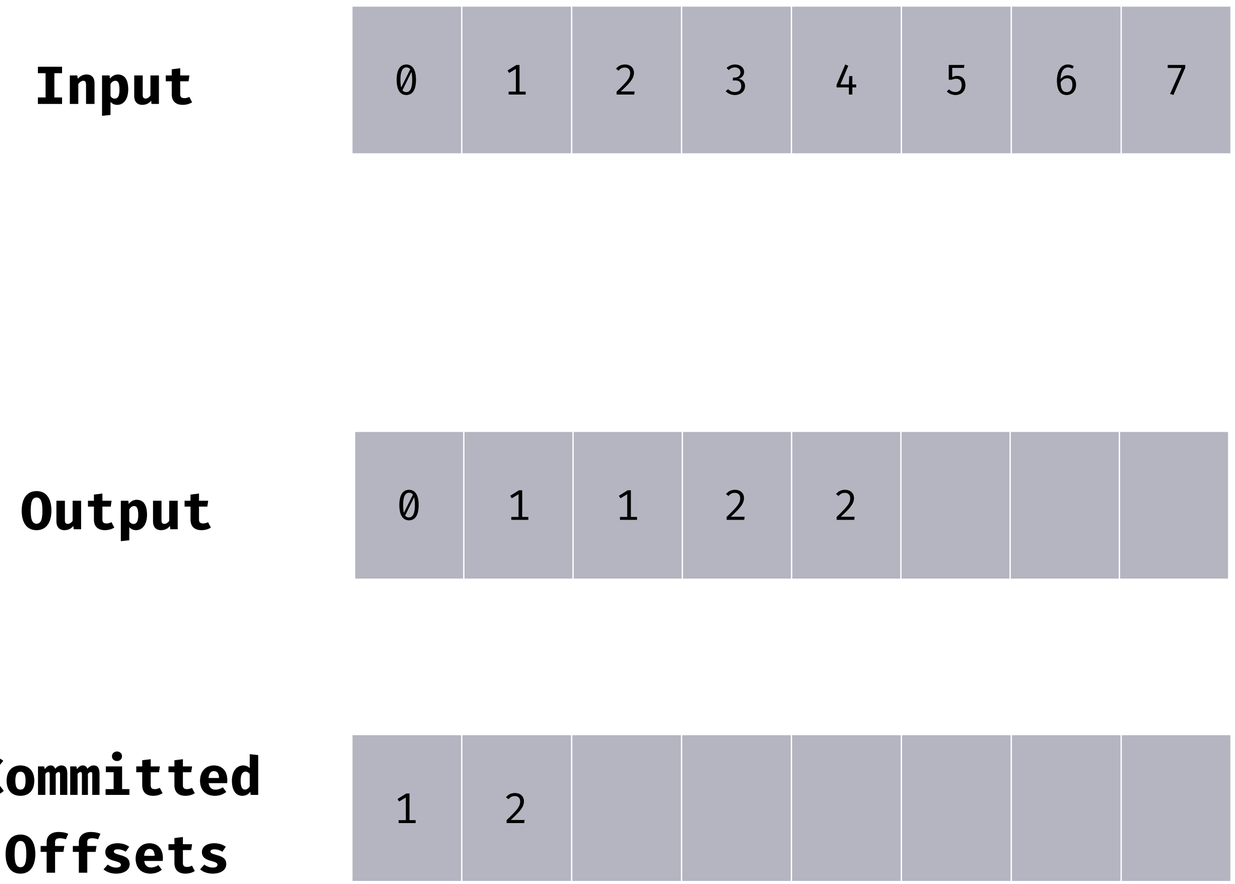



```
while (true) {
  data = c.poll()
  p.send(data)
  c.commitOffsets()
}
```

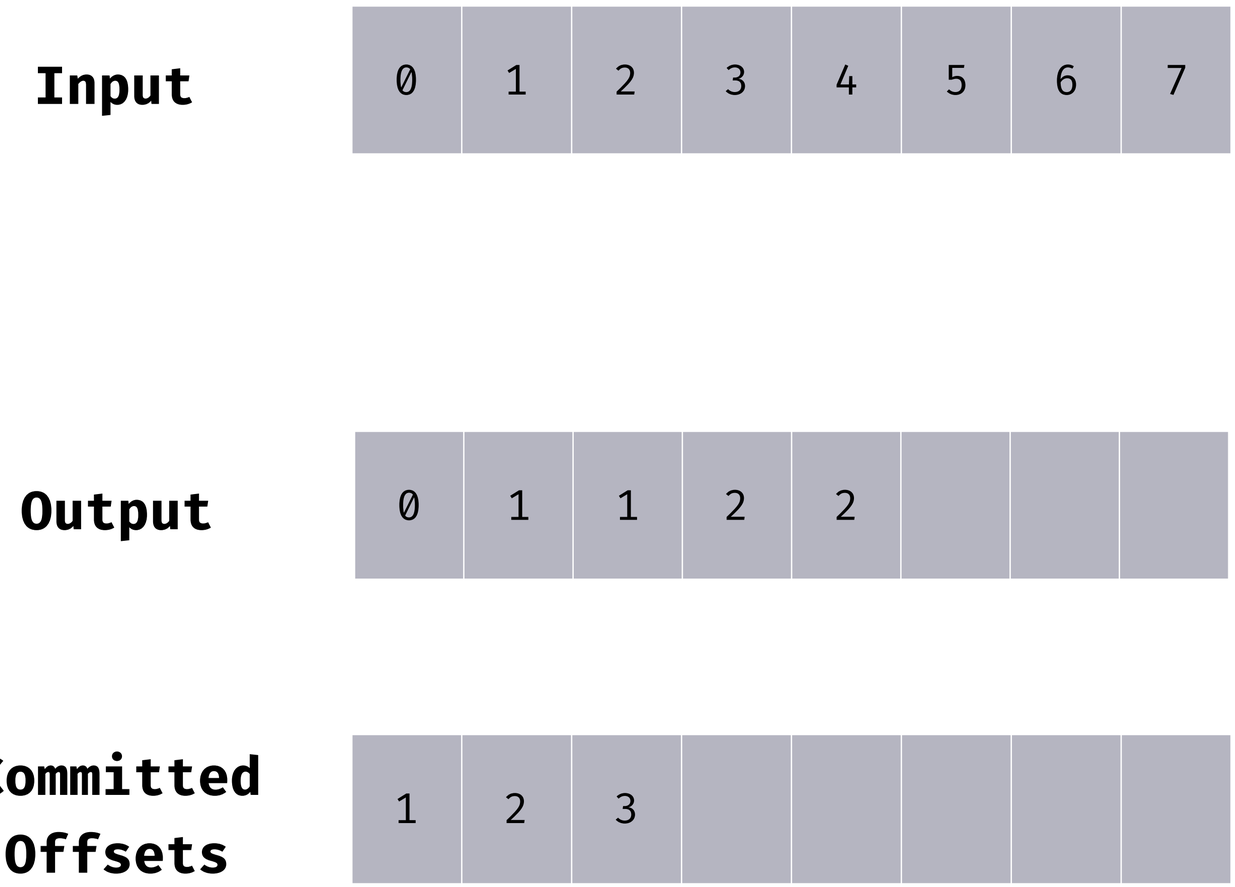
Restart!



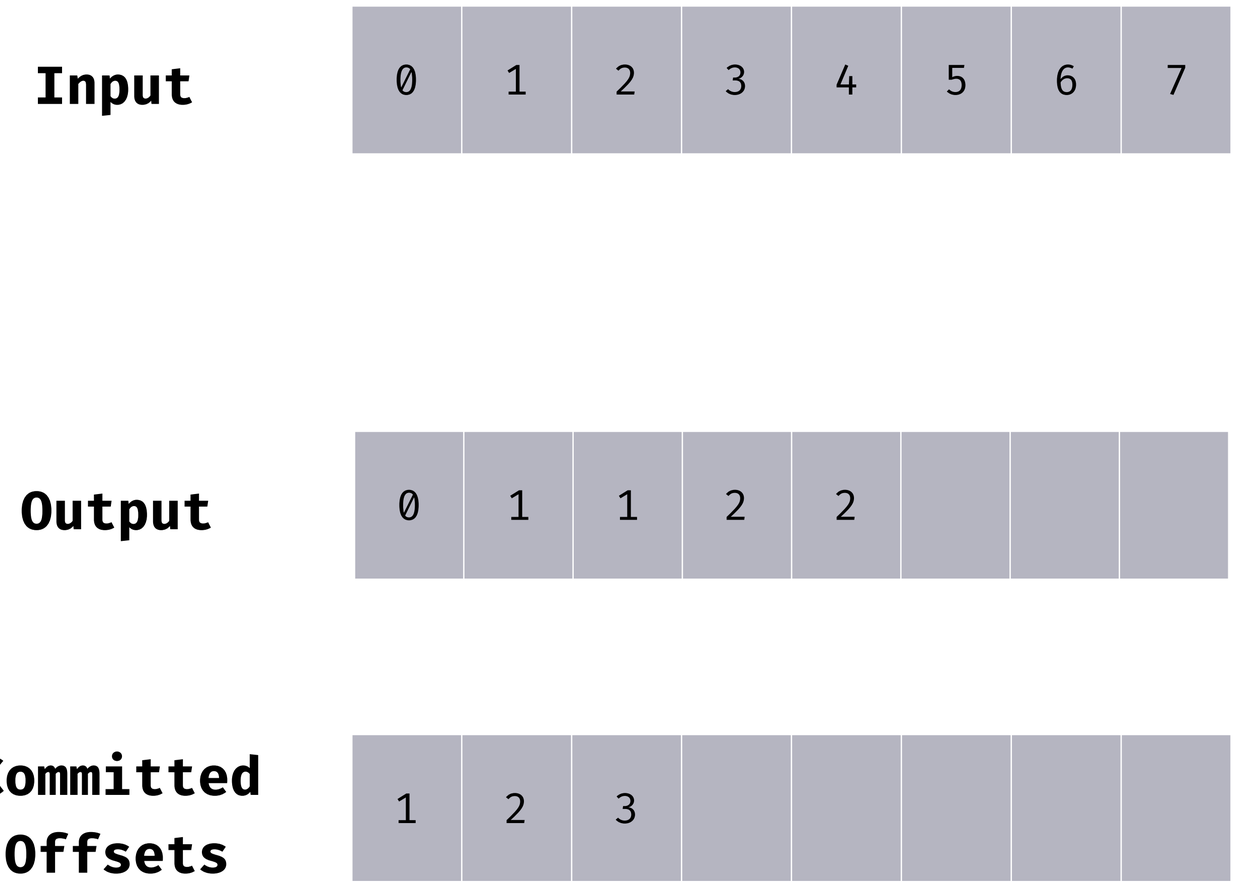

```
while (true) {
   data = c.poll()
   p.send(data)
   c.commitOffsets()
}
```




```
while (true) {
   data = c.poll()
   p.send(data)
   c.commitOffsets()
}
```




```
while (true) {
   data = c.poll()
   p.send(data)
   c.commitOffsets()
}
```




```
while (true) {
   data = c.poll()
   p.send(data)
   c.commitOffsets()
}
```

Input

0	1	2	3	4	5	6	7

Output

0	1	1	2	2			

Committed Offsets

1	2	3					

```
while  {
    dat          ll()
    p.s          )
    c.c          sets()
}
```

Input	0	1	2	3	4	5	6	7

Output	0	1	1	2	2			

Committed Offsets	1	2	3					

```
whil                {
    da          oll()
    p.          a)
    c.          ffsets()
}
```

```
while (true) {
  data = c.poll()
  p.send(data)
  c.commitOffsets()
}
```

Input

0	1	2	3	4	5	6	7

Output

0	1	1	2	2			

Committed Offsets

1	2	3					

```
while (true) {
   dat                ll()
   p.s                a)
   c.c                fsets()
}
```

```
while (true) {
  data = c.poll()
  p.send(data)
  c.commitOffsets()
}
```

Input

0	1	2	3	4	5	6	7

Output

0	1	1	2	2	3		

Committed Offsets

1	2	3					

```
while (true) {
   data = c.poll()
   p.send(data)
   c.commitOffsets()
}
```

Input

0	1	2	3	4	5	6	7

Output

0	1	1	2	2	3		

Committed Offsets

1	2	3	4				

```
while (t
   data =          ()
   p.send
   c.comm                ts()
}
```

```
while (true) {
   data = c.poll()
   p.send(data)
   c.commitOffsets()
}
```

Input

0	1	2	3	4	5	6	7

Output

0	1	1	2	2	3		

Committed Offsets

1	2	3	4				

```
while (true) {
  dat          oll()
  p.s          )
  c.c          fsets()
}
```

```
while (true) {
  data = c.poll()
  p.send(data)
  c.commitOffsets()
}
```

Input

0	1	2	3	4	5	6	7

Output

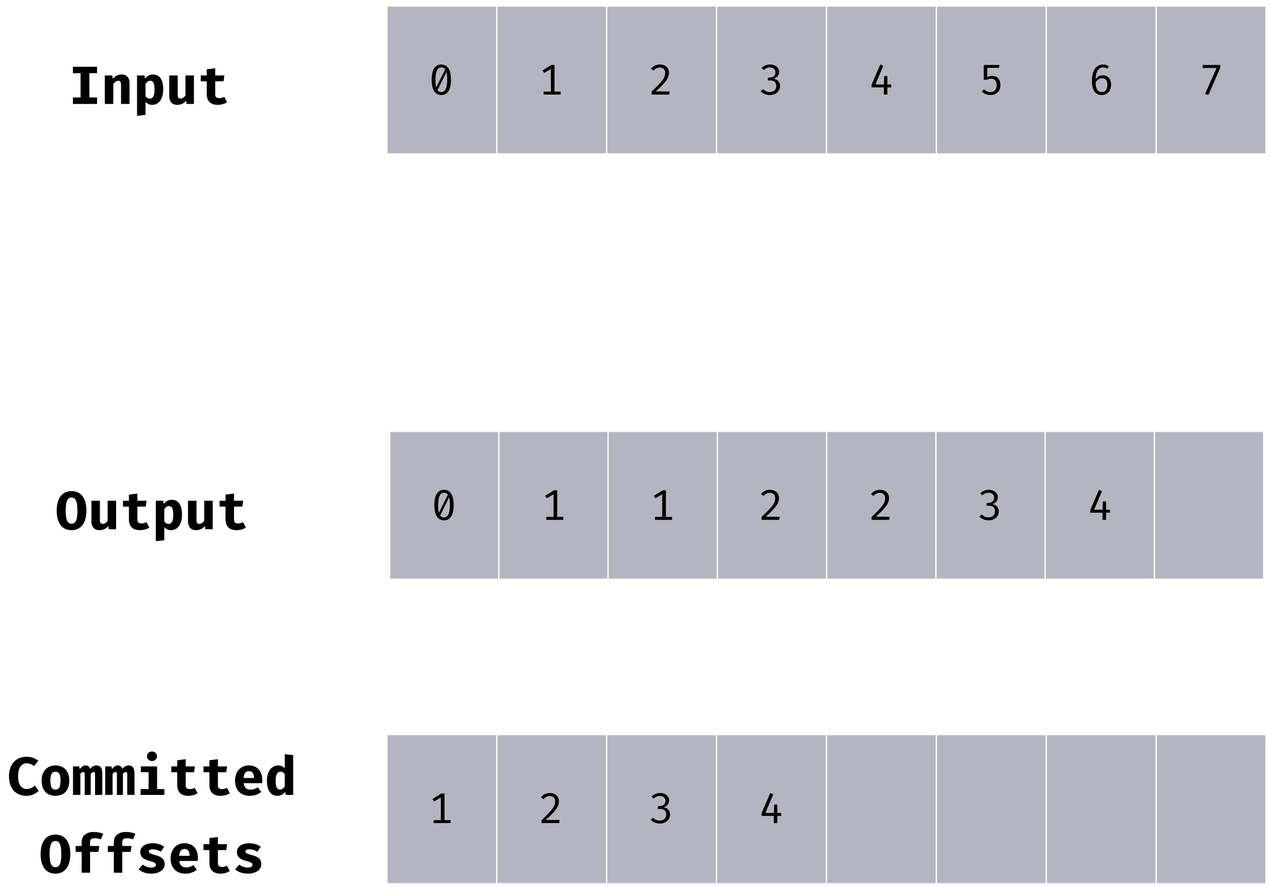
0	1	1	2	2	3	4	

Committed Offsets

1	2	3	4				

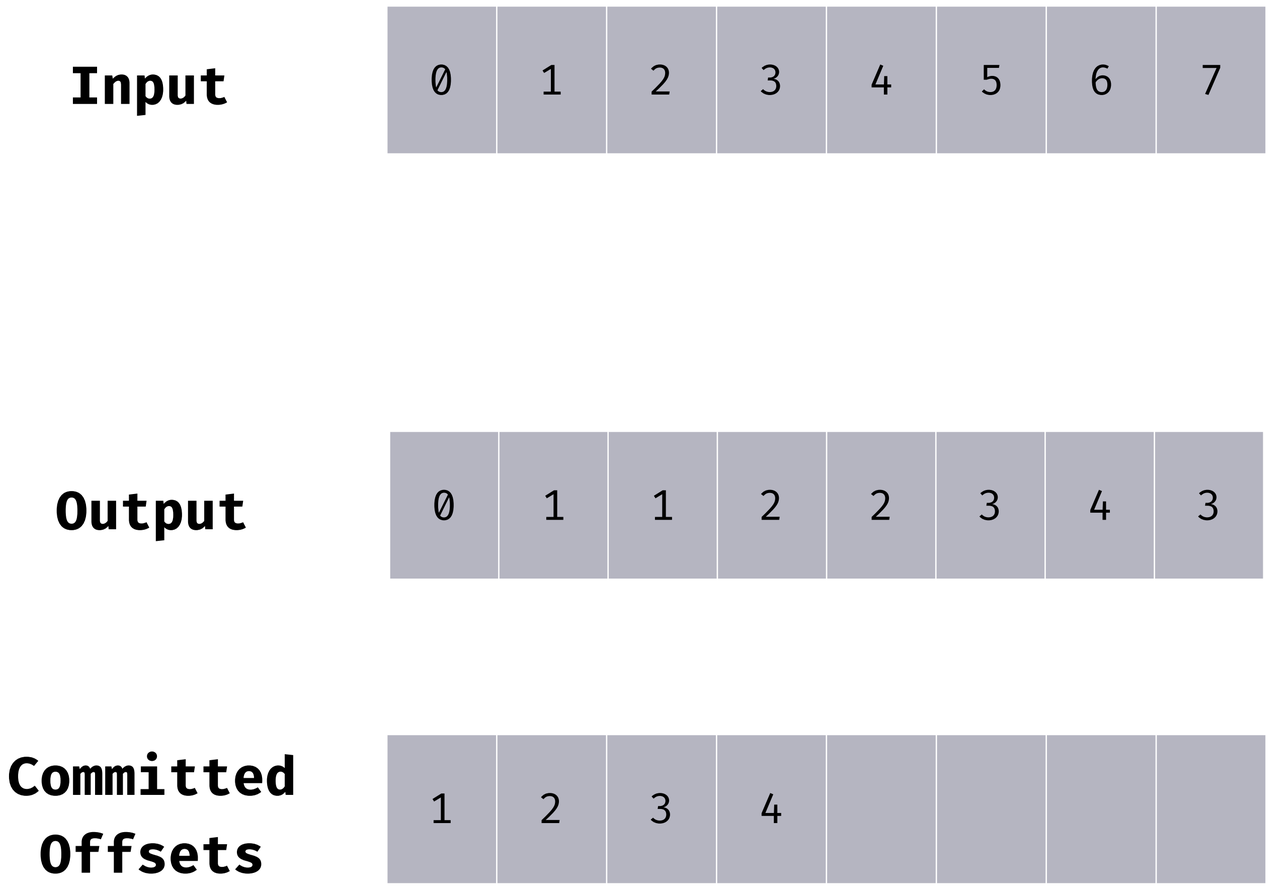
```
while (true) {
   da          poll()
   p.          ta)
   c.          ffsets()
}
```

```
while (true) {
   data = c.poll()
   p.send(data)
   c.commitOffsets()
}
```




```
while (true) {
  data = c.poll()
  p.send(data)
  c.commitOffsets()
}
```

```
while (true) {
  data = c.poll()
  p.send(data)
  c.commitOffsets()
}
```




```
while (true) {
  data = c.poll()
  p.send(data)
  c.commitOffsets()
}
```

```
while (true) {
  data = c.poll()
  p.send(data)
  c.commitOffsets()
}
```

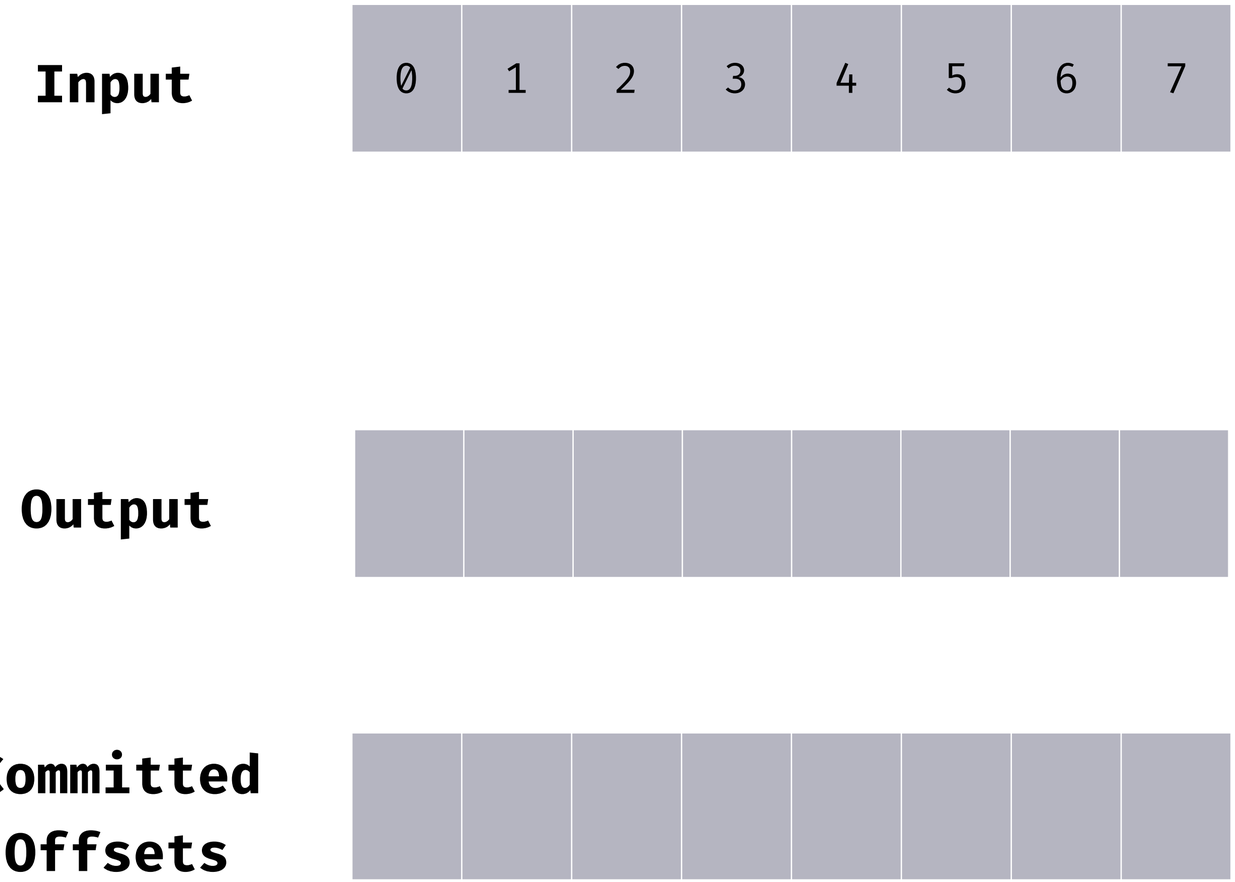
Семантика «Как минимум один раз»

- Дубликаты
  - Повторы продьюсера
  - Сбой перед записью смещений (offset)
  - Зомби

Input	0	1	2	3	4	5	6	7
Output	0	1	1	2	2	3	4	3
Committed Offsets	1	2	3	4				

Семантика «максимум один раз»

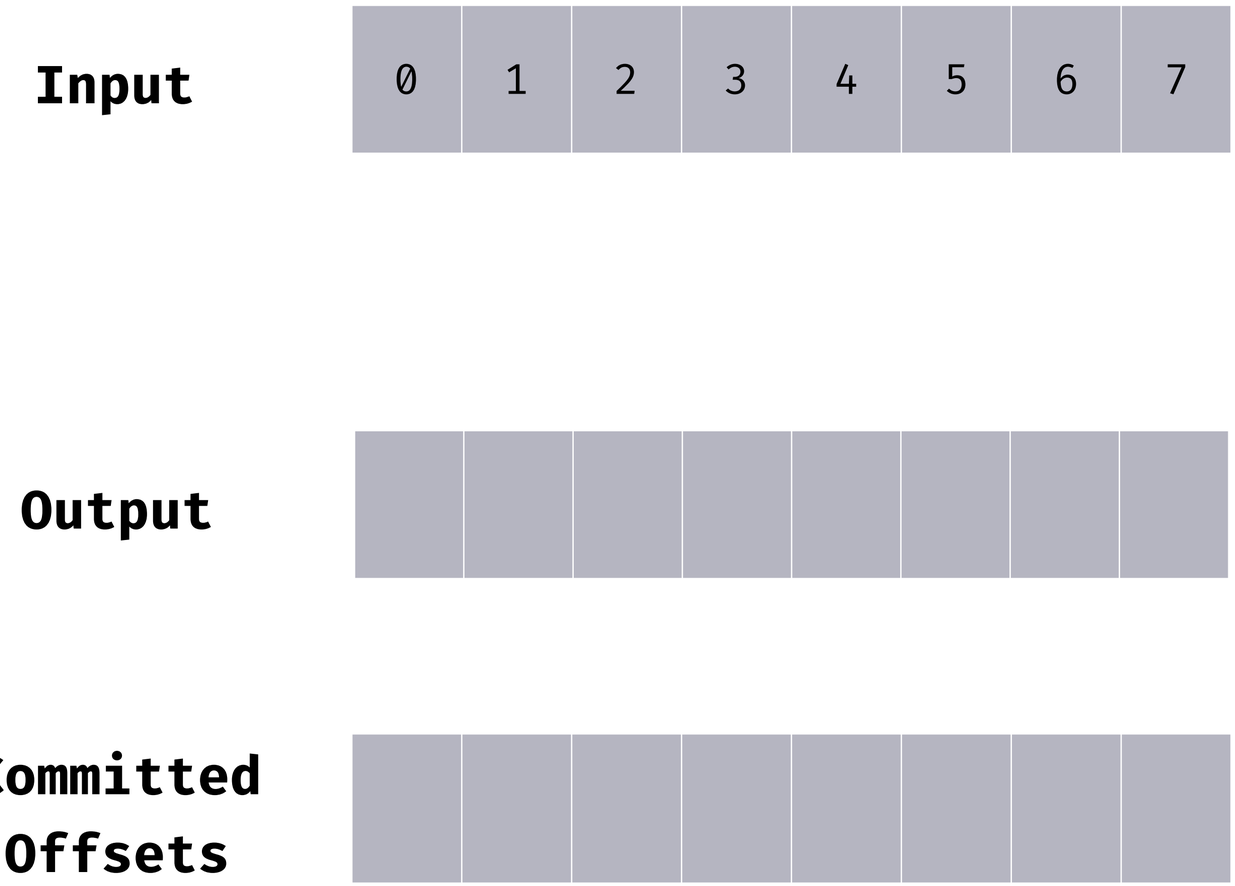
Вам это не надо




```
while (true) {
   data = c.poll()
   c.commitOffsets()
   p.send(data)
}
```

Семантика «максимум один раз»

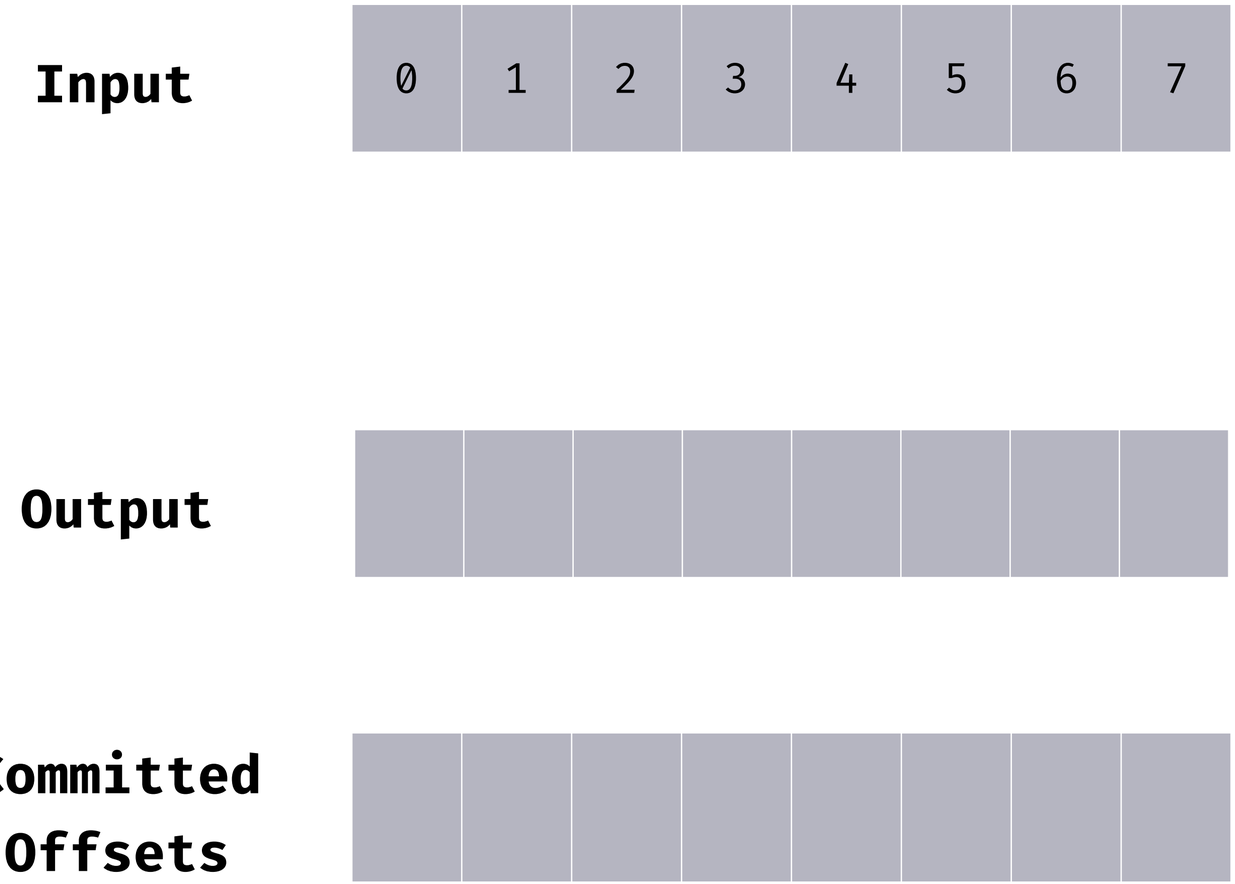
Вам это не надо




```
while (true) {
  data = c.poll()
  c.commitOffsets()
  p.send(data)
}
```

Семантика «максимум один раз»

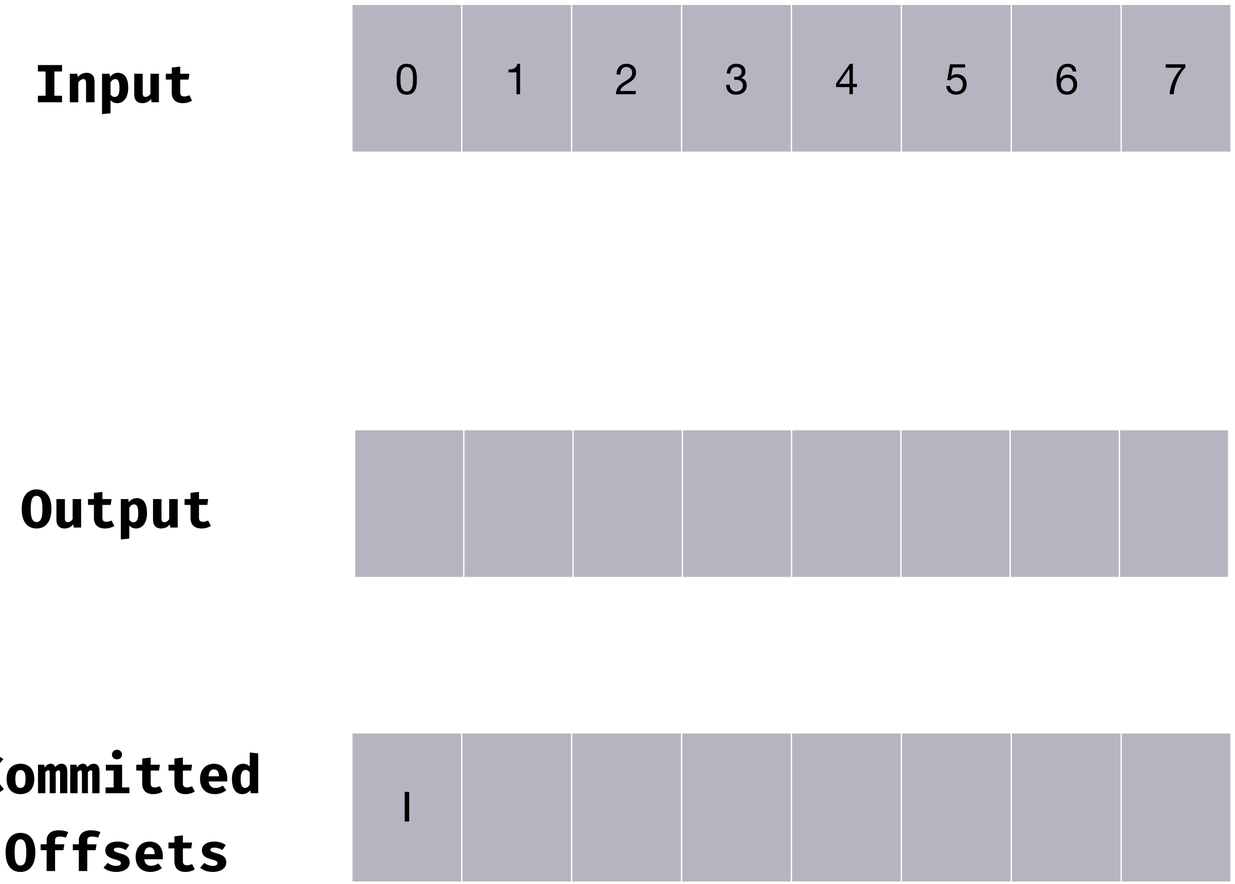
Вам это не надо




```
while (true) {
   data = c.poll()
   c.commitOffsets()
   p.send(data)
}
```

Семантика «максимум один раз»

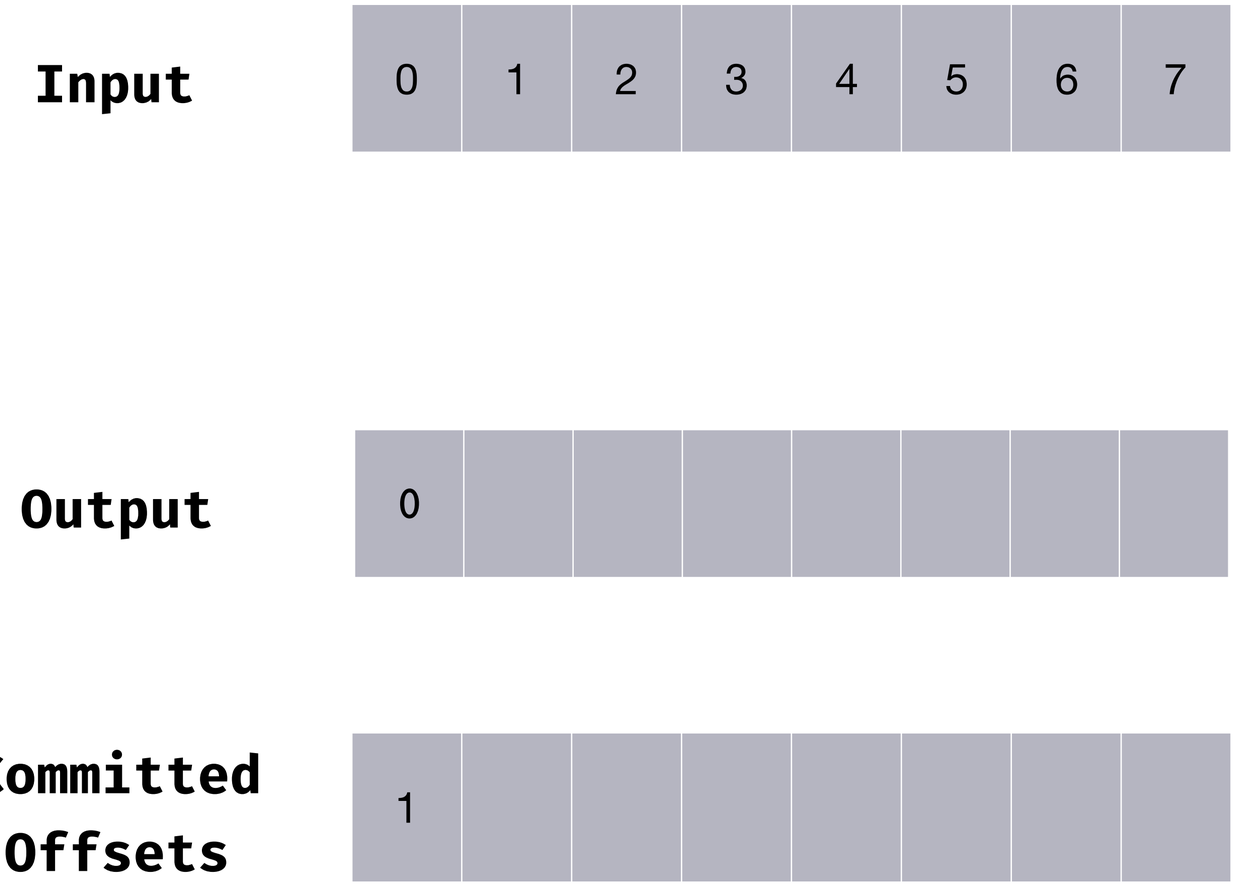
Вам это не надо




```
while (true) {
   data = c.poll()
   c.commitOffsets()
   p.send(data)
}
```

Семантика «максимум один раз»

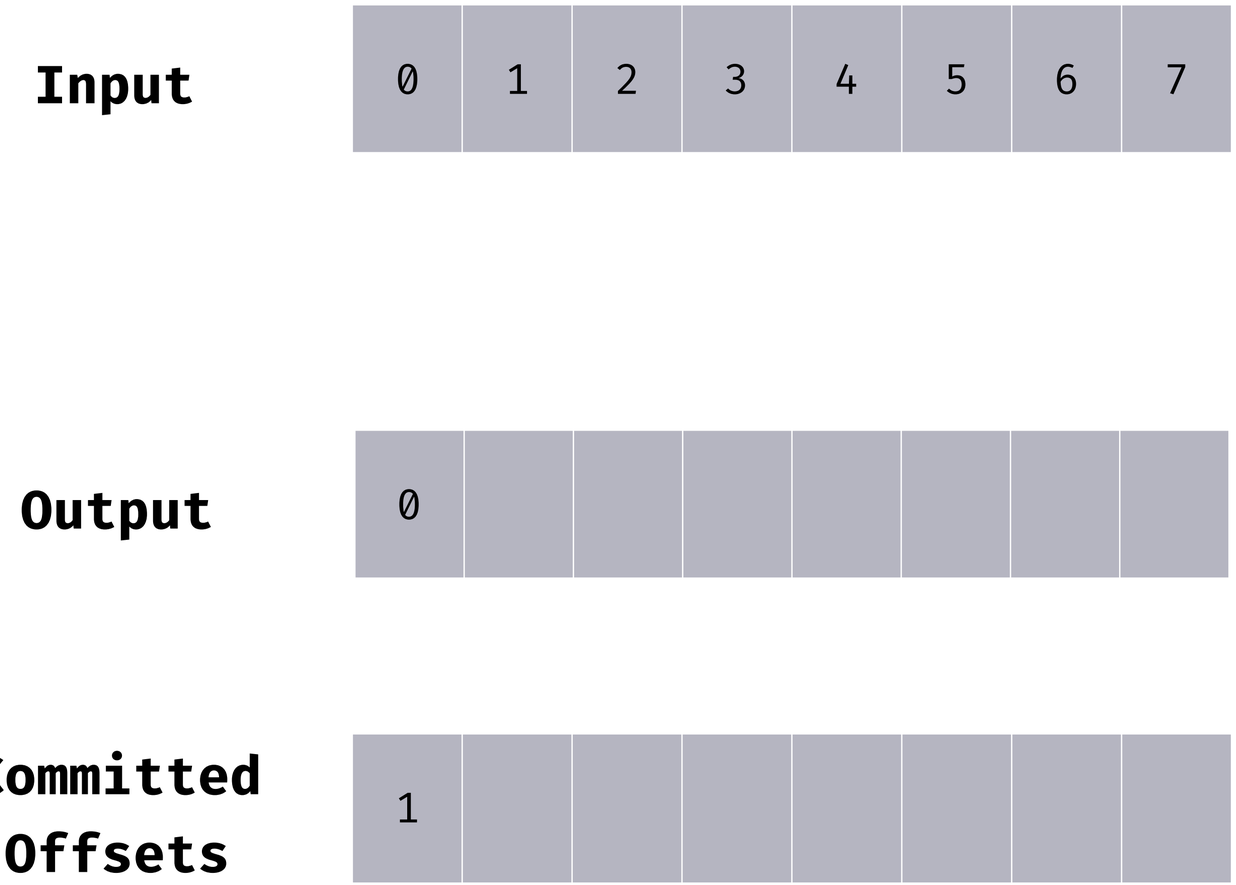
Вам это не надо




```
while (true) {
   data = c.poll()
   c.commitOffsets()
   p.send(data)
}
```

Семантика «максимум один раз»

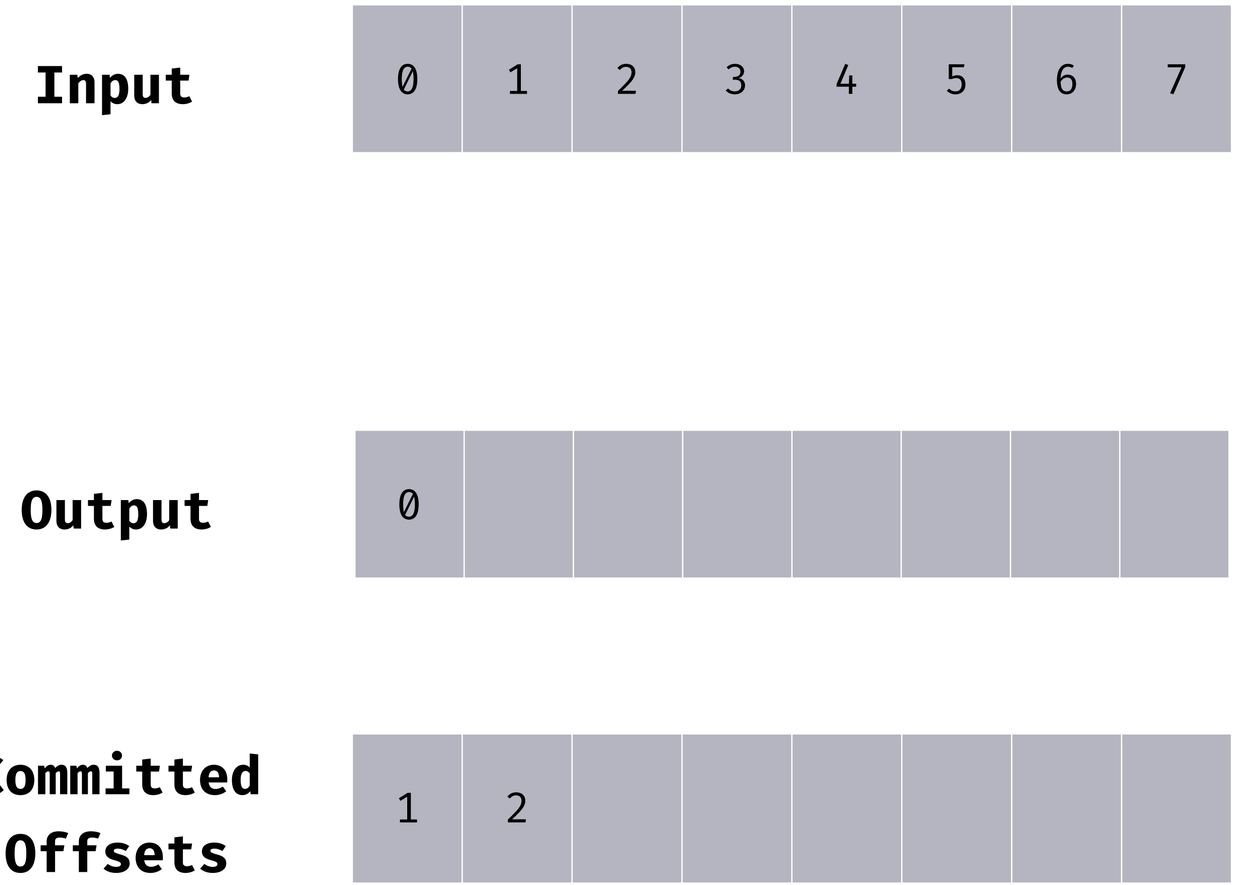
Вам это не надо




```
while (true) {
   data = c.poll()
   c.commitOffsets()
   p.send(data)
}
```

Семантика «максимум один раз»

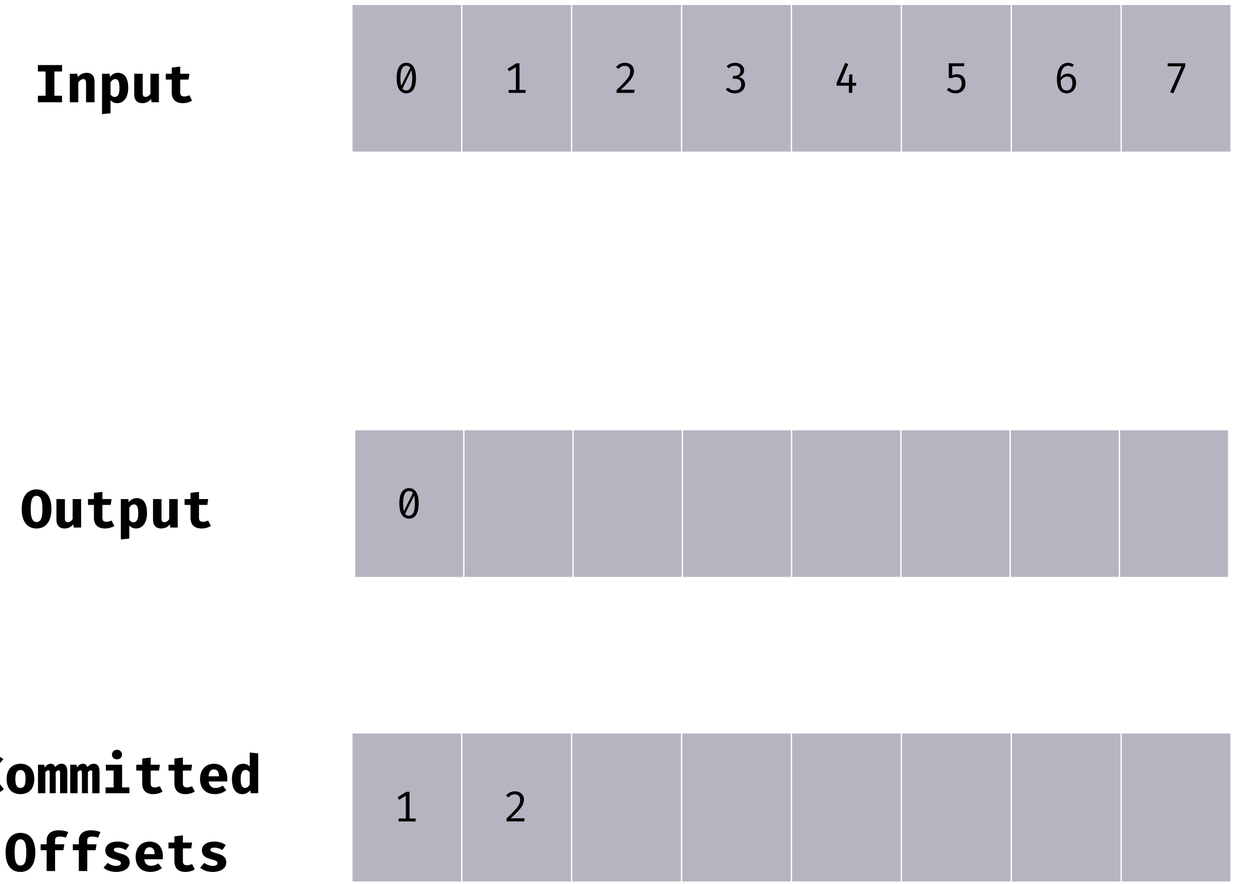
Вам это не надо




```
while (true) {
   data = c.poll()
   c.commitOffsets()
   p.send(data)
}
```

Семантика «максимум один раз»

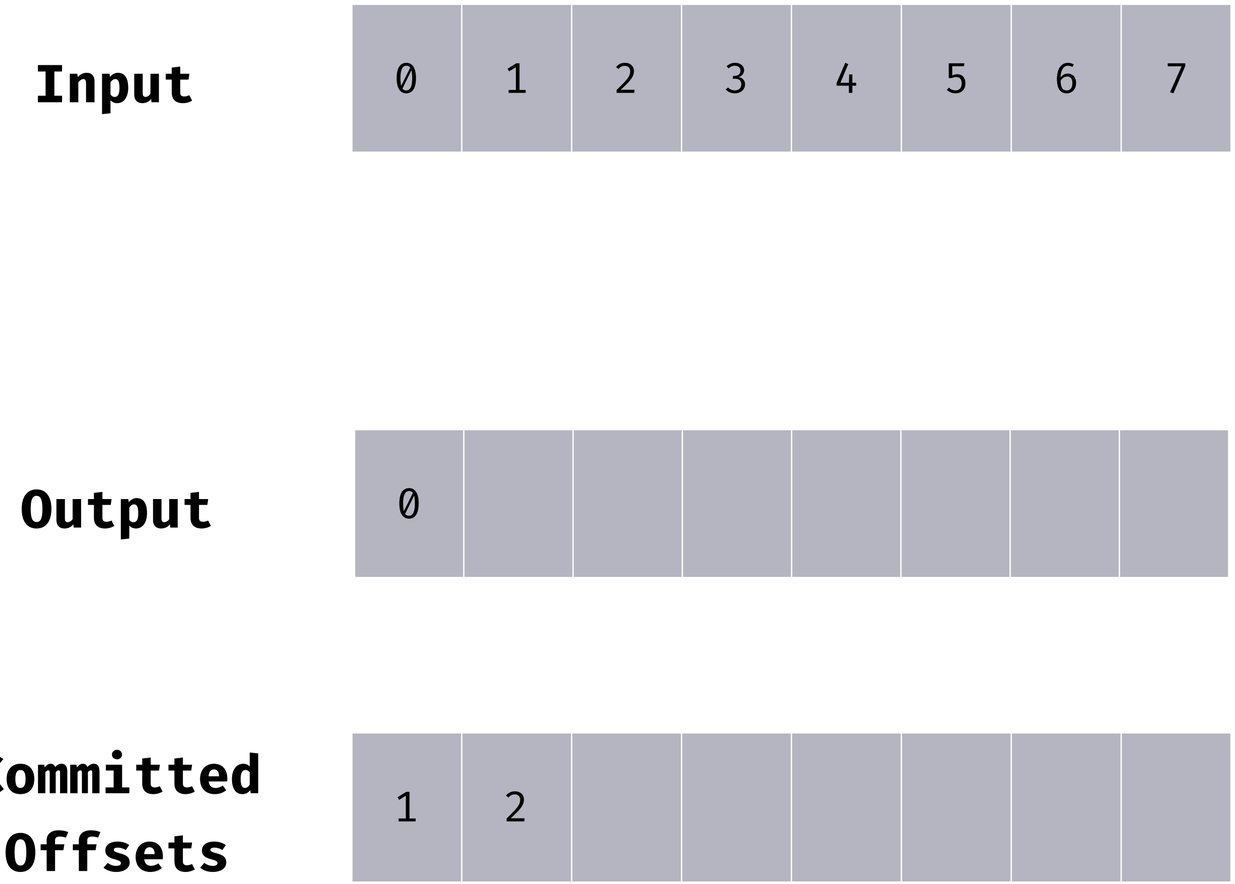
Вам это не надо




```
while (true) {
   data = c.poll()
   c.commitOffsets()
   p.send(data)
}
```

Семантика «максимум один раз»

Вам это не надо

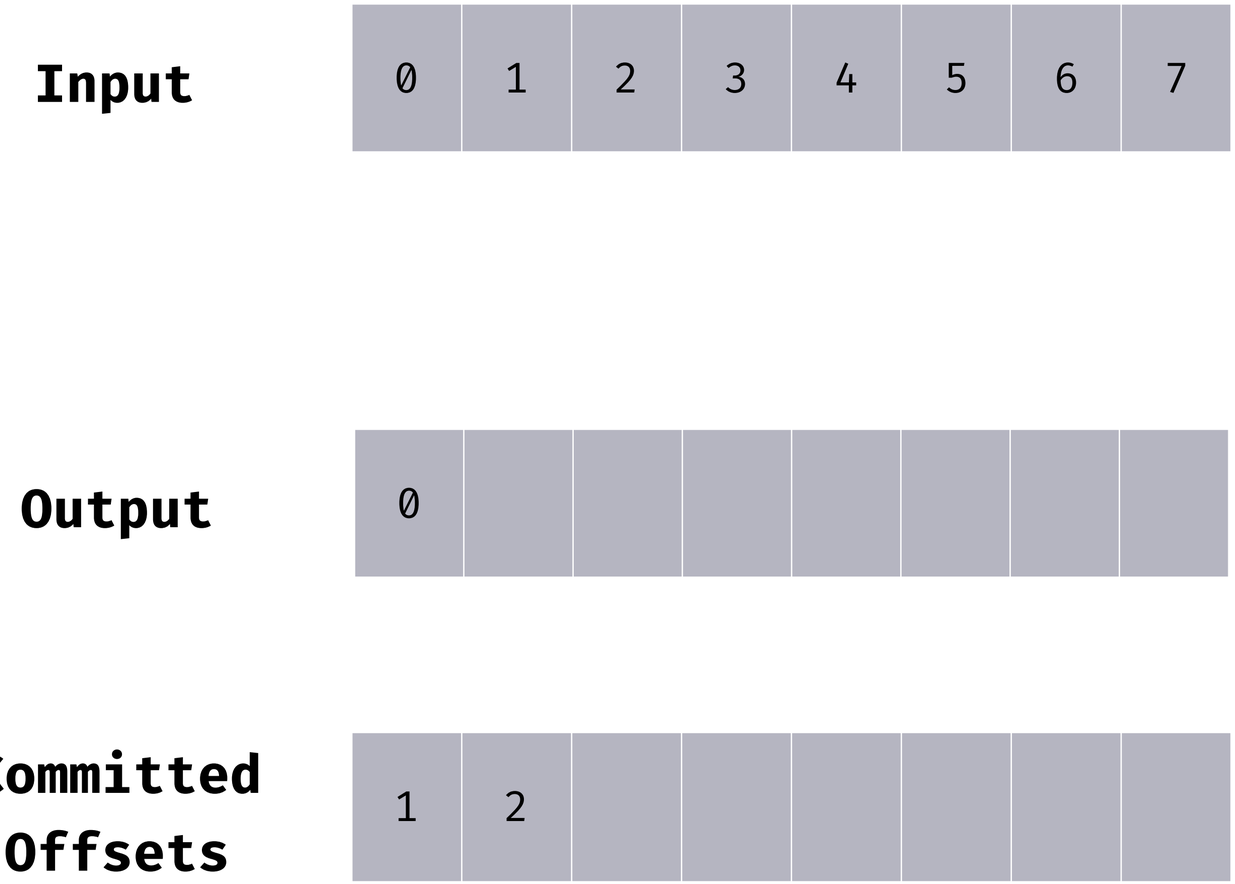



```
while (true) {
   data = c.poll()
   c.commitOffsets()
   p.send(data)
}
```

Ack lost, yolo!

Семантика «максимум один раз»

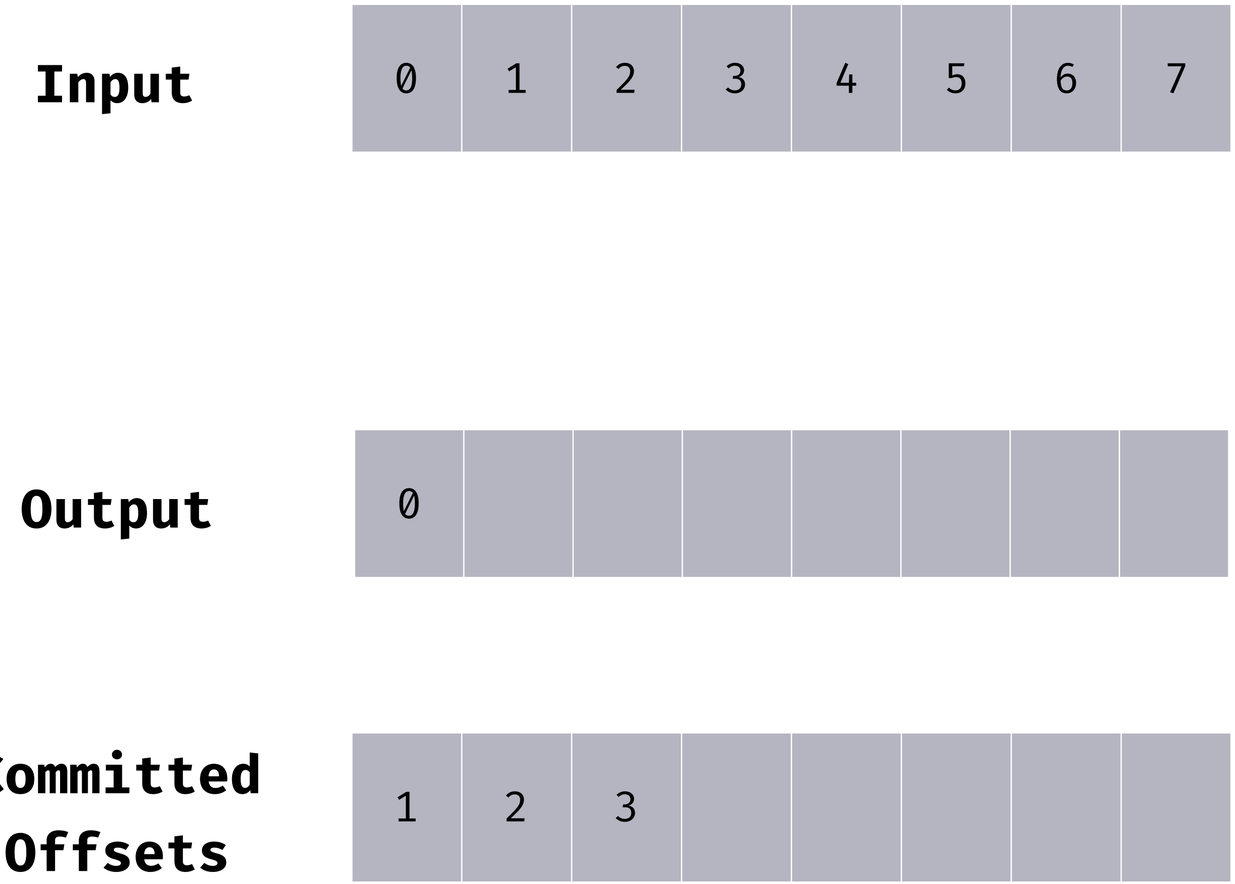
Вам это не надо




```
while (true) {
   data = c.poll()
   c.commitOffsets()
   p.send(data)
}
```

Семантика «максимум один раз»

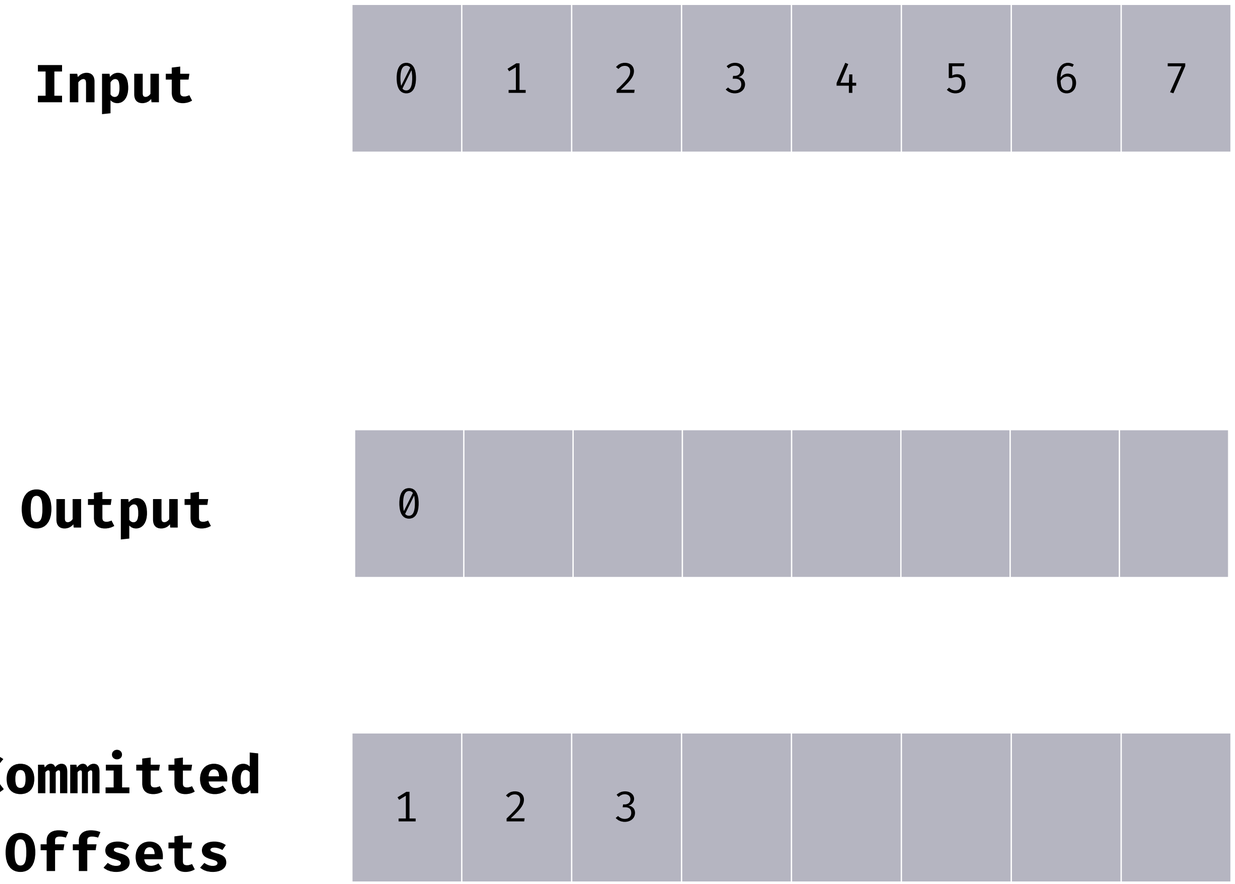
Вам это не надо




```
while (true) {
   data = c.poll()
   c.commitOffsets()
   p.send(data)
}
```

Семантика «максимум один раз»

Вам это не надо

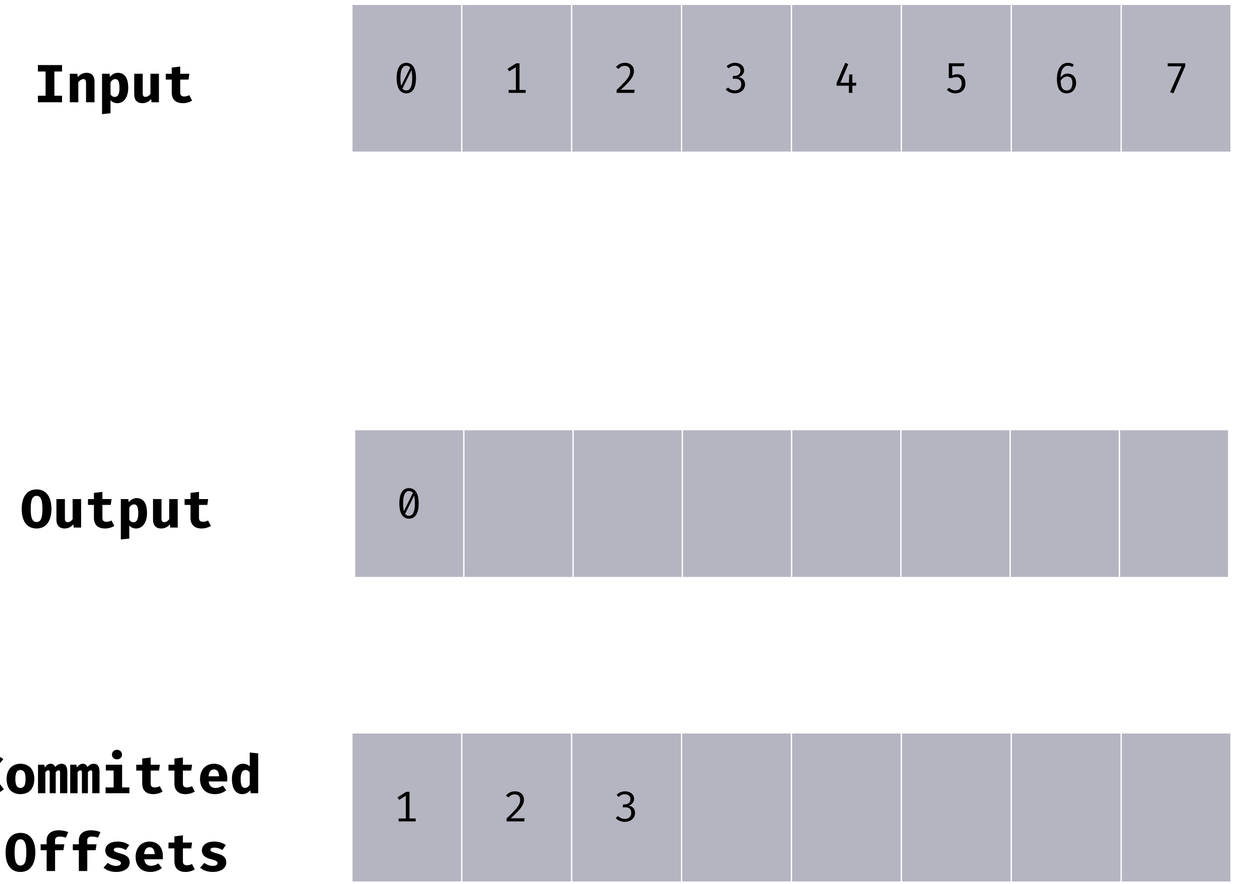



```
while (true) {
   data = c.poll()
   c.commitOffsets()
   p.send(data)
}
```

Process crash!

Семантика «максимум один раз»

Вам это не надо

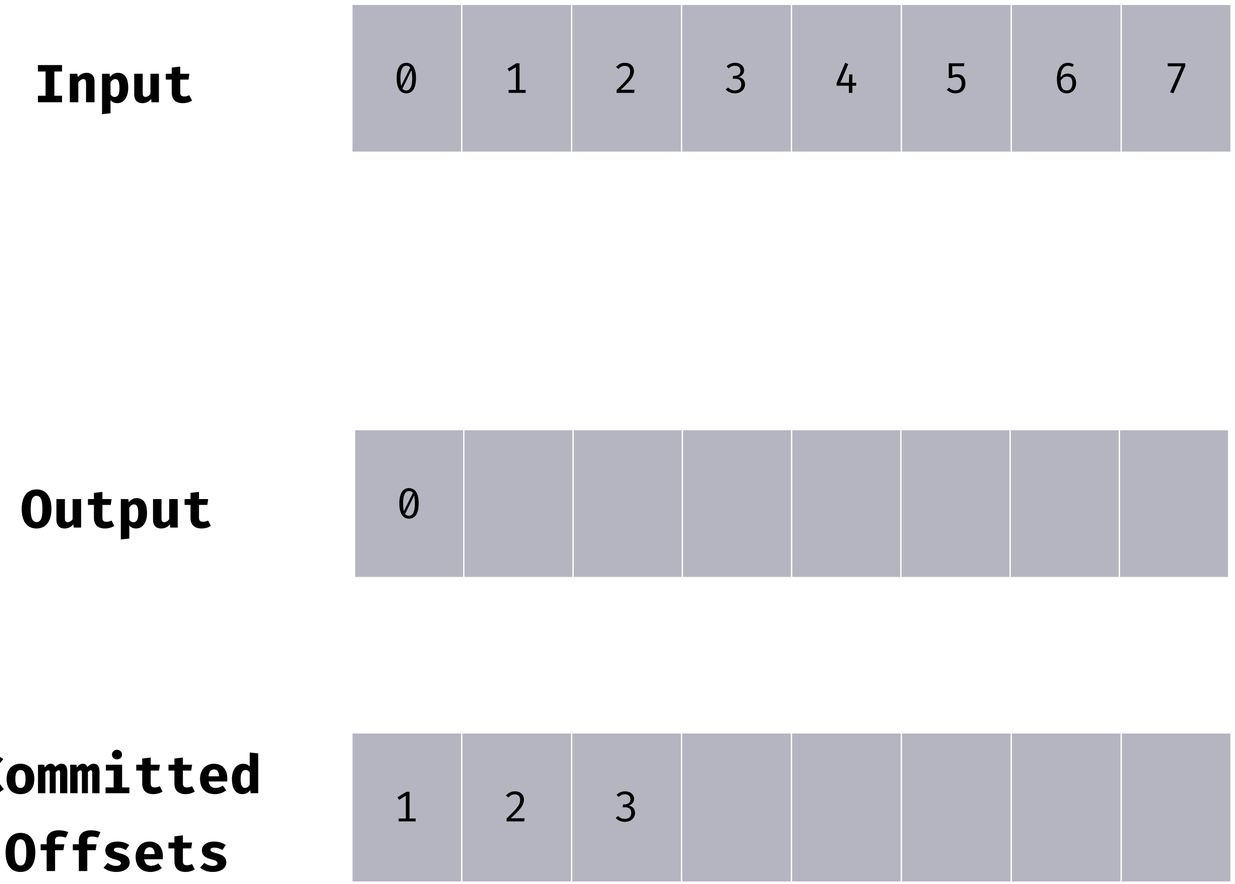



```
while (true) {
  data = c.poll()
  c.commitOffsets()
  p.send(data)
}
```

Restart!

Семантика «максимум один раз»

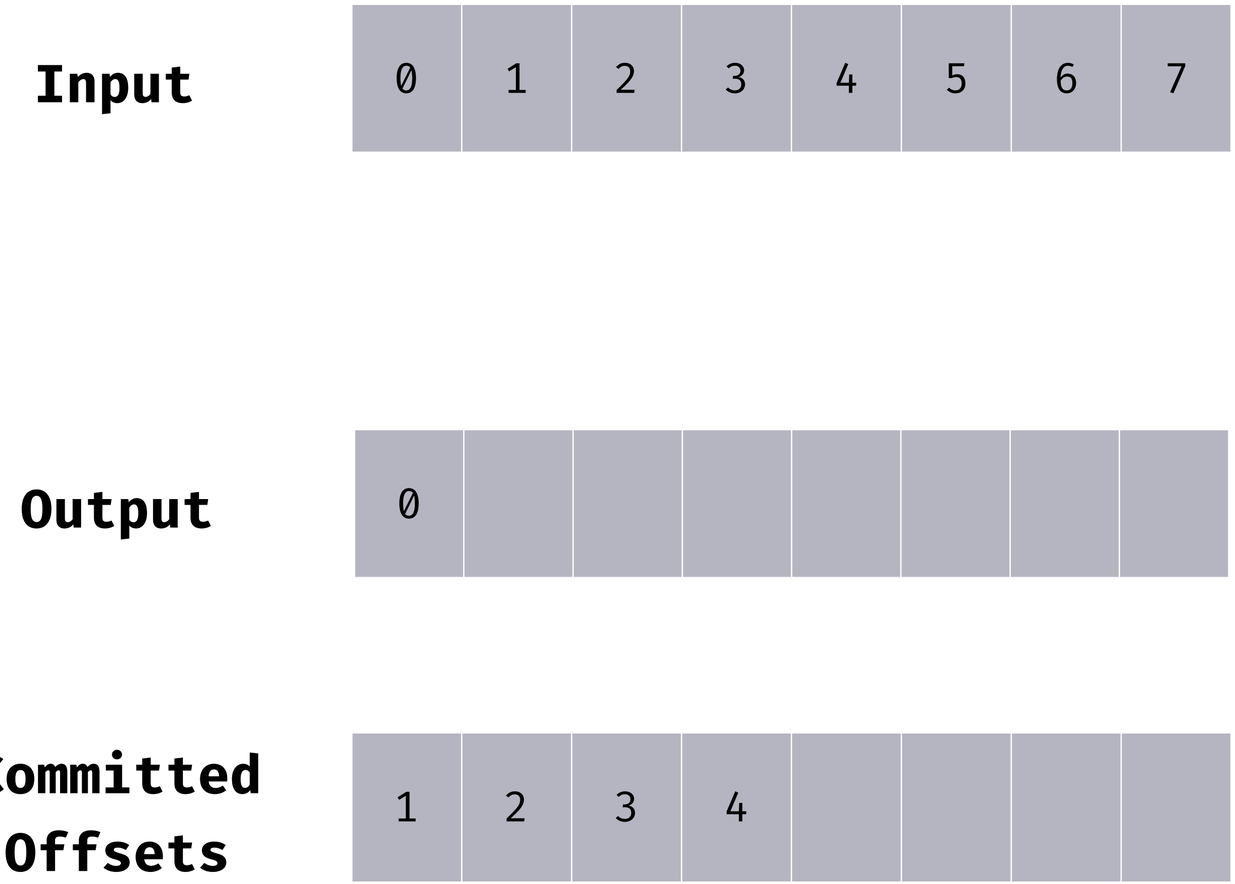
Вам это не надо




```
while (true) {
  data = c.poll()
  c.commitOffsets()
  p.send(data)
}
```

Семантика «максимум один раз»

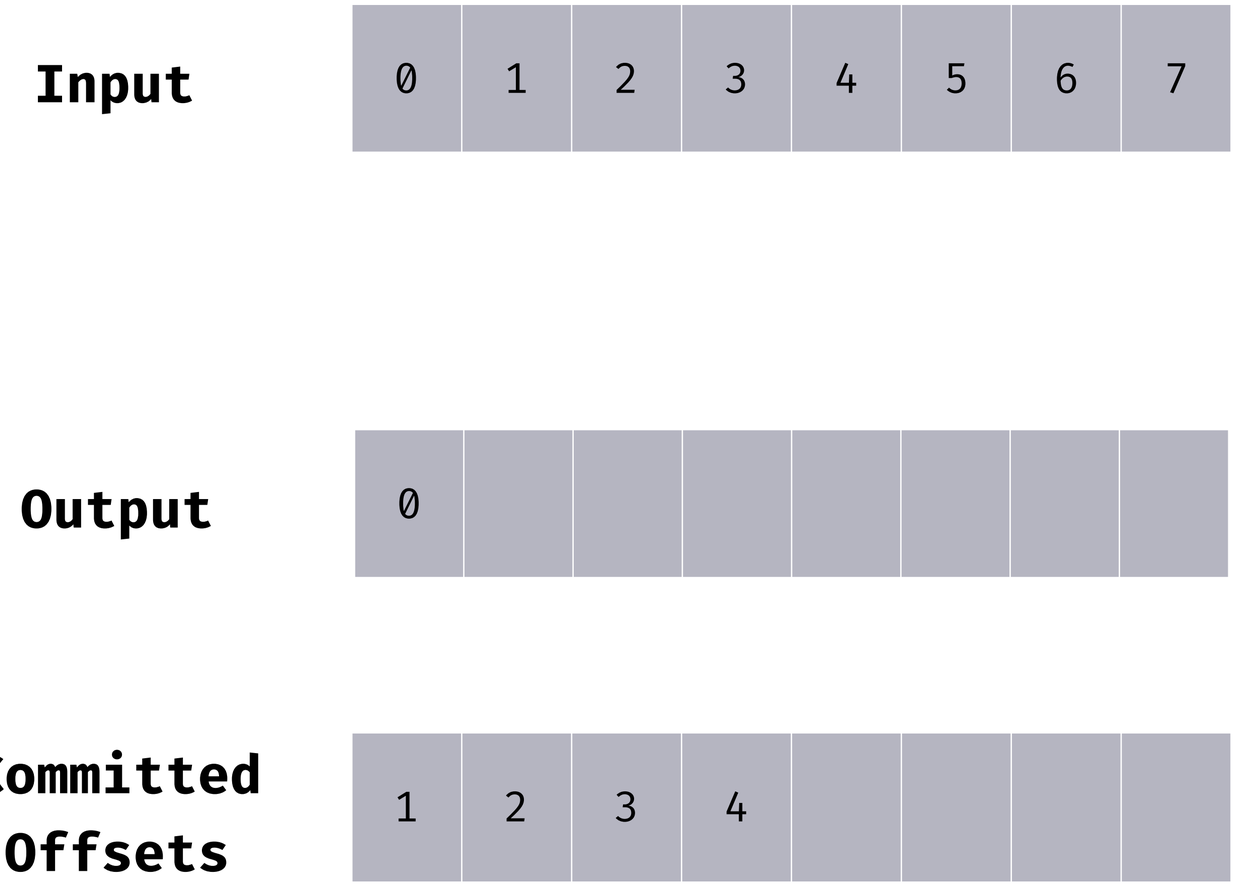
Вам это не надо




```
while (true) {
   data = c.poll()
   c.commitOffsets()
   p.send(data)
}
```

Семантика «максимум один раз»

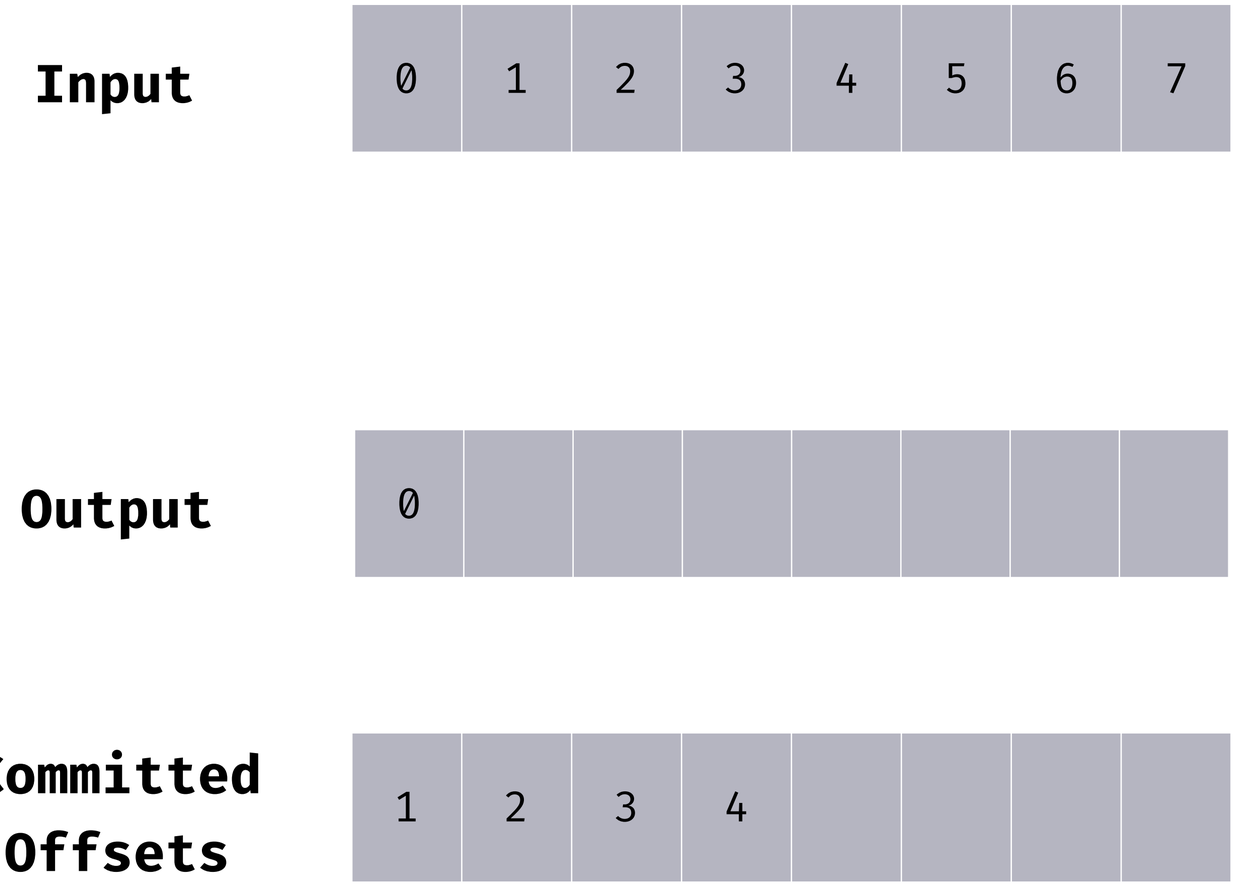
Вам это не надо




```
while (true) {
   data = c.poll()
   c.commitOffsets()
   p.send(data)
}
```

Семантика «максимум один раз»

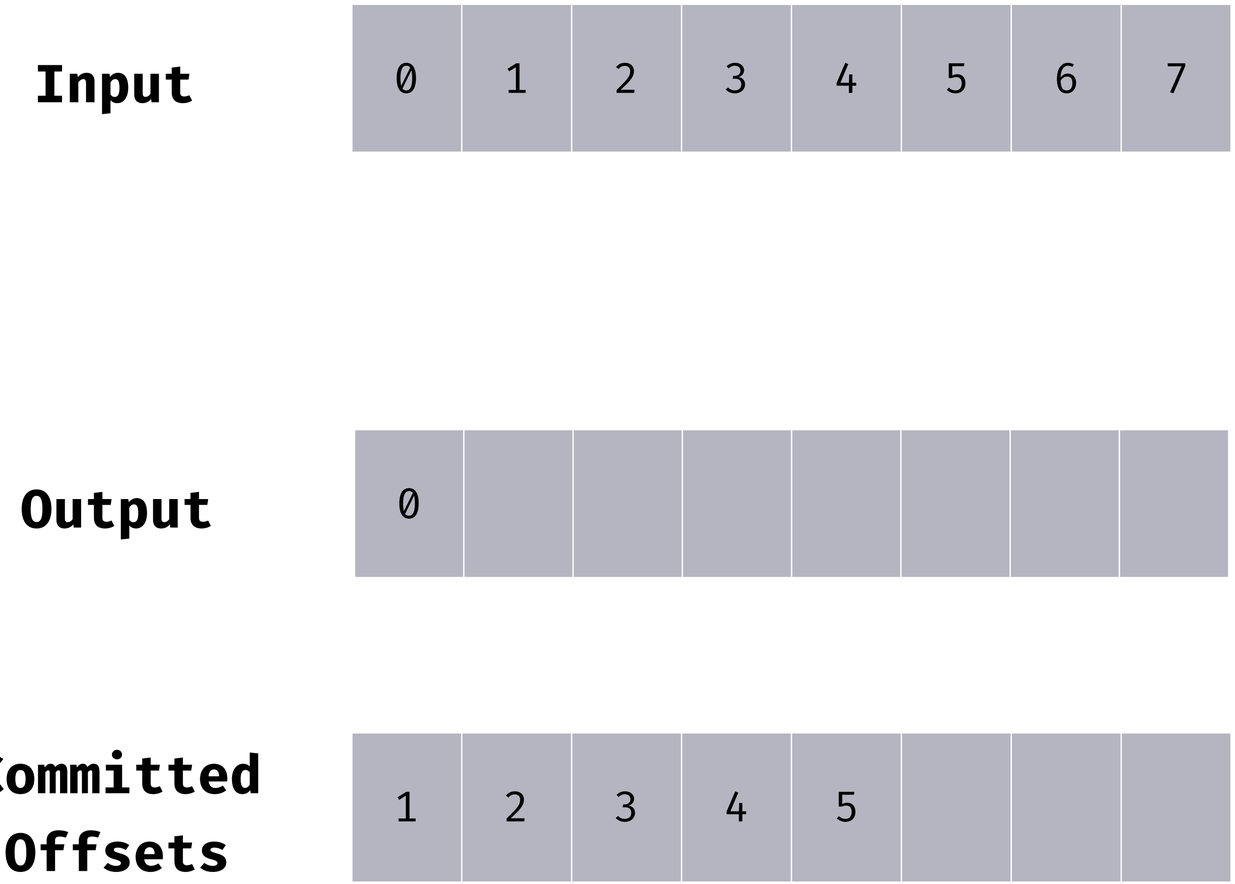
Вам это не надо




```
while (true) {
   data = c.poll()
   c.commitOffsets()
   p.send(data)
}
```

Семантика «максимум один раз»

Вам это не надо




```
while (true) {
   data = c.poll()
   c.commitOffsets()
   p.send(data)
}
```

Семантика «максимум один раз»

Вам это не надо

Input	0	1	2	3	4	5	6	7
Output	0							
Committed Offsets	1	2	3	4	5			

```
while (true) {
    data =                    ()
    c.comm                    ts()
    p.send
}
```

Семантика «максимум один раз»

Вам это не надо

Input

0	1	2	3	4	5	6	7

Output

0							

Committed Offsets

1	2	3	4	5			

```
while
  data  ll()
  c.co  sets()
  p.send(data)
}
```

```
while (true) {
  data = c.poll()
  c.commitOffsets()
  p.send(data)
}
```

Семантика «максимум один раз»

Вам это не надо

Input

0	1	2	3	4	5	6	7

Output

0							

Committed Offsets

1	2	3	4	5			

```
while (true) {
   data
   c.co
   p.se
}
```

```
while (true) {
   data = c.poll()
   c.commitOffsets()
   p.send(data)
}
```

Семантика «максимум один раз»

Вам это не надо

Input	0	1	2	3	4	5	6	7

Output	0							

Committed Offsets	1	2	3	4	5	6		

```
while (true) {
   da          poll()

c.co           sets()
   p.send(data)
}
```

```
while (true) {
   data = c.poll()
   c.commitOffsets()
   p.send(data)
}
```

Семантика «максимум один раз»

Вам это не надо

Input

0	1	2	3	4	5	6	7

Output

0	3	5					

Committed Offsets

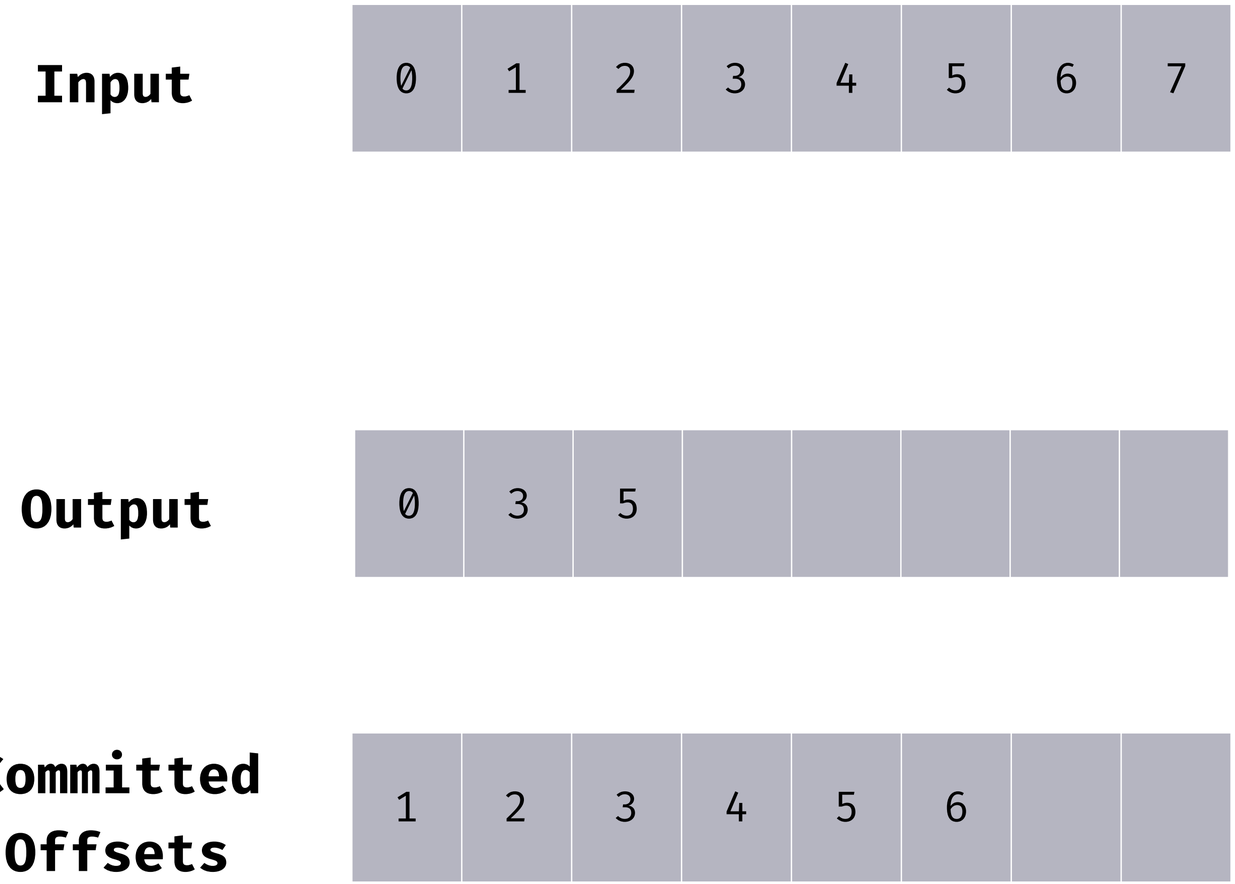
1	2	3	4	5	6		

```
while (
   data
   c.com
   p.sen
}
```

```
while (true) {
   data = c.poll()
   c.commitOffsets()
   p.send(data)
}
```

Семантика «максимум один раз»

Вам это не надо

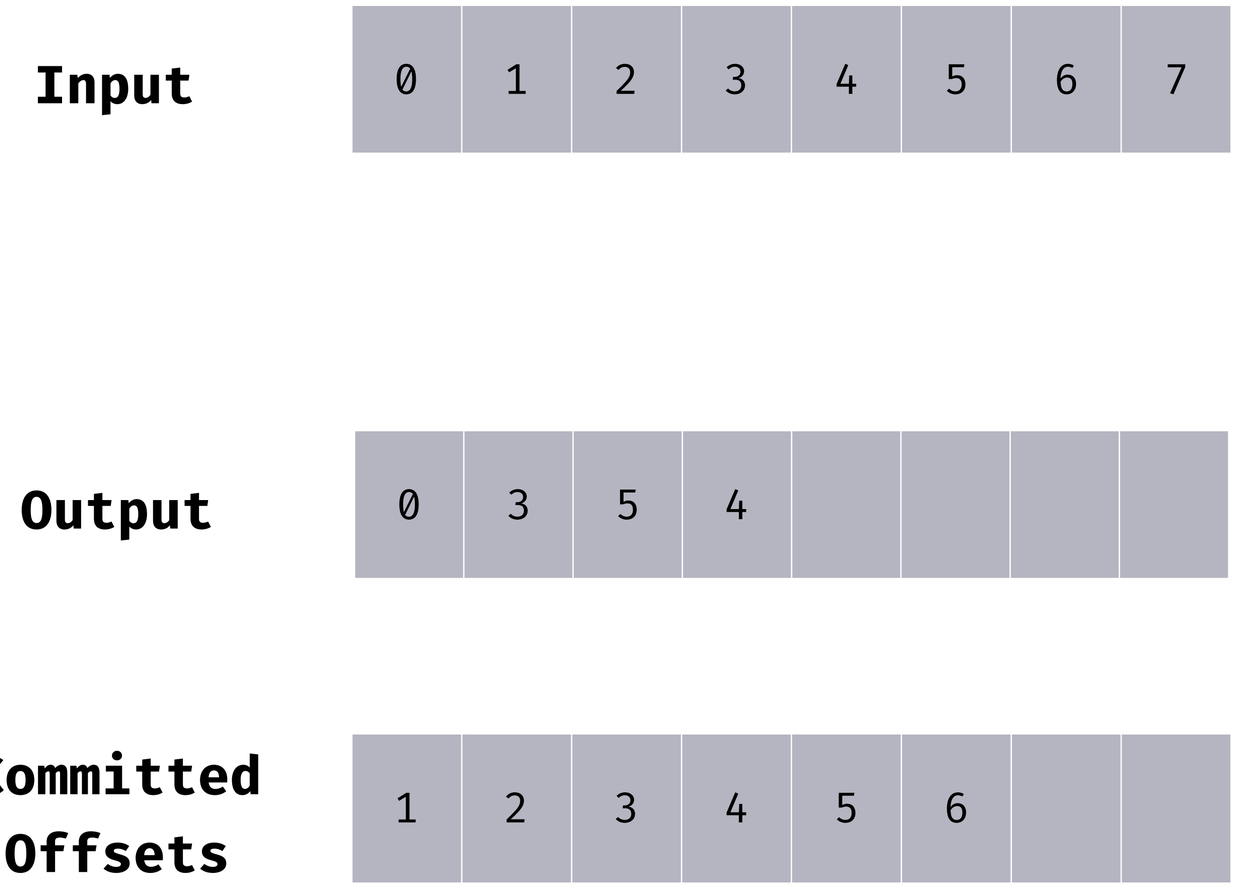



```
while (true) {
  data = c.poll()
  c.commitOffsets()
  p.send(data)
}
```

```
while (true) {
  data = c.poll()
  c.commitOffsets()
  p.send(data)
}
```

Семантика «максимум один раз»

Вам это не надо




```
while (true) {
  data = c.poll()
  c.commitOffsets()
  p.send(data)
}
```

```
while (true) {
  data = c.poll()
  c.commitOffsets()
  p.send(data)
}
```

Семантика «максимум один раз»

Вам это не надо

Input	0	1	2	3	4	5	6	7
Output	0	3	5	4				
Committed Offsets	1	2	3	4	5	6		

- Потери сообщений
  - Повторы продьюсера не разрешены
  - Сбои перед тем как записываем данные
- Зомби могут  порядок сообщений

Слабости семантики

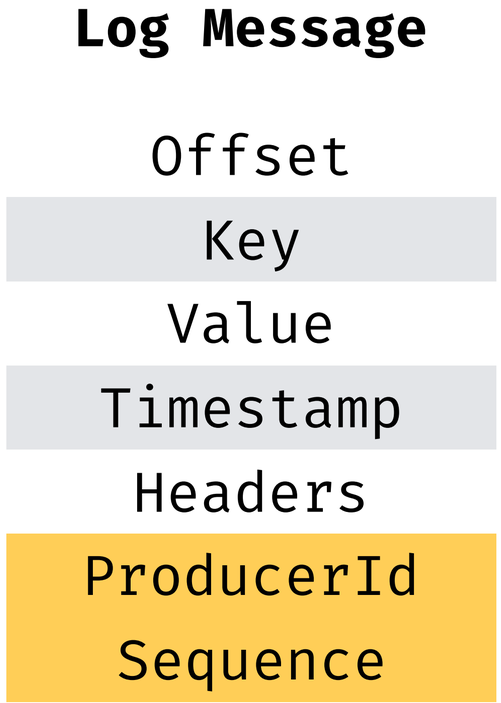
- Повторные попытки продюсера небезопасны
- Данные не пишутся атомарно с их смещениями (offsets)
- Нет защиты от зомби

Слабости семантики

- Повторные попытки продюсера небезопасны
- Данные не пишутся атомарно с их смещениями (offsets)
- Нет защиты от зомби

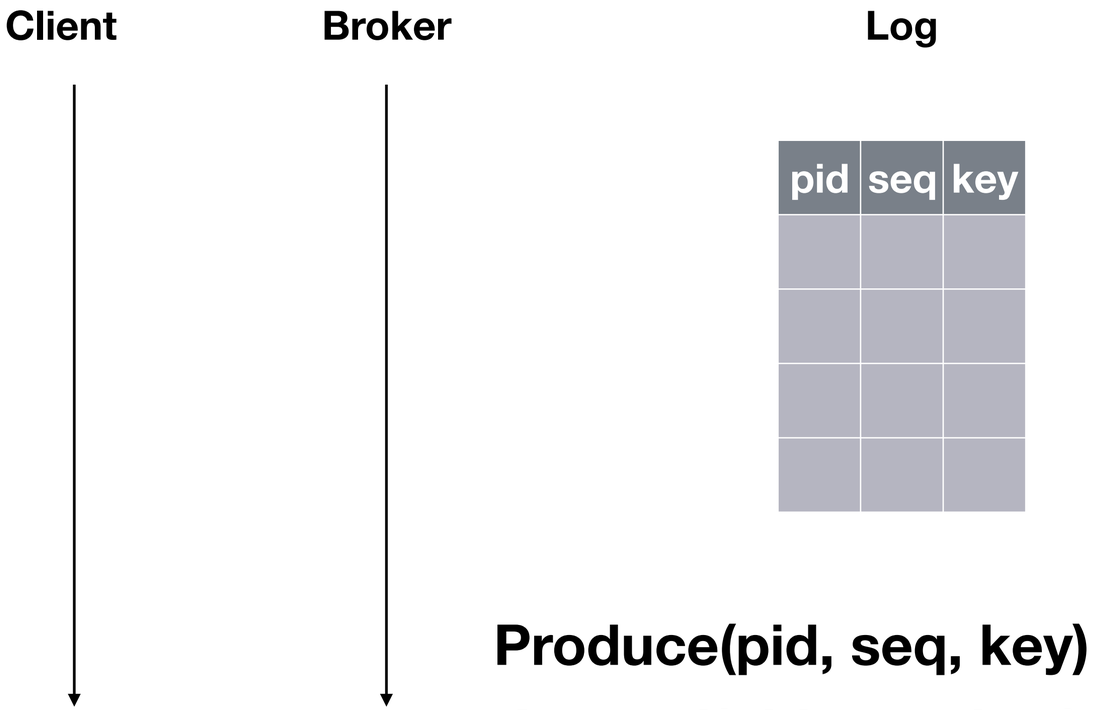
Идемпотентный продюсер

- `ProducerId` определяет экземпляр продюсера
- `Sequence` служит для дедупликации
- `enable.idempotence=true`



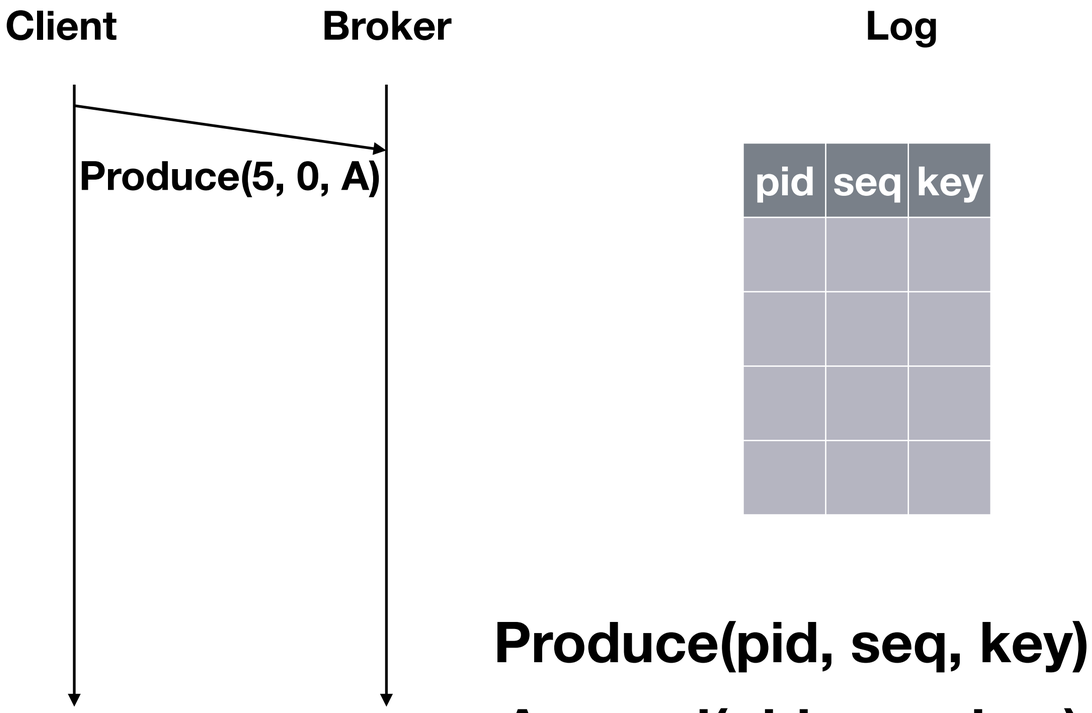
Идемпотентный продюсер

- `ProducerId` определяет экземпляр продюсера
- `Sequence` служит для дедупликации



Идемпотентный продюсер

- `ProducerId` определяет экземпляр продюсера
- `Sequence` служит для дедупликации

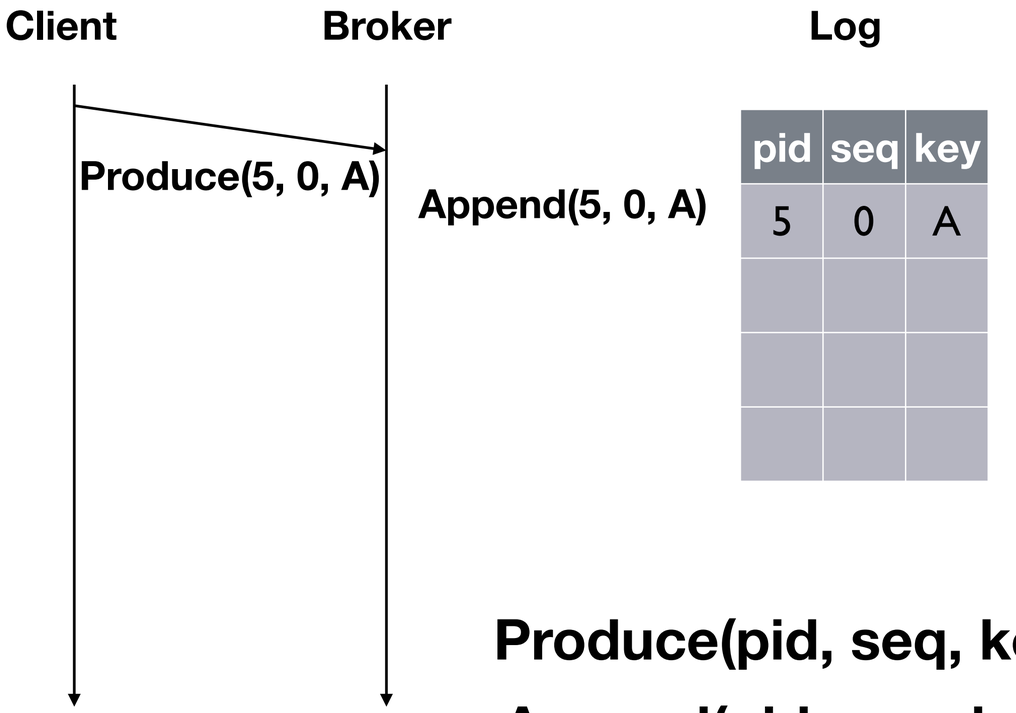


Produce(pid, seq, key)

Append(pid, seq, key)

Идемпотентный продюсер

- `ProducerId` определяет экземпляр продюсера
- `Sequence` служит для дедупликации

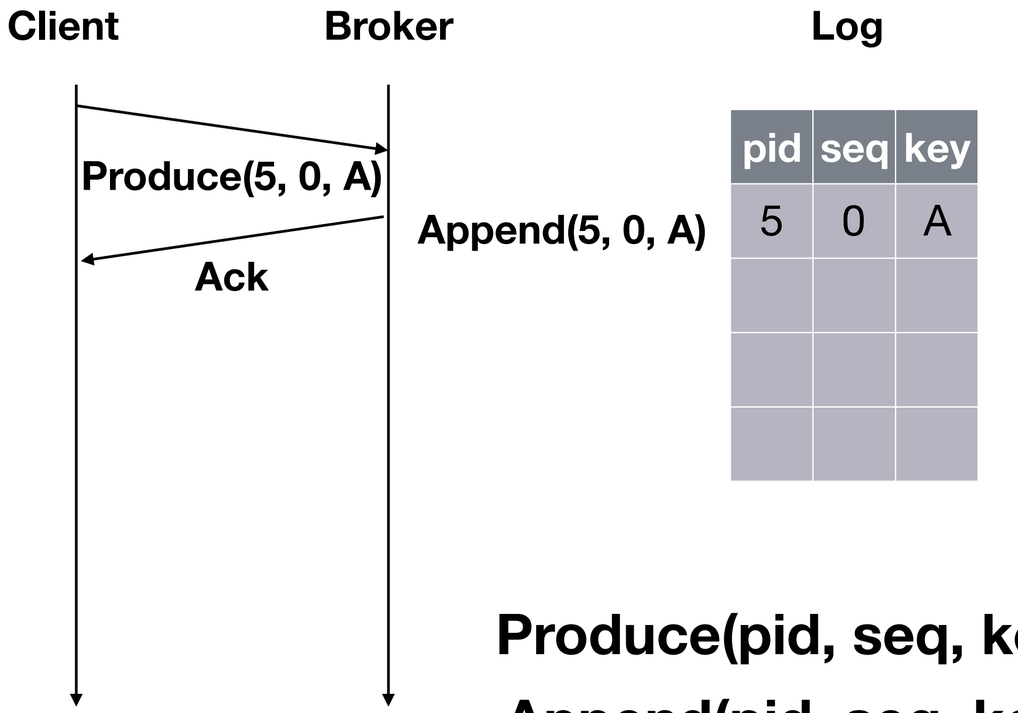


Produce(pid, seq, key)

Append(pid, seq, key)

Идемпотентный продюсер

- `ProducerId` определяет экземпляр продюсера
- `Sequence` служит для дедупликации

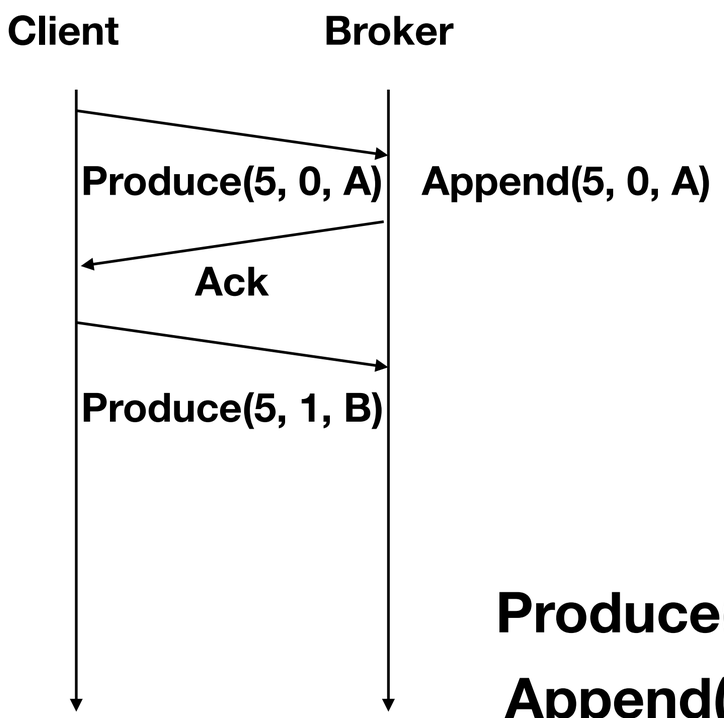


Produce(pid, seq, key)

Append(pid, seq, key)

Идемпотентный продюсер

- `ProducerId` определяет экземпляр продюсера
- `Sequence` служит для дедупликации



Log

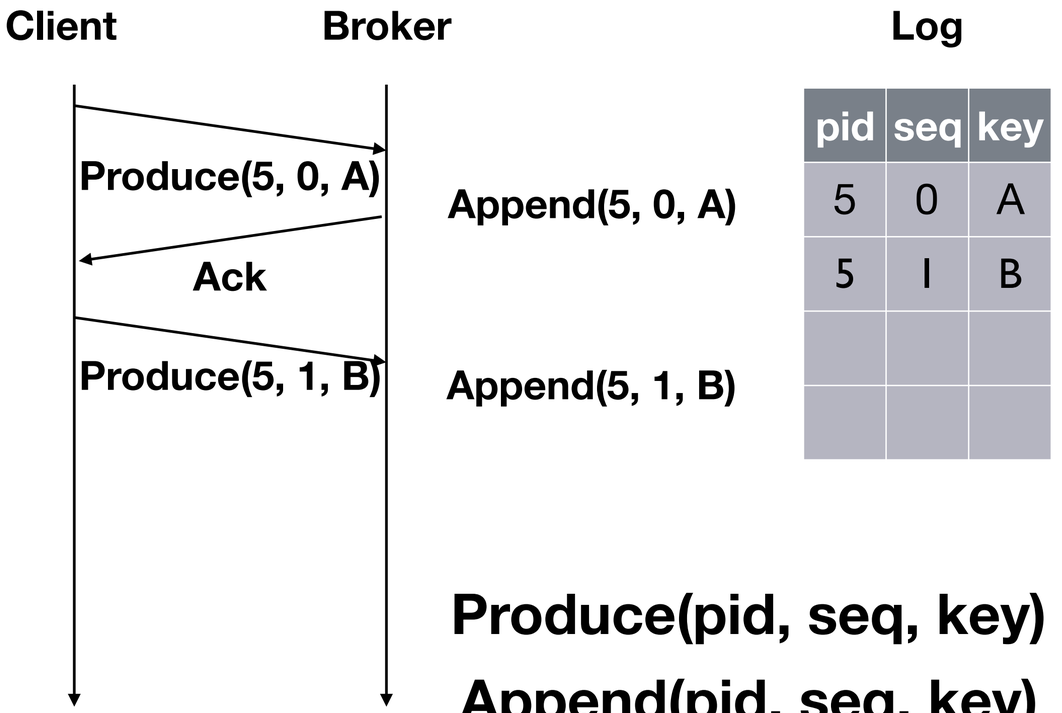
pid	seq	key
5	0	A

Produce(pid, seq, key)

Append(pid, seq, key)

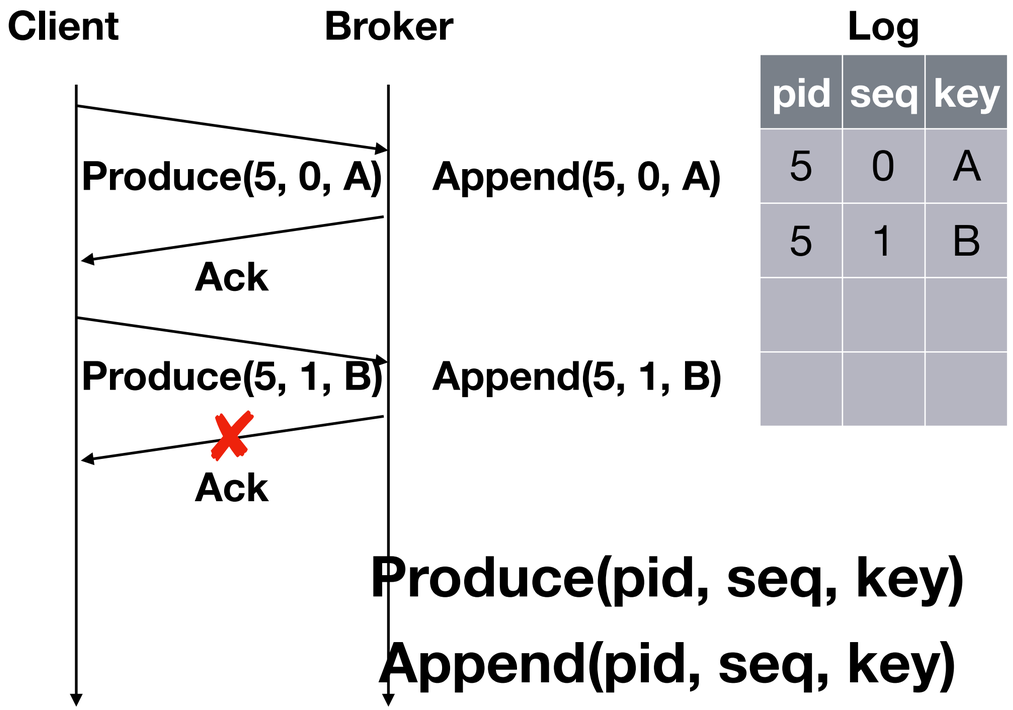
Идемпотентный продюсер

- `ProducerId` определяет экземпляр продюсера
- `Sequence` служит для дедупликации



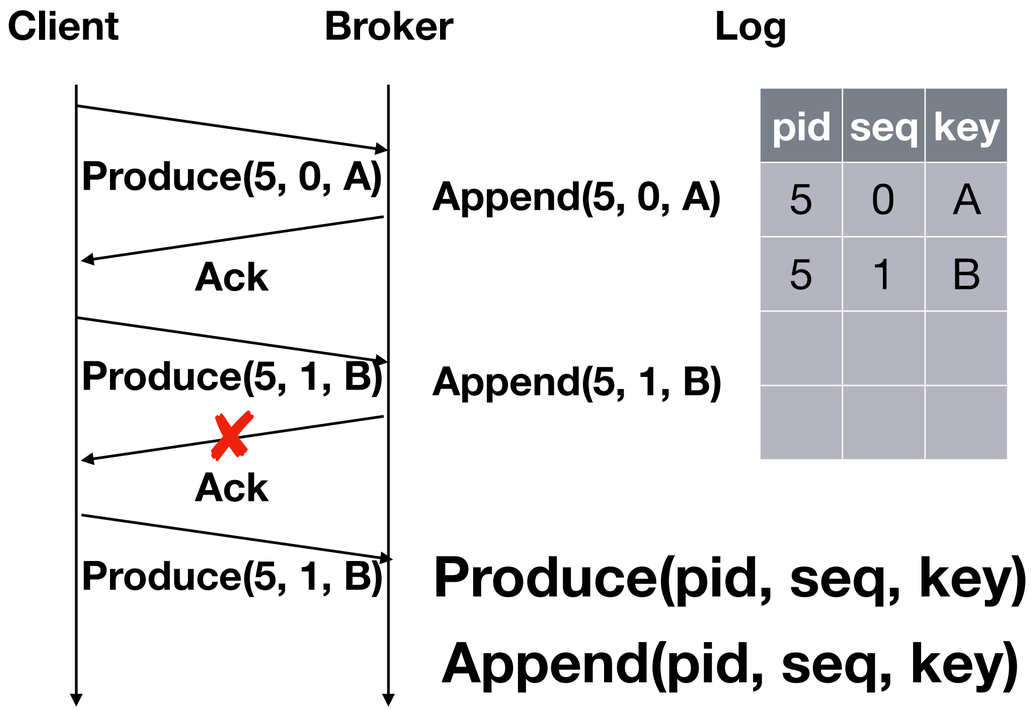
Идемпотентный продюсер

- `ProducerId` определяет экземпляр продюсера
- `Sequence` служит для дедупликации



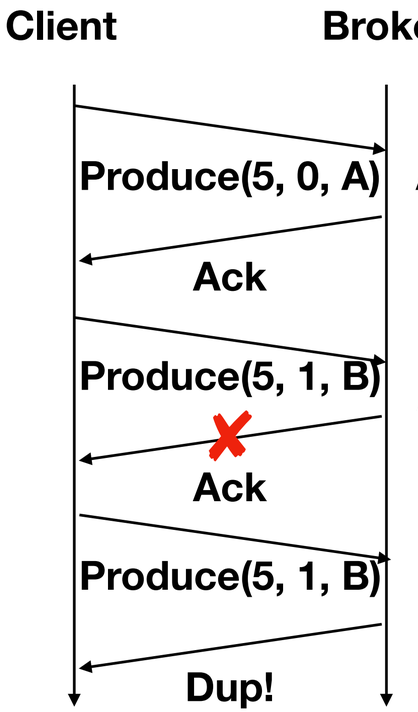
Идемпотентный продюсер

- `ProducerId` определяет экземпляр продюсера
- `Sequence` служит для дедупликации



Идемпотентный продюсер

- `ProducerId` определяет экземпляр продюсера
- `Sequence` служит для дедупликации



Append(5, 0, A)

Append(5, 1, B)

Log

pid	seq	key
5	0	A
5	1	B

Produce(pid, seq, key)

Append(pid, seq, key)

Слабости семантики

- Повторные попытки продюсера небезопасны
- Данные не пишутся атомарно с их смещениями (offsets)
- Нет защиты от зомби

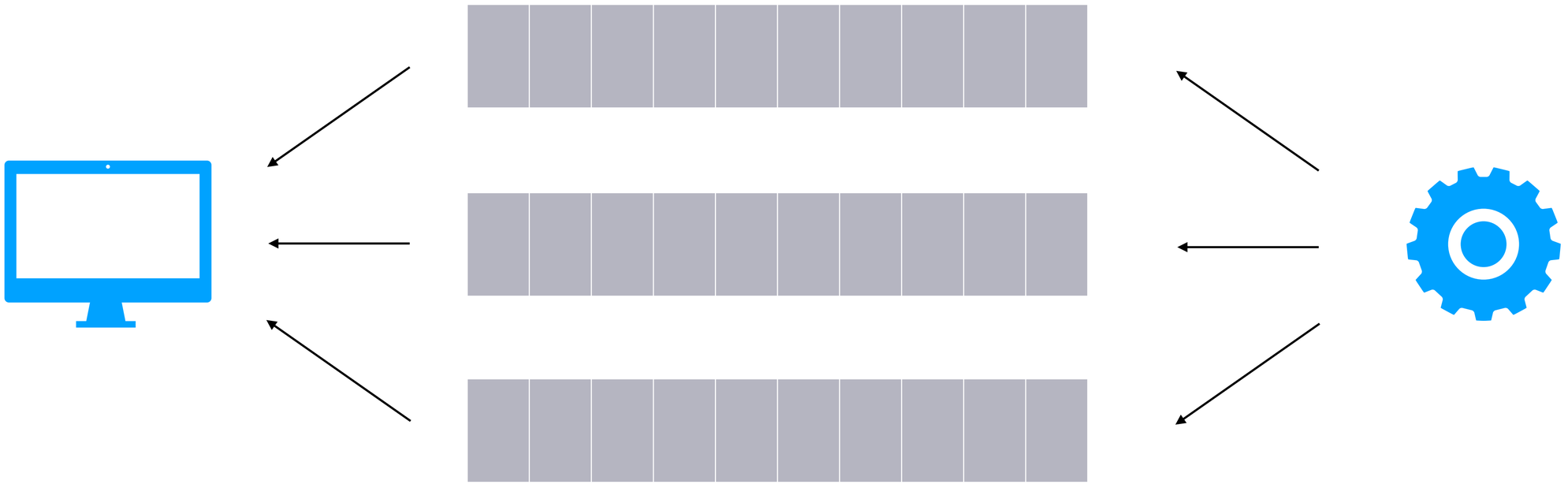
Слабости семантики

- ~~Повторные попытки продюсера небезопасны~~
- Данные не пишутся атомарно с их смещениями (offsets)
- Нет защиты от зомби

Слабости семантики

- ~~Повторные попытки продюсера небезопасны~~
- Данные не пишутся атомарно с их смещениями (offsets)
- Нет защиты от зомби

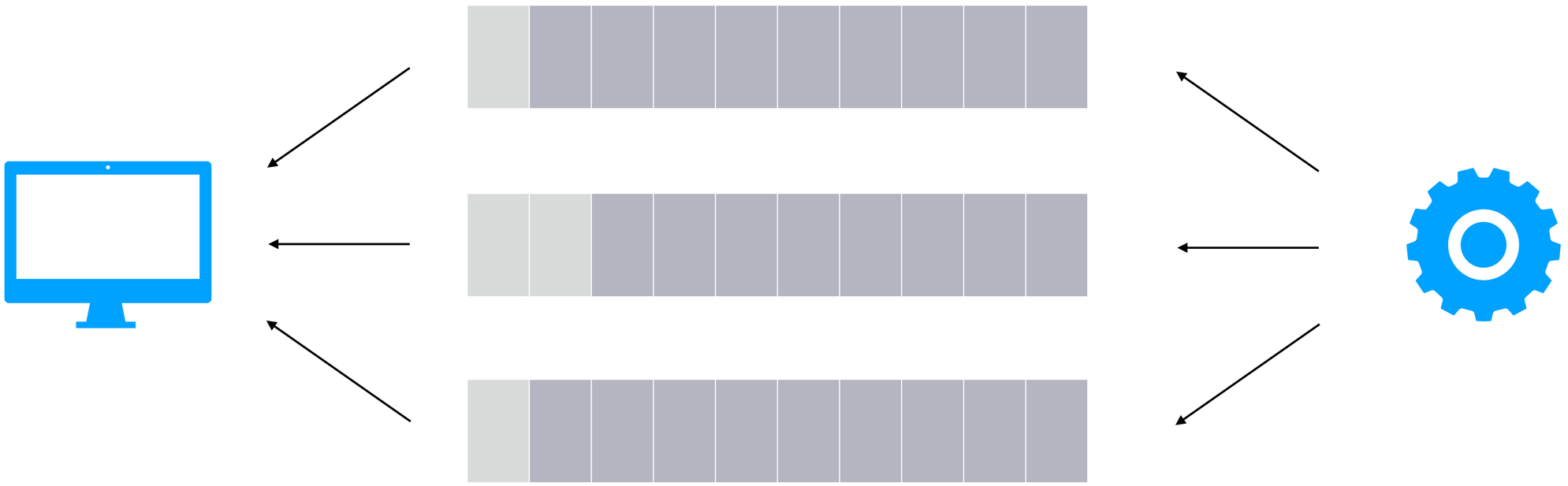
Снэпшоты Чэнди - Лэмпорта



Observer

Output topics

Processor

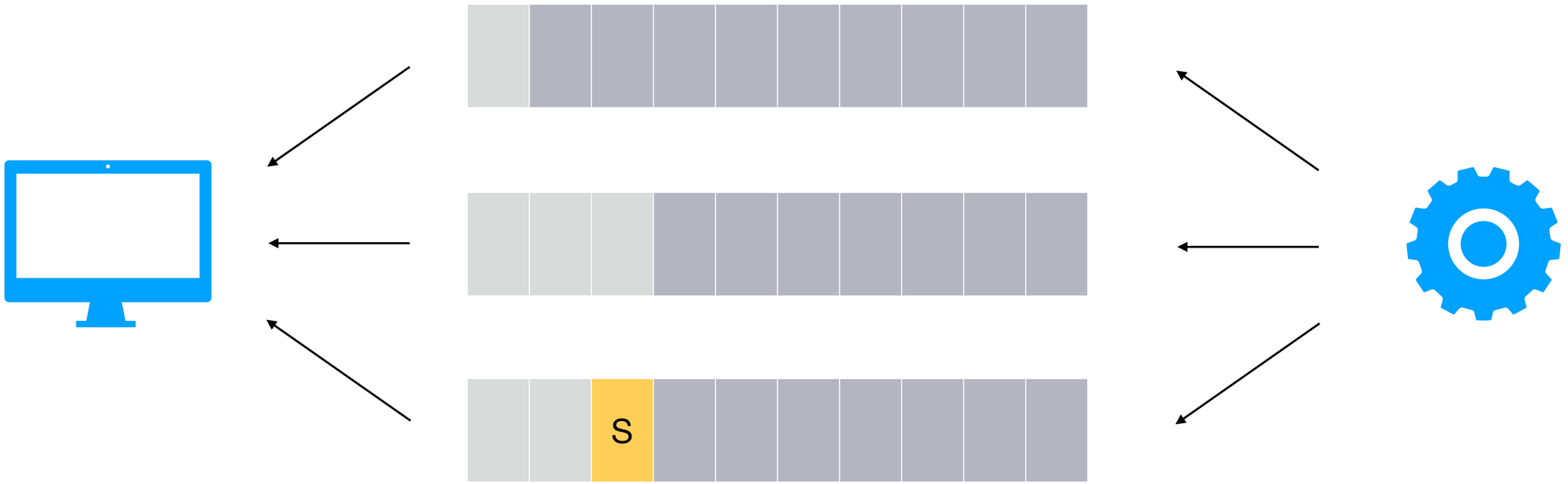


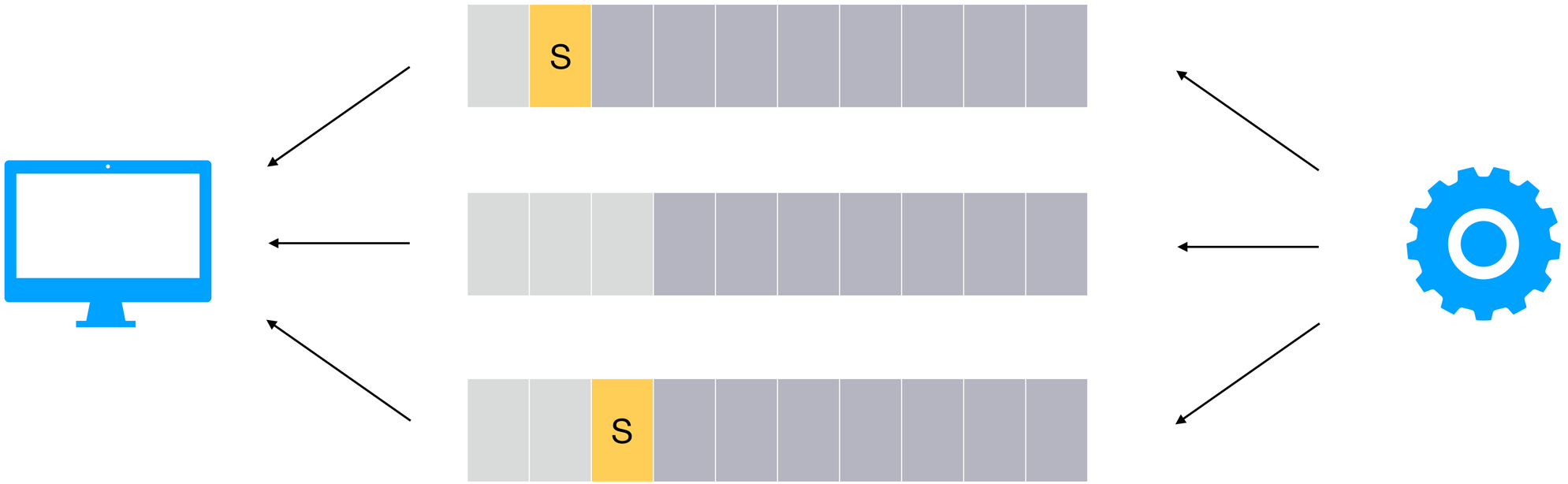
Observer

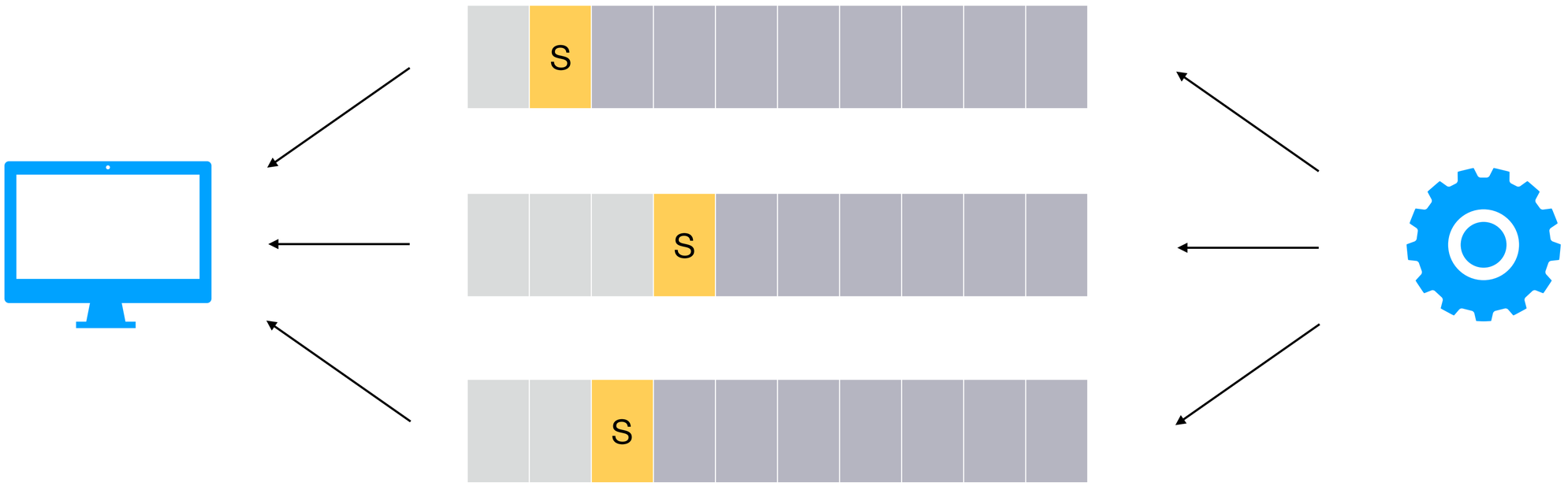
Output topics

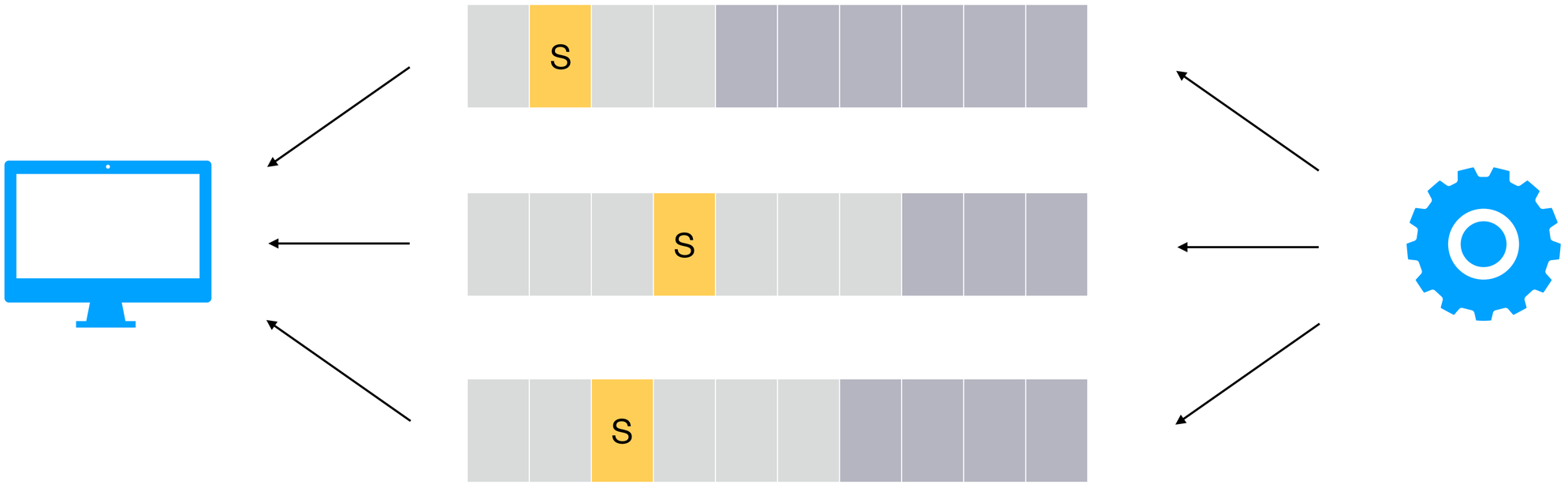
Processor

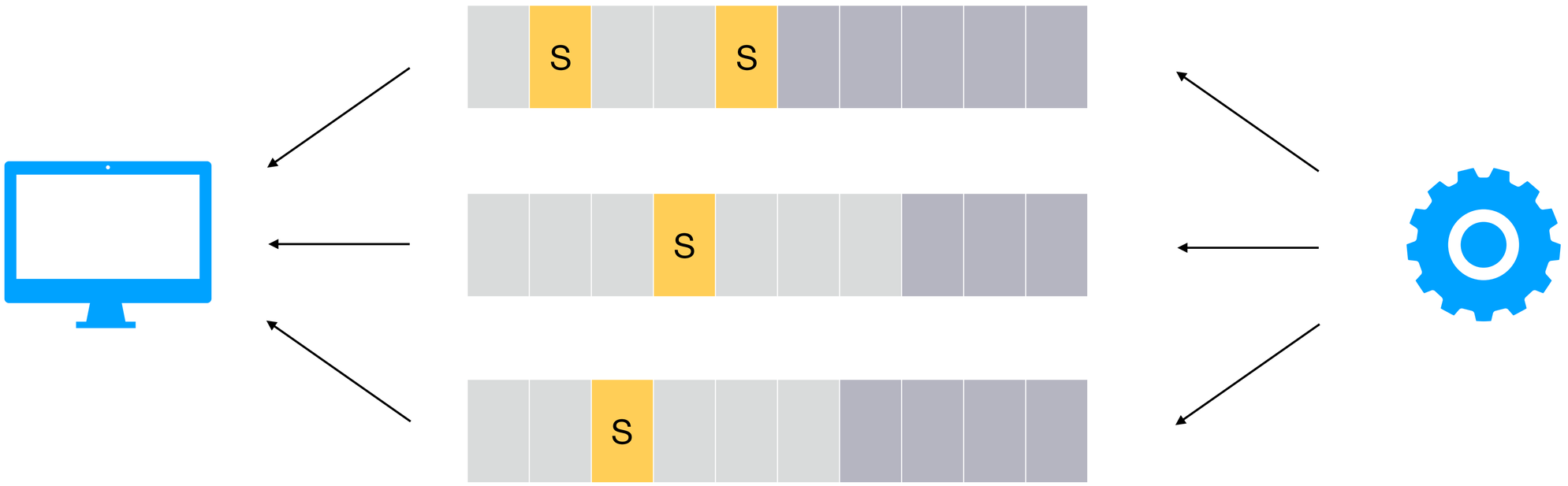

Observer
Output topics
Processor

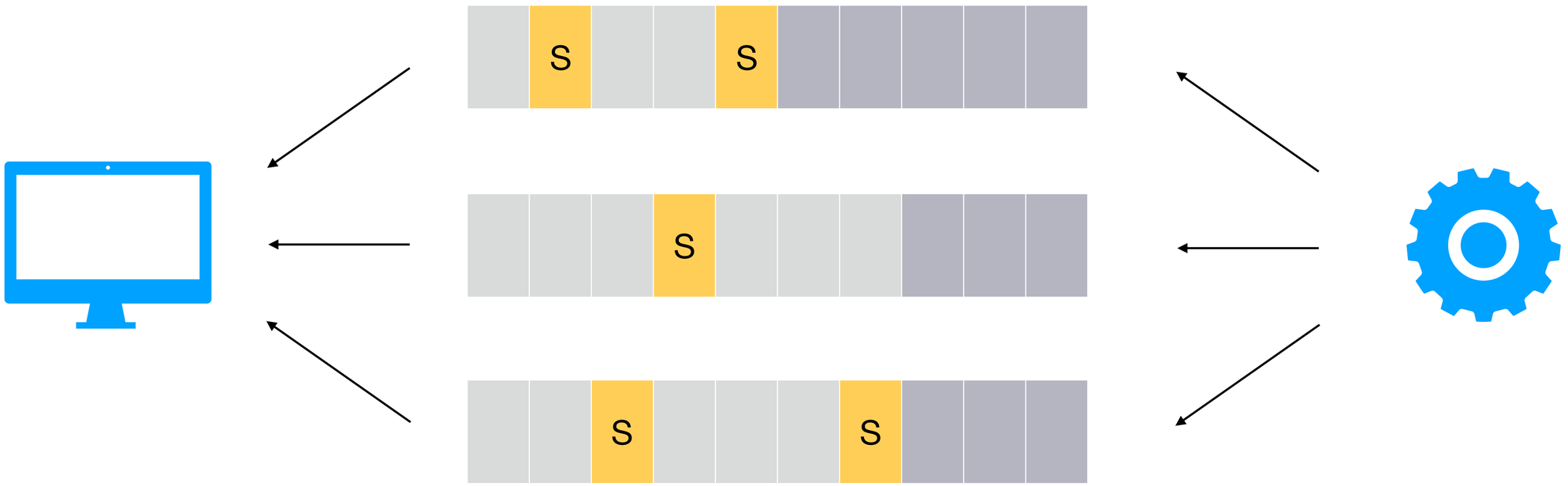

S
Observer
Output topics
Processor

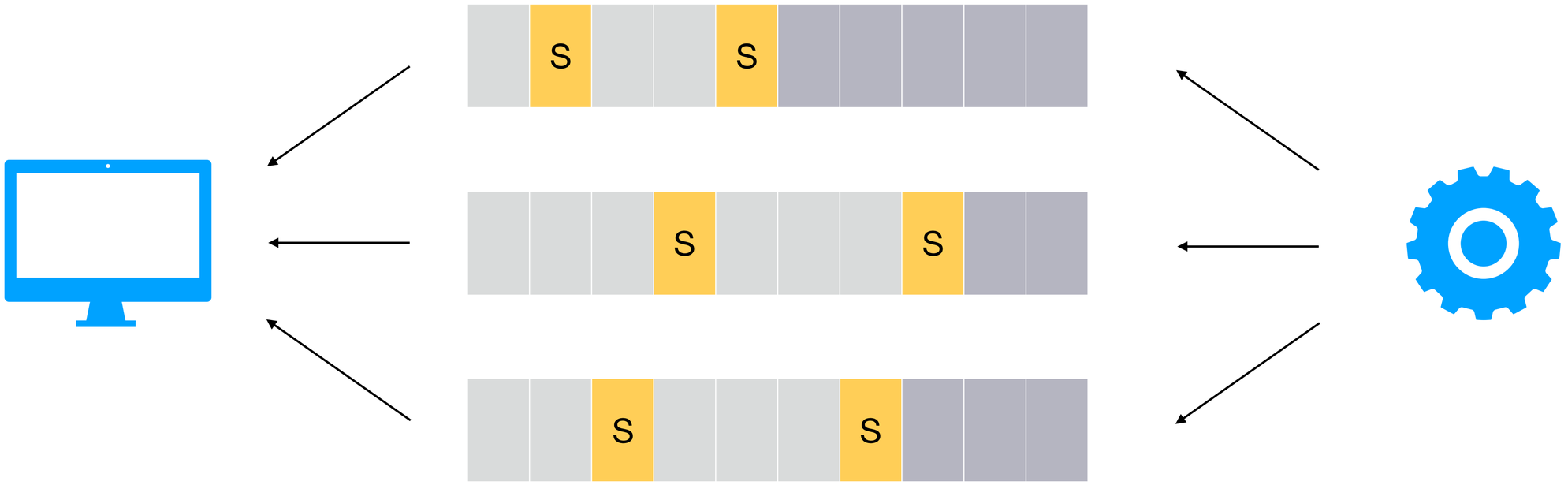

S
S
Observer
Output topics
Processor


S
S
S
Observer
Output topics
Processor

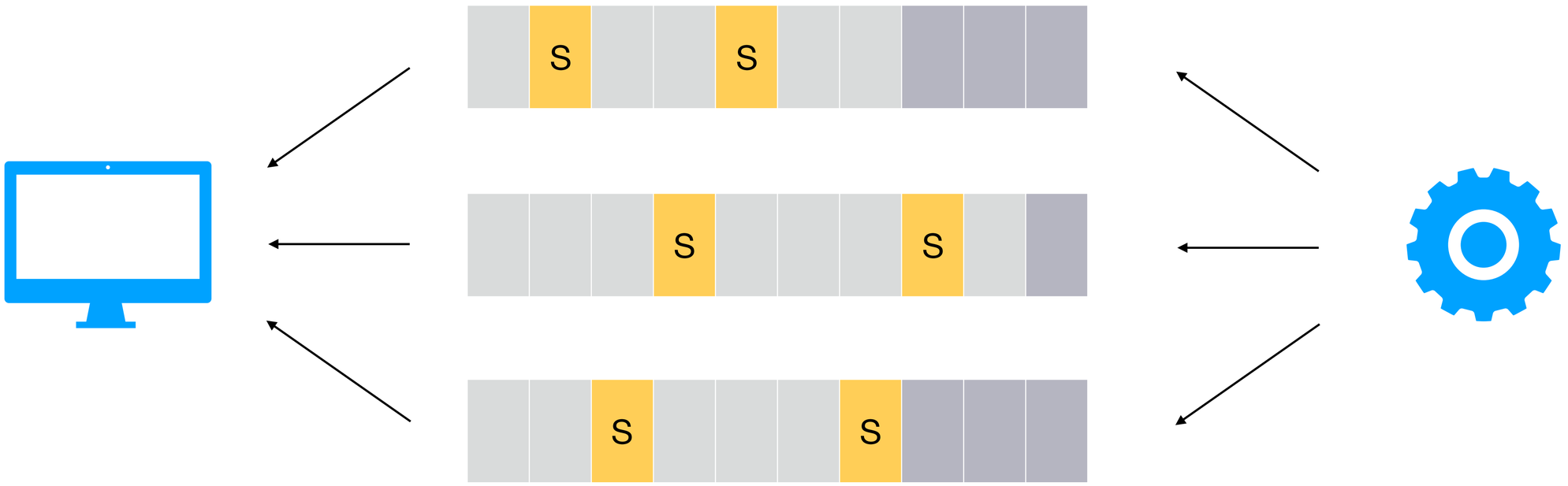

S
S
S
Observer
Output topics
Processor


S
S
S
S
Observer
Output topics
Processor


S
S
S
S
Observer
Output topics
Processor

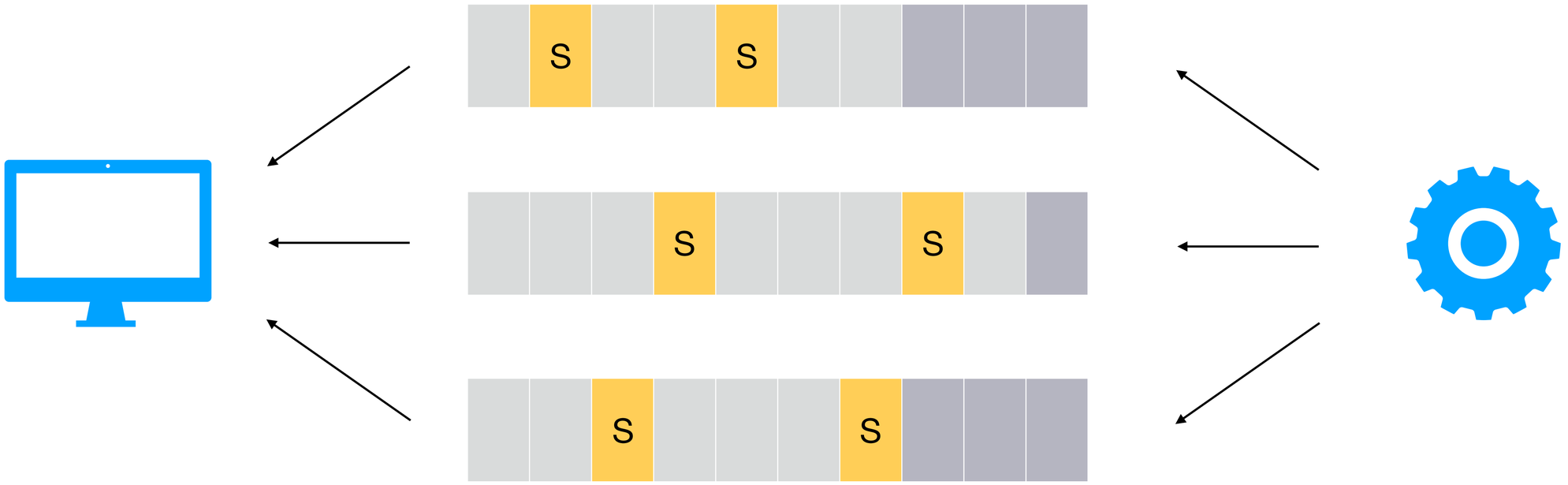


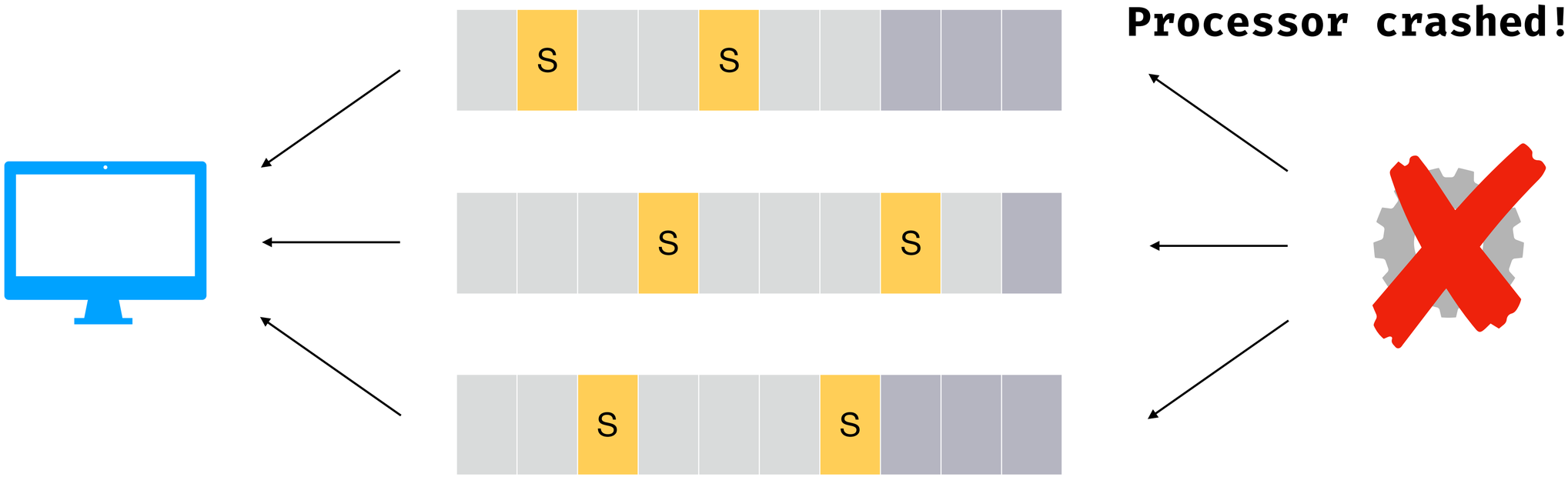
Observer **Output topics** **Processor**

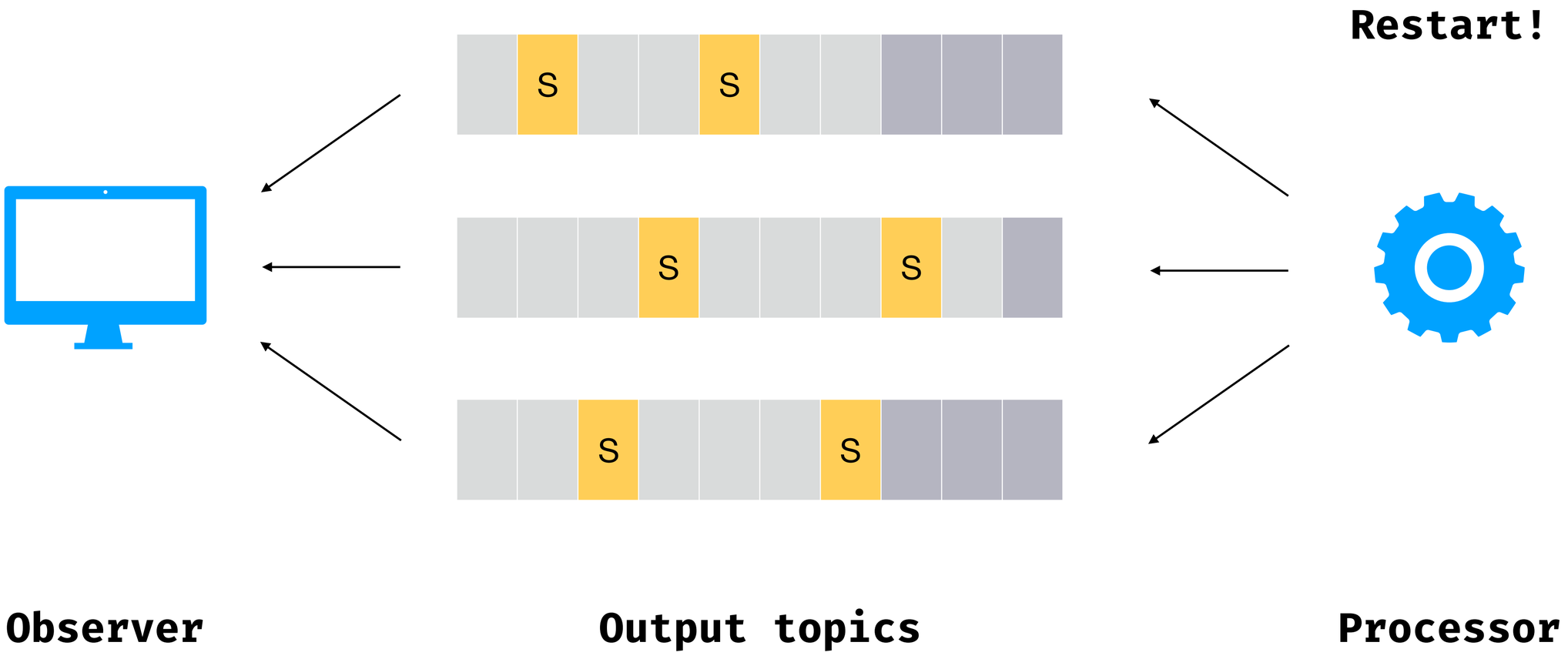

S
S
S
S
S
S
Observer
Output topics
Processor

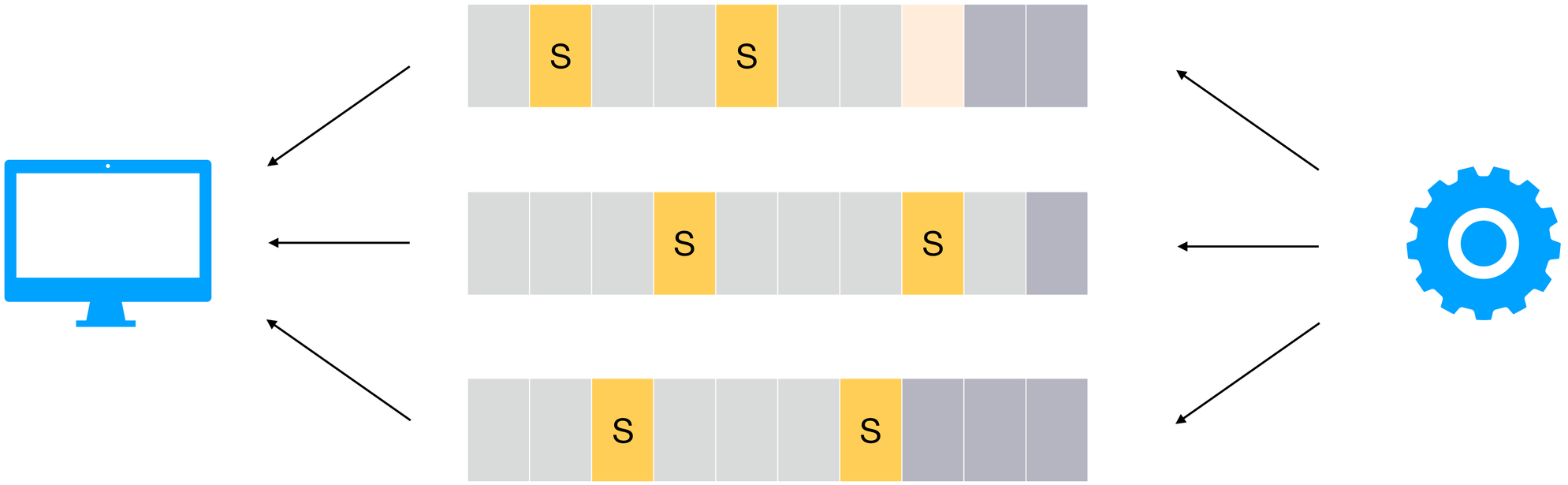
Снэпшоты Чэнди - Лэмпорта

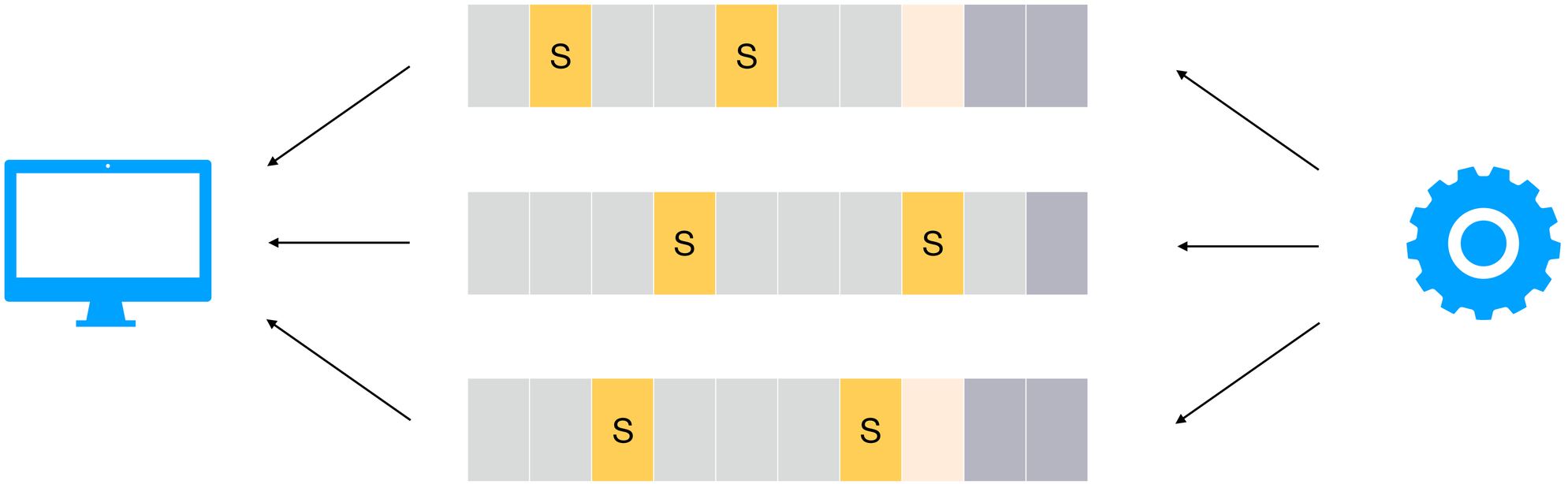
- А как же быть с со сбоями?
  - Soft: сбой во время обработки
  - Hard: сбой до или во время снятия снэпшота

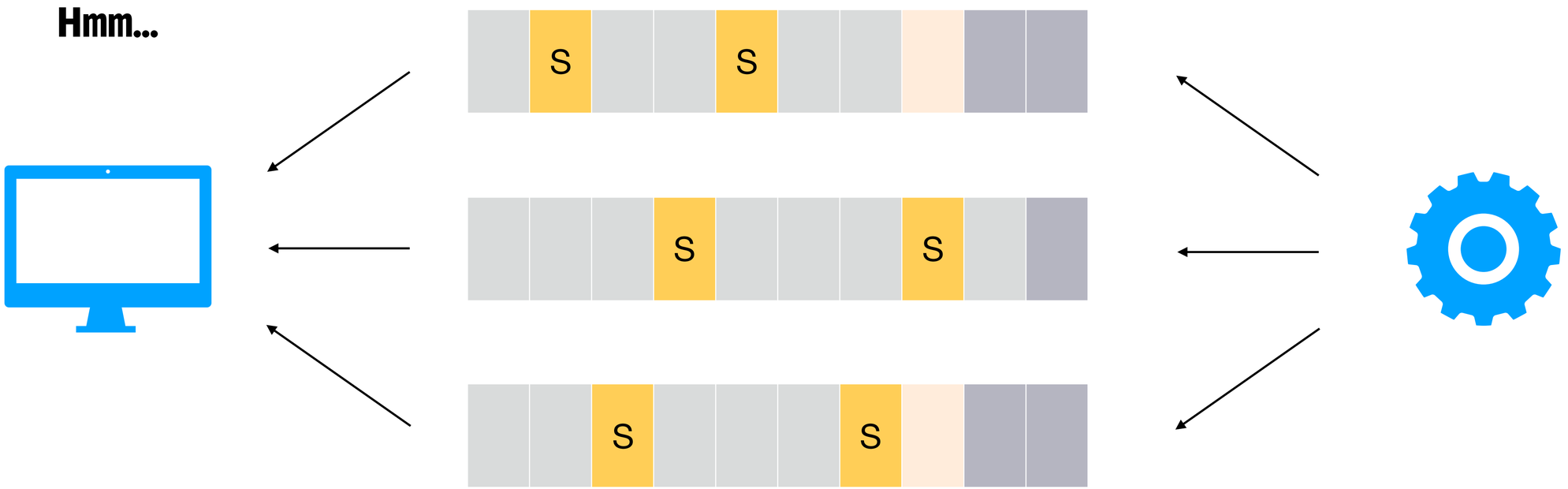

S
S
S
S
S
S
Observer
Output topics
Processor

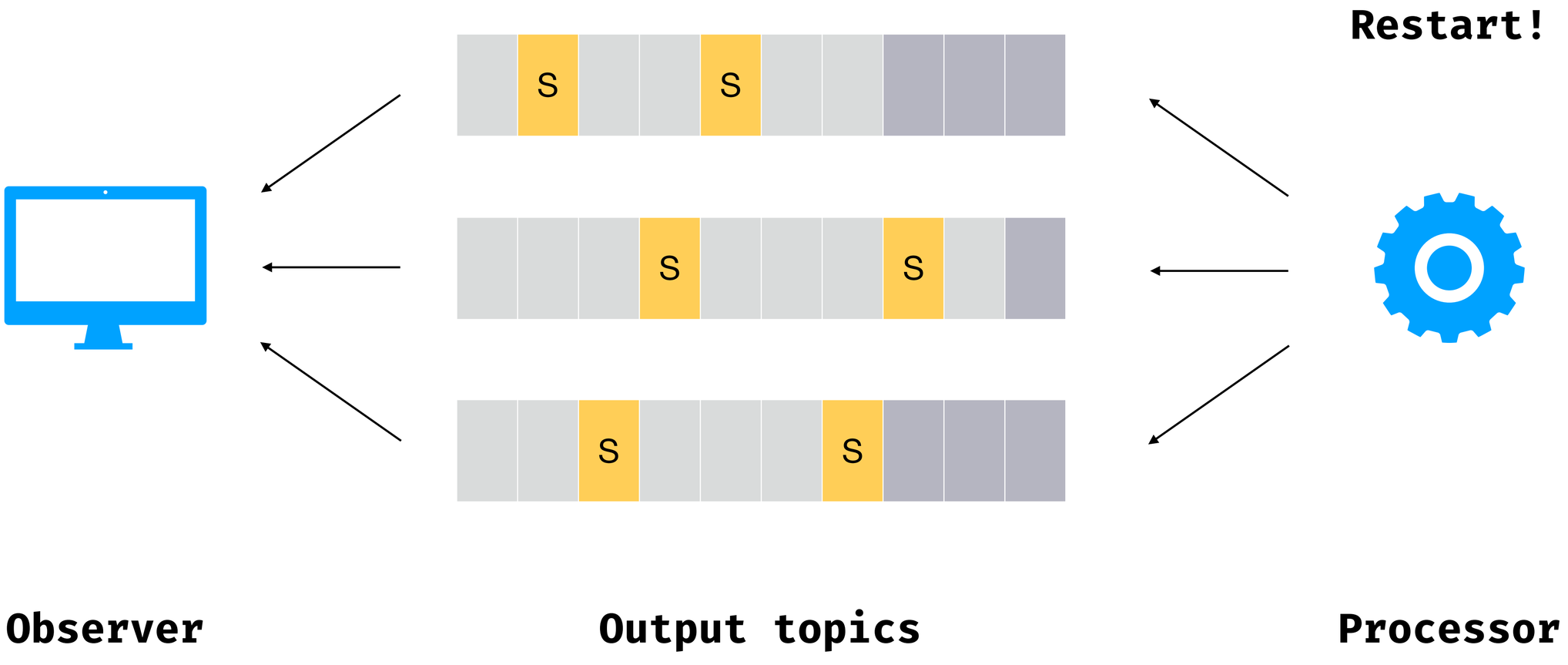

Processor crashed!
S
S
S
S
S
S
Observer
Output topics
Processor

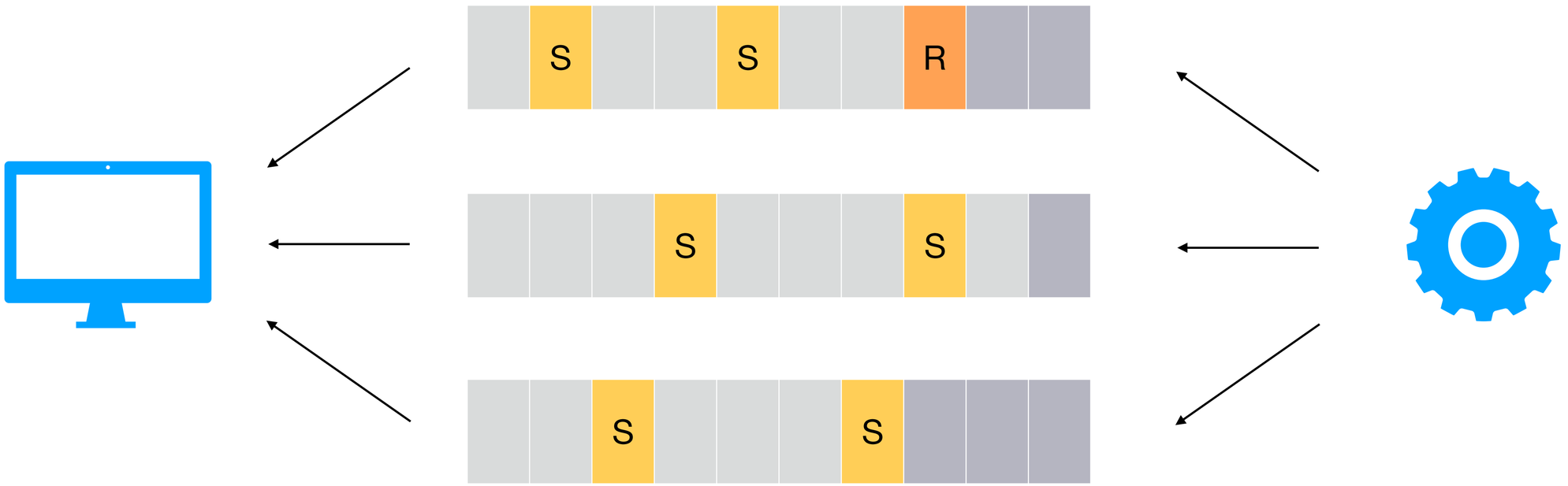

Restart!
S
S
S
S
S
S
Observer
Output topics
Processor

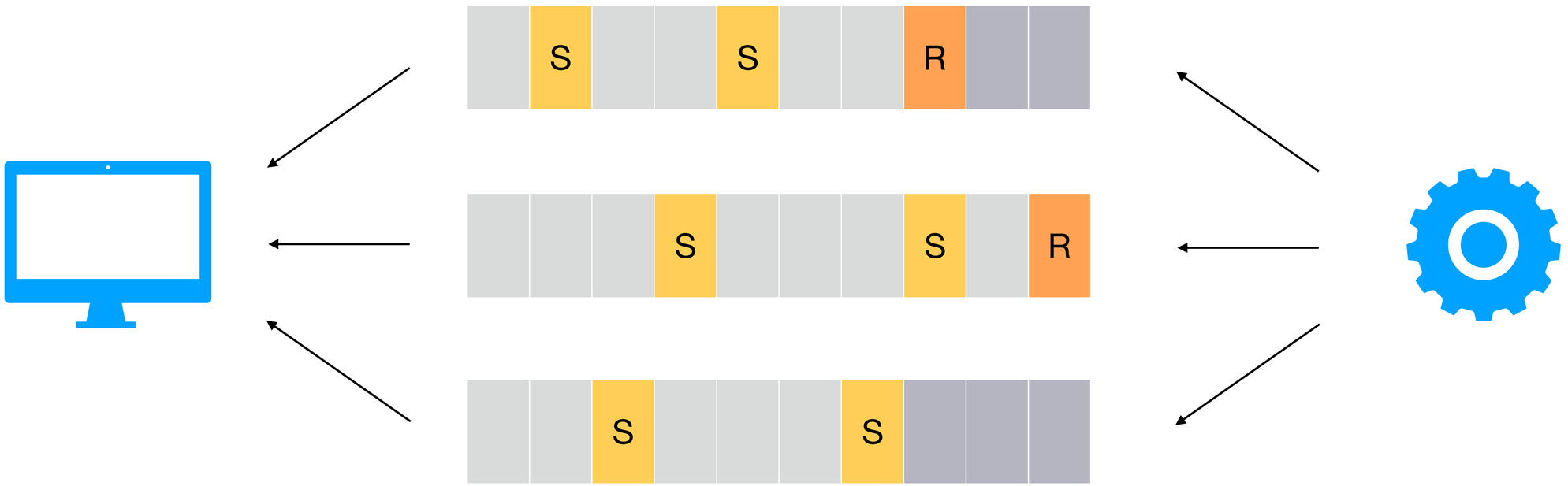

S
S
S
S
S
S
Observer
Output topics
Processor

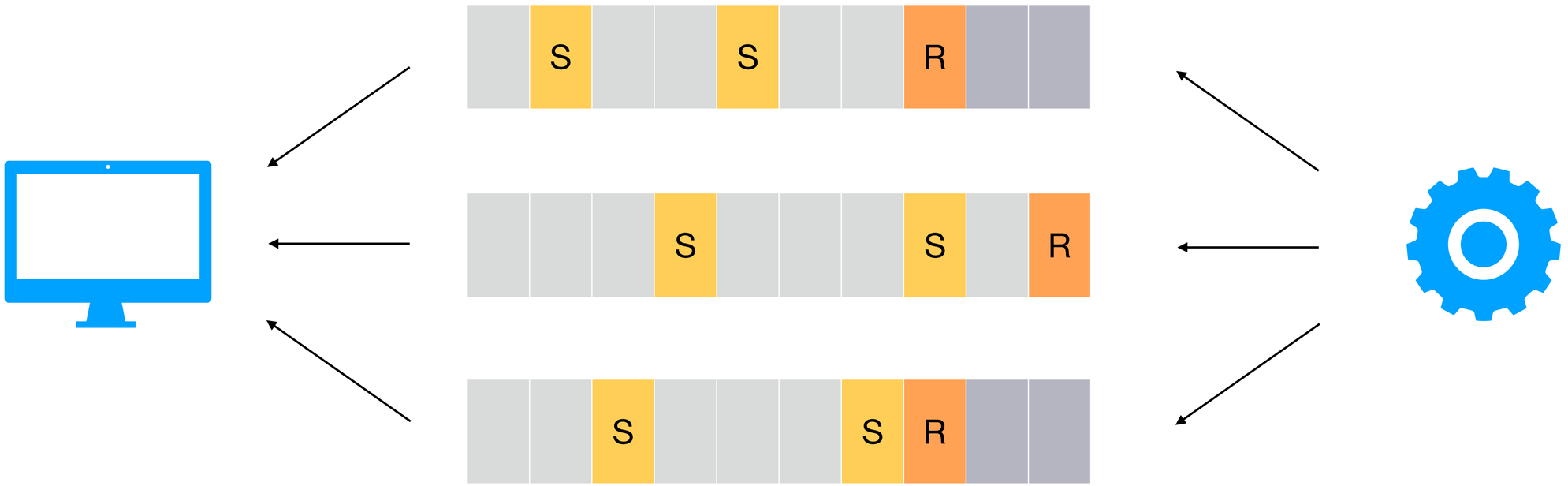

S
S
S
S
S
S
Observer
Output topics
Processor


Hmm...
S
S
S
S
S
S
Observer
Output topics
Processor


Restart!
S
S
S
S
S
S
Observer
Output topics
Processor

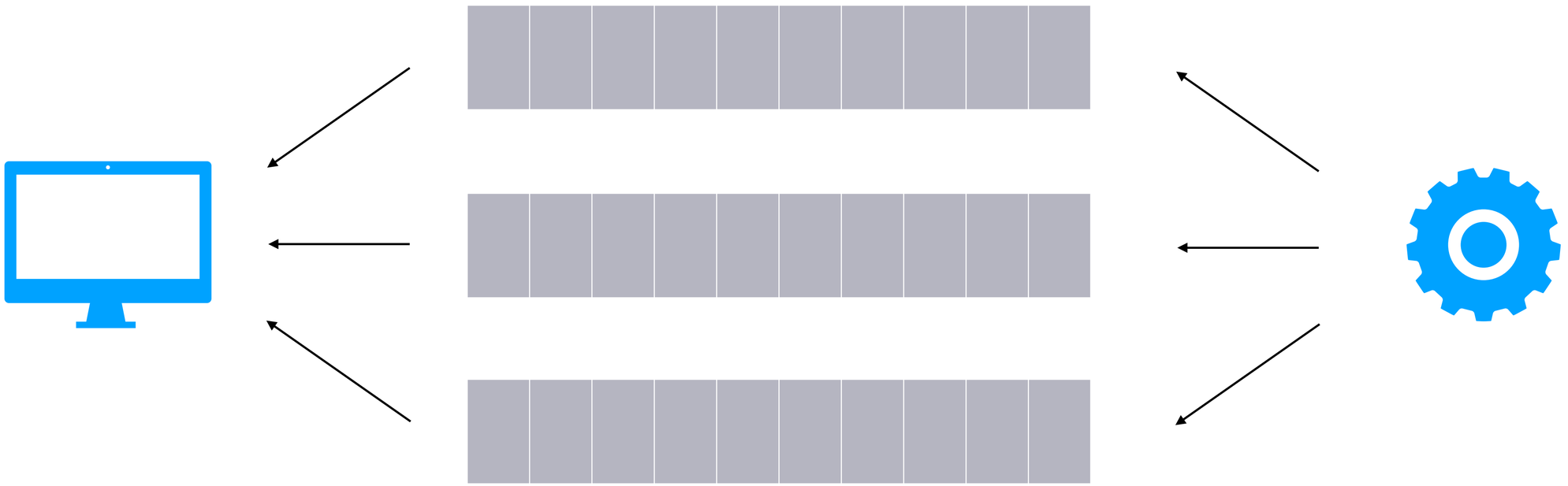

S
S
R
S
S
S
S
Observer
Output topics
Processor


S
S
R
S
S
R
S
S
Observer
Output topics
Processor


S
S
R
S
S
R
S
S
R
Observer
Output topics
Processor

Транзакции в Кафке

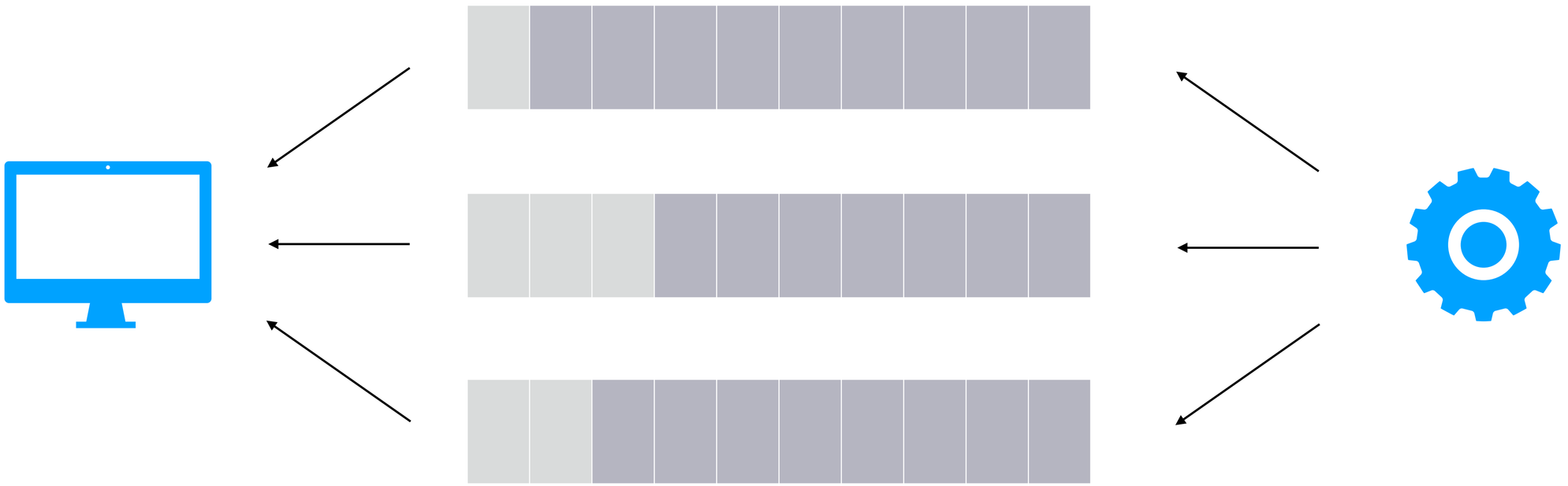
- Два типа маркеров COMMIT and ABORT
- Мягкие (Soft) типы сбоев обрабатываются записью ABORT маркера во все партиции



Observer

Output

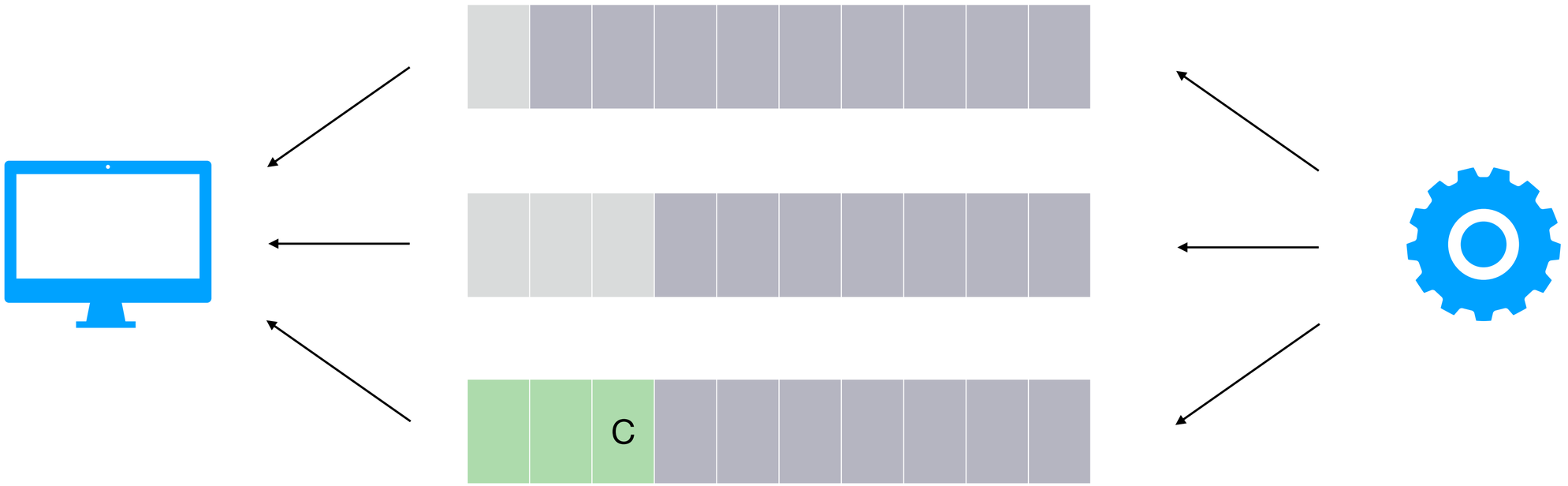
Processor



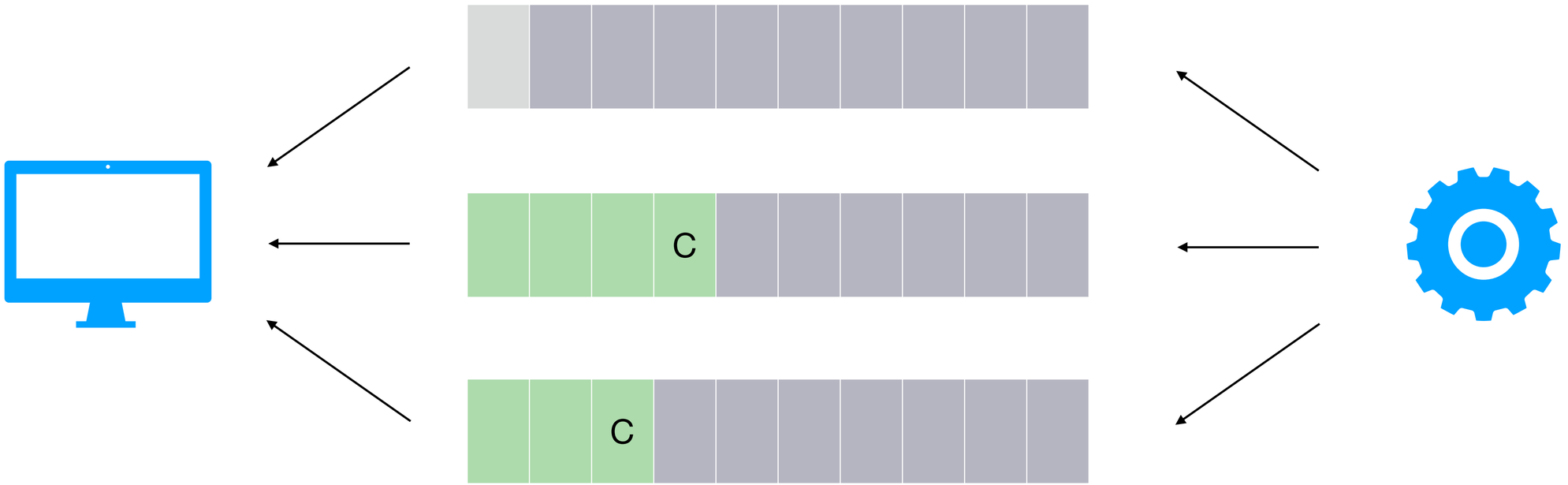
Observer

Output

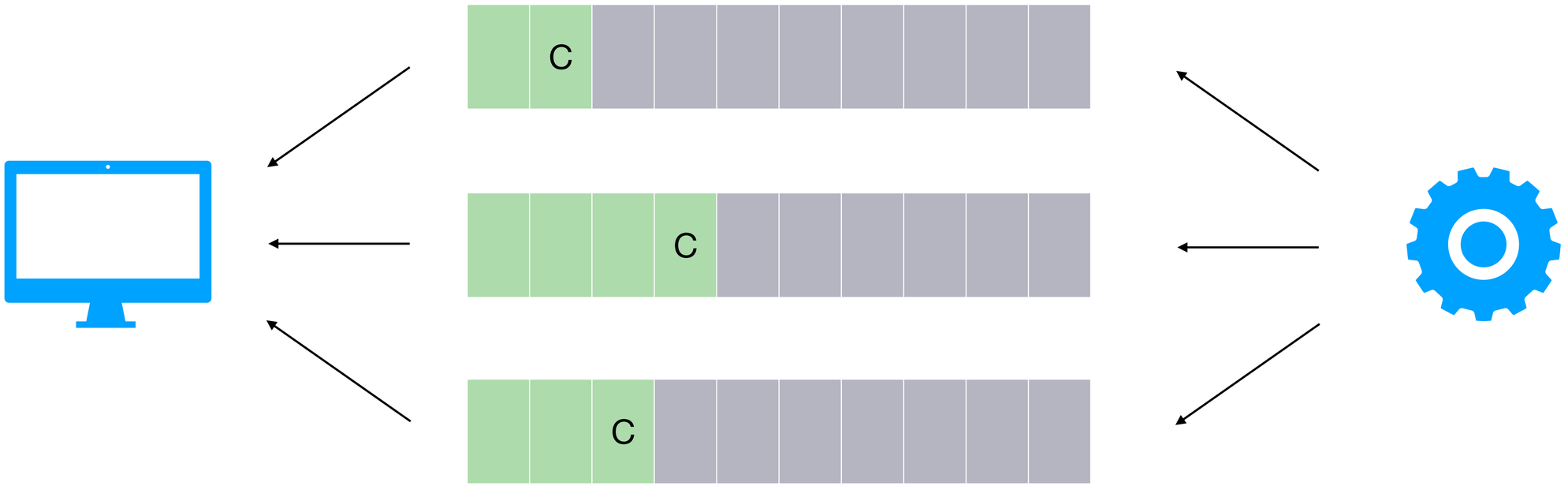
Processor

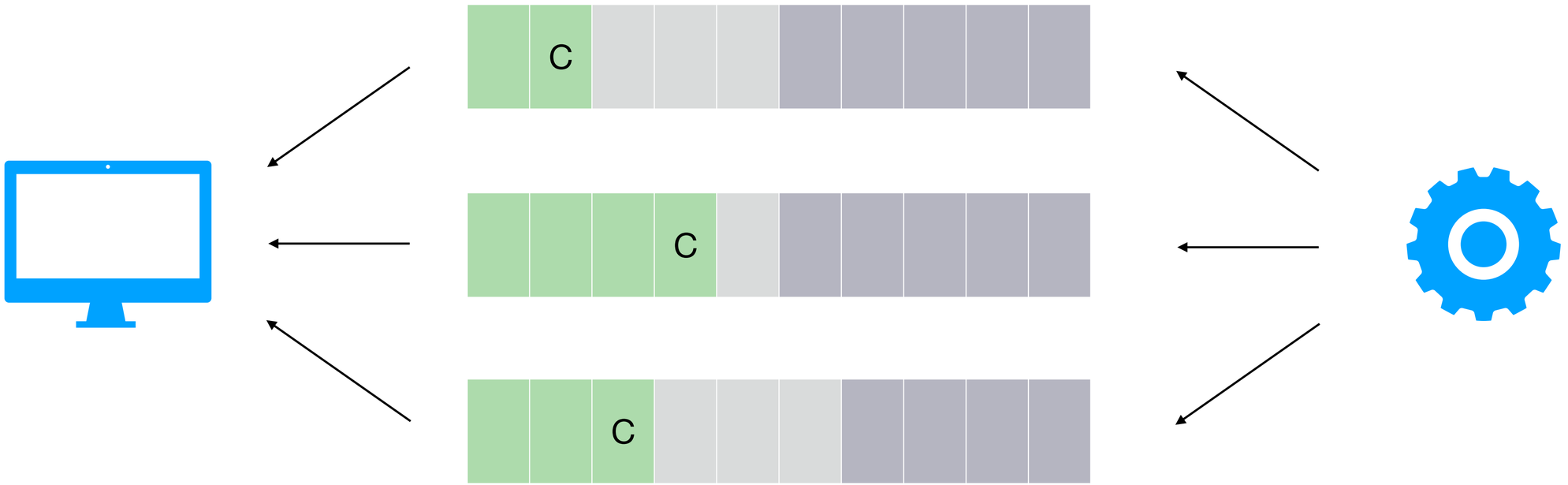


Observer **Output** **Processor**

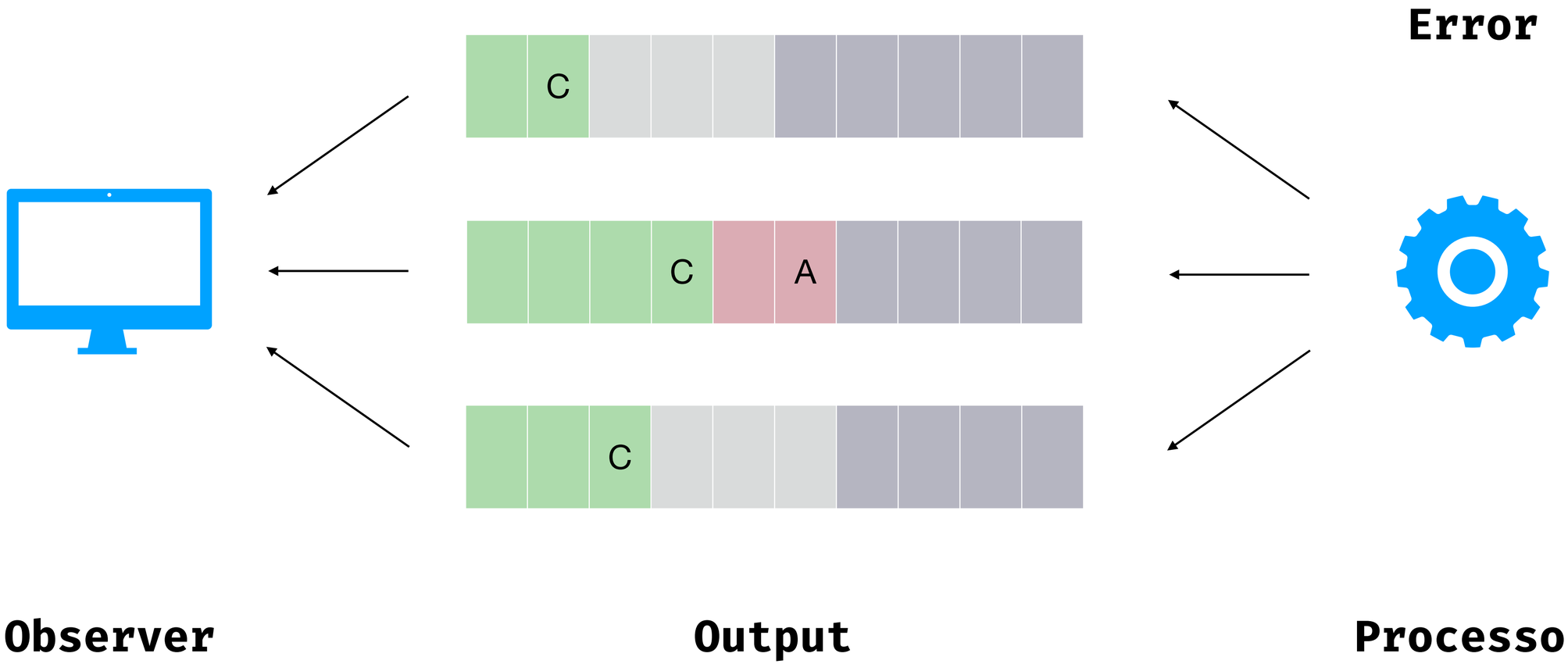


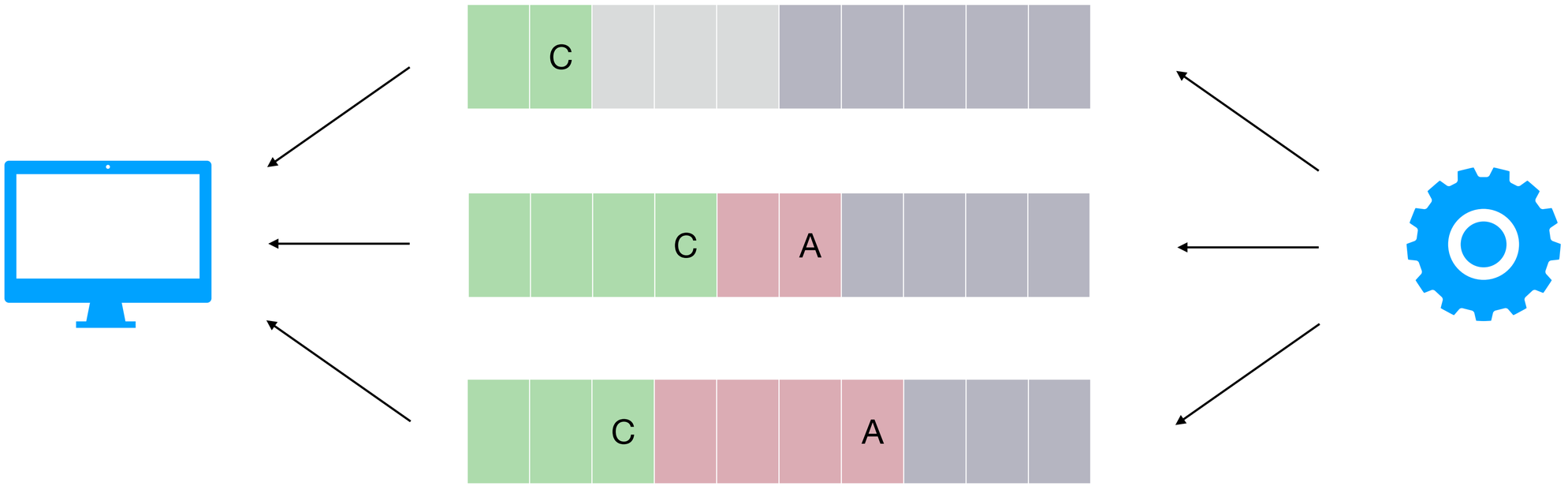
Observer **Output** **Processor**


C
C
C
Observer
Output
Processor

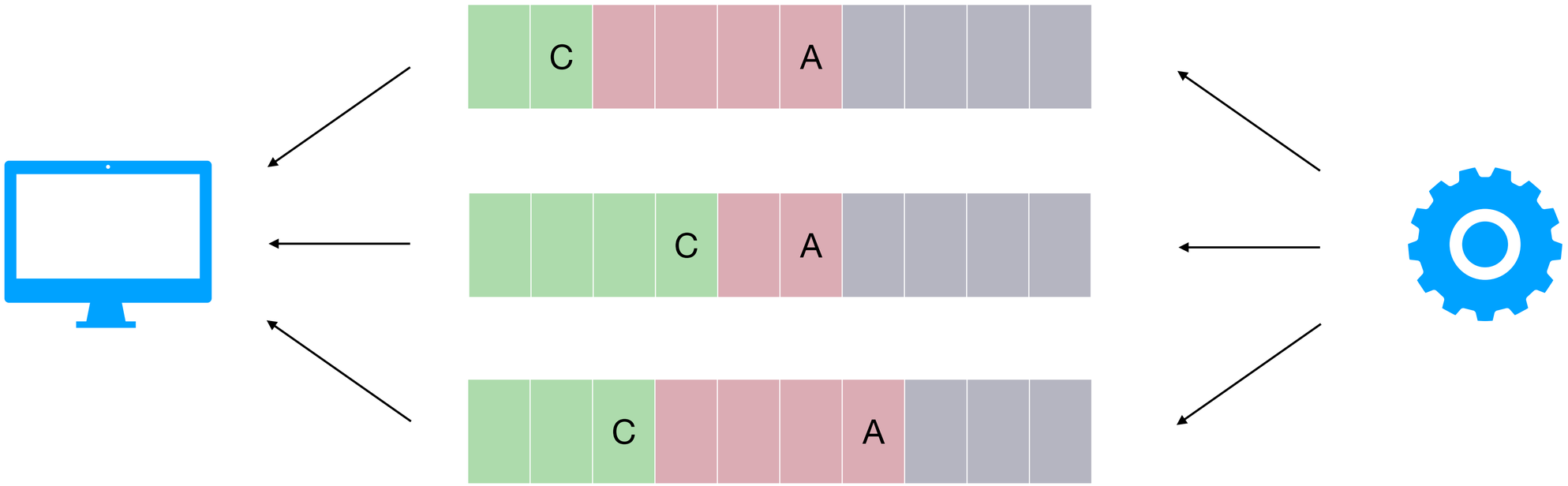


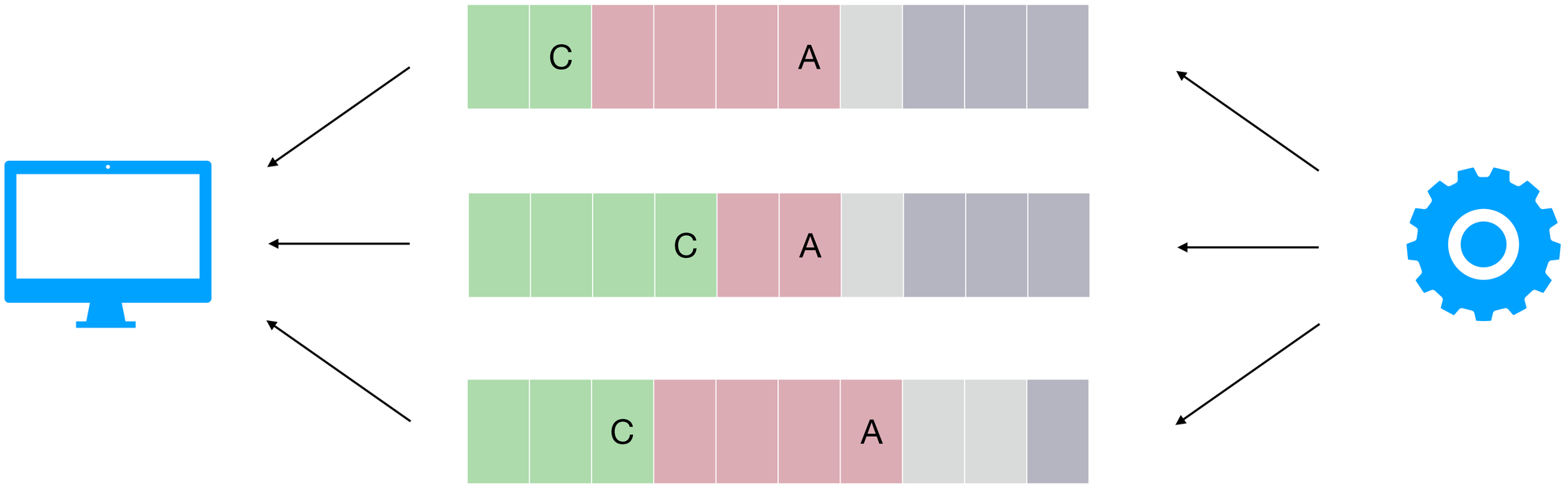
Observer **Output** **Processor**


Error
C
C
A
C
Observer
Output
Processor

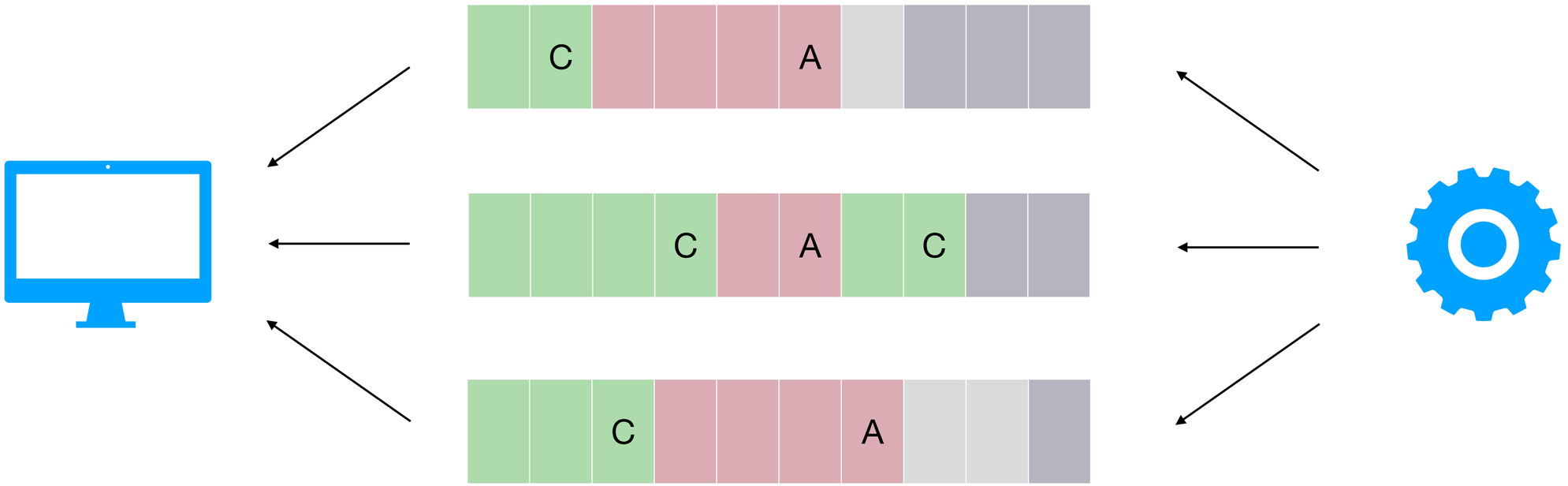


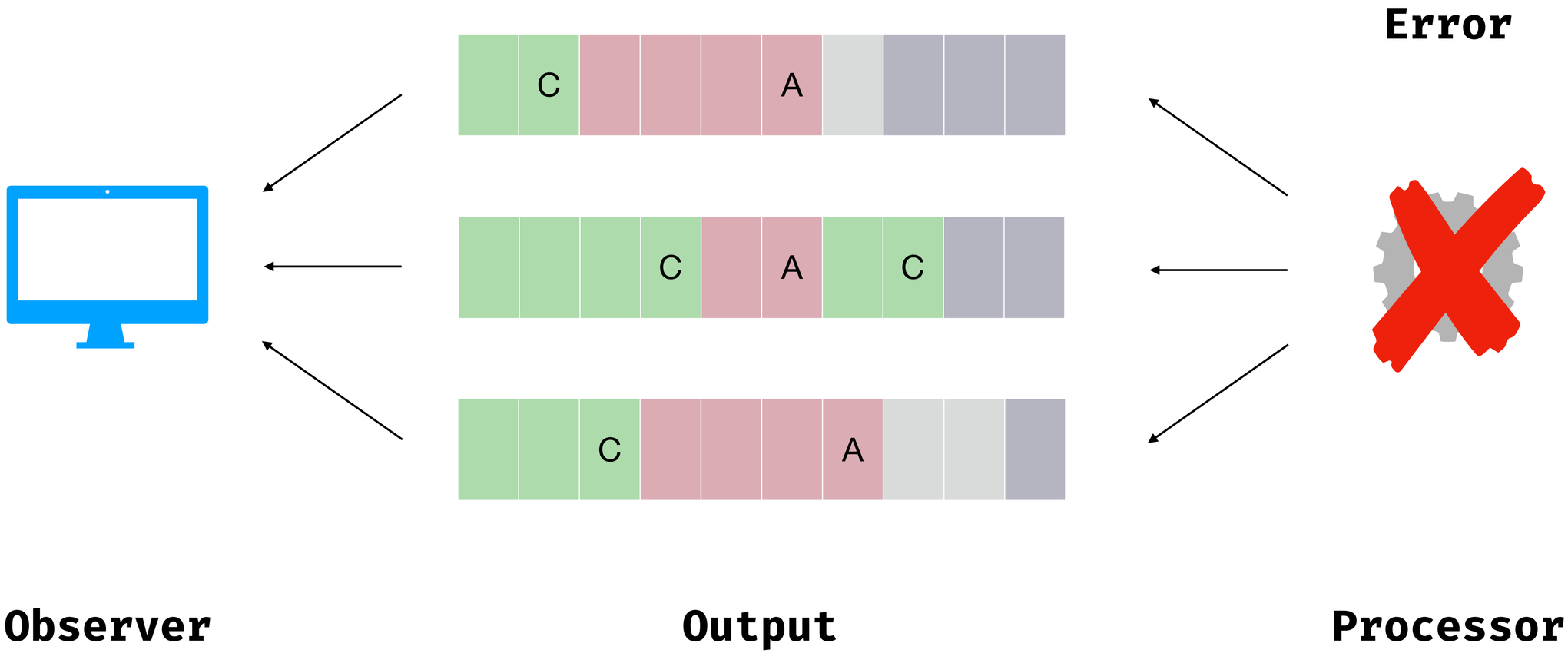
Observer **Output** **Processor**

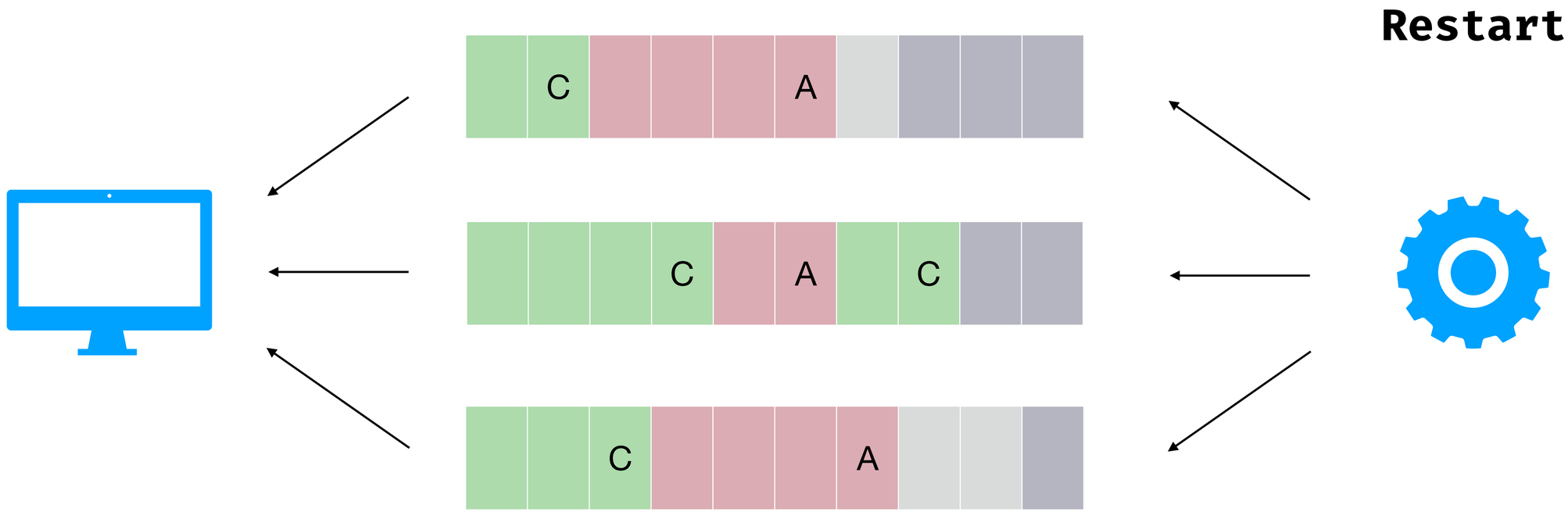

C
A
C
A
C
A
Observer
Output
Processor

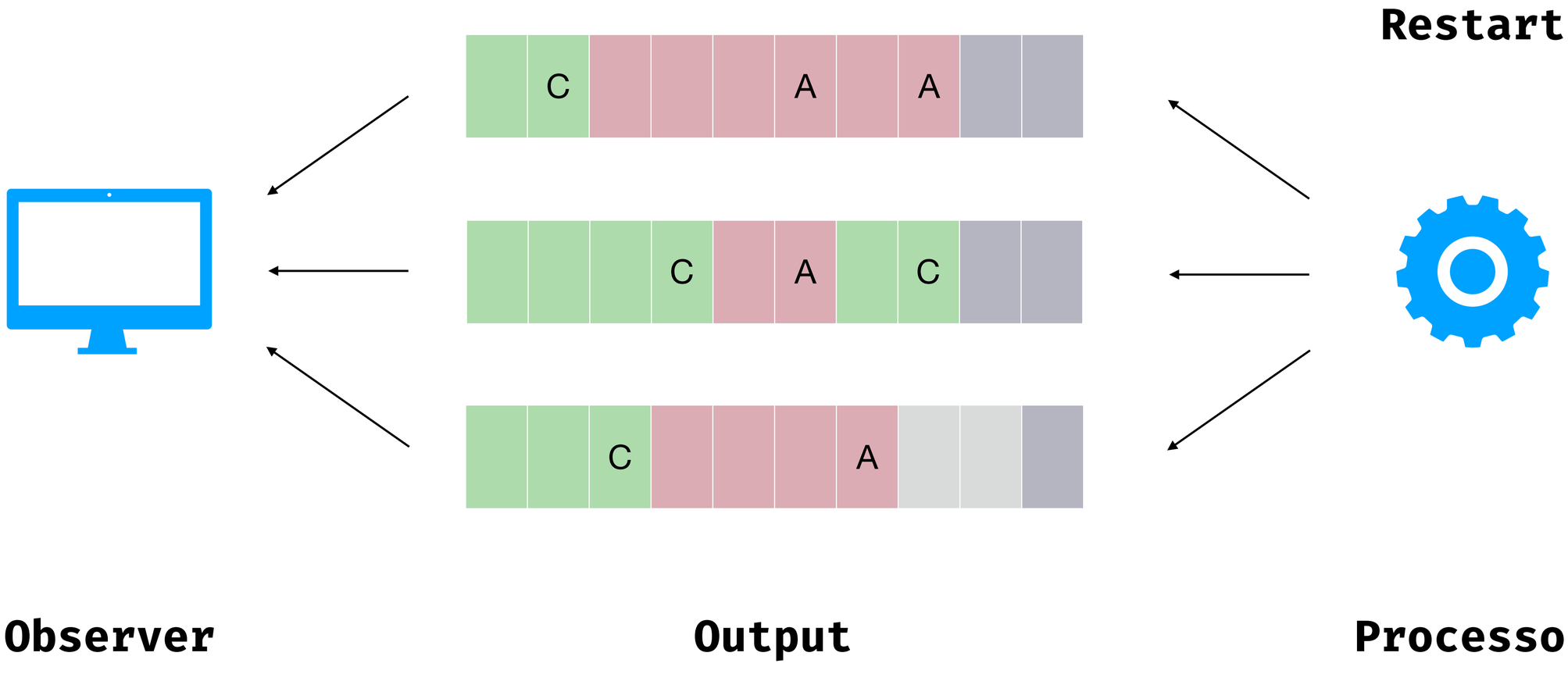


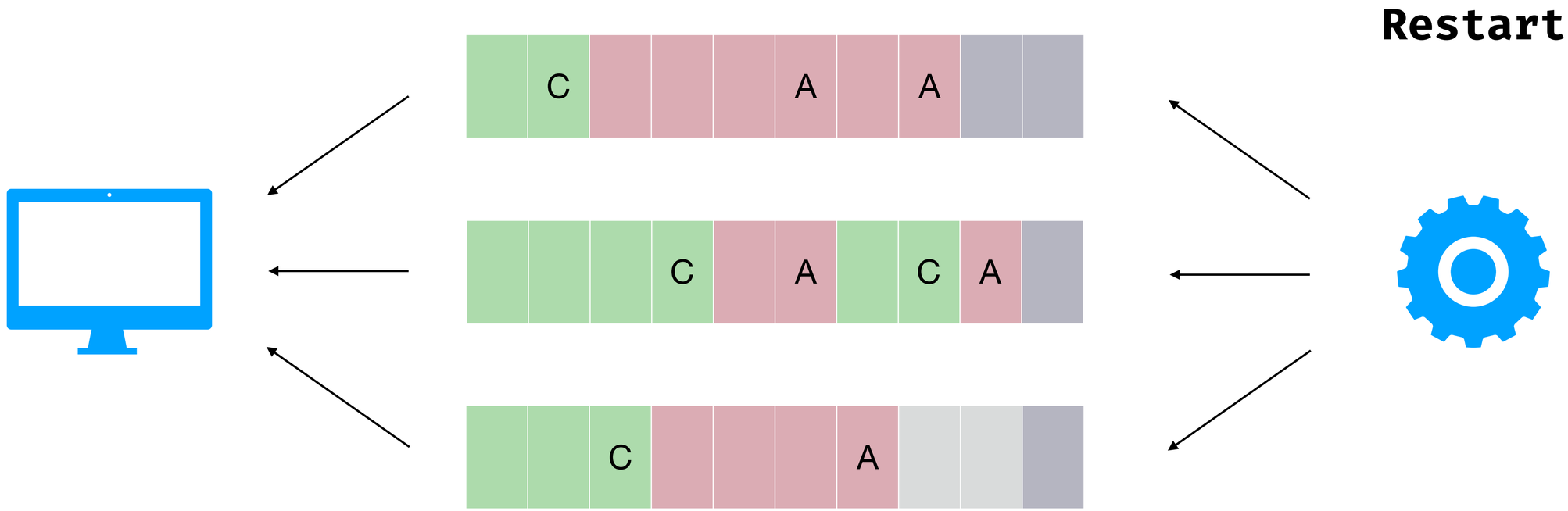
Observer **Output** **Processor**

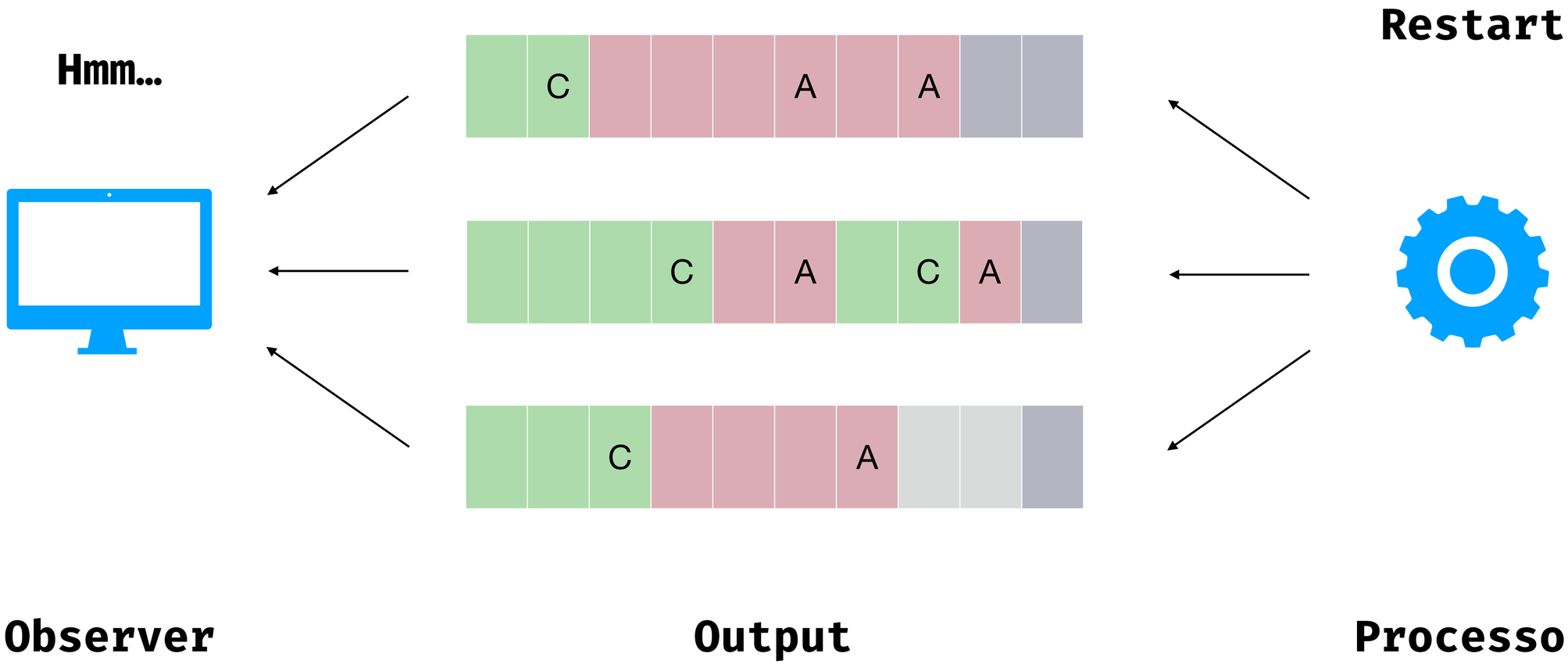

C
A
C
A
C
C
A
Observer
Output
Processor


Error
C
A
C
A
C
C
A
Observer
Output
Processor


Restart
C
A
C
A
C
C
A
Observer
Output
Processor

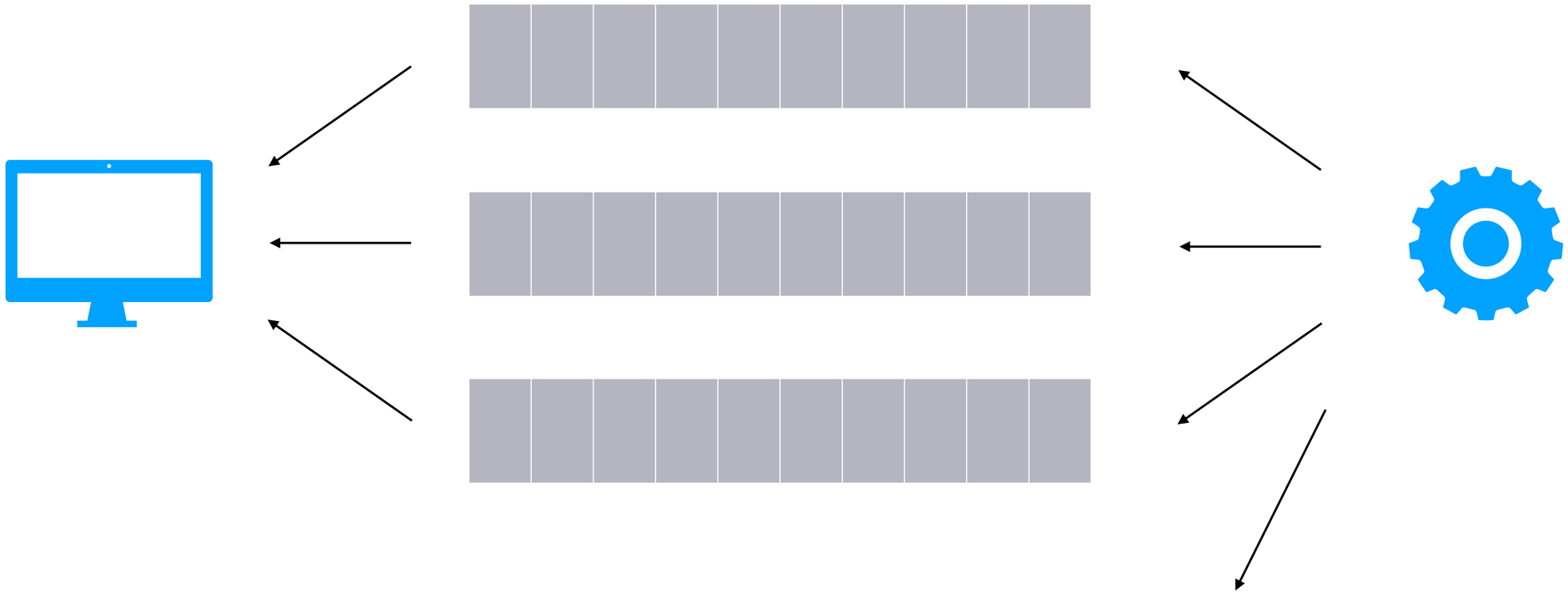

Restart
C
A
A
C
A
C
C
A
Observer
Output
Processor


Restart
C
A
A
C
A
C
A
C
A
Observer
Output
Processor

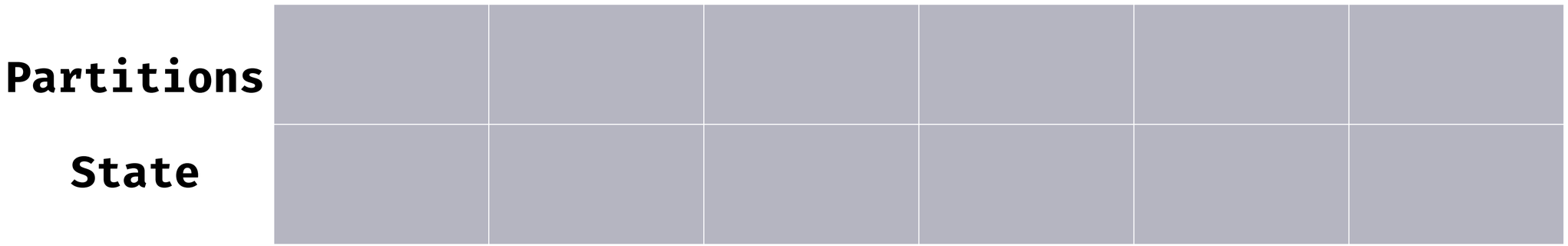

Hmm...
C
A
A
C
A
C
A
C
A
Restart
Observer
Output
Processor

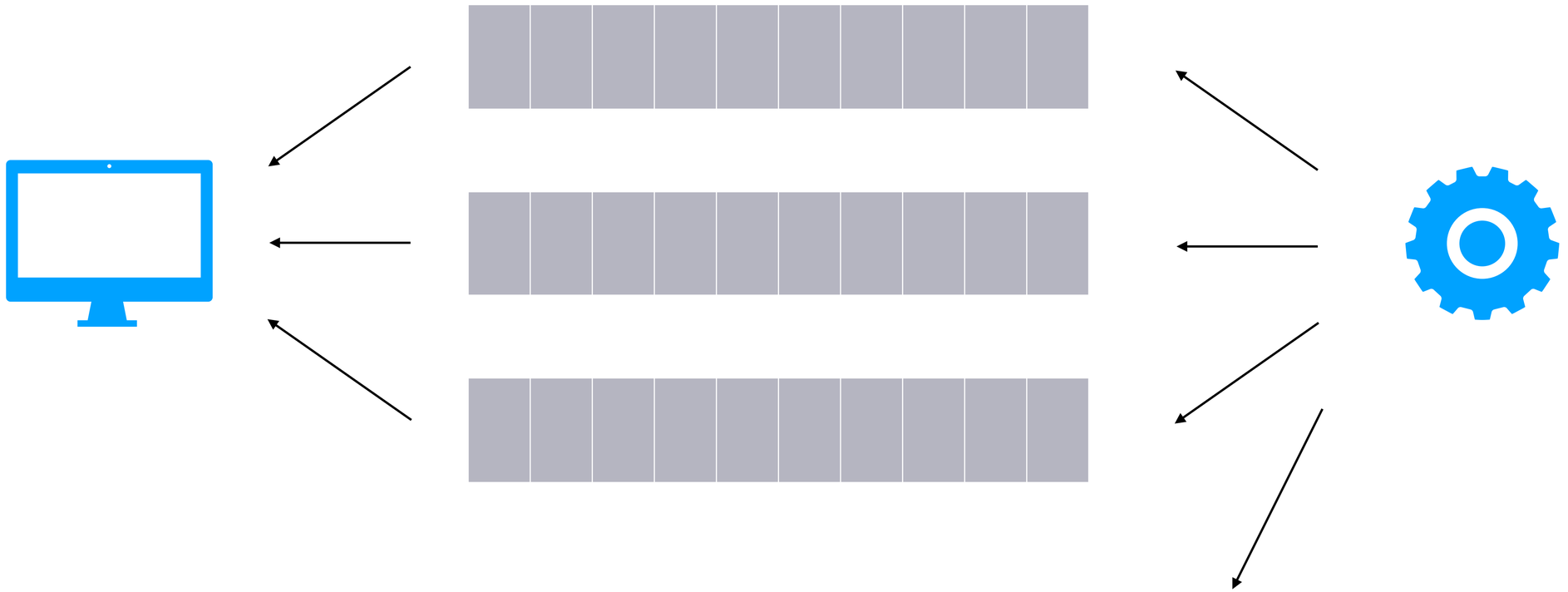
Транзакции

- 2-х фазный коммит и транзакционный лог
  - Запить «желания коммита» в лог
  - Запись маркеров в топики
  - Запись маркера о завершении транзакции (commit / abort)



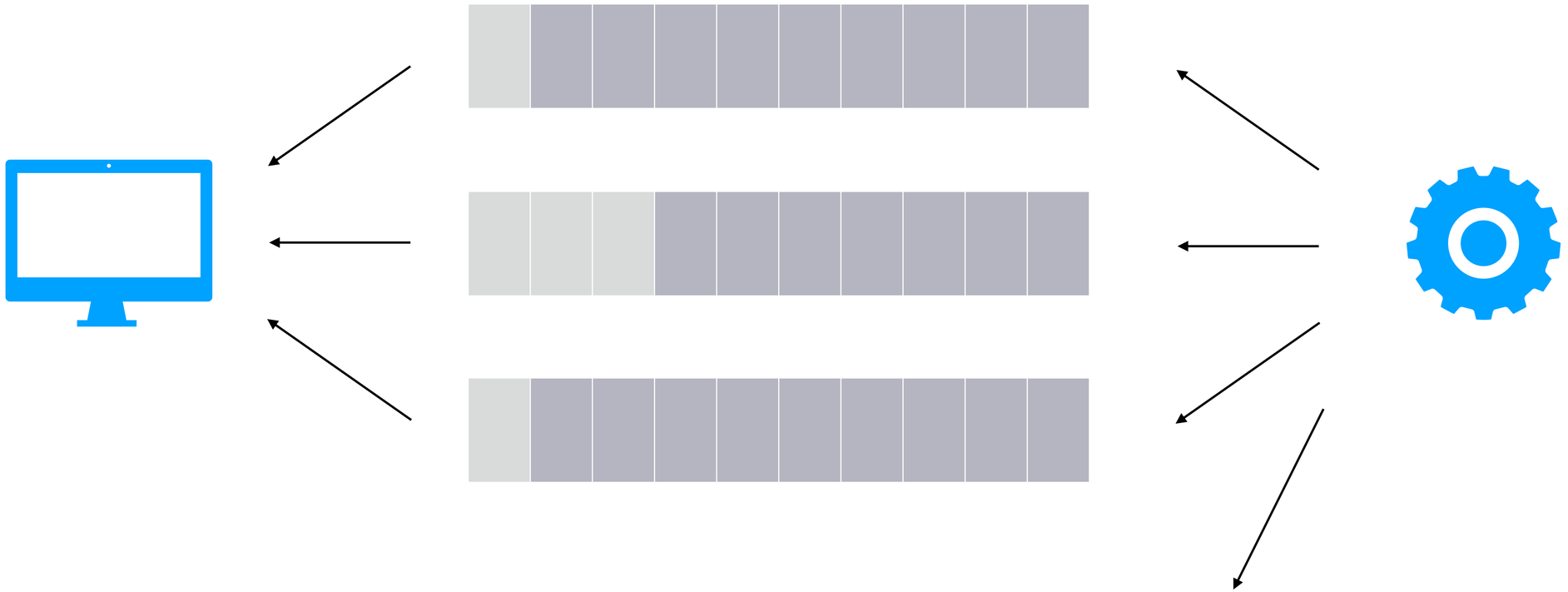
Transaction





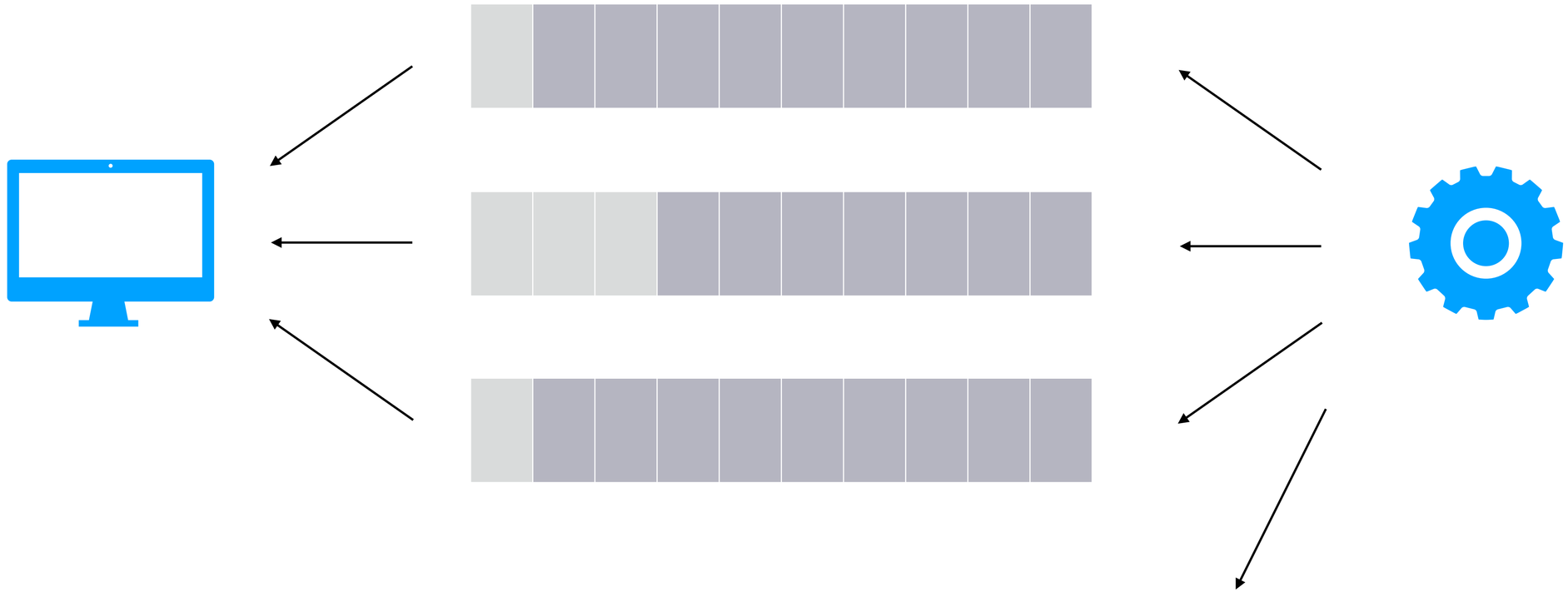
Transaction

Partitions	(0, 1, 2)					
State	ongoing					



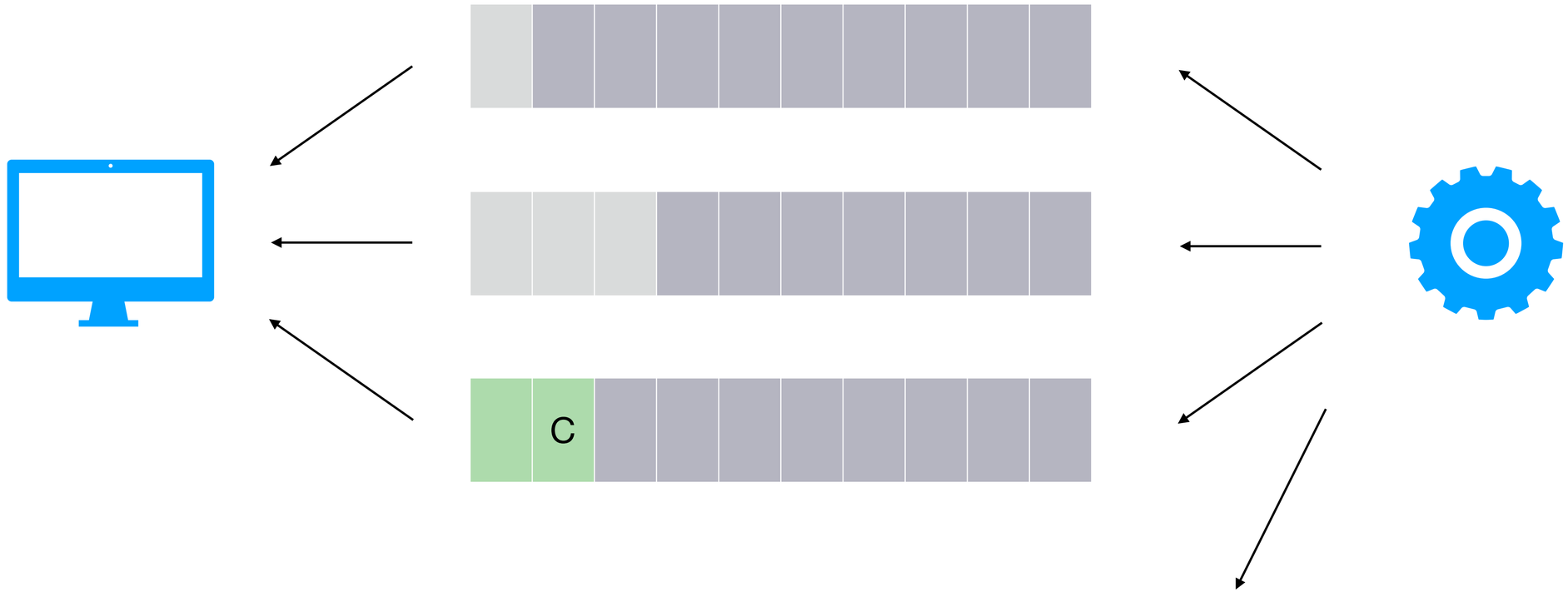
Transaction

Partitions	(0, 1, 2)					
State	ongoing					



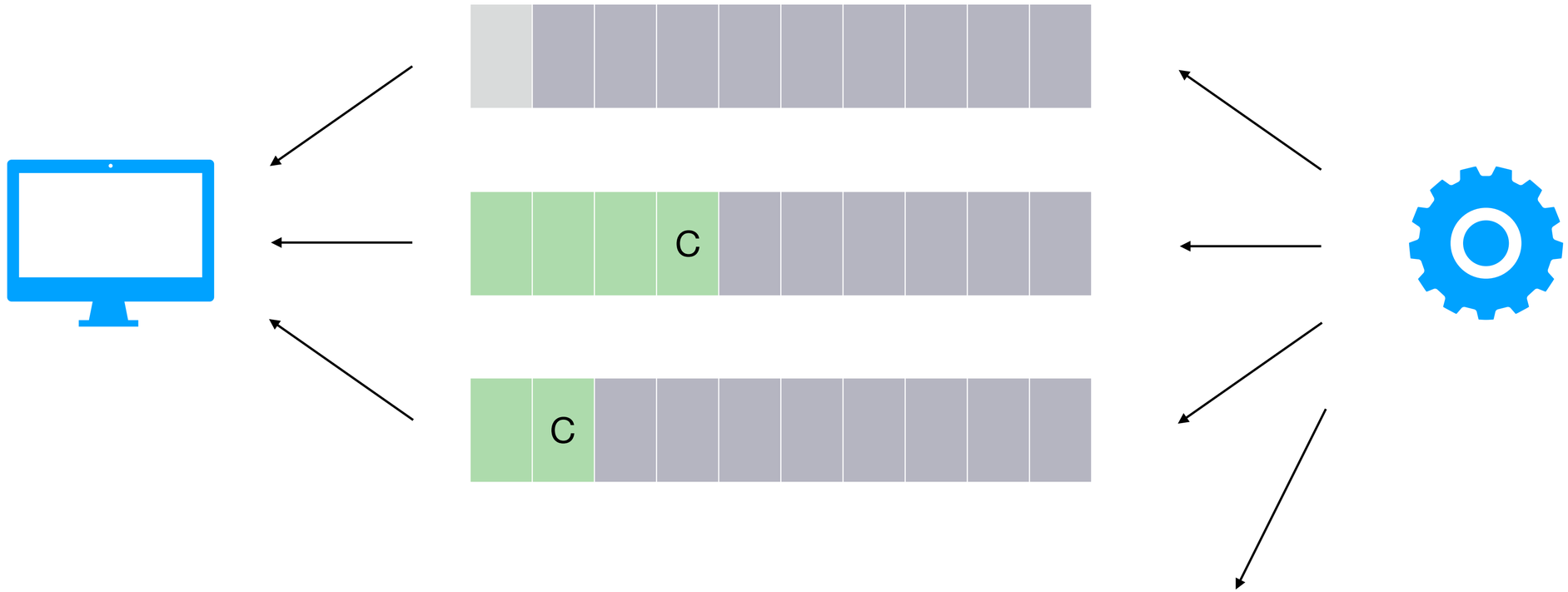
Transaction

Partitions	(0, 1, 2)	(0, 1, 2)				
State	ongoing	prepare commit				



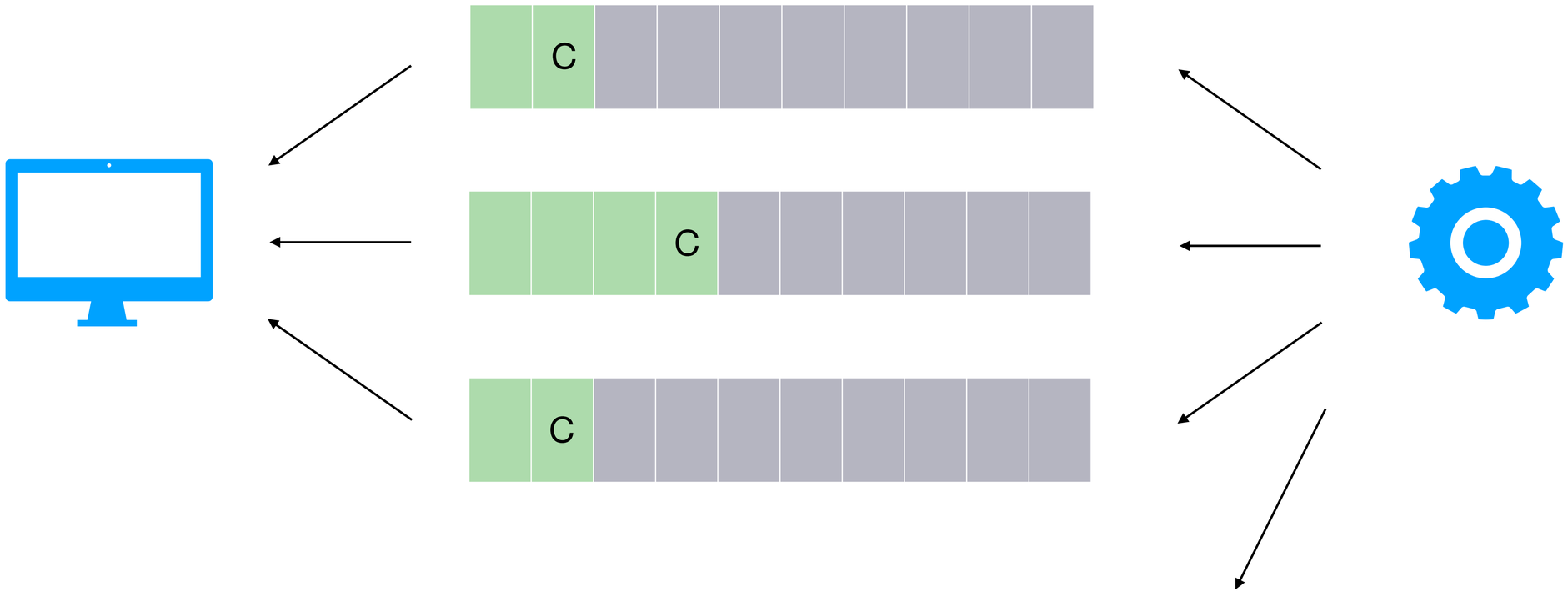
Transaction

Partitions	(0, 1, 2)	(0, 1, 2)				
State	ongoing	prepare commit				



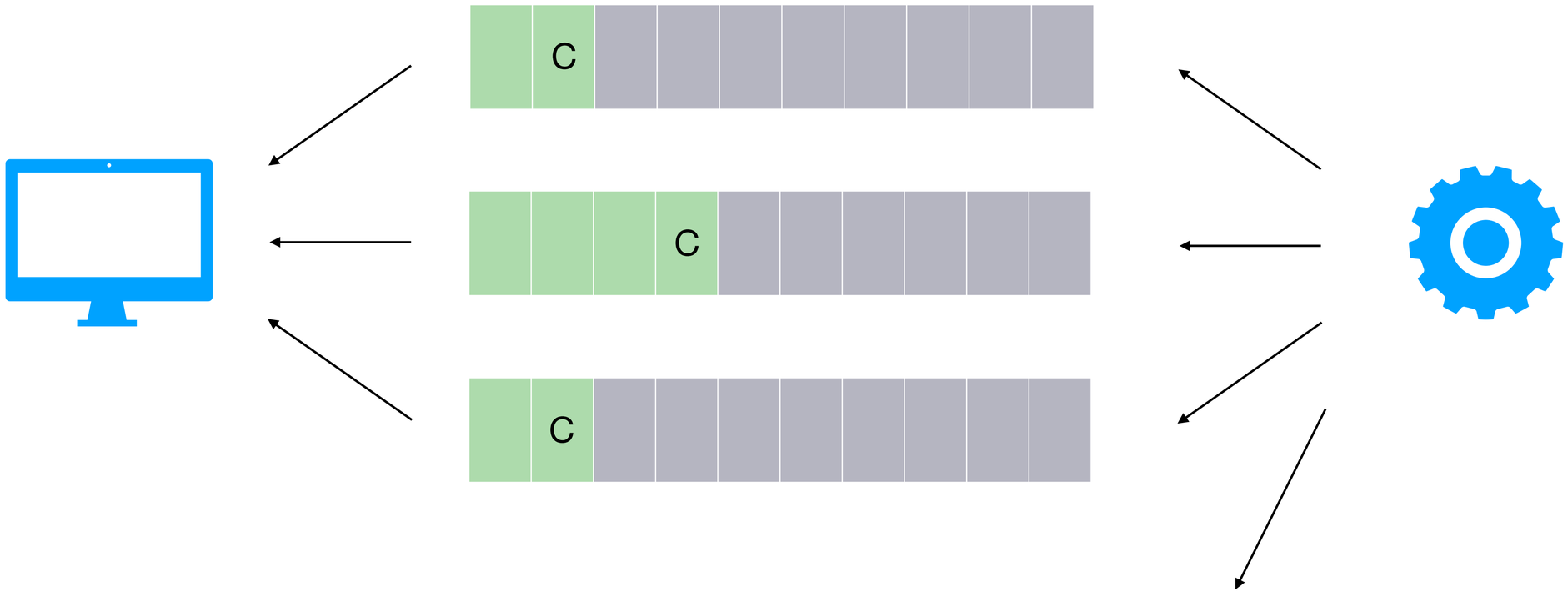
Transaction

Partitions	(0, 1, 2)	(0, 1, 2)				
State	ongoing	prepare commit				



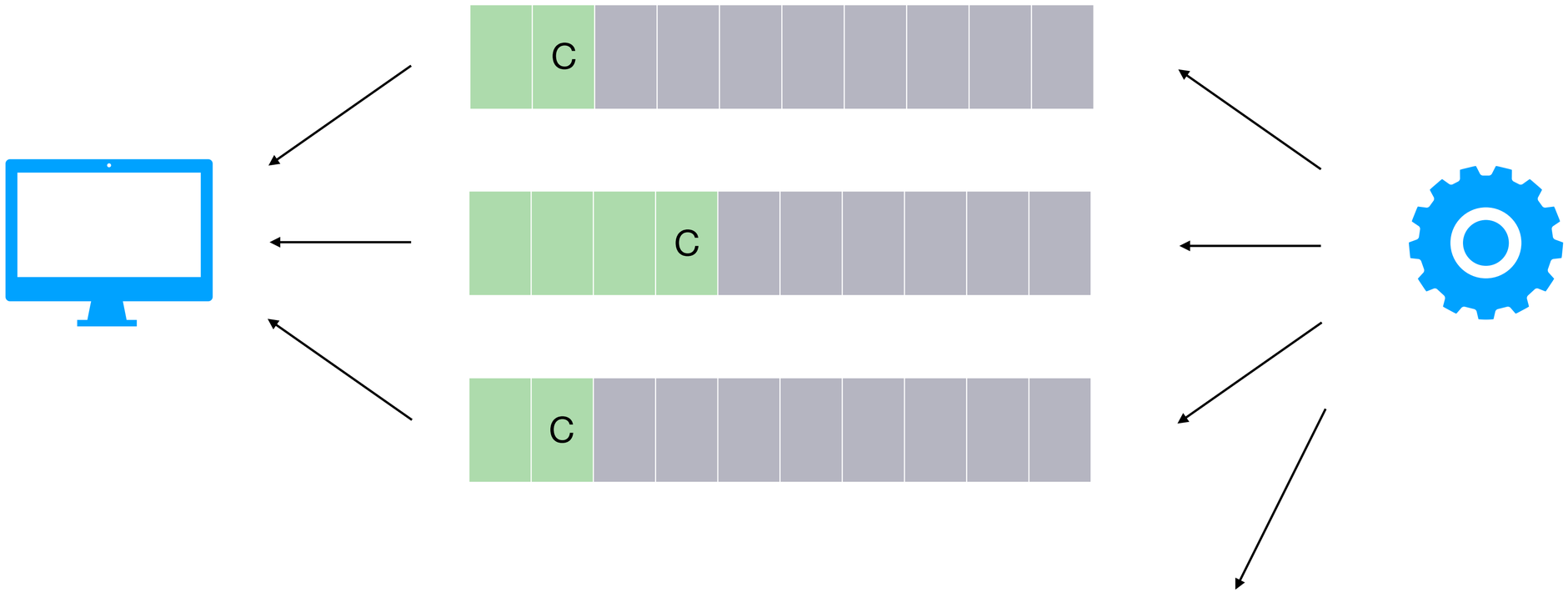
Transaction

Partitions	(0, 1, 2)	(0, 1, 2)				
State	ongoing	prepare commit				



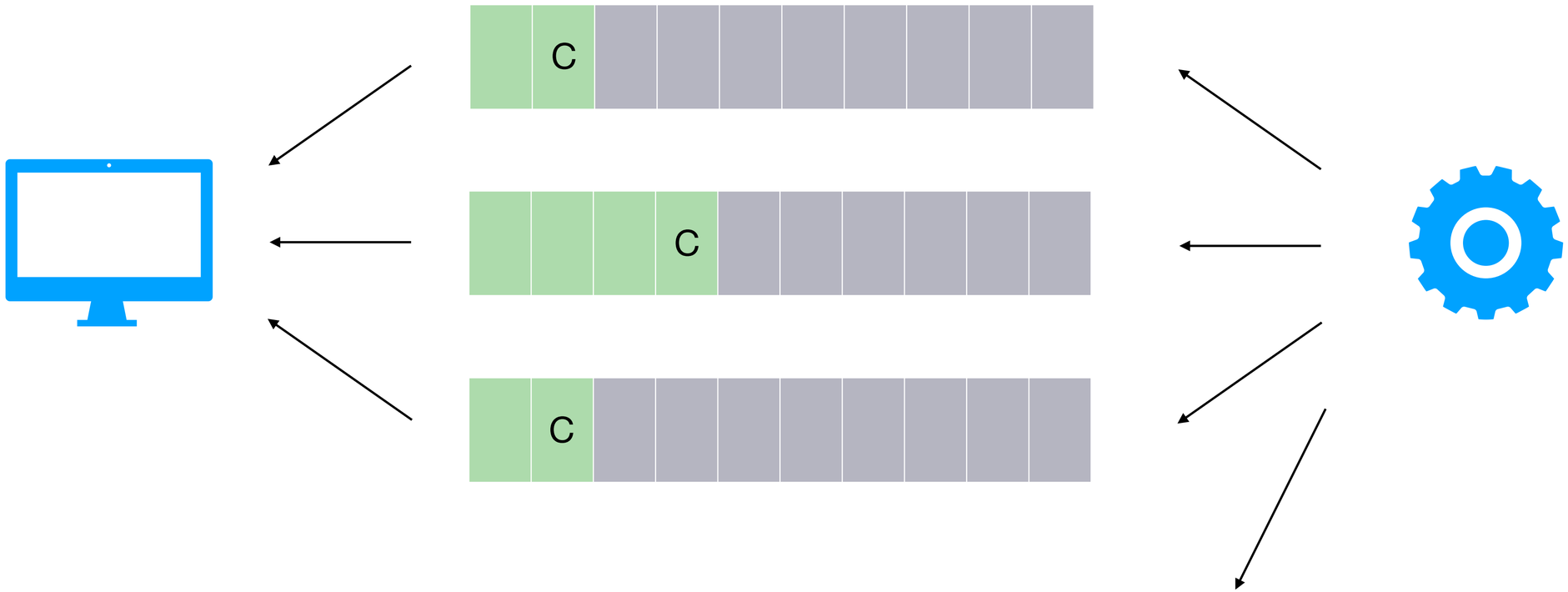
Transaction

Partitions	(0, 1, 2)	(0, 1, 2)	(0, 1, 2)			
State	ongoing	prepare commit	complete commit			



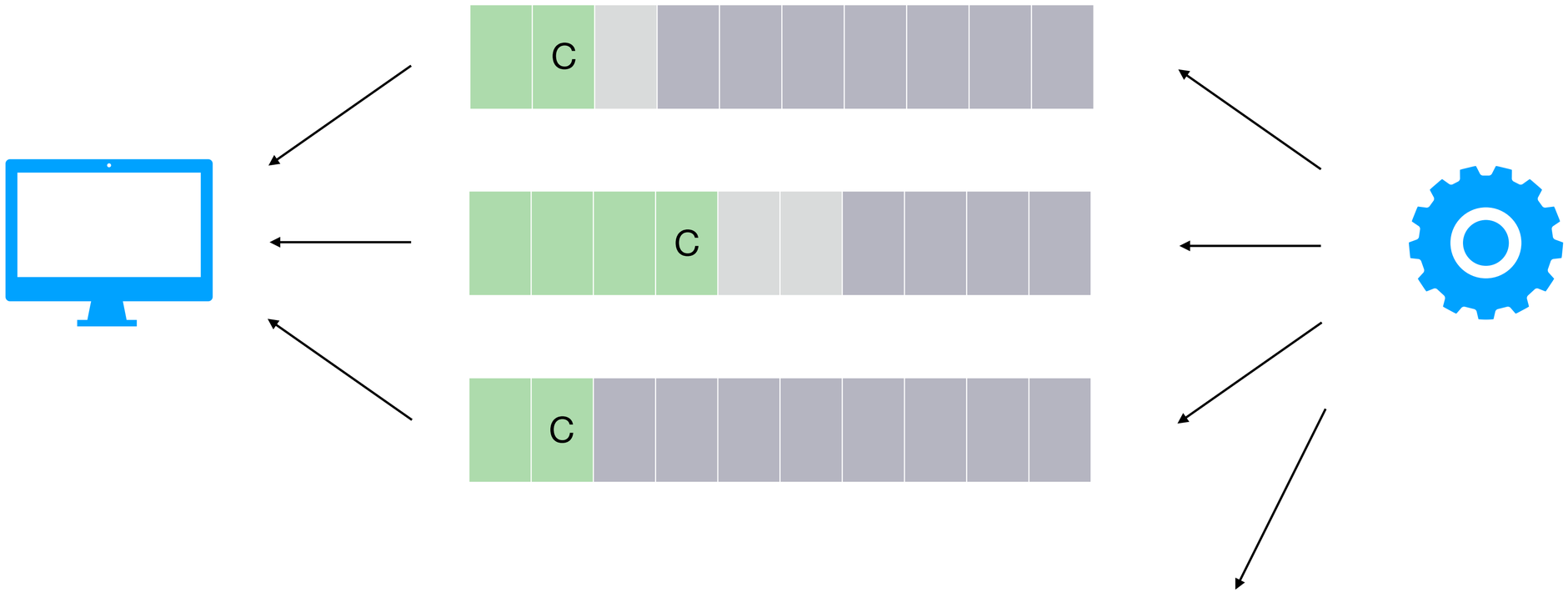
Transaction

Partitions	(0, 1, 2)	(0, 1, 2)	(0, 1, 2)			
State	ongoing	prepare commit	complete commit			



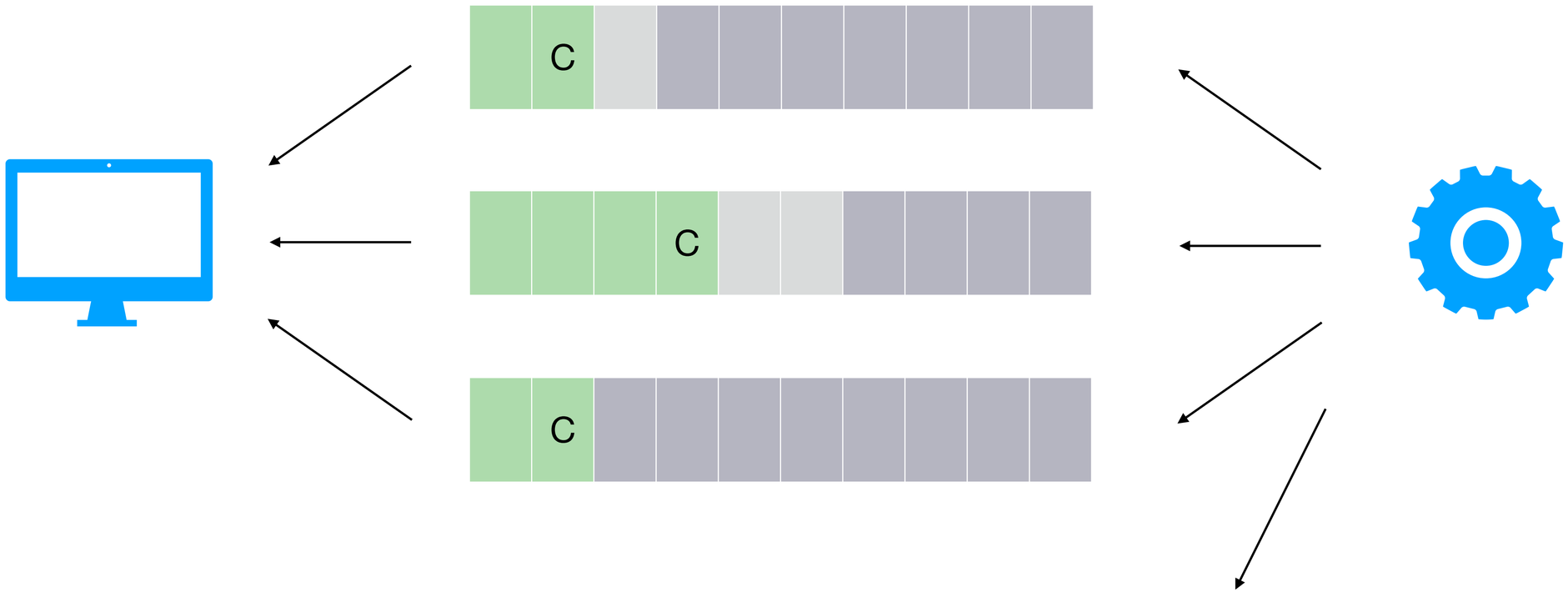
Transaction

Partitions	(0, 1, 2)	(0, 1, 2)	(0, 1, 2)	(0, 1)		
State	ongoing	prepare commit	complete commit	ongoing		



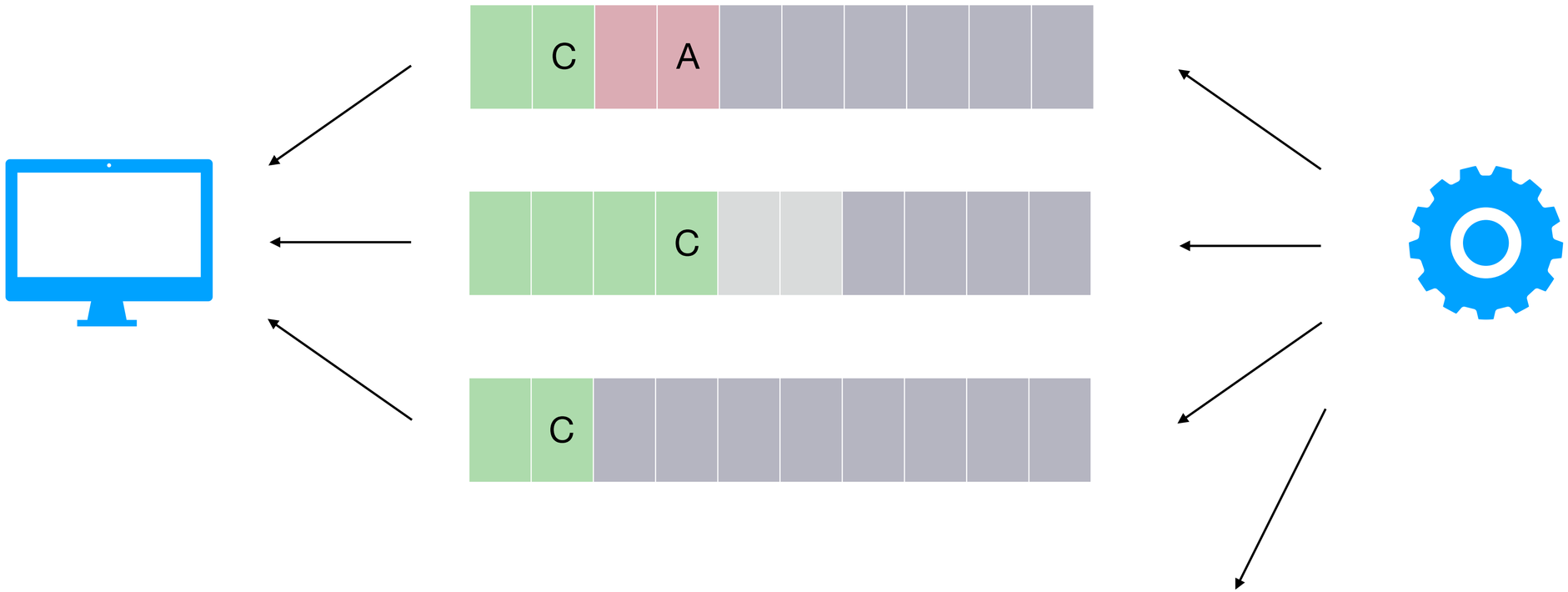
Transaction

Partitions	(0, 1, 2)	(0, 1, 2)	(0, 1, 2)	(0, 1)		
State	ongoing	prepare commit	complete commit	ongoing		



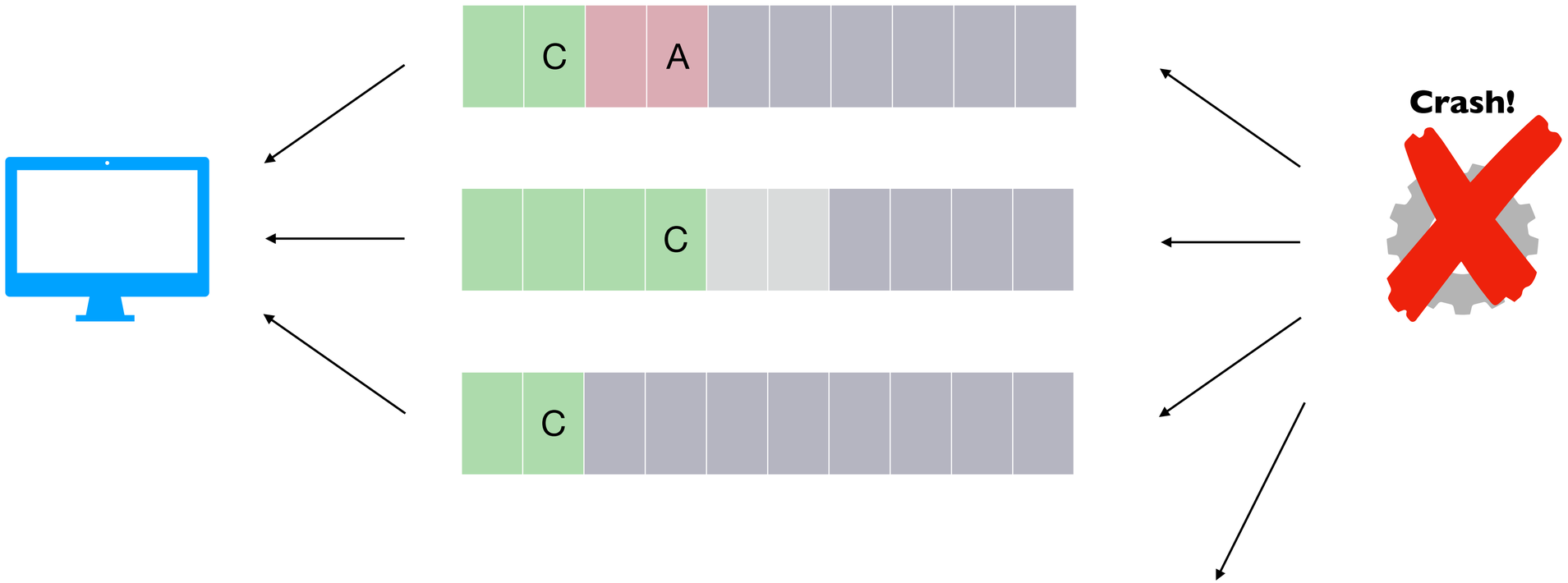
Transaction

Partitions	(0, 1, 2)	(0, 1, 2)	(0, 1, 2)	(0, 1)	(0, 1)	
State	ongoing	prepare commit	complete commit	ongoing	prepare abort	



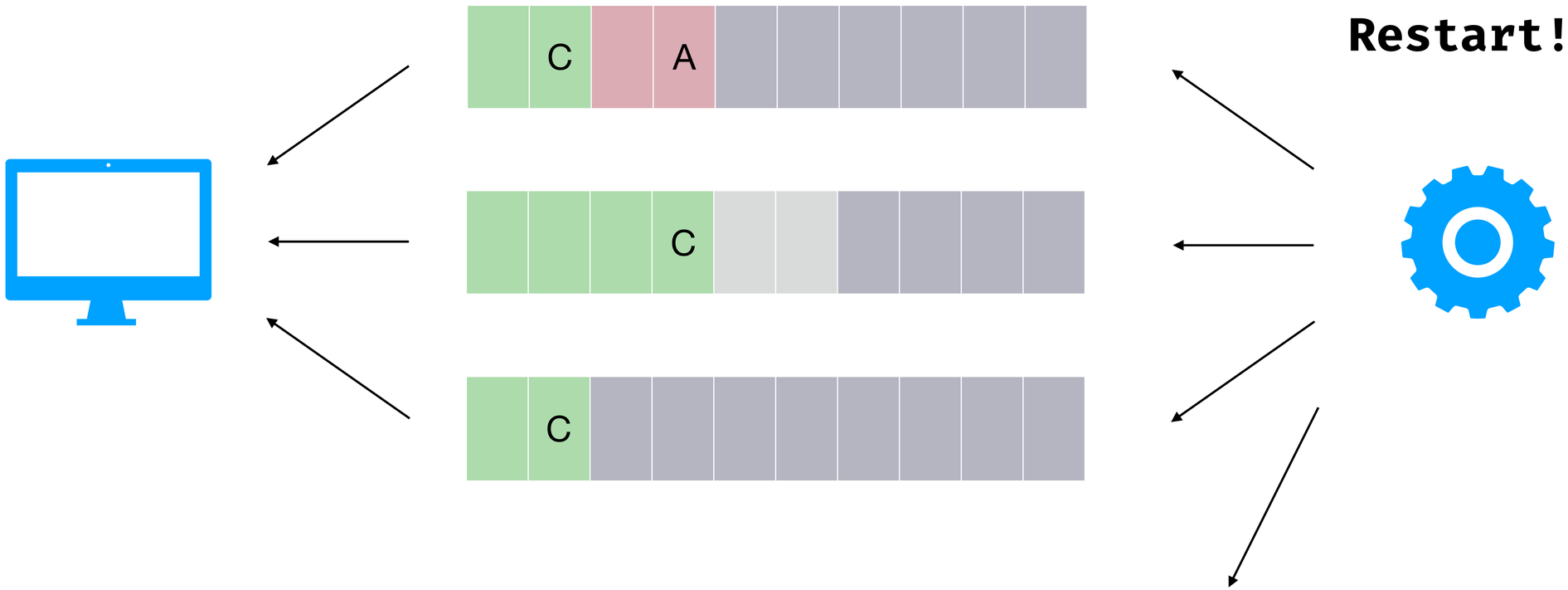
Transaction

Partitions	(0, 1, 2)	(0, 1, 2)	(0, 1, 2)	(0, 1)	(0, 1)	
State	ongoing	prepare commit	complete commit	ongoing	prepare abort	



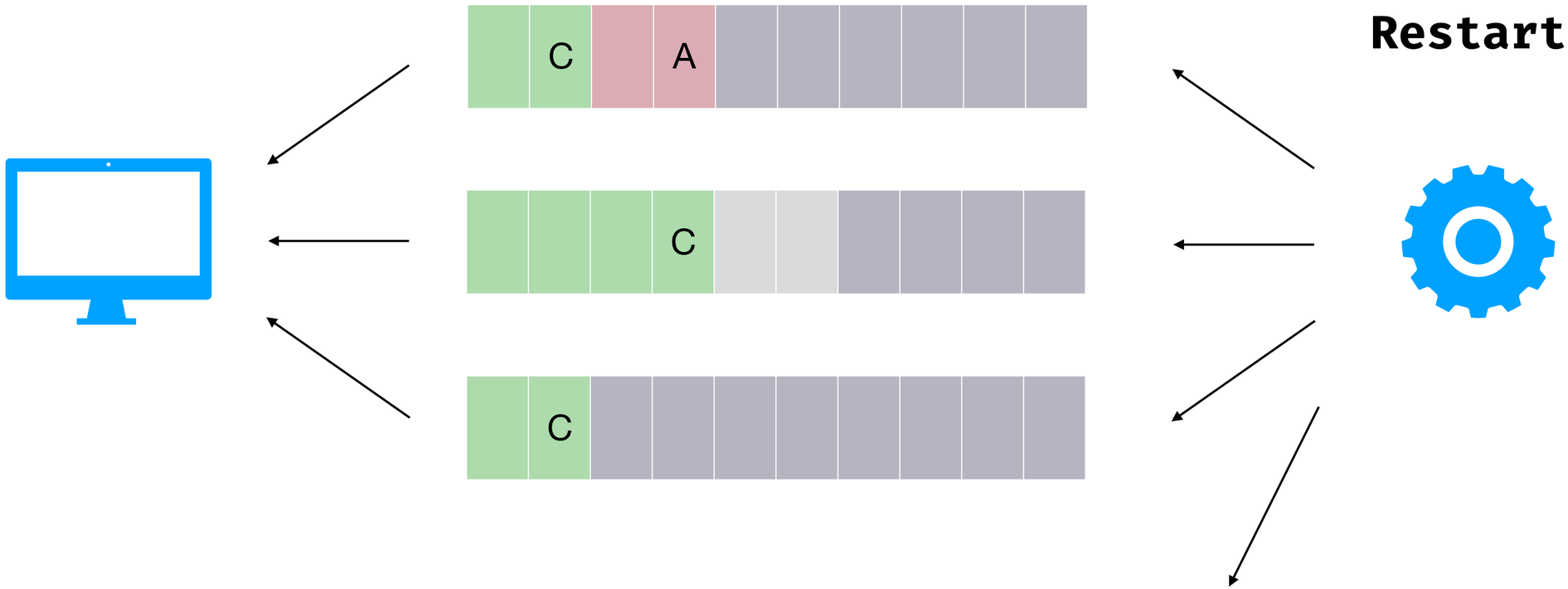
Transaction

Partitions	(0, 1, 2)	(0, 1, 2)	(0, 1, 2)	(0, 1)	(0, 1)	
State	ongoing	prepare commit	complete commit	ongoing	prepare abort	



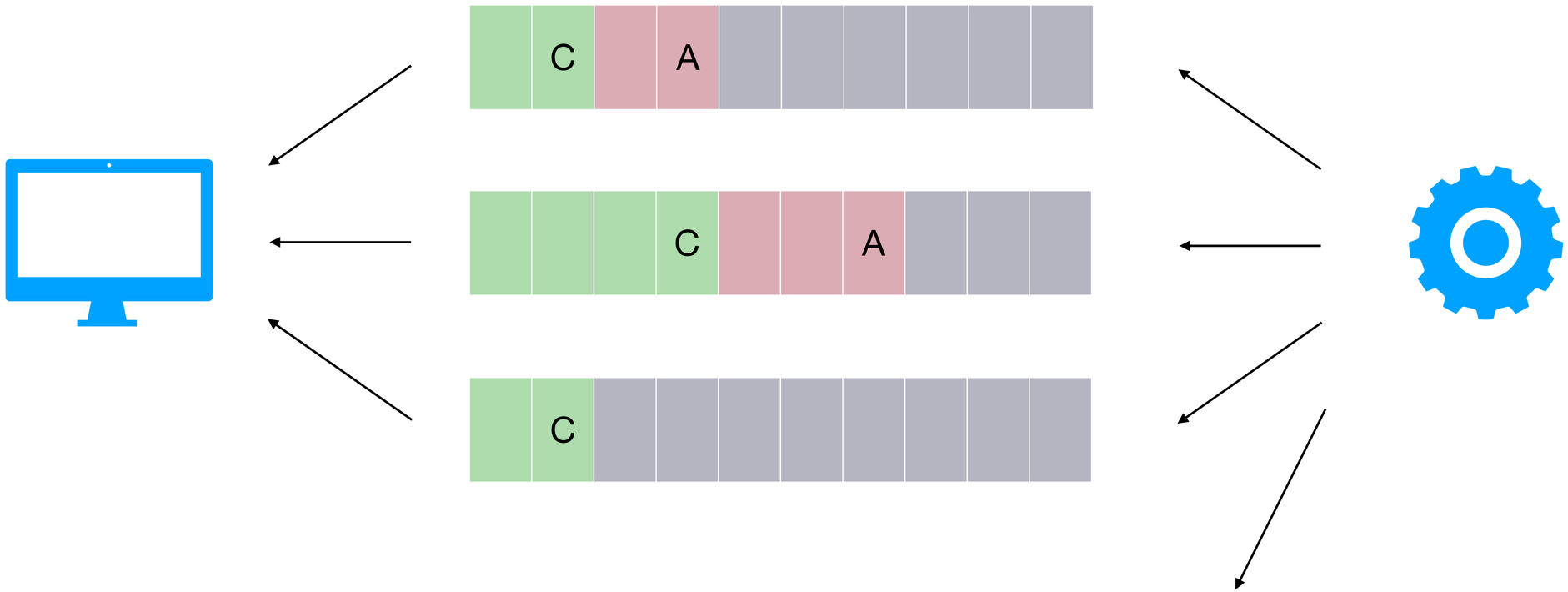
Transaction

Partitions	(0, 1, 2)	(0, 1, 2)	(0, 1, 2)	(0, 1)	(0, 1)	
State	ongoing	prepare commit	complete commit	ongoing	prepare abort	



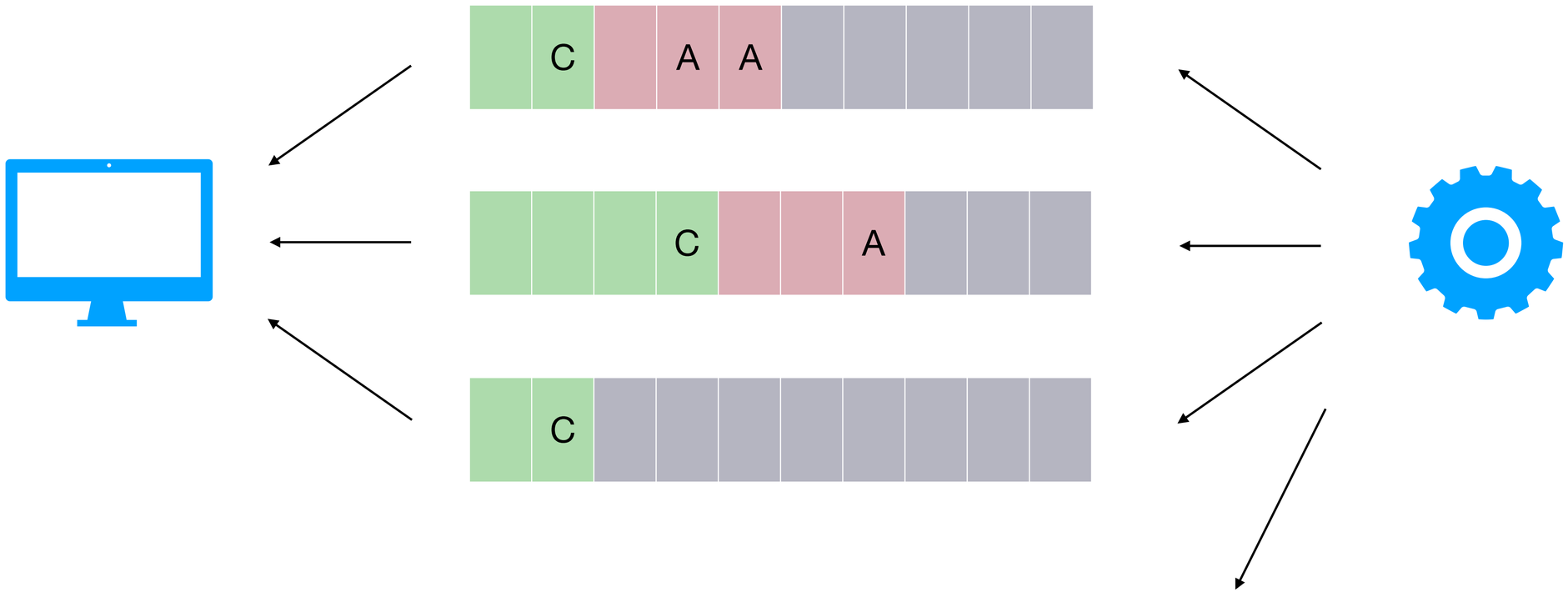
Transaction

Partitions	(0, 1, 2)	(0, 1, 2)	(0, 1, 2)	(0, 1)	(0, 1)	
State	ongoing	prepare commit	complete commit	ongoing	prepare abort	



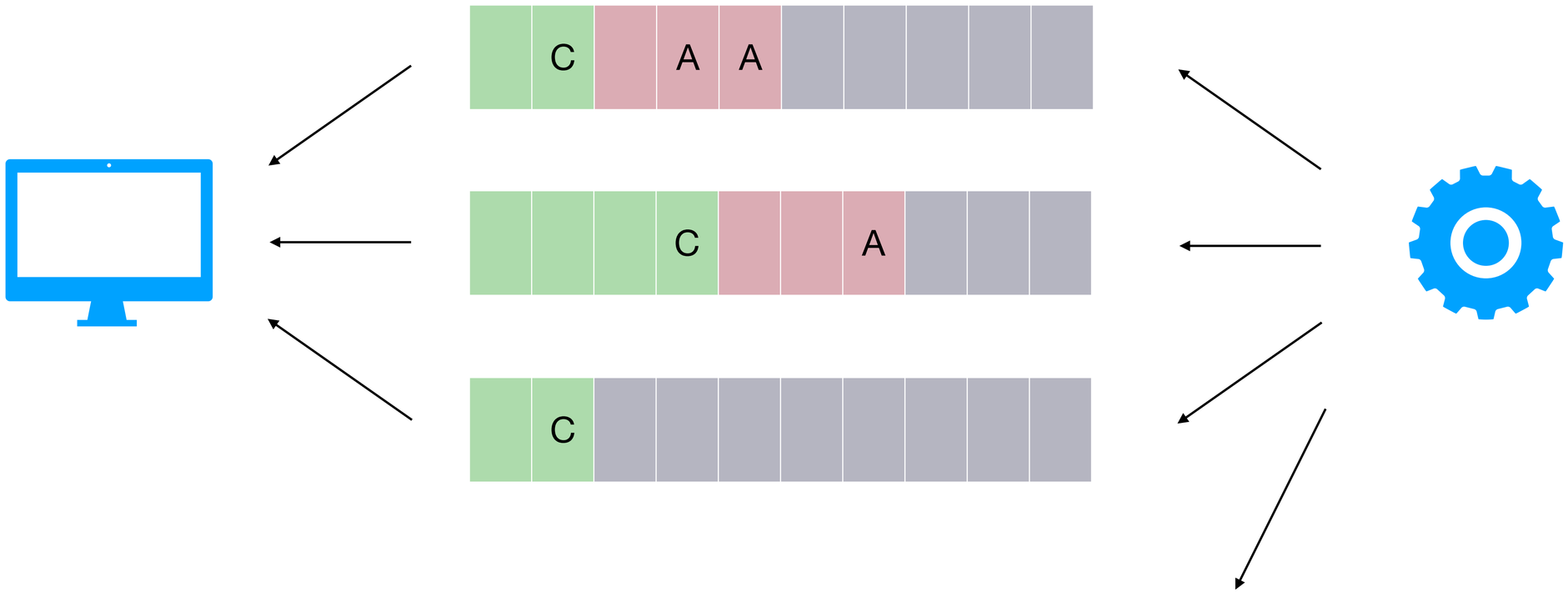
Transaction

Partitions	(0, 1, 2)	(0, 1, 2)	(0, 1, 2)	(0, 1)	(0, 1)	
State	ongoing	prepare commit	complete commit	ongoing	prepare abort	



Transaction

Partitions	(0, 1, 2)	(0, 1, 2)	(0, 1, 2)	(0, 1)	(0, 1)	
State	ongoing	prepare commit	complete commit	ongoing	prepare abort	



Transaction

Partitions	(0, 1, 2)	(0, 1, 2)	(0, 1, 2)	(0, 1)	(0, 1)	(0, 1)
State	ongoing	prepare commit	complete commit	ongoing	prepare abort	complete abort

Транзакции

- Атомарная запись на множество партиций
- Включая `__consumer_offsets`

Слабости семантики

- ~~Повторные попытки продюсера небезопасны~~
- Данные не пишутся атомарно с их смещениями (offsets)
- Нет защиты от зомби

Слабости семантики

- ~~Повторные попытки продюсера небезопасны~~
- ~~Данные не пишутся атомарно с их смещениями (offsets)~~
- Нет защиты от зомби

Слабости семантики

- ~~Повторные попытки продюсера небезопасны~~
- ~~Данные не пишутся атомарно с их смещениями (offsets)~~
- Нет защиты от зомби

Забор от Зомби

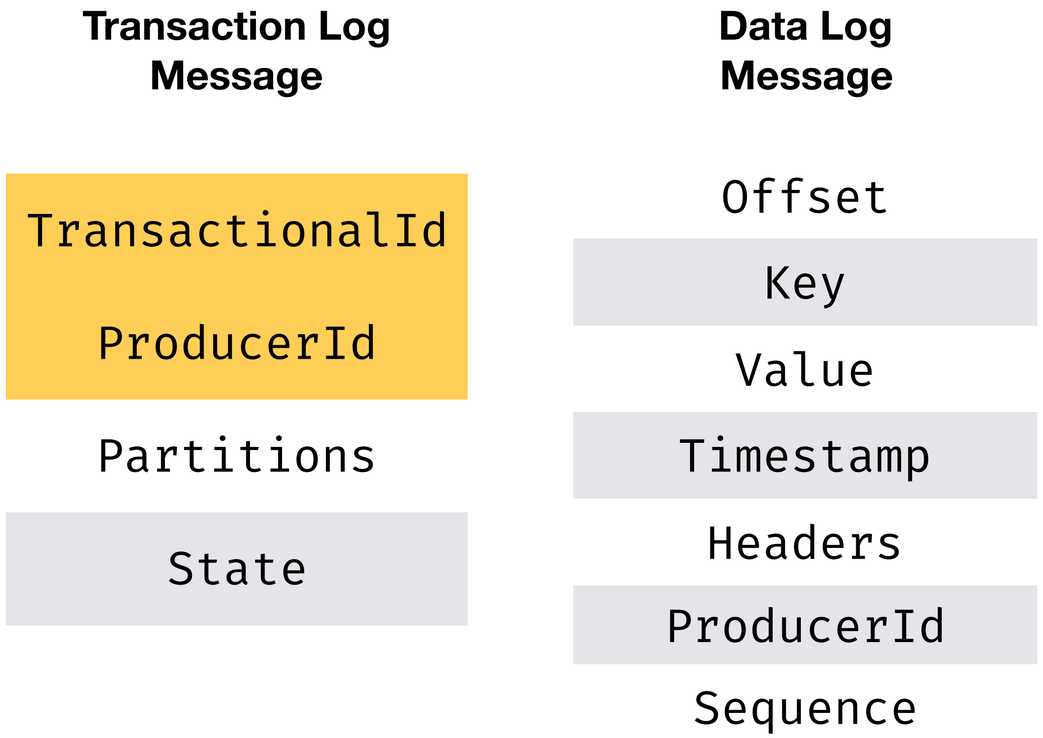
- Чтобы остановить зомби нам нужен «забор»

Забор от Зомби

- Чтобы остановить зомби нам нужен «забор»
- Но для начала, надо каким-то образом идентифицировать экземпляр продьюсера между перезапусками

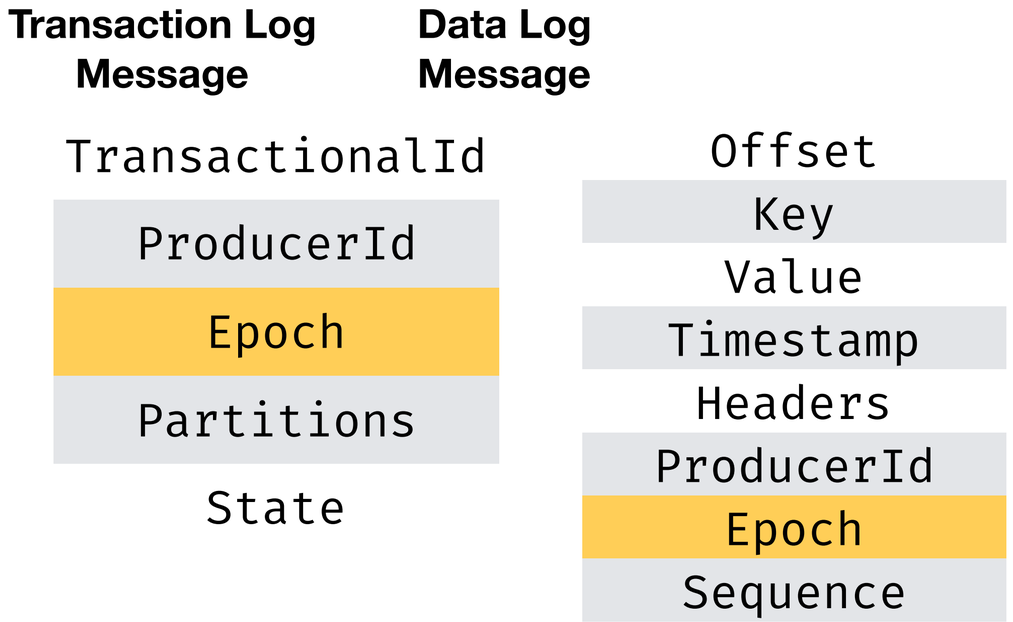
Zombie fencing

- “transactionalId” связан с producerId
- Не надо сохранять producerId
- transactional.id={id}



Zombie fencing

- Мы добавили эпоху продюсера epoch
- Epoch инкрементируется при инициализации продюсера



Zombie fencing

Producer 1

txnl.id=A

coordinator=1

Transaction Log 0 (Leader)

txnl.id				
pid				
epoch				
partitions				
state				

Zombie fencing

Producer 1

txnl.id=A

coordinator=1

InitProducerId(txnl.id=A)

Transaction Log 0 (Leader)

txnl.id				
pid				
epoch				
partitions				
state				

Zombie fencing

Producer 1

txnl.id=A

coordinator=1

InitProducerId(txnl.id=A)

Transaction Log 0 (Leader)

txnl.id	A			
pid	239			
epoch	0			
partitions	none			
state	init			

Zombie fencing

Producer 1

txnl.id=A

coordinator=1

pid=239, epoch=0

Transaction Log 0 (Leader)

txnl.id	A			
pid	239			
epoch	0			
partitions	none			
state	init			

Zombie fencing

Producer 1

txnl.id=A
coordinator=1
pid=239
epoch=0

Transaction Log 0 (Leader)

txnl.id	A			
pid	239			
epoch	0			
partitions	none			
state	init			

zombie fencing

Producer 1

txnl.id=A
coordinator=1
pid=239
epoch=0

AddPartitions([0])

Transaction Log 0 (Leader)

txnl.id	A	A		
pid	239	239		
epoch	0	0		
partitions	none	[0]		
state	init	ongoing		

Zombie fencing

Producer 1

txnl.id=A
coordinator=1
pid=239
epoch=0

OK

Transaction Log 0 (Leader)

txnl.id	A	A		
pid	239	239		
epoch	0	0		
partitions	none	[0]		
state	init	ongoing		

zombie fencing

Producer 1

txnl.id=A
coordinator=1
pid=239
epoch=0

Transaction Log 0 (Leader)

txnl.id	A	A		
pid	239	239		
epoch	0	0		
partitions	none	[0]		
state	init	ongoing		

Data partition 0 (Leader)

offset								
pid								
epoch								

zombie fencing

Producer 1

txnl.id=A
coordinator=1
pid=239
epoch=0

Produce(pid=239, epoch=0)

Transaction Log 0 (Leader)

txnl.id	A	A		
pid	239	239		
epoch	0	0		
partitions	none	[0]		
state	init	ongoing		

Data partition 0 (Leader)

offset	0							
pid	239							
epoch	0							

zombie fencing

Producer 1

txnl.id=A
coordinator=1
pid=239
epoch=0

Producer 2

txnl.id=A

coordinator=1

Transaction Log 0 (Leader)

txnl.id	A	A		
pid	239	239		
epoch	0	0		
partitions	none	[0]		
state	init	ongoing		

Data partition 0 (Leader)

offset	0							
pid	239							
epoch	0							

Zombie fencing

Producer 1

txnl.id=A

coordinator=1

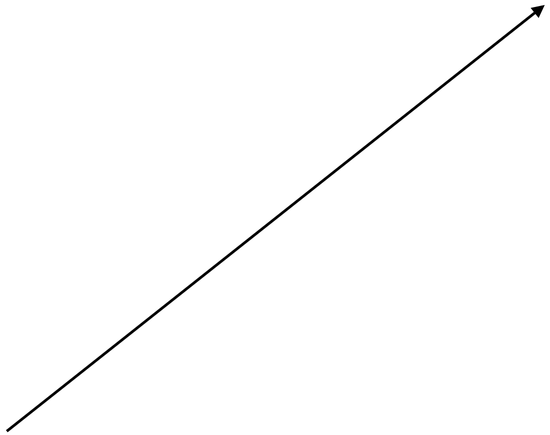
pid=239

epoch=0

Producer 2

txnl.id=A

coordinator=1



InitProducerId(txnl.id=A)

Transaction Log 0 (Leader)

txnl.id	A	A		
pid	239	239		
epoch	0	0		
partitions	none	[0]		
state	init	ongoing		

Data partition 0 (Leader)

offset	0							
pid	239							
epoch	0							

Zombie fencing

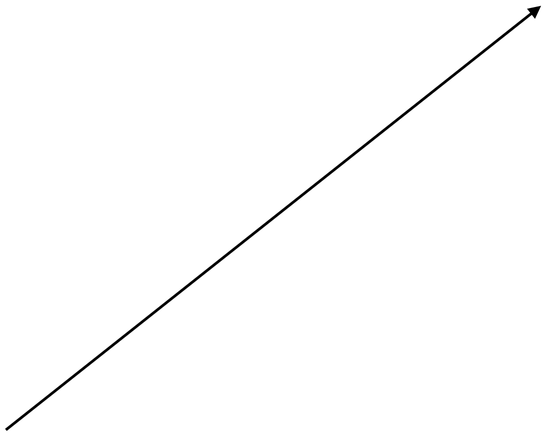
Producer 1

txnl.id=A
coordinator=1
pid=239
epoch=0

Producer 2

txnl.id=A

coordinator=1



InitProducerId(txnl.id=A)

Transaction Log 0 (Leader)

txnl.id	A	A	A	
pid	239	239	239	
epoch	0	0	1	
partitions	none	[0]	[0]	
state	init	ongoing	pr_abort	

Data partition 0 (Leader)

offset	0							
pid	239							
epoch	0							

Zombie fencing

Producer 1

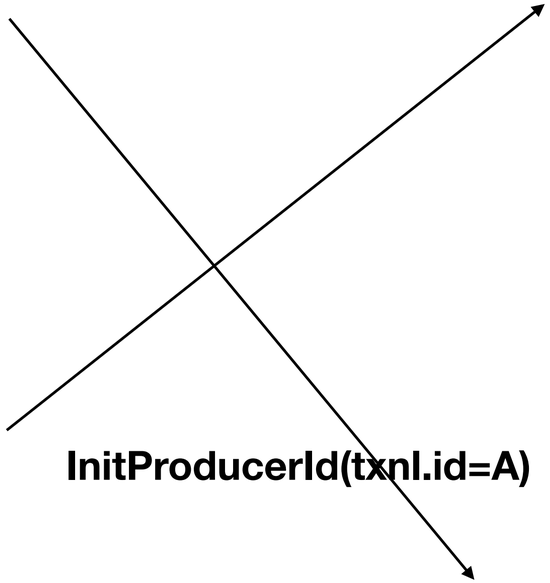
txnl.id=A

coordinator=1

pid=239

epoch=0

Produce(pid=239, epoch=0)



InitProducerId(txnl.id=A)

Producer 2

txnl.id=A

coordinator=1

Transaction Log 0 (Leader)

txnl.id	A	A	A	
pid	239	239	239	
epoch	0	0	1	
partitions	none	[0]	[0]	
state	init	ongoing	pr_abort	

Data partition 0 (Leader)

offset	0	1						
pid	239	239						
epoch	0	0						

Zombie fencing

Producer 1

txnl.id=A
coordinator=1
pid=239
epoch=0

Producer 2

txnl.id=A

coordinator=1

InitProducerId(txnl.id=A)

Transaction Log 0 (Leader)

txnl.id	A	A	A	
pid	239	239	239	
epoch	0	0	1	
partitions	none	[0]	[0]	
state	init	ongoing	pr_abort	

Data partition 0 (Leader)

offset	0	1	2					
pid	239	239	239					
epoch	0	0	1					

Zombie fencing

Producer 1

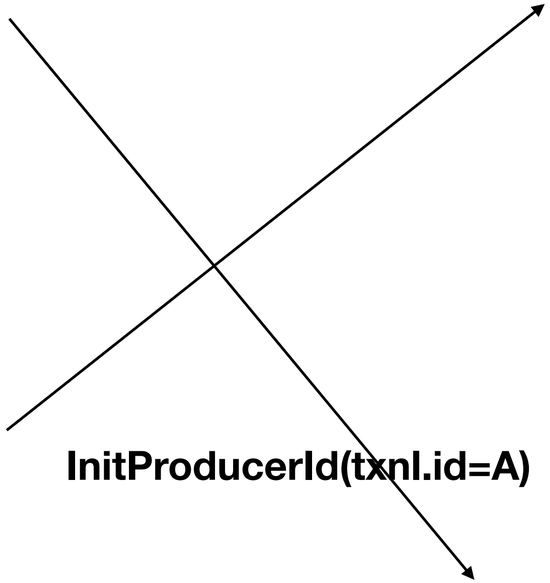
txnl.id=A

coordinator=1

pid=239

epoch=0

Produce(pid=239, epoch=0)



InitProducerId(txnl.id=A)

Producer 2

txnl.id=A

coordinator=1

Transaction Log 0 (Leader)

txnl.id	A	A	A	
pid	239	239	239	
epoch	0	0	1	
partitions	none	[0]	[0]	
state	init	ongoing	pr_abort	

Data partition 0 (Leader)

offset	0	1	2					
pid	239	239	239					
epoch	0	0	1					

Zombie fencing

Producer 1

txnl.id=A

coordinator=1

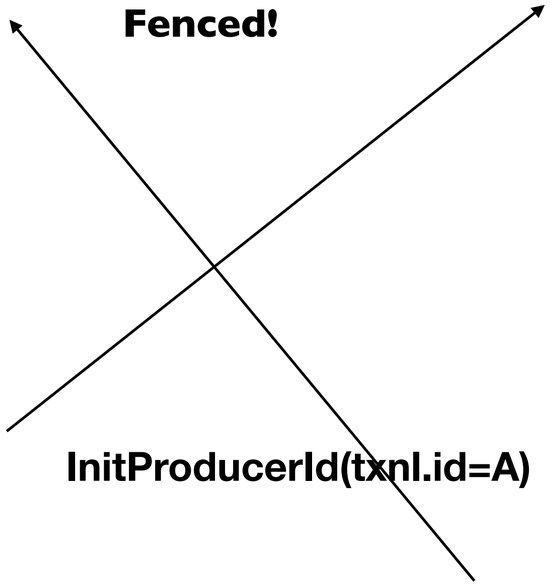
pid=239

epoch=0

Producer 2

txnl.id=A

coordinator=1



Transaction Log 0 (Leader)

txnl.id	A	A	A	
pid	239	239	239	
epoch	0	0	1	
partitions	none	[0]	[0]	
state	init	ongoing	pr_abort	

Data partition 0 (Leader)

offset	0	1	2					
pid	239	239	239					
epoch	0	0	1					

Zombie fencing

Producer 1

txnl.id=A

coordinator=1

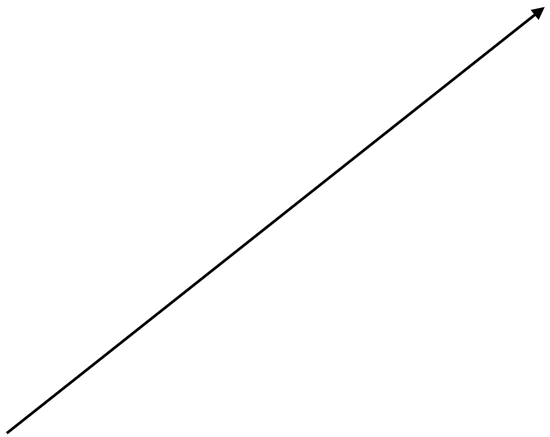
pid=239

epoch=0

Producer 2

txnl.id=A

coordinator=1



InitProducerId(txnl.id=A)

Transaction Log 0 (Leader)

txnl.id	A	A	A	A
pid	239	239	239	239
epoch	0	0	1	1
partitions	none	[0]	[0]	[0]
state	init	ongoing	p_abort	c_abort

Data partition 0 (Leader)

offset	0	1	2
pid	239	239	239
epoch	0	0	1

Zombie fencing

Producer 1

txnl.id=A

coordinator=1

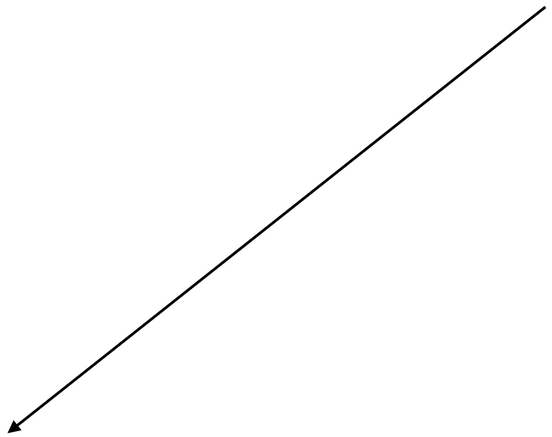
pid=239

epoch=0

Producer 2

txnl.id=A

coordinator=1



pid=239, epoch=1

Transaction Log 0 (Leader)

txnl.id	A	A	A	A
pid	239	239	239	239
epoch	0	0	1	1
partitions	none	[0]	[0]	[0]
state	init	ongoing	p_abort	c_abort

Data partition 0 (Leader)

offset	0	1	2
pid	239	239	239
epoch	0	0	1

Zombie fencing

Producer 1

txnl.id=A
coordinator=1
pid=239
epoch=0

Producer 2

txnl.id=A
coordinator=1
pid=239
epoch=1

Transaction Log 0 (Leader)

txnl.id	A	A	A	A
pid	239	239	239	239
epoch	0	0	1	1
partitions	none	[0]	[0]	[0]
state	init	ongoing	p_abort	c_abort

Data partition 0 (Leader)

offset	0	1	2
pid	239	239	239
epoch	0	0	1

Zombie fencing

Producer 1

txnl.id=A
coordinator=1
pid=239
epoch=0

CommitTxn(epoch=0)

Transaction Log 0 (Leader)

txnl.id	A	A	A	A
pid	239	239	239	239
epoch	0	0	1	1
partitions	none	[0]	[0]	[0]
state	init	ongoing	p_abort	c_abort

Producer 2

txnl.id=A
coordinator=1
pid=239
epoch=1

Data partition 0 (Leader)

offset	0	1	2
pid	239	239	239
epoch	0	0	1

Zombie fencing

Producer 1

txnl.id=A
coordinator=1
pid=239
epoch=0

Fenced!

Producer 2

txnl.id=A
coordinator=1
pid=239
epoch=1

Transaction Log 0 (Leader)

txnl.id	A	A	A	A
pid	239	239	239	239
epoch	0	0	1	1
partitions	none	[0]	[0]	[0]
state	init	ongoing	p_abort	c_abort

Data partition 0 (Leader)

offset	0	1	2
pid	239	239	239
epoch	0	0	1

Слабости семантики

- ~~Повторные попытки продюсера небезопасны~~
- ~~Данные не пишутся атомарно с их смещениями (offsets)~~
- Нет защиты от зомби

Слабости семантики

- ~~Повторные попытки продюсера небезопасны~~
- ~~Данные не пишутся атомарно с их смещениями (offsets)~~
- ~~Нет защиты от зомби~~

Семантика «exactly-once»

```
p.initTxns()
c.subscribe(inputTopics)
while (true) {
  offsets, input = c.poll()
  output = process(input)
  p.beginTxn()
  p.send(output)
  p.sendOffsets(offsets)
  p.commitTxn()
}
```

Семантика «exactly-once»

```
p.initTxns()
c.subscribe(inputTopics)
while (true) {
  offsets, input = c.poll()
  output = process(input)
  p.beginTxn()
  p.send(output)
  p.sendOffsets(offsets)
  p.commitTxn()
}
```

```
void beginTransaction();
```

Семантика «exact!y-once»

```
void sendOffsetsToTransaction(
  Map<TopicPartition, OffsetAndMetadata> offsets, String consumerGroupId)
throws ProducerFencedException;
```

Семантика «exactly-once»

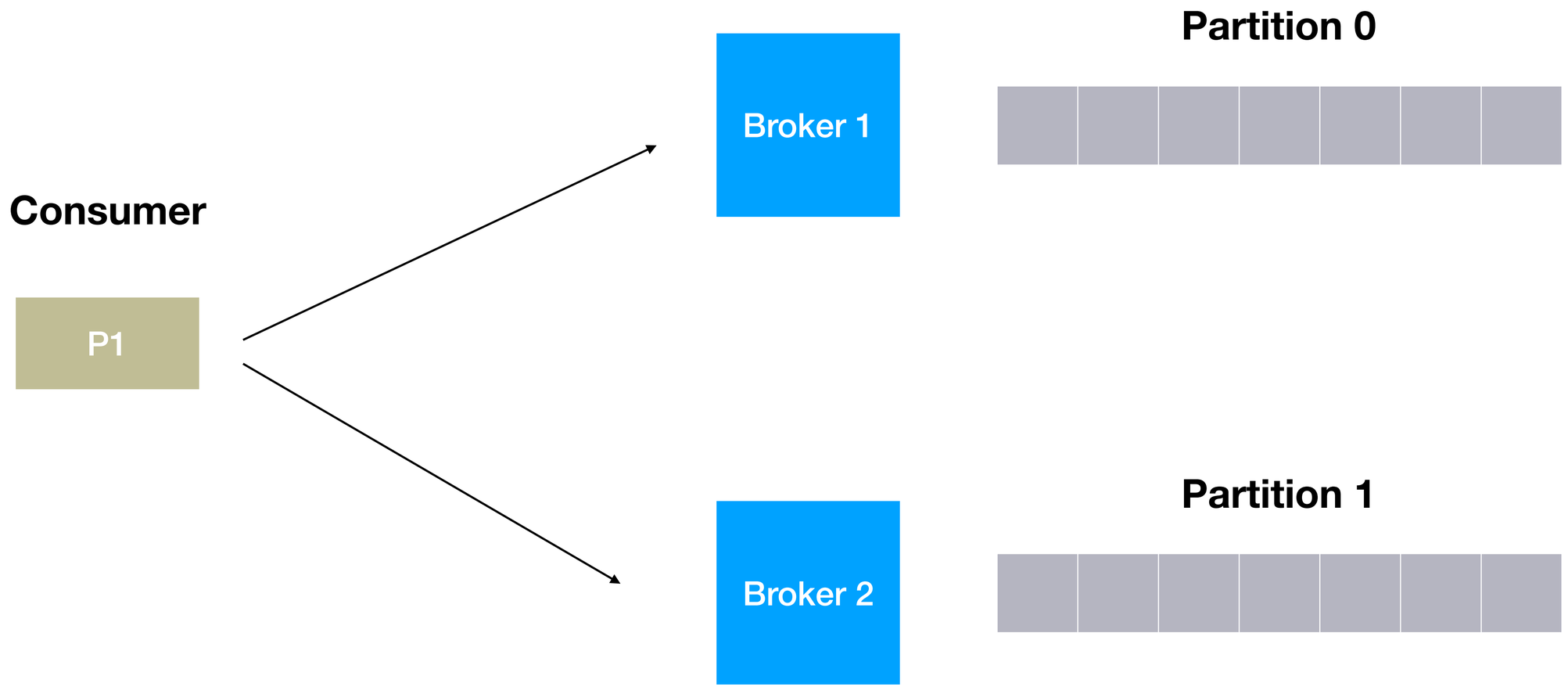
```
p.initTxns()
c.subscribe(inputTopics)
while (true) {
  offsets, input = c.poll()
  output = process(input)
  p.beginTxn()
  p.send(output)
  p.sendOffsets(offsets)
  p.commitTxn()
}
```

```
void commitTransaction()
throws ProducerFencedException;
```

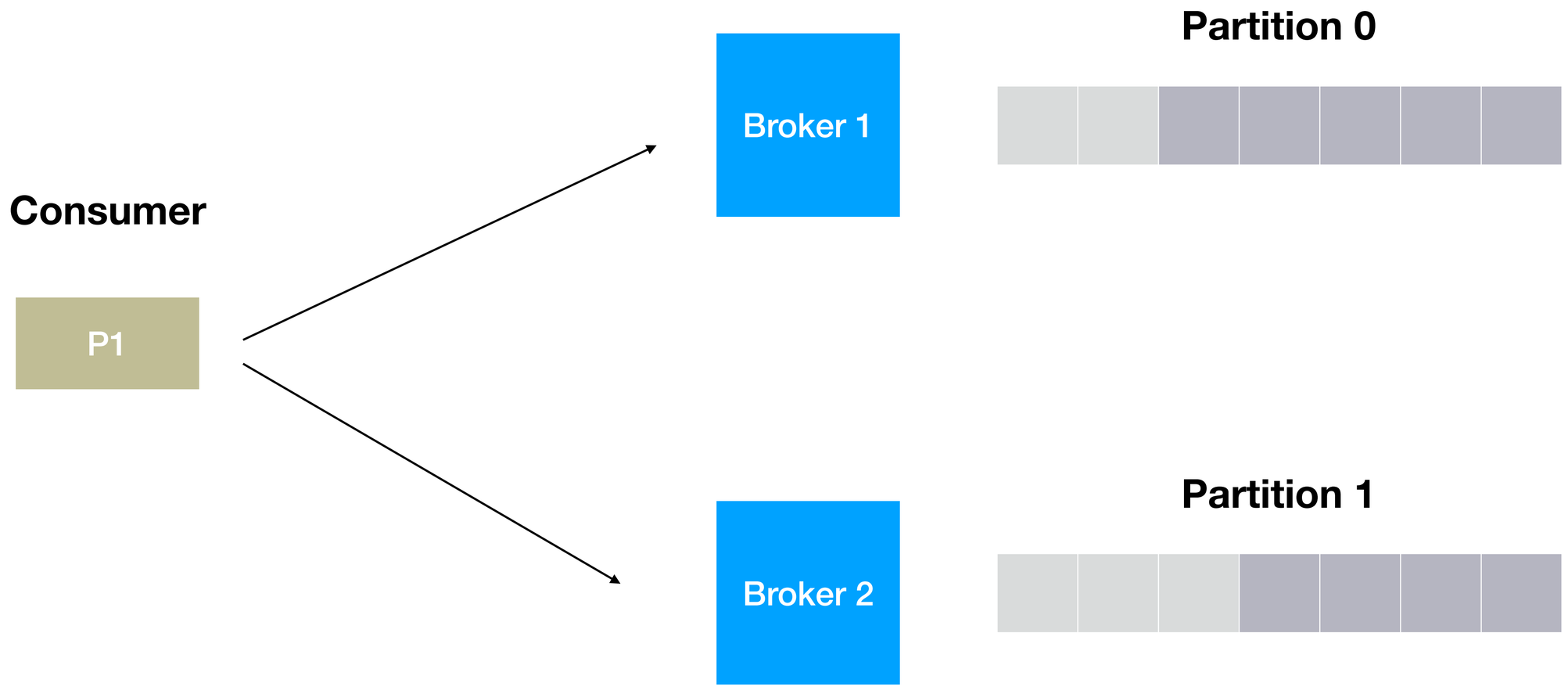
Изоляция чтения у Consumer

- Consumer `isolation.level`:
  - `read_committed`: доступны только закоммиченные сообщения
  - `read_uncommitted`: все сообщения доступны

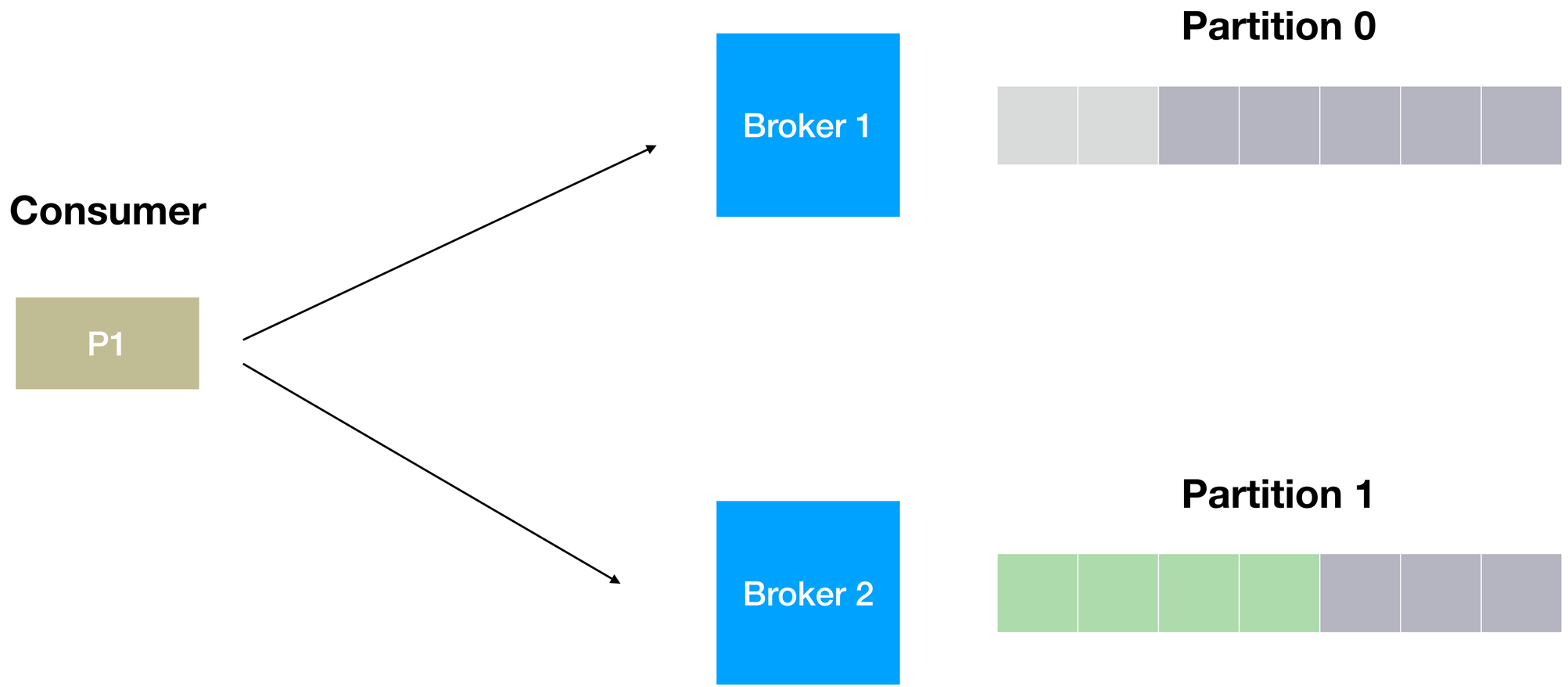
Изоляция чтения у Consumer



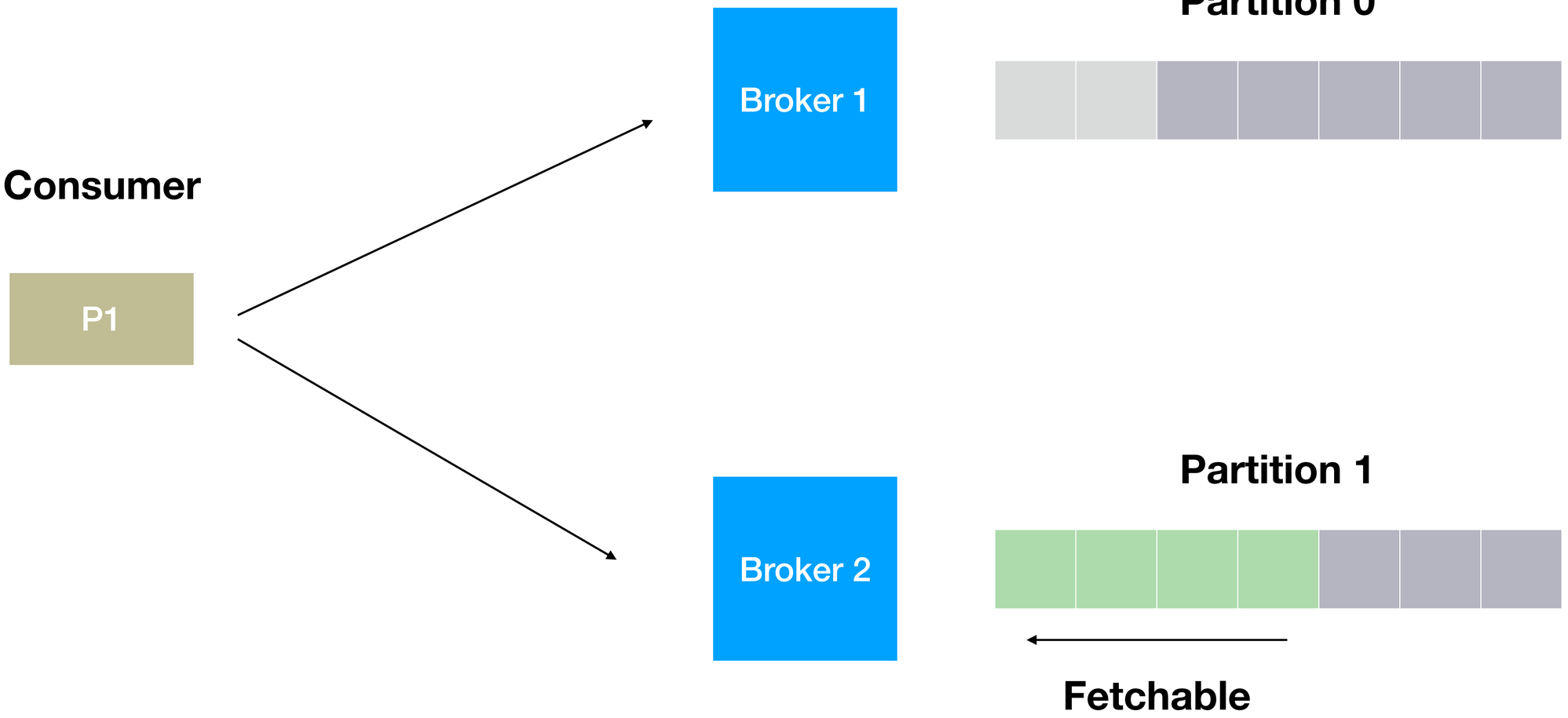
Изоляция чтения у Consumer



Изоляция чтения у Consumer



Изоляция чтения у Consumer



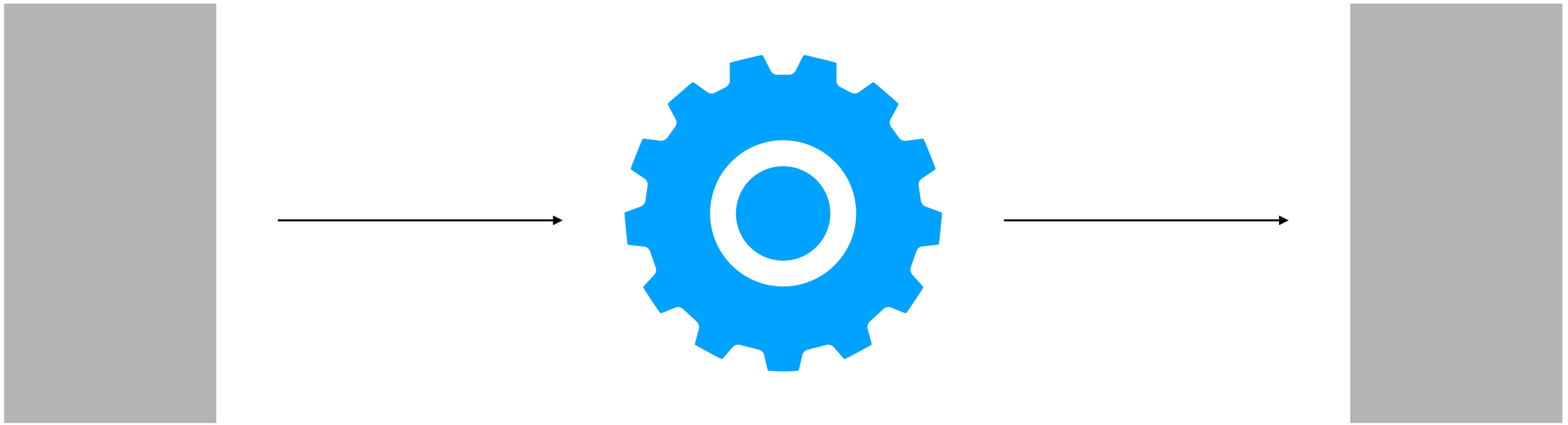
Семантика «Только один раз»

- Идемпотентный продюсер - доставка сообщений без дубликатов
- Транзакционный продюсер: возможность атомарной записи в несколько партиции (в том числе в Хранилище offsets)
- Read-committed консьюмер: только законченные сообщения будут возвращены

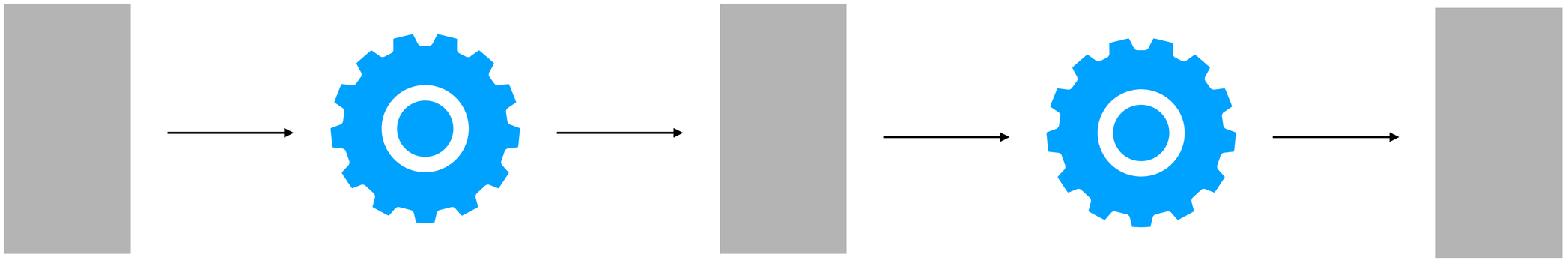
А тормозить не будет?

- Оверхед минимальный
- Чем продолжительней транзакция, тем выше end-to-end !atency

end-to-end eos

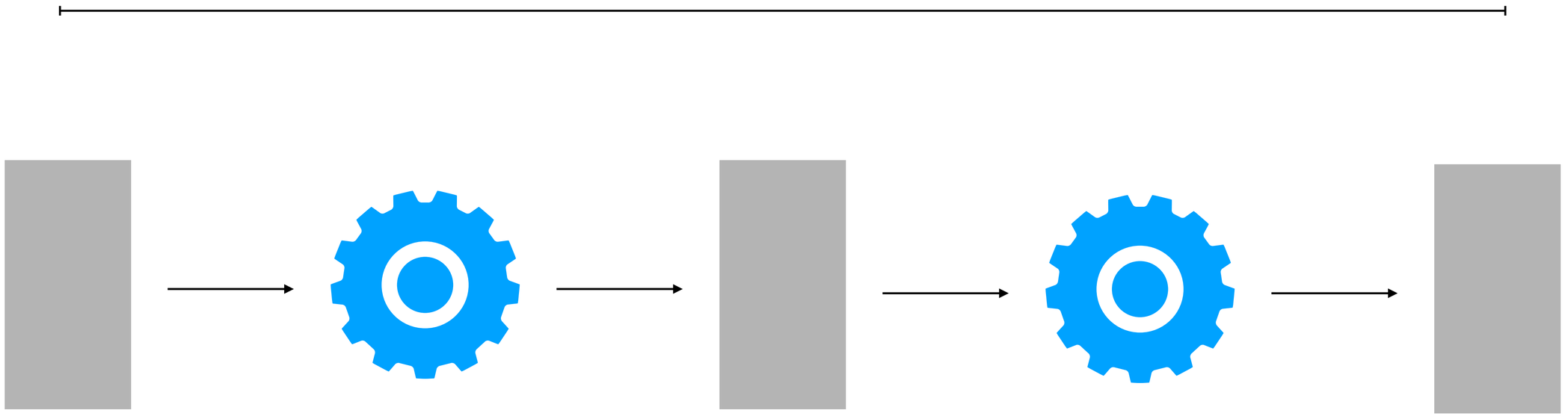


end-to-end eos



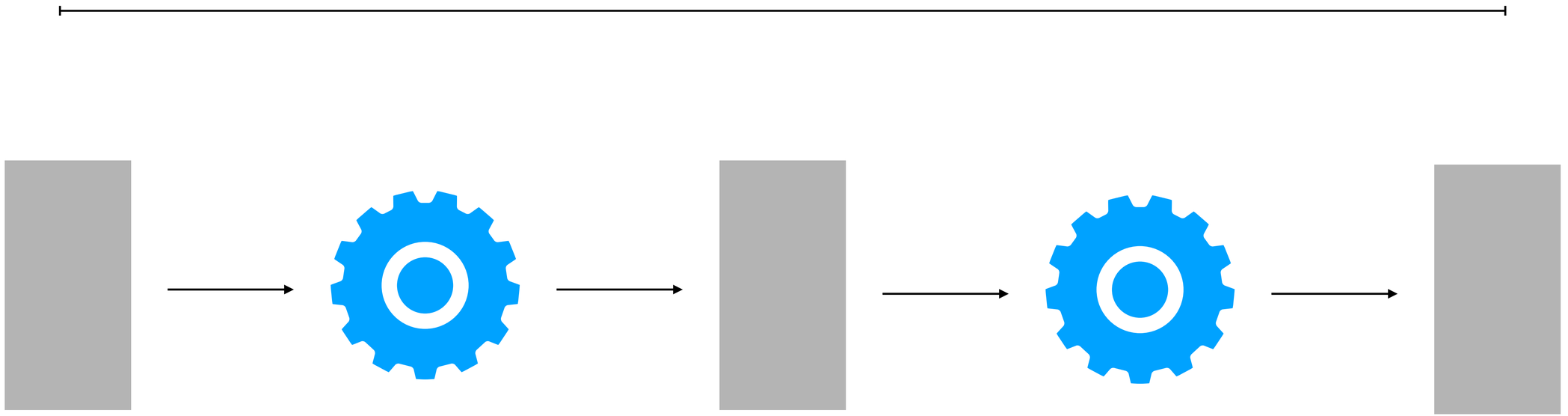
end-to-end eos

Kafka EoS



end-to-end eos

Kafka EoS



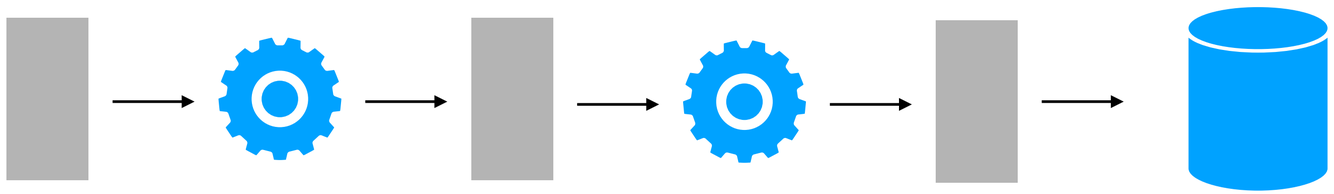
(Kafka Streams)

processing.guarantee=exactly_once

end-to-end eos

Kafka EoS

Необходима дополнительная координация



(Kafka Streams)

`processing.guarantee=exactly_once`

(Kafka Connect Sink)

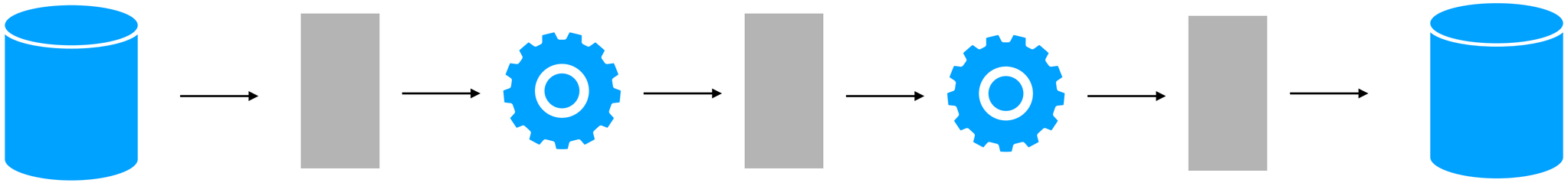
HDFS, ElasticSearch, S3,

end-to-end eos

Kafka EoS??

Kafka EoS

Необходима дополнительная координация



(Kafka Connect Source)

Work in progress

(Kafka Streams)

`processing.guarantee=exactly_once`

(Kafka Connect Sink)
HDFS, ElasticSearch, S3, etc.

Команда EOS
пацаны Ваше ребята...

- Apurva Mehta, Damian Guy, F!avio Junqueira, Guozhang Wang, Matthias J. Sax, Sriram Subramanian
- More info: goog!e KIP-98

Приходите к нам работать!

confluent

https://www.confluent.io/careers/