Протокол Totem

Оглавление

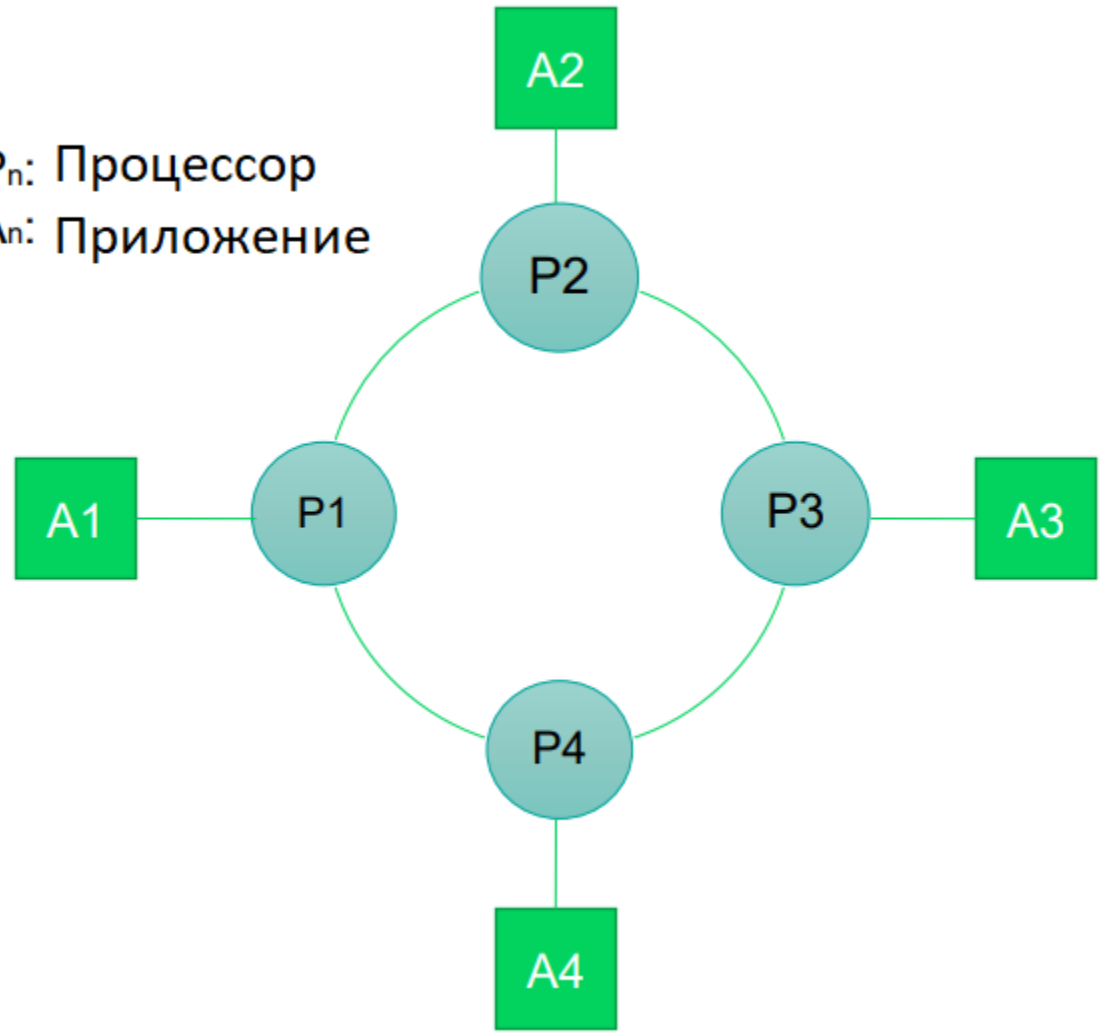
Введение

- Totem на одном кольце:
  - Поддерживает высокопроизводительные отказоустойчивые распределенные системы, которые продолжают работать несмотря на изменения сети(разделение, объединение), ошибки в узлах;
  - Обеспечивает полную упорядоченную доставку сообщений с низкими накладными расходами, высокой пропускной способностью, низкой задержкой, используя логическое кольцо с пересылками;
  - Обеспечивает быстрое определение разделения сети и сбоев процессов вместе с перестройкой;
- Totem с избыточным кольцом:
  - Основан на базовой версии;
  - Более надежен за счет дополнительных сетевых интерфейсов.

Введение

- Процессор
  - Узел corosync, который является членом закрытой группы процессов(Closed Process Group – CPG);
- Приложение
  - Программа использующая corosync.

P_n: Процессор
A_n: Приложение



Введение

- Broadcast
  - Один процессор => все процессоры
- Передача(Transmit)
  - Один процессор => следующий процессор;
- Посылка(Delivery)
  - Один процессор => ассоциированное с ним приложение.

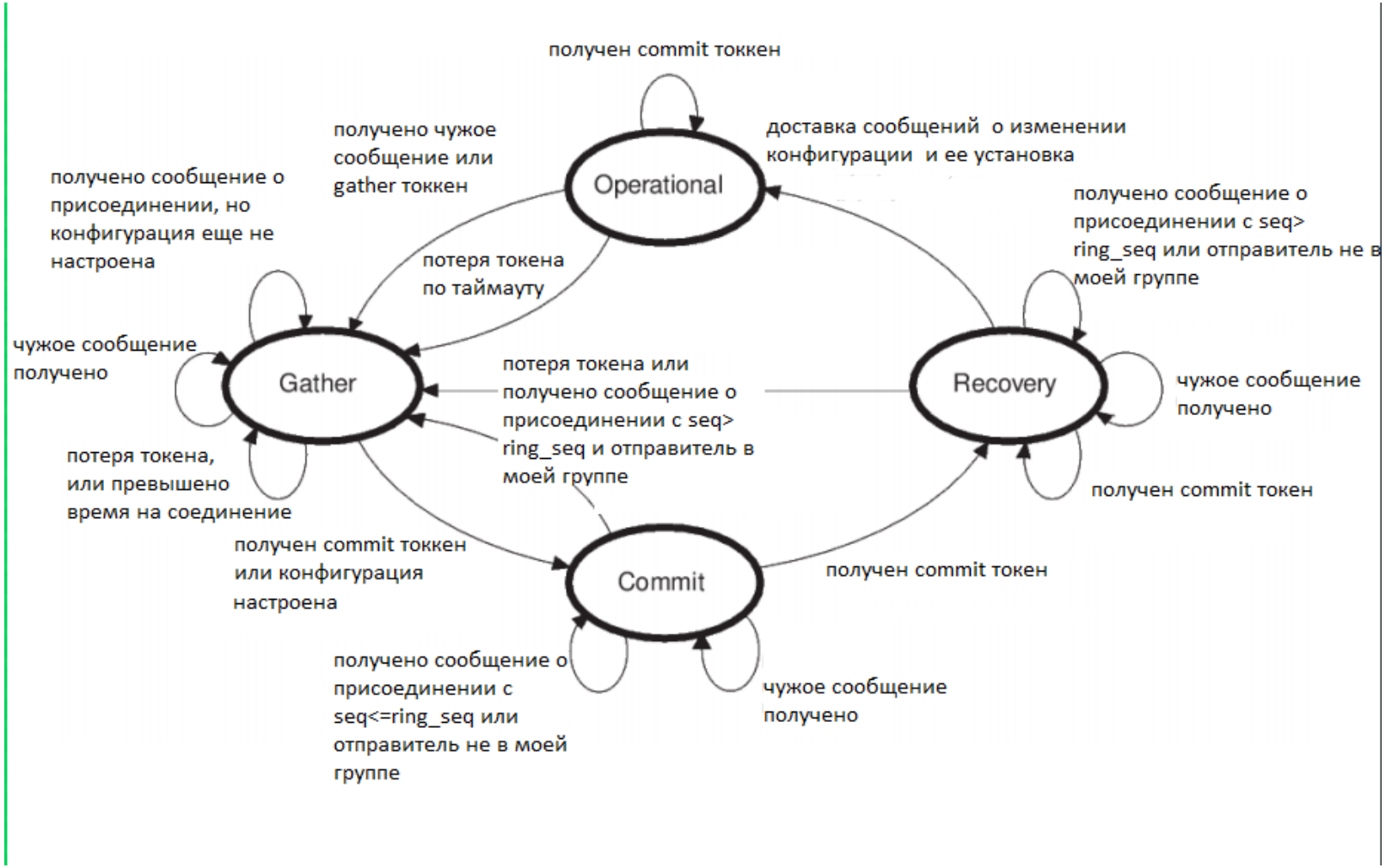
Надежный упорядоченный сервис доставки

- Надежная доставка для конфигурации C
  - Каждое сообщение имеет уникальный идентификатор;
  - Каждый процессор посылает одно сообщение строго один раз. Если процессор посылает два сообщения, то он пошлет сначала первое, потом второе;
  - Если процессор создал сообщение, то он либо пошлет его, либо случиться ошибка, но сначала он доставит сообщение о смене конфигурации;
  - Если процессор член группы C и конфигурация не изменилась, тогда процессор доставит в C все сообщения, которые были в C;
  - Если процессор доставляет сообщение созданное в C, то он становиться членом группы C;
  - Если процессы p и q были в старой конфигурации и остались в новой, то они доставят все старые сообщения до смены конфигурации.

Способы доставки сообщений

- Доставка в общем порядке
  - Доставка удовлетворяет часа Лэмпорта внутри конфигурации;
- Доставка в согласованном порядке
  - Гарантирует, что все сообщения будут доставляться в одном общем порядке . Когда процессору нужно доставить сообщение, сначала он доставляет все прошлые сообщения;
- Доставка в безопасном порядке
  - Когда процессор доставляет сообщения, он определяет каждый процессор, у которого есть такое же сообщение, и если на первом случилась ошибка, сообщение отправит другой.

Totem на простом кольце. Машина состояния.

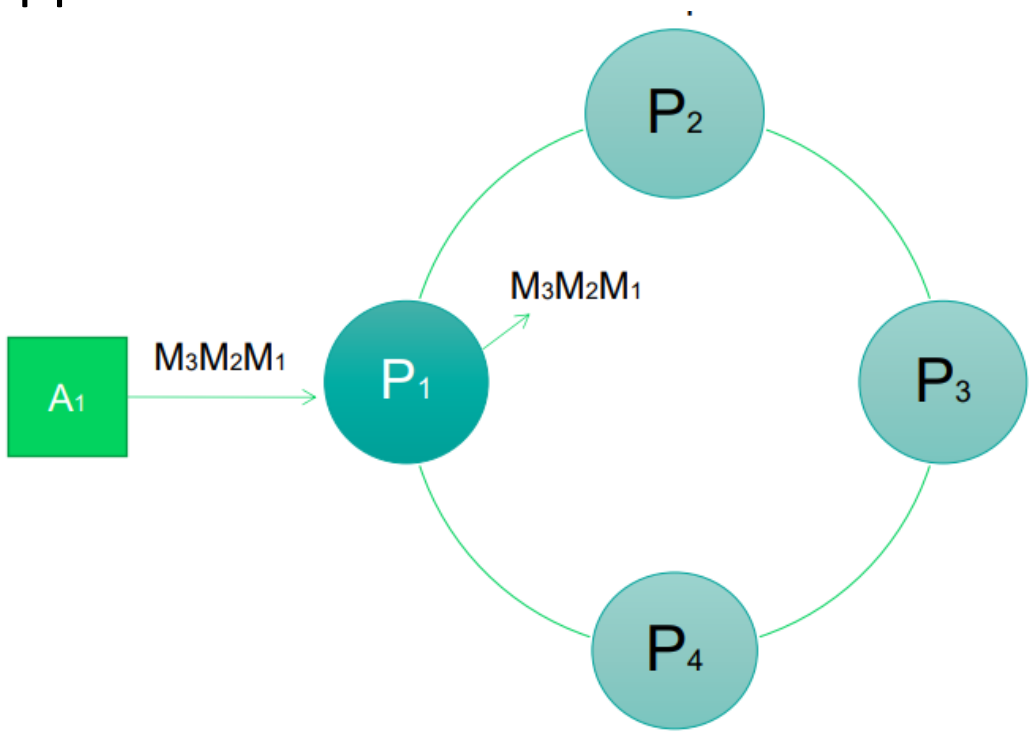


Totem Ordering Protocol

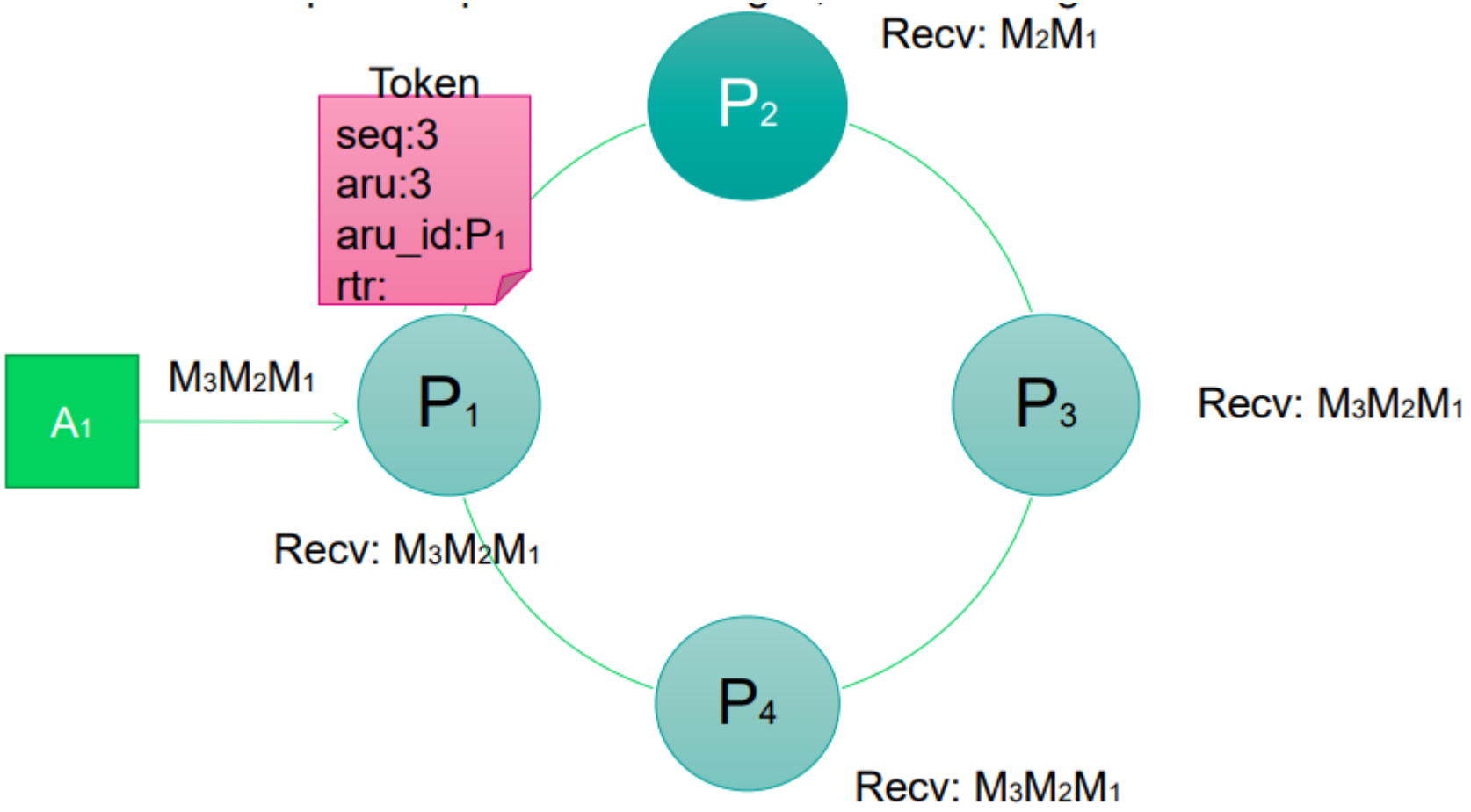
- Действует в состоянии Operational;
- Доставка сообщений приложениям обеспечивается в согласованном или безопасном порядке;
- Приложения могут уточнять в каком порядке они буду работать (согласованном или безопасном);
- Процессоры могут доставляет сообщения в общем порядке одним за одним.

Пример

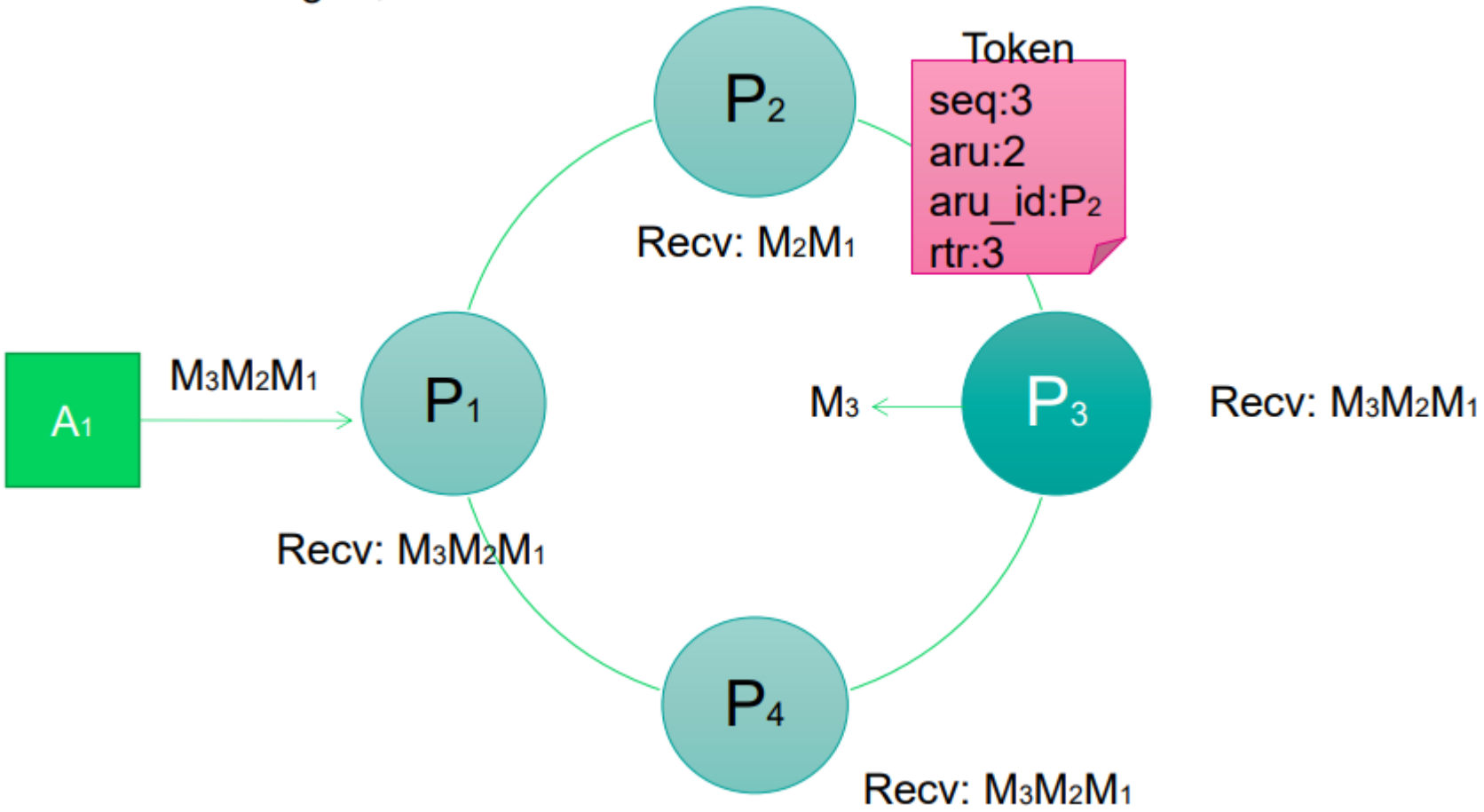
- A1 просит P1 доставит 3 кусочка сообщения: M1, M2, M3
- P1 передает M1, M2, M3 дальше, сначала сохраняя сообщения в своей очереди



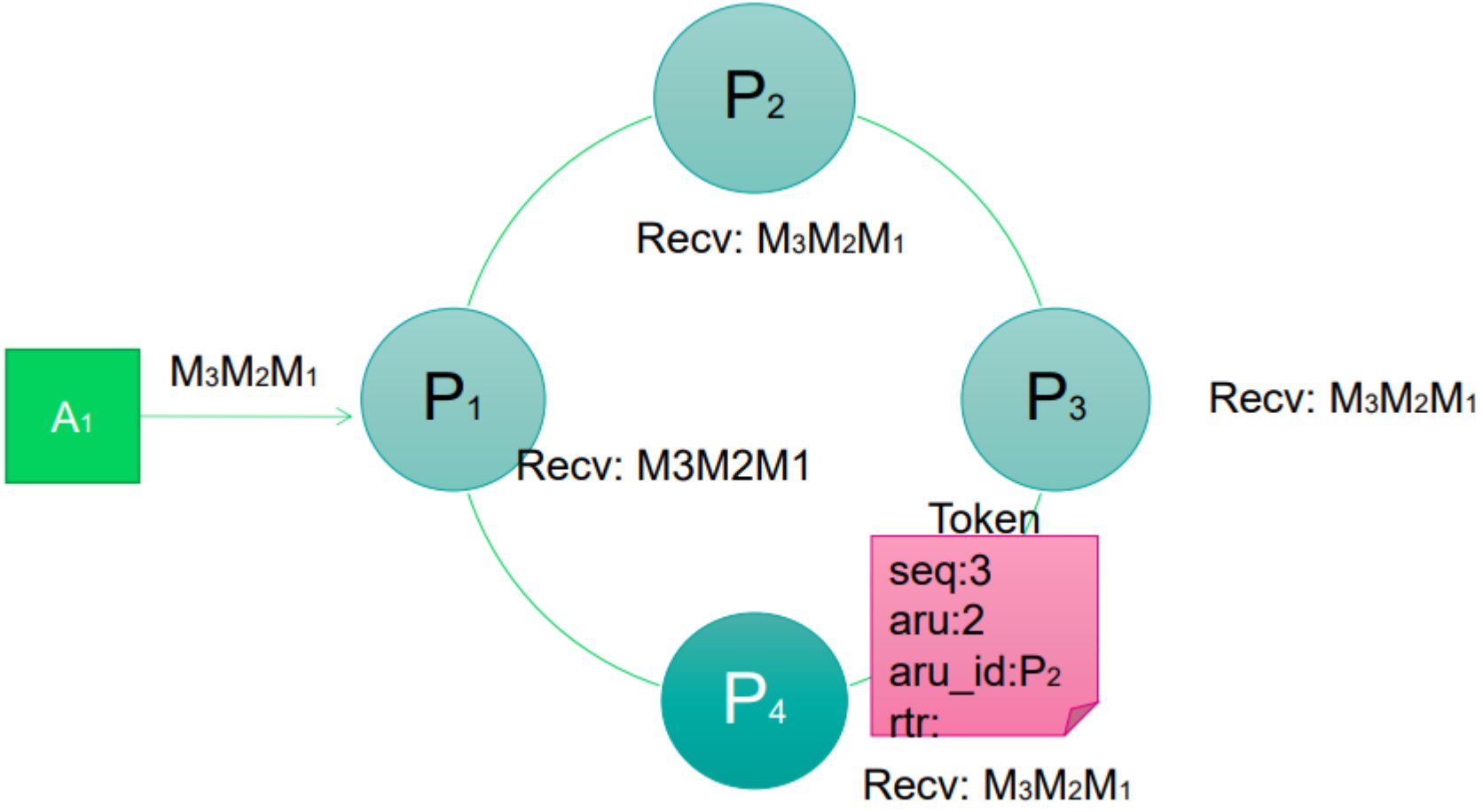
- P2 получил только M2, M1, когда P3 и P4 получили M3, M2, M1
- P1 отправляет токен P2, в токене поле seq содержит информацию о максимальном количестве сообщений,
- P2 сравнивает поле seq со своими сообщениями и видит, что не получил M3.



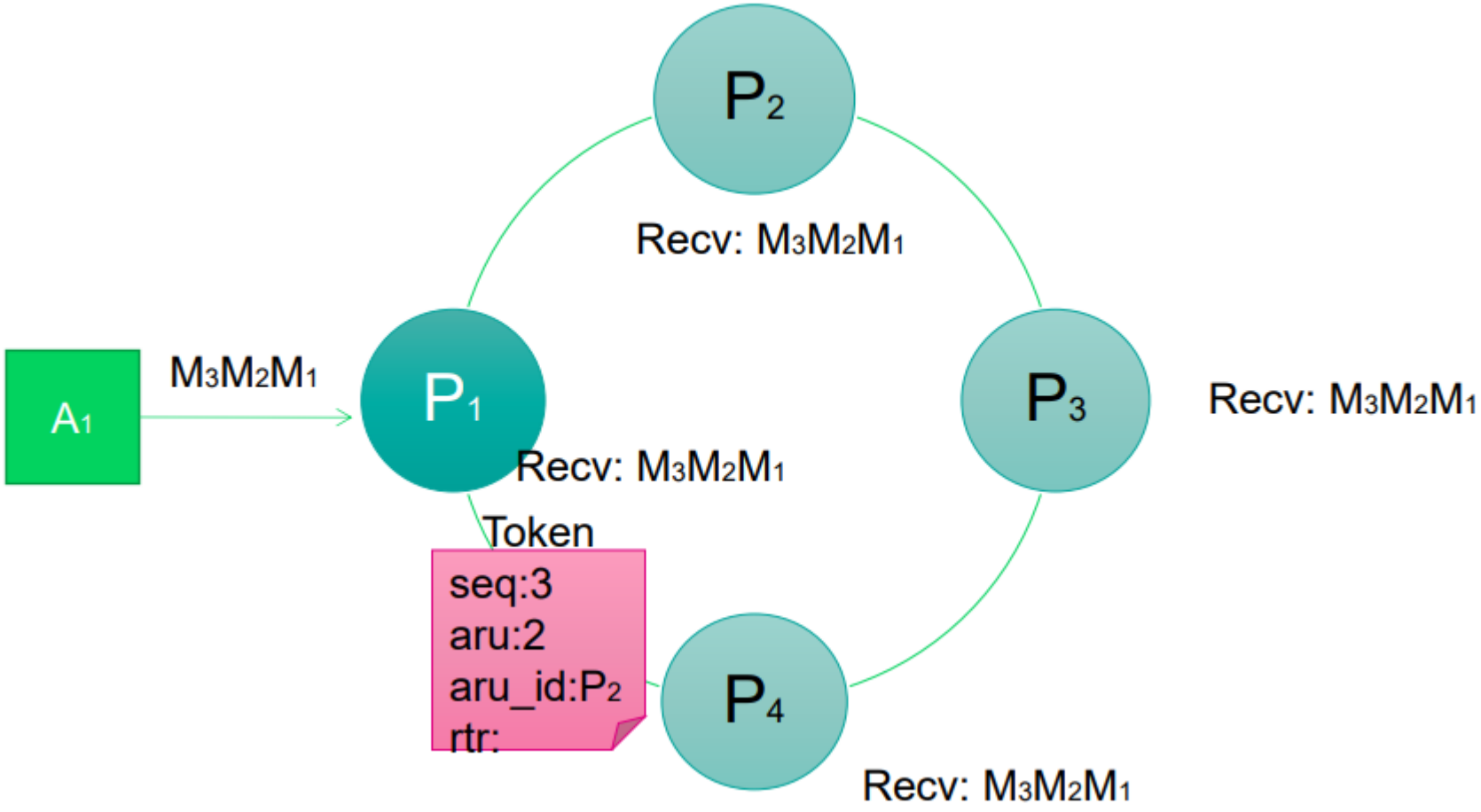
- P2 обновляет поле aru(aru = 2) в токене – количество, полученных сообщений, устанавливает rtr = 3 и передает токен P3
- Когда P3 получает токен, он делает broadcast M3 на кластер
- После P3 очищает поле rtr и передает токен P4



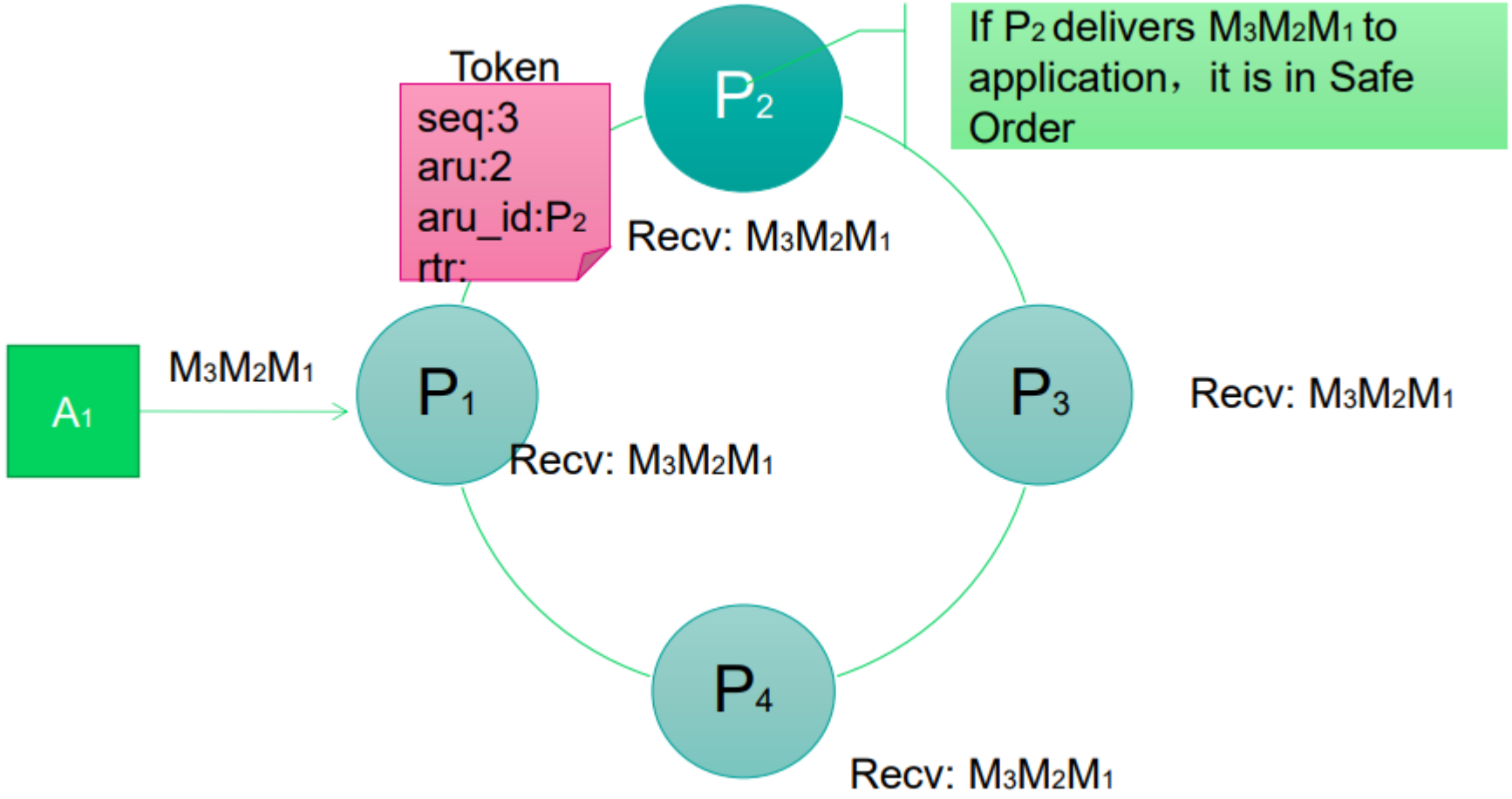
- P2 получает сообщение M2, остальные игнорируют
- P4 ничего не делает с токеном, полученным от P3, и передает его дальше



- P1 получает токен от P4, ничего с ним не делает и отправляет его P2



- P2 находит поле aru_id в токене, видит свой id и отмечает, что получил M3 (aru = 3), теперь P2 понимает, что получил все сообщения
- P2 передает токен P3



В каком порядке отправлять сообщения (согласованный/безопасный)

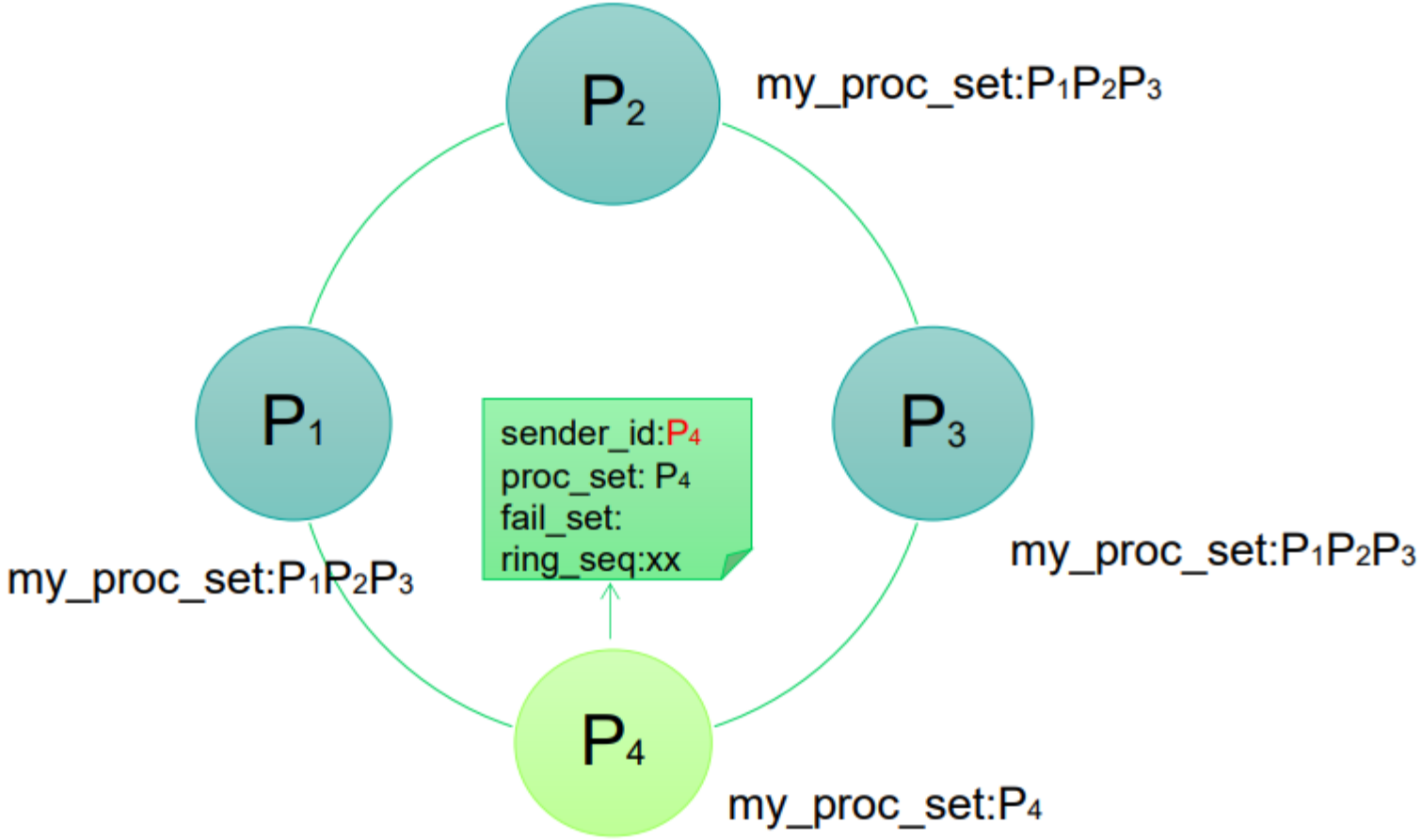
- Согласованный порядок
  - Если процессор отправляет токен приложению
- Безопасный порядок
  - Если aru > seq больше, чем в двух последовательных передачах

The Membership Protocol

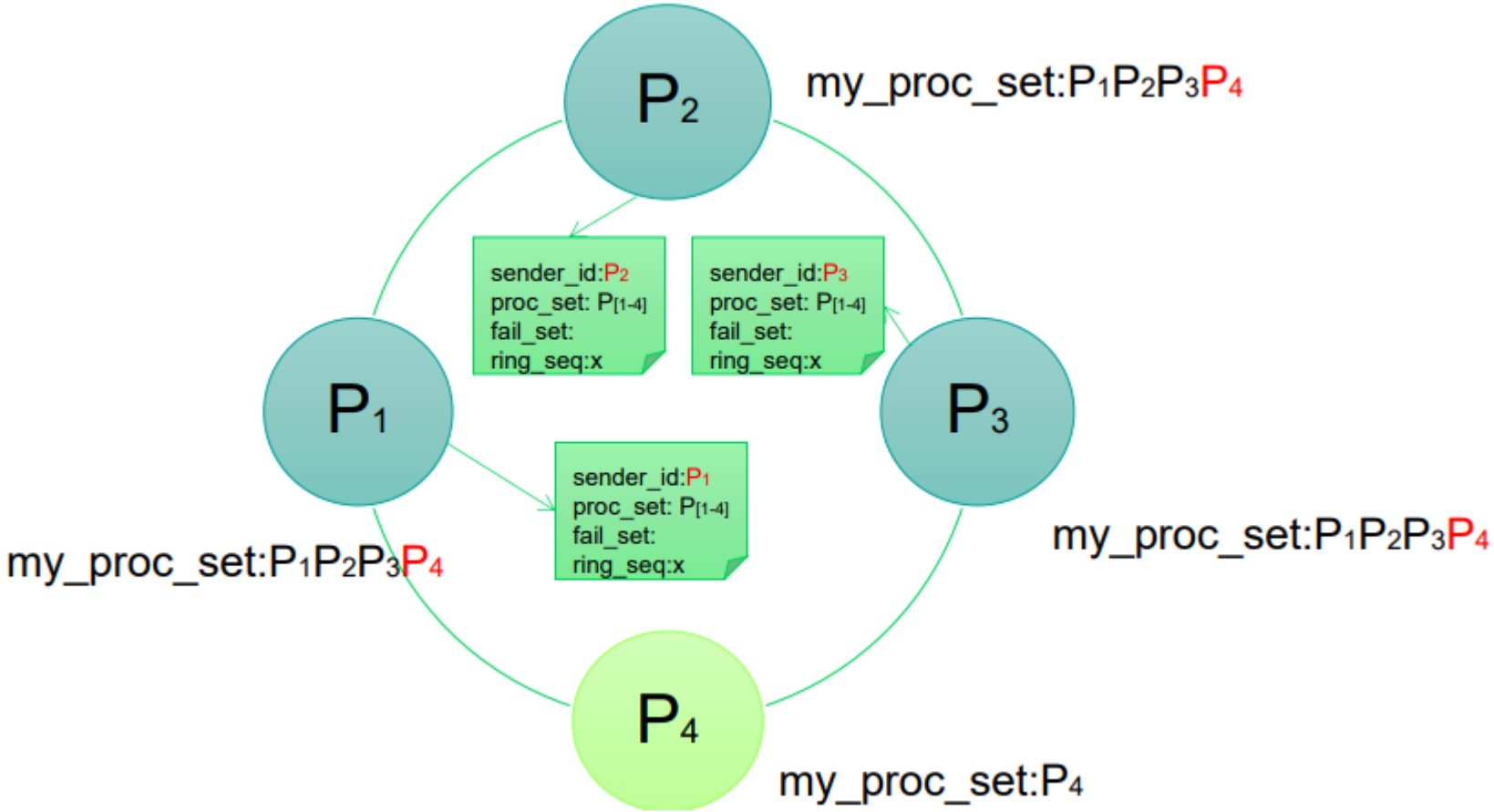
- Действует в состоянии сборки(Gather) и Commit
- Когда новый процессор присоединяется или старый процессор покидает кластер, кольцо перестраивается

Пример: присоединение нового процессора

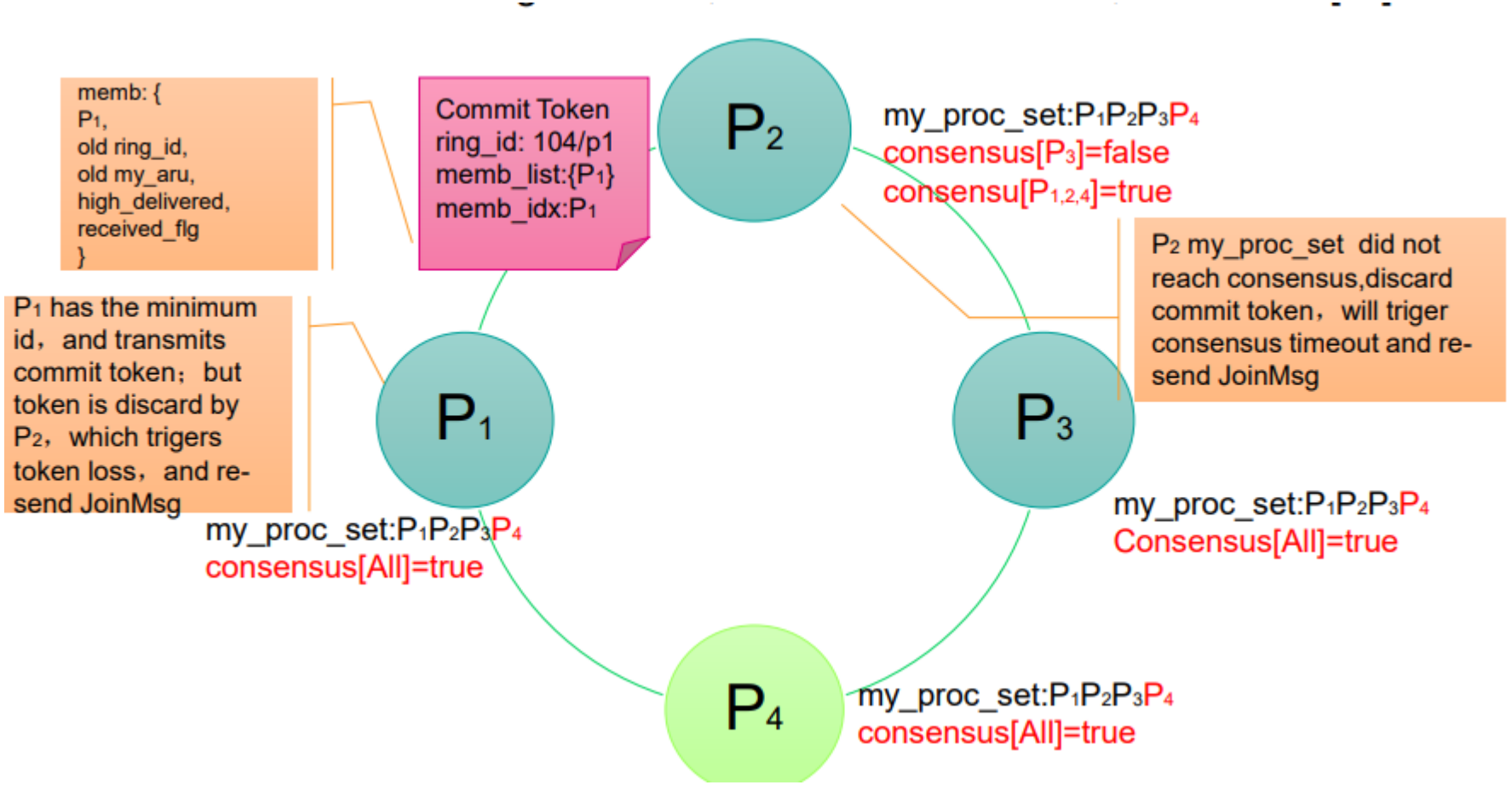
- P4 это новый процессор, который хочет присоединится
- Старое кольцо { P1, P2, P3 }. Узлы старого кольца хранят список участников в поле my_proc_set
- Когда P4 присоединяется к кластеру, он отправляет broadcast-сообщение о присоединении
- Когда P1, P2. P3 получают сообщение о присоединении, они переходят в состояние сборки



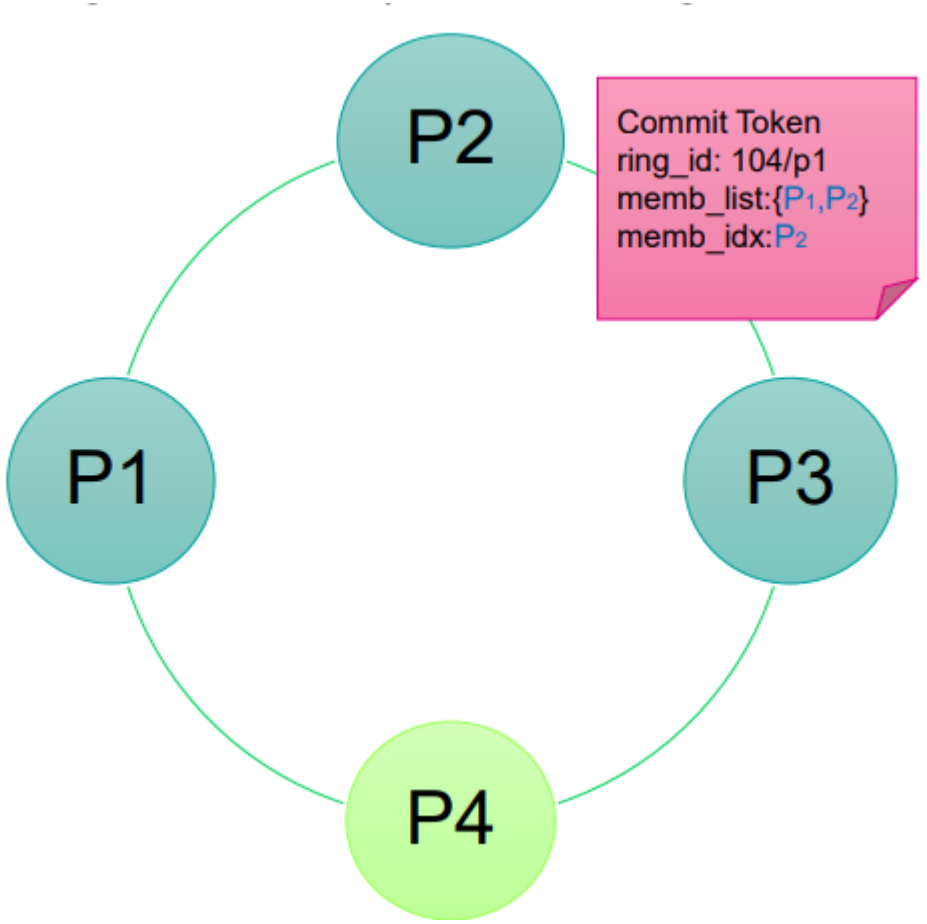
- Когда P1, P2, P3 получили сообщение о присоединение от P4, они добавляют P4 в my_proc_set
- P1, P2, P3 рассылают новое broadcast-сщщбщение о приссоединение
- Когда процессор получает сообщение о присоединение от другого процессора, он сравнивает proc_set со своим и если они совпадают, то переходит в состояние консенсуса



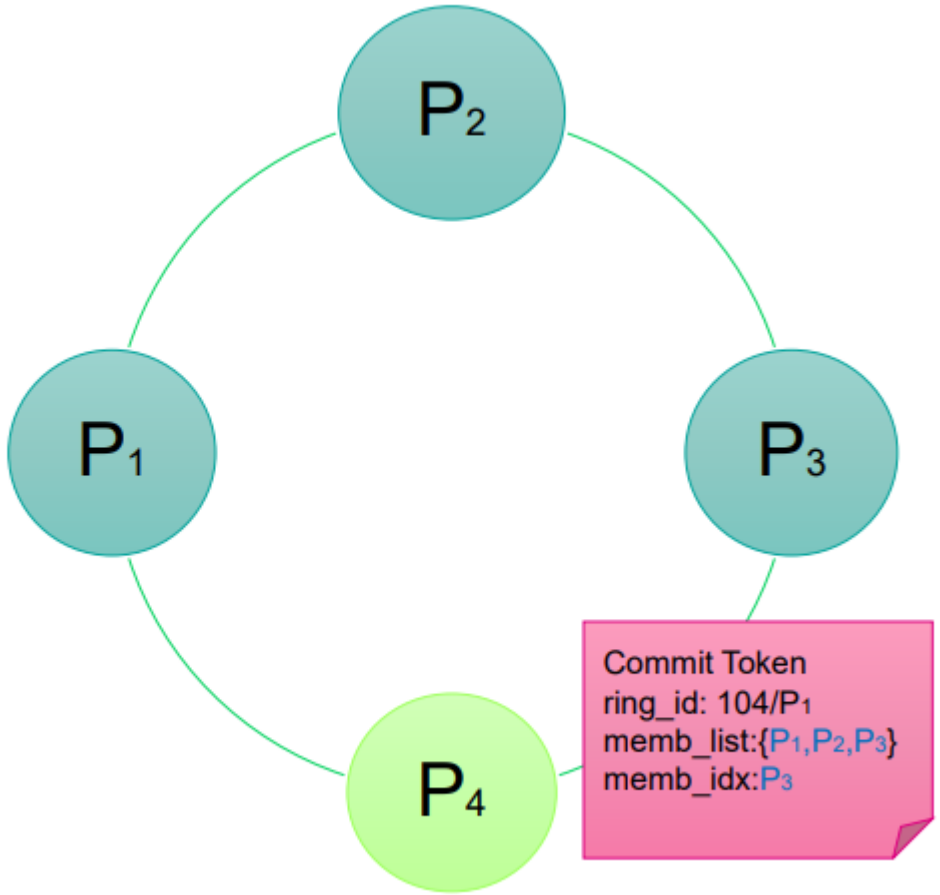
- Когда процессор находит, что все участники в my_proc_set достигли консенсуса, он отправляет Commit токен, если имеет минимальный id
- CommitToken’s ring_id.seq = max(old ring_id and JoinMsg’s ring_id) + 4
- Через некоторое время процессоры P1, P3, P4 достигнут консенсуса
- P2 не получил сообщение от P3, в P2 consensus[P3]=false



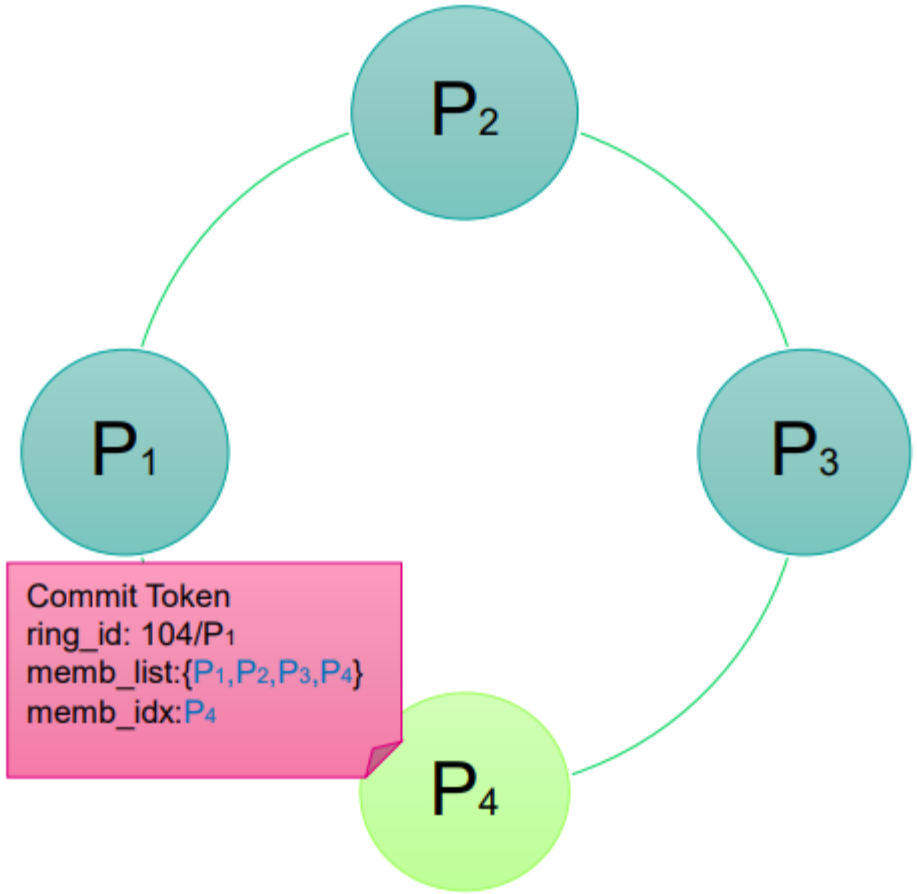
- В обычной ситуации через некоторое количество коммуникаций о присоединении, все процессоры будут в состояние консенсуса
- P2 получает Commit токен от P1, обновляет member_list и memb_idx, передает токен и переходит в состояние Commit



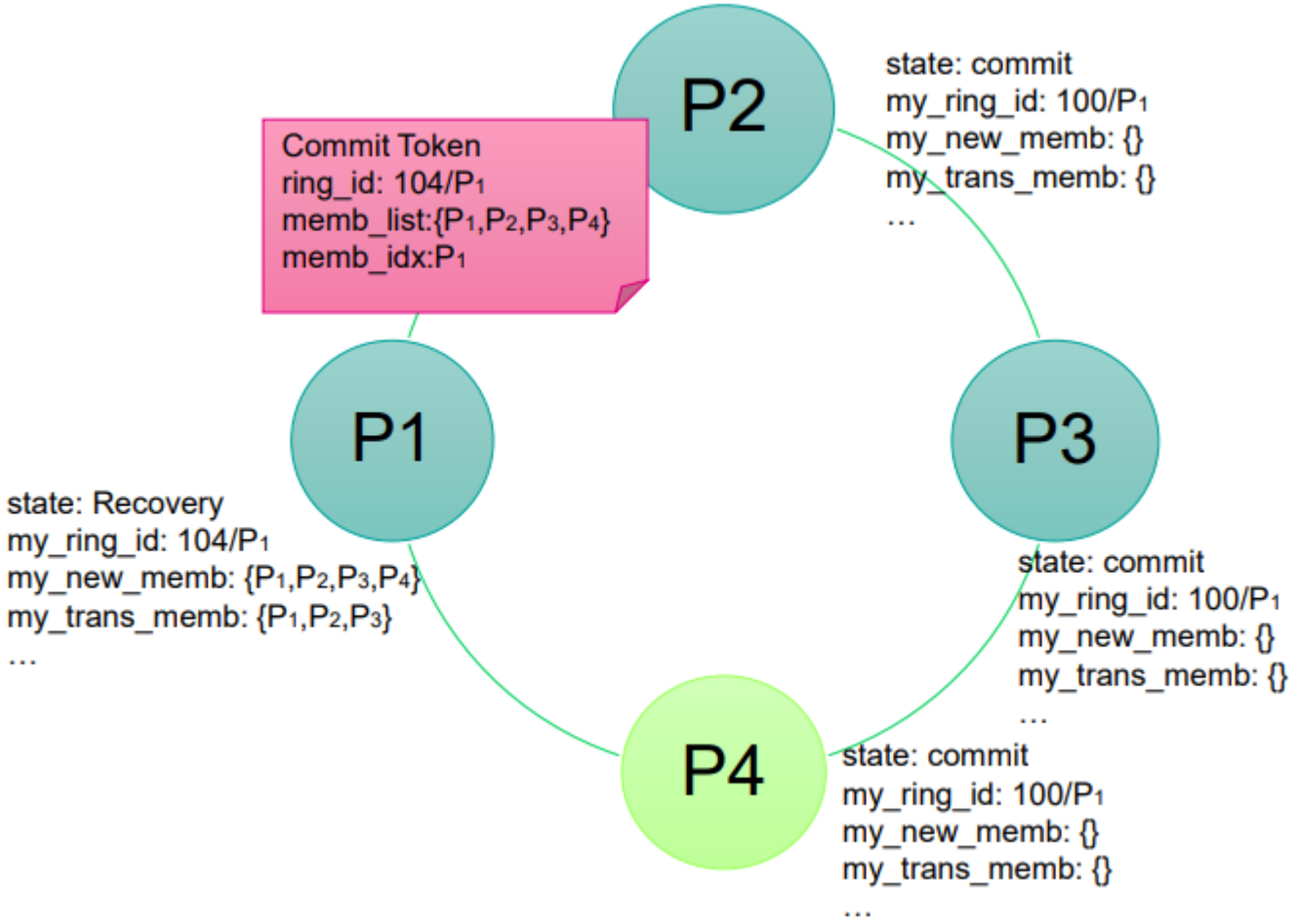
- P3 получает Commit токен от P2, обновляет member_list и memb_idx, передает токен и переходит в состояние Commit



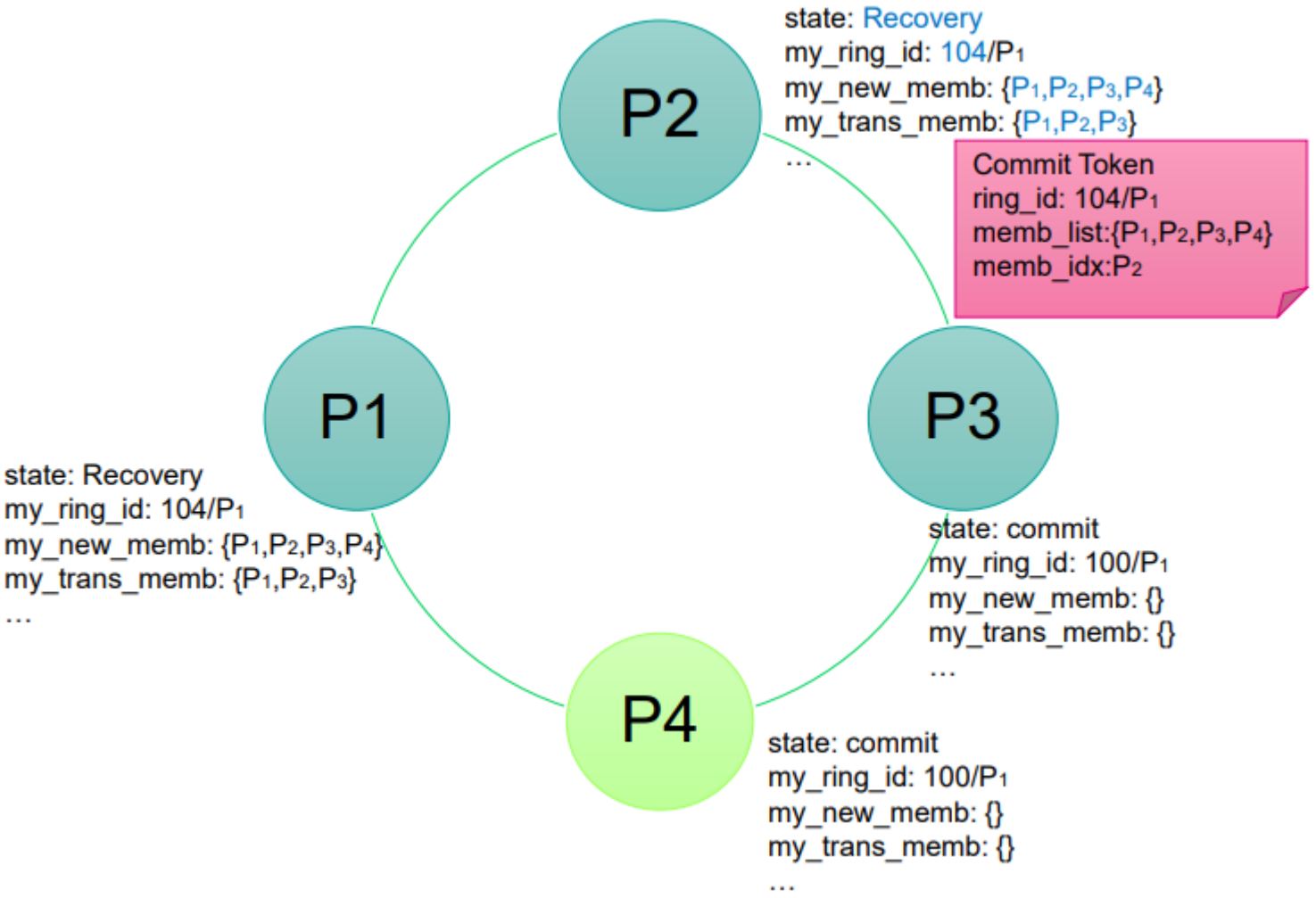
P4 получает Commit токен от P3, обновляет member_list и memb_idx, передает токен и переходит в состояние Commit



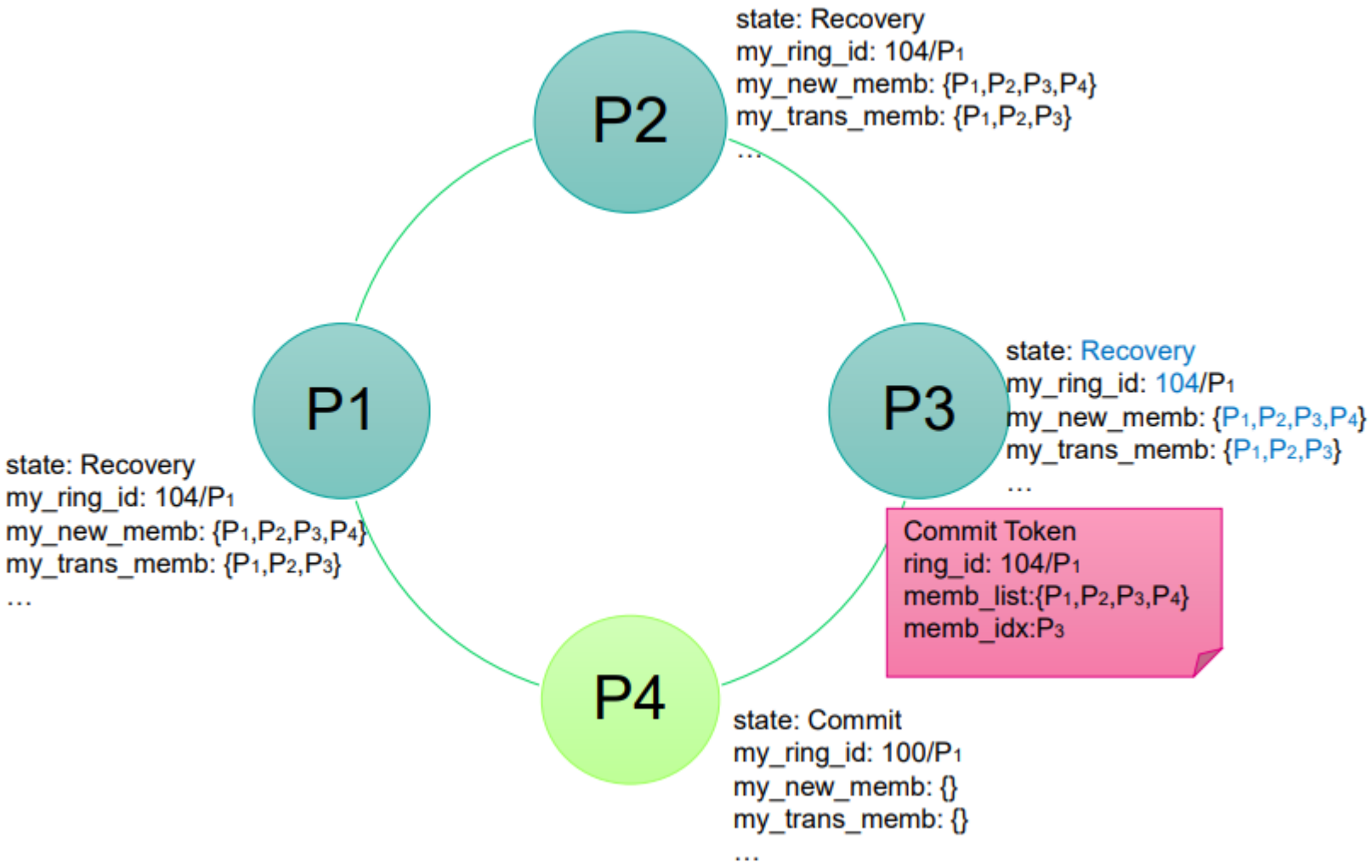
- P1 получает Commit токен от P4, так как P1 уже в состояние Commit, он понимает, что и все остальные тоже в состояние Commit
- P1 снова передает Commit токен и переходит в состояние Восстановления



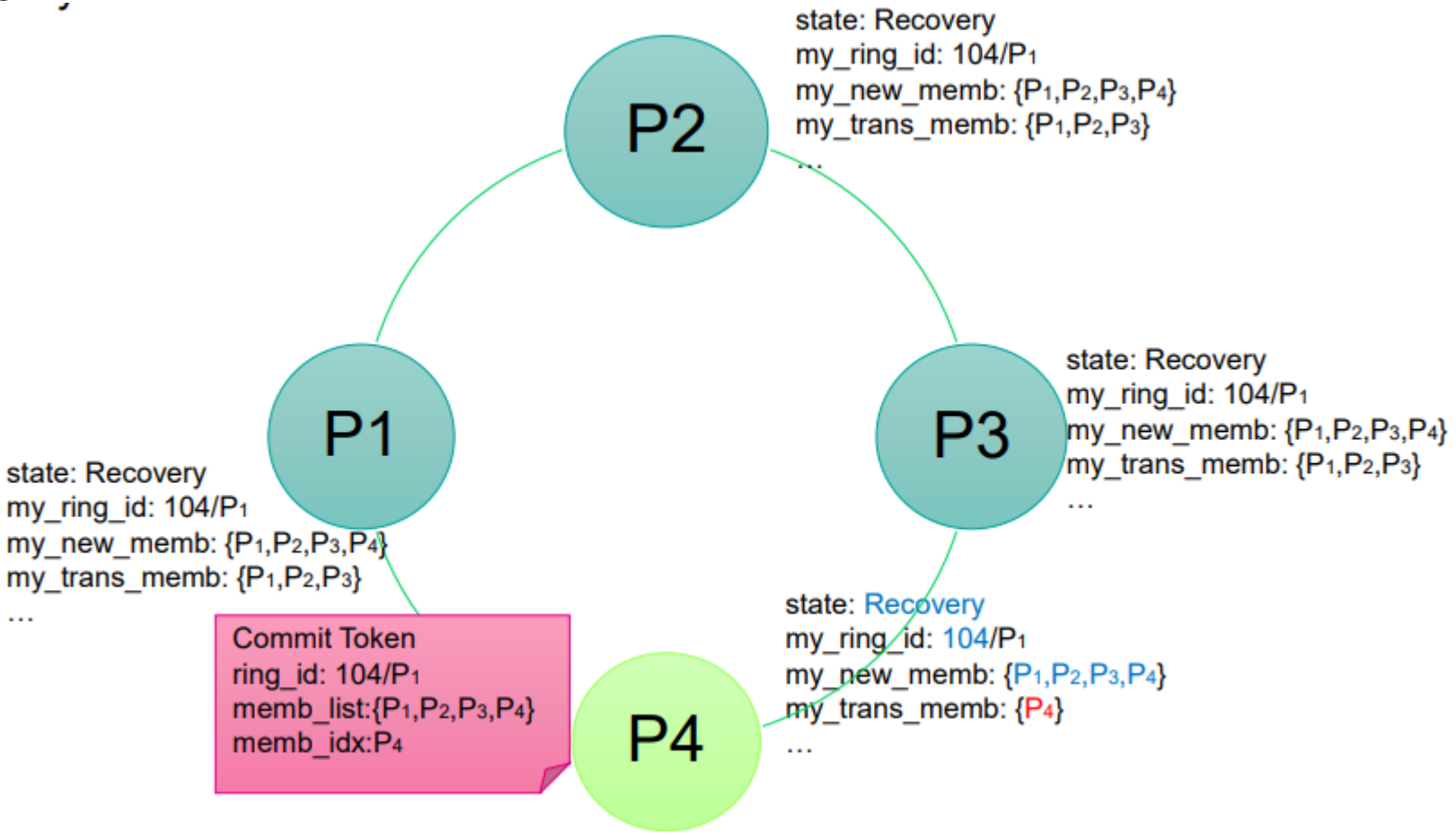
- P2 снова передает Commit токен и переходит в состояние Восстановления



- P3 снова передает Commit токен и переходит в состояние Восстановления
- Также P3 устанавливает поле ring_id (my_ring_id=CommitToken’s ring_id)



- P4 снова передает Commit токен и переходит в состояние Восстановления
- P4 — новый член, поэтому он один в my_trans_memb
- Когда P1 получает Commit токен в третий раз, все процессоры находятся в состоянии Восстановления



Протокол Восстановление (The Recovery Protocol)

- Действует в состоянии Восстановления (Recovery)
- Переход от старого кольца к новому, восстанавливает сообщения из старого кольца, чтобы сохранился согласованный или безопасный порядок
- В состояние восстановления нельзя отправлять broadcast-сообщения приложениям

Протокол Восстановления

- Шаг 1
  - Обмен сообщениями с процессорами внутри старого кольца
- Шаг 2
  - Доставка сообщений приложениям в соответствии с порядком указанным в старой конфигурации (согласованный или безопасный порядок)
- Шаг 3
  - Доставка первого конфигурационного сообщения, оно содержит участников старого кольца, которые остались в новом
- Шаг 4
  - Доставка сообщений, которые находятся в согласованном или безопасном порядке
- Шаг 5
  - Доставка второго конфигурационного сообщения приложениям, оно содержит участников нового кольца
- Шаг 6
  - Переход в состояние Operational
  - На шаге 2 — 6 не нужно обмениваться сообщениями с другими процессорами, это атомарная операция

Totem протокол с избыточным кольцом

- Основан на Totem протоколе с одним кольцом
- Более надежен даже для отключенного узла с помощью настройки дополнительного сетевого интерфейса

Totem протокол с избыточным кольцом

- Активное копирование
  - Все сообщения передаются по N каналам
  - Каждое сообщение получено N раз
  - Чем больше каналов (больше N), тем выше стоимость пропускной способности
- Пассивное копирование
  - Каждое сообщение передается по одному из N каналов
  - Каждое сообщение получено N раз
  - Пропускная способность такая же, как и одного кольца
- Активно-пассивное копирование
  - Сообщения передаются по K каналам(1 < K < N)