Программирование распределенных систем

Роман Елизаров, 2022

elizarov@gmail.com

Лекция 3

УПОРЯДОЧИВАНИЕ СООБЩЕНИЙ

Упорядочивание сообщений

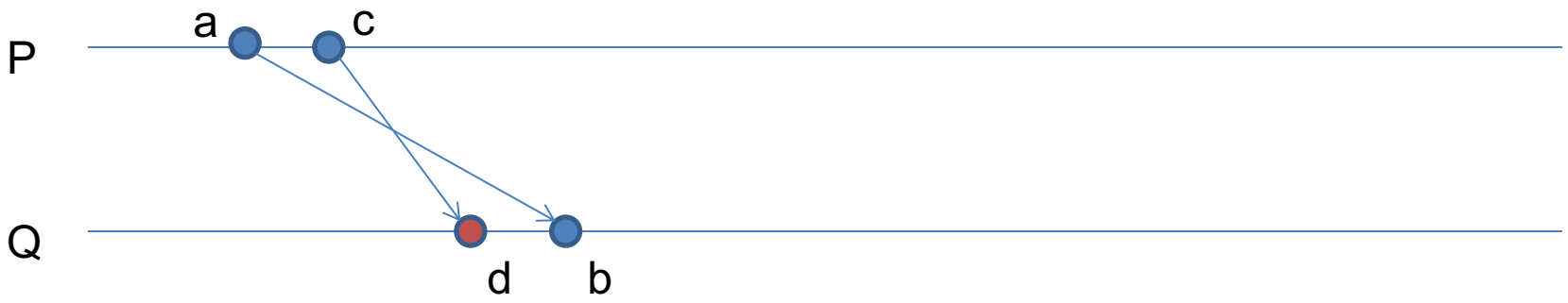
- Для обычных сообщений между парой процессов
  - Асинхронная передача (нет порядка)
  - FIFO
  - Причинно-согласованный порядок
  - Синхронный порядок
- Для multicast и broadcast сообщений
  - Общий порядок

FIFO порядок
(First In First Out order)

- Не бывает пар сообщений m и n упорядоченных неверно!

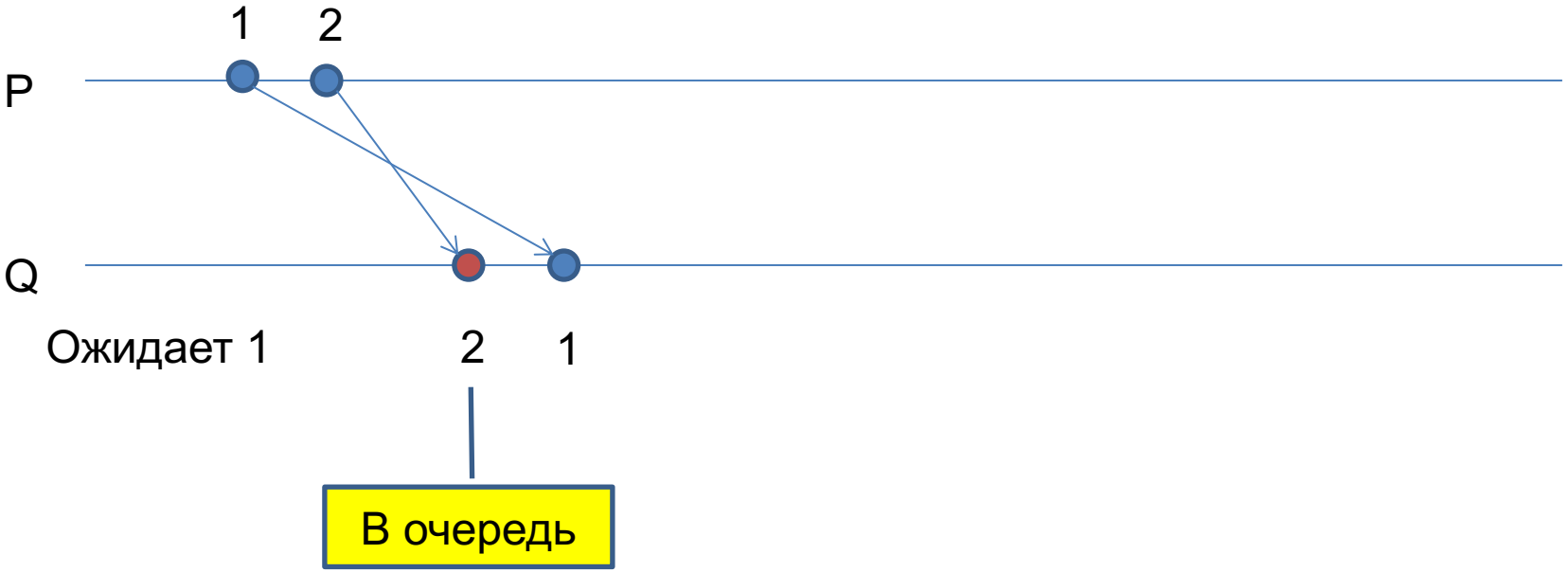
$$\nexists m, n \in M : snd(m) < snd(n) \;\&\; rcv(n) < rcv(m)$$

- Здесь “<“ это отношение порядка в одном процессе
- Пример нарушения FIFO с $m = (a, b); n = (c, d)$



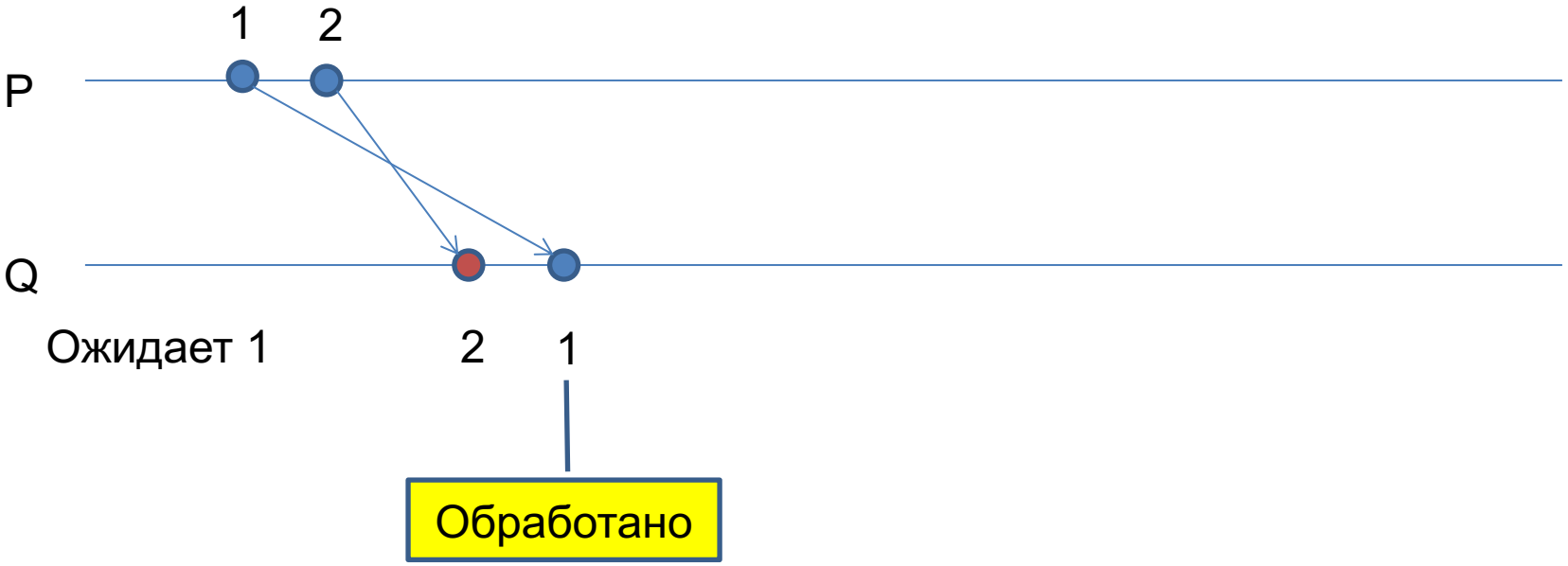
Алгоритм FIFO

- Алгоритм FIFO основан на нумерации сообщений
  - Нумеруем все сообщения между парой $P \rightarrow Q$
  - На стороне получатель есть «номер ожидаемого сообщения»
  - Условие обработки: номер пришедшего сообщения = ожидаемый
  - Сообщения пришедшие «неожиданно» складываются в очередь



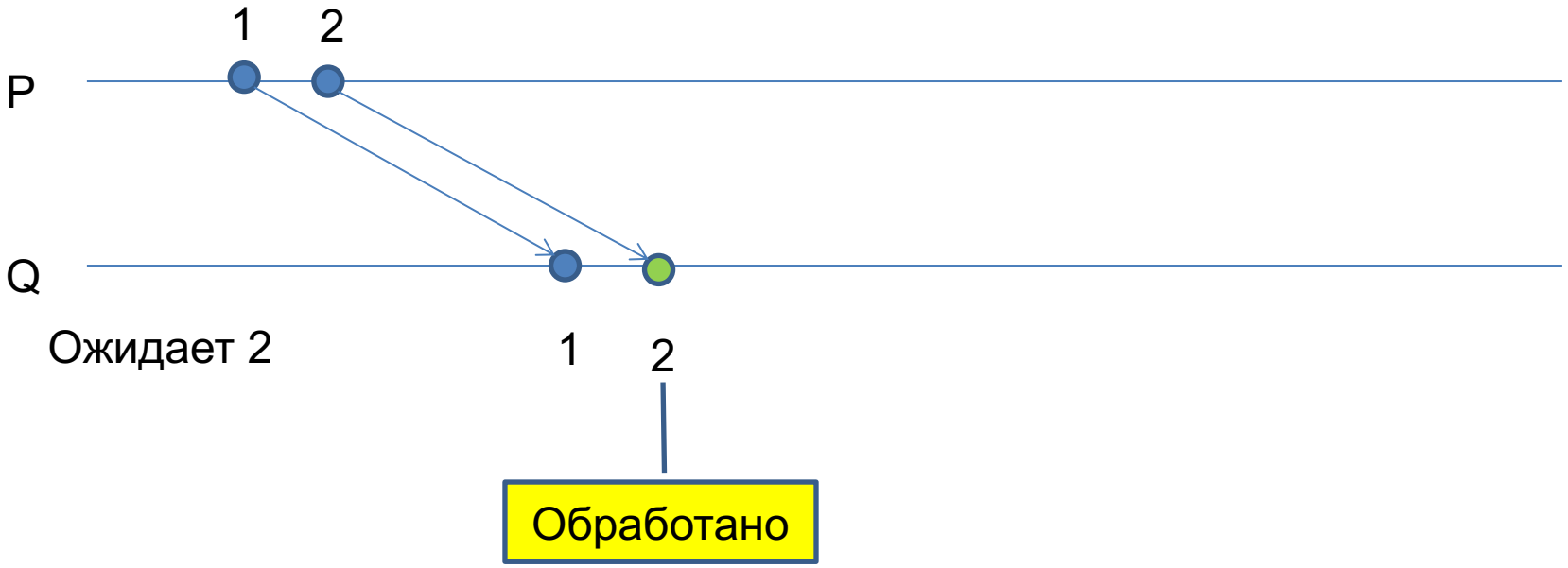
Алгоритм FIFO

- Алгоритм FIFO основан на нумерации сообщений
  - Нумеруем все сообщения между парой $P \rightarrow Q$
  - На стороне получатель есть «номер ожидаемого сообщения»
  - Условие обработки: номер пришедшего сообщения = ожидаемый
  - Сообщения пришедшие «неожиданно» складываются в очередь



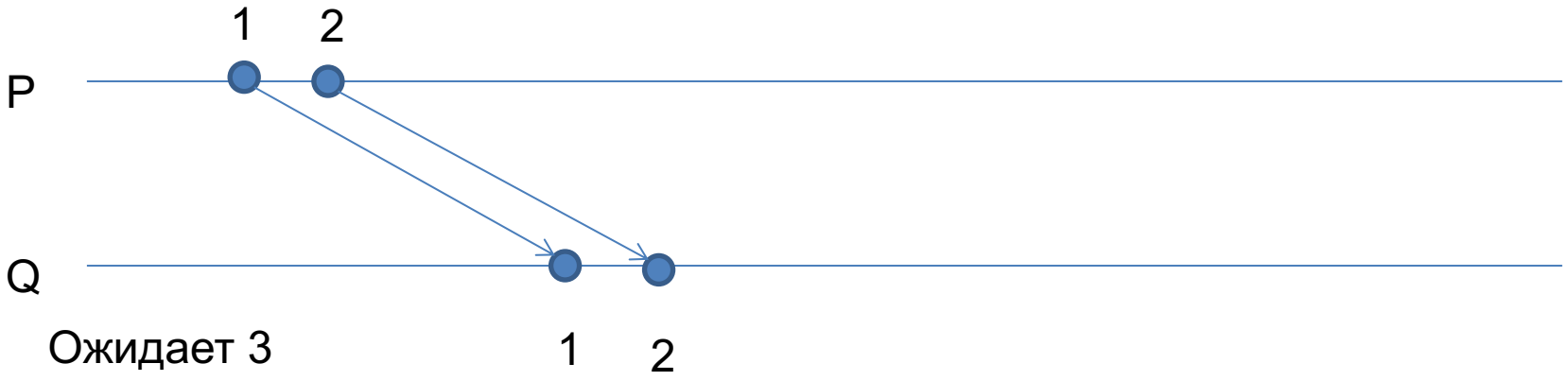
Алгоритм FIFO

- Алгоритм FIFO основан на нумерации сообщений
  - Нумеруем все сообщения между парой $P \rightarrow Q$
  - На стороне получатель есть «номер ожидаемого сообщения»
  - Условие обработки: номер пришедшего сообщения = ожидаемый
  - Сообщения пришедшие «неожиданно» складываются в очередь



Алгоритм FIFO

- Алгоритм FIFO основан на нумерации сообщений
  - Нумеруем все сообщения между парой $P \rightarrow Q$
  - На стороне получатель есть «номер ожидаемого сообщения»
  - Условие обработки: номер пришедшего сообщения = ожидаемый
  - Сообщения пришедшие «неожиданно» складываются в очередь

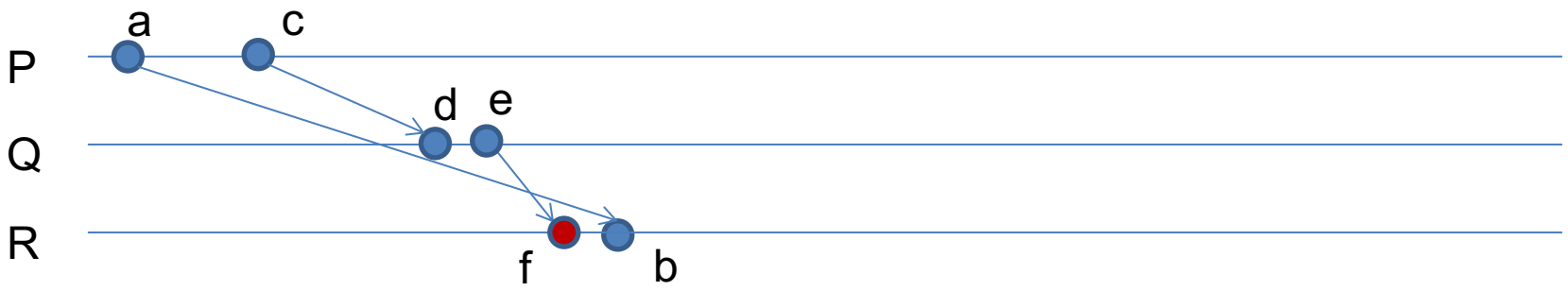


Причинно-согласованный порядок (casually consistent order)

- Не бывает пар сообщений m и n упорядоченных неверно!

$$\nexists m, n \in M: snd(m) \rightarrow snd(n) \,\&\, rcv(n) \rightarrow rcv(m)$$

- Здесь используется отношение «произошло до»
- Пример нарушения (FIFO не нарушен) с $m = (a, b);\ n = (e, f)$

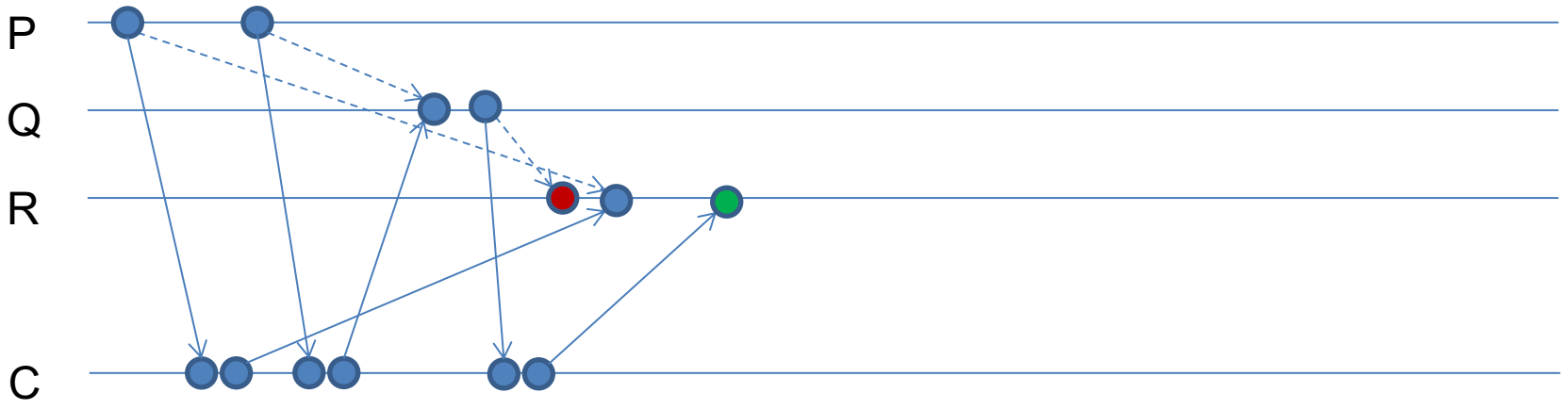


Централизованный алгоритм для причинно-согласованного порядка

- Не бывает пар сообщений m и n упорядоченных неверно!

$$\nexists m, n \in M: snd(m) \rightarrow snd(n) \,\&\, rcv(n) \rightarrow rcv(m)$$

- Выберем координатора, который будет осуществлять передачу (каналы до координатора FIFO)

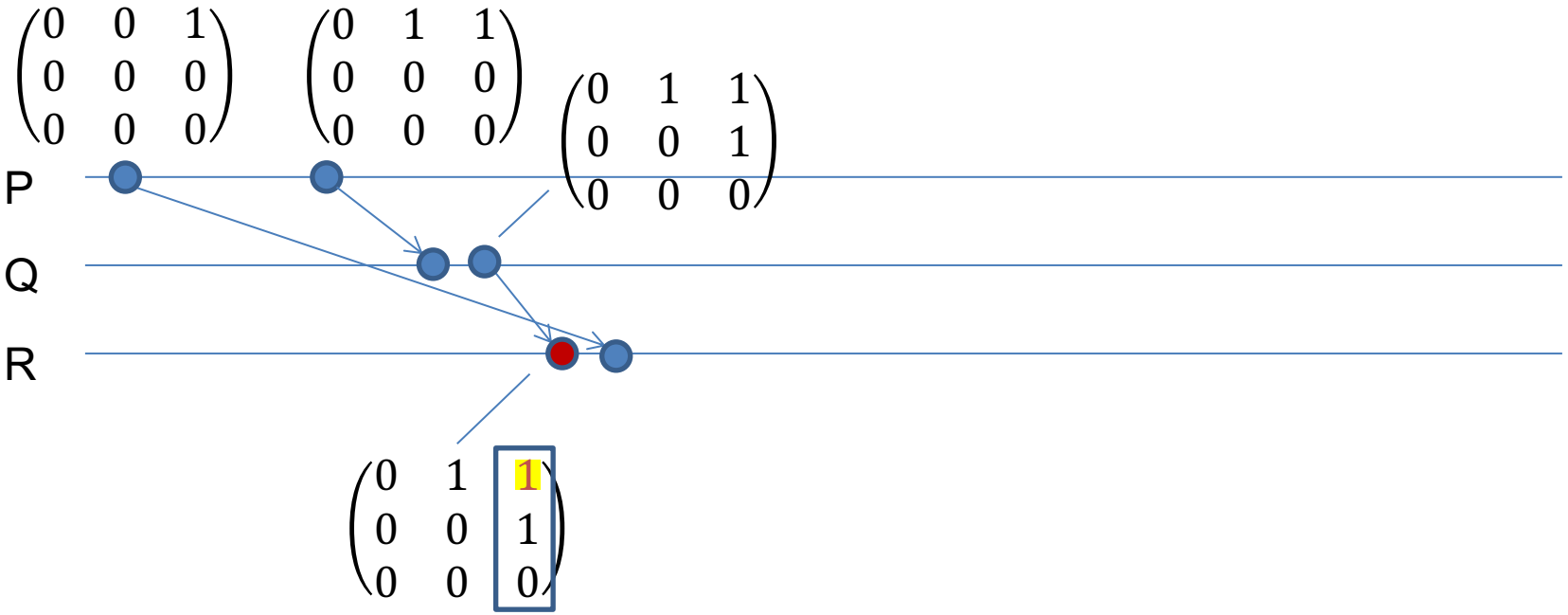


Распределенный алгоритм причинно-согласованного порядка

- Используем матричные часы
  - M_{ij} - количество сообщений посланных от P_i к P_j
  - Перед посылкой сообщения $M_{ij} = M_{ij} + 1$ и шлем матрицу M
- Условия обработки сообщения (все должны быть выполнены)
  - Если процесс P_i получил сообщение от P_j с матрицей W
  - FIFO: $W_{ji} = M_{ji} + 1$
  - Casual: $\forall k \neq j : M_{ki} \geq W_{ki}$
  - После обработки: $M = \max(M, W)$

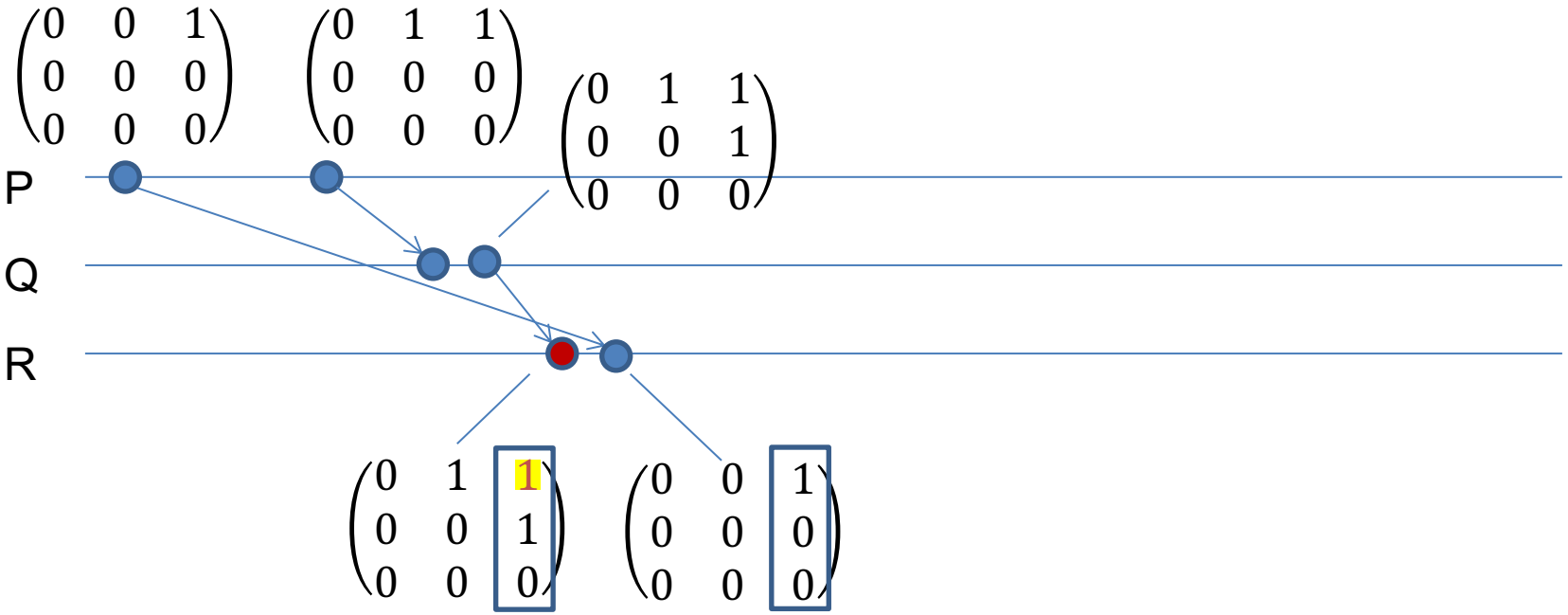
Пример причинно-согласованного порядка

- Условия обработки сообщения (все должны быть выполнены)
  - FIFO: $W_{ji} = M_{ji} + 1$
  - Casual: $\forall k \neq j : M_{ki} \geq W_{ki}$
  - После обработки: $M = \max(M, W)$



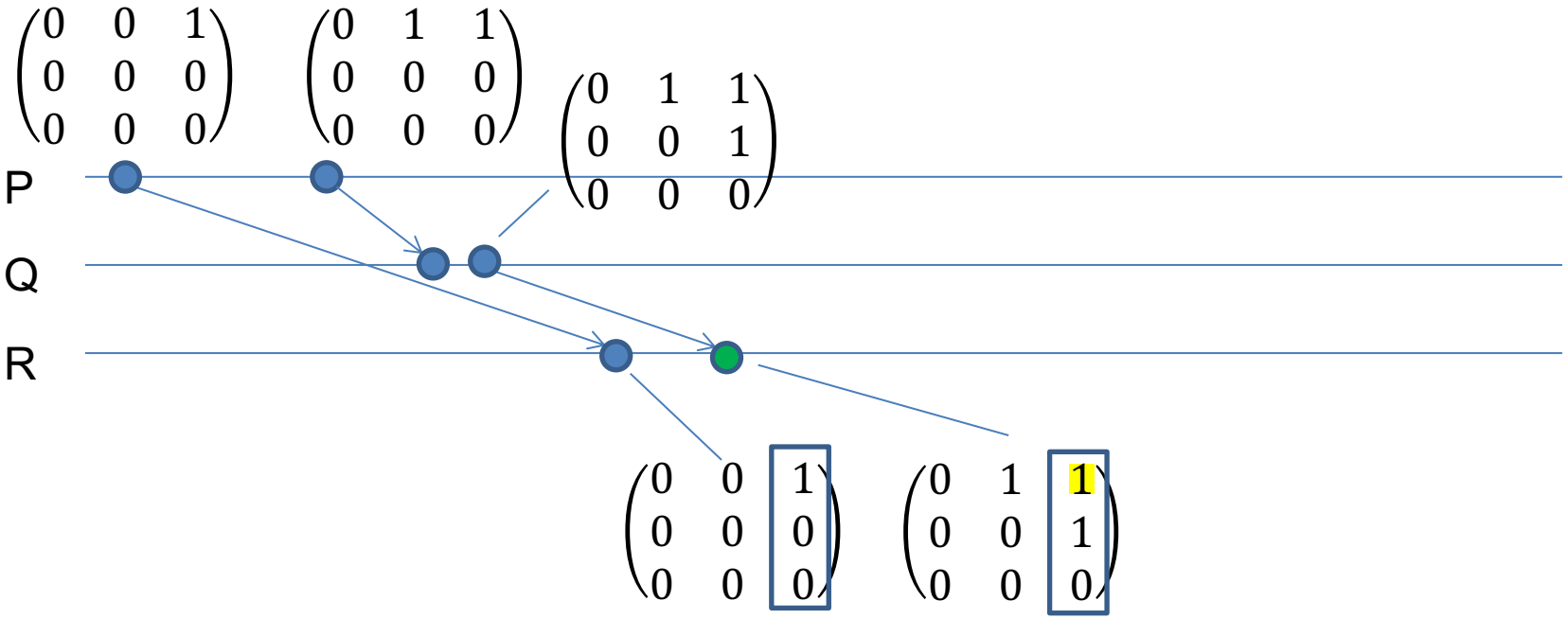
Пример причинно-согласованного порядка

- Условия обработки сообщения (все должны быть выполнены)
  - FIFO: $W_{ji} = M_{ji} + 1$
  - Casual: $\forall k \neq j : M_{ki} \geq W_{ki}$
  - После обработки: $M = \max(M, W)$



Пример причинно-согласованного порядка

- Условия обработки сообщения (все должны быть выполнены)
  - FIFO: $W_{ji} = M_{ji} + 1$
  - Casual: $\forall k \neq j : M_{ki} \geq W_{ki}$
  - После обработки: $M = \max(M, W)$

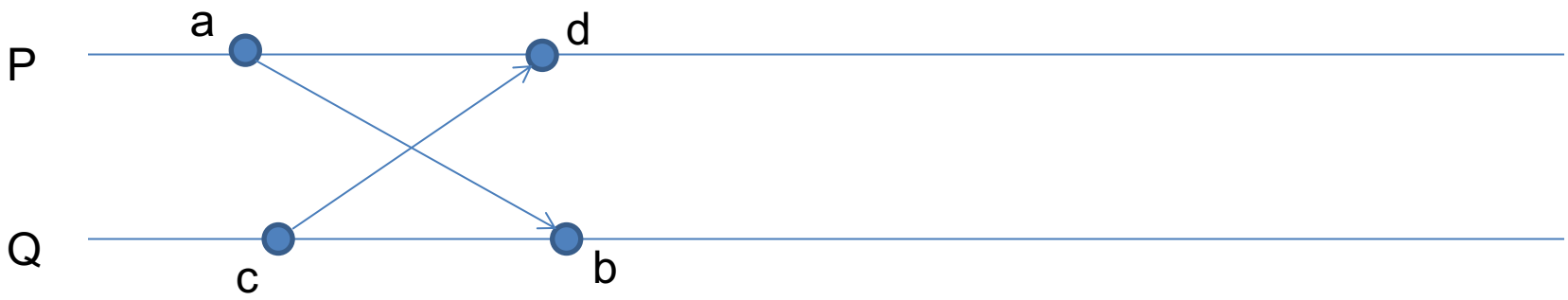


Синхронный порядок (synchronous order)

- Всем сообщениям можно сопоставить время $T(m)$ так что время событий $T\big(rcv(m)\big) = T\big(snd(m)\big) = T(m)$ и

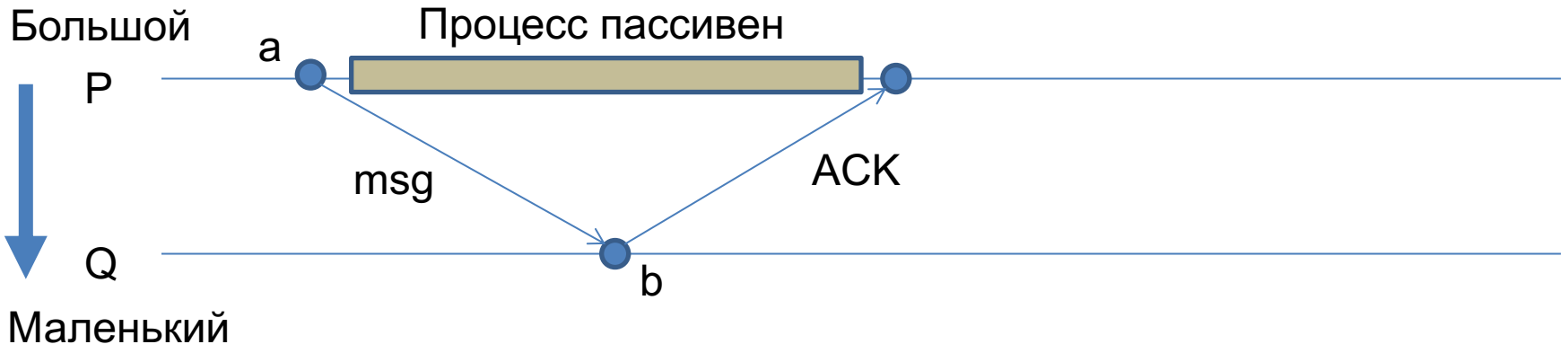
$$\forall \mathrm{e}, \mathrm{f} \in E : e \longrightarrow f \Longrightarrow T(e) < T(f)$$

- Пример нарушения синхронного порядка (причинно-согласованность не нарушена)



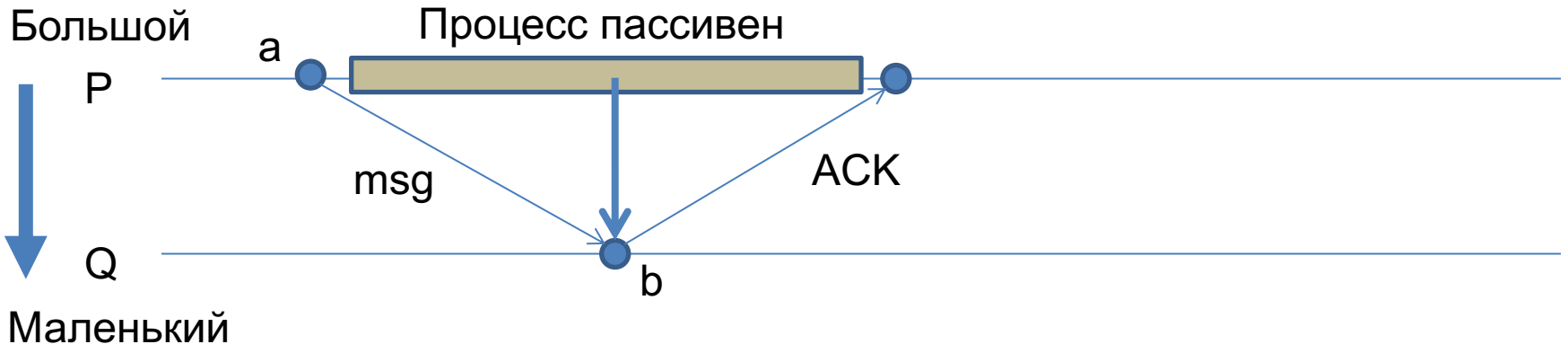
Алгоритм синхронного порядка

- Алгоритм основан на иерархии процессов
  - Процессы «большие» и «маленькие»
  - Большой шлет маленькому с подтверждением



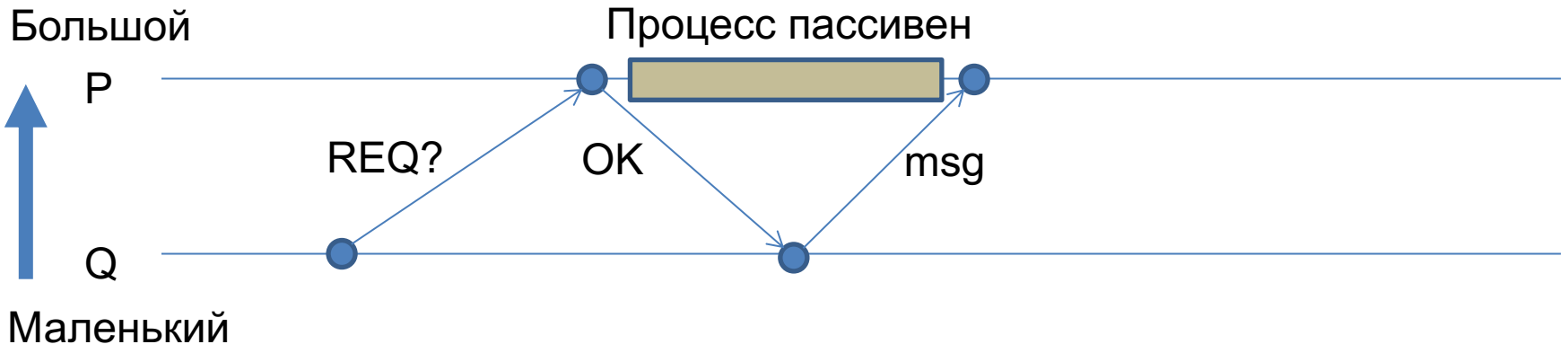
Алгоритм синхронного порядка

- Алгоритм основан на иерархии процессов
  - Процессы «большие» и «маленькие»
  - Большой шлет маленькому с подтверждением



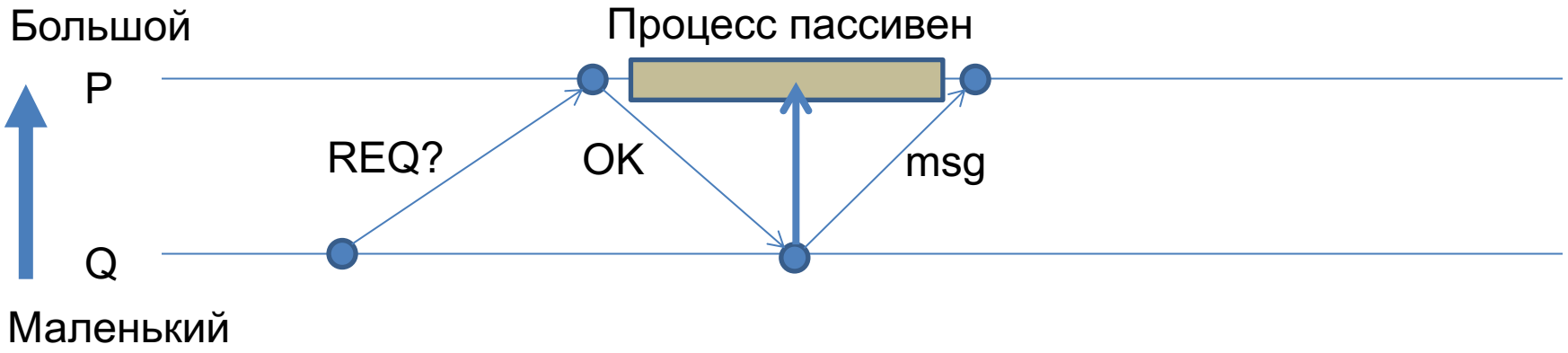
Алгоритм синхронного порядка

- Алгоритм основан на иерархии процессов
  - Процессы «большие» и «маленькие»
  - Маленький шлет большому только получив разрешение



Алгоритм синхронного порядка

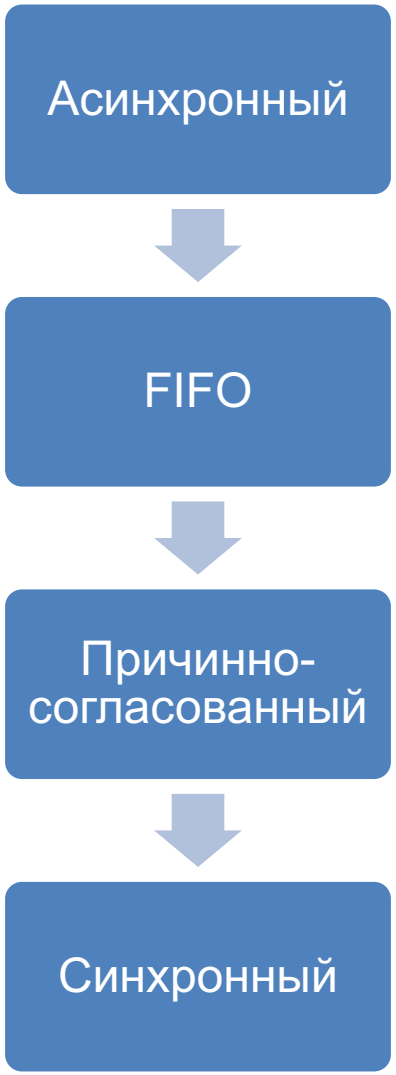
- Алгоритм основан на иерархии процессов
  - Процессы «большие» и «маленькие»
  - Маленький шлет большому только получив разрешение



Алгоритм синхронного порядка

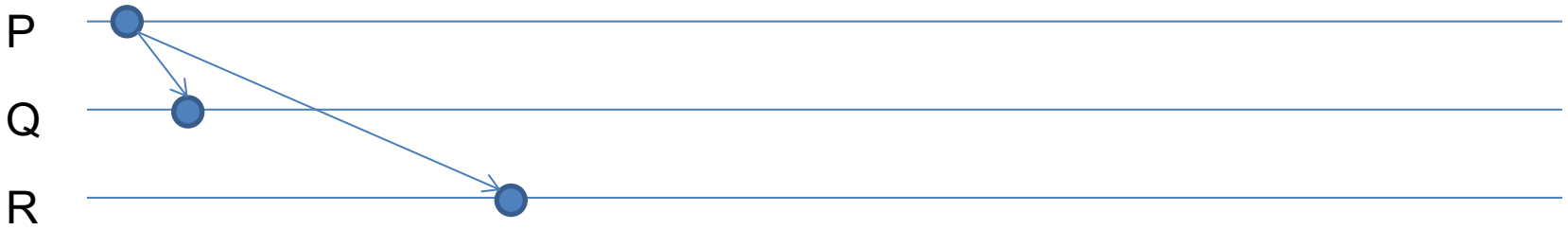
- Пассивный процесс не участвует в другой передаче сообщений
  - Поэтому «стрелки» передачи всегда можно расположить вертикально (см. иллюстрации)
- В системе не может произойти взаимная блокировка из-за иерархии процессов
  - при любой коммуникации пассивным становится больший процесс в иерархии.

Иерархия порядков



Массовая рассылка сообщений

- Broadcast – от одного всем другим процессам
- Multicast – от одного множеству других процессов

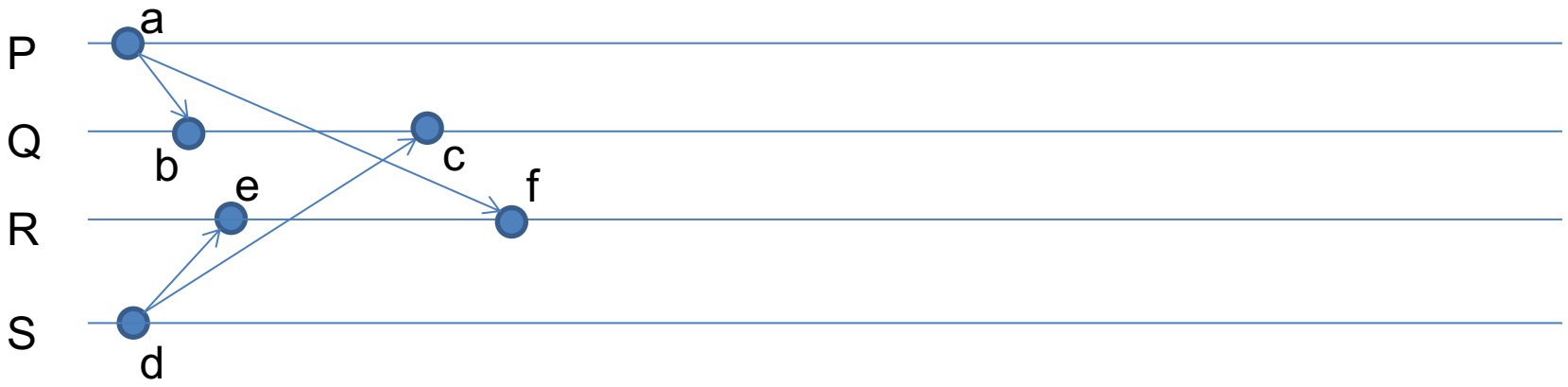


Общий порядок (total order)

- Только для случаем когда одно сообщение идет многим получателям (rcv_p с индексом процесса-получателя P)

$$\nexists m, n \in M; p, q \in P: rcv_p(m) < rcv_p(n) \;\&\; rcv_q(n) < rcv_q(m)$$

- Тривиально выполняется для всех unicast сообщений
- Пример нарушения (FIFO не нарушен!)

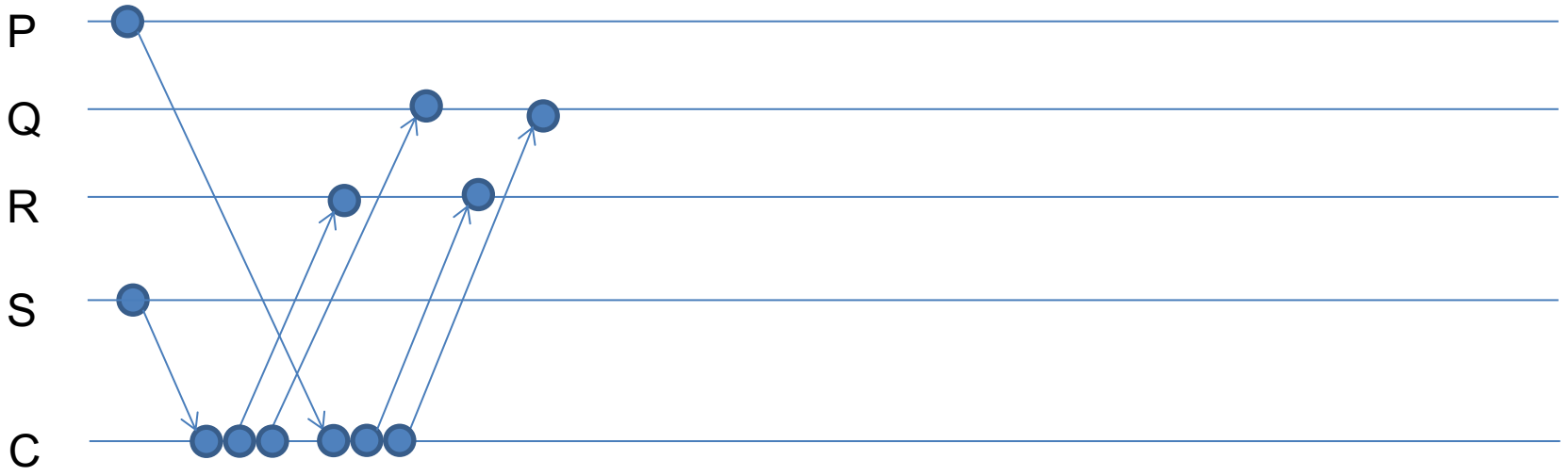


Централизованный алгоритм общего порядка

- Только для случаем когда одно сообщение идет многим получателям (rcv_p с индексом процесса-получателя P)

$$\nexists m, n \in M; p, q \in P: rcv_p(m) < rcv_p(n) \;\&\; rcv_q(n) < rcv_q(m)$$

- Выберем координатора для рассылки сообщений (работает при условии что каналы от координатора FIFO)

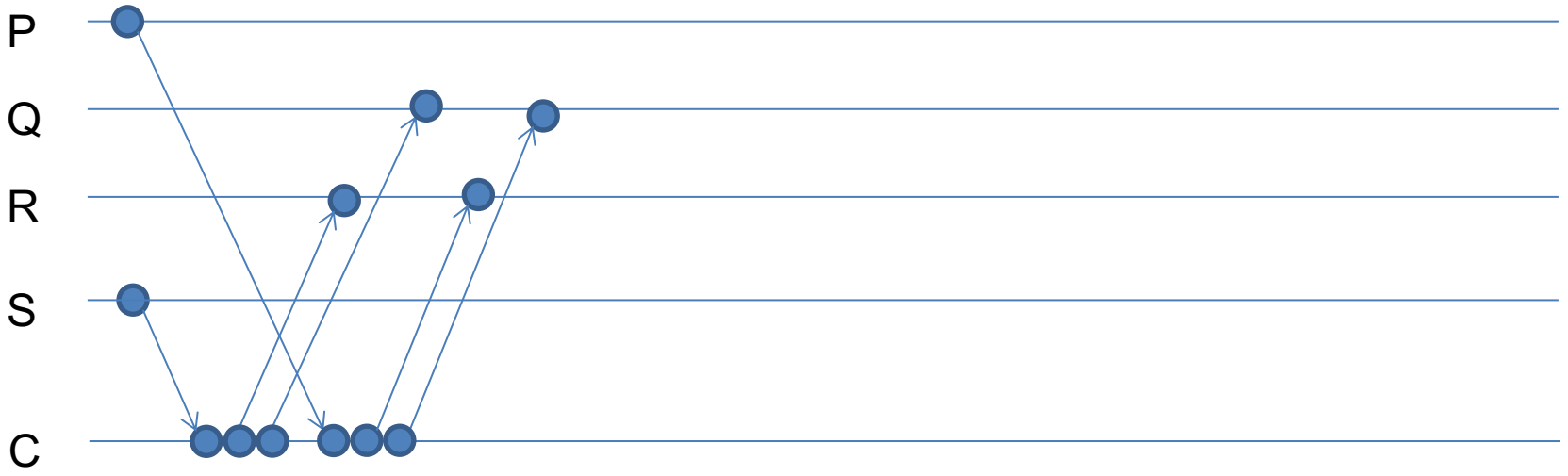


Централизованный алгоритм общего порядка

- Только для случаем когда одно сообщение идет многим получателям (rcv_p с индексом процесса-получателя P)

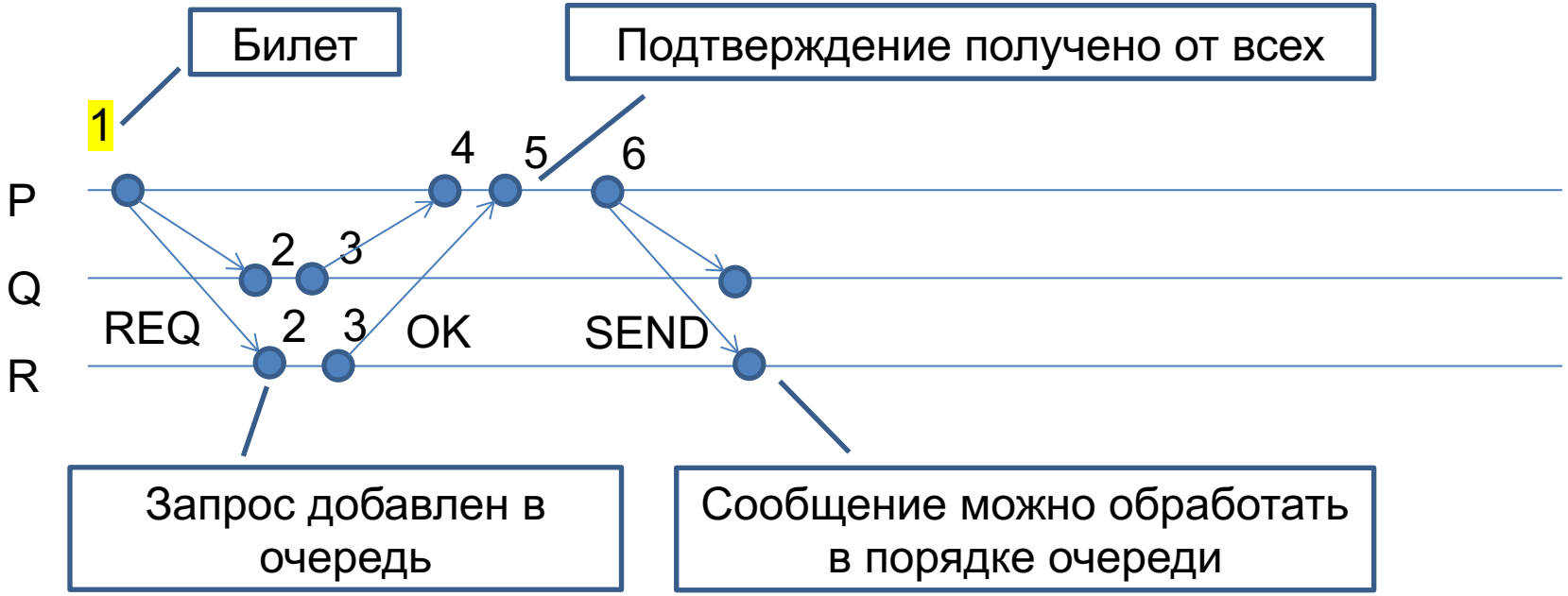
$$\nexists m, n \in M; p, q \in P: rcv_p(m) < rcv_p(n) \;\&\; rcv_q(n) < rcv_q(m)$$

Централизованный алгоритм также обеспечивает причинно-согласованный порядок



Алгоритм Лампорта для общего порядка

- Это же работает как взаимная блокировка!
  - Обобщаем алгоритм взаимной блокировки Лампорта
  - Ведем логические часты, «билетик» задает порядок обработки сообщений
- Особенности: Нужен FIFO, надо посылать сообщение всем (даже если multicast)



Алгоритм Скина для общего порядка

- Оптимизируем алгоритм Лампорта
  - Номер билета вычисляется только при получении подтверждения
  - Работает не только для broadcast (шлем всем) но и для multicast (шлем подмножеству); не нужен FIFO