ITIVITI

LET IT TRADE

Распределенные вычисления

Лекция 5-6

Алгоритмы взаимного исключения

УНИВЕРСИТЕТ ИТМО

Общие концепции

ITIVITI

- **Критическая секция** - часть программы, в которой есть обращения к совместно используемым ресурсам (структурам данных или устройствам), работа с которыми требует эксклюзивного доступа
- **Взаимное исключение** – требование, согласно которому во время выполнения критической секции одного процесса ни один другой процесс не должен выполняться в этой же критической секции (имеющей такой же идентификатор)

LET IT TRADE

УНИВЕРСИТЕТ ИТМО

Общие концепции

ITIVITI

- Отсутствие общей памяти в распределенных системах не позволяет использовать разделяемые переменные (такие как семафоры)
- Единственное средство – обмен сообщениями
- Произвольные задержки передачи сообщений и отсутствие полной информации о состоянии всей системы
- Относительная скорость выполнения процессов и их число произвольны

LET IT TRADE

УНИВЕРСИТЕТ ИТМО

Предположения

ITIVITI

- Все процессы выполняются и взаимодействуют асинхронно
- Процессы не выходят из строя
- Каналы являются надежными и коммуникационная сеть не разбивается на несвязанные друг с другом части
- Требования к свойству очередности каналов являются специфичными для каждого алгоритма
- Одна критическая секция

LET IT TRADE

УНИВЕРСИТЕТ ИТМО

Структура кода с критической секцией

P_1

```
void P1(int R)
{
  while(true) {
  /* предшествующий код */;
  request_cs(R);
  /* критическая секция */;
  release_cs(R);
  /* последующий код */;
  }
}
```

...

P_N

```
void PN(int R)
{
  while(true) {
  /* предшествующий код */;
  request_cs(R);
  /* критическая секция */;
  release_cs(R);
  /* последующий код */;
  }
}
```

1) критическая секция работает с ресурсом, идентификатором которого является число R
2) предшествующая и последующая часть кода, где нет обращений к ресурсу R
3) R – идентификатор ресурса или критической секции

Состояния процесса

ITIVITI

- Запрос на вход в критическую секцию
  - блокируется до получения разрешения на вход в CS и не должен формировать новых запросов на вход в CS
- Выполнение внутри критической секции
  - находится в течение ограниченного времени
- Выполнение вне критической секции
  - может находится сколь угодно долго
- Переходы процесса из состояния в состояние

LET IT TRADE

УНИВЕРСИТЕТ ИТМО

Требования к алгоритмам взаимного исключения

ITIVITI

- Безопасность
  - В каждый момент времени в CS может выполняться не более одного процесса
- Живучесть
  - отсутствие взаимоблокировок (бесконечного ожидания прихода сообщения, которое никогда не придет)
  - отсутствие голодания (бесконечного ожидания входа в CS, в то время как другие процессы неоднократно получают доступ к CS) = справедливость
- Справедливость
  - порядок вхождения в CS (в порядке «запрос произошел раньше другого» или нет)

Подходы к решению задачи взаимного исключения

ITIVITI

- На основе получения разрешений
  - Безопасность. Путем получения «достаточного числа» разрешений
  - Живучесть. С помощью отметок времени запросов, с помощью ациклического графа предшествования
- На основе передачи маркера (*token*)
  - Безопасность. Материализация права войти в CS в виде уникального объекта – маркера
  - Живучесть. Управление перемещением маркера: perpetuum mobile, по запросу на получение (метод распространения запросов для достижения маркера)
- Вырожденный случай – центр. алгоритм

LET IT TRADE

УНИВЕРСИТЕТ ИТМО

Централизованные vs распределенные алгоритмы

ITIVITI

- Централизованные алгоритм
  - основная часть работы выполняется одним процессом, остальные процессы играют незначительную роль в достижении общего результата
  - обычно это – получение информации от основного процесса или предоставление ему нужных данных
  - ассиметричный: различные процессы исполняют логически разные операции
  - модель клиент-сервер
- Распределенный алгоритм
  - каждый процесс играет одинаковую роль в достижении общего результата и в разделении общей нагрузки
  - симметричный: все процессы исполняют один и тот же код (логические функции)

Централизованный алгоритм взаимного исключения

- Один из процессов выбирается координатором
- Когда процесс собирается войти в КС, он посылает координатору сообщение REQUEST
- Если ни один из процессов не находится в этой КС, координатор посылает разрешение REPLY
- Процесс входит в КС после получения ответа
- Если доступ предоставлен другому процессу координатор либо не отвечает, либо отправляет «доступ запрещен» - в чем разница?

Централизованный алгоритм взаимного исключения

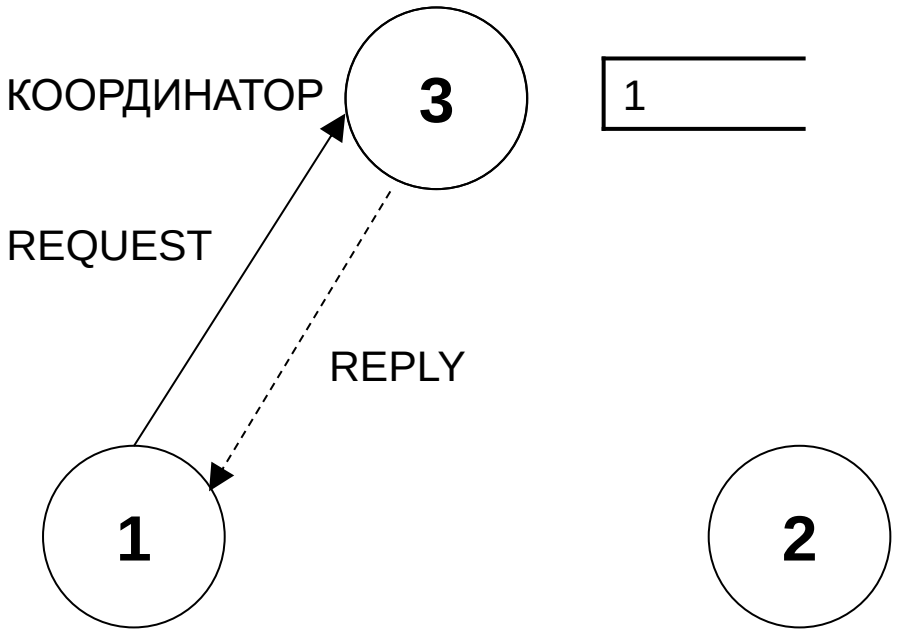
- Координатор ставит запрос в очередь и ожидает дальнейших сообщений
- Поступающие запросы помещаются в очередь координатора согласно порядку их поступления
- Когда процесс покидает КС, он посылает координатору сообщение RELEASE
- Координатор выбирает первый элемент из очереди отложенных запросов и посылает соответствующему процессу разрешающее сообщение

Централизованный алгоритм взаимного исключения

ITIVITI

LET IT TRADE

Пример работы централизованного алгоритма:

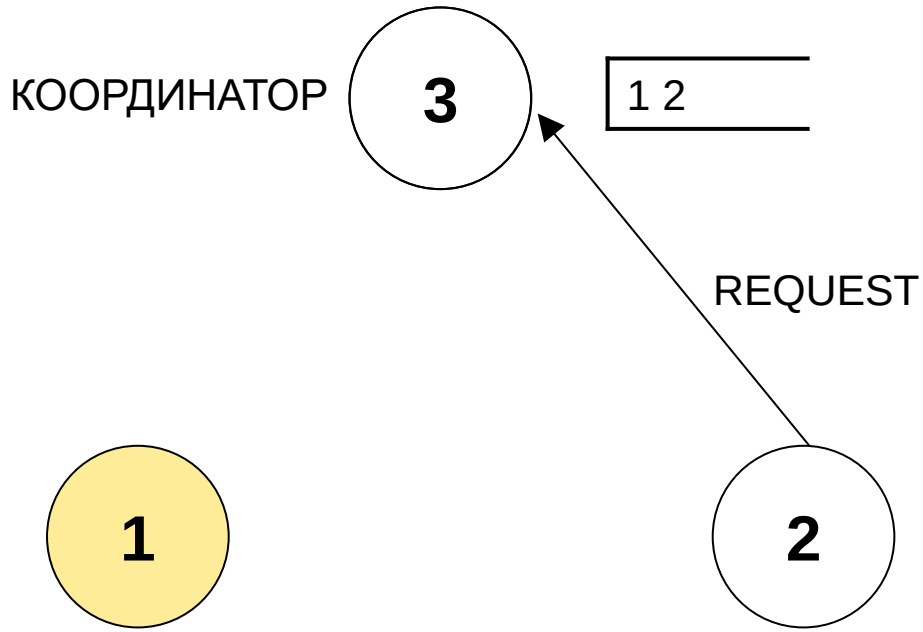


процесс 1 запрашивает вход в КС и получает ответ

УНИВЕРСИТЕТ ИТМО

Централизованный алгоритм взаимного исключения

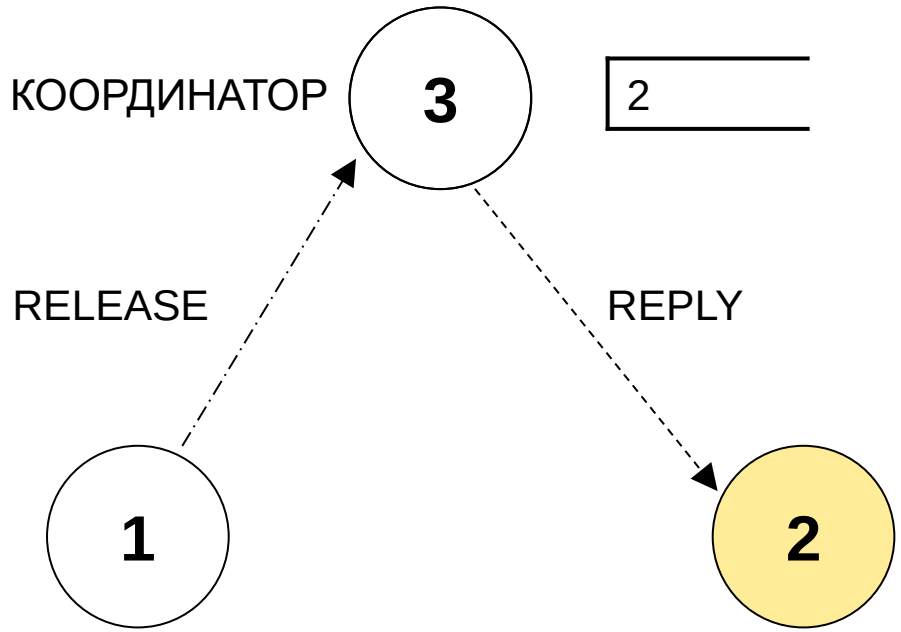
Пример работы централизованного алгоритма:



процесс 1 входит в КС; процесс 2 запрашивает вход в КС;
координатор помещает запрос процесса 2 в очередь;
процесс 2 блокируется

Централизованный алгоритм взаимного исключения

Пример работы централизованного алгоритма:



процесс 1 выходит из КС;
координатор отвечает процессу 2;
процесс 2 разблокируется и входит в КС

Свойства централизованного алгоритма

ITIVITI

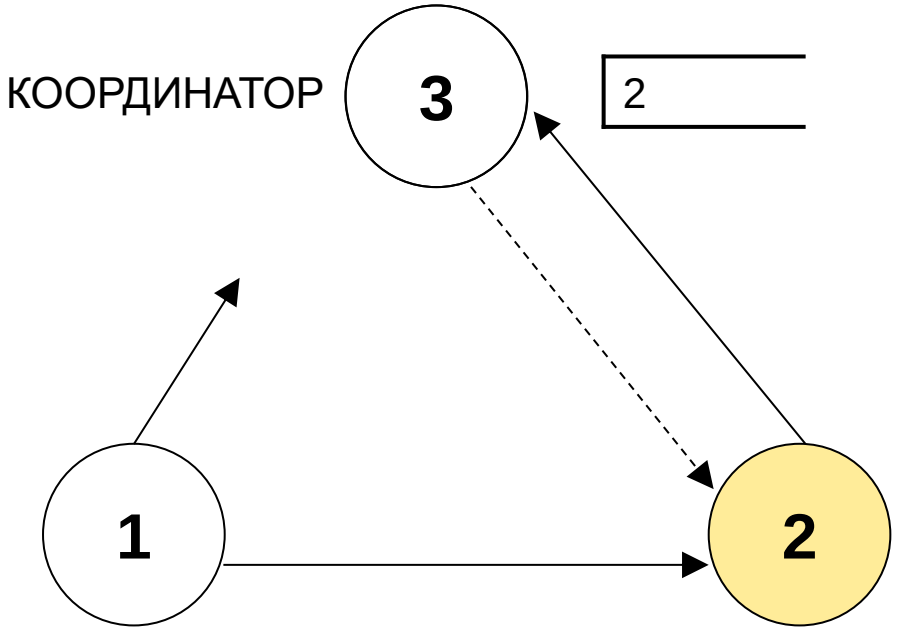
LET IT TRADE

- Безопасность?
- Живучесть?
- Справедливость?

УНИВЕРСИТЕТ ИТМО

Централизованный алгоритм взаимного исключения

Не совсем справедливость:



процесс 1 запрашивает вход в КС и отправляет сообщение процессу 2;
после получения сообщения процесс 2 запрашивает вход в КС;
запрос процесса 2 достигает координатора раньше запроса процесса 1

Алгоритм распределенной очереди Лэмпорта

ITIVITI

LET IT TRADE

Предположения:

- Каналы должны обладать свойством FIFO
- Каждое сообщение доставляется в течение конечного времени
- Сеть является полносвязной

УНИВЕРСИТЕТ ИТМО

Алгоритм распределенной очереди Лэмпорта

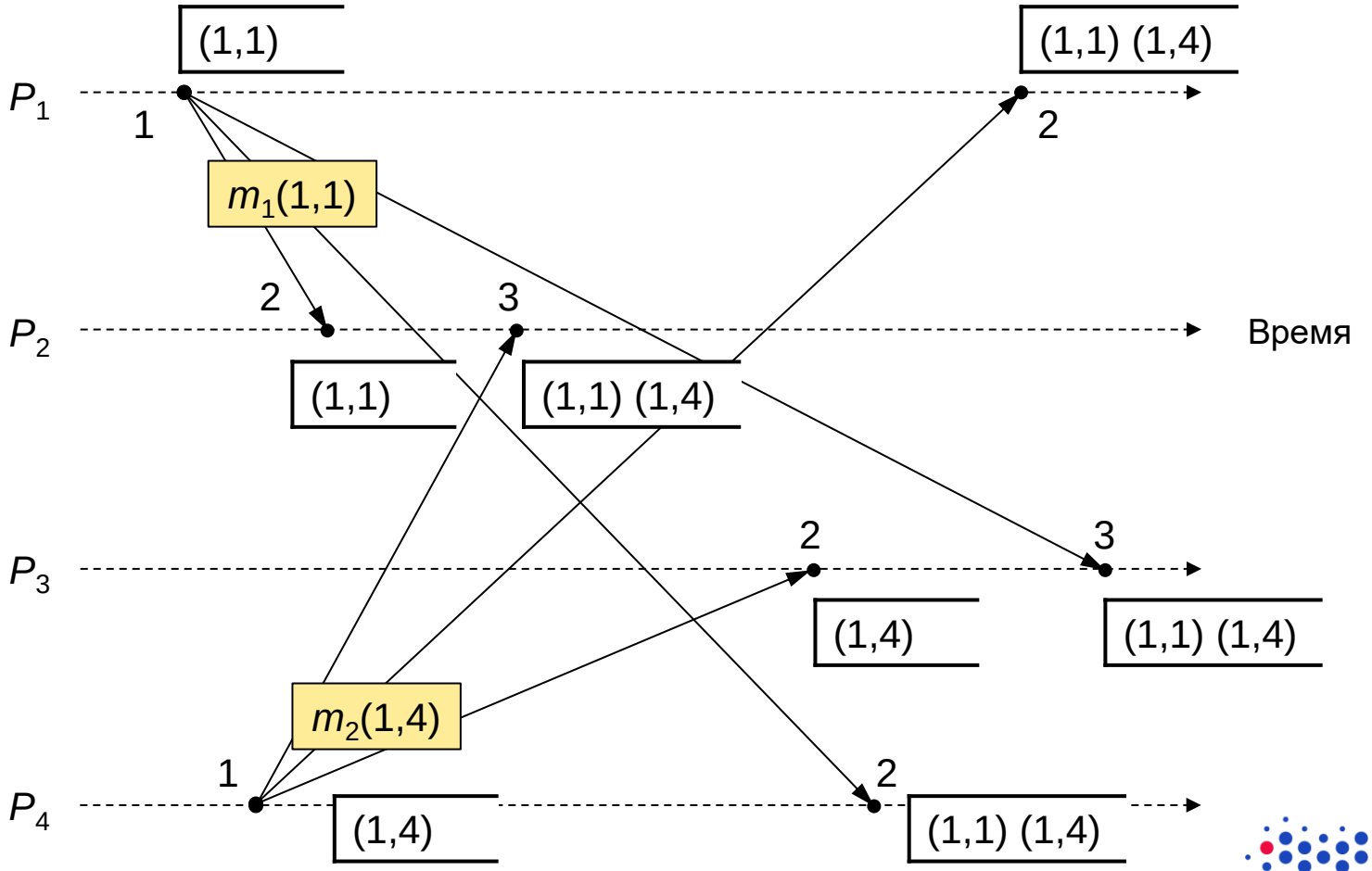
ITIVITI

- Иллюстрация линейного упорядочивания операций (вход в КС, выход из КС)
- Попытка обобщить централизованный алгоритм (очередь – отдельный объект, на всех узлах – копия этого объекта)
- Чтобы порядок запросов в очереди совпадал у всех процессов, запросам присваиваются отметки скалярного времени
- Сложность: есть угроза, что у двух различных процессов в начале очереди будут находиться разные запросы – нарушение взаимного исключения!

LET IT TRADE

УНИВЕРСИТЕТ ИТМО

Пример формирования распределенной очереди



Алгоритм распределенной очереди Лэмпорта

ITIVITI

LET IT TRADE

- **Вывод:**
  Прежде чем процесс, исходя из состояния своей собственной очереди, примет решение о входе в КС, он должен получить от всех других процессов сообщение, гарантирующее, что в каналах не осталось ни одного сообщения, которое может занять первое место в очереди

УНИВЕРСИТЕТ ИТМО

Алгоритм Лэмпорта: Запрос на вход в КС

ITIVITI

LET IT TRADE

- Q_i – очередь процесса P_i в которой хранятся все запросы на доступ к КС в порядке их отметок времени
- Когда P_i нужен доступ к КС, он отправляет REQUEST(L_i, i) всем остальным и помещает это сообщение в свою очередь Q_i
- Когда P_j получает REQUEST(L_i, i) от P_i, он помещает его в свою очередь Q_j и отправляет P_i REPLY(L_j, j) со значением своего времени (L_j, j)

УНИВЕРСИТЕТ ИТМО

Алгоритм Лэмпорта: Вход в КС

- P_i может войти в КС, если соблюдаются два условия:
- 1) REQUEST(L_i, i) процесса P_i является первым в его собственной очереди Q_i
- 2) P_i получил сообщения от всех остальных процессов с отметкой времени большей, чем (L_i, i). Соблюдение этого правила гарантирует, что P_i известно обо всех запросах, предшествующих его текущему запросу

Алгоритм Лэмпорта: Выход из КС

ITIVITI

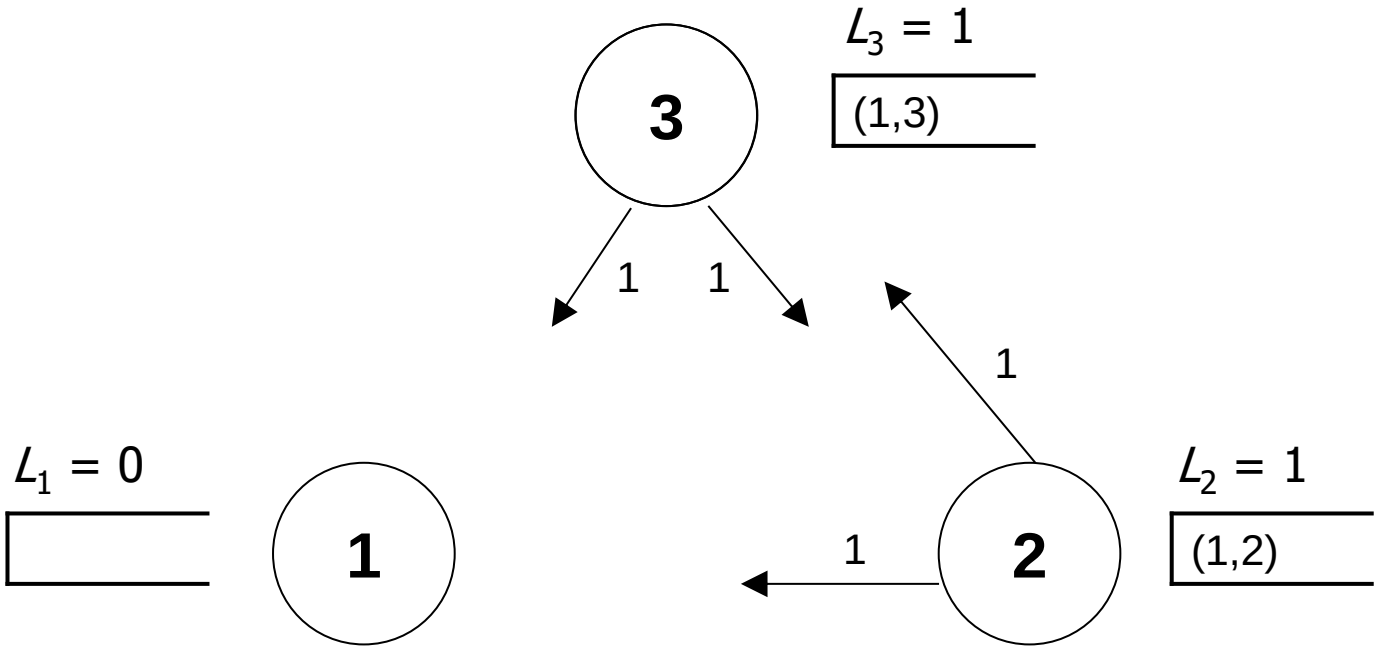
LET IT TRADE

- P_i удаляет свой запрос из начала собственной очереди Q_i и рассылает всем другим процессам RELEASE(L_i, i) с отметкой своего времени
- Получив RELEASE(L_i, i) процесс P_j удаляет запрос P_i из начала своей очереди Q_j
- После удаления P_j запроса от P_i из своей очереди Q_j, его собственный запрос может оказаться первым в Q_j, и P_j может рассчитывать на вход в КС

УНИВЕРСИТЕТ ИТМО

Взаимное исключение

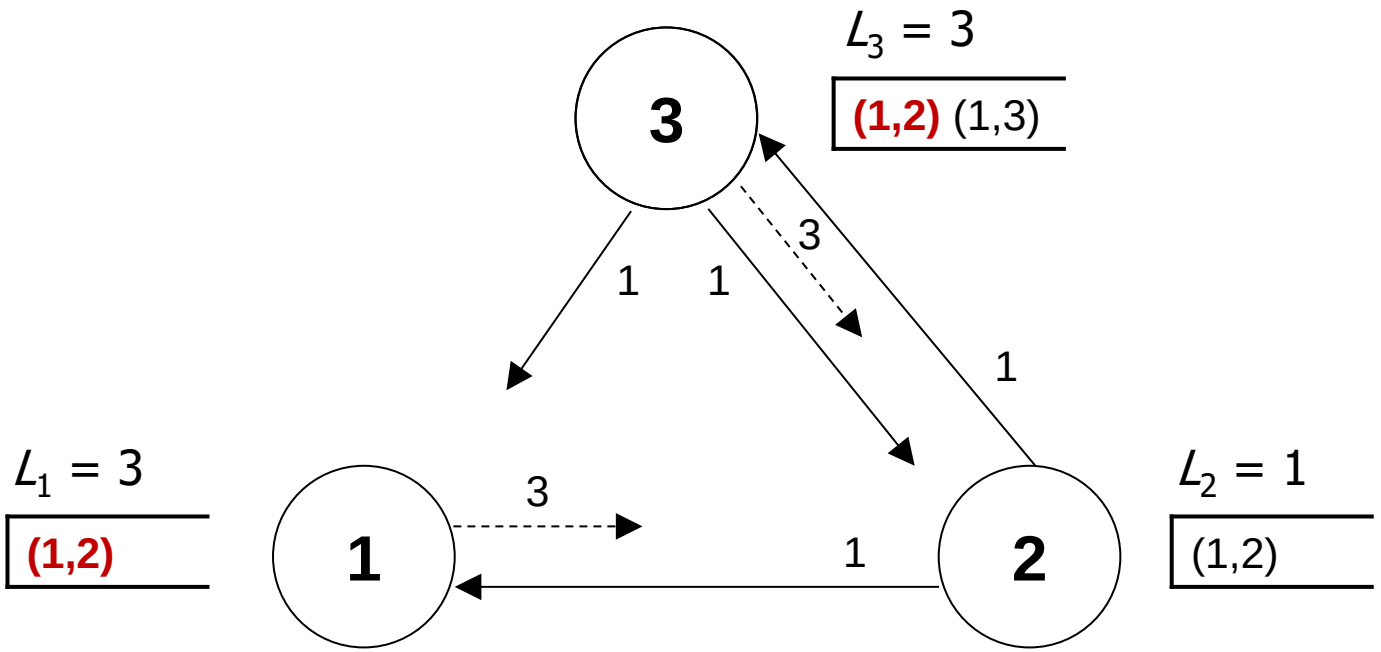
Пример работы алгоритма Л. Лэмпорта:



процессы 2 и 3 запрашивают вход в КС

Взаимное исключение

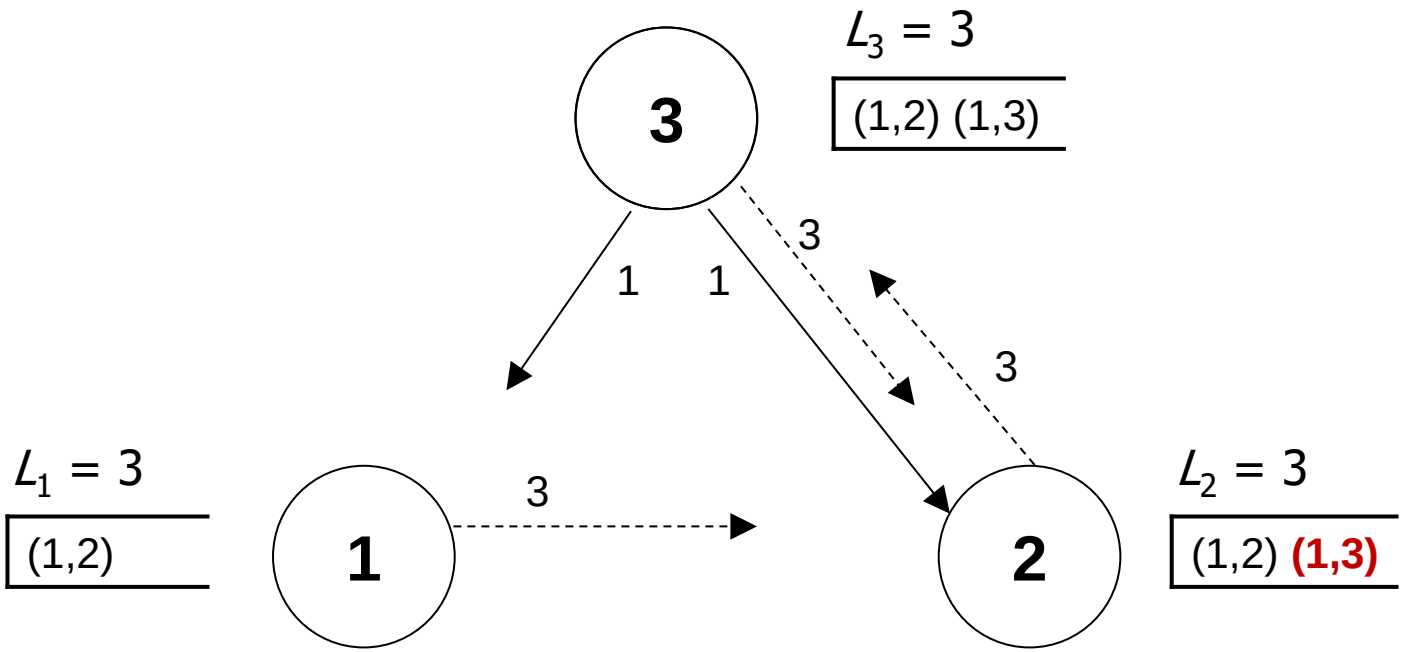
Пример работы алгоритма Л. Лэмпорта:



процессы 1 и 3 отправляют ответ

Взаимное исключение

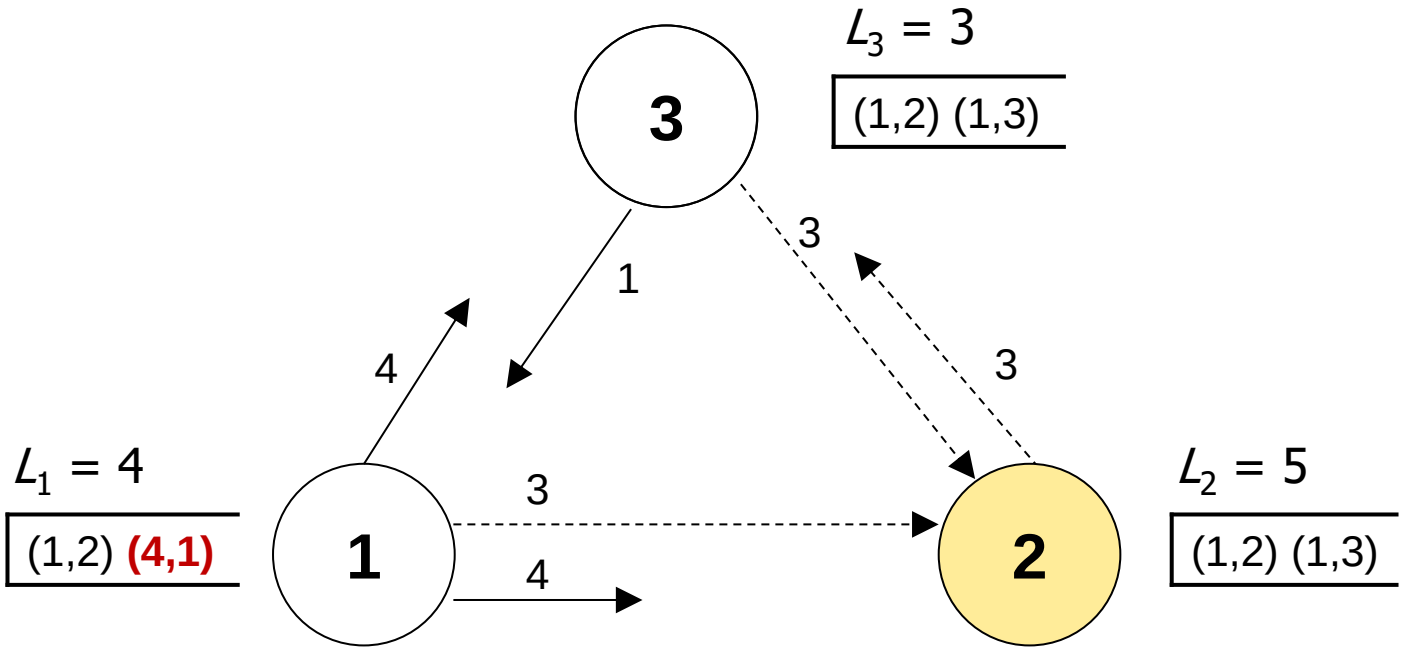
Пример работы алгоритма Л. Лэмпорта:



процесс 2 отправляет ответ

Взаимное исключение

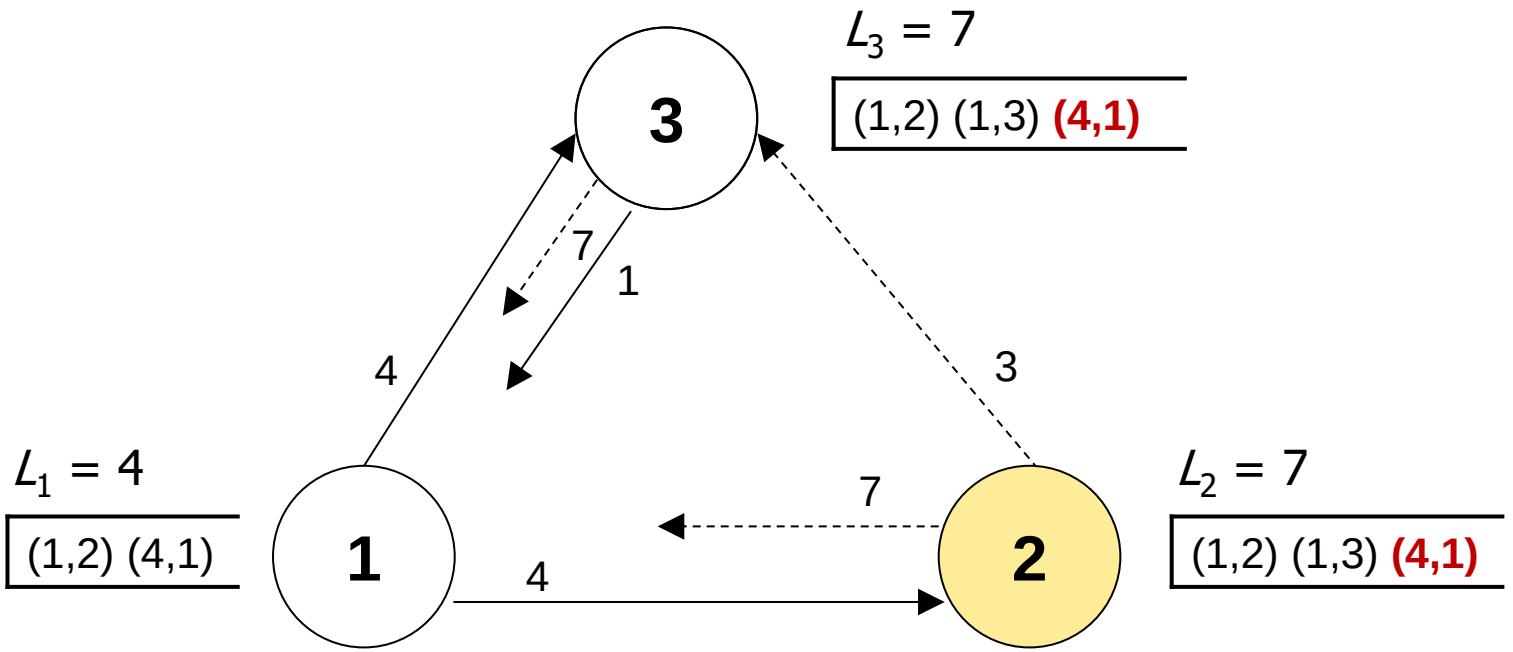
Пример работы алгоритма Л. Лэмпорта:



процесс 2 входит в КС;
процесс 1 запрашивает вход в КС

Взаимное исключение

Пример работы алгоритма Л. Лэмпорта:



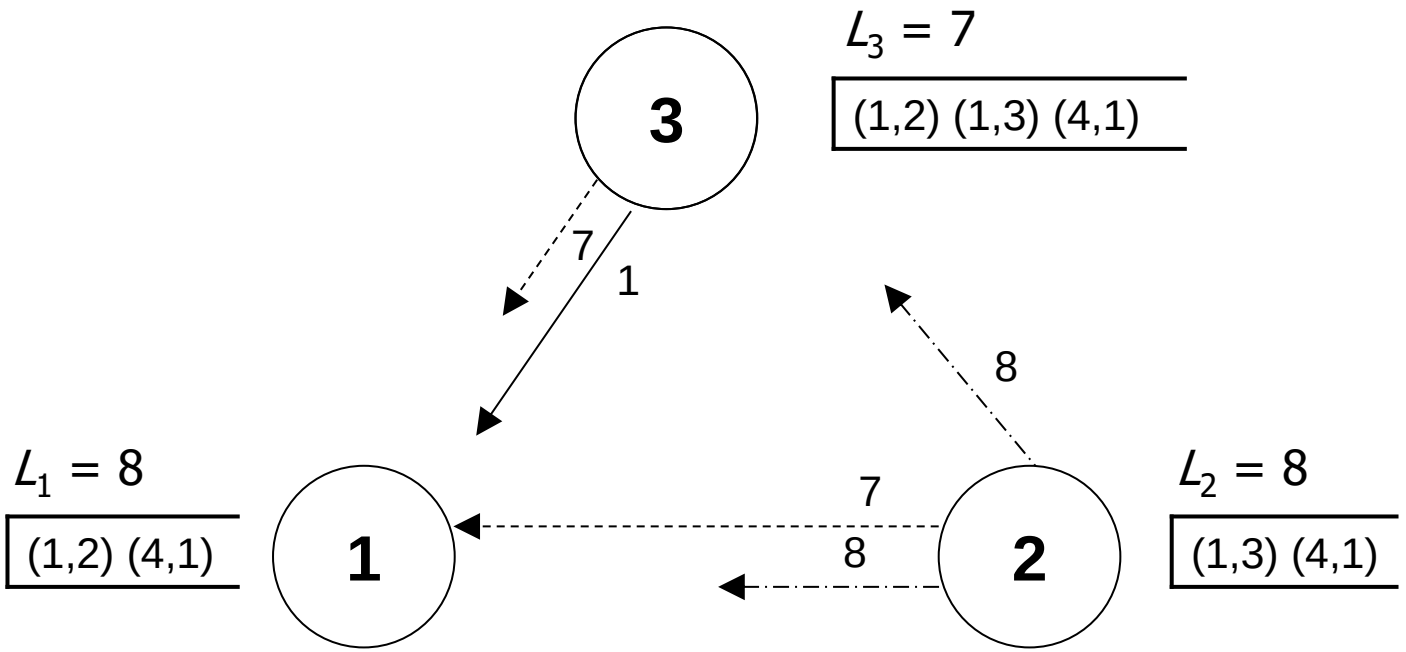
процессы 2 и 3 отправляют ответ

Взаимное исключение

ITIVITI

LET IT TRADE

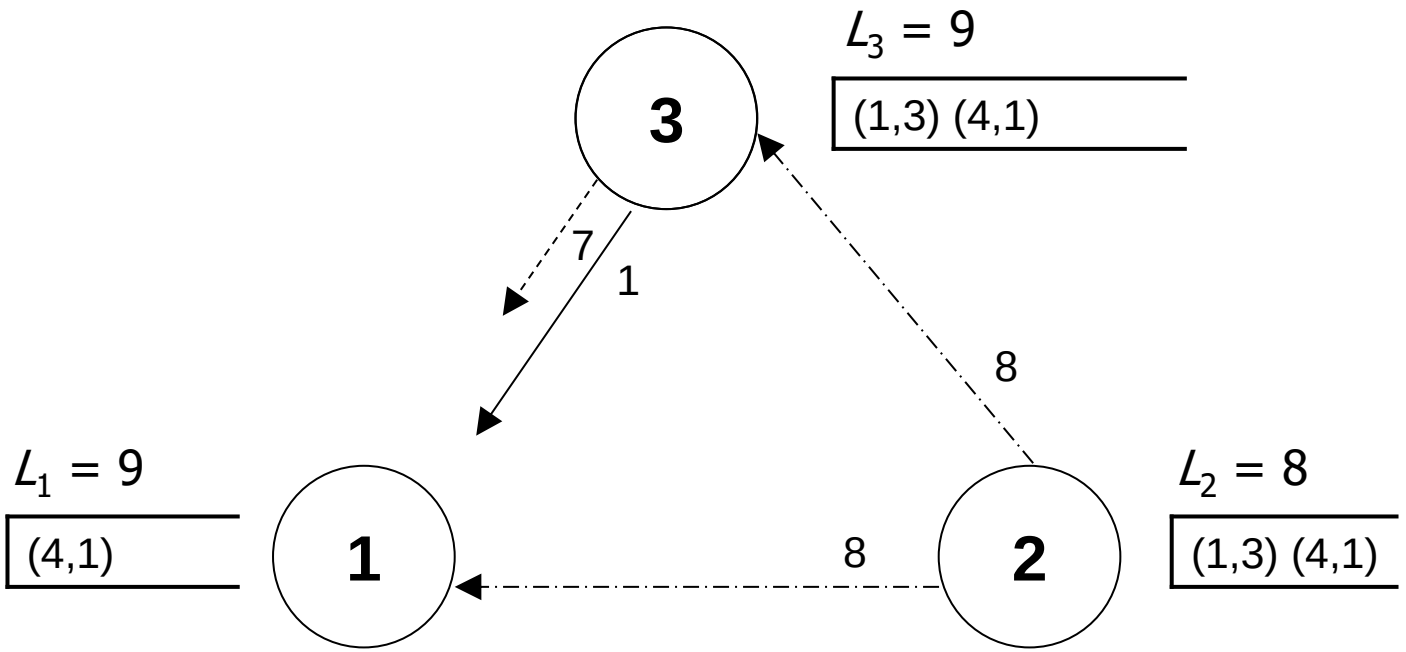
Пример работы алгоритма Л. Лэмпорта:



из-за свойств FIFO канала ответ процесса 3 не может прийти раньше его запроса в процесс 1;
процесс 2 выходит из КС

Взаимное исключение

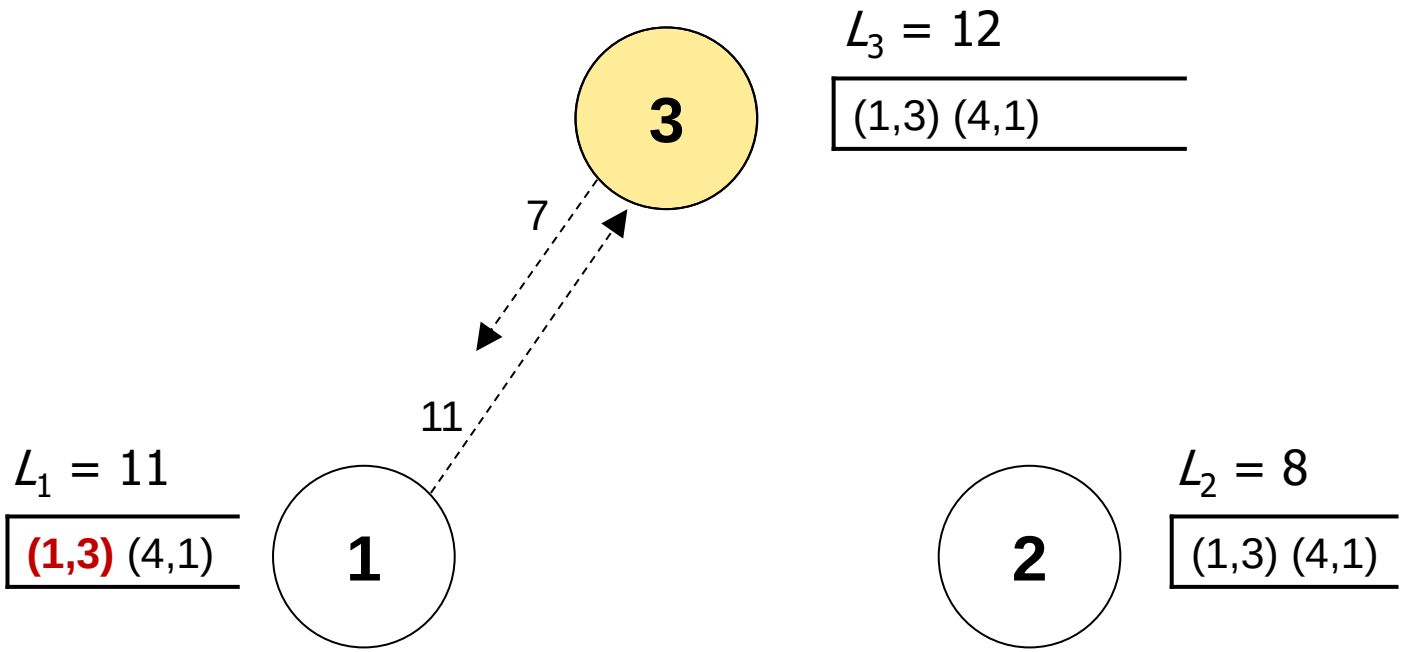
Пример работы алгоритма Л. Лэмпорта:



из-за свойств FIFO канала ответ процесса 3 не может прийти раньше его запроса в процесс 1;
запрос процесса 2 удаляется из очередей процессов 1 и 3

Взаимное исключение

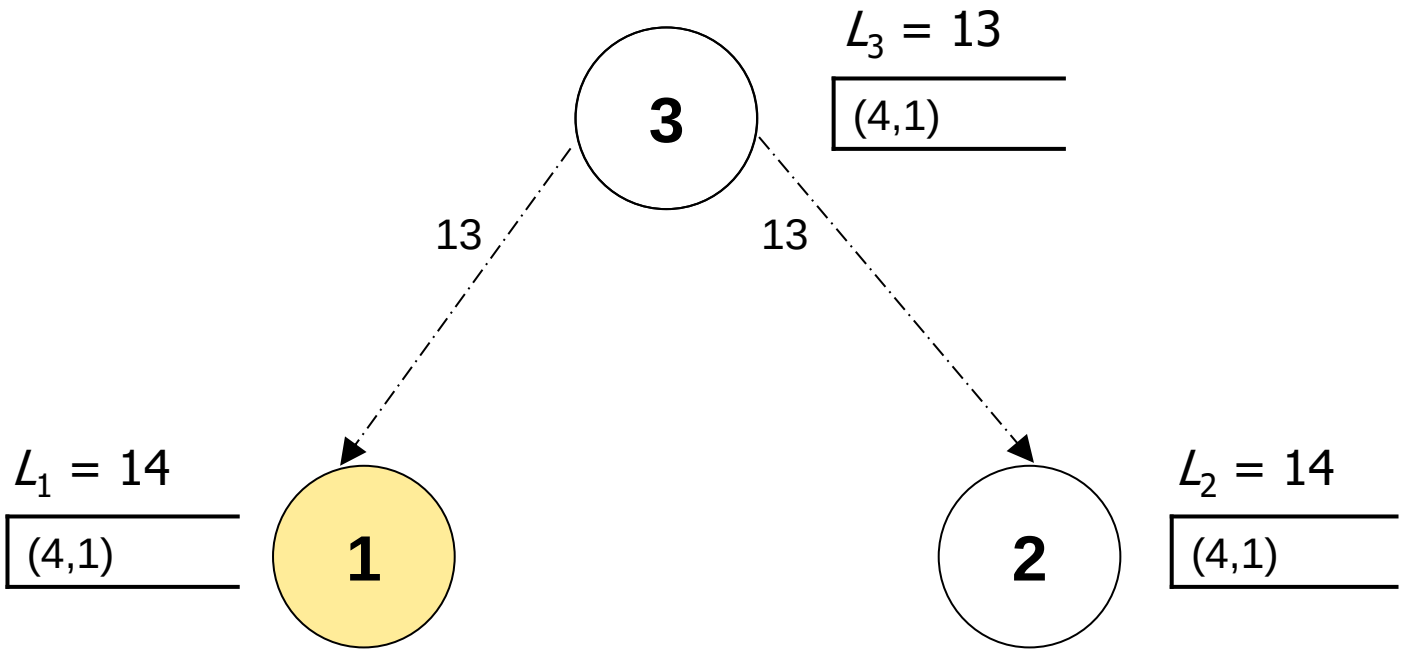
Пример работы алгоритма Л. Лэмпорта:



запрос процесса 3 достигает процесса 1;
процесс 1 отправляет ответ;
процесс 3 входит в КС

Взаимное исключение

Пример работы алгоритма Л. Лэмпорта:



процесс 3 выходит из КС;
запрос процесса 3 удаляется из всех очередей;
процесс 1 входит в КС

Свойства алгоритма Лэмпорта

- Безопасность?
  - Если P_i принимает решение войти в КС, то в системе не может быть никаких других запросов, которые были переданы перед его запросом
- Живучесть?
  - Перед запросом P_i в очереди может оказаться не более (N – 1) запросов с меньшими отметками времени
  - Как только P_i выйдет из КС, он передает RELEASE, что приводит к удалению его запроса из всех копий очереди и предоставлению доступа к КС другому процессу
- Справедливость?
  - По определению…

Свойства алгоритма Лэмпорта

ITIVITI

- Алгоритм является распределенным:
  - все процессы следуют одним и тем же правилам
  - каждый процесс принимает решение о входе в КС только на основе своей локальной информации
- Для входа-выхода из КС необходимо переслать $3(N - 1)$ сообщений
- **Вопрос:** всегда ли нужны все перечисленные сообщения?
- **Вопрос:** все ли сообщения / события должны быть отмечены скалярным временем Лэмпорта?

Оптимизация алгоритма Лэмпорта

ITIVITI

- Иногда можно не посылать сообщения REPLY
- Если P_j получает от P_i REQUEST(L_i, i) после того, как он сам отправил свой REQUEST(L_j, j) со временем (L_j, j) > (L_i, i), тогда P_j нет необходимости отправлять P_i ответ REPLY
- Такой запрос от P_j выполняет функцию REPLY
- На рисунке P_1 получает запрос от P_3 со временем (1,3) после того, как сам отправил запрос со временем (4,1). Поэтому REPLY из P_1 в P_3 со временем (11,1) можно было бы опустить
- Количество сообщений:
  от $2(N – 1)$ до $3(N – 1)$

LET IT TRADE

УНИВЕРСИТЕТ ИТМО

Алгоритм Рикарта-Агравала

ITIVITI

- Попытка оптимизировать алгоритм Лэмпорта, исключив из него сообщения RELEASE
- Снимается требование FIFO для каналов; остальные предположения остаются в силе
- Используется только два типа сообщений: запрос на вход в КС REQUEST и ответ REPLY
- Процесс рассылает REQUEST всем процессам для получения их разрешения на вход в КС
- Процесс сможет войти в КС, когда он получит ответы REPLY от всех остальных процессов

Алгоритм Рикарта-Агравала

ITIVITI

- Если процесс выполняется вне КС …
- Если процесс выполняется внутри КС …
- Если процесс находится в состоянии запроса на вход в КС …
- Для разрешения конфликтов каждый запрос отмечается скалярным временем Лэмпорта
- Если процессу нужно войти в КС, он сравнивает время своего запроса со временем запроса, полученного от другого процесса, и откладывает отправку REPLY, если время своего запроса меньше; иначе – отправляет REPLY немедленно

Алгоритм Рикарта-Агравала

- DR_i [1..N] содержит признак *отложенного ответа* (*deferred reply*) другим процессам
- DR_i [k] инициализируется нулем, $1 \le k \le N$
- Если P_i откладывает отправку ответного сообщения P_j, он устанавливает DR_i [j] = 1, а после отправки сообщения REPLY элемент DR_i [j] сбрасывается в ноль: DR_i [j] = 0

Алгоритм Рикарта-Агравала: Запрос на вход в КС

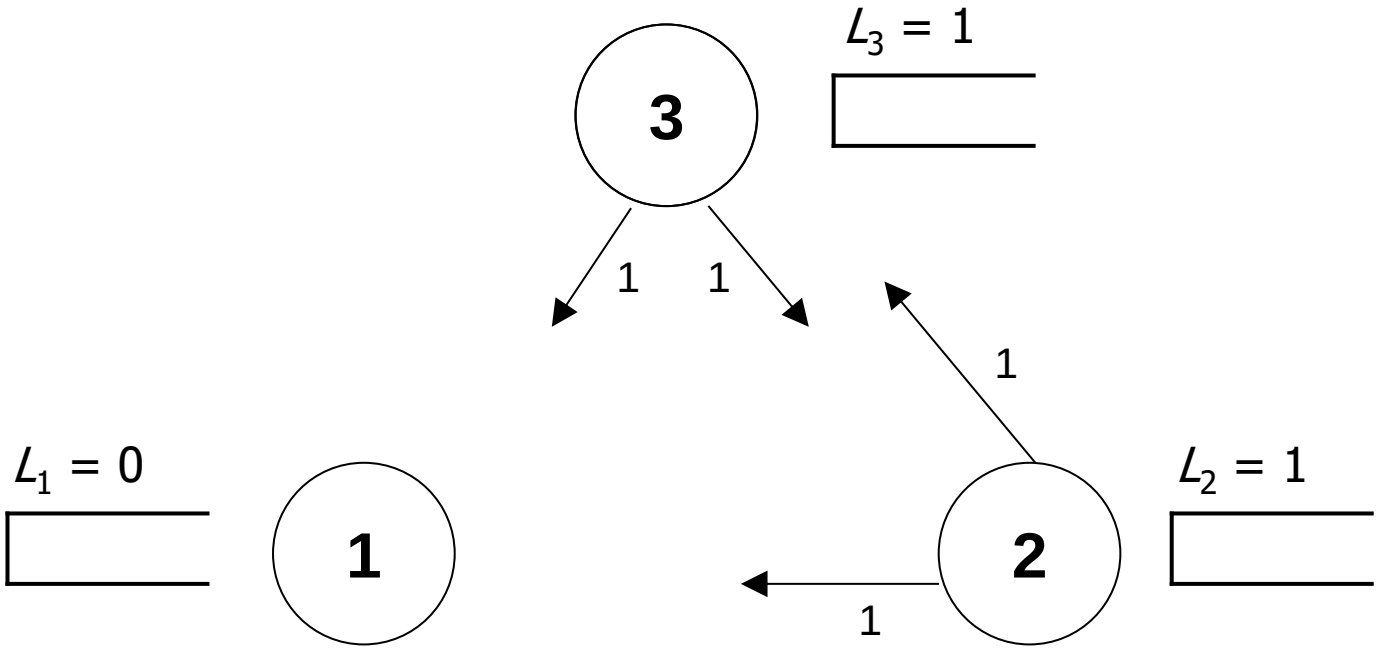
- Когда P_i нужен доступ к КС, он переходит в сост. «запроса на вход в КС» и рассылает REQUEST(L_i, i)
- Когда P_j получает REQUEST(L_i, i) от P_i, он отправляет P_i ответ REPLY если
  - P_j находится в состоянии «выполнения вне КС» или
  - P_j находится в состоянии «запроса на вход в КС» и время его запроса (L_j, j) больше времени (L_i, i) запроса процесса P_i
- Иначе P_j откладывает отправку REPLY и устанавливает $DR_j[i] = 1$

Алгоритм Рикарта-Агравала: Вход в КС и выход из КС

- **Вход в КС**
  Процесс P_i может войти в КС, если он получил сообщения REPLY от всех остальных процессов
- **Выход из КС**
  При выходе из КС P_i рассылает отложенные сообщения REPLY всем ожидающим процессам, т.е. всем процессам P_j, для которых $DR_i[j] = 1$, и затем устанавливает $DR_i[j] = 0$
- **Вопрос:** все ли сообщения / события должны быть отмечены скалярным временем Лэмпорта?

Взаимное исключение

Пример работы алгоритма Рикарта-Агравала:



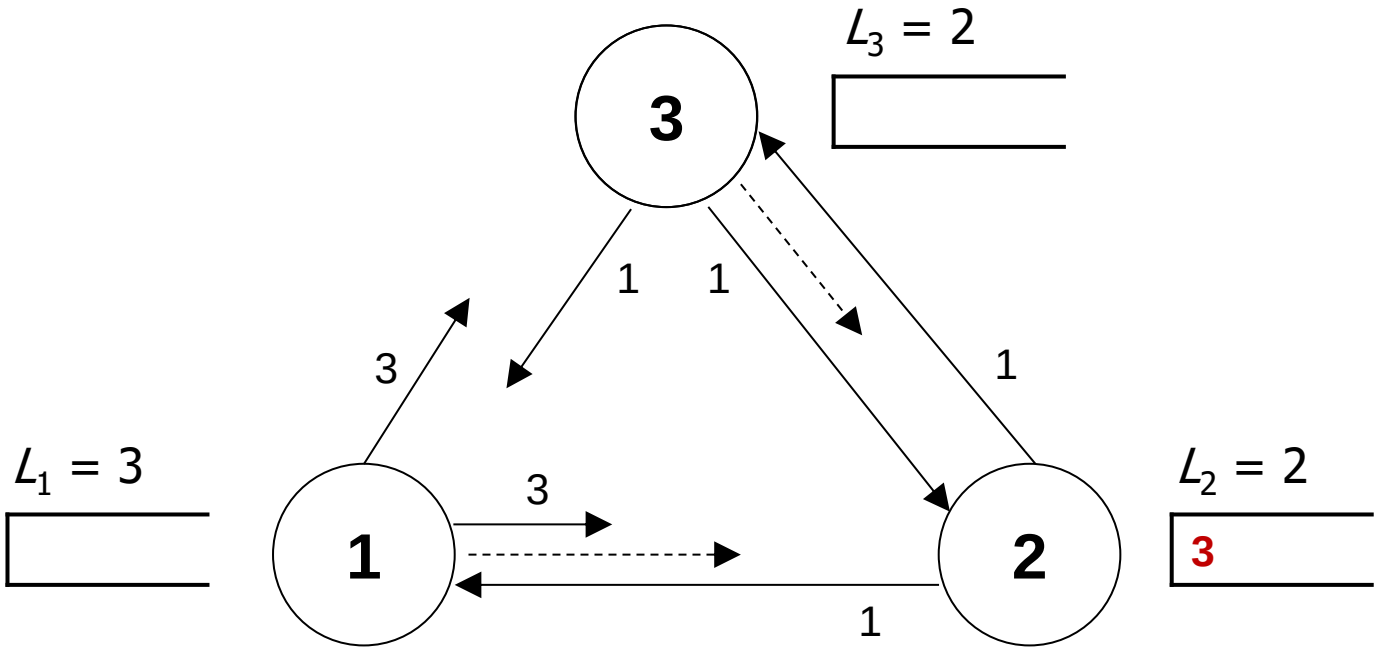
процессы 2 и 3 запрашивают вход в КС

Взаимное исключение

ITIVITI

LET IT TRADE

Пример работы алгоритма Рикарта-Агравала:



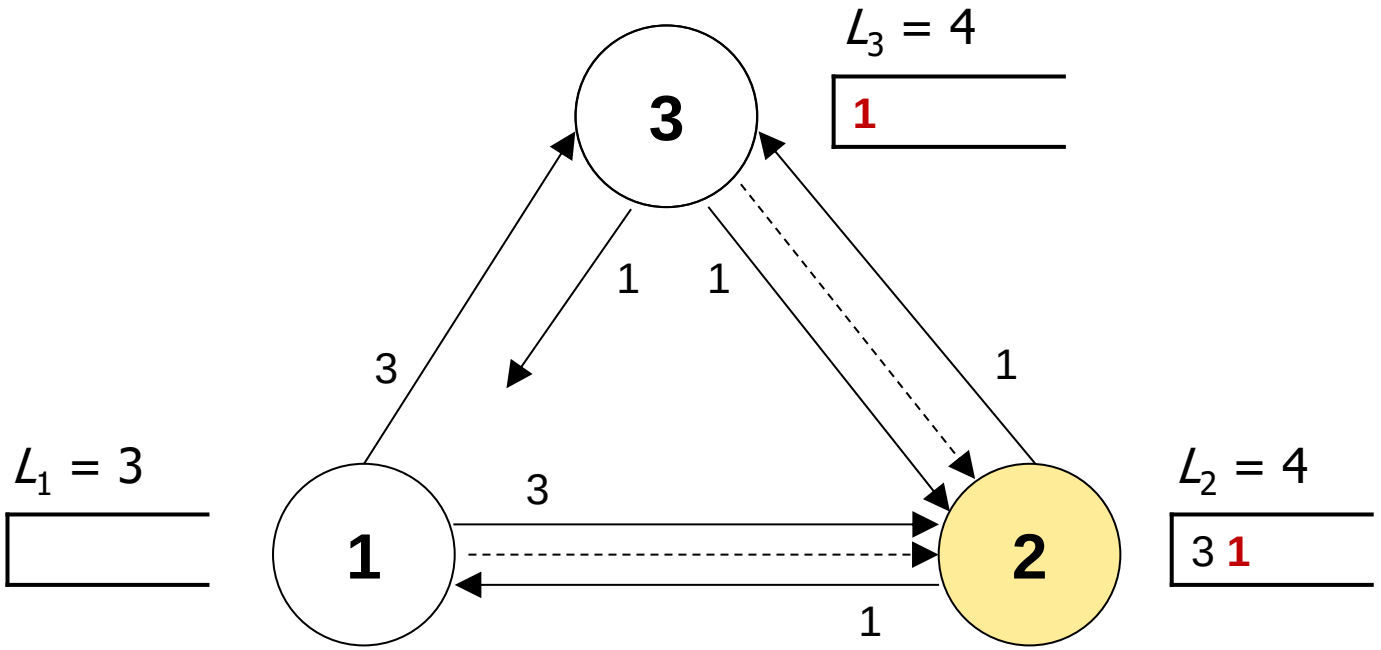
процессы 1 и 3 отправляют ответ; процесс 2 откладывает ответ
процесс 1 запрашивает вход в КС

Взаимное исключение

ITIVITI

LET IT TRADE

Пример работы алгоритма Рикарта-Агравала:



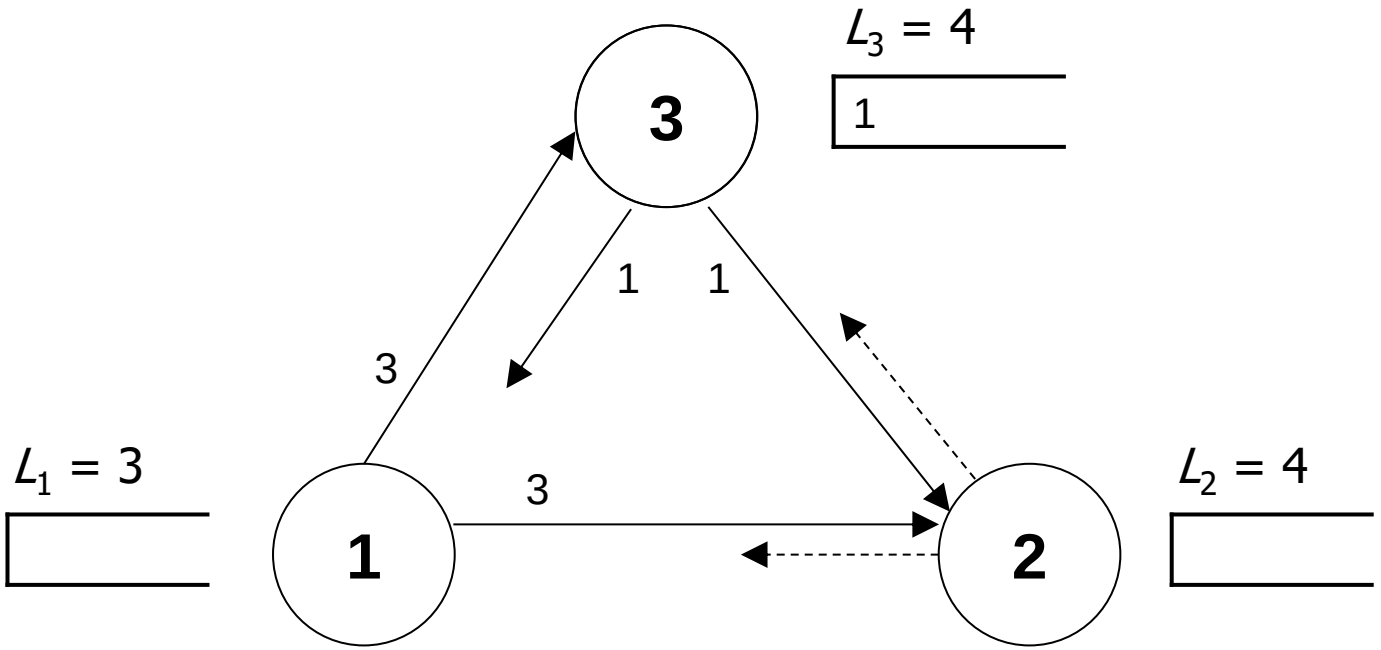
процесс 2 получает все ответы и входит в КС
запросы процесса 1 достигают процессов 2 и 3
процессы 3 и 2 откладывают ответ

Взаимное исключение

ITIVITI

LET IT TRADE

Пример работы алгоритма Рикарта-Агравала:



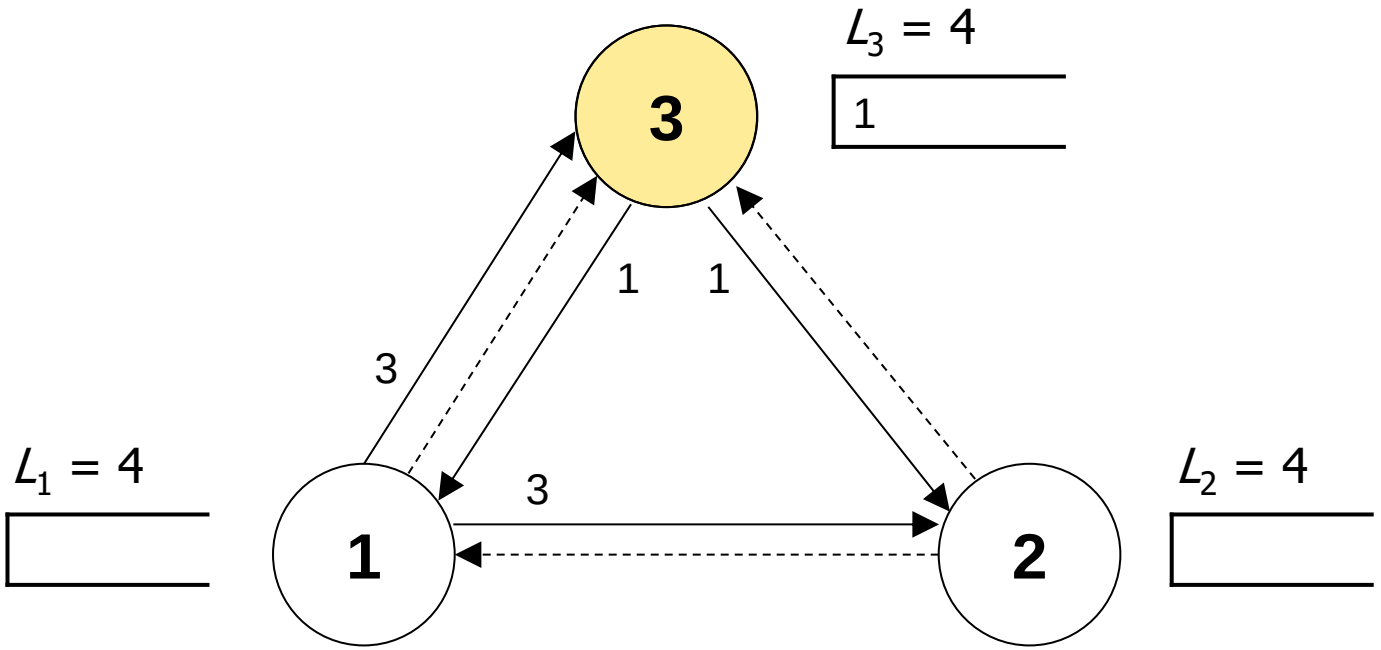
процесс 2 выходит из КС
процесс 2 рассылает отложенные ответы процессам 3 и 1

Взаимное исключение

ITIVITI

LET IT TRADE

Пример работы алгоритма Рикарта-Агравала:



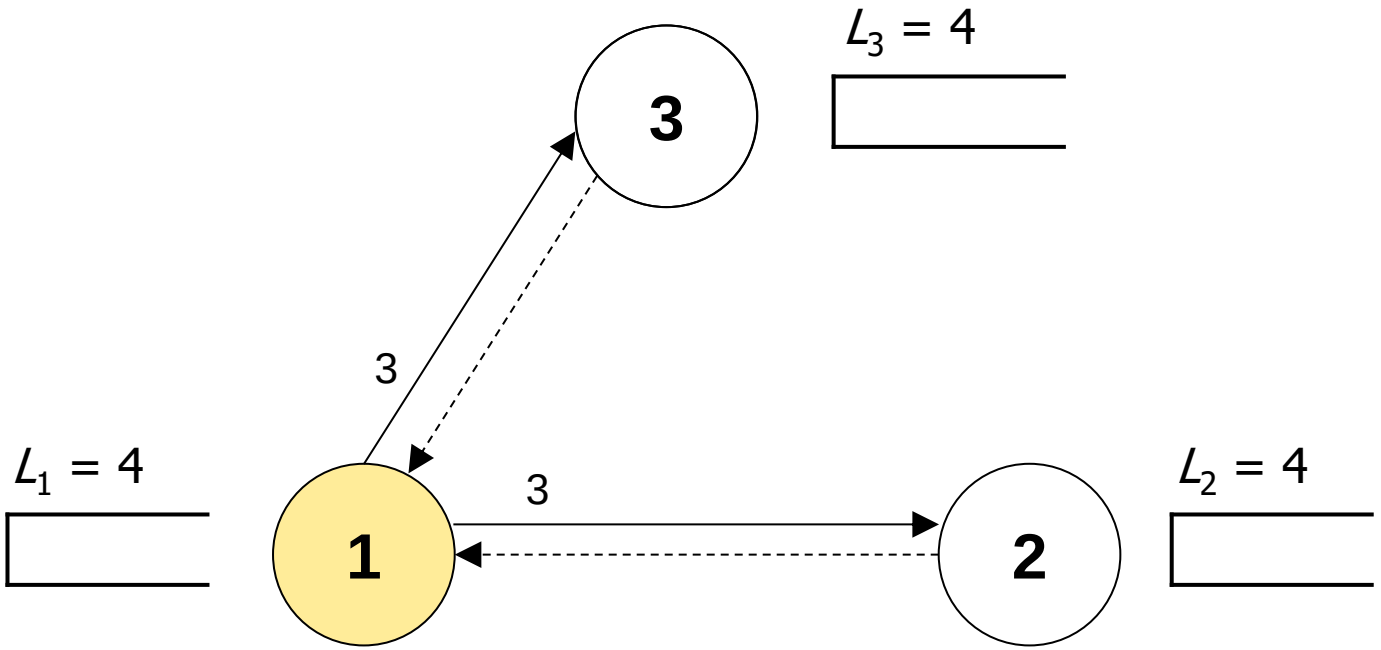
отложенные ответы процесса 2 достигают процессов 1 и 3
запрос процесса 3 достигает процесса 1, процесс 1 высылает ответ
процесс 3 получает ответы от процессов 2 и 1 и входит в КС

Взаимное исключение

ITIVITI

LET IT TRADE

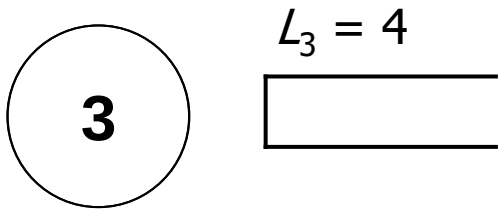
Пример работы алгоритма Рикарта-Агравала:



процесс 3 выходит из КС
процесс 3 отправляет отложенный ответ процессу 1
процесс 1 получает все ответы и входит в КС

Взаимное исключение

Пример работы алгоритма Рикарта-Агравала:



процесс 1 выходит из КС
отложенных ответов у процесса 1 нет

Свойства алгоритма Рикарта-Агравала

- Безопасность? От противного:
  - Пусть P_i и P_j выполняются в КС одновременно, и время запроса P_i меньше времени запроса P_j
  - P_j может войти в КС только когда он получит ответы на свой запрос от всех остальных процессов
  - Запрос от P_j достиг P_i после того, как P_i отправил свой запрос; в противном случае время запроса P_i должно быть больше времени запроса P_j по правилам работы логических часов
  - По условиям алгоритма P_i не может отослать ответ REPLY процессу P_j, находясь в состоянии «запроса на вход в КС» и имея меньшее время своего запроса, до тех пор, пока P_i не выйдет из КС
  - Противоречие с предположением, что P_j получил REPLY от всех процессов, в том числе, от P_i

Свойства алгоритма Рикарта-Агравала

ITIVITI

- Живучесть?
  - Построим цепь из процессов, ожидающих друг друга: P_i ждет P_j, P_j ждет P_k, P_k ждет P_m, и т.д.
  - Эта цепь имеет ограниченную длину и является ациклической: процессы располагаются в ней в порядке убывания отметок времени запросов
  - Последний процесс P_l с наименьшим временем запроса либо выполняется в КС, либо получит разрешения от всех процессов и войдет в КС
  - При выходе из КС P_l разошлет ответные сообщения всем ожидающим процессам
  - Поэтому за ограниченное число шагов любой процесс окажется последним в цепи ожидающих процессов и сможет войти в КС

Свойства алгоритма Рикарта-Агравала

ITIVITI

LET IT TRADE

- Справедливость?
  - По определению…
- Для входа-выхода из КС требуется обмен $2(N - 1)$ сообщениями:
  - $(N - 1)$ сообщений-запросов REQUEST
  - $(N - 1)$ ответных сообщений REPLY

УНИВЕРСИТЕТ ИТМО

ITIVITI

LET IT TRADE

Спасибо за внимание!

УНИВЕРСИТЕТ ИТМО