**Понятие логических часов**

Механизм логических часов позволяет контролировать порядок событий при распределенных вычислениях и, как следствие, упорядочивать события (т.е. с использованием логических часов формируется порядок событий)

Введем в рассмотрение обозначения:

Е – множество событий, происходящих в системе (в частности, событий связанных с обменом сообщениями)

Т = { | i = 1, n} – множество допустимых значений логических часов.

Функция θ – функция отображающая множество событий Е на множество на множество (упорядоченное множество Т).

Т.е. Е→ { | ≤ } или Е → T

Процесс , часы , каждому процессу Р; поставлены в соответствие часы

Если – некоторое событие в процессе , то () → (где i – номер элемента в множестве для – го процесса)

Составляющие логических часов каждого процесса :

- локальные логические часы – измерение собственного хода выполнения процесса (т.е. локальные часы – ход своего собственного выполнения);

- глобальные логические часы – представление процесса о глобальном времени (т.е. для записи информации о выполнении других процессов).

Логические глобальные часы используются на назначения временных отметок для собственных событий.

Правила изменения логических часов (типы правил, виды правил):

- Правило 1 – определяет, как процесс изменяет свои локальные часы при наступлении в нем события.

- Правило 2 - определяет, как процесс изменяет глобальные часы для отображения событий в других процессах.

Правила обеспечивают выполнение условий вида:

1) Если и - события процесса , такие что → (где → - отношение порядка),   
то () < (’)

2) Если и - события отправки сообщения процессам и получение сообщения процесса , то θ () < θ ().

Т.о. необходимо хранить значения часов и определить механизм изменения их значений

**Скалярное время**

Скалярное время (механизм скалярного времени) предполагает, что логическое локальное время процесса и его значения глобального времени представляются одной скалярной величиной, обозначенной

Для скалярных часов правила 1 и 2 определяются следующим образом:

- Правило 1

Перед выполнением любого события процесс увеличивает значение локальных часов

= ,

Данное правило удовлетворяет условию 1 не противоречивости логических часов

- Правило 2 (реализуемое для синхронизации значений логических часов разных процессов):

Каждое передаваемые сообщения сопровождаются значения процесса отправителя на момент отправки сообщения. (Значение )

Процесс , получивший сообщение, содержащее , выполняет следующие действия:

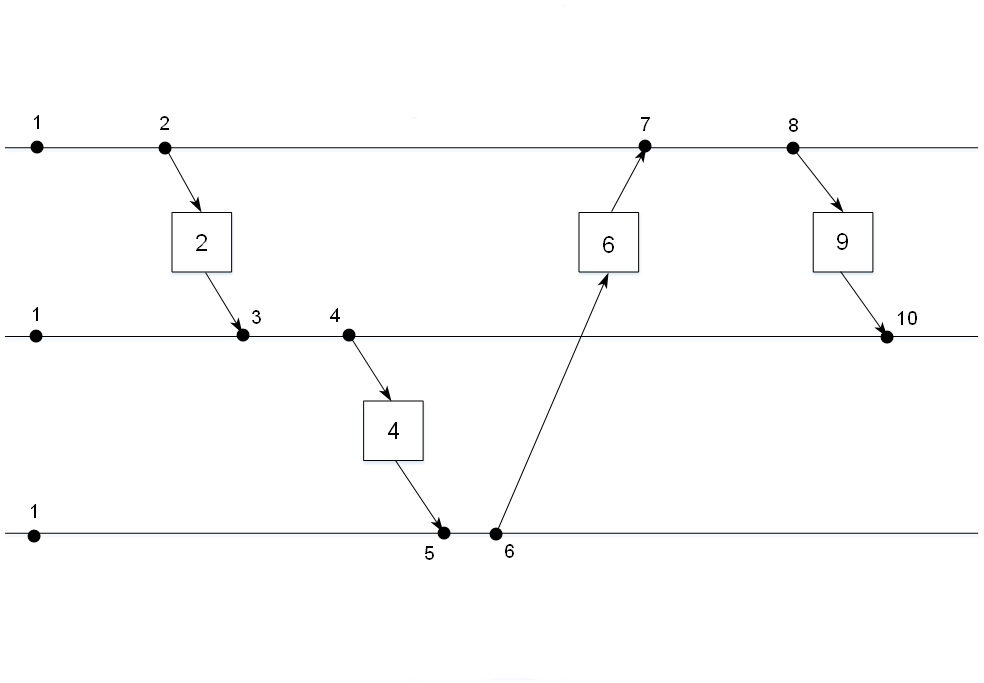
1. , )

2. Реализует действия, соответствующие правилу 1 ( = )

3. Реализует обработку сообщения в соответствии с правилом 2 разграничиваются события отправки и принятия сообщений (разные значения часов для отправки и принятия сообщений), т.е.

→ => L () < L ( )

Пример реализации взаимодействия процессов с использованием скалярных часов



Р1

Р2

Р3

В данном примере речь не идет о широковещании сообщений

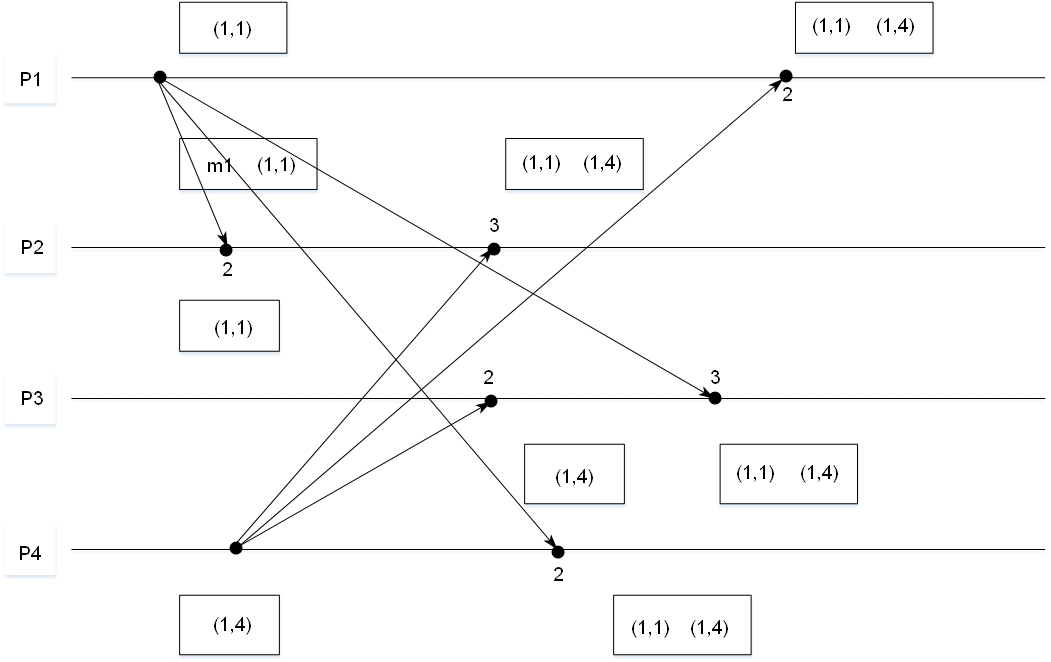
Пример реализации групповой рассылки с использованием скалярных часов.

Каждый процесс должен иметь информацию о событиях рассылки, происходящих на других процессах.

Каждый процесс поддерживает локальную очередь, в которую помещаются широковещательные сообщения, упорядоченные по возрастанию значений меток времени.

Извлечение сообщений из очереди позволяет выполнить их интерпретацию. Временная метка для каждого процесса представляется в виде: (L (m), i), где L (m) – часы для сообщения m, i – идентификатор процесса.

Размещение идентификатора процесса в нужном месте очереди обеспечивает передачу подтверждающих сообщений от всех процессов, для которых выполнено широковещание.



Подтверждения нужны для широковещания процессов о том, что в каналах не осталось синхронизации (не осталось сообщений), которые могут потеснить в очереди уже принятые.

**Взаимные исключения в распределенных системах**

Виды алгоритмов, обеспечивающих синхронизацию доступа к ресурсам:

- централизованные;

- распределенные.

Централизованные алгоритмы действия по синхронизации доступа обеспечиваются одним (ограниченной группой) процессов.

Централизованные алгоритмы архитектура «клиент – сервер».

**Вариант** – реплицированные серверы (для повышения надежности системы синхронизации).

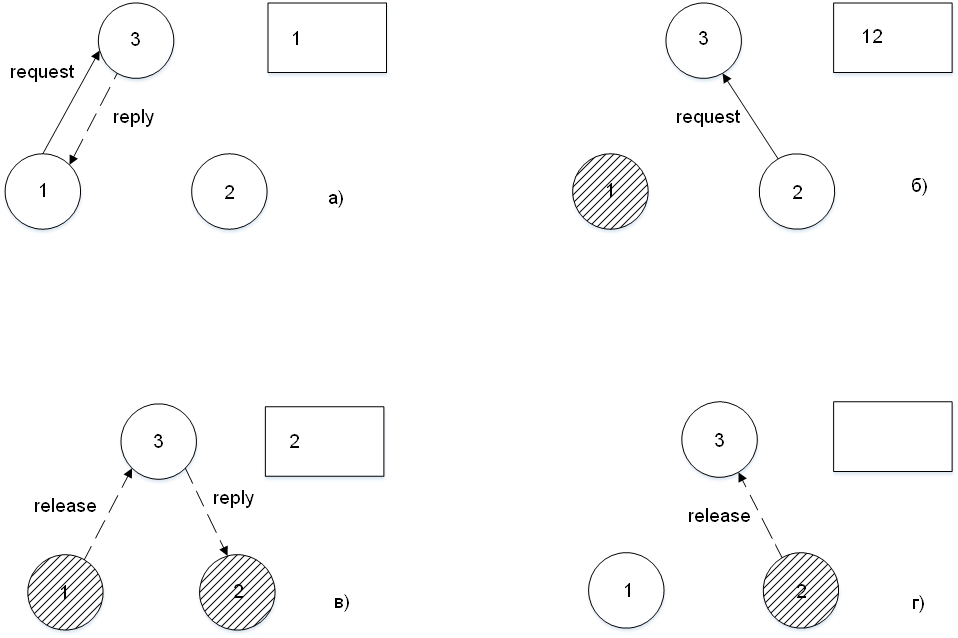
Распределенные алгоритмы – симметричные, т.е все процессы выполняют одни и те же функции.

**Централизованный алгоритм**

Разрешение на вход в критическую секцию выдается серверным процессом.

Два типа сообщений, обеспечивающих вход в критическую секцию request (запрос), reply – разрешение. Сообщение с указанием об освобождении ресурсов release. Т.о. сообщения с запросом ресурсов и с указанием необходимости освобождения ресурсов передается серверу. Сервер управляет очередью процессов к каждому из ресурсов. Если очередь не пуста, значит ресурс занят.

Сервер



**Распределенный алгоритм взаимоисключения на основе скалярных часов**

В данном алгоритме каждый процесс оперирует с локальной очередью, в которую помещаются запросы на доступ к соответствующему ресурсу. Запросы упорядочиваются в соответствии со значением их временной метки. Т.е. запросы обслуживаются в порядке их возникновения в системе. Под меткой времени запроса подразумевается пара ( , i), где – значение (текущее) скалярных часов i-го процесса, i-идентификатор процесса.

Постановка меток в очередь обеспечивается широковещательной рассылкой запросов и гарантируется передачей от всех процессов в системе процессу-идентификатору, рассылки подтверждающих сообщений.

Обозначение очереди –

Виды сообщений:

1. Запросы на вход в КС

Процессу, которому требуется ресурс, реализует широковещательную рассылку сообщения request (, i) со значением локального времени. Процесс , получив запрос от , помещает его в локальную очередь и отправляет процессу ответ reply (, i) со значением своих логических часов.

2. Условия входа процесса в КС:

а) Запрос request (, i) процесса обладает наименьшим значением временной метки среди всех запросов, находящихся в локальной очереди ;

б) Процесс получил сообщение от остальных процессов с отметкой времени, большей чем (, i) (это гарантирует, что процессу известно обо всех запросах, предшествующих его текущему запросу);

3. Реализация выхода процесса из КС

а) Процесс удаляет запрос (свой) из очереди и рассылает другим процессам сообщение release (, i) с отметкой логического времени.

б) Получив сообщение release (, i) процесс удаляет запрос процесса из своей очереди . После удаления запроса процесса в очереди запросы процесса может оказаться с меньшей временной меткой (для этого должен ранее получить подтверждение его запроса от всех процессов).

Особенности алгоритма:

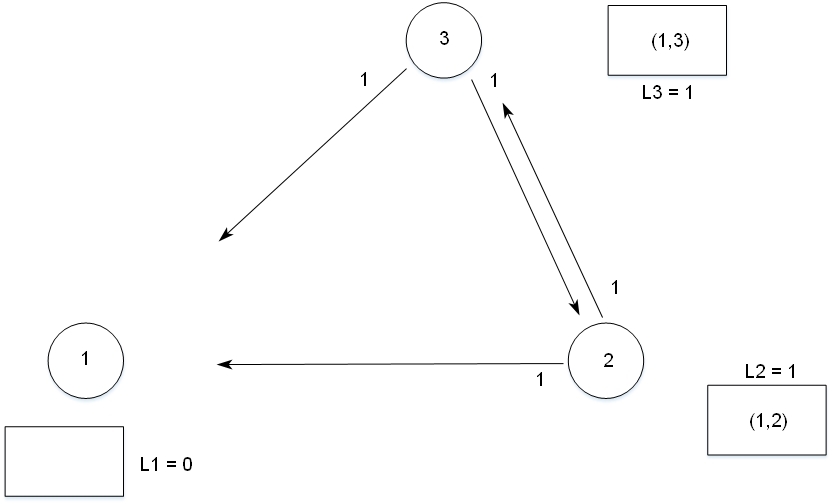
1. Рассмотренный алгоритм является распределенным, т.е. все процессы реализуют одни те же действия, каждый процесс принимает решение о входе в КС на основе своей локальной информации

2. Если каждый процесс получил подтверждения от других процессов в ответ на свой запрос, то все процессы имеют одинаковый состав своих очередей, в которых запросы упорядочены одинаковым образом.

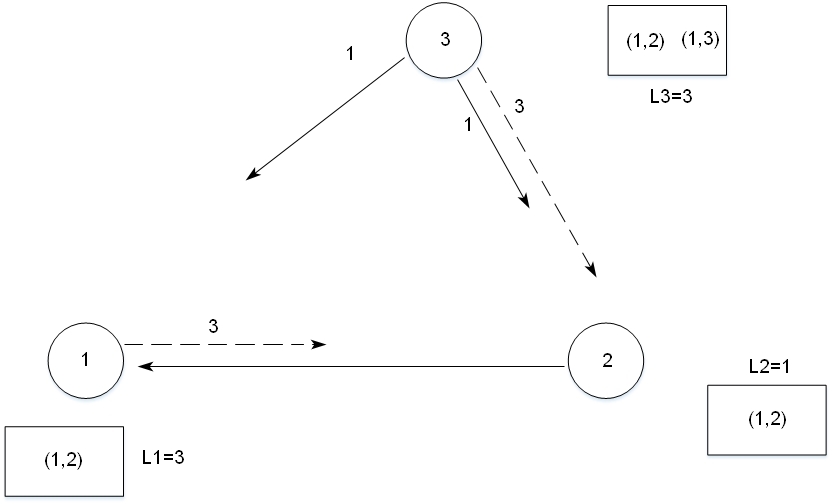
Пример реализации распределенного алгоритма синхронизации входа в критический секции

обозначение веса дуги:

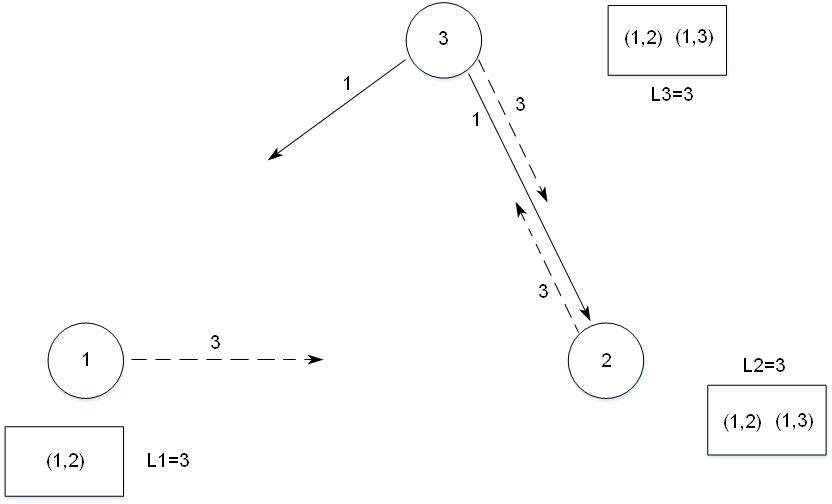
1) Процессы и одновременно запрашивают вход в КС, изменяя значения своих локальных часов и



2) Процессы и получают запрос от процесса и отправляют ему ответы. Метка (1,2) меньше метки (1,3), тогда запрос помещается в голову очереди

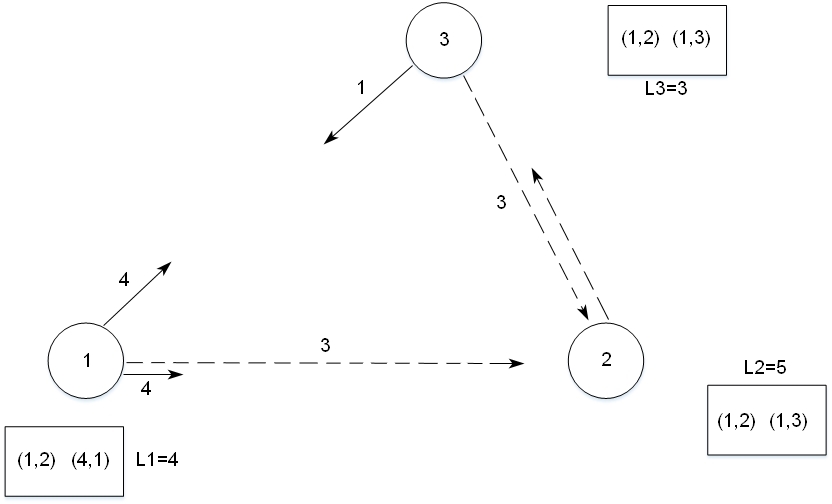


3)



Т.к. канал – это очередь то запрос от на придет раньше, чем подтверждение от на . Запрос от не дошел ни до , ни до .

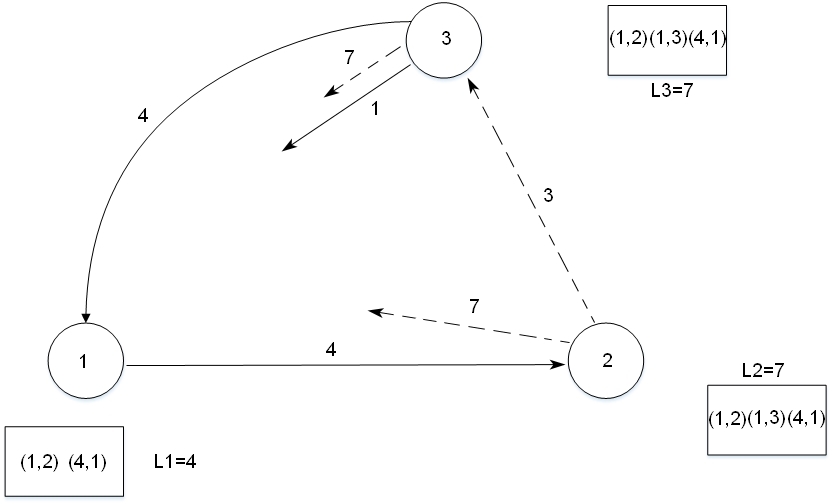
3. Процесс получает ответные сообщения от процессов и , поэтому увеличивает свои часы до



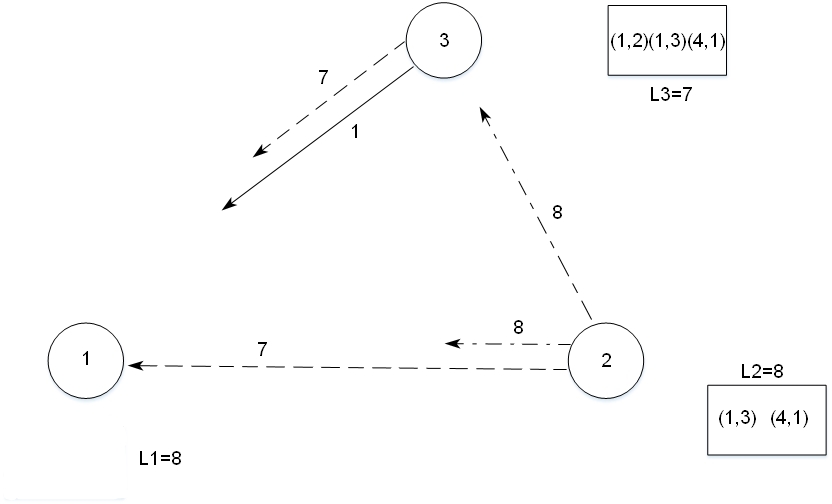
Т.к. процесс получил подтверждения процессов, и он в голове своей локальной очереди, то процесс входит в КС. Процесс рассылает запрос на вход в критическую секцию.

Процессы и посылают ответные сообщения.

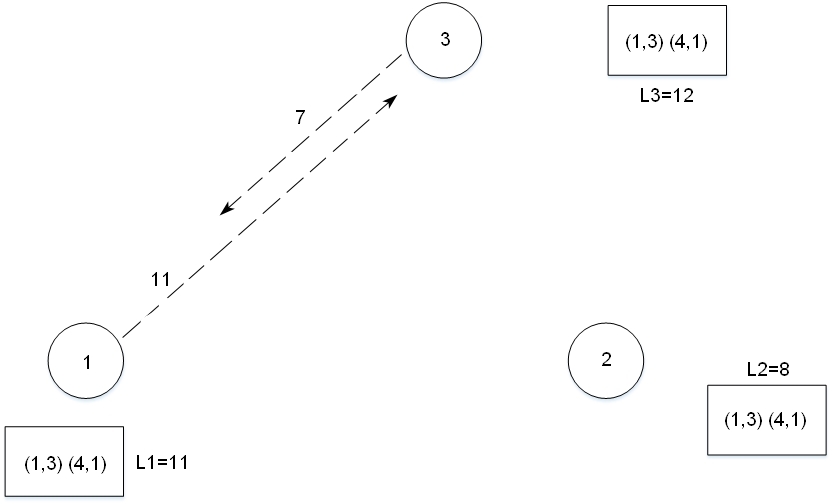
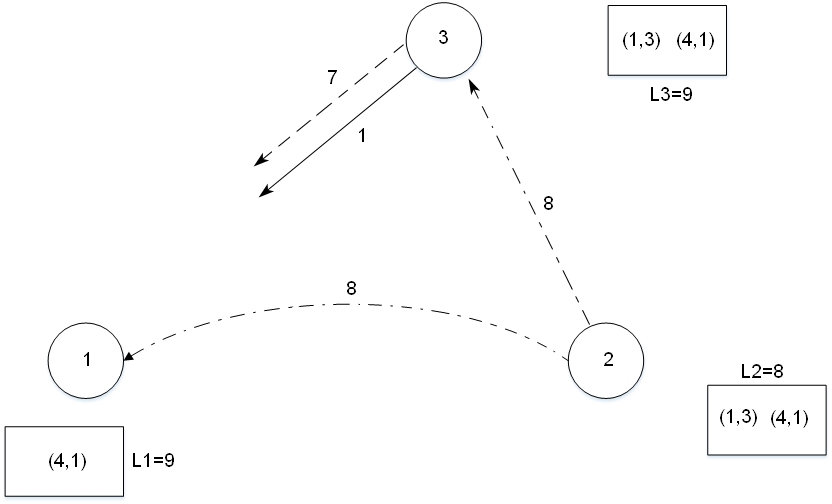
4)



5) Освобождение ресурса



6)



11 – подтверждение 1

