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:

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

T = $\{T_i \mid i = 1, n\}$ – множество допустимых значений логических часов.

Функция θ – функция отображающая множество событий E на множество на множество (упорядоченное множество T).

Т.е. E→ $\{ T_i \mid T_i \le T_{i+1} \}$ или E → T

Процесс P_i, часы θ_i, каждому процессу P; поставлены в соответствие часы θ_i

Если e_i – некоторое событие в процессе P_i, то $\theta_i(e_i) \rightarrow T_i$ (где i – номер элемента в множестве T_i для P_i – го процесса)

Составляющие логических часов каждого процесса P_i:

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

- глобальные логические часы – представление процесса P_i 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 e_i и e_i' - события процесса P_i , такие что $e_i \rightarrow e_i'$ (где → - отношение порядка), то $\theta_i(e_i) < \theta_i(e_i')$

2) Если e_i и e_i' - события отправки сообщения процессам P_i и получение сообщения процесса P_j, то $\theta(e_i) < \theta(e_j)$.

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 P_i и его значения глобального времени представляются одной скалярной величиной, обозначенной L_i

Для скалярных часов правила 1 и 2 определяются следующим образом:

- Правило 1

Перед выполнением любого события процесс P_i увеличивает значение локальных часов L_i

$$L_i = L_i + \alpha_i, \alpha_i > 0$$

Данное правило удовлетворяет условию 1 не противоречивости логических часов

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

Каждое передаваемые сообщения сопровождаются значения L_i процесса отправителя на момент отправки сообщения. (Значение $L_{msy} = L_i$)

Процесс P_i, получивший сообщение, содержащее L_{msy}, выполняет следующие действия:

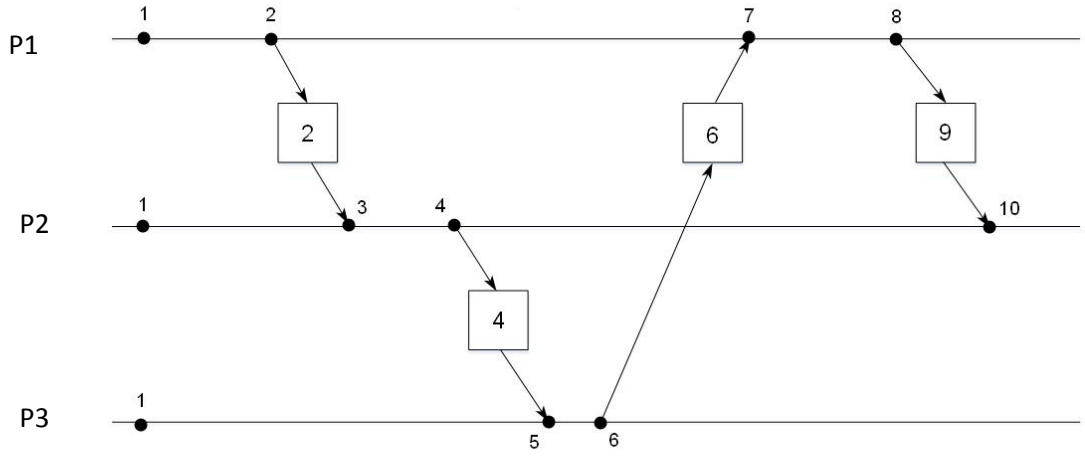
1. $L_i = \max(L_i, L_{msy})$

2. Реализует действия, соответствующие правилу 1 ($L_i = L_i + \alpha_i$)

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.

$$e_i \rightarrow e_j' \Rightarrow L(e_i) < L(e_j')$$

Пример реализации взаимодействия процессов с использованием скалярных часов



В данном примере речь не идет о широковещании сообщений

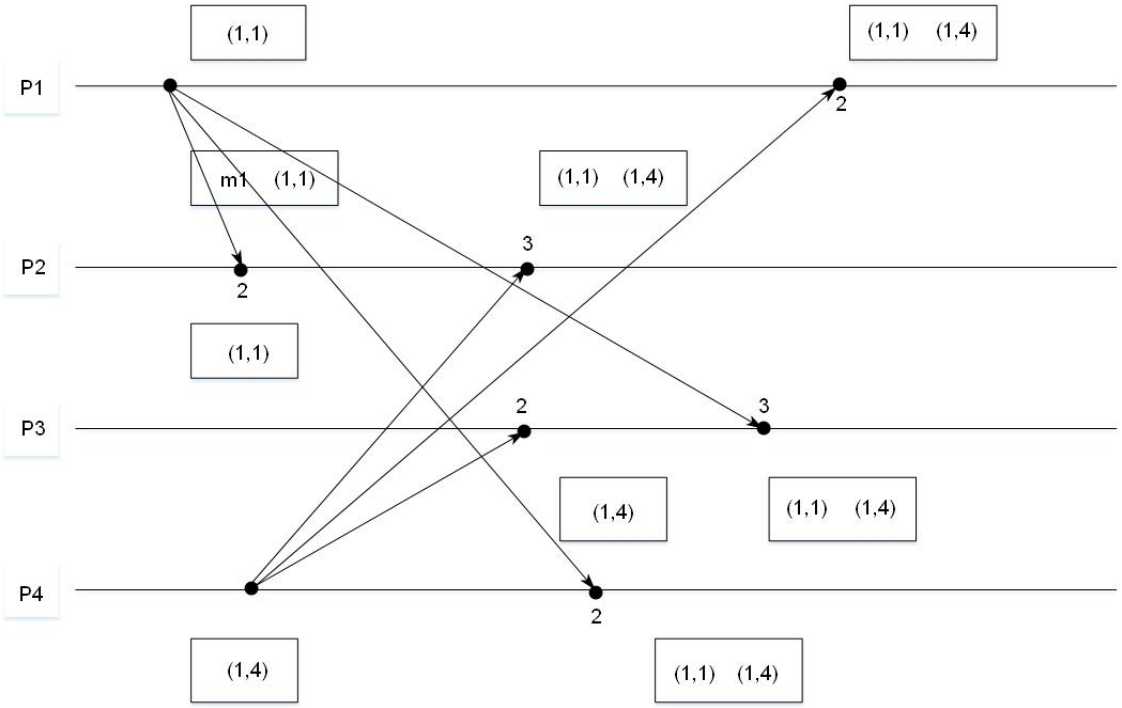
Пример реализации групповой рассылки с использованием скалярных часов.

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».

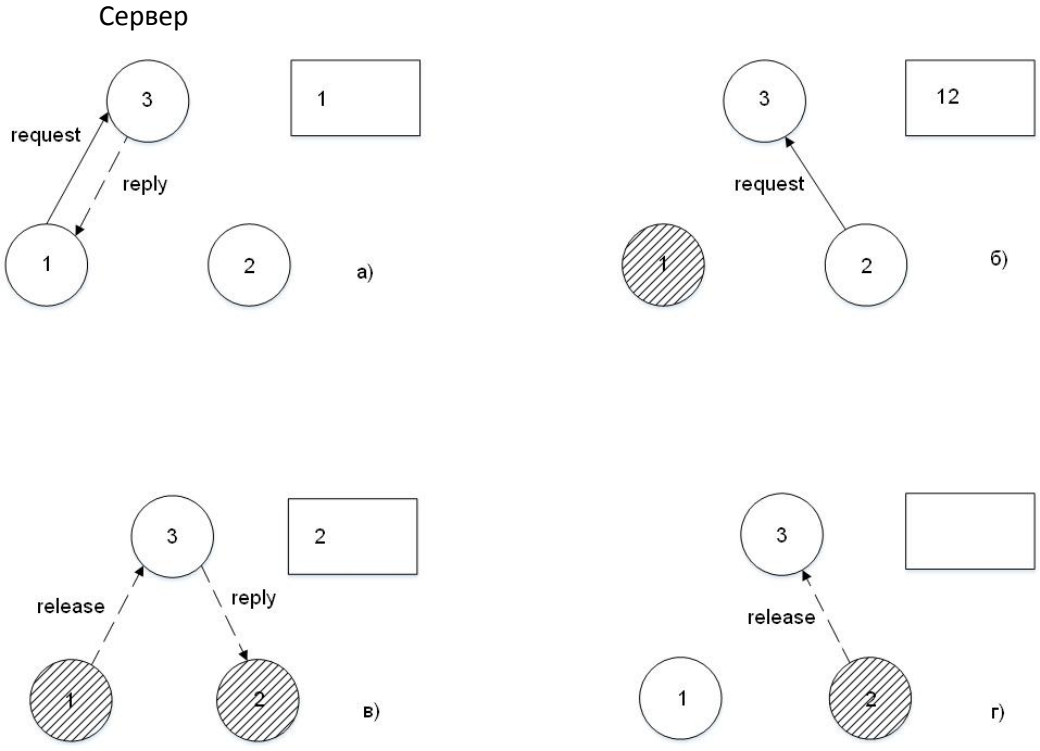
Вариант – реплицированные серверы (для повышения надежности системы синхронизации).

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

<u>Централизованный алгоритм</u>

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.



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 (L_i, i), где L_i – 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 – Q_i

Виды сообщений:

1. Запросы на вход в КС

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

2. Условия входа процесса P_i в КС:

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

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

3. Реализация выхода процесса P_i из КС

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

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

Особенности алгоритма:

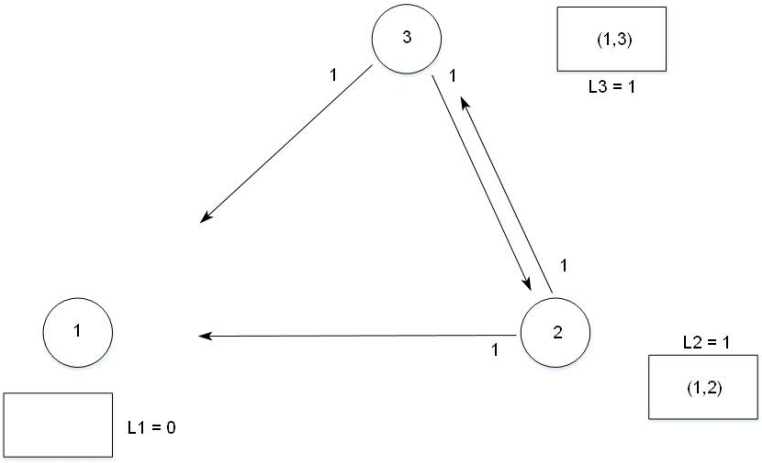
1. Рассмотренный алгоритм является распределенным, т.е. все процессы реализуют одни те же действия, каждый процесс принимает решение о входе в КС на основе своей локальной информации

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.

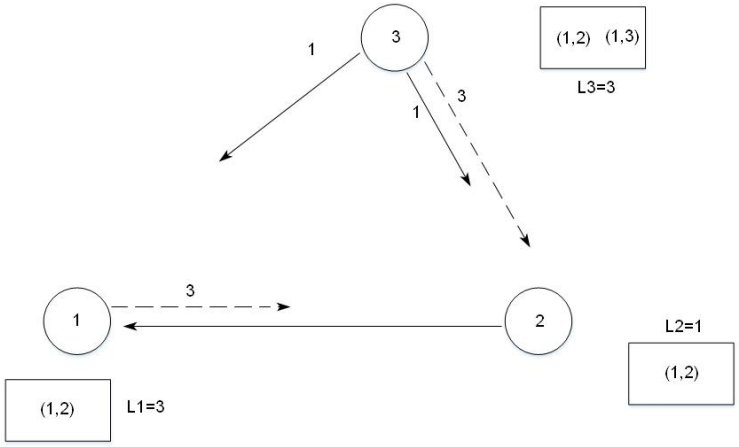
Пример реализации распределенного алгоритма синхронизации входа в критический секции

обозначение веса дуги: $\xrightarrow{L_i}$

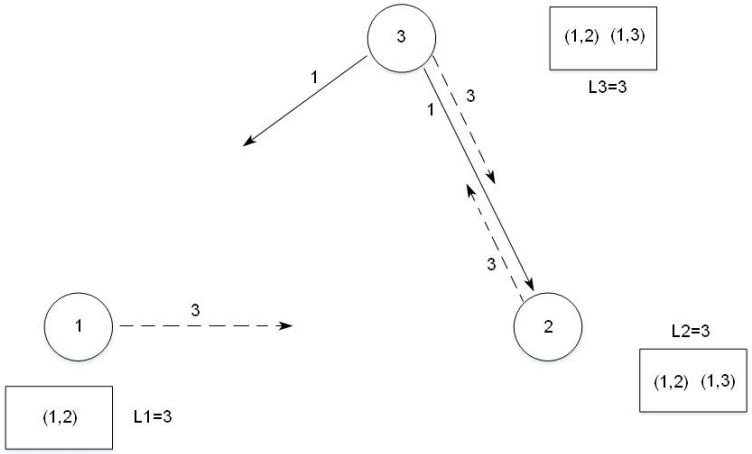
1) Процессы P_2 и P_3 одновременно запрашивают вход в КС, изменяя значения своих локальных часов L_2 и L_3



2) Процессы P_1 и P_3 получают запрос от процесса P_2 и отправляют ему ответы. Метка (1,2) меньше метки (1,3), тогда запрос P_2 помещается в голову очереди

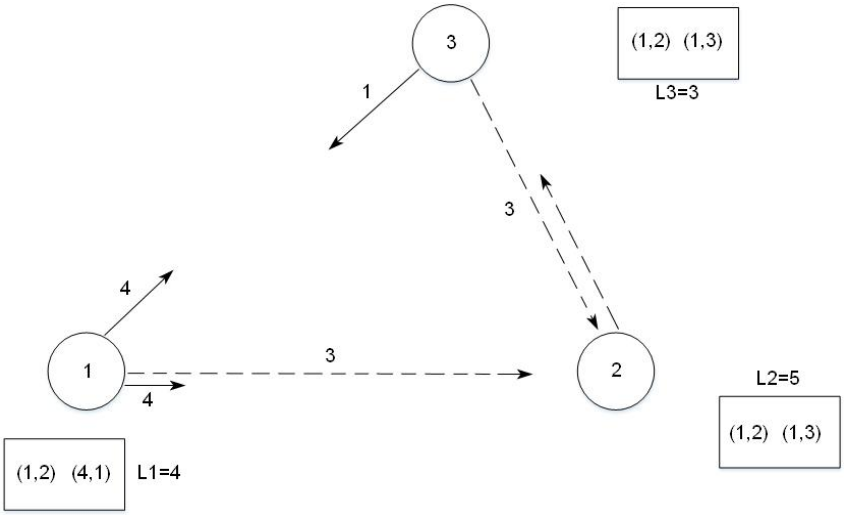


3)



Т.к. канал – это очередь $F_i F_0$ то запрос от P_3 на P_2 придет раньше, чем подтверждение от P_3 на P_2. Запрос от P_3 не дошел ни до P_1, ни до P_2.

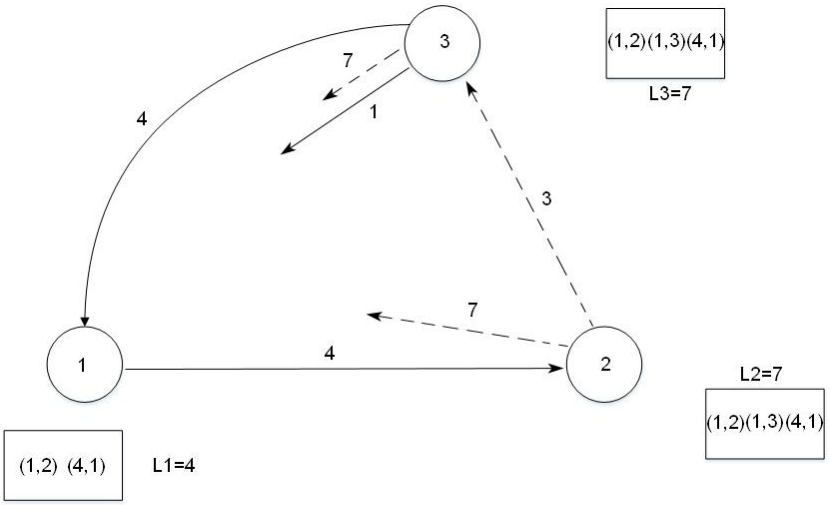
3. Процесс P_2 получает ответные сообщения от процессов P_1 и P_3, поэтому увеличивает свои часы до $L_2 = 5$.



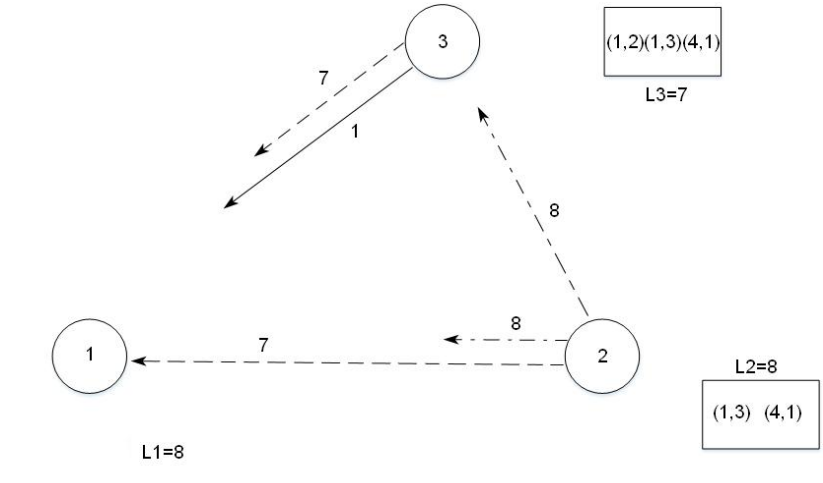
Т.к. процесс P_2 получил подтверждения процессов, и он в голове своей локальной очереди, то процесс P_2 входит в КС. Процесс P_1 рассылает запрос на вход в критическую секцию.

Процессы P_2 и P_3 посылают ответные сообщения.

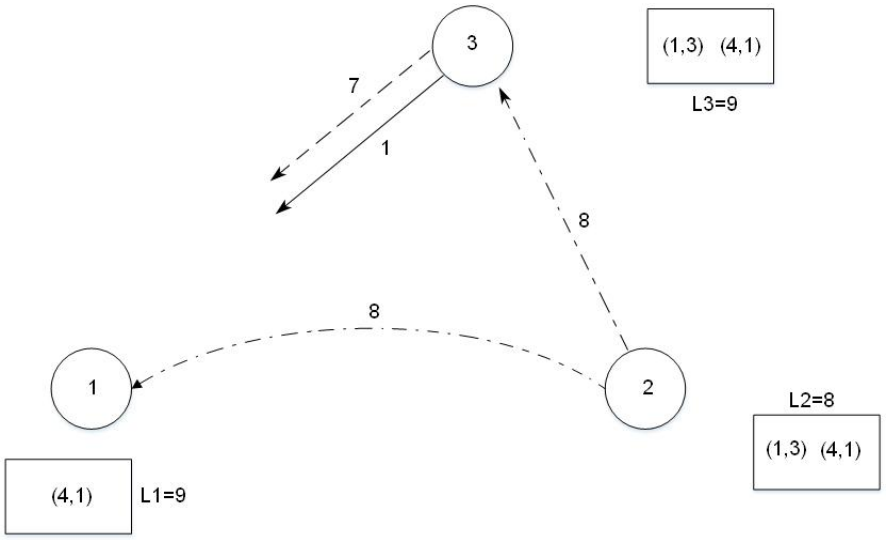
4)

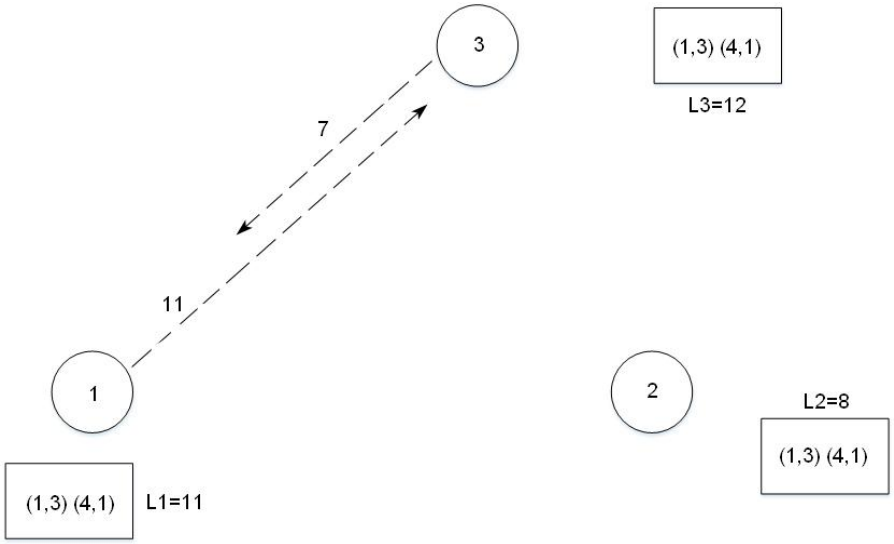


5) Освобождение ресурса



6)





11 – подтверждение 1