ОСОБЕННОСТИ ВЗАИМОДЕЙСТВИЯ ПОСРЕДСТВОМ ПЕРЕДАЧИ СООБЩЕНИЙ

- Взаимодействующие процессы не указывают идентификаторы друг друга;

- Процесс реализует отправку сообщения нескольким адресатам;

- Требуется различать типы сообщений, которыми обмениваются процессы (запрос, ответ);

В первом случае в сообщении вместо ID-отправителя указывается ID вида «Any».

Типизация сообщений

Реализация примитивов с указанием типа сообщения:

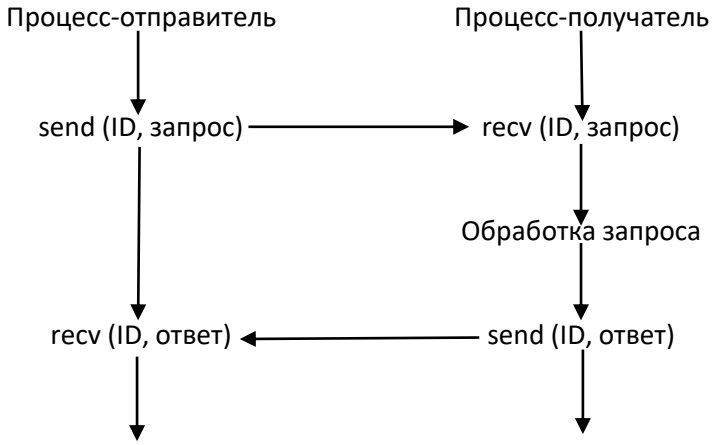
send (ID_получателя, сообщение-запрос);

recv (ID_отправителя, сообщение-запрос);

send (ID_получателя, сообщение-ответ);

recv (ID_отправителя, сообщение-ответ);

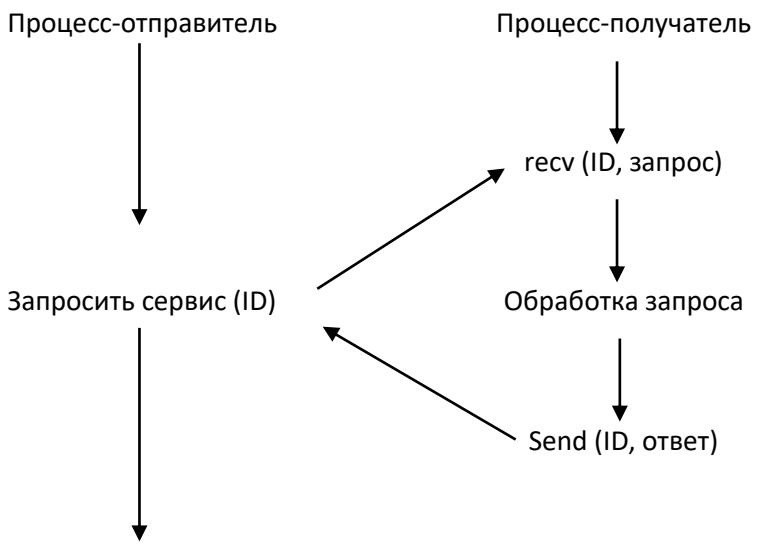
Пример взаимодействия между клиентом и сервером



Процесс – отправитель, сформировав запросы далее продолжает свое выполнение

Примитив **Запросить_сервис()** – это реализация (совместная) пары примитивов send (ID, запрос) – recv (ID,ответ) (отправитель блокируется до получения ответа).

Пример взаимодействия клиента и сервера с блокированием клиента в ситуации ответа



Широковещание и мультивещание сообщений

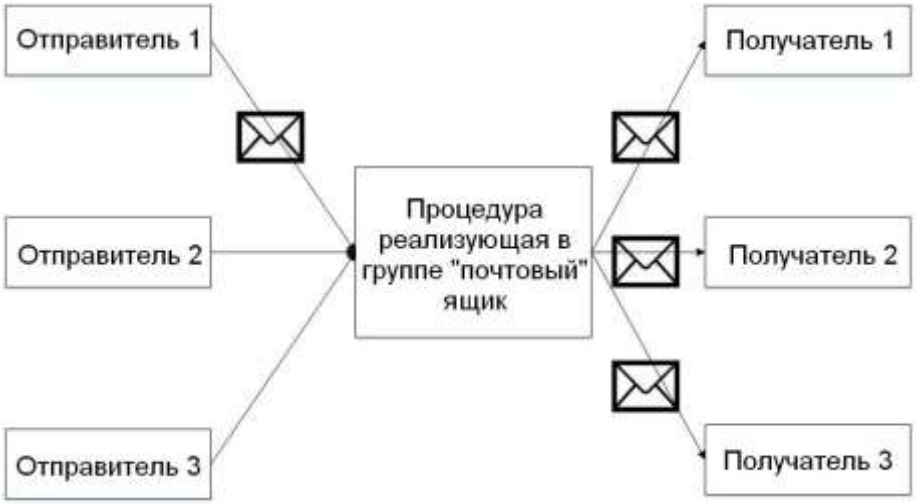
Реализуется отправка сообщения определенной совокупности процессов. Либо реализуется получение сообщений от заданной группы процессов. Перечень процессов, которым рассылаются сообщения либо от которых принимаются сообщения, предварительно д/б сформирован.

Результат – именованная группа процессов (т.е. формируется группа процессов, для которых будет реализовываться широковещание, группа именуется).

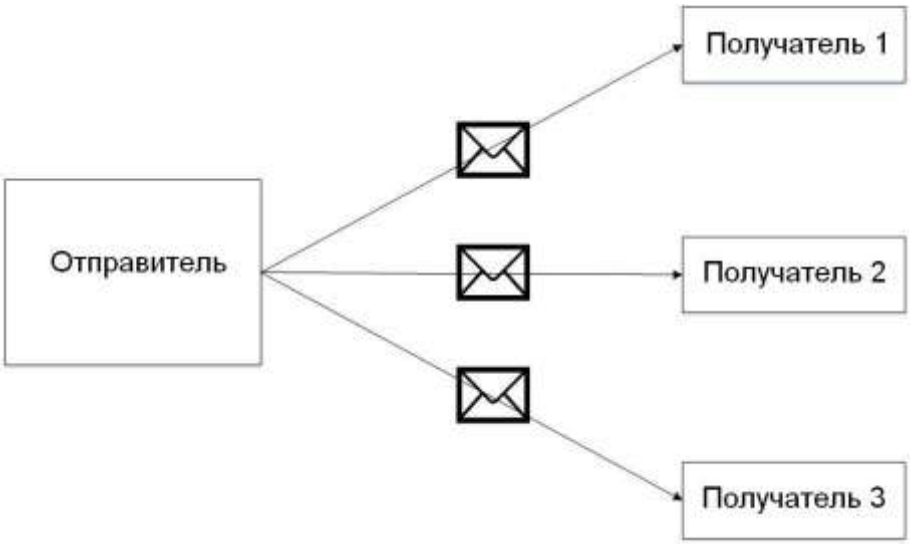
Передача сообщений группе - аналог передачи сообщения отдельному процессу. Формат сообщения, формируемого пользователем:

Имя группы / место назначения
Источник
Тип сообщения
Тело

<u>Пример широковещания (в пределах групп)</u>



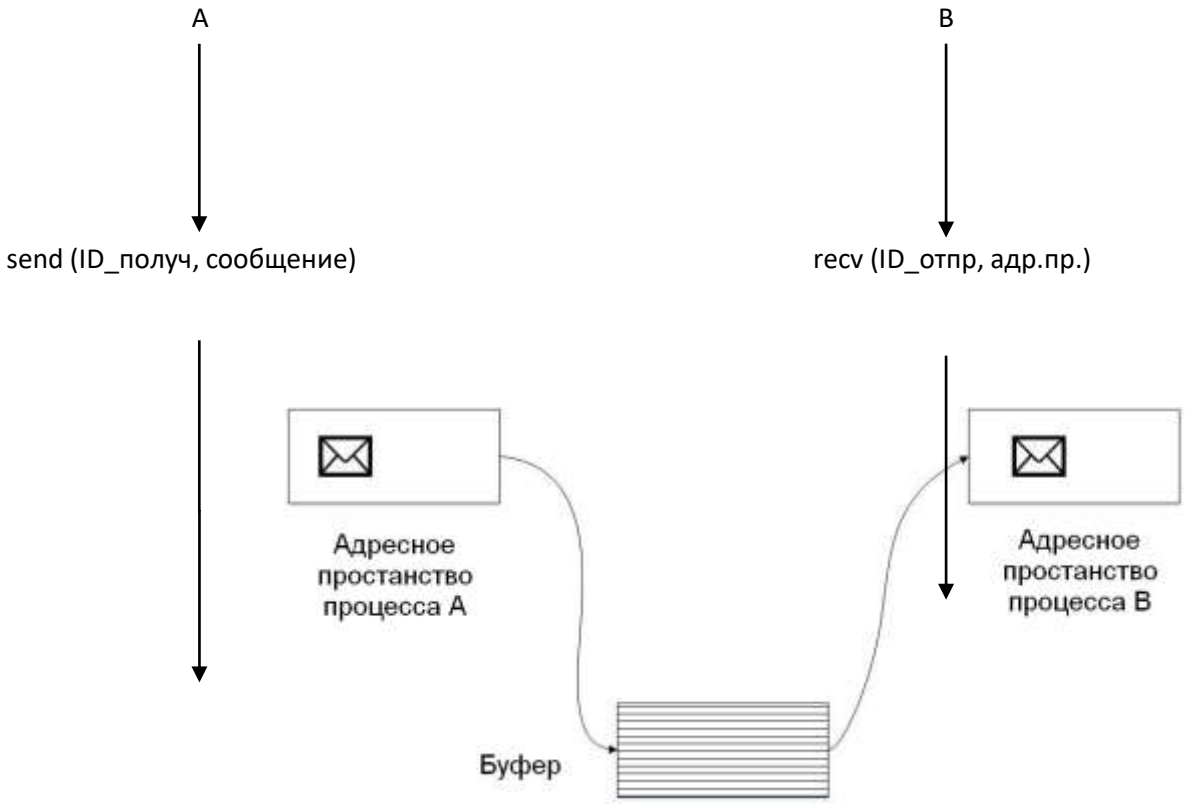
Пример мультивещания



Т.о. широковещание реализуется в пределах именованной группы процессов.

РЕАЛИЗАЦИЯ СИНХРОНИЗАЦИИ ПРИ ПЕРЕДАЧЕ СООБЩЕНИЙ

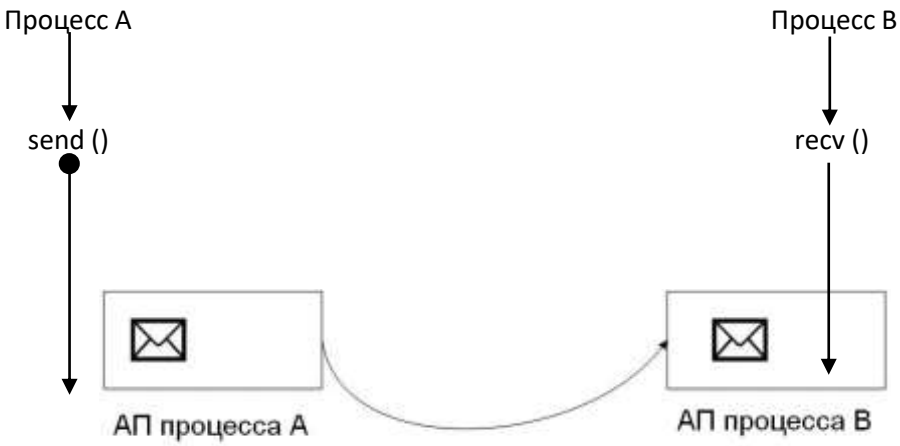
Если реализуется асинхронная передача, то выполняется двойное копирование данных (передающая сторона не блокируется).



Т.о. асинхронная передача связана с необходимостью создания буфера для хранения сообщения в случае, если процесс –приемник не готов получить данные. Также затраты связаны с копированием данных из АП процесса А в буфер и из буфера в АП процесса В.

Снижение затрат ресурсов, связанных с организацией буфера, с организацией копирования в/из буфера возможно посредством синхронной передачи (не предполагает использование буфера). Сообщение копируется от отправителя к получателю в момент синхронизации (отправитель блокируется).

Схема синхронного взаимодействия посредством передачи сообщений



● - возможное ожидание процесса отправителя

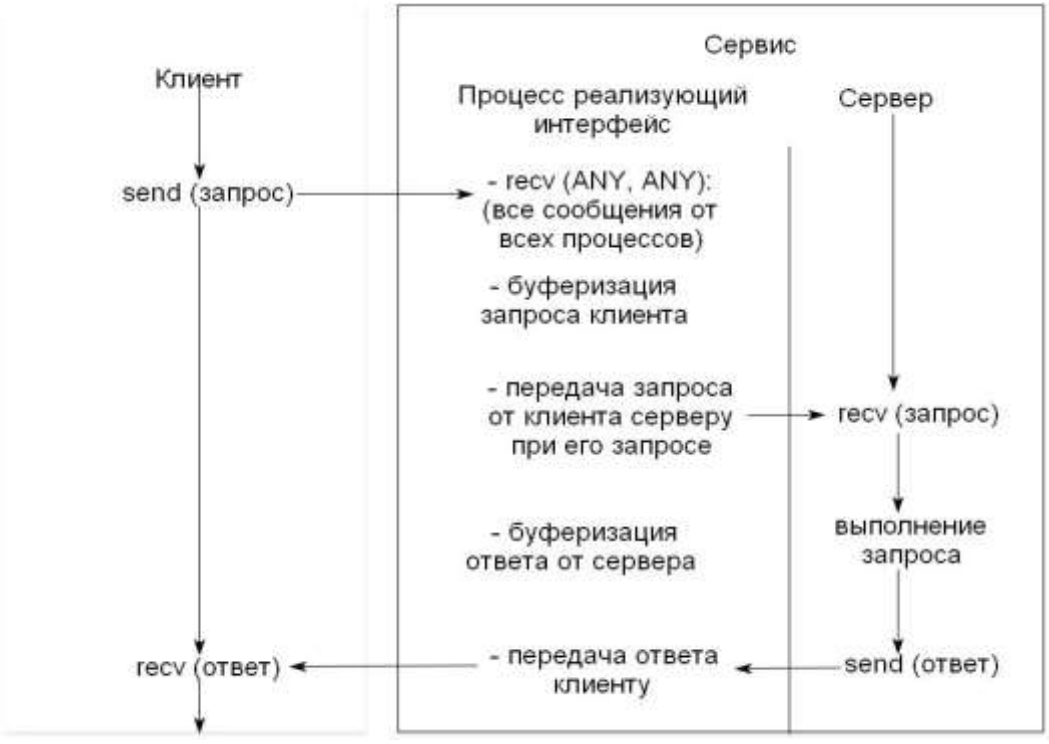
Случай, когда отправляющий процесс не может быть блокирован

Процесс является некоторым сервисом (в частности, системным сервисом), который отправил ответ клиенту на его запрос (ожидание ответа клиентам не гарантируется, в случае синхронной передачи сервис д/б заблокирован, однако заблокированным он быть не может).

Пример:

Процесс, реализующий ввод данных и передачу их клиентам (процесс, вводящий данные и передающий их клиентам, заблокирован быть не может).

Схема промежуточной буферизации запросов/ответов, позволяющая исключить блокирование при синхронной передаче



ВЗАИМОДЕЙСТВИЕ МЕЖДУ ПРОЦЕССАМИ В РАСПРЕДЕЛЕННОЙ СИСТЕМЕ

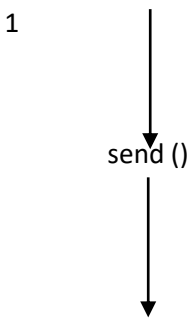
Виды взаимодействия распределено выполняющихся процессов:

1) **Взаимодействие при передаче данных**;

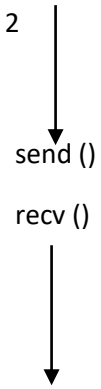
2) **Взаимодействие при синхронизации доступа к общим ресурсам**;

3) **Удаленный вызов процедур** (Remote Procedure Call, RPC) – вызывающий и вызываемый процессы находятся на разных РС. При удаленном взаимодействии (в соответствии с пунктами 1 и 2) требуется реализовывать синхронную модель.

Способы реализации синхронной модели



где send () – блокирующая передача



где send () – не блокирующая передача (ожидание подтверждения синхронизации

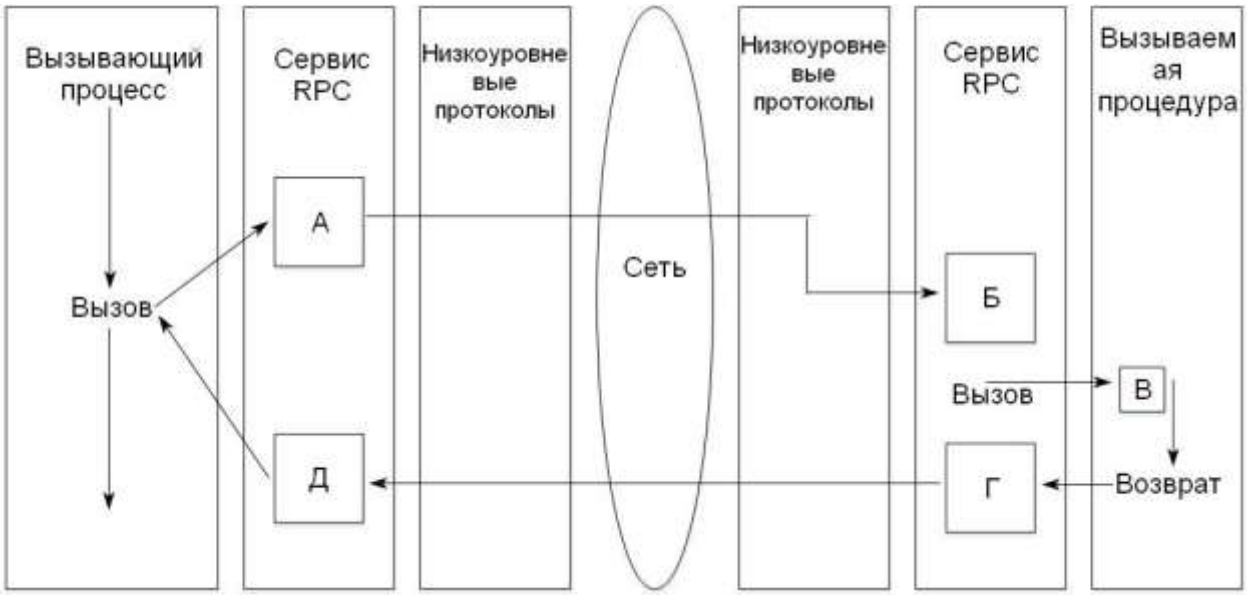
Удаленный вызов процедур (RPC) (ключевые слова: «вызов процедур»)

Вызывающий и вызываемый процессы, взаимодействующие в рамках RPC, функционируют в рамках разных АП. Поэтому взаимодействие между ними (в частности, передача аргументов в вызываемую процедуру) реализуется посредством передачи сообщений.

Механизм RPC предполагает, что вызывающий процесс передает удаленной процедуре данные, которые процедура обрабатывает. Результаты возвращаются вызвавшему процессу.

Т.о. механизм RPC реализован в системе, обеспечивающей распределенные вычисления.

Схема реализации RPC для систем, обеспечивающих распределенные вычисления



Вызывающая **Вызываемая сторона**

Вызывающий процесс реализует обращение к удаленной процедуре (вызов с использованием механизма RPC распознается как удаленный). Аргументы для вызываемой процедуры задаются обычным образом.

Т.к. процедура идентифицирована как удаленная, то управление передается механизму RPC в точку А.

Сервис (механизм) RPC реализует:

- упаковку аргументов в требуемом виде для передачи по сети;

- формирование идентификатора удаленного вызова;

Сервис RPC на вызываемой стороне реализует:

- распаковку данных в аргументы, которые могут быть переданы в вызываемую процедуру;

- фиксирует идентификатор удаленного вызова;

В соответствии с идентификатором удаленного вызова вызывающая сторона формирует подтверждение в случае получения вызывающим процессом результатов обработки.

ВЫДЕЛЕНИЕ РЕСУРСОВ И ВЗАИМОБЛОКИРОВКИ

Виды ресурсов:

1) аппаратные (процессор, память, УВВ);

2) программные (общие вызываемые процедуры);

3) объекты данных.

Граф выделения ресурсов, два вида вершин:

1) вершины-ресурсы

2) вершины-процессы

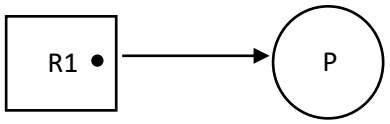
Точки – количество ресурса, которые может быть выделено процессу.

Два вида дуг:

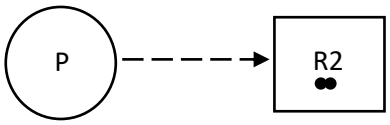
——→ ресурс выделен определенному процессу

– – – –→ ресурс запрашивается определенным процессом

Способы связи разнотипных вершин на графе выделения ресурсов

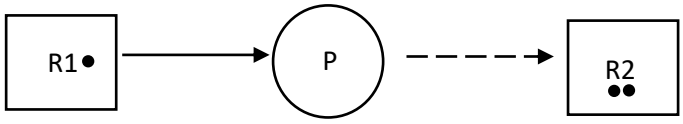


Ресурс R1 в количестве 1 выделен процессу P



Процесс P запросил ресурс R2, которого в наличии имеется 2 единицы

Граф выделения и запроса ресурса (граф взаимодействия процесса и ресурсов)



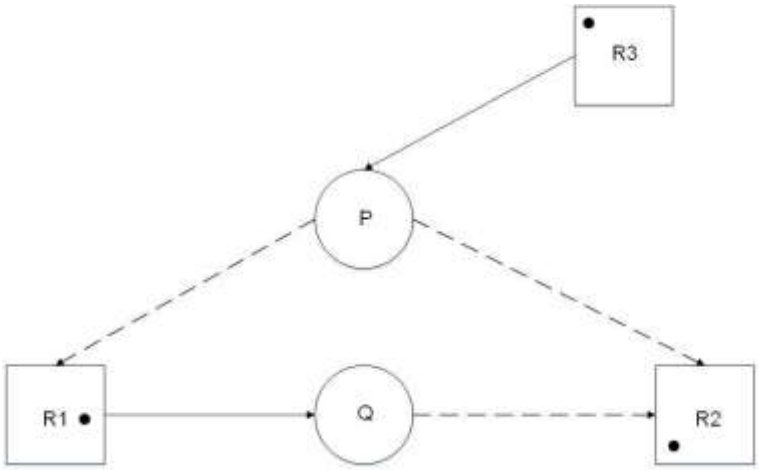
Процесс P в заблокированном состоянии, т.к. им запрошен ресурс R2, который ему не выделен.

Пример графа выделения и запроса ресурсов с возможной взаимоблокировкой

Примечание

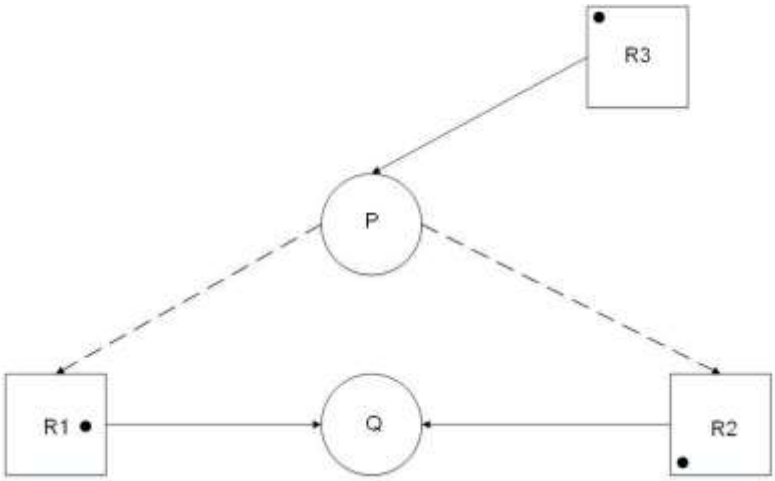
Если в графе существует цикл, в котором присутствуют по одному объекту каждого вида, то имеет место взаимоблокировка процессов.

1. Исходный граф выделения и запроса ресурсов

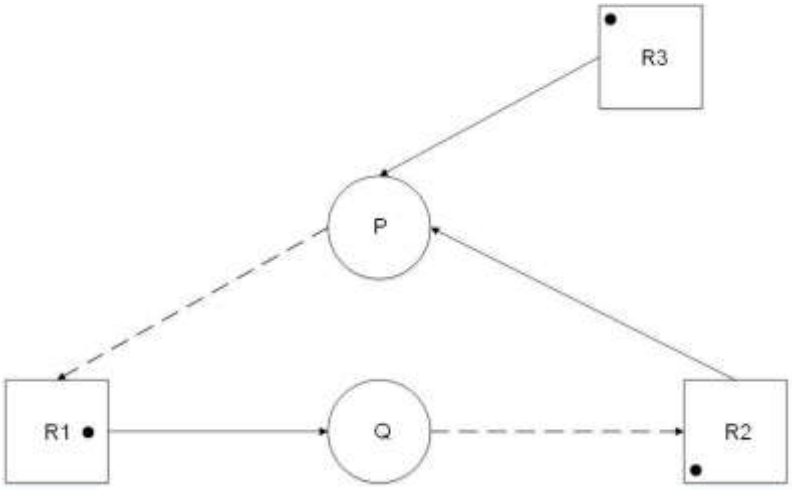


Взаимоблокировка отсутствует, но является возможной, т.к. единственной экземпляр ресурса R2 запрошен двумя процессами.

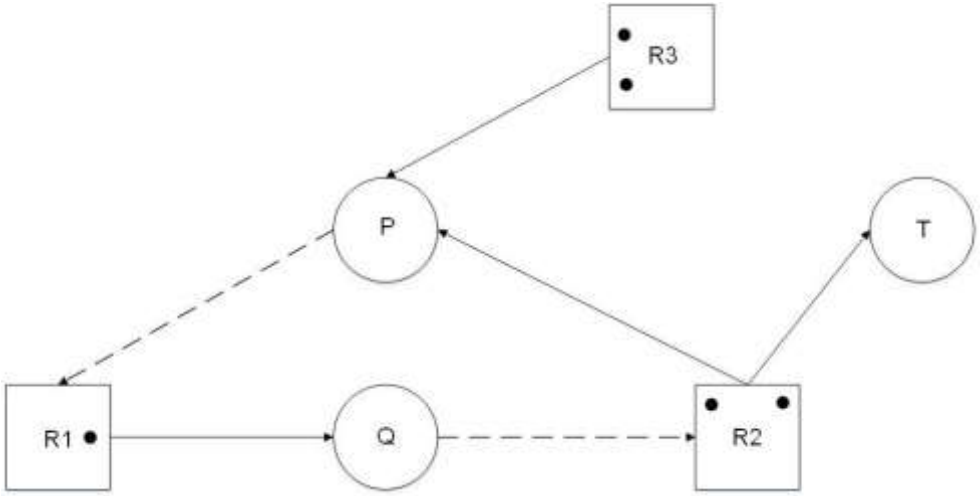
2. Модифицированный вид графа. Ресурс R2 выделен процессу Q. Взаимоблокировка отсутствует (цикл на графе отсутствует).

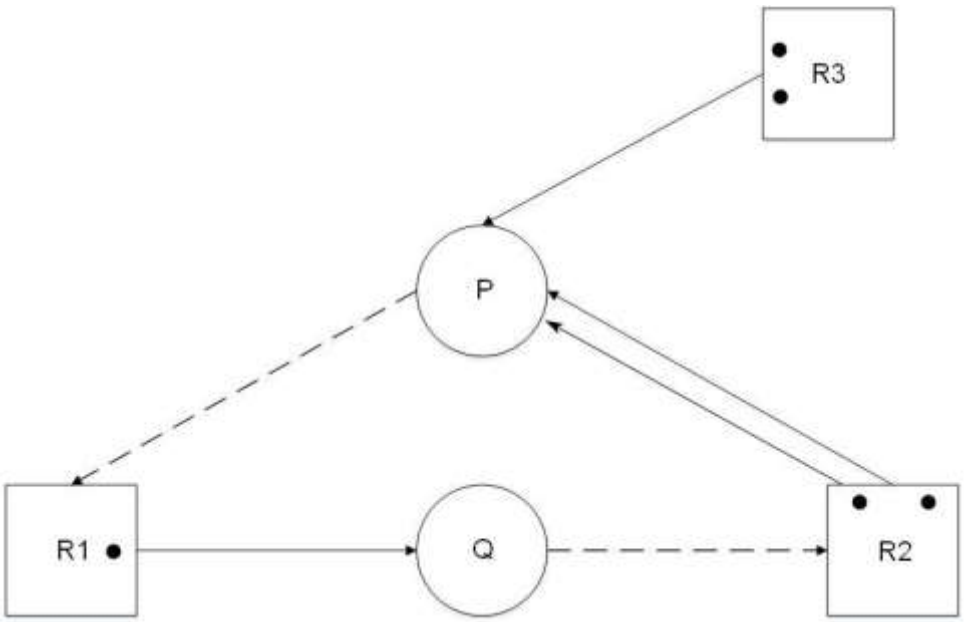


3. Модифицированный вид графа. Ресурс R2 выделен процессу P. Сформирован цикл, в котором присутствует по одному объекту каждого типа => сформирована взаимоблокировка процессов.



Пример реализации взаимоблокировки при наличии нескольких экземпляров ресурсов





ВЫЯВЛЕНИЕ ВЗАИМОБЛОКИРОВОК

С целью выявления взаимоблокировок требуется хранить информацию о запросах и о выделенных ресурсах. Известными являются типы объектов (ресурсов) и количество экземпляров каждого из них.

Система выделения ресурсов фиксирует выделенные процессом ресурсы и ресурсы, которые доступны для использования, а также информацию о запросах от процессов.

Хранение информации о выделенных ресурсах – матрица A выделенных ресурсов.

Элемент a_{ij} – число объектов j-го типа (ресурсов j-го типа), выделенных процессу i

Матрица B – матрица запросов b_{ij} – количество ресурса j-го типа, которое было запрошено i-ым процессам

Матрица выделенных ресурсов

a11	a12	...	a1m
a21	a22	...	a2m
	...		
an1	an2	...	anm

Матрица запросов

b11	b12	....	b1m
b21	b22	....	b2m
	....		
bn1	bn2	....	bnm

Общее количество ресурсов, которые будут разделены.

R = (r_1, r_2, …, r_m), где r_i – количество ресурсов i-го типа, которое может быть выделено процессам.

V = (v_1, v_1, …, v_m) – количество ресурсов, доступных к разделению между процессами на текущем шаге взаимодействия.

Пример структур для хранения данных ресурсов и запросов на ресурсы

$$A = \begin{vmatrix} 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{vmatrix} \qquad B = \begin{vmatrix} 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 \end{vmatrix}$$

R = (2 1 1 2 1)

V = (0 0 0 0 1)

Алгоритм выявления взаимоблокировки

1. Определить индексы i строк матрицы A, содержащих 0 (т.о. определяются идентификаторы процессов, которым не выделены ресурсы). Если процессу не выделены ресурсы, то он не может быть вовлечен во взаимоблокировку. Индексы соответствующих процессов исключаются из решения;

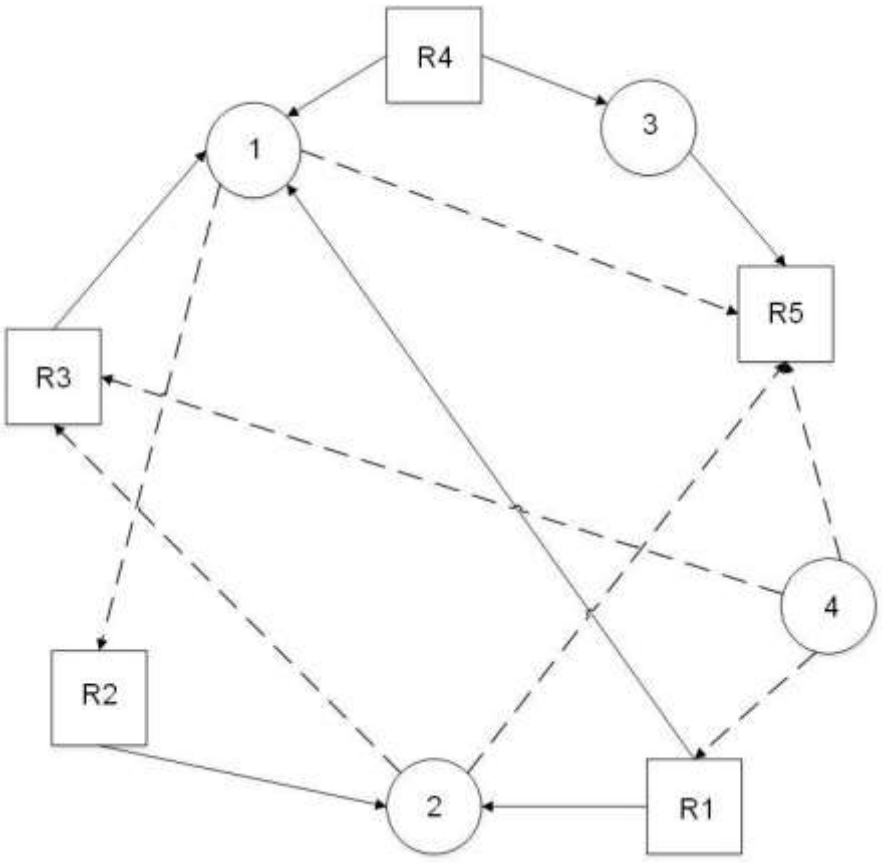
2. Инициализация текущего вектора W (используемого при реализации алгоритма): W = V;

3. Найти не помеченную строку i, для которой $B_i \leq W$ (объектиы, , которые запрошены процессом i имеются в наличие); Если строк нет, то завершение алгоритма.

4. Установить новое значение вектора W: W + Ai, исключить идентификатор процесса (строки в матрицах A и B из дальнейшего рассмотрения); Переход к шагу 3ю.

При завершении рассмотренного алгоритма не исключенные из рассмотрения номера строк матриц A и B соответствуют процессам, участвующим в взаимоблокировке.

Пример выявления взаимоблокировки. Исходный граф взаимодействия процессов и ресурсов имеет следующий вид.



Данному графу соответствуют матрицы A и B, вектора R и V следующего вида:

$$A = \begin{vmatrix} 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{vmatrix} \qquad B = \begin{vmatrix} 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 \end{vmatrix}$$

R = (2 1 1 2 1)

V = (0 0 0 1)

1.Инициализация W_0 = (0 0 0 0 1)

2.Строка 4 исключается из рассмотрения, т.к. нет выделенных ресурсу четырех ресурсов;

3. $B_3 \leq W_0$ => запросы процесса 3 могут быть удовлетворены после выделения имеющихся ресурсов.

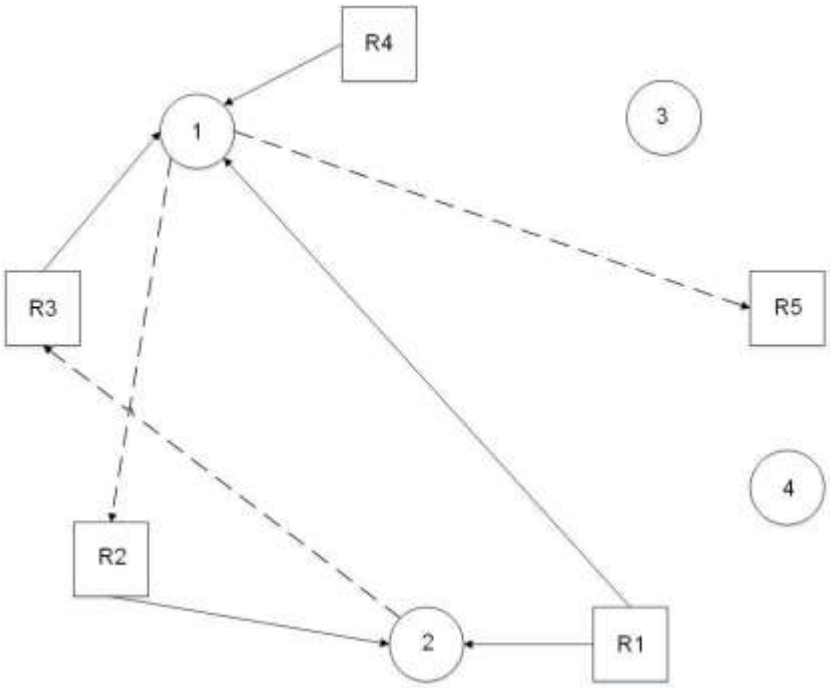
$B_3 \leq W_0$ – ресурсы, запрошенные процессом 3 могут быть ему выделены при их освобождении (идентификатор 3 исключается из рассмотрения).

Модификация вектора W_0:

$W_1 = W_0 + A_3$ = (0 0 0 1 1)

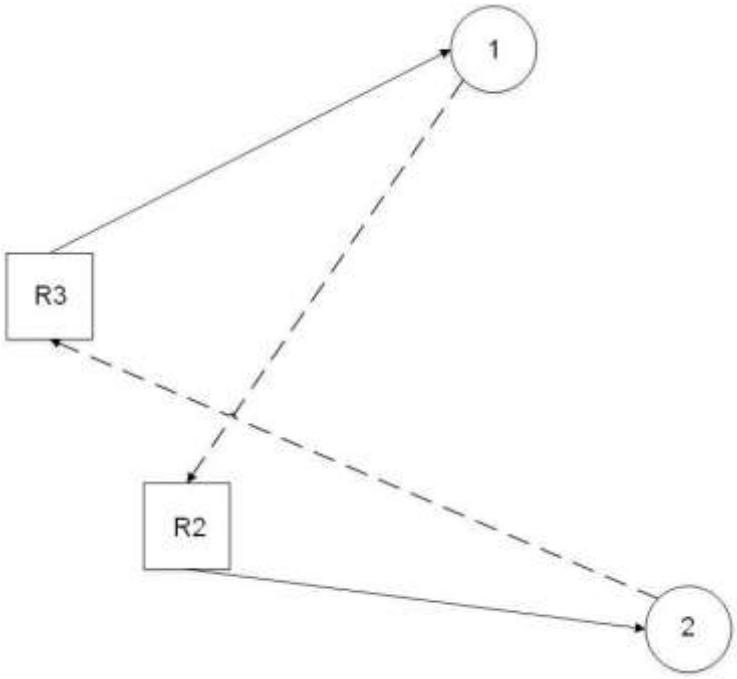
4. Т.к. ни для одной из строк матрицы B не выполняется действие $B_i \leq W_1$, то оставшимися процессами являются 1 и 2, которые взаимно блокированы.

Вид графа после шагов 2 и 3



Исключение из рассмотрения ресурса R5, т.к. он запрашивается одним процессом и при этом не является выделенным, ресурса R1, т.к. он выделен процессам 1 и 2 в полном объеме, ресурса R4, т.к. он выделен процессу 1 и не запрошен другим процессом

Итоговый вид графа процессы/ресурсы с взаимоблокировкой



Простейший способ устранения взаимоблокировки – освобождение ресурсов заблокированными процессами (либо прервать использование ресурсов избирательно).