Глобальное состояние

Курносов Михаил Георгиевич

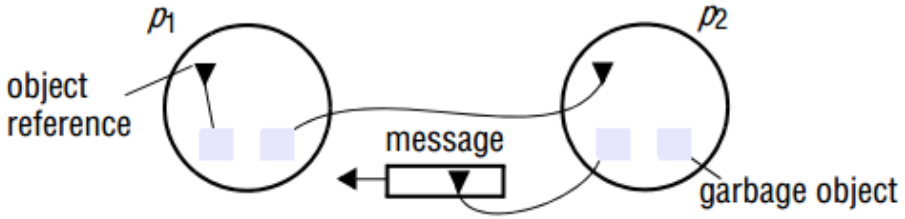
E-mail: mkurnosov@gmail.com
WWW: www.mkurnosov.net

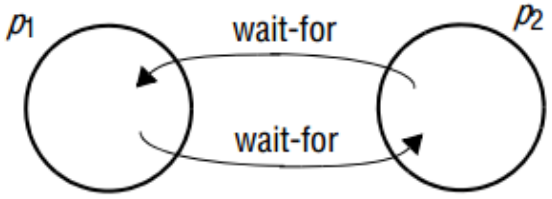
Курс «Распределенная обработка информации»
Сибирский государственный университет телекоммуникаций и информатики
Осенний семестр, 2019

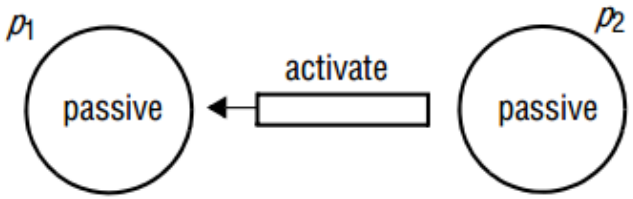
Глобальное состояние (Global state)

- **Как определить в каком состоянии находится распределенная система в заданный момент времени?**

- **Примеры**
  - ❑ Сборка мусора (Garbage collection): подсчет числа ссылок на заданный объект
  - ❑ Определение взаимной блокировки (Deadlock): обнаружение цикла в графе отношения “wait-for”
  - ❑ Определение завершения распределенного алгоритма
  - ❑ Отладка распределенного приложения

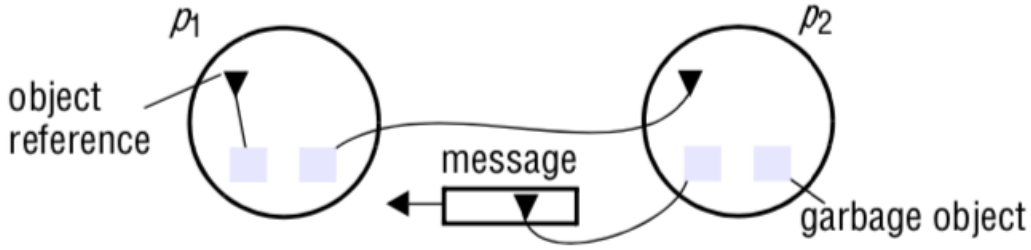






Сборка мусора (distributed garbage collection)

- Объект может быть уничтожен (освобожден) если в системе отсутствуют ссылки на него
- Сборщик мусора (garbage collector) решает задачу подсчета числа ссылок на объект в распределенной системе (определение факта отсутствия ссылок)
- **Пример**
- Процесс 1 владеет двумя объектами, на которые ссылается он и процесс 2

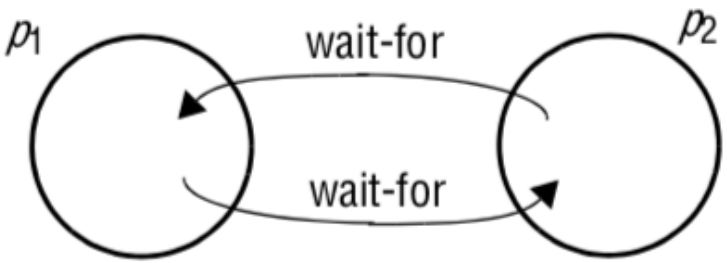


- Процесс 2 имеет два объекта: один без ссылок на него, второй – имеет ссылки из сообщения, которое передается по каналу связи

Обнаружение взаимной блокировки (deadlock detection)

- Взаимная блокировка возникает когда каждый процесс ждет ответа другого для передачи сообщения и в графе отношения ”ждет-для” присутствует цикл

```
next = (rank + 1) % commsize
prev = (rank – 1 + commsize) % commsize
MPI_Ssend(bufout, 1, MPI_INT, next, 0)
MPI_Recv(bufin, 1, MPI_INT, prev, 0)
```



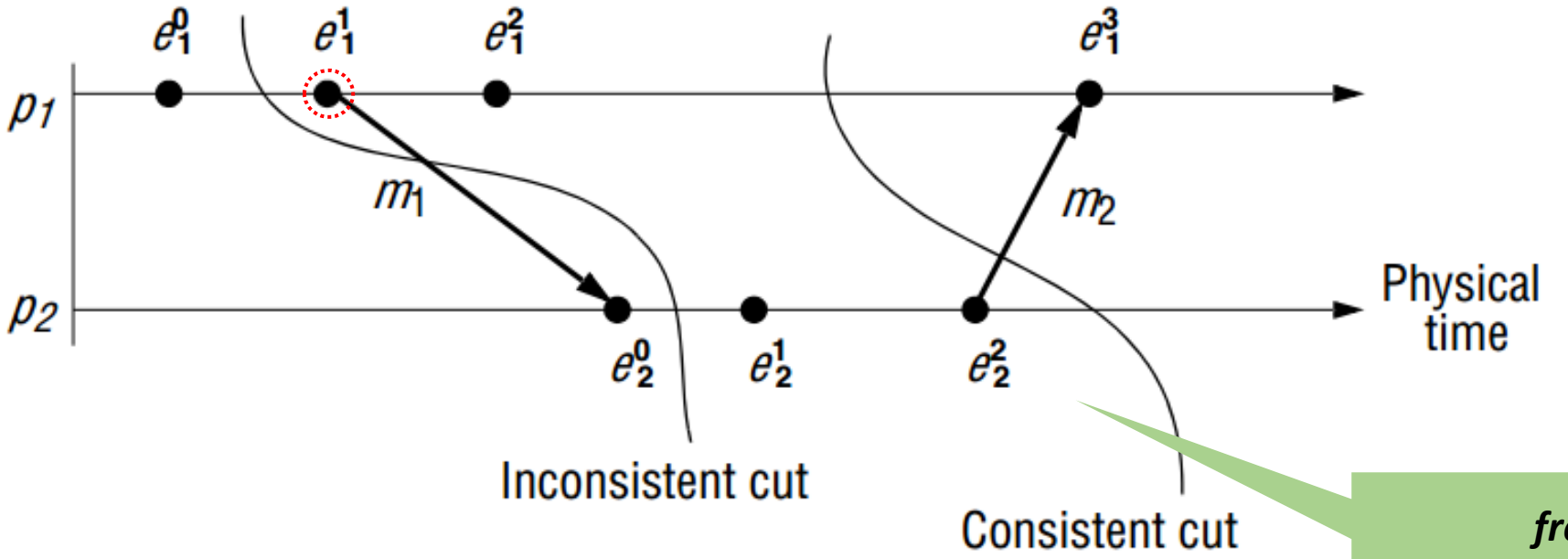
Глобальное состояние (Global state)

- **Глобальное состояние $S = (S_1, S_2, \ldots, S_n)$** – это совокупность состояний процессов распределенной системы

- **Присутствуют глобальные часы**
  - При наличии глобальных часов процессы договариваются о моменте времени *T* и достигнув его сохраняют свои состояния

- **Глобальных часов нет**
  - Можно ли корректно “собрать” глобальное состояние из локальных состояний процессов в отсутствии глобальных часов?

Глобальное состояние (Global state)



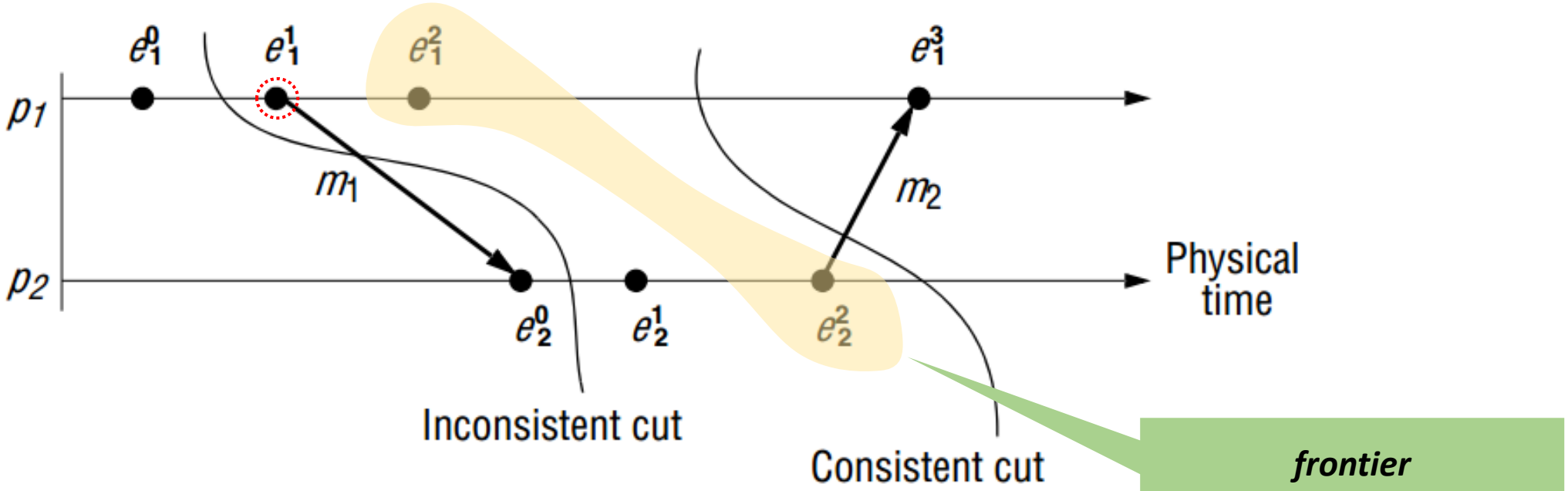
frontier
пограничные события

- Обозначим через $\boldsymbol{h_i^k} = (\boldsymbol{e_i^0}, \boldsymbol{e_i^1}, \ldots, \boldsymbol{e_i^k})$ историю *k* событий процесса p_i
- **Глобальная история** *H* системы – объединение историй процессов

$$H = h_0 \cup h_1 \cup \cdots \cup h_{p-1}$$

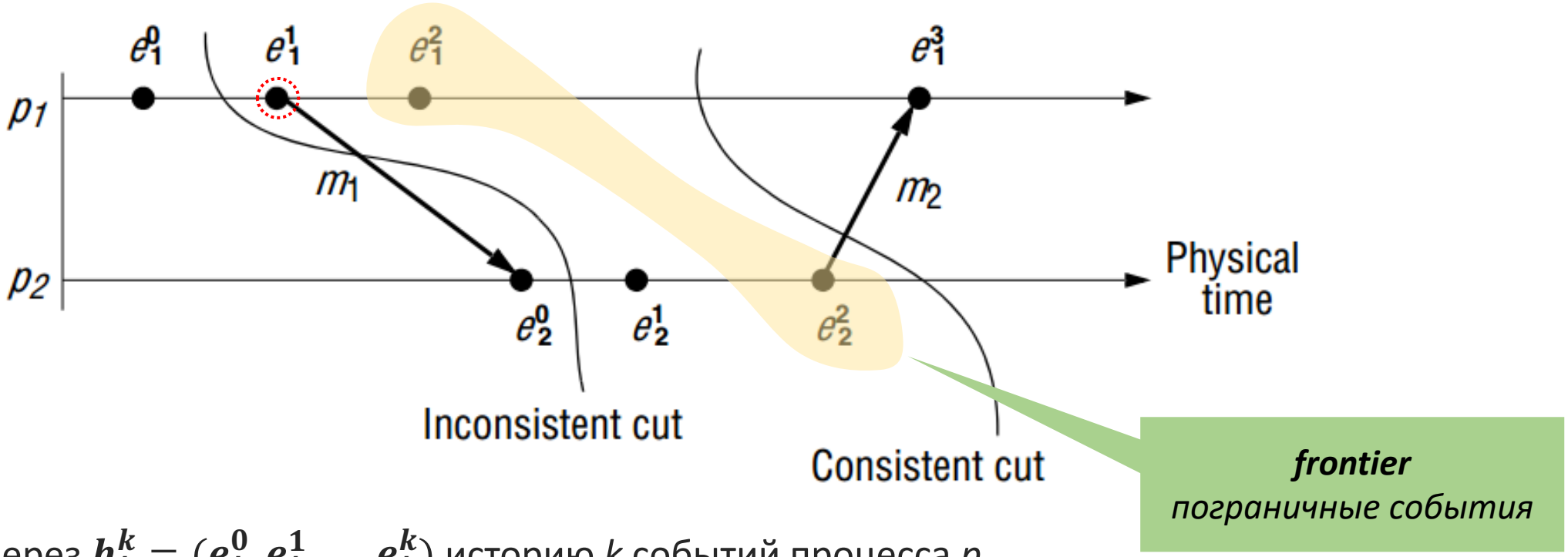
- **Глобальное состояние** *S* (global state) – состояния *p* процессов

Глобальное состояние (Global state)



- Обозначим через $\boldsymbol{h_i^k} = (\boldsymbol{e_i^0}, \boldsymbol{e_i^1}, \dots, \boldsymbol{e_i^k})$ историю *k* событий процесса p_i
- ***Сечение*** (срез, cut) $\boldsymbol{C} = \boldsymbol{h_1^{c_1}} \cup \boldsymbol{h_2^{c_2}} \cup \cdots \cup \boldsymbol{h_n^{c_n}}$ – подмножество глобальной истории событий

Глобальное состояние (Global state)



- Обозначим через $\boldsymbol{h_i^k} = (\boldsymbol{e_i^0}, \boldsymbol{e_i^1}, \dots, \boldsymbol{e_i^k})$ историю *k* событий процесса p_i
- ***Сечение*** (срез, cut) $\boldsymbol{C} = \boldsymbol{h_1^{c_1}} \cup \boldsymbol{h_2^{c_2}} \cup \cdots \cup \boldsymbol{h_n^{c_n}}$ – подмножество глобальной истории событий
- ***Согласование сечение*** (consistent cut) – это сечение, в котором для любого события *e* содержится событие предшествующее ему
- ***Согласованное глобальное состояние*** (consistent global state) – это глобальное состояние соответствующее согласованному сечению

Построение моментального снимка (Snapshot)

- **Моментальный снимок** (snapshot) – это согласованное глобальное состояние системы, включающее:
  - ❑ состояния *n* процессов
  - ❑ состояния каналов связи между процессами
- На самом деле состояния могли не иметь место одновременно, но соответствуют согласованному срезу

Алгоритм Chandy-Lamport (1985)

- K. Mani Chandy, Leslie Lamport. **Distributed Snapshots: Determining Global States of Distributed Systems** // 1985, http://research.microsoft.com/users/lamport/pubs/chandy.pdf
- **Имеется** *n* процессов
- Процессы и каналы связи между ними абсолютно надежны
- Каналы односторонние и доставляют сообщения в FIFO-порядке
- Граф из процессов и каналов между ними связный

Алгоритм Chandy-Lamport (1985)

- K. Mani Chandy, Leslie Lamport. **Distributed Snapshots: Determining Global States of Distributed Systems** // 1985, http://research.microsoft.com/users/lamport/pubs/chandy.pdf
- **Требуется** построить согласованное глобальное состояние процессов (snapshot)
- Любой процесс может инициировать построение глобального снимка в любой момент времени
- Процессы могут продолжать свое выполнение и обмениваться сообщениями во время построения снимка

Алгоритм Chandy-Lamport (1985)

- Каждый процесс имеет исходящие (outgoing) и входящие (incomming) каналы
- Каждый процесс записывает свои состояния и сообщения, присылаемые ему
- Если процесс p_i отправил сообщение m процессу p_j, но процесс p_j его еще не получил, то m учитывается как состояние канала между процессами
- Алгоритм использует *сообщение-маркер (marker)*:
  - ❑ Информирует получателя о необходимости сохранить свое состояние
  - ❑ Запускает или заканчивает процедуру записи входящих сообщений
- Алгоритм запускает любой процесс в любой момент времени
  - ❑ Запуск – прием маркера по несуществующему каналу

Алгоритм Chandy-Lamport Snapshot

Marker receiving rule for process p_i

On receipt of a *marker* message at p_i over channel c:

if (p_i has not yet recorded its state) it

records its process state now;

records the state of c as the empty set;

turns on recording of messages arriving over other incoming channels;

else

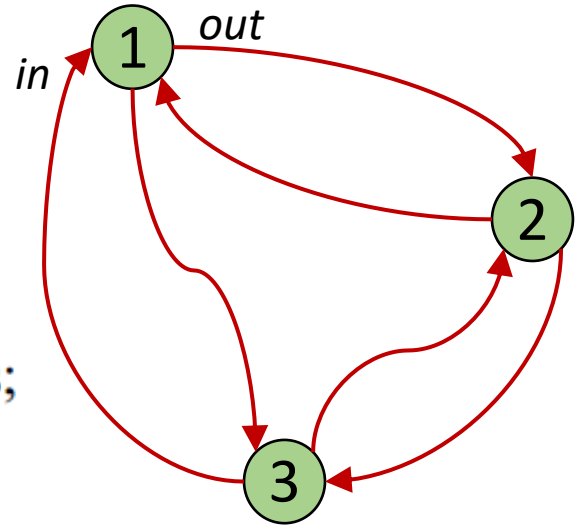
p_i records the state of c as the set of messages it has received over c since it saved its state.

end if

Marker sending rule for process p_i

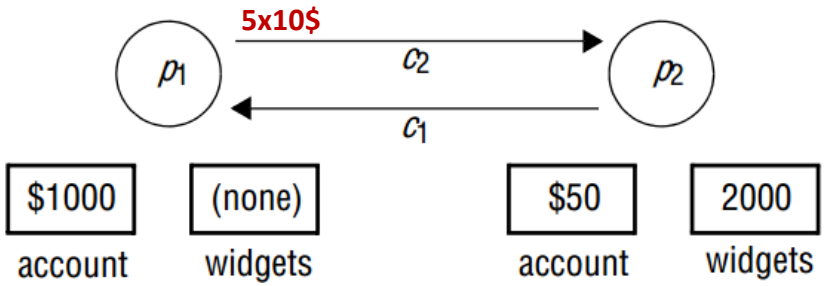
After p_i has recorded its state, for each outgoing channel c:

p_i sends one marker message over c (before it sends any other message over c).



Любой процесс запускает алгоритм и действует по правилу “receiving rule” (принял маркер по несуществующему каналу)

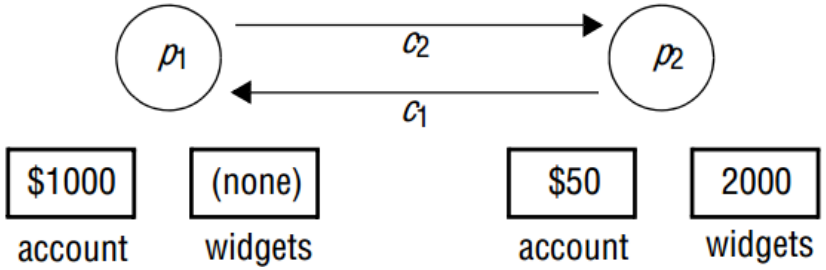
Алгоритм Chandy-Lamport



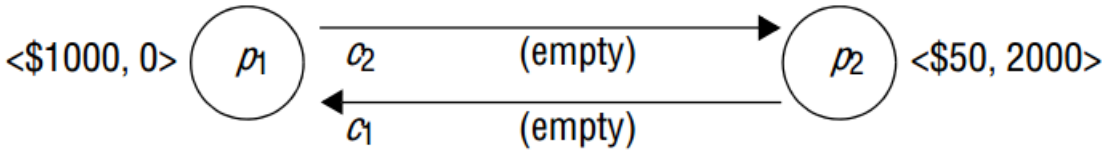
Состояние системы:
p2 получил платеж,
p1 ожидает товар (передается по каналу)

- Система из двух процессов, соединенных двумя однонаправленными каналами
- Процессы ведут торги
- Процесс p1 отправил запрос p2 на 5 предметов по 10$
- Процесс p2 должен ответить – передать по каналу c1 5 предметов по 10$

Алгоритм Chandy-Lamport

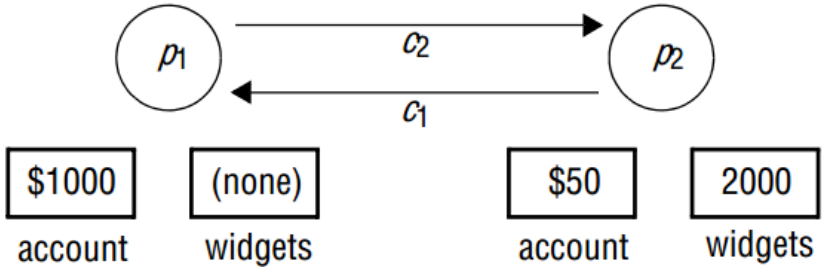


1. Global state S_0

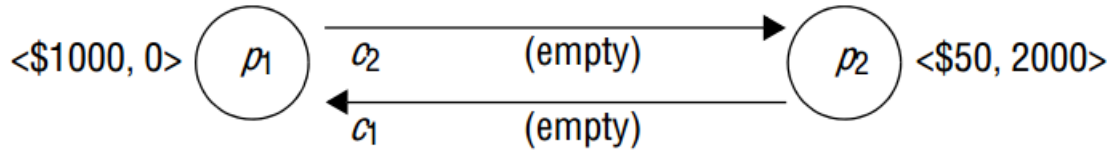


- **Процесс p1 запустил создание снимка:**
  - ❑ записал свое состояние S_0 = <$1000, 0>
  - ❑ начал записывать входящие сообщения

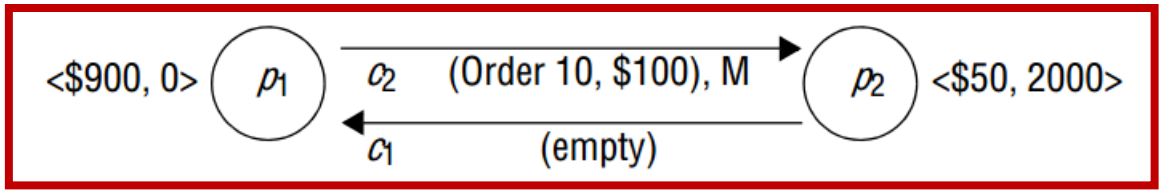
Алгоритм Chandy-Lamport



1. Global state S_0

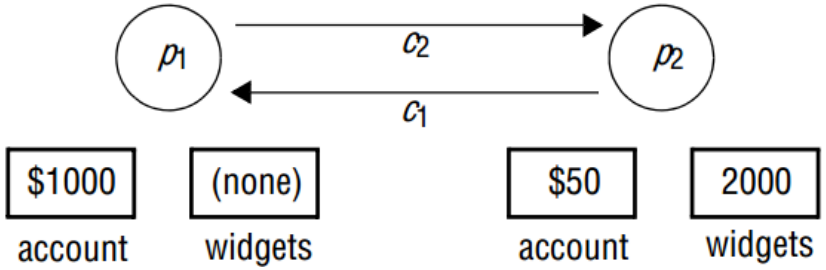


2. Global state S_1

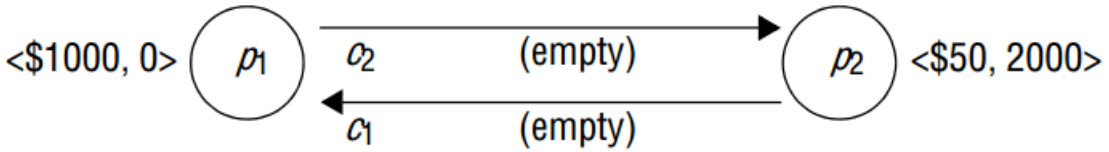


- **Процесс p1 запустил создание снимка:**
  - ❑ записал свое состояние S_0 = <$1000, 0>
  - ❑ начал записывать входящие сообщения
  - ❑ отправил маркер через *c*2
  - ❑ отправил заказ на 10 шт. (100$) => S_1

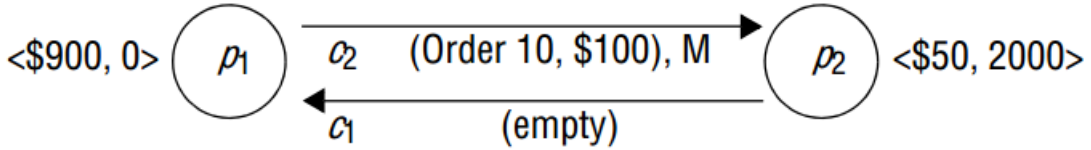
Алгоритм Chandy-Lamport



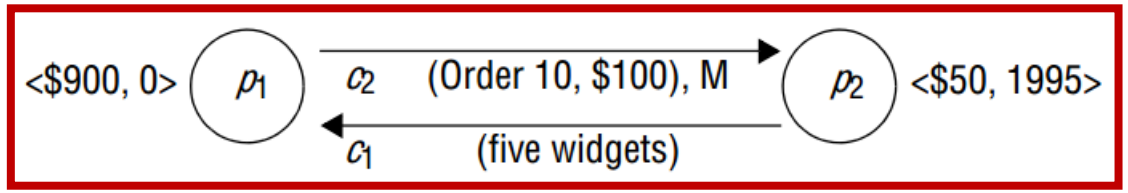
1. Global state S_0



2. Global state S_1

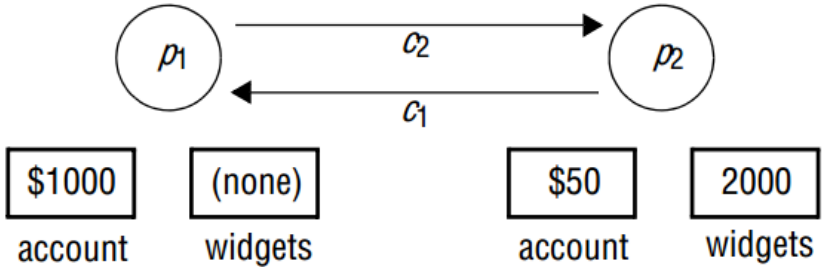


3. Global state S_2

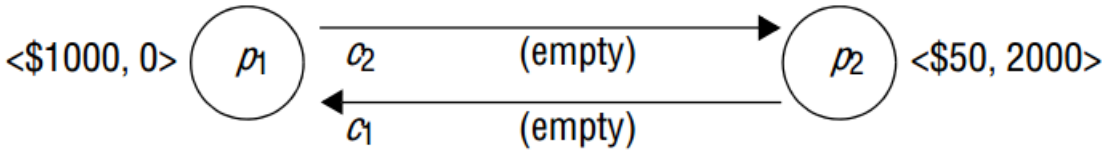


- Процесс p1 запустил создание снимка:
  - ❑ записал свое состояние S_0 = <$1000, 0>
  - ❑ начал записывать входящие сообщения
  - ❑ отправил маркер через *c*2,
  - ❑ отправил заказ на 100$ => S_1
- **Процесс p2 получил запрос от p1**
  - ❑ отправил в ответ 5 предметов через c1 => ***S_2***
  - ❑ получил маркер через c2,
    сохранил свое состояние <$50, 1995> и c2 = <>
  - ❑ отправил маркер через c1

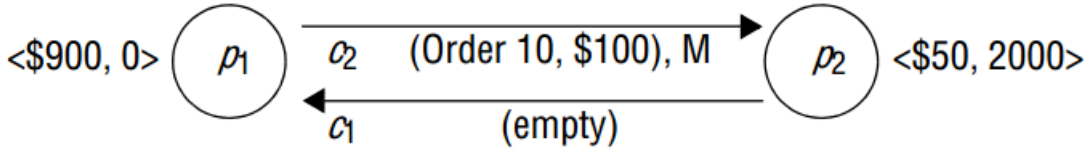
Алгоритм Chandy-Lamport



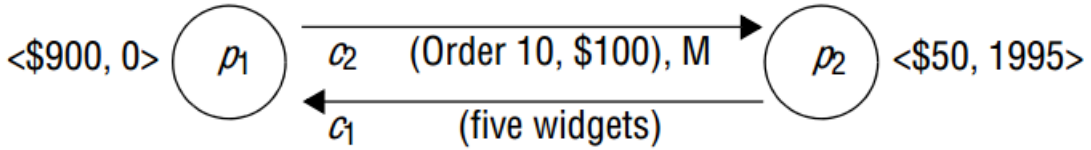
1. Global state S_0



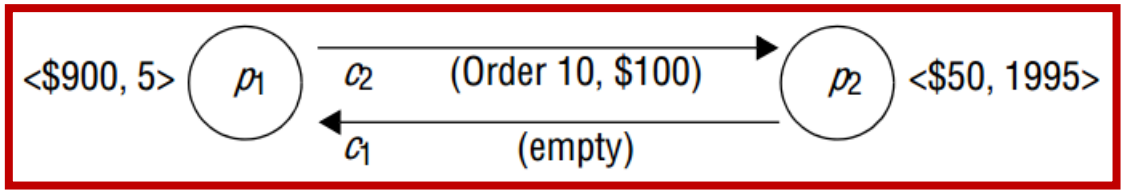
2. Global state S_1



3. Global state S_2



4. Global state S_3

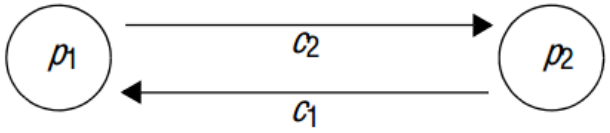


(M = marker message)

- Процесс p1 запустил создание снимка:
  - ❑ записал свое состояние S_0 = <$1000, 0>
  - ❑ начал записывать входящие сообщения
  - ❑ отправил маркер через *c*2,
  - ❑ отправил заказ на 100$ => S_1
- **Процесс p2 получил запрос от p1**
  - ❑ отправил в ответ 5 предметов через c1 => S_2
  - ❑ получил маркер через c2,
    сохранил свое состояние <$50, 1995> и c2 = <>
  - ❑ отправил маркер через c1
- **Процесс p1 получил маркер от p2**
  - ❑ записал состояние канала c1 = <5 widgets>

State: p1 <$1000, 0>, p2 <$50, 1995>, c1 = <5>, c2 = <>

Алгоритм Chandy-Lamport

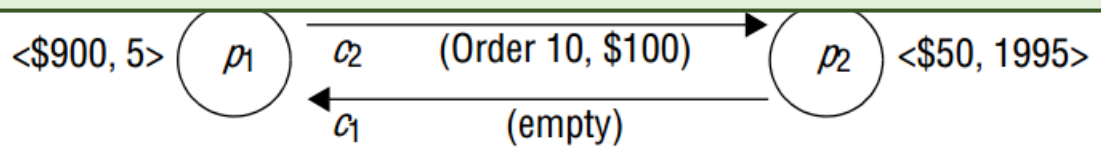


Snapshot

p_1 = <$1000, 0>; p_2 = <$50, 1995>; c_1 = <5 предметов>; c_2 = <>

Как собрать в одном процессе сохранённые состояния всех?
(All-to-one broadcast, gather)

4. Global state S_3



(M = marker message)

- **Процесс p1 получил маркер от p2**
  - ❑ записал состояние канала c1 = <5 widgets>

State: p1 <$1000, 0>, p2 <$50, 1995>, c1 = <5>, c2 = <>

Глобальное состояние (Global state)

- Если каналы не обеспечивают передачи сообщений в порядке FIFO
  - Алгоритм Лая-Янга (Lai, Yang, 1987)
- Области применения
- Обнаружение тупиков (deadlock) – алгоритм глобальной разметки