Топологии

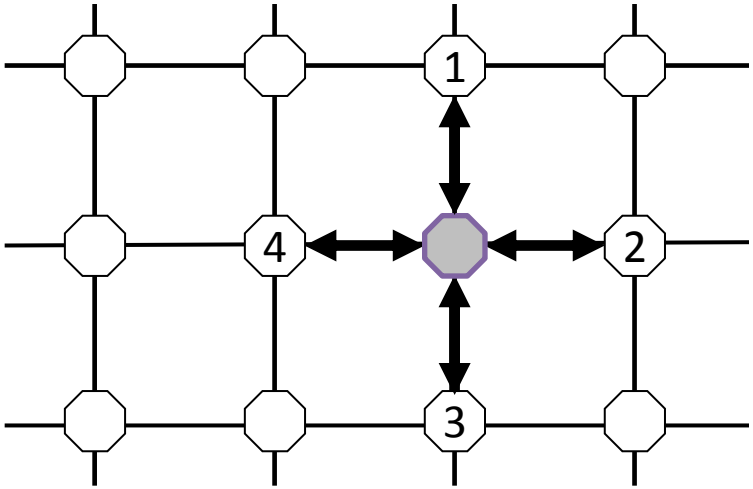
Практическое занятие #2

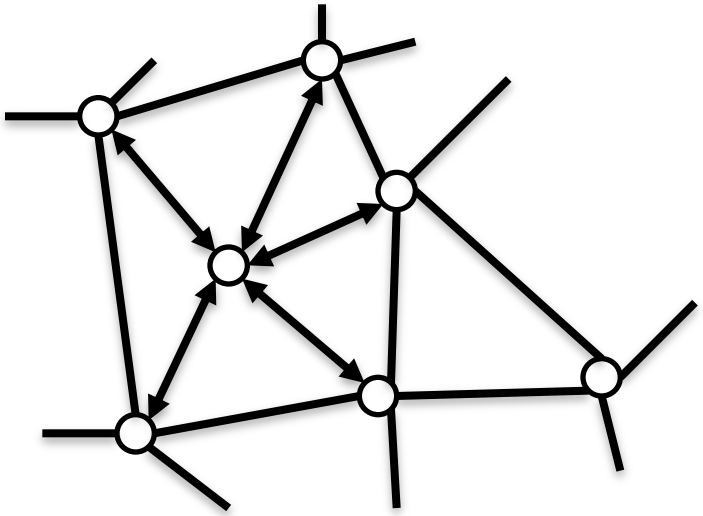
Группы и коммуникаторы MPI

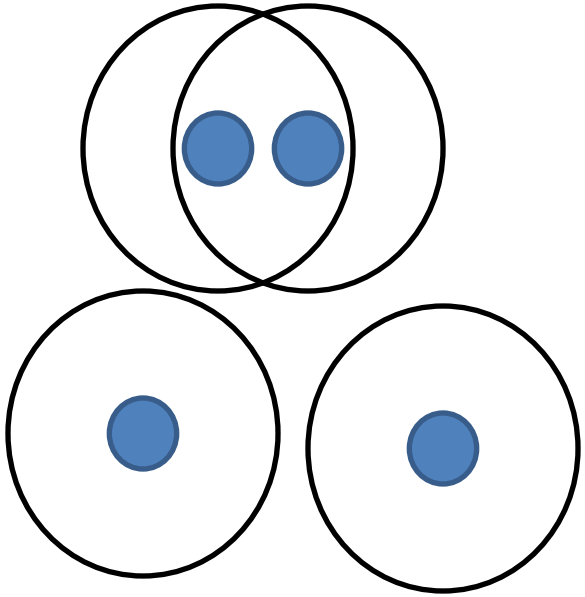
Введение в топологии OST

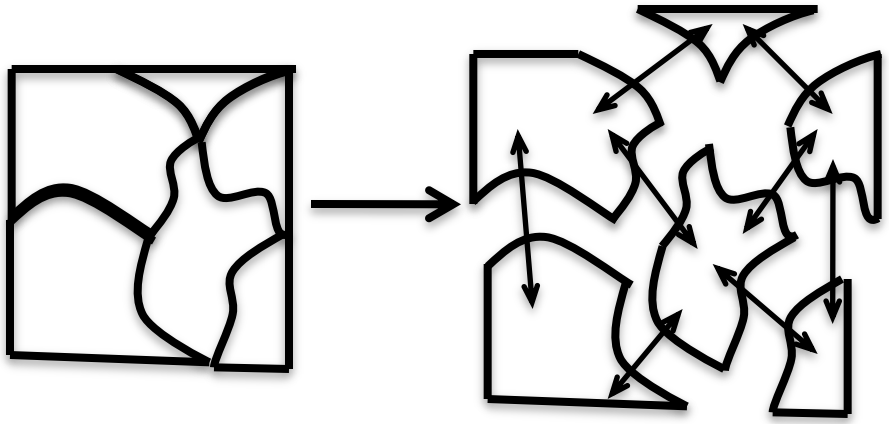
ПОНЯТИЕ СВЯЗЕЙ

Примеры связей









ГРУППЫ И КОММУНИКАТОРЫ MPI

Общая схема MPI программы

```
#include<mpi.h>
int main(int argc,char **argv){
    int rank, size;

    //Инициализация работы с MPI
    MPI_Init(&argc, &argv);

    //Параллельная часть

    //Завершение работы с MPI
    MPI_Finalize();

    return 0;
}
```

Группы MPI

Группа MPI – подмножество процессов

- Возможно выполнять независимые части задачи
- Перенумерация процессов в пределах группы
- Может использоваться для задания топологий

Группы MPI

```
// Пусть size = 8

MPI_Group orig_group, new_group;
int ranks1[4]={0,1,2,3}, ranks2[4]={4,5,6,7};

MPI_Comm_group(MPI_COMM_WORLD, &orig_group);

if (rank < size / 2) {
  MPI_Group_incl(orig_group, size / 2,
                 ranks1, &new_group);
} else {
  MPI_Group_incl(orig_group, size / 2,
                 ranks2, &new_group);
}
```

Группы и коммуникаторы MPI

```
MPI_Group_rank (new_group, &group_rank);
MPI_Group_size(new_group, &group_size);

MPI_Comm new_comm;
MPI_Comm_create(MPI_COMM_WORLD,
                new_group, &new_comm);

MPI_Comm_rank(new_comm, &comm_rank);
MPI_Comm_size(new_comm, &comm_size);

MPI_Barrier(new_comm);

printf(“group %d/%d | comm %d/%d\n",
        group_rank, group_size,
        comm_rank, comm_size);
```

ДЕМОНСТРАЦИЯ

Демонстрация

```
global 0/6 | group 0/3 | comm 0/3
global 1/6 | group 1/3 | comm 1/3
global 2/6 | group 2/3 | comm 2/3
global 3/6 | group 0/3 | comm 0/3
global 4/6 | group 1/3 | comm 1/3
global 5/6 | group 2/3 | comm 2/3
```

Топологии в OST

Топология – описание окрестностей каждой точки в пространстве объектов.

Топология в OST задается классом, в котором определена

- Функция близости пары точек
- Функция описания окрестности точки

Функция близости

Функция близости – булева функция, которая для пары точек A и B проверяет принадлежность точки B окрестности точки A.

```
class topology(ost.Topology.Abstract):
  def proximity(self, p1, p2):
    # p1, p2 - наборы(массивы) координат
    # рассматриваемых точек
    return <True/False>
```

Функция описания окрестности

Функция описания окрестности для некоторой точки пространства возвращает в виде списка координаты всех точек, которые входят в ее окрестность

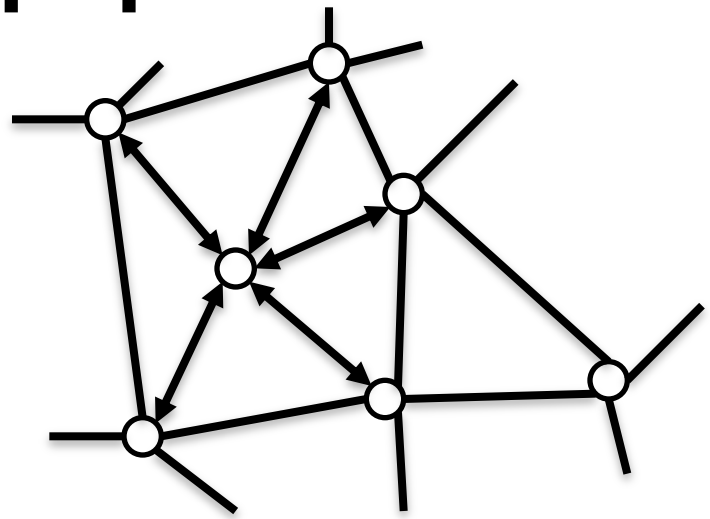
```
class topology(ost.Topology.Abstract):
  def neighborhood(self, p):
    # p - набор(массив) координат,
    # рассматриваемой точки
    return [ [p1_x1,...,p1_xn1],
             ....,
             [pk_x1,...,pk_xnk] ]
```

Функция описания окрестности

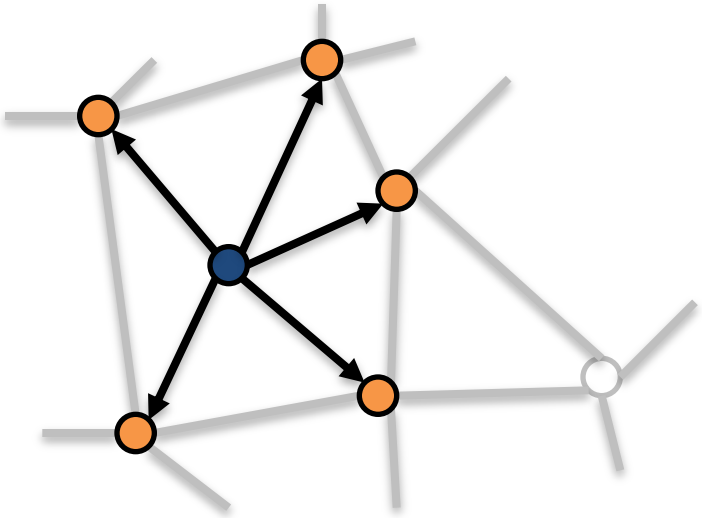
```
class topology(ost.Topology.Abstract):
  def neighborhood(self, p):
    # p - набор(массив) координат,
    # рассматриваемой точки
    return { synonym1: [p1_x1,...,p1_xn1],
             ....,
             synonymk: [pk_x1,...,pk_xnk]
           }
```

Пример: неструктурированная сетка

Все вершины графа пронумерованы 1,2,3,…



Для каждого узла явное описание списка соседей



Простое использование формата METIS для такого описания

Пример: неструктурированная сетка

```
# Пример топологии неструктурированной сетки
# на основе функции описания окрестности
class topology(ost.Topology.Abstract):
    def __init__(self):
        #Таблица, задающая ребра графа
        self.edges = { 0 : [1, 3],
                       1 : [0, 2, 3],
                       2 : [],
                       3 : [2] }

    # Функция описания окрестности вершины p
    # Возвращает список всех вершин
    # с которыми соединена данная p
    def neighborhood(self, p):
        return self.edges[p]
```

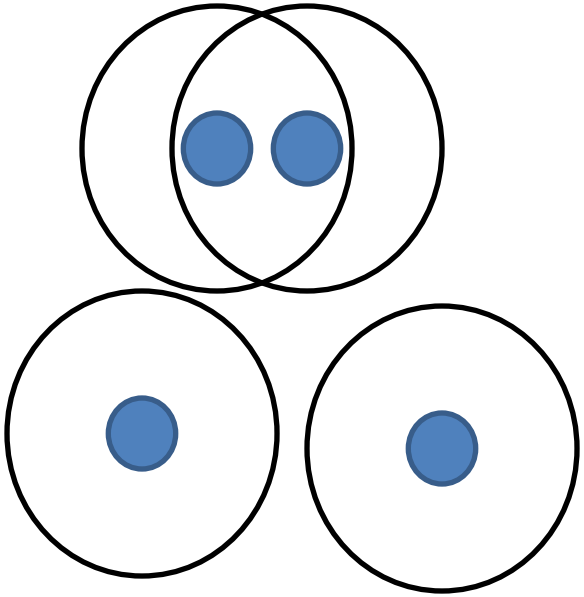
Пример: неструктурированная сетка

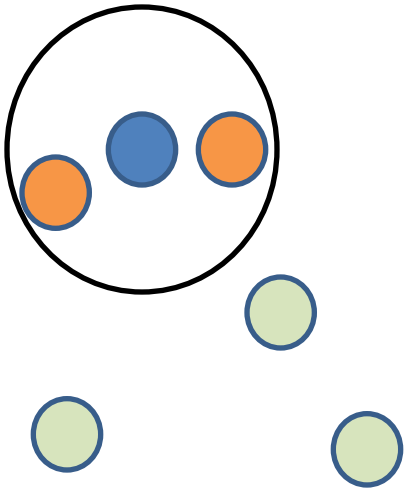
```
# Пример топологии неструктурированной сетки
# на основе функции описания окрестности
class topology(ost.Topology.Abstract):
    def __init__(self):
        #Таблица, задающая ребра графа
        self.edges = { 0 : [1, 3],
                       1 : [0, 2, 3],
                       2 : [],
                       3 : [2] }

    #Функция близости проверяет наличие вершины p2
    #в списке концов ребер, выходящих из p1
    def proximity(self, p1, p2):
        return p2 in self.edges[p1]
```

Пример: плоскость

Координаты задаются парой действительных чисел

Соседи удалены не более, чем на $\mathbf{r}$: $|\mathbf{x} - \mathbf{y}| \leq \mathbf{r}$





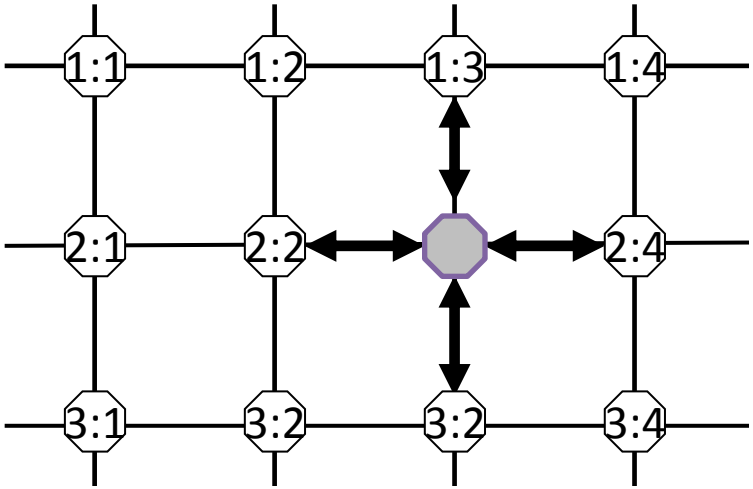
Пример: плоскость

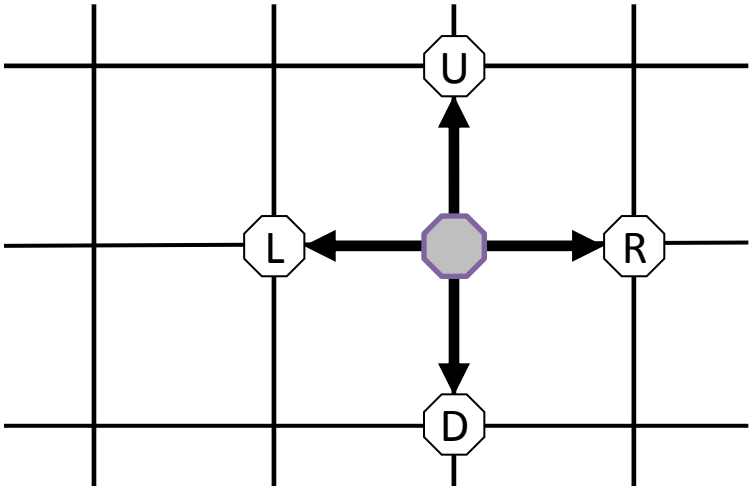
```
class topology(ost.Topology.Abstract):
    def __init__(self):
        # Задание радиуса окрестности точки
        self.radius = 2

    #Функция вычисления расстояния между точками
    def distance(self, p1, p2):
        return sqrt( (p1[0] - p2[0])**2
                    +(p1[1] - p2[1])**2)

    #Функция близости.
    #Расстояние между точками меньше, чем radius
    def proximity(self, p1, p2):
        return self.distance(p1, p2)< self.radius
```

Пример: целочисленная решетка

Координаты задаются парой целых чисел

Координаты соседей отличаются на ±1 по одной из координат





Пример: целочисленная решетка

```
class topology(ost.Topology.Abstract):

    # Функция описания окрестности вершины p
    def neighborhood(self, p):
        return { left : [p[0] - 1,p[1]],
                 right: [p[0] + 1,p[1]],
                 up   : [p[0],p[1] + 1],
                 down : [p[0],p[1] - 1]
               }
```

Глобальная и локальная топологии

Глобальная топология
общий класс топологии для всех объектов вычислительной системы

Локальная топология

класс топологии для специализации окрестности одного конкретного объекта

Прикладной объект

```
class object(ost.Object.Abstract):

    def fun_1(self, ...):
            …
    def fun_N(self, ...):


    # Функция проводящая вычисления
    def run(self):
```

Прикладной объект

```
def run(self):
 for <…>:
  # Обращение к i-ому соседу
  # по fun_j-ой функции
  self.topology.neighbors[i].link.fun_j()

  # При использовании синонимов
  # Обращение к соседу слева
  # по fun_j-ой функции
  self.left.fun_j()

self.setFinish()
```