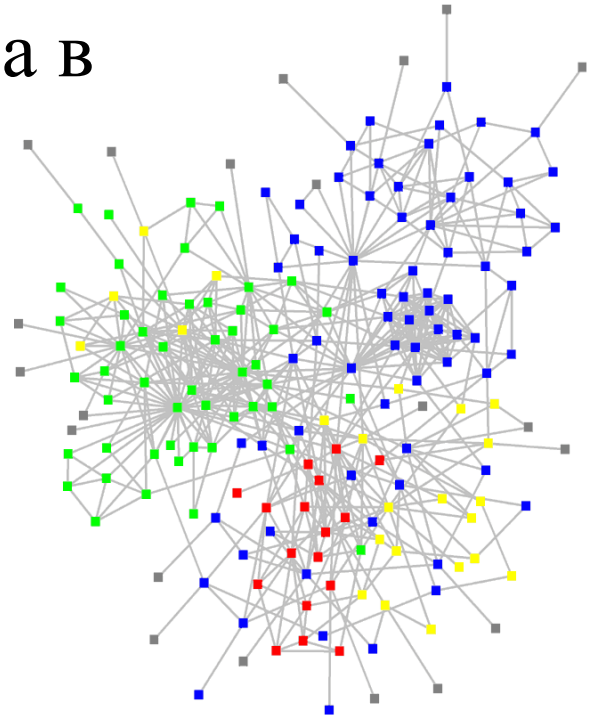
Факультет вычислительной математики и кибернетики
МГУ им. М.В. Ломоносова

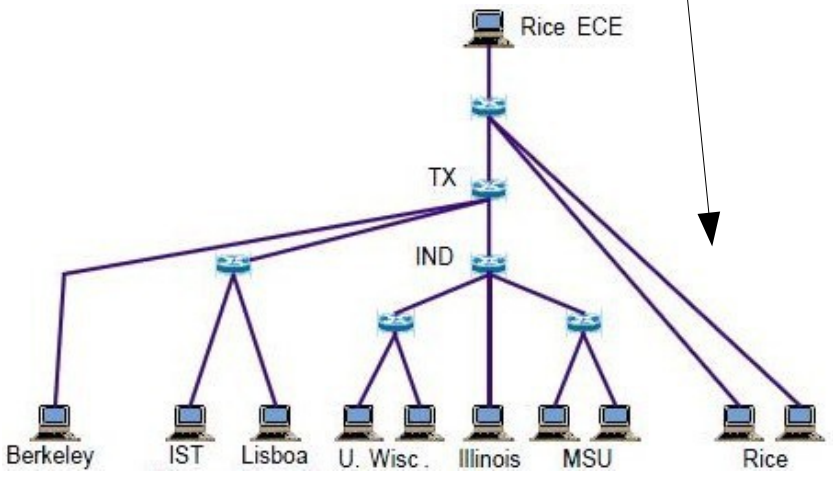
Выявление топологии коммуникационной среды вычислительного кластера по результатам нагрузочного тестирования

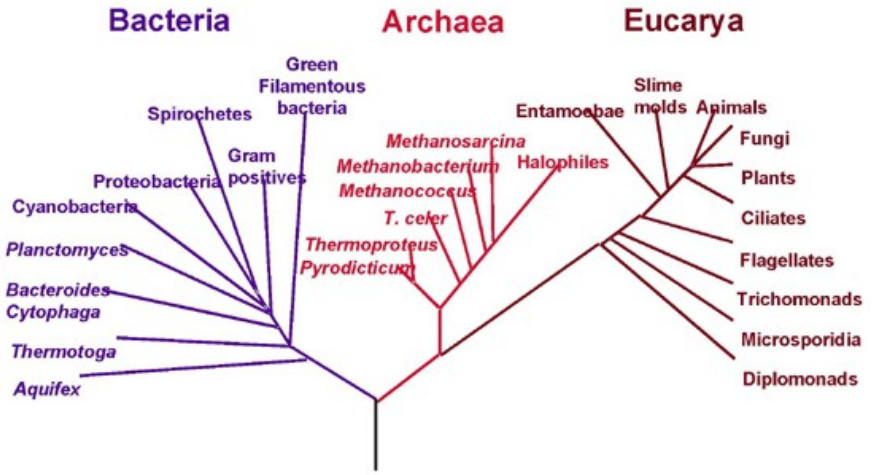
Банников П.С.
Сальников А.Н.

Смежные области

- Задача восстановления графа транзитивных взаимоотношений объектов актуальна в следующих областях:
  - Социальные сети
  - Биоинформатика
  - Транспортные сети
  - Компьютерные сети, маршрутизация







Тесты коммуникаций

- Предположим, что на кластере было проведено тестирование
  - результат — величины задержек в пространстве *N* x *N* x *M*, где *N* — множество узлов, *M* — множество длин передаваемых сообщений
    - *матрицы задержек*
  - *network_tests2*
    - на MPI

Спецификации

- Информацию об архитектуре даёт *спецификация* (*документация*) вычислителя
- Недостатки:
  - недостаточность детализации
  - несоответствие действительности
  - инертность информации
- Спецификация может попросту отсутствовать
  - например, кластер собран из узлов, соединённых коммутатором
  - внутреннее устройство коммутатора — коммерческая тайна

Реальным задачам — реальную информацию!

Зачем нужны тесты и выявление топологий

- Написание эффективных параллельных программ с учётом архитектуры конкретного вычислительного кластера (оптимальный *мэппинг*)
- Обнаружение неполадок в работе коммуникационной среды кластера
- Поиск несоответствий в спецификациях
- Возможность написания спецификации в случае отсутствия последней

Терминология

- *Топология кластера* — коммуникационные элементы кластера (вычислительные узлы, коммутаторы и т. д...) и соединяющие их физические каналы
- Сопоставим коммуникационному элементу вершину, каналу — пару дуг (ребро)
  - полученный набор вершин и дуг (рёбер) — *(не)ориентированный граф топологии*
- Отождествим понятия «*топология [кластера]*», «*граф*» и «*граф топологии*»

Задачи

- Выявление
  - по совокупности матриц задержек выявить топологию кластера
- Визуализация
  - отобразить получившуюся топологию в трёхмерном пространстве
- Сравнение
  - при наличии спецификации к кластеру сравнить выявленную топологию с описанной в спецификации

Выявление

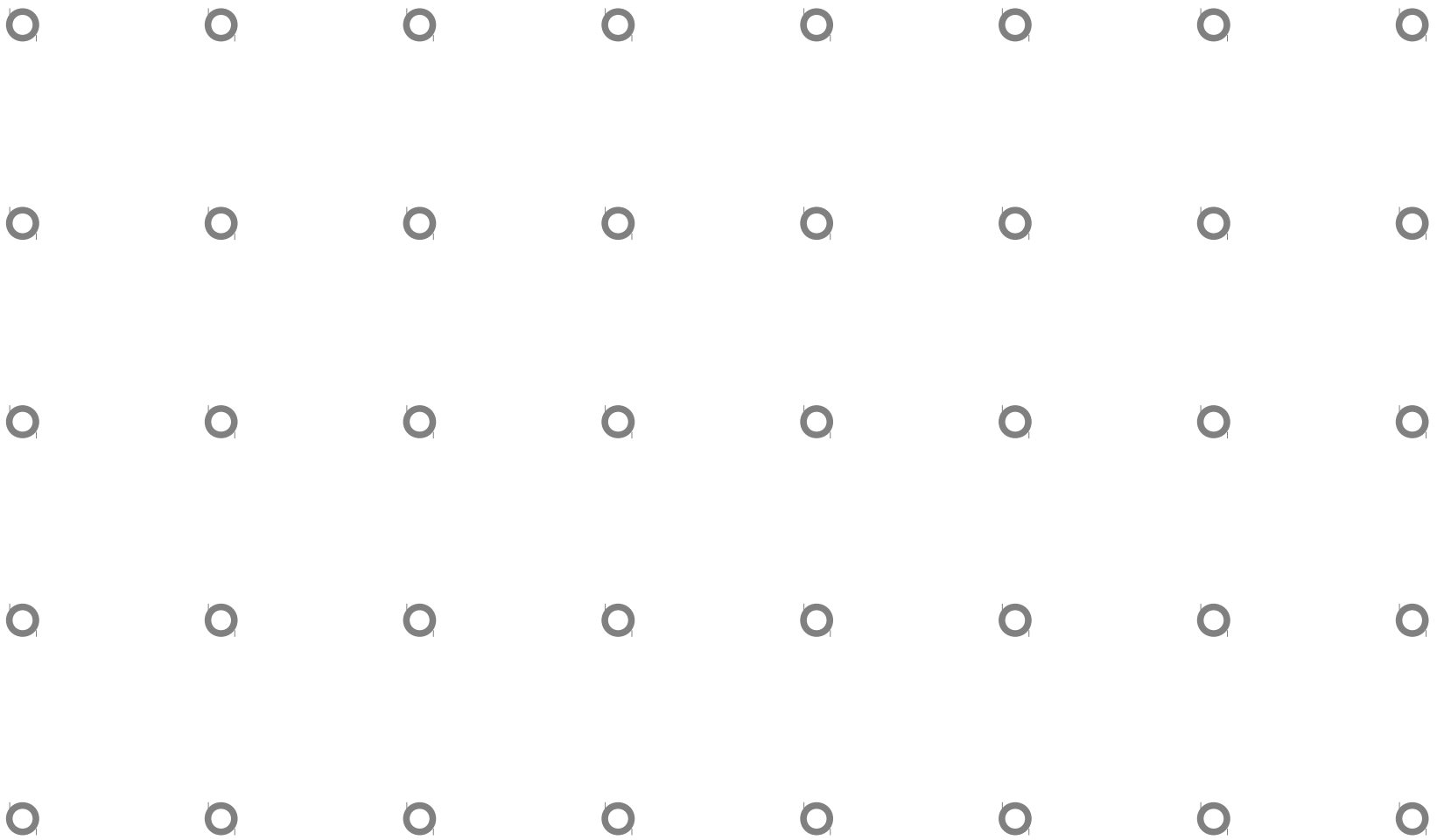
- *Требования*:
  - Восстановление топологий произвольного вида
  - Учёт процессов на общей памяти

- Из совокупности матриц задержек возьмём единственную — M
  - для неё выявим т.н. *локальную топологию*

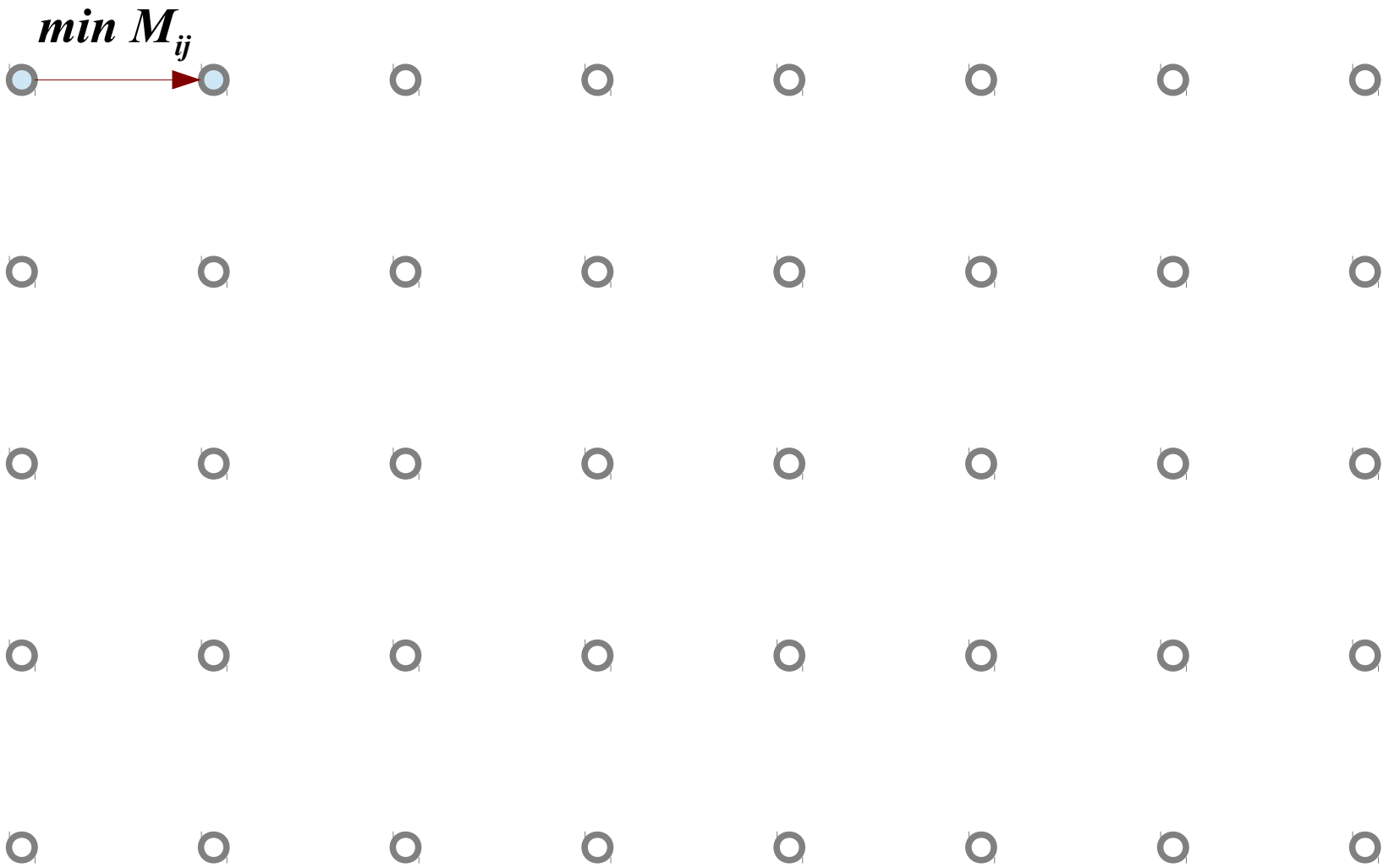
Выявление

- Этапы:
    1. выявление остовного дерева
    2. восстановление циклов

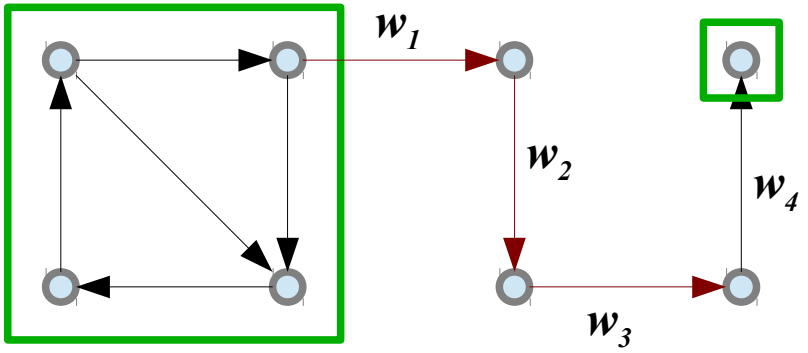
Выявление, этап 1



Выявление, этап 1

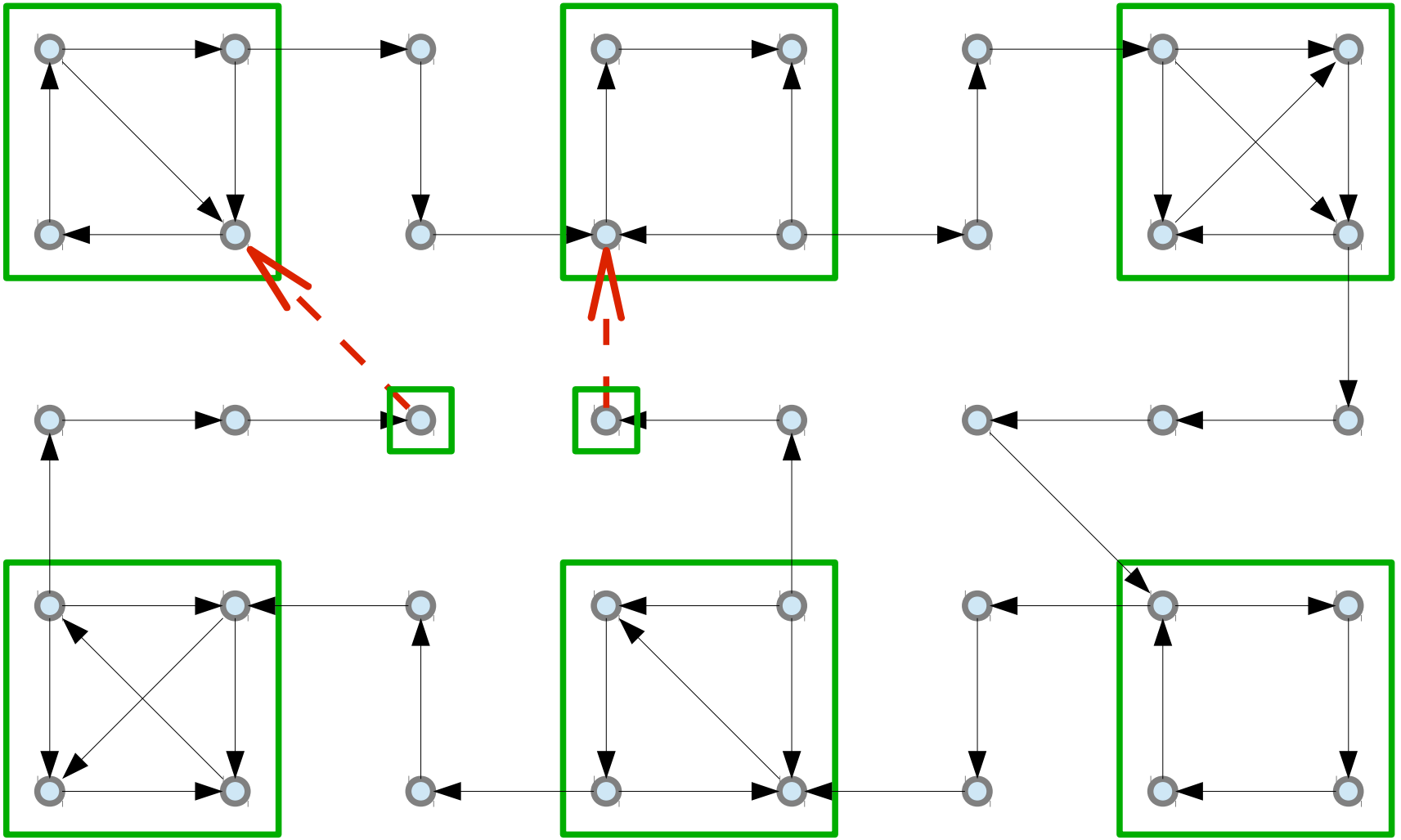


Выявление, этап 1



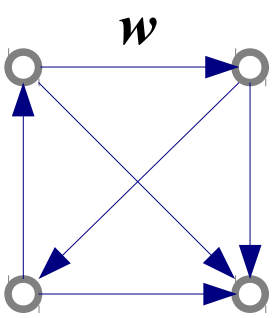
$$w_1 > w_2 > w_3 > w_4 = min$$

Выявление, этап 1

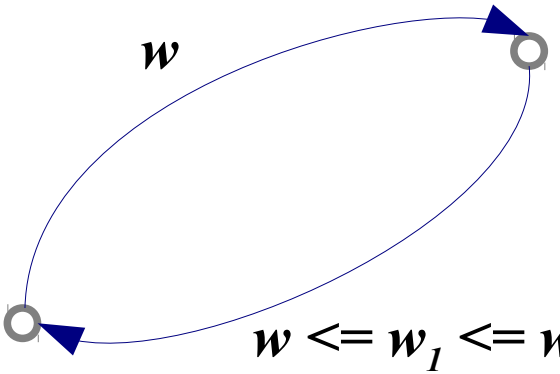


Выявление, этап 1

- Параметры:



$w <= w_1 <= w * (1+\varepsilon_s)$



$w <= w_1 <= w * (1+\varepsilon_d)$

Выявление, этап 2

- Этап 1 не допускает циклов
  - поэтому плохо применим к архитектурам, в которых есть циклы
- Если суммарный вес дуг кратчайшего пути от вершины *i* до вершины *j* больше, чем M_{ij}, и дуги *i->j* нет, то цикл замыкается
- На практике состоит из *n* вызовов алгоритма Дейкстры поиска кратчайшего пути из одной вершины во все остальные

Выявление

- *Особенности*:
  - Независимость от симметричности входной матрицы
  - Выявление топологий произвольной структуры
  - Результирующий граф имеет строго n вершин
  - Результирующий граф ориентированный
  - Временная сложность в худшем случае: $\boldsymbol{O}(n^3)$
  - Пространственная сложность: $\boldsymbol{O}(n^2)$

Выявление

- Набор *локальных топологий* объединяется в единственную топологию
  - отождествление вершин
  - отождествление однонаправленных дуг
    - *Вероятность существования* дуги — k/m,

      где k — число топологий в наборе, содержащих дугу, m — общее количество матриц задержек

Визуализация

- *Требования*:
  - сохранение длин дуг
  - отсутствие вершин с одинаковыми координатами
  - визуальное отличие дуг с разной вероятностью существования

Визуализация

- Отображение **не**ориентированного графа:
  - вершины — точки, рёбра — отрезки, соединяющие соответствующие вершины-точки
- Задача отображения графа сводится к классической задаче *многомерного шкалирования* (ММШ)[1]
  - *функция стресса*

(1) Borg I., Groenen P.J.F. «Modern Multidimensional Scaling — Theory and Applications»

Визуализация

- Алгоритм:

1. перевод матрицы задержек в матрицу длин рёбер с последующей нормировкой

$$w = \frac{M_{ij} * p_{i \to j} + M_{ji} * p_{j \to i}}{p_{i \to j} + p_{j \to i}}$$ — пересчёт весов

2. применение адаптированного метода ММШ

$$f = (f_1 + f_2)/2$$ — новая функция стресса

3. визуализация полученных точек и отрезков

$$k = \frac{p_{i \to j}^2 + p_{j \to i}^2}{p_{i \to j} + p_{j \to i}}$$ — коэффициент интерполяции

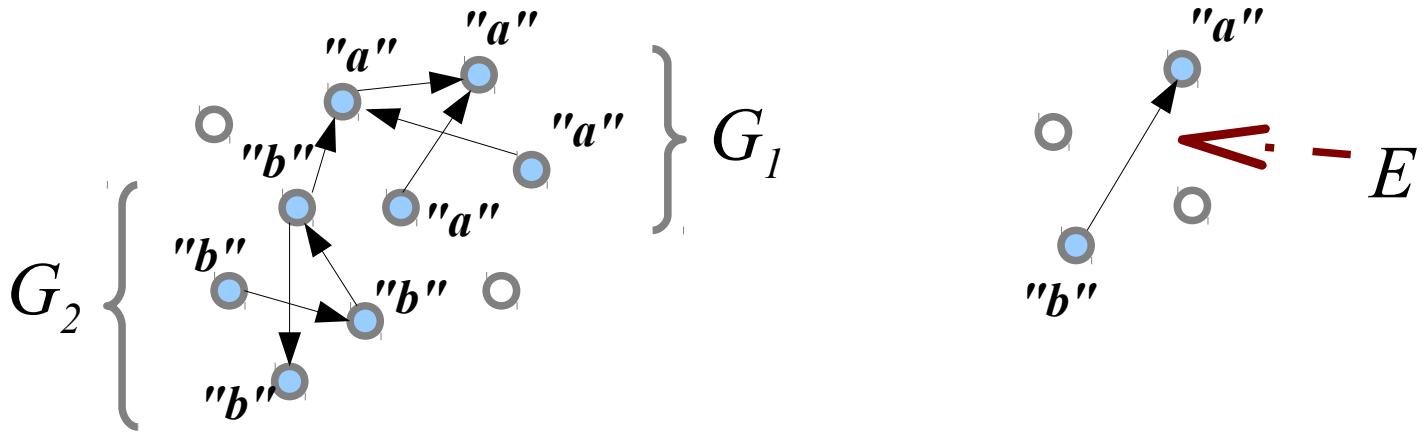
Сравнение

- Формат записи топологии из спецификации:
  - подмножество языка Graphviz/DOT
  - вершинам присвоено уникальное имя
  - веса рёбер — пропускные способности каналов
- Предполагается, что граф неориентированный
- Отождествление вершин в эталонной топологии и выявленной топологии происходит по именам

Сравнение, 1 случай

- Вычисляется *мера сходства* по формуле

$$sim = \frac{\sum_{G_1, G_2 \in \hat{V}} \left(\max_{i \in G_1,\ j \in G_2} p_{i \to j} \right)}{|E|} * 100\%$$



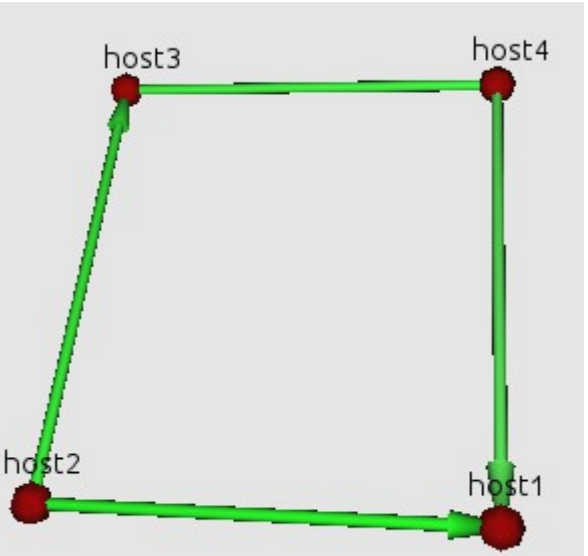
- Если мера сходства ненулевая, эталонная топология считается *регулярной*; результат алгоритма — вычисленное значение меры сходства

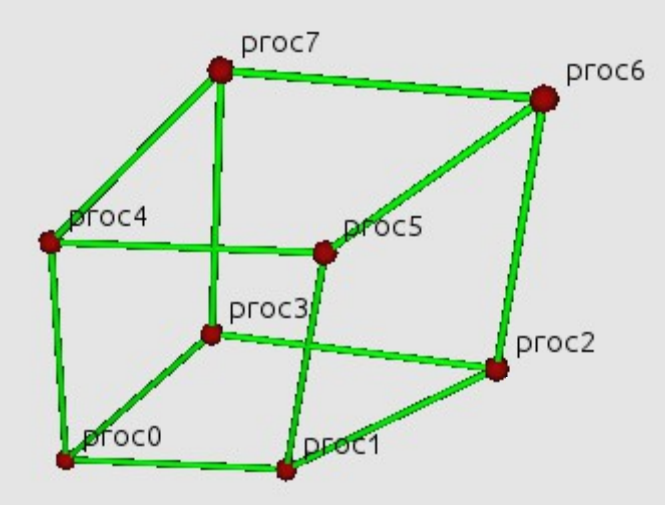
Сравнение, 2 случай

- Если мера сходства равна нулю, значит, эталонная топология *иерархическая*
- Отображается граф эталонной топологии со специальной раскраской рёбер
  - надо привести веса рёбер сравниваемых топологий к одному формату
  - цвета зависят от разницы весов одного и того же ребра в сравниваемых топологиях

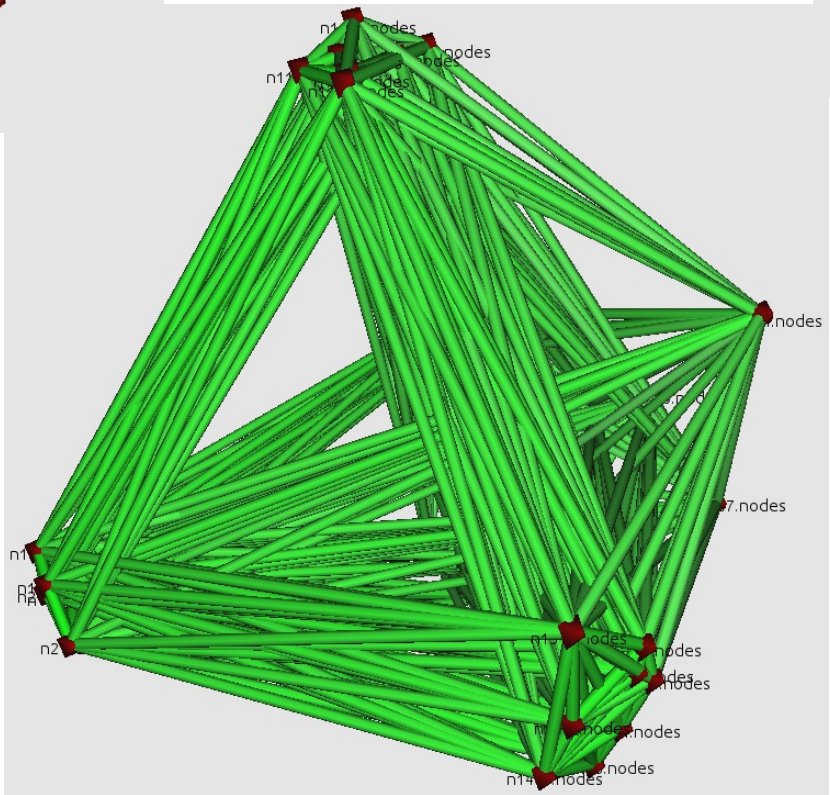
Примеры выявленных топологий

4-ядерный процессор; среднее арифметическое



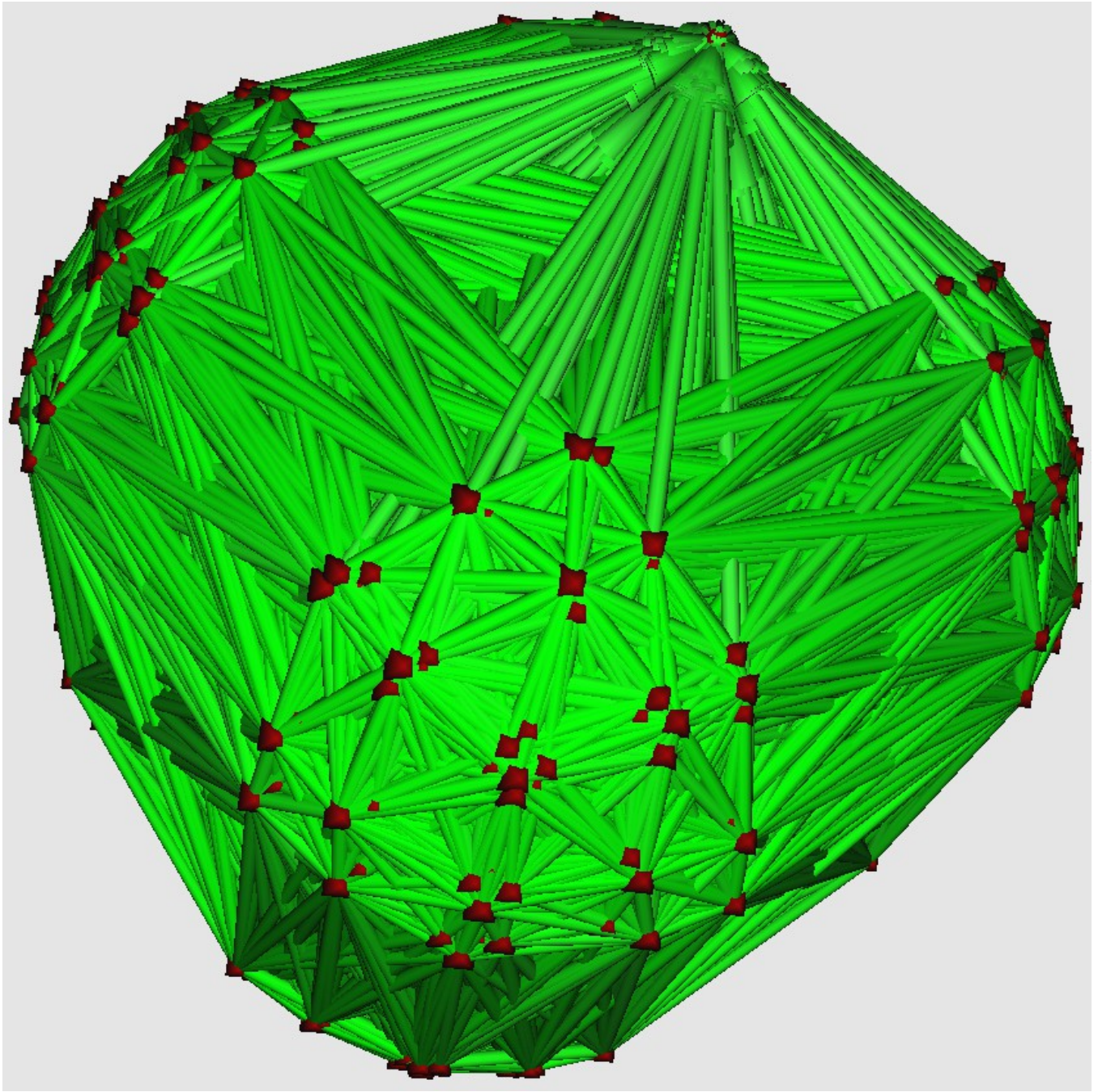


Искусственный пример: куб из 8 процессов; все длины равны 1



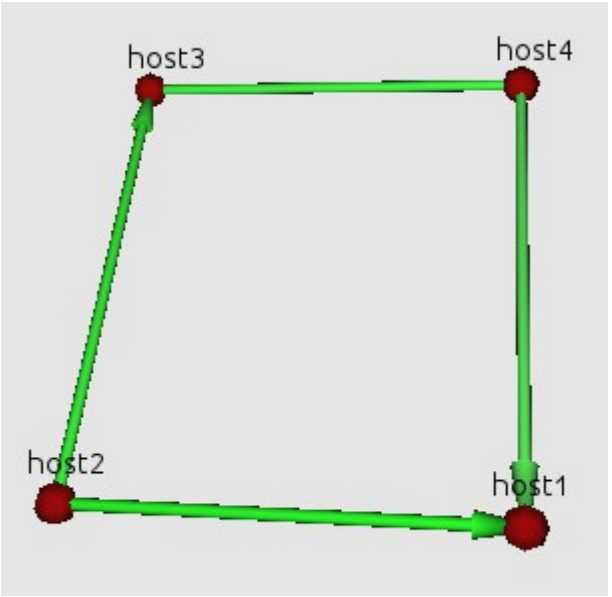
38 узлов, соединённых коммутатором; медиана

Примеры выявленных топологий

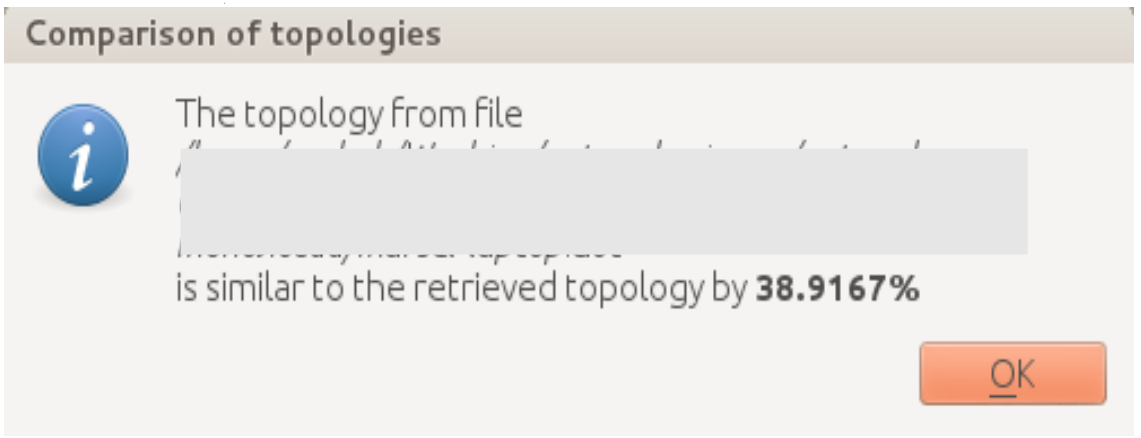


BlueGene/P,
512 процессов;
длина
сообщений 0

Пример сравнения - 1



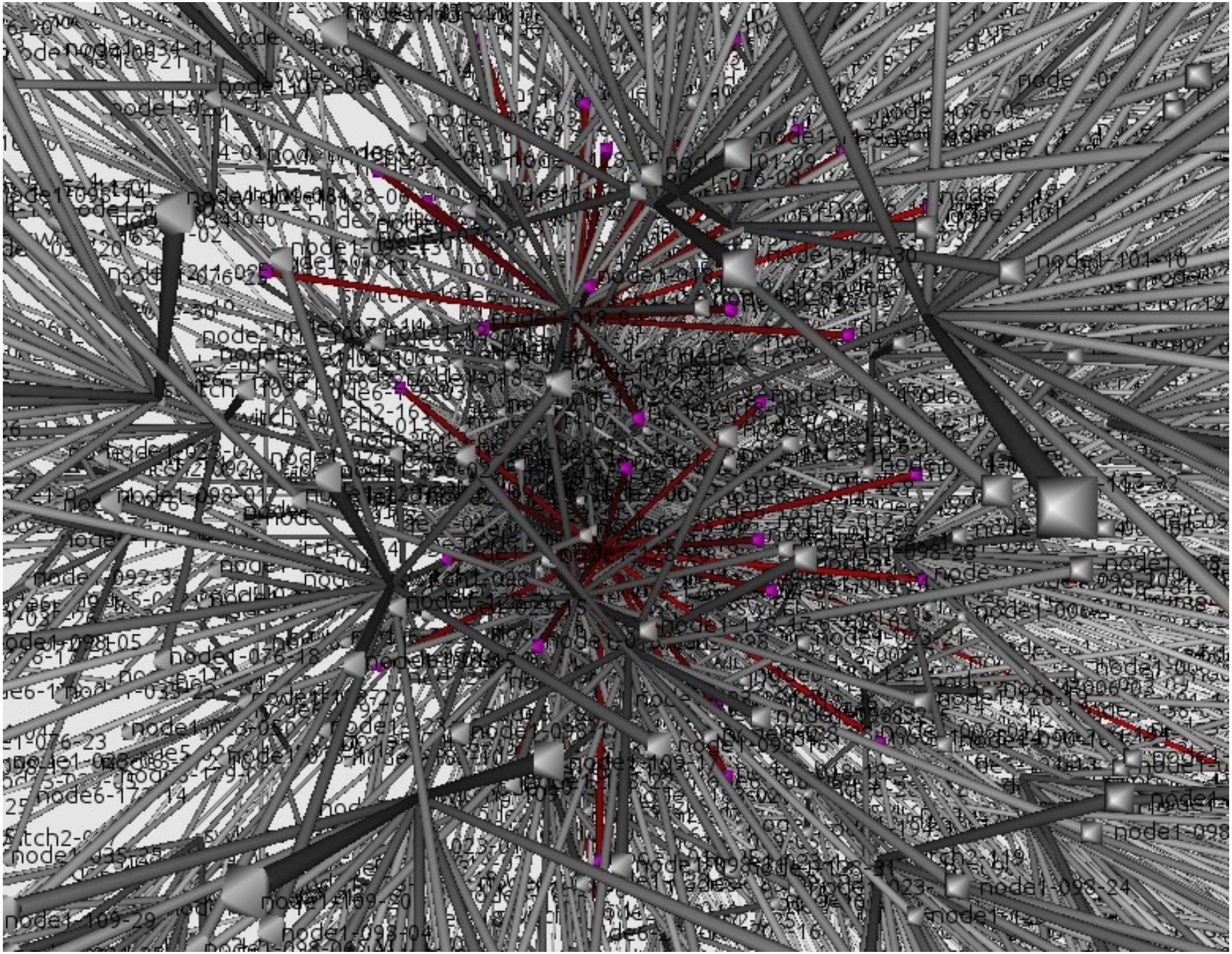

```
graph -laptop {
    v0 [label="host1"];
    v1 [label="host2"];
    v2 [label="host3"];
    v3 [label="host4"];

    v0 -- v1 [label="1000 Gbit/s"];
    v0 -- v2 [label="1000 Gbit/s"];
    v0 -- v3 [label="1000 Gbit/s"];
    v1 -- v2 [label="1000 Gbit/s"];
    v1 -- v3 [label="1000 Gbit/s"];
    v2 -- v3 [label="1000 Gbit/s"];
}
```



Пример сравнения - 2

«Ломоносов», 6761 узел

Пример сравнения - 2



Вычислительный эксперимент

- (выявление); AMD Phenom II, 1 ядро, 1.8 ГГц

Кластер	*Число узлов*	*Среднее время для одной матрицы*
BlueGene/P	128	10 мс
	256	80 мс
	1024	760 мс
«Ломоносов»	500	80 мс
«Чебышёв»	64	5 мс
	128	9 мс
MVS-50k	300	40 мс
switch	38	0.04 мс

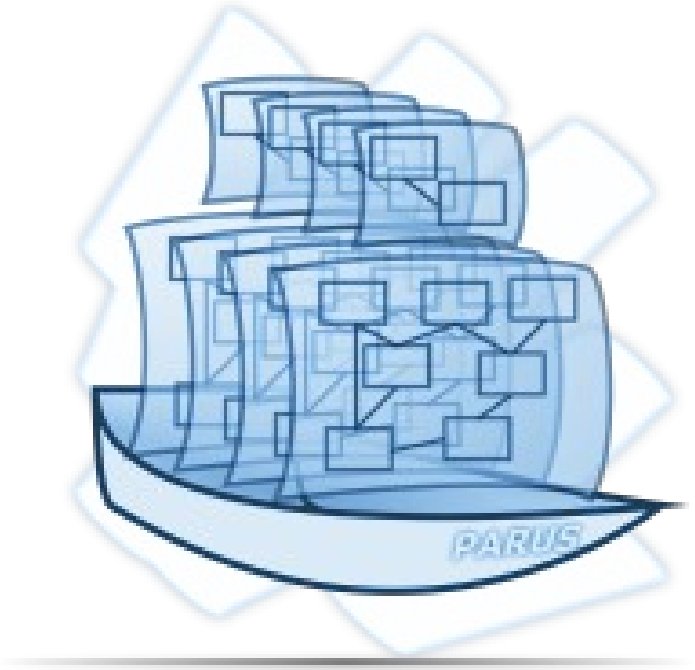
Вычислительный эксперимент - 2

- (отображение)
  - время работы сильно зависит от графа
  - от нескольких миллисекунд (искусственные примеры) до 1 часа («Ломоносов», 6761 узел)

- (визуализация)
  - 500 вершин и порядка 200.000 рёбер отрисовываются в режиме реального времени

Направления развития

- Разработка алгоритма выявления топологии, учитывающего коммутаторы и маршрутизаторы
  - расширение Neighbor Joining?
- Отыскание глобального минимума в задаче многомерного шкалирования для более точного отображения графа



Спасибо!

Исходники:

`git.code.sf.net/p/parus/code/`

папка

`network_test2/src/network_viewer_qt_v2/`

Сальников Алексей Николаевич: salnikov@cs.msu.ru
Банников Павел Сергеевич: pvl.bannikov@gmail.com

Интерпретация результатов

- Алгоритм выявления топологии:

Спецификация архитектуры	Результат алгоритма
регулярная структура (решётки, гиперкубы, торы, ...)	структура (в общем случае - нерегулярная)
иерархическая структура (деревья, звёзды, ...)	полносвязный граф

Оценки сложностей, особенности

- n - число процессов, m - число длин сообщений
- временная сложность в худшем случае: $O(n^3*m)$
- пространственная сложность: $O(n^2)$
- Особенности:
  - не требуется симметричность матриц
  - выявление связей непосредственно между вычислительными элементами
    - структура произвольна
  - никаких дополнительных вершин не создаётся, поэтому невозможен учёт коммутаторов и маршрутизаторов

Программная реализация

- Язык
  - C/C++
- GUI
  - Qt4
- Визуализация
  - OpenGL, через Qt-модуль QtOpenGL
- Распараллеливание
  - OpenMP

Программная реализация - 2