Обеспечение качества сервиса в центрах обработки данных

м.н.с. Степанов Евгений Павлович

Программа курса

Подходы:

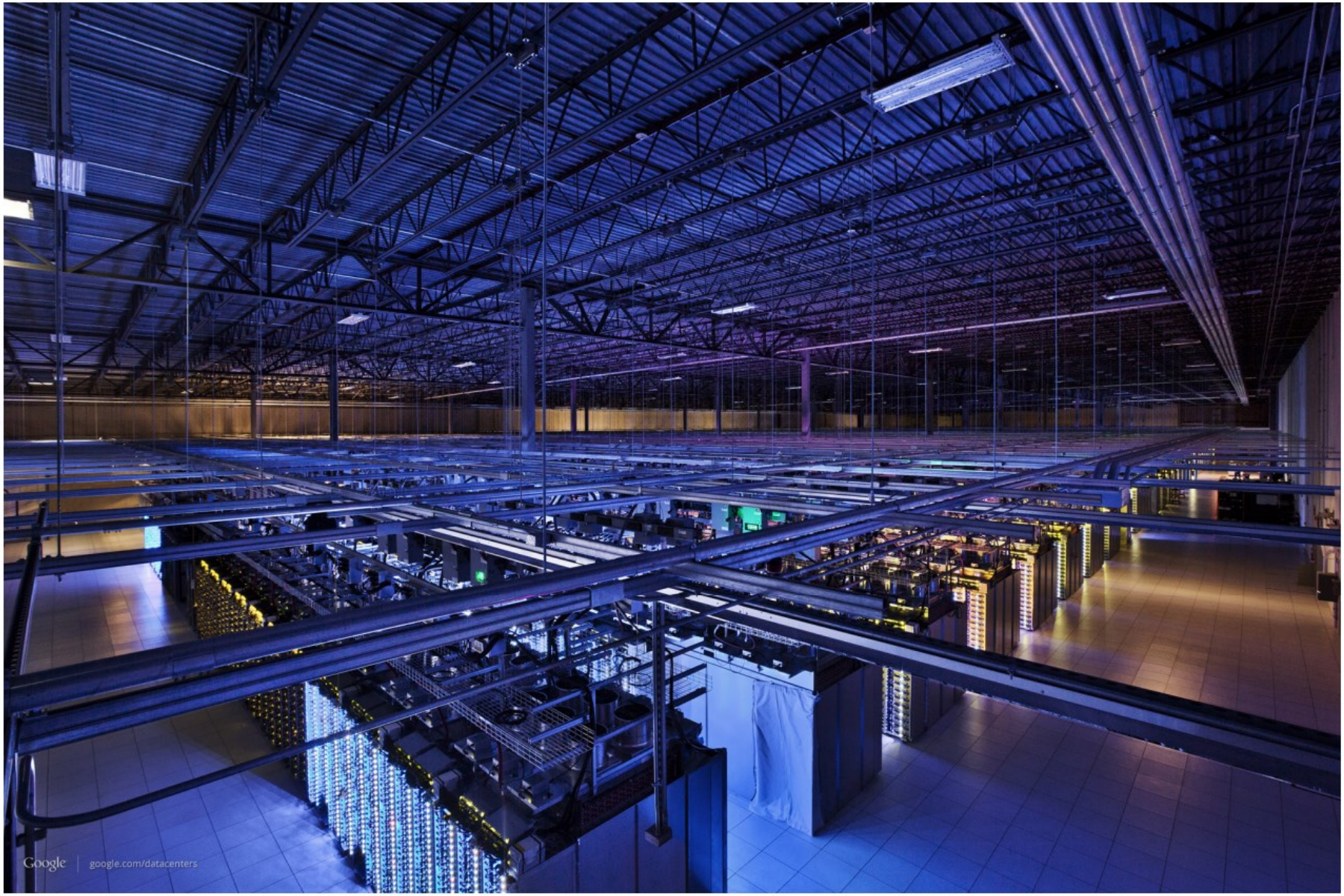
1. ***Управление перегрузкой***
   - Современные протоколы управления перегрузкой TCP
2. ***Демультиплексирование/мультиплексирование***
   - Многопоточные транспортные протоколы
   - Маршрутизация на уровне интернет провайдеров
   - Network Coding
3. ***Сегментация***
   - TCP Proxy
4. ***Балансировка***
   - Балансировка нагрузки и управление трафиком

Модели:

- Сетевое исчисление: математический подход к качеству сервиса
- NS3: моделирование поведения сети с высокой точностью

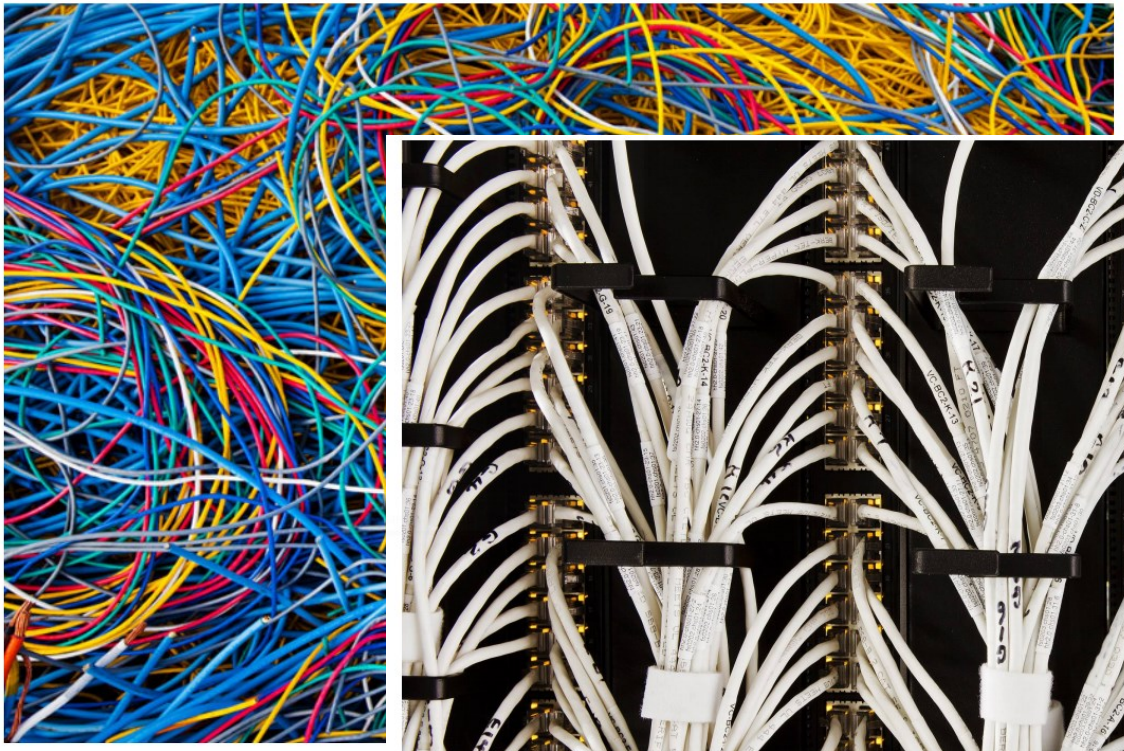
Примеры:

- Управление сетевыми ресурсами в Центрах Обработки Данных
- Пропускная способность по требованию
- Обеспечение качества сервиса в сетях доставки контента
- HTTP3/QUIC


Google | google.com/datacenters


© AP

Кабели



Эффективность современных ЦОД

страшная тайна – утилизация в 30% это хорошо

- Неравномерное потребности приложений
  - Большинство приложений требовательны к одному из ресурсов машины – остальные простаивают
- Сложность планирования вычислений
  - Требования теннантов и их приложений изменяются во времени, их сложно предсказывать
- Поддержание высокого уровня сервиса
  - Нужно быть готовыми к необходимости подключить дополнительное оборудование при возникновении отказов или обновлении

Повышение гибкости

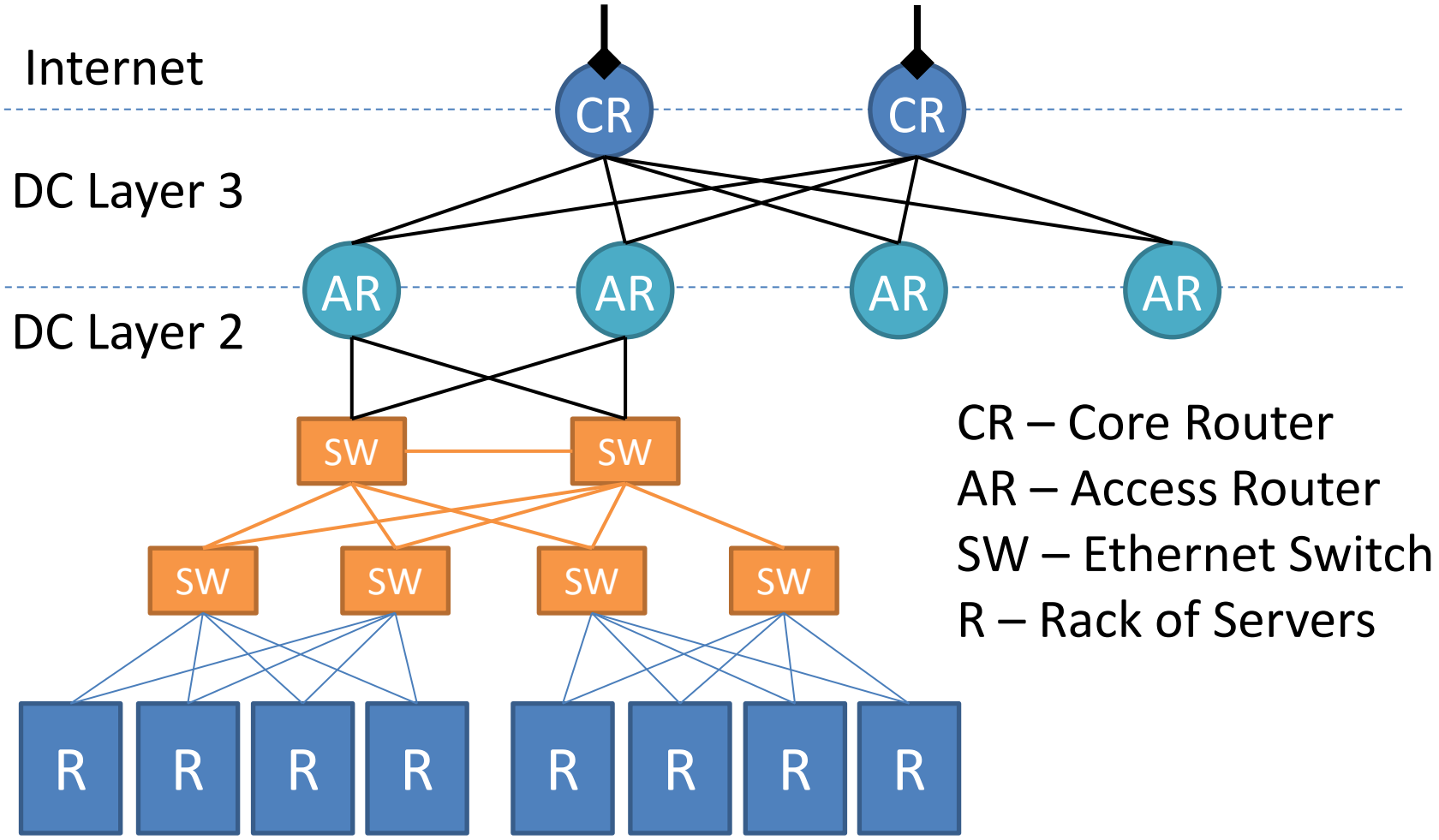
любой сервис на любом сервере

Принцип ***resource pooling*** – превращение множества серверов ЦОД в единый пул вычислительных ресурсов, которые можно распределять по задачам произвольным образом

- Гибкое управление вычислителями Распределённые файловые системы
- Быстрый обмен данными между приложениями, вне зависимости от их физического расположения

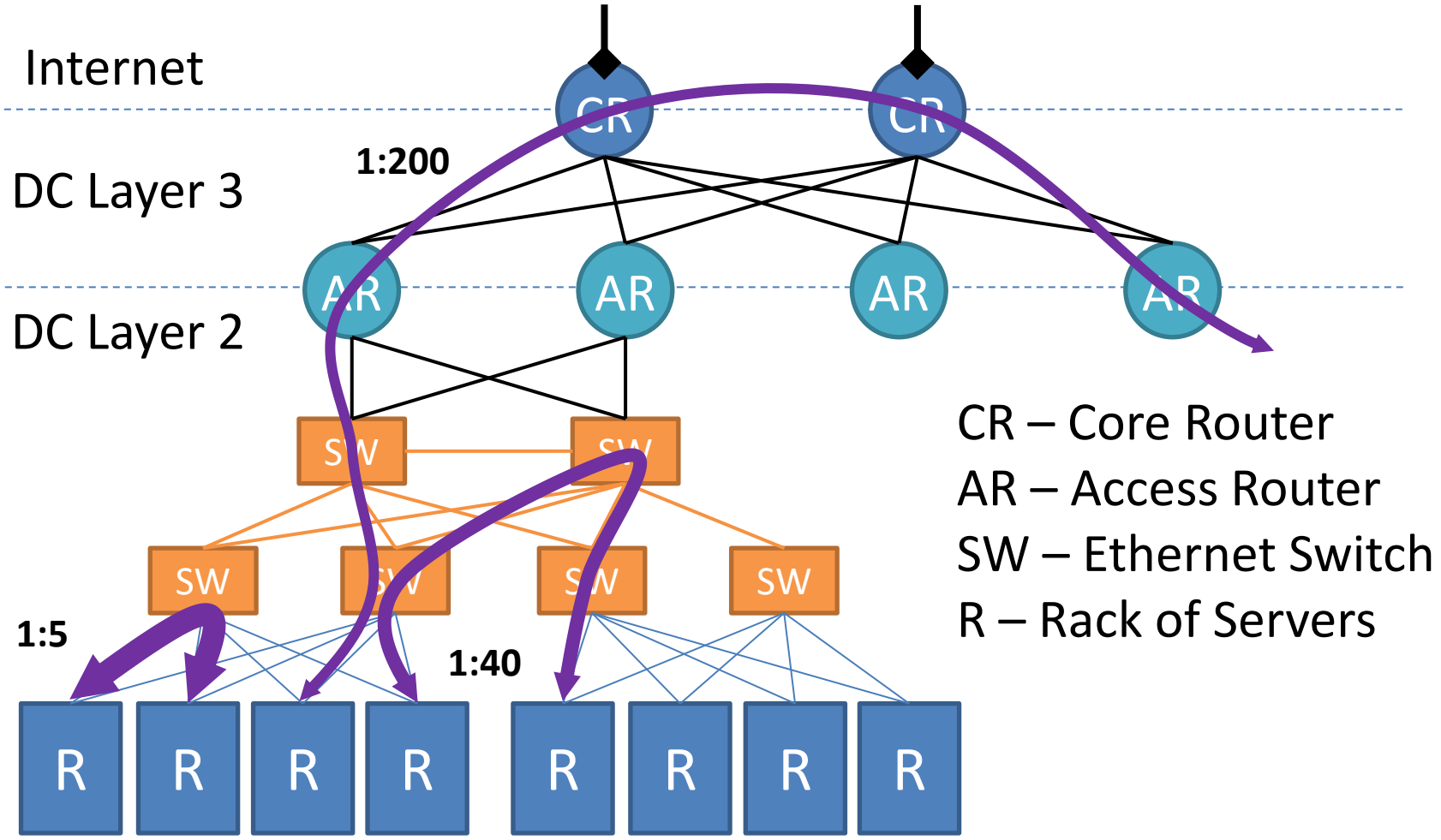
Классическая топология ЦОД

Data Center: Load balancing Data Center Services, Cisco 2004



Классическая топология ЦОД

Data Center: Load balancing Data Center Services, Cisco 2004

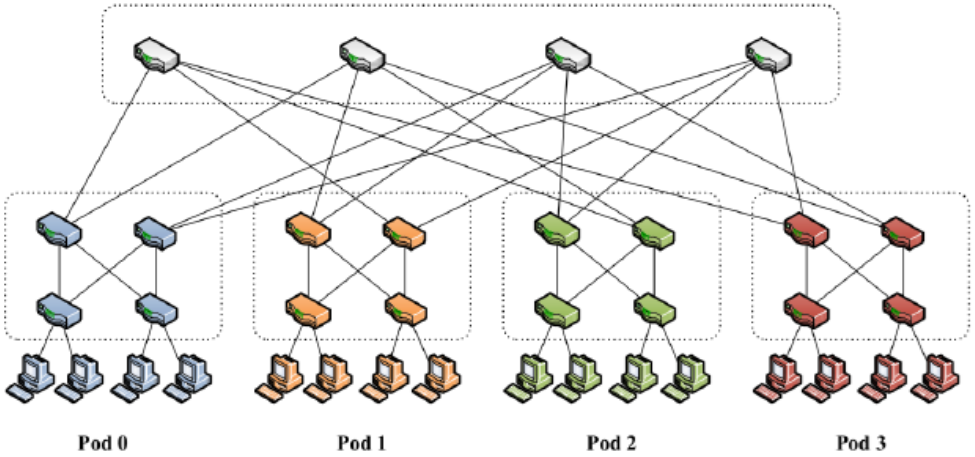


Проблемы традиционной топологии

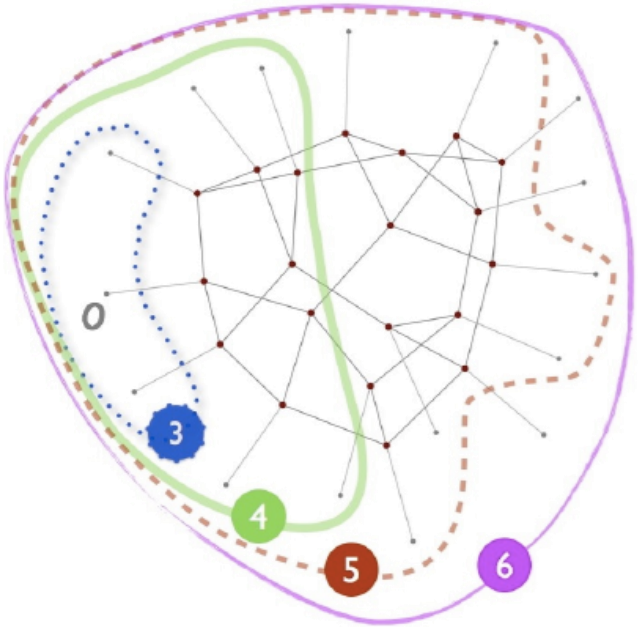
- Слишком большая нагрузка на центральные коммутационные устройства
  - Плохая масштабируемость
  - Высокая стоимость оборудования
  - Низкая отказоустойчивость
- Неравномерная связность топологии
  - Фрагментированность ресурсов
  - Низкая *Bisection-Bandwidth* – совокупная пропускная способность сети ЦОД при наихудшем распределении нагрузки по узлам

Альтернативные топологии

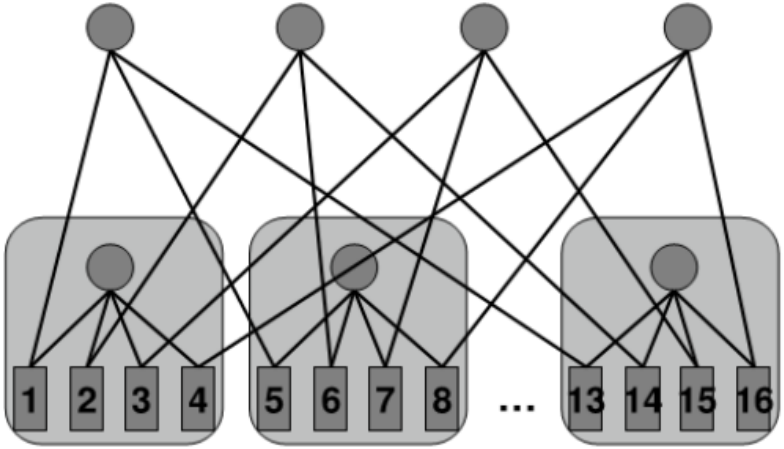
Fat-tree [SIGCOMM'08]



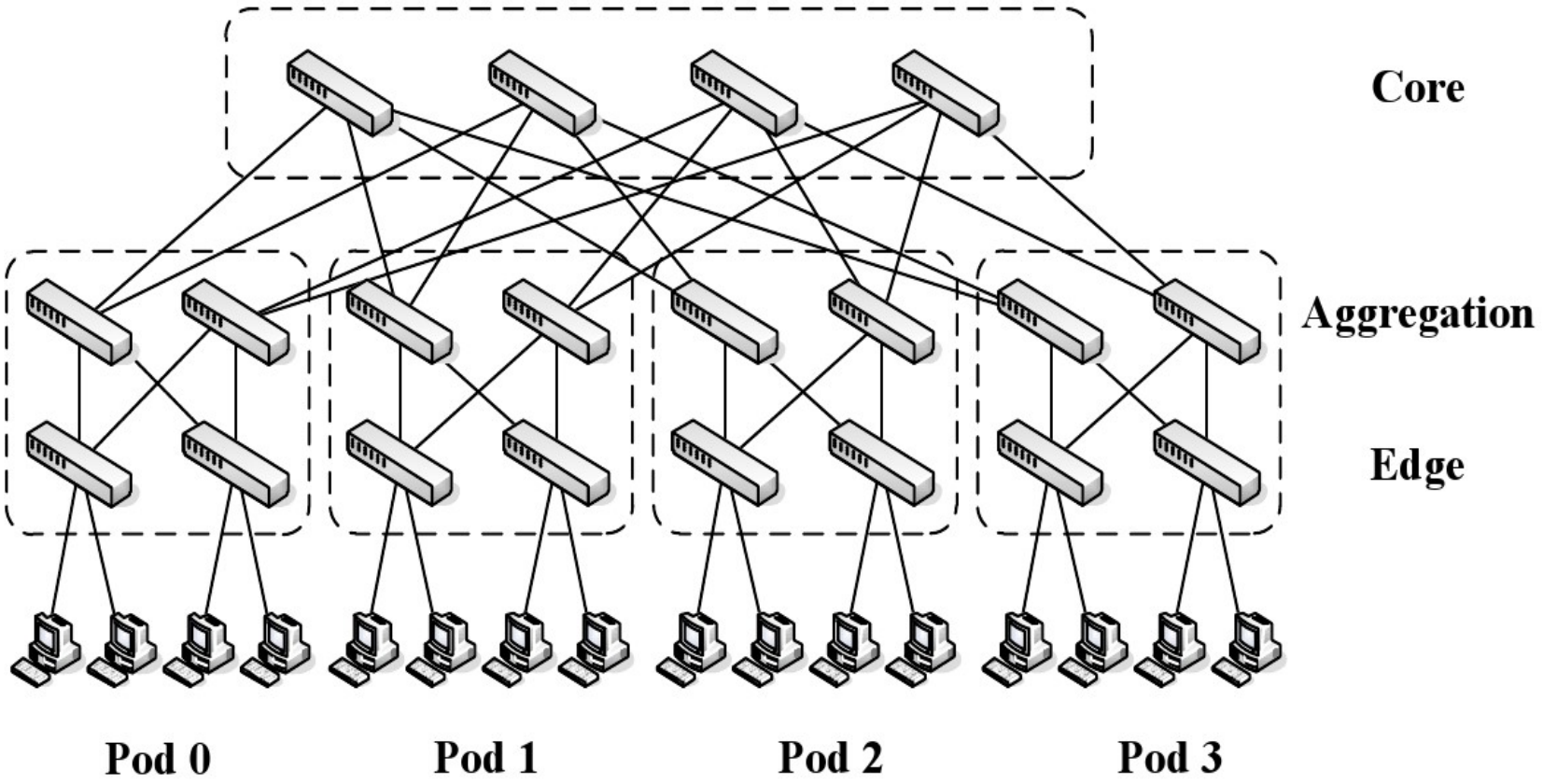
Jellyfish (random) [NSDI'12]



BCube [SIGCOMM'10]



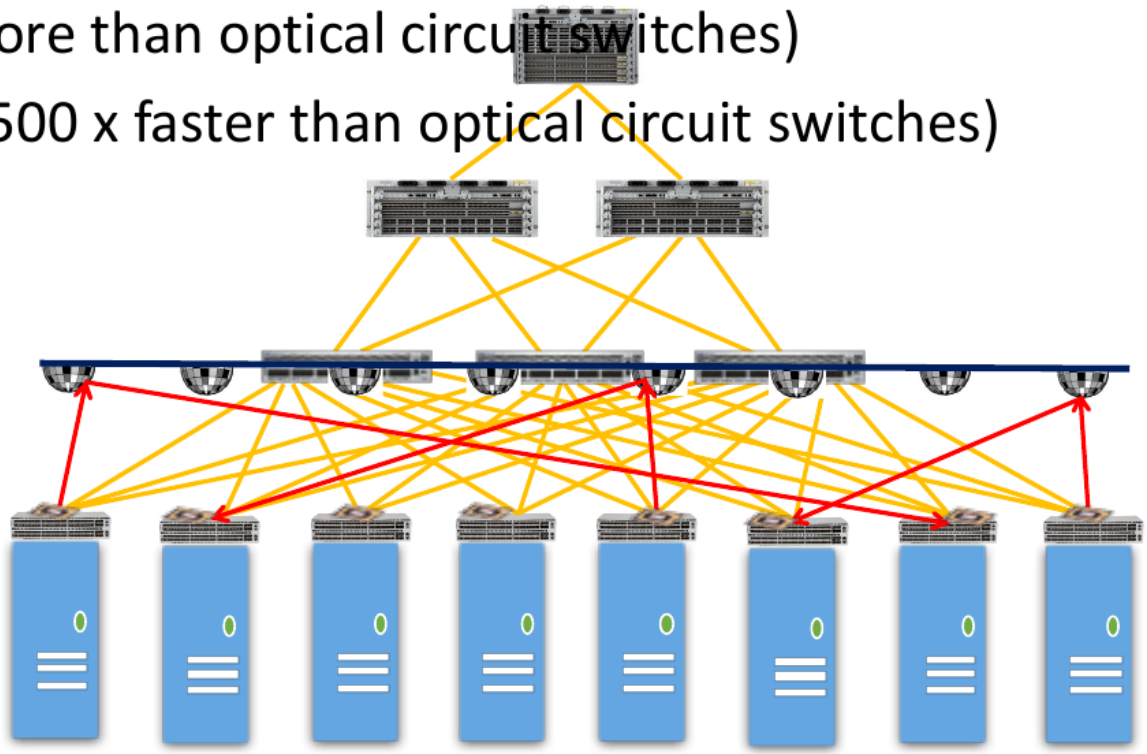
FAT Tree



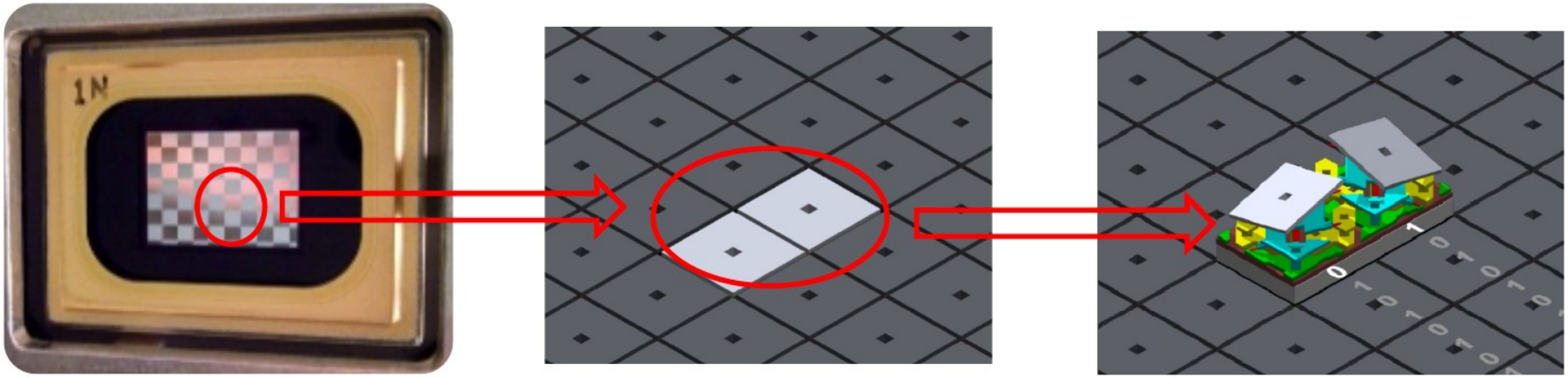
Можно ли обеспеченивать высокую высокую скорость обмена по запросу?

ProjecToR interconnect

- Free-space topology (seamless)
- 18,000 fan-out (60 x more than optical circuit switches)
- 12 us switching time (2500 x faster than optical circuit switches)



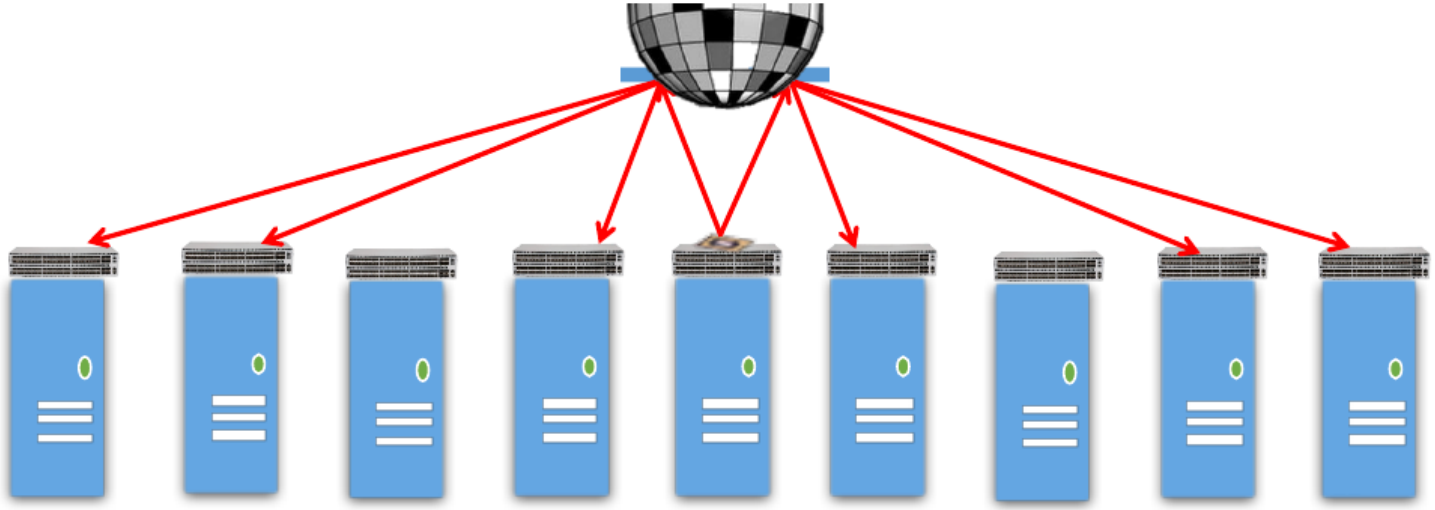
Static topology



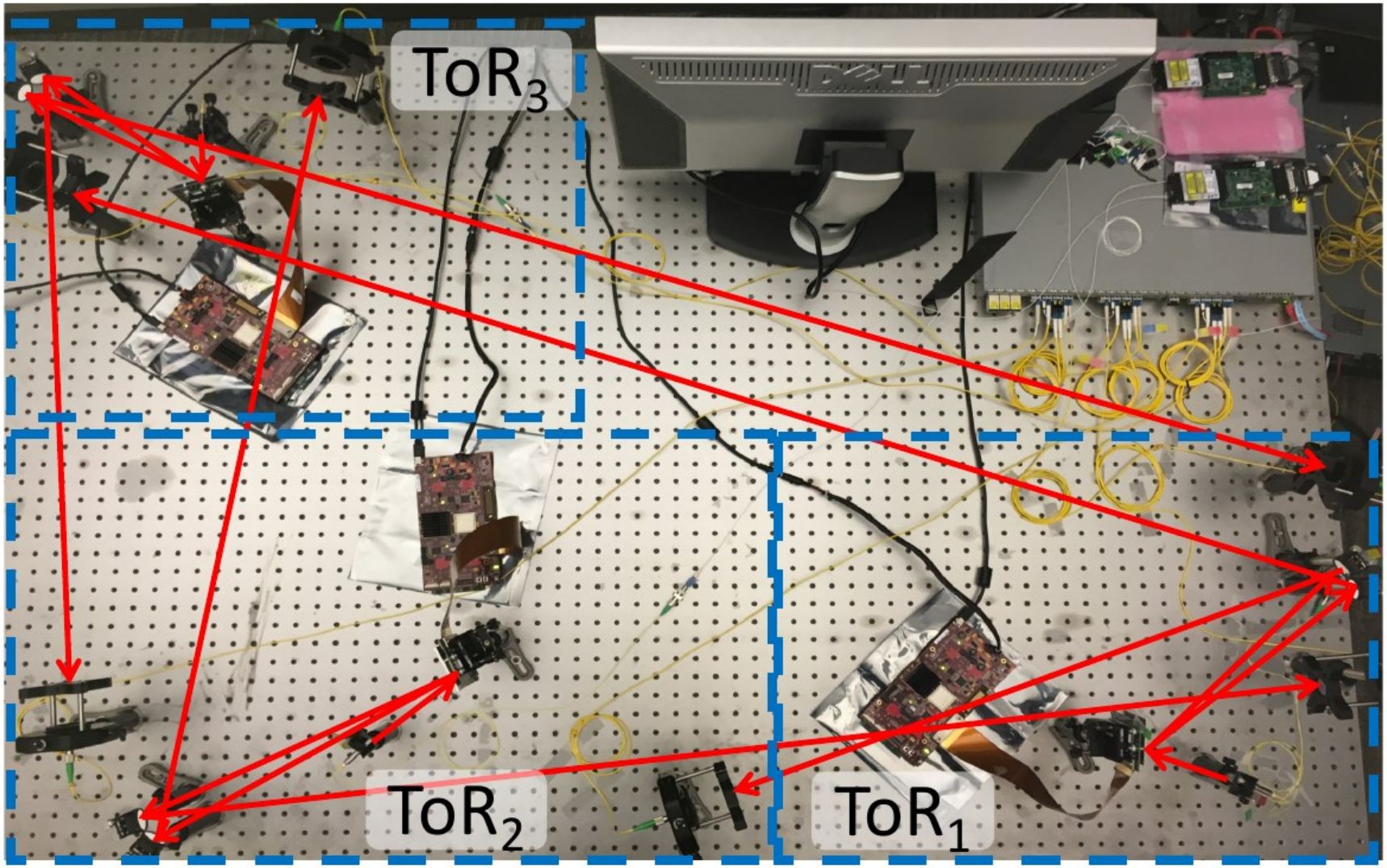
TEXAS INSTRUMENTS

Array of micromirrors (10 um)

Memory cell

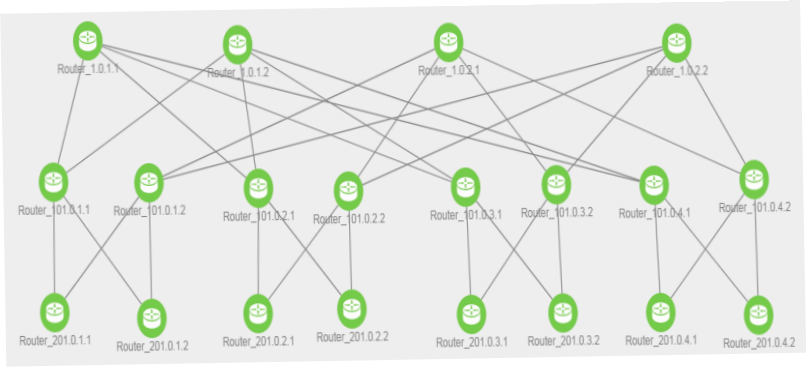


Prototype: A 3-ToR ProjecToR interconnect

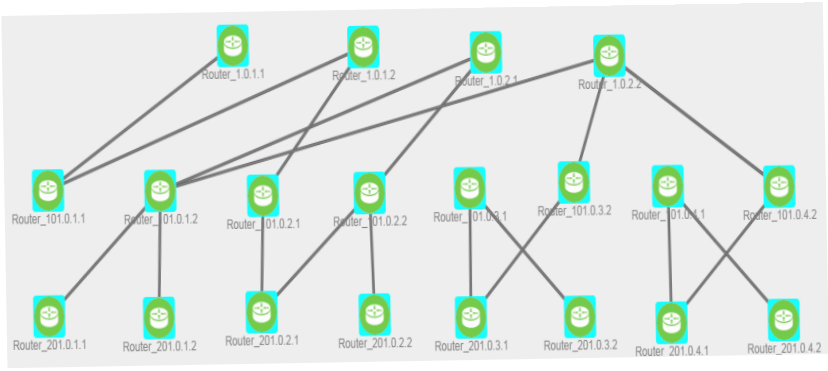


ProjecToR: Agile Reconfigurable Data Center Interconnect. Sigcomm 2016.

Flooding Topology



Fat Tree



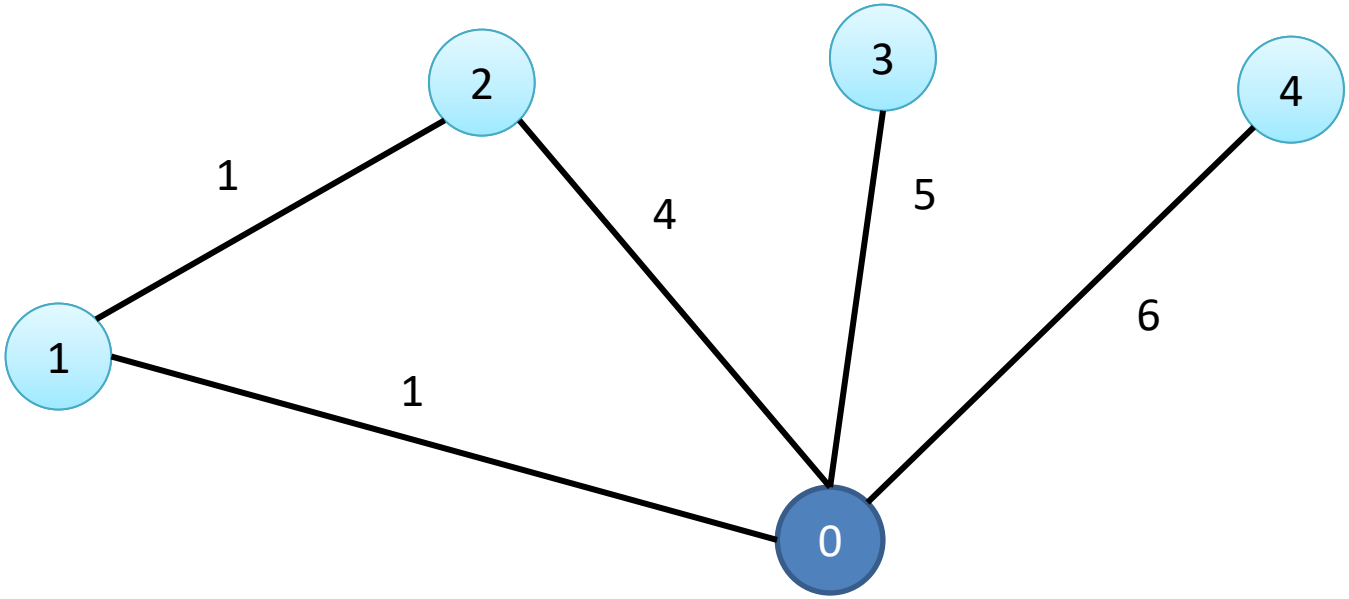
Flooding Topology

Разработка алгоритма

- 2-edge-connected FT
  - Algorithm 1 (relaxation procedure)
  - Algorithm 2 (recursive relaxation)
  - Algorithm 3 (bounded-degree spanning tree)
  - Algorithm 4 (2-edge-connected FT)
- 2-vertex-connected FT
- Special cases
  - Bipartite graphs
  - Fat tree graphs
  - Clos networks

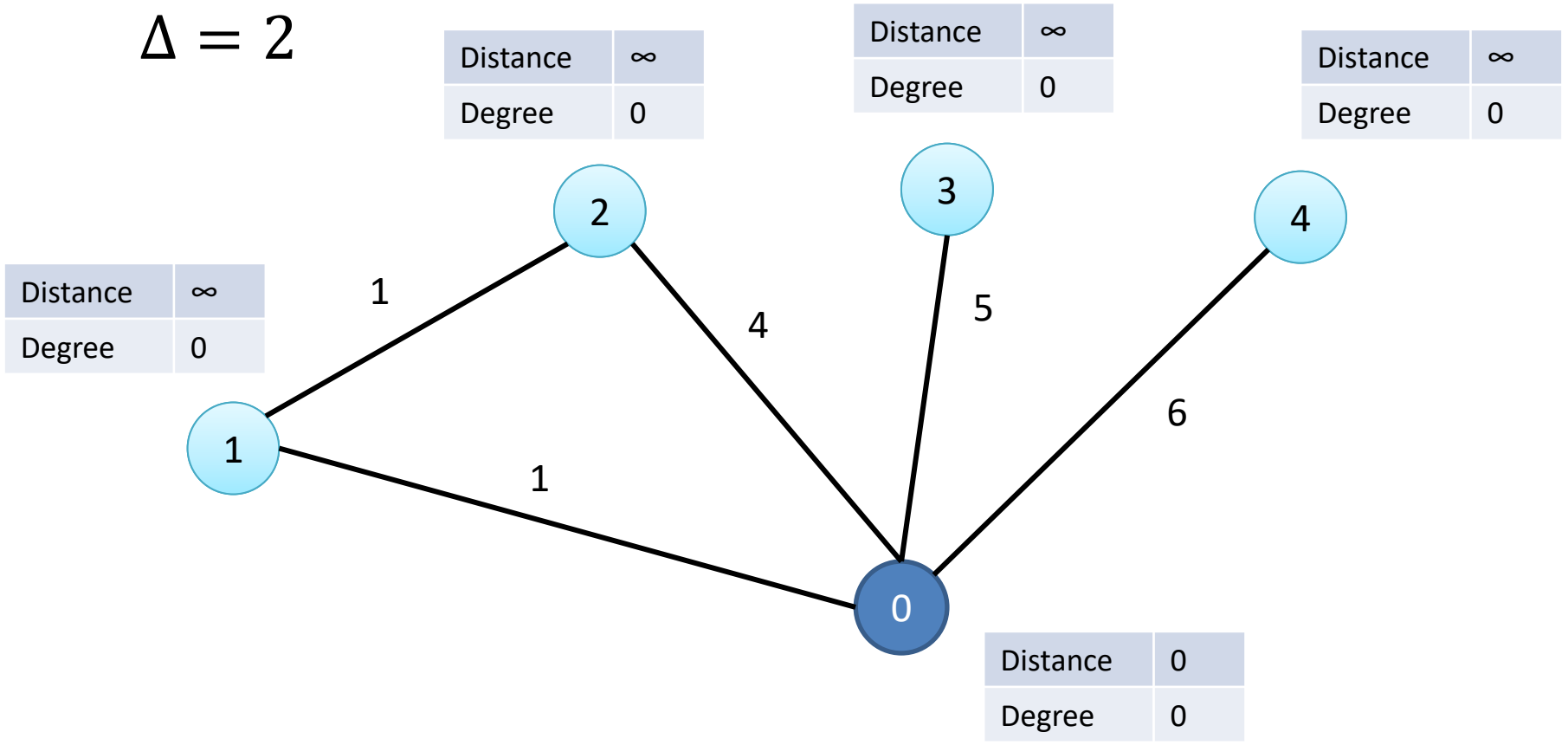
Graph №1

$\Delta = 2$



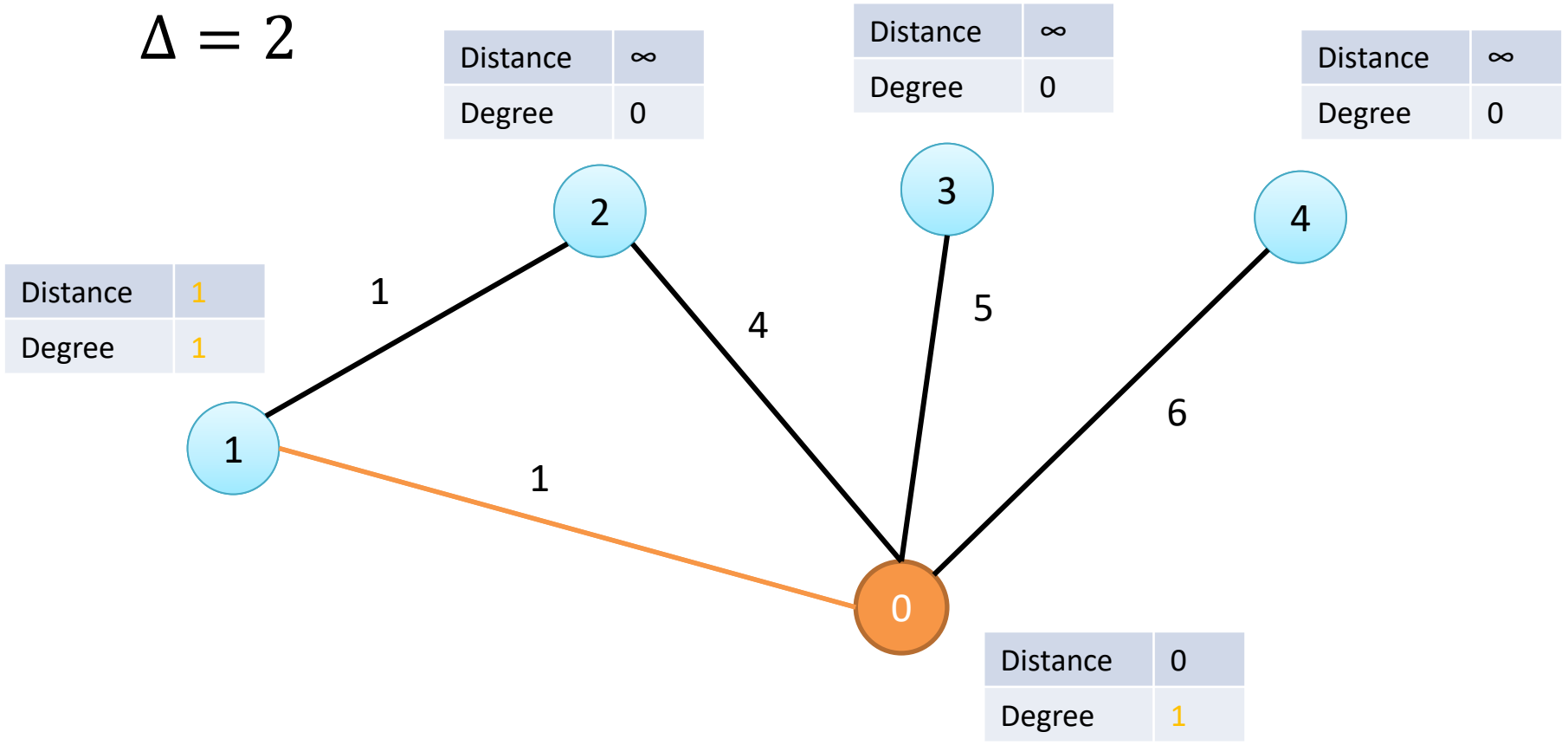
Bounded-degree spanning tree

$\Delta = 2$



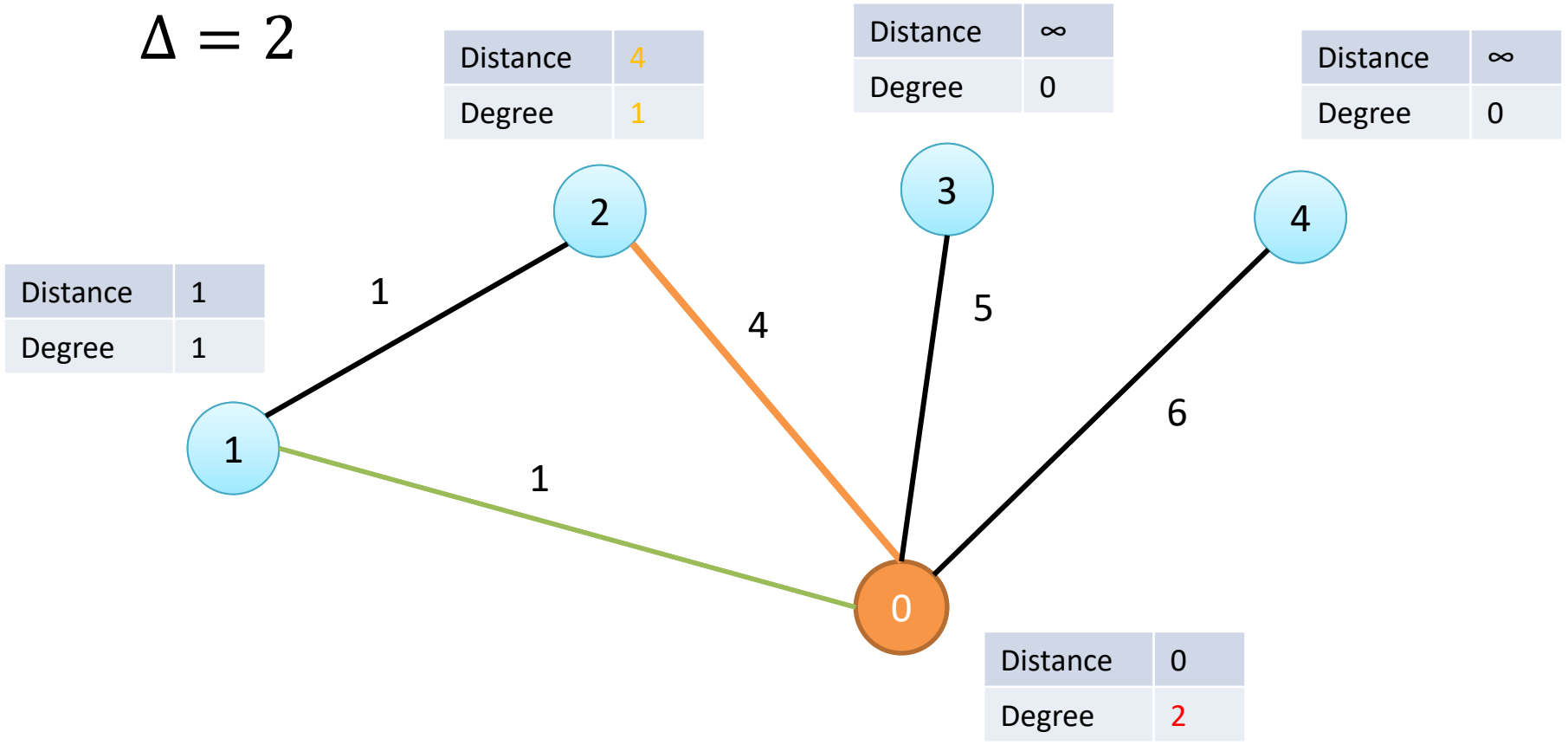
Bounded-degree spanning tree

$\Delta = 2$



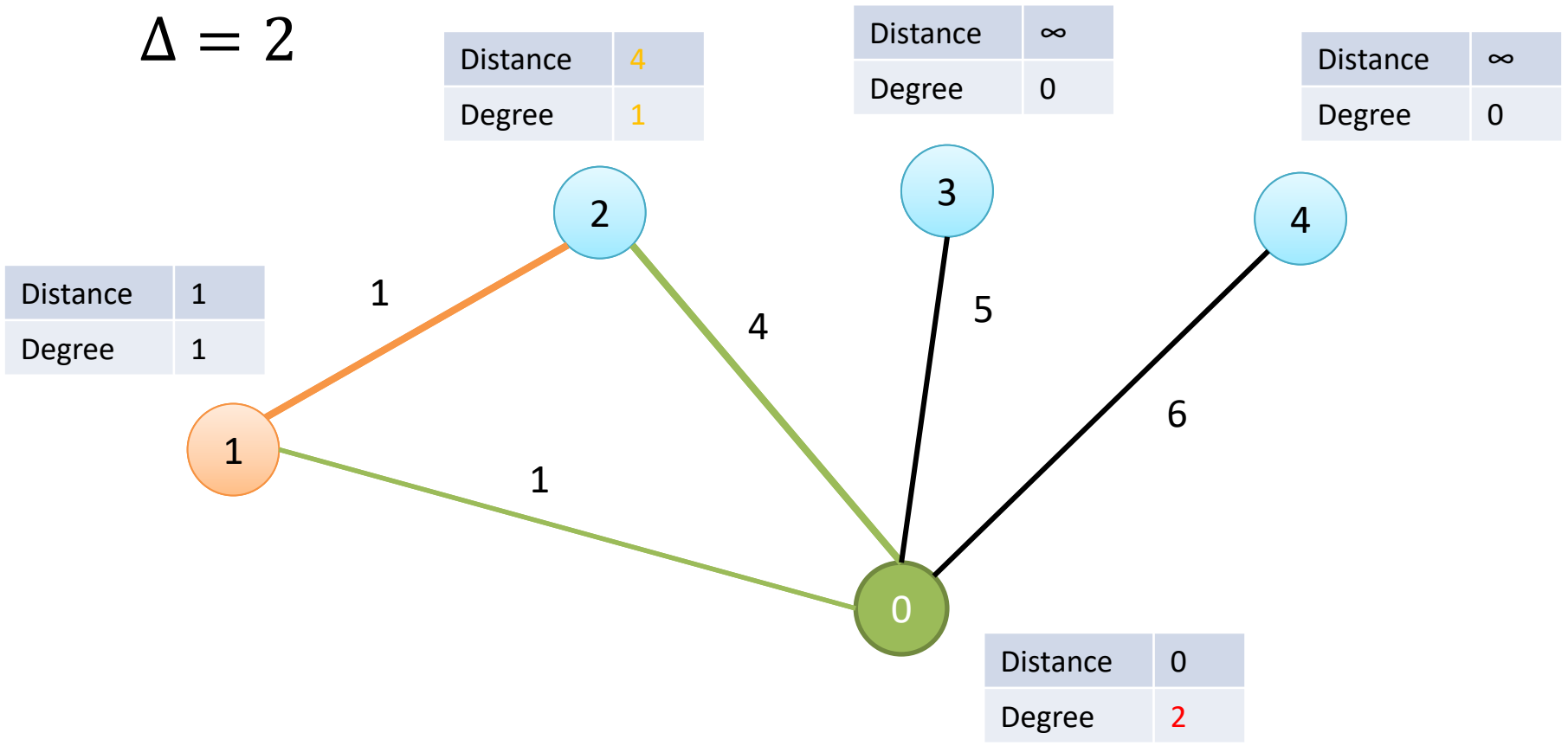
Bounded-degree spanning tree

$\Delta = 2$



Bounded-degree spanning tree

$\Delta = 2$



Relaxation

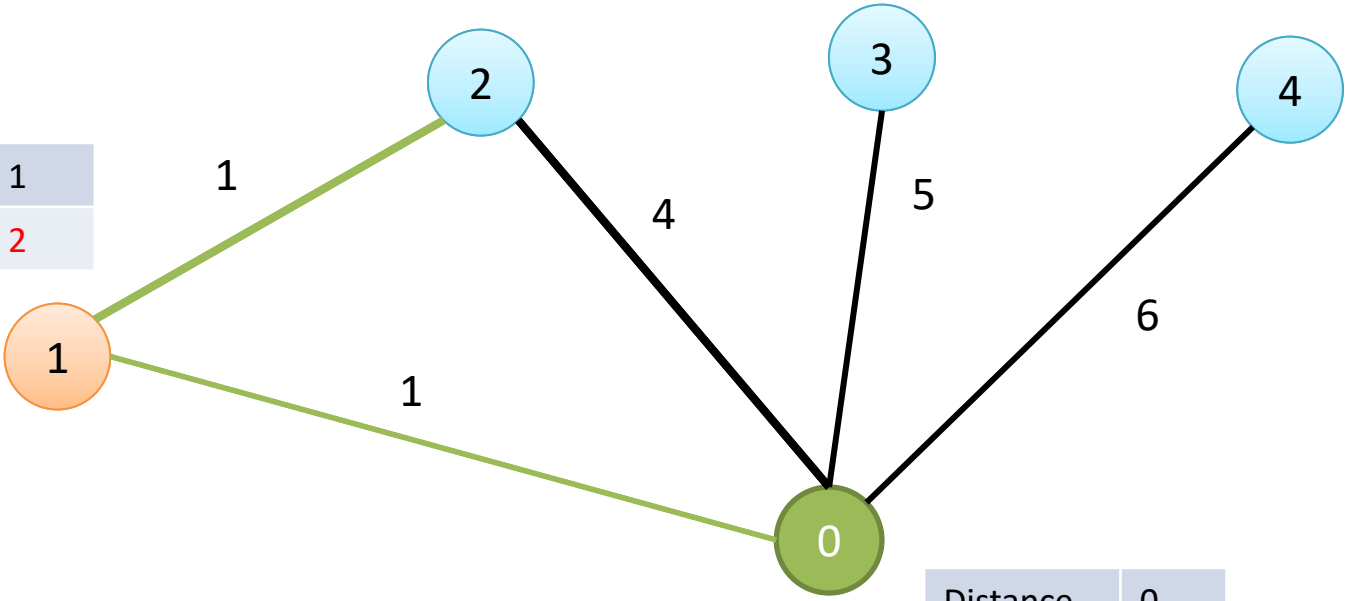
$\Delta = 2$

Distance	2
Degree	1

Distance	∞
Degree	0

Distance	∞
Degree	0

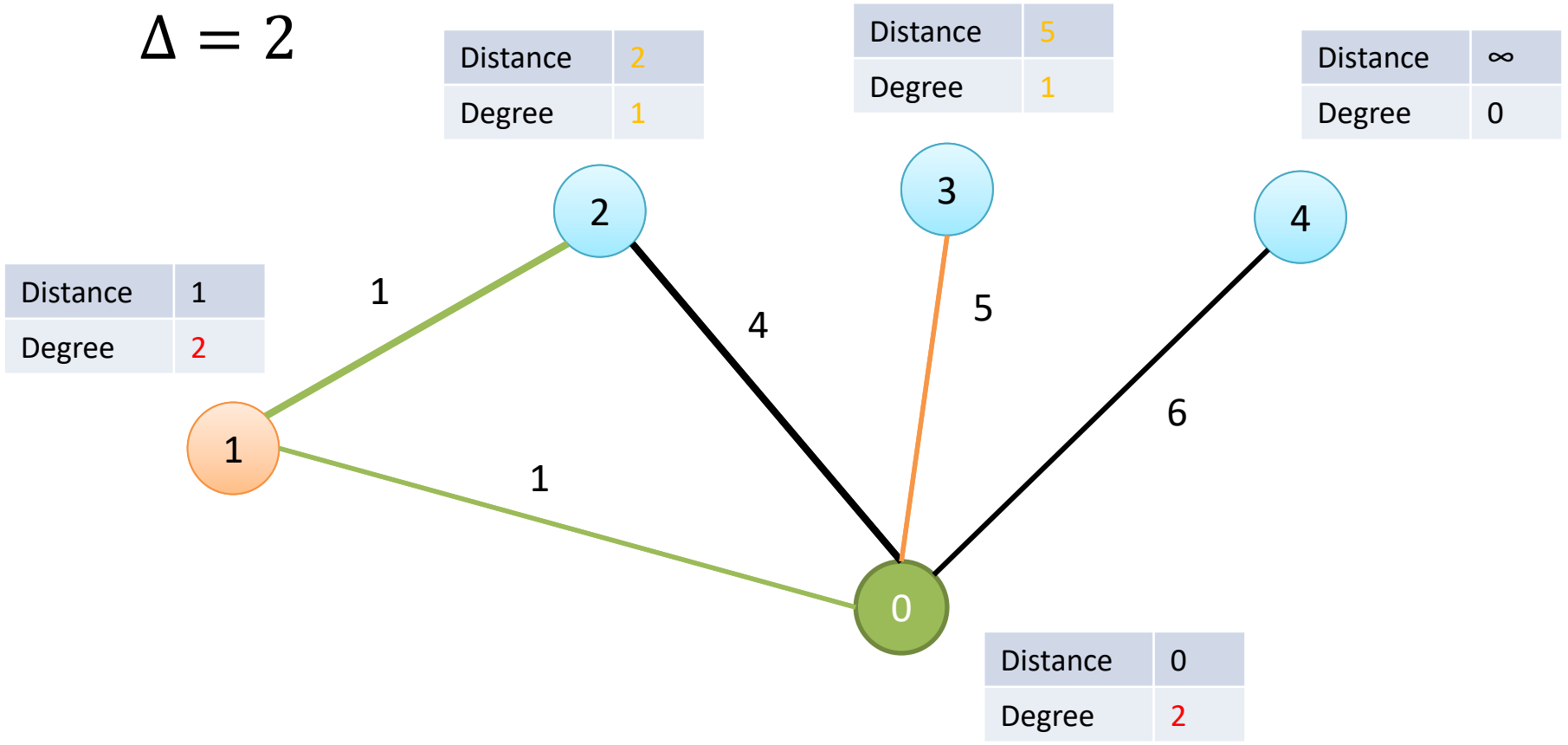
Distance	1
Degree	2



Distance	0
Degree	1

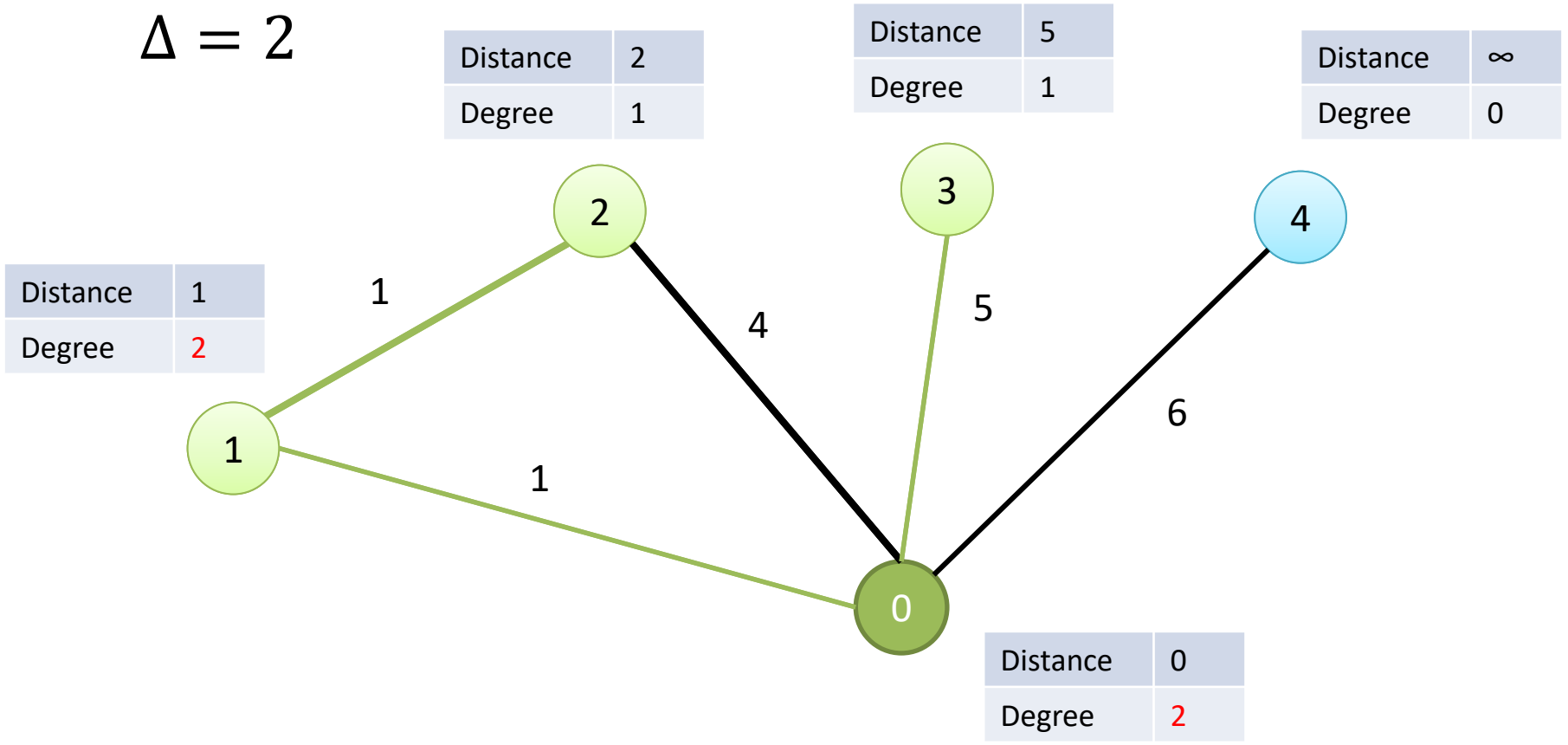
Recursive relaxation

$\Delta = 2$



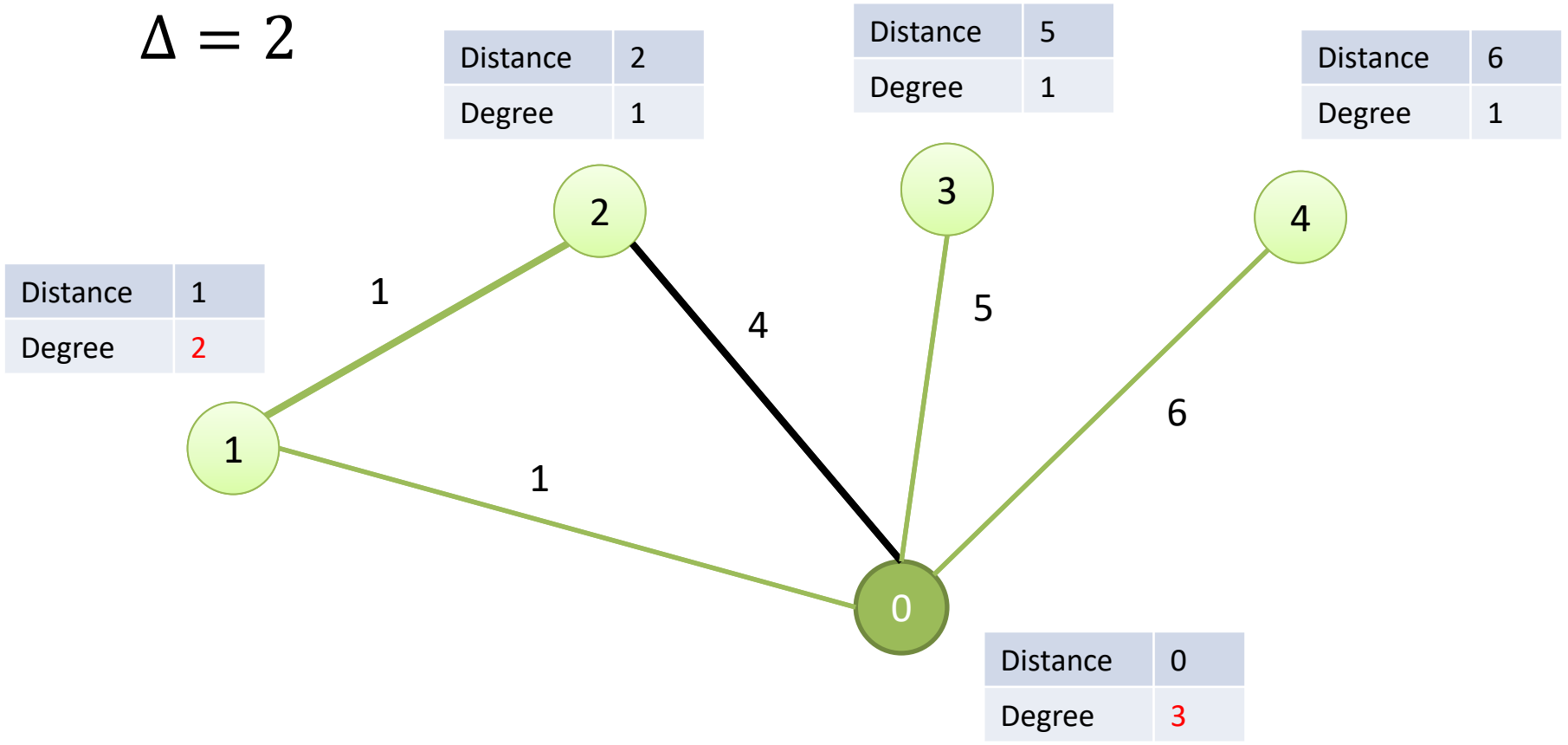
Bounded-degree spanning tree

$\Delta = 2$

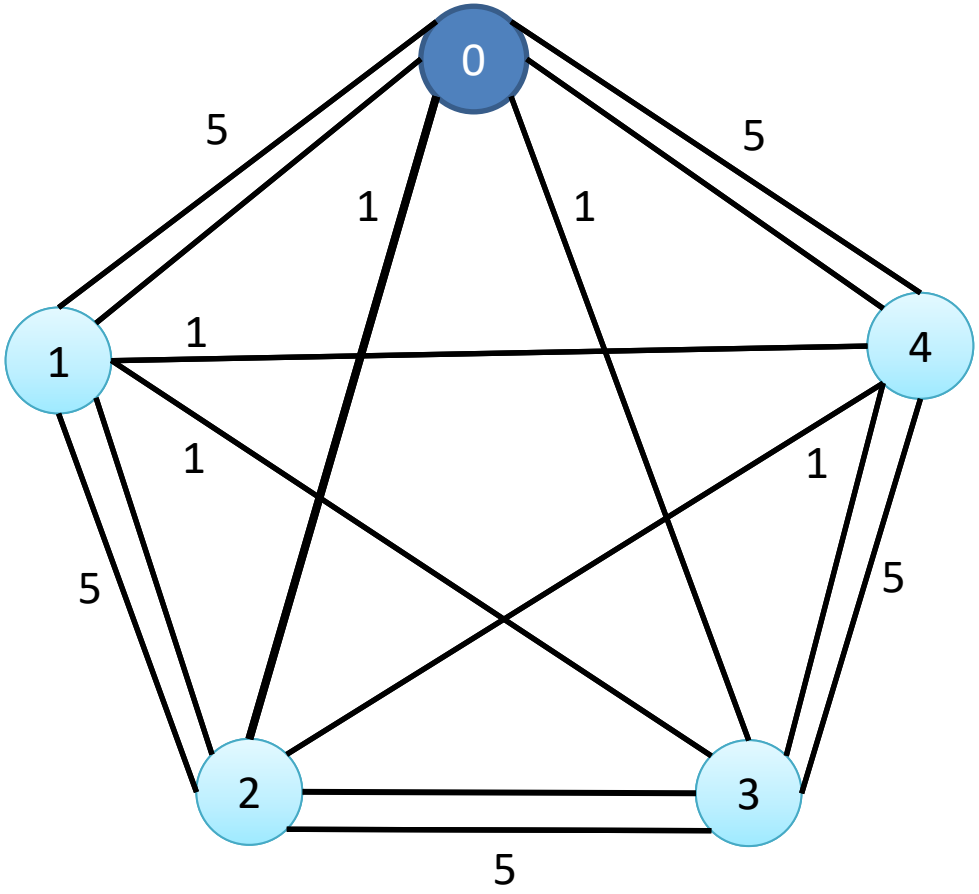


Spannify

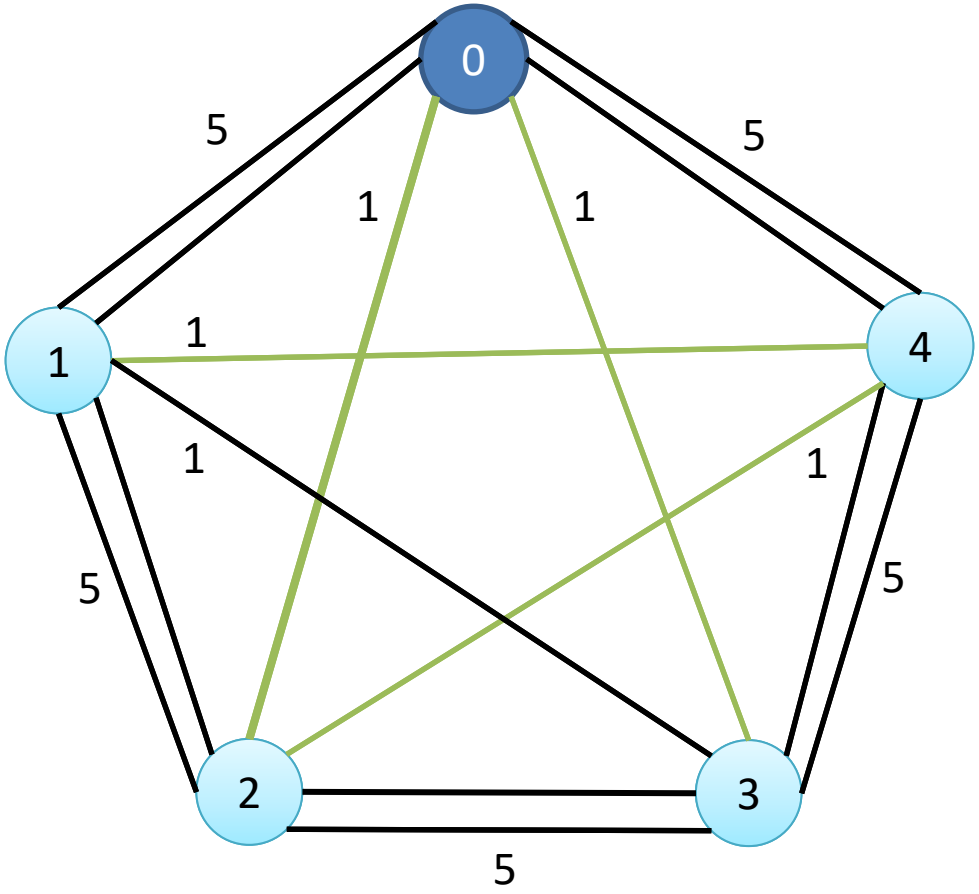
$\Delta = 2$



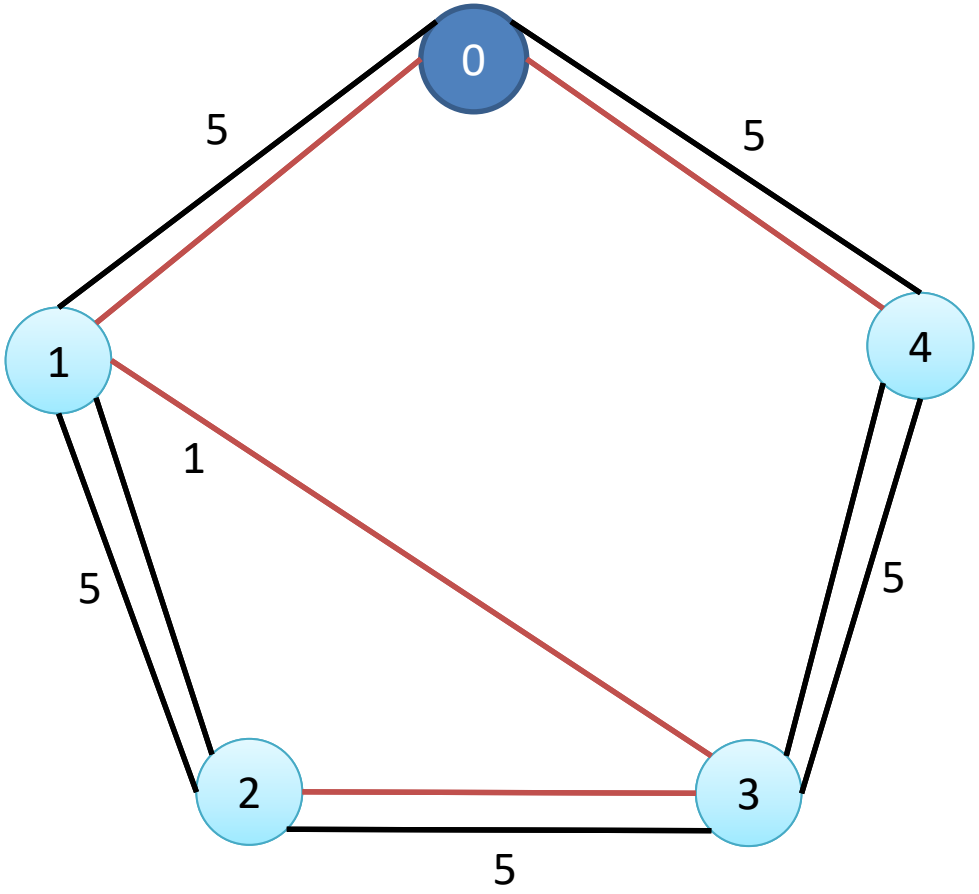
Graph №2



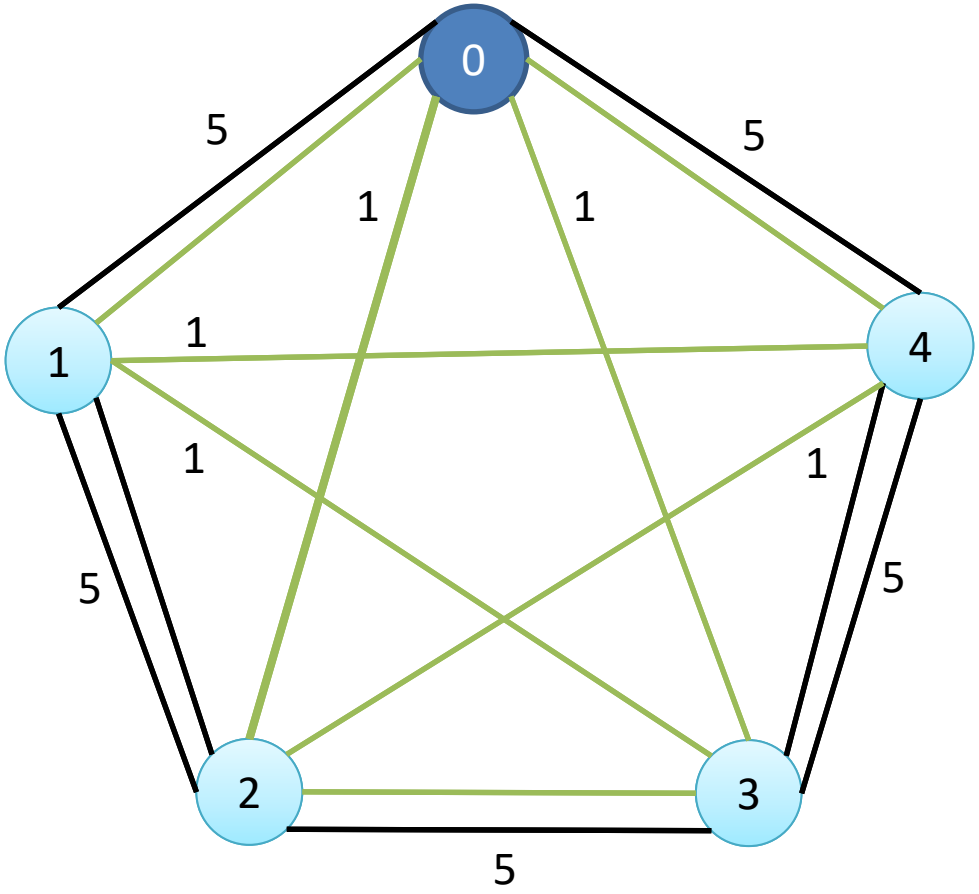
2-edge-connected FT



2-edge-connected FT



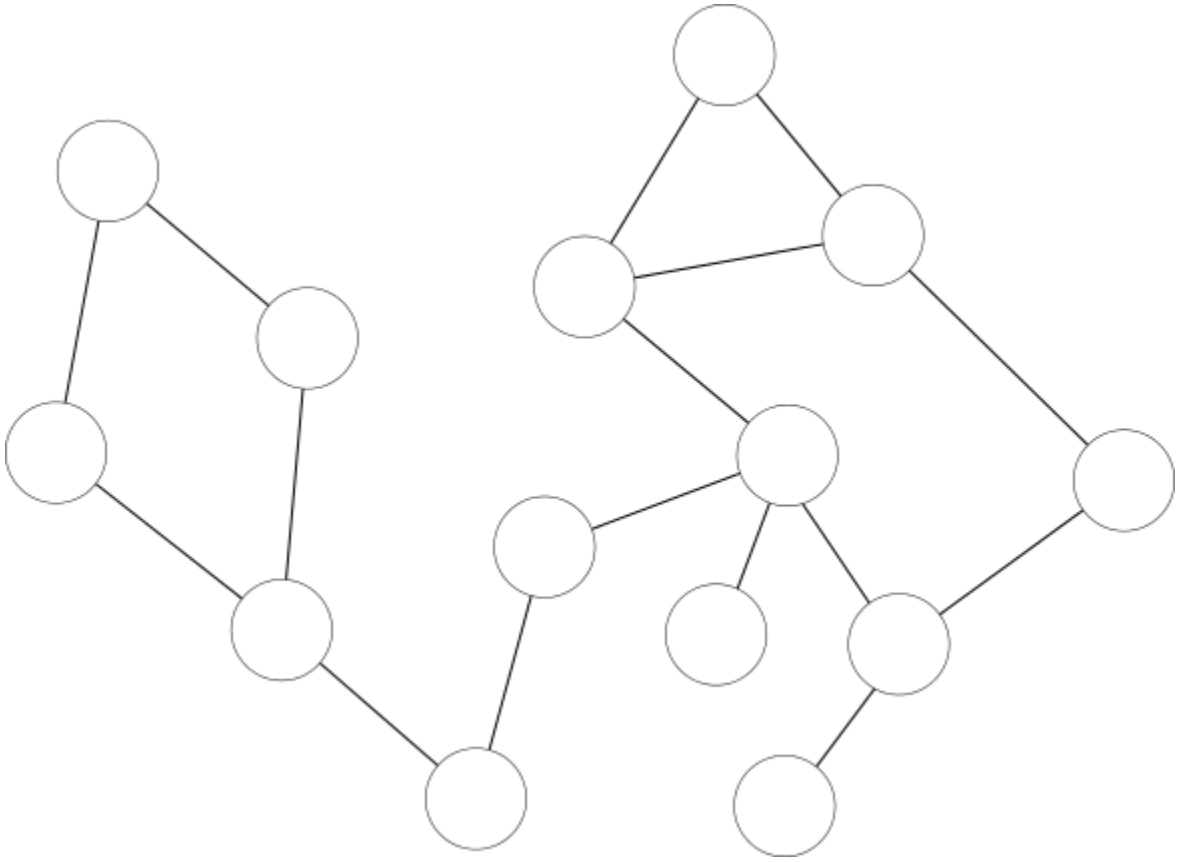
2-edge-connected FT



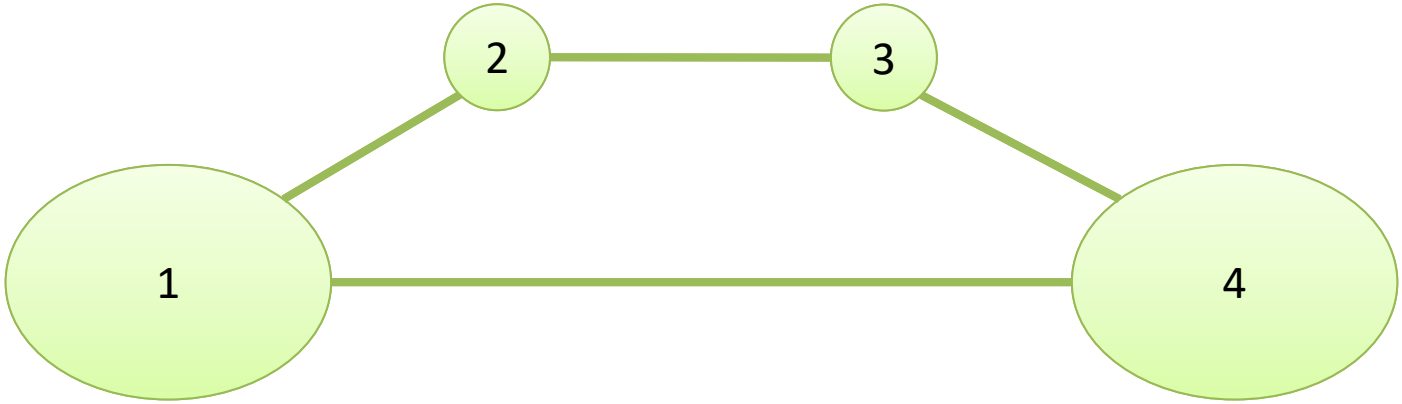
2-vertex-connected FT

- 2-edge-connected FT
- Find biconnected components
- Add edges to make whole graph biconnected

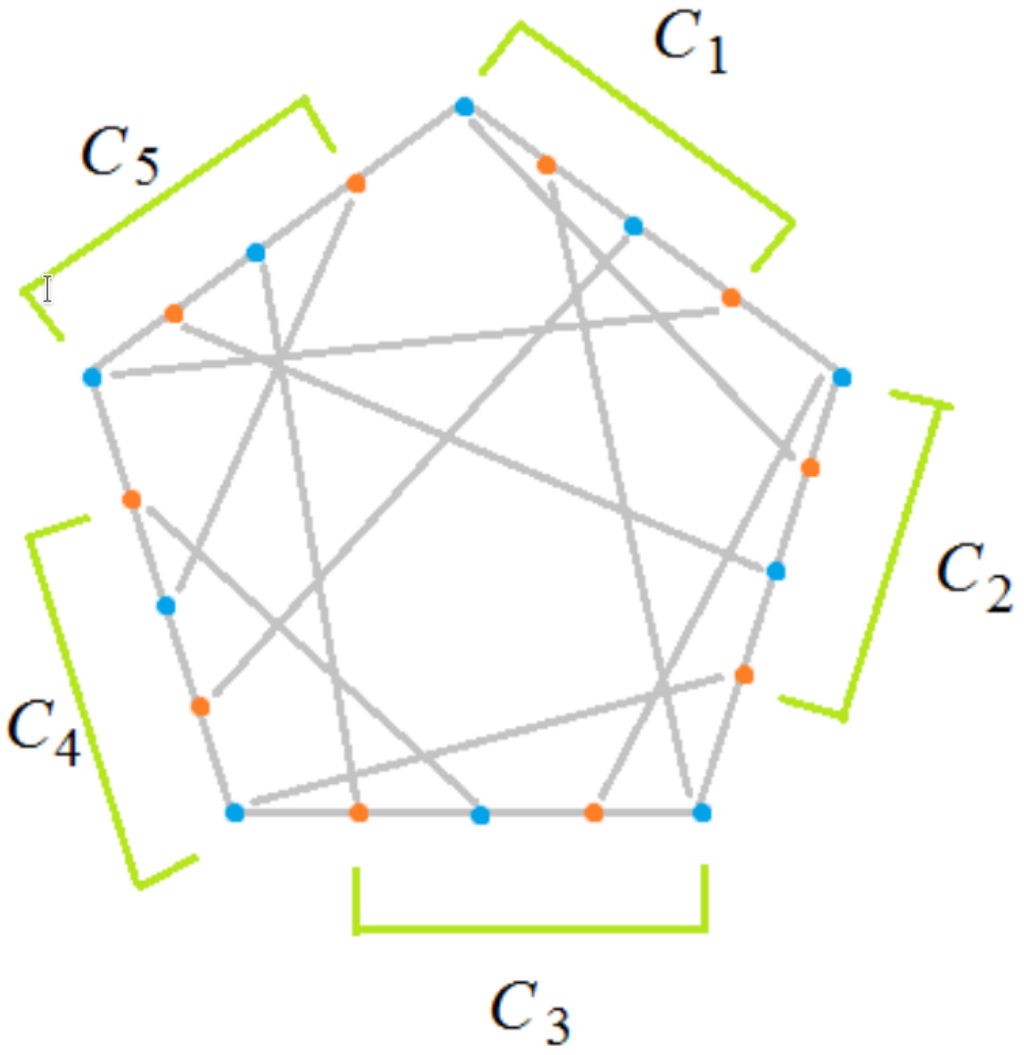
Biconnected components



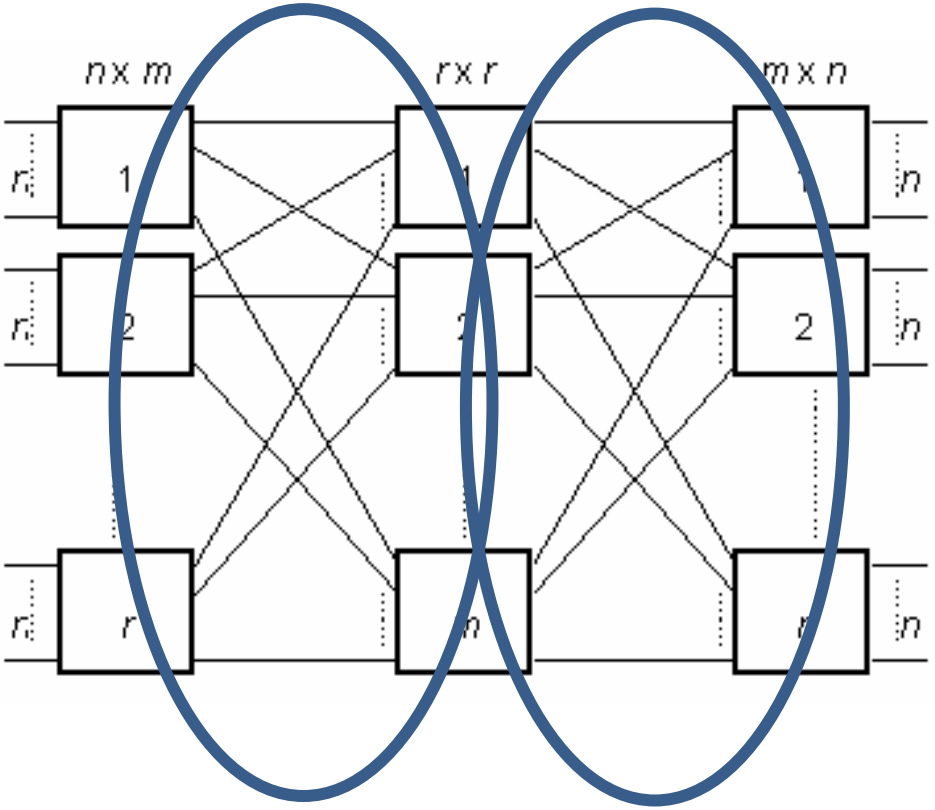
2-vertex-connected FT



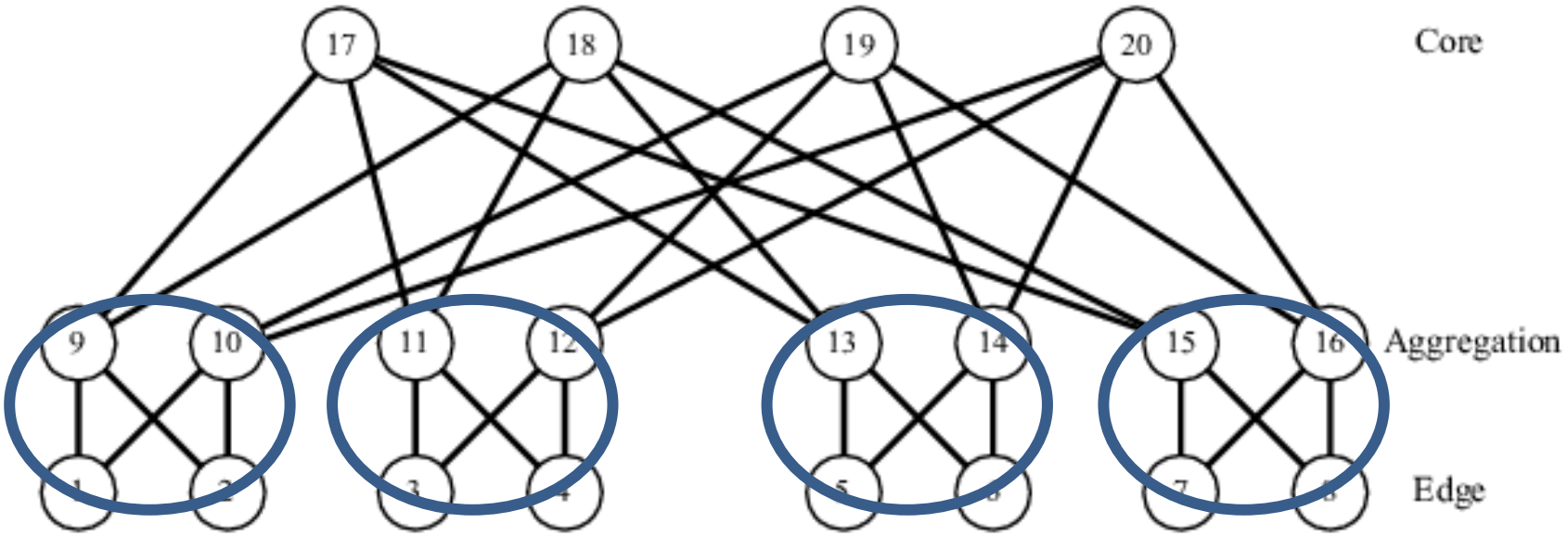
Special cases



Special cases: Clos network



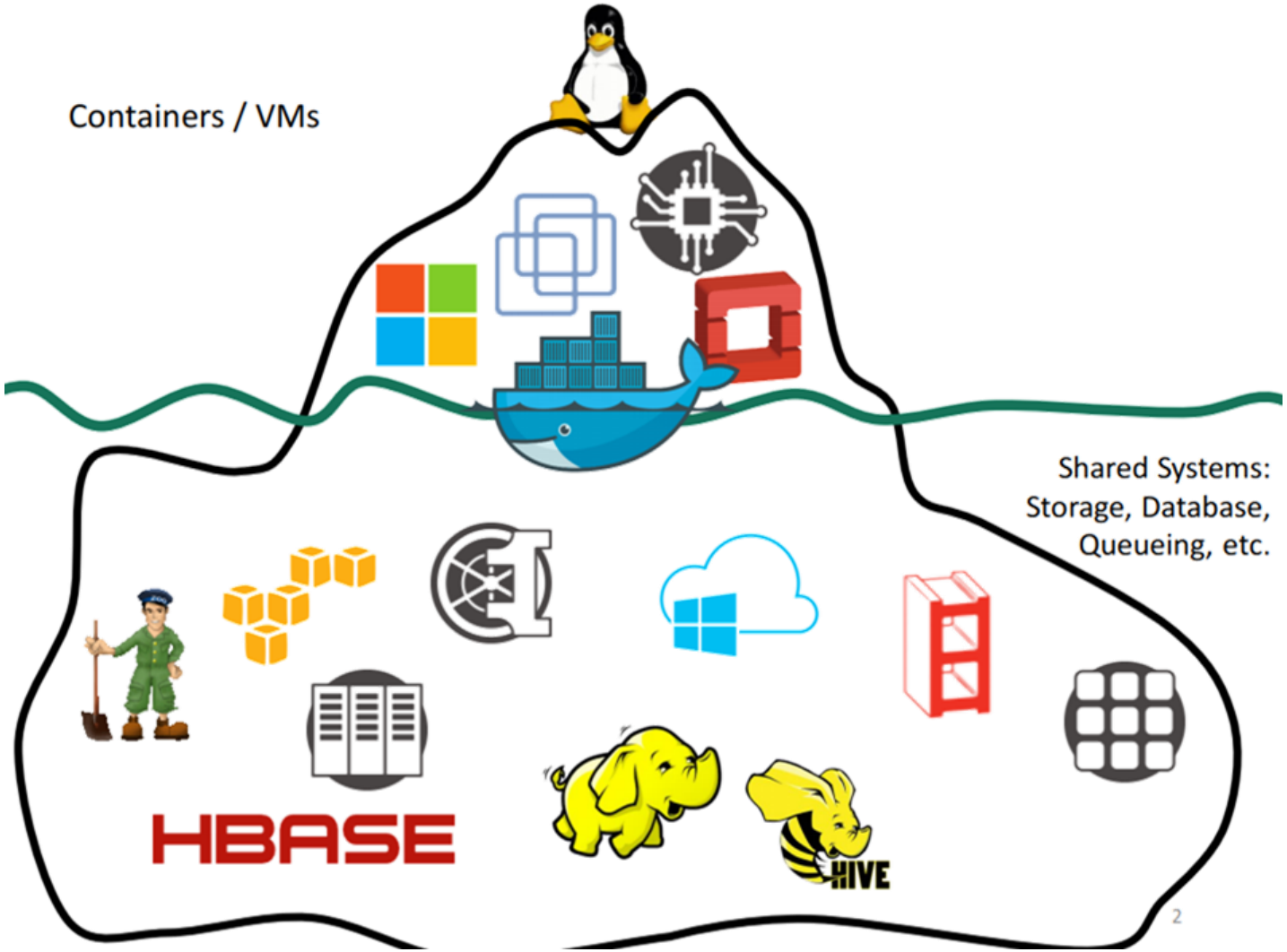
Special cases: Fat tree



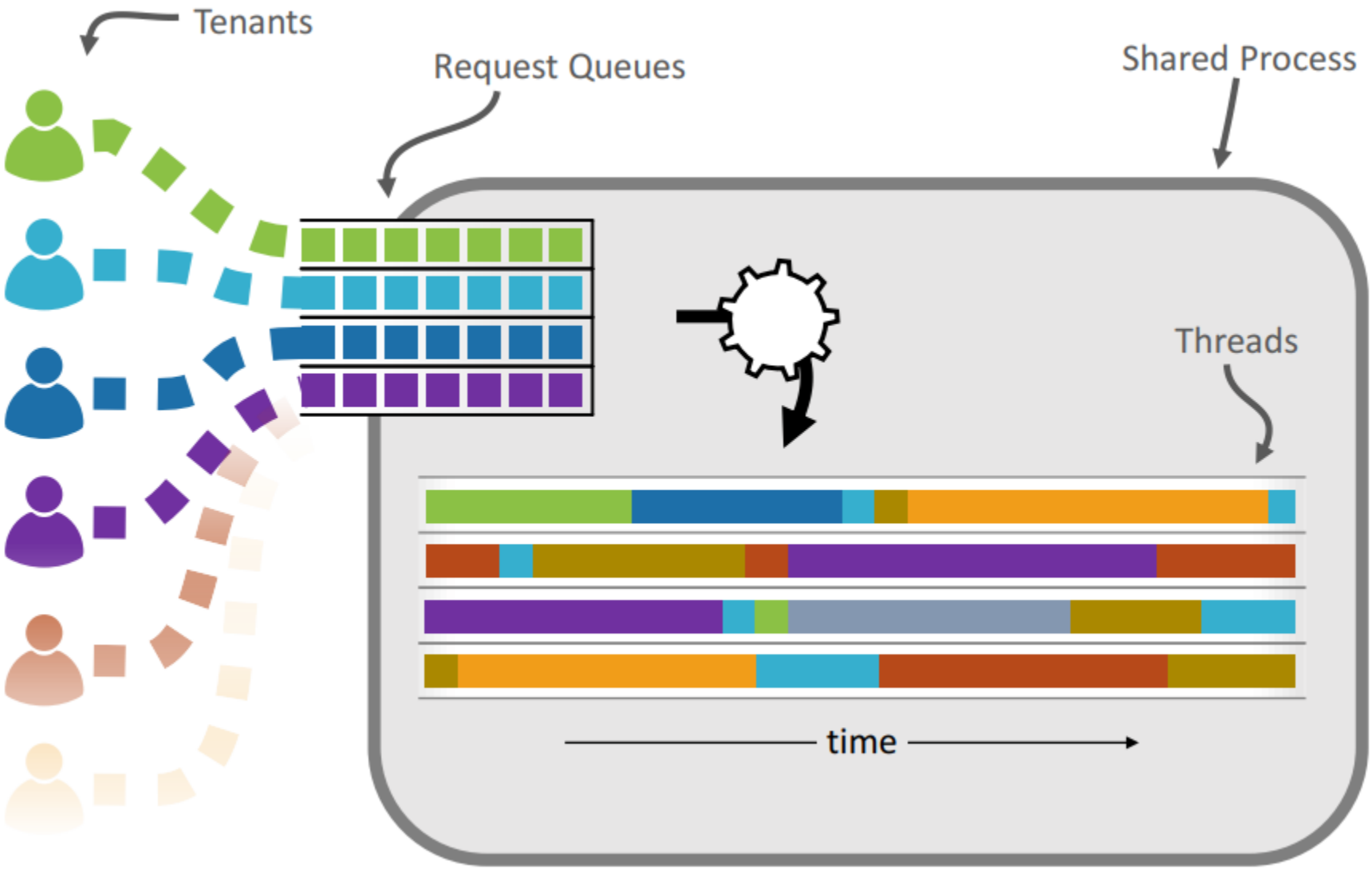
Планирование в ЦОД

- Распределение приложений по серверам
  - Выбор группы серверов внутри ЦОД
  - Разделение процессорного времени
  - Квотирование жёсткого диска
- Управление передачей пакетов
  - Распределение пропускной способности
  - Обработка пакетов на серверах приложений

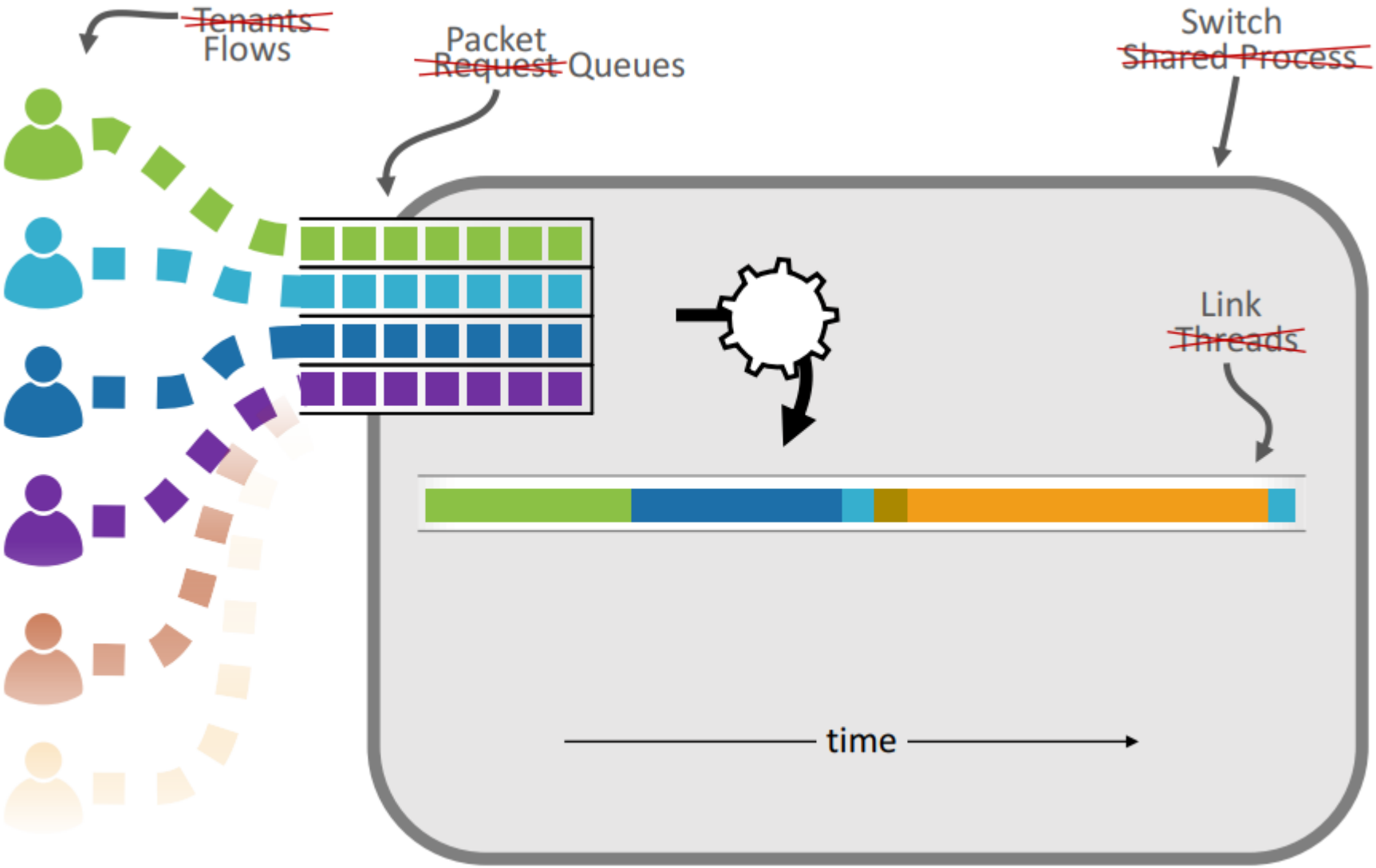
Иерархия облачных приложений



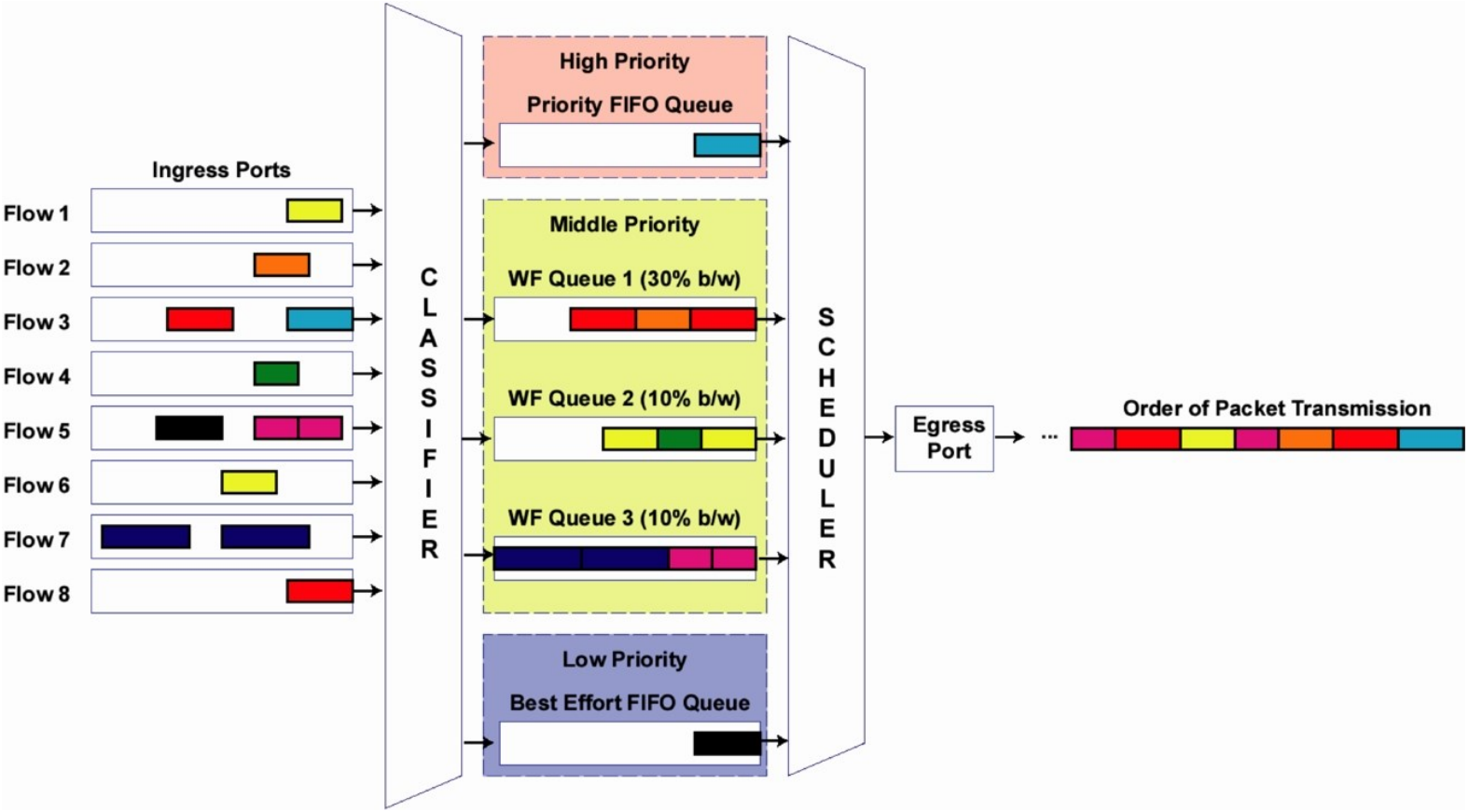
Инфраструктурные приложения обслуживают запросы от тенантов



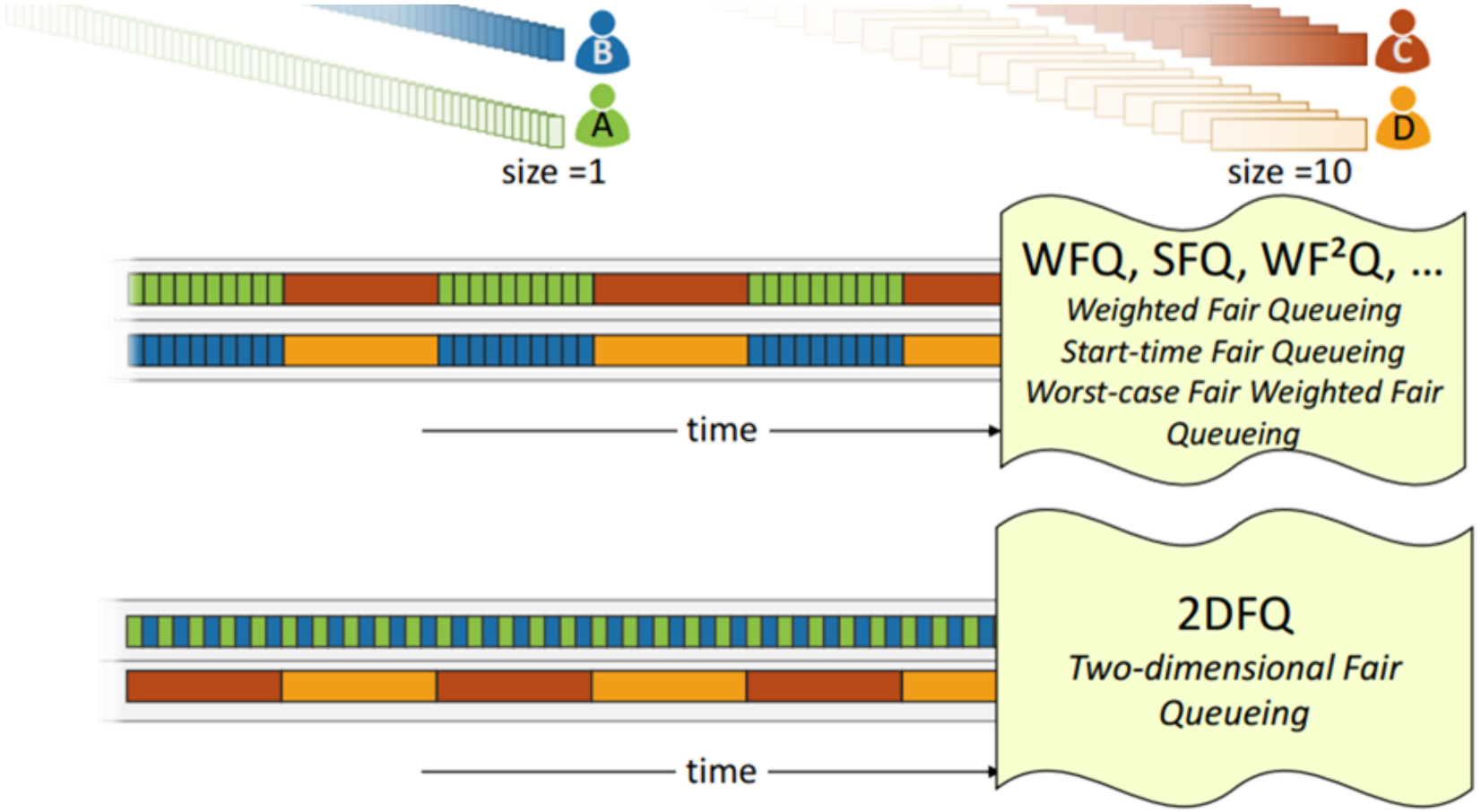
Проблема распределения ресурсов похожа на планирование пакетов



Дисциплины обслуживания и их устройство на коммутаторах

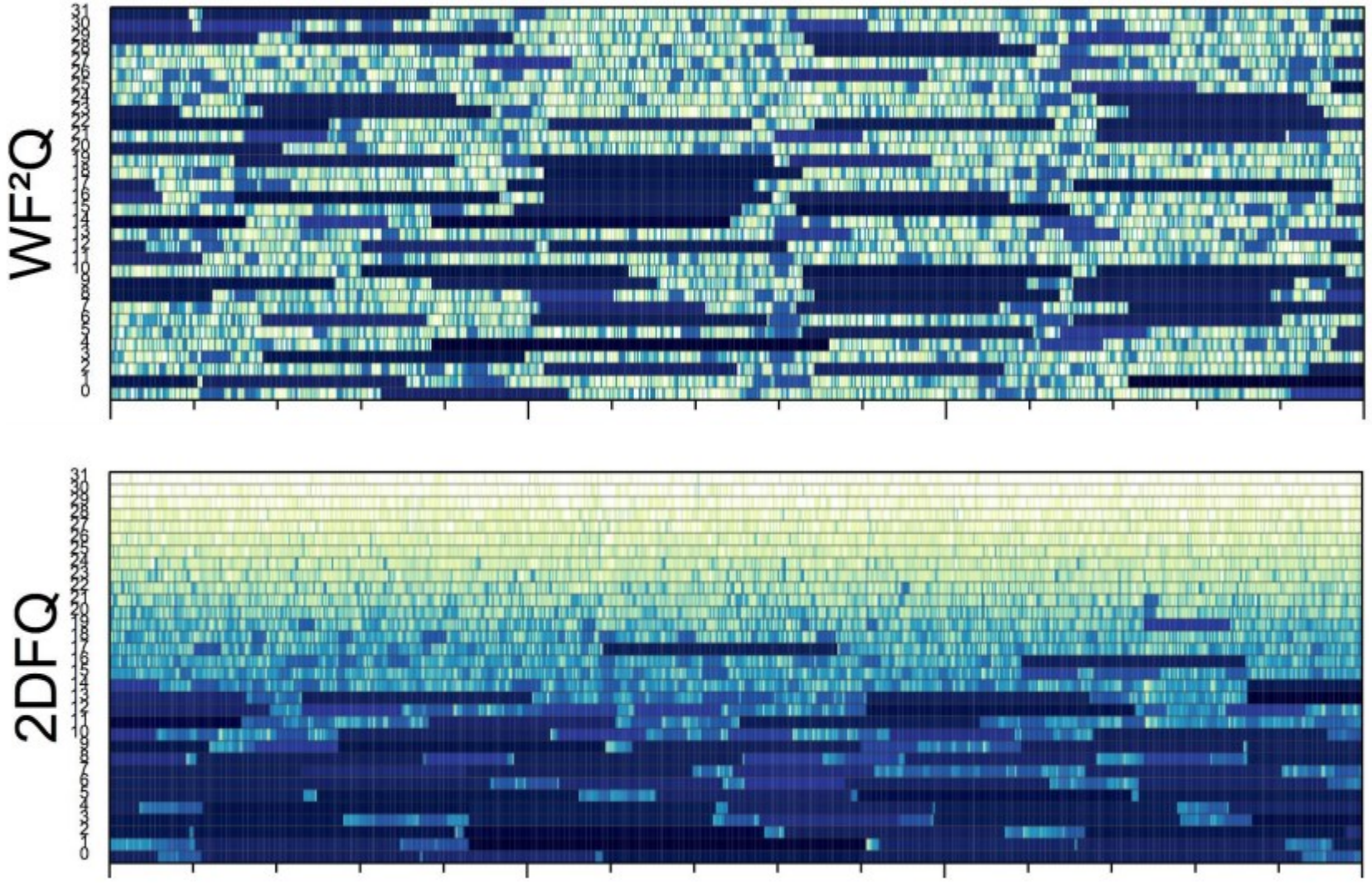


Алгоритмы планирования потоков для приложений ЦОД

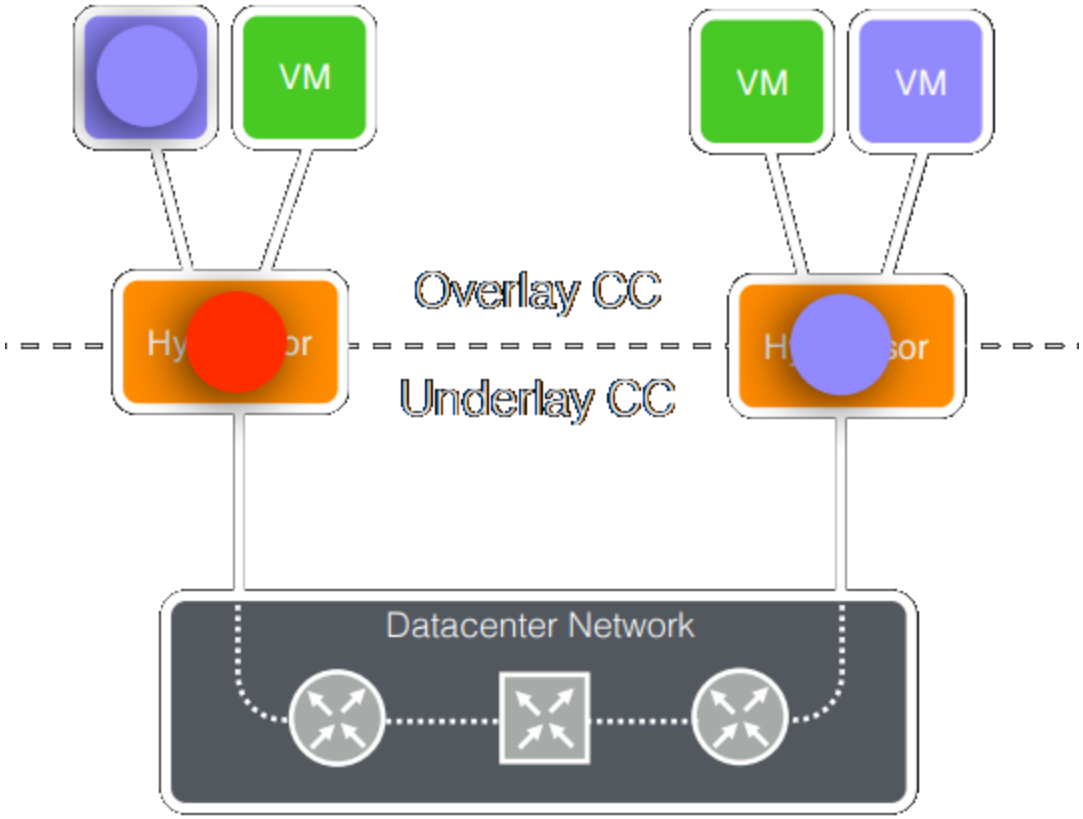


Построение таких расписаний, которые не только бы распределяли нагрузку, но и позволяли бы достичь наибольшей равномерности распределения

Распределение нагрузки с 2DFQ



Оптимизация приложений под ЦОД



В ЦОД используются более эффективные алгоритмы управления перегрузкой TCP (DCTCP, MCP, etc)

Как подменить протокол перегрузки, который использует приложение на тот протокол, который используется в ЦОД?

AC/DC TCP: Virtual Congestion Control Enforcement for Datacenter Networks. Sigcomm 2016.
Virtualized Congestion Control. Sigcomm 2016.

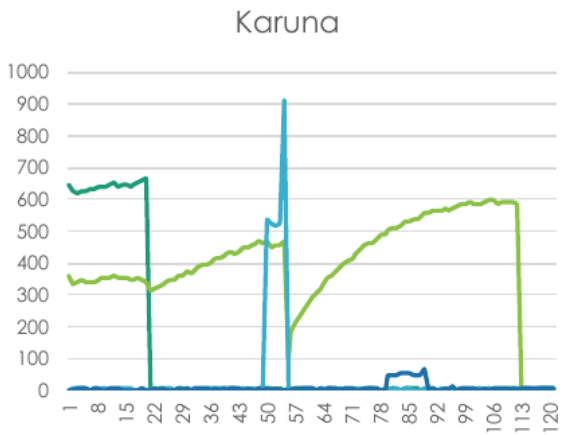
Планирование потоков данных

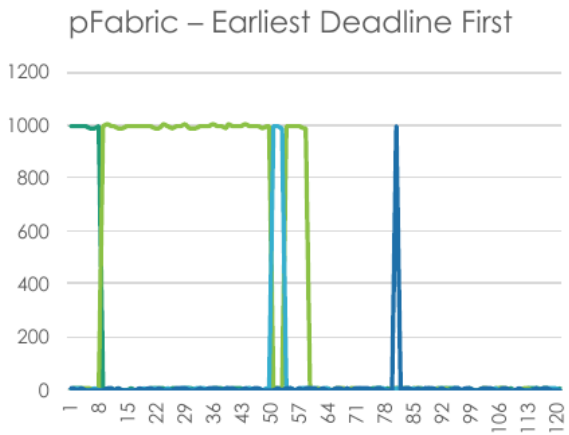
- Потоки без жёстких директивных интервалов
  - Резервное копирование
- Потоки реального времени
  - Map-reduce
- Как распределять пропускные способноси каналов между потоками?
  - Использовать для разных типов потоков разные алгоритмы управления перегрузкой TCP
  - Вычислять функцию изменения congestion window в зависимости от директивного интервала
  - Какие эвристики использовать для вычисления функции управления перегрузкой?

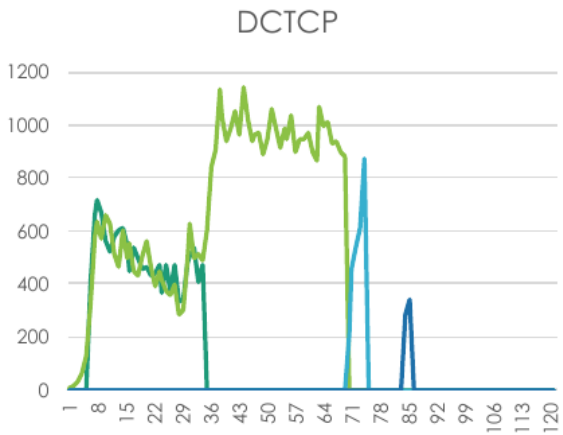
Karuna

ЦОД – место для прокрастинации

Flow	Size	Deadline	Start Time
1	14.4MB	20ms	0ms
2	48MB	120ms	0ms
3	3MB	5ms	50ms
4	0.5MB	10ms	80ms





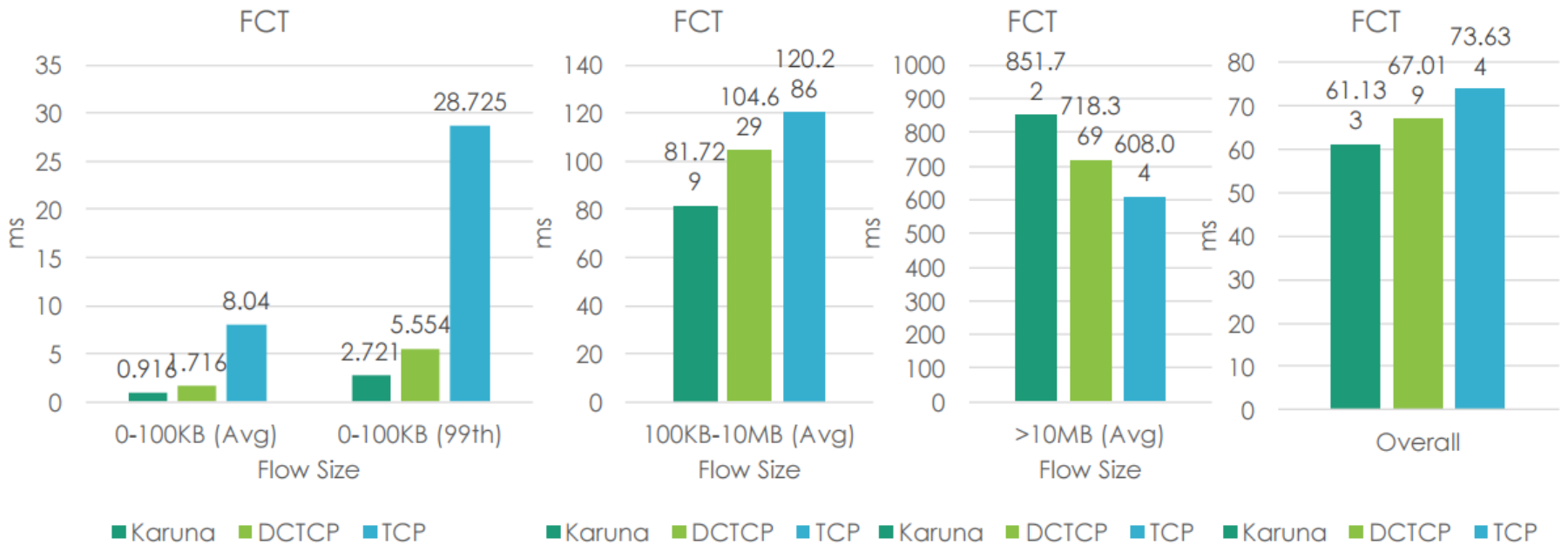


Karuna стремится передать данные прямо перед наступлением дедлайна, оставляя больше ресурсов для потоков без жёстких ограничений

Scheduling Mix-flows in Commodity Datacenters with Karuna. Sigcomm 2016.

Karuna

ЦОД – место для прокрастинации

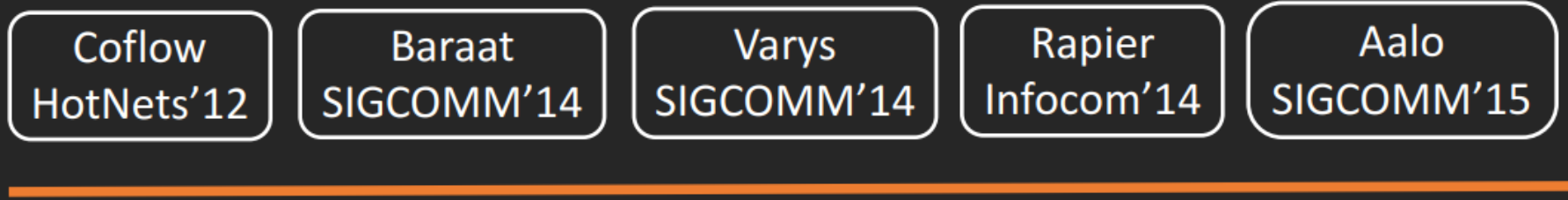


Karuna стремится передать данные прямо перед наступлением дедлайна, оставляя больше ресурсов для потоков без жёстких ограничений

Scheduling Mix-flows in Commodity Datacenters with Karuna. Sigcomm 2016.

Связанные потоки (coflows)

- Выполнение задач часто требует завершения сразу нескольких потоков
  - Map-Reduce
  - Бессмысленно ускорять отдельные потоки, если приложение всё равно не завершится
- Планирование связанных потоков может существенно ускорить работу ЦОД
  - Как идентифицировать связанные потоки?



Assumption: *all distributed data-parallel applications have to be modified to correctly use the same coflow API*

- **Difficulty 1:** Enable current Coflow API requires intrusive refactoring
- **Difficulty 2:** Hard to modify all applications and keep them up to date

Can we automatically identify and schedule coflows without manually modifying any data-parallel applications?

CODA: Toward Automatically Identifying and Scheduling Coflows in the Dark. Sigcomm 2016.

Резюме

- Несмотря на огромную мощность современных ЦОД существует большой простор для увеличения их эффективности
- Основная проблема ЦОД – внутренние ифраструктурные ограничения
- Концепция resource pooling стремиться преодолеть эти ограничения, представляя вычислительные ресурсы ЦОД в виде однородного множества
- Поддержание такой абстрации требует преодоления множества трудностей

Bandwidth on Demand

- Меняется организация приложений: указываем требование к пропускной способности соединения!
- SDN vs набор скриптов
- Маршрутизация
- Приоритизация клиентов

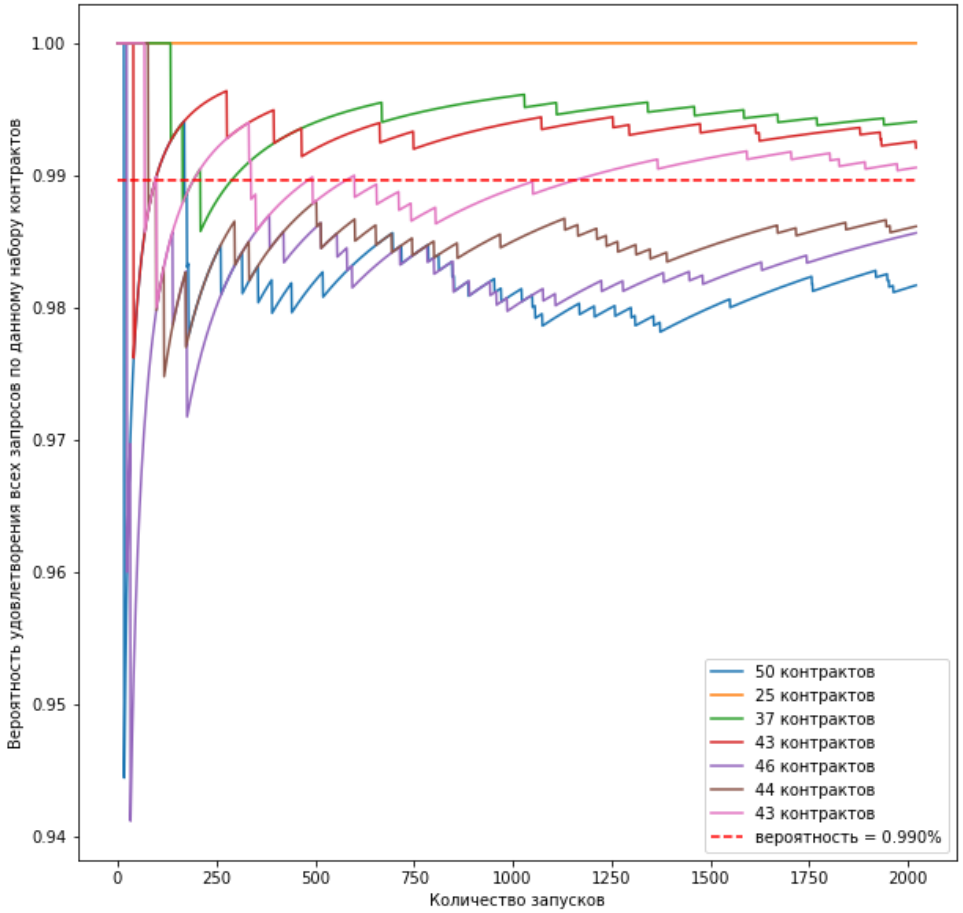
Формализация задачи

- Можно свести к задаче линейного программирования

$$\min_{M,K,E} \sum_{m=0}^{M} \sum_{k=1}^{K} \sum_{j:(v_i,v_j)\in E} q_{i,j}^{C_k}(t_m) \quad (*_{\text{Ц}_1})$$

$$(*_{\text{огр}_1}) \begin{cases} \sum_{k=1}^{K} {q_{i,j}}^{C_k}(t_m) \le \rho_{i,j} - {q_{i,j}}^{F}(t_m) & (v_i, v_j) \in E, m \in \overline{0,M}\ (1) \\ \sum_{m=0}^{M_k} \sum_{i:(v_{a_k},v_i)\in E} {q_{a_k,i}}^{C_k}(t_m) * \varepsilon = \Delta_k & k = \overline{1,K}\ (2) \\ \sum_{m=0}^{M_k} \sum_{i:(v_i,v_{a_k})\in E} {q_{i,a_k}}^{C_k}(t_m) * \varepsilon = 0 & k = \overline{1,K}\ (3) \\ \sum_{m=0}^{M_k} \sum_{i:(v_i,v_{b_k})\in E} {q_{i,b_k}}^{C_k}(t_m) * \varepsilon = \Delta_k & k = \overline{1,K}\ (4) \\ \sum_{m=0}^{M_k} \sum_{i:(v_{b_k},v_i)\in E} {q_{b_k,i}}^{C_k}(t_m) * \varepsilon = 0 & k = \overline{1,K}\ (5) \\ \sum_{i:(v_i,v_r)\in E} {q_{i,r}}^{C_k}(t_m) - \sum_{j:(v_r,v_j)\in E} {q_{r,j}}^{C_k}(t_m) = 0 & v_r \in V\backslash\{v_{a_k,b_k}\}, k = \overline{1,K}, m = \overline{0,M}\ (6) \\ {q_{i,j}}^{C_k}(t_m) - {q_{i,j}}^{C_k}(t_{m+1}) = 0 & m = \overline{0,M_k-1}, k = \overline{1,K}, (v_i,v_j) \in E\ (7) \\ {q_{i,j}}^{C_k}(t_m) \ge 0 & (v_i,v_j) \in E, k = \overline{1,K}, m = \overline{0,M}\ (8) \end{cases}$$

Оценка резерва



Программа курса

Подходы:

1. ***Управление перегрузкой***
   - Современные протоколы управления перегрузкой TCP
2. ***Демультиплексирование/мультиплексирование***
   - Многопоточные транспортные протоколы
   - Маршрутизация на уровне интернет провайдеров
   - Network Coding
3. ***Сегментация***
   - TCP Proxy
4. ***Балансировка***
   - Балансировка нагрузки и управление трафиком

Модели:

- Сетевое исчисление: математический подход к качеству сервиса
- NS3: моделирование поведения сети с высокой точностью

Примеры:

- Управление сетевыми ресурсами в Центрах Обработки Данных
- Пропускная способность по требованию
- Обеспечение качества сервиса в сетях доставки контента
- HTTP3/QUIC