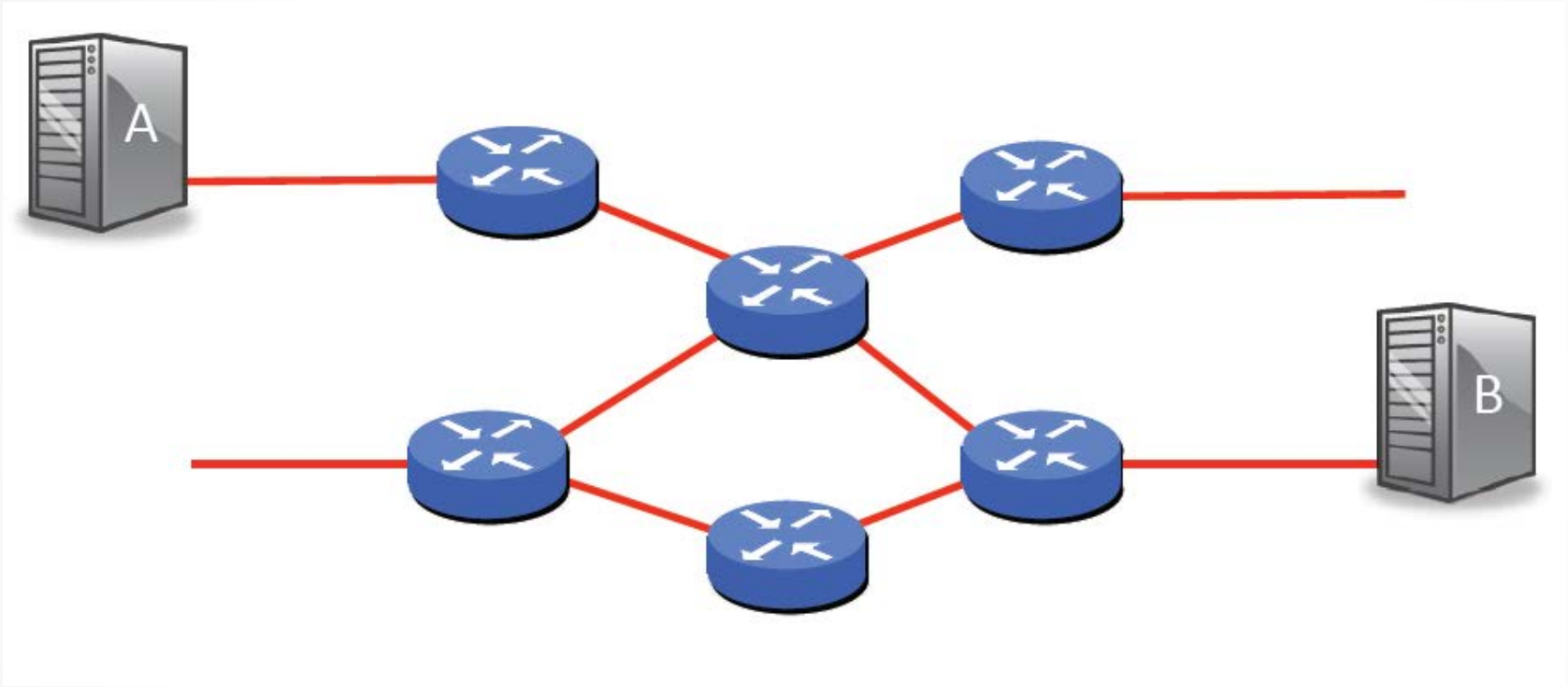
Основы маршрутизации в Интернет

(том 2 стр.30-52)

Введение в компьютерные сети

проф. Смелянский Р.Л.
Лаборатория Вычислительных комплексов
ф-т ВМК МГУ

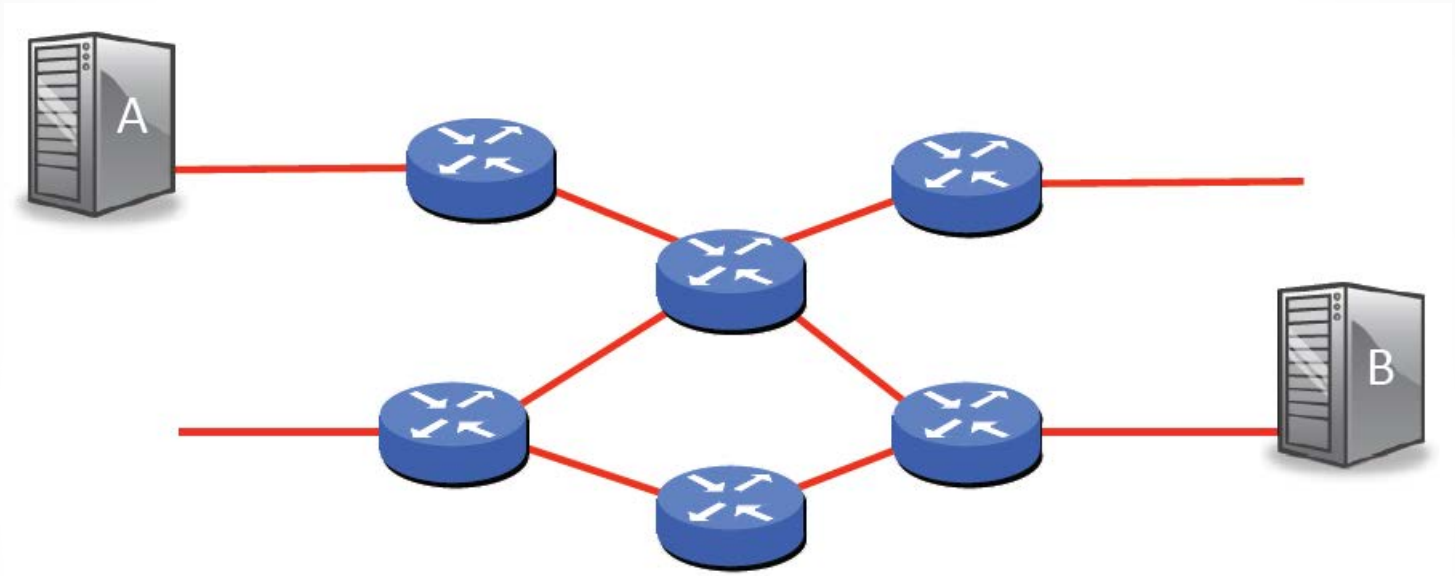
Проблема



- Кто должен определить как пакеты из A достигнут B?

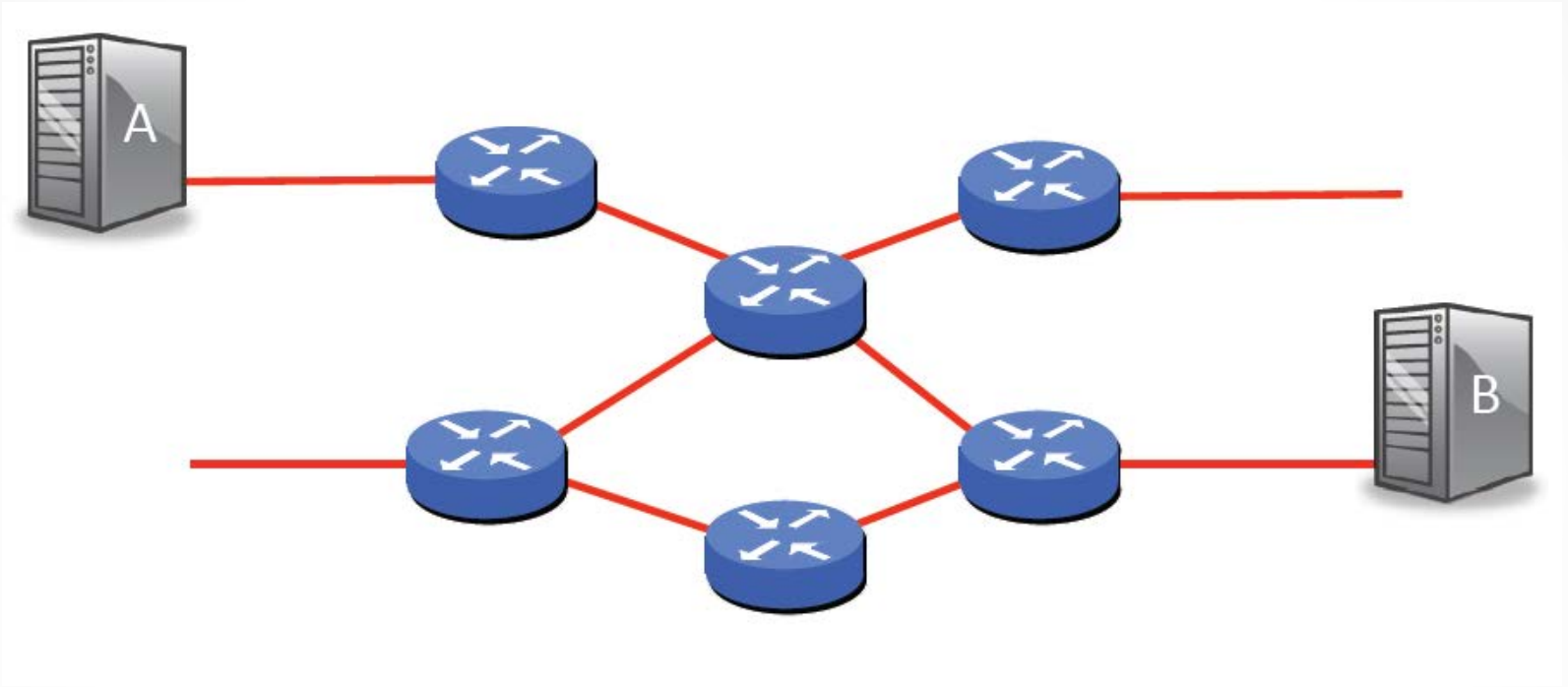
Критерий выбора маршрута?

Лавина (Flooding)



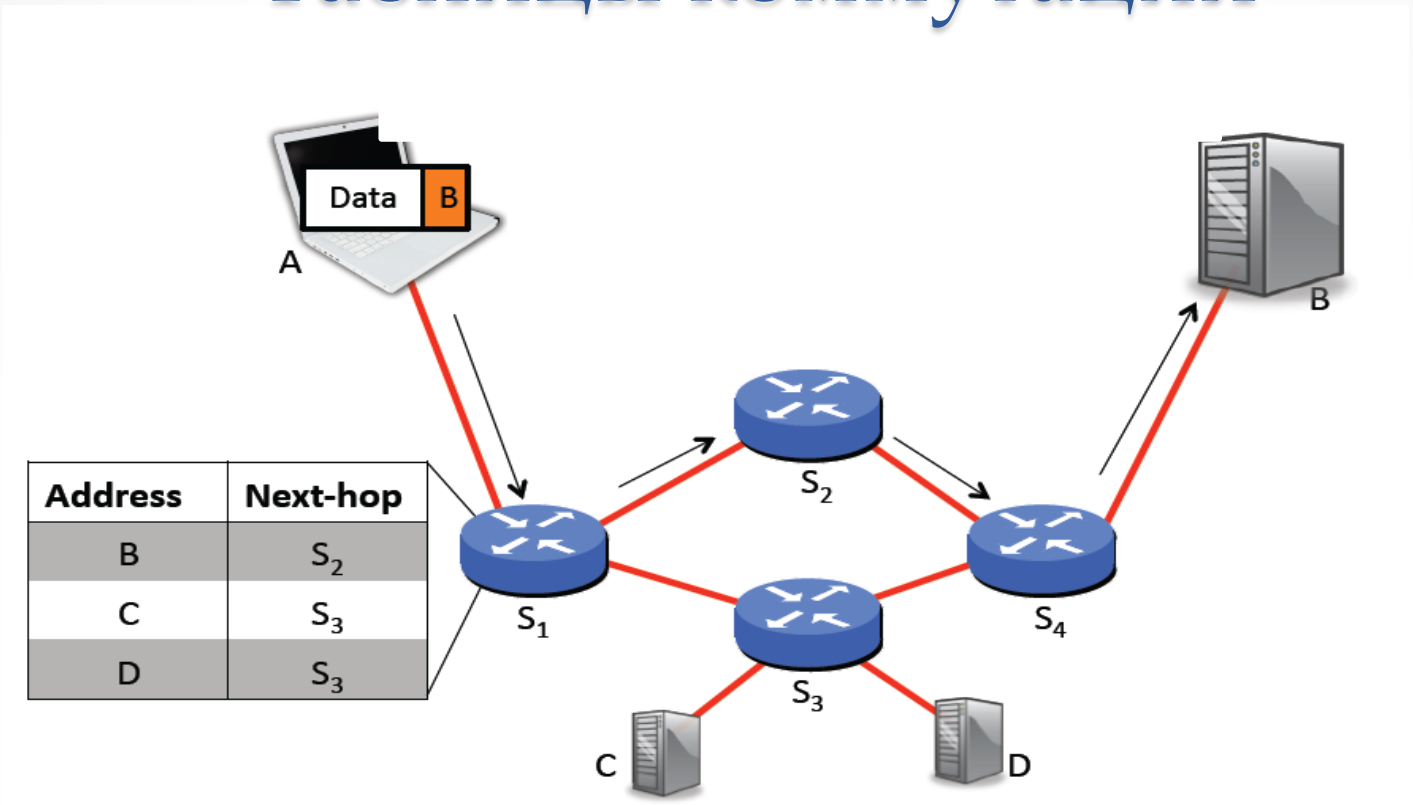
- Не эффективное использование линий
- Нагрузка на сеть
- Пакеты могут зацикливаться
- Как отличить оригинал от дубля?
- Используется когда топология не известна (или ей нельзя доверять)

Маршрутизация от источника



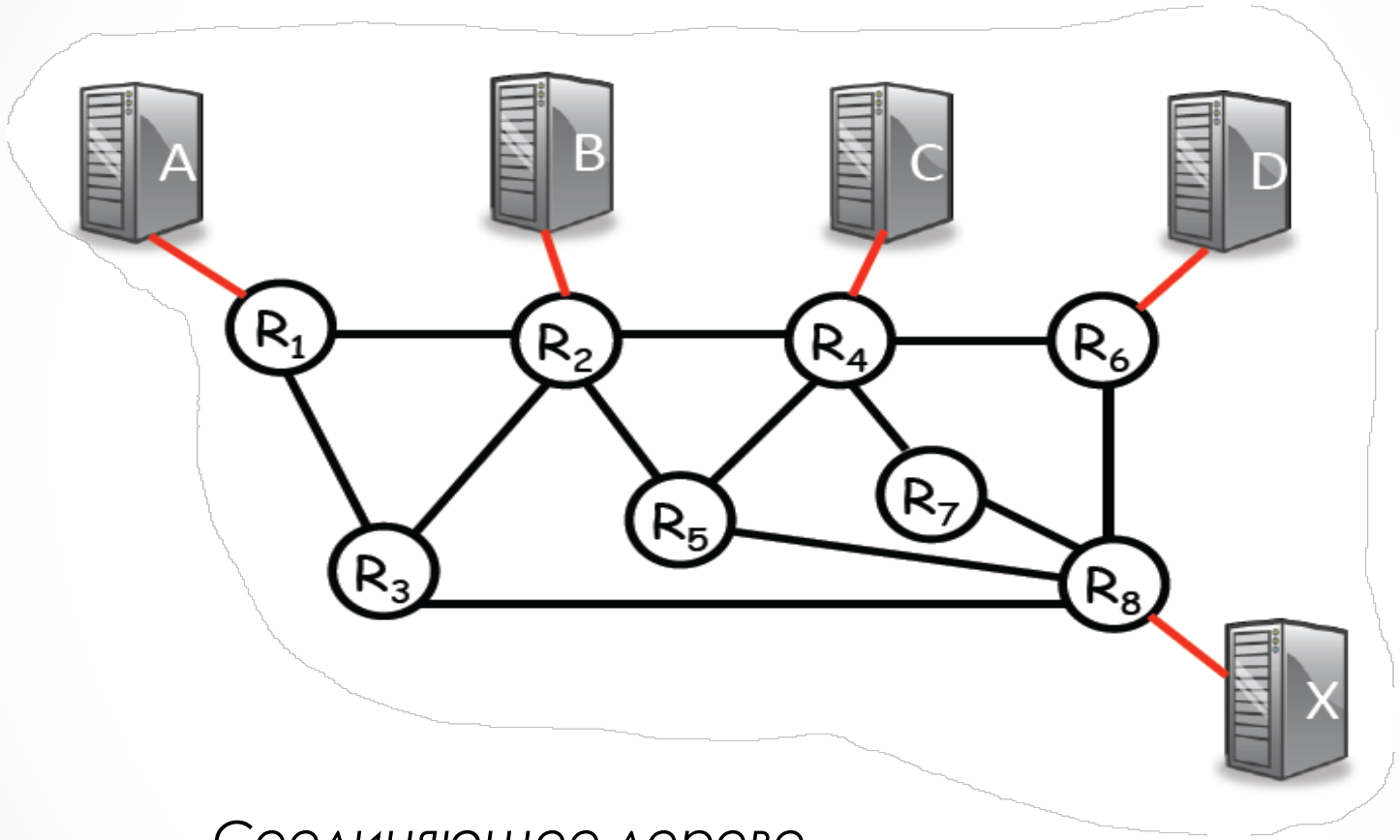
- Не требует поддержки от сети (маршрутизаторы только коммутируют)
- Пакеты содержать списки адресов, переменной длины (могут быть очень длинными)
- Выбор маршрута на конечном хосте, который должен знать топологию сети
- Используется когда пользователь хочет сам управлять маршрутизацией

Таблицы коммутации



- Оптимизация: сеть маршрутизирует по скачкам
- У каждого коммутатора должна быть своя таблица (необходимо много таблиц)
- Состояния от места назначения, а не от потока
- Как поддерживать таблицу в актуальном состоянии

Соединяющие деревья (Spanning tree)

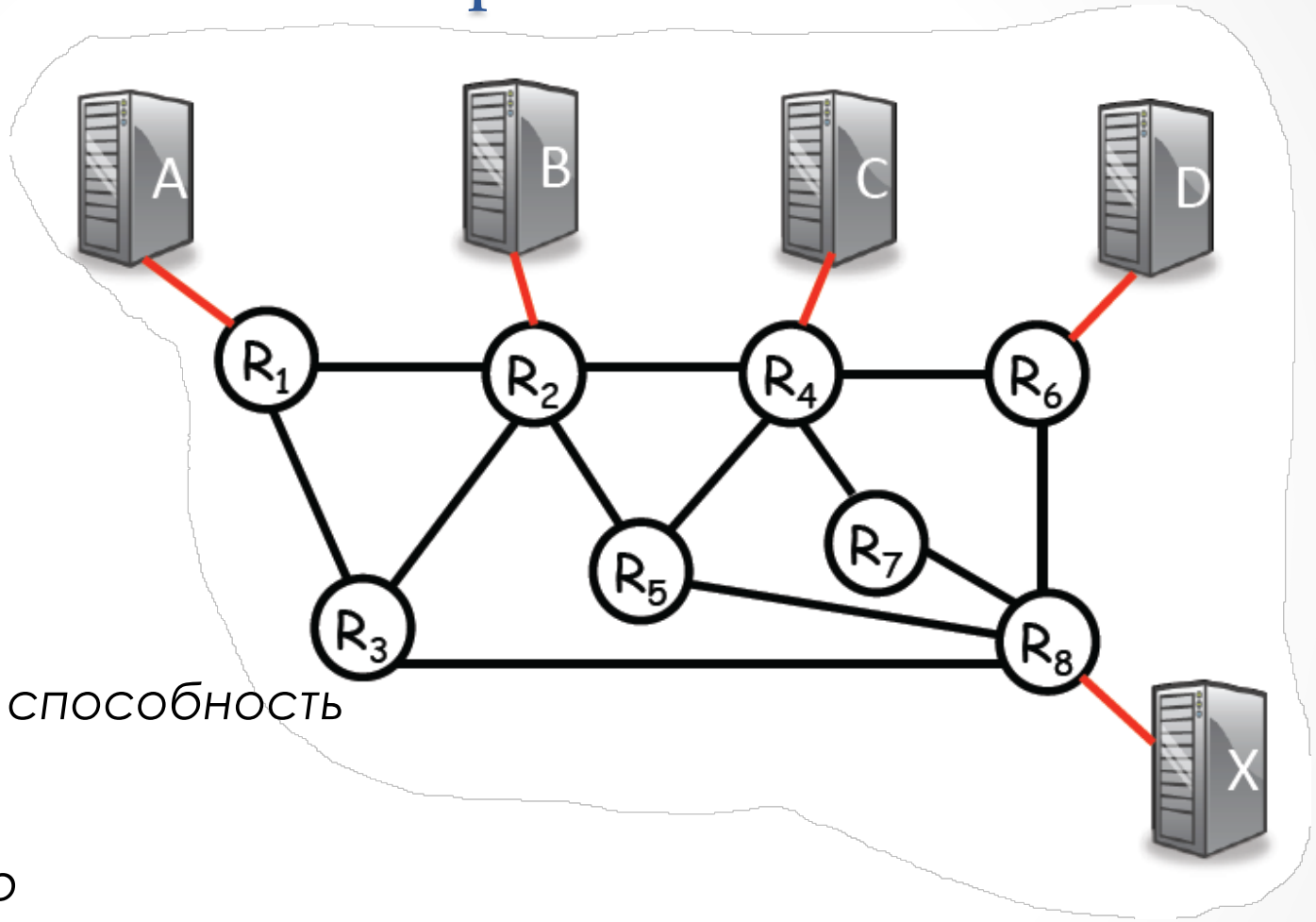


Соединяющее дерево

соединяющее – все листья достижимы

дерево – нет циклов

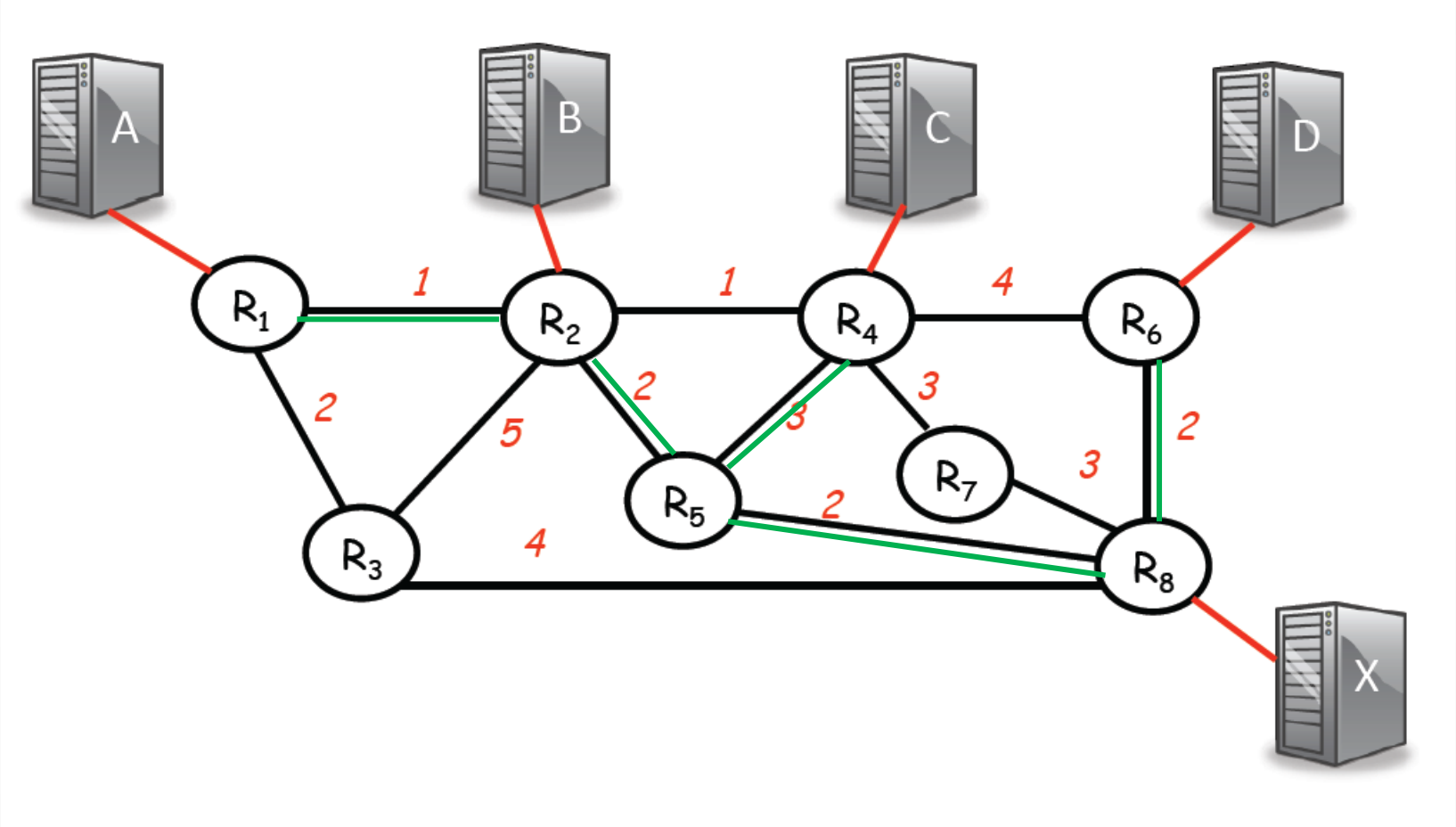
Метрики



Метрики:

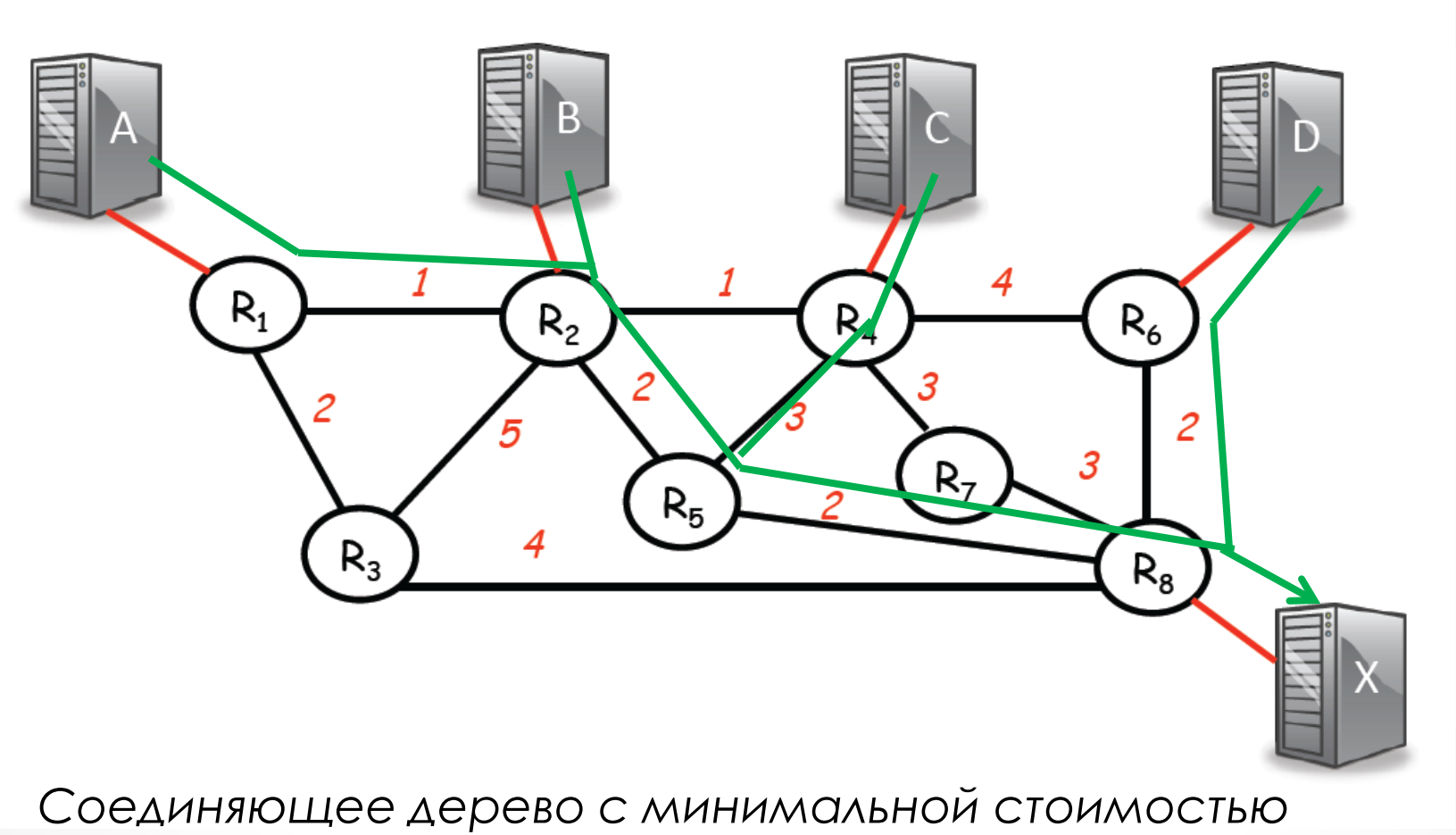
- *мин. расстояние*
- *мин. скачки*
- *мин. задержка*
- *макс. пропускная способность*
- *мин. загруженный*
- *макс. надежный*
- *с мин. стоимостью*
- *макс. безопасный*
- *...*

Пример взвешенного графа



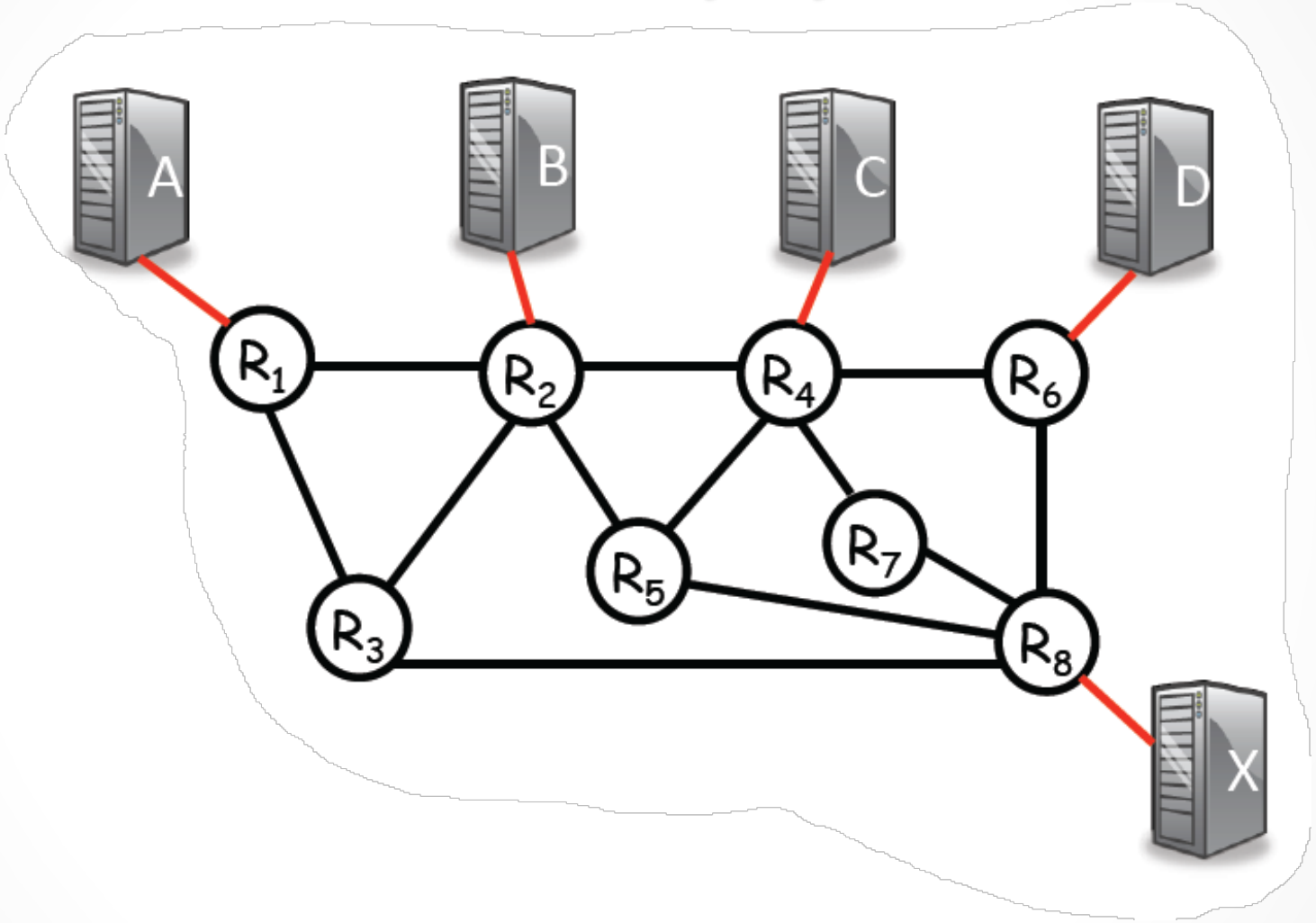
Понятие взвешенного графа

Соединяющее дерево: корень отправитель, листья – все достижимые хосты.

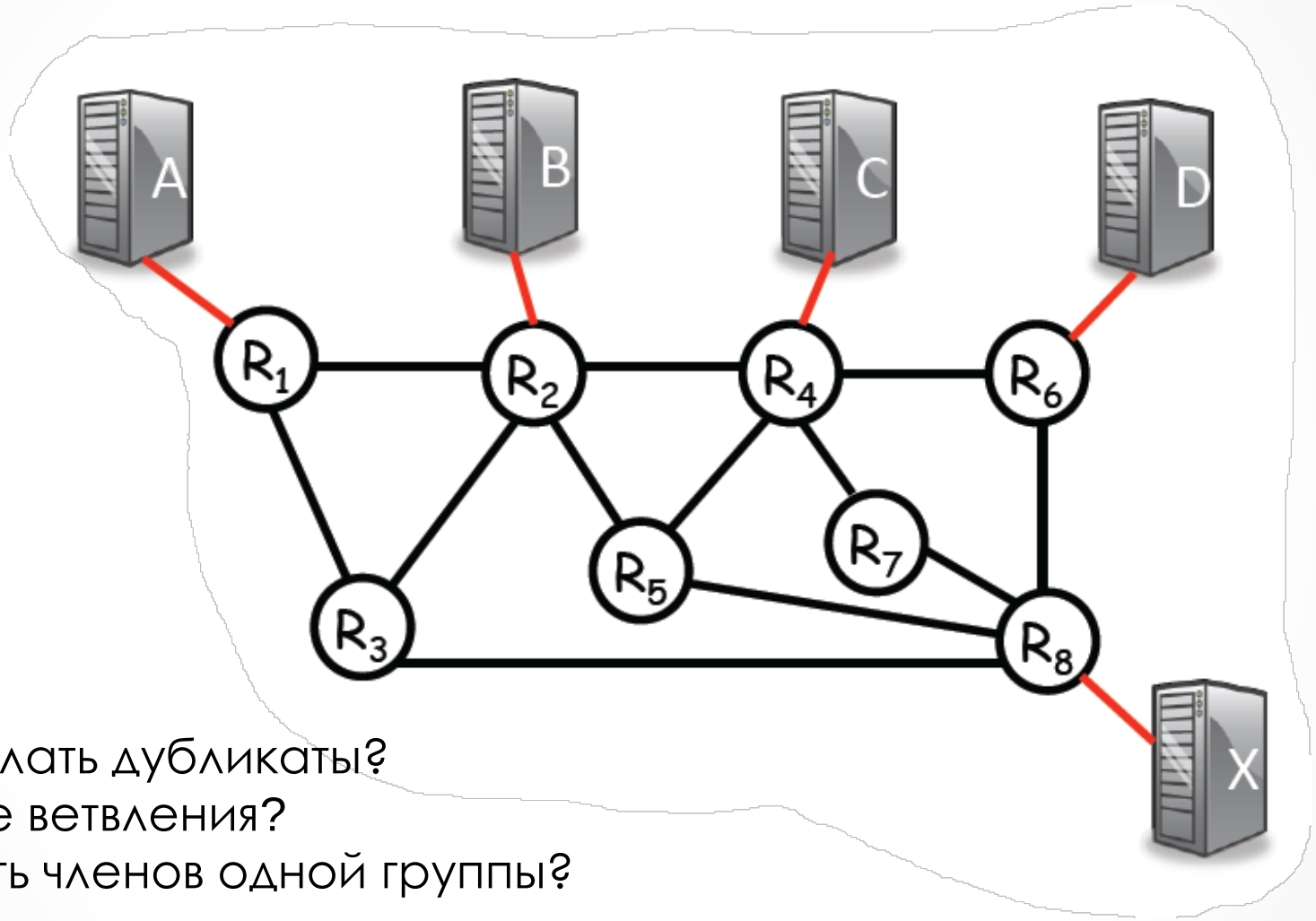


Соединяющее дерево с минимальной стоимостью

По множеству путей

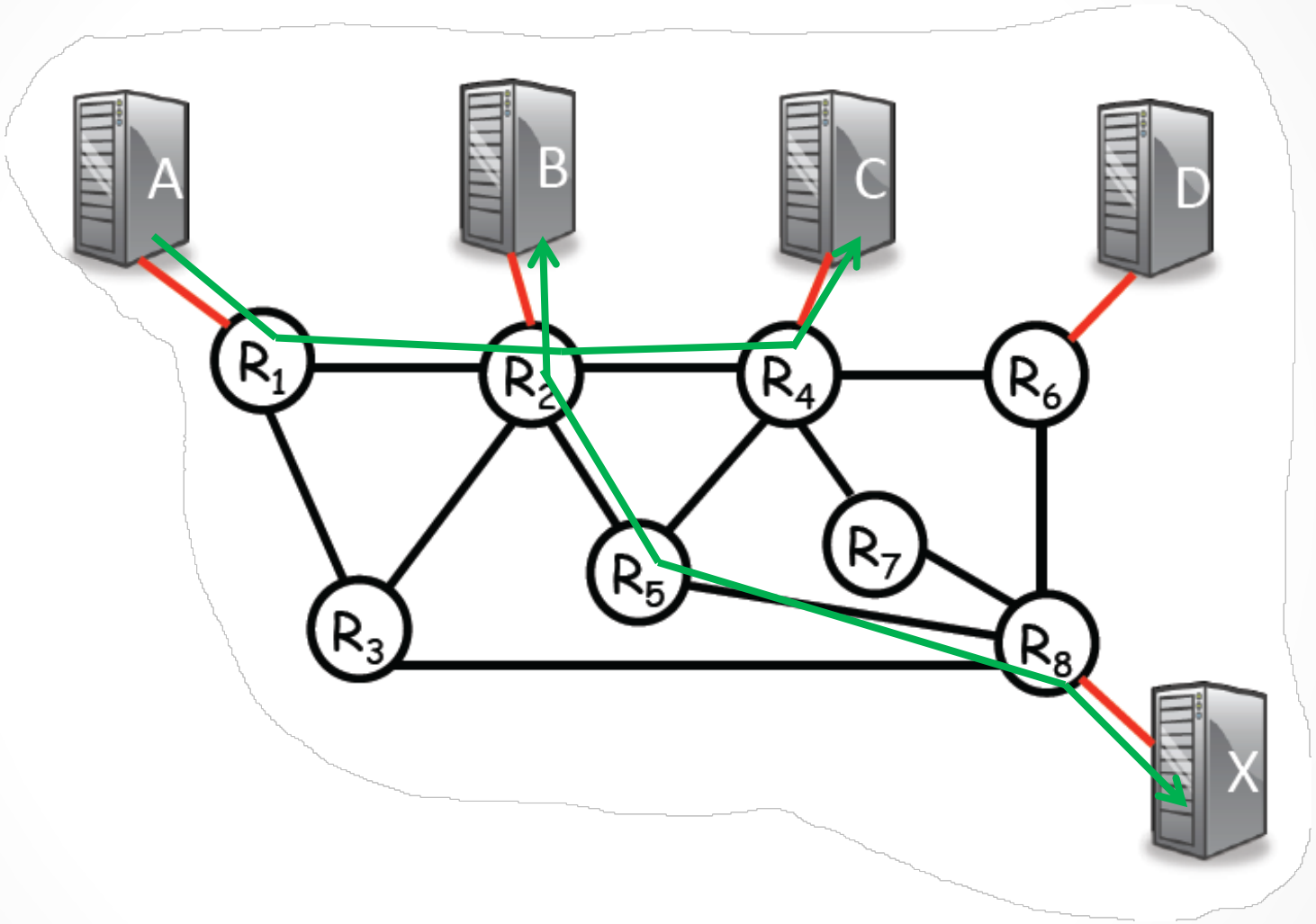


Групповая



- Кто должен делать дубликаты?
- В каждой точке ветвления?
- Как определить членов одной группы?

Групповая



Маршрутизация по вектору расстояния: алгоритм Белмана-Форда

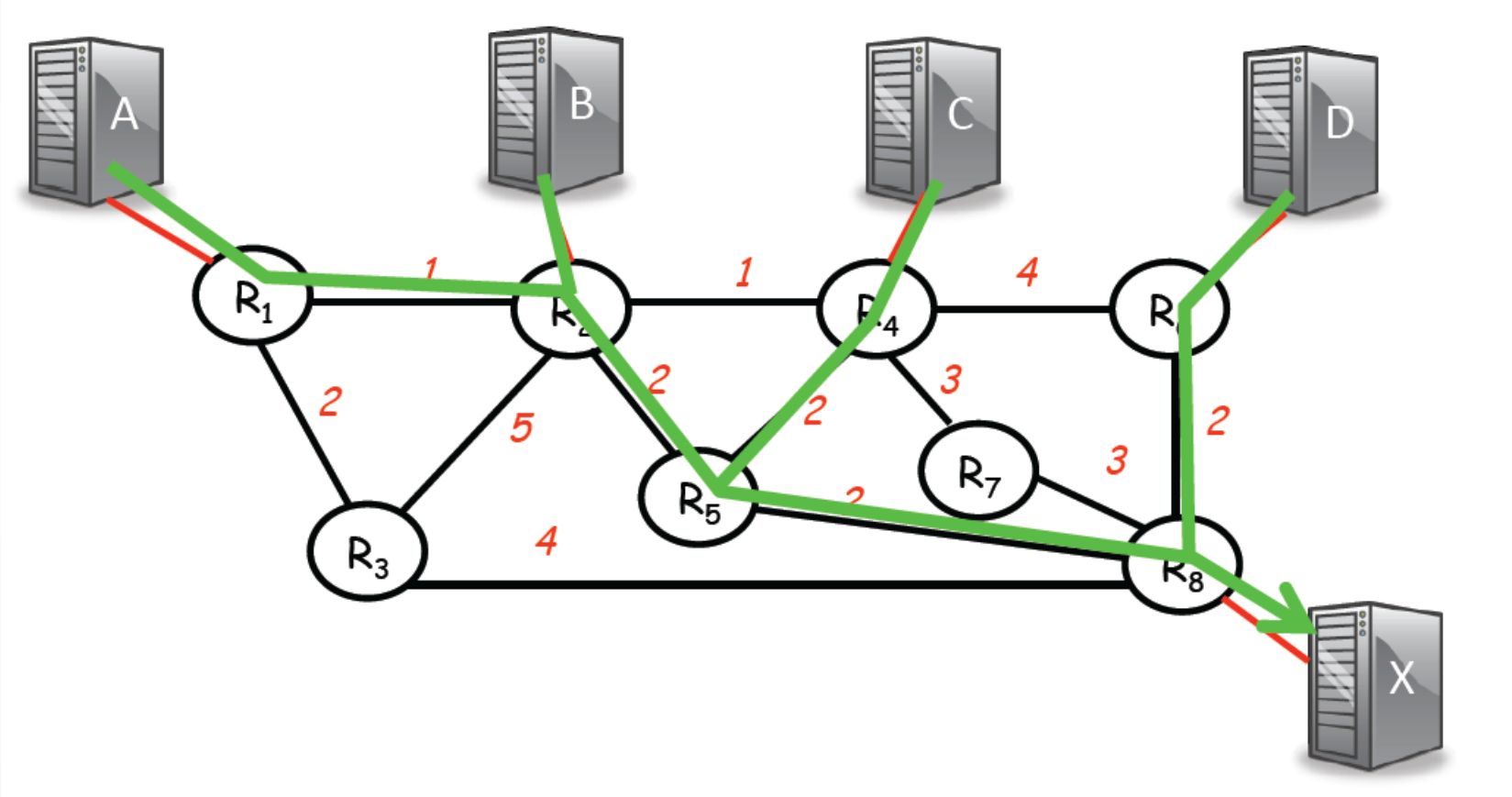
(том 2 стр.38-41)

Введение в компьютерные сети

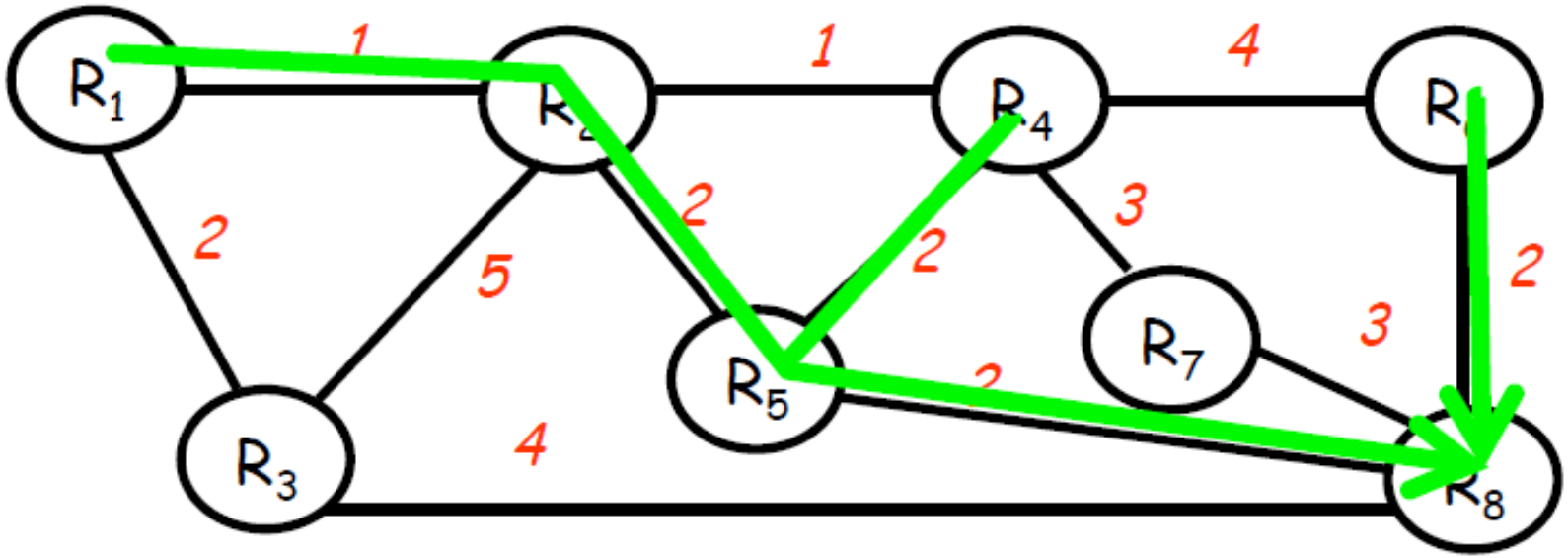
проф. Смелянский Р.Л.
Лаборатория Вычислительных комплексов
ф-т ВМК МГУ

Проблема

Как маршрутизаторы могут совместно найти соединяющее дерево минимальной стоимости?



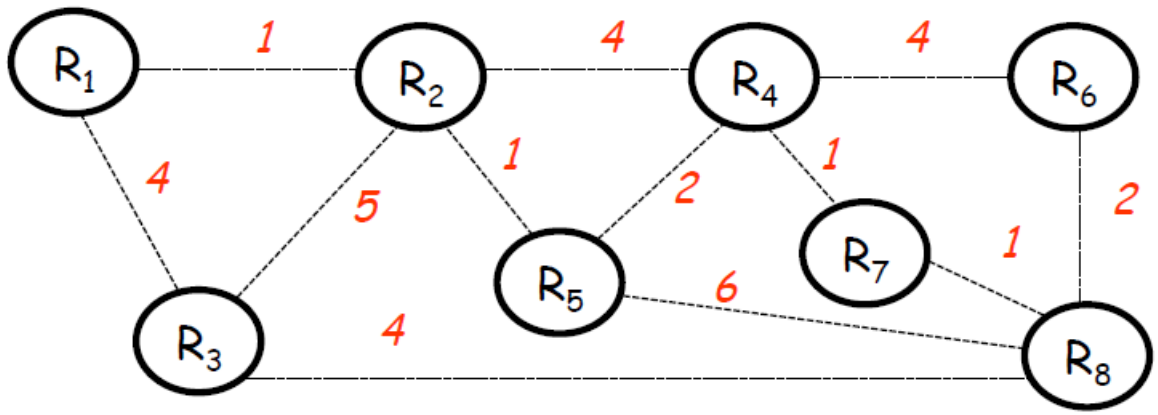
Эквивалентно нахождению соединяющего дерева минимальной стоимости только среди маршрутизаторов



Распределенный алгоритм Беллмана-Форда (т.2 стр.38-48)

- *Пусть каждый маршрутизатор знает стоимость линии к каждому своему соседу*
- *Маршрутизатор R_8 рассчитывает стоимость C_i для достижения каждого известного ему R_i*
- *Вектор $\underline{C_8} = (C_1, C_2, \ldots, C_7)$ – вектор расстояния до R_8*
- *Изначально $\underline{C} = (\infty, \infty, \ldots, \infty)$*
  1. *Каждые T секунд, R_i шлет C_i всем своим соседям*
  2. *Если R_i нашел более дешевый пусть, то он обновляет C_i у всех своих соседей*
  3. *Вернуться к 1*

Пример



R_1	∞
R_2	∞
R_3	∞
R_4	∞
R_5	∞
R_6	∞
R_7	∞

шаг 0

R_1	∞
R_2	∞
R_3	**4**
R_4	∞
R_5	**6**
R_6	**2**
R_7	**1**

шаг 1

R_1	$8,R_3$
R_2	$7,R_5$
R_3	4
R_4	$2,R_7$
R_5	6
R_6	2
R_7	1

шаг 2

R_1	$8,R_3$
R_2	$6,R_4$
R_3	4
R_4	$2,R_7$
R_5	$4,R_4$
R_6	2
R_7	1

шаг 3

R_1	$7,R_2$
R_2	$5,R_7$
R_3	4
R_4	$2,R_7$
R_5	$4,R_4$
R_6	2
R_7	1

шаг 4

R_1	$6,R_4$
R_2	$5,R_7$
R_3	4
R_4	$2,R_7$
R_5	$4,R_4$
R_6	2
R_7	1

шаг 5

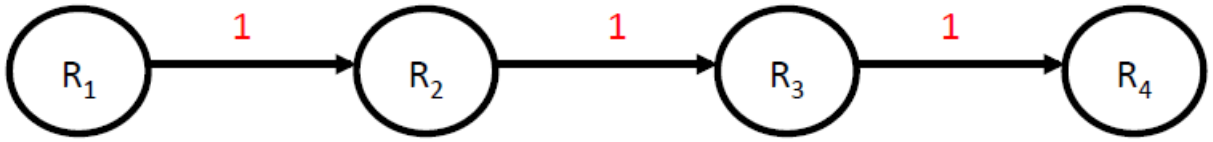
Алгоритм Беллмана-Форда

Вопросы:

1. *Каково максимальное время работы алгоритма?*
2. *Всегда ли алгоритм будет сходиться?*
3. *Что будет если измениться стоимость линии, или отключится маршрутизатор/линия?*

Проблемы с алгоритмом Б-Ф

Плохие вести распространяются медленно



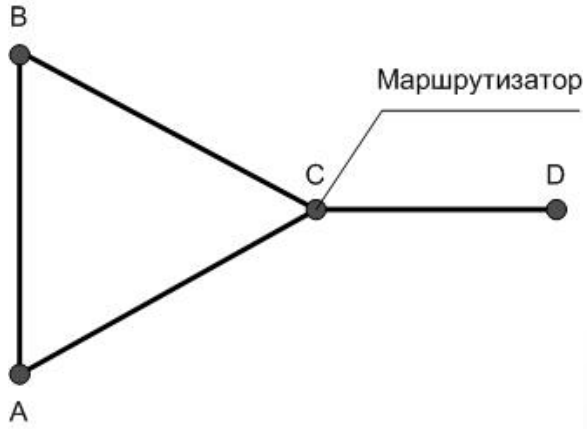
Рассмотрим расчет расстояния для R3 до R4

Time	R_1	R_2	R_3
0	3,R_2	2,R_3	1, R_4
1	3,R_2	2,R_3	3,R_2
2	3,R_2	4,R_3	3,R_2
3	5,R_2	4,R_3	5,R_2
…	*Итак до бесконечности*		…

Линия R3 - R4 не действует

Проблема счетчика до бесконечности

- *Установить ограничение на «бесконечность» (e.g. 16)*
- *Разделение направлений: т.к. R_2 получает данные о маршруте с наименьшей стоимостью от R_3, то запретить R_2 сообщать R_3 о маршрутах, проходящих через R_3*
- *Разделение направлений с бесконечностью: R_2 посылает R_3 ∞*
- *Есть и другие проблемы, связанные с алгоритмом Б-Ф*



Беллман-Форд на практике

- *Алгоритм Беллмана-Форда - пример алгоритма по вектору расстояния*
- *Этот алгоритм использовался в первых Интернет протоколах маршрутизации RIP (Routing Internet Protocol)*
- *Он не требует больших вычислений, распределенный и, в конечном счете, сходится*
- *Со временем он был вытеснен алгоритмами, которые рассчитывали соединяющее дерево для каждого маршрутизатора*

Маршрутизация по состоянию канала: алгоритм наикратчайшего пути Дейкстры

(том 2 стр.33, 41-46)

Введение в компьютерные сети

проф. Смелянский Р.Л.
Лаборатория Вычислительных комплексов
ф-т ВМК МГУ

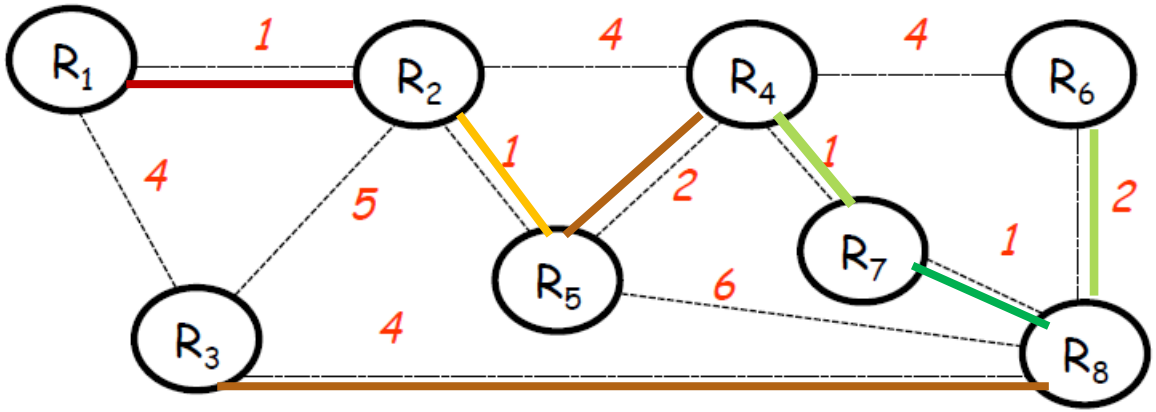
Алгоритм Дейкстры наикратчайшего пути

1. *Определение топологии сети: Маршрутизатор передает лавиной всем другим маршрутизаторам состояния своих линий для рассчета топологии сети*
   - *Периодически*
   - *Когда изменяется состояние линии*

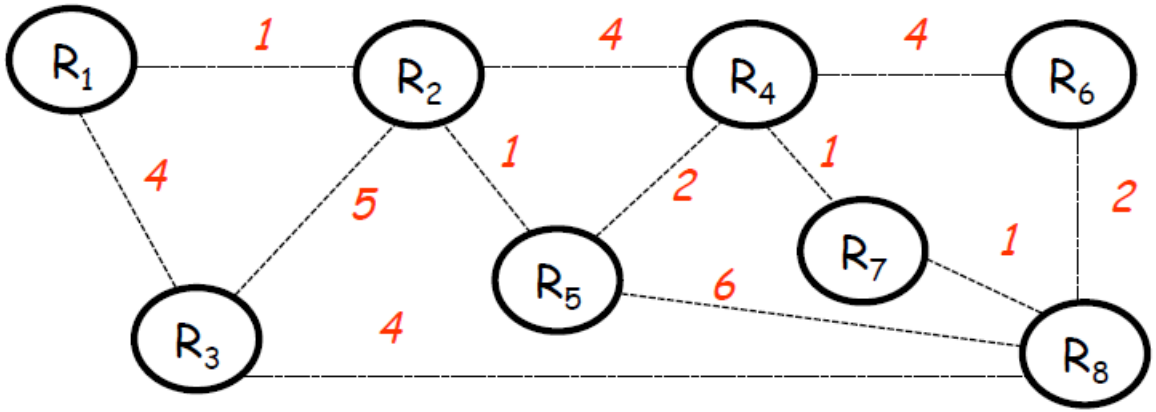
2. *Вычисление по алгоритму Дейкстры: каждый маршрутизатор независимо запускает алгоритм Дейкстры наикратчайшего пути.*

Каждый маршрутизатор находит соединяющее дерево с минимальной стоимостью до каждого другого маршрутизатора

Пример для R8

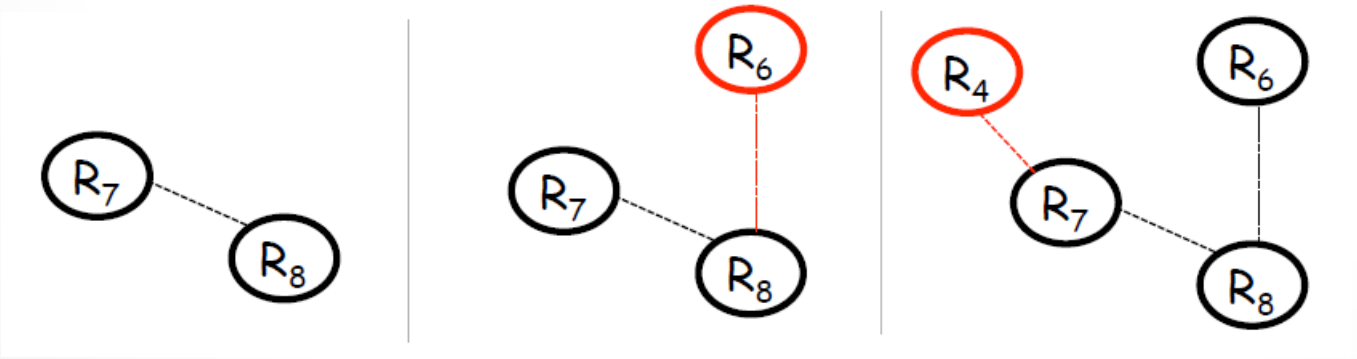


Пример для R8

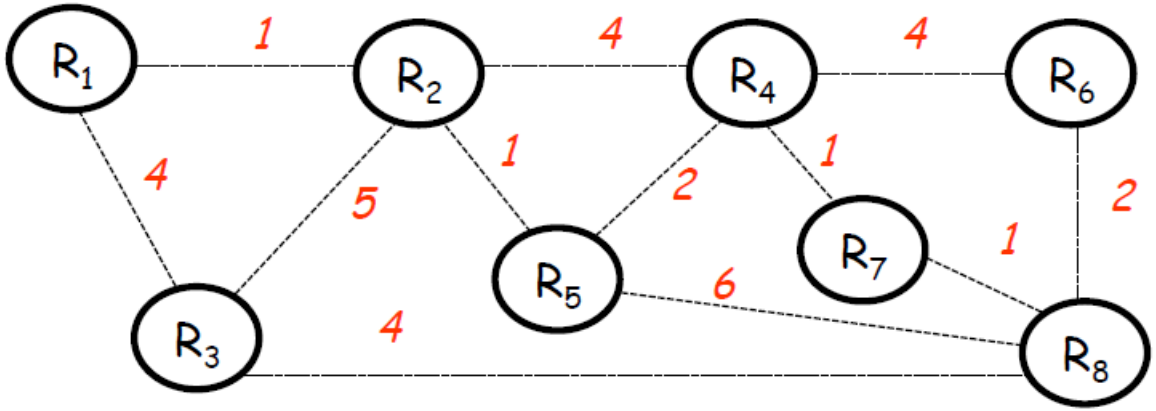


Добавляем путь стоимости

1 2 2

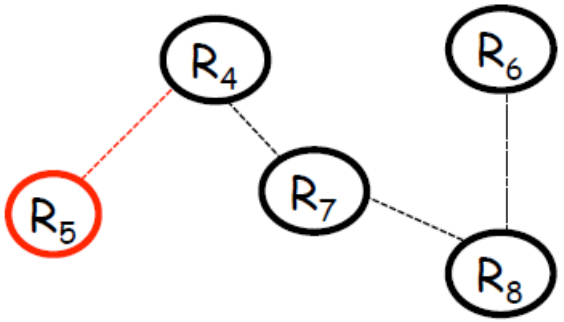


Пример для R8

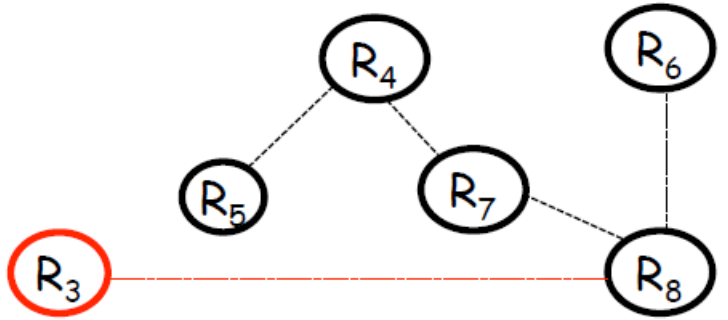


Добавляем путь стоимости

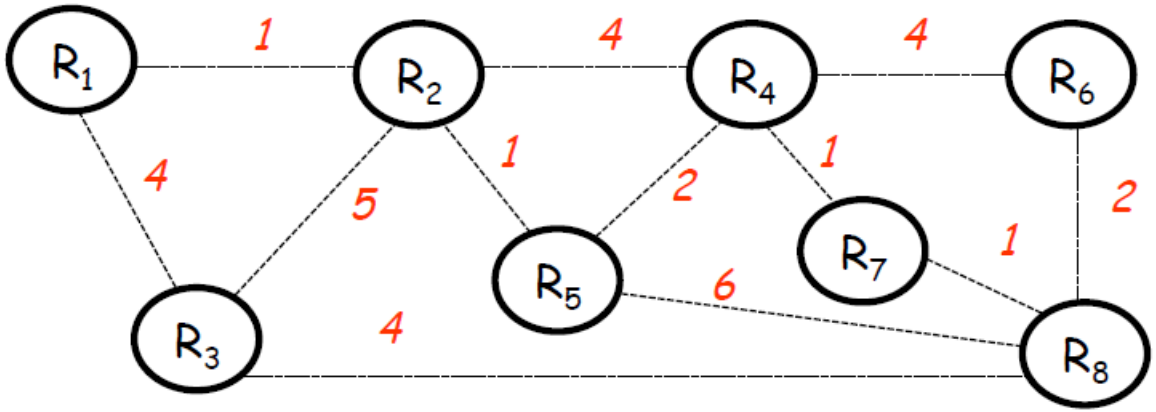
4



4

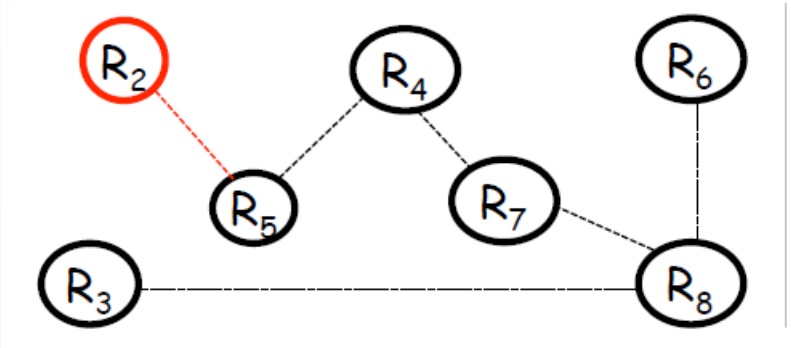


Пример для R8

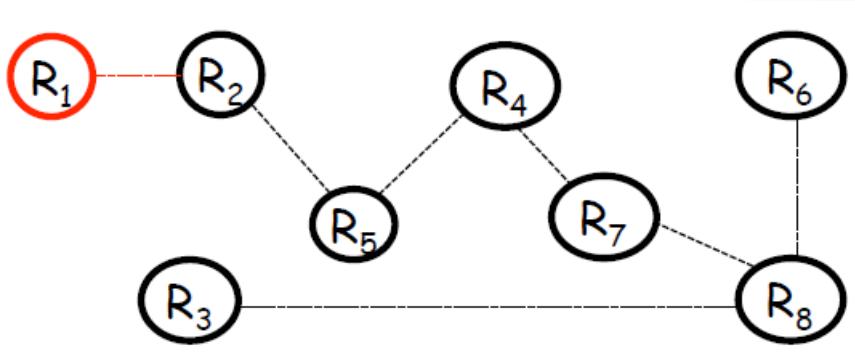


Добавляем путь стоимости

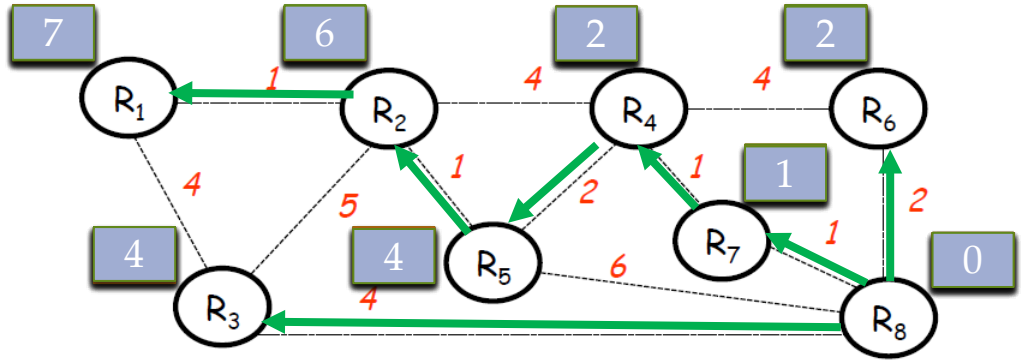
5



6



Алгоритм



	0	1	2	3	4	5	6	7
Пройден		R_8	R_8, R_7	R_8, R_7, R_6	R_8, R_7, R_6, R_4	R_8, R_7, R_6, R_4, R_3	$R_8, R_7, R_6, R_4, R_3, R_5$	$R_8, R_7, R_6, R_4, R_3, R_5, R_2$
Веса	$R_1=\infty$ $R_5=\infty$ $R_2=\infty$ $R_6=\infty$ $R_3=\infty$ $R_7=\infty$ $R_4=\infty$ $R_8=0$	$R_1=\infty$ $R_5=6$ $R_2=\infty$ $R_6=2$ $R_3=4$ $R_7=1$ $R_4=\infty$ $R_8=0$	$R_1=\infty$ $R_5=6$ $R_2=\infty$ $R_6=2$ $R_3=4$ $R_7=1$ $R_4=2$ $R_8=0$	$R_1=\infty$ $R_5=6$ $R_2=\infty$ $R_6=2$ $R_3=4$ $R_7=1$ $R_4=2$ $R_8=0$	$R_1=\infty$ $R_5=4$ $R_2=6$ $R_6=2$ $R_3=4$ $R_7=1$ $R_4=2$ $R_8=0$	$R_1=8$ $R_5=4$ $R_2=6$ $R_6=2$ $R_3=4$ $R_7=1$ $R_4=2$ $R_8=0$	$R_1=8$ $R_4=2$ $R_2=6$ $R_6=2$ $R_5=4$ $R_7=1$ $R_3=4$ $R_8=0$	$R_1=7$ $R_4=2$ $R_2=6$ $R_6=2$ $R_3=4$ $R_7=1$ $R_5=4$ $R_8=0$
Смежны	$R_3=4$ $R_7=1$ $R_5=6$ $R_6=2$	$R_4=2$ $R_3=4$ $R_6=2$ $R_5=6$	$R_4=2<6=>R_4=2$ $R_3=4$ $R_5=6$	$R_3=4$ $R_5=6>4=>R_5=4$ $R_2=\infty>6=>R_2=6$	R1=∞>8=>R1=8 R2=6<9=>R2=6 $R_5=4$	$R_2=6<7=>R_2=6$	$R_1=8>7=>R_1=7$	
Выбираем	$R_8=0$	$R_7=1$	$R_6=2$	$R_4=2$	$R_3=4$	$R_5=4$	$R_2=6$	Стоп. Вершин нет

Алгоритм Дейкстры

```
function ShortestPath_Dijkstra(G, src, d, prev)
    // Input: G = (V, E), src, dst
    // Output: d[1:n], prev[1:n]
    //   prev[i] - узел, предшествующий i в пути

    // Помещаем вершины в очередь с приоритетом
    for each i in V \ {src} do
        d[i] = Infinity
        prev[i] = -1
        PriorityQueueInsert(Q, i, d[i])
    end for

    d[src] = 0
    prev[src] = -1
    PriorityQueueInsert(Q, src, d[src])
```

Алгоритм Дейкстры

```
    for i = 0 to n - 1 do
        // Извлекаем узел ближайший к начальному
        v = PriorityQueueRemoveMin(Q)
        // Отмечаем v как посещенный
        H = H + {v}

        // Цикл по смежным вершинам узла v
        for each u in Adj(v) \ H do
            // Путь через u короче текущего пути?
            if d[v] + w(v, u) < d[u] then
                d[u] = d[v] + w(v, u)
                PriorityQueueDecrease(Q, u, d[u])
                prev[u] = v
            end if
        end for
    end for
end function
```

Алгоритм Дейкстры

Вопросы:

- *Сколько времени работает этот алгоритм?*
- *Что происходит когда изменяется стоимость линии или когда маршрутизатор/линия выходят из строя?*

Сложность алгоритма Дейкстры

Вариант реализации алгоритма Дейкстры	**Насыщенный граф** $m = O(n^2)$	**Разреженный граф** $m = O(n)$
Вариант 1 *D*[i] – это массив (поиск за время $O(n)$)	$T = O(n^2 + m) = \boldsymbol{O(n^2)}$	$T = O(n^2 + m) = \boldsymbol{O(n^2)}$
Вариант 2 *D*[i] хранятся в бинарной куче	$T = O(n \log n + m \log n) = \boldsymbol{O(n^2 \log n)}$	$T = O(n \log n + m \log n) = \boldsymbol{O(n \log n)}$
Вариант 3 *D*[i] хранятся в Фибоначчиевой куче	$T = O(m + n \log n) = \boldsymbol{O(n^2)}$	$T = O(n + n \log n) = \boldsymbol{O(n \log n)}$

Алгоритм Дейкстры на практике

- *Алгоритм Дейкстры – это пример алгоритма по состоянию канала*
  - *состояние линии знает каждый маршрутизатор*
  - *каждый маршрутизатор строит соединяющее дерево минимальной стоимости до каждого другого маршрутизатора*
- *Этот алгоритм является основой OSPF (Open Shortest Path First) - широко используемого протокола маршрутизации (том 2 стр.82-86)*

Постановки задач о кратчайшем пути

- **Задача о кратчайшем пути между парой вершин (single-pair shortest path problem)**
  Требуется найти кратчайший путь из заданной вершины *s* в заданную вершину *d*
- **Задача о кратчайших путях из заданной вершины во все (single-source shortest path problem)**
  Найти кратчайшие пути из заданной вершины *s* во все
- **Задача о кратчайшем пути в заданный пункт назначения (single-destination shortest path problem)**
  Требуется найти кратчайшие пути в заданную вершину *v* из всех вершин графа
- **Задача о кратчайшем пути между всеми парами вершин (all-pairs shortest path problem)**
  Требуется найти кратчайший путь из каждой вершины *u* в каждую вершину *v*

Алгоритмы поиска кратчайшего пути в графе

Алгоритм	Применение
Алгоритм Дейкстры	Находит кратчайший путь от одной из вершин графа до всех остальных. Алгоритм работает только для графов без ребер отрицательного веса ($w_{ij} \geq 0$)
Алгоритм Беллмана-Форда	Находит кратчайшие пути от одной вершины графа до всех остальных во взвешенном графе. Вес ребер может быть отрицательным
Алгоритм поиска A* (A star)	Находит путь с наименьшей стоимостью от одной вершины к другой, используя алгоритм поиска по первому наилучшему совпадению на графе
Алгоритм Флойда-Уоршелла	Находит кратчайшие пути между всеми вершинами взвешенного ориентированного графа
Алгоритм Джонсона	Находит кратчайшие пути между всеми парами вершин взвешенного ориентированного графа (должны отсутствовать циклы с отрицательным весом)
Алгоритм Ли (волновой алгоритм)	Находит путь между вершинами *s* и *t* графа, содержащий минимальное количество промежуточных вершин (трассировки электрических соединений на кристаллах микросхем и на печатных платах)
Алгоритмы Viterbi, Cherkassky, …	

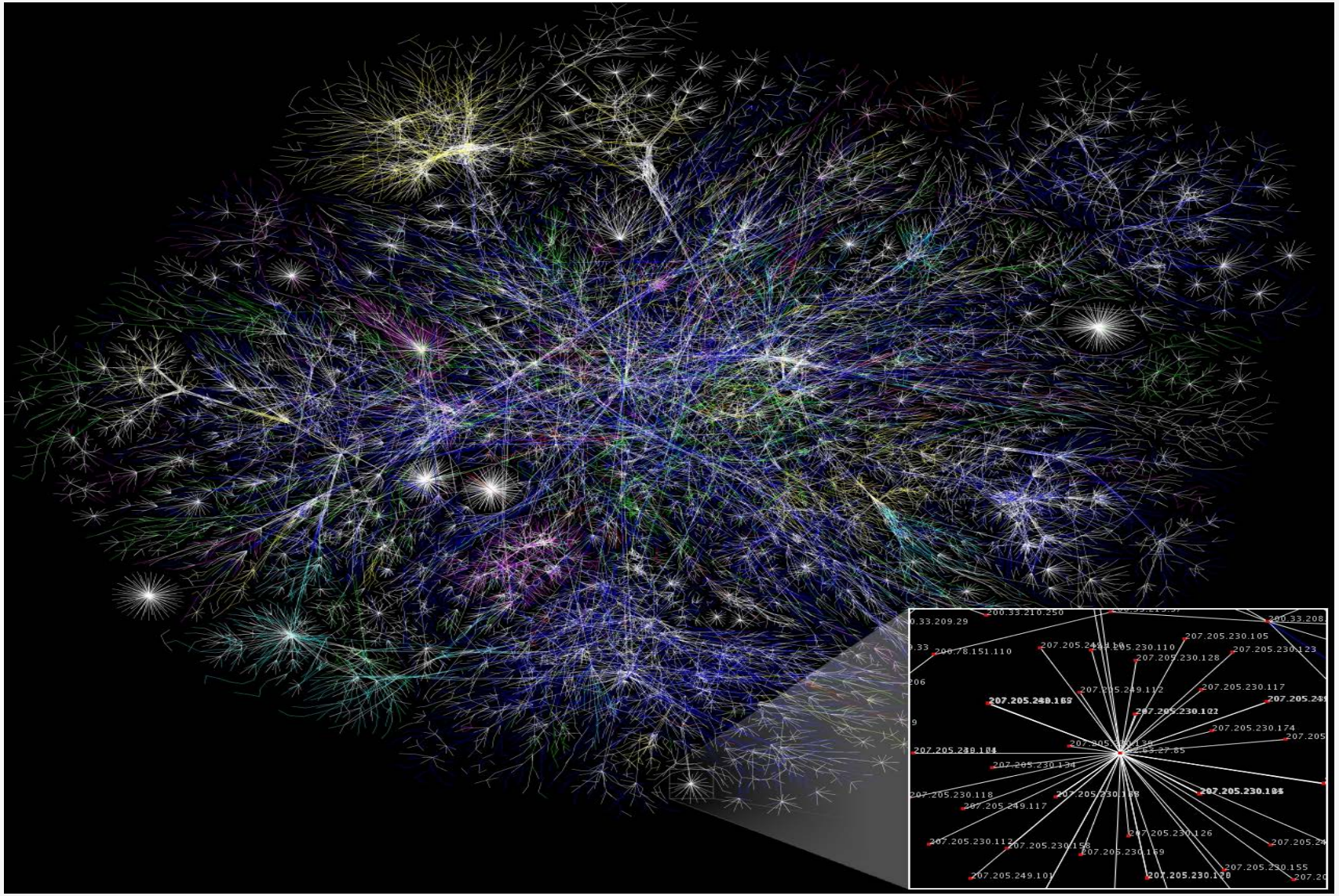
Маршрутизация в Интернет

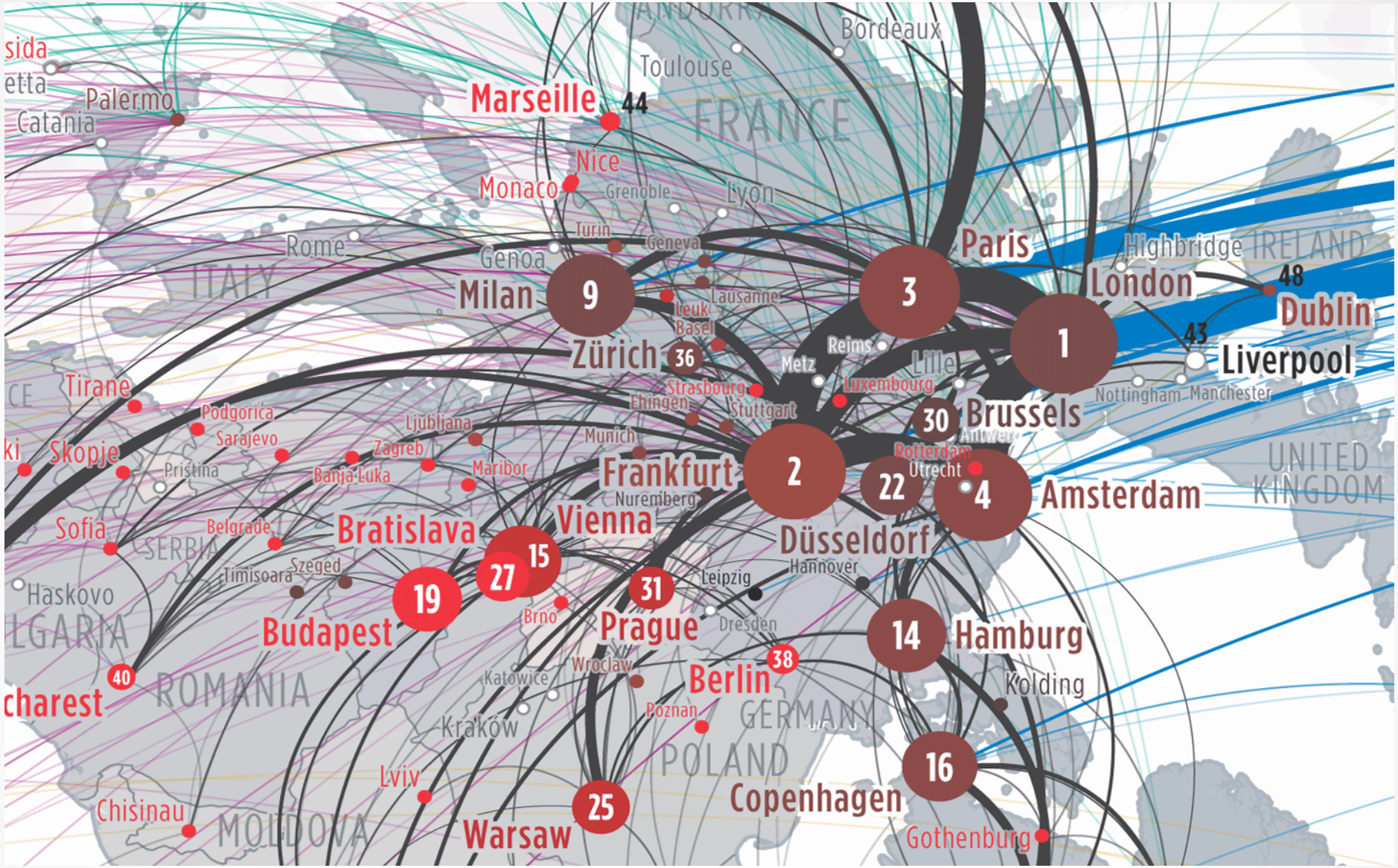
(том 2 стр.82-88)

Введение в компьютерные сети

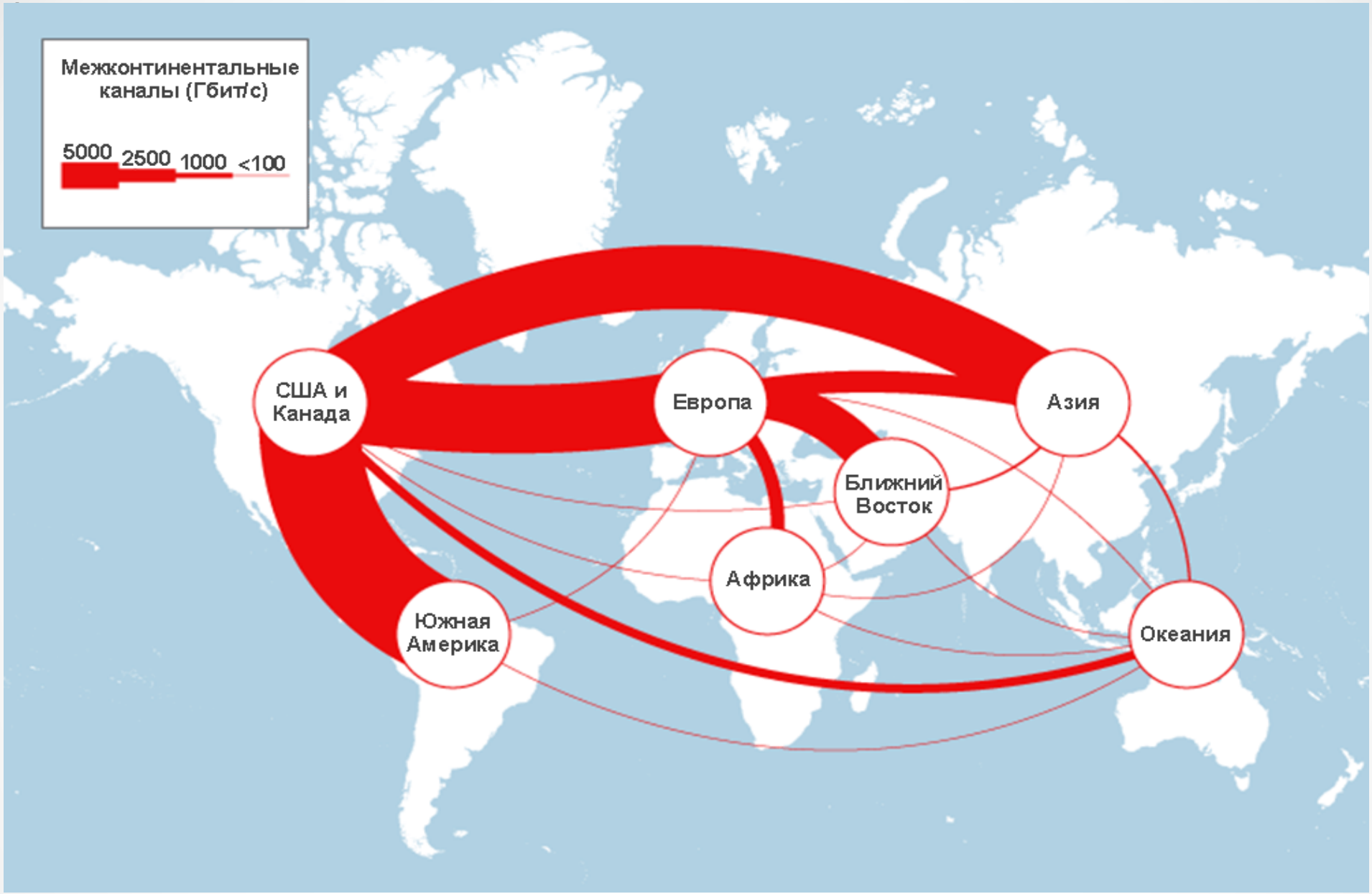
проф. Смелянский Р.Л.
Лаборатория Вычислительных комплексов
ф-т ВМК МГУ

Как быть с такой сетью ?





Если магистральные каналы связи сравнить с кровеносной системой современной цивилизации, то Европа — её сердце.


Межконтинентальные каналы (Гбит/с)
5000
2500
1000
<100
США и Канада
Европа
Азия
Ближний Восток
Африка
Южная Америка
Океания

Карта подводных сетевых коммуникаций

Автономные Системы

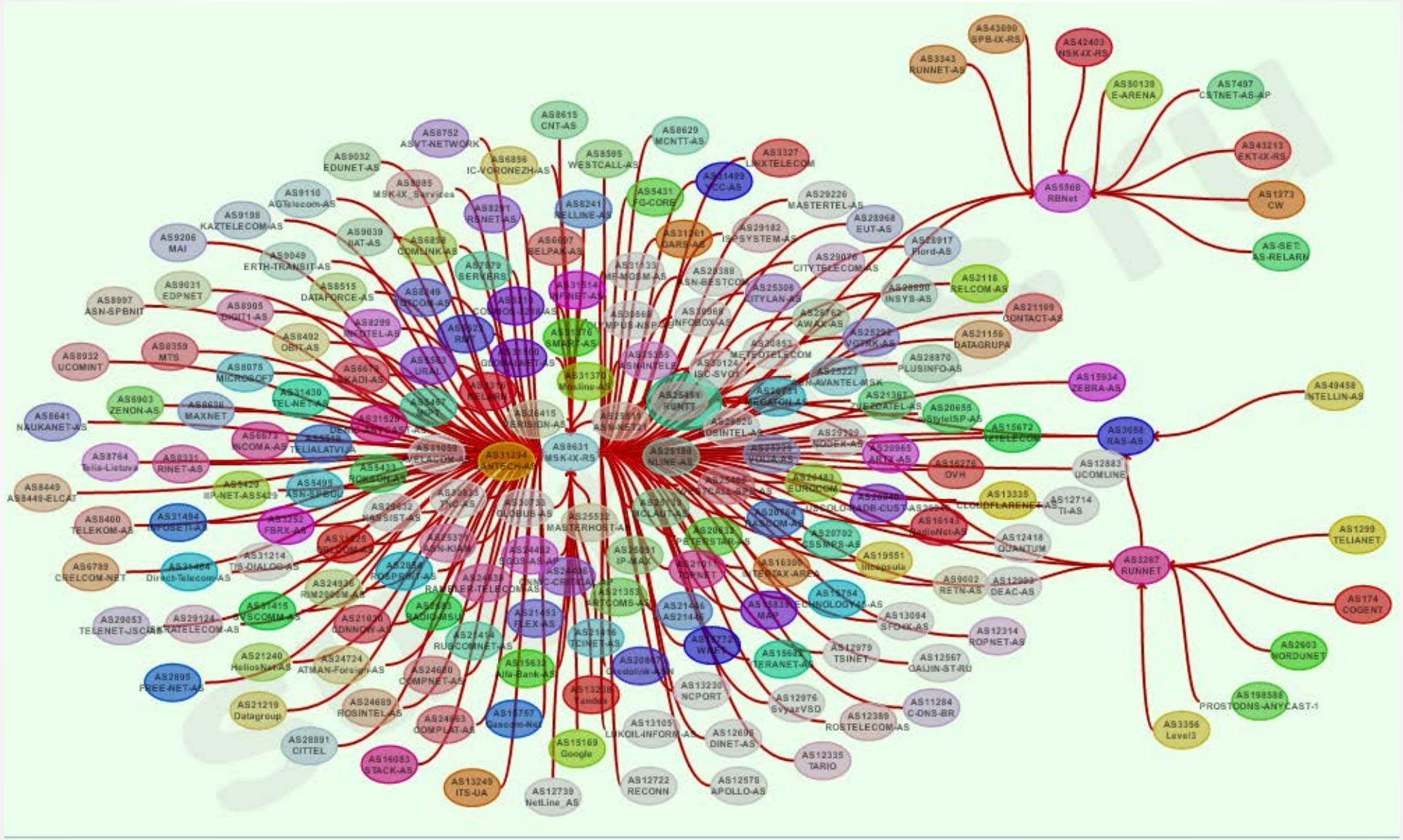
Единица иерархии в Интернет

- Внутри АС ее владелец решает как маршрутизировать потоки данных
- Между АС должен использоваться протокол BGP-4 (Border Gateway Protocol v4 RFC 1771)

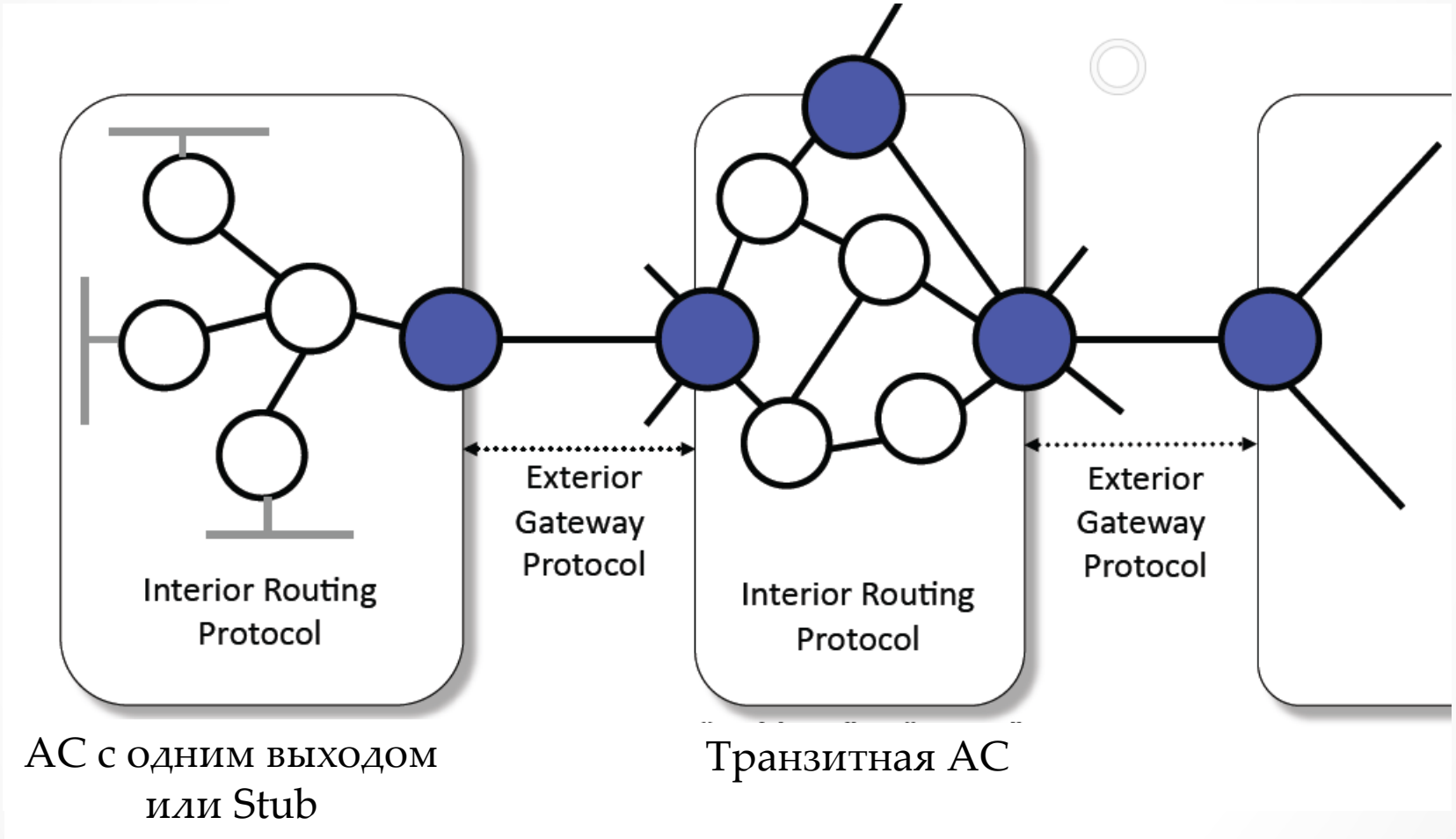
Как найти номер АС?

http://whatismyipaddress.com/ip-lookup

Связность автономных систем



Иерархическая структура Интернет



АС с одним выходом или Stub

Транзитная АС

Протоколы внутренней маршрутизации

RIP (RFC 2453)
(www.rfc.com.ru, http://www.ietf.org/rfc.html)

- используют алгоритм по вектору расстояния (алгоритм Б-Ф)
- обновление векторов каждые 30 секунд
- аутентификация при обновлениях не применяется
- изначально был использован в BSD Unix
- сегодня применяется редко

OSPF (RFC 2328)

- изменения состояний линии рассылаются лавиной по необходимости
- каждый маршрутизатор использует алгоритм Дейкстры
- изменения аутентифицируются
- АС можно разбивать на области
- Широко используется, сложный аналог

IS-IS (RFC 1142), который также широко используется

Маршрутизация через одну точку выхода

В АС выделяют одну точку выхода, так что маршрутизаторы внутри АС могут использовать маршрутизацию по умолчанию

- *Каждый маршрутизатор знает все префиксы внутри АС*
- *Пакеты для других АС пересылаются на маршрутизатор-выход по умолчанию*
- *Маршрутизатор-выход по умолчанию – пограничный шлюз для других АС*

Таблицы маршрутизации в АС с одним выходом (маршрутизатор-выход) по умолчанию, как правило, имеют не большой размер

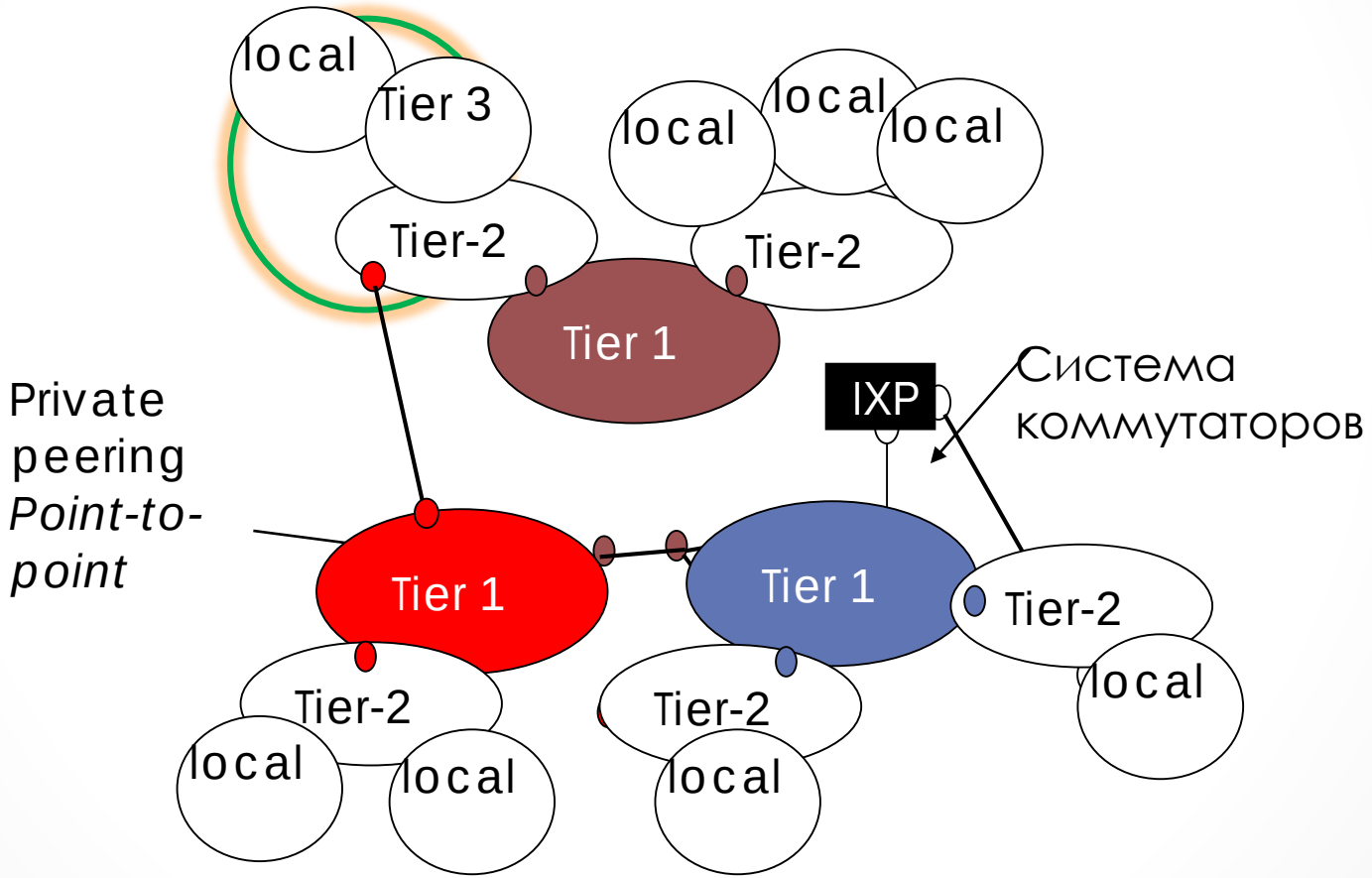
Маршрутизация через несколько точек выхода

- *Используется в транзитных АС и сетях международных компаний с разветвленной сетью офисов*
- *Каждому внутреннему маршрутизатору должно быть сообщено какую точку выхода он должен использовать для определенного префикса точки назначения*
- *Таблица маршрутизации существенно разрастается*
- *Два подхода:*
  - *«горячая картошка» – переслать ближайшему выходу*
  - *выбрать выход ближе всего к токе назначения*

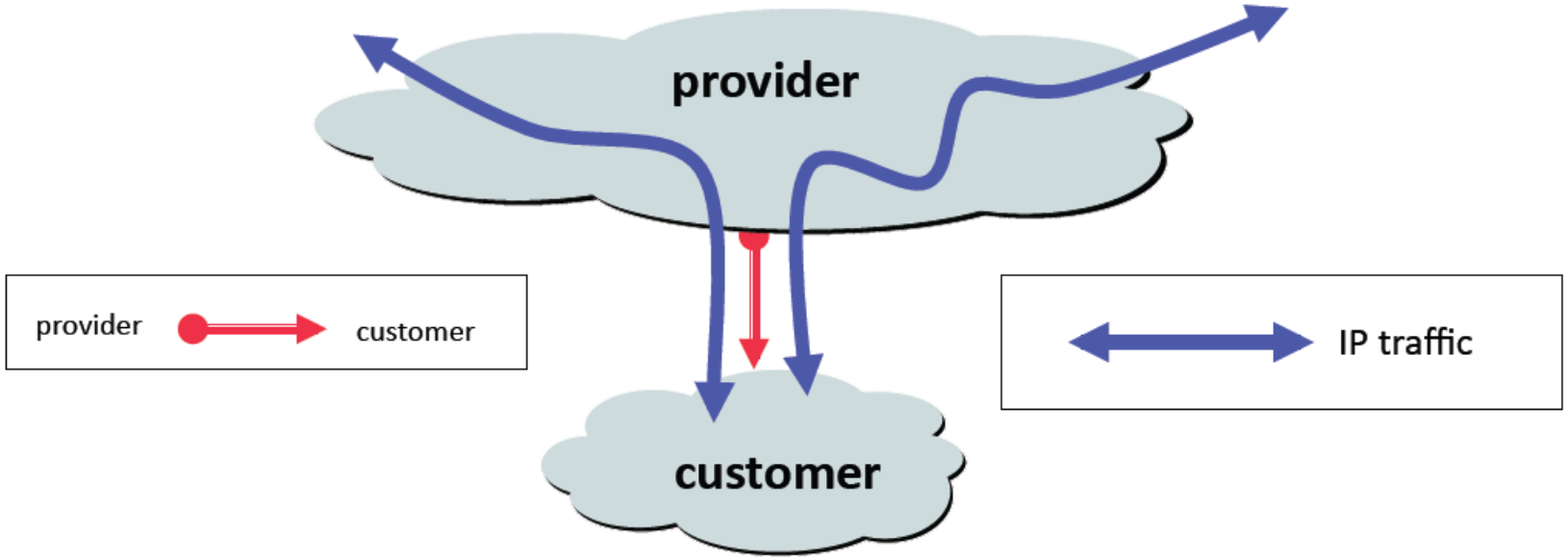
Протокол внешней маршрутизации

- *Все АС взаимодействуют, используя протокол BGP-4*
- *BGP-4 был разработан чтобы решить следующие проблемы:*
  - *Топология: Интернет плохо структурированная смесь разнообразных АС*
  - *Автономия АС: каждая АС по-своему определяет стоимость линии, поэтому невозможно построить путь с наименьшей стоимостью*
  - *Доверие: некоторые АС не могут доверять тем маршрутам, которые предлагают другие АС (два конкурирующих провайдера, защита конфиденциальности через территорию неприятеля)*
  - *Политика: разные АС преследуют разные цели (мин. число скачков vs предпочтение одного провайдера перед другими)*

Структура Интернет

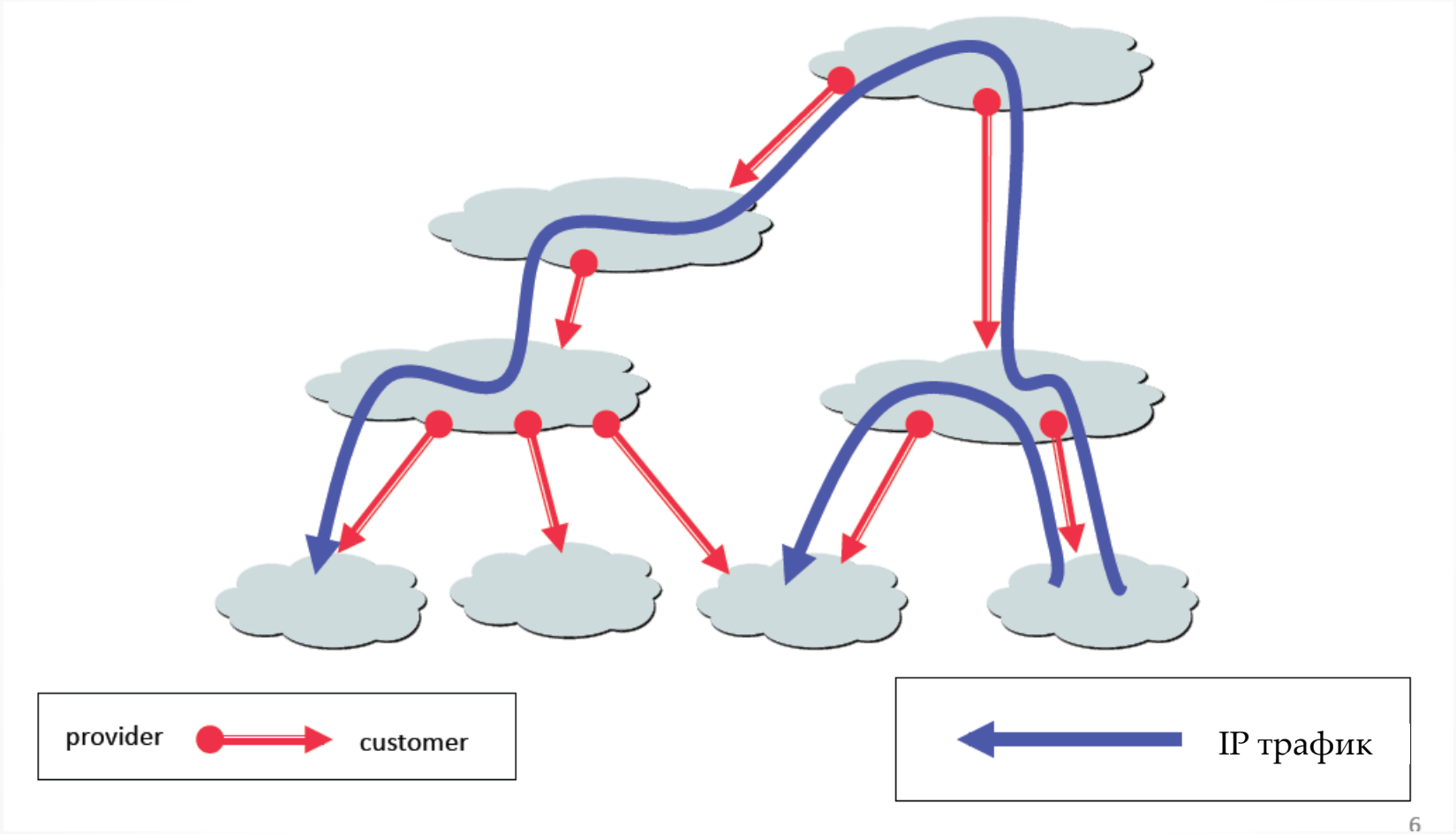


Заказчики и провайдеры

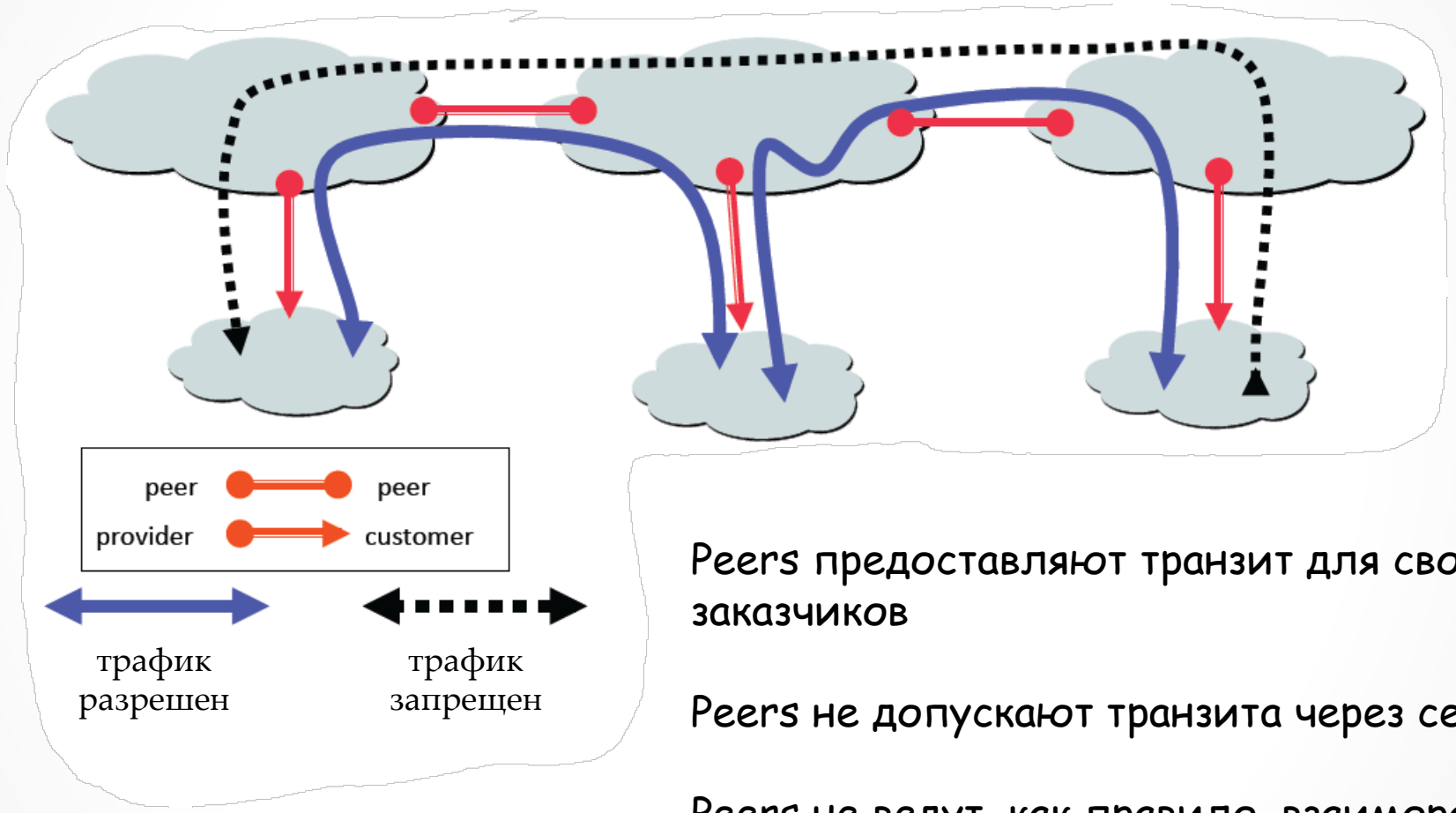


Заказчики платят провайдеру за свои пакеты

Иерархия заказчиков и провайдеров



Отношение Peering



Peers предоставляют транзит для своих важных заказчиков

Peers не допускают транзита через себя другим peers

Peers не ведут, как правило, взаиморасчетов

Итог

- *Интернет состоит из множества независимо управляемых АС*
- *Каждая АС использует свой внутренний протокол маршрутизации*
- *Оконечные АС используют простую маршрутизацию по умолчанию*
- *Транзитные АС должны сами определять какой выход использовать*
- *Для взаимодействия АС должны использовать BGP-4 протокол*

Маршрутизация в BGP

Введение в компьютерные сети

проф. Смелянский Р.Л.
Лаборатория Вычислительных комплексов
ф-т ВМК МГУ

Основы
Border Gateway Protocol (BGP-4)

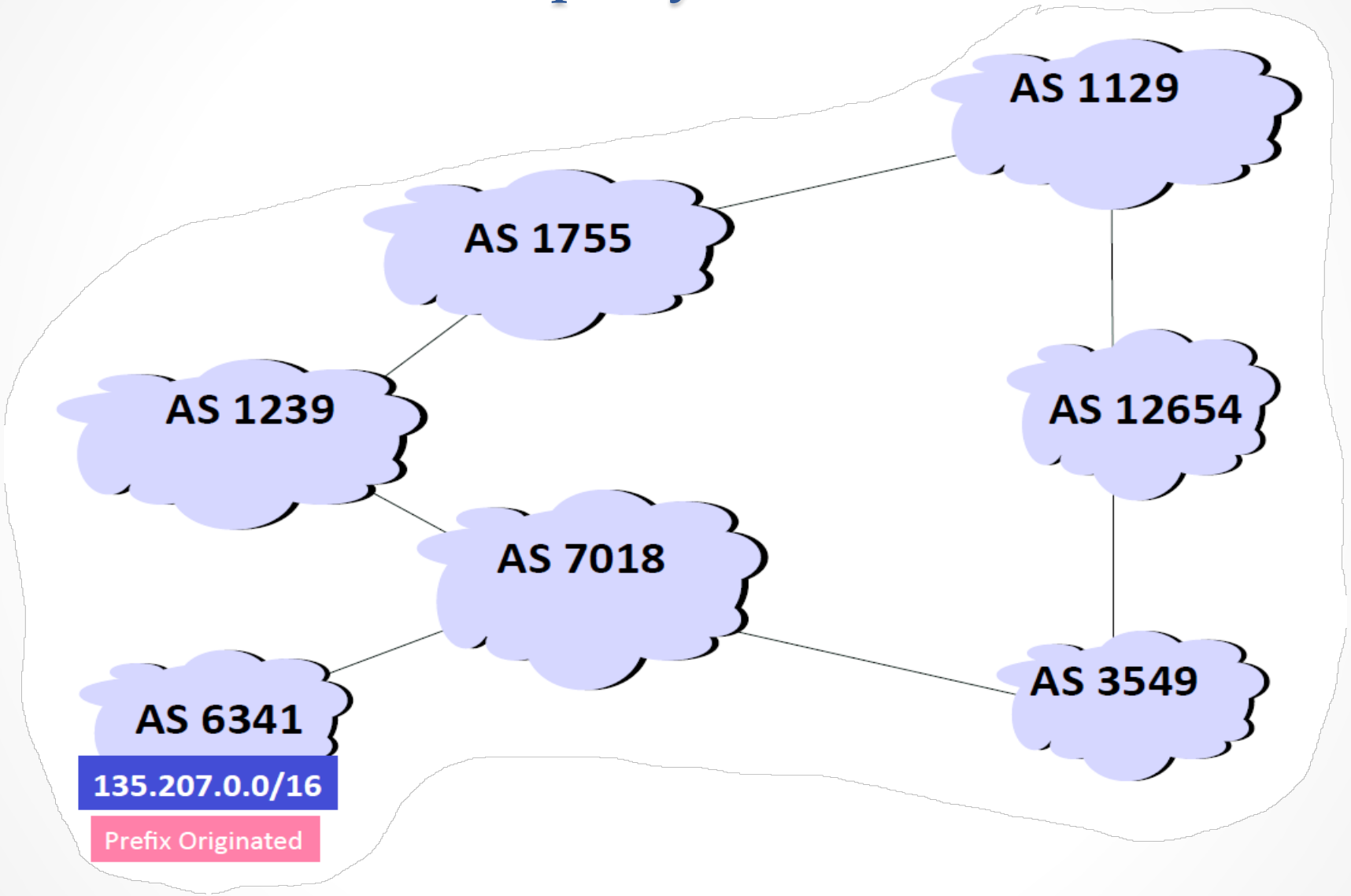
- BGP использует «вектор пути»
- Каждый BGP маршрутизатор рассылает список путей (путь – список AC)
  - AS_PATH
  - К сети 171.64/16 можно пройти по пути {AS7,AS52,AS13}
- Наличие цикла в маршруте определяется локально и такие маршруты игнорируются
- Из множества доступных маршрутов выбирается тот, которые наиболее всего соответствует политике AC
- Если маршрутизатор/линии вышли из строя, то маршрут изымается из списка

BGP сообщения

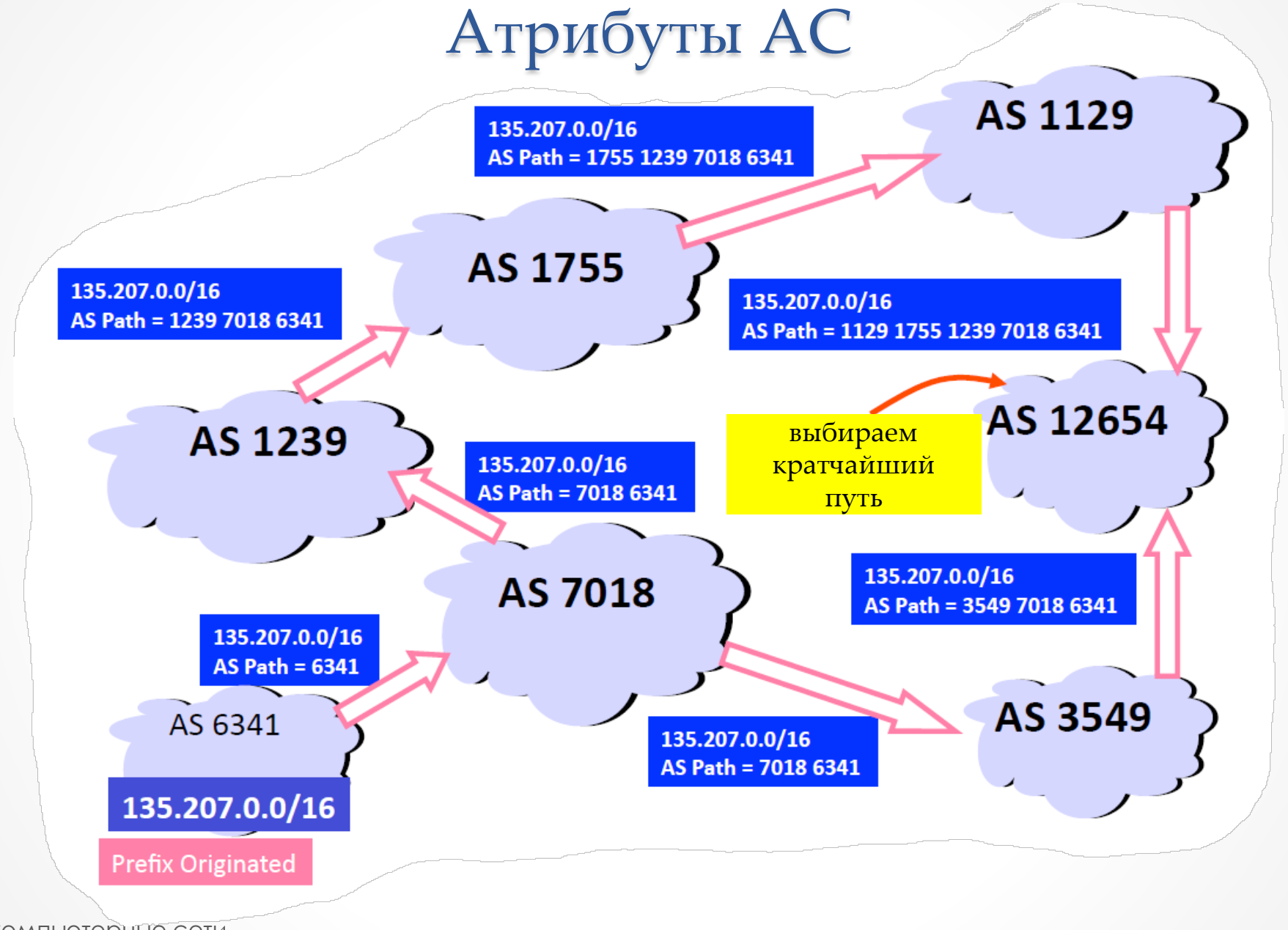
- ***Open*** Установка BGP сессии
- ***Keep Alive*** Проверка работоспособности через регулярные интервалы
- ***Notification*** Закрытие peering сессии
- ***Update*** Объявление нового или изъятие ранее объявленного маршрута
- ***BGP объявление = префикс + атрибуты маршрута***

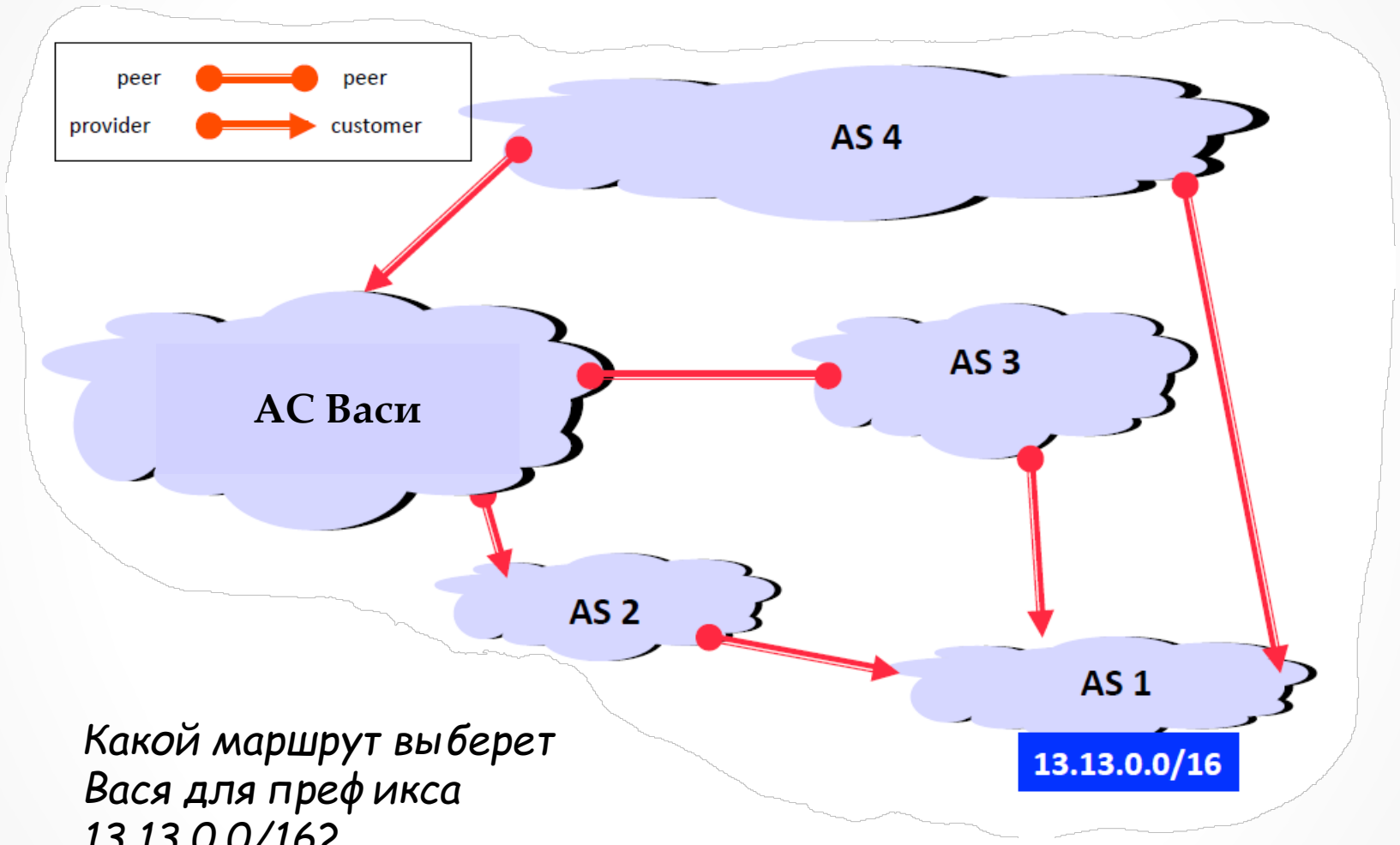
- ***Path attributes***
  - следующий скачок (hop), список AC (path), предпочтения, шлюзы выхода ….

Атрибуты АС

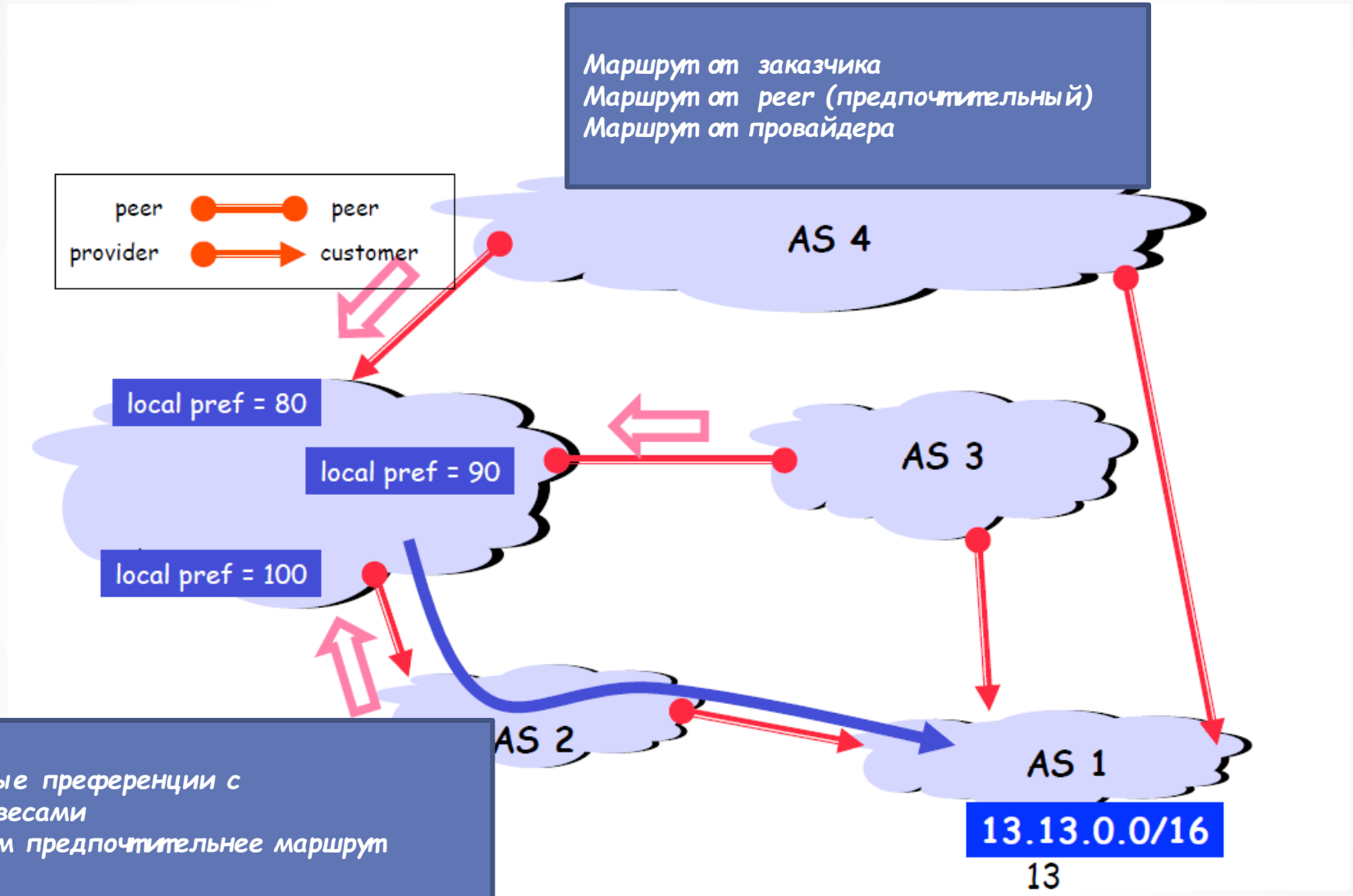

AS 1129
AS 1755
AS 1239
AS 12654
AS 7018
AS 3549
AS 6341
135.207.0.0/16
Prefix Originated

Атрибуты АС

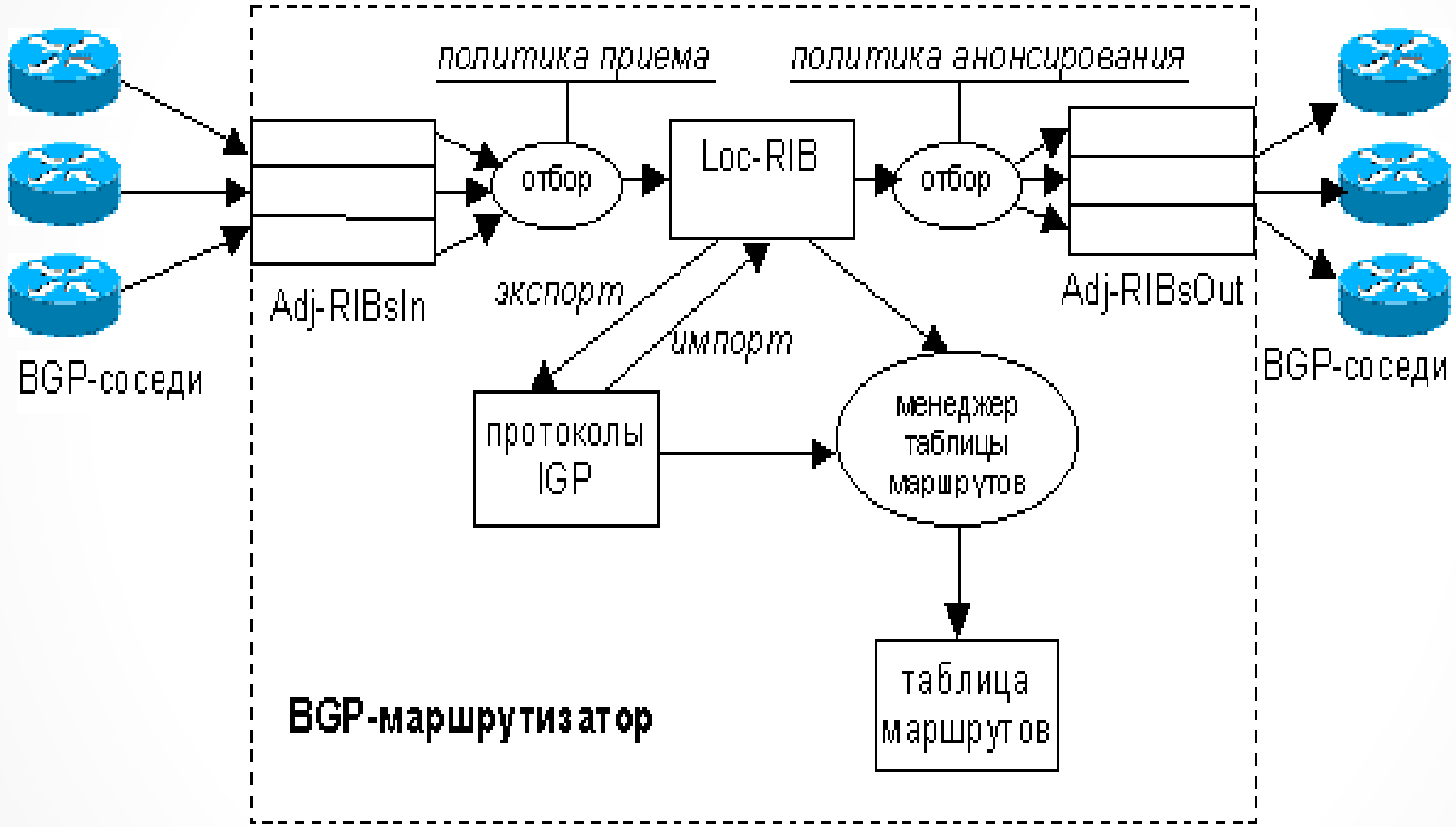

AS 1129
135.207.0.0/16
AS Path = 1755 1239 7018 6341
AS 1755
135.207.0.0/16
AS Path = 1239 7018 6341
135.207.0.0/16
AS Path = 1129 1755 1239 7018 6341
AS 12654
AS 1239
выбираем кратчайший путь
135.207.0.0/16
AS Path = 7018 6341
AS 7018
135.207.0.0/16
AS Path = 3549 7018 6341
135.207.0.0/16
AS Path = 6341
AS 6341
AS 3549
135.207.0.0/16
AS Path = 7018 6341
135.207.0.0/16
Prefix Originated



Какой маршрут выберет Вася для префикса 13.13.0.0/16?


Маршрут от заказчика
Маршрут от peer (предпочтительный)
Маршрут от провайдера
peer
peer
provider
customer
AS 4
local pref = 80
local pref = 90
local pref = 100
AS 3
AS 2
AS 1
13.13.0.0/16
13
Установить локальные преференции с соответствующими весами
Чем больше вес, тем предпочтительнее маршрут

Процедура выбора маршрута



Итог

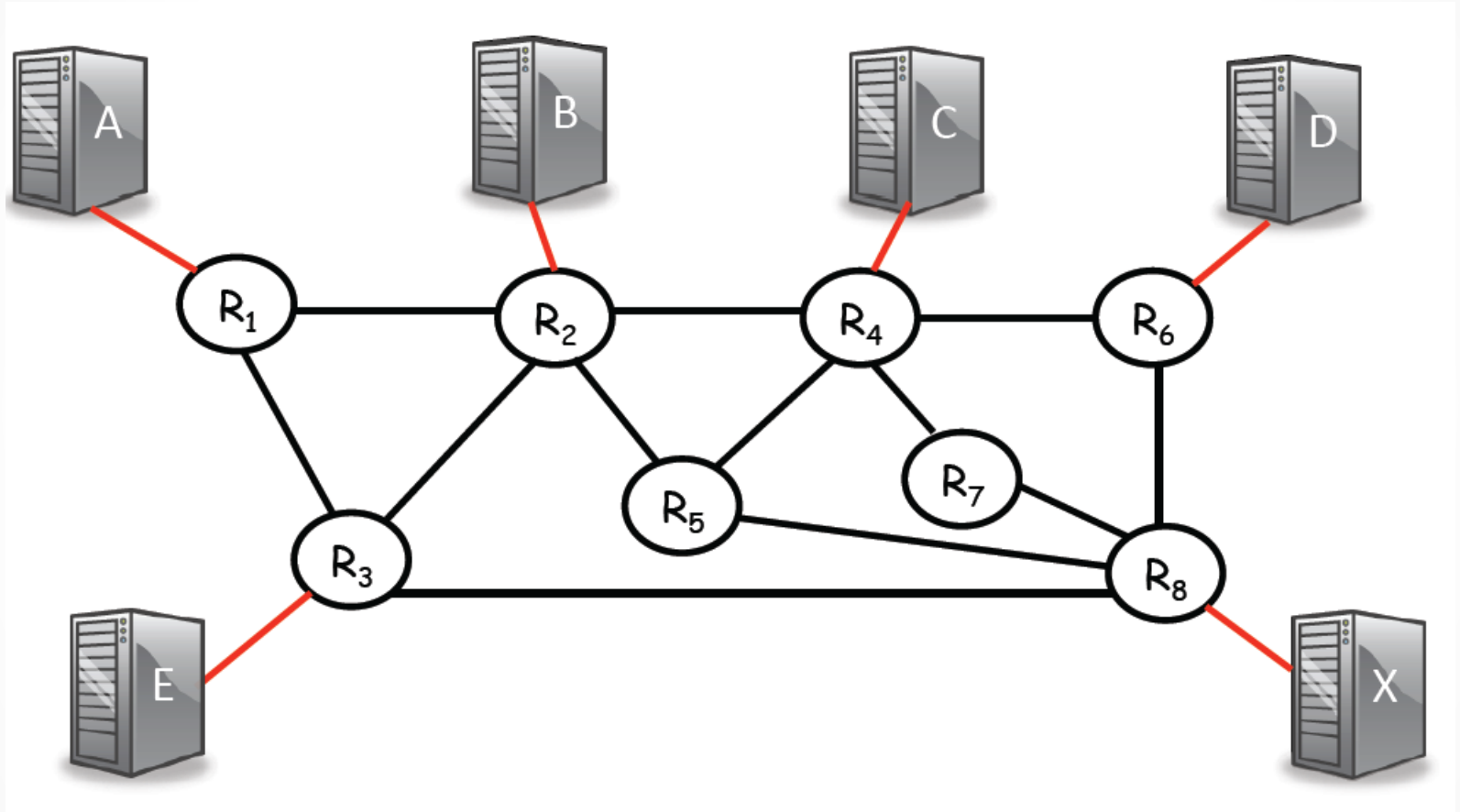
- *Все АС для взаимодействия в Интернете должны использовать BGP-4*
- *BGP-4 использует алгоритм маршрутизации по вектору пути, которые легко распознает циклы*
- *BGP-4 имеет сложный интерфейс, позволяющий каждой АС устанавливать свою локальную политику маршрутизации*
- *Каждая АС устанавливает свою политику для построения маршрутов, безопасности и локальных особенностей*

Групповая маршрутизация

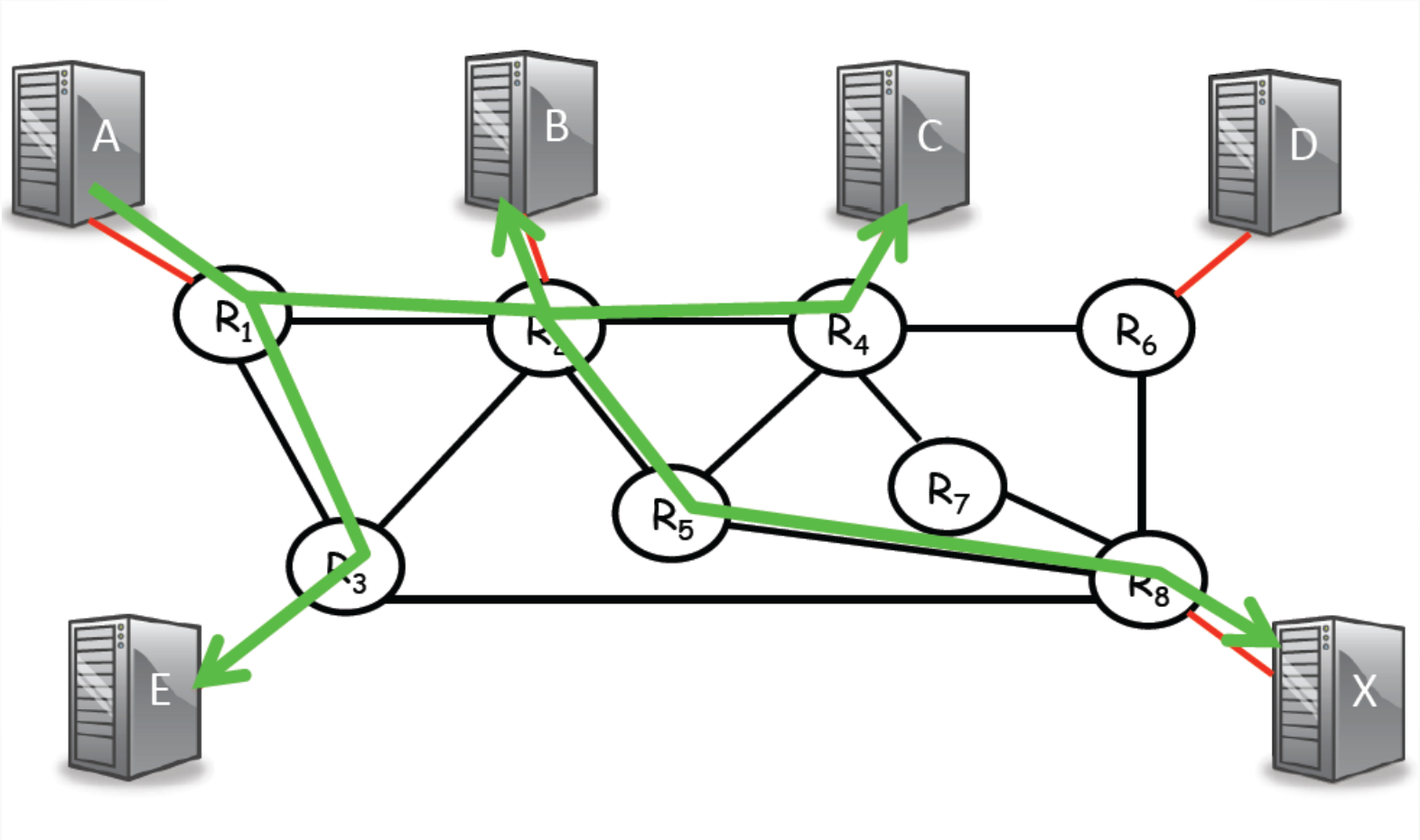
Введение в компьютерные сети

проф. Смелянский Р.Л.
Лаборатория Вычислительных комплексов
ф-т ВМК МГУ

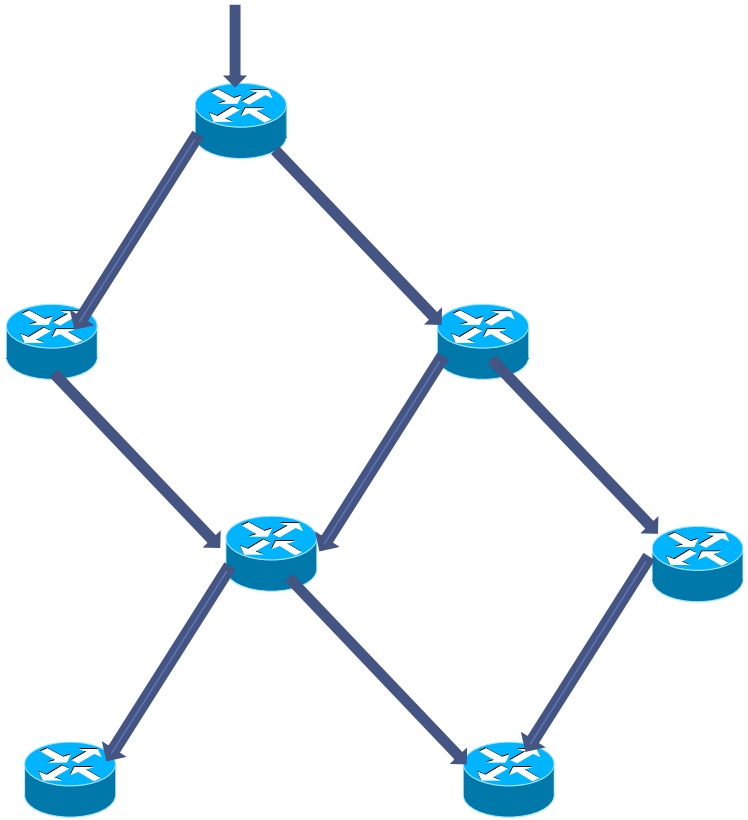
Групповая маршрутизация



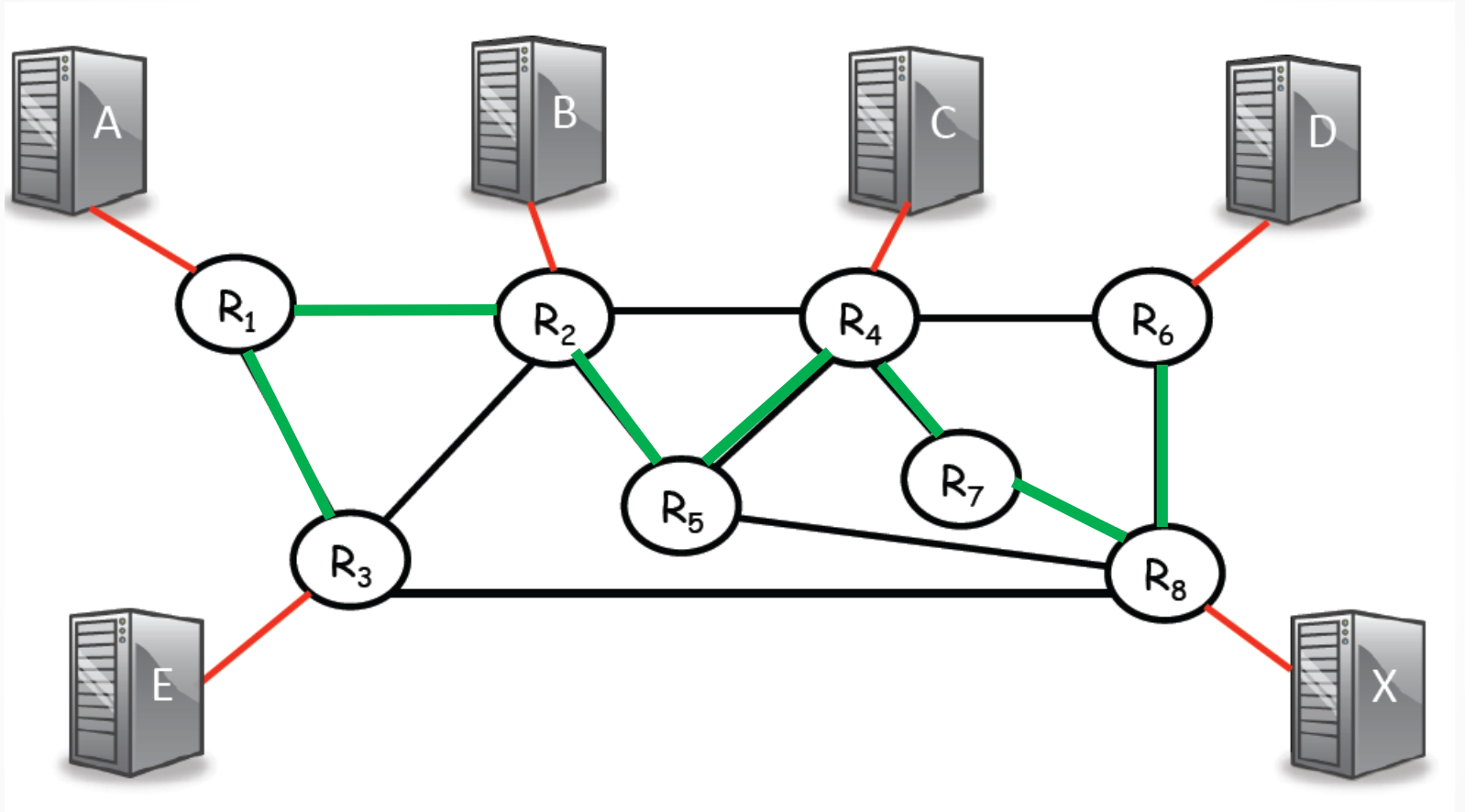
Групповая маршрутизация



Лавиной (Flooding)



Вещание по обратному пути (RPB)



aka Reverse Path Forwarding (RPB)

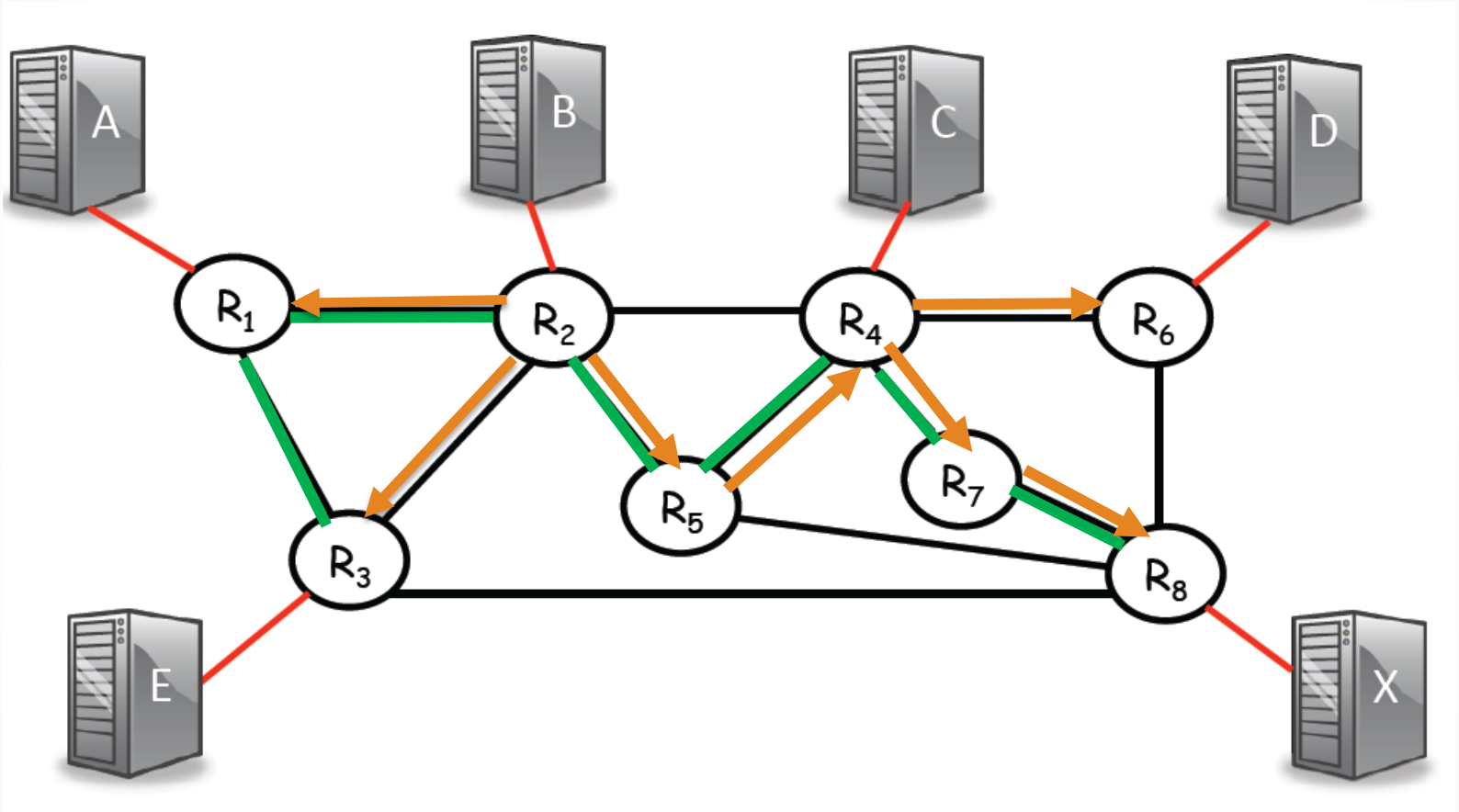
RPB + обрезка (pruning)

1. *Каждому хосту пакеты доставляют без циклов*

2. *Маршрутизаторы через которые нет путей к нужным маршрутизаторам и хостам, шлют сообщения обрезки*

3. *Результирующее дерево – дерево соединений минимальной стоимости от источника к группе интересующих хостов*

Одно дерево или несколько ?



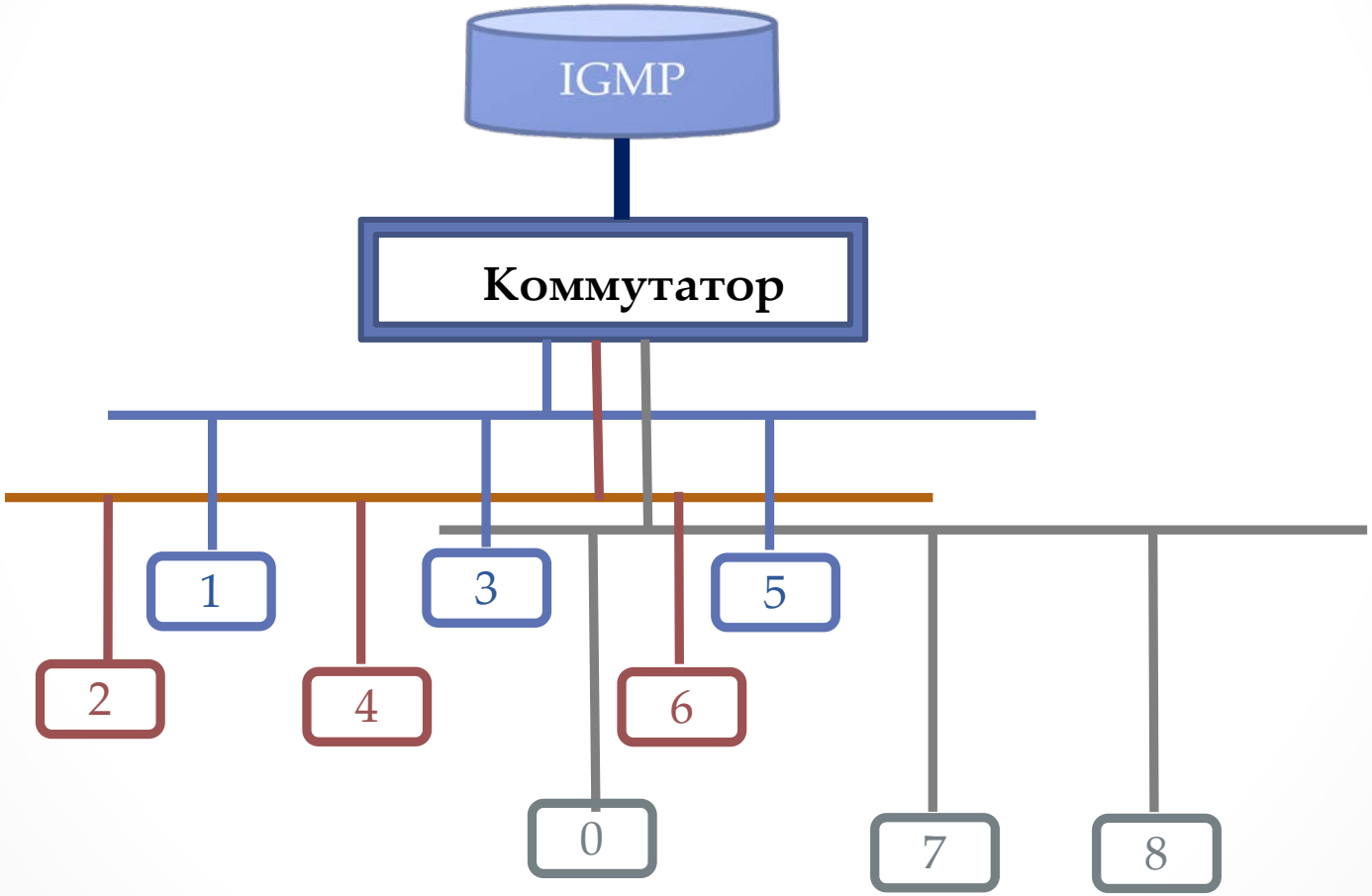
Групповая маршрутизация на практике

- Методы и принципы
  - Вещание по обратному пути и обрезка
  - одно дерево vs несколько деревьев
- Протоколы
  - Групповая адресация
  - IGMP – управление группой
  - DVMRP – первый протокол групповой маршрутизации
  - PIM – протокол маршрутизации для независимых групп

Адресация и подключение к группе

- *IPv4: сети класса D специально выделены для групповой адресации*
- *IGMP (Internet Group Management Protocol - RFC3376)*
  - *Этот протокол действует между хостом и непосредственно подсоединенным маршрутизатором на уровне L2*
  - *Хосты подписываются/запрашивают получать пакеты определенной группы*
  - *Маршрутизаторы периодически опрашивают хосты к каким группам они хотели бы быть подключенными*
  - *Если отклика нет, то членство в группе прекращается по time_out (soft-state)*

IGMP протокол



Групповая маршрутизация в Интернет

- *DVMRP*
  - *Distance Vector Multicast Routing Protocol (RFC 1075)*
  - *Первый протокол групповой маршрутизации в Интернет*
  - *Использует RPB + обрезка*
- *PIM (Protocol Independent Multicast)*
  - *Протокол независимой групповой маршрутизации*
  - *Два режима: dense и sparse*
  - *Dense (RFC 3973) – аналогичен DVMRP*
  - *Sparse (RFC 4601) – через точки rendezvous, через которые пакеты достигают небольшого количества деревьев соединений*

Групповая маршрутизация на практике

- *Актуальность групповой маршрутизации на практике постоянно возрастает*
  - *по большей части коммуникации индивидуализированы*
  - *ранние реализации были не эффективны*
  - *сегодня в основном используются для IP TV и быстрой рассылки*
  - *используется отдельными приложениями*

Интересные вопросы:

Как сделать групповое взаимодействие надежным?
Как реализовать управление потоком?
Как поддерживать разную скорость работы с разными клиентами?
Как обеспечить конфиденциальность при групповом взаимодействии?
Как быть когда коммутация на L2 и маршрутизация на L3 независимы?

Маршрутизация по соединяющему дереву

Введение в компьютерные сети

проф. Смелянский Р.Л.
Лаборатория Вычислительных комплексов
ф-т ВМК МГУ

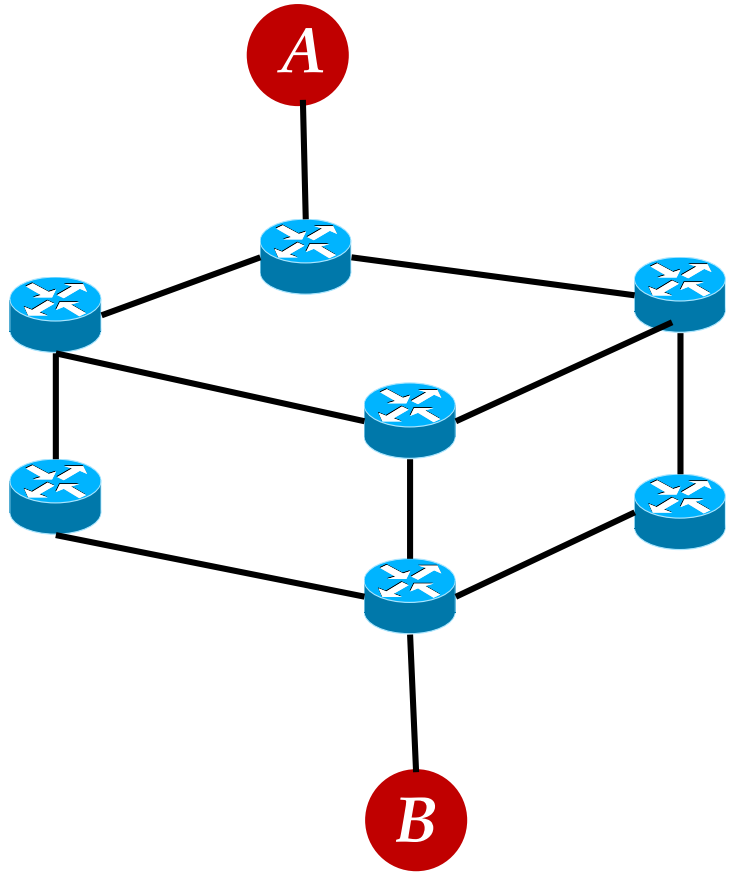
План

- *Ethernet коммутаторы «маршрутизируют» пакеты*
- *Как коммутатор находит неизвестные ему адреса, как он защищается от циклов?*
- *Коммутатор используют единое соединяющее дерево, вдоль которого передаются пакеты*

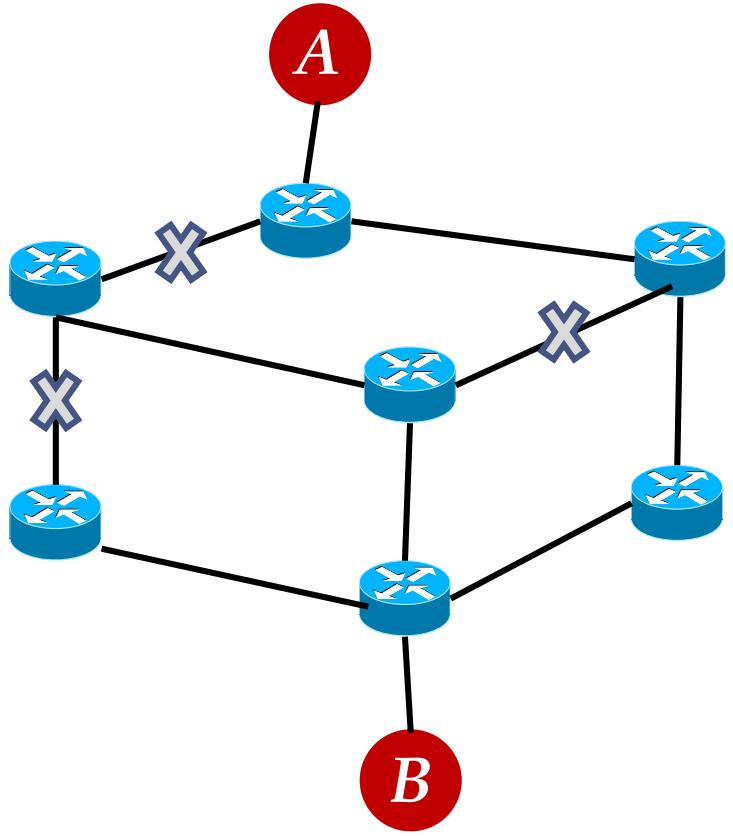
Ethernet коммутатор

1. *Анализирует заголовок каждого поступающего кадра*
2. *Если DA есть в таблице коммутации, передать кадр на надлежащий порт-выход*
3. *Если DA нет в таблице коммутации, разослать кадр по всем портам за исключением того, по которому он пришел*
4. *Коммутатор «изучает» сеть - Таблица коммутации пополняется за счет изучения адресов SA поступающих пакетов*

«Изучение» может зацикливаться



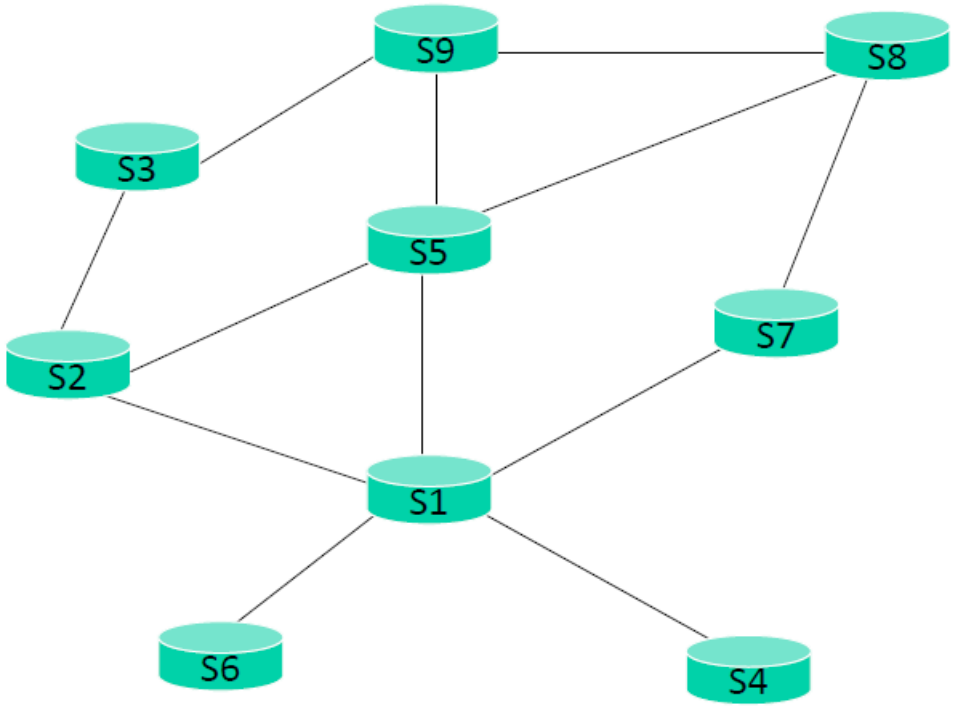
«Изучение» может зацикливаться



Предотвращение зацикливаний
(протокол соединяющего дерева - spanning tree protocol)

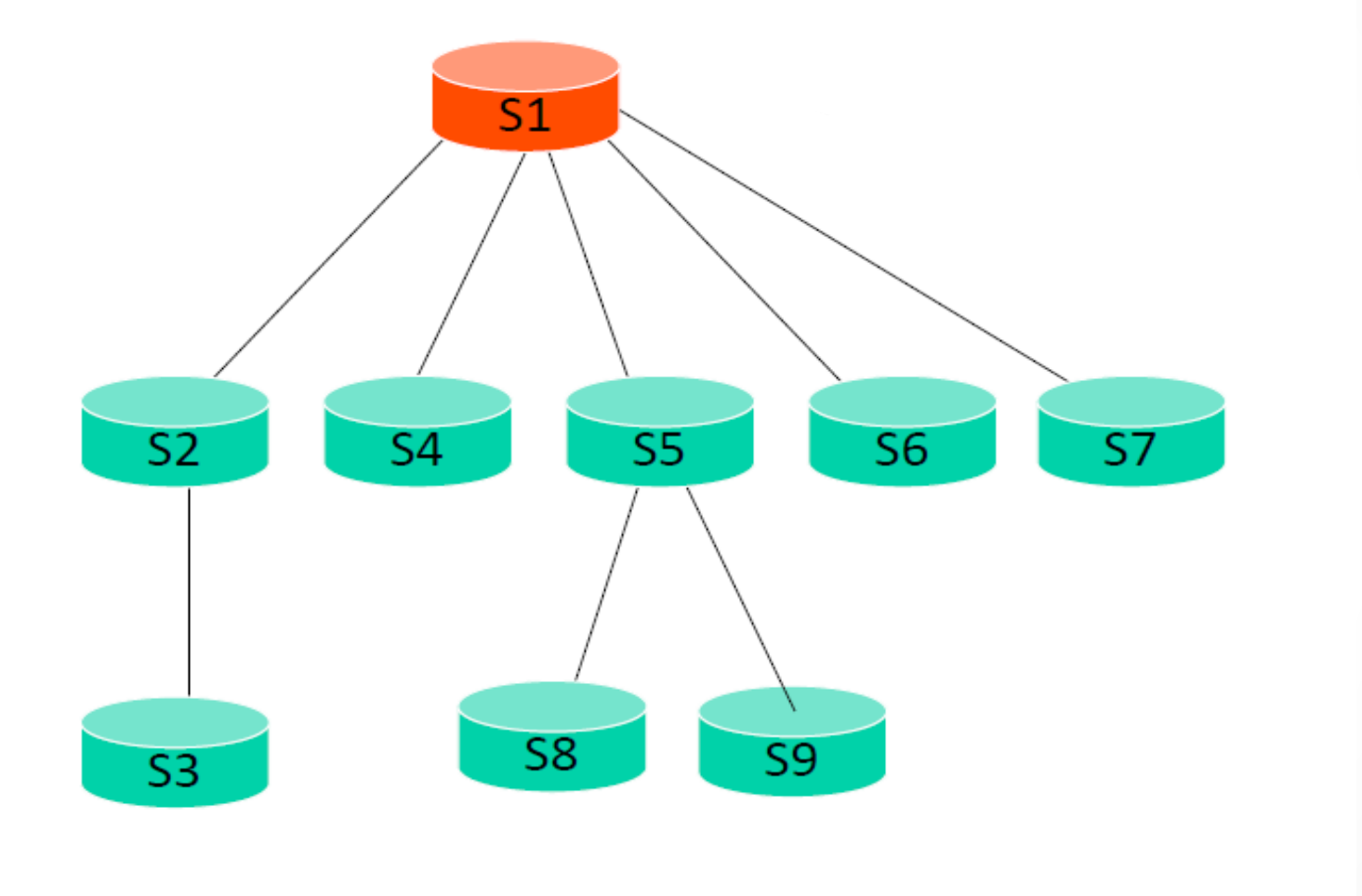
- *Топология коммутаторов – граф*
- *Протокол STP находит подграф , в котором все вершины соединены без циклов*
  - *соединяющее – к любому коммутатору есть путь*
  - *дерево – нет циклов*
- *STP распределенный протокол*
  - *какой из коммутаторов – корень дерева*
  - *каким портам разрешено рассылать кадры вдоль дерева ?*

Пример соединяющего дерева



- *выбираем корень*
- *кадр коммутируется на тот порт, который ведет от корня с наименьшим число скачков (hop)*

Соединяющее дерево для нашего примера

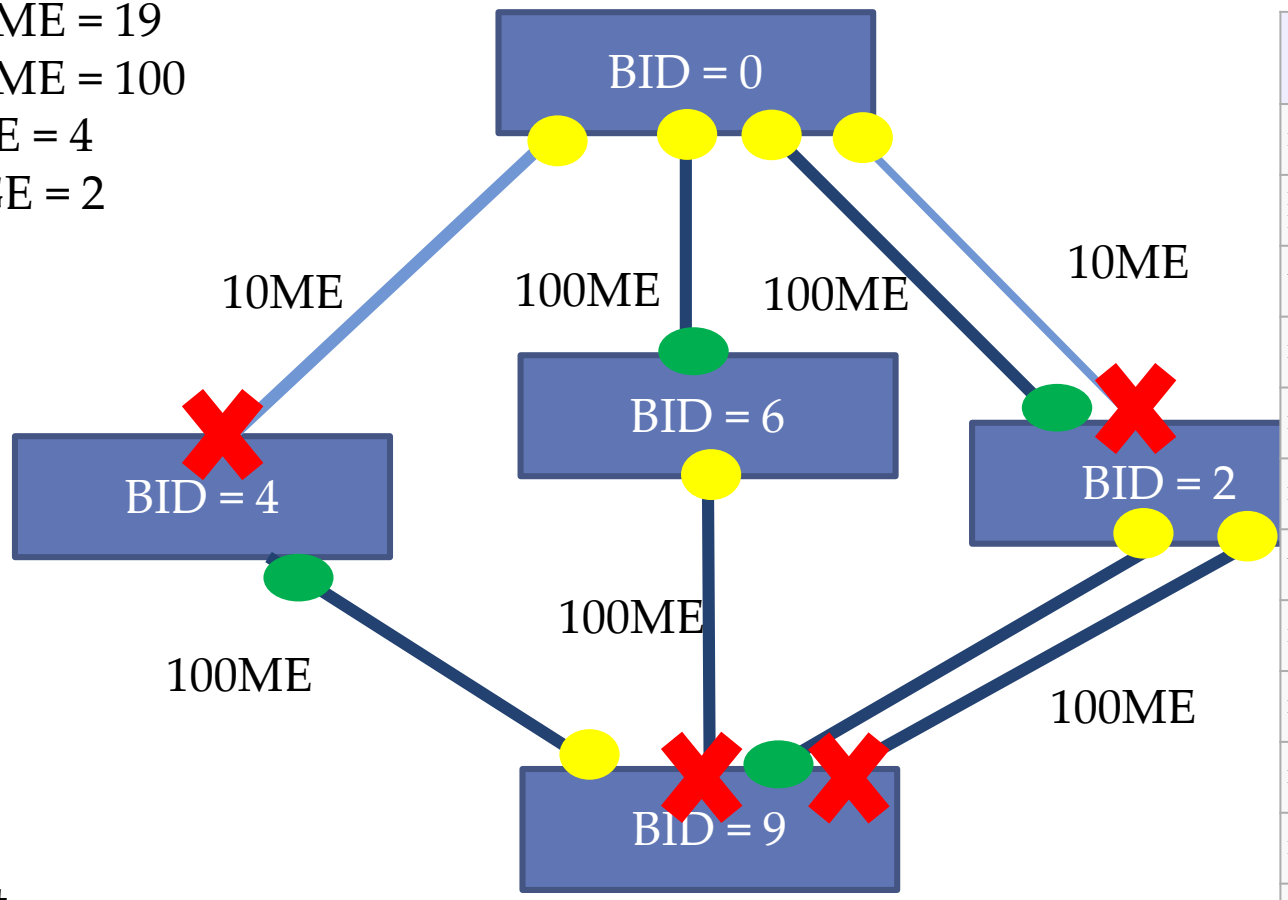


Как это работает

- После включения коммутаторов в сеть, по умолчанию каждый коммутатор считает себя корневым (root).
- Каждый коммутатор начинает посылать по всем портам конфигурационные Hello BPDU пакеты раз в 2 секунды. (BPDU (Bridge PDU)- ID отправителя, ID корня, расстояние от отправителя до корня)). Изначально все считают себя корнем (расстояние = 0).
- Если коммутатор получает BPDU с идентификатором Bridge ID меньшим, чем свой собственный, он прекращает генерировать свои BPDU и начинает ретранслировать BPDU с этим идентификатором. Таким образом в конце концов в этой сети Ethernet остаётся только один коммутатор, который продолжает генерировать и передавать собственные BPDU. Он и становится корневым (root bridge).
- Остальные коммутаторы ретранслируют BPDU корневого, добавляя в них собственный идентификатор и увеличивая счетчик пути (path cost).
- Для каждого сегмента сети, к которому присоединены два и более портов коммутаторов, происходит определение rootport, потом designated port — порта, через который BPDU, приходящие от корневого коммутатора, попадают в этот сегмент.
- После этого все порты в сегментах, к которым присоединены 2 и более портов коммутаторов, блокируются за исключением root port и designated port.
- Корневой хост продолжает посылать свои Hello BPDU раз в 2 секунды.

Построение ST

100ME = 19
10 ME = 100
1 GE = 4
10GE = 2



- Root port
- Designated port

Bridge Protocol Data Unit (BPDU) —

Название поля	Размер поля
Protocol Identifier	2 байта
Protocol Version Identifier	1 байт
BPDU Type	1 байт
Flags	1 байт
Root Identifier	8 байт
Root Path Cost	4 байта
Bridge Identifier	8 байт
Port Identifier	2 байта
Message Age	2 байта
Max Age	2 байта
Hello Time	2 байта
Forward Delay	2 байта

Заключение

- *Есть много способов маршрутизации пакетов в сети*
- *Для расчета маршрута используют протокол маршрутизации*
- *Алгоритмы маршрутизации часто используют соединяющие деревья с минимальной стоимостью до места назначения*
- *Маршрутизация с множественными путями позволяет распределять нагрузку по нескольким линиям одновременно*
- *Групповая маршрутизация обеспечивает доставку сразу нескольким хостам*