# РАЗРАБОТКА И ВНЕДРЕНИЕ НОВЫХ ГИБРИДНЫХ ПРОТОКОЛОВ МАРШРУТИЗАЦИИ ДЛЯ БОЛЬШИХ СЕТЕЙ

**Х. Хаю , Абросимов Л.И.**

Изменились требования к центрам обработки данных для удовлетворения растущих потребностей в трафике. Существует необходимость в простом, масштабируемом протоколе маршрутизации, который обладает гибкостью и простотой управления для поддержки больших сетей [1]. Протоколы дистанционно-векторной маршрутизации очень просты и легки в реализации, но они страдают от петель маршрутизации. Протоколы состояния канала, с другой стороны, обладают преимуществами быстрой сходимость, разделения домена маршрутизации за счет дополнительной сложности для реализации, настройки и устранения неполадок.

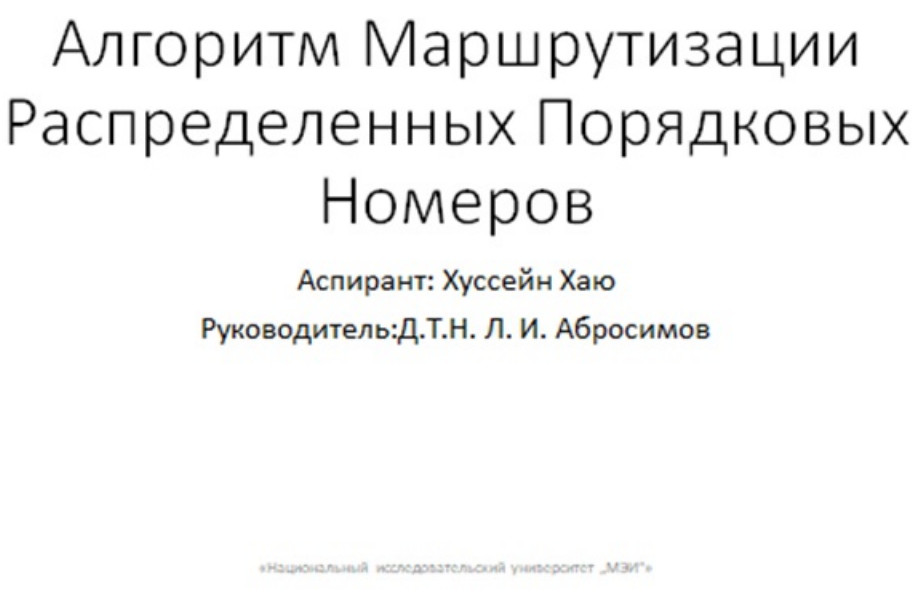
В данном докладе предлагается новый протокол без петели, который сочетает в себе простоту протоколов дистанционно-векторной маршрутизации, свободу от петель и возможность использования в крупномасштабных ячеистых сетях, как и в протоколах состояния канала.

Протокол использует гибридный алгоритм дистанционно-векторный и состояния канала.

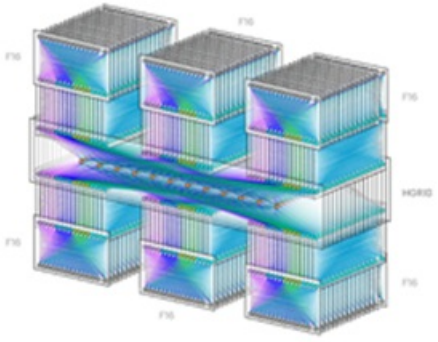
Проектирование и реализация архитектуры протокола подробно описаны в презентации

## Литература

Premji, Ariff, et al. “Use of BGP for Routing in Large-Scale Data Centers”. Tech. Rep., Aug. 2016. [Online]. Available: https://doi.org/10.17487/rfc7938



# Введение

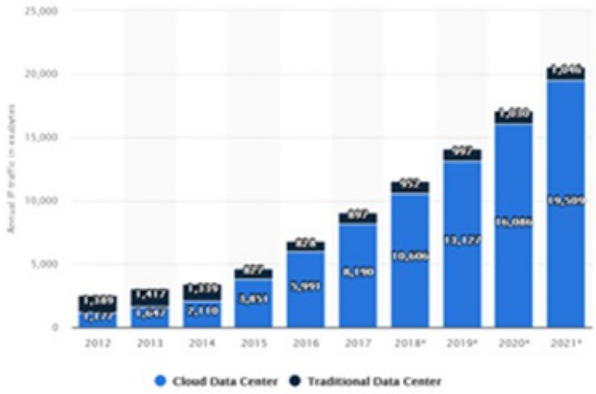


Пример сверхбольшой сети центров обработки данных (Facebook)*

- Тенденция облачных вычислений и приложений сетевой виртуализации значительно увеличила трафик данных в сетях центров обработки данных.
- Для поддержки такого большого объема трафика центры обработки данных постоянно расширяются в масштабах. Схемы топологии и требования также резко изменились.

*Reinventing Facebook's data center network

# Трафик сетевых приложений в центрах обработки данных




- Современные центры обработки данных включают в себя больше трафика, которым обмениваются серверы, который называется трафиком восток-запад.
- 71,5% этого трафика останется в центре обработки данных*

*Traffic behavior in cloud data centers: A survey

# Требования к центрам обработки данных*

- ТРЕБОВАНИЕ 1: Выберите топологию, которую можно масштабировать "горизонтально", добавляя дополнительные каналы связи и сетевые устройства того же типа, не требуя обновления самих сетевых элементов.
- ТРЕБОВАНИЕ 2: Определите узкий набор программных функций/протоколов, поддерживаемых множеством поставщиков сетевого оборудования.
- ТРЕБОВАНИЕ 3: Выберите протокол маршрутизации, который имеет простую реализацию с точки зрения сложности программного кода и простоты операционной поддержки.
- ТРЕБОВАНИЕ 4: Максимально сведите к минимуму область отказов оборудования или проблем с протоколом.
- ТРЕБОВАНИЕ 5: Разрешить некоторую разработку трафика, предпочтительно с помощью явного управления следующим переходом префикса маршрутизации с использованием встроенной механики протокола.

*Use of BGP for routing in large scale data centers.

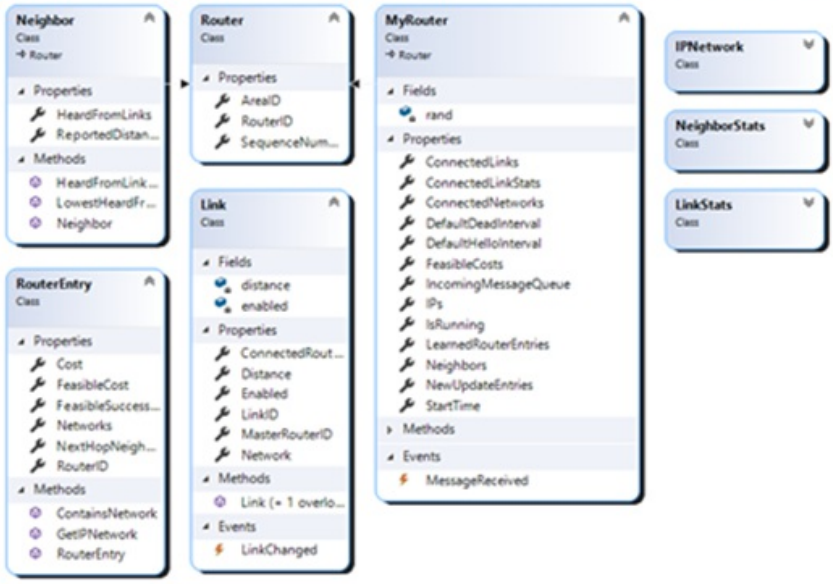
# Преимущества и недостатки существующих решений

| Техническое решение | Преимущества | Недостатки |
|---|---|---|
| Протокол маршрутизации EIGRP | быстрая конвергенция и масштабируемость в небольших и средних сетях. Поддерживает неравномерную балансировку нагрузки | Проприетарный протокол (только Cisco Systems, Inc.). Не масштабируется для сверхбольших сетей. Невозможность разделить домен маршрутизации. |
| Протоколы состояния канала: OSPF и IS-IS | Полное представление о топологии. IS-IS может работать на уровне 2 без зависимости от IP. | Сложность управления и устранения неполадок. Избыточный обмен трафиком обновлений во время повторной сходимости. Не масштабируется для сверхбольших сетей. |
| Протокол граничного шлюза BGP | Масштабируемость. меньшая сложность разработки протокола и меньшие затраты на заполнение информацией по сравнению с состоянием канала, а также простота управления и устранения неполадок. | Неоптимальные вычисленные пути. Негарантированная сходимость (постоянные колебания маршрута). |
| Программно-определяемая сеть SDN | Простота управления и автоматизации. Проектирование сетевого трафика и поддержка качества обслуживания. | Ограничения масштабируемости на уровне управления в больших сетях. Единственная точка отказа. Атаки безопасности на контроллер. |

# основные характеристики предлагаемого решения

- Возможность разделения доменов маршрутизации на поддомены для повышения масштабируемости
- Разделите проблему маршрутизации на две подзадачи
  - Вычисление путей к узлам инфраструктуры, которое достигается с использованием модифицированного алгоритма бесцикловой дистанционно-векторной маршрутизации
  - Распределение сопоставления между узлами инфраструктуры и сетями для вычисления таблиц маршрутизации
- Свобода цикла достигается путем добавления двух компонентов к метрике (порядковый номер, бит запроса на увеличение порядкового номера) и незначительных изменений в исходном алгоритме вектора расстояния

# Диаграмма классов логики маршрутизации



# Диаграмма классов сообщений

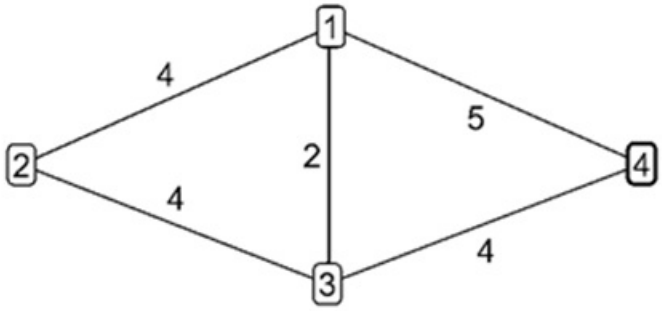

**Algorithm 1** Handling update messages algorithm

```
1: procedure UpdateMessageReceived(updateMessage)
2:     neighbor ← Neighbors[updateMessage.SourceRouterID[          ▷ Exit if failed
3:     for all updateEntry in updateMessage.UpdateEntries do
4:         newDistance ← updateEntry.Distance + GetLinkByID(updateMessage.LinkID)
5:         routerEntry ← LearnedRouterEntries[updateEntry.RouterID[
6:         if routerEntry = null then
7:             routerEntry ← CreateNewRouterEntry(updateEntry, newDistance)
8:             newUpdateEntry ← CreateNewUpdateEntry(updateEntry,newDistance)
9:             routerEntry.NextHopNeighbors.Add(neighbor)
10:            routerEntry.FeasibleSuccessors.Add(neighbor)
11:            UpdateNeighborReportedDistance(neighbor, updateEntry)
12:            UpdateLinkState(updateEntry, routerEntry, newUpdateEntry)
13:            AddDestinationsRouters(newUpdateEntry)
14:            AddOrUpdateNewUpdateEntry(newUpdateEntry)
15:            LearnedRouterEntries.Add(routerEntry)
16:        else
17:            if (
    updateEntry.RouterID = self.RouterID and updateEntry.RISN = true
    and updateEntry.SequenceNumber = self.SequenceNumber
               )) then
18:                IncreaseSelfRouterSequenceNumber(routerEntry)
19:                newUpdateEntry ← CreateSelfNewUpdateEntry()
20:                AddDestinationsRouters(newUpdateEntry)
21:                AddOrUpdateNewUpdateEntry(newUpdateEntry)
22:            else if (
    updateEntry.SequenceNumber > routerEntry.FeasibleCost.SequenceNumber
    or (updateEntry.SequenceNumber = routerEntry.FeasibleCost.SequenceNumber)
    and (newDistance < routerEntry.FeasibleCost.Distance)
               ) then
23:                oldSeqNo ← routerEntry.FeasibleCost.SequenceNumber
24:                routerEntry.FeasibleCost.Distance ← newDistance
25:                routerEntry.FeasibleCost.SequenceNumber ← updateEn-
    try.SequenceNumber
26:                routerEntry.Cost.Distance ← newDistance
27:                routerEntry.Cost.SequenceNumber ← updateEntry.SequenceNumber
28:                routerEntry.NextHopNeighbors.Clear()
29:                routerEntry.NextHopNeighbors.Add(neighbor)
30:                newUpdateEntry ← CreateNewUpdateEntry(updateEntry, newDistance)
31:                routerEntry.FeasibleSuccessors.Add(neighbor)
32:                UpdateNeighborReportedDistance(neighbor, updateEntry)
33:                UpdateFeasibleSuccessors(routerEntry)                ▷
    Some of the old feasible successor have to be removed from the feasible successors table
    if they do not satisfy the feasibility condition
34:                UpdateLinkState(updateEntry, routerEntry, newUpdateEntry)
```

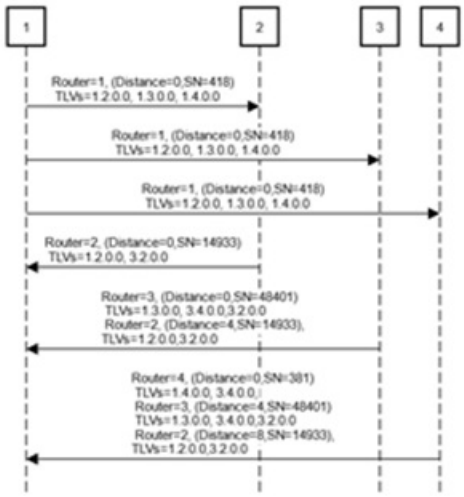
# алгоритм обработки сообщений об обновлении

```
35:                if (updateEntry.SequenceNumber > oldSeqNo) then
36:                    routerEntry.Cost.RISN ← updateEntry.RISN
37:                    newUpdateEntry.RISN ← updateEntry.RISN
38:                else
39:                    routerEntry.Cost.RISN ← routerEntry.Cost.RISN or updateEn-
    try.RISN
40:                    newUpdateEntry.RISN ← newUpdateEntry.RISN or updateEn-
    try.RISN
41:                end if
42:                AddDestinationsRouters(newUpdateEntry)
43:                AddOrUpdateNewUpdateEntry(newUpdateEntry)
44:            else if (
    updateEntry.SequenceNumber = routerEntry.FeasibleCost.SequenceNumber
               ) then
45:                UpdateNeighborReportedDistance(neighbor, updateEntry)
46:                if Reported Distance changed then
47:                    if (updateEntry.Distance < routerEntry.FeasibleCost.Distance) then
48:                        routerEntry.FeasibleSuccessors.Add(neighbor)
49:                    else
50:                        routerEntry.FeasibleSuccessors.Remove(neighbor)
51:                    end if
52:                    UpdateDistance(routerEntry)
53:                end if
54:                routerEntry.Cost.RISN ← routerEntry.Cost.RISN or updateEntry.RISN
55:                UpdateLinkState(updateEntry, routerEntry, newUpdateEntry)
56:                if There is a change in cost or RISN or link state then
57:                    newUpdateEntry ← CreateNewUpdateEntryFromRouterEn-
    try(routerEntry)
58:                    AddDestinationsRouters(newUpdateEntry)
59:                    AddOrUpdateNewUpdateEntry(newUpdateEntry)
60:                end if
61:            end if
62:        end if
63:    end for
64: end procedure
```

# Пример (Топология сети)

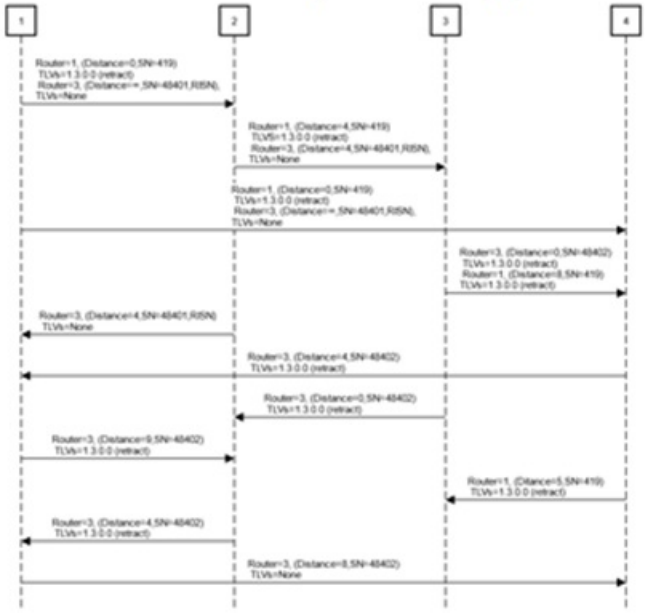



# Инициализация



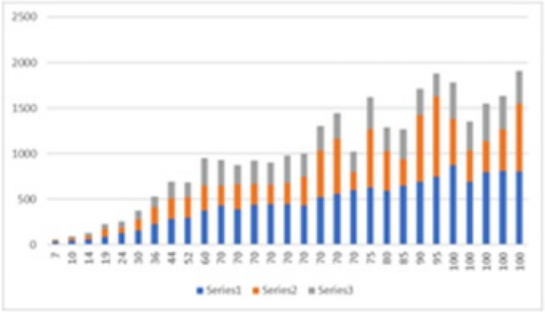

- Маршрутизатор "1" отправляет информацию о себе (порядковый номер и расстояние (0), а также о сетях, непосредственно подключенных к маршрутизатору "1") на маршрутизаторы "2", "3" и "4".
- Когда маршрутизатор узнает о другом маршрутизаторе, он отправляет эту информацию своим соседям с указанием расстояния до нового изученного

# Обмен сообщениями для повторной сходимости



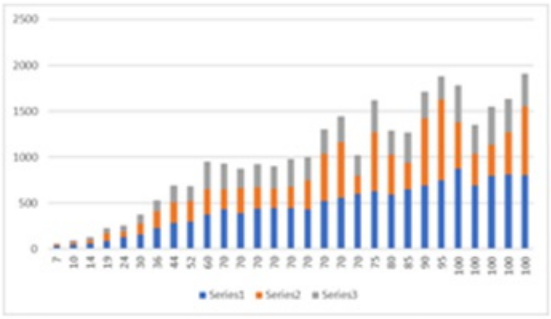

- Связь между "1" и "3" вышла из строя
- Маршрутизатор "1" втягивает сеть по связи "1-3" (префикс "1.3.0.0") и отправляет запрос на увеличение порядкового номера для маршрутизатора 3 (с бесконечным расстоянием), поскольку маршрутизатор "1" не имеет возможного узла следующего перехода для маршрутизатора "3". Запрос отправляется на маршрутизаторы "2" и "4".
- После того, как запрос на увеличение последовательности достигает маршрутизатора "3", он отправляет обновления с увеличенным порядковым номером, а также удаляет префикс "1.3.0.0", поскольку он также больше недоступен из"3".
- Обновления распространяются в сети в соответствии с правилами инкрементных запускаемых обновлений с разделением горизонта.

# Количество сообщений об обновлении по количеству связей




- серия 1 (синий) фаза инициализации
- серия 2 (оранжевая) фаза повторной сходимости после отказа одной из связей
- серия 2 (оранжевая) фаза повторной сходимости после восстановления вышедшей из строя связи

# Количество сообщений об обновлении по количеству связей




- серия 1 (синий) фаза инициализации
- серия 2 (оранжевая) фаза повторной сходимости после отказа одной из связей
- серия 2 (оранжевая) фаза повторной сходимости после восстановления вышедшей из строя связи

# заключение

- Протокол имеет гораздо меньшую базу данных для обслуживания, чем другие протоколы состояния канала. Это связано с тем, что объявляются только сети, а не сами связи, и сети также могут быть дополнительно суммированы с помощью суммирования маршрутов.
- Протокол также является дистанционно-векторным только для узлов инфраструктуры. Следовательно, количество обновлений после изменения топологии будет ограничено.