УНИВЕРСИТЕТ ИТМО

Разработка алгоритмов работы программно-конфигурируемой сети с распределенным инфраструктурным слоем

Иванов Константин, группа M4239

Научный руководитель: **Ульянцев В.И.**, к.т.н., доц. каф. КТ

Внешний научный руководитель: **Кузнецов П.В.**, PhD, Telecom ParisTech

Санкт-Петербург, 2018

Программно-конфигурируемые сети (SDN)

- Слой данных (коммутаторы)
  - Передача пакетов по сети
- Контрольный слой (контроллеры)
  - Часто ⩾ 1 контроллера – распределенный контр. слой

Контр. слой управляется SDN приложениями

- Маршрутизация
- Брандмауэр
- Балансировка нагрузки
- …

Архитектура SDN сети.

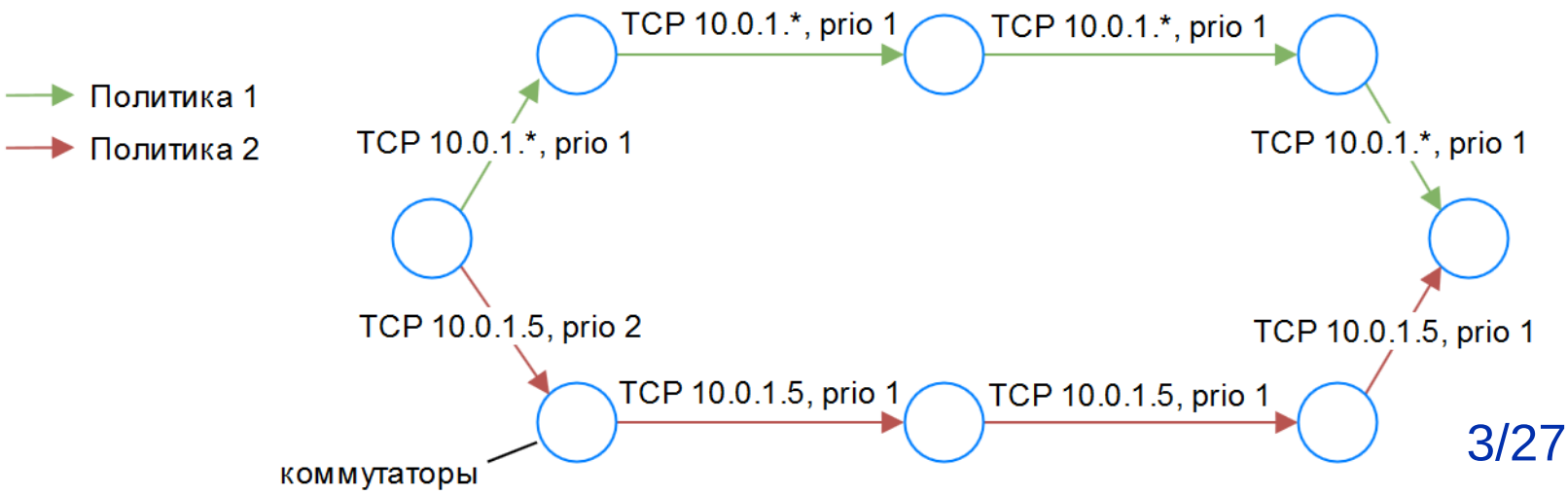
https://perso.telecom-paristech.fr/kuznetso/projects/SDN/cpc/

Конфигурация и политики сети

- Пакеты варьируются по типам

  TCP/UDP, адрес и порт назначения…
- Сетевая политика – ориентированный граф

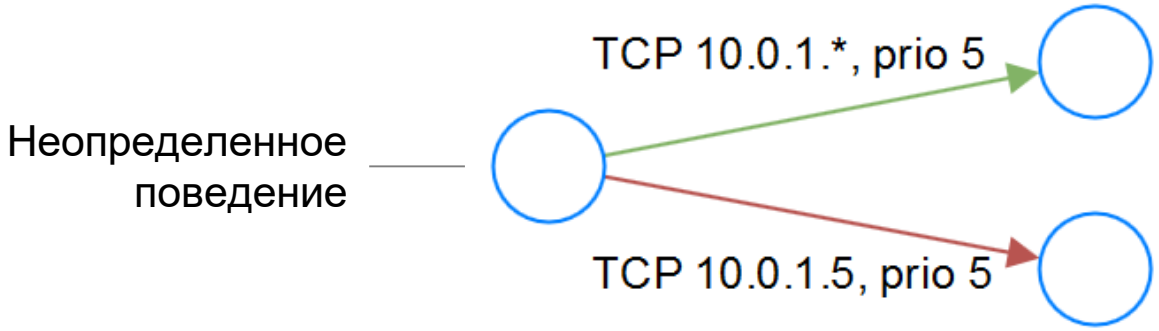
  На ребрах – маски на тип пакета и приоритеты
- Сетевая конфигурация – объединение сетевых политик



Конфликты сетевых политик

В общем случае, предлагаемые на установку политики могут конфликтовать

Когда разные SDN приложения управляют одним видом трафика



Контрольный слой не должен допускать установку политики, конфликтующей с имеющейся конфигурацией

Однако, конкурентная установка политик, касающихся одного типа пакетов и одних и тех же коммутаторов, маловероятна

Распространенность SDN

Используется в дата-центрах

- Google[1]
- Microsoft Azure[2]
- Dell EMC[3]

Реализуется компаниями[4]

- VMware
- Cisco
- AT&T

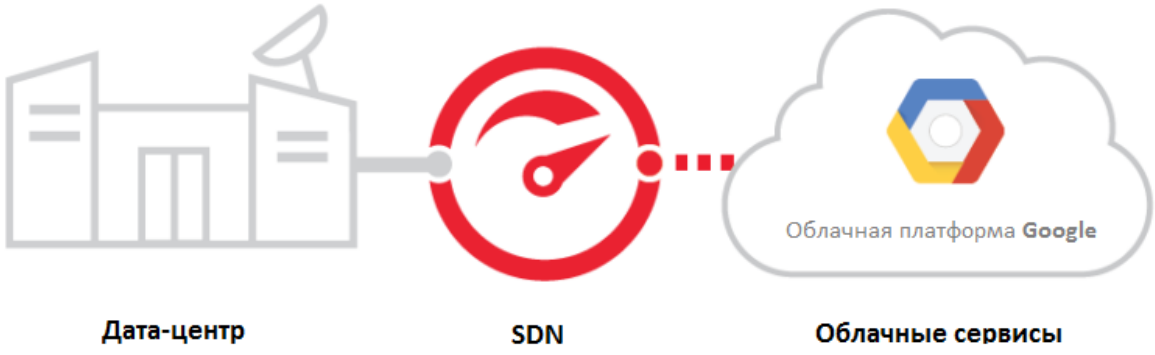


Схема управления дата-центром через SDN в Google [5]

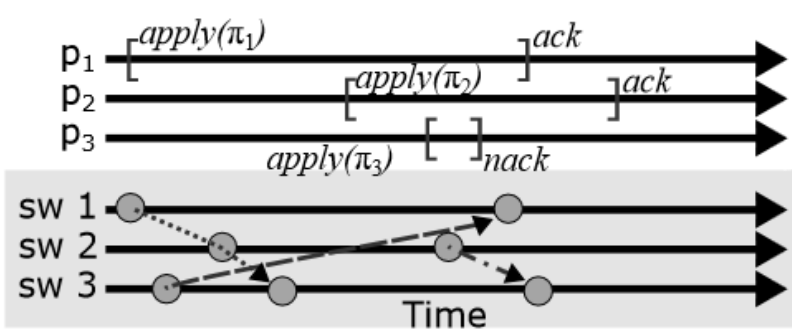
[1] B4: Experience with a globally-deployed software defined WAN / S. Jain [и др.] // ACM SIGCOMM Computer Communication Review. Т. 43. — ACM. 2013. — С. 3–14.
[2]https://searchsdn.techtarget.com/news/2240215890/Microsofts-Windows-Azure-network-is-a-massive-virtual-SDN
[3]https://www.vmware.com/products/nsx.html
[4]https://ru.wikipedia.org/wiki/Программно-определяемая_сеть
[5] http://www.axcess.com.my/simplify-business-innovative-google-cloud-services/

Линеаризуемость

Свойство *линеаризуемости* операций – работа с распределенной системой выглядит так, как если бы все операции были упорядочены и происходили атомарно

Облегчает разработку SDN приложений

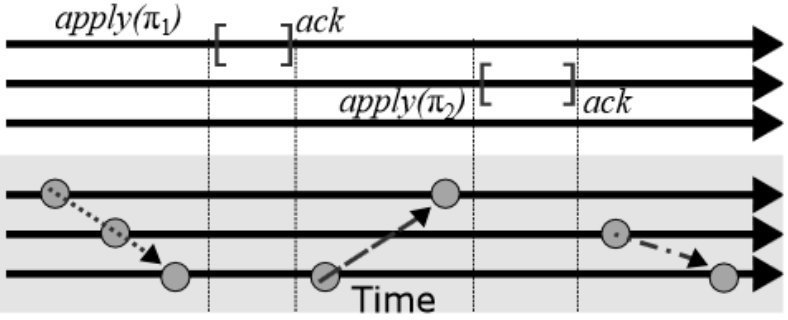
Запросы на установку политик



Перемещение пакетов по коммутаторам

≡

Линеаризованное исполнение

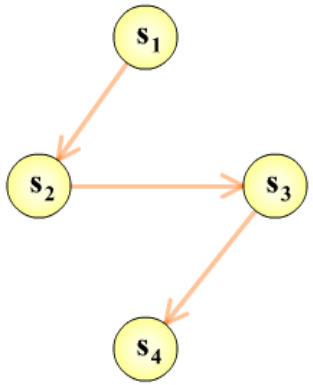


A distributed and robust sdn control plane for transactional network updates / M. Canini [и др.] // Computer Communications (INFOCOM), 2015 IEEE Conference on. — IEEE. 2015. — С. 190–198.

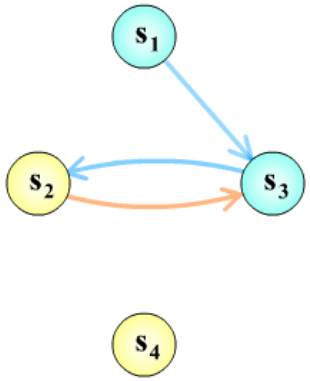
Сохранение конфигурации для пакетов

Свойство *сохранения конфигурации для пакетов* – на каждом из коммутаторов пакет обрабатывается согласно одной и той же конфигурации

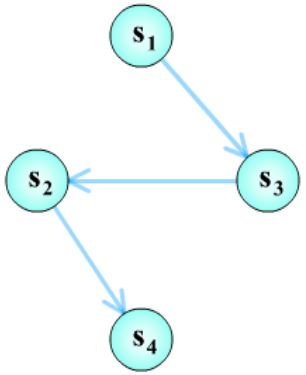
Без него возможно появление циклов и «черных дыр» в маршрутах пакетов



До обновления



В процессе обновления



После обновления

Установка политик: двухфазное обновление

- Конфигурациям сопоставляются теги (версии)
- Тегами помечаются пакеты
- Установка конфигурации C (тег $\boldsymbol{\tau}$) в 2 фазы:
  1. Добавление правила: пакеты с тегом $\boldsymbol{\tau}$ обрабатывать согласно конфигурации C
  2. Коммутаторы помечают новые пакеты в сети тегом $\boldsymbol{\tau}$

- Дает гарантию сохранения конфигурации для пакетов
- При получении пакета коммутатор уже знает соответствующую ему конфигурацию

Существующие протоколы распределенного контрольного слоя

- На сегодняшний день множество реализаций
  - Onix, Onos, HyperFlow, …
- Но мало предоставляющих гарантию линеаризуемости
  - Ravana (Zookeeper)
  - BFT-Light (BFT-SMaRt)
- Централизованные хранилища конфигурации, требуют ⩾3 коммуникационных шагов в общем случае
- *Нет* предоставляющих сохранение конфигурации для пакетов

«Generalized Paxos» [1]

- Преимущества

  2 коммуникационных шага в случае отсутствия конфликтов

- Недостатки
  - Дополнительный раунд восстановления в случае конкурентного предложения конфликтующих политик (например, используя алгоритм Classic Paxos)
  - Разные контроллеры получают политики в разном порядке
- Сложности

  Тяжеловесные сообщения

- Используется в проекте MDCC [2]

[1] Lamport, Leslie. *Generalized consensus and paxos*. Technical Report MSR-TR-2005-33, Microsoft Research, 2005.
[2] Kraska, Tim, et al. "MDCC: Multi-data center consistency." *Proceedings of the 8th ACM European Conference on Computer Systems*. ACM, 2013.

Цель и задачи

- Цели:

  Разработка протокола контрольного слоя с **гарантией согласованности уровня пакетов** и **линеаризуемости**, на основе **более оптимального хранилища** конфигурации по сравнению с аналогами
- Задачи:
  - Оптимизация Generalized Paxos с целью сокращения объема используемого трафика
  - Разработка алгоритма контроллера с гарантией сохранения конфигурации для пакетов
  - Реализация прототипа контроллера и его оценка

Требования к разрабатываемому протоколу

- Предоставляет интерфейс установки политик
  - Установка завершается успешно
  - Или отменяется (в случае конфликта)

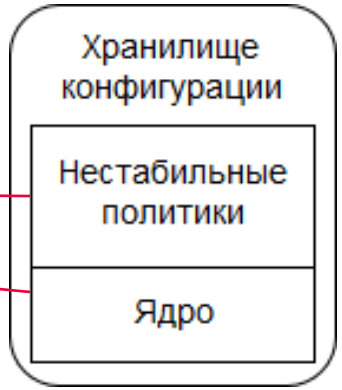
    В случае чего политика не влияет ни на один пакет
- Свойства:
  - Корректность
    - Конфигурация непротиворечива, линеаризуемость, сохранение конфигурации для пакетов…
  - Живучесть
    - Запрос на установку в конечном счете завершается
- Модель сети:

  Отказы контроллеров, задержка и потери сообщений

Методика улучшения Generalized Paxos

- Модификация Generalized Paxos – Gen-Paxos
- Сообщения содержат только предлагаемые политики
- Отказ от раундов
  - Предложенная политика отсылается немедленно
  - Двухуровневое хранение конфигурации сети

```
1  class ConfigStore:
2      ...
3      def total():
4          return core + unstable
5
6      On core change(newCore):
7          unstable = unstable.filter(newCore.agrees)
```



Структура хранилища конфигурации

Методика улучшения: псевдокод

Запрос на сохранение политики

```
propose(policy):
    Periodically until fixed:
        broadcast AcceptRequest(policy)
          to All
```

AcceptRequest

```
Any.On AcceptRequest(policy):
    If no recovery round is going:
        # returns ack or nack
        res <- addUnstable(policy)
        broadcast Accepted(policy, res)
          to All [to Target]
```

Accepted

Accepted

Результат сохранения

```
Target.On Accepted(policy, res):
    If got quorum of (policy, res):
        Fix(policy, res)
```

```
Coordinator.On Accepted(policy, res):
    If got at least one 'ack' and
      one 'nack' about policy:
        InitRecoveryRound(policy)
        # operates with Core level
```

Удаление политик

Проблема: Gen-Paxos требует, чтобы порядок применения политик не играл роли, но:

- $C \cdot Ins(P) \cdot Rem(P) \cdot Ins(P')$ - ок
- $C \cdot Ins(P) \cdot Ins(P') \cdot Rem(P)$ - не имеет смысла если P и P' конфликтуют

Решение: устанавливаемые политики нумеруются по поколениям, политики разных поколений не конфликтуют

- В хранилище добавляются такие $Ins^i(P)$, что

$$\forall Ins^i(P) \in C : conflict(P,P') \wedge Rem(P) \in C \Rightarrow j > i$$

- Если политика добавляется строго после удаления всех конфликтующих, конфликтов не будет

Установка политик на коммутаторы

- Выделенный контроллер устанавливает политики последовательно [группами] методом 2-фазного обновления
- Коммутаторы хранят множество установленных политик
- В случае отказа контроллера избирается новый

```
1   Once elected:
2       hanging, tag <- dataPlane.fetchPartiallyInstalledPolicies()
3       completeInstall(hanging, tag)
4
5       installed <- dataPlane.fetchInstalledPolicies()
6       nextTag <- size(installed)
7
8   Forever do:
9       policy <- pickUninstalled()  # or pickUninstalledMany()
10      dataPlane.install(policy, nextTag)
11      nextTag <- nextTag + 1
```

Свойства алгоритма

- Теорема: при возможности отказа контроллеров, полученный алгоритм удовлетворяет свойствам:
  - Валидность (конфигурация состоит из предложенных политик)
  - Устойчивость (установленные обновления конфигурации не пропадают)
  - Нетривиальность (политики не отвергаются беспричинно)
  - Согласованность (конфигурация непротиворечива)
  - Живучесть (в системе есть прогресс, пока имеется большинство работающих контроллеров)
  - Линеаризуемость
  - Сохранение конфигурации для пакетов
- Теоретическое доказательство

Реализация

- Был реализован прототип контроллера на языке Haskell [1]
- Модифицирована существующая[2] реализация коммутатора Open vSwitch

Два режима работы:

- С двухфазным обновлением – дает гарантии линеаризуемости и сохранения конфигурации для пакета
- Без двухфазного обновления – не дает таких гарантий

[1] https://github.com/Martoon-00/sdn-policy

[2] https://github.com/openvswitch/ovs

Тестирование Gen-Paxos

Для написания тестовых сценариев использовались библиотеки:

- Time-warp
  - Эмуляция ненадежной сети
  - Управление сценариями
- QuickCheck

  Автоматическая генерация параметров тестовых сценариев

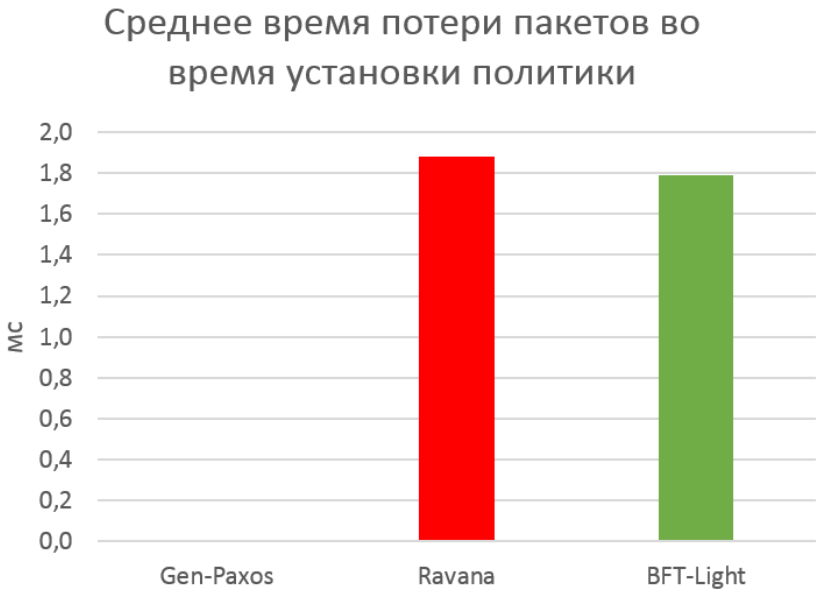
Симуляция SDN

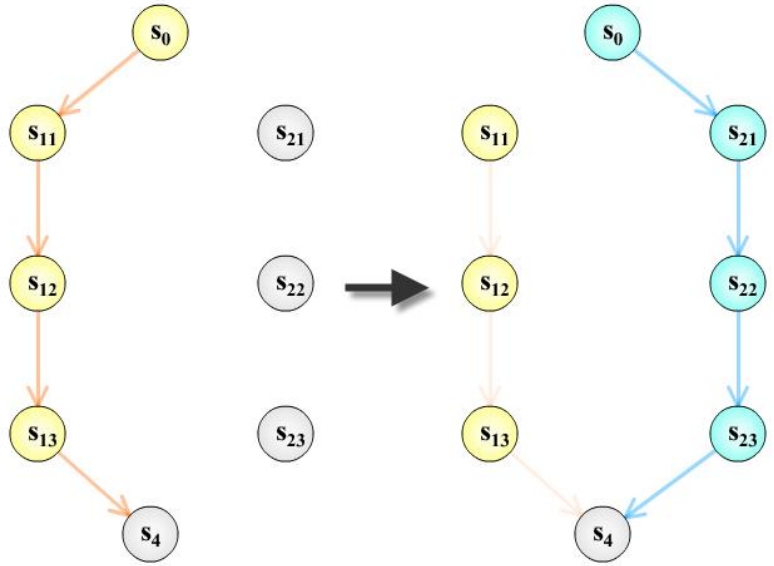
- Mininet – симуляция полноценной SDN
- Cbench – симуляция коммутаторов и входящих потоков пакетов
- На новые потоки пакетов контроллер отвечает установкой политики их обработки
- По одной виртуальной машине или пространству имен сети на контроллер

Оценка пользы сохранения конфигурации для пакетов (1)

Сетевая политика изменяет путь следования пакетов

Коммутатор может не иметь правила обработки пришедшего пакета

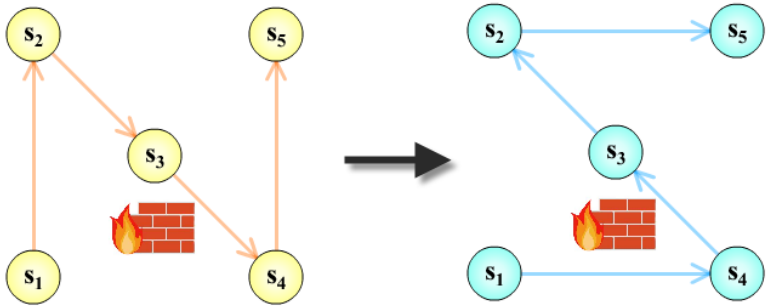




Оценка пользы сохранения конфигурации для пакетов (2)

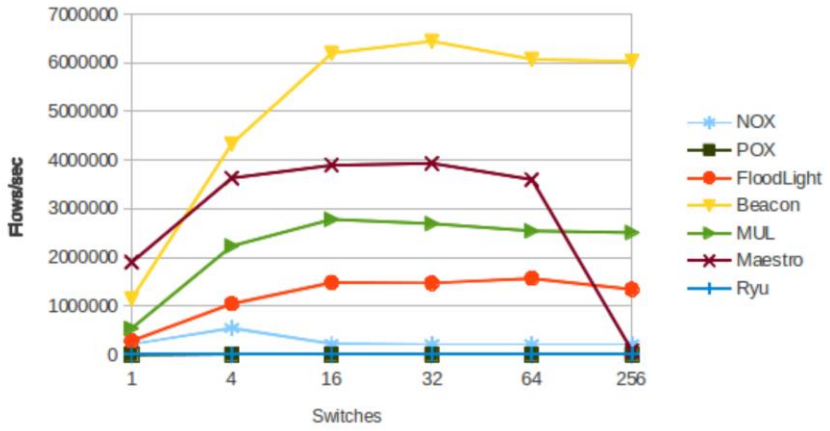
Сетевая политика изменяет путь следования пакетов

Пакеты должны проходить через брандмауэр



Производительность: сравниваемые величины

- Существующие платформы контроллера

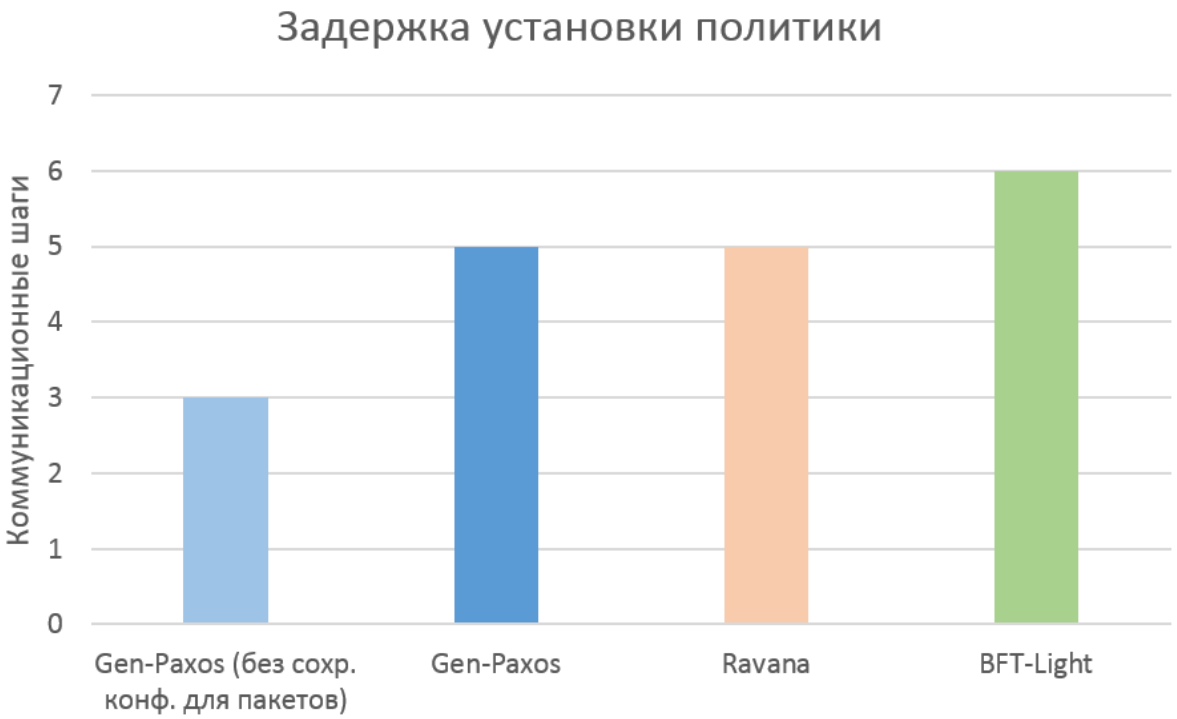


Сравнение платформ контроллера[1]

- Следует сравнивать величины, не зависящие от использованной реализации протокола

[1] Shalimov, Alexander, et al. "Advanced study of SDN/OpenFlow controllers." *Proceedings of the 9th central & eastern european software engineering conference in russia*. ACM, 2013

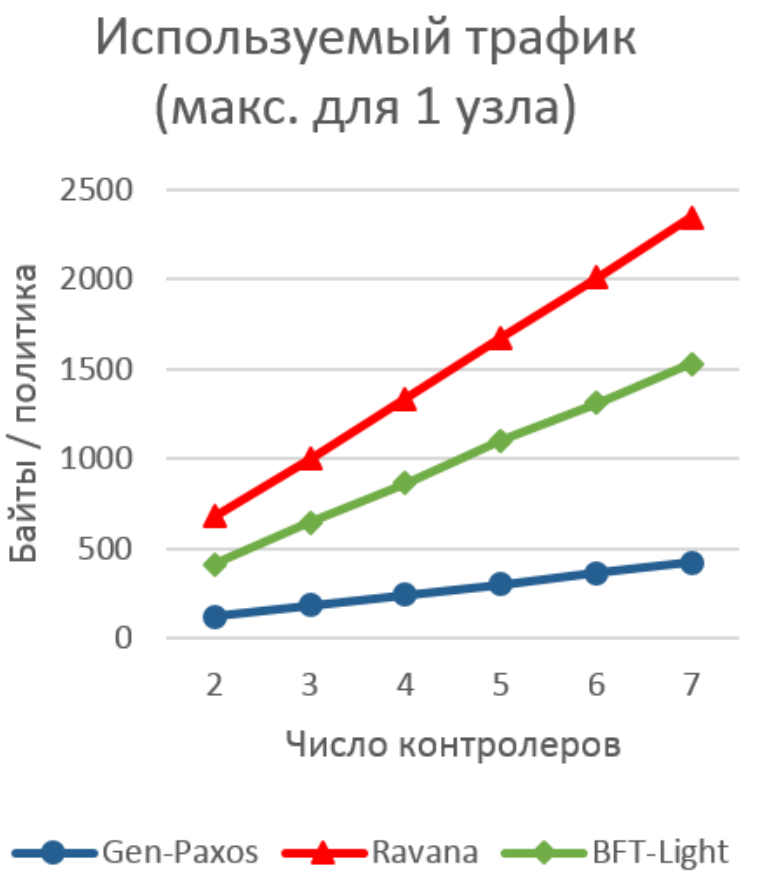
Анализ: задержка установки



- Двухфазное обновление применимо к любому алгоритму
- Вносит одинаковую задержку

  => Gen-Paxos более оптимален

Анализ: используемый контроллерами трафик



Результаты

- Был успешно реализован распределенный протокол SDN контроллера, предоставляющего свойства линеаризуемости сохранения конфигурации для пакетов
- Разработана модификация Generalized Paxos, требующая меньшее количество коммуникационных шагов для установки политики по сравнению с аналогами, и сравнимую пропускную способность
- Теоретическое доказательство гарантий корректности и живучести протокола
- Прототип проверен на множестве тестовых сценариев

Для дальнейшей работы

- Адаптация существующих улучшений Generalized Paxos
- Формальная верификация протокола
- Работоспособность в других моделях сети

- Планируется публикация

Спасибо за внимание