Московский государственный университет имени М.В.Ломоносова
Факультет Вычислительной Математики и Кибернетики
Кафедра автоматизации систем вычислительных комплексов

Синякова Марина Алексеевна

Анализ задержки потоков виртуального пласта в программно-конфигурируемой сети с помощью стохастического сетевого исчисления

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

Научный руководитель:
программист Степанов Евгений Павлович

Москва, 2019

Аннотация

Настоящая работа посвящена исследованию алгоритмов вычисления оценки задержки в сети с использованием стохастического сетевого исчисления для случая множественных потоков. В настоящее время продолжают активно разрабатываться методы обеспечения требований качества сервиса. Например, одной из ключевых особенностей разрабатываемых 5G сетей является выделение пользователям со схожими требованиями качества сервиса отдельного виртуального пласта, в котором передаются данные этих пользователей. Настройка приоритетов очередей на коммутаторах, используемых виртуальным пластом, может позволить удовлетворить требования по задержке. Однако для корректной настройки приоритетов необходимо оценивать получаемую задержку для всех пользовательских потоков. Для этого был разработан алгоритм, вычисляющий оценку задержки транспортного потока внутри виртуального пласта с использованием стохастического сетевого исчисления и с учетом приоритетов виртуальных пластов. Предложенный алгоритм реализован на языке Python3 и сводит задачу оценки задержки к задаче линейного программирования. Было проведено экспериментальное исследование разработанного алгоритма и выявлены зависимости времени работы алгоритма и количества ограничений задачи линейного программирования от количества узлов в топологии, от степени связности узлов в топологии и от количества потоков данных внутри виртуального пласта.

Содержание

1 Введение

1.1 Цель работы

Разработать метод для оценки задержки потоков виртуального пласта в программно-конфигурируемой сети (ПКС) с помощью стохастического сетевого исчисления (ССИ).

1.2 Актуальность работы

В настоящее время в связи с возросшими потребностями к качеству сервиса идет активное усовершенствование существующих сетевых технологий и разработка новых. Частным примером является разработка сетей нового поколений 5G [1]. Одной из основных особенностей является распределение сетевых ресурсов от потребностей пользователей. Например, при просмотре видео в формате 4K [2] скорость передачи данных является критической величиной и должна быть достаточно велика, чтобы видео воспроизводилось непрерывно. Возможен просмотр видео на разноформатных устройствах, что так же требует различного качества сервиса. Например, разрешение на телефоне [3] значительно отличается от разрешения на телевизоре, следовательно требуется передавать меньший объем данных в единицу времени. Кроме того, могут быть разные условия подключения к Интернет-провайдеру, где пользователь может быть подписан на недостаточную для просмотра видео в высоком разрешении скорость, что может быть обусловлено более дешевым тарифом. При передачи трафика в режиме реального времени задержка становится критической величиной. Например, при дистанционной хирургии трансляция видео должна происходить с минимально возможной задержкой, так как необходимо мгновенное взаимодействие между врачами. А при отправке сообщений электронной почты задержка является менее значимой величиной, так как нам не требуется мгновенный отклик, как, например, при обмене потоками данных между крупными биржами. Как видно из вышеперечисленных примеров, для разных типов приложений необходимо свое качество сервиса - ограничение на доступную пропускную способность каналов передачи данных и требуемая максимальная задержка. Таким образом, некоторым пользователям или под определенный вид трафика необходимо выделять отдельные ресурсы сети - виртуальные пласты [4].

В настоящей работе рассматривается только проблема обеспечения требуемой максимальной задержки. Оценка задержки будет даваться с определенной вероятностью, что делает полученную оценку менее строгой, но более приближенной к реальной работе сети. Каждому виртуальному пласту выдается некоторая очередь с заданным приоритетом на коммутаторах для передачи данных. Приоритет очереди равен приоритету виртуального пласта. Вариация задержки происходит при помощи изменения приоритета виртуального пласта. Увеличивая приоритет виртуального пласта, уменьшается задержка передачи данных внутри него. В связи с тем, что ограничения на задержку у всех виртуальных пластов разные, то и приоритеты очередей у них должны быть различны. Если ограничения на задержку одинаковые и данные виртуальные пласта

нельзя объединить в связи с различными требованиями на пропускную способность канала, то этим виртуальным пластам выдаются соседние приоритеты. Но как проверить, что выбранного приоритета достаточно для обеспечения качества сервиса? Для этого для выбранного приоритета и множества потоков необходимо уметь вычислять оценку задержки виртуального пласта - максимальную задержку по всем потокам.

Есть множество методов для вычисления оценки задержки с использованием сетевого исчисления[5], но к задаче оценки задержки потоков в виртуальном пласте они напрямую не применимы. Так как для виртуального пласта нету метода построения кривых нагрузки и обслуживания, без которых невозможно применение многих метод вычисления оценки задержки. Также для виртуального пласта нету средства для автоматического расчета оценки задержки исходя из этих кривых.

1.3 Постановка задачи

Для достижения указанной цели необходимо решить следующие подзадачи:

1. Провести обзор алгоритмов оценки задержки в ССИ с целью выбора алгоритма для использования в случае множественных потоков данных.

2. Предложить метод получения кривой нагрузки из собранных статистических данных по потокам.

3. Предложить метод получения кривой обслуживания исходя из приоритетов виртуальных пластов и заданных ограничений на доступную пропускную способность каналов передачи данных.

4. Реализовать предложенные методы и провести экспериментальное исследование для определения сложности работы всего алгоритма и выявления зависимости времени работы алгоритма от размера топологии и количества потоков.

1.4 Подход к решению

В настоящей работе под термином "виртуальный пласт" понимаем совокупность сетевых ресурсов, предназначенных для передачи данных с определенным качеством сервиса. Наиболее просто реализовать выделение виртуальных пластов можно в программно-конфигурируемых сетях (ПКС)[6], так как контролер имеет доступ ко всем ресурсам сети и может управлять ими. Под приоритетом виртуального пласта понимаем приоритет очереди на коммутаторах, которая выделена для передачи данных рассматриваемого виртуального пласта. Считаем приоритеты виртуальных пластов на всех коммутаторах одинаковы. Возможно и другое задание приоритетов в связи с тем, что на разных коммутаторах могут быть заданы разные приоритеты, но для удобства вычислений рассматривается наиболее простой вариант. Так как количество виртуальных пластов может быть произвольным, в динамике могут появляться новые виртуальные пласты и

удаляться старые, то необходима гибкость в настройке очередей коммутаторов, которую проще всего могут предоставить лишь программные коммутаторы. Поэтому в настоящей работе рассматривается взаимодействие только с программными коммутаторами, которые используются при построении ПКС.

Под термином “качество сервиса” понимается набор параметров, который характеризует производительность сетевого соединения. Набор параметров, рассматриваемых в настоящей работе, включает:

- пропускную способность – часть ширины полосы пропускания канала, предоставляемая отдельному виртуальному пласту;
- сквозную задержку – время, необходимое на доставку данных от точки входа данных в топологию сети до точки выхода данных из топологии.

Рассматриваются только сети прямого распространения. Под сетью прямого распространения понимаем такую топологию, в которой пакеты распространяются только в одном направлении - от "входов" к "выходам". На рисунке 1 изображен пример данной топологии.

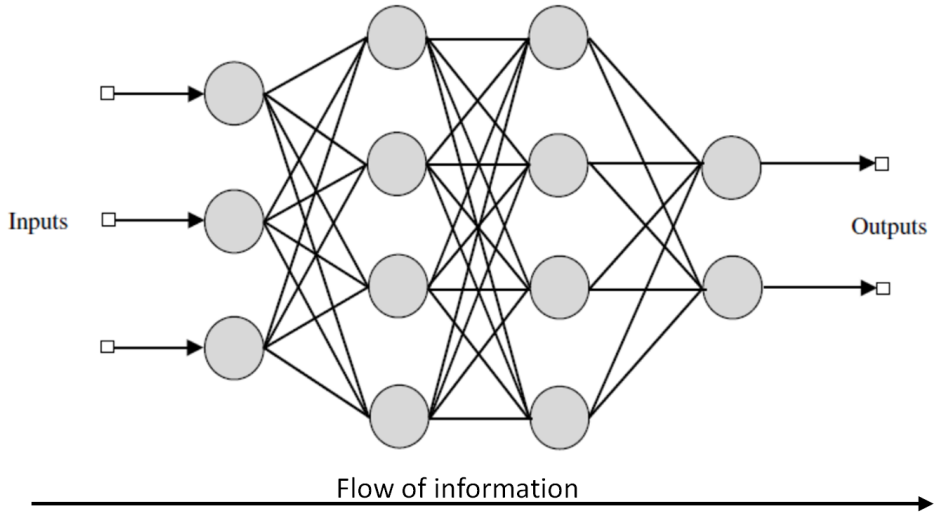


Рис. 1: *Сеть прямого распространения*

В настоящей работе используется математический аппарат сетевого исчисления для нахождения оценок задержки. Существует два вида сетевого исчисления: детерминированное сетевое исчисление и стохастическое сетевое исчисление. В связи с тем, что детерминированное сетевое исчисление дает оценку задержки в наихудшем случае, значение которой может быть недостижимо или выполнимо крайне редко. Поэтому в данной работе будет использовано стохастическое сетевое исчисление [7], которое позволяет вычислять оценку задержки, приближенную к работе реальной сети. Оценка будет верна с вероятностью, которую пользователь будет задавать с остальными ограничениями для создания виртуального пласта.

Кривую нагрузки можно задавать напрямую, однако пользователю сделать это достаточно трудно. Поэтому выходом из этой проблемы является размещение пользовательских потоков в некоторый низко приоритетный виртуальный пласт и сбор статистики по количеству поступающих в топологию байт пользовательских потоков. Поэтому пользовательский трафик в настоящей работе задается при помощи статистических данных, на основании которых необходимо построить кривую нагрузки. Статистические данные описывают количество байт, поступивших за 1 секунду. Для упрощения вычислений в данной работе будем использовать только Пуассоновское распределение для генерации статистических данных, отражающих входной трафик.

1.5 Структура работы

В главе 2 описываются основные понятия стохастического сетевого исчисления. В главе 3 приводится обзор алгоритмов вычисления оценки задержки для множественных потоков. В главе 4 содержится подробное описание работы выбранного метода вычисления оценки задержки. Глава 5 описывает работу всего алгоритма по вычислению оценки задержки внутри виртуального пласта. Глава 6 содержит результаты экспериментальных исследований. Глава 7 подводит заключение настоящей работы.

2 Введение в стохастическое сетевое исчисление

В данном разделе введем определения кривой нагрузки и кривой обслуживания, задавая их с использованием стохастического сетевого исчисления. Так же получим оценку задержки для работы одного потока, которая была бы наиболее приближена к реальной работе сети.

Введем основные определения:

- *Функция прибытия* $A(t)$ - описывает зависимость суммарного количества данных, поступивших на обработчик, от времени.
- *Функция обработчика* $S(t)$ – зависимость количества переданных данных от времени.
- *Задержка (delay)* $W(t)$ – время прохождения через обработчик той порции данных, которая поступила на него в момент времени t.
- *Отставание (backlog)* $B(t)$ – выражает количество данных, находящихся «внутри» обработчика.
- *Кривой обслуживания* (variable capacity node) обработчика s называется такая функция β_s, что для каждого временного интервала длины $\tau \geq 0$, величина $\beta_s(\tau)$ не превышает объёма данных, который s способен обработать в течение этого интервала:

  $$\forall t : \forall \tau : S(t+\tau) - S(t) \geq \beta_s(\tau)$$

  Кривая производительности β_s позволяет построить нижнюю (наихудшую с точки зрения качества сервиса) оценку функции обработчика S.
- *Кривой нагрузки* (arrival curve) для потока с функцией прибытия A называется такая функция α, что для каждого временного интервала длины τ количество переданных в течение него данных этого потока не превышает величины $\alpha(\tau)$:

  $$\forall t : \forall \tau : A(t+\tau) - A(t) \leq \alpha(\tau)$$

  Кривая нагрузки ограничивает сверху скорость поступления данных потока на каждом интервале заданной длины
- *Период занятости* - временной интервал, в течение которого отставание $b(t)$ обработчика строго положительно.

Теорема 3: (*Оценка задержки*) Пусть поток данных с функцией прибытия A, ограниченный кривой нагрузки α, обслуживается обработчиком с кривой сервиса β по дисциплине FIFO. Тогда значение задержки обслуживания этого потока не превышает

горизонтального отклонения между кривыми прибытия α и сервиса β:

$$\forall t \in \mathbb{R} : \omega(t) \leq h(\alpha, \beta) = \sup_t \{\inf\{\tau \geq 0 | \ \alpha(t) \leq \beta(t+\tau)\}\}$$

Представим ограничение для задержки в виде $P[W(t) > \omega] \leq \varepsilon'$, где ε' вероятность того, что в реальной сети задержка нарушит построенную нами оценку ω. Данную оценку можно получить из моделей EBB (exponentially bounded burstiness) и EBF (exponentially bounded functions). Для фиксированного значения τ модель EBB будет иметь вид:

$$P[A(\tau, t) > \rho(t-\tau) + b] \leq \varepsilon(b), \quad \varepsilon(b) = \alpha e^{-\theta b} \tag{1}$$

где $A(\tau, t)$ – входной трафик, пришедший в момент времени от τ до t ; ρ – скорость прибытия пакетов, которая зависит от свободного параметра $\theta \geq 0$; b – размер всплеска; $\varepsilon(b)$ – функция, обозначающая вероятность, с которой допускается нарушение построенной оценки для случайной величины $A(\tau, t)$ и зависит от параметра b, коэффициент $\alpha \geq 0$. Модель называется EBB, так как при увеличении b вероятность нарушения оценки экспоненциально уменьшается. Графическая интерпретация модели EBB изображена на рисунке 2.

Чтобы применить соотношения, справедливые для детерминированного сетевого исчисления, для получения оценки задержки, необходимо, чтобы можно было взять любое $\tau \in [0, t]$. Поэтому обобщение неравенства (1) для всех значений τ представимо в виде:

$$P[\exists \tau \in [0, t] : \ A(\tau, t) > \rho'_A(t-\tau) + b_A] \leq \varepsilon'(b_A), \tag{2}$$

где $\rho'_A = \rho_A + \delta$, а δ – калибровочный параметр, $\varepsilon'(b_A)$ – вероятность, с которой может нарушаться построенная оценка. Величины ρ_A и b_A характеристики трафика.

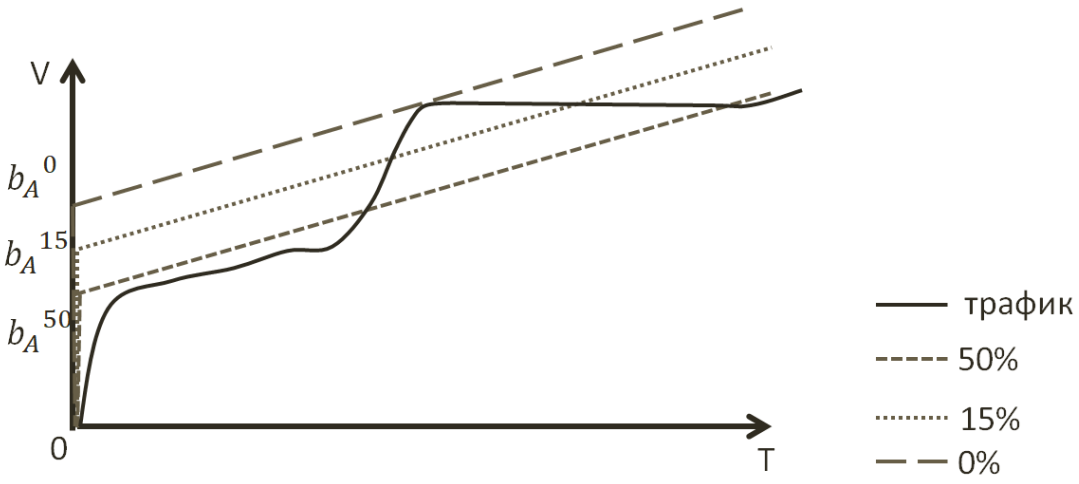


Рис. 2: *Графическое представление зависимости вероятности в модели EBB от величины* b_A.

Для обработчика применима модель EBF (exponentially bounded functions), которая

аналогична модели EBB и имеет вид:

$$P[S(\tau,t) < \rho(t-\tau) - b] \ \leq \ \varepsilon(b), \quad \varepsilon(b) \ = \ \alpha e^{-\theta b} \tag{3}$$

$$P[\exists \tau \in [0,t] \ : \ S(\tau,t) \ < \ \rho'_S(t-\tau) + b_S] \ \leq \ \varepsilon'(b_S), \tag{4}$$

Исходя из утверждений (2) и (4) получим диапазон выбора калибровочного параметра и основное ограничение на скорости трафика и обработчика. Мы вводили обозначения как $\rho'_A = \rho_A + \delta$ и $\rho'_S = \rho_S - \delta$.

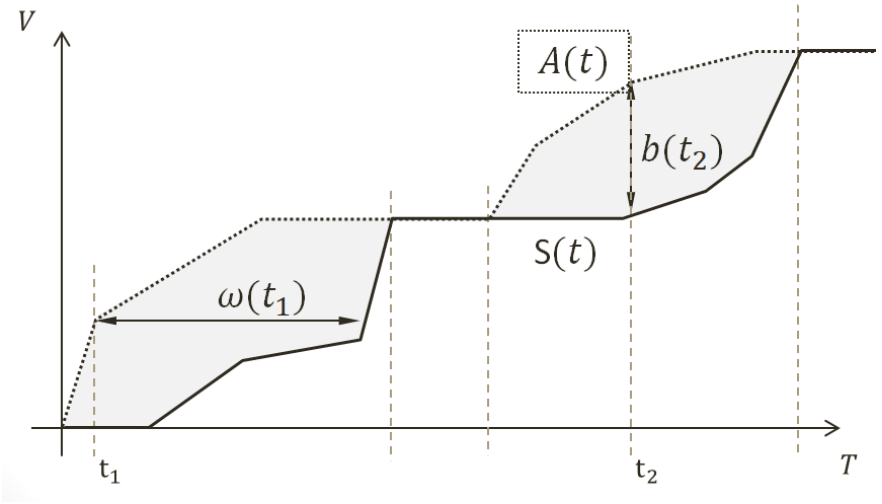


Рис. 3: *Визуализация значений задержки и отставания.*

На рисунке 3 задержка – максимальное отклонение между двумя прямыми по горизонтали. Так как кривая нагрузки и кривая обслуживания имеют вид кусочно-линейных функций, то нам для получения конечной оценки задержки необходимо пересечение данных прямых или сохранение параллельности, то есть необходимо выполнение условия $\rho_A \leq \rho_S$. Исходя из необходимого условия выполнения оценок, получаем диапазон для выбора δ, $\delta \in [0, \frac{\rho_A - \rho_S}{2}]$.

Модели EBB и EBF хороши для расчетов, но в большинстве случаев, такие оценки получить трудно, так как случайные величины (поступление трафика и работа обработчиков) в основном задаются распределением. Для работы с таким представлением случайной величины введем модель MGF (moment generation function) – производящая функция моментов.

Производящей функцией моментов (MGF) называется функция от случайной величины X с распределением $\mathbb{F}^X$, имеющая вид: $M_X(\theta) = \mathbb{E}[e^{\theta X}]$, где θ – случайный параметр.

Введем аффинную оценку для MGF трафика:

$$\mathbb{E}[e^{\theta A(\tau,t)}] \ \leq \ e^{\theta(\rho(t-\tau)+\sigma)} \tag{5}$$

Аффинная оценка MGF функции обслуживания:

$$\mathbb{E}[e^{-\theta S(\tau,t)}] \;\leq\; e^{-\theta(\rho(t-\tau)-\sigma)} \tag{6}$$

Используя неравенство Чернова $P[X \geq x] \leq e^{-\theta x}\mathbb{E}[e^{\theta}X]$ можно из аффинной оценки MGF получить модели EBB и EBF для трафика и обработчика соответственно.

Используя выше рассмотренные подходы, авторы статьи [10] рассчитали вероятности $\varepsilon'_A(b) = \dfrac{e^{\theta\sigma}}{1-e^{-\theta\delta}}e^{-\theta b}$ и $\varepsilon'_S(b) = \dfrac{e^{\theta\sigma}}{1-e^{-\theta\delta}}e^{-\theta b}$ для трафика и обработчика соответственно. $\varepsilon'(b) = \varepsilon'_A(b) + \varepsilon'_S(b)$ - это общая вероятность, с которой допускается нарушение полученной оценки для характеристик сети. Инвертированием формул вероятностей $\varepsilon'_A(b)$ и $\varepsilon'_S(b)$ получаем основные формулы для вычисления параметров b_A и b_S, которые в дальнейшем будут использованы для вычислений задержки в сети.

$$b_A \;=\; \sigma_A - \frac{1}{\theta}\left(\ln\frac{\varepsilon'}{2} + \ln\left(1-e^{-\theta\delta}\right)\right) \tag{7}$$

$$b_S \;=\; \sigma_S - \frac{1}{\theta}\left(\ln\frac{\varepsilon'}{2} + \ln\left(1-e^{-\theta\delta}\right)\right) \tag{8}$$

3 Обзор алгоритмов вычисления оценки задержки

Для вычисления оценки задержки для виртуального пласта необходимо уметь вычислять оценку задержки, когда в сети предается более чем один поток данных. В сетевом исчислении часто сначала вычисляют свертку [8] обработчиков, а затем вычисляют оценку задержку для потока, проходящего через один обработчик.

Введем основные понятия и теоремы, которые необходимы для применения ниже рассмотренных методов.

Определение: ***Система обработчиков*** - связное множество обработчиков, соединенное между собой каналами передачи данных.

Определение: ***Кросс-трафик*** - трафик потоков, маршруты которых пересекаются на рассматриваемом обработчике с маршрутом рассматриваемого потока.

Определение: ***Свертка двух функций*** - такая операция $\otimes$, что:

$$(f \otimes g)(t) = \inf_{0 \le s \le t} \{f(s) + g(t-s)\}$$

Теорема 1: (*Конкатенация системы обработчиков*) Рассмотрим поток, проходящий через тандем систем S_1 и S_2. Предположим, что S_i предлагает кривую обслуживания $\beta_i,\ i = 1, 2$ к рассматриваемому потоку. Тогда объединение двух систем дает кривую обслуживания $\beta_1 \otimes \beta_2$.

Теорема 2: (*Остаточная кривая обслуживания при произвольном мультиплексировании*) Рассмотрим узел, мультиплексирующий два потока 1 и 2 произвольным образом. Пусть β - минимальная кривая обслуживания, гарантируемая узлом; α_2 - кривая нагрузки для потока 2. $\beta_1 = [\beta - \alpha_2]_+$ является кривой обслуживания для потока 1. $\beta - \beta_1$ называют остаточной кривой обслуживания для рассматриваемого потока 2.

Но как учитывать кросс-трафик и как разделять ресурсы между потоками, проходящими через один обработчик? Было рассмотрено несколько методов оценки задержки, которые учитывают кросс-трафик, по следующим критериям.

Критерии обзора:

1. *Обработчики работают в режиме FIFO.*
   Так как одним из критериев применения сетевого исчисления является обслуживание очереди в коммутаторах по дисциплине FIFO [9], то данное ограничение накладывается и на метод вычисления оценки задержки.

2. *Точность оценки.*
   Метод должен позволяет вычислять оценку задержку отдельно для каждого потока. Так как при вычислении оценки задержки для агрегированного потока трафика накладываются слишком жесткие ограничения для каждого потока. Эти ограничения дают большую погрешность на оценку задержки, и полученное значение

является сильно завышенным относительно реального состояния сети. Оценка задержки для каждого потока должна быть максимально приближена к реальному значению в сети, чтобы полученное значение отражало реальное состояние сети.

3. *Наименьшая вычислительная сложность алгоритма.*
   Так как вычисления будут проводиться для большой топологии и многих потоков, выбранный метод должен вычислять оценку задержки с максимальной точностью, тратя на это минимальное количество ресурсов.

3.1 Анализ общего потока (TFA - Total Flow Analysis)

Идея этого метода состоит в том, чтобы сначала оценить задержку для общего потока трафика (трафик всех потоков, проходящий через данный обработчик) для одного обработчика, а затем суммировать полученные оценки для интересующего нас потока. Если использовать произвольное мультиплексирование потоков, то оценка задержки для каждого узла должна быть вычислена как максимальный период занятости на узлах, потому что общий поток в этом методе рассматривается не как FIFO.

Рассмотрим данный метод на примере, изображенном на рисунке 4.

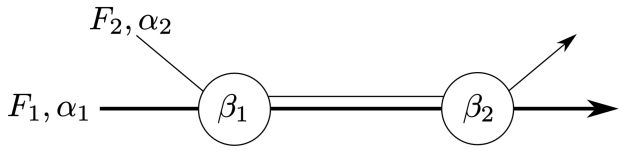


Рис. 4: *Простая модель сети из двух узлов и двух потоков*

Задержка для первого потока F_1 будет равна сумме задержек на каждом узле: $d^{TFA} = d_1 + d_2$, что подробнее описано в [10].

Данный метод не подходит для решения поставленной задачи, так как не выполнен первый критерий обзора: обработчики обслуживают трафик не в модели FIFO.

3.2 Анализ раздельных потоков (Separated Flow Analysis - SFA)

Основным недостатком TFA является то, что он не разделяет услуги, предоставляемые потокам, то есть не использует теорему конкатенации, которая обеспечивает явное преимущество перед аддитивными методом вычисления оценки задержки. Метод SFA вместо обработки общего потока сначала определяет кривую обслуживания, которое будет предоставлено рассматриваемому потоку на каждом узле, а затем применяет теорему конкатенации к тандему кривых обслуживания. Разделение основано на теореме об остаточной кривой обслуживания при произвольном мультиплексировании. В отличие от метода TFA, в методе SFA предполагается, что каждый поток обслуживается по дисциплине FIFO. Оценка задержки 1 потока для сети, изображенной на

рисунке 4, при помощи SFA может быть получена следующим образом:

$$d^{SFA} = h(\alpha_1, [\beta_1 - \alpha_2]^+ \otimes [\beta_2 - (\alpha_2 \oslash \beta_1)]^+)$$

При вычислении оценки задержки для каждого потока в рассматриваемом примере идет дважды учет второго потока, что уменьшает точность вычисления данного метода, описанного подробнее в [10]. Тем самым второй критерий обзора выполнен не оптимально.

3.3 SFA с учитыванием только одного вхождения мультиплексирования (Pay Multiplexing Only Once SFA - PMOO-SFA)

Один из недостатков SFA - это зависимость от порядка применения теоремы 2 (об остаточной кривой обслуживания при произвольном мультиплексировании) и теоремы 1 (о конкатенации системы обработчиков). Возможны две последовательности применения этих теорем. Например, в схеме на рисунке 4 сначала могла быть применена свертка двух узловых кривых обслуживания, а затем применено произвольное мультиплексирование к результирующей системе с одним узлом. Идея данного метода состоит в том, чтобы сначала объединить отдельные системы, чтобы иметь возможность платить за мультиплексирование с другими потоками только один раз. Тем самым при применении PMOO-SFA для случая на рисунке 4 получаем оценку:

$$d^{PMOO} = h(\alpha_1, [\beta_1 \otimes \beta_2 - \alpha_2]^+)$$

Для работы с ацикличным пересекающимся трафиком сложность данного алгоритма полиномиальна, что доказано в статье [11].

3.4 Линейное программирование

В этом разделе представлен альтернативный метод для вычисления оценки задержки в сетях прямого распространения (feed-forward сети) с произвольным мультиплексированием узлов. Этот метод состоит в формулировании задачи оптимизации, основанной на знании об ограничениях прихода других потоков трафика и гарантиях обслуживания, предоставляемых каждым узлом.

Кривые нагрузки и кривые обслуживания выражаются с помощью кумулятивных функций: кумулятивная функция $F(t)$ подсчитывает общий объем данных, который достиг некоторого условия до момента времени t. В частности для трафика это количество бит, поступивших в сеть к моменту времени t, а для обработчиков - это количество бит, отправленным данным обработчиком в сеть к моменту времени t.

Ограничение на количество поступающих бит в систему, накладываемое кривой нагрузи, обозначают как $F(t) - F(s) \leq \alpha(t - s)$. Это означает, что число бит, поступающих между временем s и t, не больше, чем $\alpha(t - s)$. В качестве кривой нагрузки будем

рассматривать линейные кривые вида $\alpha_{\sigma_A,\rho_A} = \rho_A t + \sigma_A$, где ρ_A - скорость прибытия пакетов, а σ_A - максимальное число пакетов, которые могут одновременно поступать в систему (всплеск трафика).

Ограничение на количество бит, отправленных в систему обработчиком, накладываемое кривой обслуживания, обозначают как $F(t) - F(s) \geq \beta(t - s)$. Разница между выходным трафиком в момент времени t_π и $t_{j\pi}$ не меньше кривой обслуживания на этом интервале. Причем кривая обслуживания имеет вид $\beta_{\sigma_S,\rho_S} = \rho_S t - \sigma_S$, где ρ_S - скорость работы обработчика, а σ_S - задержка на обработку трафика.

Для вычисления оценки задержки задаются ряд ограничений на время, входной трафик, а так же задаются соотношения между обработчиками. Целевой функцией для задачи линейного программирования является максимальная значение разности между временем входа трафика в сеть и временем его выхода ($max(t_\emptyset - u)$), что будет подробнее описано в пункте 4.5.

Сложность данного алгоритма полиномиальная [9].

3.5 Выводы

Алгоритм TFA не удовлетворяет первому и ключевому критерию о работе обработчиков в режиме FIFO, тем самым делает невозможным использование сетевого исчисления. Алгоритм SFA не полностью удовлетворяет второму критерию, так как несмотря на то, что он разделяет агрегированный трафик на потоки, метод дает менее точную оценку задержки в сравнении с тем же PMOO-SFA. После рассмотрения методов вычисления оценки задержки были выбраны 2 алгоритма, удовлетворяющие всем критериям: PMOO-SFA и линейное программирование. Так как была проведена исследовательская работа с использованием детерминированного сетевого исчисления [12], в которой было показано, что SFA-тип методов проигрывает линейному программированию, то для данной работы был выбран именно метод линейного программирования.

4 Построение ограничений для метода линейного программирования

Исходя из всех критериев был выбран метод линейного программирования вычисления оценки задержки для множественных потоков. Рассмотрим подробнее данный метод. Существует ряд ограничений, по которым строятся неравенства для линейного программирования: временные ограничения (пункт 4.1), траекторные ограничения (пункт 4.2), ограничения для входного трафика (пункт 4.3), ограничения для обработчиков (пункт 4.4) и ограничения для построения оценки задержки (пункт 4.5).

Под траекторией будем понимать последовательность обработчиков, через которые проходит поток трафика.

Рассмотрим поток i, маршрут которого заканчивается на обработчике j. Пусть Π - множество всех траекторий в сети, заканчивающихся на обработчике j. Так же туда входит $\emptyset$. Введем переменные, которые будем использовать в данном методе:

- t_π - время начала периода отставания на траектории $\pi \in \Pi$. Под обозначением $t_\emptyset$ будем понимать момент, в который из системы выходит пакет с наибольшей задержкой. Для траектории $j\pi \in \Pi, t_{j\pi} = start_j(t_\pi)$ - начало периода отставания на обработчике j, который является началом траектории $j\pi$ и содержит время t_π.
- $F_i^{(j)}(t_\pi)$ для всех $j \in i, j \neq 0$ - накопительная функция для потока i на обработчике j.
- $F_i^{(0)}(t_\pi)$ - накопительная функция, соответствующая количеству байт, поступивших в поток i до момента времени t_π.

Рассмотрим топологию, изображенную на рисунке 5, и для второго потока запишем все необходимые ограничения для вычисления оценки задержки.

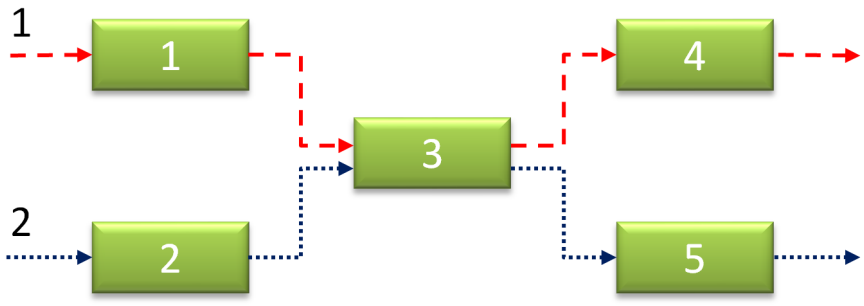


Рис. 5: *Топология сети с двумя потоками.*

4.1 Временные ограничения

Для каждого потока i строится множество Π^i, которое содержит все t_π такие, что траектория π является частью маршрута потока i. Данное множество необходимо упо-

рядочить в соответствии с некоторыми ограничениями:

- Для всех траекторий $j\pi \in \Pi \Rightarrow t_{j\pi} \leq t_\pi$.
- Для всех траекторий, выходящих из одного обработчика ($j\pi_1,\ j\pi_2 \in \Pi$), верно соотношение $t_{j\pi_1} < t_{j\pi_2} \Rightarrow t_{j\pi_1} \leq t_{\pi_1} \leq t_{j\pi_2} \leq t_{\pi_2}$. Так же необходимо учитывать ситуацию $t_{j\pi_1} = t_{j\pi_2}$, тогда возможны два варианта соотношений: $t_{j\pi_1} = t_{j\pi_2} \leq t_{\pi_1} \leq t_{\pi_2}$ и $t_{j\pi_1} = t_{j\pi_2} \leq t_{\pi_2} \leq t_{\pi_1}$.

Таким образом, для каждого потока строятся всевозможные комбинации времен соответствующих началам периодов отставания t_π, удовлетворяющие вышеописанным ограничением и отражающих траекторию потока. В зависимости от расстановки знаков неравенств получаются различные ограничения на переменные t_π, что приводит к нескольким вариантам задач линейного программирования. Для рассматриваемого примера получаем два варианта временных ограничений:

$$1:\ t_{235} \leq t_{135} \leq t_{35} \leq t_5 \leq t_\varnothing$$

$$2:\ t_{235} \leq t_{35} \leq t_{135} \leq t_5 \leq t_\varnothing$$

Для записи дальнейших неравенств выберем ограничение 1.

4.2 Траекторные ограничения

- *Начало периода отставания:* для всех обработчиков, принадлежащих потоку i ($\forall j \in i$), и для всех траекторий из этого потока ($\forall j\pi \in \Pi$) верно $F_i^{(pred(j))}(t_{j\pi}) = F_i^{(j)}(t_{j\pi})$.

  Для потока 2 и ограничения на время 1 равенство достигается в точках t_{235}, t_{35}, t_5:

$$F_2^{(0)}(t_{235}) = F_2^{(2)}(t_{235}),$$

$$F_2^{(2)}(t_{35}) = F_2^{(3)}(t_{35}),$$

$$F_2^{(3)}(t_5) = F_2^{(5)}(t_5)$$

- *Ограничения потока:* для всех потоков i и $j\pi \in \Pi,\ j \in i$ добавляем ограничения $F_i^{(0)}(t_{j\pi}) \geq F_i^{(j)}(t_{j\pi}),\ F_i^{(0)}(t_\pi) \geq F_i^{(j)}(t_\pi)$.

  Из точки t_{35} следует пара неравенств:

$$F_2^{(0)}(t_{35}) \geq F_2^{(2)}(t_{35}),\ F_2^{(0)}(t_5) \geq F_2^{(3)}(t_5)$$

  Из точки t_5 следует пара неравенств:

$$F_2^{(0)}(t_5) \geq F_2^{(3)}(t_5),\ F_2^{(0)}(t_\varnothing) \geq F_2^{(5)}(t_\varnothing)$$

Так как неравенства $F_2^{(0)}(t_5) \geq F_2^{(3)}(t_5)$ повторяются, то будет записано только одно.

- *Неубывающие функции:* для всех потоков и обработчиков, принадлежащих рассматриваемому потоку ($j \in i$), и $\pi_1, \pi_2 \in \Pi^i$:

$$t_{\pi_1} = t_{\pi_2} \;\Rightarrow\; F_i^{(j)}(t_{\pi_1}) = F_i^{(j)}(t_{\pi_2})$$
$$t_{\pi_1} \leq t_{\pi_2} \;\Rightarrow\; F_i^{(j)}(t_{\pi_1}) \leq F_i^{(j)}(t_{\pi_2}).$$

Для временного ограничения 1 верны соотношения для входного трафика в том же порядке, что и соотношения на время:

$$F_2^{(0)}(t_{235}) \leq F_2^{(0)}(t_{135}) \leq F_2^{(0)}(t_{35}) \leq F_2^{(0)}(t_5) \leq F_2^{(0)}(t_\emptyset)$$

Для каждого обработчика выполнены неравенства:

$$F_2^{(2)}(t_{235}) \leq F_2^{(0)}(t_{35})$$

$$F_2^{(3)}(t_{235}) \leq F_2^{(3)}(t_5)$$
$$F_2^{(5)}(t_5) \leq F_2^{(5)}(t_\emptyset)$$

4.3 Ограничения для входного трафика

Для всех потоков записываем ограничения для каждой пары времен. Для временны́х переменных, удовлетворяющих условиям: $t_{\pi_1} = t_{\pi_2}$ или $t_{\pi_1} \leq t_{\pi_2}$, записывается неравенство вида $F_i^{(0)}(t_{\pi_2}) - F_i^{(0)}(t_{\pi_1}) \leq \alpha_i(t_{\pi_2} - t_{\pi_1})$.

Будем считать, что кривая нагрузки α_i задается в виде $\rho_i t + \sigma_i$. Так как в качестве аргумента подается $t_\emptyset - t_\pi$, то кривая нагрузки принимает вид $\alpha_i(t_\emptyset - t_\pi) = \rho_i(t_\emptyset - t_\pi) + \sigma_i$.

Для потока 2 и задачи 1 будет верны неравенства:

$$F_2^{(0)}(t_\emptyset) - F_2^{(0)}(t_5) \leq \rho_2(t_\emptyset - t_5) + \sigma_2$$

$$F_2^{(0)}(t_\emptyset) - F_2^{(0)}(t_{35}) \leq \rho_2(t_\emptyset - t_{35}) + \sigma_2$$
$$F_2^{(0)}(t_\emptyset) - F_2^{(0)}(t_{135}) \leq \rho_2(t_\emptyset - t_{135}) + \sigma_2$$
$$F_2^{(0)}(t_\emptyset) - F_2^{(0)}(t_{235}) \leq \rho_2(t_\emptyset - t_{235}) + \sigma_2$$
$$F_2^{(0)}(t_5) - F_2^{(0)}(t_{35}) \leq \rho_2(t_5 - t_{55}) + \sigma_2$$
$$F_2^{(0)}(t_5) - F_2^{(0)}(t_{135}) \leq \rho_2(t_5 - t_{135}) + \sigma_2$$
$$F_2^{(0)}(t_5) - F_2^{(0)}(t_{235}) \leq \rho_2(t_5 - t_{235}) + \sigma_2$$
$$F_2^{(0)}(t_{35}) - F_2^{(0)}(t_{135}) \leq \rho_2(t_{35} - t_{135}) + \sigma_2$$

$$F_2^{(0)}(t_{35}) - F_2^{(0)}(t_{235}) \leq \rho_2(t_{35} - t_{235}) + \sigma_2$$

$$F_2^{(0)}(t_{135}) - F_2^{(0)}(t_{235}) \leq \rho_2(t_{135} - t_{235}) + \sigma_2$$

4.4 Ограничения для обработчиков

Для каждого обработчика, суммируя по всем потокам, проходящим через данный обработчик, записываем неравенства вида $\sum_{i \ni j} F_i^{(j)}(t_\pi) - \sum_{i \ni j} F_i^{(j)}(t_\pi) \geq \beta_j(t_\pi - t_{j\pi})$. Если выполняется соотношение $t_{j\pi_1} = t_{j\pi_2}$,тогда для t_{π_1} и t_{π_2}, связанных соотношениями $\{=, \leq\}$, добавляем неравенство $\sum_{i \ni j} F_i^{(j)}(t_{\pi_2}) - \sum_{i \ni j} F_i^{(j)}(t_{\pi_1}) \geq \beta_j(t_{\pi_2} - t_{\pi_1})$.

Будем считать, что кривая обслуживания β_i задается в виде $\rho_i t + \sigma_i$. Так как в качестве аргумента подается $t_\pi - t_{j\pi}$, то кривая обслуживания принимает вид $\beta_i(t_\pi - t_{j\pi}) = \rho_i(t_\pi - t_{j\pi}) + \sigma_i$.
Для обработчика 1:

$$F_1^{(1)}(t_{35}) - F_1^{(1)}(t_{135}) \geq \rho_1(t_{35} - t_{135}) + \sigma_1$$

Для обработчика 2:

$$F_2^{(2)}(t_{35}) - F_2^{(2)}(t_{235}) \geq \rho_2(t_{35} - t_{235}) + \sigma_2$$

Для обработчика 3:

$$(F_1^{(3)}(t_5) + F_2^{(3)}(t_5)) - (F_1^{(3)}(t_{35}) + F_2^{(3)}(t_{35})) \geq \rho_3(t_5 - t_{35}) + \sigma_3$$

Для обработчика 5:

$$F_2^{(5)}(t_\emptyset) - F_2^{(5)}(t_5) \geq \rho_5(t_\emptyset - t_5) + \sigma_5$$

4.5 Построение оценки задержки

Целевой функцией задачи линейного программирования является $max(t_\emptyset - u)$, где $t_\emptyset - u$ - задержка передачи данных, поступивших в сеть в момент времени u и остававшихся в сети до момента времени $t_\emptyset$. Переменную u вставляем во всевозможные положения в ограничении на время 1 и получаем различные варианты задач для линейного программирования. Для примера переменную u вставим между t_5 и $t_\emptyset$, получаем первое неравенство:

$$t_5 \leq u \leq t_\emptyset$$

Добавление новой переменной u влечет за собой еще ряд ограничений:

- *Время прихода:* $F_{i_0}^{(0)}(u) \geq F_{i_0}^{(last(i_0))}(t_\emptyset)$.
  Для потока 2 заключительным обработчиком является 5, поэтому неравенство примет вид:
  $$F_2^{(0)}(u) \geq F_2^{(5))}(t_\emptyset)$$

- *Позиция и монотонность:* для соотношения $t_\pi \leq u \Rightarrow F_{i_0}^{(0)}(t_\pi) \leq F_{i_0}^{(0)}(u)$, а для соотношений $u \leq t_\pi \Rightarrow F_{i_0}^{(0)}(u) \leq F_{i_0}^{(0)}(t_\pi)$.

$$F_2^{(0)}(t_5) \leq F_2^{(0)}(u) \leq F_2^{(0)}(t_\emptyset)$$

- *Ограничения для входного трафика:* для соотношения $t_\pi \geq u \Rightarrow F_{i_0}^{(0)}(u) - F_{i_0}^{(0)}(t_\pi) \leq \alpha_{i_0}(u - t_\pi)$, для обратного соотношения $t_\pi \leq u \Rightarrow F_{i_0}^{(0)}(t_\pi) - F_{i_0}^{(0)}(u) \leq \alpha_{i_0}(t_\pi - u)$. Для исходного положения u верны следующие ограничения на вход:

$$F_2^{(0)}(t_u) - F_2^{(0)}(t_{235}) \leq \rho_2(t_u - t_{235}) + \sigma_2$$

$$F_2^{(0)}(t_u) - F_2^{(0)}(t_{135}) \leq \rho_2(t_u - t_{135}) + \sigma_2$$

$$F_2^{(0)}(t_u) - F_2^{(0)}(t_{35}) \leq \rho_2(t_u - t_{35}) + \sigma_2$$

$$F_2^{(0)}(t_u) - F_2^{(0)}(t_5) \leq \rho_2(t_u - t_5) + \sigma_2$$

$$F_2^{(0)}(t_u) - F_2^{(0)}(t_\emptyset) \leq \rho_2(t_u - t_\emptyset) + \sigma_2$$

5 Алгоритм оценки задержки потока в виртуальном пласте

Пусть уже установлено $N-1$ виртуальных пластов и известны их приоритеты на коммутаторе и выделенная для них пропускная способность каналов. Рассмотрим N-ый виртуальный пласт, для которого задан начальный приоритет на коммутаторе, с учетом которого необходимо вычислить оценку задержку.

5.1 Входные данные для алгоритма

1. ***Уровень значимости и распределение χ^2, которое ему соответствует***

2. ***Топология сети с заданными очередями на коммутаторах***

   - Задается список из коммутаторов, отражающий узлы топологии сети.
   - Связи между коммутаторами задаются в виде ребер связного направленного графа, которые представляют собой пары коммутаторов, между которыми есть соединение. Граф - направленный, так как одно из ограничений работы метода линейного программирования - это сеть прямого распространения. Для выполнения этого требования необходимо задать направленный граф, который будет отражать направление движения транспортных потоков.
   - Список приоритетов виртуальных пластов на коммутаторах представляет собой список номеров виртуальных пластов в последовательности от наивысшего приоритета к наименьшему. Считаем, что список приоритетов справедлив для всех коммутаторов. Если потоки виртуального пласта не проходят через этот коммутатор, для упрощения расчетов задержка от него все равно будет учитываться.

3. ***Физическая пропускная способность каналов***

4. ***Список существующих виртуальных пластов с требованиями по качеству сервиса***
   Для каждого виртуального пласта, уже установленного в сеть, задается информация о:

   - идентификационном номере виртуального пласта;
   - максимальной скорости передачи данных внутри виртуального пласта.

5. ***Данные о виртуальном пласте, для которого выполняется оценка задержки***

   - Скорость передачи данных внутри виртуального пласта.
   - Приоритет виртуального пласта.

- Информация о всех потоках внутри данного виртуального пласта:
  - Маршрут потока, задающийся списком из коммутаторов, через которые проходит транспортный поток.
  - Величина ε, с которой допустимо нарушение построенной оценки для кривой нагрузки.
  - Значения ρ_A и b_A или файл с собранной статистикой по потоку.

5.2 Формирование кривой нагрузки

Для построения кривой нагрузки в виде $\alpha = \rho_A t + b_A$ необходимо найти значения ρ_A и b_A по заданной статистике. Статистика представляет собой количество байт, пришедших за 1 секунду. Считаем, что замеры пришедшего трафика осуществлялись каждую секунду и всего таких замеров было сделано n, причем $n \geq 3$.

Для построения кривой нагрузки необходимо описать заданную статистику каким-то распределением. Для упрощения вычислений возьмем Пуассоновское распределение и при помощи метода проверки гипотезы о виде распределения, предложенного в [13], вычислим параметр λ Пуассоновского распределения. Стоит отметить, что возможно использование и других распределений с применением того же критерия согласия Пирсона. Для того чтобы можно было применить данный метод проверки гипотезы о виде распределения, необходимо преобразовать собранную статистику. Случайной величиной x_i считаем количество байт, пришедших за 1 секунду, а значением случайной величины n_i считаем количество раз, которое данное значение встретилось собранной в статистике, где i - индекс различных значений в файле с собранной статистикой.

Применим алгоритм, описанный в [13]. Данный метод основан на проверке гипотезы при помощи критерия согласия Пирсона. В качестве гипотезы возьмем Пуассоновское распределение с параметром $\lambda = x_{cp}$, где $x_{cp} = \dfrac{\sum_i x_i * n_i}{\sum_i n_i}$. Для начала вычислим вероятность p_i, что пришедшее количество байт соответствуют значению величины x_i. Данная вероятность вычисляется по формуле $p_i = \dfrac{\lambda^i}{i!} e^{-\lambda}$. После этого для каждой случайной величины x_i вычислим критерий согласия Пирсона $K_i = \dfrac{(n_i - np_i)}{np_i}$. Просуммировав K_i, получим величину $K = \sum K_i$ - эмпирическое значение критерия согласия для рассматриваемого распределения.

Критической областью называют совокупность значений критерия, при которых выдвинутую гипотезу отвергают. Областью принятия гипотезы называют совокупность значений критерия, при которых гипотезу принимают. Критическими точками называют точки, отделяющие критическую область от области принятия выдвинутой гипотезы гипотезы. Определим границу критической области для данного распределения. Критическая область для данной гипотезы всегда правосторонняя: $[K_{kp}, +\infty)$, где K_{kp} является теоретическим значением критерия согласия Пирсона и находится по таблицам распределения χ^2, данный подход описан в [14]. Так как критерий Пирсона измеряет

разницу между эмпирическим и теоретическим распределениями, то чем больше наблюдаемое значение $K_{current} = K$, тем сильнее довод против основной гипотезы. Если полученное значение $K \in [0, K_{kp})$, то построенная гипотеза верна и собранная статистика удовлетворяет Пуассоновскому распределению с параметром λ. Иначе наблюдаемое значение критерия Пирсона попадает в критическую область $K_{current} \in [K_{kp}, +\infty)$, поэтому есть основания отвергать построенную гипотезу.

Если по критерию согласия Пирсона построенная гипотеза о виде распределения собранной статистики верна, то получаем распределение для входящего трафика, которое имеет вид $p_i = \frac{(x_{cp})^i}{i!} e^{-x_{cp}}$. Используя полученное распределение, построим оценку MGF (5) для данного трафика, где $\rho_A = \frac{\lambda(e^{\theta/\nu} - 1)}{\theta}$, а $\sigma = 0$, ν^{-1} - размер пакета входящего трафика. Варьируемая величина θ необходима для получения оптимальной оценки задержки описанной в [7].

Переходим к представлению EBB (2) и вычисляем с помощью (7) величину всплеска трафика b_A, оптимизировав данную оценку по θ. Получим кривую нагрузки $\alpha = \rho_A t + b_A$ для данного трафика.

5.3 Формирование кривой обслуживания

Для построения кривой обслуживания в виде $\beta = \rho_S t + b_S$ необходимо найти значения ρ_S и b_S. Ограничение на скорость передачи данных задается в параметрах рассматриваемого виртуального пласта. Данное ограничение берется как значение величины ρ_S. Тем самым необходимо вычислить значение b_S.

Так как кривая обслуживания дает оценку снизу, то необходимо оценить наихудший случай загруженности каналов. Рассмотрим построение требуемой оценки на примере, изображенном на рисунке 6.

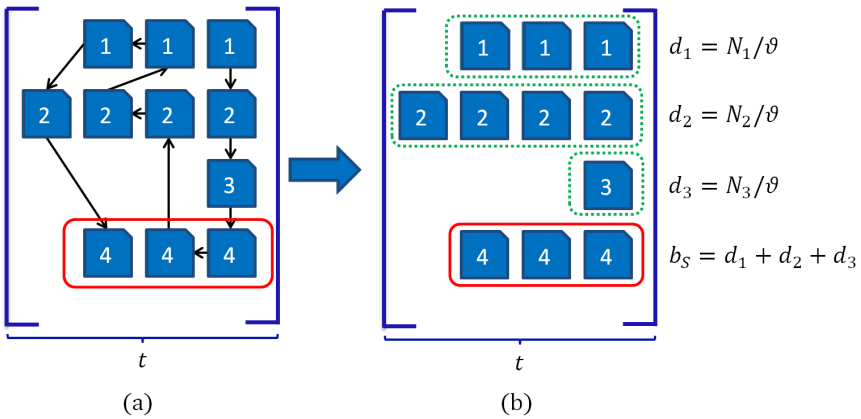


Рис. 6: *Пример оценки задержки обработчика с учетом приоритетов виртуальных пластов.*

На рисунке 6(а) изображена последовательность обработки пакетов на коммутаторе. Для того чтобы получить оценку задержки на коммутаторе для виртуального пласта

с меньшим приоритетом, необходимо знать время, которое будет затрачено на передачу пакетов для виртуальных пластов имеющих больший приоритет, что отражено на рисунке 6(b).

Рассмотрим подробнее преобразование, изображенное на рисунке 6(b). Для интервала времени t ($t = 1$ секунде) рассмотрим, какое количество пакетов может обработать обработчик за этот интервал времени. Максимальное количество пакетов зависит от пропускной способности, выделенной под каждый виртуальный пласт, то есть от скорости передачи данных внутри этого виртуального пласта. Тем самым задается максимальное количество обработанных пакетов за интервал времени для данного виртуального пласта. Для примера возьмем виртуальный пласт с наивысшим приоритетом (пакеты, обозначенные номером 1). Пусть скорость передачи данных внутри виртуального пласта 10 Mbps, тогда количество обработанных пакетов для данного виртуального пласта равен 10 Mb (на рисунке 6 $N_1 = 10$ Mb). Так как за 1 секунду из этого виртуального пласта больше чем 10 Mb данных обработчик обработать не может. Для того чтобы вычислить время d_i, необходимое на отправку пакетов с данным приоритетом, нужно поделить величину N_i на пропускную способность канала ϑ. Вычислив такую величину для всех виртуальных пластов и просуммировав эти значения у пластов с бóльшим приоритетом, получаем задержку перед отправкой пакетов на данном обработчике для виртуального пласта с приоритетом ниже. То есть если рассматривать очередь с приоритетом 4 на рисунке 6(b), то для вычисления b_S для 4-го виртуального пласта необходимо найти $\sum_{i=1}^{3} d_i$, что и будет величиной b_S.

Нерассмотренной задачей остался подбор параметра t. Один из возможных подходов основан на учете таких характеристик виртуального пласта, как размер очереди и доступная пропускная способность. Пусть для виртуального пласта на коммутаторе выделана очередь i, размер которой равен V_i. Пропускная способность внутри виртуального пласта равна ρ_S, а пропускную способность канала обозначим как ϑ. Тогда время, необходимое для обработки пакетов очереди i, вычисляется как $t_i = \dfrac{V_i}{\vartheta}$. С другой стороны время t_i для отправки пакетов из очереди i - это часть временного интервала t, которая пропорциональна части пропускной способности, занимаемой данным виртуальным пластом, причем $\sum_i t_i = t$. Тогда t_i можно задать как $\dfrac{\rho_S}{\vartheta} t$. Тогда получаем формулу для вычисления временного интервала, исходя из физического размера очереди и пропускной способности внутри виртуального пласта, которая имеет вид: $\dfrac{V_i}{\vartheta} = \dfrac{\rho_S}{\vartheta} t \;\Rightarrow\; V_i = \rho_S t \;\Rightarrow\; t = \dfrac{V_i}{\rho_S}$.

5.4 Алгоритм вычисления оценки задержки

Рассмотрим поэтапно работу алгоритма по вычислению оценки задержки внутри виртуального пласта.

После считывания всех данных из входного файла вычисляются кривые нагрузки и обслуживания, затем начинаем составлять задачу линейного программирования на основе неравенств, описанных в главе 4.

Для каждого потока формируем множество временны́х переменных, которые имеют одинаковый конечный обработчик с данным потоком. Например, есть поток, который проходит через коммутаторы $[1, 2, 3]$, то время для обработчика $3 \Rightarrow t_3$, а для обработчика $2 \Rightarrow t_{23}$ и так далее всем коммутаторам в топологии (даже тем, которые не входят в маршрут рассматриваемого потока) ставятся в соответствие временны́е значения. Тогда искомое множество для рассматриваемого потока будет состоять из всех последовательности временны́х переменных, оканчивающихся в 3 обработчике + время $t_\varnothing$. Затем необходимо для рассматриваемого потока сформировать задачи линейного программирования. При построении множества времен для данного потока параллельно строим дерево соотношений этих времен. Для примера изображенного на рисунке 5 и потока 2 данное дерево изображено на рисунке 7.

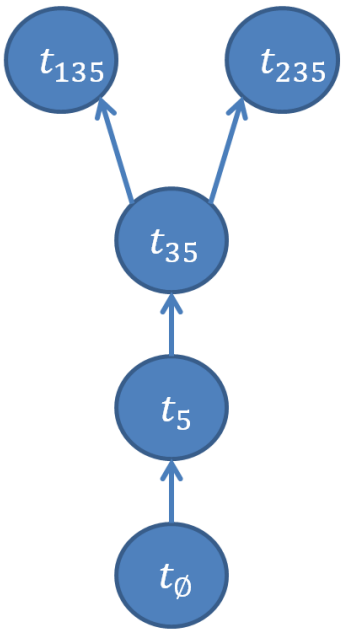


Рис. 7: *Дерево соотношений времен.*

Выполним обход по этому дереву, выбирая на каждом шаге обхода листовую вершину, которая затем удаляется из дерева, находящуюся на любом предыдущем уровне, на текущем или на один уровень ниже. Перебрав все вершины, получим последовательность обхода дерева, которая является возможной задачей линейного программирования. Множество всех получаемых последовательностей описанным алгоритмом перебора даст все возможные задачи линейного программирования. Проверив выполнение временных ограничений, описанных в пункте 4.1 и расставив знаки $[\leq, =]$, получаем временны́е ограничения для задачи линейного программирования.

Возьмем одно временное ограничение и опишем подход формирования неравенств и решения задачи линейного программирования. Исходя из сформированного на прошлом шаге множества времен, для каждого потока в данном виртуальном пласте и для каждого обработчика в этих потоках формируем множество временны́х переменных для данного потока и из него выбираем относящиеся к данному обработчику. Такое

разделение упростит нам построение ограничений линейного программирования.

Для каждого потока записываем траекторные ограничения и ограничения для входного трафика (описанные в пунктах 4.2 и 4.3 соответственно). Для каждого коммутатора, входящего в топологию, записываем ограничения для обработчиков (описанные в пункте 4.4). Теперь добавим значение u во временные ограничения, варьируя положение u между разными временны́ми переменными. Для каждого такого положения записываем дополнительные неравенства, описанные в пункте 4.5, и добавляем оптимизируемое значение $max(t_{\emptyset} - u)$, получаем готовый файл с задачей линейного программирования.

Каждый такой файл решается с помощью средства lp_solve [15]. Данное средство решает задачу линейного программирования и выдает значение оценки задержки для одного рассматриваемого потока. Перебрав все временны́е ограничения для одного потока, находим максимальную оценку задержку для него. Вычислив оценку задержки для каждого потока в рассматриваемом виртуальном пласте, выбрав из данных значений максимальное, получим оценку задержки для виртуального пласта.

5.5 Вероятность полученной оценки

При формировании кривой нагрузки задается значение ε - величина, с которой может нарушаться оценка для рассматриваемого потока. Необходимо оценить значение вероятности, с которой полученная нами оценка задержки для виртуального пласта будет корректна. Задержка для каждого потока вычисляется отдельно и единственным связывающим звеном являются ограничения на обработчики, но при построении неравенств считаем, что каждый поток проходит с уже выделенной для него пропускной способностью. Поэтому, учитывая одно из основных ограничений сетевого исчисления: отсутствие сброса и потерь пакетов, считаем потоки в виртуальном пласте независимыми. Тем самым вероятность корректности построенной оценки вычисляется как произведение вероятностей, с которыми верны кривые нагрузки.

Пусть в виртуальном пласте проходят 2 потока, для которых построены кривые нагрузки с вероятностям нарушения данной оценки $\varepsilon_1 = 1\%$ и $\varepsilon_2 = 0,5\%$ соответственно. Тогда вероятности корректности кривых нагрузки равны $99,0\%$ и $99,5\%$ соответственно, а вероятность, с которой верна построенная оценка, равна $98,5\%$ ($0,99 \times 0,995 = 0.98505$).

5.6 Реализация алгоритма

Основная работа алгоритма и все генераторы входных данных реализованы на языке Python3. Входные данные задаются в формате json. Статистика хранится в CSV файлах. Имя данного файла записывается во входной файл напротив поля "statistic имя формируется от директории, в которой лежит входной файл. Вызов lp_solve осуществляется непосредственно из основной программы при помощи функции Popen модуля subprocess. Результат выполнения программы печатается в стандартный поток вывода.

6 Экспериментальное исследование

6.1 Вычисление сложности алгоритма для линейной топологии

Целью данного экспериментального исследования является установление зависимостей времени вычисления алгоритма и скорости роста числа ограничений линейного программирования от увеличения числа узлов линейной топологии и увеличения числа потоков.

Используется линейная топология с N последовательно соединенными коммутаторами, через которую проходят M потоков, как изображено на рисунке 8.

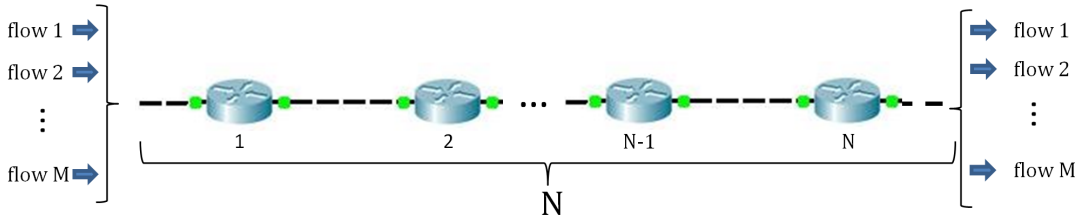


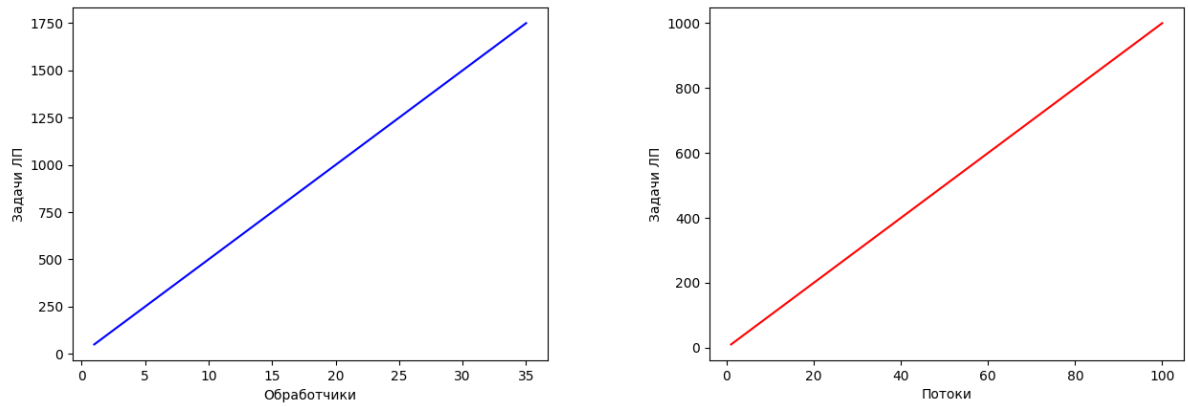
Рис. 8: *Топология сети для проведения экспериментов.*

Был написан алгоритм на языке Python3, который вычисляет оценку задержки для виртуального пласта в соответствии с алгоритмом, описанным в предыдущей главе. Считаем, что уже установлено два виртуальных пласта с доступной пропускной способностью равной 10 Mbps. Физическая пропускная способность канала равна 100 Mbps. Потоки проходят через все коммутаторы, тем самым дают максимальную нагрузку на топологию и загружают каналы. Считаем обработчики одинаковыми со скоростью обработки 10 Mbps с начальной задержкой равной 1 s. Для упрощения вычислений всем потокам сразу задается кривая нагрузки и считаем все входные потоки равными, скорость которых ограничена сверху величиной вычисляемом по формуле $\rho_A = \dfrac{0.9 \times \rho_S}{M}$ и величиной всплеска вычисляемой по формуле (7).

Был написан генератор входных файлов на языке Python3, который создает $N \times M$ входных файлов, генерируя для каждой длины топологии различное число потоков. Эксперименты были запущены для $N = 35$ и $M = 50$ и было выявлено несколько зависимостей:

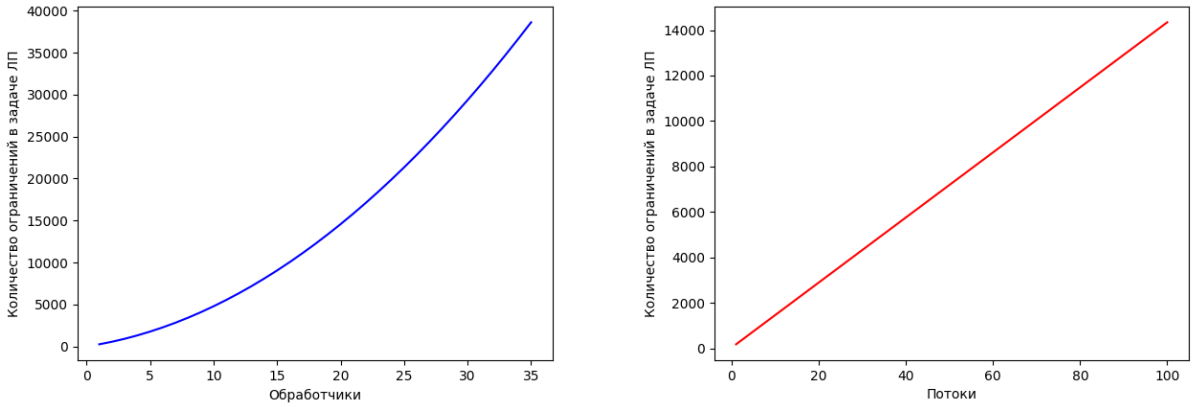
- Количество задач линейного программирования, необходимых для вычисления задержки для одного эксперимента равно $< switch\_number > \times < flow\_number >$. Линейная зависимость от числа обработчиков изображена на рисунке 9(а) при количестве потоков, равном 50, а линейная зависимость от числа потоков на рисунке 9(б) при количестве обработчиков, равном 13.

- Количество ограничений для одной задачи линейного программирования имеет зависимость $\mathcal{O}(< switch\_number >^2 \times < flow\_number >)$. Квадратичная зависимость от числа обработчиков изображена на рисунке 10(а) при количестве



а) б)

Рис. 9: Зависимость количества задач линейного программирования для вычисления оценки задержки внутри виртуального пласта от (а) - размера топологии, (б) - количества потоков.



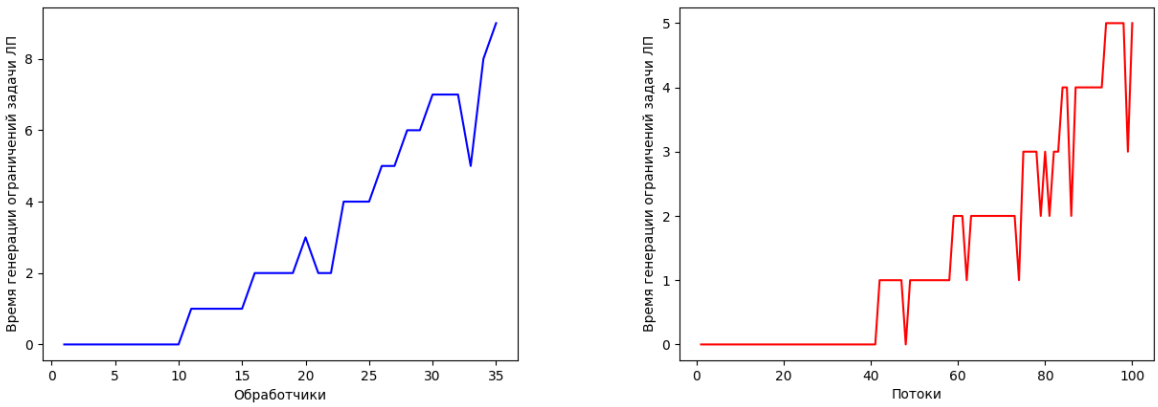
а) б)

Рис. 10: Зависимость количества ограничений для задачи линейного программирования от (а) - размера топологии, (б) - количества потоков.

потоков, равном 50, а линейная зависимость от числа потоков на рисунке 10(б) при количестве обработчиков, равном 13.

Зависимость времени генерации неравенств от величины топологии и числа потоков изображена на рисунках 11(а) и 11(б) соответственно. Стоит отметить, что при генерации временных ограничений не полным перебором, а методом описанным в пункте 5.4, время генерации неравенств для задачи линейного программирования было существенно уменьшено.

Зависимость времени вычисления задержки для виртуального пласта зависит от времени генерации ограничений для задачи линейного программирования t_{gen}, от количества задач N, которые необходимо решить для вычисления задержки внутри виртуального пласта, и от времени решения каждой задачи t_i. Тогда общее время вычисления оценки задержки имеет вид: $T = \sum_{i=1}^{N} t_i + t_{gen} + \mathcal{O}(N)$, где $\mathcal{O}(N)$ - время для нахождения максимума по всем задачам. Зависимость общего времени вычисления оценки задерж-



а) б)

Рис. 11: Зависимость времени генерации ограничений для задачи линейного программирования от (а) - размера топологии при количестве потоков равным 50, (б) - количества потоков при количестве обработчиков равным 13. Время задается в секундах.

ки от общего числа ограничений для задачи линейного программирования изображено на рисунке 12. Общее число ограничений - это сумма ограничений по всем задачам линейного программирования, решаемых для вычисления оценки задержки виртуального пласта.

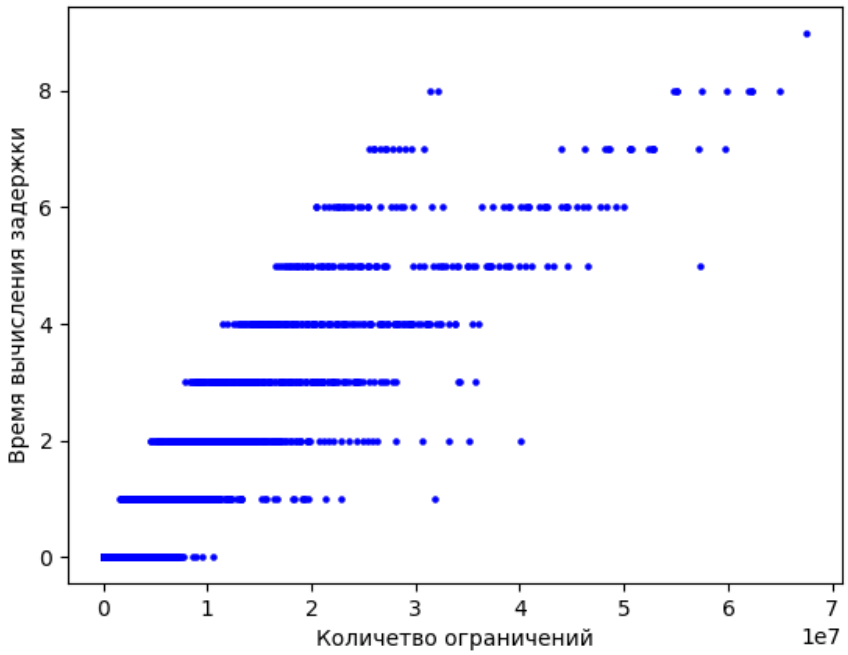


Рис. 12: Зависимость времени вычисления задержки виртуального пласта от общего количества ограничений задачи линейного программирования. Время задается в секундах.

Ступенчатость рисунка 12 обусловлена значительным ростом величины t_{gen} с увеличением числа узлов и числа потоков.

6.2 Вычисление сложности алгоритма для сети прямого распространения

Целью данного экспериментального исследования является установление зависимостей времени вычисления алгоритма и скорости роста числа ограничений линейного программирования от увеличения связности топологии.

Была рассмотрена топология из 10 обработчиков, через которую проходят 35 потоков. Начальная топология изображена на рисунке 13(а). На каждом шаге алгоритма добавляется одно соединение, которое еще не принадлежит топологии, до тех пора пока топология не станет полно связной и не примет вид, изображенный на рисунке 13(б).

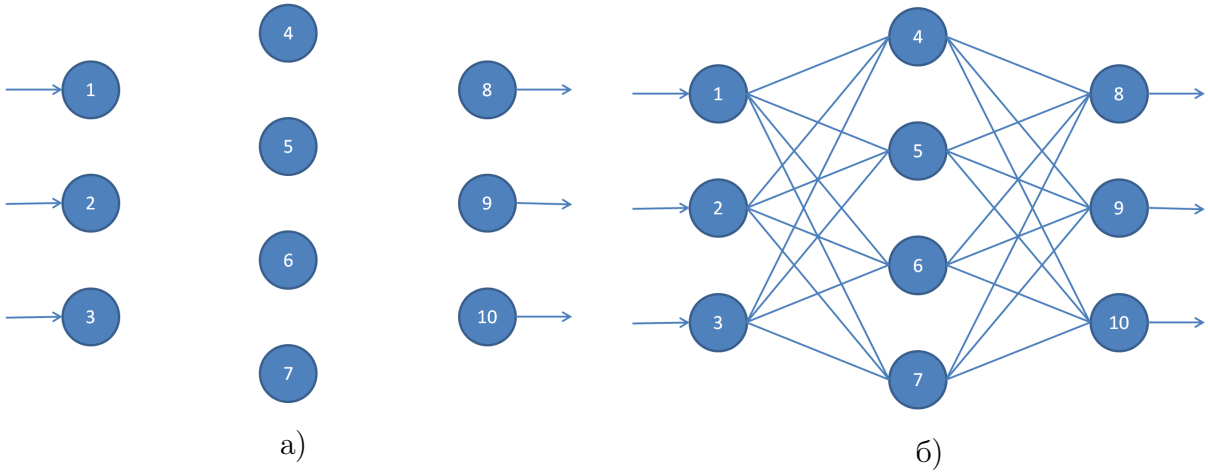
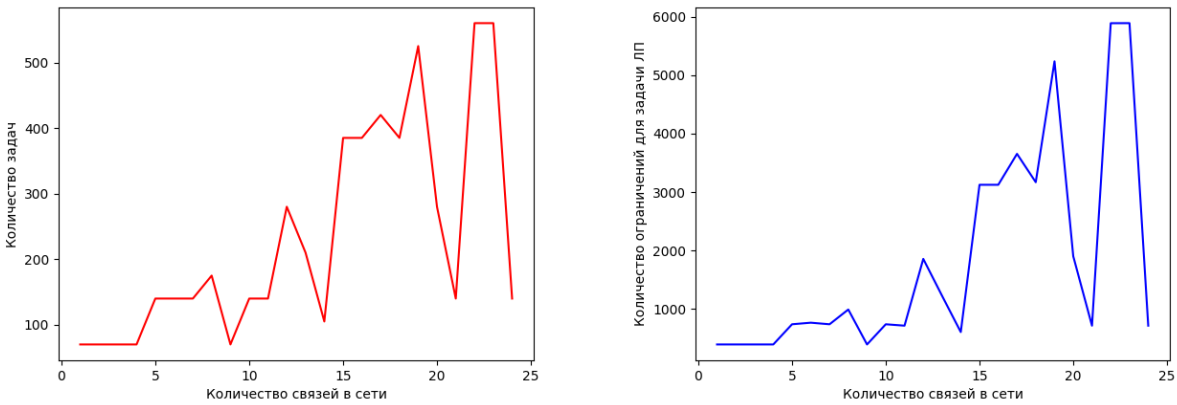


Рис. 13: Топология для проведения экспериментов.

Потоки выбираются случайным образом, но так чтобы они проходили через всю сгенерированную топологию на данный момент. Это условие возникает в связи с тем, что наличие узлов и связей без потоков никак не влияет на значение оценки задержки, и следовательно их необходимо отбросить. Но так как на каждом шаге необходимо увеличить связность сети, то для сохранения числа связей маршруты были выбраны так, чтобы они покрывали всю топологию.

Для данной топологии рассмотрим зависимости изменения числа задач и числа ограничений для задачи линейного программирования от связности сети. Зависимость числа задач линейного программирования, которые необходимо решить для вычисления оценки задержки виртуального пласта, от связности сети изображена на рисунке 14(а). Зависимость числа ограничений для одной задачи линейного программирования от связности сети изображено на рисунке 14(б).

Как видно из рисунка 14, зависимости задач и ограничений имеют схожее поведение при увеличении связности сети. Для данной топологии существуют резкие пики и провалы для количества задач и ограничений. Это обусловлено тем, что при изменении связности топологии некоторые времена начала периодов отставания (пункт 4.1) связываются более жесткими соотношениями между собой и имеют меньше вариантов расположения во временных ограничения задачи линейного программирования. А ко-



а) б)

Рис. 14: Зависимость (а) - количества задач ЛП для вычисления оценки задержки виртуального пласта, (б) - количества ограничений для одной задачи ЛП от связности сети.

личество ограничений для задачи линейного программирования напрямую зависит от построения временных ограничений для этой задачи.

Стоит отметить, что время на решение каждой задачи достаточно мало, данная зависимость изображена на рисунке 15. Сильный всплеск на значениях $18-21$ обуславливается неудачным прохождением потоков.

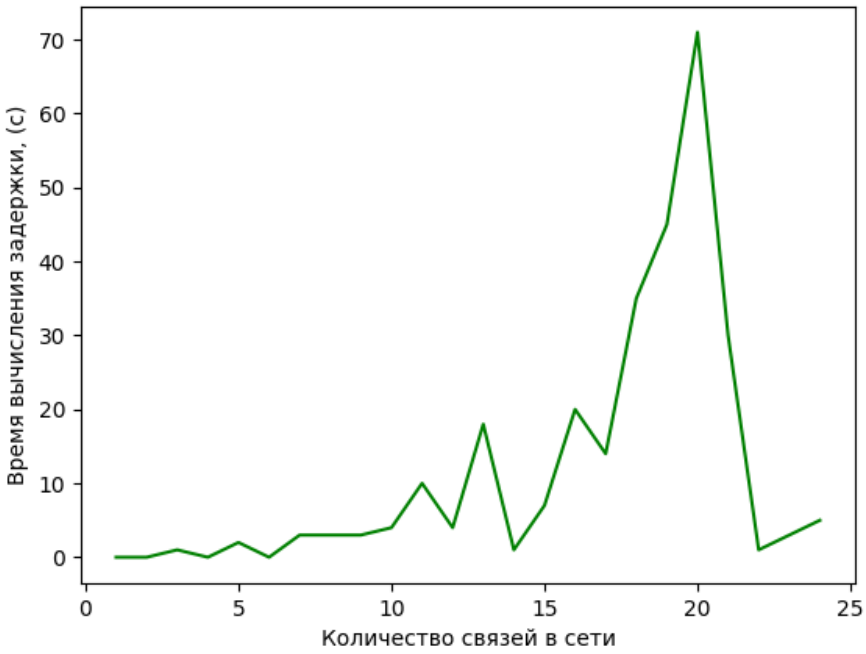


Рис. 15: Зависимость времени работы алгоритма от количества связей в топологии.

6.3 Выводы

Для вычисления оценки задержки виртуального пласта была выявлена зависимость $\mathcal{O}(< switch\_number >^2 \times < flow\_number >)$ числа неравенств для одной задачи линейного программирования от количества обработчиков и количества потоков. При вычислении оценки задержки рост времени генерации и общего времени выполнения достаточно нестабилен и имеет резки спады, что требует дальнейшего более подробного изучения.

Для топологии сети прямого распростронения невозможно построить точно зависимости от числа связей, так как велика зависимость не только от количества потоков (которое было постоянным), но и от их маршрутов, из-за чего возникают сильные всплески времени выполнения, количества задач и количества ограничений в задаче линейного программирования.

7 Заключение

В ходе выполнения настоящей работы были получены следующие результаты:

- По результатам обзора алгоритмов вычисления оценки задержки для множественных потоков, был выбран метод линейного программирования.
- Был предложен алгоритм задания кривой нагрузки по собранной статистике. Алгоритм использует критерий согласия Пирсена для определения распределения случайной величины, которая является количеством бит, поступивших в сеть в измеряемый момент времени. Для формирования кривой нагрузки по полученным параметрам трафика исходя из соответствующего ему распределения использовалась модель EBB из стохастического сетевого исчисления.
- Был предложен алгоритм формирования кривой обслуживания исходя из заданного приоритета для виртуальных пластов. Исходя из приоритета очереди на коммутаторе, соответствующей приоритету виртуального пласта, вычисляем начальную задержку для виртуального пласта. Чем ниже приоритет виртуального пласта, тем больше его начальная задержка.
- Был реализован алгоритм, который по топологии, информации о виртуальных пластах, вычисляет оценку задержки для всех внутренних потоков виртуального пласта, а затем и для самого виртуального пласта.
- Было проведено экспериментальное исследование, в ходе которого были выявлены зависимости времени генерации задачи линейного программирования, количества задач линейного программирования для вычисления оценки задержки виртуального пласта и количества ограничений в одной задаче линейного программирования от числа обработчиков и числа потоков для линейной топологии и от связности топологии для сети прямого распространения.

Дальнейший интерес для исследования представляют:

- разработка алгоритма, который позволяет удовлетворить требования качества сервиса по задержке при помощи изменения приоритетов виртуальных пластов;
- модификация предложенного алгоритма для работы с другими видами распределения трафика.

СПИСОК ЛИТЕРАТУРЫ

[1] Zhang S. An Overview of Network Slicing for 5G //IEEE Wireless Communications. – 2019.

[2] Halák J. et al. Real-time long-distance transfer of uncompressed 4K video for remote collaboration //Future Generation Computer Systems. – 2011. – Т. 27. – №. 7. – С. 886-892.

[3] Zhang D. et al. Context-aware multimedia content adaptation for mobile web //International Journal of Networked and Distributed Computing. – 2015. – Т. 3. – №. 1. – С. 1-10.

[4] Ordonez-Lucena J. et al. Network slicing for 5G with SDN/NFV: Concepts, architectures, and challenges //IEEE Communications Magazine. – 2017. – Т. 55. – №. 5. – С. 80-87.

[5] Jiang Y., Liu Y. Stochastic network calculus. – London : Springer, 2008. – Т. 1.

[6] Kim H., Feamster N. Improving network management with software defined networking //IEEE Communications Magazine. – 2013. – Т. 51. – №. 2. – С. 114-119.

[7] Fidler M., Rizk A. A guide to the stochastic network calculus. // IEEE Communications Surveys & Tutorials. – 2015. – Т. 17. – № 1. – С. 92-105.

[8] Fidler M. Survey of deterministic and stochastic service curve models in the network calculus //IEEE Communications surveys & tutorials. – 2010. – Т. 12. – № 1. – С. 59-86.

[9] Bouillard A., Thierry É. Tight performance bounds in the worst-case analysis of feed-forward networks //Discrete Event Dynamic Systems. – 2016. – Т. 26. – № 3. – С. 383-411.

[10] Schmitt J. B., Zdarsky F. A., Fidler M. Delay bounds under arbitrary multiplexing: When network calculus leaves you in the lurch.. // IEEE INFOCOM 2008-The 27th Conference on Computer Communications. – IEEE, 2008. – С. 1669-1677.

[11] Bouillard A. et al. Optimal routing for end-to-end guarantees using Network Calculus //Performance Evaluation. – 2008. – Т. 65. – № 11-12. – С. 883-906.

[12] Чемерицкий Е. В. ИССЛЕДОВАНИЕ МЕТОДОВ КОНТРОЛЯ ФУНКЦИОНИРОВАНИЯ ПРОГРАММНО-КОНФИГУРИРУЕМЫХ СЕТЕЙ.

[13] https://math.semestr.ru/group/poisson-examples.php

[14] Ковалевский А., Аркашов Н. Теория вероятностей и случайные процессы. – Litres, 2019. С.147 - С.159

[15] http://lpsolve.sourceforge.net/5.5/