Московский Государственный Университет им. М.В. Ломоносова
Факультет Вычислительной Математики и Кибернетики
Кафедра Автоматизации Систем Вычислительных Комплексов

Маршрутизация демультиплексированных соединений с учётом текущего состояния сети

Курсовая работа

Выполнил:
Звонов Андрей Денисович
321 группа

Научный руководитель:
Степанов Евгений Павлович

Москва, 2020 год

Содержание

1. Введение

1.1. Цель работы

Целью данной работы является ускорение работы транспортных соединений при помощи использования алгоритма маршрутизации демультиплексированных соединений, учитывающего текущее состояние сети.
Под состоянием сети в данной работе будем понимать данные о доступной пропускной способности каждого канала в сети в некоторый момент времени. То, каким способом были получены эти данные, выходит за рамки данной работы и не рассматривается.

1.2. Актуальность

В основные задачи существующих многопоточных протоколов не входит маршрутизация. Так, MPTCP, или MultiPath TCP, хоть и позволяет разбить поток данных на несколько подпотоков, требует заранее выбрать используемое количество подпотоков, причём это количество не изменяется во время работы. Протокол FDMP, или Flow (De) Multiplexing Protocol [2], в отличие от MPTCP, динамически изменяет количество используемых подпотоков, закрывая и открывая их по мере необходимости. Однако ни тот, ни другой не обеспечивают эффективную маршрутизацию установленного многопоточного соединения.

Как показано в [1], использование многопоточных соединений без эффективной политики их маршрутизации не даёт того выигрыша от демультиплексирования, который можно было бы получить при её наличии.

При этом существующие алгоритмы маршрутизации многопоточных соединений не являются эффективными. Например, GSPF, или жадный алгоритм, не обязательно найдёт (оптимальный) маршрут, а MCMF (Min-Cost Max-Flow) не предоставит множества маршрутов с необходимой пропускной способностью.

В связи с этим возникает необходимость разработки алгоритма, находящего множество маршрутов между двумя заданными точками с минимальным пересечением и удовлетворяющее заданному требованию к пропускной способности.

1.3. Задачи работы

Для решения данной задачи необходимо решить следующие подзадачи:

1) Провести обзор существующих алгоритмов маршрутизации для того, чтобы выбрать алгоритм, на основе которого далее будет предложено решение поставленной задачи.

2) Разработать и реализовать алгоритм маршрутизации демультиплексированных соединений, учитывающий текущее состояние сети.

3) Провести экспериментальное исследование прототипа разработанного алгоритма.

1.4. Структура работы

ыыыыыыыыыыыыыыыыыыыыыбля

2. Неформальная постановка задачи

Сведём задачу к задаче поиска маршрута на графе. Пусть нам известно состояние сети в некоторый момент времени, которое представлено в виде ненаправленного графа без кратных рёбер, для каждого ребра задана его остаточная пропускная способность в некоторый момент времени. Кроме того, обозначены точка-отправитель и точка-получатель, и известно требование к пропускной способности.
Задача будет состоять в том, чтобы найти множество маршрутов заранее неизвестной мощности (её тоже предстоит определить), и для каждого маршрута установить скорость передачи данных по нему в рамках данной задачи. Найденное множество должно удовлетворять нескольким требованиям, а именно:

- Суммарная скорость передачи до всем маршрутам должно равняться требованию;
- Найденное множество маршрутов должно иметь наименьшую стоимость среди всех множеств маршрутов

До того, как ввести функцию стоимости, отметим, что данная задача как минимум принадлежит классу $NP-hard$ (что может быть установлено путём сведения к нашей задаче задачи целочисленного линейного программирования, которая сама принадлежит классу $NP-complete$), то есть нахождение не точного, а приближённого решения в данном случае оправдано.
Стоимость каждого маршрута будет зависеть от двух величин — длины маршрута и т.н. избыточной занимаемой пропускной способности, о которой подробнее:
Пусть изначально величина максимального потока в остаточном графе сети между выбранными величинами была равна mf_1, а после выбора маршрута со скоростью, равной его максимальной пропускной способностью стала равна mf_2. Тогда, *избыточной* занятой пропускной способностью назовём отношение разности $mf_1 - mf_2$ к максимальной пропускной способности рассматриваемого маршрута.
Сумма же множества маршрутов будет складываться из индивидуальной стоимости всех входящих в него маршрутов и штрафа за каждое попарное пересечение между маршрутами множества.

3. Формальная постановка задачи

Пусть дан граф $G(V, E)$ без кратных рёбер. Кроме того, задано отображение $bw : E \to \mathbb{R}_+$ — остаточная пропускная способность каждого ребра графа в некоторый момент времени, $r > 0$ — требование к пропускной способности и $s, d \in V$ — вершина-источник и вершина-получатель, для которых необходимо решить задачу.

Введём некоторые дополнительные обозначения:

- Обозначим как P_i маршрут без циклов в G — чередующуюся последовательность вершин и рёбер, каждый два соседних элемента которой инцидентны: $P_i = \{s, e_1^i, v_1^i, ..., v_{n-1}^i, e_n^i, d\}$, где $v_j^i \neq v_k^i, j \neq k$;
- Пусть также $\mathbb{P} = \{P_i\}$ — множество всех маршрутов из s в d в ненаправленном графе G, а $P \subseteq \mathbb{P}$ - некоторое его подмножество;
- Назовём $mf(G, s, d)$ функцию, находящую величину максимального потока между вершинами s в d в графе G; данная функция будет нужна в дальнейшем для расчёта стоимости
- Пропускную способность отдельного маршрута обозначим $b : P_i \to \mathbb{R}$. Эта величина равна наименьшей из остаточных пропускных способностей всех входящих в маршрут рёбер.
- Каждому найденному маршруту будем присваивать скорость $rate : P \to R_+^{|P|}$, которая должна не превышать пропускной способности маршрута. Вообще говоря, выбор оптимальных скоростей для набора маршрутов - это отдельная задача, которая не рассматривается в рамках данной работы, поэтому в дальнейшем будем устанавливать скорость каждого вновь выбранного маршрута либо равной его максимальной пропускной способности, либо разности между требованием и уже имеющейся суммарной пропускной способностью других выбранных маршрутов.
- Также введём функцию штрафа за пересечения между парой маршрутов $isect : P_i^2 \to \mathbb{R}_+$, $isect(P_i, P_j) = |\{e \mid e \in P_i \& e \in P_j, e \in E\}| * const$, где $const$ определяется в зависимости от конкретной топологии
- Стоимостью отдельно взятого маршрута будем считать $Cost(P_i) = \frac{|P_i|-1}{2*|V|} + \frac{mf(G,s,d)-mf(G\setminus P_i,s,d)}{b(P_i)}$ — сумму двух величин: отношения длины маршрута к общему числу вершин в графе и отношения избыточной занимаемой пропускной способности к пропускной способности самого маршрута.
- Стоимость множества маршрутов $Cost(P) = \sum\limits_{P_i \in P} Cost(P_i) + \sum\limits_{P_i, P_j \in P} isect(P_i, P_j),\ i \neq j$ — сумма стоимостей всех входящих в P маршрутов и штраф за пересечения маршрутов этого множества между собой.
- Будем называть множество маршрутов $P \in \mathbb{P}$ допустимым, если выполнены следующие условия:
  - Суммарная скорость всех входящих в множество маршрутов равна заявленному требованию: $\sum\limits_{P_i \in P} rate(P_i) = r$.
  - Суммарная скорость всех проходящих через ребро графа маршрутов не превышает его пропускной способности: $\forall e \in E : \sum\limits_{P_i \ni e} rate(P_i) \leq bw(e)$

Необходимо найти допустимое множество маршрутов минимальной стоимости: $\arg\min\limits_{P \in Q} Cost(P)$, где Q — множество всех допустимых P.

4. Обзор существующих алгоритмов

Уже существует множество алгоритмов поиска маршрутов на графе, решающих различные схожие с поставленной задачи, такие как поиск заранее заданного числа непересекающихся / имеющих минимальное количество пересечений / имеющих минимальную длину маршрутов. Цель нижеследующего обзора — рассмотреть некоторые из алгоритмов решения похожих задач, определить их применимость или неприменимость к решению поставленной задачи и, возможно, выбрать один или несколько алгоритмов для того, чтобы на их основе построить алгоритм её решения.

4.1. Критерии обзора

- Входные данные алгоритма содержат заранее заданное количество маршрутов k.
  Заранее необходимое число маршрутов неизвестно, и его также необходимо определить, если алгоритм требует его в качестве входного параметра.

- Точность алгоритма.
  Под точностью алгоритма будем понимать, что в случае, если существует хотя бы одно множество маршрутов, удовлетворяющее поставленной ранее задаче, то алгоритм находит решение поставленной задачи, которое имеет минимально возможную стоимость. В противном случае (если решения не существует или алгоритм не может его найти) алгоритм должен остановиться.

- Временная сложность алгоритма.
  Безусловно, пространственная сложность - также важное свойство алгоритма, однако, так как алгоритм должен работать в реальном времени, то в первую очередь именно временная сложность является важным фактором выбора того или иного алгоритма.

4.2. Рассмотренные алгоритмы

4.2.1. Жадный алгоритм

Алгоритм работы:

1) В данном графе G найти какой-либо маршрут от s к d (например, с помощью алгоритма Дейкстры).

2) GOTO 1, если суммарная пропускная способность найденных маршрутов не достила нужного значения, иначе работа окончена.

Вышеизложенный алгоритм не является точным в нашем понимании, то есть может не найти решение, даже если оно существует и может быть найдено другими алгоритмами. Действительно, возможен случай, когда выбор наилучшего в данный момент маршрута "забьёт"узкие горлышки и требование уже не может быть достигнуто, в то время как при более разумном подходе решение может существовать. Соответственно, хотя данный алгоритм и может быть применён к поставленной задаче, он не является точным, поэтому поставленную задачу он не решает.
Однако далее при описании предложенных вариантов решения жадный алгоритм будет применена для того, чтобы по очереди выбирать маршруты из некоторого уже определённого другим алгоритмом множества.

Сложность жадного алгоритма — $O(k * |V|^2)$, где k — заданное число маршрутов.

4.2.2. k-max-min disjoint paths

Здесь рассмотрим не один конкретный алгоритм, а целое семейство алгоритмов, решающих задачу нахождения k непересекающихся маршрутов между двумя точками. Вообще говоря, как показано в [3], эта задача относится к классу NP—трудных, что даёт полное право строить не точные, а эвристические алгоритмы её решения. Подробно алгоритмы данного семейства описаны в [3, 4, 5], однако общая схема их работы следующая: сначала находится множество маршрутов от источника к получателю с помощью модифицированного алгоритма Дейкстры, после чего жадным алгоритмом выбирают k непересекающихся из них.

Данные алгоритмы находят k *непересекающихся* маршрутов, где k должно быть задано, то есть они не решают проблему выбора необходимого количества маршрутов. Кроме того, может не существовать множества именно *непересекающихся* маршрутов, удовлетворяющего заявленному требованию, хотя решение с пересекающимися маршрутами может существовать. Кроме того, на втором этапе работы алгоритма происходит "жадный"выбор, то возникает проблема, аналогичная проблеме предыдущего описанного алгоритма, а именно: алгоритм может не найти решение даже в случае, если оно существует.

Сложность алгоритмов этого семейства, описанных в [3, 4], равна $O(|E| * log|V| + V^2)$.

4.2.3. Max Flow, или задача о максимальном потоке

Данные алгоритмы находят не множество маршрутов, кое бы являлось решением задачи, а загрузку каждого канала, по которой позже можно составить множество маршрутов.
Достоинством этого подхода к решению является как то, что не нужно заранее знать количество используемых маршрутов, так и то, что он находит *какое-то* множество маршрутов, удовлетворяющее требованию, если такое множество существует.
При этом данный алгоритм не накладывает на решение никаких штрафов за пересечения между маршрутами, кроме того, он решает задачу нахождения именно *максимального* потока и при малом требовании может быть найдено плохое (то есть высокой стоимости) решение.

Таким образом, данный подход не годится для решения поставленной задачи, но, найдя величину максимального потока из точки в точку можно сразу проверить, не превышает ли её заявленное требование к пропускной способности. Более того, если требование не сильно отличается от найденной величины максимального потока, можно использовать этот подход для решения задачи.

Сложность алгоритма, решающего задачу о максимальном потоке, зависит от конкретного выбранного алгоритма. Так, для алгоритма Диница, алгоритма проталкивания предпотока она составляет $O(|V|^2 * |E|)$, а для алгоритма Эдмонда-Карпа — $O(|V| * |E|^2)$.

4.3. MCMF, или Min-Cost Max-Flow[1]

MCMF сводит задачу к задаче поиска максимального потока, и начинает работу с изменения графа. Каждое ребро в графе заменяется на множество k параллельных рёбер, где k — наперёд заданное требуемое количество маршрутов. Для каждого ребра из каждого такого множества установим стоимость, равную $1 + i * |E|$, где $|E|$ — количество рёбер в исходном графе, а i — порядковый номер ребра в своём множестве. Для всех рёбер полученного таким образом графа их пропускная способность принимается равной единице. После этого добавляется узел-сток, который соединяется с узлом-получателем одним ребром единичной стоимости и пропускной способности k.
Для полученного графа решается задача о нахождении максимального потока минимальной стоимости, после чего по потокам на каждом из рёбер восстанавливаются сами маршруты. Таким образом, благодаря добавленному узлу-стоку число найденных маршрутов будет ограничено k, а за счёт высокой стоимости использования параллельных друг другу рёбер пересе-

кающиеся по рёбрам маршруты будут выбраны в последнюю очередь.
Данный алгоритм обеспечивает нахождение k маршрутов с минимальным пересечением, где k —наперёд заданное число, то есть не решается задача нахождения необходимого количества подпотоков. Кроме того, серьёзным недостатком данного алгоритма является то, что он не учитывает ни требование к скорости, ни пропускную способность каналов, и для решения поставленной задачи данный алгоритм в существующем виде неприменим.

Сложность алгоритма можно грубо оценить сложностью используемого алгоритма решения задачи о нахождении максимального потока ($maxflow$).

4.4. Выводы

Как было установлено в результате обзора, ни один из рассмотренных алгоритмов не может быть применён к решению поставленной выше задачи. Однако, для дальнейшей модификации был выбран алгоритм MCMF. В частности, его преимуществом является то, что он не ограничивается только непересекающимися маршрутами, а лишь *избегает* их. Ниже будет детально изложена как предлагаемая модификация MCMF, так и другие подходы к решению задачи.

5. Предложенные решения

5.1. Алгоритмы, требующие предварительных действий

Так как введённая постановка задачи требует вычисления максимального потока для расчёта стоимости каждого маршрута, вполне уместно предложение заменить расчёт максимального потока в реальном времени определёнными предварительными расчётами до начала работы основного алгоритма, по результатам которых уже во время работы можно будет с некоторой точностью получать решение, используя менее сложные алгоритмы. В настоящей работе рассматриваются две модели проведения этих подготовительных расчётов и, соответственно, два варианта алгоритма.

5.1.1. Жадный с предрассчитанными маршрутами

Подготовительные шаги

$\forall s, d \in G$:

1) Найти множество всех маршрутов из s в d
2) Для каждого маршрута найти стоимость в соответствии с постановкой задачи
3) Сохранить пару (маршрут, стоимость)

В результате получается структура данных, в которой каждой паре точек на графе поставлено в соответствие множество пар маршрут-стоимость.

Основной алгоритм

В результате работы алгоритма получается множество пар (маршрут, скорость), сумма всех скоростей равна требованию к пропускной способности

0) Изначально множество найденных маршрутов пусто
1) Выбрать из множества маршрутов между точками s, d маршрут наименьшей стоимости.
2) Принять скорость маршрута равной меньшей из двух величин: максимальной пропускной способности маршрута или разности требования и суммы скоростей всех уже имеющихся маршрутов.
3) Добавить маршрут в множество найденных с рассчитанной скоростью.
4) Изменить (увеличить) стоимость всех ещё не выбранных маршрутов, имеющих пересечения с данным:
   - Для каждого из ещё не выбранных маршрутов найти разность стоимости множества маршрутов и стоимости множества маршрутов при условии добавления этого маршрута в множество.
5) Если суммарная скорость всех найденных маршрутов равно требованию, работа алгоритма окончена. Иначе GOTO 1.

Особенности подхода

\+ Нет необходимости искать маршруты алгоритма - маршруты уже посчитаны и нужно просто выбрать наиболее дешёвый

\- В худшем случае, на каждом шаге придётся искать количество общих рёбер и пересчитывать стоимость приблизительно $|V|^2 * |V|!$ маршрутов (количество маршрутов между любыми парами точек)

5.1.2. Жадный с предрассчитанными весами

Подготовительные шаги
$\forall s, d \in G$:

1) Найти множество всех маршрутов из s в d
2) Для каждого маршрута найти стоимость в соответствии с постановкой задачи
3) Добавить стоимость найденного маршрута к весу каждого ребра, через которое этот маршрут проходит

В результате подготовительных шагов получаем вес каждого ребра в графе. Перед началом работы присвоим рёбрам графа соответствующие веса.

Основной алгоритм
В результате работы алгоритма получается множество пар (маршрут, скорость), сумма всех скоростей равна требованию к пропускной способности.

0) Изначально множество найденных маршрутов пусто
1) Найти маршрут наименьшей стоимости между точками s, d (например, с помощью алгоритма Дейкстры)
2) Принять скорость маршрута равной меньшей из двух величин: максимальной пропускной способности маршрута или разности требования и суммы скоростей всех уже имеющихся маршрутов.
3) Добавить маршрут в множество найденных с рассчитанной скоростью.
4) Изменить (увеличить) вес всех рёбер, через которые проходит данный маршрут
5) Если суммарная скорость всех найденных маршрутов равно требованию, работа алгоритма окончена. Иначе GOTO 1.

Особенности подхода

\+ Найденных маршрутов как структуры данных не существует, после нахождения каждого маршрута необходимо увеличить всего не более $|E|$ рёбер на уже посчитанную величину

\- На каждом шаге необходимо находить маршрут; например, алгоритмом Дейкстры, со сложностью $O(|V|^2)$

5.1.3. Модифицированный MCMF [1]

Список литературы

[1] E. Stepanov, R. Smelianski. On Analysis of Traffic Flow Demultiplexing Effectiveness

[2] E. Chemeritskiy, E. Stepanov, R. Smelianski. Managing network resources with Flow (De) Multiplexing Protocol

[3] M. Doshi, A. Kamdar. Multi-Constraint QoS Disjoint Multipath Routing in SDN

[4] Jang-Ping S., Lee-Wei L., Jagadeesha R., Yeh-Cheng C. An efficient multipath routing algorithm for multipath TCP in SDN.

[5] C-L Li, S. T. McCormic, and D. Simchi-Levi , "The Complexity of Finding Two Disjoint Paths with Min-Max Objective Function ,"Discrete Applied Mathematics, Vol. 26, No. 1, pp. 105-115, January 1990.