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) Составить формальную постановку задачи.
2) Провести обзор существующих алгоритмов маршрутизации.
3) На основе результатов обзора существующих алгоритмам разработать алгоритм маршрутизации демультиплексированных соединений, учитывающий текущее состояние сети.
4) Реализовать прототип разработанного алгоритма.
5) Провести экспериментальное исследование прототипа разработанного алгоритма.

2. Формальная постановка задачи

Сведём задачу к задаче поиска маршрута на графе.
Пусть дан граф $G(V, E)$ без кратных рёбер. Кроме того, задано отображение $bw : E \to \mathbb{R}_+$ — доступная пропускная способность каждого ребра графа, $r > 0$ — требование к пропускной способности и $s, d \in V$ — вершина-источник и вершина-получатель, для которых необходимо решить задачу.
Введём некоторые дополнительные обозначения:

- $P_i = \{s, e_1^i, v_1^i, ..., v_{n-1}^i, e_n^i, d\}$, где $v_j^i \neq v_k^i, j \neq k$ — маршрут без циклов в G;

- $\mathbb{P} = \{P_i\}$ — множество всевозможных маршрутов из s в d
- $P \in \mathbb{P}$ - некоторое подмножество множества всех маршрутов;
- $mf(G, s, d)$ — величина максимального потока из s в d в графе G;
- $b: P_i \to \mathbb{R}$ — пропускная способность отдельного маршрута;
- $rate: P \to R_+^{|P|}$ — скорость отдельного маршрута;
- $isect: P_i^2 \to \mathbb{R}_+,\ isect(P_i, P_j) = |\{e \mid e \in P_i \& e \in P_j, e \in E\}| * const$
- $Cost(P_i) = \frac{|P_i|-1}{2*|V|} + \frac{mf(G,s,d)-mf(G\setminus P_i,s,d)}{b(P_i)}$ — стоимость отдельно взятого маршрута;
- $Cost(P) = \sum\limits_{P_i \in P} Cost(P_i) \ + \sum\limits_{P_i, P_j \in P} isect(P_i, P_j),\ i \neq j$ — стоимость набора P;
- P — допустимое множество маршрутов, если:
  - $P \in \mathbb{P}$;
  - $\sum\limits_{P_i \in P} rate(P_i) = r$;
  - $\forall e \in E : \sum\limits_{P_i \ni e} rate(P_i) \leq bw(e)$;
- Пусть Q — множество всех допустимых P.

Необходимо найти $\arg\min\limits_{P \in Q} Cost(P)$

3. Обзор существующих решений

3.1. Критерии обзора

Определим критерии сравнения:

- Входные данные алгоритма содержат заранее заданное количество маршрутов k;
- Точность решения
  (Под точностью будем иметь в виду, является ли решение оптимальным. Решение оптимально, если если оно удовлетворяет условию задачи и при наличии нескольких возможных решений имеет наименьшее количество пересечений маршрутов по рёбрам);
- Сложность алгоритма решения.

Перейдём к рассмотрению алгоритмов.

3.2. Жадный алгоритм

Алгоритм работы:

1) В данном графе G найти какой-либо маршрут от s к d (например, с посощью алгоритма Дейкстры).
2) GOTO 1, пока суммарная пропускная способность найденных маршрутов не достигнет нужного значения.

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

Кроме того, алгоритм не всегда находит оптимальное решение, и может не найти решение вообще. Действительно, возможен случай, когда выбор наилучшего в данный момент маршрута "забьёт"узкие горлышки и требование уже не может быть достигнуто. Притом, при более умном подходе решение может существовать.

3.3. k-max-min disjoint paths

В данном случае рассмотрим не отдельный алгоритм, а целое семейство. Например, подобные алгоритмы описаны в [3, 4]. Алгоритмы данного семейства сначала находят множество маршрутов от источника к получателю с помощью модифицированного алгоритма Дейкстры; после этого жадным алгоритмом выбирают k непересекающихся из них.

Данные алгоритмы находят k *непересекающихся* маршрутов, где k — задано. Как и жадный алгоритм, они не решают проблему выбора необходимого количсетва маршрутов. Кроме того, может не существовать множества *непересекающихся* маршрутов, удовлетворяющего заявленному требованию, хотя решение пересекающимися маршрутами может существовать. Также, так как на втором этапе работы алгоритма происходит "жадный"выбор, то возникает проблема, аналогичная проблеме предыдущего описанного алгоритма. То есть, алгоритм не всегда находит оптимальное решение или решение вообще.

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

3.4. Max Flow, или задача о максимальном потоке

Данные алгоритмы находят не множество маршрутов, являющееся решением задачи, а загрузку каждого канала, однако, по ней позже можно составить множество маршрутов. Достоинством этого подхода к решению является то, что не нужно заранее знать количество используемых маршрутов.

Однако, данный алгоритм решает задачу нахождения **максимального** потока, и при малом требовании оптимальное решение может быть не найдено. Таким образом, данный подход не годится для решения поставленной задачи, но годится, например, для проверки *наличия* решения поставленной задачи.

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

3.5. MCMF, или Min-Cost Max-Flow[1]

MCMF сводит задачу к задаче поиска максимального потока, и начинает работу с изменения графа. Он добавляет узел-сток, соединённый с получателем соединением с пропускной способностью k. Далее, каждое ребро заменяет на множество из k рёбер, где k — желаемое число 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 полученного графа решается задача о нахождении максимального потока; после этого маршруты, восстановленные из решения задачи о максимальном потоке, проецируются на исходный граф.

Данный алгоритм обеспечивает нахождение k маршрутов с минимальным пересечением, где k —наперёд заданное число, то есть он не решает задачу нахождения необходимого количества подпотоков. Кроме того, серьёзным недостатком данного алгоритма является то, что он не учитывает ни требование к скорости, ни пропускную способность каналов.

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

3.6. Выводы

Ни один из рассмотренных алгоритмов не обеспечивает нахождение оптимального решения; кроме того, некоторые из них требуют заранее определить количество маршрутов. Для модификации с целью выполнения поставленной задачи был выбран алгоритм MCMF, так как он не ограничивается лишь непересекающимися маршрутами. Подробно способ модификации алгоритма будет предложен далее.

4. Предложенные решения

4.1. 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.

4.1.1. 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)

4.1.2. 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 $O(|V|^2)$

4.1.3. 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.