СРАВНИТЕЛЬНЫЙ АНАЛИЗ АЛГОРИТМОВ КЛАССИФИКАЦИИ ТРАФИКА

Абдурахманов Рустам Паттахович
ТУИТ, к.т.н., профессор

Аннотация. *Процесс классификации интернет-трафика в пересылающей машине называется классификацией пакетов. Этот процесс становится очень важным в последние годы из-за огромного развития сетевых услуг. В этой статье объясняется таксономия наиболее популярных и современных алгоритмов классификации пакетов на основе древовидных чисел с детальным анализом, сравнением и разработкой нового алгоритма классификации на основе древовидных чисел.*

Ключевые слова: *классификация, IP-трафик, процесс, пакет, алгоритмы , база данных, структура данных, маршрутизаторы, контроль доступа.*

Интернет становится все более и более сложным местом для жизни из-за его использования для все большего количества критически важных задач, выполняемых организациями. Желательно, чтобы эти критически важные действия не были нарушены ни интенсивным трафиком, отправляемым другими организациями, ни злонамеренными злоумышленниками. Управле-ние трафиком, управление доступом и многие другие службы требуют распознавания пакетов на основе нескольких полей заголовков пакетов, что называется классификацией пакетов. Технология классификации пакетов быстро развивается, и в последние десятилетия были предложены разно-образные алгоритмы классификации пакетов. Тем не менее, большая часть литературы в основном сосредоточена на повышении производительности алгоритма классификации пакетов и игнорирует теоретический анализ и проблемы, возникающие в процессе реализации [1-2]. На фоне высококоско-ростных сетей алгоритмы классификации пакетов не обязаны иметь единст-венную особенность интенсивных задач проектирования по сложности во времени и пространстве, но также должны иметь хорошую масштабируе-мость и высокую гибкость для поддержки большого количества правил.

В этом работе описываются соответствующие исследовательские работы по алгоритмам классификации пакетов. Существующие алгоритмы классифика-ции пакетов можно сгруппировать в четыре класса: алгоритмы

на основе дерева , алгоритмы на основе хэша , алгоритмы параллельного поиска и эвристические алгоритмы. В этом обсуждении мы используем n для обозна-чения количества правил в базе данных классификации, к для обозначения количества полей (т. е. измерений) и w для обозначения максимальной длины полей (в битах).

Алгоритмы на основе Trie [3, 4] строят иерархические древовидные структуры, в которых, как только совпадение найдено в одном измерении, поиск выполняется в отдельном дереве, связанном с узлом, представляющим совпадение. Примерами таких алгоритмов являются алгоритмы Grid-of-tries [4] and Area-based Quad Tree (AQT) [5]. Алгоритмы на основе Trie требуют в худшем случае столько обращений к памяти, сколько битов в полях, исполь-зуемых для классификации. Структуры данных многобитового дерева более эффективны с точки зрения количества требуемых обращений к памяти. Однако эти структуры данных требуют значительно большего объема памяти. В целом схемы на основе древовидных чисел хорошо работают для одномерного поиска. Однако требования к памяти для этих схем значительно возрастают с увеличением количества измерений поиска.

Алгоритмы на основе Trie требуют памяти O(NW) и требуют доступа к памяти 2W-1 на каждый поиск, где N — количество фильтров, а W — длина IP-адреса [4]. Для двух полевых фильтров было предложено квадратное дерево на основе областей (AQT). AQT поддерживает эффективное время обновления. Производительность алгоритмов на основе trie изучается в [6]. Схемы, использующие дерево решений для разделения фильтров на несколь-ко наборов, представлены в статьях [7]. Количество фильтров в каждом наборе ограничено заранее заданными значениями, а для обхода набора фильтров используется линейный поиск. В [4] используется механизм, называемый перекрестным производством, включающий поиск BMP по отдельным полям и использование представленных предварительно рассчитанных таблиц для объединения результатов поиска отдельных префиксов. Но в этой схеме требования к памяти увеличиваются с увеличе-нием количества полей, 0(Nk), где к — количество классифицированных полей. Идея, основанная на хеше, привела к появлению двумерных фильтров. Фильтры с определенной длиной префикса группируются в кортеж, каждый кортеж затем объединяется для создания хэш-ключа, который используется для поиска кортежа. Соответствующий фильтр можно найти, поочередно проверяя каждый кортеж и отслеживая фильтр с наименьшей стоимостью.

Также было предложено множество алгоритмов на основе trie . Эти алгоритмы представляют собой широко используемые программные подходы. Маршрутизаторы на основе дерева и его структуры данных решают, как найти LMP. Во всех алгоритмах поиска следует учитывать четыре основных вопроса:

1. Требования к объему памяти: структура данных таблицы маршрутизации должна храниться в памяти, поэтому она должна использовать меньше места для уменьшения общего размера памяти.

2. Скорость поиска: наиболее серьезной проблемой интернет-маршрутиза-торов при пересылке пакета является скорость поиска выходного порта.

3. Масштабируемость. IPv6 станет следующим широко используемым идентификатором уровня 3, поэтому алгоритмы должны быть применимы как к IPv4, так и к IPv6.

4. Скорость обновления: информация о маршруте часто меняется в Интернете, а скорость обновление таблицы маршрутизации путем вставки и удаления записи имеет решающее значение.

Вот несколько примеров алгоритмов на основе trie :

а) Иерархическое дерево — это многомерная форма двоичного дерева , где каждое измерение представляет поле в наборе правил. Это также известно как попытки возврата и многоуровневые попытки. Иерархические подходы очень эффективны для обеспечения высокоскоростного поиска, поскольку пространство поиска значительно сокращается всякий раз, когда одномерный поиск завершается, поиск завершается.

б) Дерево сокращения множеств (set pruning trie) — это модифицированная форма иерархического дерева, которая решает возврат распространенная в иерархическом дереве. При сокращении наборов правила, включенные в исходный узел исходного дерева, реплицируются в целевое дерево дочернего узла, чтобы избежать обратного отслеживания. Благодаря сокращению множества все возможные совпадающие кандидаты собираются в дерево назначения. самого длинного совпадающего узла в исходном дереве. Следовательно, поиск в исходном дереве должен только определить самый длинный соответствующий узел, а затем перейти к соединенному с ним целевому дереву. Таким образом, обратного отслеживания можно избежать.

в) Ключевые идеи алгоритма «Grid-of-tries» заключаются в использовании предварительных вычислений и указателей переключения для ускорения поиска в более позднем исходном дереве на основе поиска в

более раннем исходном дереве. Этот метод также называется набором деревьев обрезки, поскольку избыточные поддеревья можно «отсечь» из дерева, разрешив множественные входящие ребра в узле. Хотя эта оптимизация действительно устраняет избыточные поддеревья , это не исключает полностью реплика-цию, поскольку фильтры могут храниться на нескольких узлах в дерево. Grid-of-Tries исключает эту репликацию, сохраняя фильтры на одном узле и используя указатели переключения для направления поиска к потенциально совпадающим фильтрам.

г) Extended Grid-of-Tries по существу изменяет указатели переключения на указатели перехода, которые направляют поиск по всем возможным совпадающим фильтрам, а не к фильтрам с самыми длинными совпадающи-ми префиксами адреса назначения и источника. Extended Grid-of-Tries начинается с построения стандартной сетки попыток с использованием префиксов адресов назначения и источника всех фильтров в наборе фильт-ров. Вместо того, чтобы хранить соответствующие фильтры в узлах префикса исходного адреса, расширенная сетка попыток сохраняет указатель на список фильтров, в которых указаны префиксы адресов назначения и источника, а также остальные три поля фильтров. При этом указатели перехода между попытками источника направляют поиск ко всем возможным совпадающим фильтрам. В худшем случае Extended Grid-of-Tries требует доступа к памяти 0(W2), где W — длина адреса.

С точки зрения требований к хранению иерархическое дерево является наиболее эффективным алгоритмом классификации. Сложность хранения равна O(dW). Здесь N — количество правил в наборе правил классифика-тора, d — количество подполей в каждом правиле, а W — максимальная глубина каждого подполя. Но с отрицательной стороны, выбор совпада-ющего правила занимает много времени, при поиске выполняется возврат с целью учесть все правила, совпавшие с пакетом, и выбирается правило с наивысшим приоритетом. Обратное отслеживание также важно, поскольку древовидная структура не позволяет эффективно показать уровни приоритета правил.

Обратное отслеживание в иерархическом дереве было необходимо, поскольку наборы правил дерева F1 являются непересекающимися наборами, а также мы не знаем заранее, какое дерево F2 содержит соответствующее правило. Поскольку набор сокращенных попыток устраняет эту необходи-мость, сложность времени поиска снижается до O(dW) вместо 0 (Wd), как в случае иерархических попыток. Но устранение процесса возврата происхо-дит за счет увеличения требований к памяти.

Сложность хранения равна 0(N d dW), как правило, требуется, чтобы реплицировалось максимальное количество раз N d. Сложность обновления правил в древовидной структуре имеет порядок 0(Nd). Сложность хранения снижается до O(NdW) по сравнению со сложностью 0(N d dW) набора сокращений trie. Сложность времени поиска O(dW) такая же, как и у дерева сокращения множеств, но значительно улучшена по сравнению с иерар-хическим деревом из-за заранее вычисленных указателей переключения. Но проблема со структурой сетки попыток заключается в том, что ее очень сложно построить, когда количество полей увеличивается. Кроме того, дополнительные обновления, которые необходимо применить к древовидной структуре, довольно сложны, поскольку необходимо изменить множество указателей. Если какой-либо узел необходимо удалить, то указатели, указывающие на этот узел, необходимо переориентировать на новый узел, созданный на его месте.

Описанные ранее алгоритмы trie страдают либо от проблем с более длительным процессом поиска, либо от большого потребления памяти, либо от увеличения сложности. Дерево сокращения множества решает проблему возврата к узлам-предкам, но это приводит к занятию большого объема памяти. Также существуют сетки попыток, но их сложность значительно возрастает с увеличением количества полей в наборе правил. Поэтому мы предложили модифицированную древовидную структуру, основанную на том факте, что более конкретные префиксы, то есть более длинные пре-фиксы, обычно имеют более высокий приоритет. Если префикс заголовка пакета сначала сопоставляется с правилом сопоставления с более кон-кретным префиксом, то в ближайшем будущем это конкретное правило станет лучшим правилом сопоставления, и поэтому поиск по дереву можно обойти для большинства входных данных, просто сравнив префикс заголовка пакета. с правилом сопоставления с наивысшим приоритетом каждой попытки уровня 2 с сохранением адресов назначения. Указатель предка или родительского дерева сохраняется в каждом корневом узле целевого дерева, так что он будет связан с этим предком всякий раз, когда начинается процесс поиска. Таким образом, это похоже на дерево обрезки множеств, использу-ющее двоичное дерево поиска с удалением пустых узлов в целевом дереве.

Скорость, с которой происходит классификация пакетов, рассчитывается с использованием количества обращений к памяти в алгоритме, поскольку поиск в памяти-самая медленная операция в процессе поиска. Подходы, основанные на иерархических деревьях, очень

эффективны для обеспечения высокоскоростного поиска, поскольку пространство поиска существенно сокращается по сравнению с одномерным поиском. Существующие методо-логии, основанные на иерархических деревьях, имеют два основных недостатка: возврат назад и пустые узлы внутри целевого дерева. Чтобы исключить обратное отслеживание, иерархическое дерево сокращения наборов использует сокращение набора правил, при котором правила, включенные в узлы-предки, копируются во все узлы-потомки. В структуре Grid- oftries он должен заранее вычислить указатели переключения в дереве назначения.

На основании вышеизложенного мы предлагаем новую схему алгорит-мов на основе древовидных чисел . Предлагаемый модифицированный алгоритм сокращения дерева множеств представляет собой комбинацию двоичного дерева поиска и дерева сокращения множеств . Конструкция дерева аналогична дереву сокращения множества , но с удалением пустых внутренних узлов в дереве назначения с использованием алгоритма дерева двоичного поиска .

Для использования предложенной схемы IP-адреса в дереве назначения должны храниться в порядке возрастания длины префиксов. При сравнении двух префиксов разной длины префикс меньшей длины приравнивается к подстроке аналогичной длины префикса большей длины. Если совпадение оказывается равным, то наблюдается следующий бит префикса большей длины. Если следующий бит равен единице, то префикс одиночной длины обозначается как больший, или же префикс с меньшей длиной обозначается как больший. Таким образом, дерево двоичного поиска санкционирует ограничение на создание двоичного дерева назначения , поскольку оно заставляет узел -предок располагаться на более высоком уровне, чем узел-потомок или узел-потомок, чтобы префикс предка сравнивался раньше. Таким образом, префикс предка более короткой длины будет сопоставляться раньше, чем префикс более длинной длины, при условии, что они находятся во взаимосвязи вложенности префиксов.

В структуре дерева двоичного поиска пустые внутренние узлы удаляют-ся, если они удовлетворяют следующим двум условиям: во-первых, они не должны иметь дочерних узлов или иметь только одного дочернего узла, а во-вторых, они не должны быть серыми узлами, т. е. они не должны содержать информацию о следующем переходе или любое соответствующее правило. Уровень 2 древовидной структуры строится с использованием IP-префиксов таблицы назначения. Запись действительного бита устанавливается в едини-цу, когда правило сохраняется в узле, или же узел

удаляется, когда у него нет сохраненного правила. Если узел удаляется, дочерние указатели, указыва-ющие на него, сбрасываются на следующего доступного предка, что обе-спечивает непрерывность процесса поиска для всех других сравнений полей. Указатель связанного списка используется для соединения пар префиксов источника и назначения, а связанные правила располагаются в списке ссылок в порядке убывания их важности.

Процесс поиска аналогичен процессу обрезки stet trie : во входящем пакете выполняется поиск самого длинного правила соответствия с теми же полями. Адрес источника входящего пакета сравнивается с trie уровня 1 для поля F1 с префиксом 0, обозначающим левый указатель. Это указывает на то, что входной адрес меньше по длине, чем префикс, хранящийся в этом узле, и префикс 1, указывающий на правый указатель. . Если пакет соответствует любому узлу в дереве исходного IP-префикса или узлов больше нет для про-хождения, то он следует за указателем следующего дерева, указывающим на следующий уровень дерева .т.е. к целевому IP-дереву. Правилу, которое лучше всего соответствует, присваивается наименьший приоритет. Теперь поиск в дереве IP- адресов назначения выполняется рекурсивно, как и в дереве IP-адресов источника.

Сложность поиска предлагаемого алгоритма равна O(dlogN), при этом W (глубина каждого поля) заменяется на logN , где N — количество правил в наборе правил. Требование к памяти меньше или максимально равно тре-бованию для дерева сокращения множества , т. е. O(N d), которое в худшем случае совпадает с иерархическим деревом сокращения множества.

Как видно из приведенных выше математических утверждений, предла-гаемый алгоритм работает лучше, чем существующие алгоритмы на основе древовидных чисел с точки зрения занимаемого пространства памяти и скорости поиска. Он требует гораздо меньше места для выполнения и имеет более высокую скорость поиска, чем существующие методы.

СПИСОК ЛИТЕРАТУРЫ:

1. Hanmandlu M, Verma O P, Susan S, Madasu V K, Color segmentation by fuzzy co-clustering of chrominance color features, Neurocomputing, 2013, 120: 235-249.

2. Jiang Y,Chung F,Wang S, Deng Z, Wang J, Qian P. Collaborative fuzzy clustering from multiple weighted views. IEEE Transactions on Cybernetics, 2015, 45(4): 688-701.pmid:25069132

3. P. Tsuchiya. "A search algorithm for table entries with non-contiguous wildcarding," unpublished report, Bellcore.

4. V. Srinivasan, S. Suri, G. Varghese, and M. Waldvogel. "Fast and Scalable Layer four Switching," Proceedings of ACM Sigcomm, pages 203-14, September 1998.

5. T.V. Lakshman and D. Stiliadis. "High-Speed Policy-based Packet Forwarding Using Efficient Multi-dimensional Range Matching", Proceedings of ACM Sigcomm, pages 191-202, September 1998.

6. V. Sahasranaman and M. Buddhikot, "Comparative evaluation of software implementation of layer 4 packet classifcation schemes," in Proc. IEEE ICNP, Nov. 2001, pp. 220-228.

7. P. Gupta and N. McKeown, "Packet classification using hierarchical intelligent cuttings," in Hot Interconnects VII, August 1999.