УДК 681.5 DOI:10.31854/1813-324X-2021-7-1-132-140

Программная методика оценки эффективности аппаратного состава серверов системы глубокой инспекции пакетов с использованием модернизированного метода Хука – Дживса

В.В. Фицов[1]

[1]Санкт-Петербургский государственный университет телекоммуникаций им. проф. М.А. Бонч-Бруевича, Санкт-Петербург, 193232, Российская Федерация
*Адрес для переписки: noldi@iks.sut.ru

Информация о статье
Поступила в редакцию 01.03.2021
Принята к публикации 17.03.2021

Ссылка для цитирования: Фицов В.В. Программная методика оценки эффективности аппаратного состава серверов системы глубокой инспекции пакетов с использованием модернизированного метода Хука – Дживса // Труды учебных заведений связи. 2021. Т. 7. № 1. С. 132–140. DOI:10.31854/1813-324X-2021-7-1-132-140

Аннотация: *Системы глубокой инспекции пакетов на сетях связи используются для распознавания приложения порождающего конкретный поток трафика. Вопросы, связанные с моделированием и проектированием систем глубокой инспекции пакетов, остаются малоизученными. В данной работе приводится программная методика оценки эффективности аппаратного состава серверов системы глубокой инспекции пакетов, использующая математическую модель такой системы и методы программного поиска. Дается описание программного поиска методом максимального элемента и методом Хука – Дживса. Предложена модернизация метода Хука – Дживса для монотонно убывающей функции. Проведено сравнение методов по числу шагов поиска.*

Ключевые слова: *глубокая инспекция пакетов, программный поиск, метод максимального элемента, метод Хука – Дживса, математическая модель.*

Введение

С распространением сетевых приложений, использующих доступные транспортные порты, не закрепляемые за приложением, в современных мультисервисных сетях обострилась проблема распознавания трафика. Оказалось, что без специальных технологий классификации трафика и инспектирования пакетов сетевое оборудование не различает данные различных приложений.

Наиболее точный анализ выполняют устройства глубокой инспекции пакетов DPI (*аббр. от англ.* Deep Packet Inspection). Для распознавания приложения по потоку пакетов используется специальный метод анализа (в том числе сигнатурного), обычно разрабатываемый и поддерживаемый вендором. Из-за большего числа существующих приложений поддержка сотен и тысяч методов довольно затратная. Кроме того, методы анализа требуют значительных аппаратных ресурсов. Наиболее производительные системы DPI используют специализированные аппаратные платы для разбора и анализа поступающих пакетов [1]. Все это вместе приводит к высокой стоимости оборудования DPI.

В ряде работ исследуются вопросы анализа при классификации трафика и глубокой инспекции пакетов. Например, в [2, 3] описываются параметры выявления потоков данных и способы анализа этих потоков. Сетевое приложение может инициировать передачу данных с помощью одного или нескольких потоков пакетов. Пакеты объединяются в поток с помощью MAC и IP-адресов, транспортных портов и типа протокола. Исследования количества анализируемых пакетов из потока трафика были проведены в [4, 5]. Алгоритмы комбинации решений были представлены в работах по классификации трафика [6, 7].

В [8] проведена оценка необходимых аппаратных ресурсов для извлечения идентификатора потока на аппаратном фильтре, для поиска/добавления идентификатора потока и поиска сигнатуры в тактах.

Б. Ньянг [9] предлагает математическую модель взаимодействия системы DPI с различными внешними серверами сетей связи. Однако такая математическая модель системы DPI не дает описания числа обслуживающих устройств и взаимодействия между серверами внутри системы. Общеизвестные подходы по определению необходимой вычислительной мощности оборудования систем глубокой инспекции пакетов основываются на параметрах максимальной скорости сетевых интерфейсов и числе одновременно установленных соединений, и не предполагают проведения расчетов. Вопросы, связанные с моделированием и проектированием систем DPI, остаются малоизученными. В связи с этим в [10] была предложена математическая модель DPI, состоящая из двух различных математических моделей. Подробнее об этом будет сказано ниже. Недостаточно внимания уделено эффективности использования аппаратных ресурсов в системах DPI из-за сложности и новизны проблемы. Использование методики оценки эффективности вариантов аппаратного состава серверов системы DPI позволило бы выявить необходимость в модернизации системы с целью повысить быстродействие и получить подходящую загрузку аппаратных ресурсов в системе.

В данной работе описывается применение разработанной программной методики для оценки эффективности работы системы DPI, кратко упомянутой ранее в [11]. Программная методика основана на представленной в [10] математической модели. Программный поиск осуществляется методом максимального элемента (ММЭ) [12] или модернизированным методом Хука – Дживса (MHJ), представленным в этой статье.

Особенности системы глубокой инспекции пакетов

Система DPI распознает приложения по потоку пакетов, проводит мониторинг трафика, собирает статистику, ограничивает скорость трафика по приложениям [13]. В значительном числе работ, например в [1, 13], рассказывается о способах применения системы DPI. А в [1, 9, 14] говорится о внутренней архитектуре системы DPI, состоящей из нескольких специализированных серверов. В [14] определяется ряд функциональных объектов системы DPI: сканирование (DPI-ScF, *аббр. от англ.* Scan Function), анализ (DPI-AnF, *аббр. от англ.* Analyser Function), выполнение действий (DPI-AcEF, *аббр. от англ.* Action Execution Function), выбор политики (PDF, *аббр. от англ.* Policy Decision Function). Зачастую функции сканирования и выполнения действий над потоками трафика (политик) объединены в сервере, который принято называть аппаратный фильтр (HWF, *аббр. от англ.* Hardware Filter) [9, 11]. Под политикой подразумевается набор правил по управлению доступом к ресурсам сети, применительно к проходящему через DPI трафику.

Каждый из серверов выполняет свои задачи и активно взаимодействует с остальными серверами системы DPI. Задачи аппаратного фильтра – выявлять потоки, пропускать или ограничивать известный трафик в соответствии с параметрами фильтрации, а также передавать на анализ неизвестный трафик. Сервер анализа проводит глубокую инспекцию пакетов потока и выявляет его принадлежность к определенному приложению. Сервер выбора политик для выявленного приложения указывает номер политики обработки его трафика. Сервера хранения информации сообщают инструкции по обработке трафика согласно выбранной политике, которые затем применяются на аппаратном фильтре.

Аппаратный фильтр системы DPI может работать в разных режимах, с точки зрения обработки пакетов неизвестного потока трафика, до того, как поток будет распознан. В первом режиме такие пакеты отбрасываются до тех пор, пока поток не будет определен. Во втором режиме пакеты неизвестного потока пропускаются согласно политике «по умолчанию», пока не будет получена уточненная политика. В третьем режиме пакеты потока буферизируются и пропускаются только по завершению анализа. Система DPI может быть представлена как сеть массового обслуживания (СеМО), состоящая из систем массового обслуживания (СМО). При расчете и проектировании такой СеМО возникает вопрос о необходимом числе обслуживающих устройств на каждом из СМО.

Математическая модель системы DPI

В [10] была предложена математическая модель DPI, состоящая из двух различных математических моделей, основанных на трудах И. Норроса [15], а также Е. Вентцель и Л. Овчарова [16]. Для аппаратного фильтра (СМО1) применяется математическая модель, основанная на [15], в которой поток поступления заявок описывается фрактальным броуновским движением. Для сервера анализа (СМО2) применяется математическая модель с бесконечной очередью и равномерным взаимодействием, о котором упоминалось в [16]. В [10] были представлены формулы расчета среднего времени нахождения заявок в системе DPI (1), и в частности для СМО1 (2) и СМО2 (4).

Время, затрачиваемое на серверах выбора политики ($\overline{T_3}$) и хранения информации ($\overline{T_4}$), не учитывалось, исходя из сравнительно небольшой нагрузки на эти сервера:

$$\overline{T_{dpi}} = \overline{T_1} + \overline{T_2} + \overline{T_3} + \overline{T_4}, \quad (1)$$

$$\overline{T_1} \approx \left(\frac{\rho}{(V_1 - \rho) \times m}\right) \times \times \exp\left(-\frac{(C-m)^{2\times H}}{2 \times \varphi(H)^2 \times a \times m} \times x^{2-2\times H}\right) + \frac{x}{m}, \quad (2)$$

где ρ – загрузка СМО; V_1 – число обслуживающих устройств СМО1; C – пропускная способность системы; m – средняя величина поступающего трафика; H – параметр Херста для самоподобного процесса; $\varphi(H)$ – коэффициент определяемый согласно (3); параметр a – характерный момент фрактального броуновского движения; x – среднее число заявок в СМО1.

$$\varphi(H) = H^H \times (1-H)^{1-H}, \quad (3)$$

$$\overline{T_2} = \frac{\left[\frac{1}{\sum_{i=1}^{h} i\times l\times\mu+\sum_{j=h+1}^{V_2} j\times\mu} + \frac{\beta}{V_2\times\mu}\times\frac{\alpha^h}{h!}\times\beta\times\frac{1}{(1-\beta)^2}\right]}{\left[\sum_{i=0}^{h}\frac{\alpha^i}{i!}+\frac{\alpha^h}{h!}\times\frac{\beta^{h+1}}{1-\beta}\right]}, \quad (4)$$

где h – максимально возможное количество групп обслуживающих устройств СМО2; l – число устройств в одной группе; μ – интенсивность обслуживания заявок СМО2; V_2 – число обслуживающих устройств СМО2.

Коэффициенты α и β определяются в (5) и (6):

$$\alpha = \frac{\lambda}{l\times\mu}, \quad (5)$$

$$\beta = \frac{\lambda}{V_2\times\mu}, \quad (6)$$

где λ – интенсивность поступающих заявок.

В программной методике формулы расчета среднего времени нахождения заявок в СМО1 (2) и СМО2 (4) будут использоваться для оценки эффективности числа обслуживающих устройств на аппаратном фильтре и сервере анализа системы DPI.

Программный поиск в методике оценки эффективности

Определить подходящее под заданные условия число обслуживающих устройств для серверов системы DPI можно, используя упомянутую выше математическую модель. При этом предлагается использовать методы программного поиска. Особенностью данной задачи является предположение о том, что с ростом числа обслуживающих устройств среднее время обработки заявки в системе будет снижаться. Таким образом, можно считать, что функция среднего времени обработки заявки от числа обслуживающих устройств будет монотонно убывающей. Данное обстоятельство значительно упрощает реализацию программных алгоритмов поиска эффективного распределения аппаратных ресурсов по серверам системы DPI.

Для программного поиска используют теорему Куна и Такера, ММЭ, метод динамического программирования, метод неопределенности множеств Лагранжа, метод Гаусса – Зейделя, метод Хука – Дживса (HJ-метод), метод Нелдера – Мида, методы случайного поиска и другие.

Одним из самых простых методов программного поиска для малого числа шагов является ММЭ. Об использовании ММЭ рассказывалось ранее в [12]. Альтернативным рекомендуемым подходом по сравнению с ММЭ может выступать HJ-метод, разработанный в 1961 г. [17], но до сих пор являющийся весьма эффективным и оригинальным. HJ-метод более сложный в программной реализации, но предполагает более быстрый поиск (т. е. требует меньшего числа шагов поиска).

Был разработан программный код на языке python с применением ММЭ и HJ-метода для оценки эффективности аппаратного состава серверов системы DPI. Алгоритм работы программной методики оценки числа устройств специализированных серверов системы DPI представлен на рисунке 1.

В ходе оценки эффективности аппаратного состава системы DPI будет проводиться расчет среднего времени нахождения заявки в системе с изменением числа обслуживающих устройств на серверах DPI согласно выбранному шагу по ММЭ или HJ. Для проведения расчетов согласно формулам математической модели, в программном коде на языке python требуются циклы для расчета математических сумм, а также элементарные математические функции библиотеки math для расчета факториала (factorial) и степени (pow) указанного числа.

В качестве шага по ММЭ выбрано увеличение числа обслуживающих устройств на 1 вплоть до 100. А начать расчет среднего времени нахождения заявки в системе DPI целесообразно с одного устройства. Т. к. математическая модель DPI дает описание для СМО1 и СМО2, то и программный поиск проводится для каждого из двух СМО. В таблице 1 даны значения среднего времени анализа потока трафика системой DPI, полученные с помощью программных функций расчета по математической модели DPI для разного числа устройств в СМО1 и СМО2. В качестве исходных данных для расчетов использовались значения набора параметров, полученных на основе статистического анализа трафика общежитий СПбГУТ, приведенные в [10].

ТАБЛИЦА 1. Среднее время нахождения заявки в системе DPI при увеличении числа обслуживающих устройств СМО1 и СМО2

TABLE 1. Average Time Spent by a Request in the DPI System with an Increase in the Number of Servicing Devices QS1 and QS 2

V_1, шт.	V_2, шт.	T_{dpi}, с	V_1, шт.	V_2, шт.	T_{dpi}, с	V_1, шт.	V_2, шт.	T_{dpi}, с
1	1	0,1369	2	1	0,1203	3	1	0,1147
1	2	0,0337	2	2	0,0171	3	2	0,0115
1	3	0,0334	2	3	0,0167	3	3	0,0112
1	4	0,0333	2	4	0,0167	3	4	0,0111

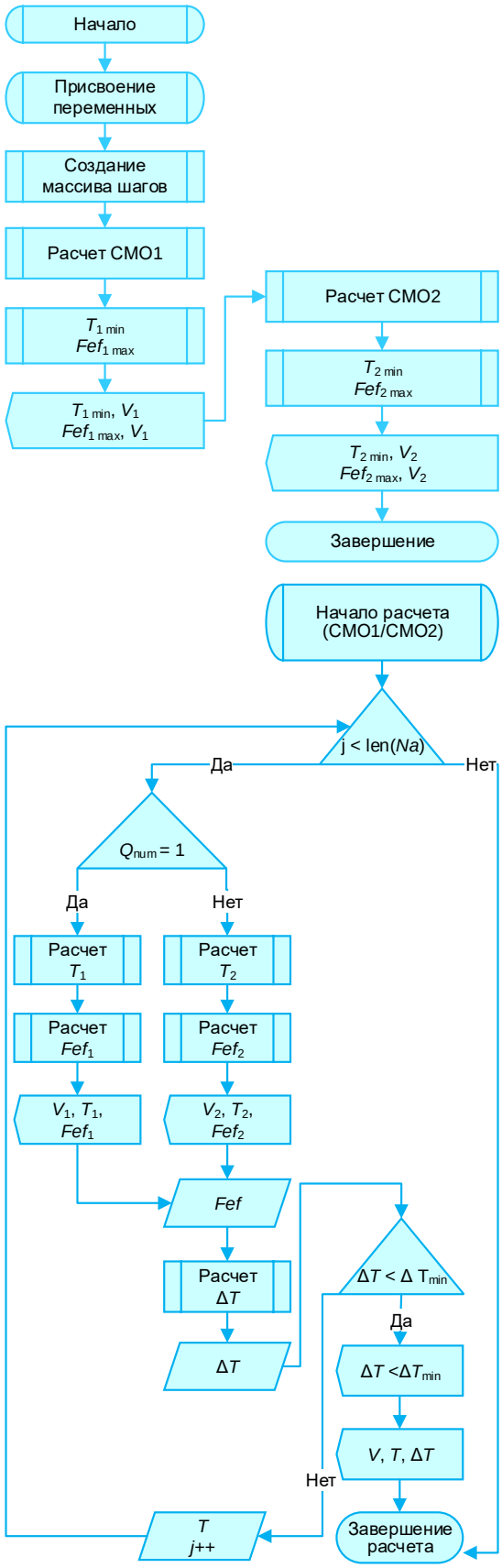

Рис. 1. Алгоритм логики работы основной программы оценки эффективного аппаратного состава серверов системы DPI

Fig. 1. Algorithm of the Logic for Main Program of the Effective Equipment Composition of the DPI Server System

Значения среднего времени нахождения заявки в системе DPI, приведенные в таблице 1, подтверждают предположение об их монотонно убывающем характере с ростом числа обслуживающих устройств на серверах системы DPI. Увеличение числа устройств в СМО1 и СМО2 с различной степенью влияет на снижение среднего времени нахождения заявки в системе DPI.

Подобные расчеты позволяют определить подходящую комбинацию числа обслуживающих устройств для выполнения требования по максимально допустимому времени обнаружения потока трафика и применения к нему соответствующих политик. Однако если исходить из того, что дальнейшее снижение времени нахождения заявки в системе DPI дает некоторое преимущество, то возникает вопрос: до какого числа обслуживающих устройств СМО1 и СМО2 целесообразно наращивать производительность системы при заданных исходных данных, предполагающих поток поступающих заявок на определенном статистикой уровне.

Во-первых, увеличение числа обслуживающих устройств имеет смысл при ощутимом снижении времени нахождения заявки в системе. Как видно из таблицы 1, с ростом числа устройств эффект в виде снижения времени нахождения заявки в системе уменьшается. Поэтому в программную методику оценки эффективности был введен параметр минимально целесообразного снижения среднего времени нахождения заявки в системе (ΔT) при увеличении обслуживающих устройств на 1. Для примера в качестве ощутимого для сети изменения времени нахождения заявки в системе DPI было выбрано значение в 0,1 мс.

Во-вторых, рост количества обслуживающих устройств означает увеличение затрат на покупку оборудования, потребление электроэнергии и прочего. В данной статье не ставится задача по исследованию экономической целесообразности того или иного решения, а только показывается наличие такого влияния и возможность его практического внедрения в метод проектирования системы DPI. Таким образом, в программную методику оценки эффективности была введена функция стоимости, ранее представленная в [12], которая учитывает стоимость оборудования, амортизацию, издержки на потребление электроэнергии, недополученную прибыль по причине более высокой загруженности сети оператора связи, при менее эффективной работе системы DPI. При этом стоимость устройств для СМО1 и СМО2 может быть различной.

Для тех же исходных данных, взятых из [10], было выявлено, что снижение среднего времени нахождения заявки в системе DPI становится менее 0,1 мс при V_1 = 18 и V_2 = 3. Далее с увеличением числа приборов время изменяется незначительно, а функция эффективности указала на комбинацию

обслуживающих устройств $V_1 = 3$ и $V_2 = 2$. Это означает, что при заданных параметрах функции эффективности не целесообразно далее увеличивать число обслуживающих устройств. В таблицу 2 сведены 3 комбинации распределения обслуживающих устройств по СМО1 и СМО2 в системе DPI. Комбинация $V_1 = 1$ и $V_2 = 1$ дана для сравнения.

ТАБЛИЦА 2. Комбинации аппаратного состава серверов системы DPI

TABLE 2. Hardware Combinations of DPI Servers

V_1, шт.	V_2, шт.	T_1, с	T_2, с	T_{dpi}, с
3	2	0,0111	0,0004	0,0115
18	3	0,0018	0,0001	0,0019
1	1	0,0333	0,1036	0,1369

Как было описано выше, методика оценки эффективности представлена следующими этапами: формирование исходных данных; определение ограничений для допустимого значения нахождения заявки в системе DPI и увеличения обслуживающих устройств серверов DPI; выполнение шагов поиска эффективного числа обслуживающих устройств, где на каждом шаге выполняется расчет по выбранным формулам (в данном случае по (2) и/или (4)); выбор подходящей комбинации числа обслуживающих устройств.

Таким образом, программная методика оценки эффективности аппаратного состава серверов системы DPI предлагает одну из комбинаций, которую можно получить самостоятельно логическим путем из таблицы 1. В случаях, когда может потребоваться несколько десятков шагов поиска ММЭ, имеет смысл применять другие методы поиска, например, HJ-метод.

Модернизация метода Хука – Дживса

Рассмотрим применение в программной методике оценки эффективности аппаратного состава серверов системы DPI HJ-метода. Он состоит из последовательности шагов исследующего поиска вокруг базисной точки, за которой в случае успеха следует поиск по образцу. Существует несколько улучшений HJ-метода. Например, в [18] предложен гибридный алгоритм HJ-методом с локальным поиском, в котором сканирование пространства переменных проводится с использованием кратного алгоритма столкновения частиц. В HJ-методе возможно проведение многомерного поиска, что может быть удачно использовано для проводимого одновременно поиска подходящего числа серверов для нескольких СМО в составе системы DPI.

HJ-метод состоит из следующих этапов. Определяется величина базисной точки для начала поиска (b_0 – точка начала расчетов). Рассчитывается значение времени нахождения заявки в системе в базисной точке. Вычисляется размер шага поиска (St), который задает точность полученного результата поиска. В случае желаемого изменения времени нахождения заявки в системе делается еще один шаг в данном направлении.

Если изменение не произошло или было противоположным, то делается шаг в противоположную сторону. В случае успеха он так же может быть повторен. При многомерном поиске такая процедура повторяется для базиса и шага другой переменной в составе времени нахождения заявки в системе.

Если шаги не приводят к успеху – это означает, что было найдено предварительное решение с точностью, соответствующей величине шага. Тогда результат поиска уточняется, для чего производится снижение величины шага (например, вдвое) вплоть до 10-кратного уменьшения величины шага.

Если процедура была успешна после выполнения одного или двух шагов, то наиболее успешная точка принимается за новую базисную точку, и начинается поиск по образцу. Упрощенный алгоритм поиска с применением базисной точки (b_j), изменения величины шага и поиска по образцу по HJ-методу представлен на рисунке 2. Последний предполагает продолжение поиска в успешном направлении (P_j – значение для проведения следующего исследования) согласно (7).

Для вычисления P_j используется значение предпоследней базисной точки (b_j) с добавлением удвоенной разницы последней точки (b_{j+1}) и b_j:

$$P_j = b_j + 2 \times (b_{j+1} - b_j). \quad (7)$$

Исходя из выше представленного описания части алгоритма работы HJ-метода, соседние базисные точки разделяют несколько шагов. Обозначим n за число необходимых шагов, а St_j является величиной шага после базисной точки (b_j).

Тогда (7) можно переписать в виде (8), который не является наиболее удобным, т. к. не всегда применим, и приведен исключительно для большей наглядности процедуры поиска по образцу:

$$P_j = b_j + 2 \times n \times St_j. \quad (8)$$

Когда изменение времени нахождения заявки в системе для нового базиса окажется меньше 0,1 мс, согласно HJ-методу, проводится уменьшение шага поиска до минимально доступного. Как уже говорилось выше, обычно на практике шаг уменьшают до 10 раз, после чего поиск считается завершенным. Более подробно HJ-метод описан в [17]. Для определения числа обслуживающих устройств серверов системы DPI необходима точность, как минимум соответствующая одному аппаратному устройству. А процедура уточнения путем уменьшения величины шага приводит к значительному увеличению числа необходимых для получения точного решения шагов.

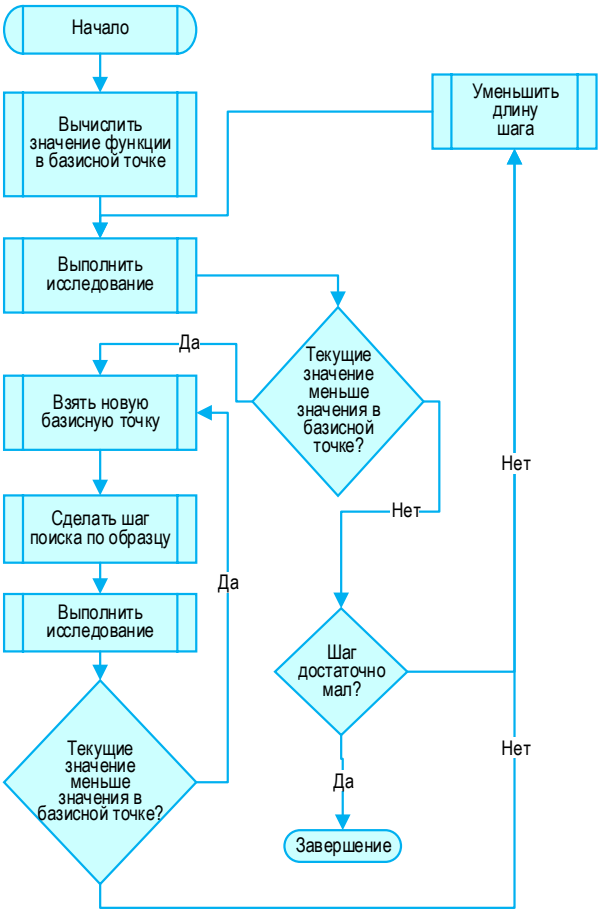

Рис. 2. Упрощенный алгоритм работы HJ-метода

Fig. 2. Simplified Algorithm of the Hooke – Jeeves Method

Применение HJ-метода несколько раз для одной и той же СМО, с различной точностью изначального шага, приводит к избыточному числу необходимых шагов. Способ, позволяющий снизить число шагов поиска по HJ-методу, будет представлен далее.

Для начала поиска HJ-метода следует выбрать базисную точку (b_0). При расчете для одной СМО можно воспользоваться упрощенной или усложненной формулой, для получения условного числа обслуживающих устройств (V_b) для начала расчетов проводимых алгоритмом.

В качестве упрощенной формулы можно использовать (9) – соотношение интенсивности поступающих заявок (λ) к интенсивности обработки заявок (μ) с учетом предполагаемого коэффициента загрузки системы (ρ_b):

$$V_b = \frac{\lambda}{\mu \times \rho_b} = b_0. \qquad (9)$$

В уточненной формуле расчет должен основываться на предполагаемом времени нахождения заявки в рассчитываемой СМО. Для расчета значения базисной точки используются исходные данные представленные в [10].

Величина шага (St) в сторону от базиса, для монотонно убывающих функций может быть задана как (10). Тогда величина шага будет снижаться по мере удаления значения времени нахождения заявки в системе (T_{dpi}) от своего максимально допустимого значения ($T_{\max}$):

$$St = b_0 \times \frac{T_{dpi}}{T_{\max}}. \qquad (10)$$

Однако при $b_0 < 4$, для сокращения числа шагов поиска оказалось более практично задавать значение шага равным минимально допустимому значению шага ($St = St_{\min}$). В случаях более 15 шагов поиска, использование стандартного HJ-метода с минимально допустимым шагом требует избыточного числа шагов.

Для случаев, когда число шагов может достигать 10–15 и более, эмпирическим путем был получен MHJ. В нем за шаг принимается десятикратное значение минимально допустимого шага ($St = 10 \times St_{\min}$). Однако, при обнаружении ситуации, в которой изменение времени нахождения заявки в системе за удельный шаг ($St/St_{\min}$) меньше минимально целесообразного изменения, следует уменьшить величину шага вдвое. А затем использовать в качестве базисной точки последнее значение, при котором изменение времени было больше минимально целесообразного значения (ΔT). Кроме того, необходимо использовать аналогичное предпоследнее значение как запасную базисную точку. Такой MHJ становится более эффективен с возрастанием числа шагов поиска, что видно из таблицы 3 и рисунка 3, где показано сравнение использования следующих методов программного поиска: ММЭ, HJ-метода, MHJ.

MHJ был положен в основу соответствующей функции в программной методике оценки эффективности аппаратного состава серверов системы DPI.

Результаты использования программной методики оценки эффективности

Используя (9) для исходных данных, представленных в [10], можно получить величину базиса для СМО1 и СМО2.

Для упрощения задачи минимально допустимая величина шага взята за одно аппаратное устройство ($St_{\min} = 1$):

– для HJ-метода задан шаг 1 ($St = St_{\min} = 1$);

– для MHJ задан шаг 10 ($St = 10 \times St_{\min} = 10$).

Результаты поиска (эффективное число обслуживающих устройств (V) и среднее время нахождения заявки в СМО (T)), необходимое число шагов (n) и заданная точность представлены в таблице 3. Результаты в этих таблицах были получены без учета функции стоимости, которая обычно прекращает программный поиск при меньшем числе устройств.

ТАБЛИЦА 3. Поиск числа обслуживающих устройств СМО1/СМО2

TABLE 3. Search of the Number of Servicing Devices QS1/QS2

Метод	ММЭ			HJ-метод			MHJ
Базис (b_0), шт.	–/–	–/–	–/–	2/2	2/2	2/2	2/2
Шаг (St), шт.	1/1	2/2	2/2	10/10	10/10	1/1	10/10
St_{min}, шт.	1/1	1/1	1/1	1/1	1/1	1/1	1/1
Точность шага, шт.	1/1	2/2	1/1	10/10	1/1	1/1	1/1
Число шагов (n), шт.	19/4	10/3	12/4	4/2	28/4	12/3	12/5
V, шт.	18/3	18/3	18/3	22/2	18/3	18/3	18/3
ΔT, мс	0,1/0,1	0,1/0,1	0,1/0,1	0,1/0,1	0,1/0,1	0,1/0,1	0,1/0,1
T, с	1,85/0,09	1,85/0,09	1,85/0,09	1,51/0,42	1,85/0,09	1,85/0,09	1,85/0,09

Следует еще раз подчеркнуть значение крайних правых столбцов таблицы 3, где представлен результат работы MHJ по сравнению с ММЭ и оригинальным HJ-методом. Таким образом, рекомендуемое число серверов для СМО1 и СМО2 (см. таблицу 2) может быть расчитано как с помощью метода ММЭ, так и с помощью MHJ.

На рисунке 3 показано сравнение ММЭ, HJ-метода и MHJ по числу шагов при изменении значения минимально целесообразного снижения среднего времени нахождения заявки в системе (ΔT).

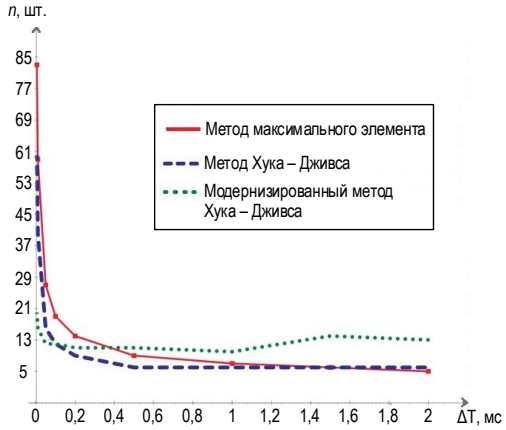

Рис. 3. Сравнение методов по числу шагов

Fig. 3. Comparison of Methods by Number of Steps

Изменение величины ΔT было выбрано для демонстрации работы методов программного поиска. Подобное влияние могут оказывать и другие исходные данные для расчетов.

ММЭ более эффективен при числе шагов менее 10, а также может быть рекомендован, когда необходима простейшая программная реализация поиска. Однако в большинстве случаев более предпочтителен MHJ, требующий более сложную программную реализацию. В случаях, когда требуется более 10 шагов ММЭ, использование MHJ снизит число необходимых шагов поиска. За счет использования поиска по образцу выполнение расчетов по HJ-методу будет выполняться быстрее, чем при использовании ММЭ.

Заключение

В данной статье были определены условия целесообразности применения для монотонно убывающей функции MHJ по сравнению с ММЭ применительно к программной методике оценки эффективного аппаратного состава серверов системы глубокой инспекции пакетов. Также была упомянута возможность многомерного поиска с помощью HJ-метода для данной задачи, которая будет приведена в последующих работах.

Представленная программная методика позволила определить эффективные численные комбинации распределения обслуживающих устройств по серверам системы глубокой инспекции пакетов для функционирования в заданных условиях с учетом следующих критериев: времени нахождения заявки в системе, минимально целесообразным изменением этого времени при увеличении на одно обслуживающее устройство, функции стоимости системы.

Использование программной методики позволяет выявить необходимость модернизации системы, с целью повысить быстродействие и эффективность использования аппаратных ресурсов в системе глубокой инспекции пакетов. А также повысить эффективность работы системы за счет перераспределения аппаратных ресурсов в режиме реального времени.

Список используемых источников

1. Сенченко Ю.Л. Некоторые аспекты высокоскоростной обработки трафика // Технологии и средства связи. 2013. № 1(94). С. 52–53.

2. Trammell B., Boschi E., Procissi G., Callegari C., Dorfinger P., Schatzmann D. Identifying Skype Traffic in a Large-Scale Flow Data Repository // Proceedings of the 3rd International Workshop on Traffic Monitoring and Analysis (TMA, Vienna, Austria, 27 April 2011). Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2011. Vol. 6613. PP. 72–85. DOI:10.1007/978-3-642-20305-3_7

3. Sommer R., Feldmann A. NetFlow: Information loss or win? // Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurment (IMW '02, Marseille, France, November, 2002). New York: Association for Computing Machinery, 2002. PP. 173–174. DOI: 10.1145/637201.637226

4. Park J., Yoon S., Kim M. Software Architecture for a Lightweight Payload Signature-Based Traffic Classification System // Proceedings of the 3rd International Workshop on Traffic Monitoring and Analysis (TMA, Vienna, Austria, 27 April 2011). Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2011. Vol. 6613. PP. 136–149. DOI:10.1007/978-3-642-20305-3_12

5. Deart V., Mankov V., Krasnova I. Agglomerative Clustering of Network Traffic Based on Various Approaches to Determining the Distance Matrix // Proceedings of the 28th Conference of Open Innovations Association (FRUCT'28, Moscow, Russia, 27–29 January 2021). IEEE, 2021. Vol. 1. PP. 81–88. DOI:10.23919/FRUCT50888.2021.9347616

6. Dainotti A., Pescape A., Sansone C. Early Classification of Network Traffic through Multi-classification // Proceedings of the 3rd International Workshop on Traffic Monitoring and Analysis (TMA, Vienna, Austria, 27 April 2011). Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2011. Vol. 6613. PP. 122–135. DOI:10.1007/978-3-642-20305-3_11

7. Sheluhin O., Kazhemskiy M. Influence Of Fractal Dimension Statistical Charachteristics On Quality Of Network Attacks Binary Classification // Proceedings of the 28th Conference of Open Innovations Association (FRUCT'28, Moscow, Russia, 27–29 January 2021). IEEE, 2021. Vol. 1. 2021. PP. 407–413. DOI:10.23919/FRUCT50888.2021.9347600

8. Cascarano N., Ciminiera L., Risso F. Optimizing Deep Packet Inspection for High-Speed Traffic Analysis // Journal of Network and Systems Management. 2011. Vol. 19. PP. 7–31. DOI:10.1007/s10922-010-9181-x

9. Niang B. Bandwidth management – A deep packet inspection mathematical model // Proceedings of the 6th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT, St. Petersburg, Russia, 6-8 October 2014). IEEE, 2014. PP. 169–175. DOI:10.1109/ICUMT.2014.7002098

10. Goldstein B., Fitsov V. Dual Mathematical Model for Calculating of Deep Packet Inspection // Proceedings of the 28th Conference of Open Innovations Association (FRUCT'28, Moscow, Russia, 27–29 January 2021). IEEE, 2021. Vol. 1. 2021. PP. 127–133. DOI:10.23919/FRUCT50888.2021.9347547

11. Фицов В. Методы построения сетевых архитектур систем DPI // Вестник связи. 2020. № 12. С. 32–37.

12. Фицов В. Применение программного кода для оптимизации числа серверов DPI методом максимального элемента // VII Международная научно-техническая и научно-методическая конференция «Актуальные проблемы инфотелекоммуникаций в науке и образовании» (Санкт-Петербург, Россия, 28 февраля – 01 марта 2018). СПб: СПбГУТ, 2018. С. 650–656.

13. Якимович С. Управление трафиком и услугами в сетях ШПД с помощью решений DPI // Вестник связи. 2010. № 12. С. 27–29.

14. Rec. ITU-T Y.2771 Framework for deep packet inspection. ITU, 2014.

15. Norros I. A storage model with self-similar input // Queueing Systems. 1994. Iss. 16. PP. 387–396. DOI:10.1007/BF01158964

16. Овчаров Л. Прикладные задачи теории массового обслуживания. М.: Машиностроение, 1969. 324 с.

17. Hooke R., Jeeves T.A. «Direct Search» Solution of Numerical and Statistical Problems // Journal of the ACM. 1961. Vol. 8. Iss. 2. PP. 212–229. DOI:10.1145/321062.321069

18. Сулимов В., Шкапов П., Носачев С. Локальный поиск методом Хука – Дживса в гибридном алгоритме глобальной оптимизации // Наука и образование. 2014. № 6. С. 107–123. DOI:10.7463/0614.0716155

* * *

Software Methodology for Estimating the Efficiency of the Hardware Composition of Deep Packet Inspection System Using the Modernized Hooke – Jeeves Method

V. Fitsov[1]

[1]The Bonch-Bruevich Saint-Petersburg State University of Telecommunications,
St. Petersburg, 193232, Russian Federation

Article info
DOI:10.31854/1813-324X-2021-7-1-132-140
Received 01st March 2021
Accepted 17th March 2021

For citation: Fitsov V. Software Methodology for Estimating the Efficiency of the Hardware Composition of Deep Packet Inspection System Using the Modernized Hooke – Jeeves Method. *Proc. of Telecom. Universities.* 2021;7(1):132–140. (in Russ.) DOI:10.31854/1813-324X-2021-7-1-132-140

Abstract: *Deep packet inspection systems on communication networks are used to identify the application generating a specific traffic flow. The issues related to modeling and design of deep packet inspection systems remain poorly understood. In this paper, a software technique for evaluating the effectiveness of the hardware composition of the servers of the deep packet inspection system is presented, using a mathematical model of such a system and software search methods. The description of the program search by the maximum element method and the Hook-Jeeves method is given. A modernization of the Hook-Jeeves method for a monotonically decreasing function is proposed. Comparison of the methods by the number of search steps is performed.*

Keywords: *programmatic search, maximum element method, Hooke-Jeeves method, mathematical model.*

References

1. Senchenko Yu.L. Some Aspects of High-Speed Traffic Processing. *Tekhnologii i sredstva sviazi*. 2013;1(94):52–53. (in Russ.)
2. Trammell B., Boschi E., Procissi G., Callegari C., Dorfinger P., Schatzmann D. Identifying Skype Traffic in a Large-Scale Flow Data Repository. *Proceedings of the 3rd International Workshop on Traffic Monitoring and Analysis, TMA, 27 April 2011, Vienna, Austria*. *Lecture Notes in Computer Science*. Berlin, Heidelberg: Springer; 2011. vol.6613. p.72–85. DOI:10.1007/978-3-642-20305-3_7
3. Sommer R., Feldmann A. NetFlow: Information loss or win? *Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurment, IMW '02, November, 2002, Marseille, France*. New York: Association for Computing Machinery; 2002. p.173–174. DOI: 10.1145/637201.637226
4. Park J., Yoon S., Kim M. Software Architecture for a Lightweight Payload Signature-Based Traffic Classification System. *Proceedings of the 3rd International Workshop on Traffic Monitoring and Analysis, TMA, 27 April 2011, Vienna, Austria. Lecture Notes in Computer Science*. Berlin, Heidelberg: Springer; 2011. vol.6613. p.136–149. DOI:10.1007/978-3-642-20305-3_12
5. Deart V., Mankov V., Krasnova I. Agglomerative Clustering of Network Traffic Based on Various Approaches to Determining the Distance Matrix. *Proceedings of the 28th Conference of Open Innovations Association, FRUCT'28, 27–29 January 2021, Moscow, Russia*. IEEE; 2021. vol.1. p.81–88. DOI:10.23919/FRUCT50888.2021.9347616
6. Dainotti A., Pescape A., Sansone C. Early Classification of Network Traffic through Multi-classification. *Proceedings of the 3rd International Workshop on Traffic Monitoring and Analysis, TMA, 27 April 2011, Vienna, Austria. Lecture Notes in Computer Science*. Berlin, Heidelberg: Springer; 2011. vol.6613. p.122–135. DOI:10.1007/978-3-642-20305-3_11
7. Sheluhin O., Kazhemskiy M. Influence Of Fractal Dimension Statistical Charachteristics On Quality Of Network Attacks Binary Classification. *Proceedings of the 28th Conference of Open Innovations Association, FRUCT'28, 27–29 January 2021, Moscow, Russia*. IEEE; 2021. vol.1. p.407–413. DOI:10.23919/FRUCT50888.2021.9347600
8. Cascarano N., Ciminiera L., Risso F. Optimizing Deep Packet Inspection for High-Speed Traffic Analysis. *Journal of Network and Systems Management*. 2011;19:7–31. DOI:10.1007/s10922-010-9181-x
9. Niang B. Bandwidth management – A deep packet inspection mathematical model. *Proceedings of the 6th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops, ICUMT, St. Petersburg, Russia, 6-8 October 2014*. IEEE; 2014. p.169–175. DOI:10.1109/ICUMT.2014.7002098
10. Goldstein B., Fitsov V. Dual Mathematical Model for Calculating of Deep Packet Inspection. *Proceedings of the 28th Conference of Open Innovations Association, FRUCT'28, 27–29 January 2021, Moscow, Russia*. IEEE; 2021. vol.1. p.127–133. DOI:10.23919/FRUCT50888.2021.9347547
11. Fitsov V. Methods for Constructing Network Architectures of Dpi Systems. *Vestnik Svyazi*. 2020;12:32–37. (in Russ.)
12. Fitsov V. Employment Software Code for Optimization the Number of Dpi-Servers by Maximal Element Method. *Proceedings of the VIIIth International Conference on Infotelecommunications in Science and Education, 28 February – 1 March 2018, St. Petersburg, Russia*. St. Petersburg: The Bonch-Bruevich Saint-Petersburg State University of Telecommunications Publ.; 2018. p.650–656. (in Russ.)
13. Yakimovich, S. Controlling Traffic and Services in Broadband Networks Using DPI Solutions. *Vestnik Svyazi.* 2010;12: 27–29. (in Russ.)
14. Rec. ITU-T Y.2771 *Framework for deep packet inspection.* ITU; 2014.
15. Norros I. A storage model with self-similar input. *Queueing Systems*. 1994;16:387–396. DOI:10.1007/ BF01158964
16. Ovcharov L. *Applied Problems of Queuing Theory*. Moscow: Mashinostroenie Publ; 1969. 324 p. (in Russ.)
17. Hooke R., Jeeves T.A. «Direct Search» Solution of Numerical and Statistical Problems. *Journal of the ACM*. 1961;8(2): 212–229. DOI:10.1145/321062.321069
18. Sulimov V.D., Shkapov P.M., Nosachev S.K. Hooke–Jeeves Method-used Local Search in a Hybrid Global Optimization Algorithm. *Science and Education.* 2014;6:107–123. DOI:10.7463/0614.0716155 (in Russ.)

Сведения об авторе:

Фицов Вадим Владленович
аспирант кафедры Инфокоммуникационных систем Санкт-Петербургского государственного университета телекоммуникаций им. проф. М.А. Бонч-Бруевича,
noldi@iks.sut.ru
https://orcid.org/0000-0002-3226-927X