Содержание

Введение

Классификация трафика представляет собой важную задачу, так как полученные результаты используются в разных приложениях, важных для администрирования сети и для конечного пользователя.

С точки зрения провайдера идентификация протоколов, приложений или типов приложений по потокам данных в сети может использоваться для:

- контроля сети и трафика в ней (например, для блокировки протоколов, таких как BitTorrent),
- предоставления высокого качества сервиса при помощи эффективного определения наиболее приоритетных потоков и изменения скорости передачи отдельных пакетов,
- изменения цен на услуги,
- планирования размещения и использования ресурсов,
- оптимизации предоставляемых сервисов и алгоритмов маршрутизации (например, для регулирования приоритетов передачи различных типов данных в случае высокой загруженности сети).

В силу того, что потребности пользователей касательно использования Сети постоянно меняются, необходимо их знать и модифицировать Сеть в соответствии с актуальными запросами. Например, на сегодняшний день видна тенденция отказа от господствующего ранее принципа асимметрии устройства сети в том смысле, что клиенты загружают намного больше информации, чем отправляют её в Сеть. Появление P2P-приложений, VoIP, видеозвонков, потоковой передачи мультимедиа и прочих вещей должно вызвать у интернет-провайдеров соответствующие ответные действия по переустройству сети под новые запросы клиентов. Помимо этого, в настоящее время возрастает число «умных устройств», которые составляют Интернет вещей: он тоже поставит перед интернет-провайдерами новые задачи для обеспечения максимальной эффективности своей работы.

Также следует вспомнить мобильные приложения, чья доля в интернет-трафике постоянно растёт. Использование смартфонов и мобильных приложений можно считать более персонализированным по сравнению с «десктопным» трафиком, поэтому получение данных о такого рода трафике позволяет более эффективно составить сетевой портрет пользователя. Выяснение интересов и предпочтений пользователей используется для маркетинга, позволяя проводить лучшие таргетированные рекламные кампании.

С точки зрения обеспечения безопасности, классификация сетевого трафика может использоваться как важный признак при выявлении неправомерных или необычных действий пользователя, аномалий в работе Сети, кибер-атак и прочих нарушений, что может повысить общую безопасность Сети.

Применяемые для классификации интернет-трафика методы изменяются вместе с глобальными изменениями в устройстве трафика. К таким глобальным изменениям, влияющим на решение задачи классификации трафика можно отнести:

- отказ от использования утверждённого списка портов в зависимости от протокола/приложения (намеренный или в связи с устареванием данного списка);
- обфускация протоколов с целью замаскировать те из них, которые блокируются или подавляются провайдером;
- широкое распространение шифрования трафика, которой не позволяет использовать для классификации содержимое полезной нагрузки пакета;
- постоянное появление новых протоколов и приложений и т.д.

По указанным выше причинам, задачу классификации сетевого трафика на данный момент нельзя считать решённой, и исследовательские группы продолжают предлагать всё новые решения, которые позволяют показывать более эффективные результаты в условиях беспрестанно меняющейся ситуации.

1. ИССЛЕДОВАТЕЛЬСКИЙ РАЗДЕЛ

1.1 История развития средств анализа сетевого трафика

Технологии для анализа сетевого трафика начали появляться в 90-х годах прошлого века. Необходимость в них возникла примерно в одно время в нескольких областях.

Из-за усложнения схем сетей и большого разнообразия сетевых устройств началось и усложнение их настройки, а также поддержки сети в работоспособном состоянии - необходим был инструмент, который позволял бы с одной стороны локализовать проблему, а с другой предоставить как можно более исчерпывающую информацию о природе проблемы. Собственно объектом, который содержит в себе всю необходимую информацию и является сетевой трафик.

Одним из инструментов, изначально предназначенным для решения этой проблемы стал сетевой сниффер [1] и анализатор Wireshark [2] (ранее Ethereal), созданный инженером Джеральдом Комбом (Gerald Comb) в 1997 году. Wireshark продолжает активно развиваться и является стандартом в определённой области сетевого анализа.

Примерно в это же время начинает активно применяться технология трансляции адресов NAT [3], созданная для того, чтобы сэкономить IP адреса, а также для того, чтобы внешний наблюдатель не мог увидеть ресурсы и устройство внутренней локальной сети. Для реализации этой технологии требовался инструмент - аппаратный или программный транслятор адресов. В результате данный функционал был внедрён в качестве составной части в большинство маршрутизаторов. Существуют и программные реализации в составе некоторых серверных операционных систем, а также в виде отдельных приложений [4].

К этому же времени относятся первые упоминания о вирусах и DoS и DDoS [5] атаках, в основном типа SYN-флуд - первое упоминание о DDoS относится к

1996 году. Для защиты от этих угроз требовался инструмент, который может анализировать и фильтровать пакеты до их попадания на основной сервер. Одним из видов таких защит стали межсетевые экраны (firewall). Первое поколение данных решений относилось к типу пакетных фильтров (packet filters), которые обрабатывали пакеты по одному (не учитывая предысторию) и анализировали только уровни L1-L3 модели OSI и (для протоколов TCP/UDP) номера портов из транспортного уровня L4 (см. рис. 1). Для определения типа трафика (web, email и т.д.) использовался список фиксированных номера портов из каталога IANA [6]. Процесс анализа заключался в сравнении данных, извлечённых из пакета, с набором заданных правил и, в зависимости от результата — блокировка или пропуск пакета в сеть с занесением события в журнал и опциональным уведомлением источника пакета о ситуации. Например, правило «Блокировка Telnet трафика» выглядело, как правило, описывающее пакеты, транспортный протокол которых — TCP, номер целевого порта — 23, а действие при выявлении такого пакета — блокировка. Одним из первых подобных решений был продукт DEC SEAL.

Ближе к концу 90х — началу 2000х годов, в связи с ростом сетевых потоков данных, актуальными стали ещё две задачи, требовавшие сетевого анализа: балансировка нагрузки между серверами и ускорение работы отдельных видов сетевых приложений. К сетевым приложениям, требовавшим ускорения, относились, прежде всего, приложения, использующие протоколы HTTP, DNS, SSL [7]. Для решения второй проблемы использовались, т.н. прокси-сервера, осуществляющие кэширование поступающих данных, минимизируя, так образом, обмены по сети. Устройства, разработанные для решения обеих этих задач (инкапсулирующие, в частности, функционал прокси-серверов) носили название контроллеры доставки приложений (Application delivery controllers, ADC). Такие решения в частности были разработаны компаниями Alteon, Radware, F5, Brocade, Cisco. В первой половине 2000х годов сетевые технологии получили бурное развитие — появились средства голосового обмена по сети (VoIP) и обмена данными в одноранговых сетях P2P (Napster, KaZaA), что, в

частности, привело к очередному резкому скачку объёмов передаваемых по сети данных. Для развивающихся сетей крупных корпораций потребовалось объединять в единую локальную сеть территориально разнесённые площадки. Более частыми и сложными стали сетевые атаки, что требовало более развитых средств защиты. Для реализации передачи управляющих сигналов и данных VoIP с использованием таких протоколов как SIP[8] и RTP[9] между различными провайдерами, как телефонной связи, так и интернета требовались специальные устройства – пограничные контроллеры сессий (session border controllers, SBC) [10], которым требовалось выделять соответствующий трафик из общего потока. Данные устройства производились в таких компаниях как Acme Packet, Audiocodes, Cisco, Genband. Для решения проблемы эффективного обмена данными между разными сегментами распределённой сети, соединёнными каналом ограниченной пропускной способности (данная проблема имеет название Channel optimization) был разработан целый спектр техник под общим названием Wan Optimizations [11]. Среди этих техник можно указать:

- Дедупликация (Deduplication)– уменьшение повторной передачи данных за счёт сохранения на обоих концах обмена повторяющихся элементов данных и последующей передачи ссылок на эти данные вместо самих данных. Может осуществляться на разных уровнях сетевого стека (в частности, TCP и IP)
- Сжатие (Compression) – передача данных по каналу в сжатом виде с последующим разжатием на другой стороне.
- Оптимизация латентности - упреждающая отправка сетевых пакетов-подтверждений TCP.
- Кэширование получаемого содержимого. Реализовывалось с помощью прокси-серверов, наиболее распространёнными из которых были Web-прокси, кэшировавшие содержимое сайтов. Примерами такого ПО являются Squid и NetCache.
- Объединение нескольких пакетов интенсивных сетевых протоколов, таких как CIFS [12], в один (protocol spoofing). Данные техники впоследствии

реализовывались как в виде отдельных сетевых устройств (Middleboxes [13]), так и программно, на мощных серверах (Network appliances). Одним из первых производителей стала компания Riverbed, впоследствии купившая анализатор Wireshark и интегрировавшая его в свои продукты. В сфере сетевой безопасности в этот период также произошли значительные изменения. Усложнение сетевых атак привело к тому, что их стало затруднительно с достаточной точностью определять по отдельным пакетам, а скорость появления новых атак — к необходимости реагирования на ещё неизвестные их виды. В совокупности это привело к появлению методов защиты на основе анализа поведения сетевых потоков (tcp session behaviour analysis). В то же время стали появляться вредоносные сайты, заражающие их посетителей, а также методы внедрения вредоносного функционала в не заражённые сайты. Для защиты от таких атак потребовалось внедрение обновляемых чёрных списков сайтов и необходимость фильтрации и блокировки по URL. Среди производителей средств защиты можно указать Arbor, BlueCoat, SonicWall.

Наиболее полное развитие технология анализа сетевого трафика получила, начиная со второй половины 2000х годов, в связи с несколькими факторами:

- Непрекращающийся рост объёмов передаваемых данных.
- Рост ширины каналов, обеспечивающих возможности для передачи этих объёмов.
- Увеличение количества разнообразия передаваемых данных, в частности тех, которые могут использоваться для составления различных профилей, как отдельных пользователей, так и различных групп.
- Рост как разнообразия сетевых угроз и атак, так и их количественные характеристик.

Эти факторы привели к росту потребностей со стороны провайдеров интернета (internet service providers, ISP) и различных компаний. Интересы этих групп различны, но, в тоже время, имеют значительные пересечения. Так,

например, общей областью интересов является защита сетевых ресурсов, которая, в свою очередь, делится на ряд направлений:

- Антивирусные решения (AV).
- Развитые межсетевые экраны Next Generation Firewalls (NGFW).
- Системы обнаружения и предотвращения сетевых атак Intrusion detection/prevention systems IDS/IPS.
- Системы защиты от DDoS-атак.

В то же время, специфичной областью интересов провайдеров интернета является [14, 15]:

- Обеспечение качество связи в часы наибольшей нагрузки (ЧНН) с учётом экономии на расширении арендуемых каналов связи.
- Получение конкурентного преимущества за счёт возможности предлагать более выгодные индивидуальные тарифы с учётом индивидуального профиля пользования сетевым каналом.
- Регулирование полосы пропускания для некоторых видов трафика. Одной из основных проблем является P2P трафик, который, может занимать значимую часть арендуемого провайдером канала (до 60- 80%[16]), приводя к тому, что чтобы обеспечить необходимое качество сервиса (quality of service, QoS) провайдеру приходится ускоренными (по сравнению с прогнозами роста абонентской базы и пользовательских потребностей) темпами расширять данный канал. Основной областью интересов компаний, предлагающих свои товары и услуги с использованием Интернета, являются «профили» пользователей с точки зрения их интересов и предпочтений. Подобные профили можно опосредованно выявить, в частности, с помощью списка сайтов, которые пользователь посещает, набора его поисковых запросов, сетевых приложений, которые он использует.

К другой группе относятся компании, предоставляющие различные интернет сервисы, например, с помощью технологии виртуализации сетевых

функций (Network Function Virtualization, NFV). К таким сервисам можно отнести:

- облачные сервисы,
- сервисы защиты,
- хранения и др.

Для этих компаний, специфичным является вопрос управления большими объёмами входящего трафика — требуется балансировка и интеллектуальное управление. В соответствии с приведённым выше историческим развитием потребностей в области сетевых сервисов происходило развитие технологий анализа сетевого трафика, ложащихся в основу аппаратных, программных и гибридных решений.

1.2 Направления развития технологий анализа сетевого трафика

Можно выделить два основных направлений развития.

- Рост «глубины» анализа для отдельного сетевого пакета, то есть увеличение уровня модели OSI, данные которого подвергаются анализу.
- Полнота учёта состояния потока, к которому относится пакет, а также других потоков, связанных с данным.

В следующих разделах будут рассмотрены оба этих направления развития.

1.2.1 Глубина анализа сетевых пакетов

По этой «оси» технологии анализа трафика развивались последовательно, каждая последующая наследовала часть предыдущих механизмов и добавляла свои. Можно выделить три уровня развития технологии, которые приведены на рисунке 1.

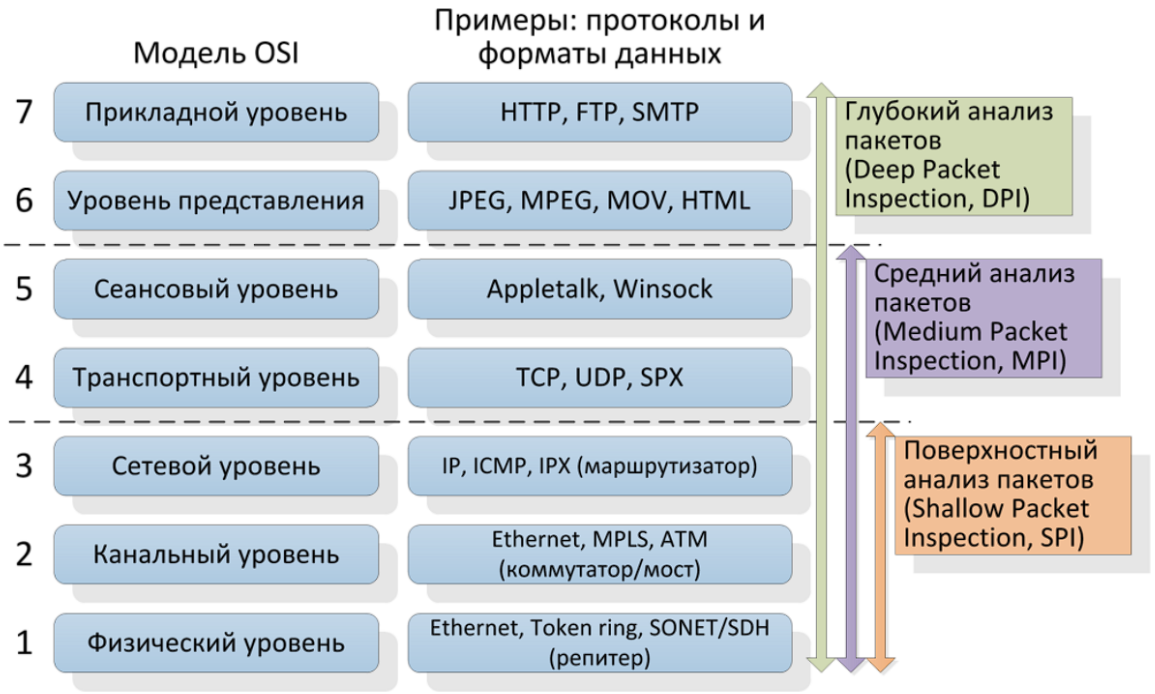

Рисунок 1 – Уровни развития технологии анализа сетевого трафика по «глубине»

Рассмотрим эти уровни более детально.

1.2.2 Поверхностный анализ пакетов (SPI)

Технология анализа трафика, основывающаяся исключительно на заголовках пакета уровней L1-L3 по модели OSI. Предъявляет низкие требования к вычислительным ресурсам, что позволяет анализировать большие объёмы трафика. Технология широко распространена, на её основе работает большинство межсетевых экранов операционных систем (в частности в ОС Windows XP/Vista и OS X), маршрутизаторов и других сетевых устройств. На её основе реализованы сетевые списки контроля доступа на уровне IP адресов и портов (Access Control List, ACL). Таким образом, данная технология хорошо подходит для разграничения доступа извне к отдельным компьютерам (IP) и сервисам (порты) внутренней сети.

1.2.3 Средний анализ пакетов (MPI)

Технология анализа трафика, основывающаяся на инспектировании сессий и сеансов связи, инициированных приложением, но устанавливаемых шлюзом-посредником (см. рис. 2). Также применяется термин «прокси приложений» (application proxy). В рамках данной технологии содержимое пакетов анализируется частично и по предопределенным правилам. Не используются сложные методы анализа типа сигнатурного. Устройства, реализующие данный функционал размещаются между провайдером интернета и конечным пользователем. Данные устройства разбирают заголовки вплоть до транспортного уровня и небольшую часть данных пакета для сопоставления разобранной части с некоторым списком разбора (parse list), с последующей реакцией в случае их обнаружения. Данные списки обычно короче списков ACL и предоставляют более широкий диапазон действий в отличие от «разрешить/запретить» в случае ACL. Эти списки также более выразительны, так как позволяют привязываться не к IP-адресам, а к формату данных пакетов и данным некоторых протоколов уровня приложения, например, URL-адресам в случае протокола HTTP. С помощью MPI можно, например, заблокировать возможность получения flash-файлов или картинок с определённых интернет сервисов (на уровне представления OSI) или заблокировать часть команд (на уровне приложения OSI) в отдельных протоколах. Набор протоколов, как правило, очень ограничен. Например, в первых версиях CheckPoint FireWall-1 (CheckPoint FW-1) поддерживались протоколы Telnet, FTP, HTTP, а в Cisco Private Internet Exchange (Cisco PI X) - FTP, HTTP, H.323, RSH, SMTP и SQLNET. Впоследствии данные наборы незначительно расширялись. Также известно, что данная технология используется в продуктах компаний McAfee и Symantec. Межсетевые экраны, использующие данную технологию, относятся ко второму поколению [17].

Данная технология более гибкая в сравнении с SPI и, помимо разграничения доступа, подходит для большего числа задач — кэширование содержимого, анализ сжатого/шифрованного трафика, ограничение функционала отдельных протоколов путём запрета отдельных команд. Благодаря подключению в режиме прокси, может служить в качестве Wan Optimizer'a (см. выше). Основной недостаток MPI — плохая масштабируемость: каждая команда и протокол требуют отдельного «шлюза» (входной-выходной порты). Кроме того, работа в режиме прокси сильно снижает скорость обработки. Для снижения нагрузки на прокси-сервер был разработан протокол ICAP [18], позволяющий прокси-серверам отправлять проходящие через них данные для проведения анализа сторонним серверам на предмет безопасности или анализа содержимого. Эта схема реализована в антивирусном продукте ClamAV, который может подключаться к прокси-серверам Squid и NetCache, упомянутым выше. Эти факторы сильно ограничивают применение данной технологии на уровне провайдеров интернета вследствие необходимости анализа большого числа протоколов и команд на широких каналах связи.

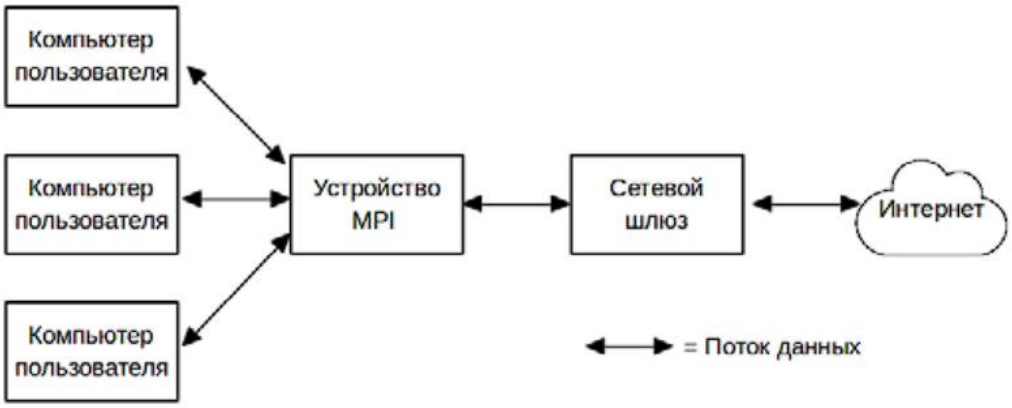

Рисунок 2 - Схема применения устройств анализа на основе технологии MPI.

1.2.4 Глубокий анализ пакетов (DPI)

Иногда употребляют более узкий термин — DPP (Deep Packet Processing), который подразумевает такие действия над пакетами, как модификация,

фильтрация или перенаправление. Сегодня оба термина часто используются как взаимозаменяемые [19]. Данная технология является логичным развитием MPI. В рамках данного подхода анализатор просматривает содержимое каждого пакета полностью. Одним из важных отличий от предыдущих технологий является то, что системы на базе DPI могут принимать решение не только по содержимому пакетов, но и по косвенным признакам, присущим каким-то определённым сетевым программам и протоколам. Для этого может использоваться статистический анализ. Например, анализ частоты встречи определённых символов, длин пакетов, расстояние между метками времени последовательных пакетов и т.д. Также, по сравнению с предыдущими подходами, значительно расширен список применений технологии: классификация, ограничение полосы, приоритезация, маркировка, кэширование и т. д. Технология DPI получила развитие, прежде всего, из-за стремительного роста вычислительных способностей процессоров, их быстродействия и, соответственно, возможностей для более полного и точного анализа сетевых данных. В отличие от MPI, данная технология изначально разрабатывалась для высокоскоростной обработки и идентификации большого числа приложений в реальном времени. Таким образом, решения на основе DPI хорошо масштабируются как по ширине сетевого канала (известны решения, работающие на каналах порядка 100 Гбит/сек), так и по числу идентифицируемых приложений (в существующих решениях — порядка нескольких тысяч). С точки зрения реализации, основной компонент любого решения DPI - модуль классификации, отвечающий за классификацию сетевых потоков. При этом в зависимости от целей применения DPI, классификация может выполняться с различной точностью: • тип протокола или приложения (например, Web, P2P, VoIP) • конкретный протокол уровня приложения (HTTP, BitTorrent, SIP) • приложение, использующее протокол (Google Chrome, µTorrent, Skype).

Важно отметить, что соответствие между классами различных уровней точности не однозначно, что показано на рисунке 3.

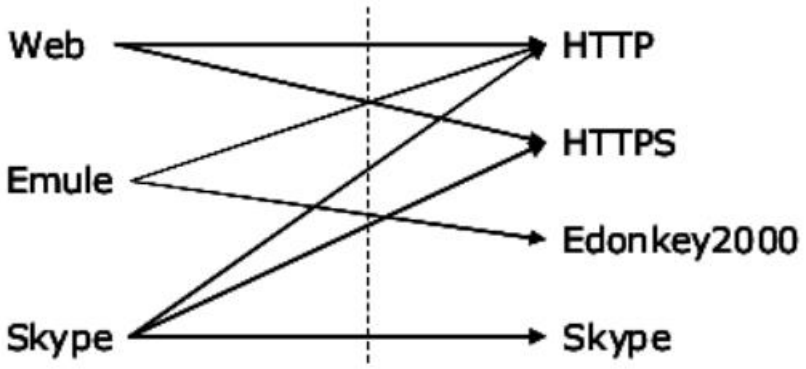

Рисунок 3 - Различие между идентификацией приложений (слева) и протоколов (справа)

Технология DPI на данный момент является текущим стандартом де-факто для средств анализа сетевого трафика и относится к области критически важных технологий необходимых для обеспечения, как сетевой безопасности, так и требований законодательства. Вследствие этого в последнее время на международном уровне был принят ряд стандартов, требований и рекомендаций по особенностям реализации, внутреннему устройству и набору функций соответствующих средств [20, 21]. Эта технология редко применяется в межсетевых экранах — это скорее область IDS/IPS систем, в качестве исключений можно указать экраны Hogwash и Shield. Однако межсетевые экраны, относящиеся к четвёртому поколению [17] могут учитывать данные IDS/IPS систем в процессе анализа.

1.3 Учёт состояния потока при анализе сетевого трафика.

Вторым направлением развития технологии анализа можно назвать учёт состояния протокола (потока) в процессе анализа — т.н. stateless/statefull виды анализа. Данное направление актуально только для протоколов, использующих транспортный протокол с установлением соединения (connection-oriented). Это означает, что перед любым обменом командами и данными происходит процесс «установления соединения», в ходе которого стороны обмениваются

фиксированной последовательностью пакетов, которая часто называется «рукопожатием» (handshake), а после завершения обмена происходит аналогичный процесс «закрытия соединения». К connection-oriented протоколам, в частности, относится протокол TCP, но не UDP. Однако следует учесть, что поверх UDP может быть реализован другой транспортный протокол, с установлением соединения. В качестве примера можно привести протокол Quick UDP Internet Connections (QUIC) [22] — протокол транспортного уровня с установлением соединения, использующий UDP. Из этого следует, что, в общем случае, нельзя полностью исключить statefull анализ для UDP пакетов.

Для описания различий описанных подходов требуется дать определение понятию «поток пакетов». Известны различные определения данного понятия. Часть из наиболее широко используемых приведена на сайте Center for Applied Internet Data Analysis (CAIDA)[23]. В данной работе мы будем использовать «односторонний поток транспортного уровня» — последовательность пакетов передающихся с заданного IP-адреса и TCP/UDP порта на данный IP-адрес и TCP/UDP порт, с указанием протокола транспортного уровня (TCP/UDP). Таким образом, поток задаётся пятёркой <srcIP, srcPort, dstIP, dstPort, protocol>. С учётом данного определения, можно сформулировать отличие statefull от stateless подхода. Оно состоит в том, что в случае statefull подхода учитывается тот факт, к какому именно потоку относится анализируемый пакет, и результат (состояние) анализа предыдущих пакетов этого же потока, если данный пакет не первый. В случае если пакет первый — проверяется, что он является корректным пакетом установления соединения. Следует также отметить, что понятие «statefull» не вполне чёткое и может иметь разные градации с различным «состоянием», что приводит к различному балансу точность анализа/ресурсоёмкость/скорость работы [24, 25]. Один из вариантов градации можно видеть на рис. 4. Список уровней учёта состояния потока, который там отражён – следующий:

• Анализ отдельных пакетов без учёта потоков и состояний (Packet Based No State, PBNS).

• Анализ пакетов в рамках потоков (Packet Based Per Flow State, PBFS).

• Анализ сообщений в рамках потока (Message Based Per Flow State, MBFS), т.е. произведена сборка IP-фрагментов в IP-пакеты (IP- нормализация) и сборка TCP-сегментов в TCP-сеансы (TCP- нормализация).

• Анализ сообщений в рамках протокола (Message Based Per Protocol State, MBPS), т.е. учитывается состояние автомата протокола (возможность принимать тот или иной тип сообщений).

Пример автомата состояний протокола HTTP приведён на рисунке 4. Вершины соответствуют состояниям, рёбра — условиям перехода, к которым могут относиться приём/отправка сообщения, результаты обработки сообщений, истечение таймаута.

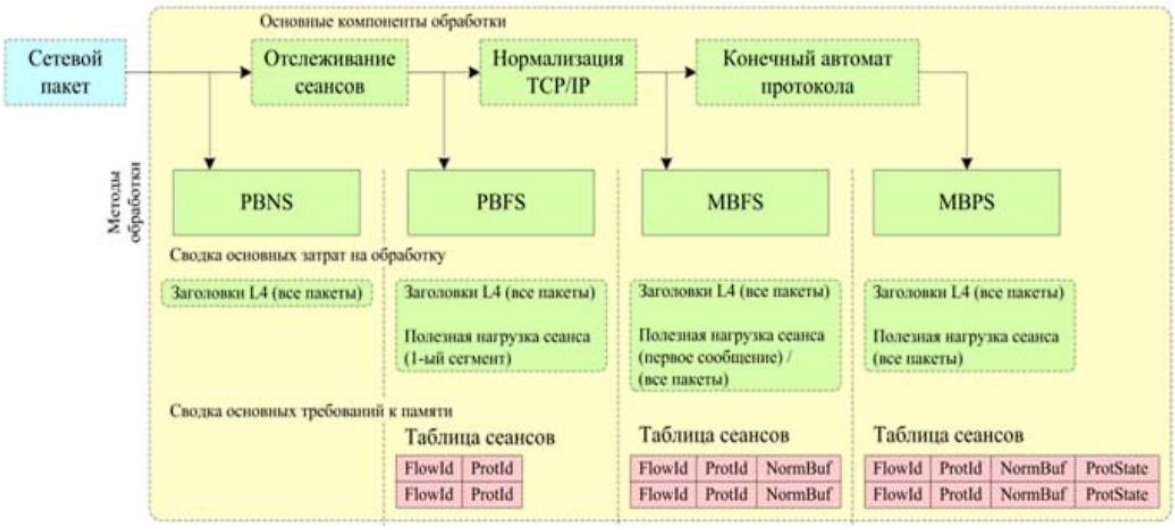

Рисунок 4 - Градации полноты учёта состояния потока.

Базовые реализации технологии DPI часто относятся к stateless-анализу, то есть анализ выполняется на уровне отдельных пакетов, состояние между анализом нескольких пакетов одного сетевого потока не сохраняется. Этого уровня точности хватает для многих практических приложений и позволяет значительно экономить ресурсы (см. рис. 4). В то же время, существуют задачи, для которых такого уровня точности не достаточно. В качестве примеров можно

привести две технологии, использующие statefull подход — инспекция пакетов с хранением состояния (statefull packet inspection, SPI) и глубокий анализ содержимого (deep content inspection, DCI).

1.3.1 Анализ сетевых пакетов с учётом состояния потоков

В рамках SPI подхода, программа или устройство, которое его реализует, в момент открытия нового соединения проверяет его на соответствие заданной политике безопасности и до закрытия хранит параметры этого соединения в памяти. С помощью таких решений, в частности, осуществляется проверка корректности соединения, например отсутствие пакетов на открытом сетевом порте после завершения соединения. Реализации SPI содержатся в большинстве современных маршрутизаторов в виде SPI-брандмауэров. Также эта технология используется в программных межсетевых экранах, учитывающих состояние (stateful firewalls), компании CheckPoint и ряде IDS/IPS систем. Межсетевые экраны, использующие эту технологию, относят к третьему поколению [17]. При данном подходе отслеживаются не только входящие и исходящие пакеты, но и состояние отдельных соединений, которое хранится в динамических таблицах. Благодаря этому при анализе очередного пакета могут учитываться не только заданные правила и политики по отношению к адресам и содержимому пакетов, но и состояние соединения, к которому относится пакет и предыдущих пакетов, которые к нему относятся, а также и других, связанных с данным, соединений. Классический пример преимущества межсетевого экрана поддерживающего состояние потока по сравнению с межсетевыми экранами без такой поддержки — обработка FTP протокола. Данный протокол открывает новый поток передачи данных на каждую соответствующую команду, причём поток открывается на случайном порте, большем 1024. Так как межсетевой экран не имеет возможности узнать, что новый поток относится к допустимому FTP протоколу — этот поток будет заблокирован. В случае наличия поддержки состояний потоков — адресная информация нового потока будет добавлена в таблицу легитимных потоков и сессия будет пропущена в сеть.

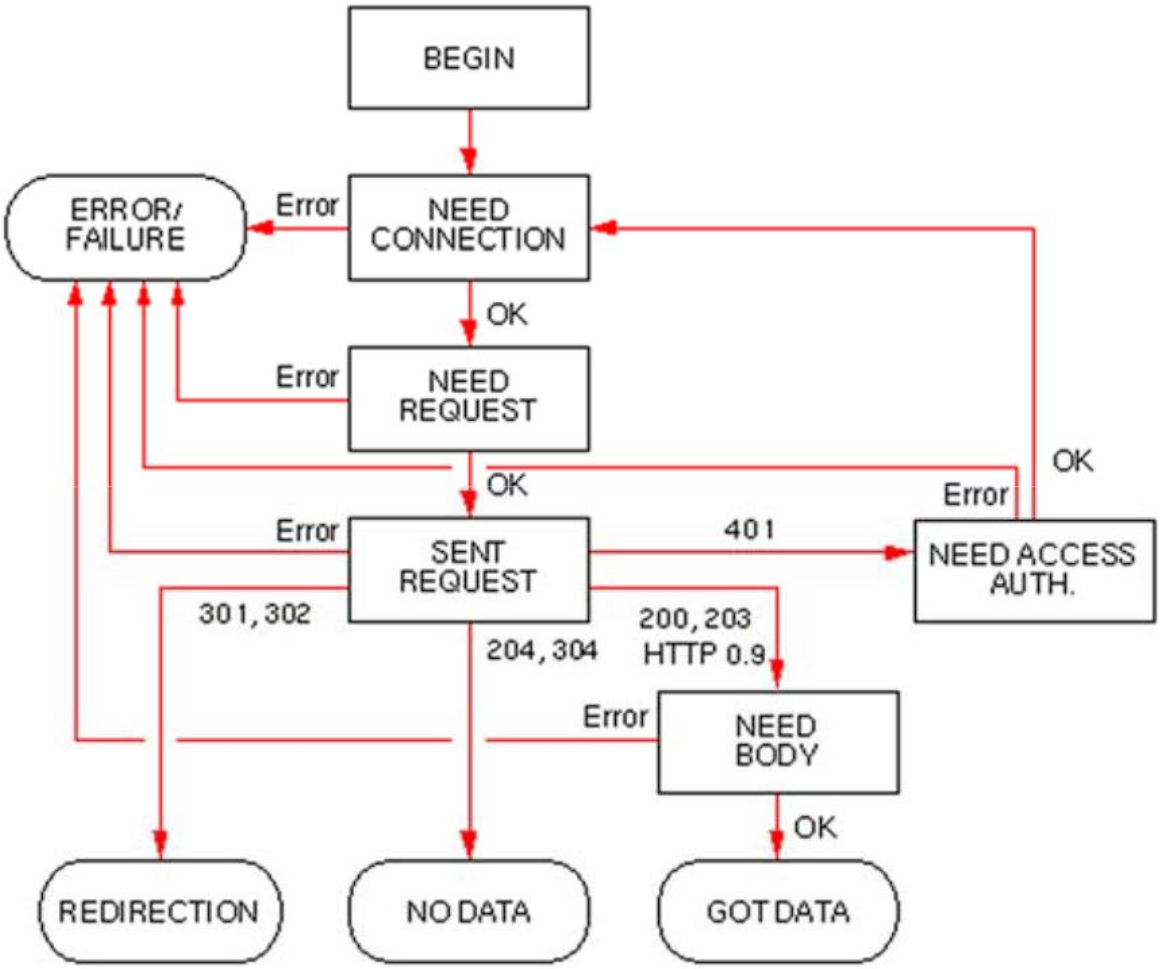

Рисунок 5 - Пример автомата состояний протокола HTTP.

1.3.2 Анализ содержимого сетевых протоколов прикладного уровня

В рамках технологии DCI выполняется не только идентификация протокола конкретного сетевого потока, но и группировка потоков в группы, отвечающие за предоставление некоторого сервиса, например сигнального протокола, например, SIP и протокола передачи данных, например, RTP, в случае VoIP. Также в процессе применения DCI, анализ не останавливается на идентификации протокола, например, HTTP, но также делается попытка определить приложение, которое его использует (например, Gmail) и собрать контент этого приложения в том виде, в каком оно было передано приложением для отправки по сети (электронное письмо). Примером использования данной технологии может служить функция прослушивания VoIP звонков по

перехваченному трафику в анализаторе WireShark [26]. С точки зрения функционала, основной вклад DCI в дополнение к модулю классификации (основной функционал DPI) — набор модулей разбора для различных протоколов прикладного уровня и различных видов данных в различных кодировках (например, MIME [27]), которые они содержат. Функции модулей разбора, сводятся к двум основным: 1. Разбор буфера данных (сетевого пакета или собранной сессии), в соответствии с форматом сообщений протокола, описанным, как правило, на одном из специальных языков типа ASN.1 [28] и P4 [29]. 2. Сборка сессий для протоколов с установлением соединения и их последующий разбор (пункт 1). Одной из тенденций последнего времени в развитии средств DPI/DCI является универсализация и централизация анализа. Данная концепция может быть обозначена как «DPI как сервис» - под этим названием она была приведена в работе [30]. Суть концепции заключается в том, что если в сети используется большое число различных средств, реализующих тот или иной анализ трафика (межсетевые экраны, системы IDS, оптимизаторы трафика и др.), то имеет смысл вынести весь анализ в отдельное устройство. Это устройство будет выполнять полный разбор сетевых данных и рассылать результаты анализа всем устройствам в зависимости от их потребностей, а те, в свою очередь, реализовывать только реакцию на поступающие данные. Переход к этой концепции в чём-то аналогичен переходу к программно-конфигурируемым сетям (Software Defined Networks, SDN) [31] в вопросах управления трафиком, при котором все решения по используемым алгоритмам маршрутизации и уровне её выполнения переходят от конкретных маршрутизаторов к выделенным устройствам - SDN-контроллерам. Такие подходы упрощают масштабирование систем и позволяют эффективно расширять функционал без дополнительных работ по интеграции и перенастройке оборудования. Концепция «DPI как сервис» может быть эффективно реализована в рамках систем унифицированного управления угрозами (Unified threat management, UTM) и унифицированного управления безопасностью (Unified security management, USM). Эти системы также являются

отражением тенденции централизации в виде объединения функционала межсетевых экранов, сетевых систем IDS/IPS, антивирусов, VPN-серверов, фильтров содержимого, балансировки нагрузки и предотвращения утечек данных в рамках единой системы. Демонстрацией этих тенденций является выделение функционала распознавания протоколов и извлечения метаданных в виде отдельных модулей. Причём эти модули могут быть, как чисто программными, так и привязываться к некоторой аппаратуре. Примерами программных реализаций являются Qosmos Intelligence Engine [32], ipoque PACE [33], Windriver Content Inspection Engine [34], Procera PacketLogic Content Intelligence [35]. Среди привязанных к аппаратуре модулей можно указать Cisco Network Based Application Recognition (NBAR) [36] и Junos OS Next-Generation Application Identification [37]. Использование этих модулей в виде составной части систем контроля и управления трафиком позволяет формулировать политики безопасности и другие виды политик в гораздо более высокоуровневых терминах, например в терминах URL, имён приложений, отдельных функциональностей в рамках этих приложений (например, блокирование передачи голоса в рамках Skype, при сохранении возможности обмена текстовыми сообщениями). По сути, набор функций данных модулей аналогичен расширению функционала технологии MPI на произвольное множество протоколов, их команд и данных, которое поддерживаются конкретным модулем распознавания протоколов. Типичная схема использование такого решения [14] приведена на рис. 6, где «Внешний интерфейс» — решение типа «DPI как сервис», PCRF - Policy and Charging Rules Function — устройство, хранящее политики и правила, применяемые к трафику, «Внутренний интерфейс» — устройство хранящее статистику, журналы, результаты применения правил к трафику, и т.д.

Рисунок 6 - Схема использования системы DPI для применения политик к сетевому трафику.

Концепция «DPI как сервис» может также рассматриваться как отделение инфраструктурной части анализа сетевого трафика от бизнес-логики в рамках отдельных прикладных задач (сбор статистики, межсетевой экран, IDS/IPS системы и др.). В следующем разделе будет рассмотрена схема работы именно инфраструктурной части анализа, так как она является, с небольшими вариациями, идентичной в различных решениях для анализа сетевого трафика. В частности, будут выделены отдельные этапы анализа с кратким описанием их особенностей, а в последующих разделах каждый этап будет рассмотрен более подробно.

1.4 Общая схема инфраструктурных алгоритмов анализа сетевого трафика.

Общая схема анализа сетевого трафика состоит из следующей последовательности шагов, каждый из которых приводит к повышению уровня представления объекта анализа. 1. Захват пакетов, проходящих через контролируемое сетевое соединение. Результатом данного шага является получение объекта анализа в виде сетевых пакетов. В зависимости от

необходимой точности и скорости последующего анализа, а также доступных вычислительных мощностей могут использоваться различные подходы.

• Слайсинг (slicing), при котором анализу подвергаются не всё содержимое пакетов, а только некоторый префикс (n первых байт). В ряде исследований (например, [38]) показано, что этот подход хорошо работает для последующей классификации трафика по протоколам. В частном случае, если перехватываемый размер равен суммарному размеру сетевых заголовков (L1-L3) является реализацией технологии SPI.

• Самплинг (sampling), при котором перехватываются не все пакеты, а только их часть, которая может выбираться по различным условиям, в зависимости от потребностей. В процессе развития технологии было предложено большое число стратегий отбора [39]. Например, для задач мониторинга типов трафика подходит вариант с выбором каждого n-го пакета (uniform sampling), где n может выбираться в зависимости от соотношения ширины канала и пропускной способности системы анализа. Задача получения информации о полном состоянии сети по результатам самплинга известна как inversion problem [40], в частности, при применении uniform sampling происходит недооценка среднего размера пакетов, так как чаще будут отбираться пакеты меньшего размера [41]. Для передачи перехваченных данных используется протокол PSAMP [42].

• Наконец, для задач, в которых требуется максимально точный анализ трафика, например для систем обеспечения сетевой безопасности, требуется перехватывать все данные всего поступающего трафика без потерь — для обозначения этого подхода используется термин lossless capture или deep packet capture (DPC). 2. Агрегирование пакетов в потоки по некоторым адресным признакам (flow generaion [43]), получение нового объекта для анализа — сетевого потока. Если при этом данные пакетов в дальнейшем анализе не учитываются, то такой вид анализа называется «анализ потоков» - flow based analysis (в отличие от packet-based анализа, при котором анализируются данные

пакетов). На рис. 7 показаны различия типичных схем packet и flow-based анализа. Flow-based анализ широко используется в силу значительно меньших требований к мощности вычислителя и пропускной способности, за счёт значительного снижения объёма данных для обработки. Такой вид анализа может выполняться как локально [43], так и удалённо от точки сбора данных [44]. Для передачи собранных данных от точки сбора до точки анализа используется большое число протоколов, часть из которых стандартизирована в виде IPFIX [45], а часть разработана отдельными производителями — Cisco NetFlow, Juniper Jflow. В рамках подхода записи, описывающие поток могут содержать разный набор данных. Наиболее общим набором таких данных является следующий:

- IP адреса источника и адресата,
- протокол транспортного уровня,
- в случае протоколов TCP/UDP — номера портов источника/адресата,
- набор счётчиков: количество переданных пакетов и байт, время создания и завершения потока.

Следует отметить, что хотя данный метод действительно значительно снижает требования к анализатору, тем не менее, он не является достаточно гибким, так как в отличие от слайсинга и самплинга не позволяет варьировать количество поступающих данных (оно зависит от входных данных). Более того в большинстве реальных задач количество потоков незначительно меньше количества пакетов (примерно на порядок) из-за большого числа очень коротких потоков, состоящих из нескольких пакетов — flash flows [46]. Для решения этой проблемы было предложено использовать самплинг для потоков [41]. Другой особенностью данного метода является то, что, вследствие ограниченности памяти, устройство, осуществляющее агрегацию пакетов, не может отслеживать один поток на протяжении произвольного промежутка времени. Для решения этой проблемы в конкретном решении обычно присутствует настройка, ограничивающая максимальную продолжительность потока (5 минут, в случае

Cisco NetFlow [40]). По истечении этого времени считается, что поток завершился, и информация о последующих пакетах агрегируется в рамках «нового» потока. Исследование точности flow- based подхода и влияния этого эффекта на точность анализа содержится в работе [47]. Также в этой публикации описан инструмент FLOW-REDUCE, осуществляющий «сборку» полной информации о потоке из фрагментов, на которые она была разбита из-за ограничений по времени.

3. Выполнение классификации по протоколу прикладного уровня или конкретному сетевому приложению. Результатом данной операции является получение нового объекта для анализа — сетевого потока конкретного протокола или приложения (в этом случае связанных потоков может быть несколько, например, в случае VoIP приложения это потоки SIP и RTP). После выполнения данной операции возможна следующая дополнительная обработка полученного объекта, конкретный вид которой зависит от решаемой прикладной задачи:

- разбор полей протокола (protocol parsing),
- сборка сессии протокола для протоколов с установлением соединения,
- извлечение данных приложения (content extraction) — страниц сайтов (HTML), файлов различных типов (исполняемые, изображения, текстовые документы, и т.д.), электронных писем, аудио-видео потоков и т. д.,
- разбор данных приложения (application content parsing).

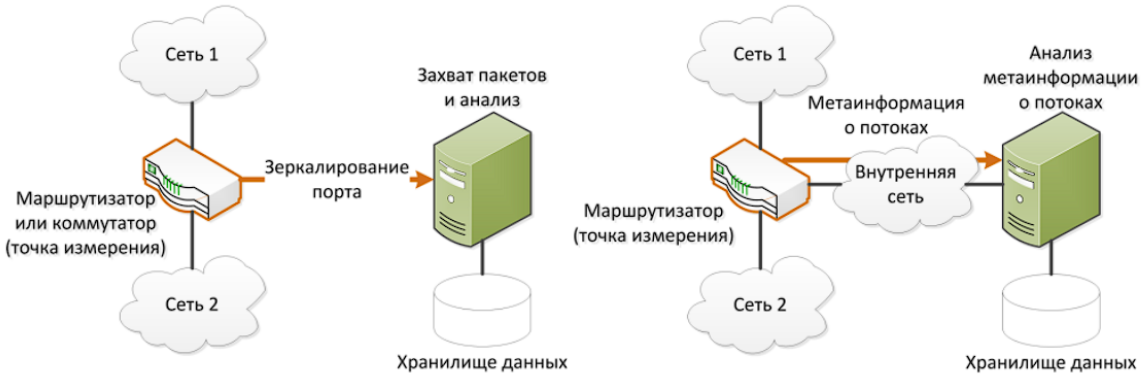

Рисунок 7 - Различия типичных схем packet (слева) и flow-based (справа) анализа.

Для полноты картины, следует сказать, что помимо указанных выше packet- based и flow-based подходов существует ещё один источник данных о сетевом трафике — т.н. база управляющей информации (Manage Information Base, MIB) [48] – виртуальная база данных, используемая для управления объектами в сети связи. Модули для накопления, хранения и обмена данными в формате MIB реализованы в большинстве устройств. Передача данных осуществляется по протоколу SNMP [49]. Данные получаемые таким путём имеют низкий объём и неспецифичны для протоколов. Например, в рамках данного подхода, можно получить сведения об общем количестве пакетов и байт прошедших через конкретный сетевой интерфейс конкретного сетевого устройства. Следует сказать, что одной из причин развития MIB и flow-based подходов, несмотря на их сравнительно низкую точность, послужила до сих пор идущая глобальная дискуссия [50] о законности и допустимости глубокого анализа трафика с точки зрения нарушения безопасности, прав на частную жизнь и т. д. На данный момент одним из следствий данной дискуссии является, в частности, то, что в научных работах, трафик, который подвергается глубокому анализу предварительно проходит процедуру «анонимизации» с помощью специальных средств [51]. Далее будут более подробно рассмотрены отдельные шаги из приведённой общей схемы анализа сетевого трафика, методы, алгоритмы и подходы, а также их особенности и ограничения применимости.

1.4.1 Захват сетевых пакетов

Программные и аппаратные средства, осуществляющие захват трафика относятся к классу сниферов (sniffers). Для решения задачи захвата трафика могут использоваться как стандартные серверные сетевые карты, так и специализированные сетевые карты, предназначенные для перехвата трафика на предельных скоростях без потерь. Специализированные карты, как правило,

реализованы на базе FPGA или ASIC и имеют встроенные средства для проставления временных меток, аппаратной фильтрации, снятия некоторых заголовков низкоуровневых протоколов, балансировки нагрузки между процессорами на многопроцессорных компьютерах с учётом IP-потоков, выявления ошибочных и дублирующихся пакетов. При этом вся обработка (в том числе и копирование данных в память компьютера из памяти сетевой карты) осуществляется без привлечения ресурсов ЦПУ. По мере развития технологий многие из описанных свойств реализуются и на базе стандартных сетевых карт. Технология реализации таких дополнительных функций носит название TCP Offload Engine (TOE). Она включает в себя следующие различные технологии, базовыми из которых являются следующие:

- Large Segment Offload (LSO) или Giant send offload (GSO)— сегментация больших TCP-пакетов при отправке
- Large Receive Offload (LRO) — сборка приходящих отдельных сетевых пакетов в большие сегменты
- Checksum Offload — проверка контрольных сумм в заголовках IPv4, IPv6, TCP и UDP
- IP Security (IPSec) Offload — шифрование/дешифрование трафика протокола IPSec.

Основной проблемой для стандартных сетевых адаптеров является не скорость передачи данных, как таковая, а количество пакетов в единицу времени. Это обусловлено особенностями внутренней реализации обработчиков пакетов на сетевых картах, драйверов сетевых карт и программных сетевых стеков ОС. Вследствие этого, стандартные сетевые карты без специализированных драйверов и сетевых стеков не обеспечивают перехват трафика без существенных потерь на скоростях более 3 Mpps (миллионов пакетов в секунду). Причины такого ограничения будут рассмотрены ниже. Ещё одной проблемой является точное проставление временных меток.

Проблемы, возникающие при переходе к сетевым соединениям, поддерживающим более высокие скорости передачи данных, связаны в основном с несколькими факторами:

- Ограниченной пропускной способностью аппаратуры.
- Архитектурными ограничениями при взаимодействии аппаратуры с ОС и ОС с пользовательскими приложениями.
- Объёмом памяти, необходимым для хранения получаемых данных.

Большинство распространённых систем анализа трафика работают, используя библиотеки Libpcap (ОС Linux) и WinPcap (ОС Windows). Данные библиотеки работают в пользовательском режиме. Для обеспечения своей работы со стороны ОС они используют драйверы уровня ядра Berkeley Packet Filter (BPF) и Netgroup Packet Filter (NPF) соответственно. Основная разница между этими драйверами заключается в схеме их работы с буферами памяти, использующимися для временного хранения пакетов, получаемых от сетевой карты. Драйвер BPF использует схему с двойной буферизацией, в то время как драйвер NPF использует кольцевой буфер [52]. Среди проблем этих решений, приводящих к снижению производительности можно выделить:

- Двойное копирование данных пакета (из карты в память ядра, из памяти ядра в память пользовательского процесса). • Большое число прерываний от сетевой карты (на каждый пакет, чтобы он был скопирован в буфер ядра).
- Большое число переключений между режимами ядра и пользователя (на каждый пакет при его копировании в память пользовательского процесса).
- Недостаточное использование параллелизма на уровне отдельных ядер и процессоров (по умолчанию все прерывания обрабатываются одним ядром).
- Проблемы с синхронизацией при доступе к данным из нескольких потоков выполнения. В случае, если полученные данные должны обрабатываться в несколько потоков между этими потоками возникает ситуация соревнования за ресурсы. В зависимости от количества копирований данных

пакетов, которые выполняются в процессе перехвата, решения разделяются следующим образом.

- 0-copy (zero-copy). Для реализации подхода с нулевым копированием требуется аппаратная поддержка со стороны сетевой карты – она должна содержать собственный DMA контроллер, копирующий данные с карты в память программы пользователя, без дополнительного копирования через память ядра. Примером может служить библиотека PF_RING ZC в связке с сетевыми картами Intel или Napatech [53]
- 1-copy. Для реализации этого подхода возможны несколько вариантов — разработка анализатора на уровне ядра, что является весьма сложной задачей или прямое отображение памяти ядра в память пользовательского процесса.
- 2-copy. Стандартное решение на базе LibPcap или WinPcap. Для решения перечисленных проблем было реализовано некоторое количество специализированных драйверов и сетевых стеков, к которым относятся, например, коммерческое решение Sniffer10G от Emulex и Myricom, а также открытая разработка PF_RING компании Ntop. Эти решения используют схему с кольцевым буфером, как более эффективную, а также оптимизированы для многопроцессорных и многоядерных компьютеров. В частности они реализуют следующий функционал:
- Обработка перехвата пакетов с использованием большого числа нитей исполнения (одна нить на входную очередь).
- Балансировка нагрузки между ядрами (одно ядро – одна входная очередь). • Пакетная фильтрация внутри сетевой карты. Для реализации этих функций используется как аппаратная поддержка со стороны архитектуры, так и поддержка со стороны ОС (специализированное API). Среди используемых технологий можно выделить следующие.
- Набор близких технологий Interrupt Moderation, Adaptive Interrupt Moderation, Interrupt Coalescing, Interrupt Blanking, Interrupt Throttling,

позволяющих управлять задержкой доставки прерываний за счёт настраиваемого таймера и обрабатывать получение/отправку множества пакетов за одно прерывание.

- MSI-X — распределение I/O прерываний по нескольким процессорам и ядрам.
- New API (NAPI) — интерфейс уровня ядра ОС Linux, позволяющий применять технику уменьшения количества прерываний (interrupt mitigation) со стороны сетевых устройств.
- Receive-side Scaling (RSS) — технология, предоставляющая возможность динамической балансировки нагрузки входящих сетевых пакетов по нескольким ядрам и процессорам (прерывания поступают на разные процессоры). Существуют реализации для масштабирования на случаи более 64 процессоров. Данная технология поддерживается в семействе ОС Windows с появлением Scalable Networking Pack. В ОС Linux аналог этой технологии называется Linux Scalable I/O. Также существует ряд аппаратных технологий от различных производителей процессоров, предназначенных для ускорения ввода/вывода.
- Intel Integrated I/O - технология прямого подключения шины PCI Express 3.0 к процессору (без отдельного PCI-контроллера), реализованная в семействе Intel Xeon E5.
- Direct Cache Access (DCA) – предоставление устройствам ввода/вывода, таким как сетевые адаптеры, возможности помещения данных напрямую в кеш процессора Intel.

1.4.2 Группировка сетевых пакетов в потоки

Группировка пакетов в потоки — достаточно стандартная и простая операция. Основное отличие разных реализаций данного функционала связано с тем, какие именно поля адресной информации и как использовать для идентификации потока. Наиболее употребляемое определение потока было дано

ранее. Так как оно использует 5-ку полей как ключевую информацию для определения принадлежности конкретного пакета к конкретному потоку, то для его обозначения и обычно используют термин 5-tuple. Также иногда используются двусторонние потоки, симметричные к перестановке пар <srcIP, srcPort> и <dstIP, dstPort>. Модуль, отвечающий за группировку пакета, обычно называют генератором потоков (flow generator). В процессе работы данный модуль хранит в памяти отображение соответствующей ключевой информации на данные конкретных потоков. При появлении нового пакета, с ним производятся следующие операции. 1. Из пакета извлекается ключевая информация, позволяющая идентифицировать, к какому потоку он принадлежит. 2. Производится поиск по текущему множеству потоков. 3. Если поток найден – в данных потока увеличиваются соответствующие счётчики – как правило, к ним относятся время жизни потока, количество пакетов и байт в потоке. Если поток не найден - создаётся новая запись потока и в неё добавляется информация о текущем пакете. В работе [38] проведена оценка вычислительных ресурсов, необходимых для выполнения первых двух операций, а также для операции классификации (в случае использования детерминированных конечных автоматов). Результаты оценки приведены в Табл. 1. Абсолютные цифры, приведённые на рисунке, на данный момент могут быть не вполне актуальны, но их ценность, прежде всего, в относительной стоимости операций.

Таблица 1 - Оценка скорости выполнения основных операций при анализе трафика

Операция	Стоимость (такты процессора)
Извлечение идентификатора потока	78
Поиск/добавление идентификатора потока	49

Поиск сигнатуры с помощью детерминированного конечного автомата (мин., ср., макс.)	13-4331-8900

Описанная выше базовая схема, хоть и является корректной, но неполной. Она содержит существенный недостаток — предполагается, что модуль располагает бесконечной памятью, так как отсутствует определение условий завершения потока и поэтому непонятно, когда следует удалять запись о потоке из отображения. В случае транспортного протокола с установкой соединения (например, TCP) в этом протоколе предусмотрена явная процедура завершения соединения (обмен FIN-ACK пакетами или посылка RST пакета). В случае протоколов без установления соединения (например, UDP) такой подход не работает, поэтому, как правило, используется один из вариантов, основанных на использовании таймера — например, обрыв соединения через 5 мин (такой вариант используется в коммутаторах Cisco NetFlow). Этот же подход используется для слишком долгих TCP-потоков [43].

1.4.3 Классификация сетевого трафика.

Тема классификации сетевого трафика сама по себе является очень обширной. Прежде чем переходить к методам, которыми она осуществляется, перечислим варианты классификации по её результатам, то есть объектам, которые получаются на выходе данного алгоритма, их свойствам и возможностям их дальнейшей обработки. По этому критерию, можно выделить три основных варианта классификации. Далее они перечислены в порядке увеличения «точности» классификации:

- Тип трафика не является достаточно содержательным способом классификации и, как правило, или не подвергается дальнейшему анализу, или подвергается достаточно простой дополнительной уточняющей классификации.

В зависимости от сферы применения, типы могут быть различными. Среди примеров, можно указать:

- P2P, видео-стриминг, веб-трафик — в случае систем сбора статистики и мониторинга, • трафик сетевой атаки/нормальный трафик — в случае систем защиты от сетевых атак,
- трафик, содержащий/не содержащий объекты копирайта, в случае систем контроля копирайта.
- Использующийся протокол прикладного уровня (protocol identification) является достаточно содержательным и может, как использоваться непосредственно — например, в системах сбора статистики и мониторинга для повышения уровня точности. Основным способом дальнейшей обработки является разбор протокола, включающий два основных функции — сборка сессии прикладного уровня, в случае необходимости извлечение данных протокола из отдельных его полей (метаинформация уровня протокола).
- Приложение, передающее данные (application identification), дает максимально детализированный уровень классификации. На этом уровне могут осуществляться те же виды обработки, что и на уровне протокола прикладного уровня, а также извлекаться и интерпретироваться данные (метаинформация) конкретного приложения, что соответствует более высокому уровню их представления. Например, поле типа «строка», определённое на уровне протокола, может соответствовать «имени пользователя» на уровне приложения.

В различных прикладных задачах результаты идентификации протоколов и приложений могут интерпретироваться и, соответственно подвергаться различной последующей обработке (как и в случае идентификации типа трафика). Например, в случае системы защиты от вредоносного кода, под протоколом может пониматься командный (command-and-control, C&C) протокол ботнета, а под приложением — конкретный вирус. Соответственно, извлекаемая метаинформация — команды ботнета, передаваемые им данные, а цель анализа — выяснение его функционала, оценка распространённости и

исследование возможностей его деактивации. В случае системы составления профиля пользователя для последующей демонстрации таргетированной рекламы (например, iMarker) в роли протокола может выступать HTTP, в роли приложения — браузер, а объектом анализа является запрос пользователя к поисковой системе, который подвергается дальнейшему текстовому анализу для извлечения ключевых слов. Выбор конкретной прикладной задачи может значительно влиять как на выбор алгоритма классификации, так и на его параметры и производительность. В качестве примера можно рассмотреть следующее сравнение. В случае системы статистики, алгоритм классификации обычно работает последовательно на пакетах каждого потока «до первого срабатывания». Схема такой классификации приведена на рисунке 8.

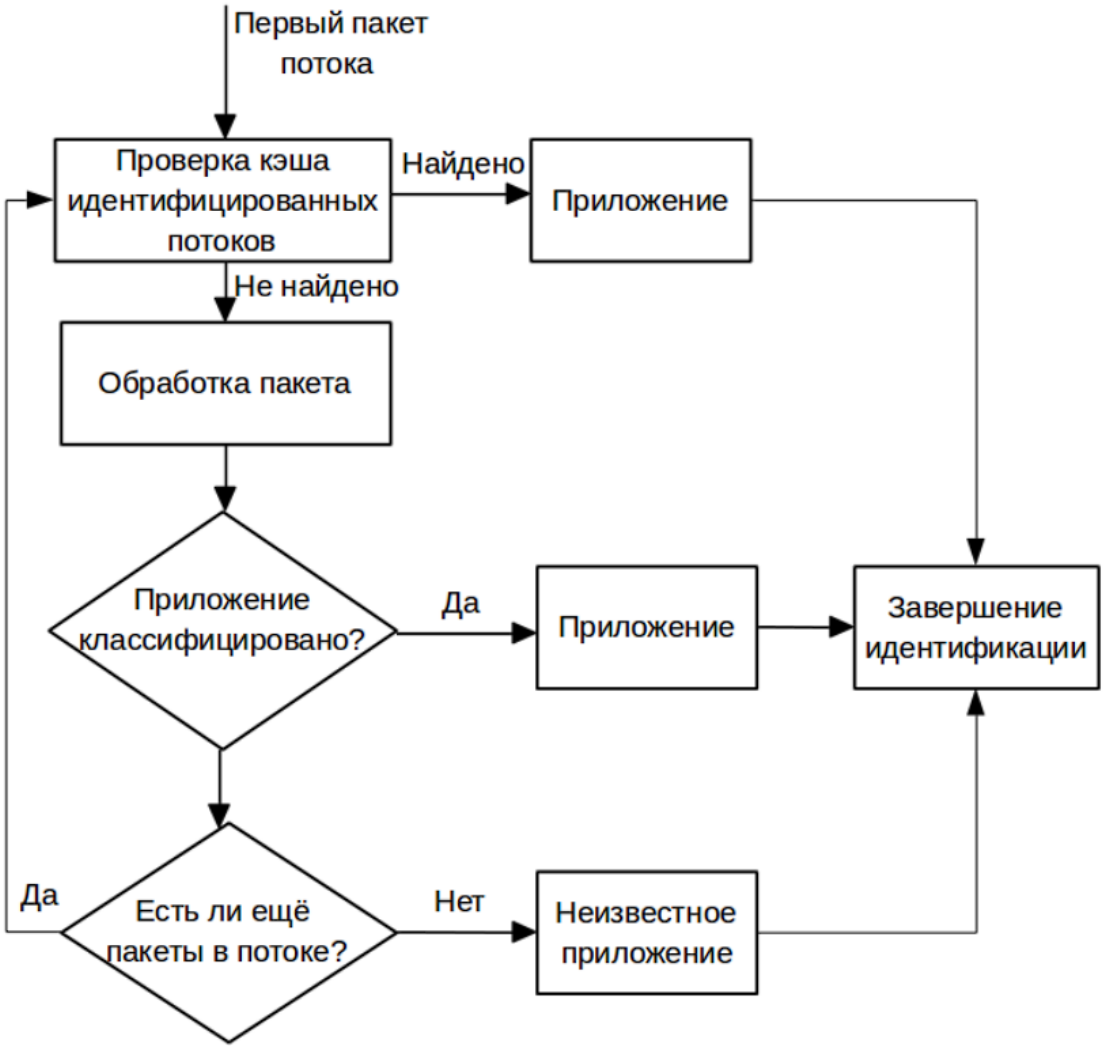

Рис. 8 - Схема классификации «до первого срабатывания».

В случае систем фильтрации по ключевым словам такой метод не подходит, так как в одном и том же сетевом потоке, в различных пакетах могут встретиться различные слова и, с точки зрения системы классификации, в этом случае данный поток попадёт сразу в несколько классов. В общем случае, очевидно, что первый подход гораздо производительнее, так как приходится анализировать значительно меньшие объёмы данных. Кроме того, в ряде подходов, для дополнительного ускорения, анализируют не всё содержимое пакета, а только некоторый его префикс (по аналогии со слайсингом). Например, в работе [54], для идентификации потоков, содержащих шифрованные и сжатые данные, используются только первые 16 байт пакетов. В работе [38] проведена оценка влияния размера анализируемого префикса пакета на точность классификации по протоколам и скорость работы классификатора на трёх снятых сетевых трассах Unibs-GT, Polito, Polito-GT. Результаты приведены на рис. 9, где на левом графике ошибки классификации обозначены как misclassified, а трафик, который не удалось классифицировать, как unknown.

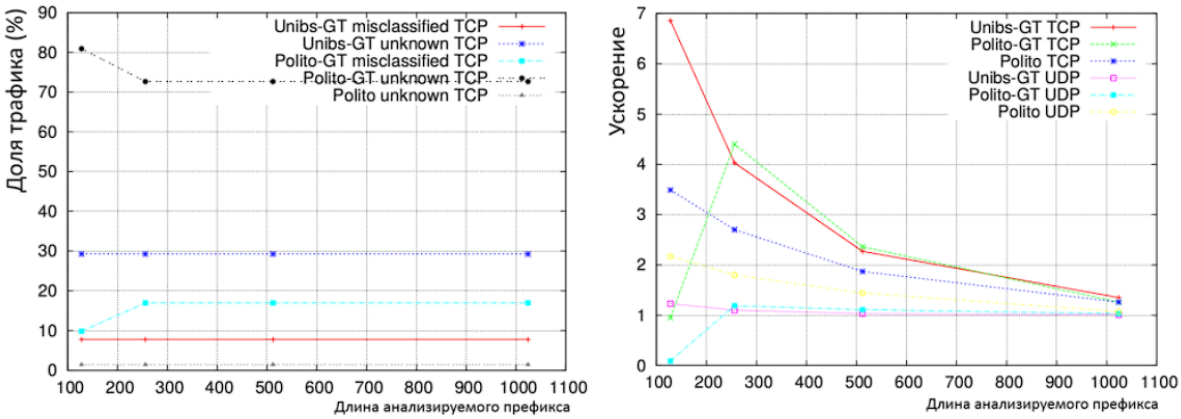

Рис. 9 - Оценка влияния длины префикса на точность классификации (слева) и скорость (справа).

Рискнок 9 - Оценка влияния длины префикса на точность классификации (слева) и скорость (справа).

На основе этих исследований, в частности делается вывод об избыточности проведения IP-дефрагментации и TCP-нормализации при решении данной задачи, так как данные алгоритмы (особенно второй) достаточно ресурсоёмки и практически не влияют на точность. Это происходит из-за того, что для

классификации, как правило, используется не более 256 байт пакетов, а минимальный размер фрагмента обычно не меньше 576 байт. То есть, для данной задачи PBFS подход более предпочтителен, чем подход MBFS (см. рис. 4). Рассмотрев виды классификации по получаемым результатам и подходы в разных прикладных задачах, перейдём к рассмотрению конкретных алгоритмов классификации. Классическим подходом к классификации является анализ содержимого пакетов (payload-based). При этом, как правило, выполняется поиск т.н. «сигнатур» (signature-based подходы) - характерных признаков, которые заранее создаются для каждого приложения или их групп. Классификация может выполняться как на уровне отдельных пакетов (stateless анализ), или может учитываться состояние потока (statefull анализ). Для повышения точности распознавания часть подходов использует уточнённые «сигнатуры» на основе автоматов состояний протоколов (см. рис. 5). При таком подходе, получаемые сообщения, после их классификации, сопоставляются с переходами в различных автоматах протоколов, и оценивается корректность последовательностей таких переходов. Эта группа подходов называется Stateful Protocol Analysis Detection [55]. Как было показано на рис. 9, классификация является наиболее нагруженным алгоритмом анализа сетевых пакетов. Исторически, из-за нехватки вычислительных мощностей, предпринимались попытки достижения увеличения производительности алгоритма за счёт выбора источника данных, используемых алгоритмом в процессе классификации, таким образом, чтобы обрабатываемые данные, будучи не менее информативными, чем содержимое пакетов, были бы более компактны. Эта группа подходов (в отличие от «сигнатурного») относится к классу «основанных на выводе» (inference-based).

Одним из важных преимуществ inference-based подходов является то, что качество анализа не зависит от представления данных в сетевых пакетах, в частности, отсутствуют ограничения при анализе сжатого/шифрованного трафика. Далее будут рассмотрены основные подходы к решению задачи классификации, их особенности и ограничения применимости.

1.4.4 Подходы на основе вывода.

Все подходы на основе вывода можно разделить на группы по двум основным параметрам:

- используемые для вывода данные,
- используемый для их анализа алгоритм. Все виды данных, в свою очередь, можно разделить на:
- характеристики отдельных пакетов в рамках отдельного потока (packet based),
- характеристики потоков в целом (flow based). К первой группе относятся подходы, использующие такие характеристики как: временные промежутки между пакетами, последовательности размеров пакетов [56], и др. Ко второй группе относятся два основных подхода.
- Подход на основе анализа портов (port-based) при котором идентификация происходит по одному из номеров портов потока, на основе базы данных о характерных статичных портах, которые используют зарегистрированные в IANA протоколы (регистрировать можно любой номер порта, а не только первые 1024). Этот метод считается малоэффективным, так как на данный момент существует большое число протоколов с динамическими номерами портов. В частности, к таким протоколам относятся практически все реализации P2P. Кроме того, часто используются схемы, при которых трафик некоторого протокола (например, HTTP) передаётся по нехарактерному для него номеру порта (не 80 в случае HTTP).
- Подходы на основе статистической информация об активности отдельных хостов в сети: в скольких и каких именно обменах данными (потоках) участвовал данный хост, сколько данных, и в какую сторону передавалось и т.д.

Эти данные сопоставлялись с набором заранее созданных шаблонов различных видов серверов. Один из таких подходов описан в работе [57]. Алгоритмы анализа данных делятся на два основных направления:

- сравнение с тем или иным видом заранее созданного шаблона,

- подход на основе машинного обучения и последующего распознавания.

Методы на основе машинного обучения в последнее время получили бурное развития. Одной из причин этого развития является доступность большого числа разнообразных данных для обучения (социальные сети, крупные БД, результаты поисковиков и т.д.). Эта группа методов на данный момент представлена большим числом алгоритмов: байесовские сети, деревья принятия решений, методы опорных векторов, методы k-средних и др. Данные методы, в свою очередь делятся на группы по методу обучения [58], который применяется для их конфигурирования:

- классификация (обучение с учителем),
- кластеризация (обучение без учителя),
- ассоциирование (association),
- численное предсказание (numeric prediction).

1.4.5 Методы на основе сигнатур

Недостатком этих методов является их высокая ресурсоёмкость, связанная с необходимостью просмотра больших объёмов данных. Однако в настоящее время вычислительные мощности позволяют использовать более точные, чем основанные на выводе, сигнатурные методы, которые, в свою очередь, делятся на две большие группы:

- поиск строк (string matching)
- поиск регулярных выражений (regexp matching).

Сигнатуры на основе строк. В процессе развития, для поиска строк применялось большое число различных алгоритмов поиска строк, обладающих различными преимуществами и недостатками, что определяло область их применения [59,60]. Наиболее известными алгоритмами являются: прямой перебор (brute force, BF), Кнут- Морис-Пратт (KMP), Бойер-Мур (BM), Ахо-Корасик (AC), AC-BM (использующийся в Snort), Wu-Manber, Commentz Walter

(CW), фильтры Блума (вероятностная структура на основе хеша). В работе [61] проводится обзор и сравнение большого числа методов поиска строк по тому как реализован алгоритм сравнения с имеющимися сигнатурами. Выделено 4 группы методов:

- Последовательное сравнение со всеми сигнатурами (Exhaustive Search).
- Дерево сравнений (Decision Tree).
- Декомпозиция (Decomposition), при которой отдельные части сигнатура обрабатываются независимо, с последующим объединением результатов.
- Ассоциативный доступ (Tuple Space), при котором сигнатуры разбиваются на группы бит, с которыми проводятся операции сравнения.

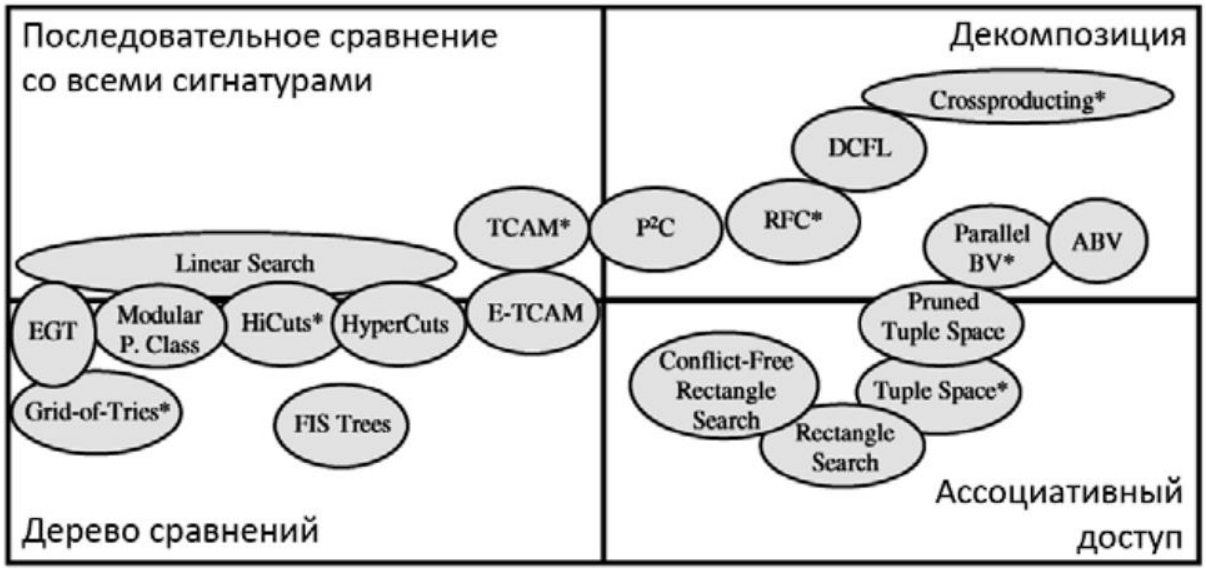

Рисунок 10 - Распределение алгоритмов поиска строковых сигнатур по данным группам.

1.4.6 Сигнатуры на основе регулярных выражений.

С ростом числа протоколов и их сложности строковое представление было признано недостаточно выразительным, в связи с чем, для описания сигнатур стали использовать регулярные языки в виде грамматик и регулярных

выражений. Для эффективного поиска сигнатур регулярный язык, описывающий сигнатуру, представляются в форме конечного автомата. Выделяют два основных вида автоматов - детерминированные или недетерминированные. Оба эти представления имеют свои достоинства и недостатки. Одна из открытых баз сигнатур такого вида используется в открытом приложении для классификации l7-filter[62]. Кроме того, такие подходы могут не срабатывать в случае, если сигнатура была разделена на несколько пакетов на уровне IP или TCP. Для решения этой проблемы, перед поиском сигнатуры необходимо выполнить IP-дефрагментацию и TCP-нормализацию соответственно. Основным достоинством недетерминированных конечных автоматов (НКА, NFA) является их компактность: объем занимаемой памяти пропорционален числу символов, входящих в регулярные выражения. Однако для обработки каждого символа входных данных недетерминированным конечным автоматам может потребоваться до O(N) обращений к памяти, где N – число состояний автомата [63]. По этой причине возможности применения НКА в высоконагруженных системах ограничены. В свою очередь, детерминированные конечные автоматы (ДКА, DFA) требуют для каждого входного символа совершить единственное обращение к памяти. Их использование может представлять трудности в связи с их большим размером: число состояний ДКА может экспоненциально расти («экспоненциальный взрыв»), и ограничено O(2 l), где l – суммарная длина регулярных выражений в каноническом представлении. В работе [38] было проведено исследование влияния разных типов регулярных выражений на рост размеров автомата. Результаты показаны на рис. 11. Было выделено 3 типа регулярных выражений, с точки зрения их влияния на размер автомата:

- выражения, привязанные к началу пакета (поиск осуществляется только в начале пакета);
- выражения, привязанные к началу пакета и содержащие звёздочку Клини (*);
- выражения, не привязанные к началу и содержащие звёздочку Клини (*).

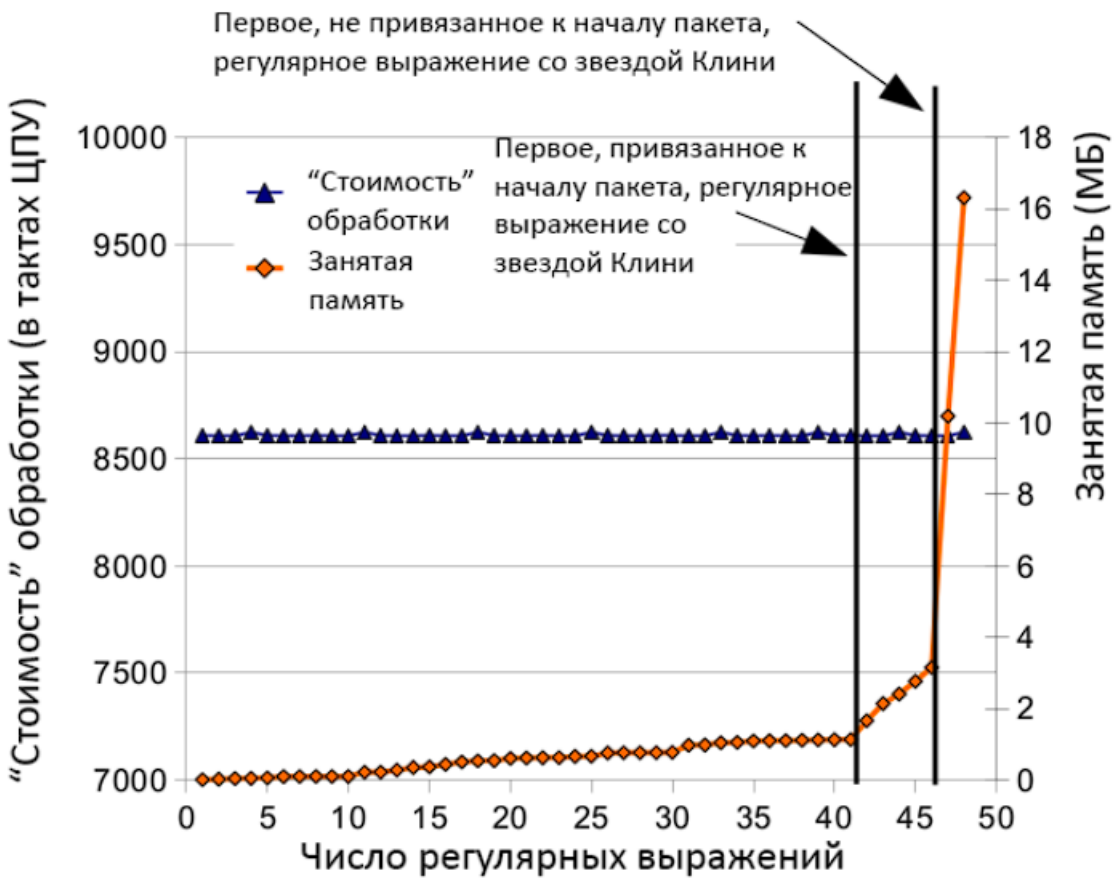

Рисунок 11 - Экспоненциальный взрыв размера DFA при добавлении регулярных выражений со звёздочкой Клини.

Для снижения размеров автоматов часто применяют различные виды сжатия. Такие автоматы носят название сжатые ДКА (Compressed DFA, cDFA). В табл. 2 приведено сравнение трёх основных видов автоматов по размеру и производительности поиска, взятое из работы [38]. Данные автоматы были построены по регулярным выражениям классификатора L7[63]. Для предотвращения экспоненциального роста размера, детерминированные автоматы были разделены на 4 части.

Таблица 2 - Сравнение размеров и скорости работы основных видов конечных автоматов.

Алгоритм	Стоимость в тактах ЦПУ (мин, ср., макс.)	Количество автоматов	Размер автомата
НКА	2.2*10**4, 4.1*10**7, 8.9*10**7	1	509 Кб
ДКА	52, 2.5*10**4, 3.6*10**4	4	230 Мб
Сжатый ДКА	268, 1.2*10**5, 1.7*10**5	4	53 Мб

Несмотря на проблемы с требованиями к памяти, детерминированные конечные автоматы (и их модификации) получили намного большее распространение в высокоскоростных системах анализа. Современные системы анализа трафика предъявляют высокие требования, как к скорости обработки данных, так и к количеству регулярных выражений, задействованных в обработке и, соответственно, размеру итогового автомата. Так как ни ДКА, ни НКА не могут удовлетворить одновременно требования и по скорости, и по размеру памяти, в настоящее время ведется большое количество исследований по разработке гибридных представлений. С точки зрения реализации, автоматы представляют собой таблицы из состояний, в каждой ячейке которых находится список возможных переходов из этого состояния в другое. Поэтому два основных направления работ сосредоточены на уменьшении числа состояний и переходов соответственно. В качестве примеров, можно привести представления D 2FA [64] и δFA [65], реализующие сжатие переходов и группу представлений MDFA [66], H-FA [67], XFA [68] и Dual FA [69], реализующих различные подходы к сокращению числа состояний.

1.4.7 Анализ данных в разных представлениях

Одну из важных проблем для классификаторов на основе содержимого представляет тот факт, что одни и те же данные (например, строка) могут быть при передаче по сети быть закодированы по-разному, в зависимости, например, от используемого протокола. В частности, под «различными представлениями» в данном разделе имеются следующие аспекты:

- Различные методы кодировки, в частности для текстовых данных — ASCII и Unicode кодировки, а для бинарных данных — различные транспортные кодировки, например представление в виде текста (binary-to-text), примером которых является Base64.
- Сжатия данных для уменьшения загруженности каналов передачи данных, например использования gzip и deflate алгоритмов для сжатия содержимого HTTP-сообщении.
- Шифрование данных для обеспечения безопасности, например использование криптографических алгоритмов RC4 и AES в протоколах SSL/TLS.

По данным различных исследований, сжатый и шифрованный трафик (иногда используется общий термин «непрозрачный», opaque) составляет всё большую долю от всех сетевых потоков данных [54]. Это является следствием большого числа факторов, таких как:

- рост популярности онлайн видеосервисов, использующие сжатие видеопотоков,
- распространённость P2P-сервисов, которые в большинстве своём используют шифрование,
- использование шифрованного соединения (HTTPS) по-умолчанию на многих популярных сайтах,
- внедрение сжатия в HTTP протоколе на многих Web-серверах.
- Проблема классификации этих видов трафика имеет несколько аспектов:

- Для корректной классификации такого трафика требуется дополнительный функционал.
- Попытка классифицировать такой трафик «в лоб» существенно снижает общую производительность классификатора, так как приходится просматривать все данные пакетов, проходя по большой части автомата и при этом результат почти наверняка будет отрицательным. То есть такой трафик представляет собой «худший случай», характеристики работы на котором алгоритмов классификации существенно хуже средних (см. табл.2).

Для решения первого аспекта проблемы используются несколько подходов:

- Генерация копий сигнатур, которые подвергаются различным видам сжатия и кодирования. Данный метод ограничен только некоторыми алгоритмами сжатия и кодирования, а также плохо масштабируется с учётом роста количества алгоритмов сжатия и их количества их параметров.
- Использование модулей, осуществляющих разжатие/перекодировку данных перед их классификацией. Этот метод имеет такие же ограничения, как и предыдущий и также плохо масштабируется. Кроме того этот метод увеличивает уязвимость системы к атакам типа zip bomb [70], при которых размер разжимаемых данных превосходит размер сжатых на несколько порядков.
- Установка системы анализа на месте или после средства, осуществляющего разжатие/расшифрование данных. Пример такого средства - прокси-сервер. Для устранения второго аспекта, требуется подавать на модуль классификации трафика только «прозрачный» трафик, для чего из всего трафика требуется предварительно отфильтровать «непрозрачную» его часть. Для решения этой задачи разработаны алгоритмы, большая часть которых использует характерное свойство «непрозрачного» трафика — повышенную энтропию 33 значений его отдельных байт. Примеры таких алгоритмов приведены в работах [71].

1.4.8 Классификация угроз

В реализациях DPI, связанных с безопасностью (например, IDS/IPS), где классификация применяется не для идентификации протоколов, а для классификации атак и угроз, разработаны подходы, специализированные под соответствующие задачи. Одним из таких подходов является статистическое выявление аномалий, когда вначале производится обучение системы на трафике, не содержащем атак («нормальном»), а затем на реальном трафике детектируются отклонения от «нормальной» картины. Такие подходы называют «статистическое детектирование аномалий» (statistical anomaly-based detection). На основе этой техники работают многие IDS и защиты от DDoS атак.

1.5 Требования, предъявляемые к современным средствам анализа содержимого сетевого трафика

Принимая во внимание приведённый обзор основных алгоритмов и схем анализа сетевого трафика можно сформулировать ряд функциональных и нефункциональных требований, предъявляемых к современным системам анализа сетевого трафика. Все требования можно разделить по подсистемам, к которым они применяются и отдельно выделить те, которые относятся ко всей системе в целом: 1. Система в целом.

- Поддержка масштабирования по пропускной способности анализируемого канала передачи данных.
- Минимизация числа перестановок пакетов в рамках отдельных потоков. • Возможность встраивания дополнительных средств предобработки сетевых пакетов перед их передачей подсистеме классификации (перекодировка, разжатие). 2. Подсистема перехвата данных.
- Поддержка разбора всех протоколов ниже сетевого уровня, встречающихся в контролируемом канале (MPLS, VLAN и т.д.) Это необходимо, для обеспечения попадания всех пакетов одного потока в одну очередь

обработки при выполнении балансировки нагрузки (хеширование должно выполняться на уровне IP-пакета).

- Использование кольцевого буфера для хранения обрабатываемых пакетов и режима zero-copy, при наличии поддержки со стороны сетевой карты, или 1-copy, при отсутствии такой поддержки для экономии ресурсов центрального процессора.
- Для эффективного использования ресурсов многопроцессорных и многоядерных машин требуется поддержка того или иного вида RSS-технологии (управления прерываниями и их распределения по разным ядрам). 3. Подсистема агрегации пакетов в потоки.
- Поддержка возможности задания типа ключевой информации, по которой определяется принадлежность пакета к потоку, для обеспечения гибкости при использовании подсистемы для решения различных прикладных задач.
- Максимизация количества одновременно обрабатываемых потоков и времени жизни каждого отдельного потока в условиях ограниченных ресурсов памяти.
- Для обработки сжатых данных необходима возможность одновременного отслеживания потока, который представлен как в сжатом, так и в разжатом виде.
- Встроенная защита от атак типа «zip-бомба».
- Отслеживание факта связанности потоков (например, потока управления и потока данных в случае FTP), в частности, для уточнения классификации. 4. Подсистема классификации.
- Сложность алгоритма поиска сигнатур должна быть не хуже чем линейной по входным данным «в среднем», а желательно и «в худшем» случае для устойчивости системы к целенаправленным атакам.
- Расширяемый набор «сигнатур» для поддержки новых протоколов, их групп и сетевых приложений.

- Хорошая масштабируемость по памяти при росте количества «сигнатур».
- Возможность предварительного разделения трафика на «прозрачный» и «непрозрачный» с целью снижения нагрузки на данную подсистему.
- Возможность анализа данных, представленных в различных кодировках.

1.6 Эволюция методов классификации трафика

Методы классификации сетевого трафика с годами развиваются и модифицируются. Это связано в первую очередь с предъявляемыми сетью требованиями и ограничениями. Изменение устройства сетевого трафика и особенностей его передачи приводит к тому, что старые методы классификации становятся малоэффективными или просто непригодными. С другой стороны, развитие методов классификации и оборудования, на котором может работать система позволяет использовать больше признаков и более развитые способы их применения для принятия решения.

К важным характеристикам методов классификации сетевого трафика относятся:

- детализация: с каким уровнем точности система производит классификацию: семейство протоколов/класс приложений или конкретные протоколы, конкретные приложения.
- скорость реакции: способна ли система производить классификацию быстро (после нескольких пакетов), что подходит для анализа в реальном времени или для классификации нужны данные о потоке полностью.
- вычислительная стоимость: сложность вычислений и затраты по использованию памяти для классификации пакета или потока.

1.6.1 Классификация по номерам портов

Первые системы классификации трафика основывались на извлечении из пакетов номеров портов и сопоставлении их со списком IANA (Internet Assigned Numbers Authority, «Администрация адресного пространства Интернет»). IANA выделяет и регистрирует номера портов, используемые для конкретных специфических целей, например, под протокол HTTP выделен порт 80. Информацию о протоколе можно уже использовать для примерного определения типа деятельности пользователя. Этот метод классификации работает очень быстро и не требует хранения данных о потоке, вычислительно прост. Это позволяет, например, удобно использовать его в межсетевых экранах для фильтрации трафика. Однако, он обладает рядом существенных недостатков, которые по мере эволюции устройства Сети негативно влияет на результаты его работы.

Номер порта определён не для всех протоколов. В списке IANA уже содержатся категории «известно несколько применений наряду с зарегистрированным» и «порт не зарегистрирован IANA». Некоторые протоколы выбирают порты для обмена данными в ходе своей работы случайным образом (как FTP). Вдобавок, некоторые протоколы могут использовать известные номера портов других протоколов, чтобы замаскироваться под них, если другой протокол является более предпочтительным с точки зрения интернет-провайдера. Например, протокол BitTorrent может таким образом маскироваться под HTTP, чтобы избегать блокировок или ограничений на скорость передачи данных. Появляющиеся в последнее время протоколы также могут не успевать получить зарезервированный за собой порт. Этот метод хорошо подходит для определения протоколов, однако не способен хорошо различать приложения. Например, браузинг веб-страниц, VoIP и просмотр видео – все будут использовать 80 (HTTP) или 443 (HTTPS) порт для своей работы. Но у этих приложений абсолютно разные сценарии использования, поэтому на практике нам хотелось бы их различать.

Широкое распространение технологий туннелирования, инкапсулирующих протоколы, шифрование на уровне IP, использование NAT (Network Address Translation, преобразование сетевых адресов) и NAPT (Network Address and Port Translation, трансляция сетевых адресов и портов) – всё это влияет на применимость данного метода. Поэтому точность систем, основанных на определении номеров портов, невысока (по разным оценкам, от 30 до 70%) и продолжает ухудшаться. В настоящее время этот признак может служить лишь одним из многих, выступая как источник дополнительной информации при принятии решения, основанного на других критериях.

1.6.2 Глубокий анализ пакетов

Следующим шагом развития классификаторов интернет трафика стало использование технологии DPI (Deep Packet Inspection, глубокий анализ пакетов). Фильтрация сетевых пакетов в этом случае проводится по их полному содержимому, то есть проводится анализ не только заголовков, но и всего трафика на уровнях модели OSI со второго и выше. Этот метод показывает высокую точность работы, а полученная с его помощью разметка зачастую принимается как эталонная для данных с неизвестными классами. Для классификации с помощью DPI создаётся библиотека сигнатур и шаблонов пакетов, и для каждого пакета производится поиск соответствий в этой библиотеке.

Было замечено, что некоторые проприетарные протоколы передают информацию на уровне битов, что привело к созданию инструментов, работающих и на этом уровне.

При всех своих достоинствах метод DPI сталкивается с существенными проблемами в своей работе. Среди главных – невозможность работы с зашифрованным трафиком, доля которого в Интернете растёт с каждым годом, и высокие требования к ресурсам. Для хранения данных пакетов и библиотеки сигнатур требуется достаточно большой объём памяти, а при росте количества известных классов растёт размер этой библиотеки и, соответственно, время на

поиск соответствий в ней. Поэтому, этот метод плохо подходит для работы в высокоскоростных сетях в режиме реального времени. Кроме того, определённую сложность представляет создание и поддержание в актуальном состоянии библиотеки сигнатур при всё увеличивающемся количестве протоколов и приложений в Сети.

Отдельно стоит вопрос защиты приватности пользователей Сети – проблема, которая актуальна для всех систем, использующих в своей работе полезную нагрузку пакетов. Законодательную сторону этого аспекта нужно учитывать при создании, обучении и работе систем глубокого анализа пакетов. Некоторые методы пытаются ограничить количество используемых данных пакета, например, первыми 40 битами [4], но полностью проблему это не решает.

1.6.3 Стохастический анализ пакетов

Стохастический анализ пакетов (SPI, Stochastic packet inspection) для классификации пакетов изучает статистические свойств их содержимого. Например, в [5] используется критерий Хи- квадрат Пирсона для изучения случайности распределения первых байтов полезной нагрузки пакета. Таким образом строится модель синтаксиса протокола, используемого приложением. В [6] потоки определяются как зашифрованные или незашифрованные на основании энтропии первого пакета. В [7] вычисление энтропии первых байтов полезной нагрузки идентифицирует тип содержимого как текст, бинарный файл или шифрованный файл, что позволяет приоритизировать передачу некоторых файлов. Однако, такую классификацию сложно назвать точной или детализированной, так как для одного и того же приложения возможно использование всех видов содержимого. Кроме того, хотя стохастический анализ и использует более простые операции, чем глубокий анализ пакетов, он всё равно использует большой объём памяти для анализа. В связи с этим, данный метод не получил широкого распространения.

1.6.4 Использование машинного обучения для классификации трафика

Тенденции изменения сетевого трафика, широкое распространение шифрования, рост скорости передачи данных, а соответственно и необходимой скорости их обработки, постоянное появление новых классов трафика - всё это потребовало появления новых способов его классификации. Для этого были предложены методы машинного обучения, которые позволяют во многом упростить работу с созданием наборов различающих характеристик классов, автоматизируя этот процесс на основе анализа большого количества примеров этих классов (собрать который значительно проще, чем проанализировать вручную). Кроме того, многие из предложенных методов работают с общими признаками потоков, а не с полезной нагрузкой пакетов, что решает проблемы, связанные с шифрованием и с защитой данных пользователей. Это же даёт преимущество в скорости классификации и уменьшает необходимый для принятия решения объём памяти.

Далее будут подробно рассмотрены именно методы машинного обучения для классификации сетевого трафика: типы классификации, используемые модели и признаки, а также наборы данных, на которых производится обучение и тестирование моделей.

1.7 Типы классификации

Трафик в сети можно классифицировать отдельными пакетами, и методы классификации на основе портов и DPI способны решать эту задачу, однако сейчас в большинстве работ классификация производится для потоков. Здесь и далее, поток – это пятёрка значений:

<IP-адрес источника, IP-адрес получателя, порт источника, порт получателя, тип транспортного протокола>.

Существуют подходы, в которых запросы отправителя и ответы получателя интерпретируются как два разных встречных потока, однако более

частым решением является объединение этих потоков в единый двунаправленный. Поскольку интернет-поток как правило подразумевает один законченный сеанс взаимодействия между отправителем и получателем (клиентом и сервером), проблем с классификацией всего потока как единого целого в один класс обычно не возникает.

Так как потоки различаются по своей продолжительности и количеству передаваемых данных, среди них иногда особо выделяют самые маленькие («мышиные», mice) потоки и самые большие («слоновьи», elephant). Из-за существенного отличия в объёме этих потоков результаты классификации иногда проверяются отдельно по доле правильно классифицированных потоков и по доле классифицированных байтов. Большое значение имеет также длина потока. Потоки или обрывки потоков, состоящие из малого количества пакетов, могут не нести достаточного количества информации для определения класса, поэтому нуждаются в специальном подходе или игнорируются.

Классификация трафика может проводиться в онлайн режиме, то есть в режиме реального времени, или офлайн, постфактум. Режим классификации определяется решаемой задачей. Так, например, сбор статистики использования сети для глобального перераспределения ресурсов, получение информации об активности пользователя для выставления ему счетов за интернет-услуги и перерасчёт этих цен - всё это не требует срочного ответа и может обрабатываться в свободном режиме, с любым количеством доступных данных и ресурсов. Другие же задачи, такие как обеспечение качества сервиса для пользователя, выявление атак и угроз, оперативное перераспределение ресурсов, требуют как можно более быстрой реакции. В этих случаях классификатор не имеет возможности ждать окончания потока и оперировать полной информацией о нём, а вынужден ограничиваться лишь частью информации, например, лишь первыми N пакетами потока. Также накладываются ограничения на используемую модель классификатора, так как она должна достаточно экономно расходовать ресурсы системы и максимально оперативно принимать решение о потоке. Кроме того, в этих случаях решение обычно принимается параллельно

для нескольких/многих потоков данных сразу, поэтому ограничения накладываются и на количество оперативной памяти, выделяемой для обработки потока.

Выбор набора классов для проведения классификации трафика зависит от решаемой задачи. В качестве примеров можно привести следующее:

1) Классификация по протоколам прикладного уровня (HTTP, SMTP, SSH и т.д.) [8, 9]. Обычно выбираются именно протоколы прикладного уровня, так как такая классификация является наиболее практически ценной.

2) Классификация по приложениям, генерирующим интернет-трафик (Skype, Torrent, браузер и т.д.) [10, 11]. Это определяет активность пользователя и позволяет строить его профиль, ограничивать деятельность конкретных приложений, решать маркетинговые задачи.

3) Классификацию по типам действий пользователя (интернет-браузинг, скачивание файлов, просмотр видео и т.п.) [12, 13]. В данном случае определяется не конкретное используемое приложение, а вид деятельности пользователя. В некотором смысле это обобщение предыдущего типа классификации.

Существуют и другие [14, 15] подходы к определению набора классов для классификации интернет-трафика, но приведённые три являются наиболее популярным, и именно они чаще всего исследуются и описываются в научных статьях.

Отдельно здесь следует упомянуть классификацию мобильного трафика [16] и классификацию трафика устройств интернета вещей [14], которые обладают особенностями, выделяющими их в отдельные задачи. Это и большая популярность новых и/или специализированных протоколов, и другие сценарии работы приложений, и другие масштабы распространения.

2. СПЕЦИАЛЬНЫЙ РАЗДЕЛ

2.1 Методы машинного обучения, используемые для классификации траффика

Задача классификации сетевого трафика, как и другие задачи классификации, обычно рассматривается как задача обучения с учителем, поэтому при её решении используются соответствующие методы машинного обучения. Среди них можно выделить:

- наивный байесовский классификатор;
- метод опорных векторов;
- метод k-ближайших соседей;
- деревья принятия решений (с разными алгоритмами построения дерева: CART, C4.5, C5.0);
- методы бэггинга (случайный лес);
- методы бустинга (Adaboost, XGBoost);
- разные виды нейронных сетей: CNN, CNN+RNN, CNN+LSTM, SAE.

Рассмотрим примеры и результаты применения вышеупомянутых методов в исследованиях по теме классификации сетевого трафика.

2.1.1 Наивный байесовский классификатор

Наивный байесовский классификатор – простой вероятностный классификатор, основанный на применении Теоремы Байеса со строгими (наивными) предположениями о независимости. По теореме Байеса,

$$P(A|B) = (P(B|A)*P(A) / P(B)),$$

где A – класс, B – признак. Для предсказания неизвестного класса вычисляется его апостериорная вероятность. Все признаки считаются независимыми, вероятности напрямую вычисляются из обучающих данных (дискретные значения) или оцениваются через нормальное распределение.

Наивный байесовский классификатор является одним из самых простых методов машинного обучения, к его достоинствам можно отнести высокую скорость обучения и работы, однако он зачастую показывает результаты хуже, чем другие методы, поэтому его применение на практике ограничено. Можно упомянуть пример статьи [11], в котором он проигрывает в сравнении с другими рассмотренными классификаторами при классификации трафика, но показывает хорошие результаты как способ обобщения предсказаний разных типов классификаторов (см. подразд. 4.5, бэггинг).

2.1.2 Метод опорных векторов

В методе опорных векторов (SVM, support vector machine) каждый объект данных представляет из себя точку в p-мерном пространстве (где p - количество признаков), и алгоритм пытается построить гиперплоскость размерности (p−1), максимально эффективно разделяющую точки, относящиеся к разным классам. Таким образом возможна классификация на два класса, для проведения многоклассовой классификации на N классов посредством метода опорных векторов могут использоваться техники:

- попарного сравнения: строится N*(N-1)/2 классификаторов, каждый из которых учится различать между собой два класса;
- один против всех: строится N классификаторов, каждый из которых учится отличать один класс от всех остальных.

К достоинствам метода опорных векторов можно отнести его эффективность в многомерных пространствах признаков. Однако, этот метод достаточно затратен по памяти и вычислительной сложности и может легко переобучаться, то есть подстраиваться только под параметры конкретного множества примеров, на которых он обучался. В то же время его результаты для задачи классификации сетевого трафика хуже, чем у большинства других классификаторов [11], поэтому в настоящее время он редко используется с этой целью.

2.1.3 Метод k-ближайших соседей

Метод k-ближайших соседей присваивает каждому классифицируемому примеру то значение класса, которое наиболее распространено среди его k ближайших согласно выбранной функции расстояния соседей, классы которых известны. В качестве функции расстояния часто выбирается евклидова метрика. Такой подход не даёт высокую точность сам по себе, но может применяться в сочетании с кластеризацией, как в [17] для улучшения результатов выделения из трафика отдельных неизвестных классов.

2.1.4 Дерево принятия решений

Дерево принятия решений (decision tree) представляет собой бинарное дерево, в вершинах которого записаны атрибуты (признаки), по которым различаются различные ситуации, на рёбрах – значения этих атрибутов, а в листьях – значения целевой функции. Чтобы классифицировать новый случай, надо спуститься по дереву до листа и выдать соответствующее значение.

Для построения дерева могут применяться разные алгоритмы, в работах по классификации сетевого трафика можно найти примеры использования:

• C4.5 [8, 18-20] – улучшенная версия одного из базовых алгоритмов построения деревьев ID3 (Iterative Dichotomiser 3), которая может использовать как дискретные, так и непрерывные признаки, позволяет задавать веса для признаков, и производит прореживание (pruning) для построенных деревьев с целью их оптимизации. Для выбора признака для разбиения используется информационный выигрыш, основанный на энтропии.

• C5.0 [10, 21] – оптимизация алгоритма C4.0, дающая преимущество по скорости работы и используемой памяти, строящая деревья, сравнимые по эффективности, но меньшего размера.

• CART [22] (Classification and Regression Trees) строит бинарное дерево решений с использованием критерия Джини.

Деревья принятия решений и решения, построенные на их основе, являются одним из самых популярных способов решения задачи классификации трафика, так как среди их достоинств можно перечислить:

- отсутствие необходимости в специальной подготовке данных (нормализация, добавление фиктивных переменных для приведения примеров к единому размеру и др.) как при работе с нейронными сетями (см. [23]),
- способность работы с разными типами переменных (как категориальными, так и интервальными),
- способность работать с большими объёмами данных,
- хорошие результаты классификации [11, 18],
- высокую скорость работы и низкую вычислительную сложность предсказания результата построенным деревом [18],
- простоту мультиклассовой классификации (в сравнении, например, с SVM).

К недостаткам этого метода можно отнести:

• неоптимальность алгоритмов построения дерева (проблема получения оптимального дерева решений является NP-полной, поэтому оптимальное решение выбирается локально в каждом узле, что влияет на оптимальность дерева в целом),

• риск переобучения (необходимо регулировать глубину дерева),

• сложность выражений некоторых ситуаций посредством дерева (например, XOR),

• неустойчивость деревьев при небольших изменениях входных данных.

С этими недостатками можно бороться применением алгоритмов бэггинга и бустинга, которые рассматриваются далее.

2.1.5 Бэггинг

Бэггинг (bagging от bootstrap aggregating) – это способ композиции нескольких более простых классификаторов в один с целью повысить его

стабильность и точность. В частности, широко используется алгоритм случайный лес (Random Forest) [9, 11, 13, 24, 25], заключающийся в использовании комитета (ансамбля) решающих деревьев. Основная идея этого метода в использовании большого ансамбля деревьев, каждое из которых само по себе даёт очень невысокое качество классификации, но за счёт их большого количества результат получается хорошим. Классификация объектов проводится путём голосования: каждое дерево комитета относит классифицируемый объект к одному из классов, и побеждает класс, за который проголосовало наибольшее число деревьев.

В приведённых статьях метод бэггинга выбирается как основной для классификации или показывает одни из лучших результатов в сравнении с другими методами. Получающиеся модели сложнее, чем для рассмотренных выше классификаторов, а время на классификацию одного примера растёт примерно линейно относительно количества простых классификаторов в ансамбле, однако именно этот метод зачастую даёт наилучший вариант компромисса между качеством и скоростью классификации.

2.1.6 Бустинг

Бустинг (boosting) – способ организации классификаторов, позволяющий упорядочить и обучить более простые классификаторы (чаще всего это решающие деревья небольшой глубины) таким образом, чтобы результирующий классификатор показывал лучшие результаты.

Примерами алгоритмов бустинга являются:

• AdaBoost (Adaptive Boosting) [13, 18, 26] – регулирует веса в процессе обучения, чтобы примерам, на которых ошибся прошлый классификатор, придавалось большее значение;

• XGBoost [24] – ошибка минимизируется алгоритмом градиентного спуска.

По качеству и скорости работы бустинг примерно сравним с бэггингом (зависит от конкретных данных; один из примеров такого сравнения можно найти в [13]).

Можно видеть, что из рассмотренных на данный момент моделей машинного обучения для классификации сетевого трафика чаще всего и с наибольшим успехом используются деревья принятия решения (с разными алгоритмами построения), а также их комбинации методами бэггинга или бустинга.

2.1.7 Нейронные сети

Искусственные нейронные сети состоят из нескольких слоёв искусственных нейронов, каждый из которых получает на вход несколько числовых значений и преобразует их в выходное значение в соответствии со своими внутренними правилами (заранее заданной функцией активации и вычисляемыми весами входных параметров). Слой сети может состоять из произвольного числа нейронов, каждый из которых может быть соединён с любыми нейронами из предыдущего и последующего слоёв.

Существуют модели так называемых рекуррентных нейронных сетей с обратной связью, когда сигнал с выходных нейронов или нейронов скрытого слоя частично передается обратно на входы нейронов входного слоя, но чаще всего нейронные сети являются сетями прямого распространения (feedforward). Значения нейронов в каждом слое вычисляются на основании значений предыдущего слоя в ходе процесса, называющегося алгоритмом прямого распространения (forward propagation). В процессе обучения изменение параметров направлено на минимизацию функции штрафа (cost function). Для коррекции весов в процессе обучения применяется алгоритм обратного распространения ошибки (backpropagation).

Для работы с большим количеством параметров во внутренних слоях сети или для определения инвариантных относительно переноса признаков свою эффективность показали свёрточные нейронные сети (CNN), использующие

набор небольших ядер для преобразования поступающей информации и уменьшения числа извлекаемых признаков [12, 23, 27-30].

Рекуррентные сети (RNN), как уже упоминалось ранее, могут использоваться для работы с признаками, имеющими зависимости во времени (например, в случае, когда текущий ответ сети должен зависеть не только от текущих признаков, но и от некоторых признаков, использовавшихся при принятии предыдущего решения). Для этого сеть особым образом сохраняет часть извлекаемой информации для дальнейшего использования. Самым часто используемым представителем является LSTM (Долгая краткосрочная память от англ. Long short-term memory) [27, 28].

Автокодировщики (Autoencoders) представляют из себя нейронные сети, состоящие из двух частей: первая половина сети учится кодировать входную информацию в сжатом виде, а вторая учится с максимально возможной точностью воссоздавать эту информацию по сжатому представлению. Поскольку получаемый метод кодирования эффективно работает только для того типа данных, на котором обучалась сеть, другие данные будут воссоздаваться с ошибкой, что позволяет использовать автокодировщики для классификации [12, 31, 32].

Генеративно-состязательная сеть (GAN, Generative adversarial network) - комбинация из двух нейронных сетей, одна из которых учится генерировать образцы данных, похожие на реальные, а вторая - отличать эти образцы. В процессе совместного обучения качество работы обеих этих сетей растёт, что позволяет генерировать искусственные данные, почти неотличимые от реальных. Такой подход может использоваться для генерации дополнительных примеров при невозможности получить их достаточное количество другим путём [9].

Также существуют примеры совместного использования нескольких типов нейронных сетей для получения лучших результатов, в частности, CNN+RNN [14, 27, 33].

2.2 Признаки сетевого трафика, используемые для его классификации

Алгоритмы машинного обучения для своей работы нуждаются в получении признаков классифицируемых примеров, на основе которых будет приниматься решение. Для задачи классификации сетевого трафика можно выделить два основных способа их получения:

- признаки выделяются на специальном этапе подготовки данных на основе некоторой внешней информации (например, знания эксперта в области); такие признаки могут включать в себя как содержимое пакетов, так и дополнительную информацию о потоке (такую, как общее количество пакетов, размер переданных данных, время получения пакетов и т.п.);
- признаки выделяются из данных (обычно это содержимое пакетов) самой моделью в процессе так называемого глубокого обучения.

Первый подход требует от экспериментатора дополнительных усилий, но позволяет лучше контролировать процесс и использовать любую доступную информацию и её производные (например, посчитанные на основе имеющихся данных статистики). Второй подход требует минимальных действий при подготовке данных (нормализовать данные, унифицировать длину примеров, переупорядочить данные, если это необходимо) и может позволить находить неочевидные на первый взгляд зависимости, но извлекаемый им набор признаков не всегда является лучшим или минимальным для решения поставленной задачи. Кроме того, второй подход обычно требует для обучения большего количества данных.

По содержанию признаков их можно несколько условно разбить на следующие классы:

- данные пакета;
- метаинформация о пакете;
- временные характеристики;
- информация о потоке.

Первая категория включает в себя содержимое полезной нагрузки пакета или все байты пакета без из разделения на заголовок и данные. Именно она чаще всего используется при глубоком обучении модели классификации [12, 23, 27]. Выделяемые из содержимого пакетов закономерности очень информативны (такой подход является в некотором смысле автоматизированным аналогом DPI). В [12] показано, что такой метод может применяться и для классификации зашифрованного трафика, однако количество информации о пакете, используемой для его классификации достаточно велико (1480 байтов). В [27] поднимается вопрос о возможности переобучения сети под искусственно сгенерированный набор данных: при таком наборе признаков модель способна запоминать IP-адреса или номера портов, которые используются для разных классов. В таком случае показанные ей в экспериментах результаты будут высокими, но такая модель не будет применима для реальных сетей. Эта возможность не учтена в [23], где используются 784 первых байтов пакета, включая все уровни заголовка.

Под метаинформацией о пакете можно понимать такие признаки, как размер пакета, размер его полезной нагрузки, направление движения, а также тип сервиса, указанный тип протокола, установленные флаги и размер окна (для TCP). Различные комбинации этих признаков используются во многих работах по теме.

Временные характеристики включают в себя интервалы времен между прибытием пакетов (этот признак весьма показателен для классификации трафика по разным сценариям использования), общее время сессии. Также, в статье [21] предложена идея использования временных вспышек трафика. Временная вспышка (burst) трафика – это группа последовательных пакетов с интервалами прибытия между ними меньше, чем между разными вспышками. Характеристики вспышек в потоке трафика (burstiness) являются мерой изменчивости распределения времён прибытия пакетов. В новом варианте исследований [10] от тех же авторов был добавлен ещё такой признак, как время бездействия потока.

Информация о потоке объединяет в себе метаинформацию всех или части пакетов (в целом или в каждом направлении отдельно) в потоке в виде некоторых её статистических характеристик: сумма по всех пакетам, минимум, максимум, среднее арифметическое, медиана, дисперсия и т.д. Также, могут вычисляться аналогичные характеристики для временного распределения пакетов. Эти признаки дополняют и расширяют информацию, получаемую из двух предыдущих категорий признаков, но требуют для своего вычисления завершения потока, что затрудняет их использование в режиме классификации в реальном времени. Для решений этой проблемы можно использовать не все, а только N первых пакетов в потоке (например, [23]. Значение N подлежит экспериментальному вычислению или выбирается на основе экспертной оценки.

Итак, разные исследования задачи классификации трафика используют разные категории признаков, одну или несколько из перечисленных выше. Задача выбора и отбора признаков весьма актуальна и решается практически в каждом новом исследовании. Существуют статьи [34] и инструменты [35], посвящённые описанию и получения максимально возможного количества доступных признаков для пакетов в потоке данных.

2.3 Выбор и подготовка набора данных, их сравнение

Препятствием для прямого сравнения различных подходов в работах исследовательских групп по классификации трафика является не только наличие нескольких разных принципов классификации (по протоколам/приложениям/типам приложений/...), но и отсутствие единого общепризнанного набора данных, на которых бы проводилось тестирование предлагаемых методов. На настоящий момент отсутствие такого стандартного набора означает, что каждая исследовательская группа самостоятельно находит/собирает данные для обучения и тестирования моделей, размечает их по своей системе своими силами и способами и публикует полученные на них результаты. Это не позволяет сопоставлять эти результаты между собой

напрямую, как делается при решении многих типовых задач, что является существенным недостатком в данной области.

Получение большой и репрезентативной выборки данных для обучения и тестирования моделей – первая проблема, которая возникает при решении задачи классификации данных. Каждая исследовательская группа должна найти для себя подходящий источник данных, убедиться в корректности его разметки соответственно поставленной задаче и оценить его полноту относительно возможных вариантов организации трафика в сети. Количество существующих приложений/протоколов огромно, поэтому производить полную классификацию вряд ли представляется возможным. Обычно исследователи выбирают лишь некоторый набор из возможных вариантов (наиболее частые/характерные/представляющие наибольший интерес) и работают только с ним. Однако в таком случае возникает проблема обработки данных, не входящих в данный набор классов. В некоторых работах эти данные просто не рассматриваются, но этот подход неизбежно сталкивается с проблемами при работе с данными реального мира. Ещё одной проблемой является частое появление новых приложений/протоколов, которые либо должны своевременно отражаться в данных, либо обрабатываться особым образом. Также, стоит учитывать, что от метода сбора данных зависит не только их репрезентативность, но и распределение вероятностей классов, что также может влиять на конечный результат.

2.3.1 Основные аспекты выбора набора данных

При выборе или получении данных особое внимание следует уделять следующим аспектам.

2.3.2 Получение правильной разметки данных

При применении систем, использующих методы машинного обучения для классификации сетевого трафика, помимо тестовой выборки для оценки

результатов нужна достаточно большая тренировочная выборка с правильно размеченными ответами. Соответственно, возникает проблема получения этой разметки. Источников здесь может быть несколько.

a) Сторонние системы классификации. Например, можно использовать методы, разбирающие и анализирующие всю информацию, содержащуюся в пакете, для построения более легковесных и оперативных классификаторов. Под это определение подходят системы DPI, такие как Wireshark [36], nDPI [37] и др. Сравнение некоторых доступных для использования систем DPI произведено в [38, 39]. Для проверки правильности разметки инструментов предлагается протестировать их работу на множестве данных, для которых специальная программа регистрирует генерирующие их приложения, что позволяет достичь заведомо высокого качества разметки. Исследование показывает, что использование инструментов DPI даёт высокую, но не идеальную точность при определении приложений.

Соответственно, при этом подходе, следует учитывать, что точность такой разметки, а соответственно, и обучаемых на ней алгоритмов, ограничена точностью этих сторонних методов. Кроме того, этот способ может быть неприменим при работе с зашифрованным трафиком.

b) Получение данных в контролируемых условиях. В этом случае нужно организовать сбор информации как можно ближе к пользователю и поставить перед ним задачу генерировать только заранее определённый трафик. Самая распространённая проблема в этом случае – фильтрация фонового трафика, доля которого может доходить до 70%. Для контроля за получением трафика можно либо собирать его в процессе исполнения специальных скриптов, генерирующих только определённые его классы (ISCX, NIMS), либо запускать параллельно со сбором трафика специальные программы, позволяющие определить его источник (UPC dataset).

c) Генерация искусственных данных на основе существующих. Ещё один способ получения дополнительных данных для тех классов, в которых этих данных недостаточно для обучения модели, – это дополнение тренировочного

множества искусственными данными. Например, в [9, 28] для этой цели используется LSTM. Как показано, такой метод позволяет улучшать качество построенной модели, особенно в тех случаях, когда некоторые из классов недостаточно представлены в тренировочном множестве. Однако, для его использования нужен хорошо обученный генератор данных, что является отдельной задачей, которую также надо решать. Кроме того, нужно следить, чтобы получаемые таким образом данные не были излишне однообразными и соответствовали реальному положению дел в сети.

2.3.3 Доступность данных

Из-за шифрования данных или в связи с необходимостью соблюдения конфиденциальности пользователей, часть данных в пакетах может быть искажена или недоступна. Например, при публикации снимков сетевых трасс может производиться специальное кодирование IP адресов или удаление полезной нагрузки пакетов. Подобные действия могут создавать трудности, особенно при разметке данных, однако этический вопрос в данном случае должен иметь преимущество. Одним из способов поиска компромисса в данной ситуации является сбор статистики поведения реальных пользователей сети интернет и написание специальных алгоритмов-ботов, которые будут имитировать это поведение с максимальной правдоподобностью, при этом не выдавая никакую конфиденциальную информацию.

Такой подход к решению проблемы получил достаточно широкое распространение и позволил получить содержательные датасеты для многих задач, связанных с анализом трафика в сети. Одним из его существенных недостатков является, однако, большая упорядоченность и детерминизм действий по сравнению с реальными пользователями, что может нарушать естественное распределение статистических признаков потоков данных.

2.3.4 Место получения данных

Получаемые для анализа потоки данных зависят от места их сбора. Получение данных как можно ближе к конечным пользователям позволяет проще решать вопрос с шифрованием данных, в то время как сбор информации у провайдера, на глобальных маршрутизаторах позволяет получить больший объём разнообразных и репрезентативных данных. В то же время, такие особенности, как изменение временного распределения в потоке, изменение длины пакетов при туннелировании, наличие на маршрутизаторе, служащем точкой сбора информации, только одного направления потока данных, - все они могут приводить к тому, что модель, обученная на данных из одного источника, может не работать на данных из другого места. Поэтому, предпочтительной стратегией в данном случае является использование в качестве тренировочных данных, получаемых из тех же точек в сети, где предполагается работа создаваемой системы.

В [40] исследуется влияние особенностей характеристик сетей на работу классификатора, в частности показано падение результатов работы классификатора при его тестировании на трафике из сети, отличной от той, на которой он обучался.

2.3.5 Репрезентативность набора данных

После выбора системы используемых классов и разметки данных требуется убедиться в том, что полученный набор данных является репрезентативным, то есть содержит достаточное количество разнообразных примеров для каждого класса (по возможности, покрывает все возможные ситуации). Следует нивелировать перекос по количеству данных для разных классов для предотвращения переобучения классификатора. Некоторые источники получения данных требуют особого внимания и предобработки на данном этапе. Например, если набор данных получен только от нескольких пользователей, то нужно постараться убедиться, что модель сможет выделить из

данных глобальные признаки, а не будет пытаться определить специфику работы каждого конкретного пользователя.

2.4 Используемые общедоступные наборы данных

Поскольку сам процесс получения большого количества релевантных данных, пусть и неразмеченных, может быть затруднителен, в исследованиях нередко используются широко доступные трассы сетевого трафика, которые можно найти в интернете. Обычно такие трассы содержат достаточно большой объём данных и различаются по месту и способу их получения, а также предоставленной разметке (если она есть). К наиболее популярным из общедоступных наборов данных сетевого трафика можно отнести следующие.

• MooreSet (2007) [18]. Около 59 Гигабайтов данных. Данные были собраны на границе сети университетского кампуса в течение 8 месяцев. Набор данных содержит только потоки TCP. Большая их часть классифицирована вручную на основе содержания на 10 классов, также присутствуют некоторые фоновые потоки и потоки без начала. Классы: web-browsing, mail, bulk, attack, p2p, database, multimedia, service, interactive, games.

• UPC (Политехнический университет Каталонии) dataset (2013) [38]. Около 36 Гб данных, собранных за 66 дней. Ради возможности публикации данных с полной полезной нагрузкой трафик был сгенерирован искусственно, авторы постарались максимально имитировать реальный трафик. Для 91% пакетов и 42% потоков присутствует название приложения, полученное с помощью специальной программы, записывающей информацию о каждом сетевом потоке, включая название приложения.

• CAIDA (2008-...). Содержит набор анонимизированных трасс, собранных на мониторах магистральных сетей связи. С 2008 по 2014 год в набор добавлялась одна часовая трасса в месяц, с 2014 года – раз в три месяца. Из пакетов удалена полезная нагрузка, IP-адреса зашифрованы инструментом Crypto-PAn (Cryptography-based Prefix-preserving Anonymization) [41]. Размер каждой трассы – несколько миллиардов пакетов.

• ISCX (2016). Содержит несколько трасс с искусственно сгенерированными данными для решения разных задач сетевого трафика (поиск аномального поведения, поиск ботнетов, классификация трафика и т.д.). Данные сгенерированы так, чтобы по возможности имитировать поведение реальных пользователей. Наиболее популярным набором данных в исследованиях по классификации трафика является VPN-nonVPN (ISCXVPN2016) [42]. Эта трасса содержит 7 категорий трафика: браузинг, электронная почта, чаты (мгновенные сообщения), стриминг, передача файлов, VoIP, p2p. При этом каждая категория представлена в двух видах - через VPN и без него. Полезная нагрузка пакетов сохранена, общий объём файлов составляет 28Гб.

• NIMS (2007). Этот набор данных был собран в специально собранной модели сети посредством прописанных сценариев её использования. Основной целью сбора трафика являлось получение SSH трафика, а в качестве фонового были собраны DNS, HTTP, FTP, P2P (limewire) и telnet. Всего датасет содержит около 700 000 потоков, из них около 35000 – SSH потоков.

Таким образом, единого набора данных для решения задач классификации трафика не существует, и каждая исследовательская группа должна найти или подготовить данные в соответствии со своими целями. При принятии решения о выборе данных для обучения и тестирования классификатора нужно учитывать приведённые выше соображения и ограничения, чтобы полученная модель соответствовала построенной задаче и была способна работать в реальных условиях.

3. ТЕХНОЛОГИЧЕСКИЙ РАЗДЕЛ

3.1 Python

Написать про python (почему python), библиотеки, jupyter. Что это такое, что они делают.

Потом технические характеристики моего ПК.

Python — высокоуровневый язык программирования общего назначения. Относится к интерпретируемым языкам. «Питон» — мультипарадигмальный язык программирования. Он поддерживает объектно-ориентированный и структурный подходы, функциональное и аспектно-ориентированное программирование. В Python используется динамическая типизация.

Python часто используют в сфере искусственного интеллекта, в частности при помощи таких библиотек, как scikit-learn (sklearn), pandas, numpy, keras и tensorflow.

Разработка приложений с использованием ИИ отличается от обычной разработки. Для работы с ИИ нужен особый стек технологий и особые навыки. Кроме того, создание приложений на основе ИИ требует глубоких исследований.

Для реализации идей, связанных с применением ИИ, необходим надежный, гибкий язык программирования с богатым инструментарием. Python – такой язык, и поэтому на нем разрабатывается множество ИИ-проектов.

Python помогает разработчикам работать продуктивно и уверенно, причем на всех стадиях проекта, от разработки до поддержки. Этот язык обладает определенными характеристиками, которые делают его наилучшим выбором для ML- и ИИ-проектов: он простой и логичный, гибкий и мультиплатформенный, имеет отличные библиотеки и фреймворки для машинного обучения и работы с ИИ, а еще за ним стоит многочисленное сообщество разработчиков. Благодаря всему этому Python является одним из самых популярных языков программирования в мире, причем не только в сфере ML и ИИ.

В Python легко писать лаконичный и читаемый код. Несмотря на то, что за машинным обучением и искусственным интеллектом стоят сложные алгоритмы

и процессы, простота Python позволяет создавать надежные системы. Разработчики могут полностью сосредоточиться на задачах, которые они пытаются решить при помощи ML, не отвлекаясь на технические нюансы языка.

Кроме того, Python прост в изучении. Написанный на нем код легко понятен человеку, что упрощает создание моделей для машинного обучения.

Python интуитивнее многих других языков программирования. Также в наличие есть множество фреймворков, библиотек и расширений, упрощающих реализацию задуманных функций.

Поскольку Python — язык общего назначения, с его помощью можно решать многие сложные задачи машинного обучения и быстро создавать прототипы для последующей их отладки.

Создание алгоритмов ИИ и машинного обучения — это сложная задача, требующая много времени. И чтобы было легче находить оптимальные пути решения задач, программистам нужна хорошо структурированная и надежная среда разработки.

Многочисленные фреймворки и библиотеки Python помогают существенно уменьшить количество времени, необходимого для разработки приложений.

Python, с его богатым стеком технологий, имеет обширный набор библиотек для искусственного интеллекта и машинного обучения.

Мультиплатформенность (в нашем случае) — это свойство языка программирования или фреймворка, позволяющее разработчикам переносить ПО на разные машины с минимальными изменениями, либо без изменений вовсе.

Одной из причин популярности Python является то, что этот язык от платформ не зависит, так как поддерживается многими из них, включая Linux, Windows и macOS.

Код Python может использоваться для создания программ для большинства операционных систем, а это означает, что программное обеспечение Python

легко распространять и использовать в этих системах без специальных интерпретаторов.

Обычно для своих вычислительных нужд разработчики используют такие сервисы, как Google или Amazon. Но они также могут использовать свои собственные машины с мощными графическими процессорами (GPU) для обучения своих моделей машинного обучения.

3.2 Scikit-learn

Scikit-learn - один из наиболее широко используемых пакетов Python для Data Science и Machine Learning. Он позволяет выполнять множество операций и предоставляет множество алгоритмов. Scikit-learn также предлагает отличную документацию о своих классах, методах и функциях, а также описание используемых алгоритмов.

Scikit-Learn поддерживает:

- предварительную обработку данных;
- уменьшение размерности;
- выбор модели;
- регрессии;
- классификации;
- кластерный анализ.

Он также предоставляет несколько наборов данных, которые вы можете использовать для тестирования ваших моделей.

Scikit-learn не реализует все, что связано с машинным обучением. Например, он не имеет комплексной поддержки для:

- нейронных сетей;
- самоорганизующихся карт (сетей Кохонена);
- обучения ассоциативным правилам;
- обучения с подкреплением (reinforcement learning).

Scikit-learn основан на NumPy и SciPy.

Scikit-learn - это пакет с открытым исходным кодом. Как и большинство материалов из экосистемы Python, он бесплатный даже для коммерческого использования. Он лицензирован под лицензией BSD.

3.3 Pandas

pandas — программная библиотека на языке Python для обработки и анализа данных. Работа pandas с данными строится поверх библиотеки NumPy, являющейся инструментом более низкого уровня. Предоставляет специальные структуры данных и операции для манипулирования числовыми таблицами и временны́ми рядами. Название библиотеки происходит от эконометрического термина «панельные данные», используемого для описания многомерных структурированных наборов информации. pandas распространяется под новой лицензией BSD.

Основная область применения — обеспечение работы в рамках среды Python не только для сбора и очистки данных, но для задач анализа и моделирования данных, без переключения на более специфичные для статобработки языки (такие, как R и Octave).

Пакет прежде всего предназначен для очистки и первичной оценки данных по общим показателям, например среднему значению, квантилям и так далее; статистическим пакетом он в полном смысле не является, однако наборы данных типов DataFrame и Series применяются в качестве входных в большинстве модулей анализа данных и машинного обучения (SciPy, Scikit-Learn и других).

Основные возможности библиотеки:

- Объект *DataFrame* для манипулирования индексированными массивами двумерных данных.
- Инструменты для обмена данными между структурами в памяти и файлами различных форматов
- Встроенные средства совмещения данных и способы обработки отсутствующей информации

- Переформатирование наборов данных, в том числе создание сводных таблиц
- Срез данных по значениям индекса, расширенные возможности индексирования, выборка из больших наборов данных
- Вставка и удаление столбцов данных
- Возможности группировки позволяют выполнять трёхэтапные операции типа «разделение, изменение, объединение» (англ. *split-apply-combine*).
- Слияние и объединение наборов данных
- Иерархическое индексирование позволяет работать с данными высокой размерности в структурах меньшей размерности
- Работа с временными рядами: формирование временных периодов и изменение интервалов и так далее.

Библиотека оптимизирована для высокой производительности, наиболее важные части кода написаны на Cython и Си.

3.4 NymPy

NumPy — это библиотека Python, которую применяют для математических вычислений: начиная с базовых функций и заканчивая линейной алгеброй. Полное название библиотеки — Numerical Python extensions, или «Числовые расширения Python».

У этой библиотеки есть несколько важных особенностей, которые сделали ее популярным инструментом. Во-первых, исходный ее код в свободном доступе хранится на GitHub.

Во-вторых, библиотека написана на языках C и Fortran. Это компилируемые языки (языки программирования, текст которых преобразуется в машинный код — набор инструкций для конкретного типа процессора. Преобразование происходит с помощью специальной программы-компилятора, благодаря нему вычисления на компилируемых языках происходят быстрее), на которых вычисления производятся гораздо быстрее и эффективнее, чем на интерпретируемых языках (языки программирования, которые не заточены под

конкретный тип процессора и могут быть запущены на разных типах устройств). К этим языкам относится и сам Python.

Где используется NumPy:

- Научные вычисления. NumPy пользуются ученые для решения многомерных задач в математике и физике, биоинформатике, вычислительной химии и даже когнитивной психологии.
- Создание новых массивных библиотек. На основе NumPy появляются новые типы массивов, возможности которых выходят за рамки того, что предлагает библиотека. Например, библиотеки Dask, CuPy или XND.
- Data Science. В основе экосистемы для анализа данных лежит NumPy. Библиотека используется на всех этапах работы с данными: извлечение и преобразование, анализ, моделирование и оценка, репрезентация.
- Machine Learning. Библиотеки для машинного обучения scikit-learn и SciPy тоже работают благодаря вычислительным мощностям NumPy.
- Визуализация данных. По сравнению непосредственно с Python возможности NumPy позволяют исследователям визуализировать наборы данных, которые гораздо больше по размеру. Например, библиотека лежит в основе системы PyViz, которая включает в себя десятки программ для визуализации.

3.5 Keras

Keras — открытая библиотека, написанная на языке Python и обеспечивающая взаимодействие с искусственными нейронными сетями. Она представляет собой надстройку над фреймворком TensorFlow. До версии 2.3 поддерживались разные версии нейросетевых библиотек, такие как TensorFlow, Microsoft Cognitive Toolkit, Deeplearning4j, и Theano. Нацелена на оперативную работу с сетями глубинного обучения, при этом спроектирована так, чтобы быть компактной, модульной и расширяемой. Она была создана как часть исследовательских усилий проекта *ONEIROS* (англ. *Open-ended Neuro-Electronic*

Intelligent Robot Operating System), а ее основным автором и поддерживающим является Франсуа Шолле (фр. *François Chollet*), инженер Google.

Эта библиотека содержит многочисленные реализации широко применяемых строительных блоков нейронных сетей, таких как слои, целевые и передаточные функции, оптимизаторы, и множество инструментов для упрощения работы с изображениями и текстом. Ее код размещен на GitHub, а форумы поддержки включают страницу вопросов GitHub, канал Gitter и канал Slack.

На сентябрь 2016 года Keras является второй по скорости роста системой глубокого обучения после TensorFlow Google, и третьей по размеру после TensorFlow и Caffe.

3.6 TensorFlow

TensorFlow — открытая программная библиотека для машинного обучения, разработанная компанией Google для решения задач построения и тренировки нейронной сети с целью автоматического нахождения и классификации образов, достигая качества человеческого восприятия. Применяется как для исследований, так и для разработки собственных продуктов Google. Основной API для работы с библиотекой реализован для Python, также существуют реализации для R, C Sharp, C++, Haskell, Java, Go и Swift.

Является продолжением закрытого проекта DistBelief. Изначально TensorFlow была разработана командой Google Brain для внутреннего использования в Google, в 2015 году система была переведена в свободный доступ с открытой лицензией Apache 2.0.

TensorFlow 9 ноября 2015 года был открыт для свободного доступа. TensorFlow является системой машинного обучения Google Brain второго поколения. В то время как эталонная реализация работает на единичных устройствах, TensorFlow может работать на многих параллельных процессорах, как CPU, так и GPU, опираясь на архитектуру CUDA для поддержки вычислений общего назначения на графических процессорах. TensorFlow доступна для 64-

разрядных Linux, macOS, Windows, и для мобильных вычислительных платформ, включая Android и iOS.

Вычисления TensorFlow выражаются в виде потоков данных через граф состояний. Название TensorFlow происходит от операций с многомерными массивами данных, которые также называются «тензорами».

3.7 Набор данных

Разработанные нейросети используют набор данных (dataset) https://www.kaggle.com/datasets/jsrojas/ip-network-traffic-flows-labeled-with-87-apps?resource=download.

3.8 Метод опорных векторов

Машины опорных векторов — семейство алгоритмов бинарной классификации, основанных на обучении с учителем, использующих линейное разделение пространства признаков с помощью гиперплоскости.

Основная идея метода заключается в отображение векторов пространства признаков, представляющих классифицируемый объекты, в пространство более высокой размерности. Это связано с тем, что в пространстве большей размерности линейная разделимость множества оказывается выше, чем в пространстве меньшей размерности. Причины этого интуитивно понятны: чем больше признаков используется для распознавания объектов, тем выше ожидаемое качество распознавания.

После перевода в пространство большей размерности, в нём строится разделяющая гиперплоскость. При этом все векторы, расположенные с одной «стороны» гиперплоскости, относятся к одному классу, а расположенные с другой — ко второму. Также, по обе стороны основной разделяющей гиперплоскости, параллельно ей и на равном расстоянии от неё строятся две вспомогательные гиперплоскости, расстояние между которыми называют **зазор**.

Задача заключается в построении разделяющей гиперплоскость таким образом, чтобы максимизировать зазор — область пространства признаков между вспомогательными гиперплоскостями, в которой не должно быть векторов. Предполагается, что разделяющая гиперплоскость, построенная по данному правилу, обеспечит наиболее уверенное разделение классов и минимизирует среднюю ошибку распознавания.

Векторы, которые попадут на границы зазора (т.е. будут лежать на вспомогательных гиперплоскостях), называют **опорными векторами** (что и дало название методу).

Теория опорных векторов была разработана В.Н. Вапником в 1990г.

Первая реализованная модель машинного обучения реализует классификацию при помощи метода опорных векторов.

Листинг 1 – Классификация при помощи SVM.

```
!pip install pyspark
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
from functools import reduce
from pyspark.sql.functions import isnan, when, count, col
import matplotlib.pyplot as plt

# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will list all files under the input directory

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
```

```
        from pyspark import SparkConf, SparkContext
        from pyspark.sql import SparkSession, SQLContext

        spark =
SparkSession.builder.master('local[*]').config("spark.driver.memory",
"15g").appName('TrafficAnalysisUsingPySpark').getOrCreate()
        print(spark)

        # Load csv to Spark DataFrame
        TRAFFIC_DATA = "/input/Dataset-Unicauca-Version2-87Atts.csv"

        traffic_df = spark.read.options(header='True',inferSchema='True') \
                        .csv(path=TRAFFIC_DATA)

        # Display the schema of DataFrame
        traffic_df.printSchema()

        # Show the first three rows (too many columns!)
        # The columns have trailing whitespaces!
        # traffic_df.show(3)

        # Show dataframe columns
        current_columns = traffic_df.columns

        new_columns = list(map(lambda item : item.replace("
","_").replace(".","_").upper().strip(),current_columns))

        final_df = reduce(lambda data, idx:
data.withColumnRenamed(current_columns[idx], new_columns[idx]),
range(len(current_columns)), traffic_df)
```

```
final_df.printSchema()

# Count rows of df
final_df.count()

# Display newly modified columns
final_df.columns

# Create a subtable by selecting some columns

sub_df = final_df.select('FLOW_ID',
 'SOURCE_IP',
 'SOURCE_PORT',
 'DESTINATION_IP',
 'DESTINATION_PORT',
 'PROTOCOL',
 'TIMESTAMP',
 'FLOW_DURATION',
 'TOTAL_FWD_PACKETS',
 'TOTAL_BACKWARD_PACKETS',
 'TOTAL_LENGTH_OF_FWD_PACKETS',
 'TOTAL_LENGTH_OF_BWD_PACKETS',
 'FLOW_BYTES_S',
 'FLOW_PACKETS_S',
 'AVERAGE_PACKET_SIZE',
 'LABEL',
 'PROTOCOLNAME')
sub_df.show(10)
```

```
        # Check out what protocol name we have
        sub_df.groupBy("PROTOCOLNAME").count().show()

        # Calculate count, mean, stddev, min and max for FLOW_DURATION
        sub_df.select('FLOW_DURATION').describe().show()

        # Check for any NaN values
        sub_df.select([count(when(isnan(c), c)).alias(c) for c in
sub_df.columns]).show()

        # Show distinct values
        sub_df.select('PROTOCOLNAME').distinct().collect()

        socmed = ['TWITTER','INSTAGRAM','FACEBOOK']

        records = sub_df.filter(sub_df.PROTOCOLNAME.isin(socmed))

        records.show(5)

        records_df = records.toPandas()

        records_df

        records_df['TIMESTAMP'] =
pd.to_datetime(records_df['TIMESTAMP'],format= '%d/%m/%Y%H:%M:%S' )

        records_df.head()

        # Create a bar plot in pandas
        records_df["PROTOCOLNAME"].value_counts().plot.bar()
```

```
        # Use seaborn to plot the same graph above
        import seaborn as sns
        plt.figure(figsize=(10,4))
        sns.countplot(x = 'PROTOCOLNAME', data = records_df)

        import networkx as nx

        G = nx.Graph()

        G =
nx.from_pandas_edgelist(records_df[records_df['PROTOCOLNAME']=='INSTAGR
AM'], 'SOURCE_IP', 'DESTINATION_IP')
        # G = nx.from_pandas_edgelist(records_df[records_df, 'SOURCE_IP',
'DESTINATION_IP')

        plt.figure(figsize=(30, 30))
        nx.draw_networkx(G, with_labels=True,node_size=60,font_size=12)

        # nx.draw_shell(G, with_labels=True,node_size=60,font_size=12)
        plt.figure(figsize=(30, 30))
        nx.draw_spring(G, with_labels=True,node_size=60,font_size=12)

        # Print out network edges
        [e for e in G.edges]

        plt.figure(figsize=(30, 30))
        nx.draw_kamada_kawai(G, with_labels=True,node_size=60,font_size=12)

        import plotly.express as px
```

```
        fig =
px.line(records_df[records_df['PROTOCOLNAME']=='INSTAGRAM'],
x='TIMESTAMP', y="AVERAGE_PACKET_SIZE",
                title="Average Packet Size vs Time")
        fig.show()

        from sklearn.metrics import explained_variance_score
        from sklearn.model_selection import train_test_split
        from sklearn.svm import SVC
        from sklearn.impute import SimpleImputer
        from sklearn.preprocessing import LabelEncoder
        from sklearn.metrics import accuracy_score

        # Label encode protocol name
        encoder = LabelEncoder().fit(records_df['PROTOCOLNAME'])
        records_df['PROTOCOLNAME'] =
encoder.fit_transform(records_df['PROTOCOLNAME'])
        records_df['PROTOCOLNAME']

        X = records_df.drop(columns = ['FLOW_ID',
         'SOURCE_IP',
         'SOURCE_PORT',
         'DESTINATION_IP',
         'DESTINATION_PORT',
         'PROTOCOL',
         'TIMESTAMP',
         'LABEL',
         'PROTOCOLNAME'])
        Y = records_df['PROTOCOLNAME']
```

```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2,
random_state=3)

imputer = SimpleImputer()
X_train = imputer.fit_transform(X_train)
X_test = imputer.transform(X_test)

model = SVC()
model.fit(X_train,Y_train)

pred = model.predict(X_test)
explained_variance_score(pred,Y_test)

accuracy_score(pred,Y_test)
```

На рисунке 12 можно увидеть наглядную визулизацию того, какие пакеты встречались чаще всего.

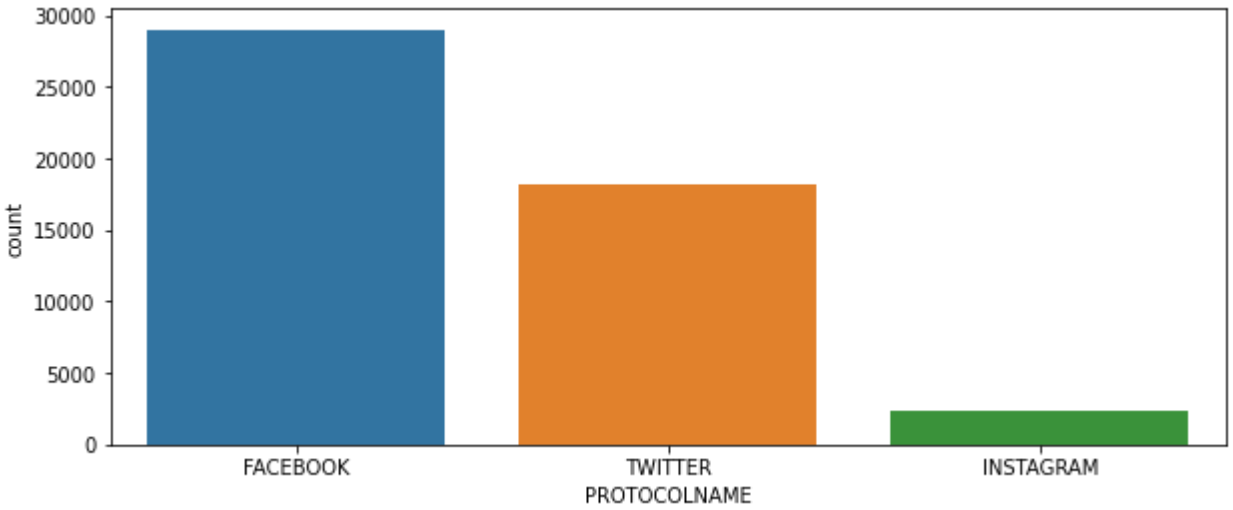

Рис. 12 – Пакеты, чаще всего встречающиеся в наборе данных

На рис.13 можно увидеть, между какими компьютерами были переданы данные, что может оказаться очень полезным, например, для контроля пользователей сети и расследования инцидентов.

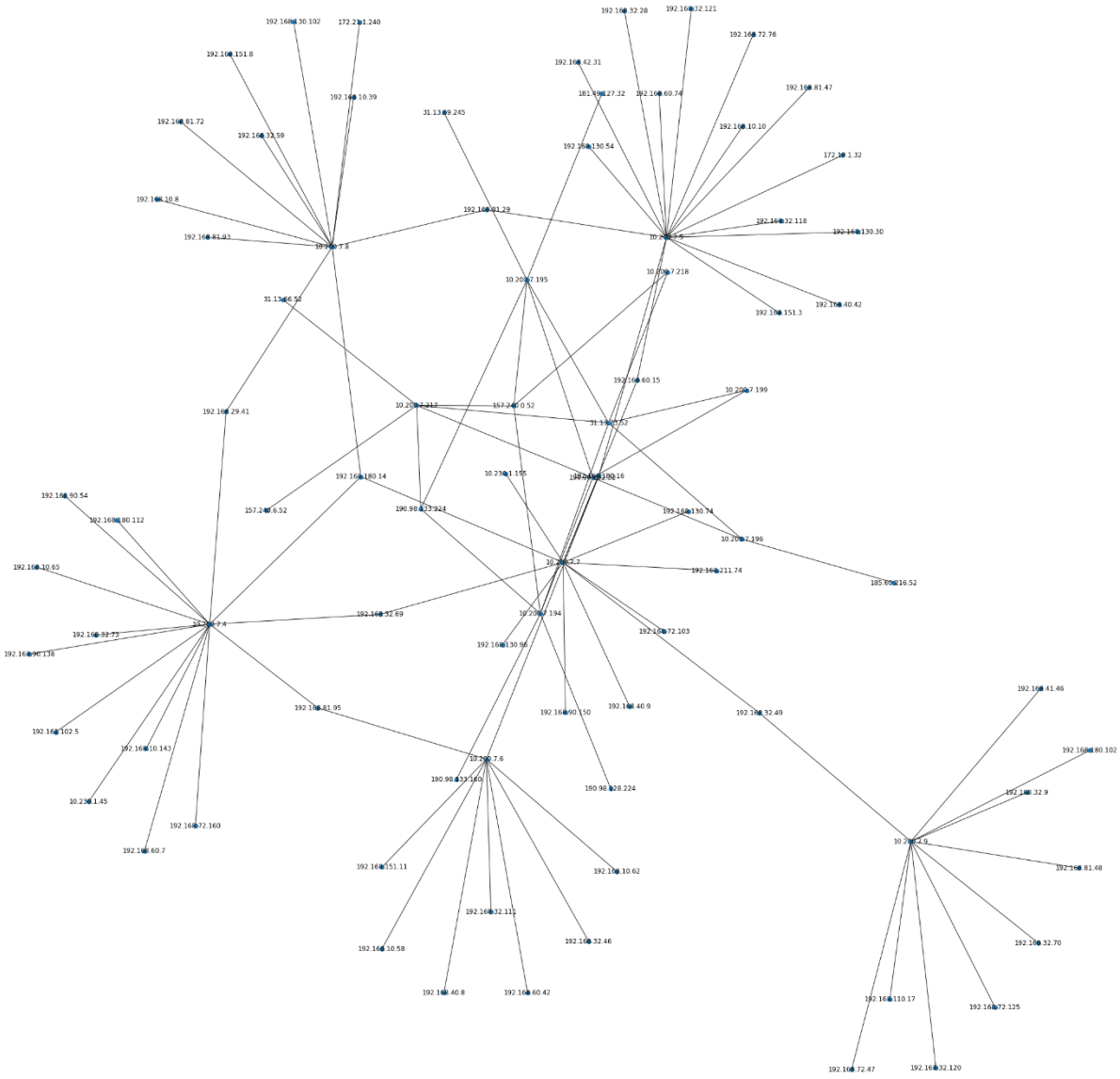

Рис. 13 – Граф коммуникаций между компьютерами в сети.

Реализованная модель имеет точность классификации 0.6513779923556628.

3.9 Рекуррентная нейронная сеть

Следующая модель использует рекуррентную нейронную сеть с 5 слоями. Данная модель показала более высокую точность, а также я добавил

визуализацию наиболее часто отправляющих и принимающих сетевые пакеты IP-адресов.

```
import time
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler, OneHotEncoder
from sklearn.metrics import confusion_matrix, classification_report
import tensorflow as tf
import os

for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

t1 = time.time()

path = '../input/ip-network-traffic-flows-labeled-with-87-apps/Dataset-Unicauca-Version2-87Atts.csv'

dataset = pd.read_csv(path)

dataset

dataset.shape

# Histogram on Source.IP
```

```
Sour_feat = pd.DataFrame(dataset['Source.IP'].value_counts()[:30])
plt.figure(figsize=(20,10))
plt.plot(Sour_feat)
plt.xticks(rotation=90)
plt.xlabel('Source.IP', {'fontsize':15})
plt.ylabel('Counts', {'fontsize':15})
plt.title('Top 30 Counts in Source.IP\n', {'fontsize':20})
plt.grid()
plt.savefig('hist Source.IP.png')
Sour_feat = Sour_feat.reset_index()['index'].values

# Histogram on Destination.IP
Dest_feat = pd.DataFrame(dataset['Destination.IP'].value_counts()[:30])
plt.figure(figsize=(20,10))
plt.plot(Dest_feat)
plt.xticks(rotation=90)
plt.xlabel('Destination.IP', {'fontsize':15})
plt.ylabel('Counts', {'fontsize':15})
plt.title('Top 30 Counts in Destination.IP\n', {'fontsize':20})
plt.grid()
plt.savefig('hist Destination.IP.png')
Dest_feat = Dest_feat.reset_index()['index'].values

Dest_feat

Sour_feat

# Filtering the dataset to contain only 30 frequently reported IP address in
Source.IP and Destination.IP
```

```
        f_dataset = dataset[dataset['Destination.IP'].isin(Dest_feat) &
dataset['Source.IP'].isin(Sour_feat)].reset_index()
        f_dataset = f_dataset.drop('index', axis=1)

        # making dummies
        dum_s = pd.get_dummies(f_dataset['Source.IP'])

        dum_d = pd.get_dummies(f_dataset['Destination.IP'])

        label = pd.get_dummies(f_dataset['ProtocolName'])

        dum_s.shape

        dum_d.shape

        label.shape

        f_dataset.columns

        # removing columns
        f_dataset = f_dataset.drop(f_dataset.select_dtypes(include =
['object']).columns, axis = 1)
        f_dataset =
f_dataset.drop(['Source.Port','Destination.Port','L7Protocol','Protocol'], axis = 1)
        f_dataset.columns

        f_dataset.shape

        p_dataset = pd.concat([f_dataset, dum_s, dum_d], axis=1)
```

```
        # normalizing the data
        scaler = MinMaxScaler()
        n_dataset = scaler.fit_transform(p_dataset)

        n_dataset.shape

        # spliting the dataset
        X = n_dataset
        y = label

        X_train, X_test, y_train, y_test = train_test_split(X, y, train_size = 0.4,
random_state = 101)

        print(X_train.shape,'\n',y_train.shape,'\n',X_test.shape,'\n',y_test.shape,'\n')

        # defining the model
        model = tf.keras.Sequential(
           layers=[tf.keras.layers.Dense(100, input_shape=[X.shape[1]]),
                tf.keras.layers.Dense(100, activation='tanh'),
                tf.keras.layers.Dense(100, activation='tanh'),
                tf.keras.layers.Dense(100, activation='tanh'),
                tf.keras.layers.Dense(y.shape[1], activation='softmax')])

        model.compile(optimizer='adam',
                  loss='categorical_crossentropy',

metrics=['accuracy',tf.keras.metrics.Precision(),tf.keras.metrics.Recall()])

        model.summary()
```

```
        # training the model
        t2 = time.time()
        history = model.fit(X_train, y_train, validation_split=0.2, epochs=100,
verbose=0)
        t3 = time.time()

        # training figures
        plt.figure(figsize=(10,10))

        plt.subplot(411)
        plt.plot(history.history['loss'])
        plt.plot(history.history['val_loss'])
        plt.legend(['loss', 'val_loss'])
        plt.xlabel('epochs')
        plt.ylabel('loss')
        plt.grid()

        plt.subplot(412)
        plt.plot(history.history['accuracy'])
        plt.plot(history.history['val_accuracy'])
        plt.legend(['accuracy', 'val_accuracy'])
        plt.xlabel('epochs')
        plt.ylabel('accuracy')
        plt.grid()

        plt.subplot(413)
        plt.plot(history.history['precision'])
        plt.plot(history.history['val_precision'])
        plt.legend(['precision', 'val_precision'])
        plt.xlabel('epochs')
```

```
        plt.ylabel('precision')
        plt.grid()

        plt.subplot(414)
        plt.plot(history.history['recall'])
        plt.plot(history.history['val_recall'])
        plt.legend(['recall', 'val_recall'])
        plt.xlabel('epochs')
        plt.ylabel('recall')
        plt.grid()

        plt.savefig('training.png')

        # prediction
        loss, accuracy, precision, recall = model.evaluate(X_test, y_test, verbose=0)
        print('loss:      {}\naccuracy:  {}\nprecision: {}\nrecall:    {}\n'.format(loss,
accuracy, precision, recall))

        # time taken
        t4 = time.time()

        print('run time           = {} sec'.format(int(t4 - t1)))
        print('training time      = {} sec'.format(int(t3 - t2)))
        print('pre-processing time = {} sec'.format(int(t2 - t1)))

        y_true = y_test.idxmax(
           axis='columns'
        ).reset_index().drop('index', axis=1).rename(columns={'0':'ProtocolName'})

        y_pred = pd.DataFrame(pd.DataFrame(model.predict(X_test),
```

```
                                columns = y_test.columns)
                        .idxmax(axis='columns'), columns=['ProtocolName'])

conf_mat = pd.DataFrame(confusion_matrix(y_true, y_pred,
                                         labels = label.columns),
                        columns = label.columns,
                        index = label.columns)

plt.figure(figsize=(24,20))

sns.heatmap(conf_mat,
            cmap = 'gray',
            linecolor = 'white',
            linewidths = 0.01,
            annot=True)

plt.title("confusion matrix", {'fontsize':35})
plt.xlabel('y_pred', {'fontsize':20})
plt.ylabel('y_true', {'fontsize':20})
plt.savefig('confusion matrix.png')

print(classification_report(y_true, y_pred))
```

На рис. 14 и рис. 15 можно увидеть, с каких IP-адресов чаще всего отправлялись пакеты, и какие IP-адреса чаще всего получали пакеты. Это может использоваться, например, для анализа вредоносного трафика в сети, или

для оптимизации сети.

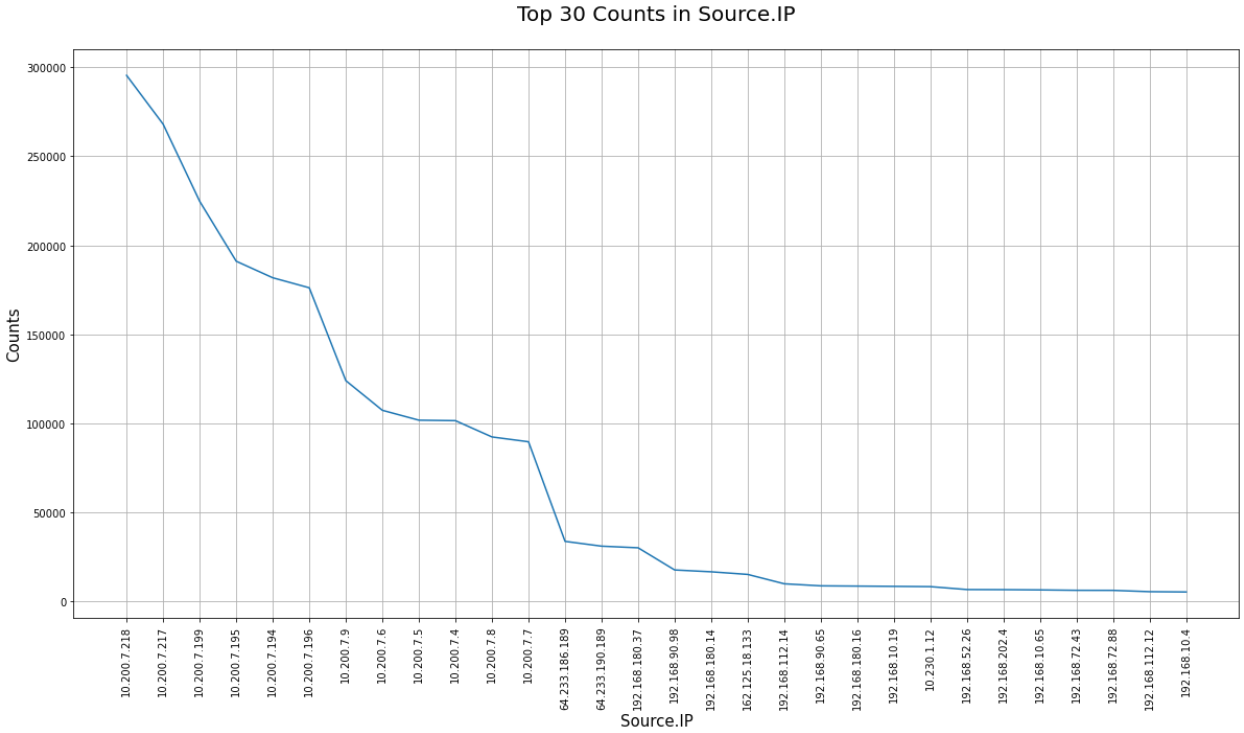

Рис. 14 – Какие IP-адреса чаще всего были указаны как Source.IP в сетевых пакетах.

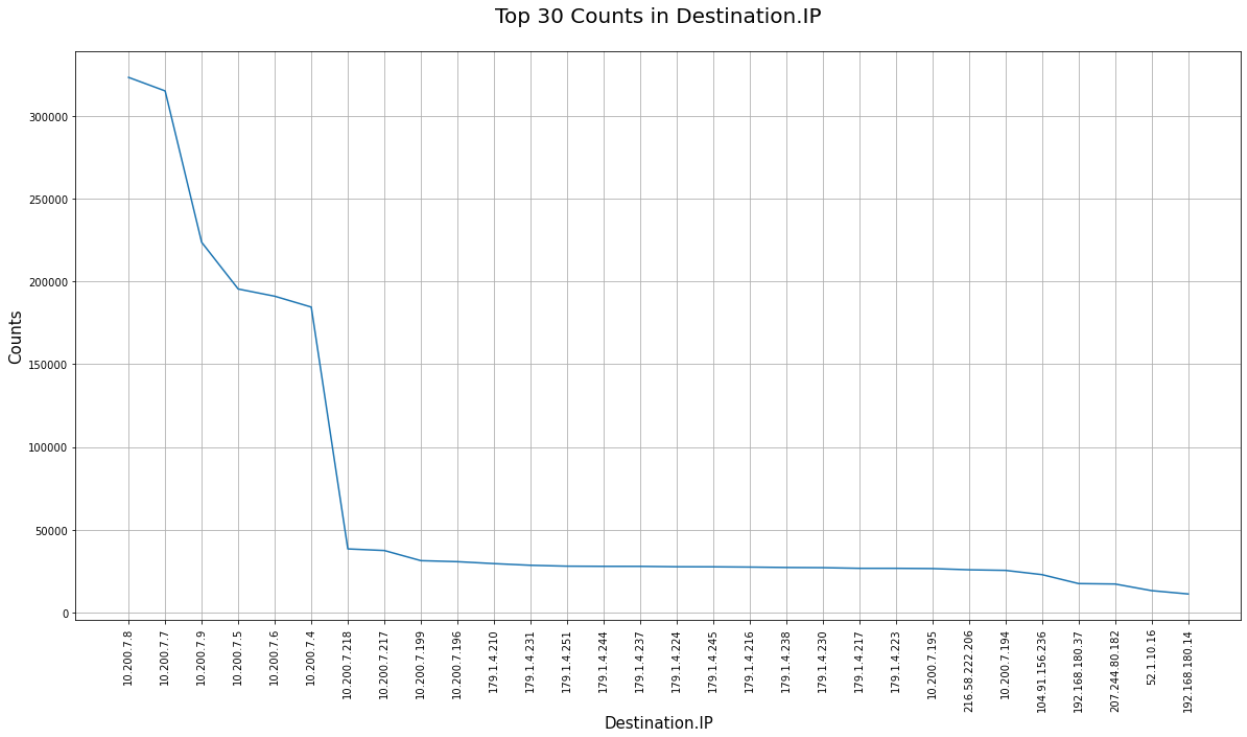

Рис. 15 – Какие IP-адреса чаще всего были указаны как Destination.IP в сетевых пакетах.

На рис. 16 мы можем увидеть, как увеличивались или уменьшались метрики нейронной сети с каждой новой эпохой.

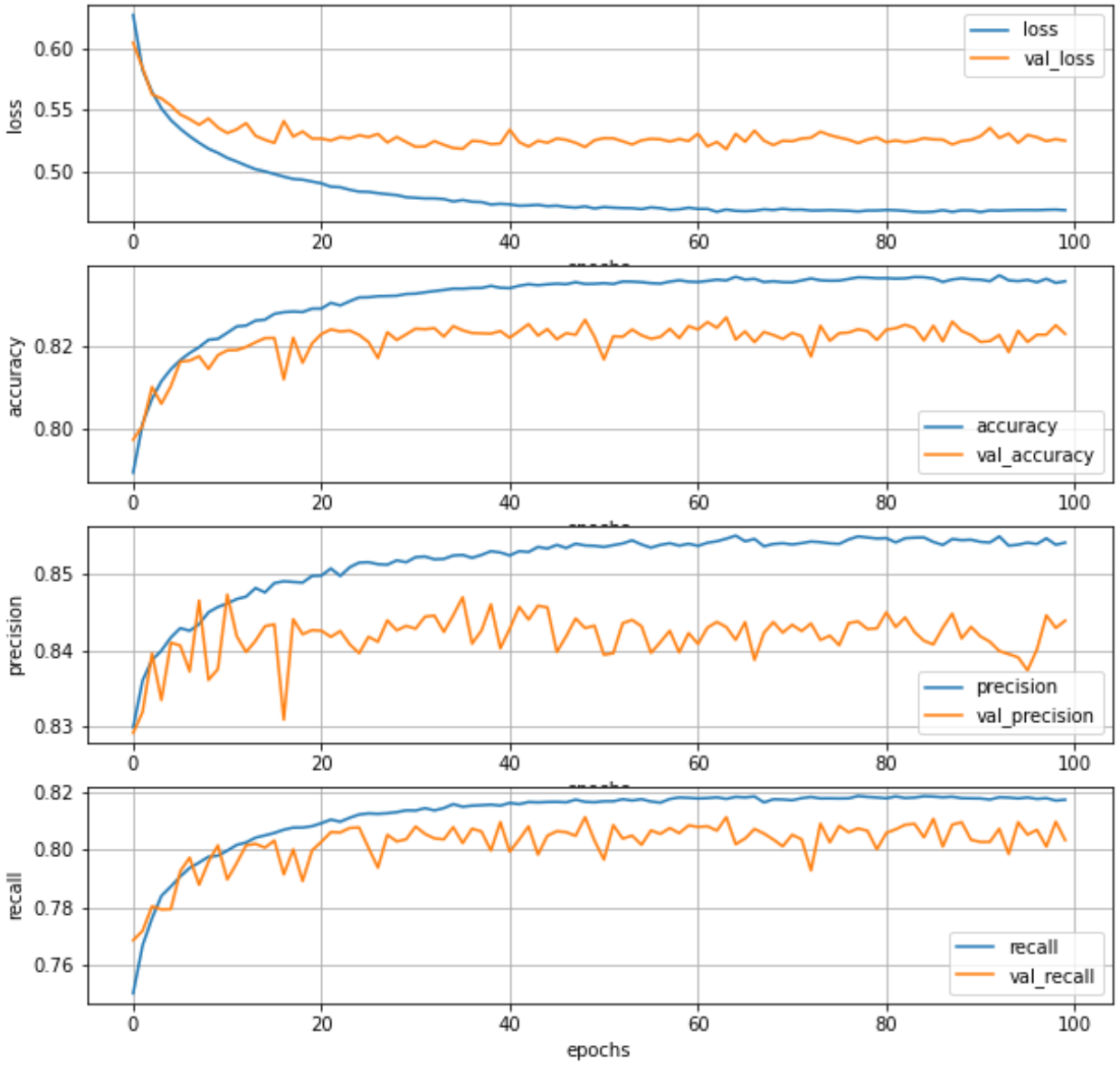

Рис. 16 – Изменение метрик нейронной сети на протяжении всех эпох.

Также в таблице 3 можно увидеть отчёт об общей точности реализованной нейронной сети.

	precision	recall	f1-score	support
accuracy			0.83	404480
macro avg	0.40	0.27	0.30	404480

weighted avg	0.81	0.83	0.81	404480

Вывод

Таким образом, в разделе 3 были реализованы две модели машинного обучения для классификации трафика. Для их реализации был выбран язык Python, который лучше всего подходит для задач машинного обучения, так как включает в себя множество библиотек. Также были приведены и обоснованы результаты работ моделей, благодаря которым уже можно делать выводы о том, какие ресурсы чаще всего используют пользователи сети, а также оптимизировать сеть в соответствии с собранными данными.

Список литературы

[1]. Sniffer. https://www.opennet.ru/base/sec/arp_snif.txt.html.

[2]. Wireshark. https://www.wireshark.org/.

[3]. NAT. https://tools.ietf.org/html/rfc1918.

[4]. Software NAT. http://www.nat32.com/v2/.

[5]. DDoS. http://ddos-protection.ru/chto-takoe-ddos.

[6]. IANA Service Name and Transport Protocol Port Nu mber Registry. http://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml.

[7]. Д. Виньяр. Deep Packet inspection: Technology and products. Workshop «DPI: architecture and experience архитектура и опыт», «Peter-Service », 3.12.2013.

[8]. SIP. https://www.ietf.org/rfc/rfc3261.txt.

[9]. RTP. https://www.ietf.org/rfc/rfc3550.txt.

[10]. Session Border Controller. http://www.voip-info.org/wiki/view/Session+Border+Controller.

[11]. Wan Optimizations. http://searchenterprisewan.techtarget.com/definition/WAN-optimization.

[12]. CIFS. https://msdn.microsoft.com/en-us/library/ee442092.aspx.

[13]. MiddleBoxes. https://tools.ietf.org/html/rfc3234.

[14]. DPI. http://nag.ru/articles/article/22432/dpi.html.

[15]. Ekonomika programmnykh i apparatnykh DPI na primere Cisco SCE i SKAT http://nag.ru/articles/article/28436/ekonomika-programmnyih-i-apparatnyih-dpi-na- primere-cisco-sce-i-skat.html.

[16]. Platformy glubokogo analiza trafika i upravleniya trafikom prilozhenii. http://www.inlinetelecom.ru/solutions/access_network/platform_depth_analysis_of_t raff ic_and_traffic_control_applications/.

[17]. David L. Cannon. CISA Certified Information Systems Auditor Study Guide, 2nd Edition, 2008, ISBN: 978-0-470-23152-4

[18]. ICAP. https://tools.ietf.org/html/rfc3507.

[19]. Sergei Medvedev. Deep Packet Inspection (DPI). Seminar «Zhivye vstrechi», Krasnoyarsk, 18.01.2014.

[20]. Recommendation ITU-T Y.2770, Requirements for deep packet inspection in next generation networks, edition 1.0, 2012.11.20

[21]. Recommendation ITU-T Y.2771, Framework for deep packet inspection, 2014.07.01

[22]. QUIC. https://tools.ietf.org/html/draft-tsvwg-quic-protocol-00.

[23]. CAIDA Flow Types. https://www.caida.org/research/traffic-analysis/flowtypes/.

[24]. P. Filimonov, M. Ivanov. Sovremennye podkh ody k klassifikatsii trafika fizicheskikh kanalov seti Internet, Trudy 18-oi Me zhdunarodnoi konferentsii «Raspredelennye komp'yuternye i kommunikatsionnye seti: upravlenie, vychislenie, svyaz'» (DCCN-2015), 19 - 22 oktyabrya 2015 g, str. 466-474.

[25]. F. Risso, M. Baldi, O. Morandi, A. Baldini, P. Monclus, “Lightweight, payload-based traffic classification: An experimental evaluation” in Proc. IEEE ICC, 2008, pp. 5869– 5875.

[26]. Wireshark, VoIP calls. https://wiki.wireshark.org/VoIP_calls.

[27]. MIME. https://tools.ietf.org/html/rfc2045.

[28]. ASN.1. https://tools.ietf.org/html/rfc6025.

[29]. P. Bosshart, D. Daly, G. Gibb, M. Izzard, N. McKeown, J. Rexford, C. Schlesinger, D. Talayco, A. Vahdat, G. Varghese, D. Wa lker. P4: Programming protocol-independent packet processors. SIGCOMM Computer Communications Review, 2013.

[30]. A. Bremler-Barr, Y. Harchol, D. Hay, Y. Koral. Deep packet inspection as a service. In CoNEXT, pages 271–282, 2014.

[31]. SDN. https://tools.ietf.org/html/rfc7426.

[32]. Qosmos ixEngine. http://www.qosmos.com/products/deep-packet-inspection-engine/.

[33]. Ipoque PACE. https://www.ipoque.com/products/pace.

[34]. Windriver Content Inspection Engine. http://www.windriver.com/products/product- overviews/PO_Wind-River-Content-Inspection-Engine.pdf.

[35]. Procera PacketLogic Conten t Intelligence. https://www.proceranetworks.com/content- intelligence.html.

[36]. Cisco NBAR. http://www.cisco.com/c/en /us/products/ios-nx-os-software/network- based-application-recognition-nbar/index.html.

[37]. Junos OS Next Generation Application Identification. https://www.juniper.net/documentation/en_US/junos15.1x49/topics/concept/services-application-identification-techniques-understanding.html.

[38]. Cascarano N, Ciminiera L, Risso F. Optimizing deep packet inspection for high-speed traffic analysis. Network System Manager. 2011; 19(1):7–31.

[39]. Duffield N., Lund C. “Predicting Resource Usage and Estimation Accuracy in an IP Flow Measurement Collection Infrastructure”. ACM Internet Measurement Conference, 2003.

[40]. Callado A., Kamienski C., Szabo G., Gero B., Kelner J., Fernandes S., Sadok D. A Survey on Internet Traffic Identification; Communications Surveys & Tutorials, IEEE Volume 11, Issue 3, 3rd Quarter 2009 Page(s):37 – 52.

[41]. Duffield, N.; Lund, C.; Thorup, M., “Learn more, sample less: control of volume and variance in network measurement”, IEEE Transactions in Information Theory, vol. 51, no. 5, pp. 1756-1775, 2005.

[42]. PSAMP. http://www.rfc-editor.org/rfc/rfc5476.txt.

[43]. Se-Hee Han, Myung-Sup Kim, Hong-Taek Ju and James W. Hong, “The Architecture of NGMON: a Passive Network Monitoring System for High-Speed IP Networks”, Accepted to appear in the Proc. of the 13th IFIP /IEEE International Workshop on Distributed Systems:Operations & Mana gement (DSOM 2002), Montre al, Canada, October 21-23, 2002.

[44]. InveaTech FlowMon. https://www.invea.com/en/products/flowmon.

[45]. IPFIX. https://tools.ietf.org/html/rfc5101.

[46]. M.-S. Kim, Y. J. Won, and J. W. Hong. Characteristic analysis of internet traffic from the perspective of flows. Comp. Comm., 29(10):1639–1652, 2006.

[47]. R. Sommer and A. Feldmann. NetFlow: Information loss or win? In ACM SIGCOMM Internet Meas. Workshop, 2002.

[48]. MIB. https://tools.ietf.org/html/rfc3418.

[49]. SNMP. https://www.ietf.org/rfc/rfc1157.txt.

[50]. Colin J. Bennett, Andrew Cl ement, Kate Milberry. Introduction to Cyber-Surveillance. Cyber-Surveillance in Everyday Life. Vol. 9, No 4 (2012)

[51]. T. Farah, and L. Trajkovic, "Anonym: A tool for anonymization of the Internet traffic." In IEEE 2013 International Conference on Cybernetics (CYBCONF), 2013, pp. 261-266.

[52]. F. Risso, and L. Degioanni, “An Architecture for High Performance Network Analysis”, Proceedings of the 6th IEEE Symposium on Computers and Communications (ISCC 2001), Hammamet, Tunisia, July 2001.

[53]. PF_RING ZC. http://www.ntop.org/products/packet-capture/pf_ring/.

[54]. Andrew M White, Srinivas Krishnan, Michael Bailey, Fabian Monrose, and Phillip Porras. Clear and Present Data: Opaque Traffic and its Security Implications for the Future. NDSS, 2013.

[55]. Network Intrusion Detection Signatures. http://www.symantec.com/connect/articles/network-intrusion-detection-signatures-part- five.

[56]. Yuji Waizumi, Yuya Tsukabe, Hiroshi Tsunoda, and Yoshiaki Nemoto. Network Application Identification Based on Comm unication Characteristics of Application Messages. International Jour nal of Electrical, Computer, Energetic, Electronic and Communication Engineering Vol:3, No:12, 2009

[57]. T. Karagiannis, K. Papagiannaki and M. Faloutsos. BLINC: Multilevel traffic classification in the dark. In Proc. of ACM SIGCOMM, August 2005

[58]. Nguyen, T. T. T. and Armitage, G.2008. A su rvey of techniques for internet traffic classification using machine learning. IEEE Communications Surveys &Tutorials 10, 4, (2008), 56-76.

[59]. T. AbuHmed, A. Mohaisen, and D. Nyang, “A survey on deep packet inspection for intrusion detection systems,” CoRR, vol. abs/0803.0037, 2008. [O nline]. Available: http://arxiv.org/abs/0803.0037.

[60]. Koloud Al-Khamaiseh, Shadi ALShagarin. A Survey of String Matching Algorithms. Int. Journal of Engineering Research and Applications. ISSN : 2248-9622, Vol. 4, Issue 7(Version 2), July 2014, pp.144-156.

[61]. D. E. Taylor, “Survey and taxonomy of packet classification techniques,” ACM Comput. Surv., vol. 37, no. 3, pp. 238–275, 2005.

[62]. L7-filter. http://l7-filter.sourceforge.net/.

[63]. J. E. Hopcroft and J. D. Ullman, “Introduction to Automata Theory, Languages, and Computation,” Addison Wesley, 1979.

[64]. S. Kumar and P. Crowley. Algorithms to Accelerate Multiple Regular Expressions Matching for Deep Packet Inspection. In Proc. of SIGCOMM, 2006.

[65]. D. Ficara, S. Giordano, G. Procissi. An Improved DFA for Fast Regular Expression Matching. In Proc. of SIGCOMM, 2008.

[66]. F. Yu, Z. Chen, Y. Diao, T. V. Lakshman, and R. H. Katz. Fast and Memory-Efficient Regular Expression Matching for Deep Packet Inspection. In Proc. of ANCS, 2006.

[67]. S. Kumar, B. Chandrasekaran, J. Turner, and G. Varghese. Curing Regular Expressions Matching Algorithms From Insomnia. In Proc. of ANCS, 2007.

[68]. R. Smith, C. Estan, S. Jha, and S. Kong. Deflating the Big Bang: Fast and Scalable Deep Packet Inspection with Extended Finite Automata. In Proc. of ACM SIGCOMM, 2008.

[69]. C. Liu, J. Wu. Fast Deep Packet Inspection with a Dual Finite Automata. IEEE Transactions on Computers, Vol. 62.

[70]. Zip Bomb. http://xeushack.com/zip-bomb/