[1. Сеть Ethernet. Принципы работы. Концентраторы и коммутаторы. 2](#_Toc453156305)

[2. Топологии соединения коммутаторов: остовное дерево (Spanning Tree), дублирующие линии (Resilient Link, LinkSafe), объединение портов (Port Trunking) 3](#_Toc453156306)

[3. Виртуальные локальные сети (VLAN) 4](#_Toc453156307)

[4. IP-адресация IPv4 (классовая и бесклассовая адресация, подсети, маска сети) 5](#_Toc453156308)

[5. IP-адресация IPv6 6](#_Toc453156309)

[6. Автоконфигурирование IP-адресов и обнаружение соседей в IPv6 7](#_Toc453156310)

[7. Групповое вещание (протокол IGMP) 8](#_Toc453156311)

[8. Служба доменных имен DNS 9](#_Toc453156312)

[9. Протокол DHCP 10](#_Toc453156313)

[10. Протокол ICMP 12](#_Toc453156314)

[11. Протоколы ARP и RARP 13](#_Toc453156315)

[12. Протокол UDP 14](#_Toc453156316)

[13. Протокол TCP. Рукопожатие. Подтверждение передачи и повторная передача. Управление потоком и контроль перегрузки. 15](#_Toc453156317)

[14. Трансляция адресов (NAT) 17](#_Toc453156318)

[15. Назначение и принципы работы протокола STUN 18](#_Toc453156319)

[16. Алгоритмы маршрутизации. Централизованные и децентрализованные. Внутренние и внешние протоколы маршрутизации 19](#_Toc453156320)

[17. Алгоритм маршрутизации RIP 21](#_Toc453156321)

[18. Алгоритм маршрутизации OSPF 22](#_Toc453156322)

[19. Алгоритм маршрутизации BGP 23](#_Toc453156323)

[20. Классификация удаленных атак на распределенные вычислительные системы 24](#_Toc453156324)

[21. Удаленные атаки на распределенные вычислительные системы 26](#_Toc453156325)

[22. Методы атаки узлов вычислительной системы 28](#_Toc453156326)

[23. Межсетевые экраны (брандмауэры) и их виды. 29](#_Toc453156327)

[24. Системы обнаружения и предотвращения вторжений (IDS и IPS). 30](#_Toc453156328)

[25. Защита информации на IP уровне. Протокол IPSec 31](#_Toc453156329)

[26. Защита WEB информации. Протокол SSL (TLS) 33](#_Toc453156330)

[27. Сертификаты X.509 34](#_Toc453156331)

[28. Протокол SET 35](#_Toc453156332)

[29. Защита электронной почты (PGP) 36](#_Toc453156333)

[30. Защита электронной почты (S/MIME) 37](#_Toc453156334)

[31. Архитектура AAA систем. 38](#_Toc453156335)

[32. Протокол RADIUS 39](#_Toc453156336)

# Сеть Ethernet. Принципы работы. Концентраторы и коммутаторы.

**Ethernet** - семейство технологий [пакетной](https://ru.wikipedia.org/wiki/%D0%9F%D0%B0%D0%BA%D0%B5%D1%82_(%D1%81%D0%B5%D1%82%D0%B5%D0%B2%D1%8B%D0%B5_%D1%82%D0%B5%D1%85%D0%BD%D0%BE%D0%BB%D0%BE%D0%B3%D0%B8%D0%B8)) передачи данных для [компьютерных сетей](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D0%B0%D1%8F_%D1%81%D0%B5%D1%82%D1%8C).

Название «Ethernet» (буквально «эфирная сеть» или «среда сети») отражает первоначальный принцип работы этой технологии: всё, передаваемое одним узлом, одновременно принимается всеми остальными (то есть имеется некое сходство с [радиовещанием](https://ru.wikipedia.org/wiki/%D0%A0%D0%B0%D0%B4%D0%B8%D0%BE%D0%B2%D0%B5%D1%89%D0%B0%D0%BD%D0%B8%D0%B5)). В настоящее время практически всегда подключение происходит через [коммутаторы (switch)](https://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D1%82%D0%B5%D0%B2%D0%BE%D0%B9_%D0%BA%D0%BE%D0%BC%D0%BC%D1%83%D1%82%D0%B0%D1%82%D0%BE%D1%80), так что кадры, отправляемые одним узлом, доходят лишь до адресата (исключение составляют передачи на [широковещательный адрес](https://ru.wikipedia.org/wiki/%D0%A8%D0%B8%D1%80%D0%BE%D0%BA%D0%BE%D0%B2%D0%B5%D1%89%D0%B0%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9_%D0%B0%D0%B4%D1%80%D0%B5%D1%81)) — это повышает скорость работы и безопасность сети.

Промышленный протокол Ethernet (EtherNet/IP) — открытый промышленный сетевой стандарт, который поддерживает неявный обмен сообщениями (обмен сообщениями ввода/вывода в реальном времени), явный обмен (обмен сообщениями) или оба и использует широко распространённые коммерческие чипы связи Ethernet и физические носители. Поскольку технология Ethernet используется с середины 1970-ых и широко принята во всём мире, то продукты Ethernet поддерживает большое количество поставщиков. Используя продукты Ethernet, вы не только следуете за общим направлением современной технологии, — у вас есть возможность иметь доступ ко всем устройствам уровня данных из Internet. EtherNet/IP возник из-за высокого спроса на использование сети Ethernet для приложений управления.

**Основные принципы работы сети Ethernet**  
Ethernet – технология локальных сетей, преимущественно функционирующих в одном здании и связывающих близко расположенные устройства. Чаще всего устройства сети Ethernet соединялись кабелем длиной не более нескольких сотен метров, а объединение в сеть распределенных по большой территории объектов было экономически невыгодно. Благодаря современным техническим достижениям, удалось существенно расширить допустимые расстояния между объектами, поэтому современные сети Ethernet могут охватывать территории в десятки километров.   
**Протоколы**  
В сетях под термином "протокол" подразумевается набор правил, регламентирующих обмен информацией. Протоколы для компьютеров – то же, что язык для людей. Поскольку эта статься излагается на русском языке, читатель, чтобы понять написаное, должен уметь читать по-русски. Аналогично, два устройства в сети смогут успешно обмениваться информацией лишь в том случае, если они оба понимают одинаковые протоколы.   
**Терминология сетей Ethernet**  
Основные операции сети Ethernet подчиняются простому набору правил. Чтобы лучше понять эти правила, важно разобраться в основной терминологии Ethernet.

**Канал передачи.** – Устройства сети Ethernet подключаются к общему каналу передачи, по которому передаются электрические сигналы. Исторически сложилось, что каналом передачи раньше был медный коаксиальный кабель, однако в наше время для этих целей чаще используется витая пара или волоконно-оптический кабель.

**Сегмент**. – Сегментом сети Ethernet называют один совместно используемый канал передачи.

**Узел.** – Узлами называются устройства, подключаемые к сегменту.

**Кадр** (или фрейм) – Узлы обмениваются короткими информационными сообщениями, которые называют кадрами. Кадр – порция информации, размер которой может меняться.

Кадры можно сравнить по функциональному назначению с предложениями человеческой речи. В русском языке есть правила, по которым строятся предложения: в каждом предложении должно быть подлежащее и сказуемое. В протоколе Ethernet предусмотрен набор правил, регламентирующих формирование кадров. Для кадра правилами прямо устанавливается максимальная и минимальная длина, а также указывается, какая в него должна вводиться обязательная информация. Например, в каждом кадре должны содержаться адрес назначения и адрес источника данных, по которым можно идентифицировать отправителя и получателя. Подобно тому, как имя соотносится с определенным человеком, адрес однозначно соответствует определенному узлу. Один адрес Ethernet не может принадлежать одновременно нескольким устройствам.   
**Канал передачи Ethernet**

Поскольку сигнал с канала передачи поступает на каждый подключенный узел, для нахождения получателя кадра очень важна роль адреса назначения.

Например, если к сети подключены несколько компьютеров и принтер, при передаче информации от одного из компьютеров к принтеру остальные компьютеры также получают и анализируют кадры данных. Получив кадр, станция сначала проверяет адрес назначения, чтобы определить, для нее ли этот кадр предназначен. При отрицательном результате проверки станция отказывается от приема этого кадра, даже не исследуя его содержимое.

Интересной особенностью системы адресации Ethernet является возможность использования широковещательного адреса. Кадр, в котором в качестве адреса назначения указан широковещательный адрес, предназначается для каждого узла сети и кадры такого типа обрабатываются всеми узлами сети.

**Концентраторы**

Концентраторы позволяют компьютерам в сети обмениваться данными. Компьютер подключается к концентратору с помощью кабеля Ethernet, и информация, передаваемая с одного компьютера на другой, проходит через концентратор. Концентратор не может определить источник или место назначения полученных данных, поэтому пересылает их всем подключенным к нему компьютерам, включая и тот, с которого была отправлена информация. Концентратор может либо передавать, либо получать данные, но не может делать и то и другое одновременно. Поэтому концентраторы работают медленнее, чем коммутаторы. Концентраторы являются наименее сложными и наименее дорогими устройствами из рассматриваемых в этом разделе.

**Коммутаторы**

Коммутаторы работают таким же образом, как и концентраторы, но при этом могут определить место назначения полученных данных, поэтому передают их только тем компьютерам, которым эти данные предназначаются. Коммутаторы могут получать и передавать данные одновременно, поэтому они работают быстрее концентраторов. Если к сети подключено четыре и более компьютеров или предполагается передача больших объемов данных (например, игра в сетевые компьютерные игры или прослушивание музыки через сеть), предпочтительно использование коммутатора. Коммутаторы стоят немного дороже концентраторов.

# Топологии соединения коммутаторов: остовное дерево (Spanning Tree), дублирующие линии (Resilient Link, LinkSafe), объединение портов (Port Trunking)

**Spanning Tree Protocol** (**STP**, протокол [остовного дерева](https://ru.wikipedia.org/wiki/%D0%9E%D1%81%D1%82%D0%BE%D0%B2%D0%BD%D0%BE%D0%B5_%D0%B4%D0%B5%D1%80%D0%B5%D0%B2%D0%BE)) — канальный протокол. Основной задачей STP является устранение петель в топологии произвольной сети [Ethernet](https://ru.wikipedia.org/wiki/Ethernet), в которой есть один или более [сетевых мостов](https://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D1%82%D0%B5%D0%B2%D0%BE%D0%B9_%D0%BC%D0%BE%D1%81%D1%82), связанных избыточными соединениями. STP решает эту задачу, автоматически блокируя соединения, которые в данный момент для полной связности коммутаторов являются избыточными.

Необходимость устранения топологических петель в сети Ethernet следует из того, что их наличие в реальной сети Ethernet с [коммутатором](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BC%D1%83%D1%82%D0%B0%D1%82%D0%BE%D1%80_Ethernet) с высокой вероятностью приводит к бесконечным повторам передачи одних и тех же кадров Ethernet одним и более коммутатором, отчего пропускная способность сети оказывается почти полностью занятой этими бесполезными повторами; в этих условиях, хотя формально сеть может продолжать работать, на практике её производительность становится настолько низкой, что может выглядеть как полный отказ сети.

STP относится ко второму уровню [модели](https://ru.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C_OSI) [OSI](https://ru.wikipedia.org/wiki/Open_Systems_Interconnection). Протокол описан в стандарте *IEEE 802.1d.* STP основан на одноимённом [алгоритме](https://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC), который разработала [Радья Перлман](https://ru.wikipedia.org/w/index.php?title=%D0%A0%D0%B0%D0%B4%D1%8C%D1%8F_%D0%9F%D0%B5%D1%80%D0%BB%D0%BC%D0%B0%D0%BD&action=edit&redlink=1) ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) [*Radia Perlman*](https://en.wikipedia.org/wiki/Radia_Perlman)).

Если в сети с мостовыми подключениями (в сегменте сети из коммутаторов) имеется несколько путей, могут образоваться циклические маршруты, и следование простым правилам пересылки данных через мост (коммутатор) приведёт к тому, что один и тот же пакет будет бесконечно передаваться с одного моста на другой (передаваться по кольцу из коммутаторов).

Алгоритм остовного дерева позволяет по мере необходимости автоматически отключать передачу через мост в отдельных портах (блокировать порты коммутатора), чтобы предотвратить зацикливание в топологии маршрутов пересылки пакетов. Для использования алгоритма остовного дерева в сетевом мосте никакой дополнительной настройки не требуется.

**Алгоритм остовного** дерева является основой протокола STP (Spanning Tree Protocol), динамически отключающего избыточные связи в сети стандарта Ethernet (для образования древовидной топологии). STP стандартизован [IEEE](https://ru.wikipedia.org/wiki/IEEE) и поддерживается многими моделями управляемых коммутаторов, в частности, включен по умолчанию на всех коммутаторах [Cisco](https://ru.wikipedia.org/wiki/Cisco).

Суть работы протокола заключается в том, что поддерживающие его коммутаторы сети Ethernet обмениваются друг с другом информацией «о себе». На основании определённых условий (обычно в соответствии с настройками) один из коммутаторов выбирается «корневым» (или «root»), после чего все остальные коммутаторы по алгоритму остовного дерева выбирают для работы порты, «ближайшие» к «корневому» коммутатору (учитывается количество посредников и скорость линий). Все прочие сетевые порты, ведущие к «корневому» коммутатору, блокируются. Таким образом образуется несвязное дерево с корнем в выбранном коммутаторе.

В коммутаторах [Cisco](https://ru.wikipedia.org/wiki/Cisco) с поддержкой VLAN протокол STP по умолчанию выполняется независимо для каждой виртуальной сети.

Кроме STP, в коммутаторах могут применяться другие методики обнаружения и устранения петель — например, сравнением таблиц коммутации (списков MAC-адресов) разных портов, или сравнением контрольных сумм проходящих пакетов (совпадение указывает на одинаковые пакеты, которые появляются из-за петель). По сравнению с описанными методами, случайным образом (или основываясь на каких-то догадках) блокирующими «дублирующие» порты, протокол STP обеспечивает древовидную структуру всего сегмента, при любом количестве резервных линий между произвольными коммутаторами, поддерживающими STP.

Выбирается один корневой мост ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *Root Bridge*).

Далее каждый коммутатор, отличный от корневого, просчитывает кратчайший путь к корневому. Соответствующий порт называется корневым портом ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *Root Port*). У любого некорневого коммутатора может быть только один корневой порт.

После этого для каждого сегмента сети, к которому присоединён более чем один мост (или несколько портов одного моста), просчитывается кратчайший путь к корневому порту. Мост, через который проходит этот путь, становится **назначенным** для этой сети ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *Designated Bridge*), а соответствующий порт — **назначенным портом** ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *Designated port*).

Далее во всех сегментах, с которыми соединены более одного порта моста, все мосты блокируют все порты, не являющиеся корневыми и назначенными. В итоге получается древовидная структура (математический [граф](https://ru.wikipedia.org/wiki/%D0%93%D1%80%D0%B0%D1%84_(%D0%BC%D0%B0%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0))) с вершиной в виде корневого коммутатора.

**Resilient Link.** Идея заключается в прокладке между двумя коммутаторами не одной, а двух линий. Пара физических интерфейсов коммутатора, сконфигурированных на дублирование, рассматривается как один порт. Таким образом должны быть сконфигурированы интерфейсы коммутаторов на обоих концах резервированной линии. В нормальном режиме передача данных происходит только по основной линии, резервная простаивает. В случае обнаружения отказа основной линии ее интерфейс отключается и обмен продолжается по резервной линии. Поскольку обнаружение отказа происходит на физическом уровне (а не по таймауту ожидания кадров BPDU), реакция и восстановление связи происходит быстро (соизмеримо с временем передачи одного кадра). Недостатком этой технологии является топологическая ограниченность: основная и резервная линии связывают одни и те же коммутаторы. Конечно, для страховки физический резервный кабель можно проложить по иной трассе, но это не спасает от отказа коммутатора (даже одного его порта). Резервная линия при исправной основной простаивает, как и при STP. Технологии не являются промышленными стандартами — совместимость аппаратуры разных производителей не гарантируется.

**Агрегирование каналов** (port trunking) — технология, которая позволяет объединить несколько физических каналов в один логический. Такое объединение позволяет увеличивать пропускную способность и надежность канала. Агрегирование каналов может быть настроено между двумя коммутаторами, коммутатором и маршрутизатором, между коммутатором и хостом.

Агрегирование каналов позволяет решить две задачи:

1. повысить пропускную способность канала
2. обеспечить резерв на случай выхода из строя одного из каналов

Большинство технологий по агрегированию позволяют объединять только параллельные каналы. То есть такие, которые начинаются на одном и том же устройстве и заканчиваются на другом.

Если рассматривать избыточные соединения между коммутаторами, то без использования специальных технологий для агрегирования каналов, передаваться данные будут только через один интерфейс, который не заблокирован STP. Такой вариант позволяет обеспечить резервирование каналов, но не дает возможности увеличить пропускную способность.

Технологии по агрегированию каналов позволяют использовать все интерфейсы одновременно. При этом устройства контролируют распространение широковещательных фреймов (а также multicast и unknown unicast), чтобы они не зацикливались. Для этого коммутатор, при получении широковещательного фрейма через обычный интерфейс, отправляет его в агрегированный канал только через один интерфейс. А при получении широковещательного фрейма из агрегированного канала, не отправляет его назад.

То есть, реальная загруженность конкретного интерфейса никак не учитывается. Поэтому один интерфейс может быть загружен больше, чем другие. Более того, при неправильном выборе метода балансировки (или если недоступны другие методы) или в некоторых топологиях, может сложиться ситуация, когда реально все данные будут передаваться, например, через один интерфейс.

# Виртуальные локальные сети (VLAN)

Изначально коммутаторы не обеспечивали возможности создания Виртуальных Локальных сетей, так как они использовались для простой пересылки фреймов между устройствами. Рынок коммутаторов начал быстро расти, когда концентраторы коллективного доступа к среде передачи данных (hubs) начали не справляться с растущими запросами на расширение полосы пропускания сети в связи с использованием приложений клиент-сервер, обеспечивающих Графический Интерфейс Пользователя (GUI).

По мере улучшения технологий и захвата рынка новомодной технологией, начали возникать VLAN. Простейший путь понять Виртуальные сети - сравнить их с физической сетью. Физическая сеть может состоять из конечных станций, связанных маршрутизатором (или маршрутизаторами), которые используют одно физическое соединение. VLAN это логическое комбинирование конечных станций в одном сегменте на Уровне 2 и Уровне 3, которые связаны напрямую, без маршрутизатора. Обычно пользователям, разделенным физически, требуется маршрутизатор для связи с другим сегментом. Коммутаторы с возможностью построения VLAN изначально были внедрены в основных учебных городках и небольших рабочих группах. Сначала коммутация разрабатывалась по мере надобности, но сейчас это является обычной практикой внедрять коммутаторы и VLAN в настольных системах.

Каждая рабочая станция в VLAN (и только эти конечные станции) обрабатывают широковещательный трафик, посылаемый другим членам VLAN. Например, рабочие станции A, B, и C присоединены в VLAN 1. VLAN 1 состоит из трех коммутаторов Catalyst 5500. Все коммутаторы расположены на разных этажах и соединены между собой опто-волокном и связаны транковым протоколом. Рабочая станция A присоединена с коммутатором A, рабочая станция B присоединена в коммутатор B и рабочая станция C присоединена в коммутатор C. Если станция A посылает широковещательный пакет, станции B и C получат этот фрейм, даже если они физически присоединены в другие коммутаторы. Рабочая станция D присоединена в коммутатор A, но объявлена в VLAN 2. Когда D посылает широковещательный пакет, станция A не увидит этот трафик, хотя она находится в том же физическом коммутаторе, но так как она находится не в той же виртуальной LAN, коммутатор не будет пересылать этот трафик на A. Помните, что VLAN работают на Уровне 2, поэтому связь между VLAN требует принятия решений маршрутизации на Уровне 3. Так же станции B и C не увидят трафик от станции D.

Виртуальные сети (VLAN) предлагают следующие преимущества:

1. Контроль за широковещательным трафиком
2. Функциональные рабочие группы
3. Повышенная безопасность

**Контроль за широковещательным трафиком**

В отличие от традиционных LAN, построенных при помощи маршрутизаторов/мостов, VLAN может быть рассмотрен как широковещательный домен с логически настроенными границами. VLAN предлагает больше свободы, чем традиционные сети. Ранее используемые разработки были основаны на физическом ограничении сетей, построенных на основе концентраторов; в основном физические границы LAN сегмента ограничивались эффективной дальностью, на которую электрический сигнал мог пройти от порта концентратора. Расширение LAN сегментов за эти границы требовало использования повторителей (repeaters), устройств, которые усиливали и пересылали сигнал. VLAN позволяет иметь широковещательный домен вне зависимости от физического размещения, среды сетевого доступа, типа носителя и скорости передачи. Члены могут располагаться там, где необходимо, а не там, где есть специальное соединение с конкретным сегментом. VLAN увеличивают производительность сети, помещая широковещательный трафик внутри маленьких и легко управляемых логических доменов. В традиционных сетях с коммутаторами, которые не поддерживают VLAN, весь широковещательный трафик попадает во все порты. Если используется VLAN, весь широковещательный трафик ограничивается отдельным широковещательным доменом.

**Функциональные рабочие группы**

Наиболее фундаментальным преимуществом технологии VLAN является возможность создания рабочих групп, основываясь на функциональности, а не на физическом расположении или типе носителя. Традиционно администраторы группировали пользователей функционального подразделения физическим перемещением пользователей, их столов и серверов в общее рабочее пространство, например в один сегмент. Все пользователи рабочей группы имели одинаковое физическое соединение для того, чтобы иметь преимущество высокоскоростного соединения с сервером. VLAN позволяет администратору создавать, группировать и перегруппировывать сетевые сегменты логически и немедленно, без изменения физической инфраструктуры и отсоединения пользователей и серверов. Возможность легкого добавления, перемещения и изменения пользователей сети - ключевое преимущество VLAN.

**Повышенная Безопасность**

VLAN также предлагает дополнительные преимущества для безопасности. Пользователи одной рабочей группы не могут получить доступ к данным другой группы, потому что каждая VLAN это закрытая, логически объявленная группа. Представьте компанию, в которой Финансовый департамент, который работает с конфиденциальной информацией, расположен на трех этажах здания. Инженерный департамент и отдел Маркетинга также расположены на трех этажах. Используя VLAN, члены Инженерного отдела и отдела Маркетинга могут быть расположены на всех трех этажах как члены двух других VLAN, а Финансовый департамент может быть членом третьей VLAN, которая расположена на всех трех этажах. Сейчас сетевой трафик, создаваемый Финансовым департаментом, будет доступен только сотрудникам этого департамента, а группы Инженерного и отдела Маркетинга не смогут получить доступ к конфиденциальным данным Финансового департамента. Очевидно, есть другие требования для обеспечения полной безопасности, но VLAN может быть частью общей стратегии сетевой безопасности. Показанный ниже рисунок говорит о том, как функционирование VLAN может расширить традиционные границы.

# IP-адресация IPv4 (классовая и бесклассовая адресация, подсети, маска сети)

Адресация в компьютерных сетях бывает двух видов: физическая адресация (на основе MAC-адреса) и логическая (на основе IP-адреса). Логическая адресация реализована на 3-ем уровне [эталонной модели OSI](http://itandlife.ru/technology/computer-networks/setevaya-model-osi-open-system-interconnection/). Далее более подробно рассматривается IP-адресация и пять классов IP-адресов, а также подсети, маски подсетей и их роль в схемах IP-адресации. Кроме того, обсуждаются отличия между публичными и частными адресами, IPv4-и IPv6-адресацией, а также одноадресными и широковещательными сообщениями.

Для обмена данными в Интернете (между различными локальными сетями) узлу необходим IP-адрес. Это логический сетевой адрес конкретного узла. Для обмена данными с другими устройствами, подключенными к Интернету, необходим правильно настроенный, уникальный IP-адрес.

IP-адрес присваивается сетевому интерфейсу узла. Обычно это [сетевая интерфейсная плата (NIC)](http://itandlife.ru/technology/computer-networks/setevye-ustrojstva-tipy-setevyx-ustrojstv-i-ix-funkcii/), установленная в устройстве. Примерами пользовательских устройств с сетевыми интерфейсами могут служить рабочие станции, серверы, сетевые принтеры и IP-телефоны. Иногда в серверах устанавливают несколько NIC, у каждой из которых есть свой IP-адрес. У интерфейсов [маршрутизатора](http://itandlife.ru/technology/computer-networks/setevye-ustrojstva-tipy-setevyx-ustrojstv-i-ix-funkcii/), обеспечивающего связь с сетью IP, также есть IP-адрес.

В каждом отправленном по сети пакете есть IP-адрес источника и назначения. Эта информация необходима сетевым устройствам для передачи информации по назначению и передачи источнику ответа.

IP-адрес представляет собой серию из 32 двоичных бит (единиц и нулей). Человеку прочесть двоичный IP-адрес очень сложно. Поэтому 32 бита группируются по четыре 8-битных байта, в так называемые октеты. Читать, записывать и запоминать IP-адреса в таком формате людям сложно. Чтобы облегчить понимание, каждый октет IP-адреса представлен в виде своего десятичного значения. Октеты разделяются десятичной точкой или запятой. Это называется точечно-десятичной нотацией.

При настройке IP-адрес узла вводится в виде десятичного числа с точками, например, 192.168.1.5. Вообразите, что вам пришлось бы вводить 32-битный двоичный эквивалент адреса — 11000000101010000000000100000101. Если ошибиться хотя бы в одном бите, получится другой адрес, и узел, возможно, не сможет работать в сети.

Структура 32-битного IP-адреса определяется межсетевым протоколом 4-ой версии (IPv4). На данный момент это один из самых распространенных в Интернете типов IP-адресов. По 32-битной схеме адресации можно создать более **4 миллиардов IP-адресов**.

Получая IP-адрес, узел просматривает все 32 бита по мере поступления на сетевой адаптер. Напротив, людям приходится преобразовывать эти 32 бита в десятичные эквиваленты, то есть в четыре октета. Каждый октет состоит из 8 бит, каждый бит имеет значение. У четырех групп из 8 бит есть один и тот же набор значений. Значение крайнего правого бита в октете – 1, значения остальных, слева направо – 2, 4, 8, 16, 32, 64 и 128.

Чтобы определить значение октета, нужно сложить значения позиций, где присутствует двоичная единица.

Нулевые позиции в сложении не участвуют.

Если все 8 бит имеют значение 0, 00000000, то значение октета равно 0.

Если все 8 бит имеют значение 1, 11111111, значение октета – 255 (128+64+32+16+8+4+2+1).

Если значения 8 бит отличаются, например, 00100111, значение октета – 39 (32+4+2+1).

Таким образом, значение каждого из четырех октетов находится в диапазоне от 0 до 255. Логический 32-битный IP-адрес представляет собой иерархическую систему и состоит из двух частей. Первая идентифицирует сеть, вторая — узел в сети. Обе части являются обязательными.

При IP-адресации в одной физической сети могут существовать несколько логических сетей, если сетевая часть адреса их узла отличается. Пример. Три узла в одной физической локальной сети имеют одинаковую сетевую часть в своем IP-адресе (192.168.50), а три других узла — другую сетевую часть (192.168.70). Три узла с одной сетевой частью в своих IP-адресах имеют возможность обмениваться данными друг с другом, но не могут обмениваться информацией с другими узлами без использования маршрутизации. В данном случае имеем одну физическую сеть и две логические IP-сети.

IP-адрес и маска подсети совместно определяют то, какая часть IP-адреса является сетевой, а какая — соответствует адресу узла. IP-адреса делятся на 5 классов. К классам A, B и C относятся коммерческие адреса, присваиваемые узлам. Класс D зарезервирован для многоадресных рассылок, а класс E – для экспериментов. В адресах класса C сетевая часть состоит из трех октетов, а адрес узла – из одного. Выбранная по умолчанию маска подсети состоит из 24 бит (255.255.255.0). Адреса класса C обычно присваиваются небольшим сетям. В адресах класса B сетевая часть и адрес узла состоят из двух октетов. Выбранная по умолчанию маска подсети состоит из 16 бит (255.255.0.0). Обычно эти адреса используются в сетях среднего размера. В адресах класса A сетевая часть состоит всего из одного октета, остальные отведены узлам. Выбранная по умолчанию маска подсети состоит из 8 бит (255.0.0.0). Обычно такие адреса присваиваются крупным организациям. Класс адреса можно определить по значению первого октета. Например, если значение первого октета IP-адреса находится в диапазоне от 192 до 223, то это адрес класса C. Например, адрес 200.14.193.67 относится к классу С.

**Классовая IP адресация** — это метод IP-адресации, который не позволяет рационально использовать ограниченный ресурс уникальных IP-адресов, т.к. не возможно использование различных масок подсетей. В классовом методе адресации используется фиксированная маска подсети, поэтому класс сети (см. выше) всегда можно идентифицировать по первым битам.

**Бесклассовая IP адресация (**Classless Inter-Domain Routing — CIDR) — это метод IP-адресации, который позволяет рационально управлять пространством IP адресов. В бесклассовом методе адресации используются маски подсети переменной длины (variable length subnet mask— VLSM).

Возможные значения масок подсети при бесклассовом методе адресации (широко применяется в современных сетях):

Каждый IP-адрес состоит из двух частей. Как узлы определяют, где сетевая часть, а где адрес узла? Для этого используется маска подсети. При настройке IP узлу присваивается не только IP-адрес, но и маска подсети. Как и IP-адрес, маска состоит из 32 бит. Она определяет, какая часть IP-адреса относится к сети, а какая – к узлу. Маска сравнивается с IP-адресом побитно, слева направо. В маске подсети единицы соответствуют сетевой части, а нули — адресу узла. Отправляя пакет, узел сравнивает маску подсети со своим IP-адресом и адресом назначения. Если биты сетевой части совпадают, значит, узлы источника и назначения находятся в одной и той же сети, и пакет доставляется локально. Если нет, отправляющий узел передает пакет на интерфейс локального маршрутизатора для отправки в другую сеть.

Чтобы вычислить количество возможных сетевых узлов, нужно взять количество отведенных для них бит в степени 2 (2 ^ 8 = 256). Из полученного результата необходимо вычесть 2 (256-2). Дело в том, что состоящая из одних единиц (1) отведенная узлам часть IP-адреса предназначена для адреса широковещательной рассылки и не может принадлежать одному узлу. Часть, состоящая только из нулей, является идентификатором сети и тоже не может быть присвоена конкретному узлу.

# IP-адресация IPv6

Маски теперь задаются только /prefix'ами ([CIDR](https://en.wikipedia.org/wiki/Classless_Inter-Domain_Routing)), классовой адресации и стандартной decimal dotted нотации в IPv6 нет. Так же теперь первый и последний адрес сети не являются зарезервированными под идентификатор сети и broadcast соответственно.

Как и в случае с IPv4, IPv6'ые адреса выделяются [через целую иерархию организаций](https://en.wikipedia.org/wiki/IPv6_address#General_allocation):  
[RIR](https://en.wikipedia.org/wiki/Regional_Internet_Registry)(/12-/23) -> [NIR](https://en.wikipedia.org/wiki/National_Internet_Registry)(??) -> [LIR](https://en.wikipedia.org/wiki/Local_Internet_registry)(/19-/32) -> ISP(/48-/56) -> LAN(=>/64)  
Также интересная деталь: Таблицы маршрутизации в IPv6 планировали изначально делать как можно более «агригабельными», что подразумевало отсутствие [Provider-Independent (PI)](https://secure.wikimedia.org/wikipedia/en/wiki/Provider-independent_address_space) адресов. Однако крупные корпорации (видимо желающие делать BGP-failover) в 2009 году всё-таки [«продавили» RIPE](https://www.ripe.net/ripe/docs/ripe-510#2e), так что получить PI-адреса в IPv6 теперь реально.

**Типы адресов и их префиксы**

Об IPv6 адресации подробно рассказано в [RFC4291](http://tools.ietf.org/html/rfc4291). Далее я опишу самые распространённые типы IPv6 адресов, их назначения и префиксы.

::/128 - Unspecified — не должен принадлежать ни одной ноде в сети;

::/0 - Default route;

::1/128 - Loopback;

fe80::/10 - Link-Local — адреса уникальные на линке.

fc00::/7 - Unique Site-Local — пришли на замену Site-Local в [RFC4193](http://tools.ietf.org/html/rfc4193). В данный момент разбит на две части: fc00::/8 и fd00::/8. [Уникальные в пределах организации](https://en.wikipedia.org/wiki/Unique_local_address), не роутящиеся в интернет адреса. Однако могут роутиться внутри site'a и между site'ами;

ff00::/8 - Multicast — о мультикасте подробнее расскажу чуть ниже. Полный список мультикаст адресов можно посмотреть тут:[IANA IPv6 Multicast Addresses](http://www.iana.org/assignments/ipv6-multicast-addresses/);

::0000/96 - IPv4-Compatible IPv6 Address — [устарели](https://en.wikipedia.org/wiki/IPv6_address#Historical_notes);

::ffff/96 - IPv4-Mapped IPv6 Address — Адреса предназначенные в основногм для Socket API. Более подробно их назначение описано в [RFC4038](http://tools.ietf.org/html/rfc4038). Прочтение этого RFC будет полезно программистам которые собираются писать Dual-Stack приложения;

**Все «не специальные» адреса относятся к:**Global Unicast — глобальные адреса, уникальные во всем интернете. Пока, впрочем, [судя по этому документу](http://www.iana.org/assignments/ipv6-address-space/), доступно только глобальное пространство 2000::/3 (Адреса от 2000:: до 3fff:ffff:ffff:ffff:ffff:ffff:ffff:ffff)

**Виды трафика**

**Unicast**— Старый добрый юникаст;

**Multicast**— Мультикаст теперь необходимое расширение, а не опциональное как в IPv4. IGMP был заменён на MLD ([Multicast Listener Discovery](https://en.wikipedia.org/wiki/Multicast_Listener_Discovery)). А процедура получения глобального мультикаст префикса стала тривиальной — теперь при получении /64 префикса провайдер автоматически получает 4.2 миллиарда глобальных мультикаст групп. Процедура подробно описана в[RFC3306](http://tools.ietf.org/html/rfc3306), а также дополнена внедрением адреса RP прямо в IPv6 адрес в [RFC3956](http://tools.ietf.org/html/rfc3956). Получение глобального мультикаст префикса для IPv4 и IPv6 описано в [RFC6308](https://tools.ietf.org/html/rfc6308). Стоит также заметить, что в IPv4 Multicast link-layer префикс был 01:00:5e, в IPv6 он стал 33:33:ff (посмотреть список групп для интерфейсов во FreeBSD можно через ifmcstat или же через ip maddr в Linux);

**Anycast**— Такой же anycast как и в IPv4. Этот тип адреса обычно анонсируется протоколом динамической маршрутизации (например BGP), сразу из нескольких мест. Это обеспечивает оптимальный, с точки зрения протокола маршрутизации, роутинг;

**Broadcast**— в IPv6 broadcast'а не существует. В место него можно использовать [All Nodes Address](http://www.iana.org/assignments/ipv6-multicast-addresses/ipv6-multicast-addresses.xml). Пакеты посланные на него будут рассылаться только на хосты **с настроенным** IPv6 адресом (при включеном MLD snooping'е); Также часть протоколов раньше использовавших Broadcast в IPv6-версии всёже обзавелись собственной multicast группой;

В IPv6 появилось такое понятие как [Scope](https://en.wikipedia.org/wiki/IPv6_address#IPv6_address_scopes), он же Zone ID терминологии Microsoft. На самом деле оно было и в IPv4, однако не было задано явно: сети 10/8, 172.16/12 и 192.168/16 яркие тому примеры.  
В случае Unicast/Anycast адресов приминимо следующее:

У каждого IPv6 enabled интерфейса есть свой Link-local адрес. Его scope, внезапно, local. Эти адреса уникальны в пределах линка, но не обязаны быть актуальными в пределах одного хоста. Так, например, VLAN созданный на интерфейсе будет иметь такой же link-local адрес, что и родительский интерфейс (так как без использования IPv6 Privacy Extensions он будет генериться из тогоже Link Layer адреса). Для того, чтобы явно указать интерфейс которому принадлежит IPv6 адрес нужно или указывать в ручную интерфейс для исходящих пакетов или использовать специальный суффикс при записи адреса: %ИндексИнтерфейса в Windows (fe80::2b0:d0ff:fee9:4143%3) или %ИмяИнтерфейса в \*BSD/Linux (fe80::2b0:d0ff:fee9:4143%em0).

В случае Multicast адресов scope указан в последних четырёх битах вторго октета IPv6 адреса: ff0s:: и может быть interface-local, link-local, admin-local, site-local, organization-local или же global.

Жизненный цикл IPv6 адреса

1. Tentative — Адрес ещё проверяется на уникальность;
2. Valid — Траффик на этот адрес будет получатся хостом, делится на 2 подсостояния:
3. Prefered state — Основное состояние, неограниченное использование адреса;
4. Deprecated state — Адрес ещё можно использовать для старых соединений, но нельзя создавать новые соединения;
5. Invalid — Адрес нельзя использовать для получения/отправки пакетов.

Отсутствует Checksum заголовка, соответственно, его не надо проверять, а также пересчитывать для каждого пакета при изменении ~~TTL~~ Hop Limit. Так как checksum больше нет, то вся ответственность за целостность информации должна лежать на протоколе более низкого уровня, так например у Ethernet фреймов есть свой честный CRC32. Так же у UDP пакетов наличие checksum теперь обязательно и UDP/IPv6 пакеты с Checksum 0000 будут просто отбрасываться принимающим хостом;

Сам TTL теперь именуется Hop Limit (скорее всего потому, что раньше одним из условий у роутера было уменьшать TTL на один каждую [секунду](http://tools.ietf.org/html/rfc791#page-14) прибывания пакета в очереди, поэтому и TIME-to-live). В связи с последним трендом с повсеместным введением MPLS/TE стоит заметить, что при прохождении IP-пакета через MPLS облако его TTL/HopLimit может не и меняется;

Роутеры теперь не занимаются фрагментацией пакетов. Хосты должны сами проводить [Path MTU discovery](http://tools.ietf.org/html/rfc1981) и разбивать пакеты. Минимальный MTU теперь равен 1280.

Были добавлены [Метки потоков](http://ru.wikipedia.org/wiki/IPv6#.D0.9C.D0.B5.D1.82.D0.BA.D0.B8_.D0.BF.D0.BE.D1.82.D0.BE.D0.BA.D0.BE.D0.B2), служат для разгрузки роутеров, более точной приоритезации трафика и балансировки. Более подробно можно почитать в RFC6437 — [IPv6 Flow Label Specification](https://tools.ietf.org/html/rfc6437). До сих пор ходят баталии об использовании этого поля IPv6 заголовка на практике. Эпик-треды можно почитать в RFC6294 — [Survey of Proposed Use Cases for the IPv6 Flow Label](https://tools.ietf.org/html/rfc6294), RFC6436 —[Rationale for Update to the IPv6 Flow Label Specification](https://tools.ietf.org/html/rfc6436) и RFC6438 — [Using the IPv6 Flow Label for Equal Cost Multipath Routing and Link Aggregation in Tunnels](https://tools.ietf.org/html/rfc6438). Единственная операционная система(из протестированных нами) которая на выставляет по умолчению flow label'ы это FreeBSD;

Изначально(в obsoleted [RFC1883](https://tools.ietf.org/html/rfc1883)) поле Traffic Class называлось Priority и занимало 4 бита, а flow label был 24 бита. В [RFC2460](https://tools.ietf.org/html/rfc2460) они стали 8 и 20 бит соответственно. Если кому интересна никрофилия можно почитать остальные [Historical notes](https://en.wikipedia.org/wiki/IPv6_address#Historical_notes) на вики;

Поддержка IPSec теперь является обязательной;

# Автоконфигурирование IP-адресов и обнаружение соседей в IPv6

Протокол из набора [Internet Protocol Suite](https://ru.wikipedia.org/w/index.php?title=Internet_Protocol_Suite&action=edit&redlink=1), используемый совместно с [IPv6](https://ru.wikipedia.org/wiki/IPv6). Он работает на [уровне слоя Интернет](https://ru.wikipedia.org/w/index.php?title=Internet_Layer&action=edit&redlink=1) Модели Интернета ([RFC 1122](https://tools.ietf.org/html/rfc1122)) и ответственен за автонастройку адреса конечных точек сети, обнаружения других узлов на линии, обнаружения адреса других узлов на уровне канала связи, обнаружение конфликта адресов, поиск доступных путей и [DNS](https://ru.wikipedia.org/wiki/DNS)-серверов, обнаружения подсетей и поддержки доступности информации о путях к другим активным соседним узлам ([RFC 4861](https://tools.ietf.org/html/rfc4861)).

Этот протокол устанавливает пять различных типов пакета [ICMPv6](https://ru.wikipedia.org/wiki/ICMPv6) для выполнения функций IPv6 сходных с [ARP](https://ru.wikipedia.org/wiki/ARP), [ICMP](https://ru.wikipedia.org/wiki/ICMP), [Router Discovery](https://ru.wikipedia.org/w/index.php?title=ICMP_Router_Discovery_Protocol&action=edit&redlink=1) и [Router Redirect](https://ru.wikipedia.org/w/index.php?title=ICMP_Redirect_Message&action=edit&redlink=1) протоколов для [IPv4](https://ru.wikipedia.org/wiki/IPv4). Тем не менее, он обеспечивает множество улучшений через взаимозаменяющиеся части IPv4 ([RFC 4861](https://tools.ietf.org/html/rfc4861), секция 3.1). Например, он включает [NUD](https://ru.wikipedia.org/w/index.php?title=NUD&action=edit&redlink=1), который повышает надежность доставки пакетов в присутствии проблемных маршрутизаторов или подключений, или непостоянных узлов.

NDP устанавливает следующие пять типов пакета [ICMPv6](https://ru.wikipedia.org/wiki/ICMPv6):

1. Запрос на доступность маршрутизаторов
2. Ответ маршрутизатора
3. Запрос доступных соседей
4. Ответ соседа
5. Перенаправление

Эти сообщения используются для обеспечения следующей функциональности:

1. Обнаружение маршрутизатора: узел может разместить маршрутизатор, находящийся на подключенной линии.
2. Обнаружение подсети: узлы могут обнаруживать работающие подсети для подключенных линий.
3. Обнаружение параметров: узлы могут запрашивать параметры линии (например, размер [MTU](https://ru.wikipedia.org/wiki/Maximum_transmission_unit)).
4. Автоматическая настройка адреса: конфигурирование адресов сетевых интерфейсов.
5. Разрешение адреса: работа между IP-адресом и адресами уровня канала связи.
6. Обнаружение следующего перехода: узлы могут находить следующий на пути пакета маршрутизатор.
7. Обнаружение недоступности соседа(NUD): определение того, что сосед более недоступен на линии.
8. Обнаружение конфликта адресов(DAD): узлы сами могут определять, занят ли адрес.
9. Перенаправление: маршрутизатор может информировать узел о других наилучших маршрутизаторах для начала пути пакета.

Рекурсивный DNS-сервер(RDNSS) и список поиска DNS(DNSSL) назначается через параметры отклика маршрутизатора(RA).  Это новая функция, и поддерживается не всем программным обеспечением.

# Групповое вещание (протокол IGMP)

Протокол группового управления в Интернете (Internet Group Management Protocol, IGMP) был разработан в 1989 году для обеспечения более эффективной рассылки информации по IP-адресам, чем традиционные методы одноадресной и широковещательной передачи. Существует три версии IGMP: IGMPvl (RFC 1112), IGMPv2 (RFC 2236) hIGMPv3(RFC 3376).

Протокол IGMP используется исключительно при взаимодействии непосредственно связанных друг с другом маршрутизатора и хоста, когда последний выступает (или желает выступать) в роли получателя трафика группового вещания.

**ПРИМЕЧАНИЕ**

*Источник не нуждается в протоколе IGMP. Любой компьютер, подключенный к Интернету, может стать источником группового вещания, при этом ему не требуется никакого дополнительного программного обеспечения, кроме того, которое включено в состав обычной реализации стека TCP/IP.*

К основным функциям протокола IGMP относятся оповещение маршрутизатора о желании хоста быть включенным в группу и опрос членов группы.

Оповещение маршрутизатора о желании хоста быть включенным в группу. Чтобы стать получателем групповых данных, узел должен «выразить» свою заинтересованность маршрутизатору, к которому непосредственно подсоединена его сеть. Для этого хост должен установить взаимодействие с маршрутизатором по протоколу IGMP. Версия IGMP для хоста прямо зависит от типа операционной системы, установленной на хосте. Так, ранние версии Windows (Windows 95) поддерживали только версию IGMPvl, более поздние (Windows 2000) — версию IGMPv2, а начиная с Windows ХР, поддерживается версия IGMPv3.

Протоколы IGMPv2 и IGMPv3 поддерживаются во многих версиях Mac OS, Linux, Unix-подобных операционных системах.

Опрос членов группы. Для выполнения этой функции один из маршрутизаторов локальной сети выбирается доминирующим. Доминирующий маршрутизатор средствами протокола IGMP периодически опрашивает все системы (групповой адрес 224.0.0.1) в непосредственно присоединенных к нему подсетях, проверяя, активны ли члены всех известных ему групп. Остальные (не выбранные) маршрутизаторы прослушивают сеть, и если обнаруживают отсутствие сообщений-запросов в течение некоторого периода (обычно 250 секунд), то повторяют процедуру выбора нового доминирующего маршрутизатора.

**В IGMPv2 определено три типа сообщений:**

1. Запрос о членстве (membership query). С помощью этого сообщения маршрутизатор пытается узнать, в каких группах состоят хосты в локальной сети, присоединенной к какому-либо его интерфейсу. Запрос о членстве существует в двух вариантах: в одном из них маршрутизатор делает общий запрос обо всех группах, в другом его интересует информация только о некоторой конкретной группе, адрес которой указывается в запросе.
2. Отчет о членстве (membership report). Этим сообщением хосты отвечают маршрутизатору, который послал в сеть запрос о членстве. В сообщении содержится информация об адресе группы, в которой они состоят. Маршрутизатор, являясь членом всех групп, получает сообщения, направленные на любой групповой адрес. Для маршрутизатора, получающего ответные сообщения, важен только факт наличия членов той или иной группы (групп), а не принадлежность конкретных хостов конкретным группам. Этот факт будет использован другими маршрутизаторами сети для продвижения пакетов группового вещания в ту часть сети, за которую «отвечает» данный маршрутизатор. Отчет о членстве хост может послать не только в ответ на запрос маршрутизатора, но и по собственной инициативе, когда он пытается присоединиться к определенной группе. После такого сообщения хост может рассчитывать на то, что трафик для этой группы действительно будет доставляться в сеть, к которой этот хост принадлежит.
3. Покинуть группу (leave group). Это сообщение хост может использовать, чтобы сигнализировать «своему» маршрутизатору о желании покинуть некоторую группу, в которой он до этого состоял. Получив это сообщение, маршрутизатор посылает специфический запрос о членстве членам только этой конкретной группы, и если не получает на него ответ (то есть это был последний хост в группе), то перестает передавать трафик группового вещания для этой группы. Слово «может» означает в данном случае, что хост может быть исключен из группы, просто не отвечая маршрутизатору на запрос о членстве (такой подход реализован в протоколе IGMPvl). Тогда маршрутизатор будет продолжать передавать нежелательный трафик группового вещания до тех пор, пока не истечет некоторый период времени с момента поступления последнего отчета о членстве. Такой подход может значительно удлинить период скрытого нахождения хоста в состоянии выхода из группы, что снижает эффективность работы сети.

Сообщения с запросами о членстве посылаются маршрутизатором регулярно с некоторой частотой. На каждом из интерфейсов с установленными средствами IGMP маршрутизаторами поддерживаются кэш-таблицы групп. Кэш-таблица содержит список всех групп, в составе которых есть хотя бы один член. Для каждой строки таблицы установлен тайм-аут. Маршрутизатор регулярно посылает запросы (по умолчанию — каждые 125 секунд), чтобы проверить, что в каждой группе еще имеются члены. Если для некоторой группы ответ не поступает в течение установленного для нее тайм-аута, то соответствующая строка удаляется из кэш-таблицы, и маршрутизатор считает, что членов этой группы в сети больше нет.

Локальная сеть может иметь несколько хостов, заинтересованных в получении трафика одной и той же группы, но маршрутизатору достаточно подтверждения только от одного хоста для того, чтобы продолжить передавать трафик в сеть для этой группы. При использовании протокола IGMPvl или IGMPv2 для ограничения числа ответов хостов на запрос маршрутизатора любой хост, состоящий в группе, вместо того чтобы немедленно ответить на запрос, сначала ждет в течение некоторого интервала времени, не появится ли в сети ответ какого-нибудь другого хоста. Если по истечении этого времени он так и не смог дождаться появления в сети ответа другого хоста, то он посылает маршрутизатору собственный отчет о членстве. (Если же используется протокол IGMPv3, то никаких пауз не устанавливается, и хосты сразу генерируют сообщения о членстве.)

Основываясь на информации, полученной с помощью IGMP, маршрутизаторы могут определять, в какие подключенные к ним сети необходимо передавать групповой трафик.

Все типы IGMP-сообщений имеют длину 8 байт и состоят из четырех полей. В зависимости от версии протокола IGMP назначение полей может несколько меняться.

Поле максимального времени ответа используется хостами для вычисления времени задержки ответа. Время задержки выбирается случайным образом из интервала от нуля до значения, заданного в этом поле.

Заметим, что поле адреса группового вещания в IGMP-сообщении не содержит адрес назначения, оно несет в себе информацию, по-разному используемую в разных типах сообщений. Например, маршрутизатор, посылая запрос о членстве, помещает в этом поле нули, а хост в сообщениях «Отчет о членстве» и «Покинуть группу» помещает в это поле адрес группы, в которую он хочет вступить или которую он хочет покинуть соответственно.

# Служба доменных имен DNS

Служба Доменных Имен предназначена для того, чтобы машины, работающие в Internet, могли по доменному имени узнать IP-адрес нужной им машины, а также некоторую другую информацию; а по IP-номеру могли узнать доменное имя машины.

Служба Доменных Имен была разработана для именования машин в глобальной сети. Основной особенностью глобальной сети является распределенное администрирование, когда один администратор физически не может уследить за выделением имен. Поэтому Служба Доменных Имен функционирует на принципе **делегирования полномочий**. Каждая машина либо знает ответ на вопрос, либо знает кого спросить. При правильном функционировании система замкнута, т.е. если запрошенная информация имеется у кого-либо, то она будет найдена и сообщена клиенту, либо, если вопрос не имеет ответа, клиент получит сообщение о невозможности получения ответа на вопрос.

Каждый клиент знает своего сервера; обычно указывается не один, а несколько серверов - если первый не отвечает, клиент обращается ко второму и так далее до исчерпания списка. В принципе неважно, к какому серверу обращаться - они дают (должны давать при правильном функционировании) одинаковые ответы на любой запрос. Поэтому для ускорения работы обычно указывают ближайший. Следует помнить, что на одной машине могут функционировать одновременно Name-сервер и программы-клиенты; поэтому если на машине запущен Name-сервер, то в качестве Name-сервера на ней должен быть прописан "я сам".

Имеется некий домен верхнего уровня, обозначаемый точкой: "**.**". Имеется девять серверов (по крайней мере на моем Name-сервере записано столько), которые отвечают за эту зону. Они не знают ни одного доменного имени - они только авторизуют серверы верхних зон. Серверы верхних зон тоже гнушаются хранить информацию о конкретных машинах и передают это право нижележащим серверам. Тут уже появляются первые упоминания о конкретных машинах, равно как и происходит авторизация нижележащих серверов.

Мне неизвестна ни одна машина с доменным именем из одного сегмента; очень редко используются доменные имена из двух сегментов; имена из трех и четырех сегментов составляют подавляющую долю всех имен Internet; имена из пяти сегментов встречаются довольно редко, а из шести и более мне неизвестны.

Допустим, клиент запросил адрес "www.организация.город.страна". Поиск информации по доменному имени происходит следующим образом:

1. Клиент спрашивает своего сервера (Если тот является сервером данной зоны, то ответит, на чем все заканчивается).
2. Сервер спрашивает корневой сервер.
3. Тот не может ответить, потому что не знает; зато знает, какой сервер отвечают за зону "страна".
4. Сервер зоны "страна" тоже не может ответить, но знает, что нужно спросить сервер зоны "город.страна".
5. Тот в свою очередь отсылает запрос серверу зоны "организация.город.страна", который сообщит нужную информацию.
6. Это приближенная модель, которая тем не менее позволяет представить работу системы DNS.

Однако эту стройную картину искажают системы кэширования и вторичных серверов. Дело в том, что получив ответ на свой вопрос, DNS-сервер получает также некоторое число, которое говорит ему о том, по истечении какого времени эта информация должна считаться устаревшей. Таким образом, все серверы, участвовавшие в поиске ответа на вопрос, заданный клиентом, могут (и скорее всего будут) помнить как ответ на заданный вопрос, так и путь, по которому шел поиск. При следующих запросах, имеющих общую правую часть с недавно сделанными запросами, поиск будет упрощен (ускорен).

Кроме того, большинство зон имеет вторичные серверы, которые содержат копии данных с первичных серверов. Сервер вышележащей зоны может направить запрос как первичному серверу, так и любому из вторичных, основываясь на своих соображениях о том, какой из них ближе.

Хочу обратить особое внимание на сходство, различие и взаимодействие систем DNS и IP-маршрутизации. Как и IP-маршрутизация, DNS работает по принципу делегирования полномочий, но выделение доменных имен совершенно не зависит от выделения IP-адресов. Для примера рассмотрим домен freebsd.org. Это - домен организации, занимающейся распространением операционной системы FreeBSD Unix. FTP-сервер, содержащий дистрибутив операционной системы и множества утилит для нее, имеет копии в нескольких десятках стран. Имена серверов выглядят так:

- ftp.freebsd.org - первичный сервер в США

- ftp.*страна*.freebsd.org - основной сервер в *стране*

- ftp*число*.*страна*.freebsd.org - дополнительный сервер в *стране*

Таким образом, машины, находящиеся в России оказались произвольно (по воле DNS-мастера из университета Bercley) включенными в домен freebsd.org; однако, они также состоят в своих зонах. Система DNS позволяет любому DNS-мастеру включить любой сервер в свою зону, хотя это включение никого ни к чему не обязывает.

Делегирование зоны ...in-addr.arpa дается только от провайдера вместе с IP-адресами. Собственно, это связано с предназначением ReverceDNS - сообщать доменное имя по IP-адресу. Наверняка мастер зоны freebsd.org держит Reverce-зону для IP-номеров, выделенных университету Bercley; но все эти серверы (кроме сервера, расположенного в университете) не входят в эту Reverce-зону, а значит, ему неподконтрольны.

Одна из проблем состит в том, что Reverce-зону можно выделить только на сеть класса A, B или C (на 16777216, 65536 или 256 адресов) и никак иначе. Можно получить права на несколько зон одного или разных классов, но что делать тем, кому выделили меньше 256 адресов? А ведь в условиях исчерпания адресного пространства не редкость выделения пула уже на 16 адресов!

DNS-услуги Internet-провайдера

Как правило, провайдер предоставляет клиенту целый комплекс услуг. В число оказываемых DNS-услуг входят:

делегирование зоны ...in-addr.arpa клиентам, имеющим пул адресов, кратный 256.

регистрация доменного имени клиента у держателя той зоны, в которой клиент хочет зарегистрироваться;

поддержание вторичного сервера прямой и обратной DNS-зон клиента;

поддержание первичного сервера этих зон, если клиент по какой-либо причине не поддерживает их сам (особенно это относится к случаю виртуальных зон и к случаю выделения малого пула адресов);

Имена зон условно можно разделить на "*организационные*" и "*географические*". В высшей зоне зарегестрированы следующие "*организационные*" зоны:

В данный момент, чтобы разгрузить домен com, собираются создать несколько новых доменов, но у меня нет достоверной информации по ним. В организационных зонах обычно размещаются непосредственно домены организаций.

Каждая страна (государство) имеет свой географический домен из двух букв

# Протокол DHCP

Наиболее полное современное описание DHCP содержится в документе RFC 2131 (март 1997 г.), который пришел на смену более ранним редакциям RFC 1531 и 1541. В настоящее время DHCP имеет статус предварительного стандарта.

DHCP появился не на пустом месте - различные схемы управления IP-адресами в сетевой среде предлагались и раньше. Однако эти схемы имеют по крайней мере один из двух недостатков - не допускают динамического назначения IP-адресов либо позволяют передавать от сервера на станцию-клиент лишь небольшое число параметров конфигурации.

При разработке протокола DHCP преследовалась цель устранить оба ограничения. Требовался механизм, который позволил бы ликвидировать стадию ручного конфигурирования компьютеров, поддерживал многосегментные сети, не требуя наличия DHCP-сервера в каждой подсети, не конфликтовал с существующими сетевыми протоколами и компьютерами, имеющими статичную конфигурацию, был способен взаимодействовать с ретранслирующими агентами протокола BOOTP и обслуживать BOOTP-клиентов, наконец, допускал управление передаваемыми параметрами конфигурации. Что касается более узких задач, то DHCP должен был обеспечивать уникальность сетевых адресов, используемых разными компьютерами сети в данный момент, сохранение прежней конфигурации клиентской станции после перезагрузки клиента или сервера, автоматическое присвоение параметров конфигурации вновь подключенным машинам.

Работа протокола DHCP базируется на классической схеме клиент-сервер. В роли клиентов выступают компьютеры сети, стремящиеся получить IP-адреса в так называемую аренду (lease), а DHCP-серверы выполняют функции диспетчеров, которые выдают адреса, контролируют их использование и сообщают клиентам требуемые параметры конфигурации. Сервер поддерживает пул свободных адресов и, кроме того, ведет собственную регистрационную базу данных. Взаимодействие DHCP-серверов со станциями-клиентами осуществляется путем обмена сообщениями.

Упомянутое выше требование поддержки базовых элементов протокола BOOTP возникло не случайно. DHCP разрабатывался как непосредственное расширение BOOTP и именно в таком качестве воспринимается BOOTP-клиентами. Этому обстоятельству в первую очередь способствует формат сообщений DHCP, во многом совпадающий с форматом, который применяется протоколом-предшественником и определен в документе RFC 951.

Сравнивая протоколы BOOTP и DHCP, нельзя не отметить появления в DHCP новых услуг. Во-первых, в этом протоколе предусмотрен механизм автоматической выдачи IP-адресов во временное пользование с возможностью их последующего присвоения новым клиентам. Во-вторых, клиент может получить от сервера все параметры конфигурации, которые ему необходимы для успешного функционирования в IP-сети.

Указанные отличия потребовали частичного расширения формата сообщений. Так, в нем появилось отдельное поле идентификатора клиента, сделана более прозрачной интерпретация адреса сервера (поле siaddr), переменный размер получило поле options, используемое, в частности, для передачи параметров конфигурации (его длина обычно находится в диапазоне 312-576 байт, хотя возможно и дополнительное расширение этого поля за счет полей sname и file).

В роли транспортного протокола для обмена DHCP-сообщениями выступает UDP. При отправке сообщения с клиента на сервер используется 67-й порт DHCP-сервера, при передаче в обратном направлении - 68-й. Эти номера портов, как и схожая структура сообщений, обеспечивают обратную совместимость DHCP с BOOTP. Конкретные процедуры взаимодействия клиентов и серверов BOOTP и DHCP регламентирует документ RFC 1542.

Протокол DHCP поддерживает три механизма выделения адресов: автоматический, динамический и ручной. В первом случае клиент получает постоянный IP-адрес, в последнем DHCP используется только для уведомления клиента об адресе, который администратор присвоил ему вручную. Оба эти варианта не таят в себе чего-либо принципиально нового, а вот динамический механизм заслуживает детального рассмотрения.

Выдача адреса в аренду производится по запросу клиента. DHCP-сервер (или группа серверов) гарантирует, что выделенный адрес до истечения срока его аренды не будет выдан другому клиенту; при повторных обращениях сервер старается предложить клиенту адрес, которым тот пользовался ранее. Со своей стороны, клиент может запросить пролонгацию срока аренды адреса либо, наоборот, досрочно отказаться от него. Протоколом предусмотрена также выдача IP-адреса в неограниченное пользование. При острой нехватке адресов сервер может сократить срок аренды адреса по сравнению с запрошенным.

1. Клиент посылает в собственную физическую подсеть широковещательное сообщение DHCPDISCOVER, в котором могут указываться устраивающие клиента IP-адрес и срок его аренды. Если в данной подсети DHCP-сервер отсутствует, сообщение будет передано в другие подсети ретранслирующими агентами протокола BOOTP (они же вернут клиенту ответные сообщения сервера).
2. Любой из DHCP-серверов может ответить на поступившее сообщение DHCPDISCOVER сообщением DHCPOFFER, включив в него доступный IP-адрес (yiaddr) и, если требуется, параметры конфигурации клиента. На этой стадии сервер не обязан резервировать указанный адрес. В принципе, он имеет право предложить его другому клиенту, также отправившему запрос DHCPDISCOVER. Тем не менее спецификации RFC 2131 рекомендуют серверу без необходимости не применять подобную тактику, а кроме того, убедиться (например, выдав эхо-запрос ICMP) в том, что предложенный адрес в текущий момент не используется каким-либо из компьютеров сети.
3. Клиент не обязан реагировать на первое же поступившее предложение. Допускается, чтобы он дождался откликов от нескольких серверов и, остановившись на одном из предложений, отправил в сеть широковещательное сообщение DHCPREQUEST. В нем содержатся идентификатор выбранного сервера и, возможно, желательные значения запрашиваемых параметров конфигурации.

Не исключено, что клиента не устроит ни одно из серверных предложений. Тогда вместо DHCPREQUEST он снова выдаст в сеть запрос DHCPDISCOVER, а серверы так и не узнают, что их предложения отклонены. Именно по этой причине сервер не обязан резервировать помещенный в DHCPOFFER адрес.

Если в процессе ожидания серверных откликов на DHCPDISCOVER достигнут тайм-аут, клиент выдает данное сообщение повторно.

1. Присутствующий в сообщении DHCPREQUEST идентификатор позволяет соответствующему DHCP-серверу убедиться в том, что клиент принял именно его предложение. В ответ сервер отправляет подтверждение DHCPACK, содержащее значения требуемых параметров конфигурации, и производит соответствующую запись в базу данных.

Если к моменту поступления сообщения DHCPREQUEST предложенный адрес уже <ушел> к другому клиенту (например, первая станция слишком долго <размышляла> над поступившими предложениями), сервер отвечает сообщением DHCPNACK.

1. Получив сообщение DHCPACK, клиент обязан убедиться в уникальности IP-адреса (средствами протокола ARP) и зафиксировать суммарный срок его аренды. Последний рассчитывается как время, прошедшее между отправкой сообщения DHCPREQUEST и приемом ответного сообщения DHCPACK, плюс срок аренды, указанный в DHCPACK.

Обнаружив, что адрес уже используется другой станцией, клиент обязан отправить серверу сообщение DHCPDECLINE и не ранее чем через 10 с начать всю процедуру снова. Процесс конфигурирования возобновляется и при получении серверного сообщения DHCPNACK.

При достижении тайм-аута в процессе ожидания серверных откликов на сообщение DHCPREQUEST клиент выдает его повторно.

1. Для досрочного прекращения аренды адреса клиент отправляет серверу сообщение DHCPRELEASE.

Приведенная последовательность действий заметно упрощается, если станция-клиент желает повторно работать с IP-адресом, который когда-то уже был ей выделен. В этом случае первым отправляемым сообщением является DHCPREQUEST, в котором клиент указывает прежде использовавшийся адрес. В ответ он может получить сообщение DHCPACK или DHCPNACK (если адрес занят либо клиентский запрос является некорректным, например из-за перемещения клиента в другую подсеть). Обязанность проверить уникальность IP-адреса опять-таки возлагается на клиента.

***Выбор адреса DHCP-сервером.***Если на момент получения запроса DHCPDISCOVER сервер не располагает свободными IP-адресами, он может направить уведомление о возникшей проблеме администратору. В противном случае при выборе адреса обычно применяется следующий алгоритм. Клиенту выделяется адрес, записанный за ним в данный момент. Если это невозможно, сервер предложит адрес, которым пользовался клиент до окончания срока последней аренды (при условии, что данный адрес свободен), либо адрес, запрошенный самим клиентом при помощи соответствующей опции (опять же, если адрес не занят). Наконец, в том случае, когда все предыдущие варианты не проходят, новый адрес выбирается из пула доступных адресов с учетом подсети, из которой поступил клиентский запрос.

Заметим, что исходя из определенной сетевым администратором политики сервер может выдать клиенту адрес, отличающийся от запрошенного (даже при доступности последнего), вообще отказать в предоставлении адреса или предложить адрес, относящийся к другой подсети. Более того, DHCP-сервер вообще не обязан реагировать на каждый поступивший запрос DHCPDISCOVER. Это предоставляет администратору возможность контролировать доступ к сети, например разрешив серверу отвечать только тем клиентам, которые предварительно зарегистрировались с помощью специальной процедуры.

*Истечение срока аренды.* По мере того как срок аренды подходит к концу, клиент может завершить работу с данным адресом, отправив на DHCP-сервер сообщение DHCPRELEASE, либо заблаговременно запросить продление срока аренды. В первом случае возвращение в сеть потребует выполнения всей процедуры инициализации заново. Во втором - станция продолжит функционировать в сети без видимого замедления работы пользовательских приложений.

При пролонгировании аренды клиент проходит два состояния - обновления адреса (RENEWING) и обновления конфигурации (REBINDING). Первое наступает примерно на половине срока аренды адреса (так называемый момент T1), второе - по истечении приблизительно 7/8 полного времени аренды (момент T2); для рассинхронизации процессов реконфигурирования разных клиентов значения этих временных меток рандомизируются с помощью случайной добавки.

В момент T1 клиент оправляет DHCP-серверу, выдавшему адрес, сообщение DHCPREQUEST с просьбой продлить срок аренды. Получив положительный ответ (DHCPACK), клиент пересчитывает срок аренды и продолжает работу в обычном режиме. Клиент ожидает прихода ответа от сервера в течение (T2 - t)/2 с (при условии, что это значение не меньше 60 с), где t - время отсылки последнего сообщения DHCPREQUEST, после чего отправляет данное сообщение повторно.

Если ответ от сервера не поступил к моменту T2, клиент переходит в состояние REBINDING и передает уже широковещательное сообщение DHCPREQUEST со своим текущим сетевым адресом. В этом случае моменты повторных выдач запросов DHCPREQUEST рассчитываются аналогично предыдущему случаю, только вместо T2 фигурирует время окончания срока аренды.

Не исключено, однако, что ответ DHCPACK не придет до окончания срока аренды. Тогда клиент обязан немедленно прекратить выполнение любых сетевых операций и заново начать процесс инициализации. Если запоздавший ответ DHCPACK все-таки поступит, клиенту рекомендуется сразу же возобновить работу под прежним адресом.

# Протокол ICMP

Протокол обмена управляющими сообщениями *ICMP (Internet Control Message Protocol)* позволяет маршрутизатору сообщить конечному узлу об ошибках, с которыми машрутизатор столкнулся при передаче какого-либо IP-пакета от данного конечного узла.

Управляющие сообщения ICMP не могут направляться промежуточному маршрутизатору, который участвовал в передаче пакета, с которым возникли проблемы, так как для такой посылки нет адресной информации - пакет несет в себе только адрес источника и адрес назначения, не фиксируя адреса промежуточных маршрутизаторов. Протокол ICMP - это протокол *сообщения об ошибках*, а не протокол *коррекции ошибок*. Конечный узел может предпринять некоторые действия для того, чтобы ошибка больше не возникала, но эти действия протоколом ICMP не регламентируются. Каждое сообщение протокола ICMP передается по сети внутри пакета IP. Пакеты IP с сообщениями ICMP маршрутизируются точно так же, как и любые другие пакеты, без приоритетов, поэтому они также могут теряться. Кроме того, в загруженной сети они могут вызывать дополнительную загрузку маршрутизаторов. Для того, чтобы не вызывать лавины сообщения об ошибках, потери пакетов IP, переносящие сообщения ICMP об ошибках, не могут порождать новые сообщения ICMP.

**Формат сообщений протокола ICMP**

Существует несколько типов сообщений ICMP. Каждый тип сообщения имеет свой формат, при этом все они начинаются с общих трех полей: 8-битного целого числа, обозначающего тип сообщения (TYPE), 8-битного поля кода (CODE), который конкретизирует назначение сообщения, и 16-битного поля контрольной суммы (CHECKSUM). Кроме того, сообщение ICMP всегда содержит заголовок и первые 64 бита данных пакета IP, который вызвал ошибку. Это делается для того, чтобы узел-отправитель смог более точно проанализировать причину ошибки, так как все протоколы прикладного уровня стека TCP/IP содержат наиболее важную информацию для анализа в первых 64 битах своих сообщений.

**Эхо-протокол**

Протокол ICMP предоставляет сетевым администраторам средства для тестирования достижимости узлов сети. Эти средства представляют собой очень простой эхо-протокол, включающий обмен двумя типами сообщений: *эхо-запрос* и *эхо-ответ*. Компьютер или маршрутизатор посылают по интерсети эхо-запрос, в котором указывают IP-адрес узла, достижимость которого нужно проверить. Узел, который получает эхо-запрос, формирует и отправляет эхо-ответ и возвращает сообщение узлу - отправителю запроса. В запросе могут содержаться некоторые данные, которые должны быть возвращены в ответе. Так как эхо-запрос и эхо-ответ передаются по сети внутри IP-пакетов, то их успешная доставка означает нормальное функционирование всей транспортной системы интерсети. Во многих операционных системах используется утилита *ping,*которая предназначена для тестирования достижимости узлов. Эта утилита обычно посылает серию эхо-запросов к тестируемому узлу и предоставляет пользователю статистику об утерянных эхо-ответах и среднем времени реакции сети на запросы.

**Сообщения о недостижимости узла назначения**

Когда маршрутизатор не может передать или доставить IP-пакет, он отсылает узлу, отправившему этот пакет, сообщение "Узел назначения недостижим" (тип сообщения - 3). Это сообщение содержит в поле кода значение, уточняющее причину, по которой пакет не был доставлен. Маршрутизатор, обнаруживший по какой-либо причине, что он не может передать IP-пакет далее по сети, должен отправить ICMP-сообщение узлу-источнику, и только потом отбросить пакет. Кроме причины ошибки, ICMP-сообщение включает также заголовок недоставленного пакета и его первые 64 бита поля данных.

Узел или сеть назначения могут быть недостижимы из-за временной неработоспособности аппаратуры, из-за того, что отправитель указал неверный адрес назначения, а также из-за того, что маршрутизатор не имеет данных о маршруте к сети назначения.

Недостижимость протокола и порта означают отсутствие реализации какого-либо протокола прикладного уровня в узле назначения или же отсутствие открытого порта протоколов UDP или TCP в узле назначения. Ошибка фрагментации возникает тогда, когда отправитель послал в сеть пакет с признаком DF, запрещающим фрагментацию, а маршрутизатор столкнулся с необходимостью передачи этого пакета в сеть со значением MTU меньшим, чем размер пакета.

**Перенаправление маршрута**

Маршрутные таблицы у компьютеров обычно являются статическими, так как конфигурируются администратором сети, а у маршрутизаторов - динамическими, формируемыми автоматически с помощью протоколов обмена маршрутной информации. Поэтому с течением времени при изменении топологии сети маршрутные таблицы компьютеров могут устаревать. Кроме того, эти таблицы обычно содержат минимум информации, например, только адреса нескольких маршрутизаторов. Для корректировки поведения компьютеров маршрутизатор может использовать сообщение протокола ICMP, называемое "Перенаправление маршрута" (Redirect). Это сообщение посылается в том случае, когда маршрутизатор видит, что компьютер отправляет пакет некоторой сети назначения нерациональным образом, то есть не тому маршрутизатору локальной сети, от которого начинается более короткий маршрут к сети назначения. Механизм перенаправления протокола ICMP позволяет компьютерам содержать в конфигурационном файле только IP-адреса его локальных маршрутизаторов. С помощью сообщений о перенаправлении маршрутизаторы будут сообщать компьютеру всю необходимую ему информацию о том, какому маршрутизатору следует отправлять пакеты для той или иной сети назначения. То есть маршрутизаторы передадут компьютеру нужную ему часть их таблиц маршрутизации.

В сообщении "Перенаправление маршрута" маршрутизатор помещает IP-адрес маршрутизатора, которым нужно пользоваться в дальнейшем, и заголовок исходного пакета с первыми 64 битами его поля данных. Из заголовка пакета узел узнает, для какой сети необходимо пользоваться указанным маршрутизатором.

# Протоколы ARP и RARP

**ARP** (англ. Address Resolution Protocol — протокол определения адреса) — протокол канального уровня. Протокол ARP (address resolution protocol, RFC-826, std-38) решает проблему преобразования IP-адреса в МАС-адрес.

Рассмотрим процедуру преобразования адресов при отправлении сообщения. Пусть одна ЭВМ отправляет сообщение другой. Прикладной программе IP-адрес места назначения обычно известен. Для определения Ethernet-адреса просматривается ARP-таблица. Если для требуемого IP-адреса в ней присутствует МАС-адрес, то формируется и посылается соответствующий пакет.

Если же с помощью ARP-таблицы не удается преобразовать адрес, то выполняется следующее:

Всем машинам в сети посылается пакет с ARP-запросом (с широковещательным МАС-адресом).

Исходящий IP-пакет ставится в очередь.

Каждая машина, принявшая ARP-запрос, в своем ARP-модуле сравнивает собственный IP-адрес с IP-адресом в запросе. Если IP-адрес совпал, то прямо по МАС-адресу отправителя запроса посылается ответ, содержащий как IP-адрес ответившей машины, так и ее МАС-адрес. После получения ответа на свой ARP-запрос машина имеет требуемую информацию о соответствии IP и МАС-адресов, формирует соответствующий элемент ARP-таблицы и отправляет IP-пакет, ранее поставленный в очередь. Если же в сети нет машины с искомым IP-адресом, то ARP-ответа не будет и не будет записи в ARP-таблицу. Протокол IP будет уничтожать IP-пакеты, предназначенные для отправки по этому адресу.

Протоколы верхнего уровня не могут отличить случай повреждения в среде ethernet от случая отсутствия машины с искомым IP-адресом. Во многих реализациях в случае, если IP-адрес не принадлежит локальной сети, внешний порт сети (gateway) или маршрутизатор откликается, выдавая свой физический адрес (режим прокси-ARP).

Функционально, ARP делится на две части. Одна — определяет физический адрес при посылке пакета, другая отвечает на запросы других машин. ARP-таблицы имеют динамический характер, каждая запись в ней «живет» определенное время после чего удаляется. Менеджер сети может осуществить запись в ARP-таблицу, которая там будет храниться «вечно». ARP-пакеты вкладываются непосредственно в ethernet-кадры.

Тип оборудования — это тип интерфейса, для которого отправитель ищет адрес; код содержит 1 для Ethernet.

Поле код операции определяет, является ли данный пакет ARP-запросом (код = 1), ARP-откликом (2), RARP-запросом (3), или RARP-откликом (4). Это поле необходимо, как поле тип кадра в Ethernet пакетах, они идентичны для ARP-запроса и отклика.

ARP-таблицы строятся согласно документу RFC-1213 и для каждого IP-адреса содержит четыре кода:ifindex Физический порт (интерфейс), соответствующий данному адресу; физический адрес MAC-адрес.  
IP-адрес, соответствующий физическому адресу;  
тип адресного соответствия это поле может принимать 4 значения:

1. Вариант не стандартный и не подходит ни к одному из описанных ниже типов;
2. Данная запись уже не соответствует действительности;
3. Постоянная привязка;
4. Динамическая привязка;

ARP запросы могут решать и другие задачи. Так при загрузке сетевого обеспечения ЭВМ такой запрос может выяснить, а не присвоен ли идентичный IP-адрес какому-то еще объекту в сети. При смене физического интерфейса такой запрос может инициировать смену записи в ARP-таблице.

В рамках протокола ARP возможны самообращенные запросы (gratuitous ARP). При таком запросе инициатор формирует пакет, где в качестве IP используется его собственный адрес. Это бывает нужно, когда осуществляется стартовая конфигурация сетевого интерфейса. В таком запросе IP-адреса отправителя и получателя совпадают.

Самообращенный запрос позволяет ЭВМ решить две проблемы. Во-первых, определить, нет ли в сети объекта, имеющего тот же IР-адрес. Если на такой запрос придет отклик, то ЭВМ выдаст на консоль сообщение Dublicate IP address sent from Ethernet address <…>. Во-вторых, в случае смены сетевой карты производится корректировка записи в АRP-таблицах ЭВМ, которые содержали старый МАС-адрес инициатора. Машина, получающая ARP-запрос c адресом, который содержится в ее таблице, должна обновить эту запись.

Вторая особенность такого запроса позволяет резервному файловому серверу заменить основной, послав самообращенный запрос со своим МАС-адресом, но с IP вышедшего из строя сервера. Этот запрос вынудит перенаправление кадров, адресованных основному серверу на резервный. Клиенты сервера при этом могут и не знать о выходе основного сервера из строя. При этом возможны и неудачи, если программные реализации в ЭВМ не в полной мере следуют регламентация протокола ARP.

**RARP**

Когда загружается система с локальным диском, она обычно получает свой IP адрес из конфигурационного файла, который считывается с диска. Однако для систем, не имеющих диска, таких как X терминалы или бездисковые рабочие станции, требуются другой способ определения собственного IP адреса.

Каждая система в сети имеет уникальный аппаратный адрес, который назначается производителем сетевого интерфейса (сетевой платы). Принцип работы RARP заключается в том, что бездисковая система может считать свой уникальный аппаратный адрес с интерфейсной платы и послать RARP запрос (широковещательный фрейм в сеть), где потребует кого-нибудь откликнуться и сообщить IP адрес (с помощью RARP отклика).

Несмотря на то что концепция довольно проста, ее реализация как правило значительно сложнее чем ARP, который был описан выше. Официальная спецификация RARP находится в RFC 903 [Finlayson et al. 1984].

Формат пакета RARP практически идентичен пакету ARP. Единственное отличие заключается в том, что поле тип фрейма (frame type) для запроса или отклика RARP установлено в 0x8035, а поле op имеет значение 3 для RARP запроса и значение 4 для RARP отклика.

RARP запрос является широковещательным, а RARP отклик обычно персональный.

# Протокол UDP

Протокол **UDP** (User Datagram Protocol, RFC-768) является одним из основных протоколов, расположенных непосредственно над IP. Он предоставляет прикладным процессам транспортные услуги, немногим отличающиеся от услуг протокола IP. Протокол UDP обеспечивает доставку дейтограмм, но не требует подтверждения их получения. Протокол UDP не требует соединения с удаленным модулем UDP ("бессвязный" протокол). К заголовку IP-пакета UDP добавляет поля *порт отправителя* и *порт получателя*, которые обеспечивают мультиплексирование информации между различными прикладными процессами, а также поля *длина* UDP-дейтограммы и *контрольная сумма*, позволяющие поддерживать целостность данных. Таким образом, если на уровне IP для определения места доставки пакета используется адрес, на уровне UDP - номер порта.

**Область использования UDP**

Примерами сетевых приложений, использующих UDP, являются **NFS** (Network File System), TFTP (Trivial File Transfer protocol, RFC-1350), **RPC** (Remote Procedure Call, RFC-1057) и **SNMP** (Simple Network Management Protocol, RFC-1157). Малые накладные расходы, связанные с форматом UDP, а также отсутствие необходимости подтверждения получения пакета, делают этот протокол наиболее популярным при реализации приложений мультимедиа, но главное его место работы - локальные сети и мультимедиа.

Прикладные процессы и модули UDP взаимодействуют через UDP-порты. Эти порты нумеруются, начиная с нуля. Прикладной процесс, предоставляющий некоторые услуги (сервер), ожидает сообщений, направленных в порт, специально выделенный для этих услуг. Программа-сервер ждет, когда какая-нибудь программа-клиент запросит услугу.

Например, сервер SNMP всегда ожидает сообщения, адресованного в порт 161. Если клиент snmp желает получить услугу, он посылает запрос в UDP-порт 161 на машину, где работает сервер. На каждой машине может быть только один агент SNMP, т.к. существует только один порт 161. Данный номер порта является общеизвестным, т.е. фиксированным номером, официально выделенным в сети Internet для услуг SNMP. Общеизвестные номера портов определяются стандартами Internet.

Данные, отправляемые прикладным процессом через модуль UDP, достигают места назначения как единое целое. Например, если процесс-отправитель производит 5 записей в порт, то процесс-получатель должен будет сделать 5 чтений. Размер каждого записанного сообщения будет совпадать с размером каждого прочитанного. Протокол UDP сохраняет границы сообщений, определяемые прикладным процессом. Он никогда не объединяет несколько сообщений в одно и не делит одно сообщение на части. Формат UDP-сообщений представлен ниже на рис. 4.4.2.1:

**Формат UDP-дейтограмм**

Длина сообщения равна числу байт в UDP-дейтограмме, включая заголовок. Поле UDP контрольная сумма содержит код, полученный в результате контрольного суммирования UDP-заголовка и поля данные. Не трудно видеть, что этот протокол использует заголовок минимального размера (8 байт). Таблица номеров UDP-портов приведена ниже (4.4.2.1). Номера портов от 0 до 255 стандартизованы и использовать их в прикладных задачах не рекомендуется. Но и в интервале 255-1023 многие номера портов заняты, поэтому прежде чем использовать какой-то порт в своей программе, следует заглянуть в RFC-1700. Во второй колонке содержится стандартное имя, принятое в Internet, а в третей - записаны имена, принятые в UNIX.

**Стандартные номера портов UDP**

Зарегистрировано ряд портов для стандартного применения и в диапазоне 1024-65535.

**Схема вычисления контрольных сумм**

Модуль IP передает поступающий IP-пакет модулю UDP, если в заголовке этого пакета указан код протокола UDP. Когда модуль UDP получает дейтограмму от модуля IP, он проверяет контрольную сумму, содержащуюся в ее заголовке. Если контрольная сумма равна нулю, это означает, что отправитель ее не подсчитал. ICMP, IGMP, UDP и TCP протоколы имеют один и тот же алгоритм вычисления контрольной суммы (RFC-1071). Но вычисление контрольной суммы для UDP имеет некоторые особенности. Во-первых, длина UDP-дейтограммы может содержать нечетное число байт, в этом случае к ней добавляется нулевой байт, который служит лишь для унификации алгоритма и никуда не пересылается. Во-вторых, при расчете контрольной суммы для UDP и TCP добавляются 12-байтные псевдо-заголовки, содержащие IP-адреса отправителя и получателя, код протокола и длину дейтограммы. Как и в случае IP-дейтограммы, если вычисленная контрольная сумма равна нулю, в соответствующее поле будет записан код 65535.

Если контрольная сумма правильная (или равна 0), то проверяется порт назначения, указанный в заголовке дейтограммы. Если прикладной процесс подключен к этому порту, то прикладное сообщение, содержащиеся в дейтограмме, становится в очередь к прикладному процессу для прочтения. В остальных случаях дейтограмма отбрасывается. Если дейтограммы поступают быстрее, чем их успевает обрабатывать прикладной процесс, то при переполнении очереди сообщений поступающие дейтограммы отбрасываются модулем UDP. Следует учитывать, что во многих посылках контрольное суммирование не охватывает адреса отправителя и места назначения. При некоторых схемах маршрутизации это приводит к зацикливанию пакетов в случае повреждения его адресной части (адресат не признает его "своим").

Так как максимальная длина IP-дейтограммы равна 65535 байтам, максимальная протяженность информационного поля UDP-дейтограммы составляет 65507 байт. На практике большинство систем работает с UDP-дейтограммами с длиной 8192 байта или менее (Ethernet допускает 1508 байт). Детальное описание форматов, полей пакетов и пр. читатель может найти в RFC-768. Смотри также RFC-2147 (IPv6 Jumbo), RFC-2508 (компрессия заголовков) и RFC-3828 (Lightweight UDP).

Нашел применение UDP и в протоколе Teredo (туннелирование IPv6 для систем NAT).

# Протокол TCP. Рукопожатие. Подтверждение передачи и повторная передача. Управление потоком и контроль перегрузки.

В стеке протоколов TCP/IP протокол *TCP (Transmission Control Protocol)* работает так же, как и протокол UDP, на транспортном уровне. Он обеспечивает надежную транспортировку данных между прикладными процессами путем установления логического соединения.

**Сегменты TCP**

Единицей данных протокола TCP является сегмент. Информация, поступающая к протоколу TCP в рамках логического соединения от протоколов более высокого уровня, рассматривается протоколом TCP как неструктурированный поток байт. Поступающие данные буферизуются средствами TCP. Для передачи на сетевой уровень из буфера "вырезается" некоторая непрерывная часть данных, называемая сегментом.

В протоколе TCP предусмотрен случай, когда приложение обращается с запросом о срочной передаче данных (бит PSH в запросе установлен в 1). В этом случае протокол TCP, не ожидая заполнения буфера до уровня размера сегмента, немедленно передает указанные данные в сеть. О таких данных говорят, что они передаются вне потока - *out of band.*

Не все сегменты, посланные через соединение, будут одного и того же размера, однако оба участника соединения должны договориться о максимальном размере сегмента, который они будут использовать. Этот размер выбирается таким образом, чтобы при упаковке сегмента в IP-пакет он помещался туда целиком, то есть максимальный размер сегмента не должен превосходить максимального размера поля данных IP-пакета. В противном случае пришлось бы выполнять фрагментацию, то есть делить сегмент на несколько частей, для того, чтобы он вместился в IP-пакет.

Аналогичные проблемы решаются и на сетевом уровне. Для того, чтобы избежать фрагментации, должен быть выбран соответствующий максимальный размер IP-пакета. Однако при этом должны быть приняты во внимание максимальные размеры поля данных кадров (MTU) всех протоколов канального уровня, используемых в сети. Максимальный размер сегмента не должен превышать минимальное значение на множестве всех MTU составной сети.

**Порты и установление TCP-соединений**

В протоколе TCP также, как и в UDP, для связи с прикладными процессами используются порты. Номера портам присваиваются аналогичным образом: имеются стандартные, зарезервированные номера (например, номер 21 закреплен за сервисом FTP, 23 - за telnet), а менее известные приложения пользуются произвольно выбранными локальными номерами.

Однако в протоколе TCP порты используются несколько иным способом. Для организации надежной передачи данных предусматривается установление *логического соединения* между двумя прикладными процессами. В рамках соединения осуществляется обязательное подтверждение правильности приема для всех переданных сообщений, и при необходимости выполняется повторная передача. Соединение в TCP позволяет вести передачу данных одновременно в обе стороны, то есть полнодуплексную передачу.

Соединение в протоколе TCP идентифицируется парой полных адресов обоих взаимодействующих процессов (оконечных точек). Адрес каждой из оконечных точек включает IP-адрес (номер сети и номер компьютера) и номер порта. Одна оконечная точка может участвовать в нескольких соединениях.

Установление соединения выполняется в следующей последовательности:

При установлении соединения одна из сторон является инициатором. Она посылает запрос к протоколу TCP на открытие порта для передачи (active open).

После открытия порта протокол TCP на стороне процесса-инициатора посылает запрос процессу, с которым требуется установить соединение.

Протокол TCP на приемной стороне открывает порт для приема данных (passive open) и возвращает квитанцию, подтверждающую прием запроса.

Для того чтобы передача могла вестись в обе стороны, протокол на приемной стороне также открывает порт для передачи (active port) и также передает запрос к противоположной стороне.

Сторона-инициатор открывает порт для приема и возвращает квитанцию. Соединение считается установленным. Далее происходит обмен данными в рамках данного соединения.

**Концепция квитирования**

В рамках соединения правильность передачи каждого сегмента должна подтверждаться квитанцией получателя. *Квитирование* - это один из традиционных методов обеспечения надежной связи. Идея квитирования состоит в следующем.

Для того, чтобы можно было организовать повторную передачу искаженных данных отправитель нумерует отправляемые единицы передаваемых данных (далее для простоты называемые кадрами). Для каждого кадра отправитель ожидает от приемника так называемую положительную квитанцию - служебное сообщение, извещающее о том, что исходный кадр был получен и данные в нем оказались корректными. Время этого ожидания ограничено - при отправке каждого кадра передатчик запускает таймер, и если по его истечению положительная квитанция на получена, то кадр считается утерянным. В некоторых протоколах приемник, в случае получения кадра с искаженными данными должен отправить отрицательную квитанцию - явное указание того, что данный кадр нужно передать повторно.

Существуют два подхода к организации процесса обмена положительными и отрицательными квитанциями: с простоями и с организацией "окна".

Метод с простоями требует, чтобы источник, пославший кадр, ожидал получения квитанции (положительной или отрицательной) от приемника и только после этого посылал следующий кадр (или повторял искаженный). Из рисунка 6.1 видно, что в этом случае производительность обмена данными существенно снижается - хотя передатчик и мог бы послать следующий кадр сразу же после отправки предыдущего, он обязан ждать прихода квитанции. Снижение производительности для этого метода коррекции особенно заметно на низкоскоростных каналах связи, то есть в территориальных сетях.

Во втором методе для повышения коэффициента использования линии источнику разрешается передать некоторое количество кадров в непрерывном режиме, то есть в максимально возможном для источника темпе, без получения на эти кадры ответных квитанций. Количество кадров, которые разрешается передавать таким образом, называется размером окна. Рисунок 6.2 иллюстрирует данный метод для размера окна в W кадров. Обычно кадры при обмене нумеруются циклически, от 1 до W. При отправке кадра с номером 1 источнику разрешается передать еще W-1 кадров до получения квитанции на кадр 1. Если же за это время квитанция на кадр 1 так и не пришла, то процесс передачи приостанавливается, и по истечению некоторого тайм-аута кадр 1 считается утерянным (или квитанция на него утеряна) и он передается снова.

Если же поток квитанций поступает более-менее регулярно, в пределах допуска в W кадров, то скорость обмена достигает максимально возможной величины для данного канала и принятого протокола.

Этот алгоритм называют алгоритмом скользящего окна. Действительно, при каждом получении квитанции окно перемещается (скользит), захватывая новые данные, которые разрешается передавать без подтверждения.

**Реализация скользящего окна в протоколе TCP**

В протоколе TCP реализована разновидность алгоритма квитирования с использованием окна. Особенность этого алгоритма состоит в том, что, хотя единицей передаваемых данных является сегмент, окно определено на множестве нумерованных байт неструктурированного потока данных, поступающих с верхнего уровня и буферизуемых протоколом TCP.

Квитанция посылается только в случае правильного приема данных, отрицательные квитанции не посылаются. Таким образом, отсутствие квитанции означает либо прием искаженного сегмента, либо потерю сегмента, либо потерю квитанции.

В качестве квитанции получатель сегмента отсылает ответное сообщение (сегмент), в которое помещает число, на единицу превышающее максимальный номер байта в полученном сегменте. Если размер окна равен W, а последняя квитанция содержала значение N, то отправитель может посылать новые сегменты до тех пор, пока в очередной сегмент не попадет байт с номером N+W. Этот сегмент выходит за рамки окна, и передачу в таком случае необходимо приостановить до прихода следующей квитанции.

**Выбор тайм-аута**

Выбор времени ожидания (тайм-аута) очередной квитанции является важной задачей, результат решения которой влияет на производительность протокола TCP.

Тайм-аут не должен быть слишком коротким, чтобы по возможности исключить избыточные повторные передачи, которые снижают полезную пропускную способность системы. Но он не должен быть и слишком большим, чтобы избежать длительных простоев, связанных с ожиданием несуществующей или "заблудившейся" квитанции.

При выборе величины тайм-аута должны учитываться скорость и надежность физических линий связи, их протяженность и многие другие подобные факторы. В протоколе TCP тайм-аут определяется с помощью достаточно сложного адаптивного алгоритма, идея которого состоит в следующем. При каждой передаче засекается время от момента отправки сегмента до прихода квитанции о его приеме (время оборота). Получаемые значения времен оборота усредняются с весовыми коэффициентами, возрастающими от предыдущего замера к последующему. Это делается с тем, чтобы усилить влияние последних замеров. В качестве тайм-аута выбирается среднее время оборота, умноженное на некоторый коэффициент. Практика показывает, что значение этого коэффициента должно превышать 2. В сетях с большим разбросом времени оборота при выборе тайм-аута учитывается и дисперсия этой величины.

**Реакция на перегрузку сети**

Варьируя величину окна, можно повлиять на загрузку сети. Чем больше окно, тем большую порцию неподтвержденных данных можно послать в сеть. Если сеть не справляется с нагрузкой, то возникают очереди в промежуточных узлах-маршрутизаторах и в конечных узлах-компьютерах.

При переполнении приемного буфера конечного узла "перегруженный" протокол TCP, отправляя квитанцию, помещает в нее новый, уменьшенный размер окна. Если он совсем отказывается от приема, то в квитанции указывается окно нулевого размера. Однако даже после этого приложение может послать сообщение на отказавшийся от приема порт. Для этого, сообщение должно сопровождаться пометкой "срочно" (бит URG в запросе установлен в 1). В такой ситуации порт обязан принять сегмент, даже если для этого придется вытеснить из буфера уже находящиеся там данные.

Другим проявлением перегрузки сети является переполнение буферов в маршрутизаторах. В таких случаях они могут централизовано изменить размер окна, посылая управляющие сообщения некоторым конечным узлам, что позволяет им дифференцировано управлять интенсивностью потока данных в разных частях сети.

**Формат сообщений TCP**

Сообщения протокола TCP называются сегментами и состоят из заголовка и блока данных. Заголовок сегмента имеет следующие поля:

Порт источника (SOURS PORT) занимает 2 байта, идентифицирует процесс-отправитель;

Порт назначения (DESTINATION PORT) занимает 2 байта, идентифицирует процесс-получатель;

Последовательный номер (SEQUENCE NUMBER) занимает 4 байта, указывает номер байта, который определяет смещение сегмента относительно потока отправляемых данных;

Подтвержденный номер (ACKNOWLEDGEMENT NUMBER) занимает 4 байта, содержит максимальный номер байта в полученном сегменте, увеличенный на единицу; именно это значение используется в качестве квитанции;

Длина заголовка (HLEN) занимает 4 бита, указывает длину заголовка сегмента TCP, измеренную в 32-битовых словах. Длина заголовка не фиксирована и может изменяться в зависимости от значений, устанавливаемых в поле Опции;

Резерв (RESERVED) занимает 6 битов, поле зарезервировано для последующего использования;

Кодовые биты (CODE BITS) занимают 6 битов, содержат служебную информацию о типе данного сегмента, задаваемую установкой в единицу соответствующих бит этого поля:

* URG - срочное сообщение;
* ACK - квитанция на принятый сегмент;
* PSH - запрос на отправку сообщения без ожидания заполнения буфера;
* RST - запрос на восстановление соединения;
* SYN - сообщение используемое для синхронизации счетчиков переданных данных при установлении соединения;
* FIN - признак достижения передающей стороной последнего байта в потоке передаваемых данных.
* Окно (WINDOW) занимает 2 байта, содержит объявляемое значение размера окна в байтах;
* Контрольная сумма (CHECKSUM) занимает 2 байта, рассчитывается по сегменту;
* Указатель срочности (URGENT POINTER) занимает 2 байта, используется совместно с кодовым битом URG, указывает на конец данных, которые необходимо срочно принять, несмотря на переполнение буфера;
* Опции (OPTIONS) - это поле имеет переменную длину и может вообще отсутствовать, максимальная величина поля 3 байта; используется для решения вспомогательных задач, например, при выборе максимального размера сегмента;
* Заполнитель (PADDING) может иметь переменную длину, представляет собой фиктивное поле, используемое для доведения размера заголовка до целого числа 32-битовых слов.

# Трансляция адресов (NAT)

Ещё одним механизмом, позволяющим поддерживать сетевую *безопасность*, является технология *NAT*.

*NAT* (*Network Address* *Translation* – преобразование сетевых адресов) – это механизм в сетях *TCP/IP*, позволяющий преобразовывать IP-адреса транзитных пакетов. Механизм *NAT* описан в *RFC* 1631, *RFC* 3022.

Преобразование адресов методом *NAT* может производиться почти любым маршрутизирующим устройством – *Интернет*-маршрутизатором, сервером доступа, межсетевым экраном. Наиболее популярным является **Source NAT** (SNAT), суть механизма которого состоит в замене адреса источника (source) при прохождении пакета в одну сторону и обратной замене адреса назначения (*destination*) в ответном пакете. Наряду с адресами источника/назначения могут также заменяться номера портов источника и назначения.

Помимо SNAT, т.е. предоставления пользователям локальной сети с внутренними адресами доступа к сети *Интернет*, часто применяется также **Destination NAT**, когда обращения извне транслируются межсетевым экраном на *сервер* в локальной сети, имеющий внутренний *адрес* и потому недоступный из внешней сети непосредственно (без *NAT*).

Существует 3 базовых концепции трансляции адресов:

1. статическая (SAT, Static Network Address Translation),
2. динамическая (DAT, Dynamic Address Translation),
3. маскарадная (NAPT, NAT Overload, PAT).

**Статический NAT** отображает локальные IP-адреса на конкретные публичные адреса на основании один к одному. Применяется, когда *локальный хост* должен быть доступен извне с использованием фиксированных адресов.

**Динамический NAT** отображает набор частных адресов на некое множество публичных IP-адресов. Если число локальных хостов не превышает число имеющихся публичных адресов, каждому локальному адресу будет гарантироваться соответствие публичного адреса. В противном случае, число хостов, которые могут одновременно получить *доступ* во внешние сети, будет ограничено количеством публичных адресов.

**Маскарадный NAT** (NAPT, *NAT* *Overload*, *PAT*, маскарадинг) – форма динамического *NAT*, который отображает несколько частных адресов в единственный публичный IP-*адрес*, используя различные порты. Известен также как *PAT* (*Port Address* *Translation*).

Механизмов взаимодействия внутренней локальной сети с внешней общедоступной сетью может быть несколько – это зависит от конкретной задачи по обеспечению доступа во внешнюю *сеть* и обратно и прописывается определенными правилами. Определены 4 типа трансляции сетевых адресов:

* Full Cone (Полный конус)
* Restricted Cone (Ограниченный конус)
* Port Restricted Cone (Порт ограниченного конуса)
* Symmetric (Симметричный)

В первых трех типах *NAT* для взаимодействия разных IP-адресов внешней сети с адресами из локальной сети используется один и тот же внешний *порт*. Четвертый тип – симметричный – для каждого адреса и порта использует отдельный внешний *порт*.

При использовании *NAT*, работающему по типу **Full Cone**, внешний *порт* устройства (маршрутизатора, сервера доступа, межсетевого экрана) открыт для приходящих с любых адресов запросов. Если пользователю из Интернета нужно отправить пакет клиенту, расположенному за *NAT*’ом, то ему необходимо знать только внешний *порт* устройства, через который установлено соединение. Например, *компьютер* за *NAT*’ом с IP-адресом 192.168.0.4 посылает и получает пакеты через *порт* 8000, которые отображаются на внешний IP-*адрес* и *порт*, как 10.1.1.1:12345. Пакеты из внешней сети приходят на устройство с IP-адресом:портом 10.1.1.1:12345 и далее отправляются на клиентский *компьютер* 192.168.0.4:8000.

Во входящих пакетах проверяется только транспортный протокол; *адрес* и *порт* назначения, *адрес* и *порт* источника значения не имеют.

При использовании *NAT*, работающему по типу **Restricted Cone**, внешний *порт* устройства (маршрутизатора, сервера доступа, межсетевого экрана) открыт для любого пакета, посланного с клиентского компьютера, в нашем примере: 192.168.0.4:8000. А пакет, пришедший из внешней сети (например, от компьютера 172.16.0.5:4000) на устройство с адресом:портом 10.1.1.1:12345, будет отправлен на *компьютер* 192.168.0.4:8000 только в том случае, если 192.168.0.4:8000 предварительно посылал *запрос* на IP-*адрес* внешнего хоста ( в нашем случае – на *компьютер* 172.16.0.5:4000). То есть, *маршрутизатор* будет транслировать входящие пакеты только с определенного адреса источника (в нашем случае *компьютер* 172.16.0.5:4000), но *номер порта* источника при этом может быть любым. В противном случае, *NAT* блокирует пакеты, пришедшие с хостов, на которые 192.168.0.4:8000 не отправлял запроса.

Механизм *NAT* **Port Restricted Cone** почти аналогичен механизму *NAT* Restricted Cone. Только в данном случае *NAT* блокирует все пакеты, пришедшие с хостов, на которые клиентский *компьютер* 192.168.0.4:8000 не отправлял запроса по какому-либо IP-адресу и порту. Mаршрутизатор обращает внимание на соответствие номера порта источника и не обращает внимания на *адрес* источника. В нашем примере *маршрутизатор* будет транслировать входящие пакеты с любым адресом источника, но *порт* источника при этом должен быть 4000. Если клиент отправил запросы во внешнюю *сеть* к нескольким IP-адресам и портам, то они смогут посылать пакеты клиенту на IP-*адрес*:*порт* 10.1.1.1:12345.

**Symmetric NAT** существенно отличается от первых трех механизмов способом отображения внутреннего IP-адреса:порта на внешний *адрес*:*порт*. Это *отображение* зависит от IP-адреса:порта компьютера, которому предназначен посланный *запрос*. Например, если клиентский *компьютер* 192.168.0.4:8000 посылает *запрос* компьютеру №1 (172.16.0.5:4000), то он может быть отображен как 10.1.1.1:12345, в тоже время, если он посылает с того же самого порта (192.168.0.4:8000) на другой IP-*адрес*, он отображается по-другому (10.1.1.1:12346).

*Компьютер* №1 (172.16.0.5:4000) может отправить пакет только на 10.1.1.1:12345, а *компьютер* №2 (169.10.2.8:6000) – только на 10.1.1.1:12346. Если любой из них попытается отправить пакеты на *порт*, с которого он не получал запроса, *NAT* заблокирует данные пакеты.

Внешний IP-*адрес*:*порт* открыт только тогда, когда клиентский *компьютер* отправляет данные во внешнюю *сеть* по определенному адресу:порту.

*NAT* выполняет три важных функции.

Позволяет сэкономить IP-адреса, транслируя несколько внутренних IP-адресов в один внешний публичный IP-адрес (или в несколько, но меньшим количеством, чем внутренних). По такому принципу построено большинство сетей в мире: на небольшой район домашней сети местного провайдера или на офис выделяется 1 публичный (внешний) IP-адрес, за которым работают и получают доступ интерфейсы с частными (внутренними) IP-адресами.

Позволяет предотвратить или ограничить обращение снаружи к внутренним хостам, оставляя возможность обращения из внутренней сети во внешнюю. При инициации соединения изнутри сети создаётся трансляция. Ответные пакеты, поступающие снаружи, соответствуют созданной трансляции и поэтому пропускаются. Если для пакетов, поступающих из внешней сети, соответствующей трансляции не существует (а она может быть созданной при инициации соединения или статической), они не пропускаются.

Позволяет скрыть определённые внутренние сервисы внутренних хостов/серверов. По сути, выполняется та же указанная выше трансляция на определённый порт, но возможно подменить внутренний порт официально зарегистрированной службы (например, 80-й порт TCP (HTTP-сервер) на внешний 54055-й). Тем самым, снаружи, на внешнем IP-адресе после трансляции адресов на сайт (или форум) для осведомлённых посетителей можно будет попасть по адресу [http://dlink.ru:54055](http://dlink.ru:54055/), но на внутреннем сервере, находящимся за NAT, он будет работать на обычном 80-м порту.

Однако следует упомянуть и о недостатках данной технологии:

Не все протоколы могут "преодолеть" NAT. Некоторые не в состоянии работать, если на пути между взаимодействующими хостами есть трансляция адресов. Опеределенные межсетевые экраны, осуществляющие трансляцию IP-адресов, могут исправить этот недостаток, соответствующим образом заменяя IP-адреса не только в заголовках IP, но и на более высоких уровнях (например, в командах протокола FTP).

# Назначение и принципы работы протокола STUN

**STUN** (сокр. от [англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *Session Traversal Utilities for*[*NAT*](https://ru.wikipedia.org/wiki/NAT), Утилиты прохождения сессий для NAT, ранее [англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *Simple Traversal of*[*UDP*](https://ru.wikipedia.org/wiki/UDP)*through NATs*, Простое прохождение UDP через серверы NAT) — это [сетевой протокол](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D1%82%D0%BE%D0%BA%D0%BE%D0%BB_%D0%BF%D0%B5%D1%80%D0%B5%D0%B4%D0%B0%D1%87%D0%B8_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85), который позволяет клиенту, находящемуся за сервером трансляции адресов (или за несколькими такими серверами), определить свой внешний [IP-адрес](https://ru.wikipedia.org/wiki/IP-%D0%B0%D0%B4%D1%80%D0%B5%D1%81), способ трансляции адреса и порта во внешней сети, связанный с определённым внутренним номером порта. Эта информация используется для установления соединения UDP между двумя [хостами](https://ru.wikipedia.org/wiki/%D0%A5%D0%BE%D1%81%D1%82) в случае, если они оба находятся за [маршрутизатором](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%80%D1%88%D1%80%D1%83%D1%82%D0%B8%D0%B7%D0%B0%D1%82%D0%BE%D1%80) NAT. Протокол определён в [рекомендации](https://ru.wikipedia.org/wiki/RFC) [RFC 5389](https://tools.ietf.org/html/rfc5389) (предыдущая версия — [RFC 3489](https://tools.ietf.org/html/rfc3489)).

Существуют протоколы, использующие пакеты UDP для передачи голоса, изображения или текста по IP-сетям. К сожалению, если обе общающиеся стороны находятся за NAT’ом, соединение не может быть установлено обычным способом. Именно здесь STUN и оказывается полезным.

STUN — это клиент-серверный протокол. [VoIP](https://ru.wikipedia.org/wiki/VoIP)-клиент может включать в себя реализацию клиента STUN, который отправляет запрос серверу STUN. Затем сервер STUN отправляет клиенту обратно информацию о том, каков внешний адрес маршрутизатора NAT, и какой порт открыт на NAT для приема входящих запросов обратно во внутреннюю сеть.

Ответ также позволяет клиенту STUN определить, какой тип трансляции адреса используется, поскольку различные типы маршрутизаторов [NAT](https://ru.wikipedia.org/wiki/NAT) обрабатывают входящие UDP пакеты по-разному. STUN работает с тремя из четырех основных типов: Full Cone NAT, Symmetric NAT, Address Restricted NAT и Port Restricted NAT. В случае ограничивающего NAT клиент должен отправить пакет на удаленный узел, прежде чем NAT начнет пропускать пакеты от удаленного узла к клиенту. STUN не будет работать с симметричным NAT’ом (также называемым «двусторонний NAT»), который часто встречается в сетях больших компаний. При симметричном NAT IP-адрес сервера STUN отличается от конечного адреса, и из-за этого адрес NAT, который видит STUN-сервер, отличается от конечного адреса, который будет использоваться для отправки пакетов клиенту.

Как только клиент обнаружил свой внешний адрес, он может передать его узлу, с которым проходит соединение. Если на пути встречаются трансляторы типа «полный конус», любая из двух сторон может начать общение. Если же выполняется трансляция типа «ограниченный конус» или «порт ограниченного конуса», обе стороны должны начать передачу данных совместно.

Нужно отметить, что методы, описываемые в рекомендации [RFC 3489](https://tools.ietf.org/html/rfc3489), не обязательно требуют использования протокола STUN; они могут использовать в рамках любого протокола, основанного на UDP.

Соединение с STUN-сервером устанавливается через **UDP-порт 3478**, однако сервер предлагает клиентам выполнить проверку также и альтернативного IP-адреса и номера порта (у серверов STUN есть два IP-адреса). RFC устанавливает, что выбор порта и IP является произвольным.

**STUN** расшифровывается как Simple Traversal of User Datagram Protocol (UDP) through Network Address Translators (NAT’s) (упрощенное прохождение UDP пакетов через NAT). Как правило протокол STUN используется в различных сетевых схемах, но нам он интересен с точки зрения VoIP. В настоящее время VoIP базируется, в основном, на основе протокола SIP. На практике, большинство устройств, использующих протокол SIP, работают за роутером или файрволом, с чем возникает ряд проблем, связанных с некорректным прохождение сигнализации или голоса. Протокол STUN помогает решить эти проблемы.

**Цель протокола STUN**

Основной целью протокола STUN, является помощь устройствам, находящимся за NAT, узнать свой публичный IP адрес. Также протокол STUN позволяет устройствам, подключенным за шлюзом, обнаружить проброс портов, сделанный самим шлюзом (NAT), т.е. какой порт другие устройства могут использовать для подключения из публичной сети. Обратите внимание, что шлюзы и маршрутизаторы не всегда делают пробросы портов, это зависит от типа NAT и от того, как он настроен.

STUN также используется для обновления привязки NAT, например, как механизм Keep-Alive при использовании Restricted Cone NAT или Port Restricted cone NAT. При прохождении трафика через такие конфигурации NAT, внутренний адрес и порт сопоставляется с конкретным внешним адресом и портом. Но если такая привязка адреса и порта не используется после истечения определенного количества времени (в зависимости от конфигурации устройства), то такую привязку NAT исключает из обслуживания. Поэтому, когда внутреннее устройство SIP пытается подключиться снова к внешнему устройству (которое может быть тем же устройством, к которому было подключение ранее), используя тот же внутренний IP и порт, маршрутизатор назначит предыдущий IP адрес и порт.

**Протокол STUN**

STUN это протокол типа «клиент-сервер». STUN сервер обычно работает на протоколах TCP и UDP и прослушивает порт 3478. Клиент обычно обращается к STUN серверу на конкретный IP адрес и порт (3478), но сервер может использовать для выполнения запросов-ответов и альтернативный IP адрес и номер порт.

**Сценарий работы STUN сервера**

Шаг 1: Компьютер A отправляет STUN запрос, через роутер 192.168.1.1, к внешнему STUN серверу с IP адресом 64.25.58.65.

Шаг 2: Роутер (192.168.1.1) перенаправляет запрос к STUN серверу (64.25.58.65) и меняет порт 5060 на 15060.

Шаг 3: STUN сервер (64.25.58.65) отправляет ответ в сторону компьютера A , через роутер с публичным IP адресом 212.128.56.125, указывая, что первоначальный запрос был получен с IP адреса 212.128.56.125 и порта 15060

Когда компьютер устанавливает сеанс связи, на основе протокола SIP, с внешним устройством, он может уведомить внешнее устройство, что отправку пакетов обратно к компьютеру А следует производить на IP адрес  212.128.56.125 и порт 15060. Как сказано выше, протокол STUN играет очень важную роль при создании сеанса связи между устройствами, находящимися за NAT.

Протокол STUN имеет и свои минусы, он не работает должным образом в сетях с использованием Symmetric NAT, т.к. Symmetric NAT создает новый адрес и порт, каждый раз, когда внутренний хост пытается подключиться к внешнему устройству.

# Алгоритмы маршрутизации. Централизованные и децентрализованные. Внутренние и внешние протоколы маршрутизации

Алгоритмы маршрутизации

Качество алгоритма определяется следующими показателями:

* Оптимальность,
* Простота и низкие непроизводительные затраты,
* Живучесть и стабильность,
* Быстрая сходимость,
* Гибкость.

**Оптимальность**

Оптимальность характеризует способность алгоритма маршрутизации выбирать "наилучший" маршрут. Наилучший маршрут зависит от показателей и от "веса" этих показателей, используемых при проведении расчета.

**Простота и низкие непроизводительные затраты**

Алгоритмы маршрутизации разрабатываются как можно более простыми, чтобы эффективно обеспечивать свои функциональные возможности, с мимимальными затратами программного обеспечения. Особенно это важно, когда маршрутизация, должна выполняться в компьютере с ограниченными физическими ресурсами.

**Живучесть и стабильность**

Алгоритмы маршрутизации должны обладать живучестью. Другими словами, они должны четко функционировать в случае неординарных или непредвиденных обстоятельств, таких как отказы аппаратуры, условия высокой нагрузки и некорректные реализации. Т.к. роутеры расположены в узловых точках сети, их отказ может вызвать значительные проблемы.

Часто наилучшими алгоритмами маршрутизации оказываются те, которые выдержали испытание временем и доказали свою надежность в различных условиях работы сети.

**Быстрая сходимость**

Алгоритмы маршрутизации должны быстро сходиться. Сходимость - это процесс соглашения между всеми роутерами по оптимальным маршрутам. Когда какое-нибудь событие в сети приводит к тому, что маршруты или отвергаются, или наоборот, становятся доступными, роутеры рассылают сообщения об обновлении маршрутизации. Такие сообщения пронизывают сети, стимулируя пересчет оптимальных маршрутов и, в конечном итоге, вынуждая все роутеры придти к соглашению по этим маршрутам. Алгоритмы маршрутизации, которые сходятся медленно, могут привести к образованию петель маршрутизации или выходам из строя сети.

В данном случае, в момент времени t1 к роутеру 1 прибывает пакет. Роутер 1 уже был обновлен и поэтому он знает, что оптимальный маршрут к пункту назначения требует, чтобы следующей остановкой был роутер 2. Поэтому роутер 1 пересылает пакет в роутер 2. Роутер 2 еще не был обновлен, поэтому он полагает, что следующей оптимальной пересылкой должен быть роутер 1. Поэтому роутер 2 пересылает пакет обратно в роутер 1. Пакет будет продолжать скакать взад и вперед между двумя роутерами до тех пор, пока роутер 2 не получит корректировку маршрутизации, или пока время жизни  данного пакета не будет исчерпано.

**Гибкость**

Алгоритмы маршрутизации должны быть также гибкими, т.е. быстро и точно адаптироваться к разнообразным обстоятельствам в сети, таким как изменения полосы пропускания сети, размеров очереди к роутеру, величины задержки сети и других переменных. Например, предположим, что сегмент сети отвергнут. Многие алгоритмы маршрутизации, после того как они узнают об этой проблеме, быстро выбирают следующий наилучший путь для всех маршрутов, которые обычно используют этот сегмент.

**Типы алгоритмов**

Алгоритмы маршрутизации могут быть:

* Статическими или динамическими
* Одномаршрутными или многомаршрутными
* Одноуровневыми или иерархическими
* Внутридоменными и междоменными
* С интеллектом в главной вычислительной машине или в роутере
* Алгоритмами состояния канала или вектора расстояний

**Статические или динамические алгоритмы**

Статические алгоритмы маршрутизации вообще вряд ли являются алгоритмами. Статические таблицы маршрутизации устанавливаются администратором сети до начала маршрутизации. Они не меняются, если только администратор сети не изменит их. Алгоритмы, использующие статические маршруты, простые и хорошо работают там, где схема сети относительно проста, и трафик сети относительно предсказуем.

Т.к. статические системы маршрутизации не могут реагировать на изменения в сети, они, как правило, считаются непригодными для современных крупных, постоянно изменяющихся сетей. Большинство современных алгоритмов маршрутизации. - динамические.

Динамические алгоритмы маршрутизации подстраиваются к изменениям в сети в масштабе реального времени. Они выполняют это путем анализа поступающих сообщений об обновлении маршрутизации. Если в сообщении указывается, что имело место изменение сети, программы маршрутизации пересчитывают маршруты и рассылают новые сообщения о корректировке маршрутизации. Такие сообщения пронизывают сеть, стимулируя роутеры заново прогонять свои алгоритмы и соответствующим образом изменять таблицы маршрутизации.

**Одномаршрутные или многомаршрутные алгоритмы**

Некоторые сложные протоколы маршрутизации обеспечивают множество маршрутов к одному и тому же пункту назначения, делает возможной мультиплексную передачу трафика по многочисленным линиям; одномаршрутные алгоритмы не могут делать этого. Т. о. многомаршрутные алгоритмы могут обеспечить заначительно большую пропускную способность и надежность.

**Одноуровневые или иерархические алгоритмы**

Некоторые алгоритмы маршрутизации оперируют в плоском пространстве, в то время как другие используют иерархиии маршрутизации. В одноуровневой системе маршрутизации все роутеры равны по отношению друг к другу. В иерархической системе маршрутизации некоторые роутеры формируют то, что составляет основу (backbone - базу) маршрутизации. Пакеты из небазовых роутеров перемещаются к базовыи роутерам и пропускаются через них до тех пор, пока не достигнут общей области пункта назначения. Начиная с этого момента, они перемещаются от последнего базового роутера через один или несколько небазовых роутеров до конечного пункта назначения. Системы маршрутизации часто устанавливают логические группы узлов, называемых доменами, или автономными системами (AS), или областями. В иерархических системах одни роутеры какого-либо домена могут сообщаться с роутерами других доменов, в то время как другие роутеры этого домена могут поддерживать связь с роутерами только в пределах своего домена. В очень крупных сетях могут существовать дополнительные иерархические уровни. Роутеры наивысшего иерархического уровня образуют базу маршрутизации.

**Внутридоменные или междоменные алгоритмы**

Некоторые алгоритмы маршрутизации действуют только в пределах доменов; другие - как в пределах доменов, так и между ними. Природа этих двух типов алгоритмов различная.

Большая часть сетевой связи имеет место в пределах групп небольших компаний (доменов). Внутридоменным роутерам необходимо знать только о других роутерах в пределах своего домена, поэтому их алгоритмы маршрутизации могут быть упрощенными. Соответственно может быть уменьшен и трафик обновления маршрутизации, зависящий от используемого алгоритма маршрутизации.

Поэтому понятно, что оптимальный алгоритм внутридоменной маршрутизации не обязательно будет оптимальным алгоритмом междоменной маршрутизации.

**Алгоритмы с интеллектом в главной вычислительной машине или в роутере**

Некоторые алгоритмы маршрутизации предполагают, что адрес конечнго узла и адрес источника определяют весь маршрут. Обычно это называют маршрутизацией от источника. Интеллект маршрутизации находится в главной вычислительной машине. В системах маршрутизации от источника роутеры действуют просто как устройства хранения и пересылки пакета, без всякий раздумий отсылая его к следующей остановке.

В других алгоритмах сами роутеры определяют маршрут через об'единенную сеть, базируясь на своих собственных расчетах.

Системы с интеллектом в главной вычислительной машине чаще выбирают наилучшие маршруты, т.к. они, как правило, находят все возможные маршруты к пункту назначения, прежде чем пакет будет действительно отослан. Однако определение всех маршрутов часто требует большого об'ема времени.

**Алгоритмы состояния канала или вектора расстояния**

Алгоритмы состояния канала (известные также как алгоритмы "первоочередности наикратчайшего маршрута") направляют потоки маршрутной информации во все узлы об'единенной сети. Однако каждый роутер посылает только ту часть маршрутной таблицы, которая описывает состояние его собственных каналов.

Алгоритмы вектора расстояния (известные также как алгоритмы Бэлмана-Форда) требуют от каждогo роутера посылки всей или части своей маршрутной таблицы, но только своим соседям.

Алгоритмы состояния каналов фактически направляют небольшие корректировки по всем направлениям, в то время как алгоритмы вектора расстояний отсылают более крупные корректировки только в соседние роутеры. Отличаясь более быстрой сходимостью, алгоритмы состояния каналов несколько меньше склонны к образованию петель маршрутизации, чем алгоритмы вектора расстояния. С другой стороны, алгоритмы состояния канала характеризуются более сложными расчетами в сравнении с алгоритмами вектора расстояний, требуя большей процессорной мощности и памяти, чем алгоритмы вектора расстояний. Вследствие этого, реализация и поддержка алгоритмов состояния канала может быть более дорогостоящей. Несмотря на их различия, оба типа алгоритмов хорошо функционируют при самых различных обстоятельствах.

**Показатели алгоритмов (метрики)**

Маршрутные таблицы содержат информацию, которую используют программы для выбора наилучшего маршрута. Ниже перечислены показатели, которые используются в алгоритмах маршрутизации:

* Длина маршрута,
* Надежность,
* Задержка,
* Ширина полосы пропускания,
* Нагрузка,
* Стоимость связи.

Сложные алгоритмы маршрутизации при выборе маршрута могут базироваться на множестве показателей, комбинируя их таким образом, что в результате получается один отдельный (гибридный) показатель.

**Длина маршрута**

Длина маршрута является наиболее общим показателем маршрутизации. Некоторые протоколы маршрутизации позволяют администраторам сети назначать произвольные цены на каждый канал сети. В этом случае длиной тракта является сумма расходов, связанных с каждым каналом.  Другие протоколы маршрутизации определяют "количество пересылок", т.е. показатель, характеризующий число проходов, которые пакет должен совершить на пути от источника до пункта назначения через изделия об'единения сетей (такие как роутеры).

**Надежность**

Надежность каждого канала сети в контексте алгоритмов маршрутизации обычно описывается в терминах отношения бит/ошибка. Некоторые каналы сети могут отказывать чаще, чем другие. Отказы одних каналов сети могут быть устранены легче или быстрее, чем отказы других каналов. При назначении оценок надежности могут быть приняты в расчет любые факторы надежности. Оценки надежности обычно назначаются каналам сети администраторами сети. Как правило, это произвольные цифровые величины.

**Задержка**

Под задержкой маршрутизации обычно понимают отрезок времени, необходимый для передвижения пакета от источника до пункта назначения через объединённую сеть. Задержка зависит от многих факторов, включая полосу пропускания промежуточных каналов сети, очереди в порт каждого роутера на пути передвижения пакета, перегруженность сети на всех промежуточных каналах сети и физическое расстояние, на которое необходимо переместить пакет. Т.к. здесь имеет место конгломерация нескольких важных переменных, задержка является наиболее общим и полезным показателем.

**Полоса пропускания**

Полоса пропускания относится к имеющейся мощности трафика какого-либо канала. При прочих равных показателях, канал Ethernet 10 Mbps предпочтителен любой арендованной линии с полосой пропускания 64 Кбайт/сек. Хотя полоса пропускания является оценкой максимально достижимой пропускной способности канала, маршруты, проходящие через каналы с большей полосой пропускания, не обязательно будут лучше маршрутов, проходящих через менее быстродействующие каналы.

# Алгоритм маршрутизации RIP

Протокол RIP (Routing Information Protocol, протокол маршрутной информации) является наиболее простым протоколом динамической маршрутизации. Он относится к протоколам типа «вектор-расстояние».

Под вектором протокол RIP определяет IP-адреса сетей, а расстояние измеряется в переходах («хопах», hope) – количестве маршрутизаторов, которое должен пройти пакет, чтобы достичь указанной сети. Следует отметить, что максимальное значение расстояния для протокола RIP равно 15, значение 16 трактуется особым образом «сеть недостижима». Это определило основной недостаток протокола – он оказывается неприменимым в больших сетях, где Возможны маршруты, превышающие 15 переходов.

Протокол RIP версии 1 имеет ряд существенных для практического использования недостатков. К числу важных проблем относятся следующие:

* **Оценка расстояния** только с учетом числа переходов. Протокол RIP не учитывает реальную производительность каналов связи, что может оказаться неэффективным в гетерогенных сетях, т.е. сетях, объединяющих каналы связи различного устройства, производительности, в которых используются разные сетевые технологии.
* **Проблема медленной конвергенции**. Маршрутизаторы,  использующие протокол RIP. Рассылают маршрутную информацию каждые 30 с, причем их работа не синхронизирована. В ситуации, когда некоторый маршрутизатор обнаружит, что какая-либо сеть стала недоступной, то в худшем случае (если проблема была выявлена сразу после очередной рассылки) он сообщит об это соседям через 30 с. Для соседних маршрутизаторов все будет происходить также. Это означает, что информация о недоступности какой-либо сети может распространятся маршрутизаторам в достаточно долго, очевидно, что сеть при этом будет находиться в нестабильном состоянии.
* **Широковещательная рассылка таблиц маршрутизации**. Протокол RIP изначально предполагал, что маршрутизаторы рассылают информацию в широковещательном режиме. Это означает, что отправленный пакет вынуждены получить и проанализировать на канальном, сетевом и транспортном уровне все компьютеры сети, в которую он направлен.

Частично указанные проблемы решаются в версии 2 (RIP2).

# Алгоритм маршрутизации OSPF

**Протокол OSPF** (Routing (Open Shortest Path First, «открой кратчайший путь первым») является более новым протоколом динамической маршрутизации и относится к протоколам типа «состояние канала».

Функционирование протокола OSPF основано на использовании всеми маршрутизаторами единой базы данных, описывающей, как и с какими сетями связан каждый маршрутизатор. Описывая каждую связь, маршрутизаторы связывают с ней метрику – значение, характеризующее «качество» канала. Например, для сетей Ethernet со скоростью обмена 100 Мбит/с используется значение 1, а для коммутируемых соединений 56 Кбит/с – значение 1785. Это позволяет маршрутизаторам OSPF (в отличие от RIP, где все каналы равнозначны) учитывать реальную пропускную способность и выявлять эффективные маршруты. Важной особенностью протокола OSPF является то, что используется групповая, а не широковещательная рассылка.

Указанные особенности, такие как групповая рассылка вместо широковещательной, отсутствие ограничений на длину маршрута, периодический обмен только короткими сообщениями о состоянии, учет «качества» каналов связи позволяют использовать OSPF в больших сетях. Однако такое использование может породить серьезную проблему – большое количество циркулирующей в сети маршрутной информации и увеличение таблиц маршрутизации. А поскольку алгоритм поиска эффективных маршрутов является, с точки зрения объема вычислений, достаточно сложным, то в больших сетях могут потребоваться высокопроизводительные и, следовательно, дорогие маршрутизаторы. Поэтому возможность построения эффективных таблиц маршрутизации может рассматриваться и как достоинство, и как недостаток протокола OSPF.

# Алгоритм маршрутизации BGP

Общая схема работы BGP такова. BGP-маршрутизаторы соседних АС, решившие обмениваться маршрутной информацией, устанавливают между собой соединения по протоколу BGP и становятся BGP-соседями (BGP-peers).

Далее BGP использует подход под названием path vector, являющийся развитием дистанционно-векторного подхода. BGP-соседи рассылают (анонсируют, advertise) друг другу векторы путей (path vectors). Вектор путей, в отличие от вектора расстояний, содержит не просто адрес сети и расстояние до нее, а адрес сети и список атрибутов (path attributes), описывающих различные характеристики маршрута от маршрутизатора-отправителя в указанную сеть. В дальнейшем для краткости мы будем называть набор данных, состоящих из адреса сети и атрибутов пути до этой сети, маршрутом в данную сеть.

Данных, содержащихся в атрибутах пути, должно быть достаточно, чтобы маршрутизатор-получатель, проанализировав их с точки зрения политики своей АС, мог принять решение о приемлемости или неприемлемости полученного маршрута.

**Реализация BGP**

Пара BGP-соседей устанавливает между собой соединение по протоколу TCP, порт 179. Соседи, принадлежащие разным автономным системам

, должны быть доступны друг другу непосредственно; для соседей из одной АС такого ограничения нет, поскольку протокол внутренней маршрутизации обеспечит наличие всех необходимых маршрутов между узлами одной автономной системы.

Поток информации, которым обмениваются BGP-соседи по протоколу TCP, состоит из последовательности BGP-сообщений. Максимальная длина сообщения 4096 октетов, минимальная - 19. Имеется 4 типа сообщений.

**Типы BGP-сообщений**

* OPEN - посылается после установления TCP-соединения. Ответом на OPEN является сообщение KEEPALIVE, если вторая сторона согласна стать BGP-соседом; иначе посылается сообщение NOTIFICATION с кодом, поясняющим причину отказа, и соединение разрывается.
* KEEPALIVE - сообщение предназначено для подтверждения согласия установить соседские отношения, а также для мониторинга активности открытого соединения: для этого BGP-соседи обмениваются KEEPALIVE-сообщениями через определенные интервалы времени.
* UPDATE - сообщение предназначено для анонсирования и отзыва маршрутов. После установления соединения с помощью сообщений UPDATE пересылаются все маршруты, которые маршрутизатор хочет объявить соседу (full update), после чего пересылаются только данные о добавленных или удаленных маршрутах по мере их появления (partial update).
* NOTIFICATION - сообщение этого типа используется для информирования соседа о причине закрытия соединения. После отправления этого сообщения BGP-соединение закрывается.

# Классификация удаленных атак на распределенные вычислительные системы

Основная цель любой классификации состоит в том, чтобы предложить такие классификационные признаки, используя которые можно наиболее точно описать классифицируемые явления или объекты. В связи с тем, что ни в одном из известных авторам научном исследовании не проводилось различия между локальными и удаленными информационными воздействиями на ВС, то применение уже известных обобщенных классификаций для описания удаленных воздействий не позволяет наиболее точно раскрыть их сущность и описать механизмы и условия их осуществления. Это связано с тем, что данный класс воздействий характеризуется сугубо специфичными признаками для распределенных вычислительных систем. Поэтому для более точного описания удаленных атак и предлагается следующая классификация.

**Итак, удаленные атаки можно классифицировать по следующим признакам:**

* По характеру воздействия
* пассивное (класс 1.1)
* активное (класс 1.2)

Пассивным воздействием на распределенную вычислительную систему назовем воздействие, которое не оказывает непосредственного влияния на работу системы, но может нарушать ее политику безопасности. Именно отсутствие непосредственного влияния на работу распределенной ВС приводит к тому, что пассивное удаленное воздействие практически невозможно обнаружить. Примером пассивного типового удаленного воздействия в РВС служит прослушивание канала связи в сети.

Под активным воздействием на распределенную ВС будем понимать воздействие, оказывающее непосредственное влияние на работу системы (изменение конфигурации РВС, нарушение работоспособности и т. д.) и нарушающее принятую в ней политику безопасности. Практически все типы удаленных атак являются активными воздействиями. Это связано с тем, что в самой природе разрушающего воздействия содержится активное начало. Очевидной особенностью активного воздействия по сравнению с пассивным является принципиальная возможность его обнаружения (естественно, с большей или меньшей степенью сложности), так как в результате его осуществления в системе происходят определенные изменения. В отличие от активного, при пассивном воздействии не остается никаких следов (от того, что атакующий просмотрит чужое сообщение в системе, в тот же момент ничего не изменится).

* По цели воздействия
* нарушение конфиденциальности информации либо ресурсов системы (класс 2.1)
* нарушение целостности информации (класс 2.2)
* нарушение работоспособности (доступности) системы (класс 2.3)

Этот классификационный признак является прямой проекцией трех основных типов угроз - раскрытия, целостности и отказа в обслуживании.

Основная цель практически любой атаки - получить несанкционированный доступ к информации. Существуют две принципиальные возможности доступа к информации: перехват и искажение. Возможность перехвата информации означает получение к ней доступа, но невозможность ее модификации. Следовательно, перехват информации ведет к нарушению ее конфиденциальности. Примером перехвата информации может служить прослушивание канала в сети (п. 3.2.1). В этом случае имеется несанкционированный доступ к информации без возможности ее искажения. Очевидно также, что нарушение конфиденциальности информации является пассивным воздействием.

Возможность искажения информации означает либо полный контроль над информационным потоком между объектами системы, либо возможность передачи сообщений от имени другого объекта. Таким образом, очевидно, что искажение информации ведет к нарушению ее целостности. Данное информационное разрушающее воздействие представляет собой яркий пример активного воздействия. Примером удаленной атаки, цель которой нарушение целостности информации, может служить типовая удаленная атака (УА) "Ложный объект РВС" (п. 3.2.3).

Принципиально другой целью атаки является нарушение работоспособности системы. В этом случае не предполагается получение атакующим несанкционированного доступа к информации. Его основная цель - добиться, чтобы операционная система на атакуемом объекте вышла из строя и для всех остальных объектов системы доступ к ресурсам атакованного объекта был бы невозможен. Примером удаленной атаки, целью которой является нарушение работоспособности системы, может служить типовая УА "Отказ в обслуживании" (п. 3.2.4).

* По условию начала осуществления воздействия

Удаленное воздействие, также как и любое другое, может начать осуществляться только при определенных условиях. В распределенных ВС существуют три вида условий начала осуществления удаленной атаки:

* Атака по запросу от атакуемого объекта (класс 3.1)

В этом случае атакующий ожидает передачи от потенциальной цели атаки запроса определенного типа, который и будет условием начала осуществления воздействия. Примером подобных запросов в ОС Novell NetWare может служить SAP-запрос (атака описана в [9]), а в сети Internet - DNS- и ARP-запросы. Удаленные атаки на объекты сети Internet, осуществляемые по запросу от атакуемой системы, рассматриваются в п. 4.2 и 4.3. Важно отметить, что данный тип удаленных атак наиболее характерен для распределенных ВС.

* Атака по наступлению ожидаемого события на атакуемом объекте (класс 3.2)

В этом случае атакующий осуществляет постоянное наблюдение за состоянием операционной системы удаленной цели атаки и при возникновении определенного события в этой системе начинает воздействие. Как и в предыдущем случае, инициатором осуществления начала атаки выступает сам атакуемый объект. Примером такого события может быть прерывание сеанса работы пользователя с сервером в ОС Novell NetWare без выдачи команды LOGOUT [9].

* Безусловная атака (класс 3.3)

В этом случае начало осуществления атаки безусловно по отношению к цели атаки, то есть атака осуществляется немедленно и безотносительно к состоянию системы и атакуемого объекта. Следовательно, в этом случае атакующий является инициатором начала осуществления атаки. Пример атаки данного вида см. в пункте 4.4.

* По наличию обратной связи с атакуемым объектом
* с обратной связью (класс 4.1)
* без обратной связи (однонаправленная атака) (класс 4.2)

Удаленная атака, осуществляемая при наличии обратной связи с атакуемым объектом, характеризуется тем, что на некоторые запросы, переданные на атакуемый объект, атакующему требуется получить ответ, а, следовательно, между атакующим и целью атаки существует обратная связь, которая позволяет атакующему адекватно реагировать на все изменения, происходящие на атакуемом объекте. Подобные удаленные атаки наиболее характерны для распределенных ВС.

В отличие от атак с обратной связью удаленным атакам без обратной связи не требуется реагировать на какие-либо изменения, происходящие на атакуемом объекте. Атаки данного вида обычно осуществляются передачей на атакуемый объект одиночных запросов, ответы на которые атакующему не нужны. Подобную УА можно называть однонаправленной удаленной атакой. Примером однонаправленных атак является типовая УА "Отказ в обслуживании" (п. 3.2.4), а также атаки, рассмотренные в п. 4.3, 4.4 и 4.6.

* По расположению субъекта атаки относительно атакуемого объекта
* внутрисегментное (класс 5.1)
* межсегментное (класс 5.2)
* По уровню эталонной модели ISO/OSI, на котором осуществляется воздействие
* физический (класс 6.1)
* канальный (класс 6.2)
* сетевой (класс 6.3)
* транспортный (класс 6.4)
* сеансовый (класс 6.5)
* представительный (класс 6.6)
* прикладной (класс 6.7)

Международная Организация по Стандартизации (ISO) приняла стандарт ISO 7498, описывающий взаимодействие открытых систем (OSI). Распределенные ВС также являются открытыми системами. Любой сетевой протокол обмена, как и любую сетевую программу, можно с той или иной степенью точности спроецировать на эталонную семиуровневую модель OSI. Такая многоуровневая проекция позволит описать в терминах модели OSI функции, заложенные в сетевой протокол или программу. Удаленная атака также является сетевой программой. В связи с этим представляется логичным рассматривать удаленные атаки на распределенные ВС, проецируя их на эталонную модель ISO/OSI.

# Удаленные атаки на распределенные вычислительные системы

В сети Internet основными базовыми протоколами удаленного доступа являются TELNET и FTP (File Transfer Protocol). TELNET - это протокол виртуального терминала (ВТ), позволяющий с удаленных хостов подключаться к серверам Internet в режиме ВТ. FTP - протокол, предназначенный для передачи файлов между удаленными хостами. Для получения доступа к серверу по данным протоколам пользователю необходимо пройти на нем процедуру идентификации и аутентификации. В качестве информации, идентифицирующей пользователя, выступает его идентификатор (имя), а для аутентификации используется пароль. Особенностью протоколов FTP и TELNET является то, что пароли и идентификаторы пользователей передаются по сети в открытом, незашифрованном виде. Таким образом, необходимым и достаточным условием для получения удаленного доступа к хостам по протоколам FTP и TELNET являются имя и пароль пользователя.

Одним из способов получения паролей и идентификаторов пользователей в сети Internet является анализ сетевого трафика. Сетевой анализ осуществляется с помощью специальной пpогpаммы-анализатоpа пакетов, перехватывающей все пакеты, передаваемые по сегменту сети, и выделяющей среди них те, в которых передаются идентификатоp пользователя и его паpоль. Сетевой анализ протоколов FTP и TELNET показывает, что TELNET разбивает пароль на символы и пересылает их по одному, помещая каждый символ из пароля в соответствующий пакет, а FTP, напротив, пересылает пароль целиком в одном пакете.

У внимательного читателя, наверное, уже возник вопрос, почему разработчики базовых прикладных протоколов Internet не предусмотрели возможностей шифрования передаваемых по сети паролей пользователей. Даже во всеми критикуемой сетевой ОС Novell NetWare 3.12 пароли пользователей никогда не передаются в открытом виде по сети (правда, этой ОС это особенно не помогает - [9]). Видимо, проблема в том, что базовые прикладные протоколы семейства TCP/IP разрабатывались очень давно - в период с конца 60-х до начала 80-х и c тех пор абсолютно не изменились. При этом точка зрения на построение глобальных сетей стала иной. Инфраструктура сети Internet и ее протоколы разрабатывались в основном из соображений надежности связи и уж никак не из соображений безопасности. Мы - пользователи сети Internet - сейчас пожинаем плоды, оставленные нам разработчиками этой морально устаревшей с точки зрения безопасности глобальной сети. Совершенно очевидно, что вычислительные системы за эти годы сделали колоссальный скачок в своем развитии. Поэтому, конечно, за эти годы подход к обеспечению информационной безопасности распределенных ВС существенно изменился. Были разработаны различные протоколы обмена, позволяющие защитить сетевое соединение и зашифровать трафик (например, протоколы SSL, SKIP и т. п.). Однако эти протоколы не сменили устаревшие и не стали стандартом для каждого пользователя (может быть, за исключением SSL). Вся проблема состоит в том, что для того, чтобы они стали стандартом, на эти протоколы должны перейти все пользователи сети, но, так как в Internet отсутствует централизованное управление сетью, то процесс перехода на эти протоколы может длиться еще многие годы. А на сегодняшний день подавляющее большинство пользователей используют стандартные протоколы семейства TCP/IP, разработанные более 15 лет назад. В результате, как показывают сообщения амеpиканских центpов по компьютеpной безопасности (CERT, CIAC), анализ сетевого трафика в сети Internet успешно пpименялся кракерами в последние годы, и, согласно матеpиалам специального комитета пpи конгpессе США, с его помощью в 1993-1994 годах было пеpе-хвачено около миллиона паpолей для доступа в различные информационные системы.

Рассмотрим обобщенную функциональную схему ложного ARP-сервера (рис. 4.3):

* ожидание ARP-запроса;
* при получении ARP-запроса передача по сети на запросивший хост ложного ARP-ответа, в котором указывается адрес сетевого адаптера атакующей станции (ложного ARP-сервера) или тот Ethernet-адрес, на котором будет принимать пакеты ложный ARP-сервер (совершенно необязательно указывать в ложном ARP-ответе свой настоящий Ethernet-адрес, так как при работе непосредственно с сетевым адаптером его можно запрограммировать на прием пакетов на любой Ethernet-адрес);
* прием, анализ, воздействие и передача пакетов обмена между взаимодействующими хостами (воздействие на перехваченную информацию см. п. 3.2.2.3).

В самом начале зарождения Internet для удобства пользователей было принято решение присвоить всем компьютерам в сети имена. Использование имен позволяет пользователю лучше ориентироваться в киберпространстве сети Internet - куда проще, понятней и наглядней для пользователя запомнить, например, имя ***www.ferrari.it***, чем четырехразрядную цепочку IP-адреса. Использование в Internet мнемонически понятных для пользователей имен породило проблему преобразования имен в IP-адреса. Такое преобразование необходимо, так как на сетевом уровне адресация пакетов идет не по именам, а по IP-адресам, следовательно, для непосредственной адресации сообщений в Internet имена не годятся. На этапе раннего развития Internet, когда в сеть было объединено небольшое количество компьютеров, NIC (Network Information Center) для решения проблемы преобразования имен в адреса создал специальный файл (hosts file), в который вносились имена и соответствующие им IP-адреса всех хостов в сети. Данный файл регулярно обновлялся и распространялся по всей сети. Но, по мере развития Internet, число объединенных в сеть хостов увеличивалось, и данная схема становилась все менее и менее работоспособной, поэтому была создана новая система преобразования имен, позволяющая пользователю в случае отсутствия у него информации о соответствии имен и IP-адресов получить необходимые сведения от ближайшего информационно-поискового сервера (DNS-сервера). Эта система получила название доменной системы имен - DNS (Domain Name System).

Для реализации системы DNS был создан специальный сетевой протокол DNS, для обеспечения эффективной работы которого в сети создаются специальные выделенные информационно-поисковые серверы - DNS-серверы. Поясним основную задачу, решаемую службой DNS. В современной сети Internet хост при обращении к удаленному серверу обычно имеет информацию только о его имени и не знает его IP-адреса, который и необходим для непосредственной адресации. Следовательно, перед хостом возникает стандартная проблема удаленного поиска: по имени удаленного хоста найти его IP-адрес. Решением этой проблемы и занимается служба DNS на базе протокола DNS.

Рассмотрим DNS-алгоритм удаленного поиска IP-адреса по имени в сети Internet:

хост посылает на IP-адрес ближайшего DNS-сервера (он устанавливается при настройке сетевой ОС) DNS-запрос, в котором указывает имя сервера, IP-адрес которого необходимо найти;

DNS-сервер, получив запрос, просматривает свою базу имен на наличие в ней указанного в запросе имени. В случае, если имя найдено, а, следовательно, найден и соответствующий ему IP-адрес, то на запросивший хост DNS-сервер отправляет DNS-ответ, в котором указывает искомыйIP-адрес. В случае, если указанное в запросе имя DNS-сервер не обнаружил в своей базе имен, то DNS-запрос отсылается DNS-сервером на один из корневых DNS-серверов, адреса которых содержатся в файле настроек DNS-сервера root.cache, и описанная в этом пункте процедура повторяется, пока имя не будет найдено (или не найдено).

Для осуществления этой удаленной атаки необходимо подготовить ложное ICMP Redirect Host сообщение, в котором указать конечный IP-адрес маршрута (адрес хоста, маршрут к которому будет изменен) и IP-адрес ложного маршрутизатора. Далее это сообщение передается на атакуемый хост от имени маршрутизатора. Для этого в IP-заголовке в поле адреса отправителя указывается IP-адрес маршрутизатора. В принципе, можно предложить два варианта данной удаленной атаки.

В первом случае атакующий находится в том же сегменте сети, что и цель атаки. Тогда, послав ложное ICMP-сообщение, он в качестве IP-адреса нового маршрутизатора может указать либо свой IP-адрес, либо любой из адресов данной подсети. Это даст атакующему возможность изменить маршрут передачи сообщений, направляемых атакованным хостом на определенный IP-адрес, и получить контроль над трафиком между атакуемым хостом и интересующим атакующего сервером. После этого атака перейдет во вторую стадию, связанную с приемом, анализом и передачей пакетов, получаемых от "обманутого" хоста. Рассмотрим функциональную схему осуществления этой удаленной атаки:

* передача на атакуемый хост ложного ICMP Redirect Host сообщения;
* отправление ARP-ответа в случае, если пришел ARP-запpос от атакуемого хоста;
* перенаправление пакетов от атакуемого хоста на настоящий маршрутизатор;
* перенаправление пакетов от маршрутизатора на атакуемый хост;
* при приеме пакета возможно воздействие на информацию по схеме "Ложный объект РВС"(п. 3.2.3.3).

**Атака на TCP пакет.** Итак, для осуществления описанной выше атаки необходимым и достаточным условием является знание двух текущих 32-битных параметров ISSa и ISSb, идентифицирующих TCP-соединение. Рассмотрим возможные способы их получения.

В том случае, когда атакующий находится в одном сегменте с целью атаки или через его сегмент проходит трафик предполагаемого объекта атаки, то задача получения значений ISSa и ISSb является тривиальной и решается путем анализа сетевого трафика. Следовательно, надо четко понимать, что протокол TCP позволяет, в принципе, защитить соединение только в случае невозможности перехвата атакующим сообщений, передаваемых по данному соединению, то есть в случае нахождения атакующего в других сегментах относительно абонентов TCP-соединения.

Поэтому наибольший интерес для нас представляют межсегментные атаки, когда атакующий и его цель находятся в разных сегментах сети. В этом случае задача получения значений ISSa и ISSb не является тривиальной. Далее предлагается следующее решение данной проблемы.

# Методы атаки узлов вычислительной системы

Не найдено

# Межсетевые экраны (брандмауэры) и их виды.

**Межсетево́й экра́н**, **сетево́й экра́н** — это комплекс аппаратных и программных средств в [компьютерной сети](https://ru.wikipedia.org/wiki/%D0%92%D1%8B%D1%87%D0%B8%D1%81%D0%BB%D0%B8%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D1%81%D0%B5%D1%82%D1%8C), осуществляющий контроль и фильтрацию проходящих через него [сетевых пакетов](https://ru.wikipedia.org/wiki/IP-%D0%BF%D0%B0%D0%BA%D0%B5%D1%82) в соответствии с заданными правилами.

Основной задачей сетевого экрана является защита сети или отдельных её узлов от [несанкционированного доступа](https://ru.wikipedia.org/wiki/%D0%9D%D0%B5%D1%81%D0%B0%D0%BD%D0%BA%D1%86%D0%B8%D0%BE%D0%BD%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D1%8B%D0%B9_%D0%B4%D0%BE%D1%81%D1%82%D1%83%D0%BF). Также сетевые экраны часто называют фильтрами, так как их основная задача — не пропускать (фильтровать) пакеты, не подходящие под критерии, определённые в конфигурации.

Некоторые сетевые экраны также позволяют осуществлять [трансляцию адресов](https://ru.wikipedia.org/wiki/NAT) — динамическую замену [внутрисетевых (серых) адресов](https://ru.wikipedia.org/wiki/%D0%9B%D0%BE%D0%BA%D0%B0%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D0%B2%D1%8B%D1%87%D0%B8%D1%81%D0%BB%D0%B8%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D1%81%D0%B5%D1%82%D1%8C#.D0.90.D0.B4.D1.80.D0.B5.D1.81.D0.B0.D1.86.D0.B8.D1.8F) или портов на внешние, используемые за пределами [локальной сети](https://ru.wikipedia.org/wiki/%D0%9B%D0%BE%D0%BA%D0%B0%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D1%81%D0%B5%D1%82%D1%8C), — что может обеспечивать дополнительную безопасность.

**Брандма́уэр** ([нем.](https://ru.wikipedia.org/wiki/%D0%9D%D0%B5%D0%BC%D0%B5%D1%86%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *Brandmauer*) — заимствованный из немецкого языка термин, являющийся аналогом английского *firewall* в его оригинальном значении (противопожарная перегородка — стена, которая разделяет смежные здания, предохраняя от распространения [пожара](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%B6%D0%B0%D1%80)). Интересно, что в области компьютерных технологий в немецком языке употребляется слово [*Firewall*](https://de.wikipedia.org/wiki/Firewall).

**Файрво́лл**, **файрво́л**, **файерво́л**, **фаерво́л** — образовано [транскрипцией](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%B0%D0%BA%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B0%D1%8F_%D1%82%D1%80%D0%B0%D0%BD%D1%81%D0%BA%D1%80%D0%B8%D0%BF%D1%86%D0%B8%D1%8F) английского термина *firewall*.

Поддерживаемый уровень сетевой [модели OSI](https://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D1%82%D0%B5%D0%B2%D0%B0%D1%8F_%D0%BC%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C_OSI) является основной характеристикой при классификации межсетевых экранов. Различают следующие типы межсетевых экранов:

* [Управляемые коммутаторы](https://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D1%82%D0%B5%D0%B2%D0%BE%D0%B9_%D0%BA%D0%BE%D0%BC%D0%BC%D1%83%D1%82%D0%B0%D1%82%D0%BE%D1%80) ([канальный уровень](https://ru.wikipedia.org/wiki/%D0%9A%D0%B0%D0%BD%D0%B0%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9_%D1%83%D1%80%D0%BE%D0%B2%D0%B5%D0%BD%D1%8C)).
* Сетевые фильтры [сетевого уровня](https://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D1%82%D0%B5%D0%B2%D0%BE%D0%B9_%D1%83%D1%80%D0%BE%D0%B2%D0%B5%D0%BD%D1%8C) (stateless). Фильтрация статическая, осуществляется путём анализа [IP](https://ru.wikipedia.org/wiki/Internet_Protocol)-адреса источника и приёмника, протокола, портов отправителя и получателя.
* Шлюзы [сеансового уровня](https://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D0%B0%D0%BD%D1%81%D0%BE%D0%B2%D1%8B%D0%B9_%D1%83%D1%80%D0%BE%D0%B2%D0%B5%D0%BD%D1%8C) (circuit-level proxy). В сетевой модели [TCP/IP](https://ru.wikipedia.org/wiki/TCP/IP) нет уровня, однозначно соответствующего сеансовому уровню OSI, поэтому к шлюзам сеансового уровня относят фильтры, которые невозможно отождествить ни с сетевым, ни с транспортным, ни с прикладным уровнем.
* шлюзы, транслирующие адреса ([NAT](https://ru.wikipedia.org/wiki/NAT), [PAT](https://ru.wikipedia.org/wiki/%D0%A2%D1%80%D0%B0%D0%BD%D1%81%D0%BB%D1%8F%D1%86%D0%B8%D1%8F_%D0%BF%D0%BE%D1%80%D1%82-%D0%B0%D0%B4%D1%80%D0%B5%D1%81)) или сетевые протоколы ([транслирующий мост](https://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D1%82%D0%B5%D0%B2%D0%BE%D0%B9_%D0%BC%D0%BE%D1%81%D1%82));
* фильтры контроля состояния канала. К фильтрам контроля состояния канала связи нередко относят сетевые фильтры сетевого уровня с расширенными возможностями (stateful), которые дополнительно анализируют [заголовки пакетов](https://ru.wikipedia.org/wiki/%D0%9F%D0%B0%D0%BA%D0%B5%D1%82_(%D1%81%D0%B5%D1%82%D0%B5%D0%B2%D1%8B%D0%B5_%D1%82%D0%B5%D1%85%D0%BD%D0%BE%D0%BB%D0%BE%D0%B3%D0%B8%D0%B8)) и умеют фильтровать [фрагментированные](https://ru.wikisource.org/wiki/ru.wikipedia.org/wiki/IP) пакеты);
* шлюзы сеансового уровня. Наиболее известным и популярным шлюзом сеансового уровня является посредник [SOCKS](https://ru.wikipedia.org/wiki/SOCKS).
* Шлюз [прикладного уровня](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%B8%D0%BA%D0%BB%D0%B0%D0%B4%D0%BD%D0%BE%D0%B9_%D1%83%D1%80%D0%BE%D0%B2%D0%B5%D0%BD%D1%8C) (application-level proxy), часто называемые [прокси-серверами](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%BA%D1%81%D0%B8-%D1%81%D0%B5%D1%80%D0%B2%D0%B5%D1%80). Делятся на прозрачные (transparent) и непрозрачные (solid).

Брандмауэр SPI ([Stateful Packet Inspection](https://ru.wikipedia.org/wiki/Stateful_Packet_Inspection), SPI), или иначе брандмауэры с динамической фильтрацией пакетов (Dynamic Packet Filtering), являются по сути шлюзами сеансового уровня с расширенными возможностями. Инспекторы состояния оперируют на сеансовом уровне, но «понимают» протоколы прикладного и сетевого уровней. В отличие от шлюза прикладного уровня, открывающего два виртуальных канала TCP (один — для клиента, другой — для сервера) для каждого соединения, инспектор состояния не препятствует организации прямого соединения между клиентом и сервером.

Существует также понятие «межсетевой экран экспертного уровня». Сетевой экран данного типа базируются на посредниках прикладного уровня или инспекторах состояния, но обязательно комплектуются шлюзами сеансового уровня и сетевыми фильтрами, иногда понимая и сетевой уровень. Зачастую имеют систему протоколирования событий и оповещения администраторов, средства поддержки удаленных пользователей (например авторизация), средства построения виртуальных частных сетей и т. д. К нему относятся почти все имеющиеся на рынке брандмауэры.

# Системы обнаружения и предотвращения вторжений (IDS и IPS).

**Система обнаружения вторжений (СОВ) (англ. Intrusion Detection System (IDS))** — программное или аппаратное средство, предназначенное для выявления фактов неавторизованного доступа (вторжения или сетевой атаки) в компьютерную систему или сеть.

IDS всё чаще становятся необходимым дополнением инфраструктуры сетевой безопасности. В дополнение к межсетевым экранам (firewall), работа которых происходит на основе политики безопасности, IDS служат механизмами мониторинга и наблюдения подозрительной активности. Они могут обнаружить атакующих, которые обошли Firewall, и выдать отчет об этом администратору, который, в свою очередь, предпримет дальнейшие шаги по предотвращению атаки. Технологии обнаружения проникновений не делают систему абсолютно безопасной. Тем не менее практическая польза от IDS существует и не маленькая.  
**Использование IDS помогает достичь нескольких целей:**

* Обнаружить вторжение или сетевую атаку;
* Спрогнозировать возможные будущие атаки и выявить уязвимости для предотвращения их дальнейшего развития. Атакующий обычно выполняет ряд предварительных действий, таких как, например, сетевое зондирование (сканирование) или другое тестирование для обнаружения уязвимостей целевой системы;
* Выполнить документирование существующих угроз;
* Обеспечить контроль качества администрирования с точки зрения безопасности, особенно в больших и сложных сетях;
* Получить полезную информацию о проникновениях, которые имели место, для восстановления и корректирования вызвавших проникновение факторов;
* Определить расположение источника атаки по отношению к локальной сети (внешние или внутренние атаки), что важно при принятии решений о расположении ресурсов в сети.

**Архитектура IDS**

**Обычно IDS включает:**

* Сенсорную подсистему, предназначенную для сбора событий, связанных с безопасностью защищаемой сети или системы;
* Подсистему анализа, предназначенную для выявления сетевых атак и подозрительных действий;
* Хранилище, в котором накапливаются первичные события и результаты анализа;
* Консоль управления, позволяющая конфигурировать IDS, наблюдать за состоянием защищаемой системы и IDS, просматривать выявленные подсистемой анализа инциденты.
* По способам мониторинга IDS системы подразделяются на network-based (NIDS) и host-based (HIDS).

Основными коммерческими IDS являются network-based. Эти IDS определяют атаки, захватывая и анализируя сетевые пакеты. Слушая сетевой сегмент, NIDS может просматривать сетевой трафик от нескольких хостов, которые присоединены к сетевому сегменту, и таким образом защищать эти хосты.  
**Преимущества NIDS:**

* Большое покрытие для мониторинга и в связи с этим централизованное управление;
* Несколько оптимально расположенных NIDS могут просматривать большую сеть.
* Не влияют на производительность и топологию сети.
* NIDS обычно являются пассивными устройствами, которые прослушивают сегменты сети без воздействия на её нормальное функционирование.

Таким образом, обычно бывает легко модифицировать топологию сети для размещения таких IDS.

**Недостатки NIDS:**

* Обладают высокой ресурсоёмкостью;
* Для NIDS может быть трудно обрабатывать все пакеты в большой или занятой сети, и, следовательно, они могут пропустить распознавание атаки, которая началась при большом трафике.
* Требуют дополнительной настройки и функциональности сетевых устройств;

Например, многие коммутаторы, на которых построены сети, не предоставляют универсального мониторинга портов, и это ограничивает диапазон мониторинга сенсора NIDS только одним хостом. Даже когда коммутаторы предоставляют такой мониторинг портов, часто единственный порт не может охватить весь трафик, передаваемый коммутатором.

* Не могут анализировать зашифрованную информацию;
* Не могут распознать результат атаки;

Host-based IDS имеют дело с информацией, собранной внутри единственного компьютера. Такое выгодное расположение позволяет HIDS анализировать деятельность с большой достоверностью и точностью, определяя только те процессы и пользователей, которые имеют отношение к конкретной атаке в ОС. НIDS обычно используют информационные источники двух типов: результаты аудита ОС и системные логи.  
**Преимущества HIDS:**

* Имеют возможностью следить за событиями локально относительно хоста, могут определять атаки, которые не могут видеть NIDS;
* Могут функционировать в окружении, в котором сетевой трафик зашифрован;
* Не требуют дополнительной функциональности сетевых устройств.

Например, на функционирование HIDS не влияет наличие в сети коммутаторов.

**Недостатки HIDS:**

* Не имеют централизованного управления;
* Могут быть блокированы некоторыми DoS-атаками или даже запрещены;
* Обладают высокой ресурсоёмкостью;
* Малое покрытие для мониторинг;

По способам определения вредоносного трафика IDS системы подразделяются на:

* signature-based (сигнатурного метода),
* anomaly-based (метода аномалий)
* policy-based (метода, основанного на политике).  
  **Система предотвращения вторжений (IPS)**

Типичное развертывание NIDS и NIPS систем. **Система предотвращения вторжений (англ. Intrusion Prevention System (IPS))** — программное или аппаратное средство, которое осуществляет мониторинг сети или компьютерной системы в реальном времени с целью выявления, предотвращения или блокировки вредоносной активности.  
  
В целом IPS по классификации и свои функциям аналогичны IDS. Главное их отличие состоит в том, что они функционируют в реальном времени и могут в автоматическом режиме блокировать сетевые атаки. Каждая IPS включает в себя модуль IDS.  
  
Как уже было сказано выше, правильное размещение систем IDS/IPS в сети не оказывает влияния на её топологию, но зато имеет огромное значение для оптимального мониторинга и достижения максимального эффекта от её защиты. На рисунке ниже приведён конкретный пример.

# Защита информации на IP уровне. Протокол IPSec

В конце шестидесятых годов американское агентство перспективных исследований в обороне DARPA приняло решение о создании экспериментальной сети под названием ARPANet. В семидесятых годах ARPANet стала считаться действующей сетью США, и через эту сеть можно было получить доступ к ведущим университетским и научным центрам США. В начале восьмидесятых годов началась стандартизация языков программирования, а затем и протоколов взаимодействия сетей. Результатом этой работы стала разработка семиуровневой модели сетевого взаимодействия ISO/OSI и семейства протоколов TCP/IP, которое стало основой для построения как локальных, так и глобальных сетей.

Базовые механизмы информационного обмена в сетях TCP/IP были в целом сформированы в начале восьмидесятых годов, и были направлены прежде всего на обеспечение доставки пакетов данных между различными операционными системами с использованием разнородных каналов связи. Несмотря на то, что идея создания сети ARPANet (впоследствии превратившейся в современный Интернет) принадлежала правительственной оборонной организации, фактически сеть зародилась в исследовательском мире, и наследовала традиции открытости академического сообщества. Ещё до коммерциализации Интернета (которая произошла в середине девяностых годов) многие авторитетные исследователи отмечали проблемы, связанные с безопасностью стека протоколов TCP/IP. Основные концепции протоколов TCP/IP не полностью удовлетворяют (а в ряде случаев и противоречат) современным представлениям о компьютерной безопасности.

До недавнего времени сеть Интернет использовалась в основном для обработки информации по относительно простым протоколам: электронная почта, передача файлов, удалённый доступ. Сегодня, благодаря широкому распространению технологий WWW, всё активнее применяются средства распределённой обработки мультимедийной информации. Одновременно с этим растёт объём данных, обрабатываемых в средах клиент/сервер и предназначенных для одновременного коллективного доступа большого числа абонентов. Разработано несколько протоколов прикладного уровня, обеспечивающих информационную безопасность таких приложений, как электронная почта (PEM, PGP и т.п.), WWW (Secure HTTP, SSL и т.п.), сетевое управление (SNMPv2 и т.п.). Однако наличие средств обеспечения безопасности в базовых протоколах семейства TCP/IP позволит осуществлять информационный обмен между широким спектром различных приложений и сервисных служб.

**Краткая историческая справка появления протокола**

В 1994 году Совет по архитектуре Интернет (IAB) выпустил отчет "Безопасность архитектуры Интернет". В этом документе описывались основные области применения дополнительных средств безопасности в сети Интернет, а именно защита от несанкционированного мониторинга, подмены пакетов и управления потоками данных. В числе первоочередных и наиболее важных защитных мер указывалась необходимость разработки концепции и основных механизмов обеспечения целостности и конфиденциальности потоков данных. Поскольку изменение базовых протоколов семейства TCP/IP вызвало бы полную перестройку сети Интернет, была поставлена задача обеспечения безопасности информационного обмена в открытых телекоммуникационных сетях на базе существующих протоколов. Таким образом, начала создаваться спецификация Secure IP, дополнительная по отношению к протоколам IPv4 и IPv6.

**Архитектура IPSec**

IP Security — это комплект протоколов, касающихся вопросов шифрования, аутентификации и обеспечения защиты при транспортировке IP-пакетов; в его состав сейчас входят почти 20 предложений по стандартам и 18 RFC.

Спецификация IP Security (известная сегодня как IPsec) разрабатывается [Рабочей группой IP Security Protocol IETF](http://www.ietf.org/html.charters/ipsec-charter.html). Первоначально IPsec включал в себя 3 алгоритмо-независимые базовые спецификации, опубликованные в качестве RFC-документов "Архитектура безопасности IP", "Аутентифицирующий заголовок (AH)", "Инкапсуляция зашифрованных данных (ESP)" (RFC1825, 1826 и 1827). Необходимо заметить, что в ноябре 1998 года Рабочая группа IP Security Protocol предложила новые версии этих спецификаций, имеющие в настоящее время статус предварительных стандартов, это RFC2401 — RFC2412. Отметим, что RFC1825-27 на протяжении уже нескольких лет считаются устаревшими и реально не используются. Кроме этого, существуют несколько алгоритмо-зависимых спецификаций, использующих протоколы MD5, SHA, DES.

Рабочая группа IP Security Protocol разрабатывает также и протоколы управления ключевой информацией. В задачу этой группы входит разработка Internet Key Management Protocol (IKMP), протокола управления ключами прикладного уровня, не зависящего от используемых протоколов обеспечения безопасности. В настоящее время рассматриваются концепции управления ключами с использованием спецификации Internet Security Association and Key Management Protocol (ISAKMP) и протокола Oakley Key Determination Protocol. Спецификация ISAKMP описывает механизмы согласования атрибутов используемых протоколов, в то время как протокол Oakley позволяет устанавливать сессионные ключи на компьютеры сети Интернет. Ранее рассматривались также возможности использования механизмов управления ключами протокола SKIP, однако сейчас такие возможности реально практически нигде не используются. Создаваемые стандарты управления ключевой информацией, возможно, будут поддерживать Центры распределения ключей, аналогичные используемым в системе [Kerberos](http://www.ixbt.com/comm/kerberos5.shtml). Протоколами ключевого управления для IPSec на основе Kerberos сейчас занимается относительно новая рабочая группа KINK (Kerberized Internet Negotiation of Keys).

Гарантии целостности и конфиденциальности данных в спецификации IPsec обеспечиваются за счет использования механизмов аутентификации и шифрования соответственно. Последние, в свою очередь, основаны на предварительном согласовании сторонами информационного обмена т.н. "контекста безопасности" – применяемых криптографических алгоритмов, алгоритмов управления ключевой информацией и их параметров. Спецификация IPsec предусматривает возможность поддержки сторонами информационного обмена различных протоколов и параметров аутентификации и шифрования пакетов данных, а также различных схем распределения ключей. При этом результатом согласования контекста безопасности является установление индекса параметров безопасности (SPI), представляющего собой указатель на определенный элемент внутренней структуры стороны информационного обмена, описывающей возможные наборы параметров безопасности.

По сути, IPSec, который станет составной частью IPv6, работает на третьем уровне, т. е. на сетевом уровне. В результате передаваемые IP-пакеты будут защищены прозрачным для сетевых приложений и инфраструктуры образом. В отличие от SSL (Secure Socket Layer), который работает на четвертом (т. е. транспортном) уровне и теснее связан с более высокими уровнями модели OSI, IPSec призван обеспечить низкоуровневую защиту.

К IP-данным, готовым к передаче по виртуальной частной сети, IPSec добавляет заголовок для идентификации защищенных пакетов. Перед передачей по Internet эти пакеты инкапсулируются в другие IP-пакеты. IPSec поддерживает несколько типов шифрования, в том числе Data Encryption Standard (DES) и Message Digest 5 (MD5).

Чтобы установить защищенное соединение, оба участника сеанса должны иметь возможность быстро согласовать параметры защиты, такие как алгоритмы аутентификации и ключи. IPSec поддерживает два типа схем управления ключами, с помощью которых участники могут согласовать параметры сеанса. Эта двойная поддержка в свое время вызвала определенные трения в IETF Working Group.

С текущей версией IP, IPv4, могут быть использованы или Internet Secure Association Key Management Protocol (ISAKMP), или Simple Key Management for Internet Protocol. С новой версией IP, IPv6, придется использовать ISAKMP, известный сейчас как IKE, хотя не исключается возможность использования SKIP. Однако, следует иметь в виду, что SKIP уже давно не рассматривается как кандидат управления ключами, и был исключён из списка возможных кандидатов ещё в 1997 г.

**Заголовок AH**

Аутентифицирующий заголовок (AH) является обычным опциональным заголовком и, как правило, располагается между основным заголовком пакета IP и полем данных. Наличие AH никак не влияет на процесс передачи информации транспортного и более высокого уровней. Основным и единственным назначением AH является обеспечение защиты от атак, связанных с несанкционированным изменением содержимого пакета, и в том числе от подмены исходного адреса сетевого уровня. Протоколы более высокого уровня должны быть модифицированы в целях осуществления проверки аутентичности полученных данных.

Формат AH достаточно прост и состоит из 96-битового заголовка и данных переменной длины, состоящих из 32-битовых слов. Названия полей достаточно ясно отражают их содержимое: Next Header указывает на следующий заголовок, Payload Len представляет длину пакета, SPI является указателем на контекст безопасности и Sequence Number Field содержит последовательный номер пакета.

Последовательный номер пакета был введен в AH в 1997 году в ходе процесса пересмотра спецификации IPsec. Значение этого поля формируется отправителем и служит для защиты от атак, связанных с повторным использованием данных процесса аутентификации. Поскольку сеть Интернет не гарантирует порядок доставки пакетов, получатель должен хранить информацию о максимальном последовательном номере пакета, прошедшего успешную аутентификацию, и о получении некоторого числа пакетов, содержащих предыдущие последовательные номера (обычно это число равно 64).

В отличие от алгоритмов вычисления контрольной суммы, применяемых в протоколах передачи информации по коммутируемым линиям связи или по каналам локальных сетей и ориентированных на исправление случайных ошибок среды передачи, механизмы обеспечения целостности данных в открытых телекоммуникационных сетях должны иметь средства защиты от внесения целенаправленных изменений. Одним из таких механизмов является специальное применение алгоритма MD5: в процессе формирования AH последовательно вычисляется хэш-функция от объединения самого пакета и некоторого предварительно согласованного ключа, а затем от объединения полученного результата и преобразованного ключа. Данный механизм применяется по умолчанию в целях обеспечения всех реализаций IPv6, по крайней мере, одним общим алгоритмом, не подверженным экспортным ограничениям.

**Заголовок ESP**

В случае использования инкапсуляции зашифрованных данных заголовок ESP является последним в ряду опциональных заголовков, "видимых" в пакете. Поскольку основной целью ESP является обеспечение конфиденциальности данных, разные виды информации могут требовать применения существенно различных алгоритмов шифрования. Следовательно, формат ESP может претерпевать значительные изменения в зависимости от используемых криптографических алгоритмов. Тем не менее, можно выделить следующие обязательные поля: SPI, указывающее на контекст безопасности и Sequence Number Field, содержащее последовательный номер пакета. Поле "ESP Authentication Data" (контрольная сумма), не является обязательным в заголовке ESP. Получатель пакета ESP расшифровывает ESP заголовок и использует параметры и данные применяемого алгоритма шифрования для декодирования информации транспортного уровня.

Различают два режима применения ESP и AH (а также их комбинации) — транспортный и туннельный.

**Транспортный режим**

Транспортный режим используется для шифрования поля данных IP пакета, содержащего протоколы транспортного уровня (TCP, UDP, ICMP), которое, в свою очередь, содержит информацию прикладных служб. Примером применения транспортного режима является передача электронной почты. Все промежуточные узлы на маршруте пакета от отправителя к получателю используют только открытую информацию сетевого уровня и, возможно, некоторые опциональные заголовки пакета (в IPv6). Недостатком транспортного режима является отсутствие механизмов скрытия конкретных отправителя и получателя пакета, а также возможность проведения анализа трафика. Результатом такого анализа может стать информация об объемах и направлениях передачи информации, области интересов абонентов, расположение руководителей.

**Туннельный режим**

Туннельный режим предполагает шифрование всего пакета, включая заголовок сетевого уровня. Туннельный режим применяется в случае необходимости скрытия информационного обмена организации с внешним миром. При этом, адресные поля заголовка сетевого уровня пакета, использующего туннельный режим, заполняются межсетевым экраном организации и не содержат информации о конкретном отправителе пакета. При передаче информации из внешнего мира в локальную сеть организации в качестве адреса назначения используется сетевой адрес межсетевого экрана. После расшифровки межсетевым экраном начального заголовка сетевого уровня пакет направляется получателю.

**Security Associations**

Security Association (SA) — это соединение, которое предоставляет службы обеспечения безопасности трафика, который передаётся через него. Два компьютера на каждой стороне SA хранят режим, протокол, алгоритмы и ключи, используемые в SA. Каждый SA используется только в одном направлении. Для двунаправленной связи требуется два SA. Каждый SA реализует один режим и протокол; таким образом, если для одного пакета необходимо использовать два протокола (как например AH и ESP), то требуется два SA.

**Политика безопасности**

Политика безопасности хранится в SPD (База данных политики безопасности). SPD может указать для пакета данных одно из трёх действий: отбросить пакет, не обрабатывать пакет с помощью IPSec, обработать пакет с помощью IPSec. В последнем случае SPD также указывает, какой SA необходимо использовать (если, конечно, подходящий SA уже был создан) или указывает, с какими параметрами должен быть создан новый SA.

SPD является очень гибким механизмом управления, который допускает очень хорошее управление обработкой каждого пакета. Пакеты классифицируются по большому числу полей, и SPD может проверять некоторые или все поля для того, чтобы определить соответствующее действие. Это может привести к тому, что весь трафик между двумя машинами будет передаваться при помощи одного SA, либо отдельные SA будут использоваться для каждого приложения, или даже для каждого TCP соединения.

**ISAKMP/Oakley**

Протокол ISAKMP определяет общую структуру протоколов, которые используются для установления SA и для выполнения других функций управления ключами. ISAKMP поддерживает несколько Областей Интерпретации (DOI), одной из которых является IPSec-DOI. ISAKMP не определяет законченный протокол, а предоставляет "строительные блоки" для различных DOI и протоколов обмена ключами.

Протокол Oakley — это протокол определения ключа, использующий алгоритм замены ключа Диффи-Хеллмана. Протокол Oakley поддерживает идеальную прямую безопасность (Perfect Forward Secrecy — PFS). Наличие PFS означает невозможность расшифровки всего траффика при компрометации любого ключа в системе.

I**KE**

IKE — протокол обмена ключами по умолчанию для ISAKMP, на данный момент являющийся единственным. IKE находится на вершине ISAKMP и выполняет, собственно, установление как ISAKMP SA, так и IPSec SA. IKE поддерживает набор различных примитивных функций для использования в протоколах. Среди них можно выделить хэш-функцию и псевдослучайную функцию (PRF).

Хэш-функция — это функция, устойчивая к коллизиям. Под устойчивостью к коллизиям понимается тот факт, что невозможно найти два разных сообщения m1 и m2, таких, что H(m1)=H(m2), где H — хэш функция.

Что касается псеводслучайных функций, то в настоящее время вместо специальных PRF используется хэш функция в конструкции HMAC (HMAC — механизм аутентификации сообщений с использованием хэш функций). Для определения HMAC нам понадобится криптографическая хэш функция (обозначим её как H) и секретный ключ K. Мы предполагаем, что H является хэш функцией, где данные хэшируются с помощью процедуры сжатия, последовательно применяемой к последовательности блоков данных. Мы обозначим за B длину таких блоков в байтах, а длину блоков, полученных в результате хэширования — как L (L<B). Ключ K может иметь длину, меньшую или равную B. Если приложение использует ключи большей длины, сначала мы должны хэшировать сам ключ с использованием H, и только после этого использовать полученную строку длиной L байт, как ключ в HMAC.

Из описания следует, что IKE использует для аутентификации сторон HASH величины. Отметим, что под HASH в данном случае подразумевается исключительно название Payload в ISAKMP, и это название не имеет ничего общего со своим содержимым.

# Защита WEB информации. Протокол SSL (TLS)

TLS и SSL упоминаются в последнее время все чаще и чаще, более актуальным становится использование цифровых сертификатов, и даже появились компании, готовые бесплатно предоставлять цифровые сертификаты всем желающим, чтобы гарантировать шифрование трафика между посещаемыми сайтами и браузером клиента. Нужно это, естественно, для безопасности, чтобы никто в сети не мог получить данные, которые передаются от клиента серверу и обратно. Как же это всё работает и как это использовать? Чтобы это понять, надо, пожалуй, начать с теории, а потом показать на практике. Итак, SSL и TLS.

**Что такое SSL и что такое TLS?**

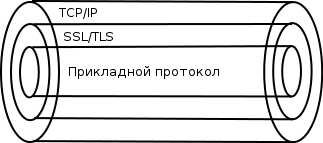
SSL — Secure Socket Layer, уровень защищенных сокетов. TLS — Transport Layer Security, безопасность транспортного уровня. SSL является более ранней системой, TLS появился позднее и он основан на спецификации SSL 3.0, разработанной компанией Netscape Communications. Тем не менее, задача у этих протоколов одна — обеспечение защищенной передачи данных между двумя компьютерами в сети Интернет. Такую передачу используют для различных сайтов, для электронной почты, для обмена сообщениями и много еще для чего. В принципе, можно передавать любую информацию таким образом, об этом чуть ниже.

Безопасная передача обеспечивается при помощи аутентификации и шифрования передаваемой информации. По сути эти протоколы, TLS и SSL, работают одинаково, принципиальных различий нет. TLS, можно сказать, является преемником SSL, хотя они и могут использоваться одновременно, причем даже на одном и том же сервере. Такая поддержка необходима для того, чтобы обеспечить работу как с новыми клиентами (устройствами и браузерами), так и с устаревшими, которые TLS не поддерживают. Последовательность возникновения этих протоколов выглядит вот так:

* SSL 1.0 — никогда не публиковался
* SSL 2.0 — февраль 1995 года
* SSL 3.0 — 1996 год
* TLS 1.0 — январь 1999 года
* TLS 1.1 — апрель 2006 года
* TLS 1.2 — август 2008 года

**Принцип работы SSL и TLS**

Принцип работы SSL и TLS, как я уже сказал, один и тот же. Поверх протокола TCP/IP устанавливается зашифрованный канал, внутри которого передаются данные по прикладному протоколу — HTTP, FTP, и так далее. Вот как это можно представить графически:



Прикладной протокол «заворачивается» в TLS/SSL, а тот в свою очередь в TCP/IP. По сути данные по прикладному протоколу передаются по TCP/IP, но они зашифрованы. И расшифровать передаваемые данные может только та машина, которая установила соединения. Для всех остальных, кто получит передаваемые пакеты, эта информация будет бессмысленной, если они не смогут ее расшифровать.

Установка соединения обеспечивается в несколько этапов:

* Клиент устанавливает соединение с сервером и запрашивает защищенное подключение. Это может обеспечиваться либо установлением соединения на порт, который изначально предназначен для работы с SSL/TLS, например, 443, либо дополнительным запросом клиентом установки защищенного соединения после установки обычного.
* При установке соединения клиент предоставляет список алгоритмов шифрования, которые он «знает». Сервер сверяет полученный список со списком алгоритмов, которые «знает» сам сервер, и выбирает наиболее надежный алгоритм, после чего сообщает клиенту, какой алгоритм использовать
* Сервер отправляет клиенту свой цифровой сертификат, подписанный удостоверяющим центром, и открытый ключ сервера.
* Клиент может связаться с сервером доверенного центра сертификации, который подписал сертификат сервера, и проверить, валиден ли сертификат сервера. Но может и не связываться. В операционной системе обычно уже установлены корневые сертификаты центров сертификации, с которыми сверяют подписи серверных сертификатов, например, браузеры.
* Генерируется сеансовый ключ для защищенного соединения. Это делается следующим образом:  
  — Клиент генерирует случайную цифровую последовательность  
  — Клиент шифрует ее открытым ключом сервера и посылает результат на сервер  
  — Сервер расшифровывает полученную последовательность при помощи закрытого ключа  
  Учитывая, что алгоритм шифрования является асимметричным, расшифровать последовательность может только сервер. При использовании асимметричного шифрования используется два ключа — приватный и публичный. Публичным отправляемое сообщение шифруется, а приватным расшифровывается. Расшифровать сообщение, имея публичный, ключ нельзя.
* Таким образом устанавливается зашифрованное соединение. Данные, передаваемые по нему, шифруются и расшифровываются до тех пор, пока соединение не будет разорвано.

**Корневой сертификат**

Чуть выше я упомянул корневой сертификат. Это сертификат авторизационного центра, подпись которым подтверждает, что сертификат, который подписан, является именно тем, который принадлежит соответствующему сервису. В самом сертификате обычно содержится ряд информационных полей, в которых содержится информация об имени сервера, которому выдан сертификат, и сроках действия этого сертификата. Если срок действия сертификата истек, он признается недействительным.

**Запрос на подпись (CSR, Certificate Sign Request)**

Для получения подписанного серверного сертификата необходимо сгенерировать запрос на подпись (CSR, Certificate Sign Request) и отправить этот запрос авторизационному центру, который вернет подписанный сертификат, устанавливаемый непосредственно на сервер, чуть ниже посмотрим, как это сделать на практике. Сначала генерируется ключ для шифрования, затем на основании этого ключа генерируется запрос на подпись, CSR-файл.

**Клиентский сертификат**

Клиентский сертификат может быть сгенерирован как для использования в устройствах, так и для использования пользователями. Обычно такие сертификаты используются при двусторонней верификации, когда клиент верифицирует, что сервер действительно тот, за кого себя выдает, и сервер в ответ делает то же самое. Такое взаимодействие называется двусторонней аутентификацией или mutual authentication. Двусторонняя аутентификация позволяет повысить уровень безопасности по сравнению с односторонней, а также может служить заменой аутентификации с использованием логина и пароля.

# Сертификаты X.509

**X.509** является стандартом ITU-T для [инфраструктуры открытых ключей](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%84%D1%80%D0%B0%D1%81%D1%82%D1%80%D1%83%D0%BA%D1%82%D1%83%D1%80%D0%B0_%D0%BE%D1%82%D0%BA%D1%80%D1%8B%D1%82%D1%8B%D1%85_%D0%BA%D0%BB%D1%8E%D1%87%D0%B5%D0%B9) (PKI) и управления привилегиями (PMI).

X.509 определяет форматы данных и процедуры распределения [открытых ключей](https://ru.wikipedia.org/wiki/%D0%9E%D1%82%D0%BA%D1%80%D1%8B%D1%82%D1%8B%D0%B9_%D0%BA%D0%BB%D1%8E%D1%87) с помощью [сертификатов](https://ru.wikipedia.org/wiki/%D0%A6%D0%B8%D1%84%D1%80%D0%BE%D0%B2%D0%BE%D0%B9_%D1%81%D0%B5%D1%80%D1%82%D0%B8%D1%84%D0%B8%D0%BA%D0%B0%D1%82) с [цифровыми подписями](https://ru.wikipedia.org/wiki/%D0%AD%D0%BB%D0%B5%D0%BA%D1%82%D1%80%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D1%86%D0%B8%D1%84%D1%80%D0%BE%D0%B2%D0%B0%D1%8F_%D0%BF%D0%BE%D0%B4%D0%BF%D0%B8%D1%81%D1%8C). Эти сертификаты предоставляются [удостоверяющими центрами](https://ru.wikipedia.org/wiki/%D0%A3%D0%B4%D0%BE%D1%81%D1%82%D0%BE%D0%B2%D0%B5%D1%80%D1%8F%D1%8E%D1%89%D0%B8%D0%B9_%D1%86%D0%B5%D0%BD%D1%82%D1%80).

В документе [RFC 1422](https://tools.ietf.org/html/rfc1422) описаны основы для [PKI](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%84%D1%80%D0%B0%D1%81%D1%82%D1%80%D1%83%D0%BA%D1%82%D1%83%D1%80%D0%B0_%D0%BE%D1%82%D0%BA%D1%80%D1%8B%D1%82%D1%8B%D1%85_%D0%BA%D0%BB%D1%8E%D1%87%D0%B5%D0%B9) на базе стандарта X.509.

В [RFC 5280](http://tools.ietf.org/html/rfc5280) определены [сертификат X.509 версии 3](http://tools.ietf.org/html/rfc5280#section-4) и [список отзыва сертификатов (CRL) версии 2](http://tools.ietf.org/html/rfc5280#section-5).

Для технологии открытых ключей необходимо, чтобы пользователь [открытого ключа](https://ru.wikipedia.org/wiki/%D0%9E%D1%82%D0%BA%D1%80%D1%8B%D1%82%D1%8B%D0%B9_%D0%BA%D0%BB%D1%8E%D1%87) был уверен, что этот ключ принадлежит именно тому удалённому субъекту (пользователю или системе), который будет использовать средства шифрования или цифровой подписи. Такую уверенность дают [сертификаты открытых ключей](https://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D1%80%D1%82%D0%B8%D1%84%D0%B8%D0%BA%D0%B0%D1%82_%D0%BE%D1%82%D0%BA%D1%80%D1%8B%D1%82%D0%BE%D0%B3%D0%BE_%D0%BA%D0%BB%D1%8E%D1%87%D0%B0), то есть структуры данных, которые связывают величины открытых ключей с субъектами. Эта связь достигается [цифровой подписью](https://ru.wikipedia.org/wiki/%D0%AD%D0%BB%D0%B5%D0%BA%D1%82%D1%80%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D1%86%D0%B8%D1%84%D1%80%D0%BE%D0%B2%D0%B0%D1%8F_%D0%BF%D0%BE%D0%B4%D0%BF%D0%B8%D1%81%D1%8C) доверенного [CA](https://ru.wikipedia.org/wiki/%D0%A3%D0%B4%D0%BE%D1%81%D1%82%D0%BE%D0%B2%D0%B5%D1%80%D1%8F%D1%8E%D1%89%D0%B8%D0%B9_%D1%86%D0%B5%D0%BD%D1%82%D1%80) под каждым сертификатом. Сертификат имеет ограниченный срок действия, указанный в его подписанном содержании. Поскольку пользователь сертификата может самостоятельно проверить его подпись и срок действия, сертификаты могут распространяться через незащищённые каналы связи и серверные системы, а также храниться в кэш-памяти незащищённых пользовательских систем. Содержание сертификата должно быть одинаковым в пределах всего [PKI](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%84%D1%80%D0%B0%D1%81%D1%82%D1%80%D1%83%D0%BA%D1%82%D1%83%D1%80%D0%B0_%D0%BE%D1%82%D0%BA%D1%80%D1%8B%D1%82%D1%8B%D1%85_%D0%BA%D0%BB%D1%8E%D1%87%D0%B5%D0%B9). В настоящее время в этой области предлагается общий стандарт для Интернет с использованием формата X.509:

* Номер версии
* Серийный номер
* [Эмитент](https://ru.wikipedia.org/wiki/%D0%AD%D0%BC%D0%B8%D1%82%D0%B5%D0%BD%D1%82)
* Субъект
* [Открытый ключ](https://ru.wikipedia.org/wiki/%D0%9E%D1%82%D0%BA%D1%80%D1%8B%D1%82%D1%8B%D0%B9_%D0%BA%D0%BB%D1%8E%D1%87) субъекта ([алгоритм](https://ru.wikipedia.org/wiki/%D0%9A%D1%80%D0%B8%D0%BF%D1%82%D0%BE%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D1%81_%D0%BE%D1%82%D0%BA%D1%80%D1%8B%D1%82%D1%8B%D0%BC_%D0%BA%D0%BB%D1%8E%D1%87%D0%BE%D0%BC), ключ)
* Период действия
* Дополнительные (необязательные) значения

Алгоритм подписи сертификата

Для описания внутренней структуры X509-сертификатов используется [ASN.1](https://ru.wikipedia.org/wiki/ASN.1). Хранятся, как правило, в виде [DER](https://ru.wikipedia.org/w/index.php?title=DER&action=edit&redlink=1) (бинарных) или [PEM](https://ru.wikipedia.org/w/index.php?title=PEM&action=edit&redlink=1)-файлов (текстовых, закодированных с помощью base64). Общепринятое расширение .cer или .crt.

# Протокол SET

Протокол защищенных электронных транзакций (SET), разработанный Visa International и MasterCard, специально предназначен для защиты транзакций в электронной коммерции. Протокол SET использует цифровые сертификаты для идентификации каждой стороны в электронной транзакции, включая покупателя, продавца и банк. Защита информации, передаваемой по Сети, дополняется методами криптографии, в которых используются открытые ключи.

Продавцы должны иметь цифровой сертификат и специальное программное течение от SET для обработки транзакций. У заказчика также должен быть цифровой сертификат и программное обеспечение электронного кошелька. Электронный кошелек устроен наподобие реального бумажника. В нем может храниться формация о нескольких кредитных или дебетовых картах, а также цифровой сертификат, удостоверяющий владельца карт. Электронный кошелек упрощает процессс совершения электронных покупок; покупателям больше не требуется повторно вводить информацию о кредитной карте в очередном электронном магазине.

Давайте рассмотрим, как выполняется транзакция электронной торговли, в которой используется протокол SET. Когда покупатель делает заказ, программное обеспечение SET продавца посылает данные заказа и цифровой сертификат продавца электронному кошельку заказчика, активизируя таким образом и программное обеспечение электронного кошелька. Заказчик выбирает карту, которая будет участвовать в транзакции. Затем данные кредитной карты и заказа зашифровываются, используя открытый ключ банка продавца, и отсылаются продавцу вместе с цифровым сертификатом покупателя. Продавец затем переправляет эту информацию в свой банк, где этот платеж должен быть обработан. Расшифровывать сообщение может только банк. Банк продавца посылает сумму платежа и свой цифровой сертификат в банк покупателя, чтобы получить подтверждение о возможности обработки транзакции. Если указанная сумма может быть списана с карточного счета, банк покупателя посылает подтверждение банку продавца. Банк продавца затем посылает разрешение на совершение сделки продавцу. В заключение, продавец посылает свое подтверждение сделки заказчику.

В протоколе SET продавец фактически не имеет доступа к частной информации клиента. Номер кредитной карты клиента не хранится на сервере продавца, что уменьшает риск мошенничества.

Хотя протокол SET специально разработан для транзакций электронной торговли и обеспечивает высокий уровень защиты, пока он не является стандартным протоколом, используемым для совершения транзакций. Часть проблемы заключается в том, что протокол SET требует специального программного обеспечения на стороне сервера, так и на стороне клиента, что приводит к дополнительным издержкам. Кроме того, подобные транзакции отнимают больше времени, чем транзакции, использующие другие протоколы, например SSL.

SET Secure Electronic Transaction LLC — организация, сформированная Visa и MasterCard для поддержки и продвижения на рынок протокола SET.

**Пример: Microsoft Authenticode**

Как вы сможете узнать, что программное обеспечение, которое вы приобрели в электронном магазине, безопасно и в него не были внесены какие-либо изменения? (каким образом вы можете убедиться в том, что не загружаете вместе с программой компьютерный вирус, который может уничтожить данные на вашем компьютере? Вы уверены в тех, от кого получили программное обеспечение? Этот список вопросов можно продолжить. С развитием электронной торговли компании, торгующие программным обеспечением, предлагают свой товар в электронных магазинах, откуда покупатели могут их загрузить непосредственно на свои компьютеры. Необходимо защищать разгружаемое программное обеспечение, чтобы быть уверенным в том, что при передаче по каналам связи оно не было изменено. Технология Microsoft Authenticode — подписи программного кода — совместно с цифровыми сертификатами VeriSign (цифровыми идентификаторами), идентифицирует издателя программного обеспечения и позволяет обнаружить изменения в коде программы, если они были туда внесены в процессе передачи данных по каналам связи. Защита Authenticode встроена в браузер Microsoft Internet Explorer. Распространители программного обеспечения должны получить цифровые сертификаты, специально разработанные для распространения программного обеспечения. Сертификаты могут быть получены у организаций, отвечающих за распределение сертификатов, таких, как, например, VeriSign. Чтобы получить удостоверение, продавец программного обеспечения должен предоставить свой открытый ключ, информацию о себе и подписать соглашение о том, что они не будут распространять программное обеспечение, которое может нанести ущерб. Это дает покупателям возможность обратиться за помощью, если загруженное программное обеспечение от сертифицированных распространителей причинило им вред.

Технология Microsoft Authenticode использует цифровую подпись. Программное обеспечение, подписанное цифровой подписью и удостоверенное цифровым сертификатом поставщика служат доказательством того, что программное обеспечение не несет в себе опасности и не было изменено.

Когда клиент пытается загрузить файл, на экране появляется диалоговое окно, в котором отображается цифровой сертификат и имя поставщика цифрового сертификата. Кроме того, даются ссылки на поставщика цифрового сертификата и поставщика программного продукта, чтобы клиент мог получить информацию о каждом из них до того, как он загрузит файл. Если Microsoft Authenticode определит, что программное обеспечение было скомпрометировано, транзакция будет прервана.

# Защита электронной почты (PGP)

**PGP** ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *Pretty Good Privacy*) — [компьютерная программа](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D0%B0%D1%8F_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0), также библиотека функций, позволяющая выполнять операции шифрования и цифровой подписи сообщений, файлов и другой информации, представленной в электронном виде, в том числе прозрачное шифрование данных на запоминающих устройствах, например, на жёстком диске.

Первоначально разработана [Филиппом Циммерманном](https://ru.wikipedia.org/wiki/%D0%A6%D0%B8%D0%BC%D0%BC%D0%B5%D1%80%D0%BC%D0%B0%D0%BD%D0%BD,_%D0%A4%D0%B8%D0%BB%D0%B8%D0%BF%D0%BF) в [1991 году](https://ru.wikipedia.org/wiki/1991_%D0%B3%D0%BE%D0%B4).

[Шифрование](https://ru.wikipedia.org/wiki/%D0%A8%D0%B8%D1%84%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) PGP осуществляется последовательно хешированием, сжатием данных, шифрованием с симметричным ключом, и, наконец, шифрованием с открытым ключом, причём каждый этап может осуществляться одним из нескольких поддерживаемых алгоритмов. Симметричное шифрование производится с использованием одного из семи симметричных алгоритмов ([AES](https://ru.wikipedia.org/wiki/Advanced_Encryption_Standard), [CAST5](https://ru.wikipedia.org/wiki/CAST5), [3DES](https://ru.wikipedia.org/wiki/Triple_DES), [IDEA](https://ru.wikipedia.org/wiki/IDEA), [Twofish](https://ru.wikipedia.org/wiki/Twofish), [Blowfish](https://ru.wikipedia.org/wiki/Blowfish), [Camellia](https://ru.wikipedia.org/wiki/Camellia_(%D0%B0%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC))) на сеансовом ключе. Сеансовый ключ генерируется с использованием криптографически стойкого [генератора псевдослучайных чисел](https://ru.wikipedia.org/wiki/%D0%93%D0%B5%D0%BD%D0%B5%D1%80%D0%B0%D1%82%D0%BE%D1%80_%D0%BF%D1%81%D0%B5%D0%B2%D0%B4%D0%BE%D1%81%D0%BB%D1%83%D1%87%D0%B0%D0%B9%D0%BD%D1%8B%D1%85_%D1%87%D0%B8%D1%81%D0%B5%D0%BB). Сеансовый ключ зашифровывается открытым ключом получателя с использованием алгоритмов [RSA](https://ru.wikipedia.org/wiki/RSA) или [Elgamal](https://ru.wikipedia.org/wiki/Elgamal) (в зависимости от типа ключа получателя). Каждый открытый ключ соответствует имени пользователя или адресу электронной почты. Первая версия системы называлась Сеть Доверия и противопоставлялась системе [X.509](https://ru.wikipedia.org/wiki/X.509), использовавшей иерархический подход, основанной на [удостоверяющих центрах](https://ru.wikipedia.org/wiki/%D0%A6%D0%B5%D0%BD%D1%82%D1%80_%D1%81%D0%B5%D1%80%D1%82%D0%B8%D1%84%D0%B8%D0%BA%D0%B0%D1%86%D0%B8%D0%B8), добавленный в PGP позже. Современные версии PGP включают оба способа.

**Ключи**

Пользователь PGP создаёт ключевую пару: открытый и закрытый [ключ](https://ru.wikipedia.org/wiki/%D0%9A%D0%BB%D1%8E%D1%87_(%D0%BA%D1%80%D0%B8%D0%BF%D1%82%D0%BE%D0%B3%D1%80%D0%B0%D1%84%D0%B8%D1%8F)). При генерации ключей задаются их владелец (имя и [адрес электронной почты](https://ru.wikipedia.org/wiki/%D0%90%D0%B4%D1%80%D0%B5%D1%81_%D1%8D%D0%BB%D0%B5%D0%BA%D1%82%D1%80%D0%BE%D0%BD%D0%BD%D0%BE%D0%B9_%D0%BF%D0%BE%D1%87%D1%82%D1%8B)), тип ключа, длина ключа и срок его действия. Открытый ключ используется для шифрования и проверки [цифровой подписи](https://ru.wikipedia.org/wiki/%D0%A6%D0%B8%D1%84%D1%80%D0%BE%D0%B2%D0%B0%D1%8F_%D0%BF%D0%BE%D0%B4%D0%BF%D0%B8%D1%81%D1%8C). Закрытый ключ — для декодирования и создания цифровой подписи.

PGP поддерживает три типа ключей [RSA](https://ru.wikipedia.org/wiki/RSA) v4, RSA legacy (v3) и [Diffie-Hellman/DSS](https://ru.wikipedia.org/wiki/Diffie-Hellman/DSS) ([Elgamal](https://ru.wikipedia.org/wiki/Elgamal) в терминологии [GnuPG](https://ru.wikipedia.org/wiki/GnuPG)).

Для ключей RSA legacy длина ключа может составлять от 1024 до 2048 бит, а для Diffie-Hellman/DSS и RSA — от 1024 до 4096. Ключи RSA legacy содержат одну ключевую пару, а ключи Diffie-Hellman/DSS и RSA могут содержать один главный ключ и дополнительные ключи для шифрования. При этом ключ электронной подписи в ключах Diffie-Hellman/DSS всегда имеет размер 1024. Срок действия для каждого из типов ключей может быть определён как неограниченный или до конкретной даты. Для защиты ключевого контейнера используется секретная фраза.

**Цифровая подпись**

PGP поддерживает [аутентификацию](https://ru.wikipedia.org/wiki/%D0%90%D1%83%D1%82%D0%B5%D0%BD%D1%82%D0%B8%D1%84%D0%B8%D0%BA%D0%B0%D1%86%D0%B8%D1%8F) и проверку целостности посредством [цифровой подписи](https://ru.wikipedia.org/wiki/%D0%AD%D0%BB%D0%B5%D0%BA%D1%82%D1%80%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D1%86%D0%B8%D1%84%D1%80%D0%BE%D0%B2%D0%B0%D1%8F_%D0%BF%D0%BE%D0%B4%D0%BF%D0%B8%D1%81%D1%8C). По умолчанию она используется совместно с шифрованием, но также может быть применена и к открытому тексту. Отправитель использует PGP для создания подписи алгоритмом [RSA](https://ru.wikipedia.org/wiki/RSA) или [DSA](https://ru.wikipedia.org/wiki/DSA). При этом сначала создаётся [хеш](https://ru.wikipedia.org/wiki/%D0%9A%D1%80%D0%B8%D0%BF%D1%82%D0%BE%D0%B3%D1%80%D0%B0%D1%84%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B0%D1%8F_%D1%85%D0%B5%D1%88-%D1%84%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D1%8F) открытого текста (также известный как дайджест), затем — цифровая подпись хеша при помощи [закрытого ключа](https://ru.wikipedia.org/wiki/%D0%90%D1%81%D0%B8%D0%BC%D0%BC%D0%B5%D1%82%D1%80%D0%B8%D1%87%D0%BD%D0%BE%D0%B5_%D1%88%D0%B8%D1%84%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) отправителя. Для формирования хеша могут использоваться алгоритмы [MD5](https://ru.wikipedia.org/wiki/MD5), [SHA-1](https://ru.wikipedia.org/wiki/SHA-1), [RIPEMD-160](https://ru.wikipedia.org/wiki/RIPEMD-160), [SHA-256](https://ru.wikipedia.org/wiki/SHA-256), [SHA-384](https://ru.wikipedia.org/wiki/SHA-384), [SHA-512](https://ru.wikipedia.org/wiki/SHA-512). В новых версиях PGP поддержка MD5 осуществляется для сохранения совместимости с ранними версиями. Для подписи используются алгоритмы RSA или DSA (в зависимости от типа ключа).

**Сжатие данных**

В целях уменьшения объёма сообщений и файлов и, возможно, для затруднения криптоанализа PGP производит [сжатие данных](https://ru.wikipedia.org/wiki/%D0%A1%D0%B6%D0%B0%D1%82%D0%B8%D0%B5_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85) перед шифрованием. Сжатие производится по одному из алгоритмов [ZIP](https://ru.wikipedia.org/wiki/ZIP), [ZLIB](https://ru.wikipedia.org/wiki/ZLIB), [BZIP2](https://ru.wikipedia.org/wiki/BZIP2). Для сжатых, коротких и слабосжимаемых файлов сжатие не выполняется.

**Сеть доверия**

Как при [шифровании](https://ru.wikipedia.org/wiki/%D0%A8%D0%B8%D1%84%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) сообщений, так и при проверке [цифровой подписи](https://ru.wikipedia.org/wiki/%D0%AD%D0%A6%D0%9F) необходимо, чтобы принятый получателем открытый ключ действительно принадлежал отправителю. При простом скачивании открытого ключа он может быть подменён. С первых версий PGP поддерживает [сертификаты](https://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D1%80%D1%82%D0%B8%D1%84%D0%B8%D0%BA%D0%B0%D1%82_(%D0%BA%D1%80%D0%B8%D0%BF%D1%82%D0%BE%D0%B3%D1%80%D0%B0%D1%84%D0%B8%D1%8F)) открытых ключей, с помощью которых подмены (или случайные ошибки передачи) легко распознаются. Однако недостаточно просто создать сертификат, защищённый от модификации, так как при этом гарантируется лишь целостность сертификата после его создания. Пользователи также должны каким-нибудь способом проверить, что открытый ключ в сертификате действительно принадлежит отправителю. С первых версий продукты PGP включают в себя внутреннюю схему проверки сертификатов, названную сеть доверия ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *web of trust*). Заданная пара «имя пользователя — открытый ключ» может быть подписана третьим лицом, удостоверяющим соответствие ключа и владельца. В таких подписях может быть несколько вложенных уровней доверия. Хотя многие программы читают и пишут эту информацию, очень немногие учитывают этот уровень сертификата, принимая решение о принятии или отклонении сертификата.

Протокол сети доверия был впервые описан Циммерманном в 1992 году в руководстве PGP версии 2.0: «С течением времени вы будете накапливать ключи других людей, которых вы можете назвать доверенными рекомендателями. Кто-нибудь ещё может выбрать своих доверительных рекомендателей. И все будут постепенно накапливать и распространять со своими ключами набор заверенных подписей других людей, ожидая, что любой получатель доверяет по крайней мере одной или двум подписям. Это позволяет создать децентрализованную устойчивую к сбоям сеть всех открытых ключей.»

Механизм сети доверия обладает преимуществами над [централизованной инфраструктурой управления открытыми ключами](https://ru.wikipedia.org/wiki/%D0%A6%D0%B5%D0%BD%D1%82%D1%80_%D1%81%D0%B5%D1%80%D1%82%D0%B8%D1%84%D0%B8%D0%BA%D0%B0%D1%86%D0%B8%D0%B8), например, используемой в [S/MIME](https://ru.wikipedia.org/wiki/S/MIME), но не получил повсеместного применения. Пользователи хотели проверять корректность сертификатов вручную или не проверять вовсе.

**Сертификаты**

В последних спецификациях OpenPGP доверенные подписи могут использоваться для поддержки создания центров сертификации. Доверенность сертификата означает, что ключ действительно принадлежит указанному владельцу и может использоваться для подписи сертификатов одним уровнем ниже. Сертификат уровня 0 означает обычную подпись. Уровень 1 означает, что при помощи подписанного ключа можно создавать сертификаты уровня 0. При помощи сертификата уровня 2 можно создавать сертификаты уровня 1. Уровень 2 практически идентичен степени доверия, с которой полагаются пользователи на списки доверенных сертификатов, встроенные в браузеры.

Все версии PGP включают в себя способ отмены сертификата. Это необходимо, если требуется сохранять безопасность связи при потере или [компрометации](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%80%D0%BE%D0%BC%D0%B5%D1%82%D0%B0%D1%86%D0%B8%D1%8F_(%D0%BA%D1%80%D0%B8%D0%BF%D1%82%D0%BE%D0%B3%D1%80%D0%B0%D1%84%D0%B8%D1%8F)) закрытого ключа. Отмена сертификата похожа на списки отзыва сертификатов в централизованной инфраструктуре открытых ключей. Современные версии PGP также поддерживают сроки истечения сертификатов.

Проблема корректного определения принадлежности открытого ключа владельцу характерна для всех криптографических систем с [асимметричным шифрованием](https://ru.wikipedia.org/wiki/%D0%90%D1%81%D0%B8%D0%BC%D0%BC%D0%B5%D1%82%D1%80%D0%B8%D1%87%D0%BD%D0%BE%D0%B5_%D1%88%D0%B8%D1%84%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5). У неё не существует достаточно хороших решений. Оригинальная схема PGP позволяет решить пользователю, использовать ли схему проверки сертификатов, в то время как большинство других инфраструктур открытых ключей требуют проверки каждого сертификата.

# Защита электронной почты (S/MIME)

**S/MIME** (Secure/[Multipurpose Internet Mail Extensions](https://ru.wikipedia.org/wiki/MIME)) — стандарт для [шифрования](https://ru.wikipedia.org/wiki/%D0%A8%D0%B8%D1%84%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) и [подписи](https://ru.wikipedia.org/wiki/%D0%AD%D0%BB%D0%B5%D0%BA%D1%82%D1%80%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D1%86%D0%B8%D1%84%D1%80%D0%BE%D0%B2%D0%B0%D1%8F_%D0%BF%D0%BE%D0%B4%D0%BF%D0%B8%D1%81%D1%8C) в [электронной почте](https://ru.wikipedia.org/wiki/%D0%AD%D0%BB%D0%B5%D0%BA%D1%82%D1%80%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D0%BF%D0%BE%D1%87%D1%82%D0%B0) с помощью [открытого ключа](https://ru.wikipedia.org/wiki/%D0%9A%D1%80%D0%B8%D0%BF%D1%82%D0%BE%D0%B3%D1%80%D0%B0%D1%84%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B0%D1%8F_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D1%81_%D0%BE%D1%82%D0%BA%D1%80%D1%8B%D1%82%D1%8B%D0%BC_%D0%BA%D0%BB%D1%8E%D1%87%D0%BE%D0%BC).

S/MIME предназначена для обеспечения криптографической безопасности электронной почты. Обеспечиваются аутентификация, целостность сообщения и гарантия сохранения авторства, безопасность данных (посредством шифрования).

Большая часть современных почтовых программ поддерживает S/MIME.

Первая версия S/PEPE была разработана в 1995 году рядом антивирусных компаний. Это была одна из нескольких спецификаций для обеспечения безопасности сообщений. Pretty Good Privacy (PGP) является примером другой, спецификацией для обеспечения безопасности сообщений. Первая версия S/MIME не была признана как единый стандарт для защиты сообщений. В 1998 году ситуация начала меняться.

В отличие от первой версии, S/PEPE второй версии был представлен в Internet Engineering Task Force (IETF) для рассмотрения в качестве стандарта. Благодаря этому шагу, S/PEPE выделился среди остальных сертификатов защиты сообщений. S/MIME версии 3.0, добился широкого признания в качестве стандарта для защиты сообщений. S/PEPE версии 3.0 поддерживают продукты Microsoft.

Обычный личный сертификат удостоверяет личность владельца только путём связывания воедино почтового адреса и сертификата. Он не удостоверяет ни имени, ни рода деятельности. Более полное удостоверение можно получить, обратившись к специализированным ЦПВЛС, которые предоставляют дополнительные (нотариально эквивалентные) услуги.

В зависимости от политик ЦПВЛС, ваш сертификат и всё его содержимое могут быть открыто опубликованы для ознакомления и проверки. В таком случае, ваше имя и почтовый адрес становятся доступными для всех, в том числе и для поиска. Другие ЦС могут публиковать только серийные номера. Это необходимый минимум для обеспечения целостности инфраструктуры открытого ключа.

Корректное использование стандарта S/MIME накладывает некоторые ограничения на применение традиционных приложений электронной почты и рабочей среды, в которой они используются:

Отправителю и получателю необходимо согласовать применение клиентских приложений электронной почты, которые поддерживают данный стандарт. В противном случае, почтовый клиент получателя отображает в письмах файлы-вложения «smime.p7s», которые получатель обычно не может корректно интерпретировать.

Эффективное применение S/MIME требует комплексного подхода к обеспечению безопасности. Это означает, что необходимо обеспечивать защиту сообщений не только по пути следования от отправителя к получателю, но и в рабочей среде отправителя и получателя. В частности, несоблюдение этого требования может привести к утечке конфиденциальной информации либо несанкционированной модификации сообщений, равно как и компрометации секретных ключей непосредственно на компьютерах пользователей.

S/MIME принципиально несовместим с [веб-почтой](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D0%BF%D0%BE%D1%87%D1%82%D0%B0). Это обусловлено тем, что [криптография открытых ключей](https://ru.wikipedia.org/wiki/%D0%9A%D1%80%D0%B8%D0%BF%D1%82%D0%BE%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D1%81_%D0%BE%D1%82%D0%BA%D1%80%D1%8B%D1%82%D1%8B%D0%BC_%D0%BA%D0%BB%D1%8E%D1%87%D0%BE%D0%BC), лежащая в основе стандарта S/MIME, обеспечивает защиту [конфиденциальности](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BD%D1%84%D0%B8%D0%B4%D0%B5%D0%BD%D1%86%D0%B8%D0%B0%D0%BB%D1%8C%D0%BD%D0%BE%D1%81%D1%82%D1%8C) и [целостности](https://ru.wikipedia.org/wiki/%D0%A6%D0%B5%D0%BB%D0%BE%D1%81%D1%82%D0%BD%D0%BE%D1%81%D1%82%D1%8C_%D0%B8%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%86%D0%B8%D0%B8) сообщений на пути от отправителя до получателя. В то же время [конфиденциальность](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BD%D1%84%D0%B8%D0%B4%D0%B5%D0%BD%D1%86%D0%B8%D0%B0%D0%BB%D1%8C%D0%BD%D0%BE%D1%81%D1%82%D1%8C) и [целостность](https://ru.wikipedia.org/wiki/%D0%A6%D0%B5%D0%BB%D0%BE%D1%81%D1%82%D0%BD%D0%BE%D1%81%D1%82%D1%8C) сообщений недостижимы при традиционном использовании веб-почты, так как провайдер сервиса веб-почты имеет возможность как читать сообщения, так и модифицировать их. В то же время попытки использования подписи или шифрования сообщений на стороне сервера являются [компрометацией](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%80%D0%BE%D0%BC%D0%B5%D1%82%D0%B0%D1%86%D0%B8%D1%8F_(%D0%BA%D1%80%D0%B8%D0%BF%D1%82%D0%BE%D0%B3%D1%80%D0%B0%D1%84%D0%B8%D1%8F)) секретных ключей пользователей. Кроме того, основное преимущество веб-почты - её доступность с любого компьютера, где есть веб-обозреватель - противоречит требованию контроля защищенности рабочей среды при использовании S/MIME.

Цифровые подписи являются наиболее часто используемые службой S/MIME. Как следует из названия, цифровые подписи это цифровой аналог традиционных, правовых подписей на бумажном документе. Как и юридические подписи, цифровые подписи обеспечивают следующие функции безопасности:

* **Неподдельность** (уникальность) подписи. Концепция неподдельности является наиболее знакомой в контексте бумажных контрактов: подписанный договор является юридическим документом. Цифровые подписи обеспечивают те же функции, и все чаще в некоторых областях, признаются юридически обязательными. Потому что в SMTP не предусмотрено средств аутентификации, он не может обеспечить строгое выполнение обязательств.
* **Целостность** данных. Дополнительная услуга безопасности, которую обеспечивает цифровая подпись, является целостность данных. Целостность данных является результатом специфических операций. С услугой целостности данных, когда получатель сообщения, подписанное цифровой подписью, проверяет цифровую подпись, получатель может быть уверен, что получил то самое сообщение, которое было отправлено, и не было изменено в процессе передачи. Таким образом, цифровые подписи в состоянии обеспечить гарантию того, чего не могут обеспечить обычные подписи на бумаге, т.к. бумажный документ, может быть изменен после его подписания.
* **Проверка подлинности** подписи служит для проверки идентичности. Она проверяет ответ на вопрос "кто вы". Потому что нет никакого способа узнать, кто на самом деле отправил сообщение. Проверка подлинности цифровой подписью решает эту проблему, позволяя получателю знать, кто отправил сообщение.

# Архитектура AAA систем.

Несанкционированный доступ, а также возможность фальсификации и обмана в сетевой среде дают нарушителям потенциальную возможность получения доступа к сетевому оборудованию и сетевым службам. Архитектура ААА позволяет сильно ограничить возможности нарушителей, оставляя законным пользователям сети право иметь доступ к сетевым ресурсам.

Архитектура защиты ААА

Защита сетевого доступа — независимо от того, рассматривается она в применении к территориальной сети предприятия, удаленному доступу или Internet — имеет модульную архитектуру, состоящую из следующих трех компонентов:

* **Аутентификация.** Требует от пользователей доказательства того, что они действительно являются теми, за кого себя выдают, например, посредством ввода имени пользователя и пароля, использования системы запросов/подтверждений, идентификационных карт или какого-то другого метода.  
  *"Я — пользователь student, и мой пароль validateme доказывает это"*
* **Авторизация.** После аутентификации пользователя сервис авторизации решает, к каким ресурсам разрешается доступ данному пользователю и какие действия разрешается ему выполнять.  
  *"Пользователь student может иметь доступ к узлу NT\_Server посредством Telnet"*
* **Аудит.** Запись того, что пользователь действительно делал, к чему имел доступ и в течение какого времени, осуществляется с целью учета, контроля и выяснения стоимости. С помощью аудита можно проследить за тем, как используются сетевые ресурсы. Аудит может быть применен для анализа практики сетевого доступа и обнаружения сетевых вторжений.  
  *"Пользователь student получал доступ к узлу NT\_Server посредством Telnet 15 раз"*

Серверы защиты ААА

Средства ААА в продуктах Cisco поддерживают контроль доступа либо с помощью локальной базы данных на сервере сетевого доступа, либо с помощью удаленной базы данных зашиты, использующей протокол защиты ААА. Оба подхода имеют свои плюсы и минусы. В этом разделе мы рассмотрим применение средств ААА с локальной и удаленной базами данных зашиты, а также стандарты удаленной базы данных зашиты, поддерживаемые средствами ААА Cisco.

**ААА и локальная база данных защиты**

Если требуется обеспечить доступ к сети небольшому числу удаленных пользователей через один-два сервера сетевого доступа, можно хранить информацию об их именах и паролях на сервере сетевого доступа. Такой подход называют локальной аутентификацией, или аутентификацией с помощью локальной базы данных зашиты.

Ниже указаны особенности использования средств ААА с локальной базой данных зашиты:

* Локальная аутентификация подходит для малых сетей с небольшим числом удаленных пользователей и серверов сетевого доступа.
* Имена пользователей, пароли и параметры авторизации хранятся в локальной базе данных защиты на сервере сетевого доступа.
* Удаленные пользователи проходят аутентификацию и авторизацию с помощью локальной базы данных зашиты.
* Авторизация и аудит при использовании локальной базы данных зашиты имеют ограниченную поддержку.
* Контроль доступа с помощью локальной базы данных защиты позволяет сэкономить на установке и поддержке удаленной базы данных зашиты.
* Аутентификация с использованием локальной базы данных зашиты обычно выполняется следующим образом. Сначала с помощью команд ААА необходимо в локальной базе данных зашиты каждого сервера сетевого доступа указать соответствующие параметры для каждого из возможных пользователей. Процесс ААА состоит из следующих шагов:
* Удаленный пользователь устанавливает соединение РРР с сервером сетевого доступа.
* Сервер сетевого доступа запрашивает у пользователя имя и пароль.
* Сервер сетевого доступа выполняет аутентификацию имени и пароля с помощью локальной базы данных.
* Сервер сетевого доступа выполняет процедуру авторизации, в результате чего пользователь получает право доступа к сетевым сервисам и другим ресурсам в соответствии со значениями, заданными в локальной базе данных.
* Сервер сетевого доступа следит за трафиком пользователя и создает записи аудита в соответствии со значениями, заданными в локальной базе данных.

**ААА и удаленная база данных защиты**

По мере роста сети все более остро встает вопрос о необходимости использования удаленной базы данных защиты, которая обеспечивала бы информацию об именах пользователей и паролях всем серверам сетевого доступа и маршрутизаторам сети. Удаленная база данных защиты размешается на сервере защиты. Удаленную базу данных защиты удобно использовать тогда, когда имеется большое число серверов сетевого доступа, контролирующих доступ к сети. Такая база данных позволяет централизованно управлять файлами профилей (т.е. параметрами доступа) удаленных пользователей, что избавляет от необходимости менять файлы профилей каждого удаленного пользователя на всех серверах сетевого доступа. Удаленная база данных зашиты помогает создать и реализовать согласованную политику защиты удаленного доступа для всех подразделений корпорации.

Рассмотрим особенности использования средств ААА с удаленной базой данных защиты:

* Аутентификация с помощью удаленной базы данных защиты оптимальна для средних и больших сетей с большим числом удаленных пользователей и множеством серверов сетевого доступа, когда затраты на содержание сервера защиты могут быть оправданы.
* Имена пользователей, пароли и параметры авторизации централизованно хранятся в удаленной базе данных защиты на сервере защиты.
* Удаленные пользователи проходят процедуры аутентификации и авторизации с помощью удаленной базы данных защиты.
* Авторизация и аудит поддерживаются сервером сетевого доступа с использованием удаленной базы данных защиты.
* Удаленная база данных защиты может использоваться для контроля доступа к серверу сетевого доступа или к сети через сервер сетевого доступа. Некоторые протоколы удаленной базы данных защиты поддерживают контроль доступа к маршрутизаторам, коммутаторам Ethernet и брандмауэрам. Удаленная база данных защиты применяется для контроля доступа к сетевому оборудованию, поддерживающему стандартные протоколы удаленного доступа.

Централизованный контроль посредством удаленной базы данных защиты позволяет сэкономить средства, избавляя от необходимости управлять каждым сервером сетевого доступа в отдельности. Для защиты базы данных необходимо, чтобы содержащий ее хост был защищен с максимальной степенью надежности.

Аутентификация с использованием удаленной базы данных защиты обычно выполняется так. Сначала необходимо заполнить локальную базу данных защиты каждого сервера сетевого доступа, описав соответствующие параметры для каждого из возможных пользователей.

# Протокол RADIUS

RADIUS ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *Remote Authentication in Dial-In User Service*) — [протокол](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D1%82%D0%BE%D0%BA%D0%BE%D0%BB_%D0%BF%D0%B5%D1%80%D0%B5%D0%B4%D0%B0%D1%87%D0%B8_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85) для реализации аутентификации, авторизации и сбора сведений об использованных ресурсах, разработанный для передачи сведений между центральной платформой и оборудованием. Этот протокол применялся для системы [тарификации](https://ru.wikipedia.org/wiki/%D0%A2%D0%B0%D1%80%D0%B8%D1%84) использованных ресурсов конкретным пользователем/абонентом. Центральная платформа и оборудование [Dial-Up](https://ru.wikipedia.org/wiki/Dial-Up) доступа ([NAS](https://ru.wikipedia.org/w/index.php?title=Network_Access_Server&action=edit&redlink=1)[[1]](https://ru.wikipedia.org/wiki/RADIUS#cite_note-1) с системой автоматизированного учёта услуг ([биллинга](https://ru.wikipedia.org/wiki/%D0%91%D0%B8%D0%BB%D0%BB%D0%B8%D0%BD%D0%B3))),

RADIUS используется как [протокол AAA](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D1%82%D0%BE%D0%BA%D0%BE%D0%BB_AAA)

Однако протокол предусматривает передачу данных любого типа, что реализуется посредством **VSA** (**V**endor **S**pecific **A**ttributes).

Будучи частью биллинговой системы, RADIUS-сервер является интерфейсом взаимодействия с телекоммуникационной системой/сервером (например [маршрутизатором](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%80%D1%88%D1%80%D1%83%D1%82%D0%B8%D0%B7%D0%B0%D1%82%D0%BE%D1%80) или [коммутатором](https://ru.wikipedia.org/wiki/Softswitch)) и может реализовывать для такой системы следующие сервисы:

**Общие**

* Создание и хранение учётных записей пользователей (абонентов)
* Управление учётной записью пользователя (абонента) из персонального интерфейса (например веб-кабинета)
* Создание карточек доступа (логин/PIN-код) для предоставления услуг, с некоторым лимитом действия (Dial-Up доступа в Интернет и карточной IP-телефонии)
* Ручная и автоматическая блокировка учётной записи абонента по достижению заданного критерия или лимита
* Сбор и анализ статистической информации о сессиях пользователя и всей обслуживаемой системы (в том числе [CDR](https://ru.wikipedia.org/wiki/Call_Detail_Record))
* Создание отчётов по различным статистическим параметрам
* Создание, печать и отправка счетов к оплате
* Аутентификация всех запросов в RADIUS-сервер из обслуживаемой системы (поле Secret)

**Аутентификация**

* Аутентентификация и авторизация через RADIUS-сервер
* Проверка учётных данных пользователя (в том числе шифрованных) по запросу обслуживаемой системы
* Авторизация
* Выдача состояния блокировки учётной записи пользователя
* Выдача разрешения к той или иной услуге
* Сортировка данных на основе анализа статистической информации (например динамическая маршрутизация) и выдача результата сортировки по запросу

**Учёт (Accounting)**

* Онлайн-учёт средств абонента: уведомления о начале и конце сессии со стороны обслуживаемой системы
* Промежуточные сообщения о продолжении сессии (Interim-пакеты)
* Автоматическое принудительное завершение действия сессии на обслуживаемой системе в рамках услуги (packet of disconnection)

BOOT message — специальный пакет, который отправляется телекоммуникационной системой на RADIUS-сервер при запуске (перезапуске) системы, с целью принудительного завершения всех сессий

В нас­то­ящее вре­мя про­токол RA­DI­US ис­поль­зу­ет­ся для дос­ту­па к вир­ту­аль­ным част­ным се­тям (VPN), точ­кам бесп­ро­вод­но­го (Wi-Fi) дос­ту­па, Et­hernet ком­му­тато­рам, DSL и дру­гим ти­пам се­тево­го дос­ту­па. Бла­года­ря отк­ры­тос­ти, прос­то­те внед­ре­ния, пос­то­ян­но­му усо­вер­шенс­тво­ванию, про­токол RA­DI­US сей­час яв­ля­ет­ся фак­ти­чес­ки стан­дартом для уда­лен­ной аутен­ти­фика­ции.

Для определения принципа работы протокола RA­DI­US необходимо рассмотреть рисунок, приведённый выше[[*где?*](https://ru.wikipedia.org/wiki/%D0%92%D0%B8%D0%BA%D0%B8%D0%BF%D0%B5%D0%B4%D0%B8%D1%8F:%D0%98%D0%B7%D0%B1%D0%B5%D0%B3%D0%B0%D0%B9%D1%82%D0%B5_%D0%BD%D0%B5%D0%BE%D0%BF%D1%80%D0%B5%D0%B4%D0%B5%D0%BB%D1%91%D0%BD%D0%BD%D1%8B%D1%85_%D0%B2%D1%8B%D1%80%D0%B0%D0%B6%D0%B5%D0%BD%D0%B8%D0%B9)]. Ноутбуки и IP-телефон представляют устройства пользователя, с которых необходимо выполнить аутентификацию и авторизацию на сетевых серверах доступа (NAS): точке Wi-Fi доступа, маршрутизаторе, VPN-сервере и IP АТС. На рисунке приведены несколько из возможных вариантов NAS, существуют и другие сетевые устройства доступа. RADIUS реализуется в виде интерфейса между NAS (RADIUS-клиент) и RADIUS-сервером — программным обеспечением, устанавливаемым на компьютере (сервере) или ином специализированном устройстве. Сервер взаимодействует с устройством пользователя не напрямую, а только через сетевой сервер доступа.

Пользователь отправляет запрос на сетевой сервер для получения доступа к определённому сетевому ресурсу, используя сертификат доступа. Сертификат отправляется на сервер через сетевой протокол канального уровня (например, [PPP](https://ru.wikipedia.org/wiki/PPP) в случае коммутируемого доступа, [DSL](https://ru.wikipedia.org/wiki/DSL) в случае использования соответствующих модемов и т.д.). NAS, в свою очередь, посылает сообщение запроса доступа на RADIUS-сервер (RADIUS Access Request). Запрос включает сертификаты доступа, представленные в виде имени пользователя и пароля или сертификата безопасности, которые получены от пользователя. Запрос может содержать дополнительные параметры: сетевой адрес устройства пользователя, телефонный номер, информацию о физическом адресе, с которого пользователь взаимодействует с NAS.

Сервер проверяет информацию на корректность, используя схемы аутентификации:

* PAP (Pass­word Aut­henti­cati­on Pro­tocol) (RFC1334)– прос­той аутен­ти­фика­ци­он­ный про­токол, ко­торый ис­поль­зу­ет­ся для аутен­ти­фика­ции поль­зо­вате­ля по от­но­шению к се­тево­му сер­ве­ру дос­ту­па (NAS). РАР ис­поль­зу­ет­ся РРР про­токо­лом. Прак­ти­чес­ки все сер­ве­ра дос­ту­па под­держи­ва­ют РАР. РАР пе­реда­ет не­зашиф­ро­ван­ный па­роль че­рез сеть и, сле­дова­тель­но, яв­ля­ет­ся не­защи­щен­ным про­токо­лом. По­это­му РАР, обыч­но, ис­поль­зу­ет­ся в том слу­чае, ког­да сер­вер не под­держи­ва­ет за­щищен­ные про­токо­лы, та­кие как СНАР, ЕАР и т.п.
* CHAP (англ. Chal­lenge Hand­sha­ke Aut­henti­cati­on Pro­tocol) ([RFC 1994](https://tools.ietf.org/html/rfc1994)) — ши­роко расп­рос­тра­нён­ный ал­го­ритм про­вер­ки под­линнос­ти, пре­дус­матри­ва­ющий пе­реда­чу не са­мого па­роля поль­зо­вате­ля, а кос­венных све­дений о нём. При ис­поль­зо­вании CHAP сер­вер уда­лен­но­го дос­ту­па отп­рав­ля­ет кли­ен­ту стро­ку зап­ро­са. На ос­но­ве этой стро­ки и па­роля поль­зо­вате­ля кли­ент вы­чис­ля­ет хеш-код MD5 (Mes­sa­ge Di­gest-5) и пе­реда­ет его сер­ве­ру. Хеш-функ­ция яв­ля­ет­ся ал­го­рит­мом од­носто­рон­не­го (не­об­ра­тимо­го) шиф­ро­вания, пос­коль­ку зна­чение хеш-функ­ции для бло­ка дан­ных вы­чис­лить лег­ко, а оп­ре­делить ис­ходный блок по хеш-ко­ду с ма­тема­тичес­кой точ­ки зре­ния не­воз­можно за при­ем­ле­мое вре­мя. Сер­вер, ко­торо­му дос­ту­пен па­роль поль­зо­вате­ля, вы­пол­ня­ет те же са­мые вы­чис­ле­ния и срав­ни­ва­ет ре­зуль­тат с хеш-ко­дом, по­лучен­ным от кли­ен­та. В слу­чае сов­па­дения учёт­ные дан­ные кли­ен­та уда­лён­но­го дос­ту­па счи­та­ют­ся под­линны­ми.
* MD5 (Mes­sa­ge-Di­gest al­go­rithm 5) ([RFC 1321](https://tools.ietf.org/html/rfc1321)) — ши­роко ис­поль­зу­емая крип­тогра­фичес­кая функ­ция с 128 би­товым хе­шем. Най­ден ряд уяз­ви­мос­тей в ал­го­рит­ме MD5, в си­лу че­го в США де­пар­та­мент U. S. De­part­ment of Ho­meland Se­curi­ty не ре­комен­ду­ет ис­поль­зо­вание MD5 в бу­дущем, и для боль­шинс­тва пра­витель­ствен­ных при­ложе­ний c 2010 го­да США тре­бу­ет­ся пе­рей­ти на се­мей­ство ал­го­рит­ма SHA-2.
* Про­токол EAP (Ex­tensib­le Aut­henti­cati­on Pro­tocol) ([RFC 3748](https://tools.ietf.org/html/rfc3748)) поз­во­ля­ет про­верять под­линность при подк­лю­чени­ях уда­лен­но­го дос­ту­па с по­мощью раз­личных ме­ханиз­мов про­вер­ки под­линнос­ти. Точ­ная схе­ма про­вер­ки под­линнос­ти сог­ла­совы­ва­ет­ся кли­ен­том уда­лен­но­го дос­ту­па и сер­ве­ром, вы­пол­ня­ющим про­вер­ку под­линнос­ти (им мо­жет быть сер­вер уда­лен­но­го дос­ту­па или RA­DI­US сер­вер). По умол­ча­нию в марш­ру­тиза­цию и уда­лен­ный дос­туп вклю­чена под­держ­ка про­токо­лов EAP-TLS и MD5-Chal­lenge (MD5-за­дача). Подк­лю­чение дру­гих мо­дулей ЕАР к сер­ве­ру, ис­поль­зу­юще­му марш­ру­тиза­цию и уда­лен­ный дос­туп, обес­пе­чива­ет под­держ­ку дру­гих ме­тодов ЕАР. Про­токол EAP поз­во­ля­ет вес­ти сво­бод­ный ди­алог меж­ду кли­ен­том уда­лен­но­го дос­ту­па и сис­те­мой про­вер­ки под­линнос­ти. Та­кой ди­алог сос­то­ит из зап­ро­сов сис­те­мы про­вер­ки под­линнос­ти на не­об­хо­димую ей ин­форма­цию и от­ве­тов кли­ен­та уда­лен­но­го дос­ту­па. Нап­ри­мер, ког­да про­токол EAP ис­поль­зу­ет­ся с ге­нера­тора­ми ко­дов дос­ту­па, сер­вер, вы­пол­ня­ющий про­вер­ку под­линнос­ти, мо­жет от­дель­но зап­ра­шивать у кли­ен­та уда­лен­но­го дос­ту­па имя поль­зо­вате­ля, иден­ти­фика­тор и код дос­ту­па. Пос­ле от­ве­та на каж­дый та­кой зап­рос кли­ент уда­лен­но­го дос­ту­па про­ходит оп­ре­делен­ный уро­вень про­вер­ки под­линнос­ти. Ког­да на все зап­ро­сы бу­дут по­луче­ны удов­летво­ритель­ные от­ве­ты, про­вер­ка под­линнос­ти кли­ен­та уда­лен­но­го дос­ту­па ус­пешно за­вер­ша­ет­ся.