Network: Часть 5. Сетевой уровень

24.10.2026 07:33 edited 24.10.2026 07:33

Эпоха простых решений

В январе 1974 года Винт Серф и Боб Кан опубликовали в IEEE Transactions on Communications статью «A Protocol for Packet Network Intercommunication»: разнородные сети с разными адресами, размерами пакетов и надежностью должны были обмениваться данными через единый промежуточный протокол. Полвека спустя на этой идее держится вся мировая связь. По состоянию на октябрь 2026 года глобальная таблица маршрутизации BGP содержит 1 085 491 IPv4-префикс, объявленный примерно 79 744 автономными системами, — и каждый пакет между ними обрабатывается механизмами сетевого уровня: адресацией, поиском маршрута, пересылкой, управлением перегрузкой и фрагментацией.

Парадокс сетевого уровня в том, что самый важный уровень глобальной сети спроектирован намеренно ненадежным. IP не гарантирует доставку, порядок и целостность полезной нагрузки и не защищает от подделки адреса отправителя. Современный магистральный маршрутизатор пересылает сотни терабит в секунду, а решения о том, куда направить каждый пакет, принимаются протоколом, который в 1981 году описывал сеть из нескольких сотен хостов. И тем не менее эта «глупая» сеть пережила рост в миллионы раз, исчерпание собственного адресного пространства, три десятилетия попыток заменить ее новым протоколом и бесчисленные аварии.

Этот парадокс порождает ряд критических вопросов для любого, кто проектирует или эксплуатирует сети:

1. **Архитектурный вопрос.** Почему Интернет выбрал дейтаграммную модель без установления соединения — в противовес виртуальным каналам, на которых настаивала телекоммуникационная индустрия 1970–1990-х? Чем пришлось заплатить?
2. **Вопрос масштабируемости.** Таблица маршрутизации выросла с сотен записей в ARPANET до миллиона с лишним префиксов. Какие механизмы — иерархия, агрегация, CIDR — не дали ей поглотить всю память маршрутизаторов, и почему она продолжает расти, хотя могла бы быть на 44% меньше?
3. **Экономический вопрос.** Интернет маршрутизируется не по географии, а по экономике: BGP — протокол политик, а не метрик, и путь между двумя зданиями в одном городе может пролегать через другой континент, потому что так дешевле по контрактам провайдеров.
4. **Вопрос перегрузки.** В октябре 1986 года пропускная способность участка ARPANET упала в тысячу раз — с 32 Кбит/с до 40 бит/с. Коллапс перегрузки и «раздувание

буферов» (bufferbloat) до сих пор определяют, почему видеозвонок «заикается» при работающем интернете.

5. **Вопрос безопасности.** В сетевом уровне нет встроенной аутентификации: любой может объявить чужие адреса или подделать адрес отправителя. Это привело к перехватам маршрутов, отключениям целых стран и кражам криптовалюты, ответ индустрии — RPKI, фильтрация и MANRS — внедрен лишь частично.
6. **Вопрос эволюции.** Адресов IPv4 не хватает с 1990-х, их хватило еще на тридцать лет — благодаря NAT, «костылю», который никто не планировал делать постоянным. Переход на IPv6 идет с 1998 года и только к 2026-му достиг половины пользователей.

Дальше мы пройдем весь сетевой уровень: от исторических корней и двух конкурирующих моделей сети — через IP-адресацию, ARP, внутреннее устройство маршрутизатора и алгоритмы маршрутизации — к BGP и экономике Интернета, перегрузке, фрагментации, NAT, anycast, MPLS и SDN, туннелям и безопасности. В финале проследим путь одного пакета от домашнего ноутбука до сервера в дата-центре. Материал опирается на стандарты IETF, классическую литературу по сетям и практику эксплуатации современных сетей.

Рождение сетевого уровня: от коммутации пакетов к «катенету»

Коммутация пакетов: Баран, Дэвис и спор о приоритете

Идея, лежащая в основе сетевого уровня, старше самого Интернета. В начале 1960-х годов Пол Баран из RAND Corporation искал способ построить связь, которая пережила бы ядерный удар. В одиннадцатитомном отчете «On Distributed Communications» (1960–1964) он предложил распределенную сеть узлов, передающую сообщения небольшими блоками по принципу «горячей картошки»: каждый узел немедленно пересылает блок дальше. AT&T идею отвергла — телефонная монополия не увидела причин ломать проверенную модель коммутации каналов.

Независимо от Барана, в 1965 году Дональд Дэвис из британской Национальной физической лаборатории (NPL) предложил аналогичную схему, ввел сам термин «пакет» и построил реальную сеть NPL Mark I (1967–1970). Спор о том, кому принадлежит приоритет — Барану, Дэвису или Леонарду Клейнроку из UCLA, чья теория массового обслуживания легла в основу математического описания таких сетей, — тянется десятилетиями. Важно другое: к концу 1960-х идея была «в воздухе», и реализовали ее американцы.

В 1969 году агентство ARPA поручило компании BBN построить ARPANET. Первый процессор передачи сообщений (IMP — предок маршрутизатора) на базе Honeywell DDP-516 был доставлен в UCLA 2 сентября 1969 года, команду BBN возглавлял Фрэнк Харт. 29 октября 1969 года студент Чарли Клайн попытался передать слово LOGIN из Лос-Анджелеса в Стэнфордский исследовательский институт, система успела отправить буквы L и O, после чего рухнула. Первым сообщением в истории Интернета стало «Lo».

Первые IMP решали задачи, которые сегодня мы назвали бы задачами сетевого уровня: выбирали маршрут для каждого пакета по таблицам, строившимся распределенным алгоритмом на основе векторов расстояний с метрикой «число переходов». Алгоритм страдал теми же болезнями, что и современный RIP, — медленной сходимостью и «счетом до бесконечности», — и в 1979 году был заменен алгоритмом на основе состояния каналов, прообразом OSPF.

Cyclades Луи Пуже и рождение дейтаграммы

Решающий концептуальный шаг сделал не американец, а француз. Луи Пуже, руководивший сетью Cyclades (1971–1974), перевернул привычную схему: вместо того чтобы сеть отвечала за надежную доставку (как в X.25 и в самой ARPANET), Cyclades переложила эту обязанность на конечные хосты, а сеть передавала простые неподтверждаемые пакеты — дейтаграммы (само слово «datagram» ввел Пуже). Маршрутизаторы не хранили состояния о потоках: каждый пакет обрабатывался независимо, что делало сеть простой, дешевой и устойчивой к сбоям узлов.

В 1974 году Пуже написал записку о «катенете» (catenet — «цепочечная сеть»), системе объединенных сетей, в которой пакеты путешествуют между разнородными сетями через шлюзы. Записка легла на стол Винту Серфу и Бобу Кану, которые как раз бились над объединением ARPANET, пакетных радиосетей PRNET и спутниковой сети SATNET. Идеи Пуже — дейтаграмма, «глупая» сеть и «умные» хосты — напрямую повлияли на архитектуру TCP/IP.

Сама Cyclades погибла: французское государство сделало ставку на сети X.25 (Transpac, обслуживавшая впоследствии знаменитый Minitel), и к концу 1970-х эксперимент Пуже закрыли. Победа X.25 во Франции оказалась пирровой — мир выбрал модель Cyclades.

ARPANET, «катенет» и протокол 1974 года

Статья Серфа и Кана 1974 года описывала единый протокол межсетевого взаимодействия: TCP, включавший тогда и маршрутизацию дейтаграмм, и надежную доставку. К 1978 году стало ясно, что задачи нужно разделить, и Серф, Кан, Постел и Коэн пришли к двухуровневой схеме — TCP для соединений и IP для доставки дейтаграмм через границы сетей. Узлы, соединяющие сети, в RFC того времени назывались шлюзами (gateways), термин «маршрутизатор» вошел в обиход позже.

Ключевым был принцип «согласования судеб» (fate sharing), описанный позже Дейвом Кларком: состояние соединения хранится на конечных хостах, там же, где его участники. Если промежуточный узел гибнет, соединение не теряется безвозвратно: хосты живы, состояние у них, и они могут восстановить связь по новому пути. Для сети, создававшейся под сценарии военных отказов, это было важнее любой оптимизации производительности.

RFC 791 и «День флага» 1 января 1983 года

В сентябре 1981 года вышел набор документов, определивший Интернет на следующие сорок лет: RFC 791 (IP), RFC 792 (ICMP) и RFC 793 (TCP), отредактированные Джоном Постелом. А 1 января 1983 года ARPANET перевели с NCP на TCP/IP — переход, вошедший в историю как Flag Day («День флага»). Постел готовил его годами, рассылая администраторам хостов письма с обещанием отключить всех, кто не обновится к сроку, операция прошла на удивление гладко. Хостов тогда было несколько сотен — сравните с 2026 годом, когда любая подобная замена протокола растягивается на десятилетия.

Философия дизайна: аргумент «конец-в-конец» и песочные часы

В 1984 году Джером Солтзер, Дэвид Рид и Дэвид Кларк сформулировали «аргумент конец-в-конец» (end-to-end argument): функцию можно полностью и корректно реализовать только на конечных узлах, поэтому помещать ее в глубь сети имеет смысл лишь как оптимизацию. Надежность, порядок доставки, безопасность — обязанности краев сети, а не ее ядра. Отсюда — «глупая сеть и умные хосты», принцип, до сих пор отличающий Интернет от телефонии.

В 1988 году Дэвид Кларк опубликовал «The Design Philosophy of the DARPA Internet Protocols», разложив цели архитектуры по приоритетам: главное — работа в условиях отказов (fate sharing), далее — поддержка разнородных сервисов, далее — экономическая эффективность и масштабируемость. Ту же идею выражает модель «песочных часов»: IP — узкая перемычка, над которой любое число приложений (HTTP, DNS, видео, еще не придуманные протоколы), а под ней — любая физика (Ethernet, Wi-Fi, оптика, спутники, даже — по первоапрельскому RFC 1149 — почтовые голуби). Узость перемычки — источник и силы Интернета, и его консерватизма: изменить IP сложнее, чем что-либо еще в истории техники.

Две модели сети: дейтаграммы против виртуальных каналов

Прежде чем разбирать IP, нужно понять фундаментальный архитектурный выбор, который не зря помещают в самое начало любого разговора о сетевом уровне: сеть может работать по одной из двух моделей, и от этого выбора зависят почти все ее свойства.

Дейтаграммная сеть

В дейтаграммной модели каждый пакет отправляется независимо и несет полный адрес назначения. Соединение не устанавливается: маршрутизаторы не хранят состояния о потоках и принимают решение по каждому пакету заново. Пакеты одного потока могут идти разными маршрутами, приходить в разном порядке, дублироваться или теряться — сеть этого не замечает и не исправляет.

Такая модель проста: маршрутизатору нужна лишь таблица «префикс назначения → следующий переход», а не таблица миллионов соединений. Она устойчива: при отказе канала трафик просто перераспределяется, и ничье «состояние» не теряется. И она идеально подходит для burst-трафика данных, плохо предсказуемого по природе. Именно эту модель выбрал IP.

Виртуальные каналы: X.25, Frame Relay, ATM

В модели виртуальных каналов (virtual circuit) перед передачей данных устанавливается соединение: сеть прокладывает путь и записывает в каждом транзитном узле состояние — «входящий идентификатор канала X преобразуется в исходящий Y на порту Z». Все пакеты соединения идут одним маршрутом, в порядке отправки, и сеть может гарантировать им полосу пропускания и задержку.

Так был устроен X.25 (стандарт CCITT 1976 года) — протокол телефонных администраций всего мира, с тяжелым канальным уровнем и потаксовой тарификацией. Так работал Frame Relay в 1990-х. Так был спроектирован ATM — амбициозная технология 1980–1990-х, обещавшая объединить голос, видео и данные в одной сети из ячеек фиксированного размера (53 байта) и которой телекоммуникационная индустрия прочила роль фундамента связи XXI века.

Почему Интернет выбрал дейтаграммы

Спор дейтаграмм и виртуальных каналов был не только техническим, но и культурно-экономическим. Телефонные компании мыслили соединениями: соединение — понятная единица тарификации, гарантированного качества и управления. Пионеры Интернета мыслили пакетами: состояние в сети считалось злом, потому что оно не переживает отказов и не масштабируется.

Практика рассудила спор. Во-первых, состояние в каждом узле на каждое соединение означало, что сеть знает о соединениях то, что знать не должна, — и ломается при перезагрузке узла. Во-вторых, виртуальные каналы плохо переносят аварийную перемаршрутизацию: при обрыве пути соединение перестраивается целиком, тогда как дейтаграммы просто начинают ходить другим маршрутом. В-третьих, экономика: маршрутизаторы Cisco на обычном железе оказались дешевле ATM-коммутаторов, а Ethernet — дешевле ATM-интерфейсов. В-четвертых, аргумент конец-в-конец: если надежность все равно нужна на хостах, дублировать ее в сети бессмысленно.

К середине 1990-х стало ясно, что IP+Ethernet выиграл, хотя остатки ATM доживали в магистралях провайдеров до 2000-х. Цена победы оказалась реальной: дейтаграммная модель не гарантирует качества обслуживания и порядка доставки, поэтому эти задачи пришлось решать «поверх и рядом» — TCP, очередями, DiffServ, инженерией трафика. И, по иронии истории, виртуальные каналы вернулись в Интернет под новыми именами — MPLS и сегментной маршрутизации.

Сравнение моделей

Сведем сравнение двух моделей. Дейтаграммная модель не требует установки соединения, переживает отказ маршрутизатора (пакеты находят другой путь) и не хранит состояния, но не гарантирует порядок и с трудом поддается обеспечению качества обслуживания. Сеть с виртуальными каналами соединение устанавливает, порядок гарантирует и ресурсы резервирует — но отказ узла рвет все проходящие через него соединения, а маршрутизаторы хранят таблицу на каждое соединение. Интернет выбрал дейтаграммы — вероятно, самое важное архитектурное решение в истории связи: все остальные свойства сетевого уровня, от «best effort» IP до политик BGP, вытекают из него.

Функции сетевого уровня

Сетевой уровень решает одну главную задачу: доставка пакета от источника к получателю через множество промежуточных сетей. Канальный уровень доставляет кадры внутри одной сети, сетевой сшивает такие сети в интерсеть (internet в общем смысле, Internet — как имя собственное). Из этой задачи следуют пять функций.

Адресация

Каждому интерфейсу устройства присваивается глобальный идентификатор — IP-адрес. В отличие от MAC-адреса, который «вшит» в сетевую карту и имеет смысл только внутри своей широковещательной зоны, IP-адрес иерархичен: старшие биты обозначают сеть, младшие — хост. Именно иерархичность позволяет агрегировать маршруты и масштабировать систему, плоское адресное пространство (как у MAC) для миллиона сетей не сработало бы.

Маршрутизация

Маршрутизация (routing) — вычисление путей: как пакету добраться от источника до получателя. Это «мозг» сетевого уровня: протоколы маршрутизации обмениваются информацией и строят таблицы маршрутов (RIB — Routing Information Base).

Пересылка

Пересылка (forwarding) — использование таблиц: пакет прибыл на входной интерфейс, и нужно за доли микросекунд решить, на какой выходной интерфейс его отправить. Это уже не логика, а операция на скорости: поиск по таблице (FIB — Forwarding Information Base), выполняемый аппаратно миллиарды раз в секунду. Разделение маршрутизации и пересылки — одна из ключевых идей уровня: медленный «мозг» строит таблицы, быстрые «рефлексы» по ним работают.

Управление перегрузкой

Если входящий трафик устойчиво превышает исходящую пропускную способность, очереди растут, задержки увеличиваются, пакеты теряются. Здесь важна одна мысль: перегрузка — не ошибка, а нормальное, ожидаемое состояние пакетной сети. Сетевой

уровень может предотвращать перегрузку (ограничивать и сглаживать трафик) и реагировать на нее (сигнализировать отправителям, сбрасывать или помечать пакеты). Механизмы — от «дырявого ведра» до ECN и современных AQM — разобраны ниже.

Фрагментация

Разные сети имеют разный максимальный размер пакета (MTU). Если пакет не помещается в MTU следующей сети, его делят на фрагменты и собирают на конечном узле. В IPv4 фрагментировать умеют промежуточные маршрутизаторы, в IPv6 это запрещено — фрагментирует только источник. Обе схемы порождают свои проблемы, вплоть до полноценных атак.

К этим пяти функциям на практике добавляют еще две: межсетевое взаимодействие (internetworking) как принцип — «внутри сети может быть что угодно, снаружи все видят IP», — и качество обслуживания (QoS), попытку заставить сеть, построенную по принципу «старайся как можешь», вести себя предсказуемо для чувствительного к задержкам трафика.

IP: протокол, на котором держится все

Заголовок IPv4

Разберем IPv4-заголовок (RFC 791) — по полям, в том порядке, в котором они лежат на проводе.

Version (4 бита) — версия протокола (4 или 6, поле кажется избыточным, пока не вспомнишь, что две версии сосуществуют уже тридцать лет). **IHL** (4 бита) — длина заголовка в 32-битных словах: минимум 5 (20 байт), максимум 15 (60) — запас под опции.

DSCP и ECN (вместе 8 бит, в RFC 791 поле называлось Type of Service) — качество обслуживания: 6 бит кода DSCP задают класс пакета, а 2 бита ECN позволяют сети сигнализировать о перегрузке, не сбрасывая пакет. История этого поля — история двух неудач и одного медленного успеха.

Total Length (16 бит) — полная длина пакета с заголовком, максимум 65 535 байт: из-за этого лимита «джамбо-кадры» в 9000 байт помещаются в IP легко, а объекты крупнее приходится фрагментировать.

Identification, Flags, Fragment Offset — трио полей фрагментации: Identification одинаков у всех фрагментов одного пакета, Flags содержат биты DF («не фрагментировать») и MF («дальше будут фрагменты»), Fragment Offset — смещение фрагмента в единицах по 8 байт.

TTL (8 бит) — время жизни. Формально в секундах, фактически с начала 1990-х — в переходах: каждый маршрутизатор уменьшает его на единицу, при нуле пакет уничтожается, а отправителю уходит ICMP Time Exceeded. Это предохранитель от петель маршрутизации.

Protocol (8 бит) — какому протоколу следующего уровня отдать полезную нагрузку: 1 — ICMP, 6 — TCP, 17 — UDP, 47 — GRE, 50 — ESP (IPsec). Это «демультиплексирование» сетевого уровня: ниточка, связывающая L3 с транспортным.

Header Checksum (16 бит) — контрольная сумма только заголовка, пересчитывается на каждом переходе, потому что изменился TTL. Целостность полезной нагрузки IP не проверяет — за это отвечают TCP/UDP своими контрольными суммами.

Source / Destination Address (по 32 бита) — отправитель и получатель, они не меняются на всем пути пакета — в отличие от MAC-адресов, переписываемых на каждом переходе.

Options — запись маршрута, метка времени, маршрутизация от источника. Последняя позволяла отправителю диктовать путь пакета — и была заброшена: она ломает независимую маршрутизацию и использовалась в атаках, поэтому провайдеры ее фильтруют.

Best effort: чего IP не гарантирует

Принцип работы IP называется best effort delivery — «доставка по мере сил». IP не гарантирует доставку, порядок, отсутствие дублей, целостность полезной нагрузки и защиту от подделки адреса источника. Все, что делает сеть, — честно пытается доставить каждый пакет по отдельности и, если не может, обычно сообщает об этом по ICMP (а часто не сообщает).

На вопрос «почему IP ненадежный?» правильный ответ таков: потому что надежность намеренно вынесли в TCP — чтобы сама сеть оставалась простой и масштабируемой. Это прямое следствие аргумента конец-в-конец: повторные передачи, упорядочивание и контроль перегрузки имеют смысл только между конечными хостами. UDP (протокол 17) демонстрирует обратную сторону выбора: приложению, которому надежность TCP не нужна (голос, видео, DNS, игры), IP дает ровно то, что есть, — «голую» доставку дейтаграмм.

Адресация IPv4: классы, подсети, CIDR

Первая схема адресации IPv4 была классовой (classful): первый октет определял класс — A (1–126, сеть плюс 24 бита хостов), B (128–191, плюс 16 бит), C (192–223, плюс 8 бит). Схема казалась элегантной и провалилась с треском: организации получали целые сети класса B на 65 тысяч адресов, даже если им нужно было триста. К началу 1990-х Интернет столкнулся с «двумя кризисами»: исчерпанием адресного пространства класса B и взрывным ростом таблицы маршрутизации.

Первым ответом стали подсети (subnetting, RFC 950, 1985): организация делила свою сеть на части с помощью маски, невидимой для внешнего мира. Вторым — и решающим — стал CIDR (Classless Inter-Domain Routing, RFC 1518/1519, сентябрь 1993 года): классы отменили вовсе, а префикс сети стал произвольной длины — `192.168.1.0/24` означает «первые 24

бита фиксированы, оставшиеся 8 бит — хосты». CIDR позволил выделять блоки «по размеру» и, что важнее, агрегировать маршруты: одну строку `10.0.0.0/8` в таблице вместо 256 строк `10.x.0.0/16`.

Сегодня маршрутизаторы работают не с адресами, а с префиксами. Поиск маршрута — это поиск самого длинного совпадающего префикса (longest prefix match): если в таблице есть и `10.0.0.0/8`, и `10.1.0.0/16`, пакет к `10.1.2.3` уйдет по второму маршруту, потому что он специфичнее. Из этого свойства вырос класс BGP-атак: объявление более специфичного префикса перехватывает трафик жертвы.

Частные адреса и специальные диапазоны

Часть пространства IPv4 зарезервирована под особые нужды: RFC 1918 (февраль 1996) выделил три диапазона для частных сетей — `10.0.0.0/8` (16,7 млн адресов), `172.16.0.0/12` (1 млн), `192.168.0.0/16` (65 тысяч, знакомые каждому `192.168.1.x` — оттуда). Дополнительно существуют `127.0.0.0/8` — петлевой адрес, `169.254.0.0/16` — link-local на случай отсутствия DHCP-сервера (RFC 3927), `100.64.0.0/10` — пространство для операторского NAT (RFC 6598), `224.0.0.0/4` — мультикаст и `240.0.0.0/4` — зарезервированный запас, который так и не понадобился.

Исчерпание IPv4: хронология и рынок адресов

О том, что 32-битных адресов (4,3 млрд) не хватит, говорили с конца 1980-х. Хронология конца растянулась на десятилетие: 3 февраля 2011 года IANA распределила последние пять блоков /8 между региональными регистраторами, дальше адреса заканчивались у APNIC (апрель 2011), RIPE NCC (сентябрь 2012), LACNIC (июнь 2014), ARIN (сентябрь 2015), а 25 ноября 2019 года RIPE NCC объявил о полном исчерпании пула IPv4 и открыл «лист ожидания».

Параллельно, в апреле 2011 года, Microsoft купила на аукционе банкротства Nortel 666 622 IPv4-адреса за 7,5 млн долларов — примерно по 11 долларов за адрес. Так родился рынок IPv4: адреса стали товаром. Цены росли до пика около 50 долларов за адрес к 2024 году, затем скорректировались, и к 2026 году диапазон составляет от 11 до 32 долларов в зависимости от размера блока. Экономический смысл ясен: дешевле купить адреса, чем мигрировать на IPv6, — и именно эта арифметика, а не технология, определяет темпы миграции.

IPv6: история создания и архитектура

Работа над преемником IPv4 началась в 1990-х под именем IPng (IP: Next Generation). Конкурировали три предложения: SIPP Стива Диринга, TUBA (TCP поверх OSI-протокола CLNP) и PIP Пола Фрэнсиса. В январе 1995 года RFC 1752 зафиксировал рекомендацию: развивать SIPP со 128-битными адресами. Спецификация вышла как RFC 1883 (декабрь 1995), затем RFC 2460 (1998), а действующая редакция — RFC 8200 (июль 2017). Номер «5» пропущен не случайно: его занял экспериментальный протокол потока ST-II (RFC 1190, 1990), из которого выросли многие идеи RSVP и мультикаста.

Что изменилось в IPv6:

- **Адрес — 128 бит**: $3{,}4 \cdot 10^{38}$ значений. Адресное пространство перестало быть ограничивающим ресурсом — им стала таблица маршрутизации.
- **Фиксированный 40-байтовый заголовок.** Контрольную сумму убрали (ее дублируют канальный и транспортный уровни), опции вынесли в цепочку расширенных заголовков, обрабатываемых только там, где нужно. Поля идут в порядке: Version, Traffic Class, Flow Label, Payload Length, Next Header, Hop Limit, затем 128-битные адреса источника и назначения — заголовок стал проще для аппаратной обработки.
- **Никакой фрагментации в транзите**: слишком большой пакет маршрутизаторы не режут, а возвращают источнику ICMPv6 Packet Too Big, фрагментирует только отправитель.
- **Поле Flow Label** (20 бит) — попытка вернуть в дейтаграммную сеть понятие «потока»: пакеты одного потока можно обрабатывать одинаково (например, не распрыгивать ECMP по разным путям).
- **TTL переименован в Hop Limit**, Type of Service — в Traffic Class, Protocol — в Next Header. **Широковещательной рассылки нет** — ее заменил мультикаст: даже ARP-подобные функции реализованы через специальные мультикаст-группы.

Адреса записываются в шестнадцатеричном виде через двоеточие, ведущие нули опускаются, последовательность нулевых групп заменяется на `::` — например, `2001:db8::1`. Стоит упомянуть автоконфигурацию SLAAC (RFC 4862): хост может получить адрес без DHCP, послушав объявления маршрутизатора, — но ранняя схема формирования адреса из MAC (EUI-64) позволяла отслеживать пользователя между сетями, и ее пришлось закрывать «расширениями приватности» (RFC 4941) — временными случайными адресами.

Почему IPv6 внедрялся тридцать лет

Мировой запуск IPv6 прошел 6 июня 2012 года (World IPv6 Launch, годом раньше был пробный World IPv6 Day) — Google, Facebook, Yahoo, Akamai и другие включили IPv6 «на постоянной основе». С тех пор прошло четырнадцать лет: по статистике Google, доля пользователей, приходящих к его сервисам по IPv6, в 2026 году составляет примерно 45–50%. Разброс по странам огромен: Индия и Франция перевалили за 60–70%, США — около половины, а многие корпоративные сети остаются практически чисто IPv4-ными.

Причины не технологические: IPv6 лучше IPv4 почти по всем инженерным параметрам. Проблема в экономике. NAT снял срочность миграции, dual-stack требует годами эксплуатировать две сети одновременно, прямой выгоды «здесь и сейчас» оператор не видит. Получился коллективный пат: все признают необходимость IPv6, но платить за срочный переход никто не хочет. Среди серверов миллиона крупнейших доменов IPv6 поддерживает уже большинство, но до «выключения IPv4» еще далеко. Сетевой уровень живет в состоянии перманентного апгрейда — ровно то, к чему архитектура «песочных часов» оказалась готова.

Механизмы перехода: dual stack, туннели, трансляция

За тридцать лет индустрия выработала три семейства механизмов сосуществования версий. **Двойной стек** (dual stack) — узел поддерживает IPv4 и IPv6 одновременно, выбор версии при подключении делает Happy Eyeballs (RFC 8305): клиент параллельно пробует оба адреса и остается на том, что сработал быстрее. **Туннелирование** — IPv6-пакеты упаковываются в IPv4 (6in4, 6to4, Teredo), исторически это позволило островам IPv6 общаться через океан IPv4. **Трансляция** — NAT64/DNS64 (RFC 6144–6147): DNS-резолвер синтезирует IPv6-адрес из IPv4, а шлюз транслирует пакеты. Крупнейший пример — T-Mobile US, работающий как IPv6-only с механизмом 464XLAT (RFC 6877).

ARP: как IP находит конкретное устройство

Разрешение адресов

Между сетевым и канальным уровнем есть зазор: IP знает адрес получателя (`192.168.1.10`), но Ethernet-кадру нужен MAC-адрес. Протокол ARP (Address Resolution Protocol, RFC 826, ноябрь 1982 года, Дэвид Пламмер) закрывает этот зазор одним вопросом: «Чей это IP? Скажи мне свой MAC».

Работает это так. Хост отправляет широковещательный кадр на MAC-адрес `FF:FF:FF:FF:FF:FF` — его слышат все устройства сегмента, кадр содержит вопрос «Кто имеет 192.168.1.10?». Устройство с таким IP отвечает уникастом: «Это я, мой MAC AA:BB:CC:DD:EE:FF». Ответ кэшируется в ARP-таблице (обычно на несколько минут), и следующие пакеты тому же адресату уходят без опроса. Есть и вариации: gratuitous ARP (узел сам объявляет «этот IP теперь мой» — используется при отказе кластеров и, к сожалению, при атаках), proxy ARP (маршрутизатор отвечает за чужие IP), RARP (обратный протокол для бездисковых станций, предок BOOTP и DHCP).

Ключевое ограничение: ARP работает только внутри одной широковещательной зоны. Роутеры не пропускают широковещательный трафик — иначе один ARP-запрос разбудил бы весь Интернет.

Когда цель в другой сети: шлюз по умолчанию

Если нужно отправить пакет на `8.8.8.8`, ARP не поможет: Google DNS не в моем сегменте. Хост сравнивает адрес назначения с маской своей подсети, понимает, что адрес «чужой», и отправляет пакет шлюзу по умолчанию (default gateway) — обычно домашнему роутеру: делает ARP-запрос к IP шлюза и вкладывает пакет в кадр, адресованный его MAC.

Отсюда — фундаментальное правило, отличающее понимающего инженера от заучившего определения: **IP-адрес назначения не меняется на всем пути, а MAC-адрес назначения переписывается на каждом переходе.** Пакет «для 8.8.8.8» на каждом участке упакован в кадр до ближайшего соседа: от ноутбука — до роутера, от роутера —

до маршрутизатора провайдера, и так далее, пока на последнем участке кадр не будет адресован MAC-у самого сервера. IP — сквозной адрес «куда», MAC — локальный адрес «кому передать сейчас».

Проблемы безопасности ARP и NDP в IPv6

ARP беззащитен по конструкции: он не аутентифицирует ответы, а многие стеки принимают «незапрошенные» обновления кэша. Отсюда ARP-спуфинг: атакующий в общем сегменте рассылает ответы «IP шлюза — это мой MAC» и становится «человеком посередине» для всех соседей. Защиты — статические записи, Dynamic ARP Inspection на управляемых коммутаторах, привязка IP-MAC-порт, — но они требуют управляемой инфраструктуры, которой в произвольной сети нет.

В IPv6 место ARP занял протокол NDP (Neighbor Discovery, RFC 4861), работающий поверх ICMPv6 и мультикаста: те же задачи (разрешение адресов, поиск маршрутизаторов, определение доступности соседей) и те же болезни. Попытка закрыть их криптографически (SEND, RFC 3971) не прижилась — вместо аутентификации администраторы полагаются на фильтрацию и изоляцию сегментов.

Внутри маршрутизатора

Маршрутизатор «пересылает пакеты» — эту фразу знает каждый, но что происходит внутри, когда пакет входит в порт магистрального устройства, разбирают редко. А между тем именно внутренняя архитектура маршрутизатора объясняет, почему сетевой уровень устроен так, а не иначе.

Обработка пакета шаг за шагом

Пакет, пришедший в маршрутизатор, проходит конвейер. Входной порт принимает сигнал физического уровня, собирает кадр канального уровня (например, Ethernet), проверяет его контрольную сумму (FCS) и снимает заголовок — из кадра «вытаскивается» IP-пакет. Проверяется IP-заголовок: версия, длина, контрольная сумма. Маршрутизатор уменьшает TTL, при нуле пакет уничтожается, а отправителю уходит ICMP Time Exceeded — так сеть защищается от петель. Контрольная сумма пересчитывается — она покрывает TTL.

Дальше — главный шаг: поиск маршрута по адресу назначения. Маршрутизатор ищет в таблице пересылки самую длинную совпадающую префикс-запись (longest prefix match) и получает «следующий переход» — соседа, которому нужно передать пакет, и выходной интерфейс. Пакет ставится в очередь выходного порта, а когда доходит черед — упаковывается в новый кадр с новыми MAC-адресами и новой FCS и уходит в провод.

Разделение на routing и forwarding здесь видно физически: протоколы маршрутизации (OSPF, BGP) медленно наполняют таблицу маршрутизации (RIB), из нее лучшие пути загружаются в таблицу пересылки (FIB), и уже она используется для каждого пакета.

Routing — «мозг», forwarding — «рефлексы». Первый пакет потока может уйти по медленному пути через центральный процессор (slow path), но миллионы последующих обрабатываются аппаратно на скорости линии (fast path).

Таблицы маршрутизации

Пример простой таблицы маршрутизации:

```
Сеть             Следующий переход
192.168.1.0/24   подключена локально (порт eth0)
10.0.0.0/8       192.168.1.1
0.0.0.0/0        шлюз провайдера (маршрут по умолчанию)
```

Маршрут по умолчанию (`0.0.0.0/0`) — частный случай агрегации: «все, что не совпало ни с чем более специфичным, отправляй туда». Домашний роутер хранит считаные строки, магистральный маршрутизатор провайдера — полную таблицу BGP, превысившую в 2026 году миллион префиксов. Отсюда практическое следствие: рост таблицы — не абстрактная статистика, а требование к дорогой аппаратуре.

Архитектура: входные порты, коммутирующая матрица, выходные порты

Конструктивно маршрутизатор состоит из входных портов (прием, разбор, поиск маршрута), коммутирующей матрицы (switching fabric — внутренняя «шина», переносящая пакеты со входа на выход) и выходных портов (буферизация, планирование, передача). Исторически сменилось три поколения матриц: **память** (первые маршрутизаторы — по сути компьютеры — копировали пакет под управлением процессора), **шина** (просто, но весь трафик делит ее пропускную способность) и **перекрестная матрица** (crossbar) — сеть параллельных путей, позволяющая одновременно передавать несколько пакетов на разные выходы, она и ее производные используются в современных устройствах.

С перекрестной матрицей связана классическая проблема head-of-line blocking: пакет, которому выходной порт занят, блокирует всю очередь своего входного порта, даже если другие выходы свободны. Решения — виртуальные очереди на выход (VOQ, отдельная очередь под каждый выходной порт) и ускорение матрицы (speedup) относительно скорости линии. На выходных портах пакеты буферизуются — и размер этих буферов, как увидим ниже, определяет задержки во всем Интернете.

TCAM, ASIC и аппаратное ускорение

Магистральные маршрутизаторы обязаны принимать решение о пересылке за сотни наносекунд — на интерфейсе 100 Гбит/с при минимальных пакетах ни один процессор с таким темпом не справится. Поэтому поиск longest prefix match выполняется в специализированной памяти TCAM (ternary content-addressable memory): она параллельно сравнивает адрес со всеми записями за один такт. TCAM дорога, прожорлива и ограничена

по объему, поэтому полная таблица DFZ (default-free zone) должна укладываться в лимиты FIB — и каждый лишний префикс, объявленный кем-то в мире, буквально стоит памяти на всех магистральных маршрутизаторах планеты.

Помимо TCAM, современная пересылка строится на сетевых ASIC и программируемых конвейерах: язык P4 (2015–2016, чипы Barefoot Tofino) позволяет задавать, как именно пакет обрабатывается в железе, — не только IP-маршрутизацию, но и произвольные заголовки и логику. Граница между «жесткой» пересылкой и «программным» управлением размывается — это готовит почву для SDN. Революцию же аппаратной пересылки произвел в 1998 году Juniper M40, где поиск и пересылку делало железо, а не Unix-подобный процессор, как у тогдашних Cisco, скорость выросла на порядки, и рынок магистральных устройств изменился навсегда.

Алгоритмы маршрутизации

Маршрутизация отвечает на вопрос «какой путь лучший?» — и ответ сложнее, чем кажется, потому что «лучший» определяется метрикой (число переходов, задержка, стоимость, политика), а сама сеть распределена: никто не видит ее целиком. Практические алгоритмы делятся на два семейства — вектор расстояний и состояние каналов, — плюс отдельный класс «вектор путей» для BGP.

Кратчайший путь: алгоритм Дейкстры

Базовая абстракция — граф: узлы-маршрутизаторы, ребра-каналы с весом. Задача — найти путь минимальной стоимости от источника до всех узлов, ее решает алгоритм Дейкстры (1959). Стоимостью может быть что угодно: единица за переход, величина, обратная пропускной способности (так делает OSPF), задержка или денежная цена транзита. Дейкстра — вычислительное ядро протоколов состояния каналов, откуда узлы узнают граф — забота протоколов.

Вектор расстояний: RIP и «счет до бесконечности»

В семействе векторов расстояний (distance vector) каждый маршрутизатор хранит таблицу «сеть → расстояние до нее» и периодически сообщает соседям свою таблицу (или ее часть). Сосед прибавляет стоимость канала и выбирает минимум — так знание расходится по системе. Математическая основа — уравнение Беллмана — Форда: распределенный алгоритм, в котором каждый узел вычисляет свой оптимум из оптимумов соседей.

Классическая реализация — RIP (RFC 1058, 1988): метрика — число переходов, максимум 15, рассылка полной таблицы каждые 30 секунд. RIP примитивно прост и до сих пор встречается в малых сетях, RIPng (RFC 2080) перенес его на IPv6, RIPv2 (RFC 2453) добавил поддержку CIDR и аутентификацию.

Главная проблема семейства — «счет до бесконечности» (count to infinity). Если сеть отвалилась, информация распространяется медленно и своеобразно: узел A узнает о пропаже, но сосед B считает, что путь через A жив, и сообщает A «у меня до сети X — 5

переходов», A принимает маршрут, не понимая, что он ведет обратно через B в никуда. Метрика растет, пока не упрется в «бесконечность» (в RIP — 16): на практике это минуты черной дыры и петли маршрутизации. Частичные лекарства: split horizon («не рассказывай соседу то, что узнал от него»), poison reverse (рассказывай, но с бесконечной метрикой), hold-down timers, ограничение метрики. Полностью проблему решает только переход к состоянию каналов.

Ранняя ARPANET использовала векторы расстояний с 1969 года и на практике столкнулась со всеми их болезнями — вплоть до того, что в периоды нагрузки обновления маршрутов вставали в очереди за пользовательскими пакетами. В 1979 году сеть перешла на алгоритм на основе состояния каналов — спор двух семейств решался не теорией, а авариями.

Состояние каналов: OSPF и IS-IS

В протоколах состояния каналов (link state) каждый маршрутизатор знает всю карту сети. Работа строится в пять шагов: обнаружить соседей (hello-пакеты), измерить стоимость каналов, собрать описание своих связей (LSA) и разослать его всем надежной рассылкой (flooding), при которой каждый узел ретранслирует полученное один раз, построить из всех LSA идентичную у всех карту топологии, прогнать на ней алгоритм Дейкстры.

OSPF (Open Shortest Path First, OSPFv2 — RFC 2328, 1998, Джон Мой) — самый распространенный внутренний протокол Интернет. Он бесплатен и открыт (в противовес проприетарному EIGRP от Cisco, опубликованному как RFC 7868 лишь в 2016 году), поддерживает аутентификацию, равнозначные пути (ECMP) и — критично для больших сетей — области (areas): маршрутизаторы знают все о своей области и лишь агрегированную информацию об остальном мире, а магистральная область 0 связывает остальные. OSPFv3 (RFC 5340) адаптирован для IPv6.

IS-IS — потомок протокола OSI (ISO 10589, адаптация RFC 1142), переживший свой стек: он оказался удачнее спроектирован для масштабирования, и именно его выбрали многие крупные провайдеры для внутренней маршрутизации. Сравнение семейств фундаментально: link state знает сеть полностью (дороже по памяти и CPU, быстрее по сходимости, устойчивее к петлям), distance vector знает только направления и расстояния (дешевле, медленнее, уязвимее).

Иерархическая маршрутизация и агрегация

Если бы каждый маршрутизатор хранил маршрут до каждой сети мира, таблица в миллион с лишним префиксов стояла бы в каждом домашнем роутере, а обновления захлестнули бы сеть. Спасение — иерархия: Интернет делится на автономные системы, внутри них — на области OSPF или зоны IS-IS, внутри зон — на локальные сети. Маршрутизатор хранит детальные маршруты о своей зоне и агрегированные — о внешнем мире: вместо миллионов строк одна.

Агрегация (summarization) — это CIDR в действии. По данным CIDR Report на октябрь 2026 года, из 1 085 491 префикса полной таблицы идеально агрегируются 608 802 — то есть 44% записей существуют лишь потому, что их владельцам нужны мультихоминг (подключение к нескольким провайдерам), инженерия трафика или защита от перехвата более специфичными анонсами. Иерархическая маршрутизация масштабируется ровно настолько, насколько участники готовы жертвовать специфичностью ради компактности, — и это постоянное напряжение между оптимизацией каждого и емкостью системы всех.

Устойчивость, сходимость и петли

После любого изменения топологии сети нужно время на сходимость (convergence) — период, пока все маршрутизаторы придут к согласованному состоянию, в это время возможны транзитные петли и временные черные дыры. У векторов расстояний сходимость медленная (минуты), у состояния каналов — быстрая (секунды), но дорогая по ресурсам. Нестабильный канал вызывает route flapping — непрерывные объявления и отзыв маршрута, против этого применяют демпфирование (RFC 2439), временно подавляющее «нервные» маршруты. Отдельная задача — скорость реакции на отказ: телефония и финансовые транзакции требуют восстановления за десятки миллисекунд, поэтому вокруг базовых протоколов вырос слой механизмов быстрого восстановления — BFD (RFC 5880), обнаруживающий смерть канала за миллисекунды, IP Fast Reroute и LFA (Loop-Free Alternates). Последним рубежом обороны от петель остается TTL: какие бы ошибки ни содержали таблицы, пакет умрет через 255 переходов. Философия уровня в чистом виде — не идеальная корректность, а ограниченность ущерба.

Интердоменная маршрутизация: BGP и экономика Интернета

Все протоколы выше работают внутри одной сети — там, где одна администрация, одна политика и одна цель. Интернет же состоит из десятков тысяч независимых сетей, и протокол, который сшивает их между собой, принципиально отличается от OSPF или RIP: он выбирает маршруты не по расстоянию, а по договорам.

Автономные системы

Автономная система (AS, RFC 1930) — сеть под единым административным управлением с единой политикой маршрутизации: провайдер, облако, университет, оператор ЦОД. Каждой AS выдан номер (ASN): исторически 16-битный, с середины 2000-х — 32-битный, потому что и номеров стало не хватать. На октябрь 2026 года в глобальной таблице видно около 79 744 автономных систем, из них свыше 28 тысяч объявляют ровно один префикс, крупнейший по числу префиксов — Amazon (AS16509, около 17,5 тысячи). Интернет — это не одна сеть, а конфедерация тысяч суверенов, и BGP — их дипломатический протокол.

BGP: протокол политик, а не метрик

BGP (Border Gateway Protocol) работает поверх TCP: маршрутизаторы обмениваются информацией о достижимости префиксов через сессии на порту 179. История протокола: RFC 1105 (1989), BGP-2 и BGP-3 (1991) и, наконец, BGP-4 (RFC 1654, 1994, действующая

спецификация RFC 4271, 2006) — именно он добавил поддержку бесклассовой адресации, без CIDR таблица маршрутизации задохнулась бы еще в девяностых. Внешний BGP (eBGP) соединяет разные AS, внутренний (iBGP) разносит внешние маршруты внутри своей. Каждое объявление несет атрибуты: AS_PATH — список пройденных автономных систем (он же — защита от петель: увидев собственный номер, маршрутизатор отвергает объявление), NEXT_HOP, LOCAL_PREF (чем больше, тем лучше), MED — подсказка соседу, и сообщества — метки для гибкой политики. Выбор маршрута идет по строгому порядку: локальные предпочтения, длина AS_PATH, происхождение, MED.

Главное, что нужно понять про BGP: **он не ищет кратчайший путь в физическом смысле**. AS_PATH измеряется числом автономных систем, а не километрами: путь «короче» может быть дороже и невыгоднее. Провайдер может предпочесть длинный маршрут, потому что он проходит через клиентов (которым платят не за трафик, а наоборот), а не через транзитных поставщиков. Модель, формализованная Лисинь Гао и Дженнифер Рексфорд, называется valley-free routing («маршрутизация без долин»): анонсы распространяются вверх (к провайдерам) и вбок (пирам) только для клиентского и собственного трафика, но не для транзита чужого трафика между двумя провайдерами. Нарушение этой негласной конституции Интернета и называется утечкой маршрутов (route leak).

Пиринг, транзит и точки обмена

Экономика связности сводится к двум контрактам. **Транзит**: провайдер А платит провайдеру В за доставку трафика куда угодно. **Пиринг**: две сети обмениваются трафиком своих клиентов бесплатно (settlement-free) или по согласию — это экономит деньги и уменьшает задержки. Физически пиринг происходит в точках обмена трафиком (IXP): DE-CIX во Франкфурте, AMS-IX в Амстердаме, LINX в Лондоне, MSK-IX в Москве — крупные IXP переключают сотни сетей и десятки терабит в секунду. На вершине иерархии — с полтора десятка глобальных сетей tier-1, которые никому не платят за транзит, потому что пирятся между собой без расчетов, ниже — tier-2 и тысячи сетей доступа. Топология системы подчиняется степенному закону (Гао и Рексфорд, 2000): несколько узлов с гигантской связностью, огромное число — с минимальной.

Вывод, который стоит проговорить явно: **Интернет маршрутизируется не по географии, а по экономике**. Трафик между двумя провайдерами в одной стране может уйти через Франкфурт, а связь внутри Африки десятилетиями шла через Европу — потому что прямые контракты отсутствовали, а транзит через бывшие метрополии был дешевле.

Почему не существует глобально оптимального маршрута

Внутри одной AS оптимизация возможна: одна администрация, одна функция цели, OSPF считает минимум стоимости. Между AS общей функции цели нет: у каждой системы своя выгода и свои контракты. Децентрализованная игра десятков тысяч эгоистичных участников не обязана сходиться к глобальному оптимуму — и не сходится. «Лучший маршрут» в масштабах Интернета — понятие не математическое, а переговорное, любая попытка «улучшить Интернет централизованно» упирается именно в это.

Хроника инцидентов: от AS7007 до наших дней

История BGP-инцидентов — лучший учебник по устройству интердоменной маршрутизации.

- **25 апреля 1997 года, AS7007.** Маршрутизатор провайдера MAI Network Services начал объявлять чужие маршруты как свои, с обрезанным AS_PATH. Полмира переключило трафик в черную дыру — первое массовое доказательство того, что BGP верит любому объявлению.
- **24 февраля 2008 года, YouTube и Pakistan Telecom.** Провайдеру велели заблокировать YouTube внутри страны, он объявил более специфичный префикс /24 из адресного пространства YouTube — и транзитный партнер PCCW разослал анонс по всему миру. YouTube лег глобально почти на два часа. Учебниковый пример того, как внутренняя цензурная мера становится глобальной аварией.
- **8 апреля 2010 года, China Telecom.** Около 18 минут через китайскую сеть анонсировалось примерно 15% маршрутов всего Интернета, включая адреса американских правительственных ведомств, детали спорны, но факт массовой переадресации подтвержден.
- **28 августа 2017 года, Google и Япония.** Провайдер NTT Communications из-за ошибки конфигурации транслировал более специфичные маршруты к префиксам Google, значительная часть японского трафика час ходила через сеть Google как через транзитную.
- **24–25 апреля 2018 года, MyEtherWallet.** Злоумышленники перехватили префиксы Amazon Route 53 и перенаправили DNS-запросы пользователей криптовалютного кошелька на фишинговый сервер. Украдено около 150 тысяч долларов в эфире — первый громкий случай, когда BGP-перехват напрямую конвертировался в деньги атакующих.
- **24 июня 2019 года, Cloudflare и «оптимизатор» Verizon.** Автоматический оптимизатор маршрутов Noction в небольшой французской сети DQE сгенерировал некорректные более специфичные анонсы, включая префиксы Cloudflare, Alliance Networks пропустила их наверх, к Verizon, а тот разнес по миру. Пользователи потеряли доступ к сайтам за Cloudflare более чем на час, Verizon часами не отвечал на экстренные обращения, а лимитов на число принимаемых префиксов у него не стояло.
- **4 октября 2021 года, Facebook.** Неудачная команда обслуживания отозвала все BGP-анонсы Facebook (около 126 префиксов). Компания исчезла из глобальной таблицы — вместе с WhatsApp, Instagram и всем, что было завязано на ее DNS. Восстановление заняло около шести часов — отчасти потому, что внутренние инструменты Facebook зависели от той же сети, потери оценили в десятки миллионов долларов. Самый крупный BGP-инцидент в истории устроил сам пострадавший: причиной была конфигурационная ошибка, а не атака.
- **8 июля 2022 года, Rogers.** Крупнейший канадский оператор потерял сеть примерно на 15 часов из-за сбоя обслуживания, отозвавшего маршруты и сломавшего DNS, не работали платежи Interac и часть вызовов экстренных служб.

- **27 июня 2024 года, Cloudflare 1.1.1.1.** Публичный DNS-резолвер деградировал несколько часов из-за комбинации перехвата и утечки: ошибившаяся сеть объявила префиксы 1.1.1.1 без валидных RPKI-подписей, но сети без проверки происхождения анонсы приняли.
- **22 января 2026 года, утечка самого Cloudflare.** Ошибка автоматизации политик на маршрутизаторе в Майами заставила Cloudflare анонсировать чужие IPv6-префиксы своим провайдерам и пирам — утечка типов 3 и 4 по RFC 7908, продлившаяся 25 минут. Ирония в том, что виновником стала компания, годами описывающая чужие утечки, — а вызвало ее безобидное на вид изменение: из политик удалили список префиксов, и пустой список совпал со «всеми маршрутами».

Общий вывод из хроники: почти все крупные инциденты — не злой умысел, а ошибки конфигурации, помноженные на полное доверие BGP к анонсам, и распространяются они через третьи руки, потому что у интердоменной системы нет центра, способного остановить лавину.

Защита маршрутизации: RPKI, BGPsec и MANRS

Ответом на перехваты стал RPKI (Resource Public Key Infrastructure, RFC 6480, 2012): владелец адресного пространства публикует подписанный объект ROA (Route Origin Authorization) — «префикс X вправе объявлять AS Y». Маршрутизаторы с включенной валидацией происхождения (ROV) отбрасывают анонсы, противоречащие ROA. По данным APNIC, в марте 2026 года около 60% наблюдаемых IPv4-маршрутов были RPKI-валидными, 38% — без записей и 2% — невалидными. Прогресс есть, но незакрытые 40% означают, что перехват по-прежнему возможен, — а инцидент июня 2024 года с 1.1.1.1 показал: подписать свои префиксы недостаточно, нужно, чтобы проверку включили остальные.

BGPsec (RFC 8205, 2017) подписывает весь AS_PATH, защищая от подделки пути, но уперся в стоимость: подписи раздувают анонсы, массового внедрения нет. Практичнее «легкие» механизмы: профиль ASPA (стандартизирован в конце 2024 года), проверяющий анонсы по данным о клиент-провайдерских отношениях, и атрибут OTC (Only-to-Customer, RFC 9234, 2022), помогающий ловить утечки. Инициатива MANRS (с 2014 года, под эгидой Internet Society) сводит гигиену к четырем правилам: фильтрация чужих префиксов, защита от спуфинга, координация с соседями, глобальная валидация. Соблюдение добровольное — и в этом весь Интернет: безопасность уровня строится не принуждением, а накоплением коллективной дисциплины.

ICMP: служебный протокол Интернета

IP молча выбрасывает пакеты, которые не может доставить, — если бы не ICMP (Internet Control Message Protocol, RFC 792, сентябрь 1981), сеть была бы не только ненадежной, но и немой. ICMP — это обратная связь сетевого уровня: сообщения об ошибках и простейшая диагностика. Формально это не самостоятельный протокол, а часть IP: сообщения ICMP путешествуют в обычных IP-пакетах с номером протокола 1 (в IPv6 — 58), и обрабатывать их обязан каждый узел.

Типы сообщений

Основные сообщения ICMPv4: **Destination Unreachable** (тип 3) с кодами «сеть/хост/порт недоступен» и особенно важным «требуется фрагментация при установленном DF» — именно оно питает обнаружение MTU пути, **Time Exceeded** (тип 11) — TTL обнулился, **Redirect** (тип 5) — маршрутизатор подсказывает хосту ходить до сети напрямую через соседа, **Echo Request / Echo Reply** (типы 8 и 0) — основа ping, **Parameter Problem** (тип 12) — битый заголовок. Исторически существовал и Source Quench (тип 4) — примитивный сигнал перегрузки, он показал себя вредным и был похоронен RFC 6633 в 2012 году.

В IPv6 роль ICMP радикально выросла: ICMPv6 (RFC 4443) несет не только ошибки, включая **Packet Too Big** (аналог «фрагментация нужна»), но и весь протокол обнаружения соседей NDP — объявления маршрутизаторов, запросы адресов, определение доступности. Отфильтровать ICMPv6 в IPv6-сети — значит сломать ее базовые функции, а не «повысить безопасность», как это иногда получалось с ICMPv4.

ping и traceroute: два инструмента, сделавшие сеть наблюдаемой

ping написал Майк Муус в декабре 1983 года в Ballistic Research Laboratory, название — от звука гидролокатора: эхо-запрос ушел, эхо-ответ вернулся. Сорок с лишним лет спустя это по-прежнему первая команда, которую вводит любой сетевой инженер, когда «ничего не работает».

traceroute появился в 1987 году: Вэн Джейкобсон из LBL реализовал идею, подсказанную Стивом Дирингом, — и она оказалась гениальной в своей наглости. Утилита отправляет серию зондов с последовательно растущим TTL: зонд с TTL=1 убивает первый маршрутизатор, который присылает в ответ ICMP Time Exceeded — так мы узнаем первый переход. TTL=2 — второй, и так далее, пока зонд не достигнет цели. Каждый «умерший» пакет выдает имя и адрес своего палача: путь через Интернет восстанавливается из трупов. Реализации различаются зондами (UDP на высокие порты в Unix, ICMP Echo в Windows tracert, TCP SYN в современных утилитах), а paris-traceroute и dublin-traceroute борются с тем, что ECMP распрыгивает зонды одного «потока» по разным путям.

ICMP в современной сети: фильтрация и «черные дыры»

Отношение к ICMP прошло полный круг. В 1990-х сообщения ICMP считались диагностикой, затем — вектором атак (smurf-атаки конца 1990-х лавинообразно усиливали эхо-запросы через широковещательные адреса), и администраторы начали фильтровать ICMP «на всякий случай». Побочный эффект оказался тяжелее болезни: заблокировав ICMP, сеть теряет PMTUD, ломает traceroute и лишает отправителей информации об ошибках — соединения не рвутся честно, а «зависают». Современная практика — выборочный пропуск: разрешать Echo и, критически важно, Fragmentation Needed / Packet Too Big, ограничивать их скорость, игнорировать Redirect на транзитных интерфейсах.

Перегрузка, очереди и качество обслуживания

Перегрузка — состояние, при котором спрос на пропускную способность превышает предложение, признаки ее — рост задержек, переполнение очередей и потери пакетов. Причины — не только «слишком много трафика»: ограниченная буферизация, дисбаланс скоростей каналов, всплески (burst traffic) и обратные связи между транспортными протоколами и сетью. Ключевой тезис: перегрузка — нормальное состояние пакетной сети, а не сбой, сеть, спроектированная без запаса на перегрузку, обречена.

Коллапс 1986 года и работа Вэн Джейкобсона

В октябре 1986 года Интернет впервые узнал, что такое коллапс перегрузки: на участке ARPANET между Беркли и LBL пропускная способность упала в тысячу раз — с 32 Кбит/с до 40 бит/с. Почти все ресурсы сети тратились на повторные передачи пакетов, которые никто не успевал принять. Исследования Вэна Джейкобсона показали механику: TCP-отправители синхронно увеличивали скорость, синхронно забивали очереди, синхронно теряли пакеты и синхронно перезапускались — автоколебания, раскачивающие сеть.

Доклад Джейкобсона «Congestion Avoidance and Control» на SIGCOMM 1988 года стал одним из самых цитируемых в истории сетевых технологий: экспоненциальное сглаживание таймеров повторной передачи, медленный старт (slow start), избегание перегрузки с аддитивным ростом и мультипликативным уменьшением окна (AIMD). Производительность участка выросла в разы не от нового железа, а от правильного алгоритма. Проблемы решались и раньше: в январе 1984 года Джон Нейгл опубликовал RFC 896, описав коллапс в сети Ford Aerospace и предложив алгоритм борьбы с лавиной мелких пакетов — «алгоритм Нейгла», который до сих пор отключают флагом TCP_NODELAY ради интерактивных приложений.

Open-loop и closed-loop: дырявое ведро и ведро с токенами

Методы борьбы делятся на предотвращение (open-loop — решаем проблему, не глядя на текущее состояние: контроль допуска, сглаживание трафика, правильное проектирование) и реакцию (closed-loop — сеть обнаруживает перегрузку и сигнализирует: ICMP в прошлом, ECN сегодня, потери пакетов как грубый, но работающий сигнал — всегда).

Два классических механизма сглаживания — «дырявое ведро» (leaky bucket) и «ведро с токенами» (token bucket). Дырявое ведро, предложенное Джонатаном Тернером в 1986 году, работает как очередь с постоянным темпом обслуживания: сколько бы ни влилось, вытекает равномерно. Ведро с токенами устроено наоборот: токены поступают в бак со скоростью R, для отправки пакета нужен токен, если бак вместимостью B полон, отправитель может выплеснуть накопленное разом. Средняя скорость ограничена R, но всплески до B разрешены — практика выбирает token bucket, потому что реальный трафик данных всплесками и живет. Комбинация обоих механизмов задает два параметра полисинга — устойчивую скорость и допустимый «горб»: в таких терминах формулируются контрактные обязательства провайдеров.

Очереди: FIFO, drop tail и справедливая очередь

Когда пакеты приходят быстрее, чем уходит канал, они буферизуются. Простейшая дисциплина — FIFO с отбрасыванием хвоста (drop tail): очередь полна — новый пакет выбрасывается. Просто, дешево и вредно: переполнение во многих узлах одновременно синхронизирует потери у множества TCP-потоков, и они синхронно уходят в медленный старт — сеть дышит пилообразными провалами пропускной способности (глобальная синхронизация TCP). К тому же FIFO несправедлива: агрессивный поток захватывает больше очереди и больше канала. Ответом стала справедливая очередь (fair queueing, Деммерс, Кешав и Шенкер, SIGCOMM 1989): идеальная модель — каждому потоку свой сервер, реальная — аппроксимации вроде WFQ и SFQ. Позже идея вернулась в массовые устройства как fq_codel — стандартная дисциплина очередей ядра Linux с 2017 года.

RED и ECN: сигнализировать вместо сбрасывания

Sally Floyd и Вэн Джейкобсон в 1993 году предложили RED (Random Early Detection): не ждать переполнения очереди, а начинать случайно отбрасывать (или, в варианте WRED, взвешенно по классам) пакеты заранее, когда средняя длина очереди превышает порог. Случайность принципиальна: разные TCP-потоки получают потери в разные моменты и десинхронизируются. RED стал мостиком к ECN (Explicit Congestion Notification, RFC 3168, 2001): зачем портить пакет, если можно его пометить? Два бита в IP-заголовке объявляют: «отправитель ECN-способен» (ECT) и «на маршрутизаторе была перегрузка» (CE). TCP снижает окно, как если бы пакет потерялся, — но без потери. Развертывание ECN заняло полтора десятилетия: пионерами стали дата-центры (DCTCP, 2010, использует долю пометок), затем подтянулись операторы и клиентские стеки.

Bufferbloat: как буферы съели задержку

В 2009–2011 годах Джим Геттис и Кэтлин Николс описали системную болезнь, получившую имя **bufferbloat** («раздувание буферов»): производители домашнего и операторского оборудования ставили огромные очереди — «чтобы не терять пакеты». Результат оказался обратным: на забитом канале домашнего роутера пакеты стоят в многомегабайтной очереди секундами, задержка вырастает до 5–10 с, VoIP и игры становятся невозможными, а TCP, видя отсутствие потерь, продолжает разгоняться. Потери — плохой сигнал перегрузки, но их отсутствие в гигантском буфере — сигнал еще хуже: сеть молчит, пока не начинает «захлебываться» задержкой.

Лекарства — алгоритмы активного управления очередями (AQM). **CoDel** (Controlled Delay, Николс и Джейкобсон, 2012) отслеживает не длину очереди, а минимальную задержку пакетов в ней: если она устойчиво превышает целевую (5 мс), пакеты начинают отбрасываться с сокращающимся интервалом — это быстро гасит перегрузку и не наказывает короткие всплески. **FQ-CoDel** добавляет справедливое разделение по потокам, **PIE** (RFC 8033) решает ту же задачу ПИ-регулятором. Все три стали частью DOCSIS 3.1 и ядра Linux. Финальный аккорд — архитектура **L4S** (RFC 9330–9332, январь 2023): две раздельные очереди для трафика с «тонкой» реакцией на ECN-пометки и для

классического — задержка в единицах миллисекунд при полном использовании канала. Тридцать семь лет между коллапсом 1986 года и L4S — и все это время проблема была одна: как честно сказать отправителю, что сеть перегружена, не роняя пакеты.

QoS: IntServ, RSVP и DiffServ

Разным приложениям нужно разное: видеозвонку — малая задержка и джиттер (потери до нескольких процентов он переживет), передаче файла — надежность при безразличии к задержке, веб-серфингу — что-то среднее. Сеть «старайся как можешь» этого не понимает: все пакеты равны.

Первая попытка — **IntServ** (RFC 1633, 1994): приложение резервирует ресурсы под поток сигнальным протоколом RSVP (RFC 2205, 1997), каждый маршрутизатор хранит состояние резервирования и гарантирует полосу и задержку. Модель красивая и не масштабируемая: состояние на поток в ядре Интернета — это ровно те виртуальные каналы, от которых сеть когда-то отказалась. IntServ выжил в нишах, а RSVP переродился в RSVP-TE — протокол инженерии трафика для MPLS.

Вторая попытка оказалась живучей — **DiffServ** (RFC 2474/2475, декабрь 1998): 6 бит поля DSCP в IP-заголовке задают класс обслуживания пакета, маршрутизаторы применяют к классам поведение per-hop (PHB): EF (Expedited Forwarding) — низкая задержка для голоса, AF (Assured Forwarding) — гарантированная доля с градациями важности, BE (Best Effort) — все остальное. Никакого состояния на поток: только классы, — масштабируемо и просто. Главный изъян: DSCP должен одинаково пониматься всеми сетями на пути, а между доменами отметки стираются — никто не хочет бесплатно возить чужой «премиум-трафик». Поэтому DiffServ прекрасно работает внутри доверенного домена — провайдера, корпорации, дата-центра (голос в VoLTE помечен EF) — и почти нигде не работает сквозным образом через весь Интернет. Честный компромисс песочных часов: глобальная простота против локальной управляемости.

Фрагментация и Path MTU Discovery

MTU: почему пакеты приходится резать

У каждой сети канального уровня есть максимальный размер кадра, а у IP-пакета внутри — MTU (Maximum Transmission Unit). Ethernet — 1500 байт, PPPoE — 1492 (8 байт съедают заголовки PPP), FDDI — 4352, туннели — 1500 минус накладные расходы (GRE, IPsec, VXLAN отъедают от 24 до 50+ байт). Пакет, собранный для «толстой» сети, может не пролезть в «узкую» по пути — и тогда IPv4-маршрутизатор делит его на фрагменты.

Как это работает

Фрагменты одного пакета несут одинаковый Identification и разные Fragment Offset, у всех, кроме последнего, взведен флаг MF. Фрагменты идут независимо и могут прийти в любом порядке, получатель буферизует их и собирает исходный пакет. Сборка выполняется

только на конечном узле: промежуточные маршрутизаторы фрагменты не пересобирают — иначе каждый хранил бы состояние фрагментов, а это снова виртуальные каналы, только без преимуществ.

Следствия суровы: потеря любого фрагмента убивает весь пакет — пакет из десяти фрагментов при вероятности потери фрагмента 1% доходит с вероятностью около 90%. Сборка требует памяти и таймеров — вектор атак на ресурсы. Только первый фрагмент содержит заголовки L4: фаерволы, фильтрующие по портам, видят их лишь у первого куска, а остальные пропускают «вслепую».

Атаки на фрагментацию

Фрагментация подарила безопасности отдельный жанр. Атака **teardrop** (1997) отправляла фрагменты с перекрывающимися смещениями — стеки Windows и Linux при сборке путались и падали. Работа Птасека и Ньюшэма «Insertion, Evasion, and Denial of Service» (1998) показала, как перекрывающиеся фрагменты позволяют «спрятать» полезную нагрузку атаки от IDS: система обнаружения и конечный узел собирают один набор фрагментов в разные пакеты. **Tiny fragment** — крошечный первый фрагмент, в который не помещаются порты TCP, — обходил ранние фильтры. Современный ответ: запрет фрагментации, где она не нужна (в IPv6 ее в транзите нет), нормализация пакетов на IDS и фильтрация аномальных фрагментов.

Path MTU Discovery и черные дыры

Правильная стратегия — не резать пакеты в пути, а заранее знать минимальный MTU на всем маршруте. Для этого существует PMTUD (Path MTU Discovery, RFC 1191, 1990, Могул и Диринг): отправитель взводит бит DF («не фрагментировать»), маршрутизатор, которому пакет велик, отбрасывает его и возвращает ICMP Fragmentation Needed со своим MTU, отправитель уменьшает размер и пробует снова.

На практике PMTUD знаменит «черными дырами»: если где-то по пути администратор отфильтровал весь ICMP, пакеты с DF молча умирают, отправитель ничего не узнает, и соединение «зависает» — симптом, знакомый любому, кто настраивал VPN (RFC 2923 посвящен этой патологии целиком). Лекарства: MSS clamping (роутер правит опцию MSS в TCP-рукопожатии, RFC 6691), PLPMTUD (RFC 4821, RFC 8899) — зондирование пути пакетами разного размера без опоры на ICMP, — и здравый смысл: не фильтровать ICMP Fragmentation Needed. В IPv6 фрагментации в транзите нет вовсе, поэтому PMTUD через ICMPv6 Packet Too Big — обязательная часть работы сети, сломать его — значит сломать IPv6-связность целиком.

NAT: костыль, на котором держится Интернет

История и причины

NAT (Network Address Translation) придумали как временную меру: RFC 1631 (май 1994, Кьелль Эгеванг и Пол Фрэнсис) описывал трансляцию адресов как мост, который протянет IPv4 до победного внедрения IPv6. Прошло тридцать два года, IPv6 у половины пользователей, NAT стал не мостом, а несущей конструкцией — практически каждый домашний роутер в мире сегодня является NAT-устройством, а у крупных мобильных операторов стоят «двойные NAT» (CGNAT). Причина — нехватка адресов IPv4, вторая, неожиданная причина — «бесплатная» безопасность: извне внутрь сети без явной настройки не добраться.

Как NAT работает внутри роутера

Классический случай — NAPT (Network Address Port Translation), он же PAT: транслируются не только адреса, но и порты. Внутренний хост `192.168.1.5` отправляет пакет на `8.8.8.8` с портом 1234. Роутер подменяет пару (IP, порт) на свою внешнюю — (`80.123.45.6`, 54321) — и записывает в таблицу трансляции строку `192.168.1.5:1234 ↔ 80.123.45.6:54321`. Когда приходит ответ, роутер находит строку и подменяет адресата обратно. Записи живут, пока живо соединение, — и именно эта таблица есть то состояние, которое NAT добавляет в «бессостоятельную» дейтаграммную сеть. Один публичный адрес обслуживает так тысячи соединений: теоретический лимит — около 64 тысяч трансляций.

Последствия: конец сквозной модели

NAT ломает сквозную (end-to-end) адресацию: из Интернета нельзя просто «постучаться» к конкретному хосту. Это бьет по всем протоколам на входящих соединениях: P2P, игровые серверы, IP-телефония, удаленный доступ. Сообщество ответило индустрией обходов: **STUN** (RFC 5389) позволяет хосту узнать свой внешний адрес и «тип» NAT (симметричный пробивается хуже всего), **TURN** (RFC 5766) — ретранслятор, гарантированный, но платный по трафику путь, **ICE** (RFC 8445) — алгоритм, перебирающий все варианты и выбирающий работающий, на этом стеке построен WebRTC и весь современный P2P. Для локальных сетей есть UPnP IGD, NAT-PMP и PCP (RFC 6887) — протоколы, которыми устройства просят роутер «пробросить порт» автоматически, их безопасность — отдельная головная боль (уязвимости вроде CallStranger показали, что UPnP из интернета лучше не слушать).

Отдельно страдают протоколы, переносящие IP-адреса внутри полезной нагрузки: FTP в активном режиме, SIP, H.323, IPsec ESP. Для них придумали ALG (Application Level Gateways) — модули роутера, заглядывающие в payload и правящие адреса, ALG настолько хрупки, что современный стиль — «выключите ALG и используйте правильный протокол». IPsec спасли режимом NAT-T (инкапсуляция ESP в UDP/4500).

CGNAT и рынок адресов

Операторский NAT (CGN, Carrier-Grade NAT) ставит трансляцию у провайдера: тысячи абонентов делят пул публичных адресов, а их внутренние адреса живут в диапазоне `100.64.0.0/10` (RFC 6598), чтобы не конфликтовать с домашними `192.168.x.x`. Двойная трансляция усугубляет все проблемы NAT и добавляет новые: несколько абонентов на одном публичном адресе ломают геолокацию, лимиты «запросов в секунду на IP» у сервисов и криминалистику — чтобы понять, кто из тысячи абонентов выходил с адреса в данный момент, провайдер обязан вести журналы распределения портов, что само по себе вопрос приватности.

Экономически CGNAT — способ не покупать IPv4 по рыночной цене и не торопиться с IPv6. Мобильные сети здесь впереди: T-Mobile US и Reliance Jio работают IPv6-only, а IPv4-доступность дают трансляцией — схема, которая и задумывалась в 2011–2012 годах как магистральный путь перехода.

NAT как «файрвол»: миф и реальность

Распространено мнение, что NAT — это безопасность. Реальность сложнее: да, NAT блокирует нежелательные входящие соединения, — но простейший фаервол делает то же и лучше: stateful-фаервол понимает, какой пакет принадлежит установленному соединению, а NAT — нет. NAT ломает сквозную криптографию (IPsec в транспортном режиме через NAPT не ходит), мешает журналированию и дает ложное чувство защищенности. Корректная формулировка: **NAT — адресный транслятор с побочным эффектом фильтрации, а не средство защиты**, безопасность дает фаервол, а дефицит адресов лечит IPv6.

Anycast, multicast и многопутевая маршрутизация

Помимо привычного unicast («один адрес — один узел»), сетевой уровень знает еще две модели адресации и одну модель распределения нагрузки, каждая со своей судьбой.

Anycast: один адрес — много узлов

Anycast — механизм, при котором один и тот же IP-адрес объявляется из многих точек сети (обычно через BGP), а маршрутизация сама приводит отправителя к «ближайшему» экземпляру — ближайшему в смысле политики BGP, а не километров. Отправитель не знает, к какому физическому серверу попадет, разные его пакеты могут прийти к разным экземплярам, поэтому поверх anycast безопасно строить прежде всего протоколы без состояния — в первую очередь DNS.

Именно DNS сделал anycast промышленным стандартом: из тринадцати «корневых» логических серверов Интернета к середине 2020-х физически развернуто полторы тысячи с лишним экземпляров по всему миру. Публичные резолверы Google (8.8.8.8), Cloudflare (1.1.1.1), Quad9 (9.9.9.9) — тоже anycast: пользователь в Дели и пользователь во Франкфурте обращаются к одному адресу, но попадают в разные дата-центры. Для CDN и защиты от DDoS anycast — способ рассредоточить атаку по точкам присутствия. Практика

эксплуатации описана в RFC 4786 (2007), главные требования — стабильность анонсов (переключения экземпляров рвут соединения) и согласованность состояния там, где оно нужно.

Multicast: красивая идея, не ставшая массовым сервисом

Multicast решает задачу «один отправитель — группа получателей» без дублирования трафика: пакет адресуется группе (в IPv4 — диапазон `224.0.0.0/4`, в IPv6 — `ff00::/8`), маршрутизаторы строят деревья доставки и реплицируют пакеты только в точках ветвления. Экономия при вещании на миллионы очевидна.

История мультикаста романтична: в марте 1992 года на заседании IETF в Сан-Диего Стив Касснер и Вэн Джейкобсон разослали аудио на десятки площадок по миру — родилась MBone, сеть туннелей, связавшая острова мультикаста через немультискастный океан. В ноябре 1994 года MBone транслировала концерт Rolling Stones — один из первых массовых интернет-стримов. Протоколы управления группами (IGMP на краю, PIM в ядре) были созданы и стандартизированы.

Но в глобальном Интернете мультикаст так и не стал общим сервисом. Причины: состояние на группу в каждом маршрутизаторе (снова масштабирование), асимметрия экономики (отправитель экономит, платят транзитом получатели), сложность диагностики и слабость контроля источников. Мультикаст выжил там, где один административный домен: IPTV у операторов, биржевые рассылки котировок, внутренние рассылки ЦОД. В IPv6 он стал обязательной частью архитектуры, но массовых «вещательных» сервисов не породил — нишу заняли CDN и адаптивный стриминг через обычный unicast.

ECMP: многопутевость как способ использовать весь металл

Реальные сети избыточны: между двумя маршрутизаторами почти всегда несколько равноценных каналов, между ЦОД — десятки путей. ECMP (Equal-Cost Multi-Path) позволяет держать в таблице несколько следующих переходов для одного префикса и распределять между ними потоки — не попакетно (иначе порядок внутри TCP-потока сломается), а по хэшу от кортежа (source IP, dest IP, протокол, порты).

ECMP — фундамент современного Интернета: без него невозможны ни агрегация каналов у провайдеров, ни сети ЦОД (где топология Clos дает десятки равных путей). Его болезни тоже фундаментальны: хэш бывает неудачным (два «слона» — огромных потока — попадают на одно соединение и забивают его, пока остальные простаивают), при изменении топологии часть потоков перескакивает на другие пути и обгоняет уже отправленные пакеты (reordering ломает TCP). Ответы — балансировка по flowlet (пачкам пакетов внутри потока), адаптивная маршрутизация в новейших fabric-коммутаторах, алгоритмы вроде Hedera и CONGA.

MPLS и инженерия трафика

Метки вместо префиксов

MPLS (Multiprotocol Label Switching, архитектура — RFC 3031, январь 2001) вставляет между заголовками L2 и L3 короткую метку (label, 20 бит). Краевой маршрутизатор (LER) анализирует IP-пакет один раз и навешивает метку, внутренние (LSR) дальше смотрят только на метку: «метка 42 пришла с порта 3 → замени на 77 и отправь в порт 5». Таблица меток мала, поиск тривиален — это коммутация, а не маршрутизация. Стек меток позволяет вкладывать туннели друг в друга.

Исторически MPLS продавали как ускоритель: в конце 1990-х поиск longest prefix match в софте был медленным. К середине 2000-х мотив устарел — TCAM и ASIC сделали IP-поиск аппаратным. Но MPLS не умер, а переродился: его настоящая ценность оказалась не в скорости.

Зачем он нужен на самом деле

Первая ценность MPLS — **инженерия трафика** (traffic engineering). IP-маршрутизация гоняет весь трафик по одному «лучшему» пути, оставляя резервные простаивать, MPLS с RSVP-TE позволяет явно проложить туннель по выбранному маршруту — в обход перегруженного узла, с гарантированной полосой. Вторая — **VPN** (RFC 4364): провайдер продает клиентам изолированные сети, где адресные пространства могут перекрываться, маршруты разделяются VRF, а транспорт обеспечивается метками. Третья — **независимость от IP**: поверх MPLS возят IPv4, IPv6, Ethernet-кадры и что угодно — универсальный транспорт магистралей, где сам IP лишь один из пассажиров.

Практически все магистральные сети провайдеров 2000–2020-х годов построены на MPLS в ядре с IP/BGP по краям. Пользователь Интернета этого не видит — и не должен: MPLS живет внутри автономных систем, между доменами по-прежнему царит BGP.

Сегментная маршрутизация: SR-MPLS и SRv6

Новое поколение — Segment Routing (SR, архитектура RFC 8402, 2018): путь пакета задается не состоянием в промежуточных узлах, а списком сегментов в самом пакете — «иди через узел A, потом через канал B, потом к C». Каждый сегмент — номер (в SR-MPLS это метка, в SRv6 — 128-битный IPv6-адрес с программой поведения). Состояние из ядра исчезает: источник или контроллер «зашивает» путь в заголовок, а транзитные узлы следуют инструкции.

SRv6 (RFC 8986, 2021) особенно интересен: сегмент — это IPv6-адрес, а его нижние биты — функция узла («переслать», «применить политику»), то есть программа обработки вшита в сам адрес. К середине 2020-х SRv6 активно внедряют крупные операторы — как замену MPLS там, где нужен один стек IPv6 на все случаи. Так виртуальные каналы, проигравшие дейтаграммам в 1980-е, совершили третий заход — в форме «маршрутизации от источника», совместимой с дейтаграммной философией.

SDN: отделение управления от пересылки

Идея и предыстория

Классический маршрутизатор совмещает два начала: data plane (пересылка пакетов по таблицам) и control plane (протоколы, наполняющие таблицы), сеть в целом — результат эмерджентного поведения тысяч независимых решений. SDN (Software-Defined Networking) предлагает разделение: пересылку оставить простым железу, а управление вынести в логически централизованный программный контроллер, который видит сеть целиком и диктует потоковые правила устройствам.

Идея зрела долго: еще в 1990-х существовали коммутаторы с программным управлением (5ESS в телефонии, «softswitch» в VoIP). Прорывом стал 2008 год: статья «OpenFlow: Enabling Innovation in Campus Networks» Ника МакКеона и коллег из Стэнфорда описала протокол, по которому контроллер добавляет, изменяет и удаляет правила «заголовок пакета → действие» в коммутаторах. Инновации в маршрутизации стало можно писать на контроллере, не дожидаясь прошивок вендоров.

Контроллеры, whitebox-коммутаторы и практика

Экосистема выросла быстро: контроллеры NOX/POX/Floodlight, промышленные ONOS и OpenDaylight, коммерческие платформы VMware NSX, Cisco ACI, «белые» коммутаторы (whitebox) — стандартное железо с открытой сетевой ОС. В гиперскейл-ЦОД SDN стал нормой: сеть программируется тем же конвейером автоматизации, что и виртуальные машины.

Но централизация оказалась палкой о двух концах. Контроллер — единая точка отказа (лечится кластеризацией, но согласованность распределенного контроллера — сама по себе сложная задача), а полное видение сети упирается в масштаб: миллионы потоков — миллионы правил. И главное — Internet-scale SDN не состоялся: у глобального Интернета нет владельца, которому можно передать контроль, BGP с его политическим плюрализмом «анти-SDN» по природе. SDN победил внутри доменов (ЦОД, кампусы, операторское ядро) и остановился на их границах.

Сети дата-центров: Clos и overlay

Современный ЦОД строится по топологии Clos (leaf-spine): серверы подключаются к leaf-коммутаторам, те — к spine-коммутаторам полным бипартитным графом, между любой парой серверов несколько равноценных путей, ECMP распределяет потоки, емкость наращивается добавлением spine-рядов. Структура, придуманная Чарльзом Клозом в 1952 году для телефонных коммутаторов, оказалась идеальной для облаков.

Поверх физической IP-сети разворачивается overlay — виртуальные сети тенантов: VXLAN (RFC 7348, 2014) упаковывает Ethernet-кадры в UDP/IP с 24-битным идентификатором VNI (16 миллионов изолированных сегментов против 4094 у VLAN), а EVPN поверх BGP (RFC 7432) разносит адреса виртуальных машин между leaf-коммутаторами. L3-адресация

физического мира стала транспортом для «виртуального L2/L3», в котором живут рабочие нагрузки, — песочные часы в квадрате: IP снизу держит overlay, IP внутри overlay держит приложения.

Туннели, VPN и мобильность

IP поверх IP

Туннелирование — упаковка пакета одного протокола в пакет другого — универсальный инструмент сетевого уровня: так работают механизмы перехода IPv4/IPv6, MPLS-транспорт, VXLAN и VPN. Простейший вариант — IP-in-IP (RFC 2003): IP-пакет целиком становится полезной нагрузкой другого IP-пакета. GRE (RFC 2784) добавляет универсальность — внутри может быть не только IP, но и Ethernet, так устроены многие «виртуальные провода». Туннели — и способ обойти ограничения, и способ их наложить: маршрутизация через туннель скрывает от оператора реальные конечные точки трафика.

IPsec: криптография сетевого уровня

IPsec — семейство протоколов, встроенное непосредственно в IP: AH аутентифицирует пакет целиком (и почти не используется — несовместим с NAT), ESP (Encapsulating Security Payload, протокол 50) шифрует и аутентифицирует полезную нагрузку, IKE согласовывает ключи. Режимы: транспортный (защищается полезная нагрузка) и туннельный (пакет упаковывается целиком — основа VPN «сайт-сайт»).

История IPsec поучительна. Разработанный в середине 1990-х как обязательная часть IPv6, он так и не стал повсеместным: обязаловка тихо отмерла, IKE славится сложностью конфигурации, NAT ломает транспортный режим. IPsec выжил в трех нишах: VPN между филиалами (вытесняемый WireGuard), удаленный доступ (вытесняемый моделью zero trust) и… криптография BGP: сессии между маршрутизаторами провайдеров защищает именно ESP в транспортном режиме. Ирония архитектуры: протокол, которому прочили роль защиты всего Интернета, стал рабочим инструментом защиты самой маршрутизации.

Там, где IPsec требовал сертифицированных шлюзов, современный минимализм предлагает WireGuard (Джейсон Доненфелд, 2015–2017, в ядре Linux с 2020 года): около 4000 строк кода, один криптографический примитив (Noise), работа поверх UDP. Сетевой уровень для него — просто транспорт: WireGuard создает виртуальный интерфейс, а его маршрутизация управляется обычными средствами ОС. Очередное подтверждение того, что простота побеждает полноту.

Mobile IP: мобильность как свойство уровня

Когда хост перемещается между сетями, его IP-адрес меняется, и соединения рвутся. Mobile IP (RFC 5944 для IPv4, RFC 6275 для IPv6) решает задачу в духе самого уровня: за узлом закреплен постоянный «домашний адрес», в чужой сети он получает «адрес

пребывания» (care-of address) и регистрирует его у домашнего агента, а тот перехватывает пакеты на домашний адрес и туннелирует их на адрес пребывания. Для корреспондента узел неподвижен.

На практике Mobile IP почти не используется — его затмили решения выше и ниже: приложения научились переживать смену адреса (QUIC привязывает сессию к connection ID, а не к IP), а сотовые операторы решили мобильность в своей плоскости: GTP туннелирует весь трафик телефона через фиксированную точку в сети оператора, так что телефон, с точки зрения Интернета, всегда «дома». Задачу, которую уровень решить может, мир решает там, где удобнее, — и Mobile IP остался красивым учебным примером туннельной мобильности.

Безопасность сетевого уровня

Отсутствие аутентификации источника

IP-пакет можно отправить с любым адресом источника: поле Source Address сетью не проверяется. Этот спуфинг — родовой порок архитектуры, заложенный не от небрежности, а от философии: сеть доставки не знает, кто платит за билеты. Последствия — весь жанр DDoS-атак с усилением и отражением: атакующий шлет маленькие запросы с поддельным адресом жертвы на открытые серверы (DNS, NTP, memcached), а те заливают жертву большими ответами, усиление достигает десятков тысяч раз (memcached — до 51 000×, что позволило атаке на GitHub в феврале 2018 года достичь 1,35 Тбит/с).

Ответ известен с 2000 года: BCP 38 (RFC 2827) — ingress filtering, проверка на границе сети, что исходящие пакеты несут адреса из ее префиксов. Позже добавились uRPF (RFC 3704) — проверка обратимости пути в маршрутизаторах, и SAVI — привязка адресов к портам коммутаторов доступа. По многолетним измерениям проекта Spoofman (CAIDA), доля сетей, из которых нельзя отправить поддельный адрес, выросла с единиц процентов в середине 2000-х до двух третей к середине 2020-х, — но правило BCP 38 до сих пор добровольное, и «спуфингуемые» сети остаются сырьевой базой ботнетов. Здесь как в зеркале видна политическая экономия сетевого уровня: исправление известного всем дефекта десятилетиями упирается не в технологию, а в отсутствие принуждения.

Атаки на сами механизмы уровня

Кроме спуфинга, уязвимы все служебные механизмы: ARP-отравление, ICMP Redirect, заставляющий хост поменять маршрут на атакующего, ложные ICMP-сообщения об ошибках, рвущие соединения, фрагментационные атаки, обходящие средства инспекции, BGP-перехваты и утечки, DHCP-голод, DNS-отравление кэша (продемонстрированное Дэном Камински в 2008 году) — формально уровень выше, но эксплуатирующее anycast- и UDP-природу сетевой доставки. Сетевой уровень — поверхность атаки целиком: почти все перечисленное существует потому, что служебные протоколы уровня проектировались в эпоху доверия.

Что сделано и чего не хватает

Современный набор защит: RPKI для происхождения маршрутов (60% валидных к марту 2026 года), ASPA/OTC для отношений и утечек, BGPsec для пути (почти не внедрен), фильтрация по BCP 38/uRPF/SAVI, лимиты префиксов на BGP-сессиях (которые Verizon в 2019 году не выставил), координация через RFC 9234. Защита строится снизу вверх, договором участников, а не указом сверху — MANRS как «джентльменский набор». Уязвимость системы определяется слабейшим участником, а цена ошибки отдельного оператора глобальна: это мы видели и в 2008-м с YouTube, и в 2021-м с Facebook, и в январе 2026-го с Cloudflare.

Путь пакета через Интернет: собираем все вместе

Проследим один HTTP-запрос от ноутбука до сервера, используя весь разобранный аппарат. Это упражнение полезно тем, что показывает: ни один из механизмов не работает изолированно.

1. **DNS.** Приложение знает только имя. Резолвер (обычно anycast-адрес вроде 1.1.1.1 или 8.8.8.8) итеративно спрашивает корневые серверы, серверы зоны и авторитетные серверы домена, получая IP сервера. Каждый из этих запросов сам прошел весь описываемый ниже путь — Интернет держится на рекурсии собственного протокола.
2. **Локальное решение.** Стек хоста сравнивает IP назначения с маской подсети: адрес «чужой», пакет пойдет через шлюз по умолчанию — домашний роутер.
3. **ARP.** Хост ищет в кэше MAC роутера, если записи нет — широковещательный запрос «кто имеет 192.168.1.1?», ответ, кэш. Пакет укладывается в Ethernet-кадр: source MAC — ноутбук, destination MAC — роутер, destination IP — конечный сервер (неизменно!).
4. **NAT.** Роутер выделяет порт на внешнем адресе, записывает строку в таблицу трансляции, переписывает source IP:port. Теперь для Интернета пакет исходит от провайдерского адреса роутера.
5. **Магистраль провайдера.** Маршрутизаторы уменьшают TTL, делают longest prefix match по FIB (заполненной OSPF/IS-IS и iBGP), ECMP выбирает один из равнозначных путей, возможно, пакет едет внутри MPLS- или SRv6-туннеля, а DSCP определяет очередь на выходе. На границе AS eBGP уже выбрал лучший внешний маршрут по политике: local pref, длина AS_PATH, MED.
6. **Между системами.** Пакет пересекает цепочку автономных систем — возможно, через IXP одним переходом, возможно, через транзитных провайдеров на другом континенте, путь выбирался не по километражу, а по valley-free правилам и контрактам. На каждой границе — новый MAC, новый TTL, тот же IP назначения.
7. **Дата-центр назначения.** Anycast или L4-балансировщик приводит пакет к стойке, overlay-сеть (VXLAN/EVPN) доносит его до виртуальной машины, фаерволы проверяют законность соединения.
8. **Ответ.** Сервер отвечает на внешний адрес NAT и переведенный порт, роутер находит строку в таблице трансляции и восстанавливает внутренний адрес, TCP собирает

сегменты в поток, приложение получает ответ. Путь занял десятки миллисекунд и 10–20 переходов — каждый со своей очередью, политикой и вероятностью ошибки.

Что остается неизменным на всем пути: IP-адреса источника и получателя (если не считать NAT). Что меняется на каждом переходе: MAC-адреса кадра, TTL, контрольная сумма заголовка. Что может случиться в пути: потеря (перегруженная очередь), задержка (буферы), переупорядочивание (ECMP при сдвиге маршрутов), фрагментация (узкий MTU), отбрасывание фильтрами, перехват (BGP-инцидент). И на все это сетевой уровень дает один ответ: best effort — и инструменты диагностики (ping, traceroute), чтобы понять, где именно все сломалось.

Итоги

Сетевой уровень — глобальная система доставки пакетов через множество независимых сетей, построенная на компромиссе между простотой, масштабируемостью и управляемостью. Все, что мы разобрали, укладывается в несколько сквозных линий.

Первая линия — простота как стратегия. Дейтаграммная модель, best effort IP, отсутствие состояния в ядре, аргумент конец-в-конец: сеть намеренно сделана «глупой», чтобы выжить, масштабироваться и не зависеть от технологий краев. За это заплачено надежностью (ее восстановил TCP), качеством обслуживания (его чинят DiffServ, очереди, AQM — и до сих пор не починили глобально) и безопасностью (ее латают BCP 38, RPKI, MANRS — добровольно и медленно). Каждая «дыра» сетевого уровня — не просчет проектировщиков, а осознанная цена выбора, которую каждое поколение инженеров пересматривает заново.

Вторая линия — иерархия против роста. CIDR, агрегация, автономные системы, области OSPF, ECMP, туннели: все это способы держать таблицу в миллион префиксов в рамках памяти железа и времени сходимости протоколов. Полная таблица могла бы быть на 44% меньше при идеальной агрегации — разница между идеалом и реальностью есть стоимость мультихоминга, инженерии трафика и недоверия между участниками. Масштабируемость Интернета — не свойство алгоритмов, а результат непрерывного торга между компактностью общих структур и специфичностью частных интересов.

Третья линия — экономика как протокол. BGP маршрутизирует не пакеты, а контракты. Valley-free правила, пиринг против транзита, IXP как рыночные площади, anycast как способ продавать близость — топология Интернета повторяет топологию его бизнеса. Отсюда и хрупкость: у системы нет центра не только технического, но и политического, глобально оптимального маршрута не существует в принципе, а единственная «полиция» — взаимный интерес участников не сжигать общее поле. Хроника инцидентов от AS7007 (1997) до утечки Cloudflare (2026) — история того, как доверительный протокол живет во враждебном мире и пока выживает.

Четвертая линия — перманентная эволюция. Сетевой уровень менялся трижды и не остановился: IPv4 пережил исчерпание адресов благодаря NAT и рынку адресов, IPv6 за тридцать лет дошел до половины пользователей, внутри уровня MPLS сменил ATM, SDN и

SRv6 сменяют классические плоскости управления, L4S закрывает гештальт буферизации 1986 года. Песочные часы работают: перемычка IP остается узкой, а миры над ней и под ней обновляются непрерывно.

Для инженера практический вывод звучит так. Во-первых, понимать сетевой уровень — значит понимать, где именно ломается связь: ping и traceroute покрывают диагностику доступности, MTU и ICMP-фильтрация объясняют «зависания» VPN, таблицы трансляции NAT — недоступность сервисов, BGP-мониторинг — глобальные «штормы». Во-вторых, почти все реальные аварии уровня конфигурационные: пустой список префиксов, снятый лимит, неотфильтрованный анонс, заблокированный ICMP, поэтому дисциплина (фильтры по умолчанию, лимиты сессий, RPKI, BCP 38) важнее любого отдельного протокола. В-третьих, сетевой уровень учит скромности: сеть из миллиона независимых участников не управляется указом — она эволюционирует соглашениями.

Интернет работает не потому, что он идеален, а потому, что он достаточно прост, чтобы масштабироваться, и достаточно распределен, чтобы переживать ошибки каждого отдельного участника. Сетевой уровень — место, где эта философия превращена в инженерные решения: TTL вместо безупречных таблиц, best effort вместо гарантий, политика вместо оптимума, BCP вместо полиции. Следующий шаг вверх — транспортный уровень, где TCP и UDP возьмут на себя все, от чего IP сознательно отказался: надежность, порядок, поток и соединение.