

Сеть на вырост

Как проектировать надежную
и масштабируемую
IT-инфраструктуру



ИИ
Елена Светлова

Елена Светлова

Сеть на вырост: Как проектировать надежную и масштабируемую IT-инфраструктуру

*<https://litres.ru/74607757>
SelfPub; 2026*

Аннотация

Платежи зависят, пользователи жалуются, каналы загружены — а решение снова сводится к просьбе «сделать сеть побыстрее»? Надёжность не появляется после покупки нового оборудования: её проектируют, измеряют и проверяют.

«Сеть на вырост» поможет превратить запросы бизнеса в понятные технические требования, SLA и сценарии роста. Вы разберётесь, как выбирать топологию под реальные потоки трафика, планировать адресацию и сегментацию, настраивать маршрутизацию и резервирование. Узнаете, как оценивать ёмкость и задержки, проектировать WAN, Wi-Fi и гибридную инфраструктуру, находить первопричины сбоев по телеметрии и журналам, автоматизировать изменения с проверкой и готовым откатом.

Книга адресована сетевым инженерам, архитекторам и ИТ-руководителям. Она поможет принимать обоснованные решения, снижать риск простоев и развивать инфраструктуру вместе с бизнесом — без лишней сложности и дорогостоящих переделок.

Книга создана с помощью ИИ. Обложка: GPT Image - Лицензия

Содержание

От бизнес-задачи к техническим требованиям: цели, ограничения, SLA и сценарии роста	6
Основы сетевой архитектуры: уровни, компоненты, протоколы и физическая инфраструктура	20
Топологии и архитектурные модели: кампус, филиалы, дата-центр и распределённая сеть	32
Адресное пространство и сегментация: IPv4, IPv6, VLAN, подсети и сетевые сервисы	47
Конец ознакомительного фрагмента.	55

Елена Светлова
Сеть на вырост: Как
проектировать надежную
и масштабируемую
IT-инфраструктуру

**Сеть на вырост: Как проектировать надежную и
масштабируемую IT-инфраструктуру**

Елена Светлова

От бизнес-задачи к техническим требованиям: цели, ограничения, SLA и сценарии роста

В переговорной спорили уже двадцать минут, хотя слово было всего одно — «надежность».

— Нам нужна сеть, которая не падает, — сказал руководитель продукта.

— Сеть не падала ни разу за последний месяц, — возразил ИТ-директор.

— А мы получили триста жалоб, что оплата зависает, — заметила начальница поддержки.

Финансовый директор поднял глаза от ноутбука:

— Только не предлагайте решение, которое стоит как новый склад.

На экране презентации по-прежнему красовалась исходная заявка: **«Сделать сеть побыстрее и надежнее к запуску нового сервиса»**. В этой фразе было много уверенности и почти не было требований. Что именно значит «побыстрее»? Для каких пользователей? Насколько надежнее? Что происходит, если сеть всё-таки недоступна? И сколько компания готова заплатить за снижение риска?

На эти вопросы нельзя ответить выбором маршрутизатора. Пока неизвестно, какой бизнес-сценарий нужно обеспе-

чить, новый маршрутизатор — всего лишь дорогая версия догадки.

Запрос на скорость, который оказался запросом на оплату

Компания «СеверМаркет» готовила запуск сервиса экспресс-заказов. Покупатель должен был выбрать товар, оплатить его в приложении и получить доставку в течение часа. Запуск планировали через шесть месяцев — как раз к сезонному пику продаж. Команда продукта пришла к ИТ с просьбой «ускорить сеть»: тестовые платежи иногда проходили с задержкой, а иногда приложение сообщало об ошибке.

Первый важный вопрос звучал не «Какое оборудование у вас стоит?», а так:

— Что именно не может сделать пользователь — и в какой момент?

Владелец продукта ответил не сразу. Затем описал сцену: человек собрал корзину, нажал «Оплатить», увидел вращающийся индикатор и через несколько секунд решил, что ничего не произошло. Нажал кнопку ещё раз. Иногда платёж проходил, иногда возникала ошибка, а иногда покупатель закрывал приложение и уходил.

Итак, бизнес-проблема была не в абстрактной скорости. Она касалась **завершения оплаты без неоправданной задержки и без потери доверия пользователя**. Сеть могла быть одной из причин, но требовалось выяснить её вклад наряду с приложением, базой данных, внешним пла-

тёжным провайдером и мобильным интернетом покупателя.

Это различие принципиально. Время отклика приложения — не то же самое, что задержка сети. Если платёж занимает три секунды, а сеть между компонентами отвечает за 40 миллисекунд, покупка сетевого оборудования сама по себе не решит проблему. Если же сетевые задержки скачут от 30 до 700 миллисекунд при нагрузке, этот участок уже стоит исследовать. Начинают с пользовательского сценария и измерений, а не с заранее выбранного виновника.

Четыре интервью — четыре значения слова «надежность»

Совещание продолжилось короткими разговорами с теми, кто видел проблему с разных сторон. Это не формальность: бизнес-требование редко лежит готовым документом на одном столе. Его приходится собирать по частям — как маршрут, известный каждому только на своём участке.

Владелец продукта описал пользователя, действие и ожидаемый результат:

— Нам важно, чтобы человек мог завершить оплату. Если ответ задерживается, он повторяет попытку. Мы боимся не только потерянной продажи, но и двойного списания.

Из этой фразы появились первые уточнения: какие устройства и регионы важны; какие этапы образуют путь оплаты; какое время ожидания пользователи считают приемлемым; как приложение обрабатывает повтор запроса; какой компонент сообщает об успешной операции.

Финансовый директор перевёл разговор в язык затрат и последствий:

— Сколько стоит час сбоя? И сколько стоит сделать так, чтобы он случался реже?

На вопрос нельзя честно ответить, умножив все незавершённые заказы на средний чек: часть покупателей вернётся, часть платежей восстановится, а часть заказов потеряется безвозвратно. Но финансовой команде можно предложить модель: оценить потерянную маржу, расходы на поддержку и компенсации, влияние на репутацию, а затем сравнить эти потери с ценой резервирования каналов, оборудования и эксплуатации. Бюджет — не досадная помеха проекту, а одно из его условий. Если на снижение риска выделена определённая сумма, это может исключить некоторые варианты архитектуры. Лучше узнать об этом до проектирования, а не после тендера.

Служба поддержки добавила то, чего не было в графиках продукта:

— Пользователь говорит «не работает», а мы не всегда понимаем, где именно. Проблему замечают по звонкам, а не по мониторингу. Если сбой затрагивает только один регион, мы узнаём об этом слишком поздно.

Значит, надёжность — это не только редкие отказы. Это ещё и способность быстро обнаружить проблему, определить её область и восстановить услугу. Показатель, который нельзя заметить, трудно улучшить; инцидент, который нель-

зя локализовать, обычно длится дольше, чем должен.

ИТ-команда показала текущую инфраструктуру. Оказалось, что два интернет-канала заведены в здание по одной физической трассе. На схеме они выглядели резервированием, но общий кабельный колодец превращал их в два билета на один и тот же автобус. Кроме того, резервный канал никто не проверял под полной нагрузкой, а переключение после отказа было настроено не для всех сервисов.

Так обнаружилось скрытое ограничение: «два провайде-ра» ещё не означает «два независимых пути». На надёжность влияют общие точки отказа — помещение, ввод кабеля, питание, пограничное устройство, DNS, облачный узел, команда дежурства. Резервирование на схеме и резервирование в реальной аварии — разные вещи.

Превращаем пожелания в измерения

После интервью команда переписала исходную просьбу. Вместо «сделать сеть быстрее» появилось: «Обеспечить предсказуемый сетевой путь для операций оплаты, измерять его задержку и потери, поддерживать работу сервиса во время сезонного пика и восстанавливать связность в согласованный срок».

Формулировка стала лучше, но проектировать по ней всё ещё рано. Её нужно разложить на проверяемые параметры.

Задержка. Среднее значение удобно для презентации и опасно для пользователя. Если большинство операций укладывается в 40 мс, а каждая двадцатая задерживается до 500

мс, именно эти редкие задержки могут создавать ощущение нестабильности. Поэтому для критичных пользовательских путей смотрят на процентилях: например, р95 — значение, ниже которого укладываются 95% измерений, а р99 — 99%. Важно заранее определить точки измерения, время суток, направления трафика и тип нагрузки.

Потери пакетов. Даже небольшая доля потерь способна вызвать повторные передачи и увеличить время ответа. Для платежного сценария недостаточно сказать «потерь быть не должно»: нужно назвать границы, место измерения, длительность окна и способ сбора данных.

Пропускная способность и запас. Канал, который справляется со средней нагрузкой, может захлебнуться в пике. Более того, по мере приближения загрузки к пределу очереди часто растут непропорционально: пакеты ждут своей очереди дольше, задержка становится нестабильной. Поэтому сеть нельзя планировать ровно под текущую среднюю нагрузку. Нужны данные о пиковом трафике, резерв на рост, а также проверка поведения при отказе одного из каналов или узлов.

Доступность. Её надо определить для конкретной услуги, а не для абстрактного набора устройств. «Маршрутизатор отвечает на ping» не доказывает, что пользователь может завершить оплату. Измерение должно отражать доступность критичного пользовательского пути — например, с помощью синтетических проверок из нескольких точек, кото-

рые регулярно выполняют тестовый запрос до согласованного сервиса.

Время восстановления и сохранность данных. RTO (Recovery Time Objective) — сколько времени допустимо потратить на восстановление услуги. RPO (Recovery Point Objective) — какой объём данных, измеренный во времени, допустимо потерять при восстановлении. Для сети обычно важнее RTO связности; RPO относится к данным и состоянию приложения. Но в платёжном сценарии они соприкасаются: восстановить сеть недостаточно, если после разрыва непонятно, прошла ли транзакция. Значит, нужно отдельно согласовать, как сервис проверяет статус операции и защищается от повторного списания.

На этом этапе у команды появился ещё один полезный принцип: **каждый показатель должен иметь способ проверки**. Требование «задержка низкая» не принимается тестом. Требование «p95 сетевой задержки между площадкой приложения и платёжным шлюзом не выше 50 мс в заданном окне нагрузки, измерение по согласованному методу» — уже можно проверить и обсудить.

SLA — обещание, за которым стоит цена

Слова SLA, SLO и внутренний целевой показатель часто используют как синонимы. Это всё равно что называть одним словом прогноз погоды, обещание зонтика и собственный план не промокнуть.

• **SLA (Service Level Agreement)** — соглашение об уров-

не услуги, обычно с условиями, исключениями и последствиями невыполнения. Оно может быть договорным обязательством перед клиентом или внутренним соглашением между командами.

● **SLO (Service Level Objective)** — целевой уровень услуги, к которому команда стремится и который регулярно измеряет. Например, доступность критичного пути оплаты не ниже 99,95% за календарный месяц.

● **Внутренний целевой показатель** — более строгая рабочая цель, оставляющая запас до обещанного уровня. Если внешнее обязательство — 99,95%, команде может быть разумно планировать 99,97%, чтобы отдельные инциденты и неточность измерения не съели весь резерв.

Обещанная доступность должна сопровождаться ответами на четыре вопроса: **что именно считаем доступным, откуда измеряем, за какой период считаем и что исключаем из расчёта**. Также нужен бюджет, соответствующий обещанию. Резервные каналы, независимые маршруты, мониторинг и круглосуточное реагирование стоят денег и требуют обслуживания. SLA без такого обеспечения — не план, а пожелание, написанное шрифтом покрупнее.

Полезно почувствовать масштаб процентов. Если считать непрерывную работу в течение года, доступность 99,9% соответствует примерно 8 часам 46 минутам простоя, а 99,99% — примерно 53 минутам. В 30-дневном месяце 99,95% — это около 22 минут. Реальные договоры могут считать до-

ступность иначе: например, включать только определённые часы, исключать согласованные работы или учитывать лишь подтверждённые инциденты. Поэтому одной цифры в презентации недостаточно: метод измерения — часть требования.

У «СеверМаркета» согласовали пока не клиентское обещание, а проектный ориентир: измерять доступность полного платёжного пути синтетическими проверками каждую минуту из трёх точек; цель — не ниже 99,95% за месяц после запуска; внутренний SLO для команды — 99,97%. Плановые работы и ошибки внешнего провайдера не исчезли из мира, но стороны решили явно описать, как их учитывать, прежде чем включать обязательство в договор.

Сезонный пик — не сюрприз, если спросить о нём заранее

До этого разговора «рост» означал: «Наверное, пользователей будет больше». Теперь команда попросила продукт и аналитику описать конкретные сценарии.

1. Обычный день: текущий объём заказов, известные суточные пики.

2. Сезонный максимум через шесть месяцев: прогноз по одновременным пользователям и числу операций, включая ожидаемую рекламную кампанию.

3. Быстрый рост: сценарий, при котором новая функция окажется популярнее прогноза.

4. Работа при отказе: поведение сервиса при потере од-

ного канала или площадки.

5. Изменение географии: подключение новых магазинов, регионов или облачных компонентов.

Для каждого сценария нужны исходные данные и допущения: текущая нагрузка, темп роста, запас, срок, к которому сеть должна быть готова. Если прогноз неточен — а он будет неточен, — это не повод отказаться от планирования. Это повод зафиксировать диапазон и предусмотреть этапы пересмотра. Например: спроектировать базовую ёмкость под подтверждённый пик, установить порог, после которого требуется расширение, и провести нагрузочное испытание за несколько недель до кампании.

У команды оказалось полгода, но это не означало, что шесть месяцев можно потратить на закупку. Сначала следовало измерить текущий путь платежа, найти узкие места и проверить предположения; затем — определить целевую нагрузку, собрать архитектуру и испытать её в условиях, похожих на пик. Если проверка покажет, что задержка возникает внутри приложения, сеть не придётся перестраивать ради чужой проблемы. Если же один канал не выдерживает нагрузку после отказа второго, это станет основанием для конкретного изменения.

Таблица, которая не даёт бизнесу и ИТ разойтись

Итог совещания команда оформила в таблицу трассировки: от цели — к сценарию, от сценария — к метрике, от метрики — к ограничению и проверке. Такая таблица помогает

увидеть, зачем нужен каждый технический показатель, и не забыть, как доказать, что требование выполнено.

Цель

Сценарий

Метрика

Ограничение

Проверка

Покупатель завершает оплату без неоправданной задержки

Пиковая нагрузка в сезонной кампании

p95 сетевой задержки между согласованными точками — не более 50 мс; доля потерь — не более 0,1 % в заданном окне

Измеряется только участок сети; полный отклик приложения учитывается отдельно

Нагрузочный тест и синтетические проверки; фиксированные точки, окно и формат отчёта

Сервис оплаты остаётся доступным

Обычная работа и отказ одного канала

SLO доступности платёжного пути — 99,97% в месяц; проектный минимум — 99,95%

Требуется независимость путей и мониторинг; доступность внешнего платёжного сервиса учитывается отдельно по согласованным правилам

Синтетические проверки из трёх точек; испытание переключения; месячный отчёт

Сбой не превращается в долгую остановку

Потеря основного канала

Восстановление связности — не более 5 минут

Предел зависит от механизма переключения и доступности дежурной команды

Плановая имитация отказа с замером времени и проверкой журналов

Сезонный рост не ухудшает качество

Прогнозный пик через шесть месяцев и сценарий роста выше прогноза

Поддержка утверждённого пикового числа операций с согласованным запасом; значения фиксируются после анализа телеметрии

Лимит бюджета и сроки поставки; прогноз пересматривается перед кампанией

Нагрузочный тест на утверждённом профиле; подтверждение резерва после отказа одного канала

Это пока каркас, а не спецификация оборудования. И это правильно: таблица фиксирует, **что должна обеспечивать система и как это проверить**, но оставляет проектировщику пространство выбрать подходящую архитектуру.

К ней добавили согласованное задание:

Обеспечить платёжный путь нового сервиса для пользователей «СеверМаркета» в обычном режиме и в период сезонного пика через шесть месяцев. После уточнения прогноза зафиксировать целевой профиль нагрузки и провести испытание с дополнительным резервом, согласованным владельцем продукта и ИТ-командой. Доступность платёжного пути измерять синтетическими проверками из трёх точек и оценивать за календарный месяц; целевой SLO — 99,97%, минимальный проектный уровень — 99,95%. Для согласованных сетевых участков контролировать p95 задержки и потери пакетов, отдельно от полного времени отклика приложения. При отказе основного канала восстановить связность не более чем за пять минут; переключение и работу под нагрузкой проверить до запуска. Ограничения бюджета, зависимости от внешнего платёжного сервиса и правила учёта плановых работ документировать до утверждения клиентского SLA.

Так выглядит переход от «сделать сеть побыстрее» к заданию, которое можно обсуждать, проектировать и принимать. Оно не обещает невозможного и не прячет спорные места в слове «надёжность». Вместо этого оно связывает пользовательскую задачу с измерениями, сроками, ценой и проверкой результата.

Короткое упражнение перед стартом проекта

Возьмите ближайшую сетевую инициативу и попробуйте

заполнить для неё одну строку таблицы. Начните не с модели оборудования, а с пяти вопросов:

1. Кто пользователь и что он пытается сделать?
2. Какой отказ или задержка мешают этому сценарию?
3. Какой показатель докажет, что стало лучше?
4. Какие ограничения — бюджет, срок, старые системы, договоры — нельзя игнорировать?
5. Как и когда вы проверите результат?

Если на последний вопрос ответить нечем, требование пока нельзя принять. Если на первый — «всем нужна хорошая сеть», проект ещё не начал разговор с бизнесом.

На совещании «СеверМаркета» никто не нашёл устройство с надписью «никогда не падать». Зато команда выяснила, что именно должна защищать сеть: путь оплаты в период максимального спроса, в пределах определённого бюджета и с восстановлением, которое проверено не на словах. Это и есть полезный результат сбора требований — не список покупок, а общий договор о том, что считать успехом.

Сначала выясните, какой сценарий бизнес не может себе позволить потерять. Затем определите, насколько надёжно он должен работать, сколько простоя допустимо и какими данными это будет доказано. И только после этого выбирайте архитектуру и оборудование. Сеть, спроектированная таким образом, начинается не с маршрутизатора. Она начинается с правильного вопроса.

Основы сетевой архитектуры: уровни, компоненты, протоколы и физическая инфраструктура

В 9:17 утра Марина открывает ноутбук, подключается к корпоративному Wi-Fi и набирает адрес внутреннего портала. Значок беспроводной сети бодро сообщает: «Подключено». Сам портал работает — коллега за соседним столом только что открыл его. Но у Марины браузер крутит колесо, а затем выдаёт ошибку.

— Это сеть, — говорит служба поддержки. — Сервер исправен, — отвечает команда приложения. — Wi-Fi подключён, — возражают специалисты беспроводной сети.

Каждый прав, но только в пределах своей проверки. Проблема в том, что слово «сеть» здесь используют как название тёмного леса, в котором якобы всё связано со всем. Инженер поступает иначе: берёт один запрос Марины и прослеживает его путь — от сетевой карты до приложения. Не «работает ли сеть вообще?», а «до какого именно шага доходит этот запрос и что происходит на каждом шаге?».

Сначала — уточнить, что именно не работает

Фраза «не открывается приложение» может означать десятки разных вещей. Ноутбук не получил адрес. Wi-Fi слишком загружен. Имя сервера не разрешается. Маршрут поте-

рян. Межсетевой экран отклоняет соединение. Сервер отвечает, но браузер не доверяет его сертификату. Наконец, сеть может быть исправна, а само приложение — нет.

Архитектура помогает превратить эту россыпь вариантов в последовательность проверяемых гипотез. Для этого полезна слоистая модель: не потому, что реальная сеть аккуратно разложена по этажам, а потому, что каждый слой решает свой класс задач и оставляет свои наблюдаемые следы.

У модели OSI семь уровней. В повседневной работе чаще используют более компактное представление TCP/IP:

Уровень TCP/IP

Что обеспечивает

Примеры

Доступ к сети

Передачу по физической среде и в локальном сегменте
Wi-Fi, Ethernet, VLAN

Межсетевой

Доставку между IP-сетями

IPv4, IPv6, маршрутизация

Транспортный

Связь между процессами на узлах

TCP, UDP

Прикладной Смысл обмена и сетевые сервисы DNS, DHCP, HTTPS, NTP

Это не четыре отдельных коробки. Это четыре взгляда на один обмен. Инженер может спросить: «Есть ли радиосвязь?», «Есть ли IP-маршрут?», «Установилось ли TCP-соединение?», «Приняло ли приложение запрос?» — и каждый вопрос проверит другую часть пути.

До приложения: адрес, шлюз и имя

В нашем примере ноутбук Марины уже подключён к Wi-Fi, но одного подключения недостаточно. Устройство должно получить сетевые настройки: собственный IP-адрес, маску или длину префикса, адрес шлюза и адрес DNS-сервера. Обычно этим занимается DHCP.

Предположим, ноутбук получил адрес 10.24.18.57/24, шлюз 10.24.18.1, а DNS-сервер — 10.24.2.53. В типичном сценарии DHCP клиент отправляет широковещательный запрос, сервер отвечает, а устройство завершает обмен и получает аренду адреса. Если DHCP-сервер находится в другой подсети, запрос может передать ему DHCP relay на маршрутизаторе или L3-коммутаторе. Поэтому отсутствие адреса не обязательно означает поломку Wi-Fi: радио может работать, а путь до DHCP-сервера — нет.

Теперь браузеру нужно найти портал, например

portal.corp.example. Человек запоминает имя; IP-сеть доставляет данные по адресу. DNS преобразует одно в другое. Но DNS-запрос — уже отдельный сетевой обмен: ноутбук обращается к резолверу, обычно по UDP-порту 53, а при необходимости DNS может использовать TCP. Если резолвер находится не в локальной подсети, ноутбук отправит запрос через шлюз. Для этого ему нужно узнать MAC-адрес шлюза — например, с помощью ARP в IPv4 или Neighbor Discovery в IPv6.

Третий сервис часто остаётся за кадром, пока не начинает влиять на результат: синхронизация времени. NTP обычно работает поверх UDP, порт 123. Часы важны не только для аккуратных журналов. TLS проверяет сроки действия сертификатов; если часы устройства сильно спешат или отстают, исправный сервер может показаться браузеру недоверенным.

Именно это происходит с Мариной. DNS возвращает корректный адрес портала, а соединение с ним устанавливается. Но часы ноутбука отстают на восемнадцать минут: запросы к корпоративному серверу времени из пользовательского сегмента блокирует ошибочное правило межсетевого экрана. Сертификат портала ещё не считается действительным по времени устройства. Для пользователя результат выглядит как «сайт не работает». Для инженера это уже конкретная цепочка: сеть доставки довела соединение до проверки TLS, а зависимый сервис синхронизации времени не выпол-

нил свою роль.

Путь запроса: от радиоканала до серверного порта

Разберём путь точнее. Пусть портал находится по адресу 10.60.8.20, а Марина открывает <https://portal.corp.example>. Для ясности представим обычное HTTPS-соединение поверх TCP. Некоторые браузеры могут использовать HTTP/3 поверх QUIC и UDP — в реальной диагностике это нужно учитывать, но базовые принципы маршрута остаются.

1. Сетевая карта и радиосреда. Ноутбук передаёт кадры по Wi-Fi к точке доступа. На этом участке важны уровень сигнала, качество радиоканала, помехи и занятость эфира. Wi-Fi — общая среда: устройства делят время передачи, а повторные отправки могут замедлять обмен, даже если значок подключения не гаснет. Поэтому проверка «подключён» ещё не доказывает, что радиосвязь хорошая.

2. Точка доступа и коммутируемая сеть. В нашем примере точка доступа локально передаёт клиентский трафик в проводную сеть и связывает SSID корпоративного Wi-Fi с VLAN 118. В другой архитектуре точка доступа может туннелировать его к контроллеру — важно знать, где именно заканчивается радиосегмент и где трафик выходит в обычную сеть. Далее кадры проходят через порт коммутатора доступа. Коммутатор использует MAC-адреса и таблицу коммутации, чтобы пересылать кадр в нужном направлении; VLAN разделяет локальные широковещательные домены.

3. Шлюз и IP-маршрутизация. Адрес портала нахо-

дится не в подсети Марины, поэтому ноутбук отправляет кадр не непосредственно серверу, а MAC-адресу шлюза 10.24.18.1. IP-пакет при этом адресован порталу: источник — ноутбук, назначение — сервер. Маршрутизатор принимает кадр, извлекает IP-пакет, выбирает следующий переход по таблице маршрутизации и создаёт новый канальный кадр для следующего участка. На каждом маршрутизируемом переходе меняется L2-обрамление; IP-адреса обычно сохраняются, хотя NAT может их преобразовать. Также маршрутизатор уменьшает TTL в IPv4 или Hop Limit в IPv6.

4. Политики и транспорт. Пакет проходит через межсетевой экран, который проверяет, разрешено ли этому источнику соединяться с сервером на TCP-порт 443. В отличие от простого маршрутизатора, экран может отслеживать состояние соединения и учитывать правила доступа. Затем устанавливается TCP-соединение: ноутбук и сервер обмениваются служебными пакетами, прежде чем передавать данные приложения. Если политика запрещает TCP 443, браузер не дойдёт до проверки сертификата.

5. TLS и приложение. После установления TCP браузер и сервер начинают TLS-обмен: согласуют параметры шифрования, проверяют сертификат и только затем могут защищённо передавать HTTP-запрос. У Марины TCP работает, сервер отвечает, но браузер отклоняет сертификат из-за неверного времени. Значит, наблюдаемая ошибка возникла выше, чем физическая передача и IP-маршрут. После вос-

становления доступа к серверу времени и синхронизации часов тот же запрос проходит проверку и портал открывается.

Типовая цепочка выглядит так:

```text

**Ноутбук**

| **Wi-Fi: радиоканал, кадры 802.11**

▼ Точка доступа | VLAN 118: Ethernet или туннель к контроллеру ▼ Коммутатор доступа ▼ Шлюз VLAN / L3-коммутатор | IP-маршрутизация ▼ Межсетевой экран | проверка политик, при необходимости NAT ▼ Маршрутизатор / магистральная сеть ▼ Серверная сеть ▼ Сервер портала: TCP 443 → TLS → HTTPS-приложение

Это пример, а не универсальный шаблон: в другой сети шлюз и экран могут быть совмещены, между кампусом и сервером может находиться WAN, а точка доступа может передавать трафик через контроллер. Архитектура должна описывать реальные границы и переходы, а не идеализированную картинку.

**Инкапсуляция: почему у каждого уровня свои улики**

Когда браузер отправляет данные, каждый уровень добавляет к ним информацию, необходимую следующему участнику пути. HTTP-данные помещаются в TLS, TLS — в TCP-сегмент, тот — в IP-пакет, а IP-пакет — в кадр для конкретного канального участка. Это называется инкапсуляцией.

Упрощённо один фрагмент обмена выглядит так:

```text

Кадр Wi-Fi/Ethernet

- └── IP-пакет: ноутбук → сервер
- └── TCP-сегмент: временный порт → 443
- └── TLS-данные
- └── HTTP-запрос

Название единицы данных меняется вместе с уровнем: кадр на канальном, пакет на сетевом, сегмент TCP или датаграмма UDP на транспортном. Это не просто терминологическая аккуратность. Она помогает понять, где искать сбой. Коммутатор доступа обычно принимает решения по канальным адресам; маршрутизатор — по IP и таблице маршрутов; межсетевой экран может учитывать адреса, порты и состояние соединения; приложение — содержание запроса.

Заголовки также показывают, почему одного «адреса устройства» недостаточно. MAC-адрес нужен для доставки в пределах локального канального участка. IP-адрес помогает доставлять пакеты между подсетями. Порт указывает на конкретный сетевой процесс на конечном узле. Имя DNS удобно человеку и приложению, но само по себе не пересылает пакеты. При диагностике полезно спрашивать не «какой адрес у сервера?», а «какой тип адреса, на каком участке и для какой задачи?».

Физическая инфраструктура — не декорация

Логическая схема может выглядеть безупречно и всё

равно опираться на плохое соединение. В проводной сети медные кабели, патч-панели, розетки, порты и трансиверы должны соответствовать скорости и типу линии. Оптика соединяет устройства световым сигналом; тип волокна и оптических модулей должен быть совместим. PoE позволяет питать точки доступа и телефоны через Ethernet, но при проектировании нужно учитывать бюджет питания коммутатора.

Индикатор порта показывает, что устройства установили связь, а не то, что линия безошибочна. Счётчики CRC-ошибок, отбрасываний и повторных передач могут указать на повреждённый кабель, неудачную оптику или перегруженный интерфейс. В Wi-Fi проверяют не только уровень сигнала, но и отношение сигнал/шум, загрузку канала, потери и повторы. Полезно записывать, где находится физический порт, куда ведёт кабель, к какой VLAN он относится и кто отвечает за этот участок.

Физический уровень задаёт ограничения для всех верхних. Например, максимальный размер IP-пакета на пути ограничен MTU участков. Если отправитель считает, что может передать крупные пакеты, а где-то по пути они не проходят и обнаружение Path MTU работает некорректно, некоторые приложения могут «зависать» странным образом: короткие обмены идут, а передача больших объёмов — нет. Значит, осмысленная диагностика связывает оптику и кабель с IP и приложением, а не считает их разными мирами.

Кто за что отвечает

Послойная модель полезна и организационно. Команда Wi-Fi отвечает за радиопокрытие, точки доступа и их контроллеры; команда кампусной сети — за коммутаторы, VLAN и шлюзы; команда безопасности — за правила межсетевого экрана; команда платформы — за DNS, DHCP и синхронизацию времени; команда приложения — за службу и её конечную точку. В небольшой организации эти роли могут принадлежать одним людям, но границы систем всё равно существуют.

Для каждой границы стоит документировать контракт: какой интерфейс соединён с каким устройством, какая VLAN и подсеть используются, кто является шлюзом, какие протоколы и порты разрешены, где находится сервис и кто его сопровождает. Это особенно важно при изменениях. Запись «точка доступа подключена к порту 17» мало помогает, если неизвестно, куда ведёт порт, как настроен trunk и какая VLAN выдаётся клиентам.

Наконец, исправный путь в одну сторону ещё не гарантирует исправный обмен. Ответ сервера должен вернуться к клиенту; обратный маршрут и политики могут отличаться. Проверка только исходящего направления не обнаружит асимметричную маршрутизацию или правило, блокирующее ответы.

Упражнение: нарисуйте путь конкретного пакета

Выберите приложение, которым пользуетесь на работе. Нарисуйте «сеть офиса целиком»: проследите один конкрет-

ный запрос — например, открытие внутренней страницы.

1. Запишите имя приложения и IP-адрес, который возвращает DNS. Уточните, откуда устройство получает DNS-сервер.

2. Зафиксируйте адрес клиента, маску, шлюз и VLAN. Если данных нет — найдите их командами `ipconfig /all` в Windows или `ip address` и `ip route` в Linux.

3. Нарисуйте физический путь: устройство → точка доступа или розетка → коммутатор → шлюз → межсетевой экран → серверная сеть → сервер.

4. На каждом переходе подпишите, что передаётся: радиокадр или Ethernet, локальный MAC-адрес назначения, IP-источник и IP-назначение, транспорт и порт.

5. Отдельно отметьте DNS, DHCP и синхронизацию времени. Они могут быть необходимы для работы приложения, но не обязаны находиться в пути каждого его пакета.

6. Для каждого участка задайте проверяемый вопрос: есть ли линк и нет ли ошибок? Правильно ли выдан адрес? Разрешается ли имя? Достижим ли шлюз? Установилось ли TCP-соединение? Проходит ли TLS-проверка? Получило ли приложение ответ?

Сверять нужно не только схему, но и наблюдения: состояние интерфейсов, таблицы MAC-адресов и маршрутов, журналы меж сетевого экрана, DNS-ответы и ошибки приложения. Цель не в том, чтобы выполнить все возможные команды. Цель — найти границу, за которой успешная передача

сменяется неуспешной.

Сеть перестаёт быть магией, когда у каждого симптома появляется место в цепочке. Wi-Fi — не то же самое, что IP-доступность. IP-доступность — не то же самое, что открытый TCP-порт. Открытый порт — не гарантия успешного TLS, а успешный TLS — ещё не подтверждение, что приложение обработало запрос. Слои не заменяют архитектуру, но дают ей общий язык: из набора коробок и кабелей сеть превращается в систему с понятными интерфейсами, зависимостями и точками проверки.

Топологии и архитектурные модели: кампус, филиалы, дата-центр и распределённая сеть

В понедельник в девять утра головной офис компании «Северный контур» не мог открыть CRM. Сама CRM работала — в облаке, быстро и без ошибок. Не работала дорога к ней: запросы из филиалов шли сначала в головной офис, затем в корпоративный дата-центр, проходили через общий шлюз и только после этого выходили в интернет. Один путь, много поворотов и слишком много машин на одной развязке.

Когда-то такая схема была разумной. У компании было три офиса, собственный сервер и одна главная площадка. Теперь — головной офис, дата-центр, двадцать восемь филиалов, облачная CRM и план открыть ещё десять точек за год. Сеть выросла не просто в размере: у неё изменился характер движения. И проблема оказалась не в том, что «не хватает скорости», а в том, что прежняя форма сети больше не соответствовала жизни компании.

На архитектурной сессии директор по инфраструктуре развернул карту площадок и спросил: «Какую топологию нам выбрать?» Инженер ответил вопросом: «Для какого трафика, какого масштаба и какого отказа?» Это был более по-

лезный вопрос. Топология — не модный рисунок из презентации и не обещание, что «всё будет резервировано». Это устройство дорог, по которым данные попадут к своим целям, и правила, по которым сеть переживёт поломку одной из этих дорог.

Сначала понять, кто куда едет

На карте «Северного контура» было три основных типа трафика.

- Пользователи в филиалах обращались к облачной CRM и видеосервису.
- Торговые приложения филиалов связывались с системами в дата-центре.
- Сотрудники головного офиса обменивались данными с внутренними сервисами и друг с другом.

Это не три одинаковых потока. Облаку важен быстрый и надёжный выход в интернет. Торговому приложению — предсказуемый доступ к дата-центру. Видеозвонку — стабильная задержка и отсутствие потерь, а не просто высокая пропускная способность на бумаге. Если отправлять весь трафик по одной и той же дороге только потому, что она уже проложена, сеть будет устроена удобно для схемы, но не обязательно для бизнеса.

Полезно различать физическую и логическую топологии. Физическая показывает, где проходят кабели и линии связи, к каким коммутаторам подключены устройства и какие площадки соединены напрямую. Логическая — какие сети ви-

дят друг друга, где проходят маршруты, какие туннели связывают площадки и через какие политики проходит трафик.

Эти карты могут не совпадать. Физически филиалы могут подключаться к двум региональным узлам, а логически — к десяткам других площадок через защищённую оверлейную сеть. И наоборот: несколько отдельных физических линий могут предоставлять один логический путь. Такое расхождение полезно, но его нельзя считать волшебством: логический резерв не поможет, если оба «разных» маршрута идут по одному кабельному каналу или зависят от одного оператора.

Поэтому хорошая архитектура начинается с нескольких простых вопросов:

1. Какие приложения и площадки должны взаимодействовать?
2. Какой путь для каждого взаимодействия приемлем по задержке и пропускной способности?
3. Что должно продолжать работать при отказе устройства, линии, площадки или оператора?
4. Сколько сложных связей команда сможет надёжно построить, проверить и поддерживать?

Ответы постепенно проявляют подходящую форму сети.

Кампус: порядок вместо бесконечных перемигиваний

Головной офис «Северного контура» занимал два здания. В каждом были этажные коммутаторы доступа: к ним подключались компьютеры, телефоны, точки Wi-Fi и принтеры. Эти коммутаторы сходились к уровню распределения, где

объединялись сети этажей, применялись политики и маршрутизировался трафик. Выше находилось ядро — быстрый транзит к дата-центру, интернету и другим частям кампуса.

Так устроена иерархическая модель: **доступ — распределение — ядро**. Её сила не в том, что три уровня обязательны, а в разделении ролей. Доступ подключает пользователей и устройства; распределение объединяет локальные сегменты и задаёт правила; ядро быстро переносит трафик между крупными блоками.

Для большой площадки такая структура похожа на городской транспорт: местные улицы не должны по отдельности вести к каждому зданию города, а магистрали не должны разбираться с каждым дворовым выездом. Разделение упрощает диагностику и позволяет добавлять новые корпуса, не переплетая соединения между всеми коммутаторами.

Но иерархия не означает, что нужно строить три этажа ради красоты. Небольшому офису или компактному кампусу может хватить модели **collapsed core**, где функции распределения и ядра объединены. Это дешевле и проще в управлении. Цена — меньший запас для роста и более крупная область воздействия при отказе центрального оборудования. Пара резервных устройств не снимает это ограничение автоматически: надо ещё продумать питание, соединения, маршрутизацию и переключение.

Для «Северного контура» выбрали кампусную иерархию с парой узлов распределения, независимыми линиями от

крупных блоков и чёткими границами маршрутизации. Там, где отказ одного этажного коммутатора допустим до приезда техника, сделали менее дорогой резерв. Там, где отключение корпуса останавливало важную операцию, предусмотрели более независимые пути и оборудование.

Именно так следует относиться к резервированию: не как к флажку «есть/нет», а как к ответу на вопрос, какие последствия отказа организация согласна принять. Две линии в одном кабельном лотке дают резерв от поломки порта, но не обязательно от строительных работ. Два коммутатора в одной стойке могут пережить отказ устройства, но не потерю питания в серверной.

Филиалы: лучи, кольца и управляемые обходы

С филиалами картина другая. В небольшой точке обычно не нужен сложный кампус: несколько пользовательских устройств, кассовое или производственное оборудование, беспроводная сеть и канал к центральным сервисам. Классическая модель — **hub-and-spoke**, или «звезда»: филиалы подключаются к одному или нескольким центральным узлам.

У такой схемы есть достоинства. Её легко объяснить, контролировать и защищать: политика доступа задаётся в центральных точках, а число связей не превращается в клубок. Она хорошо работает, если приложения действительно размещены в центре и филиалам не нужно интенсивно общаться друг с другом.

Слабое место очевидно: центральный узел становится важным перекрёстком. Если через него идёт и доступ к дата-центру, и интернет-трафик филиалов, и связь между площадками, его отказ или перегрузка затрагивает всех. Даже без отказа такой маршрут может быть неразумным: филиал в Казани отправляет запрос в облачный сервис через центральный офис в Москве и получает ответ по длинной дороге.

Первый шаг к устойчивости — два центральных узла с независимыми подключениями. Но «независимость» надо проверять, а не рисовать: разные ли у каналов операторы, трассы и точки входа? Работают ли оба узла при потере одного? Не зависит ли переключение от единственного устройства или специалиста, который помнит пароль?

Следующий вариант — **частичная mesh-связность**: напрямую соединяются только те площадки, которым это действительно нужно. Например, два региональных центра получают пути друг к другу, а малые филиалы подключаются к ближайшему узлу. Такая модель уменьшает число обходных маршрутов и может снизить задержку для выбранных потоков, не превращая каждую площадку в отдельный проект.

Полная mesh-связность соединяет каждую площадку с каждой. Для небольшого числа критичных узлов это может быть оправдано. Но количество возможных пар быстро растёт: при n площадках их $n(n-1)/2$. Для тридцати филиалов это уже 435 пар. На практике не каждую из них обяза-

тельно строить отдельной линией, однако логические связи, маршруты, политики и сценарии проверки всё равно требуют внимания. Полная mesh — не «резерв на всякий случай», а выбранная цена за прямые пути и дополнительные возможности обхода.

Программно-определяемая WAN, или SD-WAN, может упростить управление такими связями: централизованно задавать политики, выбирать канал по условиям и создавать защищённые туннели поверх разных транспортов. Но SD-WAN — не топология сама по себе. Это способ управлять связностью. Он не создаст второй физический путь, если у филиала есть только один канал, и не избавит от необходимости разбираться, куда в итоге пойдёт трафик.

Для «Северного контура» малые филиалы подключили к двум региональным узлам, а критичные площадки получили дополнительные прямые или резервные связи. Для облачной CRM предусмотрели локальный выход в интернет там, где это разрешали требования безопасности. Так запрос перестал делать крюк через штаб-квартиру. Трафик к внутренним системам по-прежнему шёл по контролируемому пути в дата-центр.

Это разделение особенно важно при выборе **локального выхода в интернет**. Он уменьшает задержку и нагрузку на центральные каналы, но переносит на филиал часть требований к защите, фильтрации, мониторингу и резервированию. Экономия на маршруте не должна обернуться десятию раз-

ными и непроверенными правилами безопасности.

Дата-центр: когда большинство разговоров происходит внутри

В кампусе много трафика направлено к общим сервисам. В современном дата-центре серверы и приложения часто активно обмениваются данными между собой: фронтенд обращается к базе, микросервис — к очереди, система аналитики — к хранилищу. Это называют **восточно-западным трафиком**. Он делает архитектуру, рассчитанную только на «серверы в центре и пользователей по краям», недостаточной.

Для таких задач подходит модель **leaf-spine**. Каждый коммутатор leaf подключён ко всем коммутаторам spine, а серверы и другие конечные устройства подключаются к leaf. Путь между двумя leaf проходит через spine. Это даёт предсказуемую структуру и позволяет распределять потоки по нескольким путям с помощью равностоимостной маршрутизации — ESMR.

Модель удобна, когда много систем обменивается трафиком внутри дата-центра и нужно расширять сеть добавлением узлов соответствующего уровня. Но leaf-spine не означает автоматически «без блокировок» или «неограниченную скорость». Всё зависит от числа и пропускной способности соединений, возможностей оборудования и профиля нагрузки. Отношение суммарной пропускной способности серверных портов к пропускной способности восходящих соединений

называют коэффициентом переподписки. Если нагрузка одновременно пойдёт по всем потокам, узкие восходящие каналы могут стать бутылочным горлышком.

Эта модель также требует дисциплины в маршрутизации, адресации и эксплуатации. Больше путей дают больше возможностей распределять трафик, но неверная настройка может ухудшить предсказуемость или затруднить поиск неисправности. Если в небольшом дата-центре почти весь трафик идёт от серверов к внешним пользователям, сложность leaf-spine может быть избыточна. Архитектуру выбирают под обмен данными, а не под привлекательность схемы.

Дальность — тоже элемент топологии

Карта площадок — это не только точки, но и расстояния между ними. По оптоволокну свет распространяется примерно со скоростью 200 тысяч километров в секунду. Это значит, что только физическое прохождение 1000 километров добавляет около 5 миллисекунд в одну сторону. Реальный маршрут может быть длиннее прямого, а оборудование и обработка пакетов добавляют свою задержку.

Пять миллисекунд не всегда заметны пользователю. Но к ним добавляются путь обратно, серверная обработка, перегруженные каналы и дополнительные сетевые переходы. Для веб-страницы такая сумма может стать ощутимой; для синхронной транзакции или интерактивного приложения — критичной. Поэтому близость на карте важна, но важнее фактический маршрут и требования приложения.

В облачной архитектуре это приводит к нескольким практическим решениям: выходить к облаку из ближайшего региона, использовать частное подключение там, где оправданы его стоимость и профиль трафика, или оставить резервный защищённый путь через интернет. Нет смысла покупать выделенный канал для редких некритичных обращений, если обычный защищённый доступ удовлетворяет требованиям. И нет смысла экономить на пути для транзакции, от которой зависит ежедневная работа филиала.

Архитектурная сессия: сравнить не моды, а последствия

На встрече инженеры нанесли на карту четыре модели. В таблице они оценивали не «хорошо» или «плохо», а примерную цену, устойчивость, задержку и сложность эксплуатации. Оценки относятся к типичным сценариям: конкретный проект может изменить их в любую сторону.

Модель

Цена

Устойчивость

Задержка

Сложность эксплуатации

Иерархический кампус

Средняя; растёт с резервированием

Хорошая при независимых путях и продуманной архитек-

туре

Предсказуемая внутри площадки

Средняя; роли уровней облегчают поиск проблем

Филиалы «звездой» к одному узлу

Низкая или средняя

Центральная точка — общий риск

Может быть высокой для удалённых сервисов

Низкая; просто контролировать, но легко перегрузить центр

Два региональных узла и выборочная mesh-связь

Средняя или высокая

Хорошая при независимых каналах и корректном переключении

Ниже для выбранных прямых путей

Средняя или высокая; растёт число сценариев

Полная mesh между площадками

Высокая

Много возможных путей, но результат зависит от общей инфраструктуры

Потенциально низкая между узлами

Высокая; много связей и политик

Leaf-spine в дата-центре

Средняя или высокая

Хорошая при нескольких независимых путях

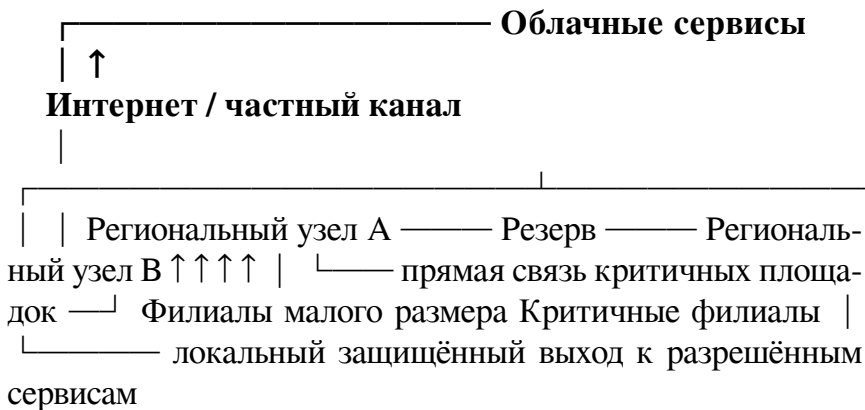
Предсказуемая между стойками

Средняя или высокая; нужна дисциплина маршрутизации и измерений

Таблица помогла не найти единственного победителя, а увидеть, где именно компании нужна каждая модель. Кампусу — иерархия. Небольшим филиалам — управляемая связь с региональными узлами. Критичным площадкам — выборочные прямые маршруты. Дата-центру — leaf-spine там, где интенсивность внутреннего обмена оправдывает эту архитектуру. Облаку — близкий и контролируемый выход, не зависящий от единственной центральной дороги.

В результате схема выглядела примерно так:

```text



## Головной кампус

доступ → распределение → ядро

| Дата-центр leaf ↔ spine | внутренние приложения

Это не готовый чертёж для копирования. На нём не показаны все маршруты, зоны безопасности, резервные каналы и ограничения конкретных сервисов. Его польза в другом: видно, кто к кому подключается, где сосредоточены зависимости и какие потоки обходят центральную площадку.

## Как выбрать модель для своей сети

Попробуйте провести собственную короткую архитектурную сессию.

**1. Нарисуйте площадки и расстояния.** Отметьте число филиалов, дата-центр, регионы облачных сервисов и каналы связи. Не ограничивайтесь логическими соединениями: запишите операторов, физические трассы и точки входа, если эти данные доступны.

**2. Добавьте поверх карты потоки приложений.** Для каждого важного приложения укажите источник, цель, направление трафика и допустимую задержку. Разделите потоки между площадками и внутри дата-центра.

**3. Опишите отказ, а не только резерв.** Что произойдёт при потере линии, центрального узла, электропитания или всего региона? Какую работу нужно сохранить немедленно, а какую можно восстановить позже?

**4. Сравните цену владения.** Учитывайте не только каналы и оборудование, но и мониторинг, тестирование пере-

ключения, сопровождение политик, обучение сотрудников и время диагностики.

**5. Проверьте, что резерв действительно независим.** Два маршрута, проходящие через одну шахту, одного оператора или одну точку питания, могут оказаться одним маршрутом под разными именами.

**6. Зафиксируйте компромисс.** Для каждого решения напишите, что оно улучшает, чего стоит и какое ограничение оставляет.

Этот разбор быстро показывает, что универсальной топологии нет. Простая звезда может быть отличным выбором для десятка небольших филиалов и плохим — для сети, где каждый филиал постоянно обращается к облаку. Полная mesh может оправдать себя для небольшой группы узлов с жёсткими требованиями к доступности, но стать дорогой и трудноуправляемой при массовом расширении. Leaf-spine может дать удобную основу для плотного обмена между серверами, но не решить проблем внешних каналов или неверно выбранных границ безопасности.

Сеть становится зрелой не тогда, когда на схеме появляются самые сложные фигуры, а когда её форма соответствует реальным потокам и последствиям отказов. «Северный контур» не стал соединять каждый филиал с каждым и не отправил всё через штаб-квартиру. Он разделил роли: кампус объединяет локальные сети, региональные узлы связывают филиалы, дата-центр обслуживает внутренние приложения,

облачные сервисы доступны по более короткому пути. Резервирование появилось там, где его цена оправдывалась последствиями простоя.

Перед тем как проектировать адреса и VLAN, стоит добиться ясности в этой карте: какие устройства должны быть соседями по сети, какие — только обмениваться маршрутизируемым трафиком, а какие не должны видеть друг друга вовсе. Именно от этих границ зависит следующий уровень архитектуры.

# **Адресное пространство и сегментация: IPv4, IPv6, VLAN, подсети и сетевые сервисы**

В понедельник утром новый филиал должен был впервые подключиться к корпоративной сети. К полудню сотрудники уже входили в почту, телефоны звонили, принтеры проснулись. Но складская система не открывалась: её сервер находился по адресу, который сеть филиала считала локальным, а головной офис — тоже локальным. Один и тот же адрес оказался по обе стороны туннеля.

Проблема не была в маршрутизаторе и не решалась заменой коммутатора. За несколько лет в разных местах компании адреса назначали «на глаз»: где-то использовали диапазон для новой лаборатории, где-то — «временную» подсеть из старого проекта, а пару принтеров настроили вручную. Теперь соединить площадки оказалось труднее, чем построить между ними канал.

Так сеть предъявляет счёт за адресные решения, принятые без плана. Адресация кажется технической бухгалтерией — пока не начинает определять, какие площадки вообще могут общаться, насколько сложны маршруты, где проходят границы безопасности и сколько времени занимает подключение нового офиса.

## **Адрес — это не просто номер**

У каждого узла в сети должен быть адрес, но хорошая адресная схема делает больше, чем позволяет найти этот узел. Она помогает понять, где он находится, к какой зоне относится и через какие маршруты до него добраться. Если адреса организованы иерархически, их можно объединять в более крупные блоки и объявлять в сети короткие сводные маршруты. Если они разбросаны случайно, маршрутизаторам приходится хранить больше отдельных записей, а инженерам — разбираться в карте, которую никто не рисовал.

Адресация влияет и на безопасность. Когда ноутбуки сотрудников, кассы, камеры, серверы и оборудование управления находятся в одной подсети, сеть не знает, что это разные категории устройств. Их трафик может свободно перемещаться внутри общего сегмента, если его не ограничивают другие механизмы. А когда устройства разделены на осмысленные зоны, между ними можно применять разные правила доступа.

Наконец, адресный план влияет на автоматизацию. DHCP, DNS, мониторинг, инвентаризация и политики безопасности работают надёжнее, когда сеть имеет предсказуемую структуру. Схема, понятная только человеку, который её создал, — не схема, а личная память, уязвимая к отпуску и увольнению.

**Шаг 1. Найдите хаос, прежде чем строить порядок**  
Начинать планирование с выбора красивых диапазонов

опасно. Сначала нужно выяснить, что уже используется.

Соберите сведения о площадках, VPN-соединениях, облаках, дата-центрах и сетях, которые могут появиться после слияния с другой компанией. Проверьте адреса маршрутизаторов, серверов, пользовательских устройств и сетевых сервисов. Ищите статические настройки, забытые DHCP-области и подсети, назначенные подрядчиками. Сопоставьте данные маршрутизаторов и DHCP с фактическими сведениями из DNS, систем мониторинга и IPAM — системы учёта адресов. Ни один источник не покажет всей картины, но расхождения между ними быстро обнаружат места, где документация давно отстала от жизни.

Особое внимание уделите пересечениям с удалёнными сетями и облачными платформами. Частный адрес вроде 10.20.4.15 не уникален в масштабе интернета, а значит, его могут одновременно использовать множество организаций. Но внутри вашей сети, её VPN и подключённых облаков он должен быть однозначным. Диапазон 100.64.0.0/10 тоже стоит учитывать отдельно: его используют операторы связи для общего адресного пространства, и это не частный диапазон RFC 1918. Не выбирайте его для корпоративной адресации без чёткой причины и проверки.

И составьте список устройств, которым адреса назначены вручную. «У принтера всегда был такой IP» — не аргумент в пользу архитектуры, но важная улика для миграции. Если адреса вручную прописаны на сотне камер, сначала нужно

спланировать, как изменить их настройки, а не просто объявить новый стандарт.

## **Шаг 2. Спроектируйте структуру блоков**

IPv4-адрес состоит из 32 бит; всего это около 4,3 миллиарда адресов, причём часть пространства зарезервирована для специальных целей. На одну организацию такой ёмкости уже не приходится надеяться — поэтому в локальных сетях обычно применяют частные адреса, а доступ в интернет организуют через трансляцию адресов и другие механизмы.

Запись CIDR, например /24, показывает длину сетевой части адреса: чем больше число после косой черты, тем меньше подсеть. Подсеть /24 содержит 256 адресов, из которых в традиционной IPv4-сети обычно 254 доступны узлам; /23 содержит 512 адресов, обычно 510 доступных. В реальном проекте часть диапазона может быть зарезервирована под шлюз, инфраструктуру и особые устройства. Не нужно каждый раз переводить маску в двоичный вид. Важно понять масштаб блока и границы, которые он задаёт.

Простой и масштабируемый план строится от крупного к мелкому:

1. Выберите общий корпоративный диапазон.
2. Выделите отдельные блоки регионам, дата-центрам или типам площадок.
3. Назначьте каждой площадке собственный непрерывный блок.
4. Разделите блок площадки на подсети, соответствующие

сетевым зонам.

**5.** Зафиксируйте, какие адреса отведены под шлюзы, инфраструктуру, DHCP и резервы.

Например, организация может зарезервировать под свои площадки часть частного пространства 10.64.0.0/10. Главный кампус получает 10.64.0.0/16, а восточный регион — блок 10.80.0.0/16. Из регионального блока можно выделять филиалам более мелкие диапазоны. При свёртывании маршрутов множество маршрутов филиалов представляется одним более крупным блоком.

Агрегация уменьшает объём маршрутной информации, но это не волшебство. Если объявить агрегат для диапазона, в котором есть незаполненные или недоступные части, трафик к ним может попасть на сеть, не знающую точного маршрута. Поэтому сводные маршруты нужно проектировать вместе с внутренней маршрутизацией и механизмами обработки недоступных адресов. Агрегация полезна, когда соответствует реальной иерархии сети, а не просто красиво выглядит в таблице.

### **Шаг 3. Разделите сеть по ролям, а не по случайности**

VLAN — это способ разделить канальную сеть на отдельные логические широковещательные домены. Проще говоря, устройства в разных VLAN не ведут себя как участники одной общей локальной сети, даже если подключены к одним и тем же коммутаторам. Чтобы трафик прошёл из одной

VLAN в другую, нужен маршрутизируемый путь, а на нём можно применять правила доступа.

Типичная ошибка — считать, что сама VLAN уже обеспечивает полноценную безопасность. Она задаёт границу второго уровня, но не запрещает устройствам взаимодействовать через маршрутизатор. Для этого нужны списки контроля доступа, межсетевой экран или другие механизмы политики. И наоборот: устройства в одной VLAN обычно могут обращаться друг к другу напрямую, если трафик не ограничен дополнительными средствами.

Создавайте сегменты по смыслу и требованиям. Сотрудникам, гостям, голосовым устройствам, серверам, камерам, системам управления и оборудованию здания могут требоваться разные правила. Не обязательно создавать отдельную VLAN для каждой команды или каждого этажа: избыточная сегментация тоже усложняет эксплуатацию. Хороший вопрос для каждой зоны звучит так: «У этих устройств одинаковые требования к доступу, адресации, надёжности и управлению?» Если ответ нет, возможно, им не место в одном сегменте.

Номер VLAN и IP-подсеть часто связаны соглашением, но одно не обязано автоматически кодировать другое. VLAN ID имеет значение в контексте канальной сети, а IP-подсеть — в маршрутизации. Номер VLAN может повторяться на разных площадках, если между ними не растягивают один и тот же второй уровень. Обычно растягивать такие домены

через большие расстояния без веской причины не стоит: это связывает площадки сильнее, чем требуется, и увеличивает область последствий некоторых сбоев.

#### **Шаг 4. Закрепите договорённости в рабочей таблице**

Ниже — пример, а не универсальный шаблон. Он показывает, как адресный план соединяет площадки, зоны и VLAN в одну систему.

**Площадка или блок**

**VLAN**

**Назначение**

**Пример IPv4-подсети**

Главный кампус 10.64.0.0/16

10

Пользовательские устройства

10.64.10.0/23

Главный кампус

20

Голосовые устройства

10.64.20.0/24

Главный кампус

30

Серверы  
10.64.30.0/24

Главный кампус

# Конец ознакомительного фрагмента.

Текст предоставлен ООО «Литрес».

Прочитайте эту книгу целиком, [купив полную легальную версию](#) на Литрес.

Безопасно оплатить книгу можно банковской картой Visa, MasterCard, Maestro, со счета мобильного телефона, с платежного терминала, в салоне МТС или Связной, через PayPal, WebMoney, Яндекс.Деньги, QIWI Кошелек, бонусными картами или другим удобным Вам способом.