YELLOW PAPER

Полная техническая спецификация чисто платёжного L1-блокчейна. Технологии описаны по шагам жизненного цикла транзакции; каждый выбор обоснован и защищён против современных альтернатив – с формулами, схемами и честными оговорками. Для инженеров, которые решают, присоединяться ли к команде, и аналитиков, оценивающих перспективность.

Xync Network L1 · v1.0, июль 2026

Содержание

0. Модель системы. Почему «только платежи» — это суперсила

Сеть состоит из n валидаторов, из них не более f байзантийских (произвольно злонамеренных: лгут, молчат, сговариваются), n = 3f+1. Сеть частично синхронна: сообщения доходят за неизвестную, но конечную задержку δ. Примитивы: ed25519, SHA-256. Кворум q = 2f+1; любые два кворума пересекаются минимум по f+1 узлам, т.е. хотя бы по одному честному:

$$|Q_1 \cap Q_2| \geq q + q - n = (2f+1) + (2f+1) - (3f+1) = f+1 > f$$

Домен сознательно сужен до денег: **отправить**, **попросить**, **обменять** (255 валют + XYNC) — ни виртуальной машины, ни произвольного общего состояния. Это не «недо-платформа», а источник каждого нашего преимущества, потому что открывает доступ к теореме, недоступной универсальным чейнам:

Теорема домена (FastPay, 2020). Если каждым аккаунтом управляет один владелец, для корректности платёжной системы не нужен тотальный порядок транзакций — достаточно частичного порядка по каждому отправителю. *Интуиция:* конфликтовать могут только траты одного аккаунта; их упорядочивает сам владелец своим счётчиком seq. Переводы разных людей коммутируют: `apply(tx_A); apply(tx_B) ≡ apply(tx_B); apply(tx_A)` — балансы получаются одинаковые.

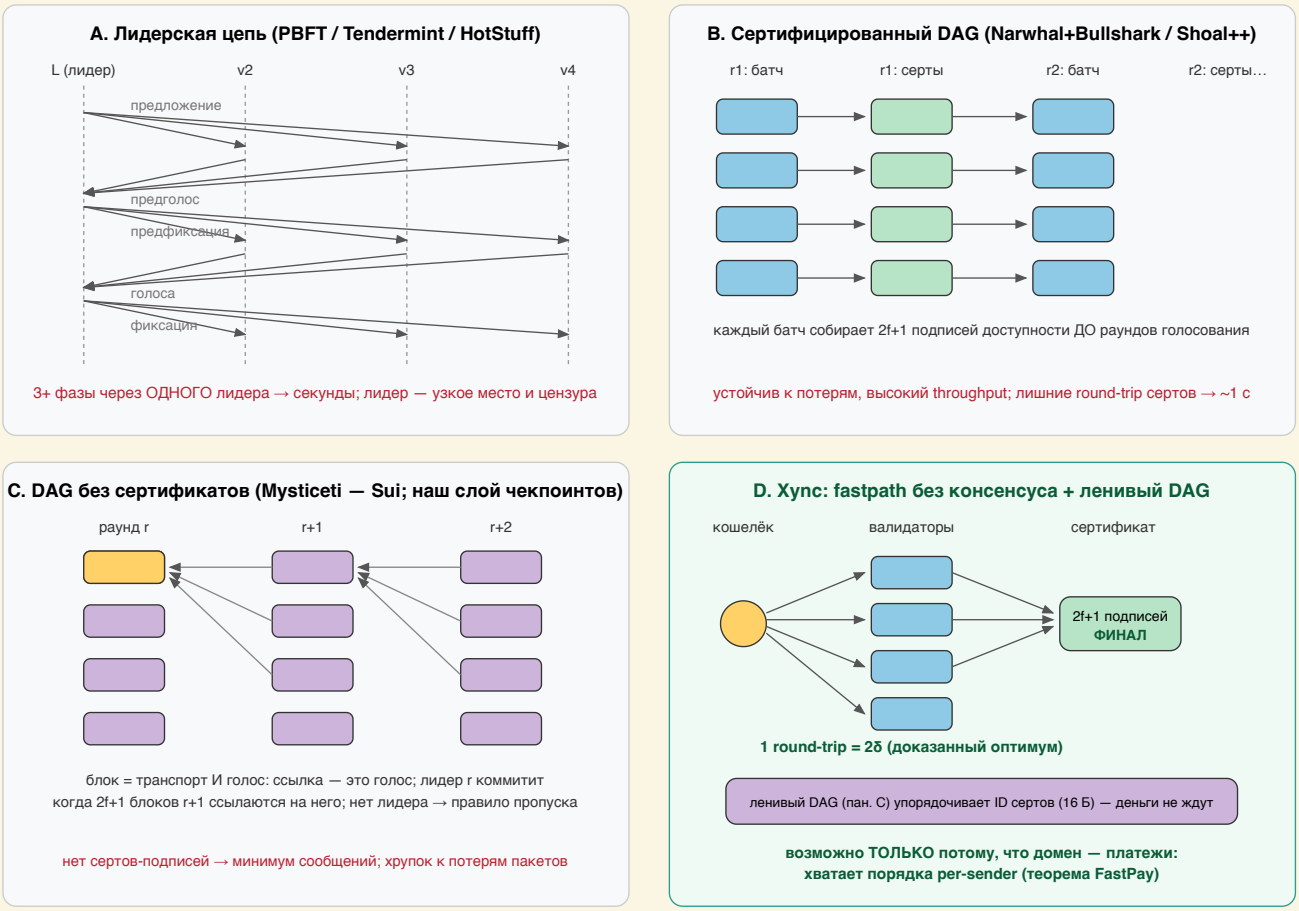
Следствие: консенсус (выбор единого порядка) на критическом пути платежа **избыточен**. Финализация возможна за один сетевой круг, и 2δ — доказанный нижний предел любой BFT-финальности (pod, 2025). Universal-чейны обязаны упорядочивать всё (смарт-контракты разделяют состояние) — им теорема недоступна, и поэтому они соревнуются в том, как быстро крутится консенсус, а мы его просто убрали с горячего пути.

Где проходит граница слоёв. Теорема покрывает ровно те операции, у каждого затронутого аккаунта которых есть владелец: перевод, запрос денег и P2P-своп (§14.2 — две ноги, каждую подписывает её владелец; такие свопы тоже коммутируют и живут в fastpath). Всё, что меняет **общее** состояние — резервы пулов, доли поставщиков ликвидности, позиции эмитентов, — владельца не имеет: два независимых обмена через один пул не коммутируют, потому что читают и пишут одни и те же резервы. Поэтому такие операции едут исключительно через ленивый DAG, в едином порядке (§8, §14.4). Это и есть честная цена мультивалютности: обмен через пул финализируется за ~1–2 с, а не за 0.3 с. Инвариант «общее изменяемое состояние меняется только в коммите по порядку DAG» проверяется машинно (§14.7).

1. Большое сравнение консенсусов 2020–2026 и защита выбора

Выбор алгоритма согласования – главное архитектурное решение любого L1: он определяет и задержку платежа, и стоимость байта, и то, кто способен остановить или цензурировать сеть. Поэтому ниже не перечень модных названий, а разбор всех живых семейств по состоянию на июль 2026: как каждое устроено, чем платит за свою скорость и почему в платёжной сети проигрывает выбранной нами связке. Proof-of-Work в обзор не входит вовсе: его вероятностная финальность измеряется десятками минут, а энергозатраты несовместимы с нодой в телефоне – оба свойства дисквалифицируют его до начала сравнения.

Схемы обмена сообщениями в семействах BFT (n=4 валидатора, f=1)



1.1. Как работает каждое семейство

A. Лидерские цепочки (PBFT 1999 → Tendermint/CometBFT → HotStuff → Jolteon/MonadBFT). Один лидер предлагает блок, все голосуют в 2–3 фазы (prevote/precommit), смена лидера по таймауту. HotStuff сделал голоса линейными (O(n) вместо O(n²)) и конвейеризуемыми; MonadBFT (2025) добавил спекулятивную финальность и защиту от tail-fork. Фундаментальные пределы семейства: каждый байт проходит через лидера (узкое место пропускной способности и точка цензуры/MEV), а финальность = несколько последовательных

фаз через весь мир: на практике 1–6 с. Для сети, которая обещает необратимый платёж быстрее, чем человек убирает телефон в карман, это дисквалифицирующая цена.

B. Certified DAG (Narwhal+Tusk/Bullshark 2022 → Shoal 2023 → Shoal++/Sailfish 2024 → Raptr 2025). Прорыв Narwhal: отделить РАССЫЛКУ данных от ПОРЯДКА. Каждый валидатор параллельно публикует батчи, собирает на каждый 2f+1 подписей доступности (сертификат батча), а консенсус упорядочивает уже маленькие сертификаты — отсюда чудовищная пропускная способность (100k+ TPS у всех). Bullshark строит поверх раунды-волны с якорями; Shoal++ выжал латентность до ~0.7–1 с тремя параллельными DAG и репутацией лидеров; Raptr (Aptos, 2025) добавил оптимистичный путь, сохранив устойчивость: при 1% потерь пакетов латентность растёт лишь на ~15%. Цена семейства: каждый батч ждёт раунд сертификации (лишний RTT и подписи) ДО того, как войдёт в порядок.

C. Uncertified DAG (Mysticeti 2024 → Mahi-Mahi 2024). Ход конём: выкинуть сертификаты совсем. Блок валидатора — одновременно транспорт и голос: ссылка на блок = голос за него (implicit voting). Лидер раунда r закоммичен, когда 2f+1 блоков раунда r+1 на него ссылаются; финальность за 3 подраунда (~0.4–0.65 с в проде Sui, v2 дал ещё −25–35%). Подписей на порядок меньше — идеально по вычислительной нагрузке. Слабость, подтверждённая независимо (arXiv 2025/877, Autobahn): без сертификатов доступности данные приходится дотягивать на критическом пути — 0.05% потерь пакетов раздувают латентность до 10×; один медленный валидатор добавляет 2 задержки на раунд.

D. Hashgraph (2016, открыт как Hiero/Apache-2.0 в 2024). Gossip-o-gossip'е: узлы обмениваются историей общения, и каждый локально вычисляет виртуальные голоса без их пересылки. Математически элегантно, aBFT. Почему не берём: финальность в секундах (2–5 с в Hedera), избыточная пересылка метаданных истории, нет отделения данных от порядка, свежих результатов по латентности семейство не показало — оно проигрывает и B, и C по обоим нашим приоритетам.

E. Alpenglow (Solana, 2025–2026). Замена PoH+TowerBFT: Votor — голосование в 1–2 раунда (при ≥80% стейка финал в один раунд, 100–150 мс), Rotor — рассылка erasure-кодированными шардами через семплированных ретрансляторов. SIMD-0326 принят 98%, тестнет с мая 2026, мейннет ~Q4 2026. Это нынешняя вершина ТОТАЛЬНОГО порядка. Цена: лидер-центричность (MEV/цензура), сложнейшая сетевая инженерия (erasure-коды, взвешенный выбор ретрансляторов), нулевая обкатка в бою.

F. Без консенсуса — consensusless (FastPay 2020 → Linera → Sui fastpath → pod 2025). Если у каждого затронутого аккаунта есть единственный владелец, договариваться о порядке не с кем и не о чем: валидаторы независимо проверяют транзакцию и независимо её подписывают, а 2f+1 подписей и есть финальность — один сетевой круг, без раундов, блоков и лидеров. pod.network довёл идею до промышленного вида: ~150 мс, 300k+ TPS, непрерывный поток аттестаций вместо цепочки блоков. Sui применяет ту же механику как «быструю полосу» для объектов с владельцем. Слабость ровно одна: общего порядка нет, поэтому произвольные смарт-контракты невозможны (нам они и не нужны), а для редких общесетевых событий нужен отдельный механизм — им у нас служит DAG-слой.

1.2. Сводная таблица

Протокол	Семейство	Финальность	Сообщений на tx	Статус 2026	Дисквалифицирующий фактор для Xync
Tendermint/CometBFT	A	1–6 с	O(n) через лидера ×3 фазы	prod (Cosmos)	латентность, лидер
HotStuff/Jolteon	A	~1–2 с	O(n) ×3	prod (Aptos до '25)	латентность
MonadBFT	A	~0.8 с спекулятивно	O(n)	prod (Monad)	спекулятивность + лидер
Hashgraph	D	2–5 с	gossip²	prod (Hedera)	латентность, трафик
DAG-Rider/Tusk	B	4+ раундов	сертификация каждого батча	research	латентность
Narwhal+Bullshark	B	~2–3 с	то же	был prod (Sui)	Sui сам ушёл на C
Shoal++	B	~0.7–1 с	то же	prod (Aptos)	лишний RTT сертификации
Sailfish	B/C	1 RBC + 1δ	–	research 2024	обкатка
Mysticeti (v2)	C	0.4–0.65 с	минимум	**prod (Sui)**	хрупкость к дропам
Mahi-Mahi	C	~3 подраунда	минимум	research	асинхронная сложность
Raptr	B+opt	суб-сек	оптимистичный путь	testnet/prod '26	нет fastpath для UX
Alpenglow	E	**0.1–0.15 с**	Rotor-шарды	мейннет ~Q4'26	лидер, необкатанность
FastPay/pod	**F**	**2δ ≈ 0.15–0.4 с**	**n аттестаций**	prod (pod)	нет порядка (для нас не минус)

1.3. Решение Xync: F + C = fastpath + ленивый DAG

Ни одно семейство не подходит целиком, потому что в платёжной сети живут события двух совершенно разных сортов, и мы обслуживаем каждый своим слоем.

Деньги идут по быстрой полосе — без консенсуса (семейство F, панель D схемы). Кошелёк рассылает транзакцию всем n валидаторам, каждый независимо её проверяет и подписывает, и первые же 2f+1 подписей, собранные вместе, образуют сертификат финальности. Один сетевой круг, никаких раундов и лидеров.

Всё остальное едет по медленной полосе — через ленивый DAG (семейство C, Mysticeti-lite). Регистрация аккаунта, наказание мошенника, смена набора валидаторов, чекпоинт, раздача комиссий — это события, для которых порядок принципиален (кто получил индекс #1000, чей fraud proof засчитан первым). Их немного, они не срочны, и они не задерживают ни один платёж. «Ленивый» означает буквально: пока событий нет, слой не производит ни байта.

Почему гибрид бьёт каждую чистую альтернативу:

- *Чистый C (Mysticeti-only):* платёж ждал бы 3 подраунда (0.5–1 с вместо 0.15–0.4) и, главное, наследовал бы хрупкость к потерям пакетов НА ПУТИ ДЕНЕГ. У нас DAG не на критическом пути: его 10×-деградация при дропах замедлит регистрации, но не платежи.
- *Чистый E (Alpenglow-путь):* сопоставимые миллисекунды, но ценой лидера (MEV/цензура), erasure-инженерии и года обкатки. Мы получаем ту же скорость из теоремы домена, а не из героической инженерии.
- *Чистый B (Shoal++/Raptr):* максимальная живучесть под потерями, но каждый батч платит RTT сертификации — при том, что для платёжного UX сертификация У НАС УЖЕ ЕСТЬ (аттестации fastpath) и она же даёт финальность.
- *Чистый F (pod-путь):* некому детерминированно исполнять регистрации, штрафы, смену валидаторов и делать чекпоинты для лёгких клиентов — pod выносит это во внешние механизмы; нам дешевле ленивый DAG.
- *D (Hashgraph):* проигрывает по латентности и трафику обоим слоям гибрида по отдельности.

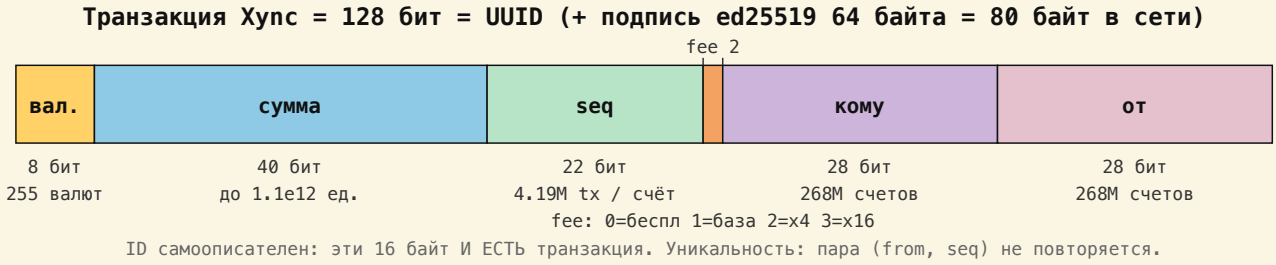
Честная цена гибрида (см. также §17): (1) эквивокация владельца — даже случайная, из-за бага кошелька — неотличима от атаки и карается; (2) любая будущая функциональность с общим изменяемым состоянием поедет только через DAG-слой с его латентностью; (3) fastpath требует от кошельков дисциплины seq (железное правило, §4).

2. Аккаунты и идентичность

Аккаунт: `{index, pubkey, seq, balances{cur: amount}}`. Индексы компактные (28 бит), выдаются последовательно системной транзакцией регистрации (~1 USD в казну — анти-Sybil аккаунтного уровня и финансирование, §11). #0 — оператор Xync: наследие живого прода XyncPay, при миграции id клиентов сохраняются 1:1 (§12).

Защита выбора. Хэш-адреса (BTC/ETH, 20–32 Б) бесплатны, но два таких адреса не влезли бы даже в половину 16-байтовой транзакции. Компактный индекс = дорогая одноразовая регистрация + дешёвая каждая транзакция; для платёжной сети трафик доминирует над взносом. Побочно: «красивые» короткие номера — бесплатная геймификация и статус (ICQ). Регистрационный взнос смягчается продуктово: реферальный ваучер (казна платит за приглашённого) и «первый входящий платёж ≥$2 окупает регистрацию получателя».

3. Транзакция: 128 бит




```
currency 8 | amount 40 | seq 22 | fee 2 | to 28 | from 28  = 128 бит
подпись ed25519(DOMAIN_TX ‖ tx_id)                          = 64 байта
итого на проводе                                            = 80 байт
```

Тело **и есть** идентификатор («tx = UUID»): ни отдельного хэша, ни сериализации, ноль накладных. Уникальность навсегда: пара (from, seq) не повторяется в течение жизни сети (счётчик не обнуляется).

Поле	Обоснование
currency 8	~100 фиатов + ~150 монет; реестр в genesis, per-currency scale (USD:3, BTC:8, XYNC:6)
amount 40	до $2^{40}-1 \approx 1.1 \cdot 10^{12}$ мин. единиц: $1.1 млрд/tx при scale 3 – достаточно
seq 22	4 194 303 исходящих на аккаунт пожизненно: физлицу навсегда, конвейеру 1000 tx/день – 11 лет, дальше саб-аккаунты (естественно для бухучёта)
fee 2	4 уровня покрывают реальный спрос на приоритет: 0=free (§5), 1=base, 2=×4, 3=×16; множители – параметр genesis
to/from 28	268 435 455 аккаунтов; v2 (33 бита, 8+ млрд людей) вводится по ДЛИНЕ КАДРА (80 Б = v1) без бита версии и без хардфорка

Защита выбора. (а) *timestamp в ID* (ранний дизайн, и ровно так работает сегодняшний прод XyncPay – `Transaction.pack`): коллизии одинаковых переводов в одну секунду, недоверенное время («проверить баланс в секунду отправки» нельзя – порядок определяет сеть, а не часы отправителя), нет replay-защиты. seq решает всё разом и дарит правило дырок (§5) и безопасную оффлайн-подпись (§13). (б) *ID = hash(тела)*: +16 Б каждой транзакции и потеря самоописываемости. (в) *UTXO*: для чистых платежей хуже account-модели по размеру и сложности кошелька. (г) *эпохи seq* (промежуточная версия): усложняли подписи, допускали повтор ID через эпоху – вырезаны в пользу 22-бит счётчика.

Сравнение: Bitcoin ≈ 250 Б, Ethereum ≈ 110 Б, Solana ≈ 230 Б, Sui ≈ 300 Б, **Xync = 80 Б**. При 4096 TPS (потолок лесенки, §8) это всего 320 КБ/с полезного трафика данных.

4. Шаг 1: подписание

Кошелёк знает свой seq (это его собственный счётчик — сети для подписи не нужно) и подписывает 16 байт. **Железное правило кошелька:** подписанные байты персистятся ДО первой отправки; любой ретрай шлёт идентичную копию; локальный счётчик seq инкрементится при КАЖДОЙ подписи, даже неотправленной. Переподписание того же seq с другим телом — криптографический автограф даблспенда (§10): протокол не может и не должен отличать «баг ретрая» от атаки.

5. Шаг 2: приём, правило дырок, резервирование, free-полоса

Валидатор на входящую tx (от клиента или из gossip):

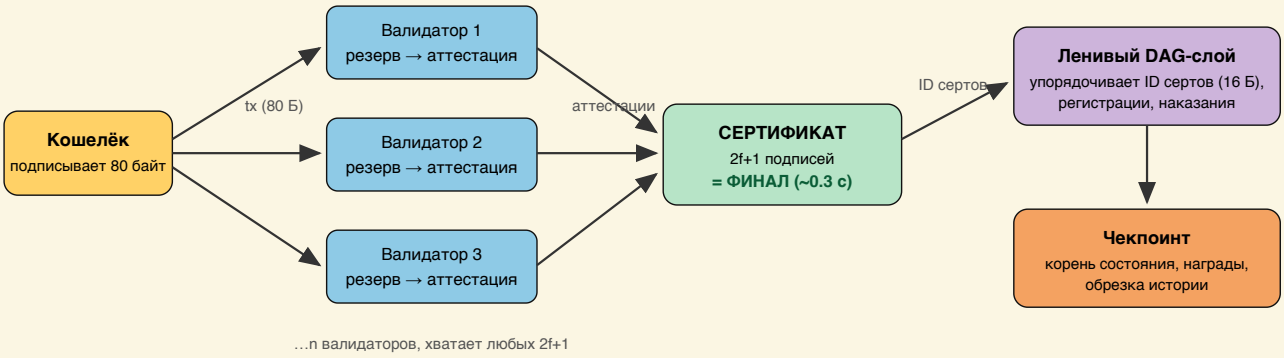
1. кодек-диапазоны, существование аккаунтов, бан-флаг;
2. **правило дырок**: если tx.seq > ожидаемого — валидатор не знает всех предыдущих tx отправителя; транзакция ПАРКУЕТСЯ до прибытия недостающих (никогда не валидируем через пропуск). Как только конвейер сдвигается — парковка доигрывается по порядку;
3. подпись + **доступный** баланс (подтверждённый − зарезервированный) на сумму и комиссию;
4. резерв средств, и только затем аттестация.

Резерв — локальная гарантия: «через меня этот отправитель эти деньги дважды не потратит». Конвейер pending ограничен (8 на аккаунт).

Free-полоса (fee=0), устройство и подводные камни. Валидация fee=0 — протокольная обязанность ВСЕХ валидаторов: финальность требует кворума, поэтому «бесплатность» не может быть настройкой одной ноды. Что настраивается — так это **приём от пользователей** (`ACCEPT_FREE`, альтруистичный режим): обычная нода отвечает «идите на благотворительную ноду», альтруистичная принимает и рассылает. Анти-спам: жёсткий протокольный лимит `free_tx_per_day` на аккаунт (фарм аккаунтов упирается в $1-регистрацию: 20 бесплатных tx/день не окупают доллар), при перегрузке free-полоса деградирует первой, и она не участвует в fee_pool — альтруизм не размывает доходы валидаторов. Живой пример: api.xync.net — старый бэкенд XyncPay — навсегда остаётся бесплатным входом для клиентов старого фронтенда (§12).

6. Шаг 3: аттестация и финализация (fastpath)

Жизненный цикл платежа: финальность за один round-trip (fastpath), порядок — позже (ленивый DAG)



Деньги финальны на шаге сертификата. DAG никогда не задерживает платёж — он лишь ведёт учёт.

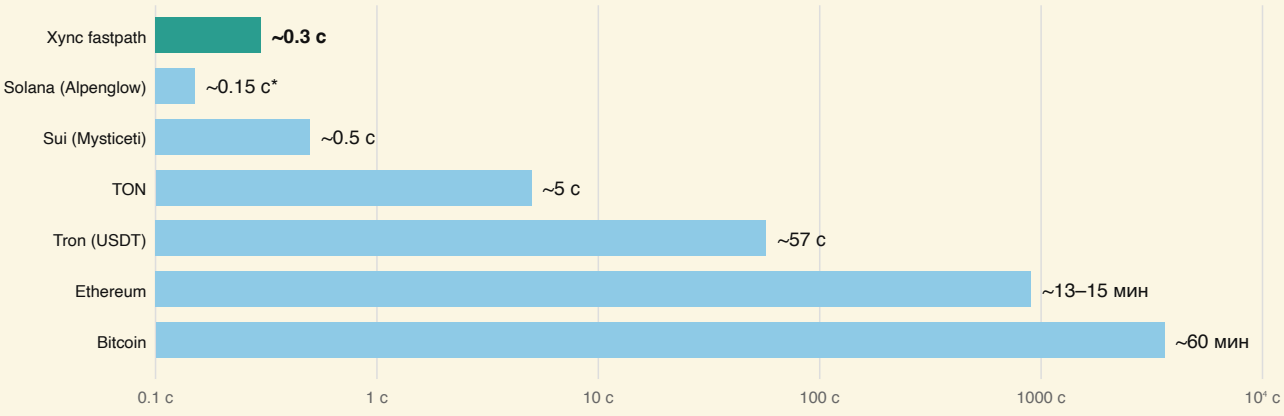
Успешный резерв → аттестация `ed25519(DOMAIN_ATT ‖ tx_id)` → gossip. q = 2f+1 аттестаций = **сертификат** = финальность. Сертификат самодостаточен (несёт 80-байтовую tx и подписи; ~280 Б при n=4), применяется любой нодой немедленно и – важно для §13 – **проверяется оффлайн**.

Теорема (безопасность fastpath). Пусть существуют сертификаты C_1, C_2 для разных тел с одним (from, seq). Их кворумы пересекаются по ≥ f+1 валидаторам, значит, минимум один честный подписал оба тела. Но честный валидатор после резервирования seq аттестует не более одного тела на (from, seq) – противоречие. ∎

Живучесть: любые 2f+1 живых валидаторов достаточны и для аттестаций, и для DAG – f отказов/злодеев не останавливают ни деньги, ни бухгалтерию.

Латентность: `T_final = 2δ + ε`, где ε – крипто (verify 30–60 мкс на ядро, батч-верификация ×2 быстрее). Целевые 150–400 мс глобально – доказанный оптимум класса. Пропускная способность фастпаса масштабируется тривиально: переводы разных отправителей независимы (embarrassingly parallel), узкое место – подписи: одно ядро ≈ 15–30k verify/с, узел с 8 ядер – >100k TPS теоретического потолка до сети.

Время до необратимой финальности (лог-шкала, типичные значения, 2026)



* Mainnet Alpenglow ожидается в Q4 2026. Карточные сети авторизуют за ~2 с, но расчёт идёт днями и допускает чарджбэки.

Сходимость состояния. Сертификаты одного отправителя применяются строго по seq (опоздавшие буферизуются); переводы разных отправителей коммутируют → все честные ноды сходятся байт-в-байт БЕЗ тотального порядка. Проверяется симуляцией на 46 инвариантов: корни состояний четырёх независимых Ledger'ов совпадают до бита.

7. Шаг 4: сетевой слой

Решения приняты по итогам исследования 07.2026 и защищены против альтернатив. Прототип использует WebSocket-заглушку (отлаживаемость); интерфейс p2p-сервиса (`/broadcast` + маршрутизация типов) не изменится при замене транспорта.

7.1. Транспорт: QUIC (не «сырой» UDP и не TCP)

Решение: **QUIC для всего трафика данных**; лёгкие датаграммы discovery – по тому же UDP-сокету (демультиплексирование по первому байту).

TLS 1.3 встроен (не нужен отдельный Noise-слой), 0-RTT переподключение, мультиплексирование стримов без head-of-line blocking, connection migration (мобильный переход Wi-Fi→LTE не рвёт соединение – критично для Telegram MiniApp). Решающий довод: **WebTransport = QUIC в браузере**; certhash позволяет браузеру подключаться к валидатору без CA-сертификата. Браузерная нода получает тот же транспорт бесплатно – а это прямая дорога к цели «полноценная нода в кошельке» (WASM).

Альтернатива	Почему нет
Сырой UDP + свой reliability-слой (путь Solana до QUIC)	Пришлось бы реализовать ретрансмиссии, контроль перегрузки, шифрование. Solana сама мигрировала на QUIC после DoS-инцидентов 2022 г.
TCP (+TLS)	Head-of-line blocking между потоками; нет миграции соединения при смене сети (мобильные!); 2–3 RTT на установку против 0–1 RTT у QUIC.
libp2p-стек целиком	Тянет мультиформаты, мультиплексоры, протокол-неготиацию – избыточная вычислительная и байтовая нагрузка. Берём только идеи (DCUtR-подобный hole punching).
WebRTC	Сложный стек (ICE/DTLS/SCTP), придуман для медиа; для точка-точка данных QUIC проще и быстрее.

Миф «сырой UDP лучше пробивает NAT» опровергнут замером 2025 г. (4.4 млн попыток, 167 стран): успех hole punching **QUIC ≈ TCP** – выбор QUIC не жертвует пробиваемостью.

7.2. Обнаружение нод: гибрид Kademlia + PEX

Решение: **подписанные node-записи** (ENR-стиль, ed25519) в Kademlia-DHT поверх UDP – глобальный поиск; **Peer Exchange** поверх уже открытых QUIC-линков – дешёвое уплотнение меша; **DNS-сиды** – бутстрап.

Альтернатива	Оценка
Только Kademlia-DHT (Ethereum discv5)	Надёжный глобальный поиск O(log N), но каждый lookup – серия RTT; для «найди ещё соседей» дорого. Берём как базу.
Только gossip/PEX (обмен addr в Bitcoin)	Почти бесплатно по трафику, но медленная сходимость и уязвимость к eclipse (соседи навязывают свой список). Берём как дополнение, не основу.
Только UDP-broadcast	Работает в LAN, в интернете не существует. Нет.
Централизованный реестр/трекер	Точка отказа и цензуры. Нет – но DNS-сиды как *бутстрап* стандартны (Bitcoin, Ethereum EIP-1459).

Node-записи подписаны ключом ноды — подделать чужой адрес нельзя. Анти-eclipse: пиры выбираются из разных /16-подсетей плюс случайные Kademlia-бакеты, а не только через PEX.

7.3. NAT: три яруса нод вместо «все равны»

Измеренная реальность: hole punching успешен ~**70%** (симметричные NAT не пробиваются). Поэтому протокол спроектирован так, чтобы инбаунд был нужен как можно меньшему числу нод:

1. **Публичные валидаторы/релеи** — белые IP, принимают всех. Relay-функция дёшева: через неё идёт только координация пробива и трафик непробившихся.
2. **Домашние full-ноды за NAT** — исходящие QUIC-линки к валидаторам + hole punching между собой (DCUtR-подобная координация через релей; observed-address в discovery-ping заменяет STUN). Не пробились (~30%) — circuit relay fallback; объём мал, ведь нода за NAT не обязана принимать инбаунд вообще.
3. **Браузерные/мобильные ноды** — только исходящий WebTransport. Инбаунд не нужен by design: подписка = исходящий стрим, gossip-push приходит по уже открытым соединениям.

Альтернатива	Почему нет
UPnP/NAT-PMP	Работает на части домашних роутеров, выключен у провайдеров CGNAT. Как оппортунистическая оптимизация — да, как основа — нет.
TURN/relay для всех	Центральные затраты растут линейно с сетью; убивает децентрализацию. Только fallback для ~30%.
«Все ноды с белым IP»	Отсекает 95% энтузиастов — прямой вред децентрализации.

Экономия трафика: не-валидаторы не ретранслируют консенсусный трафик и подписываются только на нужные стримы (свои аккаунты + заголовки чекпоинтов) — режимы нод в §9.

7.4. Gossip-дисциплина

Тело tx пересекает сеть один раз; дальше — 16-байтовые ID. Приоритетные полосы: fraud-тревога > аттестации/сертификаты > DAG-блоки.

Слой	Прототип (этот репозиторий)	Production (Rust)
Транспорт	WebSocket-меш все-со-всеми	QUIC (quinn), WebTransport для браузеров
Обнаружение	Статический список из genesis	discv5-подобная DHT + PEX + DNS-сиды
NAT	Не требуется (docker-сеть)	DCUtR-подобный пробив + relay fallback
Gossip	Flood с дедупом по `msg_id` (оптимум для n=4)	Push/pull-гибрид, приоритетные полосы, erasure-рассылка блоков (Rotor-идея) при росте сети

8. Шаг 5: ленивый DAG-порядок

Платежи финальны и без общего порядка (§6), но несколько видов событий его всё-таки требуют: регистрация выдаёт следующий свободный индекс аккаунта, штраф списывает конкретную сумму с конкретного баланса, `val_add` меняет размер кворума для всех сразу, а чекпоинт обязан получиться байт-в-байт одинаковым на каждой ноде. Такие события редки и не срочны – значит, слой, который их упорядочивает, имеет право быть медленным, лишь бы он был дешёвым и не стоял на пути денег. Схема – панель C выше.

Блок вместо голоса. Валидатор публикует блок `{author, round, refs, cert_ids, sys_txs, sig}`. Ссылка одного блока на другой означает сразу две вещи: «я его получил» и «я за него голосую» – отдельных сообщений-голосов в протоколе нет вообще, и это главная экономия семейства C. Готовые сертификаты fastpath внутрь блока не кладутся: они давно разошлись по сети своим gossip'ом, и в блоке лежат только их 16-байтовые идентификаторы. Так работает сквозной принцип «каждый байт пересекает сеть ровно один раз».

Откуда берутся раунды. Общих часов у сети нет, поэтому время отсчитывается кворумом: раунд r+1 открывается в тот момент, когда собрано q блоков раунда r (threshold clock). Медленную ноду никто не ждёт. Лидер раунда назначается по кругу – round-robin (в v2 несколько якорей одновременно плюс репутация по аптайму, как в Shoal). Блок лидера раунда r считается закоммиченным, когда q блоков раунда r+1 на него сослались; если лидер молчит, его раунд просто пропускают, и сеть не останавливается ни на секунду.

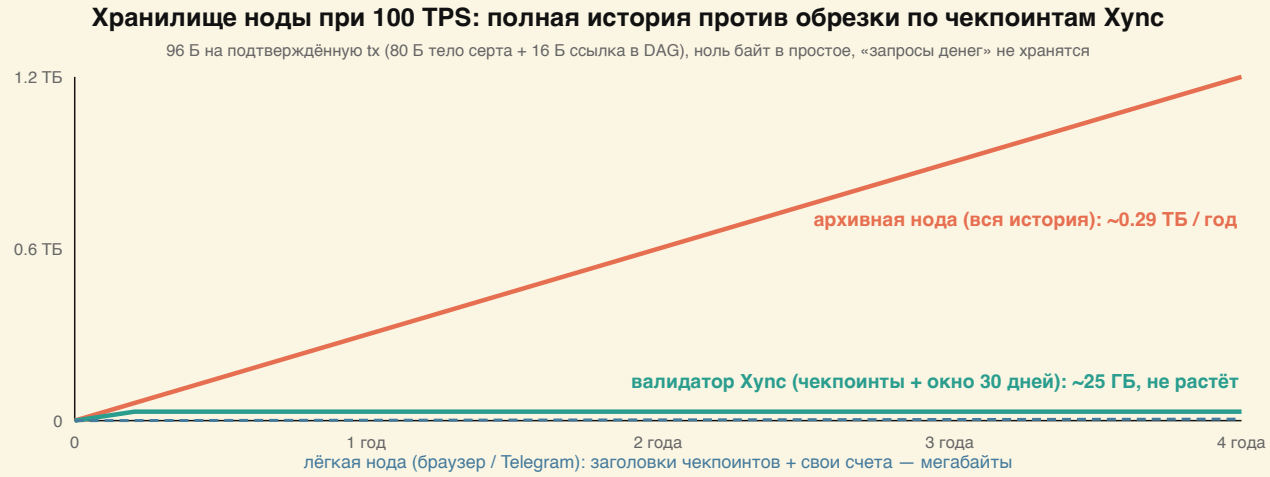
Как из графа получается список. Закоммиченный блок лидера («якорь») тянет за собой всю свою каузальную историю – все блоки, на которые он ссылается прямо или через цепочку ссылок. Каждая нода обходит эту историю одним и тем же детерминированным способом и сортирует результат по паре (round, author). Порядок выходит одинаковый у всех, хотя ноды ни о чём между собой не договаривались – согласие вытекает из формы графа, а не из голосования. Содержимое не-лидерских блоков раунда попадает в коммит следующего якоря; для денег эта отсрочка безразлична, они финальны ещё с fastpath.

Темп – адаптивная лесенка. Каждый блок стоит трафика, а торопиться DAG-слою некуда. Поэтому нода тем дольше выжидает перед публикацией, чем меньше сертификатов накопила. Production-параметры `[[4096, 1 c], [1024, 5 c], [64, 15 c], [1, 60 c]]` читаются так: набралось 4096 сертификатов – режем блок через секунду; набрался один – ждём минуту. Нижняя ступень принципиальна: **при нулевой активности блоков нет вообще, простаивающая сеть стоит 0 байт.** Для сравнения: Ethereum штампует пустой блок каждые 12 секунд, Solana – каждые 400 мс, и хранят эту пустоту все ноды и навсегда.

9. Шаг 6: чекпоинты, хранение, лёгкие ноды

Каждые K коммитов – чекпоинт: SHA-256-корень канонического состояния (+ подписи кворума в v2), граница обрезки истории и точка входа лёгких клиентов.

```
S(нода) = 100 Б · A  +  96 Б · TPS · W  +  заголовки чекпоинтов
          состояние     окно истории
A – аккаунтов, W – окно в секундах; 96 = 80 (серт) + 16 (ID в DAG)
```



Числа: 1 млн аккаунтов ≈ 100 МБ состояния; 100 TPS с 30-дневным окном ≈ 25 ГБ – валидатор живёт на Raspberry Pi. Лёгкая нода: заголовки чекпоинтов + свои аккаунты с пруфами – мегабайты, старт секунды, никакой «синхронизации с генезиса». Архив всей истории – добровольные ноды за гранты казны. «Запросы денег» – эфемерные объекты с TTL, в чейн не попадают вовсе.

Режимов нод два: **validator** (стейк, участие в кворумах, награды) и **light**. Привычного третьего типа – «full-нода без стейка» – здесь нет намеренно. Держать полное состояние и при этом добровольно отказываться от валидаторских наград не имеет смысла: железо-то одно и то же. Поэтому «full» у нас – не отдельный класс, а тот же бинарь валидатора, запущенный без ключа в наборе; такой режим нужен биржам и юридически осторожным операторам, которым важно всё видеть и ни за что не голосовать.

Защита выбора. Mina-стиль рекурсивных SNARK даёт константную историю, но прувер неподъёмен для браузера/мобильных; чекпоинты кворума дают ту же практику (мгновенный старт, обрезка) при тривиальной верификации. Классические блоки-контейнеры дублировали бы тела tx.

10. Наказания: fraud proofs

Две разные подписанные tx с одним (from, seq) – самодостаточная улика (80+80 Б + подпись репортёра). Поймавшая нода: (1) мгновенно замораживает отправителя локально и рассылает тревогу высшего приоритета – все замораживают и снимают его резервы; (2) кладёт улику системной транзакцией в DAG. Исполнение в порядке коммита (детерминизм!): **бан навсегда, штраф `fraud_fine_xync` (или весь XYNC-баланс), 50% репортёру, 50% сжигается.** Гонки репортёров исключены порядком DAG: на всех нодах выигрывает один и тот же. Мотивация дизайна – превратить каждый узел сети в охотника: поимка мошенника прибыльнее самой попытки мошенничества.

11. Sybil-устойчивость, стейк и динамический набор валидаторов

11.1. Два разных Sybil-вопроса – два разных ответа

Словом «Sybil» называют две совершенно несвязанные угрозы, и путать их дорого. Первая – дешёвые *аккаунты*, вторая – дешёвая *власть*.

Дешёвые аккаунты. Если завести аккаунт ничего не стоит, их заводят миллионами: ради спама, ради фарма бесплатной полосы, ради захвата коротких красивых индексов. Наш ответ – платная регистрация ~$1, уходящая в казну сети. Доллар незаметен человеку, который заводит один аккаунт, и разорителен для фермы, которой их нужен миллион. Побочные эффекты приятные: казна пополняется, а короткие индексы обретают цену и потому не расхватываются впустую.

Альтернативный анти-Sybil для аккаунтов	Почему нет
PoW при регистрации	Считать хэши на телефоне – значит жечь батарею пользователя ради нашей проблемы.
KYC	Закрывает вход без разрешения – то самое свойство, ради которого сеть и строится.
Соц-графы, proof-of-personhood	Фермы обходят их дешевле, чем честный человек проходит; плюс требуют доверия к графу.

Дешёвая власть. Если бы голос в кворуме давался за аккаунт, тот же миллион аккаунтов купил бы сеть целиком. Поэтому аккаунты не голосуют вообще: право подписывать аттестации и DAG-блоки даёт **исключительно стейк**, вес в кворуме пропорционален его размеру, а подпись двух противоречащих блоков карается слэшингом. Миллион купленных аккаунтов не даёт ни грамма власти над финализацией – он даёт ровно миллион пустых кошельков.

11.2. Динамический набор валидаторов

Набор валидаторов не зашит в genesis. Системные транзакции `val_add` / `val_remove` за подписью governance-ключа едут через DAG-слой, а значит, применяются всеми нодами в одной и той же точке порядка – размер набора n меняется у всех одновременно, и кворум пересчитывается автоматически:

n	q = 2f+1	f	что это значит
1	1	0	одна нода = один оператор
2	2	0	нужны обе подписи
3	3	0	нужны все три
4	**3**	**1**	**первый настоящий BFT: сеть переживает отказ или злонамеренность любой одной ноды**
7	5	2	переживает двух

Ни добавление, ни удаление ноды не требует перезапуска сети — это инвариант, который проверяет `selfcheck`. Новая нода поднимает состояние снапшотом с последнего чекпоинта, а не реплеем с генезиса, и находит соседей по p2p сама.

12. Миграция живого продукта: пять фаз без остановки

Сеть строится не с чистого листа: у неё уже есть работающий предшественник – платёжный сервис XyncPay с живыми клиентами. Поэтому децентрализация для нас не запуск нового чейна, а *операция на бьющемся сердце*: продукт не имеет права остановиться ни на минуту, а пользователь не должен заметить вообще ничего.

Исходная точка (07.2026): бэкенд `api.xync.net` валидирует всё единолично, история – в Postgres. Наблюдение по коду прода: его `Transaction.pack` уже кодирует 16 байт (typ+ts | cur | receiver | amount | sender). Новая сеть – тот же формат по духу, с исправлениями (seq вместо ts, fee-уровень, 28-битные адреса, §3). Поэтому миграция – это **снапшот балансов плюс якорь истории**, а не перепись старых транзакций в новый формат.

Три принципа: клиент ничего не замечает; каждая фаза обратима, пока не объявлена финальной; никакого big bang.

Фаза 1 – «тень» (1-2 недели). Рядом с продом поднимается нода нового протокола: n=1, кворум 1 (сеть корректно работает с одного валидатора). Бэкенд после своей обычной валидации дублирует каждую tx в ноду асинхронно – ошибки не блокируют прод. Ежесуточная сверка балансов Postgres ↔ `GET /account/{id}`. Прод остаётся единственным источником истины. *Выход:* ноль расхождений N дней подряд на живом трафике.

Фаза 2 – «переключение истины» (день X). Стоп записи на 1-2 минуты; `scripts/migrate_from_postgres.py` строит genesis: балансы по балансовой логике прода (verified-received − signed-sent), реестр валют из `cur`, ключи из `user.pub` / `prv`, а `legacy_root` – SHA-256-якорь всей центральной истории. Прошлое остаётся аудируемым, не занимая ни байта состояния: полный дамп Postgres публикуется отдельно и сверяется с якорем. Отчёт скрипта заранее показывает юзеров без ключей и отрицательные балансы. Откат: dual-write в обе стороны первые недели, Postgres – полное зеркало.

Фаза 3 – «плюс ноды» (месяцы 1-3). Механика описана в §11.2 и перезапуска сети не требует. Интересна здесь не она, а то, как с каждой добавленной нодой меняется мера доверия к оператору Xync:

n	что меняется доверенчески
1	как сейчас, но уже в новом протоколе
2	у второго оператора появляется вето: украсть без его подписи нельзя
3	сговор двух из трёх невозможен без следов в подписях
4	**первый настоящий BFT: сеть переживает отказ или злонамеренность любой одной ноды – включая сам Xync**
7	Xync можно вывести из обязательного пути платежа совсем

Риск фазы: латентность fastpath растёт с географией – кворум это самые быстрые 2f+1 из n, поэтому операторов выбирают по связности.

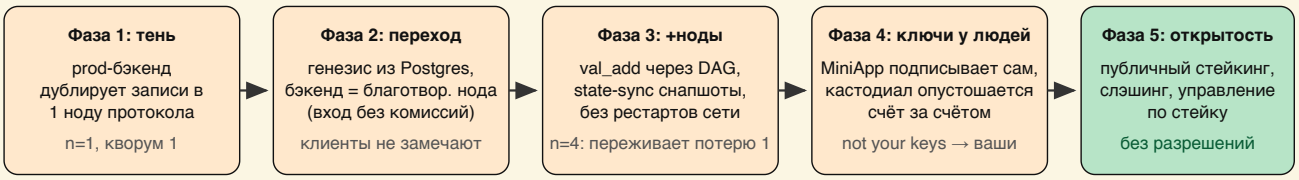
Фаза 4 – «ключи юзерам» (параллельно фазе 3). Сейчас ключи кастодиальные. MiniApp генерирует ключ локально, sys_tx `rekey` (подписанная СТАРЫМ ключом) меняет pubkey аккаунта, `User.prv` затирается. «Not your keys» решается по одному юзеру, без общего дедлайна; шлюз продолжает обслуживать некастодиальных как relay.

Фаза 5 – «открытый набор» (месяцы 6+). Вход валидатора по минимальному стейку XYNC, награды из комиссий (§15), слэшинг за эквивокацию блоков. Governance-ключ: оператор → мультисиг команды → стейк-голосование.

Старый бэкенд не выключается никогда: он становится **вечной благотворительной нодой** (валидатор в альтруистичном режиме, `ACCEPT_FREE=1`) и остаётся дефолтной точкой входа для клиентов старого фронтенда – бесплатно, в пределах протокольного дневного лимита (§5).

Вопрос миграции	Решение
Старая история	Не реплеится; `legacy_root` в genesis – криптоякорь; дамп Postgres публикуется для аудита
id клиентов	Сохраняются 1:1 (кодек допускает #0 = оператор)
Запросы денег (`status=request` в проде)	В genesis не едут – они эфемерны; живые запросы пере-создаёт шлюз
signed-но-не-verified на момент снапшота	Кредит уже учтён балансовой формулой прода; после старта шлюз пере-подаёт их как обычные tx
Валюты	id из `cur` сохраняются; код 0 зарезервирован под XYNC (в проде не использовался – проверяется скриптом)

Плавная децентрализация ЖИВОГО платёжного продукта (XyncPay)

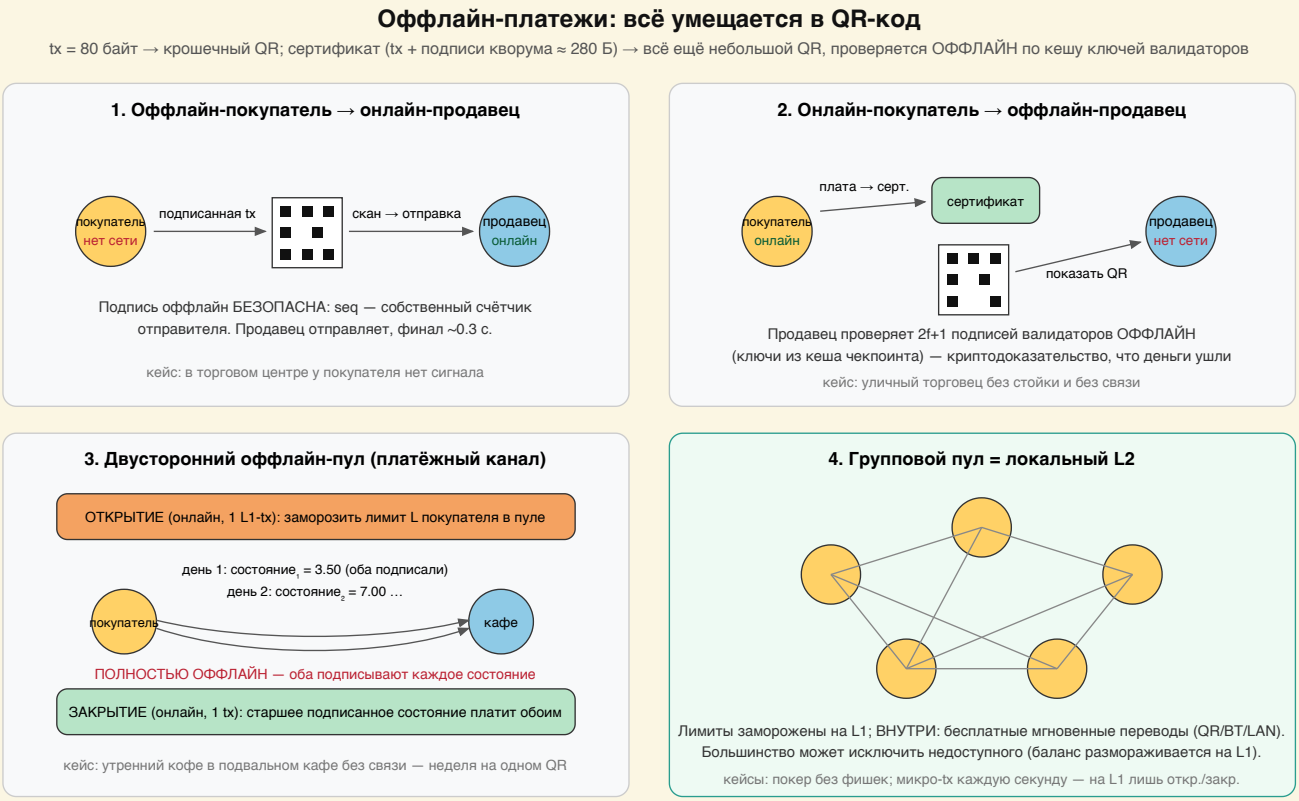


Рубеж доверия: при n = 4 валидаторах (кворум 3) оператор теряет единоличную власть над платежами — каждый перевод требует подписей независимых сторон; каждая фаза обратима, пока не объявлена финальной.

Старая история закреплена в генезисе (хеш legacy_root) — новая цепь криптографически ссылается на своё централизованное прошлое.

13. Оффлайн-платежи и пуллы (v2)

Два свойства протокола делают оффлайн-класс возможным БЕЗ изменений формата: (1) **оффлайн-подпись безопасна** – seq является собственным счётчиком отправителя, состояния сети для подписи не нужно; (2) **сертификат проверяем оффлайн** – 2f+1 подписей валидаторов сверяются с набором, закэшированным из последнего чекпоинта.



Подписанная tx – 80 байт, сертификат при n=4 – ~280 байт: и то, и другое влезает в обычный QR-код.

Режим 1: оффлайн-покупатель → онлайн-продавец. Покупатель без связи показывает QR с подписанной tx; продавец сканирует и сам отправляет её в сеть, финальность ~0.3 с у него на глазах. Два нюанса: (а) кошелёк покупателя обязан резервировать seq локально – инкремент при каждой подписи, даже неотправленной, иначе следующая оффлайн-подпись создаст конфликт, то есть самодонос; (б) покупатель не узнает об исполнении, пока не выйдет в сеть, и кошелёк должен честно показывать «подписано, ожидает доставки».

Режим 2: онлайн-покупатель → оффлайн-продавец. Покупатель платит со своего телефона, получает сертификат и показывает его QR продавцу. Оффлайн-проверка продавца: (а) кворум подписей против кэшированного набора валидаторов; (б) `to` – мой аккаунт, валюта и сумма верные; (в) анти-повтор: пара `(from, seq)` ещё не предъявлялась мне (локальный список увиденных). Кэш набора валидаторов возрастом в дни и недели достаточен: набор меняется редко и только через DAG, а при смене старые ключи остаются валидными для старых сертификатов.

Режим 3: двусторонний пулл – платёжный канал. Открытие (онлайн, одна L1-tx): `channel_open(payer, payee, cur, limit)` замораживает лимит на балансе покупателя – механика резервов в Ledger уже есть. Оффлайн-платежи: покупатель подписывает нарастающие состояния `state_k = (channel_id, k, spent_k)`, где `spent` монотонно растёт и не превышает лимит; продавец при скане СОподписывает (очная встреча – обмен двумя QR), хранят оба. Закрытие (онлайн, одностороннее): любой предъявляет максимальное СОподписанное состояние, payee получает `spent_k`, payer – остаток лимита.

Защита от «отката» – предъявления устаревшего состояния – устроена асимметрично, и это стоит разобрать. Выплата **закрывающему** мгновенна в размере его доли по предъявленному состоянию; доля **контрагента** удерживается короткое окно (до следующего чекпоинта), в котором тот может предъявить состояние с бóльшим `k` и забрать больше. Почему окно защищает именно payee: старое состояние отдаёт payee меньше, а остаток лимита возвращается payer'у – значит откат выгоден payer'у, и только ему. Payee откатывать бессмысленно: его доля в старых состояниях меньше. Для кофейни всё это невидимо – она закрывает канал вечером сама и получает деньги мгновенно.

Режим 4: групповой пулл – локальный L2. Участники вкладывают лимиты `deposit_i` одной L1-tx заморозки. Внутри живёт вектор балансов `B = (b_1..b_m)` с инвариантом `Σb_i = Σdeposit_i`; внутренняя транзакция `k` – подписи обоих контрагентов перевода над `(pool_id, k, B_k)` плюс видимость для остальных при контакте (локальная сеть, Bluetooth, QR). Полный оффлайн всей группы допустим, пока участники в контакте друг с другом. Исключение отсутствующего: большинство подписывает `evict(pool_id, member, B_last_seen)` – онлайн-tx разморозки его доли по последнему подтверждённому состоянию; вернуться можно только онлайн через `join`. Закрытие: финальный вектор за подписями большинства → одна L1-tx выплат. Между открытием и закрытием сеть не видит **ничего**: внутренние переводы бесплатны и не ограничены по частоте – «стриминг каждую секунду». Кейсы: покер без фишек, команда в походе, маркет без связи, посекундная оплата контента.

Гарант, который не держит деньги. Правило «закрытие подписывает большинство» даёт эскроу бесплатно, без единой новой строки протокола. Пусть в пулле трое: покупатель, продавец и выбранный ими гарант. Покупатель вносит цену, продавец – залог, гарант не вносит ничего. Большинство здесь – двое.

- *Сделка прошла честно.* Покупатель и продавец подписывают финальный вектор вдвоём – этого уже большинство. Гарант не участвует, не подписывает и не получает ни копейки. В обычной жизни его как бы нет.
- *Спор.* Гарант подписывает вектор вместе с той стороной, которую считает правой, – снова двое из трёх, пулл закрывается.
- *Гарант жуликоват.* Утащить деньги он не может физически: его единственной подписи не хватает, а вторую ему даст только тот, кого он не обокрал. Деньги всё время лежат в замороженных депозитах L1, а не на счету посредника.

Отсюда и «бесплатный»: сеть не берёт процента, а гарант не берёт платы за сделки, которых не касался. Интернет нужен ровно дважды – при открытии и при закрытии; сама сделка происходит оффлайн, обменом QR-кодами. Честное ограничение: гарант, сговорившись с

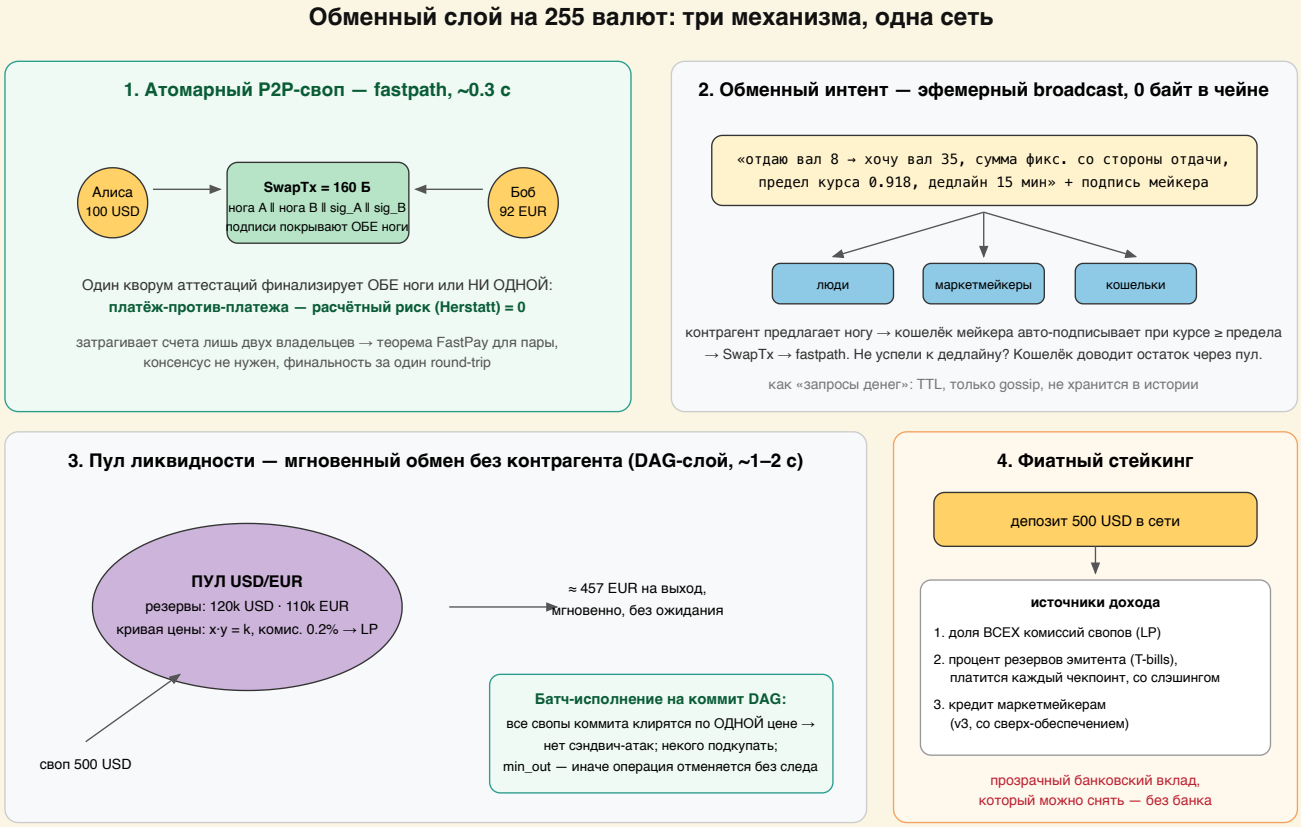
одной стороной, обкрадывает другую – это предел любой схемы «2 из 3», и цена доверия здесь ровно одна, зато явная: выбирайте гаранта вдвоём.

Событие	L1-транзакций
Открытие пулла (любого)	1
Тысячи внутренних платежей	0
Исключение участника	1
Закрытие	1

Честные ограничения. Режимы 1-2 асимметричны по информированности: подписавший оффлайн узнаёт о финальности позже, и UX обязан это показывать. Каналы вводят в Ledger новый класс состояния (открытые пуллы) и два новых вида системных tx – это v2, в прототипе их пока нет. Групповой пулл требует от кошельков локального транспорта (BT/LAN/QR) и протокола сплетен внутри группы – сложность клиентская, не сетевая. Долгий полный оффлайн обеих сторон канала – риск, что контрагент уже закрыл его онлайн; периодический выход в сеть для сверки есть норма гигиены, о которой напоминает кошелёк.

14. Мультивалютный слой: 255 валют как протокольный примитив

Фундамент уникальности XyncNet: помимо нативного XYNC, в протокол **вшиты 255 валют/монет** – не токены-контракты поверх VM (как ERC-20), не мосты-обёртки, а поле `currency` в каждой транзакции и реестр в состоянии сети. Следствие: любая операция с любой валютой наследует ВСЕ свойства протокола – 80 байт, финальность 2δ, оффлайн-сертификаты, free-полосу.



14.1. Реестр валют

`{code 1..255 -> name, scale, issuer_acct, listing_deposit}` – genesis + листинг через governance. Scale индивидуален (USD:3, BTC:8, JPY:0): поле amount всегда интерпретируется в минимальных единицах своей валюты. Эмитент фиатной валюты – регулируемая организация с листинг-депозитом в XYNC (слэшится при нарушениях) и обязательным proof-of-reserves через оракулы (§16). XYNC (код 0) – единственная валюта без эмитента.

Защита выбора. Альтернатива «токены как контракты» (ERC-20/SPL): каждый токен – своя логика, свой аудит, свои баги, а перевод стоит исполнения кода VM. У нас валюта – это ЧИСЛО в реестре: нулевой attack-surface, нулевой оверхед, все валюты ведут себя одинаково предсказуемо. Альтернатива «мосты между чейнами»: мосты – рекордсмены по взломам (миллиарды долларов: Ronin $624M, Wormhole $326M, Nomad $190M…); у нас 255 валют живут в ОДНОЙ безопасности одного набора валидаторов – класс «bridge risk» отсутствует по построению.

14.2. SwapTx: атомарный P2P-обмен в fastpath (160 байт)

Новый тип кадра (версия по длине: 80 Б = перевод, 160 Б = своп):

```
body_A(16) ‖ body_B(16) ‖ sig_A(64) ‖ sig_B(64)
body_A: {cur X, amount x, seq_A, fee, to=B, from=A}
body_B: {cur Y, amount y, seq_B, fee, to=A, from=B}
sig_i = ed25519(DOMAIN_SWP ‖ id_A ‖ id_B)   — подпись над ПАРОЙ
pair_id = sha256_16(id_A ‖ id_B)            — id объекта для аттестаций
```

Валидатор резервирует обе ноги атомарно (оба seq, оба баланса – или ничего); одна аттестация над `pair_id`; один кворум финализирует обе ноги. **Теорема безопасности пары:** конфликт по любому из (from_A, seq_A), (from_B, seq_B) не соберёт второй кворум – пересечение кворумов содержит честного валидатора, который зарезервировал пару целиком. Это PvP-расчёт (payment-versus-payment): «одна нога исполнена, вторая нет» невозможно по построению – в традиционном FX это главный расчётный риск (Herstatt risk), убивший не один банк. Своп остаётся в fastpath (0.3 с), потому что трогает счета только двух владельцев – теорема FastPay применима к паре так же, как к одиночному переводу.

Атомарный резерв (реализация `reserve_swap`). Потребности считаются словарём на КАЖДУЮ сторону и только по ИСХОДЯЩИМ ногам – получаемое от контрагента в доступность не идёт (иначе своп мог бы «оплатить сам себя»). Краевой случай: если сторона отдаёт XYNC, её основная сумма и XYNC-комиссия складываются в одну строку словаря – наивная реализация, считающая их раздельно, недобрала бы резерв. Резерв – одна запись по `pair_id`; `pending_seqs` обеих сторон пополняются вместе; снятие резерва (таймаут кворума, fraud-заморозка) откатывает обе ноги.

Финализация – точка синхронизации двух seq-цепочек (`settle_swap`). Своп применяется, только когда ОБЕ ноги дошли до своей очереди (`seq == side.seq + 1` у обеих). Если хоть одна «в будущем», сертификат буферизуется у ОБЕИХ сторон и доигрывается, как только вторая нога доедет. **Дедлока нет:** каждая seq-цепочка монотонна и независима, а своп – единственная их общая точка. Идемпотентность – guard по `pair_id` ДО любых мутаций (gossip приносит сертификат 2-4 раза). В `settled` кладётся и `pair_id`, и оба id ног – так ногу нельзя переиграть повторно уже одиночным 80-байтовым кадром.

Разделение доменов = атомарность на уровне подписи. Подпись ноги сделана над `DOMAIN_SWP ‖ id_A ‖ id_B`, а не над телом ноги. Поэтому байты одной ноги нельзя предъявить как валидный одиночный перевод (тот проверяется под `DOMAIN_TX`): ни один из участников не может «вытащить» выгодную ему ногу из свопа и исполнить её отдельно.

Обобщённые fraud proofs. Улика даблспенда теперь принимает кадры 80 ИЛИ 160 Б. Из каждого извлекается множество трат `(from, seq, id)` (у перевода одна, у свопа две); улика валидна, если есть пересечение по `(from, seq)` с РАЗНЫМИ телами И все подписи обоих кадров верны по своим доменам. Следствие: попытка потратить один seq и переводом, и ногой свопа – такая же доказуемая улика, ведущая к тому же бану + штрафу (§10), без отдельной машинерии.

14.3. Обменные интенты: broadcast-запрос «хочу обменять»

Эфемерный слой (как «запросы денег» – ноль байт в истории):

```
ExchangeIntent { maker, give_cur, want_cur, fixed: give|want, amount,
                 limit_rate, deadline_ms, sig(DOMAIN_XIN ‖ intent_id) }
```

Сумма указывается либо в отдаваемой, либо в желаемой валюте (флаг fixed); deadline задаёт срочность и TTL. Интент разлетается по gossip; контрагент (человек или маркет-мейкер-бот) присылает встречный полу-своп (proposal); кошелёк maker'а автоподписывает SwapTx, если курс не хуже limit_rate → fastpath. Стратегия дедлайна: ждать P2P-встречку до `deadline − margin`, остаток исполнить через пул (§14.4).

Курс – строго в целых, без float. Курс представлен как `rate = amount_want · 10⁹ // amount_give` (масштаб 1e9). Встречные суммы округляются ПРОТИВ taker'а: при `fixed = give` желаемое `want` округляется вверх (`ceil`), при `fixed = want` отдаваемое `give` – вниз (`floor`). Причина строгой целочисленности – не только детерминизм: курс сравнивается на РАЗНЫХ узлах и в РАЗНЫХ кошельках, и `float` дал бы расхождения в последнем бите, а с ними – споры о том, проходит ли сделка по limit_rate.

Где живёт автоподпись. SwapTx под интент доподписывает КОШЕЛЁК maker'а (`wallet watch`), а не нода: приватные ключи пользователей нода не держит принципиально (инвариант поверхности безопасности). Анти-спам: интенты подписаны, обязателен TTL, лимит активных интентов на maker'а (по умолчанию 8) – иначе бесплатное эфемерное объявление стало бы вектором флуда.

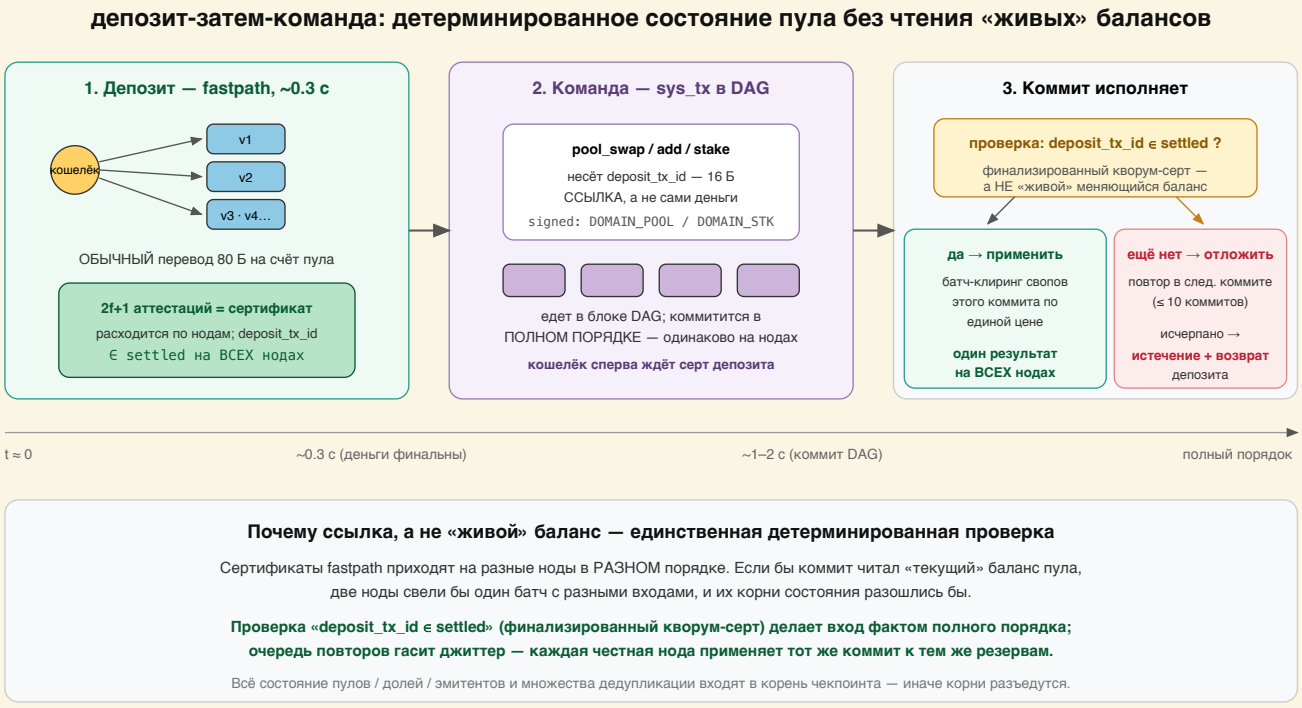
14.4. Пулы ликвидности: мгновенный обмен без контрагента

Пул – типизированный объект состояния (НЕ смарт-контракт): `{cur_a, cur_b, ra, rb, total_shares, fee_bps, acct}`. Операции `pool_create/add/remove/swap` – системные транзакции **DAG-слоя**: резервы пула – общее изменяемое состояние, ему нужен тотальный порядок (осознанная граница fastpath из §1.3; латентность обмена через пул ~1–2 с, переводов не касается).

Почему пул в DAG, а не в fastpath. Теорема FastPay опирается на то, что у объекта ОДИН владелец: пересечение кворумов ловит его двойную трату. У пула владельцев нет – любой свопает против общих резервов, и «живой» резерв в момент обработки у разных нод РАЗЛИЧАЕТСЯ (сертификаты доезжают в разном порядке). Единственный способ получить байт-в-байт одинаковое состояние – исполнять операции пула в тотальном порядке DAG.

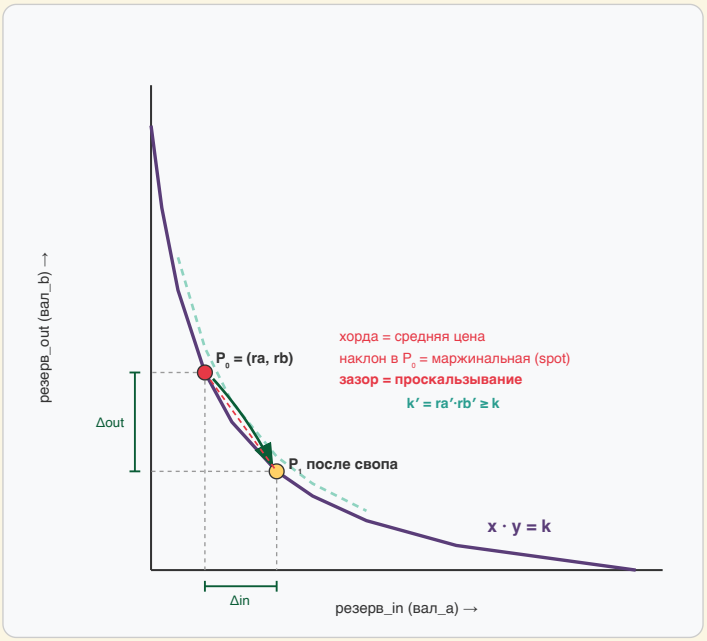
Deposit-then-command (ключ детерминизма). Резервы и выплаты пула меняются ТОЛЬКО в `ledger.commit()`; «живой» баланс в коммите не читается никогда. Деньги входят в пул лишь как УЖЕ финализированный fastpath-депозит (обычный 80-Б перевод на счёт пула), а команда несёт 16-байтовую ССЫЛКУ `deposit_tx_id`. В коммите команда исполняется, только если `deposit_tx_id ∈ settled`; иначе откладывается и ретраится (до `POOL_CMD_MAX_RETRIES = 10` коммитов), затем протухает с возвратом. Дедуп депозитов и команд – множества `used_deposits` и `pool_cmds`, оба входят в корень чекпоинта. Счёт пула создаётся с пустым

pubkey: `reserve()` отвечает на него `no_account` – тратить с пула fastpath'ом нельзя в принципе (подписать некому), а входящие переводы работают.



Цена – константное произведение, в целых (`xync/state/pool.py`). Комиссия удерживается на входе: `eff = in · (10⁴ − fee_bps) // 10⁴`; выход по кривой – `out = r_out · eff // (r_in + eff)` (floor от `r_out − k/(r_in+eff)`, k = ra·rb). В резервы кладётся ПОЛНЫЙ вход (gross), а выплата считается от `eff` – разница (комиссия) оседает в резервах и достаётся LP. Доли: первая ликвидность – `isqrt(da·db)` (среднее геометрическое, как в Uniswap-v2: первый LP задаёт стартовый курс, а корень делает доли инвариантными к масштабу депозита); добавление – `min(S·da//ra, S·db//rb)` по лимитирующей стороне, излишек возвращается; вывод – пропорция обоих резервов вниз.

Константное произведение x·y = k: проскальзывание, клин комиссии и почему k не падает



Целочисленная кривая, округление в пользу пула

Формула (Uniswap-v2 в целых числах)

```
eff = amount·(10000 − fee_bps) // 10000
out = reserve_out·eff // (reserve_in + eff)
```

оба floor — никогда не в пользу трейдера

Клин комиссии = доход LP

БРУТТО-вход идёт в резервы; выплаты считаются от EFF (< брутто). Удержанная комиссия остаётся в пуле — это доход ликвидности.

Инвариант: k никогда не убывает

- неттинг встречных потоков по spot оставляет резервы без изменений (нейтрально);
- floor в curve_out ⇒ (ra+eff)·rb′ ≥ ra·rb;
- floor в пропорц. выплате оставляет пыль в пуле;
- брутто внутрь, eff наружу ⇒ комиссия не платится.

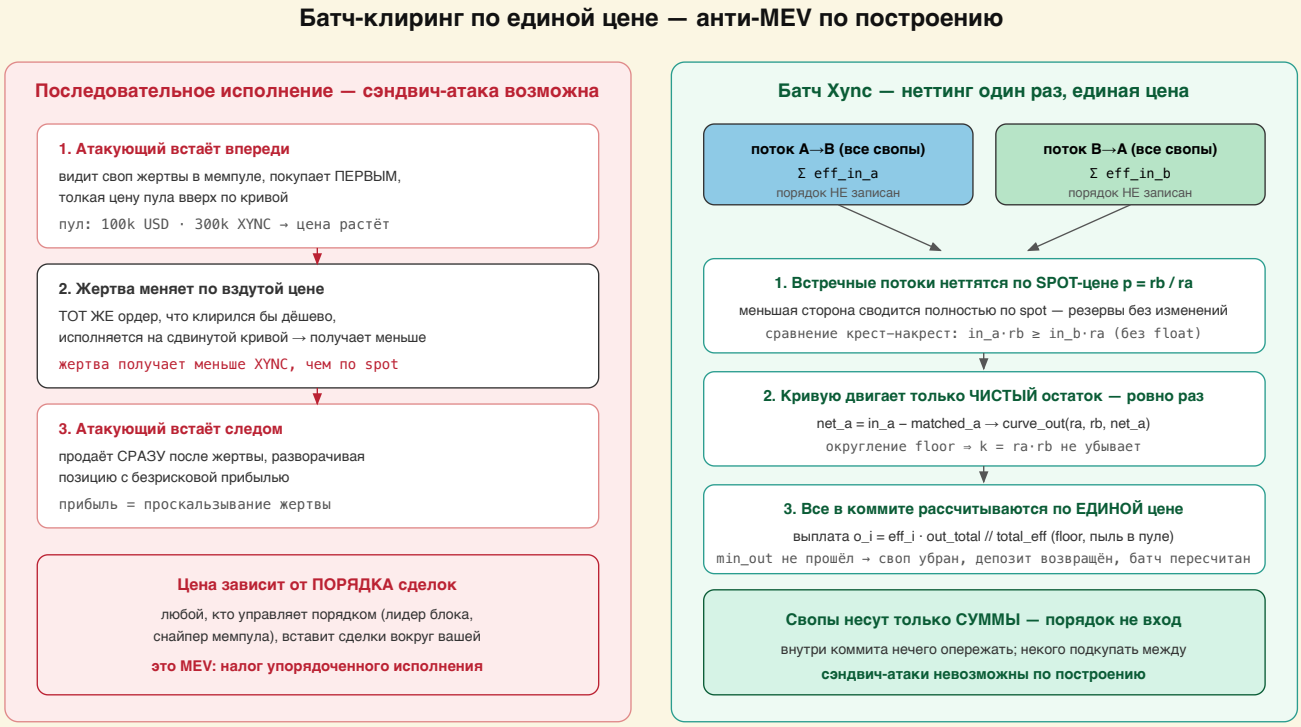
⇒ k′ = ra′·rb′ ≥ k, всегда

Почему константное произведение (не ордербук)

чистая функция двух резервов — нет VM, нет матчинг-движка, нет кода на каждую сделку; всегда даёт цену, побайтово одинаково на всех нодах.

Анти-MEV: батч-исполнение по единой клиринговой цене. Все `pool_swap` одного DAG-коммита против пула исполняются как ОДИН обмен:

1. встречные потоки взаимозачитываются по МГНОВЕННОМУ курсу пула p = rb/ra. Доминирующее направление определяется перекрёстным умножением `in_a·rb ≥ in_b·ra` (без деления и float); рецессивная сторона матчится целиком по споту – обмен по мгновенной цене резервы не двигает;
2. НЕТТО-остаток доминирующей стороны проходит через кривую `curve_out` РОВНО ОДИН раз – единственное движение цены за весь батч;
3. каждому свопу – доля общей выплаты pro-rata его входа: `o_i = eff_i · out_total // total_eff` (floor);
4. `min_out` проверяется по ИТОГОВОЙ индивидуальной выплате; не прошедший своп исключается, его депозит возвращается, а батч ПЕРЕСЧИТЫВАЕТСЯ (до фикспоинта, ≤ n итераций). `expire_round < текущего` → возврат без участия.



Почему это конструктивный анти-MEV. В клиринг входят только СУММЫ по направлениям – порядок свопов внутри батча на результат не влияет. Значит, внутри коммита сэндвич математически невозможен: нельзя «встать перед» жертвой, когда все получают одну цену. Между коммитами нет лидера, которому платят за вставку транзакции: порядок задаёт детерминированная линеаризация DAG (§8), а не аукцион за место в блоке.

Инвариант k = ra·rb не убывает (проверяется `assert` в отладке и секцией 9 selfcheck). По шагам: спот-взаимозачёт резервы не меняет → k сохраняется; `curve_out` округляет выход ВНИЗ → нетто-сделка даёт k′ ≥ k; floor в pro-rata оставляет «пыль» в пуле → k растёт ещё чуть. Все округления – в пользу пула; поэтому LP не может потерять на арифметике округлений, а комиссия монотонно наращивает резервы.

14.5. Фиатный стейкинг: вклад без банка

`stake(cur, amount, strategy)` – DAG-операция тем же паттерном deposit-then-command (подпись `DOMAIN_STAKE`). Источники доходности (честно, по нарастанию зрелости):

1. **Комиссии обменов (нативный, день 1)** – `strategy = "lp:<pid>"`. Стейк работает ликвидностью в пуле: вкладчик получает долю fee 0.2% со ВСЕХ обменов пары. В прототипе реализован «zap» одной стороной – половина депозита свопается по кривой в парный XYNC, затем добавляется ликвидность обеими сторонами (дребезг возвращается); инвариант k пула при этом не нарушается. Риск – impermanent loss, маркируется в кошельке.
2. **Процент эмитента (RWA)** – `strategy = "issuer"`. Средства уходят на счёт эмитента валюты (`issuer_acct` из реестра); позиция записывается в `issuer_stakes[cur][acct]`. На КАЖДОМ чекпоинте держателям начисляется `due = pos · rate_bps // 10⁴` (floor против держателя),

списываясь со счёта эмитента. Ставка объявлена on-chain; выплаты приходят автоматически, резервы — под оракульным proof-of-reserves (§16). Профиль «как вклад, но прозрачно».

3. **Кредит маркет-мейкерам (v3):** пул стейка кредитует ММ под сверх-обеспечение XYNC (LTV<60%), ликвидация через пулы §14.4.

Детерминизм начисления. Выплата процента исполняется в `_checkpoint` (там же, где распределяется комиссионный пул — тот же класс операций), итерация по валютам и держателям — в СОРТИРОВАННОМ порядке, суммы — целочисленные. Если у эмитента не хватило средств, выпускается сигнал `issuer_default` (в прототипе — только событие; слэшинг листинг-депозита — v3), а позиции держателей остаются нетронутыми: сеть не наказывает вкладчика за дефолт эмитента, лишь фиксирует его ончейн.

Вывод (`unstake`) — обратная операция: issuer → выплата со счёта эмитента, lp → `pool_remove`. Сеть выигрывает трижды: глубже ликвидность обменов, больше комиссий, реальный cash-flow-продукт для нетехнических пользователей.

14.6. Сводка новых типов операций (реализовано, v2)

Операция	Кадр/слой	Латентность	Домен подписи
SwapTx	160 Б, fastpath	~0.3 с	DOMAIN_SWP
ExchangeIntent	эфемерный gossip	0 байт в чейне	DOMAIN_XIN
pool_create/add/remove/swap	sys_tx, DAG	~1–2 с	DOMAIN_POOL
stake/unstake	sys_tx, DAG	~1–2 с	DOMAIN_STAKE

14.7. Инварианты обменного слоя и честные ограничения

Обменный слой реализован (4 PR) и защищён теми же принципами, что и платёжное ядро. Ключевые инварианты и чем каждый проверяется:

Инвариант	Как обеспечен	Проверка
Граница слоёв: общее состояние – только в commit по порядку DAG	deposit-then-command; «живой» баланс в коммите не читается	selfcheck §9
Детерминизм коммита (байт-в-байт у всех нод)	целочисленная математика, сортированные итерации, ноль float	два прогона §9, совпадение корней
k = ra·rb не убывает	floor в curve_out и pro-rata, gross в резервы	`assert` + §9
Идемпотентность входящего	guard по pair_id; `used_deposits`, `pool_cmds`	§7, §9
Каждый тип – свой домен подписи	SWP / XIN / POOL / STAKE	§7, §10
Всё commit-only состояние – в корне чекпоинта	пулы, доли, issuer-позиции, дедуп-множества в `_state_dump` и снапшоте	§9, state-sync

Итого 46 инвариантов `selfcheck` (секции 7–10 – обменный слой) плюс e2e на реальной сети из 4 валидаторов.

Честные ограничения прототипа. Все они перечислены здесь; журнал, в котором каждое принято и датировано, – `docs/decisions.md`:

- **Гейт депозита по локальному `settled`, а не «сертификат закоммичен в DAG».** На практике сходится (кошелёк ждёт финализацию депозита ~0.3 с прежде, чем слать команду; коммит наступает на ~1–2 с позже, и сертификат за это время расходится ко всем; ретраи поглощают джиттер). Строго-детерминированный гейт по закоммиченному cert_id – план v2.
- **Риск «сжигания seq» в two-step P2P-свопе кошелька:** протухший полу-своп может занять seq; в прототипе кошелёк не ведёт локальный seq-lock. Чистое решение (seq-lock с TTL) – v2/Rust-порт.
- **lp-zap упрощён** до пар `cur/XYNC`, и свопы zap-ов не батчатся с общим пулом; issuer-стейк – основной сценарий §14.5 – этим не затронут.
- **Процент эмитента** платится в валюте стейка (без FX-курса, чтобы не вносить float) и pro-rata ЧЕКПОИНТОВ, а не реального времени.
- **issuer_default – только сигнал**; слэшинг листинг-депозита эмитента – v3.

15. Токеномика

Эмиссия: дезинфляция с полом.

```
E(y) = max(E₀ · λʸ, E_floor),   E₀ = 4%/год, λ = 0.85, E_floor = 0.75%/год
```

В первый год сеть печатает 4% от предложения, каждый следующий год – на 15% меньше предыдущего, и так до пола в 0.75% годовых, ниже которого эмиссия не опускается никогда. Пол – не жадность, а бюджет безопасности: валидаторам нужно платить и через двадцать лет, а комиссии молодой сети их не прокормят. Отвергнутые альтернативы: нулевая эмиссия с первого дня (нечем платить, пока нет оборота); фиксированный процент навсегда (хроническое давление на цену вниз); адаптивная эмиссия в стиле Ethereum (лишний контур обратной связи между ценой токена и безопасностью – платёжной сети такая рефлексивность только вредит).

Комиссия и её судьба. Комиссия платится только в XYNC: базовая ≈ 0.001 XYNC, умноженная на множитель fee-уровня (§3). Каждая собранная комиссия делится натрое.

Доля	Кому	Смысл
50%	валидаторам	плата за работу: подписи, хранение, трафик
25%	сжигается	делает сеть дефляционной при достаточном обороте
25%	казне	гранты архивным нодам, реферальные ваучеры, разработка

Регистрационные взносы целиком уходят в казну. Штраф с пойманного мошенника делится пополам: половина тому, кто предъявил улику, половина сжигается.

Как делится валидаторская половина. Не поровну и не только по стейку. Вес ноды в раздаче = `стейк × аптайм × скорость аттестаций`, плюс бонус тому, чей блок стал якорем коммита. Аптайм в формуле стоит намеренно: валидатор, который спит, не зарабатывает ничего, каким бы крупным ни был его стейк, – это и есть экономическое выражение требования «время бесперебойной работы».

Когда сеть становится дефляционной. Сожжённое превышает напечатанное, как только годовой объём комиссий переваливает через порог V*:

```
0.25 · fee · V* > E(y) · Supply
```

Никакого обещания «дефляция с первого дня» мы не даём: до порога сеть инфляционна, и это честная плата за то, чтобы валидаторы дожили до оборота.

Чего награда не измеряет. Она пропорциональна **числу** транзакций, а не их суммам: перевод миллиона стоит ровно столько же, сколько перевод доллара. Платёжная сеть не имеет права штрафовать за крупные переводы – если пользователю нужен приоритет, он покупает его fee-уровнем, а не размером платежа. Бесплатная полоса (fee=0) наград не порождает и предложение не размывает: альтруист платит трафиком, а не чужой долей.

Честная оговорка. Комиссия только в XYNC — это трение при онбординге: чтобы отправить свои доллары, нужен посторонний токен. Ровно на это жалуются пользователи Ethereum. Три смягчения: стартовый XYNC-грант, выдаваемый при платной регистрации; автоматический обмен «на сдачу» внутри кошелька (§14.4 делает его одной операцией); и, наконец, сама доля комиссии — параметр governance, а не константа протокола.

16. Оракулы: фиатные события как криптофакты

Сеть, в которой живут доллары и евро, обязана уметь узнавать, что банковский перевод действительно случился. Обычно это делают доверенные посредники. Нам это не нужно, потому что нужный факт уже подписан криптографически — просто не нами.

Ключевое наблюдение. Каждое письмо от банка несёт DKIM-подпись: почтовый сервер банка подписывает заголовки и тело своим ключом, а публичный ключ лежит в DNS банка. Значит, письмо «мы зачислили вам 1000 €» — это не скриншот, который рисуется за минуту, а проверяемый криптографический факт, подделать который может только сам банк.

Лестница зрелости. Мы поднимаемся по ней по мере роста сети, и каждая ступень строго сильнее предыдущей:

1. **Комитет M-из-N.** Валидаторы получают письмо, каждый сам проверяет DKIM-подпись и голосует аттестацией через DAG-слой. Просто, но письмо целиком видят M валидаторов — приватности нет.
2. **zkEmail.** Пользователь строит ZK-пруф утверждения «существует письмо от домена `bank.example` с полями Y, подписанное валидным DKIM-ключом», не раскрывая ни остального содержимого, ни самого письма. Приватность возникает by construction, а не по обещанию.
3. **zkTLS.** То же самое, но напрямую из банковского API, без почтового посредника вообще.

Класс не гипотетический: zkP2P v2 (2025) работает в проде с Venmo, Revolut и Wise.

Что это даёт. Некастодиальный вход и выход фиата: эскроу разблокируется пруфом банковского перевода, а не словом обменника. Автогашение «запроса денег» по банковской квитанции. Проверяемые резервы эмитентов валют реестра (§14.1) — не аудиторский отчёт раз в квартал, а пруф по требованию.

Риски, которые мы видим, и что с ними делаем. Банк ротирует DKIM-ключ — реестр хранит историю ключей, старые пруфы остаются валидными. Фишинговый домен, похожий на банковский, — принимаются только домены из белого списка в реестре валют. Банк отзывает уже проведённый платёж — на крупные суммы эскроу-окно дольше банковского срока отзыва.

Все оракульные события — системные транзакции DAG-слоя. Латентности платежей они не касаются вообще.

17. Сводное сравнение и честные ограничения

	Xync	Solana	Sui	TON	Tron	Ethereum
Финальность	**~0.3 с**	~0.15 с*	~0.5 с	~5 с	~57 с	~13 мин
Размер tx	**80 Б**	~230 Б	~300 Б	~300 Б	~200 Б	~110 Б
Простой сети	**0 Б**	пустые блоки	пустые чекпоинты	пустые	пустые	пустые
Нода в браузере	**by design**	нет	нет	нет	нет	light-протоколы
Free-tier	**есть**	нет	нет	нет	нет	нет
Оффлайн-приём платежа	**сертификат в QR**	нет	нет	нет	нет	нет
Смарт-контракты	**нет (осознанно)**	да	да	да	да	да

* Alpenglow, мейннет ~Q4 2026.

Ограничения, о которых мы говорим сами:

1. **Кривой кошелёк может сам себя забанить.** Протокол не отличает баг ретрая от попытки двойной траты и не должен: железное правило подписи (§4) обязательно для любой реализации. В mainnet смягчаем градацией штрафов и governance-апелляцией, но не отменой правила.
2. **DAG-слой чувствителен к плохой сети.** Потери пакетов раздувают его латентность в разы. Денег это не касается — они финализируются fastpath'ом, — а план Б на случай регулярной деградации зафиксирован: оптимистичный путь в духе Raptr.
3. **Домен узок.** Всё, что не «отправить, попросить, обменять», — не к нам: ни NFT, ни деривативов, ни произвольной логики. Это осознанный выбор, из которого растут все преимущества выше, и одновременно потолок применимости сети.
4. **Регуляторика платежей.** Самый недетерминированный риск. Стратегия: некастодиальность по умолчанию, лицензированные шлюзы-партнёры на стыке с фиатом, географическое распределение валидаторов.

18. Дорожная карта

Этап	Содержание	Критерий
0 (готово)	прототип: 4 валидатора × 5 микросервисов, fastpath+DAG, обменный слой, fraud proofs, динамический набор, миграционные скрипты	selfcheck 46/46, 113 тестов (включая e2e на реальной сети), `docker compose up`
1	тень прода: dual-write из xync-hybrid, сверка	0 расхождений N дней
2	cutover: genesis из Postgres, бэкенд = благотворительная нода	прод на протоколе, n=1
3	+ноды партнёров	n=4: f=1, оператор теряет единоличную власть
4	Rust-ядро (codec/ledger/dag — уже изолированная спецификация), QUIC	кросс-тест Rust⇄Python на одном testnet
5	WASM light-нода: браузерное расширение, TG MiniApp	старт < 3 с на телефоне
6	ключи юзерам (`rekey`), открытый стейкинг, каналы §13	вход валидатора без разрешения

Python-прототип — исполняемая спецификация: `xync/common/codec.py`, `xync/state/ledger.py`, `xync/consensus/dag.py` переносятся на Rust механически, тест-векторы прилагаются. Присоединяйтесь: команда XyncPay, t.me/XyncPayBot.