|  |  |  |  |
| --- | --- | --- | --- |
|  | СОГЛАСОВАННО |  | УТВЕРЖДАЮ |
|  | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |  | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
|  | Технический директор |  | Генеральный Директор  ООО «ШИВА НЕТВОРК» |
|  | \_ / (Ф.И.О.) |  | \_\_\_\_\_\_\_\_\_\_\_\_/ (Миронов В.О.) |
|  | «\_\_» 2025 г. |  | «\_\_» 2025 г. |



**Дорожная карта**

**проекта**

**"Блокчейн с аппаратным консенсусом времени"**

**Фаза 0: Предварительная подготовка и исследование (1-2 месяца)**

1. **Глубокое исследование существующих решений:**
   * Изучить Proof-of-History (PoH) Solana, Proof-of-Elapsed-Time (PoET) Intel и другие алгоритмы, использующие время или доверенное оборудование.
   * Изучить концепции "блокчейн-часов" и их роль в распределенных системах.
   * Исследовать существующие решения для глобальной синхронизации времени, такие как Everdeen, и их устойчивость к атакам.
2. **Детальное описание возможностей PCI-карты:**
   * Какую точность синхронизации она нам обеспечит (наносекунды, микросекунды)?
   * Какие протоколы синхронизации нужны (PTP, NTP, GPS)?
   * Насколько она устойчива к внешним воздействиям и манипуляциям?
   * Есть ли у нее криптографические возможности (например, для подписи временных меток)?
3. **Определение модели угроз:**
   * Какие атаки возможны на систему времени (подделка времени, задержки, DDoS на синхронизацию)?
   * Как PCI-карта и предлагаемый консенсус будут противостоять этим атакам?
4. **Выбор языка и основных библиотек:**
   * Rust - как основа для блокчейна благодаря безопасности и производительности.
   * Определить библиотеки для криптографии (например, **sha2**, **ed25519-dalek**), сетевого взаимодействия (например, **tokio**, **libp2p** для P2P).

**Фаза 1: Разработка базового блокчейна (3-4 месяца)**

1. **Определение структуры блока:**
   * **index: u64**
   * **timestamp: u64** (получаемый с PCI-карты)
   * **prev\_hash: [u8; 32]**
   * **data: Vec<Transaction>** (или просто **String** для начала)
   * **hash: [u8; 32]**
   * **Новое поле:** **hardware\_signature: Signature** (криптографическая подпись временной метки или всего блока вашей PCI-картой).
2. **Реализация хэширования:**
   * Функция для вычисления хэша блока (например, SHA-256).
3. **Создание генезис-блока:**
   * Первый блок в цепочке.
4. **Реализация цепочки блоков:**
   * Структура **Blockchain** с вектором блоков.
   * Методы для добавления блоков и проверки целостности цепочки (проверка хэшей, **prev\_hash**).
5. **Интеграция с PCI-картой (драйвер/API):**
   * Разработать или использовать существующий драйвер/API для получения точной временной метки с вашей PCI-карты.
6. **Базовый механизм "Proof-of-Accurate-Time" (PoAT) v0.1:**
   * На этом этапе просто убедится, что каждый новый блок включает временную метку, полученную с PCI-карты, и что эта метка находится в пределах допустимого отклонения от системного времени или времени других доверенных источников.
   * Начнем с простой проверки: временная метка нового блока должна быть больше временной метки предыдущего блока и не слишком сильно отличаться от локального времени узла.

**Фаза 2: Разработка механизма консенсуса и сети (4-6 месяцев)**

1. **Разработка P2P-сети:**
   * **Обнаружение узлов:** реализовать механизм для поиска других узлов в сети (например, через DNS-сиды или DHT).
   * **Связь между узлами:** используем **tokio** и **libp2p** для асинхронной связи. Определите протоколы для обмена блоками, транзакциями и сообщениями консенсуса.
   * **Синхронизация цепочки:** Механизм для новых узлов, чтобы загрузить всю цепочку от существующих узлов.
2. **Усовершенствование PoAT (Proof-of-Accurate-Time) v1.0:**
   * **Валидаторы времени:** определим, какие узлы являются "валидаторами времени" (те, у кого есть PCI-карты).
   * **Предложение блоков:** Валидатор, который должен создать следующий блок, использует свою PCI-карту для получения временной метки.
   * **Подтверждение временных меток:** Другие валидаторы времени получают предложенный блок и проверяют его временную метку, сравнивая ее со своей собственной аппаратной временной меткой.
   * **Механизм голосования:** Реализовать механизм голосования (например, на основе BFT-консенсуса, такого как PBFT или HotStuff), где валидаторы голосуют за действительность временной метки и блока. Блок считается действительным, если его временная метка подтверждена большинством (например, 2/3) валидаторов.
   * **Обработка отклонений:** что происходит, если временная метка узла сильно отклоняется? Исключение из консенсуса, штрафы.
3. **Транзакции:**
   * Определить структуру **Transaction** (отправитель, получатель, сумма, подпись).
   * Реализовать пул транзакций (mempool).
   * Включить транзакции в блоки.
   * Реализовать базовую логику обработки транзакций (например, проверка баланса).
4. **Механизмы предотвращения атак:**
   * Защита от Sybil-атак (возможно, через стейкинг или ограниченное количество доверенных валидаторов).
   * Механизмы для обнаружения и изоляции узлов с некорректными временными метками.

**Фаза 3: Тестирование, оптимизация и дополнительные функции (3-5 месяцев)**

1. **Тестирование:**
   * **Модульное тестирование:** для всех компонентов (хэширование, структуры, P2P-логика).
   * **Интеграционное тестирование:** Проверка взаимодействия между компонентами.
   * **Сетевое тестирование:** Развертывание нескольких узлов в тестовой сети, имитация задержек, отключений.
   * **Тестирование производительности:** TPS (транзакций в секунду), время финализации блока.
2. **Оптимизация:**
   * Оптимизация производительности кода Rust.
   * Оптимизация сетевого взаимодействия.
3. **Дополнительные функции (по желанию):**
   * **Смарт-контракты:** если это необходимо для нашего сценария использования. Это значительно усложнит проект.
   * **Механизм вознаграждения:** как валидаторы будут вознаграждаться за свою работу?
   * **Хранение данных:** Оптимизация хранения блокчейна (например, использование базы данных для состояния).
   * **API для взаимодействия:** Предоставление API для внешних приложений для взаимодействия с блокчейном.
   * **Пользовательский интерфейс:** Простой CLI или веб-интерфейс для мониторинга сети.

**Фаза 4: Развертывание и поддержка (Постоянно)**

1. **Документация:**
   * Подробная техническая документация.
   * Руководства по развертыванию и использованию.
2. **Развертывание:**
   * Развертывание тестовой сети.
   * Постепенное развертывание основной сети.
3. **Мониторинг и поддержка:**
   * Мониторинг состояния сети, производительности.
   * Обновления и исправления ошибок.

**Ключевые аспекты и вызовы:**

* **Точность и надежность PCI-карты:** это основа нашего консенсуса. Любые уязвимости или неточности в карте будут влиять на весь блокчейн.
* **Синхронизация PTP/NTP между узлами:** даже с аппаратными часами, сетевые задержки могут влиять на согласованность временных меток. Вам нужно будет тщательно продумать, как узлы будут синхронизироваться друг с другом, используя PTP или другие протоколы, и как это будет влиять на консенсус.
* **Византийская отказоустойчивость (BFT):** Консенсус должен быть устойчив к злонамеренным или неисправным узлам. BFT-алгоритмы, такие как PBFT, могут быть хорошей отправной точкой.
* **Масштабируемость:** как ваша система будет масштабироваться с увеличением количества узлов и транзакций?
* **Децентрализация vs. Доверие:** если валидаторов с PCI-картами будет мало, это может привести к централизации. Баланс между доверием к оборудованию и децентрализацией будет ключевым.
* **Экономическая модель:** как будет поддерживаться сеть? Будет ли вознаграждение для валидаторов?