U/UNet

Technical Whitepaper

Author:

Tyaglov Ivan

v1.22.3

2022

Термины и сокращения

БД	База данных
РСУБД	Распределенная система управления базами данных
СУБД	Система управления базами данных
ACert	Авторизационный сертификат сети U
AKLong	CRC64 от Auth Key Public
AKP	Auth Keys Pair
AKPublic	Auth Key Public
AKShort	CRCn от Auth Key Public, n - зависит от сложности и настроек сети
API	Application Programming Interface
CP	Custom polygon
CRC	Cyclic Redundancy Check
CUDI	Cross UDomain Interface
DDBMS	Distributed DataBase Management System
DLL	Dynamic Link Library
DLLApp	Dynamic Link Library Application
ECC	Elliptic-Curve Cryptography
ECDH	Elliptic Curve Diffie–Hellman
ECDSA	Elliptic Curve Digital Signature Algorithm
ECEKP	Elliptic Curve Encryption Keys Pair
EVM	Ethereum Virtual Machine
HP	Hybrid polygon
ILP	Insured liability polygon
ILPP	Insured liability proprietary polygon
IP	Internet Protocol
KYC	Know Your Customer
MKP	Master Key Pair для восстановления доступа
NAT	Network Address Translation
NCA	Network Control Application
NFT	Non-Fungible Token
P2P	Point-to-Point
REST	Representational State Transfer
RP	Registry polygon
RPC	Remote Procedure Call
SP	Segmented polygon

SSK	Secret Session Key
TCP	Transmission Control Protocol
U	Название разрабатываемой p2p технологии
U2A	U2Application
U2C	U2Certificate
U2N	U2Network
U2P	U2Peer
U2S	U2Service
UApp	UApplication - приложение сети U
UAppNet	UApplication Network - оверлей-сеть построенная на технологии U
UCoin	Базовая монета сети UNet
UDP	User Datagram Protocol
UNet	Набор приложений и создаваемая ими сеть UNet, созданная на технологии U

От автора

Прежде всего, несколько моментов.

Первое, я настоятельно рекомендую не пропускать обзор, т.к. далее за ним следует более детальный разбор, подразумевающий, что читатель знаком с обзором.

Второе, так как проект в стадии активной разработки, то некоторые определения, механизмы, реализации и параметры могут измениться. Основной раздел активно переписывается и дополняется. Следите за актуальностью версии.

И третье, далее я хотел бы объяснить причины, которые побудили на разработку подобного проекта и обозначить цели, которые планируется достичь. Если вам это не интересно, или вы не согласны с моим мнением, можете смело переходить к следующей части.

Изначально проект разрабатывался как очередной блокчейн-проект. В процессе работы, стало понятно, что это немного не то, чего хочется достичь.

Если отвлечься и поговорить об идее Web3. Обозначить ее как нечто, базирующиеся на блокчейне, децентрализованное, анонимное (или хотя бы конфиденциальное), с независимыми финансами в виде тех же койнов или токенов, то может показаться, что, в целом, цель достигнута, но так ли это?

У вас не создается ощущения, что хоть технология и есть, но она очень специфическая. Она вплетается в нашу жизнь, как надстройка или дополнение, и ей что-то не дает взлететь? Под технологией, я подразумеваю то, что определяет ее: сам блокчейн, различные механизмы консенсуса, смарт контракты и те механизмы, которые реализованы на них.

Давайте будем честны, она решает только очень узкий аспект задач, хотя его и стараются активно расширить. При этом она не лишена целого ряда критических, на мой взгляд, проблем: экологическая сторона вопроса, спорность различных proof-методов, мнимая анонимность с вытекающими проблемами, пулы и их соотношение мощностей, когда сеть контролируют несколько пулов (участников сети), а остальные просто “променяли свой голос” на стабильный доход (участники пула), и т.д. Мы пользуемся этой технологией и пытаемся развивать только потому, что все это приносит деньги (ну и может некоторые еще верят в идею).

Создается впечатление, что это игра ради игры, блокчейн-проекты ради блокчейна. Многие идеи и реализации очень сильно переоценены. Я далее немного утрирую некоторые моменты. Все не так однозначно, и у каждой медали есть другая сторона, и это стоит учитывать.

Первое, все основные proof-алгоритмы сводятся к преобладанию ресурсами, у кого их больше тот и водит: больше оборудования, больше койнов. Система

стремится к централизации или олигополии. Почему? Потому что так проще работать самой системе. А чего здесь отличного от того, что происходит с банковской и мировой финансовой системой? Может нет смысла раздувать систему? Сформировать олигопольный центр сразу и потратить силы на его защиту и ограничение его влияния?

Второе, в блокчейне все сохранено и публично. Согласен, но хранить всю цепочку крайне неудобно, и если в начале проблем не будет, то что делать, когда данных будут петабайты? Опять - специализация, и только крупные игроки будут это хранить. И оправдано ли оно вообще? Возьмем NFT, революционный механизм цифровой собственности. А теперь сравним его с играми в библиотеке Steam. Как вы думаете, что вероятнее случиться из этого?

- Люди потеряют интерес к конкретному блокчейну, забьют на его поддержку (вспоминаем про петабайты данных) и он посыпется вместе со всеми NFT и контрактами.
- Компания Valve обанкротится, закроет Steam, не передав никому права на него. Все потеряют доступ к цифровым покупкам.

Сложно сказать, что вероятнее, но, первый сценарий не такой уж нереальный. Более того, если подумать, что если никто не захотел перехватить игровую аудиторию у банкротящейся крупной компании, то значит в мире случилось что-то, что игры потеряли ценность. А это значит, что даже если бы это все было в блокчейне, то и интерес и к нему бы погас. Так в чем принципиальная разница?

Мы пытаемся построить идеальную технологию, забывая, что мы находимся в неидеальном мире, и главные проблемы находятся не в алгоритмах, а в том, как этот продукт будет жить, как им будут пользоваться, как его могут ограничивать и блокировать извне. Что будет с большинством блокчейн-проектов, если сеть разрежут на несколько частей (изолируют друг от друга)? Форк? А потом, когда связь наладиться? Вы ведь понимаете, что так не должно быть.

Вы можете ответить, что еще не прошло достаточно времени, технология не успела еще развиться, но позвольте возразить. Возьмем смартфоны. Они не сильно старше технологии блокчейна (если считать от iPhone, как прорыва, определившего технологию). Вот только мы уже давно не представляем жизнь без смартфонов, они проникли во все уровни нашей жизни и стали частью ее, как будто были всегда, чего не скажешь о блокчейне.

Ответьте, с появлением блокчейна и смарт контрактов сервисы, которыми мы пользуемся, стали конфиденциальным и децентрализованным? Нет. Чтобы банально отправить сообщение или позвонить, мы также продолжаем пользоваться централизованными системами. Чтобы отправить файл своей жене, я вынужден его заливать на чей-то сервер, а потом она его оттуда скачивает. А если это личные данные, фотографии, документы? Еще страшнее, что большинство этих сервисов, по сути, принадлежат нескольким компаниям. О какой конфиденциальности или даже безопасности может идти речь? Да, есть

конфиденциальные решения, но они не просты, не повсеместны, и, как правило, они точно не имеют отношения к web3 и блокчейну.

На мой взгляд, вся современная затея с криптой - крайне неправильное использование ресурсов и мощностей. Так не должно быть. Мы не должны сжигать мегаватты электроэнергии, чтобы провести несколько транзакций. Нам не нужно хранить всю информацию обо всем или платить за каждый чих по смарт-контракту. Это все накладывает очень большие ограничения на возможности масштабирования этой технологии и становлении ее по-настоящему массовой.

При этом, технология нужна, очень нужна. Задумайтесь, так ли много в вашей жизни реальных децентрализованных и независимых решений, а сколько - централизованных? Вы слушаете музыку, заказываете такси, делитесь фотографиями, делаете покупки в онлайн-магазине, ищите информацию в сети, читаете любимые статьи и книги, просите смарт-колонку включить мелодию или же просто свет в доме. Почти все это централизованно и собрано почти в одних руках. И в этих руках очень большая власть и возможности. Мы осознанно принимаем эти условия, потому что нет выбора. Но так ли это? Нужны ли нам посредники, чтобы общаться, делать покупки, да в конце концов включить свет у себя дома?

И вот мы постепенно подходим к вопросу, а что делать? Какое решение? А нет его, мы и вправду упираемся в саму концепцию блокчейна, и прыгнуть дальше просто не можем. Единственный выход, это допустить некоторые упрощения, дать возможность ошибке, отказаться от блокчейна, как от базовой технологии или как от цели. Мы можем подойти к этой проблеме с другой стороны. Главное, это не блокчейн, главное - это участники сети, все и каждый. Да, это будет соответственно уже не Web3 (если понимать, что основа блокчейн), а что-то другое, назовем WebU.

Давайте попробуем обозначить основные тезисы новой системы:

1. Нам нужно понять, что реальная конфиденциальность, безопасность и независимость в сети достижимы только через P2P связь со сквозным шифрованием, когда нет посредников.
2. С современной сетевой инфраструктурой реализовать повсеместную возможность свободной P2P защищенной связи без посредников - нельзя. Но если подумать, то можно свести роль этих самых посредников к минимуму (или просто заменить их на другие каналы связи), и вот тогда этот пункт с натяжкой можно считать реализуемым.
3. Нужен единый реестр/механизм, который позволит найти в сети любого участника и наладить с ним связь. Эта система должна быть децентрализована, сегментирована, стойкая к изоляции подсети (а значит сегментация по географическому признаку) и способна к масштабированию, чтоб охватить миллиарды устройств.

4. Получив возможность связаться с каждым, следующим этапом нам нужна будет возможность формировать группы для решения коллективных задач. Значит нужна технология, которая позволит формировать сети, объединять множество участников в единый механизм, в малые и большие группы. Главное, нужно понять, что подсетей должно быть много, много цепочек, много разных решаемых задач, но они должны иметь единый механизм взаимодействия друг с другом.
5. Для распределенной работы нужен набор инструментов. Нужны системы управления базами данных, шаринг файлов, отложенная доставка для оффлайна. Вот здесь нам и пригодится блокчейн и все другие альтернативные технологии.
6. Любая система, где будет производиться обмен ресурсами и совместная работа, требует возможности материального обмена, требуется финансовая система.
7. Чтобы технология получила массовое развитие, она должна быть доступна и предоставлять механизмы разработки для независимых разработчиков.

Очень грубо, но сейчас смартфон в вашем кармане обладает ресурсами и вычислительными мощностями не хуже, чем те же персональные компьютеры лет 10 назад, а может даже и лучше. Он есть почти у каждого из нас, и почти всегда в сети. Представляете какие это мощности, если их объединить вместе?

Мы можем создать мир, где найти и наладить прямое соединение с любым человеком не будет проблемой. Нам не нужны будут посредники в большинстве приватных задач. Чтобы купить кофе с доставкой в соседнем здании, вам не нужно будет отправлять пакеты данных на другой конец света и ждать оттуда ответа. Позвонить, написать письмо, отправить файл или даже провести стрим с мероприятия можно будет не затрагивая дата-центры, посылая пакеты исключительно тем, кому они нужны. Можно долго описывать возможные кейсы использования. Более детально о них можно прочесть в разделах о сторонних приложениях.

И да, такая система априори не будет лишена всех тех недостатков, о которых я говорил выше. Большинство из них можно будет сгладить, а с чем-то даже не стоит пытаться бороться. Наша конечная цель заключается в другом, в создании слоя свободного, безопасного и независимого интернета.

Мое личное мнение, но свободный, безопасный и независимый интернет - это не означает тотальную свободу и беззаконие. Контроль должен быть. Любая технология, которая гипотетически опасна для безопасности других, должна иметь механизмы защиты этих самых других. При этом эти механизмы не должны использоваться во вред добросовестным участникам сети.

Это почти нерешаемая задача, но я верю, что если общество и власть объединят усилия, а технология будет противовесом, то возможно все.

Обзор U/UNet

Это краткий обзор всего документа, цель которого - демонстрация структуры технологии U, и производного продукта - оверлей сети UNet, как конечной цели проекта.

U

U - это технология построения многоуровневых P2P сетей для решения распределенных задач с использованием различных консенсусных механизмов.

U условно можно поделить на несколько интерфейсных уровней, где каждый последующий уровень строится на базе предыдущего:

- Peer
- Certificate
- Network
- Service
- Application

Peer (U2P)

U2P уровень базируется на UDP (в будущем и TCP, как расширение). Он определяет протокол передачи пакетов непосредственно PeerToPeer (от участника к участнику). Предоставляет разработчику единый более высокоуровневый интерфейс.

Решает задачи:

- Контроль ошибок передачи.
- Контроль за доставкой пакета. Позволяет определить доставлен ли пакет, а также получать ответные пакеты (ответы на запросы).
- Дробление большого пакета на отдельные. Контроль за их отправкой с возможностью восстановления передачи в случае обрыва. Позволяет на программном уровне отправлять большие по размеру пакеты (до ~468 Тб), не смотря на технические ограничения сети.
- Автоматическая проверка соединения и определение оптимальных параметров передачи.
- Потоковая передача данных.
- Сквозное шифрование при наличии валидного сертификата шифрования.
- Маршрутизация передачи пакетов (по цепочке, по заданному маршруту).
    - Минимизация ресурсов сети для передачи пакета большому кол-ву получателей.
    - Обход сетевых барьеров, когда прямое соединение не доступно.
- Контроль версий с возможностью динамического изменения протокола.

Certificate (U2C)

U2C определяет протоколы шифрования, работы с цифровыми ключами и взаимодействия в этой области с другими участниками сети.

Решает задачи:

- Генерация и перевыпуск доверенного сертификата (пары ключей шифрования, ограниченных временем работы) и валидация его с другими участниками сети (для генерации необходим либо участник сети с доверенным сертификатом, либо наличие другого канала связи между участниками для валидации).
- Цифровая подпись данных.
- Формирование ключа симметричного шифрования для защищенной сессии (сквозное шифрование) для уровня U2P.
- Формирование ключа симметричного шифрования для защищенной групповой сессии (когда пакет должны прочитать несколько получателей, а доставка его будет осуществляться по маршруту).

Network (U2N)

U2N определяет протоколы формирования U сети и подсетей. Предоставляет разработчикам интерфейс для работы с сетью в целом.

Нужно понимать, что U сеть - это множество небольших сетей, которые решают свои определенные задачи. Они могут объединяться, использовать данные друг друга, а могут быть полностью приватными и изолированными. Их объединяют правила, по которым они формируются, добиваясь таким образом консенсуса построения сети (каждый знает свое место изначально).

Решает задачи:

- Определяет принципы и правила работы участников в сети.
- Позволяет участникам сети обходить NAT для P2P соединения.
- Реализует механизмы построения сети участниками и формирования оптимальной топологии для решения поставленных задач.
- Контроль за работоспособностью сети.
- Построение карты сети.
- Регистрация сертификатов участников.
- Определение узких мест (бутылочное горлышко).
- Наблюдение за работоспособностью и перестройка в случае сбоя.

Service (U2S)

U2S - набор контрактов и протоколов, который позволяет всем приложениям системы работать по одним стандартам и протоколам. Это инструмент для

разработчиков с помощью, которого можно разрабатывать распределенные приложения.

Основная задача набора сервисов - разработка распределенных приложений без использования виртуальной машины. Такой подход позволяет упростить и ускорить работу системы, хотя и не лишен недостатков. Набор сервисов будет пополняться по мере развития технологии и потребностей.

На данный момент определены следующие сервисы:

- Events - подписка на события и вызов событий в сети.
- Streams - создание и контроль потоковой передачи данных.
- Storages - хранение, шаринг и кэширование файлов.
- Routes - расширение встроенного функционала маршрутизации пакетов.
- Mementos - отложенная отправка пакетов участникам вне сети.
- Polygons - система управления распределенными базами данных.
- API Connectors - механизм создания кастомных сервисов, интерфейсов между приложениями вне стандартных сервисов.
- Virtual Machines - альтернативный подход к разработке приложений, создание совместимости с блокчейн проектами на виртуальных машинах, например EVM(Ethereum Virtual Machine).

Application (U2A)

U2A - уровень распределенных приложений (UApp) запускаемых в сети U. Определяет правила их работы и взаимодействия.

Каждое UApp создает (регистрирует) набор необходимых сервисов U2S. Основное взаимодействие между участниками сети идет на уровне сервисов с привязкой к приложению. При необходимости приложение может создать кастомный сервис-интерфейс, который будет работать уже по внешней логике(вне библиотеки U). Доступ к UApp (набору его сервисов и данным) осуществляется через локальное API.

Решает задачи:

- Набор инструментов для разработки конечного распределенного приложения в U сети.
- Определяет правила установки и одновременной работы множества приложений внутри одного домена (множество устройств одного участника сети).
- Контроль за соблюдением правил и протоколов участников сети на уровне приложений.
- API для взаимодействия с приложениями на устройстве (вне системы U).

UNet

UNet - это глобальная оверлей сеть построенная на технологиях U.

Основные задачи сети:

- Единая база валидных участников сети
- Единый механизм быстрого поиска любого участника сети и установление с ним связи.
- Обеспечение региональной сегментации и устойчивости сети к внешним блокировкам и дроблениям.
- Базис, на данных которого можно разрабатывать различные совместимые друг с другом UApps.
- Предоставление UCoin, как платежного средства на начальном этапе существования сети (пока не появятся мосты с другими финансовыми инструментами).

NCA UNet

NCA(Network Control Application) UNet - это базисное UApp сети UNet. Оно определяет правила и механизмы работы сети UNet.

Основные задачи этого приложения схожи с задачами сети. По сути именно это приложение и создает сеть UNet.

Более подробнее о том, как именно работает это приложение можно прочесть в основном разделе этого документа.

Side UApps

Side UApps - это список возможных приложений/проектов, которые могут быть разработаны сторонними разработчиками. Он предоставлен в качестве демонстрации возможного пути развития технологии.

- UWallet - инструмент для работы с финансами и монетами(coins) в сети UNet.
- UVoice - чаты, группы, паблики, стримы, новостные ленты.
- UPage - личные страницы, сайты, информационные ленты.
- UStore - магазин, витрина.
- UNeed - сервис частных объявлений.
- UShare - файлообменный сервис.
- UVirtualNet - сервис создания виртуальных сетей на базе UNet.
- UAuth - авторизация физического пользователя, KYC.
- UWill - сервис самоуправления на базе UAuth.
- UDelivery - доставка товаров для работы в связке с UStore, UNeed.
- UTaxi - такси, может также работать в связке с UDelivery.

U

UFramework

UFramework библиотека, которая объединяет в себе уровни: Peer, Certificate, Network, Service, Application. Она используется для разработки конечных решений.

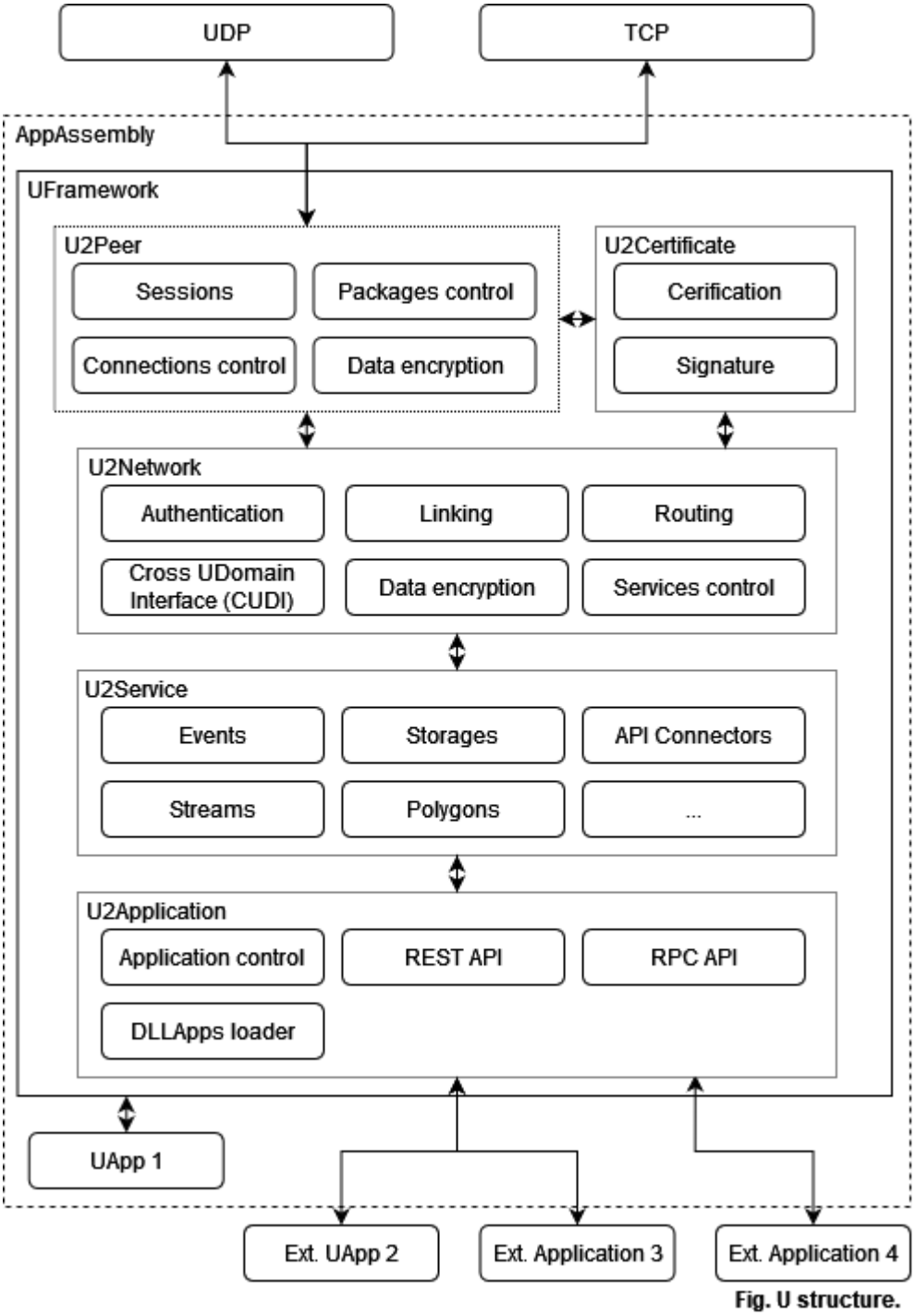

Fig. U structure.

AppAssembly

Сборка приложения U. Это сборка, основными элементами, которой является библиотека UFramework и управляющее сетью приложение (NCA). Также в сборку могут быть включены и другие дополнительные распределенные приложения (UApp).

Нужно различать понятие сборки, как программного комплекса в контексте устройства, так и узла сети - запущенной сборки на устройстве в контексте сети.

User domain

Это виртуальная область, среда, создаваемая всеми устройствами пользователя с запущенной системой U (схожей сборкой) на них и авторизованной под одним аккаунтом.

В этой области можно устанавливать и запускать UApps.

Единая виртуальная область позволяет оставлять ресурсоемкие приложения на домашних компьютерах или арендованных серверах, а управление, контроль и результат иметь на мобильных и маломощных устройствах.

Синхронность данных и исполняемых на них процессах достигается формированием вертикали подчиненности устройств. Определяется самим пользователем исходя из приоритетов.

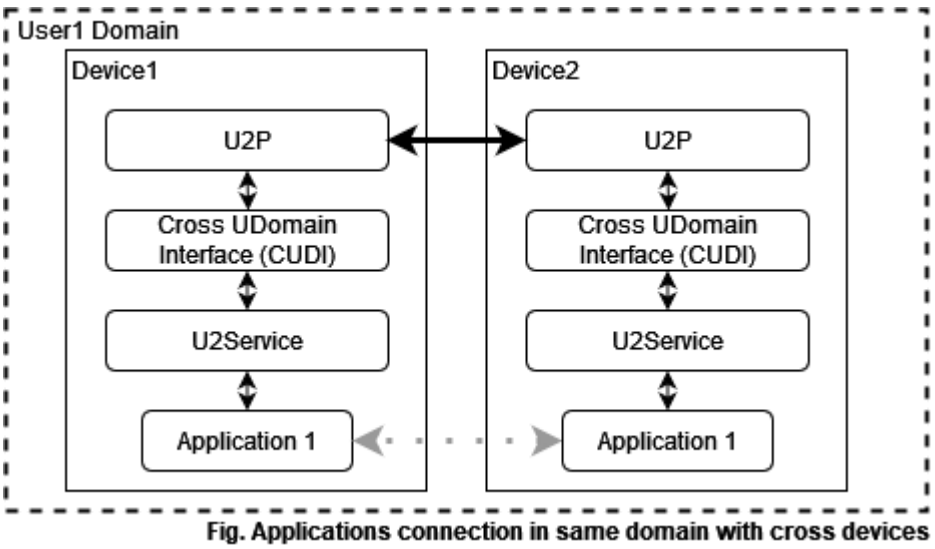

Fig. Applications connection in same domain with cross devices

UApplication

UApplication (UApp) - это программа, которая запущена в доменах участников сети, при этом он имеет одинаковую логику работы и настройки. Он объединяет этих участников в совместную подсеть для решения поставленных задач.

Сеть или подсеть создаваемая UApp, объединяющая всех участников сети, у которых установлено это же приложение называется UApplication Network (UAppNet).

Приложения могут взаимодействовать, как между собой в пределах одного домена, так и за его пределами с приложениями в других доменах. Для этого используется U2Services для стандартизации протокола взаимодействия [U2Services]. Например: UVoice (чат) обращается к UWallet (кошелек) для перевода монет между участниками чата.

Используя бесплатные шаблоны приложений пользователи могут запускать свои приложения и создавать свои сети. Используя один и тот же код приложения, но разные настройки, можно создавать и запускать разные приложения в одном и том же домене.

Существует несколько типов приложений.

Считаю, что все типы приложений, за исключением proprietary, должны быть open source и распространяется со свободной лицензией для соблюдения всех правил безопасности и открытости

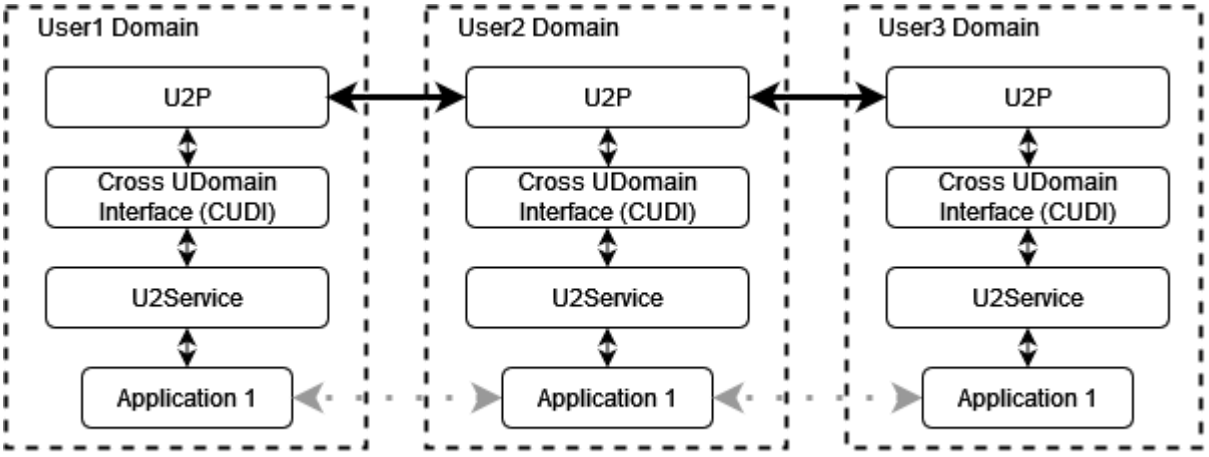

Fig. Cross domain Application connection.

Network Control Application

Network Control Application (NCA) - это приложение, которое определяет правила работы в основной сети (UMainNet): правила авторизации, маршрутизации и так далее.

Примером такого приложения является UNet для сети UNet [NCA UNet].

Другие UAppNets строятся, как ответвления от UMainNet, становясь ее частью. Приложения, которые формируют подсети, конфигурируются для взаимодействия с NCA.

При этом для создания небольшой сети не требуется NCA, и приложение может пользоваться базовыми инструментами U. Такая сеть будет закрыта для доступа из любой другой сети.

Далее рассмотрим основные типы сетей.

Open

Сеть созданная приложением доступна для всех. Каждый участник является узлом этой сети или может свободно стать таковым. Пример: files sharing.

Conditionally open

Сеть созданная приложением условно делится на две части. Одна - создатели, администраторы, которые имеют привилегированный доступ и возможности. Вторая - пользователи услуг и сервисов создаваемых сетью. При этом либо вся логика работы, либо все данные приложения открыты. Пример: паблик или открытая группа.

Private

Приватная сеть, схожа с conditionally open, но она полностью закрыта для внешнего мира. Пример: приватный чат или группа.

Proprietary

Административная часть сети, а также некоторая часть логики ее работы закрыты. Открытыми является только то, что сами разработчики открыли. Если эта сеть взаимодействует с данными имеющими ценность (токены, койны, ценная информация), в случае, когда есть опасность даблспендинга, то для ее работы необходимо соблюдение условий страхования ответственности, в противном случае сеть будет считаться небезопасной.

U2Peer (U2P)

U2P определяет механизм передачи данных point-to-point.

Формируется как транспортный протокол над UDP и TCP.

- UDP используется приоритетно. Он позволяет в большинстве ситуаций сформировать стабильное P2P соединение за NAT без внешнего IP.
- TCP используется только если один из P2P соединения имеет внешний IP, имеются проблемы с UDP и это не stream трансляция.

U2P определяет два вида сессий подключения:

- Anon - ограниченное по возможностям соединение, созданное для формирования Secured соединения.
- Secured - полнофункциональное защищенное соединение.

Реализует механизмы:

- Контроль версий с возможностью динамического изменения протокола.
- Проверка целостности сообщения.
- Защита соединения используя выданный алгоритм шифрования.
- Динамическая проверка пропускной способности сети между P2P.
- Сегментация сообщения и формирование пакетов с динамической длиной, подстраиваясь под пропускные возможности местной сети. Гипотетический размер сообщения в текущих версиях может быть до $470 * (2^{32+8} - 1)\, byte \approx 468\, Tbyte$ (без учета служебных заголовков). Протокол контролирует прием-передачу и проводит сверку полученных сегментов. Существуют настраиваемые лимиты на прием и передачу данных с определенных узлов.
- Ответа на определенное сообщение.
- Stream трансляция данных.
- Ретрансляция пакетов по маршрутному листу. Позволяет формировать соединения для тех кто не может соединиться напрямую (отсутствует техническая возможность), снимает нагрузку с нагруженных узлов (например: стрим с большим количеством подписчиков). Эта возможность опциональна и доступна только, если сам клиент разрешает ее использование.

U2Network (U2N)

U2N определяет логику работы системы на уровне сети.

Реализует механизмы:

- Авторизация
- Аутентификация
- Соединение
- Маршрутизация
- Контроль сервисов
- Cross UDomain Interface (CUDI)

Авторизация, аутентификация и шифрование

Рассматривать отдельно U2C мы не будем. Это просто криптографическая библиотека.

В U авторизация, аутентификация и шифрование непрерывно связаны, поэтому далее пойдет речь о всех используемых ключах шифрации и их назначении.

AKP и ACert

Для регистрация учетной записи в U необходимо сформировать пару ECC ключей AuthKeyPair (AKP), которая будут использоваться для шифрации и аутентификации. Публичный ключ и возможность его подтвердить приватным ключом являются по сути идентификатором в системе. AKP упаковываются в сертификат ACert, ограничивая срок работы ключа.

MKP

Также опционально, можно сгенерировать MasterKeyPair (MKP), для резервной смены AKP и восстановления доступа (конфигурация NCA), тогда MKP также упаковывается дополнительно в сертификат ACert.

AKLong, AKShort

AKP формируется по определенным правилам, кодируя дополнительную информацию (64 бита) о пользователе в саму себя. Этот код называется AKLong. Для этого используется $Hash_{CRC64}(AKPublic)$. Таким образом для регистрации придется генерировать много ключей перебором, пока не попадется с нужными параметрами. Кодируемая информация и ее формат определяется NCA [Network Control Application].

Функция CRC была выбрана, так как дала наиболее равномерное статистическое распределение для ECC ключей. В данном случае ее использование как Hash функции не создает проблем безопасности, т.к. кодируемая информация публична.

Для быстрой идентификации в сети используется $Hash_{CRC\,N}(ECPubKey)$, где N - сложность сети(определяется ее размером и NCA). Этот код называется AKShort. Это позволяет быстро определить хозяина или получателя пакета, использовать нужный сессионный ключ для дешифрации или отправить в место назначения.

Вероятность гипотетической ситуации коллизии AKShort, когда найдется несколько аккаунтов с одним кодом, дополнительно уменьшается уникальностью точки подключения. В случае возникновения коллизии U2P для общения с этой точкой переключается на пакеты со связанными заголовками. Они имеют дополнительный согласованный код для идентификации.

SSK

При создании P2P соединения, во время рукопожатия и формирования сессионного ключа шифрации (SSK) используется ECDH и AKP.

Далее SSK может использоваться с различными поточными шифрами, в зависимости от индивидуальной настройки P2P соединения. Для тестов используется HC-256 и ARC4.

Signing with an AKP

Для подписания публичных сообщений и действий используется ECDSA с использованием AKP.

Соединение, топология и маршрутизация

Основной проблемой большинства узлов сети является то, что они не имеют внешнего IP. Это значит, с таким узлом нельзя наладить связь не по его инициативе. А если у обоих узлов нет внешнего IP то, наладить связь без помощи посредника (Agent) вообще нельзя.

Давайте разберем все возможные ситуации и определим, как можно сформировать соединение.

1. Оба узла имеют внешний IP

   Самая простая ситуация. Нет никаких ограничений для формирования P2P соединения между User1 и User2.

2. Только один узел (User1) имеет внешний IP

   В этом случае User1 не сможет по своей инициативе установить связь с User2. Это значит, что связь необходимо будет поддерживать (пробрасывать PingPong пакеты) до тех пор пока в ней есть необходимость.

3. Оба узла не имеют внешний IP

   Это крайний случай, когда нет никакой возможности наладить связь между узлами без посредника. Так как в большинстве случаев будет именно эта ситуация, то вся система строится в приоритете на этот вариант.

Как обойти проблему 3й ситуации. Нам нужен стабильный посредник с внешним IP. С ним можно поддерживать постоянно связь, для автоматического формирования соединения U1<>U2, либо если есть сторонний канал связи достаточно каждому узлу кинуть запрос о сетевом состоянии посреднику, а после вручную прописать данные подключения.

Автоматический вариант (упрощенно): U1 кидает запрос A, что хочет соединиться с U2. Если у A нет соединения, то в ответ идет отказ. Далее A кидает запрос U2, что с ним хочет соединиться U1 и прикладывает сетевое состояние U1. Если U2, это соединение не нужно, то он кидает отказ и он по цепочке возвращается к U1. Если все ок, то U2 кидает пробивной пакет на U1, и возвращает A, что он ждет соединения. A возвращает ответ U1 о том, что U2 готов, отправил пробивной пакет и прикладывает сетевое состояние U2. U1 получив, что все хорошо, кидает свой пакет приветствия U2 по данным, которые получил от A. И вот тогда, если все прошло штатно соединение создается.

Зачем все так сложно? Нам необходимо, чтобы данные о U1 и U2 прописались в NAT-таблицах друг друга. А для этого нужно кидать UDP пакеты вслепую. Ни о каком TCP соединении здесь не может быть и речи.

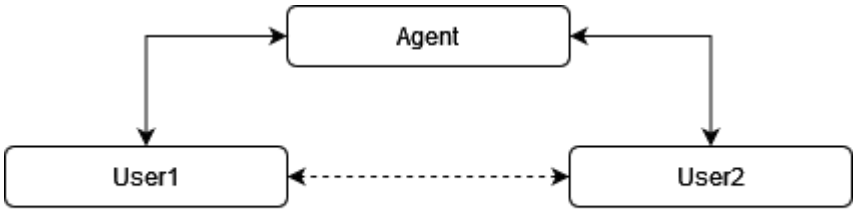

Fig. Connection through an agent

Зачем нужна топология и маршрутизация на этом уровне? Внимательно прочитав, то как можно сформировать соединение U1<>U2, вы могли заметить, что посредник не обязательно должен иметь внешний IP. Для соединения U1 и U2 нам достаточно того, чтобы они были с ним одновременно соединены. Но если пойти дальше, то вместо одного посредника у нас может быть цепочка посредников. Тогда для формирования соединения, нам нужно чтобы так или иначе по цепочке мы могли передать пакет от U1 к U2.

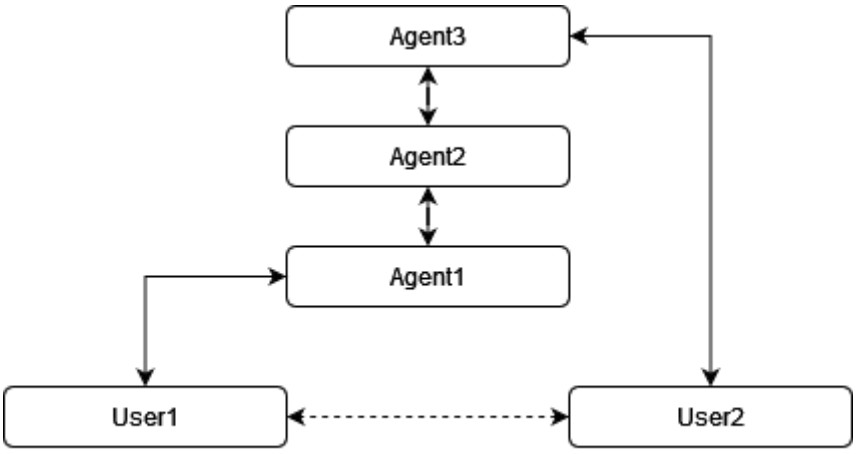

Fig. Connection through agents

Уровень предоставляет механизмы опроса сети, построения топологической схемы и поиска пользователя в сегменте. Используя эти механизмы и поддерживая постоянное соединение с одним из участников сегмента, всегда можно подключиться к другим участникам.

Нужно понимать, что применять такие механики на больших сетях очень накладно. Поэтому их применение ограничено небольшими группами, в которых не реализовано NCA. В противном случае сеть будет терять свою продуктивность.

Контроль сервисов

Несмотря на то, что для сервисов выделен отдельный уровень U2S, базовый контроллер сервисов находится на уровне U2N. Он работает на более низком уровне и является мостом между CUDI и сервисными интерфейсами.

Решаемые задачи:

- Регистрация и контроль сервисов.
- Контроль доступа к сервисам.
- Предоставление списка сервисов.
- Обеспечения связи между сервисами через CUDI.

Cross UDomain Interface (CUDI)

Набор инструментов и сетевых пакетов, который позволяет взаимодействовать сервисам и приложениям между доменами, а также внутри домена, но между разными устройствами.

По сути это расширение базовых пакетов U2P дающее возможность контролировать источники и получателей пакетов, а также предоставлять информацию о рабочих на сетевом узле сервисах и приложениях.

Признаки контроля:

- домен
- устройство
- приложение
- сервис

U2Service (U2S)

U2S - это набор контрактов и протоколов, который позволяет всем приложениям системы работать по одним стандартам и протоколам. Это инструмент для разработчиков с помощью, которого можно разрабатывать распределенные приложения.

Существует много разных сервисов и каждый отвечает за свой функционал.

Запущенные приложения способны запускать свои сервисы, со своими настройками и характеристиками, а также подключаться к сервисам других экземпляров приложений в других доменах. То есть между доменами соединение возможно только между одним и тем же приложением, но запущенным в разных доменах. [Fig. Cross domain Application connection].

Каждый сетевой узел при подключении может предоставить полный список доступных публичных сервисов и доступных приватных. Таким образом при формировании P2P соединения каждый из участников знает по каким приложениям можно взаимодействовать друг с другом.

Каждый из сервисов обладает большим наборов настроек, позволя формировать нужные уровни доступа для разных пользователей и приложений.

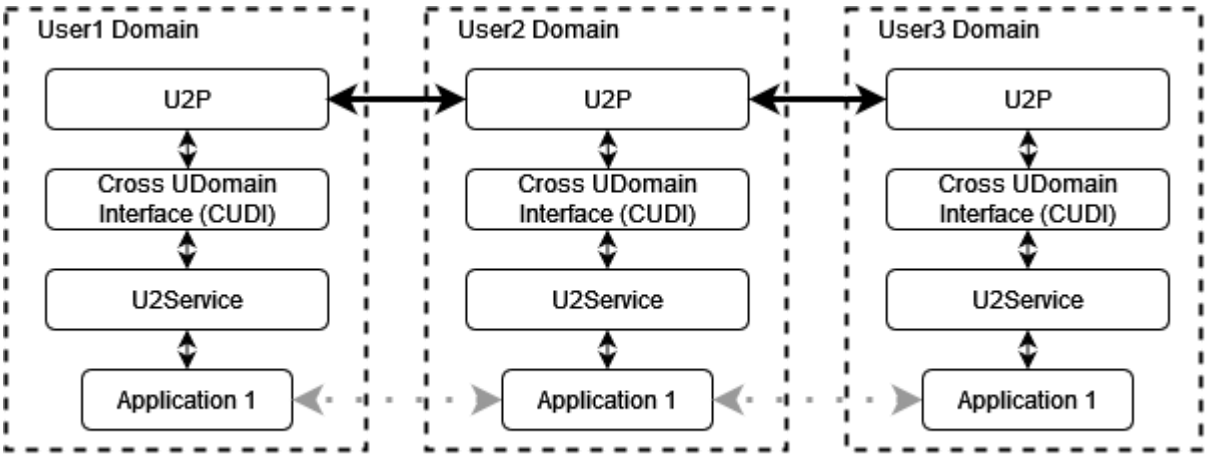

Fig. Cross domain Application connection.

API Connectors

Это сервис основной задачей, которого является формирования API между приложениями вне стандартных сервисов.

API Connectors предоставляет инструменты для формирования связи между приложениями одного типа (т.е. одинаковое приложение запущенное на разных устройствах) и связи между разными приложениями в пределах одного устройства.

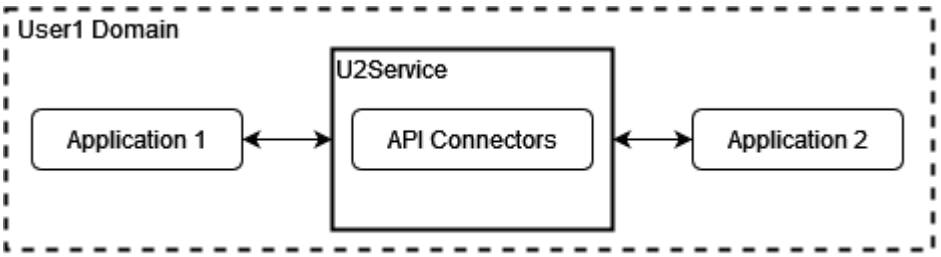

Fig. Applications connection in same domain.

Events

Механизм позволяющий создавать и подписываться на события. Так приложение может создать точку подписки на событие. События бывают приватными (для ограниченного списка пользователей) так и публичными. Настройками также ограничивается кол-во подключений и виды приложений. В момент вызова приложением события, всем подписавшимся на него уйдет уведомление о событии.

Streams

Участники сети могут создавать свои трансляции, подписываться на транслируемые стримы данных других участников. Этот функционал позволяет

быстро и просто организовывать трансляцию потоковых данных без проверки на доставку, например: аудио и видео.

Совмещая с сервисом Routings можно создавать обширные распределенные трансляции.

Storages

Сервис хранения, шаринга и кэширования файлов.

Приложения могут создавать или использовать существующие папки и раздавать их содержимое в сеть. Скачивать файлы с других устройств и кэшировать их для использования на своей стороне.

Это сервис позволяет приложениям просто и удобно использовать различные распределенные файлы в своей работе.

Примеры: хранение своей и кэширование чужих иконок из чата; используя сервис Polygons можно создать файловый регистр и делиться файлами в сети.

Routings

Сервис позволяет приложениям создавать и управлять сложными маршрутами соединения для поддержания более качественной работы сети. Основывается на встроенных механизмах U, дополняя их более сложной логикой.

Использование сервиса определяется настройками приложения, уровней доступа, коммерческой составляющей. Не каждый захочет транслировать чужие сообщения без веской причины, а потребность в таких трансляциях будет.

Топологии ретрансляций:

- Everyone with everyone (EwE). Базовое соединение. Имеет смысл только в очень маленьких сетях.
- Tree. Удобно в случае коммерческих больших трансляций.
- Ring. Используется в небольших сетях с непродолжительным временем жизни.
- Ring of rings/Trees of rings. Гибрид дерева и кольца. Удобно в случае больших некоммерческих трансляций, когда необходимо охватить наибольшее кол-во отдельных доменов. Здесь достаточно много вариаций с разными характеристиками для каждой ситуации.
- Cross rings. Используется в средних по размеру сетях, но с гарантированной доставкой сообщений и поддержкой активного соединения.

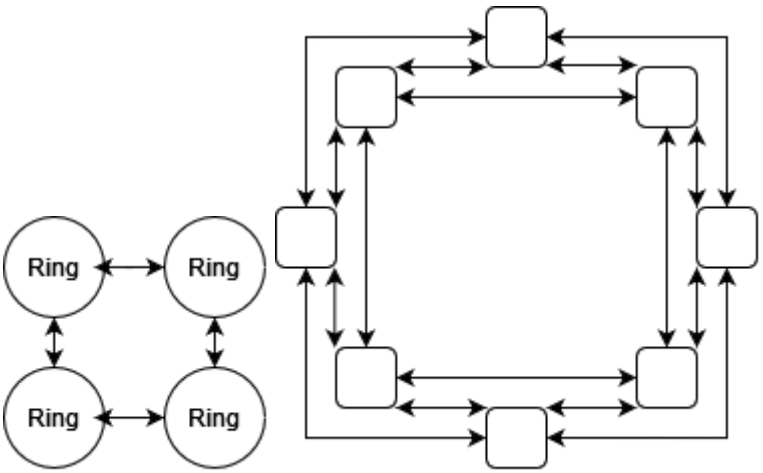

Fig. Ring of rings

Fig. Cross rings

Пример. Создан чат, в котором не более 5 участников. В этом случае каждый может сформировать прямую связь друг с другом и это не будет проблемой. Но количество стало уже больше, около 10. И это уже проблема. Оптимальнее было сразу создать маршрут - пересекающиеся кольца (cross rings), для минимизации количества соединений и сохранения скорости реакции системы. А далее при еще большей нагрузке (более 100 участников), множественные соединения настолько усложняют работу, что можно пожертвовать скорость доставки, и построить дерево колец (tree of rings).

Mementos

Сервис отложенной доставки сообщений. Позволяет приложениям воспользоваться услугами стороннего домена для отложенной доставки сообщения. Используется, если пользователя нет в сети, а у вас нет возможности или вы не хотите дожидаться его возвращения в сеть.

PolygonsKit

PolygonsKit - Это набор алгоритмов для создания Распределенных Систем Управления Базами Данных (РСУБД), Distributed Database Management System (DDBMS).

Используя его можно объединять участников UAppNet для решения поставленных задач по работе с данными.

Основная задача полигонов - дать понятный и удобный инструмент разработчикам, который позволит создавать и управлять распределенными БД в своих приложениях.

Polygon

Полигон создан для решения конкретной задачи, прописанной в исходном коде и определенной в конфигураторе. Он создан на базе U протоколов, обеспечивая совместимость с другими полигонами.

Домены UAppNet принимают на себя обязательства и роли полигона, создавая РСУБД.

Полигоны могут быть:

- полностью изолированы и решать приватные задачи;
- открытыми, иметь открытые данные или соединения для работы с другими полигонами с совместными данными.

Roles

Каждый полигон базируется на участниках с разными ролями.

Owner

Хозяин полигона или хозяева (совместное владение), кто создали и инициализировали полигон, либо получили в собственность по регистру. Они обеспечивают транзакции и хранение БД. Хозяева могут делегировать их задачи по БД спартанцам или хранителям. Может быть использовано страхование ответственности.

Spartan

Спартанец - участник РСУБД, тот кто проводит транзакции и хранит чужую БД за комиссию или бесплатно. Может быть использовано страхование ответственности. Спартанец может быть использован как щит чтобы скрыть хозяев. Хозяева могут спрятаться и быть недоступными из сети, но при этом могут напрямую быть подключенными к спартанцам для валидации их работы.

Keeper

Хранители это участники РСУБД кто хранит чужую БД за плату или бесплатно. У хранителя две задачи. Первая, хранить БД и обеспечивать доступ к копии, разгрузка центральных узлов. Вторая, мониторинг соблюдения правил. Если хранитель найдет форк-цепочку, то он может сразу получить награду за это (нужно понимать, что форк-цепочка здесь может появиться, то если кто-то жульничал) .

Types

Registry polygon (RP)

Легкая РСУБД для случаев, когда проблема double-spending не имеет смысла. Основная задача - создание подписанных записей в таблице. Каждый спартанец и хозяин имеет собственную область доступа к данным, в которой он работает. Пересечение областей доступа возможно путем создания таблицы доступов во время инициализации. Консенсус достигается простым соблюдением правил. Если кто-то их сломает, он просто потеряет доверие как член сети.

Особенности.

- Простота реализации
- Контроль доступа
- Записи могут быть удалены или отредактированы
- Скорость транзакции обратно пропорциональна количеству спартанцев.
- Масштабирование (используя Events)
- Не может использоваться, когда есть опасность дабл спендинга.

Insured liability polygon (ILP)

Используется для создания больших РСУБД решений с открытым исходным кодом. Гарантия честности обеспечивается страхованием ответственности спартанцев.

Особенности.

- Не может быть быстрой (если много пользователей).
- Proof of stake (от размера страховки) абсолютного большинства.
- Хозяева не могут проводить транзакции, только спартанцы.
- Работа без валидации хозяев.
- Ретрансляция цепи (использование Хранителей)
- Страховка ответственности Спартанцев.
- Использование только сторонней цепь, как депозит безопасности (чужая цепь для формирования страховки и ее выплаты).

Insured liability proprietary polygon (ILPP)

Используется для создания проприетарных РСУБД решений, где скорость важна. Гарантия качества обеспечивается страхованием ответственности как хозяев так и спартанцев.

Особенности.

- Небольшая (число участников) => быстрая скорость транзакции
- Proof of stake большинства (>50%; =100%)
- Ретрансляция цепи (использование Хранителей)

- Работа в тени (использование спартанцев как щит)
- Страхование ответственность спартанцев и хозяев
- Определение чужой цепи для депозита страховки (в случае проблем с ценностью собственной цепи/продукта)

Hybrid polygon (HP)

Комбинирование нескольких разных полигонов и соединение их в общую сеть. Это позволяет создать гибкую систему, с учетом сегментации сети и поставленных перед ней задач.

Segmented polygon (SP)

Полигон, соединяющий полигоны разных сегментов сети. Используется для масштабирования на основе сегментации.

Custom polygon (CP)

Используя протоколы U, каждый может создавать собственные полигоны с использованием различных технологий. Мы не ограничиваем это. Используйте все, что хотите. Это может быть блокчейн-система с proof-of-work, централизованная конфигурация серверов или что угодно.

Consensus and liability insurance

Сокрытие проведенной транзакции от хранителей или создание форков (если обнаружатся две разные транзакции подписанные спартанцем после одной и той же) приводит к нарушению смарт-контракта и потери страхового взноса. Он делится между тем, кто обнаружил проблему (50% - абсолютно любому участнику сети) и теми, кто ее подтвердил (хранители, спартанцы или хозяева делят оставшиеся 50%).

Во время критической транзакции участники (не участники DDBMS) сверяют хеш по индексу записи, и в случае отсутствия у них этой информации могут поинтересоваться у спартанцев, хранителей, владельцах о верности данных. Если будет несоответствие, то можно смело сообщать любому из них о нарушении цепочки и получать вознаграждение. Осознанное игнорирование и создание ложной цепочки не имеет смысла, так как проверочные хэши все равно не совпадут.

U2Application (U2A)

U2A предоставляет инструменты по регистрации приложения в домене, установке и подключению UApp к сборке, контролю за установленными приложениями.

Некоторые UApp могут быть интегрированы в сборку AppAssembly на этапе разработки. Но это метод крайне неудобен и не подходит для поставленных задач.

U2A предоставляет еще два варианта подключения приложений к сборке:

- RPC и REST API
- DLLApps loader

Благодаря RPC и REST API UApp может выполняться вообще за пределами AppAssembly и взаимодействовать со сборкой только через API. Также API используется для взаимодействия внешних приложений не относящимися к системе U с UApp.

UApp может быть динамически подгружено в среду выполнения AppAssembly, как сторонняя библиотеки, благодаря поддержке библиотекой UFramework динамической подгрузки UApp.

UNet

Оверлей сеть использующая технологии и протоколы описанные ранее.

Ее ядром является набор приложений с RP и ILPP полигонами (на начальном этапе, с возможностью будущей передачей управления сообществу). Каждое из приложений решает основные проблемы подобных сетей: роутинг, контроль масштабирования, контроль и проведение быстрых транзакций, обеспечение залоговой базы для страхования ответственности в других приложениях.

UNet сегментирована по административному признаку. С масштабированием сети эта сегментация будет увеличиваться. Вопрос - по какому признаку делать сегментацию был очень острым, но такое решение позволяет решить сразу множество проблем. Так крипто-система сможет органично сплестись с местной законодательной базой и будет учитывать местные особенности. Мы чаще всего взаимодействуем с пользователями близкими нам по территории. Территориальная сегментации решает вопрос быстрого и прямого доступа. Поэтому стоить DDBMS из участников, которые ближе всего друг к другу, наиболее оптимальное решение. В случае локдауна это позволит продолжить сети работать в автономном режиме без сбоев в той территории, которая отделена.

UNet должна создать экосистему для развития децентрализованных приложений. Позволить людям взаимодействовать друг с другом напрямую минуя ненужных посредников. Создавать свои группы, проекты, системы и сети. Это будет просто и обыденно, как скачать приложение из стора.

NCA UNet

Это базовое приложение сети UNet. Его полигоны формируют начальную инфраструктуру для функционирования сети.

UPRoute

При установке и регистрации каждый пользователь UNet задает свое местоположение и приложение подбирает пару ключей, таким образом чтобы hash публичного ключа соответствовал определенному шаблону. В младших байтах хэша кодируется административное местоположение участника. Таким образом зная публичный ключ, или его хэш, всегда можно точно определить к какому сегменту сети относится этот участник сети. Так как к каждому сегменту привязаны несколько спартанцев, то можно всегда очень быстро через них наладить прямое P2P соединение. Так как известны страна и регион пользователя то теперь легко определить можно ли ему предоставить доступ в соответствии с местным законодательством.

UPRoute - segmented RP-polygon with limited access to control of segments. Долевое владение распределено между 5ю участниками с 50%+1 большинством на каждом

условном сегменте. Owners in shadow. Спартанцем можно стать, только после подтверждения владельцами заявки на спартанца. Спартанцы этой сети обязаны быть онлайн 24/7 в своем сегменте и обеспечивать роутинг для участников сегмента. Спартанцы имеют возможность принимать заявки на спартанцев этого полигона. При подаче заявки создается смарт-контракт, будущий спартанец должен внести страховой взнос в полигон UPCoin. Кол-во хранителей неограниченно.

При сегментировании всем участникам, которые попадут не в свой сегмент (крайний бит в хэше не совпадает), официальные полигоны UNet предложат переместиться в нужный для них сегмент без комиссии в течение 30 дней.

Состоит из трех таблиц: Сегменты, Заявки и Спартанцы.

Public spartans methods:

UNetNodes[] GetNodes(UInt32? segment);

void CreateOfferOfNewSpartan(UPCoinSpartanContract contract);

После стабилизации сети и формирования системы управления [UWill] долевое владение будет передано созданному управляющему органу.

UPCoin

Этот полигон обеспечивает транзакции UCoin внутри сети UNet. Служит как side-chain for provide security deposit.

UCoin - выпущенный коин в лимитированном количестве как платежное средство на начальном этапе работы сети.

UPCoin - segmented ILPP-polygon.

Спартанцами данного полигона являются спартанцы полигона UPRoute. В качестве платы за работу в UPRoute они получают возможность получать комиссию за транзакции в UPCoin. Размер комиссии определяется хозяевами UPCoin. Спартанцы сегмента после проведения транзакций на сумму более 30% от стоимости своей страховки, должны инициировать процедуру аудиторской проверки/сверки с хозяевами сети по проведенным транзакциям. Для формирования блока транзакций необходимо получить согласие одного из хозяев сети. Информирую таким образом о текущем сумарном лимите по транзакциям. Даже если этот хозяин сети будет скомпрометирован, другие хозяева это смогут определить по сообщениям либо от спартанцев, либо от хранителей.

Хозяева сегмента полигона также страхуют свою ответственность. В случае нарушения протокола, спартанцы могут предоставить доказательства другим сегментам и зарегистрировать свое вознаграждение. Они хранят весь сегмент транзакций и поддерживают взаимосвязь с вышестоящими полигонами контроля.

Межсегментная транзакция проводится с участием спартанцев двух сегментов. После, во время сверки хозяева двух сегментов проведут взаимную повторную сверку.

Ввод и вывод средств с внешних токенов через спартанцев осуществляют основные сегментные полигоны UPCoin.

Public spartans methods:

UCoinTrasaction DrawInFromExternalToken(ExternalToken token);

UCoinTrasaction DrawOutToExternalToken(ExternalToken token, decimal value);

UCoinTrasaction Transfer(UNetPKHash target, decimal value);

UCoinTrasaction Transfer(UNetPublicKey target, decimal value);

UPCoinSpartanContract CreateSpartanContract(UInt32 segment);

UCoinContract CreateContract(UCoinContract contract);

UCoinTrasactionsResponce Check(UNetPKHash target, UInt64 fromTrId, Int16 take);

UCoinTrasactionsResponce Check(UNetPublicKey target, UInt64 fromTrId, Int16 take);

UCoinTrasaction Eureka(UCoinTrasaction tr1, UCoinTrasaction tr2);

UCoinTrasaction Eureka(UCoinTrasaction tr1, UCoinTrasactionsResponce trsResponce);

Side applications

Сторонние приложения - это концептуальные предложения UApps, которые могут быть разработаны на базе технологии U и стать частью инфраструктуры UNet. Нужно понимать, что представленный список ниже условный. Его задача задать общий тон развития идеи. По мере развития технологии список будет пополняться и корректироваться.

UVoice

Один из первых проектов, которые планируется разработать. Это приложение для P2P общения. Переписка, голосовые сообщения, обмен файлами, звонки аудио/видео, стримы. UVoice должно стать наглядной демонстрацией технологии и будет развиваться совместно с развитием U.

UWallet

Прямой необходимости в этом приложении на начальном этапе проекта нет. UWallet должно стать своего родом мостом между UNet и другими финансовыми

системами как фиатными, так и блокчейн. В NCA UNet базово заложен финансовый механизм, но он ограничен. Рано или поздно в системе появятся различные мосты, новые валюты и так далее. Вот тогда и появится смысл в разработке приложения агрегатора.

UPage

Достаточно простая идея для создания личных страничек и сайтов. Достаточно построить реестр доменных имен со сроком регистрации и возможностью продления.

На узле организовать ретрансляцию HTTP сервиса. В более упрощенном варианте, это может быть вообще просто доступ к папке со статическим контентом.

UStore

На базе UPage можно реализовать механизм онлайн-магазина, витрины и так далее. Также приложение может предоставлять интерфейсы для заказа товара и ретрансляции заказов в сторонние сервисы для доставки.

UShare

Развивая идею шаринга файлами, можно создать приложение с реестром файлов или пакетов, который позволит делиться этими файлами. Ближайший аналог - это torrent. Только в данном случае, у нас не будет единых торрент-трекеров, т.к. реестр распределенный.

UTaxi

Для работы таксистам нужен агрегатор, который будет собирать и раздавать заказы. И здесь возникает ситуация, при которой они становятся зависимы от агрегатора, и по сути работают на него, отдавая немалую часть дохода.

Теперь представим, что каждый таксист вместо приложения агрегатора, запускает у себя UApp, которое вместе с другими создают сеть таксистов города. В этом приложении прописаны все правила работы, выдачи заказов, определения стоимости, поощрения и наказания. Все эти правила равны и открыты для всех участников сети. Более того, сами участники заинтересованы в честности, качестве услуг предоставляемых ими. А это - самоконтроль качества, при том жесткий, т.к. ошибка одного влияет на других.

Клиенты просто подключаются к этому приложению, как гости, и делают заказ. Всю его обработку осуществляет сеть из устройств таксистов.

UDelivery

Приложение схожее с UTaxi, но отличающееся спецификой: доставка не пассажиров, а грузов и товаров.

UNeed

Приложение объявлений. Куплю, продам, обменяю, ищу и т.д. Требует нестандартного подхода к сегментации. Наиболее сложным является аспект агрегации таких объявления, т.к. сами объявления могут находиться на устройствах их выложивших. Но стоит учитывать, что такие объявления работают, как правило, в масштабах административного деления, а значит есть признак для кластеризации и построения иерархии.

UVirtualNet

Расширяя возможности протокола, можно создать приложение для создания приватных виртуальных сетей. Это откроет огромные возможности для взаимодействия пользователей в сети, т.к. по сути убирает большинство интерфейсных ограничений. Но здесь, скорей всего, будут ограничения в виду сложностей работы с большим кол-вом участников.

UAuth

KYC приложение, реализация которого, возможна, на мой взгляд, только с помощью властей тех регионов, где оно будет запускаться. Это относительно спорное решение в плане свободного интернета, но открывающее огромные возможности в построении по-настоящему цифровой экономики.

UWill

Последнее и наиболее утопичное приложение среди всех представленных. Его идея заключается в построении на базе UAuth механизмов принятия законодательных актов и их исполнения в масштабах сети UNet. Этот проект сам по себе заслуживает отдельного исследования и технического описания, поэтому здесь озвучена только идея.