TON: TELEGRAM OPEN NETWORK

Новицкий И. О., Шилин Л. Ю.
Кафедра систем управления, Белорусский государственный университет информатики и радиоэлектроники
Минск, Республика Беларусь
E-mail: ionovitsky@gmail.com

Введение

Больше года назад стало известно о планах мессенджера Telegram выпустить собственную децентрализованную сеть Telegram Open Network. Тогда стал доступен объемный технический документ, который, предположительно, был написан Николаем Дуровым и описывал структуру будущей сети. Недавно в сеть был выложен документ с описанием технической составляющей новой сети TON

I. Уровни системы

- TON Blockchain. Самый низкий уровень системы;
- TON P2P Network. Пиринговая сеть, на основе которой будет построена работа системы;
- TON Storage. Файловое хранилище, которое независимо от блокчейна будет построено на вышеупомянутой пиринговой сети;
- TON Proxy. Это сервис, цель которого повысить анонимность участников сети. Любой пакет можно отправить не напрямую, а через туннели-посредники с дополнительным шифрованием, подобно I2P или TOR;
- TON DHT. Распределенная хэш-таблица для хранения произвольных значений. Она тоже построена поверх TON Network и помогает TON Storage находить «раздающие» узлы, а TON Proxy — промежуточные ретрансляторы;
- TON Services. Платформа для произвольных сервисов. Обмен данными производится через TON Network/TON Proxy, а логика приложений создается в смарт-контрактах самого TON Blockchain;
- TON DNS. Преобразователь url адресов в 256-битные адреса — аккаунтов, контрактов, сервисов и узлов;
- TON Payments. Платежная часть системы, использует собственную валюту «грамм», также поддерживает возможность работать с валютой других блокчейн сетей.

II. ADNL: Abstract Datagram Network Layer (Абстрактная датаграмма сетевого уровня)

Как и у в других распределенных сетях, работа сети TON поддерживается узлами. На низком уровне узлы имеют IPv4/IPv6-адреса и общаются по протоколу UDP, на более высоком — обладают абстрактными адресами и реализуют протокол ADNL. Чтобы один узел мог послать пакет другому, он должен знать один из его публичных ключей (и, следовательно, адрес, который им определяется). Он зашифровывает пакет этим ключом и добавляет в начало пакета 256-битный адрес получателя — поскольку один узел может иметь несколько таких адресов, это позволит ему определить, какой ключ использовать для расшифровки.

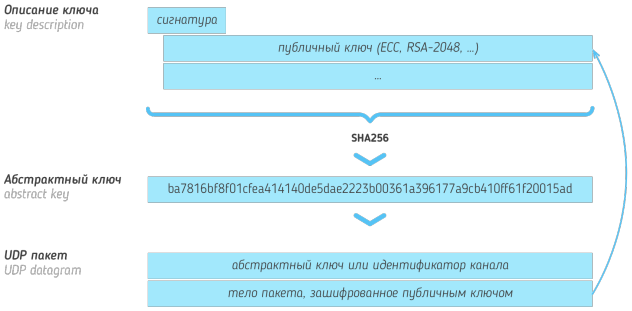

Рис. 1 – Процесс передачи сообщения от узла к узлу

Кроме того, вместо адреса получателя в начале пакета данных может находиться идентификатор канала. В таком случае обработка пакета уже зависит от конкретных договорённостей между узлами — например, отправленные в некий канал данные могут предназначаться другому узлу и должны быть ему переадресованы (это и есть сервис TON Proxy). Другим частным случаем может быть взаимодействие напрямую между узлами, но с шифрованием по индивидуальной паре ключей для этого канала.

Наконец, специальным случаем является нулевой канал — если узел ещё не знает публичных ключей соседних узлов, он может посылать им пакеты без шифрования совсем. Это предназначено только для инициализации — как только узлы пришлют информацию о своих ключах, их стоит использовать для дальнейшего взаимодействия.

Вышеописанный протокол (256 бит идентификатора канала + содержимое пакета) называется ADNL. Документация TON упоминает возможность реализации аналога TCP поверх него или собственной надстройки — RLDP (Reliable Large Datagram Protocol).

III. TON DHT: Распределённая хэш-таблица

Как в случае с другими распределёнными системами, TON предполагает реализацию DHT — распределённой хэш-таблицы. Более конкретно — таблица является Kademlia-подобной.

В абстрактном смысле, DHT ставит в соответствие 256-битным ключам некие бинарные значения произвольной длины. При этом ключи в таблице — это хэши от определённой TL-структуры (сами структуры тоже хранятся вместе с DHT). Это очень похоже на формирование адресов узлов — и они действительно могут присутствовать в DHT (например, по такому ключу может находиться IP-адрес узла соответствующего заданному абстрактному адресу, если он не скрывает его). Но в общем случае, прообразы ключей (их описания, key descriptions) — это метаданные, которые указывают на владельца записи в хэш-таблице (то есть публичный ключ какого-то узла), тип хранимого значения и правила, по которым эта запись может впоследствии изменяться. Например, правило может разрешать изменять значение только владельцу — или запрещать изменение значения в меньшую сторону (чтобы защититься от replay-атак).

Кроме 256-битных ключей вводится понятие DHT-адресов. Разница с обычными адресами узлов в том, что DHT-адрес обязательно привязан к IP-адресу. Если узел не скрывает своего IP, он может использовать обычный адрес для DHT. Но чаще для нужд DHT будет заводиться отдельный, полу-постоянный адрес.

Рис. 2 – Формирование адреса получателя

Над ключами и DHT-адресами вводится понятие расстояния — в этом всё совпадает с таблицами Kademlia — расстояние между ключами равно XOR (побитовому исключающему ИЛИ) от них. Как и в таблицах Kademlia, значение, соответствующее некоему ключу, должно храниться на s узлах, имеющих наименьшее расстояние до этого ключа (s тут — относительно небольшое число).

Для того, чтобы узел DHT мог взаимодействовать с другими такими узлами, он держит в памяти таблицу маршрутизации DHT — DHT- и IP-адреса узлов, с которыми он взаимодействовал до этого, сгруппированные по расстоянию до них. Таких групп 256 (они соответствуют старшему выставленному биту в значении расстояния — то есть узлы на расстоянии от 0 до 255 попадут в одну группу, от 256 до 65535 — в следующую, и т.д.). Внутри каждой группы хранится ограниченное число «лучших» узлов (в плане пинга до них).

Каждый узел должен поддерживать несколько операций: сохранение значения для ключа, поиск узлов и поиск значений. Поиск узлов подразумевает выдачу по заданному ключу ближайших к нему узлов из таблицы маршрутизации; поиск значений — то же самое, за исключением ситуации, когда узлу известно значение для ключа (тогда он просто возвращает его). Соответственно, если узел хочет найти в DHT значение по ключу, он посылает запросы небольшому числу ближайших к этому ключу узлов из своей таблицы маршрутизации. Если среди их ответов нет искомого значения, но есть другие адреса узлов, то запрос повторяется уже к ним.

Список литературы

1. Николай Дуров. Telegram Open Network [Электронный ресурс] // URL: https://test.ton.org/ton.pdf