Proof-of-Union — алгоритм консенсуса в блокчейн системах базируемый на сотрудничестве узлов

Коваленко Геннадий Александрович

1. Введение

В настоящее время существует огромное количество консенсус алгоритмов для блокчейн систем, каждый из которых имеет свои преимущества и недостатки присущие только ему, либо целому классу сходных алгоритмов. Так или иначе, в данное время лидирует две концепции консенсуса - основанные на майнинге (PoW) [1] и форжинге (PoS) [2], которые в свою очередь представляют конкурентную и последовательную модели генерации блоков непосредственно. Такое разделение либо предполагает крайне большое расходование материальных ресурсов, либо представляет собой необходимость комбинации с другими методами консенсуса [3], что приводит к сложности реализации, а следовательно и к проблеме доказуемой безопасности конечного решения [4, с.319]. Альтернативной моделью конкуренции и последовательности может являться алгоритм объединения узлов (PoU), решающий общую задачу сообща и главным преимуществом которого является простота реализации, сродни PoW и быстрота генерации блоков, эквивалентная PoS.

2. Становление

Зарождение PoU представляет собой вполне закономерное развитие консенсус алгоритмов, представленных в качестве синтеза уже ранее созданных и применяемых методов. Обобщая, можно представить эволюцию блокчейн систем в следующих этапах становления.

1. Конкуренция. Представляет собой генезис блокчейн систем, тезис и начало развития непосредственно. В такой модели каждый узел выполняет однотипные и сложно-математические операции полностью децентрализовано, несвязанно между собой, что порождает выполнение одинаковых действий многократно, в большей степени без всякой полезной нагрузки. Отрицательным качеством безусловно является необходимость поддерживать конкуренцию, быть соперником относительно всей другой сети, удерживать свои мощности, что в конечном счёте приводит, во-первых, к необходимости постоянного потребления майнинг-аппаратуры (процессоры, видеокарты, интегральные схемы и т.д.), во-вторых, к логичному завышению цен на подобную аппаратуру, методом искусственного дефицита средств производящими монополиями относительно большего спроса [5], в-третьих, к неостановимому поглощению электроэнергии и скорейшему разрушению потреблённой аппаратуры [6]. Положительным качеством же является простота описания алгоритма и независимость от сети, что и приводит к доступности анализа конечной безопасности.

2. Последовательность. Развивается как отрицание, антитезис конкуренции, избавляясь от самых негативных её черт. В такой модели узлы выполняют валидацию блоков поочерёдно, один за другим по общему алгоритму выбора валидатора. Сложность реализации последовательной концепции начинается на уровне сетевого консенсуса, когда строится

сильная необходимость и даже зависимость валидации текущего и главного proposer валидатора всеми другими узлами сети, что приводит к возможности существования постоянных разветвлений посредством проблем «ничего на кону» и «двойной траты». Для такого случая существуют дополняющие алгоритмы, способные обеспечить сетевой консенсус между узлами, подобно ядру Tendermint, базируемом на задаче византийских генералов [7]. Безусловным положительным качеством такой модели является отсутствие в необходимости конкурировать, что приводит к достаточно простой аппаратной составляющей, не приводящей к трате большого количества электроэнергии и быстрому устареванию используемой техники.

3. Объединение. Развивается как синтез конкуренции и последовательности, вбирая в себя простоту, монолитность и скорость, энергоэффективность. В такой модели не существует более валидаторов как отдельных и индивидуальных участников сети, теперь более блок не подписывается кем-то конкретным и определённым. Единственным и монопольным валидатором в конечном счёте становится сама сеть валидаторов, потому и не имеет значения кто станет proposer’ом блока, что приводит к финальному отрицанию последовательности как таковой. Также вторым отличием от двух предыдущих классов консенсусов является концепция слияния разных взглядов и видений на генерацию блока, вместо выбора единственно верного из предложенного множества, что приводит к однозначному и наиболее лёгкому поиску решения методом объединения информации.

3. Определение

Главным отличием PoU консенсуса является зацикленность системы на сохранение транзакций больше, чем самой связи между блоками, что является противоречивым явлением для блокчейн структур в целом. Суть заключается в том, что существует у таких алгоритмов N-ое количество блоков, подобно транзакциям, находящимся в статусе ожидания, pending’е, и в это же время существуют блоки, генерируемые после них, наперёд. Таким образом, можно сказать, что помимо транзакций, существуют также блоки, находящиеся в своеобразном mempool пространстве.

Будучи блоками неподтверждёнными, они не согласованы с сетью полностью. В таком периоде они перезаписываются и видоизменяются, находясь в этапе полиморфизма своих внутренностей. Блоки же подтверждённые располагаются ранее всех неподтверждённых, они устоявшиеся и представляют уже статичность внутренностей, неизменность заложенных транзакций.

Механизм доказательства объединения состоит из трёх основных и последовательных функций выдающих блок в разный период его состояния.

1. Принятие (accept). При вызове данной функции происходит образование нового блока на базе взятых транзакций с mempool’а. При таком действии не происходит какого бы то ни было согласования с сетью, а следовательно само подтверждение является неполным, не приводящим к окончательному решению консенсуса. После данного этапа начинается временная задержка перед последующим вызовом функции accept, необходимая для согласования с сетью уже на основе второй функции merge.

2. Слияние (merge). При вызове данной функции происходит изменение ранее созданного блока с блоком принятым из вне. В такой функции особенно важен сам механизм соединения двух разных блоков в один целостный, чтобы отправитель мог прийти к точно

такому же результатному блоку, как и получатель. Одним из самых простых решений может являться сортировка транзакций с помещением *N*-ого количества из полученного множества в блок. Все отброшенные транзакции после слияния попадают снова в mempool, ожидая следующего блока.

3. Фиксирование (commit). При вызове данной функции происходит конечное согласование блока с сетью методом подсчёта количества одинаковых блоков и выбора наиболее встречаемых из полученного множества. Предполагается, что большая часть валидаторов будет говорить истину, а потому и сохранение итогового блока в конечном счёте будет основываться на количественной характеристике сети. Качественная характеристика может уже налагаться дополнительной бизнес-логикой к ядру блокчейн сети базируемом на PoU алгоритме.

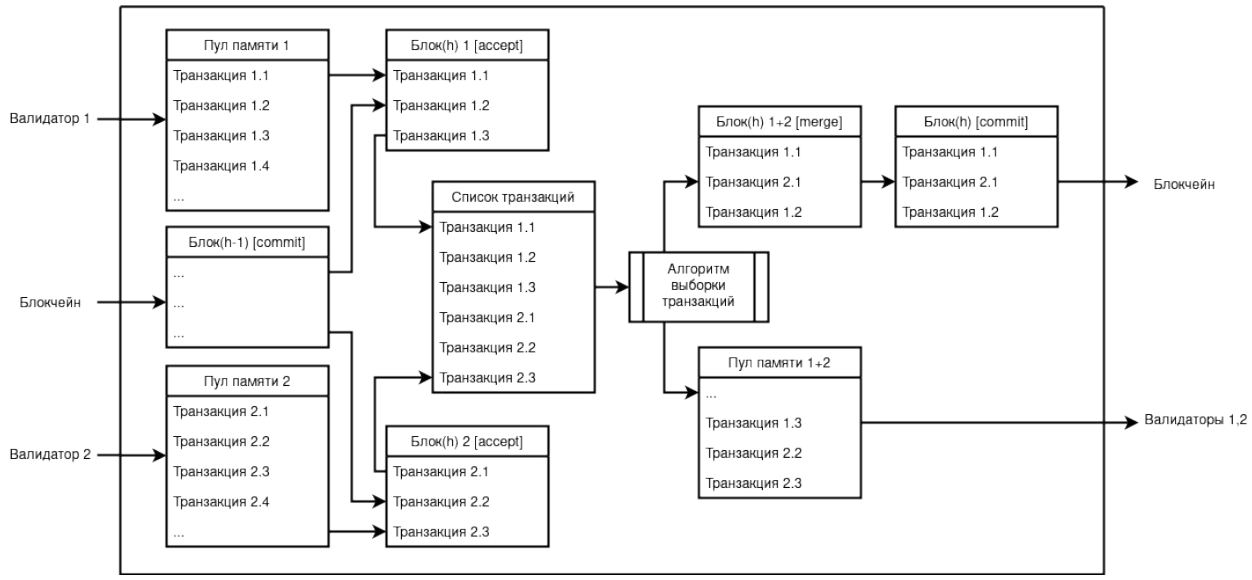

Рис. 1. Общая схема консенсуса Proof-of-Union при двух валидаторах,
где h - высота блока, (1+2) - объединение двух позиций

С экономической точки зрения такой алгоритм консенсуса приведёт к невозможности вознаграждать валидаторов внутренним механизмом бизнес-логики блокчейн технологии, т.к. каждый отдельный валидатор будет генерировать блок исходя из всех других валидаторов сети. Единственным способом поощрения валидации блока останется поощрение всех валидаторов непосредственно, без учёта их различий.

4. Безопасность

У алгоритма PoU существует ряд неоднозначных явлений, присущих в большей части только ему. Так например, одним из важнейших моментов при анализе безопасности является невозможность доказательства «правильной» цепочки. Иными словами, PoU базируется на основе динамической генерации блоков, когда не существует объективной характеристики определить «вернейшую» цепочку из всего множества уже существующих, т.к. сам алгоритм основывается на концепции их слияния, а не выбора одного возможного элемента. Таким образом, новый подключающийся валидатор должен довериться одной ветви развития блокчейна из предложенного ограниченного множества.

Пример программного кода [8] процедуры объединения блоков:

```
import (
	"bytes"
)
func (chain *ChainT) Merge(height Height, txs []Transaction) bool {
	chain.mtx.Lock()
	defer chain.mtx.Unlock()
	var (
		lastBlock = chain.Block(height)
		resultTXs []Transaction
	)
	if chain.Height() != height {
		return false
	}
	resultTXs = append(resultTXs, lastBlock.Transactions()...)
	for _, tx := range txs {
		if !tx.IsValid() {
			return false
		}
		if chain.TX(tx.Hash()) != nil {
			continue
		}
		resultTXs = append(resultTXs, tx)
	}
	if len(resultTXs) == TXsSize {
		return false
	}
	sort.SliceStable(resultTXs, func(i, j int) bool {
		return bytes.Compare(resultTXs[i].Hash(), resultTXs[j].Hash()) < 0
	})
	appendTXs := resultTXs[:TXsSize]
	deleteTXs := resultTXs[TXsSize:]
	chain.updateBlock(height, NewBlock(lastBlock.PrevHash(), appendTXs), deleteTXs)
	return true
}
```

Продолжая, можно отметить, что функция commit представляет собой необходимую меру для решения проблематики функции merge, когда таковая в чистом виде может приводить к разветвлениям сети, при условии, что сам новый merge-блок появляется в момент нового вызова accept частью сети. Такой случай крайне легко было бы воспроизводить злоумышленниками (без функции commit), постоянно разветвляющими сеть.

Далее, если у сети не существует дополнительной бизнес-логики, а сам алгоритм базируется только на реализации PoU в чистом виде, то представляется возможной атака заполнения блока транзакциями наиболее релевантными по алгоритму выборки транзакций. Это в свою очередь может привести к спам-атаке, которая, как итог, не позволит обычным клиентам сохранять свою информацию в блокчейне.

5. Заключение

В данной работе был выявлен новый класс формирования и установки консенсуса основанный на доказательстве объединения, представленный как альтернатива конкурентной и последовательной моделям. Плюсом алгоритма стала простота его реализации и скорость генерации блоков. Помимо плюсов сам алгоритм представляет и неоднозначность безопасности, т.к. базируется на кардинально противоположной позиции конкурентной и последовательной моделям консенсуса, и требует объединения разных взглядов на выдвинутый валидаторами блок.

Список литературы

1. Накамото, С. Биткойн: система цифровой пиринговой наличности [Электронный ресурс]. — Режим доступа: https://bitcoin.org/files/bitcoin-paper/bitcoin_ru.pdf (дата обращения: 19.12.2020).
2. King, S., Nadal, S. PPCoin: Peer-to-Peer Crypto-Currency with Proof-of-Stake [Электронный ресурс]. — Режим доступа: https://web.archive.org/web/20171211072318/https://peercoin.net/assets/paper/peercoin-paper.pdf (дата обращения: 29.01.2022).
3. PROOF OF STAKE - это скам [Электронный ресурс]. — Режим доступа: https://habr.com/ru/post/600113 (дата обращения: 29.01.2022).
4. Шнайер, Б. Секреты и ложь. Безопасность данных в цифровом мире / Б. Шнайер. — СпБ.: Питер, 2003. - 368 с.
5. Взгляд изнутри: цены на видеокарты и чего ждать от рынка завтра? [Электронный ресурс]. — Режим доступа: https://habr.com/ru/company/asus/blog/571400 (дата обращения: 29.01.2022).
6. Самохин, В., Самохин, Д., Бабкин, Е., Петров, И. Актуальность вопросов энергосбережения на майнинг-фермах [Электронный ресурс]. — Режим доступа: https://cyberleninka.ru/article/n/aktualnost-voprosov-energosberezheniya-na-mayning-fermah (дата обращения: 29.01.2022).
7. Герасимов, И., Чижов, И. Алгоритм консенсуса платформы Tendermint и механизм Proof Of Lock Change [Электронный ресурс]. — Режим доступа: https://cyberleninka.ru/article/n/algoritm-konsensusa-platformy-tendermint-i-mehanizm-proof-of-lock-change (дата обращения: 29.01.2022).
8. Донован, А., Керниган, Б. Язык программирования Go / А.А. Донован, Б.У. Керниган. — М.: ООО «И.Д. Вильямс», 2018. - 432 с.