Pastry
и другие алгоритмы одноранговых сетей

TRINITY LAB

2021

Pastry

Разработчики: Microsoft Labs Research, Rice University, Purdue University, University of Washington, 2001

Назначение: основополагающий алгоритм для создания одноранговых Интернет-приложений распределенного хранения файлов (поиска, обмена), организации группового общения

Узел описывается состоянием, состоящим из:

- Таблица маршрутизации размерностью $\lceil log_{2^b} N \rceil$ строк и 2^b-1 столбцов – список ID узлов, у которых n первых цифр их идентификатора ID совпадают с ID текущего узла
- «Соседи» - список ID узлов из $|M| = 2^b$ элементов, ближайших по количеству хопов к текущему узлу
- «Листья» - список ID узлов из $|L| = 2^b$ элементов из двух частей: в первой части содержатся те ID, чьи значения ID численно являются ближайшими превосходящими ID текущего узла, вторая часть списка – те ID, которые имеют численно ближайшие меньшие значения к ID текущего узла

ID узла 10233102, уникальный в диапазоне [0; 2^{128}-1]

Листья

Меньше		Больше	
10233033	10233021	10233120	10233122
10233001	10233000	10233230	10233232

Таблица маршрутизации

-0-2212102	1	-2-2301203	-3-1203203
0	1-1-301233	1-2-230203	1-3-021022
10-0-31203	10-1-32102	2	10-3-23302
102-0-0230	102-1-1302	102-2-2302	3
1023-0-322	1023-1-000	1023-2-121	3
10233-0-01	1	10233-2-32	
0		102331-2-0	
		2	

Соседи

10233033	10233021	10233120	10233122
10233001	10233000	10233230	10233232

Pastry

Разработчики: Microsoft Labs Research, Rice University, Purdue University, University of Washington, 2001

Назначение: основополагающий алгоритм для создания одноранговых Интернет-приложений распределенного хранения файлов (поиска, обмена), организации группового общения

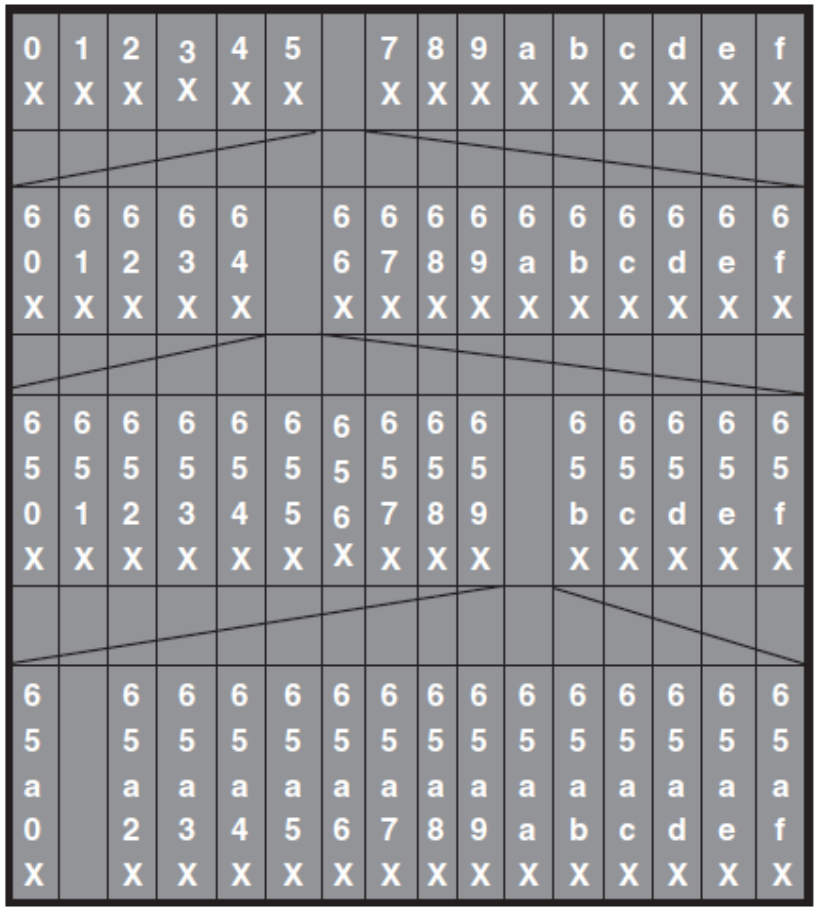
Узел описывается состоянием, состоящим из:

- Таблица маршрутизации размерностью $\lceil log_{2^b} N \rceil$ строк и 2^b-1 столбцов – список ID узлов, у которых n первых цифр их идентификатора ID совпадают с ID текущего узла
- «Соседи» - список ID узлов из $|M| = 2^b$ элементов, ближайших по количеству хопов к текущему узлу
- «Листья» - список ID узлов из $|L| = 2^b$ элементов из двух частей: в первой части содержатся те ID, чьи значения ID численно являются ближайшими превосходящими ID текущего узла, вторая часть списка – те ID, которые имеют численно ближайшие меньшие значения к ID текущего узла

ID узла 65a1x

Таблица маршрутизации

ID приведены в 16-й системе счисления, b = 4, “x” обозначает суффикс арбитрирования



Pastry

Алгоритм поиска

Сложность верхняя асимптотическая граница: $O\lceil log_{2^b} N \rceil$

Цель: узлу N требуется найти в P2P сети узел E с ID = K и переслать этому узлу сообщение

1. Если узел с ID = K попадает в диапазон ID в списке Листья, то передать сообщение узлс с ID численно ближайшим к искомому
2. Иначе, определить общий префикс (N, K)
3. Поиск в таблице маршрутизации такого узла E, у которого ID численно имеет прификс (E, K) > префикс (N, K), отправить сообщение узлу E
4. Если поиск не удался, то повторный поиск по объединенному множеству из Листья, Соседи, Таблице маршрутизации с целью найти такой узел E, у которого самый длинный общий префикс (E, K), отправить сообщение узлу E

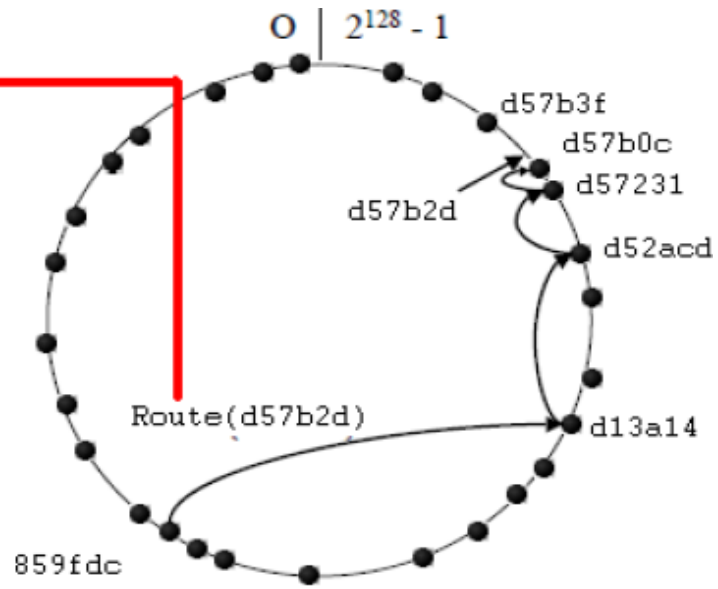
Пример

Узел с **ID=859fdc** ищет узел с ID=d57b2d

Последовательность обращения к узлам при поиске:

- узел с ID = d13a14
- узел с ID = d52acd
- узел с ID = d57231
- узел с ID = d57b0c
- узел с ID = d57b2d **найден**

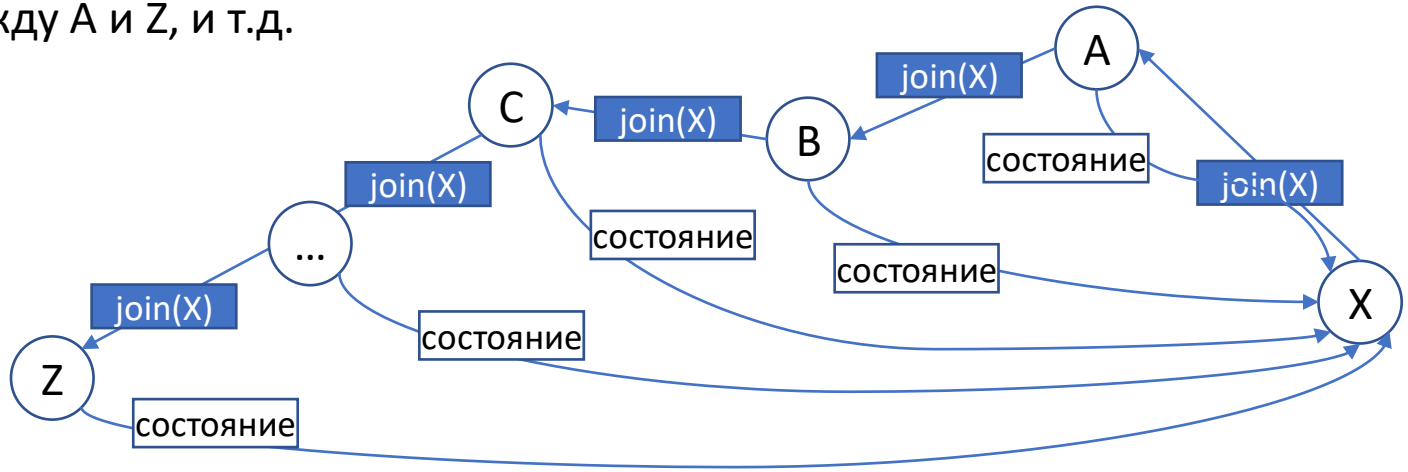
0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8
0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8
5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8
5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9
0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x



Pastry

Алгоритм добавления узла

1. Новый узел с ID=X подключается к сети и должен знать существующий узел A
2. Новый узел X шлет узлу A сообщение «*join(X)*», внутри которого указан ID нового узла
3. Узел A пересылает сообщение *join(X)* узлу Z, у которого ID численно ближайшее к X
4. Все узлы на маршруте переходов между A…Z отправляют в X свое состояние (списки, таблицу маршрутизации)
5. Узел X на основании полученной информации формирует свое состояние – вписывает ID узлов в свои списки Листья, Соседи и Таблицу маршрутизации:
   - Если узел A ближайший по количеству хопов (задержкам) к X, то список Соседей A используется для задания списка Соседей в X
   - Если ID узла Z численно наиболее близко к ID узла A, то Листья Z берутся за основу для Листьев узла X
   - Заполнение строк таблицы маршрутизации – в первую строку записывается ID узла B, являющийся первым узлом на маршруте переходов из A в Z. ID узлов X и B совпадают по первым цифрам. Во вторую строку записывается ID узла C – это второй узел между A и Z, и т.д.



Pastry

Алгоритм добавления узла

1. Новый узел с ID=X подключается к сети и должен знать существующий узел A
2. Новый узел X шлет узлу A сообщение «*join(X)*», внутри которого указан ID нового узла
3. Узел A пересылает сообщение *join(X)* узлу Z, у которого ID численно ближайшее к X
4. Все узлы на маршруте переходов между A и Z отправляют в X свое состояние (списки, таблицу маршрутизации)
5. Узел X на основании полученной информации формирует свое состояние – вписывает ID узлов в свои списки Листья, Соседи и Таблицу маршрутизации:
   - Если узел A ближайший по количеству хопов (задержкам) к X, то список Соседей A используется для задания списка Соседей в X
   - Если ID узла Z численно наиболее близко к ID узла A, то Листья Z берутся за основу для Листьев узла X
   - Заполнение строк таблицы маршрутизации – в первую строку записывается ID узла B, являющийся первым узлом на маршруте переходов из A в Z. ID узлов X и B совпадают по первым цифрам. Во вторую строку записывается ID узла C – это второй узел между A и Z, и т.д.
6. Узел X при необходимости может запросить дополнительно состояния у других узлов. X сравнит списки соседей и таблицы маршрутизации с состояниями, полученными от A..Z, выберет ближайшие по хопам ID и запишет в состояние X.
7. Узел X сообщает о своем существовании и пересылает свое состояние всем, кто у него в списках Листья, Соседи, Таблица маршрутизации с кем у него еще не было общения

- Количество сообщений, требующееся при добавления нового узла: $O\left(log_{2^b} N\right)$
- Сообщения идентифицируются **временной меткой с целью различать сообщения от 2 и более новых узлов**


Chord VS Pastry

Chord vs Pastry: сравнение общих принципов

Параметры	Chord	Pastry	Kademlia
Геометрия сети	Одноранговое кольцо из узлов, количество взаимодействующих узлов ограниченно логарифмическим числом	Одноранговая сеть типа Plaxton, префиксная маршрутизация	XOR метрика для оценки дистанции между точками в пространстве ключей (ID) узлов в сети
Широковещание	Есть, публикация *DHT-based lightweight broadcast algorithms in large-scale computing infrastructures*	Есть, требуется граф сети подобный как в системе Scribe, публикация *SplitStream: High-Bandwidth Multicast in Cooperative Environments*	Есть, публикация *Kadcast: A Structured Approach to Broadcast in Blockchain Networks*
Алгоритм поиска	Сообщение передается узлу, ближайшему к искомому ID в кольце, без **учета задержек**	Узел хранит списки листья, соседи и таблицу маршрутизации. Поиск по префиксу ID узла по ближайшему окружению узлов (учитываются хопы при заполнении списка соседей и таблицы маршрутизации)	Поиск по ключу (ID) с использованием логики XOR. Узел хранит таблицу маршрутизации для каждого бита собственного ID. Узел знает ближайшие узлы и часть остальных. При поиске рекурсивно опрашивается несколько узлов, у которых `1` после выполнения XOR текущего ID и искомого. Упрощенная аналогия – поиск по бинарному дереву из `1` и `0`
Сложность поиска, добавл.и удаления узла	$O(log_2 N)$, где N – количество узлов	$O(log_{2^b} N)$, где N-количество узлов, b – количество бит ID	$O(log_B N) + c$, где N-количество узлов, $B=2^b$, b – количество бит ID, c – константа малой величины
	$(log_2 N)^2$	$log_{2^b} N$	$O(log_B N) + c$
Актуализация таблиц маршрутизации	Периодический опрос для обнаружения новых узлов, обновление ID предшественника (predecessor), последователя (successor) и таблицы связей (fingers)	Ближайшие узлы по количеству хопов Соседи периодически обмениваются уведомлениями о работоспособности, часть узлов из окружения известны по списку Листья	Периодический опрос узлов по таблице маршрутизации для уведомления о работоспособности
Требование к сети	Новый узел должен знать любой узел в сети	Новый узел должен знать ближайший по хопам узел	Новый узел должен знать любой узел в сети

Chord vs Pastry: сравнение общих принципов

Параметры	Low Latency Chord! опубликован 2006	Pastry	Kademlia
Геометрия сети	Одноранговое кольцо из узлов, количество взаимодействующих узлов ограниченно логарифмическим числом	Одноранговая сеть типа Plaxton, префиксная маршрутизация	XOR метрика для оценки дистанции между точками в пространстве ключей (ID) узлов в сети
Широковещание	Есть, публикация *DHT-based lightweight broadcast algorithms in large-scale computing infrastructures*	Есть, требуется граф сети подобный как в системе Scribe, публикация *SplitStream: High-Bandwidth Multicast in Cooperative Environments*	Есть, публикация *Kadcast: A Structured Approach to Broadcast in Blockchain Networks*
Алгоритм поиска	Сообщение передается узлу, ближайшему к искомому ID в кольце, **с учетом задержек**	Узел хранит списки листья, соседи и таблицу маршрутизации. Поиск по префиксу ID узла по ближайшему окружению узлов (учитываются хопы при заполнении списка соседей и таблицы маршрутизации)	Поиск по ключу (ID) с использованием логики XOR. Узел хранит таблицу маршрутизации для каждого бита собственного ID. Узел знает ближайшие узлы и часть остальных. При поиске рекурсивно опрашивается несколько узлов, у которых `1` после выполнения XOR текущего ID и искомого. Упрощенная аналогия – поиск по бинарному дереву из `1` и `0`
Сложность поиска, добавл.и удаления узла	$O(log_2 N)$, где N – количество узлов	$O(log_{2^b} N)$, где N-количество узлов, b – количество бит ID	$O(log_B N) + c$, где N-количество узлов, $B=2^b$, b – количество бит ID, c – константа малой величины
	$(log_2 N)^2$	$log_{2^b} N$	$O(log_B N) + c$
Актуализация таблиц маршрутизации	Периодический опрос для обнаружения новых узлов, обновление ID предшественника (predecessor), последователя (successor) и таблицы связей (fingers)	Ближайшие узлы по количеству хопов Соседи периодически обмениваются уведомлениями о работоспособности, часть узлов из окружения известны по списку Листья	Периодический опрос узлов по таблице маршрутизации для уведомления о работоспособности
Требование к сети	Новый узел должен знать любой узел в сети	Новый узел должен знать ближайший по хопам узел	Новый узел должен знать любой узел в сети

Другие алгоритмы организации одноранговых сетей

Краткое описание алгоритмов

	Tapestry	P-Grid	CAN
Geometry	Uniformly at random from a large identifier space (typically 160bit with a globally defined radix)	Binary tree	*d*-dimensional Cartesian coordinate space
Routing algorithm	Matching Key and prefix in Node-ID	Binary tree search and prefix matching	Forward the search message to neighbour node closer to the destination
Routing performance	$\log_\beta N$, where N is the size of the identifier space, and β is the radix used	$O(\log N)$, where N is number of data items in the overlay)	$(d/4)(N^{1/d})$, N is number of nodes, and d is dimension
Join/Leave performance		$O(\log N)$	$2d$
Routing table maintenance			Through periodic update messages. Controlled flooding is used in case a node loses multiple entries simultaneously
Bootstrapping	A new node knows a nearby Tapestry node.	Know at least one node.	Know at least one node. May get this node through DNS

	Ulysses	Cycloid	Kelips
Geometry	Butterfly topology with shortcut links	Cube-Connected-Cycles graph	No geometry to the address space, each node knows about other nodes
Routing algorithm	For a Ulysses network with k levels and n nodes. to search a key α, in each step, the query gets locked in one additional dimension, after the first k steps the query reaches a node (Q, l) such that α lies within the zone Q in all the k dimensions.	Uses the outside leaf sets of the finger table to find closest cubical neighbour or the closest cyclical neighbour. Inside leaf sets are used to find appropriate node.	Routing table includes nodes in the affinity group, nodes in all the foreign groups, and filetuples. Querying node maps file name to affinity group and sends lookup request to topologically closest node in affinity group.
Routing performance	log_2log_2n	$O(d)$, where d is network dimension, n is number of nodes, $n = d.2^d$	$O(1)$
Join/Leave performance	Find corresponding node with a randomly generated key, then split the zone. log_2log_2n	$O(d)$	No structure or invariant. Join is complete with node participating in gossip stream. Node leaving are updated through the gossip system.
Bootstrapping	A joining node needs to know an existing node in Ulysses network.	No specific bootstrapping mechanism discussed .	Bootstrapping node allowing joining node to join gossip stream.

Краткое описание алгоритмов

	OneHop	EpiChord	D1HT
Geometry	128-bit random node ID ordered in a ring modulo 2^{128}.	Circular node ID space, logarithmic degree mesh	Hashing keys and peers into an ID space *[0, N]*, $N >> n$
Routing algorithm	every node maintains a full routing table	Use multiple parallel queries to locate node that owns the key	every node maintains a full routing table
Routing performance	*O(1)*	*O(1)* under lookup intensive workloads, and *O(log N)* in the worst case	*O(1)*
Join/Leave performance			$\lceil log_2 n \rceil$, where n is number of nodes
Routing table maintenance	Nodes run stabilisation routine sending keep-alive messages to successor and predecessor.	Routing entries are flushed whenever lifetime expires, or the corresponding node does not respond to queries. If slice entries are insufficient, lookup to midpoint of slice, add routing entries from the lookup response.	Propagate join/leave messages with TTL values. Use a *Quarantine* mechanism to handle highly dynamic nodes
Bootstrapping	A new node knows an existing OneHop node.	Knows at least one existing node	The new node must know an existing D1HT peer already in the system

Библиографический список:

1. Rowstron, A. Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems / A. Rowstron, P. Druschel // *Proceedings of IFIP/ACM International Conference on Distributed Systems Platforms and Open Distributed Processing.* – 2001. - pp. 329-350.

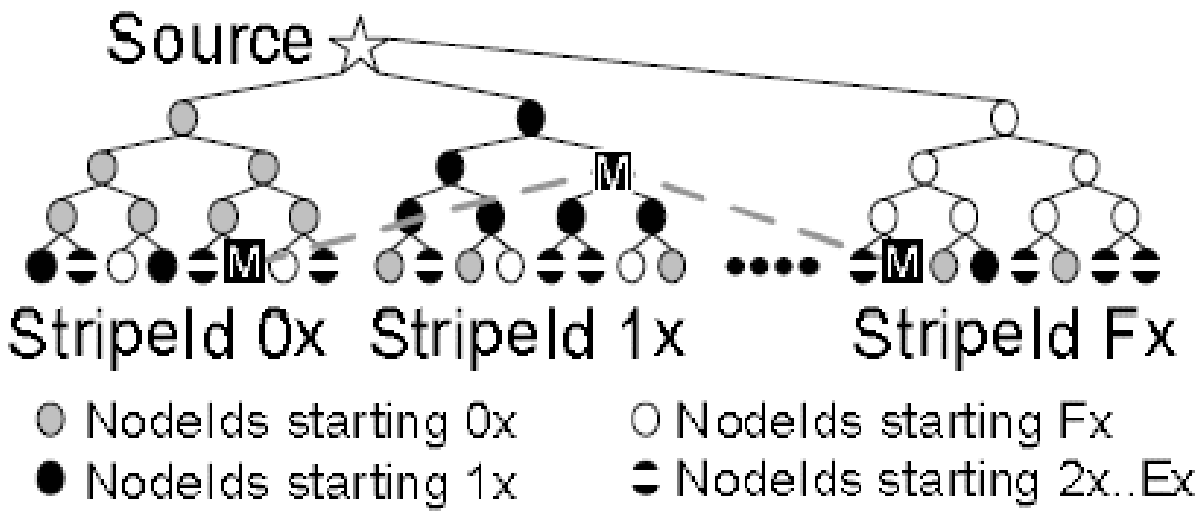
2. Dhara, K. Overview of Structured Peer-to-Peer Overlay Algorithms / K. Dhara, Y. Guo, M. Kolberg, X. Wu // *Handbook of Peer-to-Peer Networking.* – 2010. - pp.223-256.

3. Tarkoma, S. Publish/Subscribe Systemc Design and Principles // *John Wiley & Sons Ltd*. – 2012. - 346 p.

Алгоритмы широковещательной рассылки сообщений

1. Split Stream алгоритм для Pastry сети с использованием коммуникационной системы Scribe

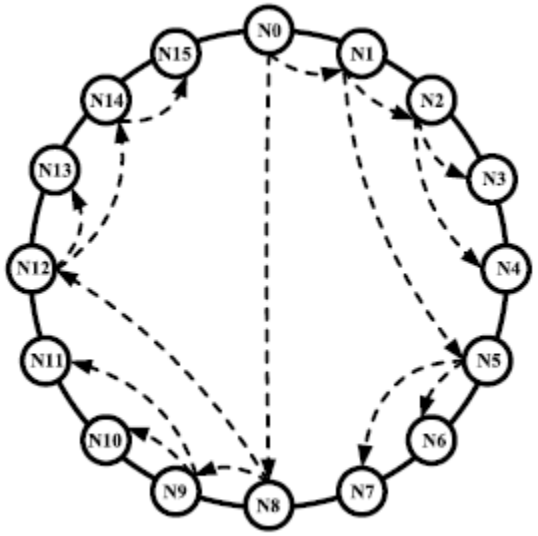
Castro, M. SplitStream: High-Bandwidth Multicast in Cooperative Environments / M. Castro, P. Druschel, A.-M. Kermarrec, A. Nandi, A. Rowstron, A.Singh // *ACM SIGOPS Operating Systems Review*. – 2004. – pp.298-313



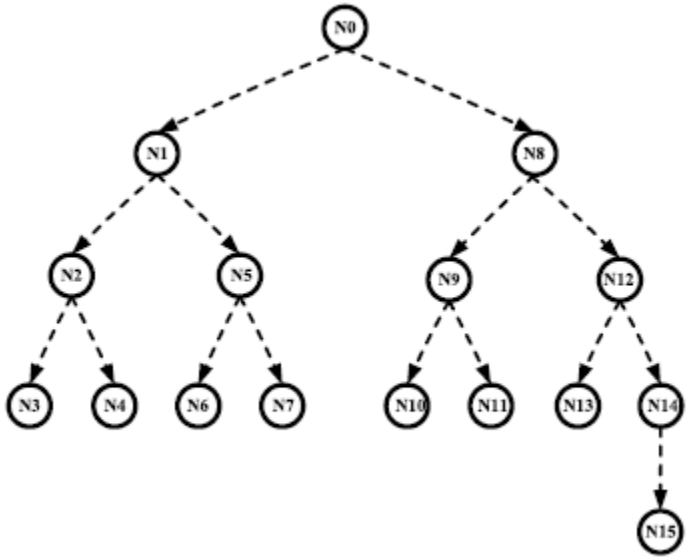
Алгоритмы широковещательной рассылки сообщений

2. Partition-based broadcast алгоритм для Chord сети

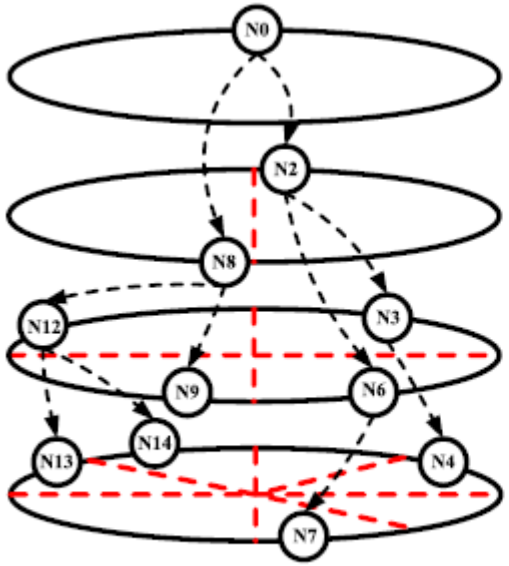
Huang, K. DHT-based lightweight broadcast algorithms in large-scale computing infrastractures / K. Huang, D. Zhang // Future Generation Computer Systems. – 2010. – pp.291-303



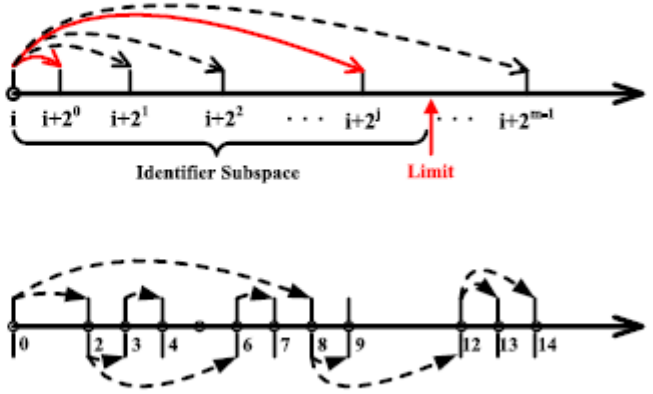
(a) Finger routing paths rooted at N_0.



(b) Balanced DBT rooted at N_0.



(a) Hierarchical binary partition of the node identifier space from N_0.



(b) Alternately covering all the destination nodes to construct balanced DBT rooted at N_0.

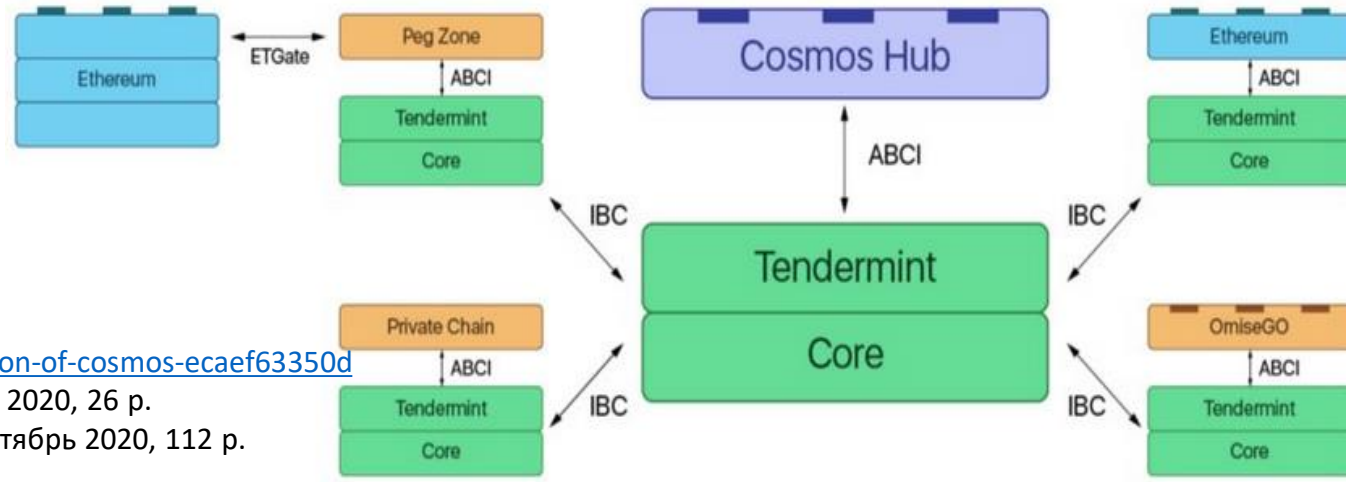
Ethereum, Polkadot

Ethereum	Polkadot
• **Kademilia p2p сеть** • **Geth** - собственной разработки клиент передачи сообщений между узлами	• Сеть строится на Validators, Nominators, Collators. • Validators генерируют блоки для работы Polkadot сети, выполняют проверку состояний, выдаваемых Collators • Nominators служат механизмом, определяющим будет ли Validator попадать в множество активных валидаторов, хранят стейк валидаторов • Collators собирают транзакции блокчейнов (*parallel chain, parachain)* и генерируют состояние доказательств перехода для Validators, отправляют и получают сообщения от разных блокчейнов через XCMP • **Kademilia p2p сеть** • **Gossiping** – широковещательное распространение инфы в сети Polkadot, объединяющей разнородные блокчейны (*parallel chain, parachain).* Узлы поддерживают «пул распространения» сообщений. Отправка происходит с отсрочкой: 1) узел помещает сообщение в пул до тех пор, пока сообщение не будет помечено как с истекшим сроком. 2) узел рассылает сообщение всем узлам, когда получит на это разрешение. Каждое сообщение связано с темой. Темы используются для группировки сообщений. Для узла определяется одноранговый узел, который напрямую связан с ним ребром в графе gossip. Состав узлов может меняться со временем. Когда новый узел подключается к сети, все сообщения пула сообщений пересылаются ему. • **XCMP** – (под)протокол передачи сообщений между разными *parachain* с аутентификацией. Понятный пример: есть сервер онлайн игры, а есть кошелёк на другом сервере с другой технологией. Задача XCMP безопасным образом связать эти два сервера, работающих в виде отличающихся chain • XCMP похож на TCP, назначение: быстрая доставка по сети Polkadot; сохранение порядка доставки сообщений из одного parachain в другой; гарантирует, что поступающие сообщения действительно были отправлены в истории chain. • Передача сообщений внутри chain задача самого chain, не входит в задачи XCMP • **Общие пространные концепты с техническими элементами, детали не раскрываются** Источники: • https://polkaworld.medium.com/whats-the-significance-of-xcmp-to-polkadot-ba9c4a283fd1 • https://research.web3.foundation/en/latest/polkadot/networking.html • Статья, Overview of Polkadot and its Design Considerations, май 2020, 41 p. • Polkadot на Youtube, https://youtu.be/xBfC6uTjvbM

Cosmos

Cosmos

- Cosmos – сеть блокчейнов
- В основе Cosmos лежит **Tendermint** технология, обеспечивающая сетевое взаимодействие и консенсус в сети
- Поверх Tendermint используется протокол **Application Blockchain protocol (ABCI)**, предоставляющий унифицированные сокеты и интерфейсы для подключения пользовательских приложений, клиент-серверная архитектура, есть реализация на Go, на github есть проекты на JavaScript, C++, Java
- Пользователь могут написать свое приложение со своей логикой, создав свой блокчейн поверх Tendermint+ABCI
- Для соединения разных блокчейнов и передачи валюты разработан протокол **Inter-Blockchain Communication Protocol (IBC)** с документацией на 112 страниц. Деталь: 2/3 валидаторов исходного блокчейна должны подтвердить токены, передающиеся в другой блокчейн
- Cosmos состоит из 2х классов: концентраторы (hub) и зоны
- **Зона** – конкретный блокчейн
- **Hub** – промежуточный блокчейн для организации связи между зонами
- Для подключения к Cosmos блокчейнов, построенных не на Tendermint, требуется использовать:
- либо алгоритм быстро завершающегося консенсуса *fast finality* в блокчейне
- либо использовать **Peg-Zone** – прокси блокчейн, который отслеживает состояние другого блокчейна
- Команда Cosmos разрабатывают Peg-Zone для Ethereum, по состоянию на 2018
- Готовят к выпуску среду разработки блокчейнов – Cosmos-SDK, по состоянию на 2018



Источники:

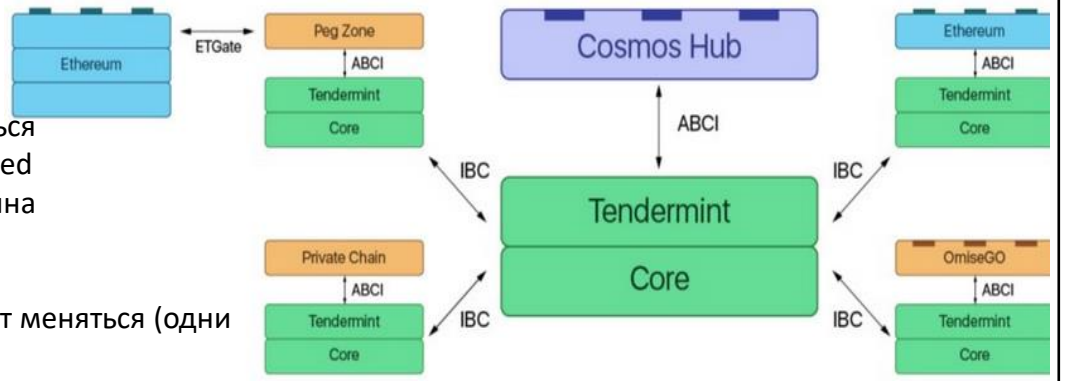
- Cosmos концепт, https://blog.cosmos.network/understanding-the-value-proposition-of-cosmos-ecaef63350d
- Статья, Goes, C. The Interblockchain Communication Protocol: An Overview, июнь 2020, 26 p.
- Техническая спецификация IBC, The Interblockchain Communication Protocol, октябрь 2020, 112 p.
- ABCI концепт, https://docs.tendermint.com/master/spec/abci/

Cosmos: Tendermint

Cosmos

Организация сети **Tendermint** – **еще более общие пространные концепты с техническими элементами, нераскрывающие алгоритмическую часть**

- Сеть строится на доверенных узлах, узел может иметь несколько IP адресов, как формируется ID узла – не ясно
- Сложная процедура аутентификации нового узла (Diffie-Helman алгоритм обмена ключами), зашифрованные TCP соединения
- Новый узел подключается к **seed**, который сообщает список существующих узлов, используя протокол **Peer exchange (PEX)**
- **PEX** – создан для BitTorrent (uTorrent и др) для одноранговового обмена файлами. Узел скачивает с сервера (tracker) список ID других узлов с указанием какие файлы на них хранятся с хешами. Дальше работает логика BitTorren – новый узел скачивает файл с соседнего узла. Соседний узел выбирается с согласованием с сервером, чтоб сбалансировать нагрузку (раняя версия BitTorrent)
- Новый узел должен знать ID блокчейна, к которой принадлежит, от доверенного узла
- Новый узел должен знать высоту цепочки блоков и хэш от доверенного узла
- Новый узел запрашивает исходные данные для одноранговых узлов для своей цепочки, и запускает протоколы Tendermint с теми, к которым по логике блокчейна ему требуется подключиться
- Если новый узел не может подключиться к узлам своего блокчейна, то узел возвращается к к узлу seed
- В сети имеются сторожевые доверенные узлы и узлы валидаторы. Узлы валидаторы блоков блокчейна не общаются с сетью напрямую, а только через сторожевые узлы.
- Валидаторы могут напрямую общаться с другими валидаторами напрямую через VPN
- Сторожевые узлы образуют множество надежных узлов. Предполагается, что сторожевые узлы могут меняться (одни выключаются, другие включаются).



- Сторожевы узла всегда должны ожидать наличия прямых входящих подключений от узла валидатора и его резервных копий
- Они не сообщают адрес узла валидатора в PEX
- Сторожевые узлы, которые доверяют друг другу, могут пожелать поддерживать постоянные соединения друг с другом через VPN, но сообщать друг о друге только в PEX.
- Известно, что используется мультиплексное соединение для поддержания независимых потоков с различными гарантиями качества обслуживания поверх одного TCP-соединения. Каждый поток представлен в виде абстракции «канал», и каждый канал имеет глобально уникальный байтовый идентификатор. Каждый канал также имеет относительный приоритет (отправка сообщения низкого приоритета не прерывается сообщением высокого приоритета). Идентификатор байта и относительные приоритеты каждого канала настраиваются при инициализации соединения.
- **В описании Tendermint на офф. Сайте есть пометки TODO:** https://docs.tendermint.com/master/spec/p2p/connection.html

Источники:

- Tendermint концепт https://docs.tendermint.com/master/spec/p2p/peer.html#

LLChord: Low Latency Chord

Routing table:

- **Fingers** – расширенный список, состоящий из ID узлов, перечисленных по ходу часовой стрелке (классический Chord) и в обратном направлении. Размер Fingers = обычный размер таблицы Fingers x2
- **Задержки** – список задержек от текущего узла к узлам из Fingers++

Спасибо за внимание!

Обсуждение