III TRINITY LAB

Review of p2p Chord project

Задача: разработка Хорды с широковещанием для мерцающей сети 2000+ узлов

- Отсутствуют материалы по Хорде
- Выполнен научно-технический поиск:
 - Обзорные материалы про р2р алгоритмов 2 книги, научные публикации:
 - 1. Tarkoma S. Publish/Subscribe Systems. Design and Principles // John Wiley & Sons Ltd. 2012. 346 p.
 - 2. Korzun D., Gurtov A. Structured Peer-to-peer Systems //Springer New York. 2013. 375 p.

3. Dhara K., Guo Y., Kolberg M., Wu X. Overview of Structured Peer-to-Peer Overlay Algorithms //Handbook of Peer-to-Peer Networking.

2010. pp.223-256.

4. Другие

Chord-based design	Parameters being improved	Means of improvement	Key differences with CR-Chor
EpiChord [19]	Lookup latency, number of hops, availability under chum.	Increased finger table size, parallel lookups.	Maintenance of many neighbors. Iterative lookups
F-Chord [4]	Number of hops, availability.	Modified finger selection procedure.	No independent variation of routing state. No multi-path routing.
Freebie fingers [18]	Number of hops.	Additional O(log N) fingers without causing additional traffic.	No independent variation of routing state. No multi-path routing.
Bidirectional latency-sensitive Chord [32], LLChord [15], topology matching [8]	Lookup latency. ([32] also reduces the number of hops via bidirectional routing similarly to [23].)	Proximity neighbor selection (PNS).	No independent variation of routing table sizes. No multi-path routing.
Fiat's et al. S-Chord [9]	Resilience to Byzantine join attack.	Increased finger table size, more messages sent during lookup and join procedures.	No multi-path routing.
Zig-zag routing [7,27]	Resilience to sybil attack.	Trust profile for individual nodes, altered routing algorithm.	Iterative lookups. No independent variation of routing table sizes. No multi-path routing.
Cyclone [1]	Resilience to data-forwarding attack.	Increasing the number of disjoint paths. Multi-path routing.	No independent variation of routing table sizes.
Mesaros's et al. S-Chord [23]	Lookup performance in number of hops.	Bidirectional routing.	No independent variation of routing table sizes. No multi-path routing.
RChord [35]	Lookup performance in number of hops. Resilience to routing attacks.	Bidirectional routing.	No independent variation of routing table sizes. No multi-path routing.
P-Chord, PL-Chord [17]	Lookup availability depending on node lifetime, number of hops.	Bidirectional routing, supernodes.	No independent variation of routing table sizes. Doesn't consider malicious nodes.
H-Chord [5]	Number of hops.	Neighbor-of-neighbor routing.	No independent variation of routing table sizes. No resistance to malicious nodes and churn. No multi-path routing.
Chord enhancement by [2]	Number of hops. Lookup latency.	Neighbor-of-neighbor routing. Proximity route selection.	No independent variation of routing table sizes. No resistance to malicious nodes and churn. No multi-path routing.

Задача: разработка Хорды с широковещанием для мерцающей сети 2000+ узлов

- Отсутствуют материалы по Хорде
- Выполнен научно-технический поиск:
 - Обзорные материалы про р2р алгоритмов 2 книги, научные публикации:
 - 1. Tarkoma S. Publish/Subscribe Systems. Design and Principles // John Wiley & Sons Ltd. 2012. 346 p.
 - 2. Korzun D., Gurtov A. Structured Peer-to-peer Systems //Springer New York. 2013. 375 p.
 - 3. Dhara K., Guo Y., Kolberg M., Wu X. Overview of Structured Peer-to-Peer Overlay Algorithms //Handbook of Peer-to-Peer Networking.

2010. pp.223-256.

4. Другие

Chord-based			Means of improvement reased finger table size,	Key differences with CR-Chord Maintenance of many
EpiChord [19]	i.	Chord	Pastry	Kademlia
F-Chord [4] Freebie fingers	Geometry	Circular Node-ID space, logarithmic	Plaxton-style mesh network, prefix	XOR metric for dis- tance between points
Bidirectional la Chord [32], topology m	Routing algorithm	degree mesh Search query for- warded to "closer" node	matching Key and prefix of Node-ID	in the key space. (XOR) Matching Key and Node-ID based routing done
Fiat's et al. S-C Zig-zag routing Cyclone [1]	Routing performance	O(logN), where N is the number of peers	$O(log_BN)$, where N is number of peers, and $B=2b$, b is number of bits of NodeID	parrallely. O(log _B N)+c, Where N is number of peers, B=2b, b is number of bits of Node-ID, and c is a
Mesaros's et al	Join/Leave performance	$(logN)^2$	log_BN	small constant $log_B N + c$
P-Chord, PL-C	Routing table maintenance	Periodic stabiliza- tion protocol at nodes to learn	Neighboring nodes periodically ex- change keep-alive	Failure of peers will not cause network- wide failure. Repli-
H-Chord [5] Chord enhance	минини	about newly joined nodes, update suc- cessor and prede- cessor, and fix fin-	messages. The leaf sets of nodes with adjacent Node-Id overlap.	cate data across mul- tiple peers.
	Bootstrap- ping	ger tables. A new node knows an existing Chord	A new node knows a nearby Pastry	A new node knows an existing Kadem-

Задача: разработка Хорды с широковещанием для мерцающей сети 2000+ узлов

- Отсутствуют материалы по Хорде
- Выполнен научно-технический поиск:
 - Обзорные материалы про р2р алгоритмов 2 книги, научные публикации:
 - 1. Tarkoma S. Publish/Subscribe Systems. Design and Principles // John Wiley & Sons Ltd. 2012. 346 p.
 - 2. Korzun D., Gurtov A. Structured Peer-to-peer Systems //Springer New York. 2013. 375 p.
 - 3. Dhara K., Guo Y., Kolberg M., Wu X. Overview of Structured Peer-to-Peer Overlay Algorithms //Handbook of Peer-to-Peer Networking.
 - 2010. pp.223-256.
 - 4. Другие
- Найдено улучшение Хорды для уменьшения задержек

JiangYi, Jinyuan Y. A Low Latency Chord Routing Algo for DHT // 1st Int. Symp. on Pervasive Computing and App. 2006. pp. 825-830

		xisting extensions to basic Chord		
Chord-based			Means of improvement reased finger table size,	Key differences with CR-Chord Maintenance of many
EpiChord [19]	ho	Chord	Pastry	Kademlia
F-Chord [4]	Geometry	Circular Node-ID	Plaxton-style mesh	XOR metric for dis
Freebie fingers		space, logarithmic degree mesh	network, prefix routing	tance between point in the key space.
Bidirectional la Chord [32], topology m	Routing algorithm	Search query for- warded to "closer" node	Matching Key and prefix of Node-ID	(XOR) Matching Key and Node-II based routing done parrallely.
Zig-zag routing	Routing performance	O(logN), where N is the number of peers	$O(log_BN)$, where N is number of peers, and $B=2b$, b is number of bits	$O(log_BN)+c$,
Cyclone [1] Mesaros's et al			of NodeID	Node-ID, and c is small constant
	Join/Leave performance	$(logN)^2$	log_BN	log _B N+c
RChord [35]	Routing	Periodic stabiliza-	Neighboring nodes	Failure of peers wil
P-Chord, PL-C	table maintenance	tion protocol at nodes to learn	periodically ex- change keep-alive	not cause network wide failure. Repli
H-Chord [5]		about newly joined nodes, update suc- cessor and prede-	messages. The leaf sets of nodes with adjacent Node-Id	cate data across multiple peers.
Chord enhance		cessor, and fix fin- ger tables.	overlap.	
	Bootstrap- ping	A new node knows an existing Chord	A new node knows a nearby Pastry	A new node knows an existing Kadem

Задача: разработка Хорды с широковещанием для мерцающей сети 2000+ узлов

- Отсутствуют материалы по Хорде
- Выполнен научно-технический поиск:
 - Обзорные материалы про р2р алгоритмов 2 книги, научные публикации:
 - 1. Tarkoma S. Publish/Subscribe Systems. Design and Principles // John Wiley & Sons Ltd. 2012. 346 p.
 - 2. Korzun D., Gurtov A. Structured Peer-to-peer Systems //Springer New York. 2013. 375 p.
 - 3. Dhara K., Guo Y., Kolberg M., Wu X. Overview of Structured Peer-to-Peer Overlay Algorithms //Handbook of Peer-to-Peer Networking.

2010. pp.223-256.

- 4. Другие
- Найдено улучшение Хорды для уменьшения задержек

JiangYi, Jinyuan Y. A Low Latency Chord Routing Algo for DHT // 1st Int. Symp. on Pervasive Computing and App. 2006. pp. 825-830/

• Найден алгоритм широковещания Хорды

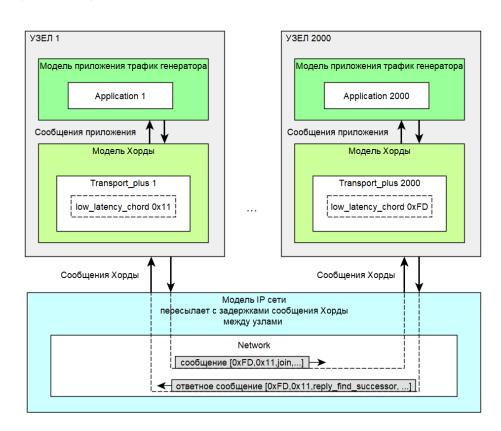
Huang K., Zhang D. DHT-based lightweight broadcast algorithms in large-scale computing infrastractures // Future Generation Computer Systems 2010. pp.291-303

	Chord Circular Node-ID space, logarithmic degree mesh Search query forwarded to "closer" node O(logN), where N is the number of peers	Pastry Plaxton-style mesh network, prefix routing Matching Key and prefix of Node-ID O(log _B N), where N is number of peers, and B=2b, b is number of bits	
Routing algorithm Routing	Circular Node-ID space, logarithmic degree mesh Search query forwarded to "closer" node O(logN), where N is the number of	Plaxton-style mesh network, prefix routing Matching Key and prefix of Node-ID O(log _B N), where N is number of peers, and B=2b, b is number of bits	XOR metric for distance between points in the key space. (XOR) Matching Key and Node-ID based routing done parrallely. O(log _B N)+c, Where N is number of peers, B=2b, b is
Routing algorithm Routing	space, logarithmic degree mesh Search query forwarded to "closer" node O(logN), where N is the number of	network, prefix routing Matching Key and prefix of Node-ID O(log _B N), where N is number of peers, and B=2b, b is number of bits	tance between points in the key space. (XOR) Matching Key and Node-ID based routing done parrallely. O(log _B N)+c, Where N is number of peers, B=2b, b is
algorithm Routing	degree mesh Search query forwarded to "closer" node O(logN), where N is the number of	routing Matching Key and prefix of Node-ID O(log _B N), where N is number of peers, and B=2b, b is number of bits	in the key space. (XOR) Matching Key and Node-ID based routing done parrallely. O(log _B N)+c, Where N is number of peers, B=2b, b is
algorithm Routing	warded to "closer" node $O(logN)$, where N is the number of	prefix of Node-ID $O(log_BN)$, where N is number of peers, and $B=2b$, b is number of bits	Key and Node-ID based routing done parrallely. O(log _B N)+c, Where N is number of peers, B=2b, b is
Routing	node $O(logN)$, where N is the number of	$O(log_BN)$, where N is number of peers, and $B=2b$, b is number of bits	based routing done parrallely. $O(log_BN)+c$, Where N is number of peers, $B=2b$, b is
•	O(logN), where N is the number of	N is number of peers, and $B=2b$, b is number of bits	parrallely. $O(log_BN)+c$, Where N is number of peers, $B=2b$, b is
•	is the number of	N is number of peers, and $B=2b$, b is number of bits	$O(log_BN)+c$,
•	is the number of	N is number of peers, and $B=2b$, b is number of bits	Where N is number of peers, $B=2b$, b is
performance		peers, and $B=2b$, b is number of bits	of peers, $B=2b$, b is
	peers	is number of bits	
			number of bits of
		of NodeID	Made ID and a in a
		of NodelD	Node-ID, and c is a small constant
Join/Leave	$(logN)^2$	log_BN	log_BN+c
performance	(10811)	108814	loggit ic
Routing	Periodic stabiliza-	Neighboring nodes	Failure of peers will
table	tion protocol at	periodically ex-	not cause network-
maintenance	nodes to learn	change keep-alive	wide failure. Repli-
	about newly joined	messages. The leaf	cate data across mul-
	nodes, update suc-	sets of nodes with	tiple peers.
		•	
	,	overlap.	
D	0		
Bootstrap-	A new node knows	A new node knows	A new node knows
		about newly joined	about newly joined nodes, update successor and predecessor, and fix finger tables. messages. The leaf sets of nodes with adjacent Node-Id overlap.

Задача: разработка Хорды с широковещанием для мерцающей сети 2000+ узлов

Выполнено:

- Подготовлен черновик укрупненного описания Хорды, конечный автомат
- Разработка кода началась 27 сентября, первый камит
- Модель состоит из:
 - > Модель Трафик генератора
 - ➤ Модель Хорды
 - Модель упрощенной сети



Задача: разработка Хорды с широковещанием для мерцающей сети 2000+ узлов

Выполнено:

- Подготовлен черновик укрупненного описания Хорды, конечный автомат
- Разработка кода началась 27 сентября, первый камит
- Модель состоит из:
 - Модель Трафик генератора
 - ➤ Модель Хорды
 - Модель упрощенной сети
- Модель выводит детальные логи
 о работе Хорды
 и скриншоты fingers с подсветкой

```
UPDATE set TIMER ACK mID 48043 retryC 0 reqC (
                               UPDATE isNowUpdate = yes
                                       pushNewMessage RX ACK mID 48043 pid 1 f t ACK a 0 ps
trp4.11chord
                               UPDATE # RX ACK mID 48043 pid 1 f t ACK a 0 ps 0 r 0 d 04 s
               TxRx
               TxRx
                                       pushNewMessage RX_SUCCESSOR mID 48043 pid 1 f t SUCCES
                                                                                                            "curr time": "275 s",
trp4.11chord
trp4.llchord
                               UPDATE _ # TIMER_UPDATE mID 0 retryC 0 reqC 0
                               UPDATE issueMessagePushTimers TX FIND SUCCESSOR mID 48703 ret
                               UPDATE isNowUpdate = yes
trp4.11chord
               TxRx
                                       sendMessage TX FIND SUCCESSOR mID 48703 pid 1 f t FIND
                                       pushNewMessage RX_ACK mID 48703 pid 1 f t ACK a 0 ps
                               UPDATE isNowUpdate = yes
                               UPDATE _# RX_SUCCESSOR mID 48703 pid 1 f t SUCCESSOR a 0 ps
               TxRx
                                       setFinger ccwfinger3 02
```

Задача: разработка Хорды с широковещанием для мерцающей сети 2000+ узлов

Разработка началась 27 сентября Простота описания Хорды ввела в заблуждение.

Для работы Хорды в мерцающей сети были добавлены:

- •10 приоритетов сообщений срочная отправка сообщений при подключении узла и обновлении fingers
 - JOIN = 1, наивысший приоритет
 - NOTIFY,
 - ACK,
 - SUCCESSOR,
 - FIND SUCCESSOR,
 - PREDECESSOR,
 - FIND PREDECESSOR,
 - BROADCAST,
 - MULTICAST.
 - SINGLE = 10

Задача: разработка Хорды с широковещанием для мерцающей сети 2000+ узлов

Разработка началась 27 сентября Простота описания Хорды ввела в заблуждение.

Для работы Хорды в мерцающей сети были добавлены:

- •10 приоритетов сообщений срочная отправка сообщений при подключении узла и обновлении fingers
- 5 таймеров, счетчики, буферизация запросных сообщений повторной отправки гарантированная доставка сообщений с подтверждением для мерцающей сети

Задача: разработка Хорды с широковещанием для мерцающей сети 2000+ узлов

Разработка началась 27 сентября Простота описания Хорды ввела в заблуждение.

Для работы Хорды в мерцающей сети были добавлены:

- •10 приоритетов сообщений срочная отправка сообщений при подключении узла и обновлении fingers
- 5 таймеров, счетчики, буферизация запросных сообщений повторной отправки гарантированная доставка сообщений с подтверждением для мерцающей сети

Задача: разработка Хорды с широковещанием для мерцающей сети 2000+ узлов

Разработка началась 27 сентября Простота описания Хорды ввела в заблуждение.

Для работы Хорды в мерцающей сети были добавлены:

- •10 приоритетов сообщений срочная отправка сообщений при подключении узла и обновлении fingers
- 5 таймеров, счетчики, буферизация запросных сообщений повторной отправки гарантированная доставка сообщений с подтверждением для мерцающей сети
- •Автоматическая перезагрузка Хорды

Задача: разработка Хорды с широковещанием для мерцающей сети 2000+ узлов

Разработка началась 27 сентября Простота описания Хорды ввела в заблуждение.

Для работы Хорды в мерцающей сети были добавлены:

- •10 приоритетов сообщений срочная отправка сообщений при подключении узла и обновлении fingers
- 5 таймеров, счетчики, буферизация запросных сообщений повторной отправки гарантированная доставка сообщений с подтверждением для мерцающей сети
- •Автоматическая перезагрузка Хорды
- •14 очередей с настраиваемым max_deep разрешение конфликтов одновременной обработки принятых и отправляемых сообщений, таймеров и событий перезагрузки Хорды



Задача: разработка Хорды с широковещанием для мерцающей сети 2000+ узлов

Разработка началась 27 сентября Простота описания Хорды ввела в заблуждение.

Для работы Хорды в мерцающей сети были добавлены:

- •10 приоритетов сообщений срочная отправка сообщений при подключении узла и обновлении fingers
- 5 таймеров, счетчики, буферизация запросных сообщений повторной отправки гарантированная доставка сообщений с подтверждением для мерцающей сети
- Автоматическая перезагрузка Хорды
- •14 очередей с настраиваемым max_deep разрешение конфликтов одновременной обработки принятых и отправляемых сообщений, таймеров и перезагрузки Хорды
- •Интервальная случайная нумерация с запоминанием идентификация сообщений минимальная защита для отбрасывания сообщений от неизвестных узлов
 - [0; 65535] делится на 131 интервал шириной 500 чисел
 - генерируется (псевдо)случайное число r в одном из интервалов
 - messageID = (nodeAddr.id << 16) + r</p>

Задача: разработка Хорды с широковещанием для мерцающей сети 2000+ узлов

Разработка началась 27 сентября Простота описания Хорды ввела в заблуждение.

Для работы Хорды в мерцающей сети были добавлены:

- •10 приоритетов сообщений срочная отправка сообщений при подключении узла и обновлении fingers
- 5 таймеров, счетчики, буферизация запросных сообщений повторной отправки гарантированная доставка сообщений с подтверждением для мерцающей сети
- •Автоматическая перезагрузка Хорды
- •14 очередей с настраиваемым max_deep разрешение конфликтов одновременной обработки принятых и отправляемых сообщений, таймеров и перезагрузки Хорды
- •Интервальная случайная нумерация с запоминанием идентификация сообщений минимальная защита для отбрасывания сообщений от неизвестных узлов
- •Конфигурационные параметры для работы каждого уровня

Задача: разработка Хорды с широковещанием для мерцающей сети 2000+ узлов

Разработка началась 27 сентября Простота описания Хорды ввела в заблуждение.

Для работы Хорды в мерцающей сети были добавлены:

- •10 приоритетов сообщений срочная отправка сообщений при подключении узла и обновлении fingers
- 5 таймеров, счетчики, буферизация запросных сообщений повторной отправки гарантированная доставка сообщений с подтверждением для мерцающей сети
- Автоматическая перезагрузка Хорды
- •14 очередей с настраиваемым max_deep для разрешение конфликтов одновременной обработки принятых и отправляемых сообщений, таймеров и перезагрузки Хорды
- •Интервальная случайная нумерация с запоминанием идентификация сообщений минимальная защита для отбрасывания сообщений от неизвестных узлов
- •Конфигурационные параметры для работы каждого уровня

Для Хорды задается 15 конфигурационных параметров + max_deep, size, priority для очередей, режим логов, путь сохранения и др.:

```
{netwAddr; seed; TrxSuccOnJoin;TrxSucc;TrxPred; Tupdate; TrxAck;
  TrxDuple; CtxJoin; CtxFindSucc; CtxRetry; CrxDuple; fingersSize; needsACK; fillFingersMinQty}
```

Задача: разработка Хорды для мерцающей сети 2000+ узлов

4 ноября – ставится задача – через 4-5 недель продемонстрировать механизм установки fingers Хорды, отложить другие механизмы

The pseudocode to stabilize the chord ring/circle after node joins and departures is as follows:

```
// create a new Chord ring.
n.create()
   predecessor := nil
    successor := n
// join a Chord ring containing node n'.
n.join(n')
    predecessor := nil
    successor := n'.find successor(n)
// called periodically. n asks the successor
// about its predecessor, verifies if n's immediate
// successor is consistent, and tells the successor about n
n.stabilize()
    x = successor.predecessor
    if x \in (n, successor) then
        successor := x
    successor.notify(n)
// n' thinks it might be our predecessor.
n.notify(n')
    if predecessor is nil or n'∈(predecessor, n) then
        predecessor := n'
// called periodically. refreshes finger table entries.
// next stores the index of the finger to fix
n.fix fingers()
    next := next + 1
    if next > m then
        next := 1
    finger[next] := find successor(n+2^{next-1});
// called periodically. checks whether predecessor has failed.
n.check predecessor()
    if predecessor has failed then
        predecessor := nil
```

The pseudocode to find the *successor* node of an id is given below:

```
// ask node n to find the successor of id
n.find successor(id)
    // Yes, that should be a closing square bracket to match
    // It is a half closed interval.
    if id ∈ (n, successorl then
        return successor
    else
        // forward the query around the circle
        n0 := closest preceding node(id)
        return n0.find successor(id)
// search the local table for the highest predecessor of id
n.closest preceding node(id)
    for i = m downto 1 do
        if (finger[i] \in (n, id)) then
            return finger[i]
    return n
```

4 ноября — ставится задача — через 4-5 недель продемонстрировать механизм установки *fingers* Хорды, отложить другие механизмы

30 ноября – сделана установка *fingers* Хорды + другие механизмы. fingers не устанавливаются – ошибки в описании Хорды

Обнаруженные ошибки в описании Хорды:

1. Бесконечная рекурсия

```
// ask node n to find the successor of id
   n.find successor(id)
        // Yes, that should be a closing square bracket to match the open
        // It is a half closed interval.
        if id ∈ (n, successor] then
            return successor
        else
            // forward the query around the circle
8
            n0 := closest preceding node(id) 	

    return n0.find successor(id)

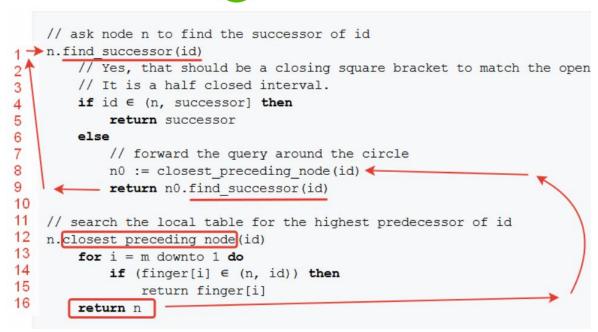
10
    // search the local table for the highest predecessor of id
    n.closest preceding node (id)
13
        for i = m downto 1 do
14
            if (finger[i] ∈ (n, id)) then
15
                return finger[i]
16
        return n
```

4 ноября – Костя ставится задача – через 4-5 недель продемонстрировать механизм установки fingers Хорды, отложить другие механизмы

30 ноября – сделано механизм установки Хорды + другие механизмы. fingers не устанавливаются – ошибки в описании Хорды

Обнаруженные ошибки в описании Хорды:

1. Бесконечная рекурсия Исправлено



18

4 ноября – ставится задача – через 4-5 недель продемонстрировать механизм установки fingers Хорды, отложить другие механизмы

30 ноября – сделано механизм установки Хорды + другие механизмы. fingers не устанавливаются – ошибки в описании Хорды

Обнаруженные ошибки в описании Хорды:

2. Нет запрета на обновление finger у самого себя

4 ноября — ставится задача — через 4-5 недель продемонстрировать механизм установки fingers Хорды, отложить другие механизмы

30 ноября – сделано механизм установки Хорды + другие механизмы. fingers не устанавливаются – ошибки в описании Хорды

Обнаруженные ошибки в описании Хорды:

2. Нет запрета на обновление finger у самого себя Исправлено



4 ноября – ставится задача – через 4-5 недель продемонстрировать механизм установки fingers Хорды, отложить другие механизмы

30 ноября – сделано механизм установки Хорды + другие механизмы. fingers не устанавливаются – ошибки в описании Хорды

Обнаруженные ошибки в описании Хорды:

2. Нет запрета на обновление finger у самого себя Исправлено



3. Не видит finger, хотя он есть Исправлено

4 ноября – ставится задача – через 4-5 недель продемонстрировать механизм установки fingers Хорды, отложить другие механизмы

30 ноября – сделано механизм установки Хорды + другие механизмы. fingers не устанавливаются – ошибки в описании Хорды

Обнаруженные ошибки в описании Хорды:

- 2. Нет запрета на обновление finger у самого себя Исправлено
- 3. Не видит finger, хотя он есть Исправлено

```
// search the local table for the highest predecessor of id
n.closest preceding node (id)
    for i = m downto 1 do
                                           4 ∈ (1; 4) - не выполняется
        if (finger[i] ∈ (n, id)) then
                                            будет возращен return n =1
            return finger[i]
    return n
```

4. Механизм подключения узла к сети не позволяет в полной мере добавить узел в

кольцо

```
// join a Chord ring containing node n'
 n.join(n')
     predecessor := nil
     successor := n'.find successor(n)
III TRINITY I AB
```

2|+1=3 -> null2|+2 = 4 -> null

 $2|+4=6 \rightarrow null$ 2|+8 = 10 -> null

4 ноября – ставится задача – через 4-5 недель продемонстрировать механизм установки fingers Хорды, отложить другие механизмы

30 ноября – сделано механизм установки Хорды + другие механизмы. fingers не устанавливаются – ошибки в описании Хорды

Обнаруженные ошибки в описании Хорды:

- 2. Нет запрета на обновление finger у самого себя Исправлено
- 3. Не видит finger, хотя он есть Исправлено

4. Механизм подключения узла к сети не позволяет в полной мере добавить узел в

кольцо

III TRINITY LAB

```
// join a Chord ring containing node n'
n.join(n')
   predecessor := nil
   successor := n'.find_successor(n)
```

Join(2)

2 | +1 = 3 -> null

2 | +2 = 4 -> null

2 | +4 = 6 -> null

2 | +8 = 10 -> null

4 ноября – ставится задача – через 4-5 недель продемонстрировать механизм установки fingers Хорды, отложить другие механизмы

30 ноября – сделано механизм установки Хорды + другие механизмы. fingers не устанавливаются – ошибки в описании Хорды

Обнаруженные ошибки в описании Хорды:

- 2. Нет запрета на обновление finger у самого себя Исправлено
- 3. Не видит finger, хотя он есть Исправлено

4. Механизм подключения узла к сети не позволяет в полной мере добавить узел в

2|+8 = 10 -> null

кольцо

III TRINITY LAB

```
// join a Chord ring containing node n'
n.join(n')
    predecessor := nil
    successor := n'.find_successor(n)
```

2 Successor(2) 1 $2|+1=3 \rightarrow \text{null}$ $2|+2=4 \rightarrow \text{null}$ $2|+4=6 \rightarrow \text{null}$

Join(2)

4 ноября – ставится задача – через 4-5 недель продемонстрировать механизм установки fingers Хорды, отложить другие механизмы

30 ноября – сделано механизм установки Хорды + другие механизмы. fingers не устанавливаются – ошибки в описании Хорды

Обнаруженные ошибки в описании Хорды:

- 2. Нет запрета на обновление finger у самого себя Исправлено
- 3. Не видит finger, хотя он есть Исправлено

4. Механизм подключения узла к сети не позволяет в полной мере добавить узел в

кольцо

```
// join a Chord ring containing node n'
n.join(n')
    predecessor := nil
    successor := n'.find_successor(n)
```

III TRINITY LAB

```
find_successor (3)

2 | +1 = 3 -> null

2 | +2 = 4 -> null

2 | +4 = 6 -> null

2 | +8 = 10 -> null
```

4 ноября – ставится задача – через 4-5 недель продемонстрировать механизм установки fingers Хорды, отложить другие механизмы

30 ноября – сделано механизм установки Хорды + другие механизмы. fingers не устанавливаются – ошибки в описании Хорды

Обнаруженные ошибки в описании Хорды:

- 2. Нет запрета на обновление finger у самого себя Исправлено
- 3. Не видит finger, хотя он есть Исправлено

4. Механизм подключения узла к сети не позволяет в полной мере добавить узел в

кольцо

```
// join a Chord ring containing node n'
n.join(n')
    predecessor := nil
    successor := n'.find_successor(n)
```

III TRINITY LAB

4 ноября – ставится задача – через 4-5 недель продемонстрировать механизм установки fingers Хорды, отложить другие механизмы

30 ноября – сделано механизм установки Хорды + другие механизмы. fingers не устанавливаются – ошибки в описании Хорды

Обнаруженные ошибки в описании Хорды:

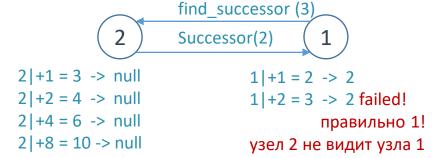
- 2. Нет запрета на обновление finger у самого себя Исправлено
- 3. Не видит finger, хотя он есть Исправлено

4. Механизм подключения узла к сети не позволяет в полной мере добавить узел в

кольцо

```
// join a Chord ring containing node n'
n.join(n')
    predecessor := nil
    successor := n'.find_successor(n)
```

III TRINITY LAB



4 ноября – ставится задача – через 4-5 недель продемонстрировать механизм установки fingers Хорды, отложить другие механизмы

30 ноября – сделано механизм установки Хорды + другие механизмы. fingers не устанавливаются – ошибки в описании Хорды

Обнаруженные ошибки в описании Хорды:

- 2. Нет запрета на обновление finger у самого себя Исправлено
- 3. Не видит finger, хотя он есть Исправлено

```
// search the local table for the highest predecessor of id n.closest_preceding_node(id)

for i = m downto 1 do

if (finger[i] ∈ (n, id)) then

return finger[i]

return n

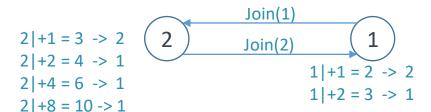
return n
```

4. Механизм подключения узла к сети не позволяет в полной мере добавить узел в кольцо Исправлено Принудительно обновляю fingers узла 2 при

// join a Chord ring containing node n'
n.join(n')
 predecessor := nil
 successor := n'.find_successor(n)

III TRINITY LAB

добавлении узла 1, запускаю fix_fingers



4 ноября – ставится задача – через 4-5 недель продемонстрировать механизм установки fingers Хорды, отложить другие механизмы

30 ноября – сделано механизм установки Хорды + другие механизмы. fingers не устанавливаются – ошибки в описании Хорды

Обнаруженные ошибки в описании Хорды:

- 2. Нет запрета на обновление finger у самого себя Исправлено
- 3. Не видит finger, хотя он есть Исправлено

```
// search the local table for the highest predecessor of id n.closest_preceding_node(id)

for i = m downto 1 do

if (finger[i] ∈ (n, id)) then

return finger[i]

return n

TPИМЕР:

4 ∈ (1; 4) - НЕ ВЫПОЛНЯЕТСЯ

будет возращен return n = 1 failed
```

4. Механизм подключения узла к сети не позволяет в полной мере добавить узел в

кольцо

III TRINITY LAB

```
// join a Chord ring containing node n'
n.join(n')
   predecessor := nil
   successor := n'.find_successor(n)
```

Join(1)
2|+1 = 3 -> 2
2|+2 = 4 -> 1
2|+4 = 6 -> 1
2|+8 = 10 -> 1

4 ноября — ставится задача — через 4-5 недель продемонстрировать механизм установки fingers Хорды, отложить другие механизмы

30 ноября – сделано механизм установки Хорды + другие механизмы. fingers не устанавливаются – ошибки в описании Хорды

Обнаруженные ошибки в описании Хорды:

- 2. Нет запрета на обновление finger у самого себя Исправлено
- 3. Не видит finger, хотя он есть Исправлено

```
// search the local table for the highest predecessor of id
n.closest preceding node (id)
    for i = m downto 1 do
                                            4 ∈ (1; 4) - не выполняется
        if (finger[i] \in (n, id)) then
                                            будет возращен return n =1
             return finger[i]
    return n
```

4. Механизм подключения узла к сети не позволяет в полной мере добавить узел в Принудительно обновляю fingers узла 2 при кольцо Исправлено

```
// join a Chord ring containing node n'
n.join(n')
    predecessor := nil
    successor := n'.find successor(n)
```

III TRINITY I AB

```
// called periodically. refreshes finger table entries.
// next stores the index of the finger to fix
n.fix fingers()
    next := next + 1
    if next > m then
        next := 1
```

30

добавлении узла 1, запускаю fix_fingers

finger[next] := find successor($n+2^{next-1}$);

4 ноября – ставится задача – через 4-5 недель продемонстрировать механизм установки fingers Хорды, отложить другие механизмы

30 ноября – сделано механизм установки Хорды + другие механизмы. fingers не устанавливаются – ошибки в описании Хорды

Обнаруженные ошибки в описании Хорды:

- 2. Нет запрета на обновление finger у самого себя Исправлено
- 3. Не видит finger, хотя он есть Исправлено

```
// search the local table for the highest predecessor of id n.closest_preceding_node(id)

for i = m downto 1 do

if (finger[i] ∈ (n, id)) then

return finger[i]

return n

TPUMEP:

4 ∈ (1; 4) - Не ВЫПОЛНЯЕТСЯ
будет возращен return n = 1 failed
```

4. Механизм подключения узла к сети не позволяет в полной мере добавить узел в кольцо Исправлено Принудительно обновляю fingers узла 2 при

```
// join a Chord ring containing node n'
n.join(n')

predecessor := nil

successor := n'.find_successor(n)

// called periodically. refreshes finger table e
// next stores the index of the finger to fix
n.fix_fingers()
next := next + 1
if next > m then
```

III TRINITY I AB

// called periodically. refreshes finger table entries.
// next stores the index of the finger to fix
n.fix_fingers()
next := next + 1
if next > m then
next := 1
finger[next] := find successor(n+2^{next-1});
n = 14
find_successor(14+2⁰)

```
// ask node n to find the successor of id
                                              Пример 1:
                                                                                              succ = 2 + 1 = 3 -> 4
n.find successor(id)
   // Yes, that should be a closing square bra
                                              узел 2 спрашивает find_succ(6) у узла 4
   // It is a half closed interval.
   if id ∈ (n, successor] then
                                              id = 6, n = 4, succ = 2
        return successor
                                                                                                     find_succ(6)?
    else
       // forward the query around the circle Bbl30B 4.find_successor(6)
       n0 := closest preceding node(id)
                                              6 \in (4; 2] false
                                                                                          6
       return n0.find successor(id)
// search the local table for the highest prede
                                                                                                 succ = 4 | +1 = 5 -> 2
n.closest preceding node(id)
   for i = m downto 1 do
       if (finger[i] \in (n, id)) then
                                               Затем 4.closest preceding node(6)
            return finger[i]
                                                   4.finger[1] 2 \in (4; 6) false
    return n
```

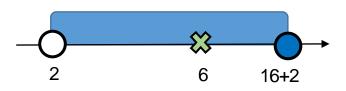
return n //не правильно

```
// ask node n to find the successor of id
n.find successor(id)
    // Yes, that should be a closing square bra
   // It is a half closed interval.
   if id ∈ (n, successor] then
        return successor
    else
        // forward the query around the circle Bbl30B 4.find_successor(6)
        n0 := closest preceding node(id)
        return n0.find successor(id)
// search the local table for the highest prede
n.closest preceding node(id)
    for i = m downto 1 do
        if (finger[i] \in (n, id)) then
            return finger[i]
    return n
```

Сергей предложил:

6 ∈ (4; макс размер кольца + 2]

 $6 \in (4; 16 + 2]$ true



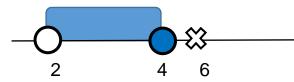
Затем return successor //2 работает

Пример 1:

узел 2 спрашивает find_succ(6) у узла 4

id = 6, n = 4, succ = 2

 $6 \in (4; 2]$ false

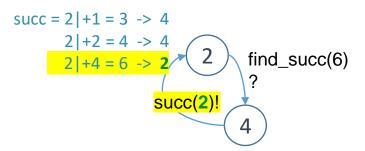


succ = 2 + 1 = 3 -> 4

find_succ(6)?

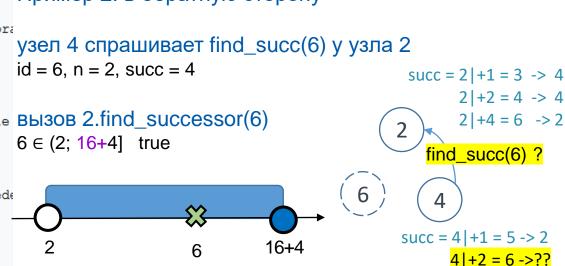
6 succ = 4 | +1 = 5 -> 2

Затем 4.closest_preceding_node 4.finger[1] $2 \in (4; 6)$ false return n //не правильно

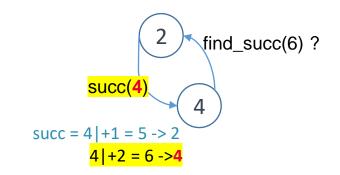


```
// ask node n to find the successor of id
n.find successor(id)
    // Yes, that should be a closing square bra
   // It is a half closed interval.
   if id ∈ (n, successor] then
        return successor
    else
        // forward the query around the circle Bbl30B 2.find_successor(6)
        n0 := closest preceding node(id)
        return n0.find successor(id)
// search the local table for the highest prede
n.closest preceding node(id)
    for i = m downto 1 do
        if (finger[i] \in (n, id)) then
            return finger[i]
    return n
```

Пример 2: в обратную сторону

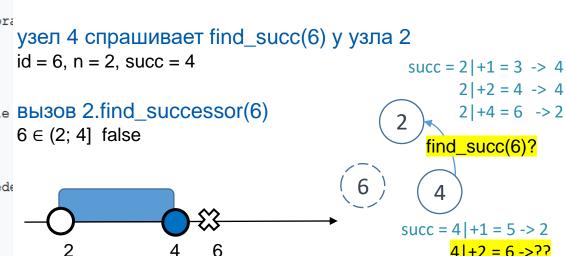


Затем return successor //4 не правильно



```
// ask node n to find the successor of id
n.find successor(id)
    // Yes, that should be a closing square bra
   // It is a half closed interval.
   if id ∈ (n, successor] then
        return successor
    else
        // forward the query around the circle Bbl30B 2.find_successor(6)
        n0 := closest preceding node(id)
        return n0.find successor(id)
// search the local table for the highest prede
n.closest preceding node(id)
    for i = m downto 1 do
        if (finger[i] \in (n, id)) then
            return finger[i]
    return n
```

Пример 2: в обратную сторону



Затем closest_preceding_node выполнит finger[3] $2 \in (2; 6)$ false finger[2] $4 \in (2; 6)$ true return finger[2] //4 не правильно

```
// ask node n to find the successor of id
n.find successor(id)
    // Yes, that should be a closing square bra
   // It is a half closed interval.
   if id ∈ (n, successor] then
        return successor
    else
        // forward the query around the circle Bbl30B 2.find_successor(6)
        n0 := closest preceding node(id)
        return n0.find successor(id)
// search the local table for the highest prede
n.closest preceding node(id)
    for i = m downto 1 do
        if (finger[i] \in (n, id)) then
            return finger[i]
    return n
```

Пример 2: в обратную сторону

узел 4 спрашивает find_succ(6) у узла 2

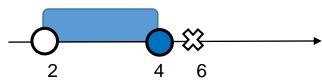
$$id = 6$$
, $n = 2$, $succ = 4$

succ = 2 | +1 = 3 -> 42|+2=4 -> 4

 $2|+4=6 \rightarrow 2$

find_succ(6)?

 $6 \in (2; 4]$ false



succ = 4 | +1 = 5 -> 24|+2=6->??

Затем closest_preceding_node выполнит

finger[3] $2 \in (2; 6)$ false finger[2] $4 \in (2; 6)$ true return finger[2] //4 не правильно

добавлено: если проверять так: finger[3] $2 \in [2; 6)$ true, if finger[3] == nreturn REPLY finger[3] //2 pa6otaet else return n0.find_successor(finger[3])

find_succ(6)? succ(2)!

succ = 4 | +1 = 5 -> 2

```
// ask node n to find the successor of id
                                            Пример 3: не работает предложенное решение
n.find successor(id)
   // Yes, that should be a closing square bra
                                            узел n = 4
   // It is a half closed interval.
   if id ∈ (n, successor] then
                                            successor = 2
       return successor
   else
       // forward the query around the circle
                                            получается условие:
       n0 := closest preceding node(id)
       return n0.find successor(id)
                                            (4; 2]
// search the local table for the highest prede
n.closest preceding node(id)
                                            по предложенному решению должно быть:
   for i = m downto 1 do
                                            (4; 2] \rightarrow (4; 16+2) \rightarrow (4; 18)
       if (finger[i] \in (n, id)) then
           return finger[i]
   return n
```

не работает в ситуации, когда надо проверить id = 1

получается id=1 HE принадлежит интервалу (4; 18]

```
// ask node n to find the successor of id
n.find_successor(id)
    // Yes, that should be a closing square bra
    // It is a half closed interval.
    if id ∈ (n, successor] then
        return successor
    else
        // forward the query around the circle
        n0 := closest preceding node(id)
        return n0.find successor(id)
// search the local table for the highest prede
n.closest preceding node(id)
    for i = m downto 1 do
        if (finger[i] ∈ (n, id)) then
            return finger[i]
    return n
```

Пример 3: не работает предложенное решение

```
узел n = 4
successor = 2
```

получается условие: (4; 2]

по предложенному решению должно быть: (4; 2] -> (4; 16+2] -> (4; 18]

не работает в ситуации, когда надо проверить id = 1

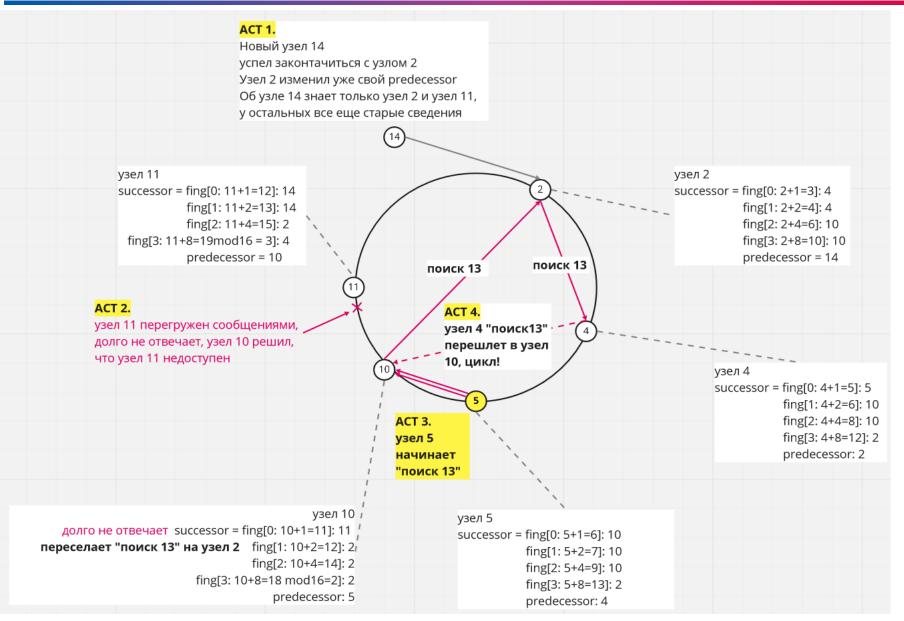
получается id=1 НЕ принадлежит интервалу (4; 18]

Исправлено

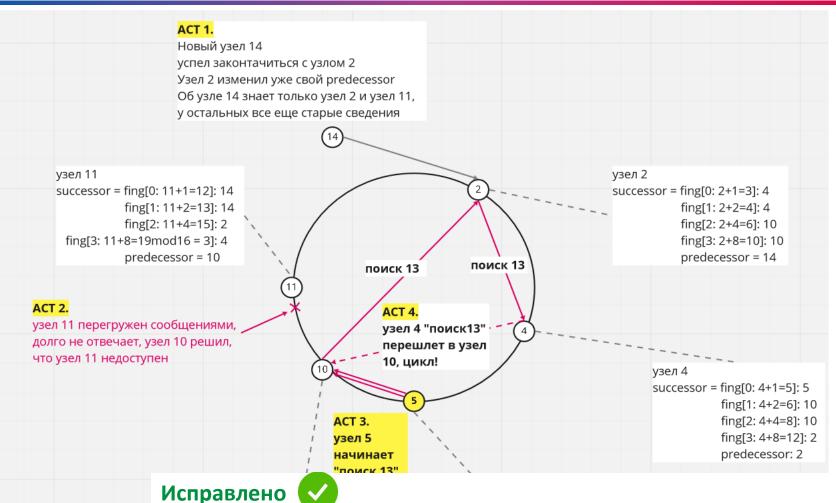
Придуман другой способ проверки (см. функцию isInRangeOverZero в inc.h).

В Хорде на JAVA также не правильно будет проверяться

Проект Хорда. Циклится по внутреннему кольцу



Проект Хорда. Циклится по внутреннему кольцу



долго не отвечает success переселает "поиск 13" на уз

fing

- Добавлен retransmit counter в сообщение. Если сообщение пересылается больше раз, чем размерность logN, то сообщение отбрасывается.
- Если истек таймер, а ответ так и не был получен (сообщение было отброшено по retransmit counter),

то запускается stabilize в узле. Также и в узле10 будет выполнено stabilize для своих сообщений и сообщения правильно будут передаваться по кольцу.

III TRINITY LAB

Спасибо за внимание! Вопросы?