III TRINITY LAB

Review of p2p Chord project

Проект Хорда

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

- Отсутствуют материалы по Хорде
- Выполнен научно-технический поиск:
  - Обзорные материалы про p2p 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

Table 11.4 Comparison of CR-Chord with existing extensions to basic Chord

Chord-based design	Parameters being improved	Means of improvement	Key differences with CR-Chord
EpiChord [19]	Lookup latency, number of hops, availability under churn.	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.

Проект Хорда

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

- Отсутствуют материалы по Хорде
- Выполнен научно-технический поиск:
  - Обзорные материалы про p2p 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

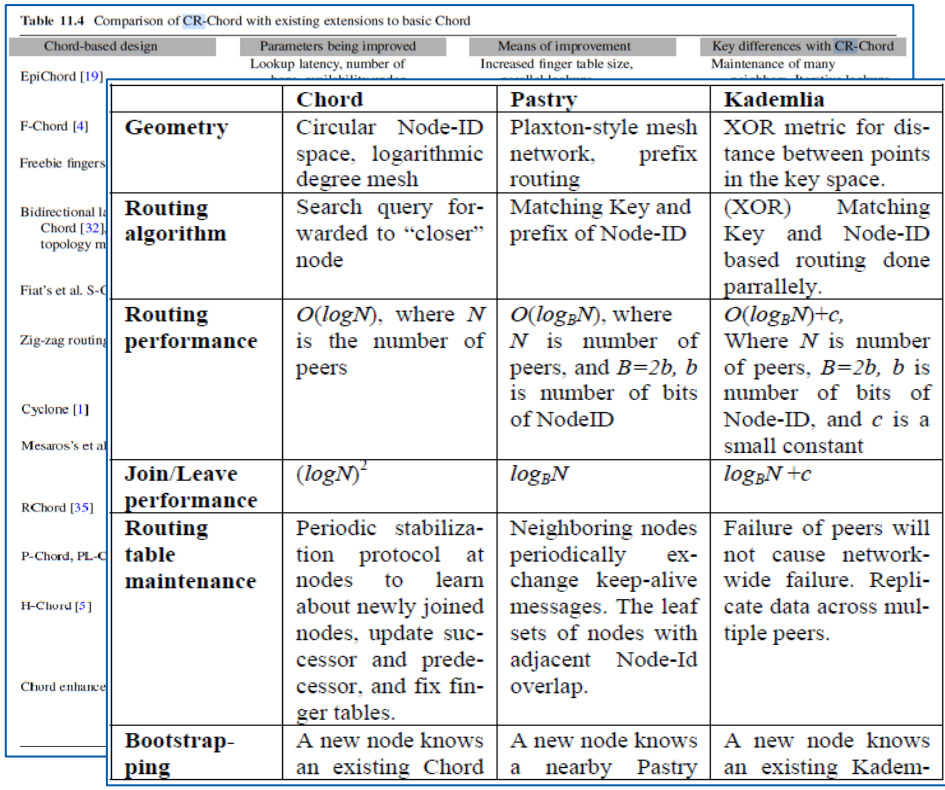


Table 11.4 Comparison of CR-Chord with existing extensions to basic Chord

Chord-based design	Parameters being improved	Means of improvement	Key differences with CR-Chord
EpiChord [19]	Lookup latency, number of	Increased finger table size,	Maintenance of many
F-Chord [4]			
Freebie fingers			
Bidirectional l… Chord [32] topology m…			
Fiat's et al. S-C…			
Zig-zag routing			
Cyclone [1]			
Mesaros's et al			
RChord [35]			
P-Chord, PL-C…			
H-Chord [5]			
Chord enhance…			

	Chord	Pastry	Kademlia
Geometry	Circular Node-ID space, logarithmic degree mesh	Plaxton-style mesh network, prefix routing	XOR metric for distance between points in the key space.
Routing algorithm	Search query forwarded to "closer" node	Matching Key and prefix of Node-ID	(XOR) Matching Key and Node-ID based routing done parrallely.
Routing performance	$O(logN)$, where N is the number of peers	$O(log_B N)$, where N is number of peers, and $B=2b$, b is number of bits of NodeID	$O(log_B N)+c$, Where N is number of peers, $B=2b$, b is number of bits of Node-ID, and c is a small constant
Join/Leave performance	$(logN)^2$	$log_B N$	$log_B N+c$
Routing table maintenance	Periodic stabilization protocol at nodes to learn about newly joined nodes, update successor and predecessor, and fix finger tables.	Neighboring nodes periodically exchange keep-alive messages. The leaf sets of nodes with adjacent Node-Id overlap.	Failure of peers will not cause network-wide failure. Replicate data across multiple peers.
Bootstrapping	A new node knows an existing Chord	A new node knows a nearby Pastry	A new node knows an existing Kadem-

Проект Хорда

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

- Отсутствуют материалы по Хорде
- Выполнен научно-технический поиск:
  - Обзорные материалы про p2p 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

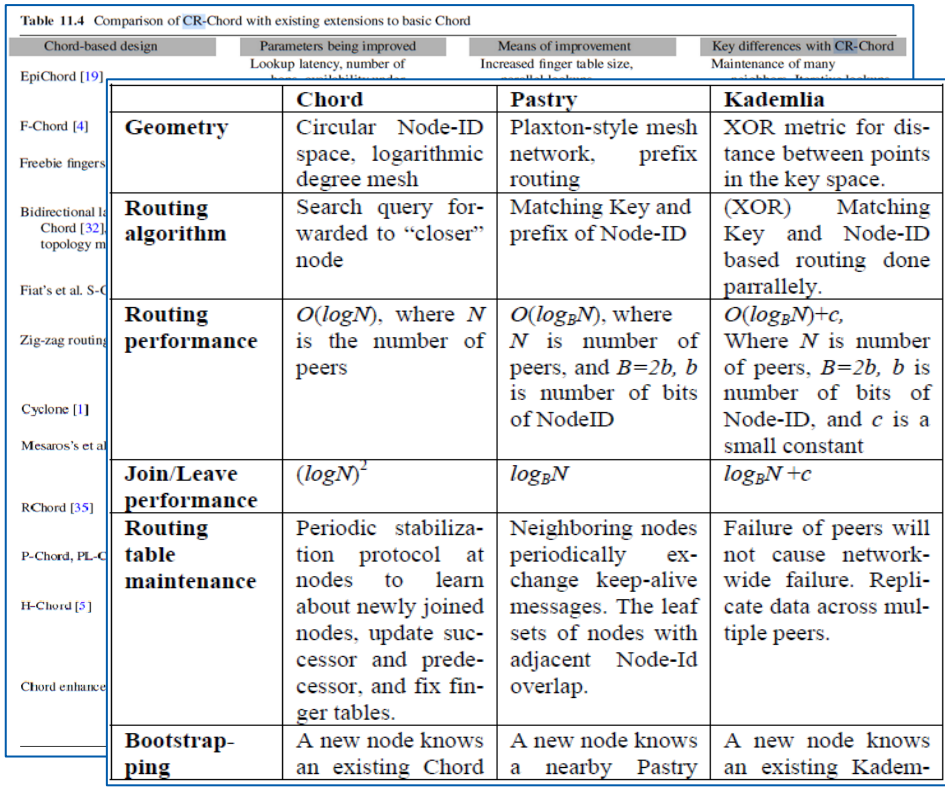


Table 11.4 Comparison of CR-Chord with existing extensions to basic Chord

Chord-based design	Parameters being improved	Means of improvement	Key differences with CR-Chord
EpiChord [19]	Lookup latency, number of	Increased finger table size,	Maintenance of many
F-Chord [4]			
Freebie fingers			
Bidirectional l… Chord [32] topology m…			
Fiat's et al. S-C…			
Zig-zag routing			
Cyclone [1]			
Mesaros's et al…			
RChord [35]			
P-Chord, PL-C…			
H-Chord [5]			
Chord enhance…			

	Chord	Pastry	Kademlia
Geometry	Circular Node-ID space, logarithmic degree mesh	Plaxton-style mesh network, prefix routing	XOR metric for distance between points in the key space.
Routing algorithm	Search query forwarded to "closer" node	Matching Key and prefix of Node-ID	(XOR) Matching Key and Node-ID based routing done parrallely.
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	$O(log_BN)+c$, Where N is number of peers, $B=2b$, b is number of bits of Node-ID, and c is a small constant
Join/Leave performance	$(logN)^2$	log_BN	log_BN+c
Routing table maintenance	Periodic stabilization protocol at nodes to learn about newly joined nodes, update successor and predecessor, and fix finger tables.	Neighboring nodes periodically exchange keep-alive messages. The leaf sets of nodes with adjacent Node-Id overlap.	Failure of peers will not cause network-wide failure. Replicate data across multiple peers.
Bootstrapping	A new node knows an existing Chord	A new node knows a nearby Pastry	A new node knows an existing Kadem-

Проект Хорда

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

- Отсутствуют материалы по Хорде
- Выполнен научно-технический поиск:
  - Обзорные материалы про p2p 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

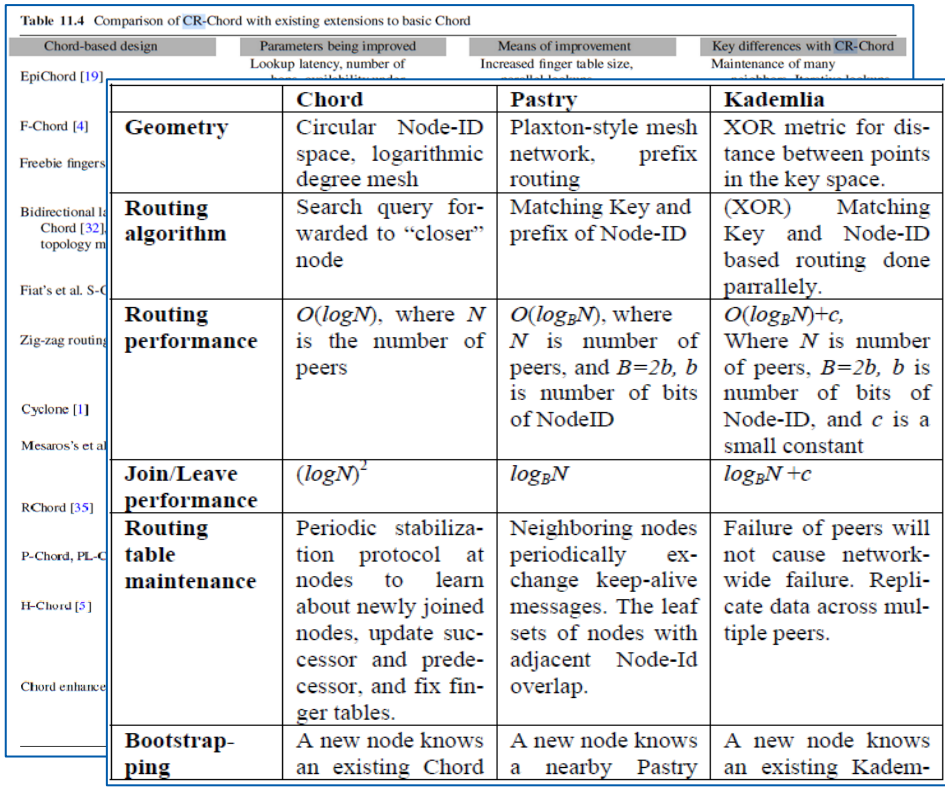


Table 11.4 Comparison of CR-Chord with existing extensions to basic Chord

Chord-based design	Parameters being improved	Means of improvement	Key differences with CR-Chord
EpiChord [19]	Lookup latency, number of	Increased finger table size,	Maintenance of many
F-Chord [4]			
Freebie fingers			
Bidirectional l… Chord [32] topology m…			
Fiat's et al. S-C…			
Zig-zag routing			
Cyclone [1]			
Mesaros's et al…			
RChord [35]			
P-Chord, PL-C…			
H-Chord [5]			
Chord enhance…			

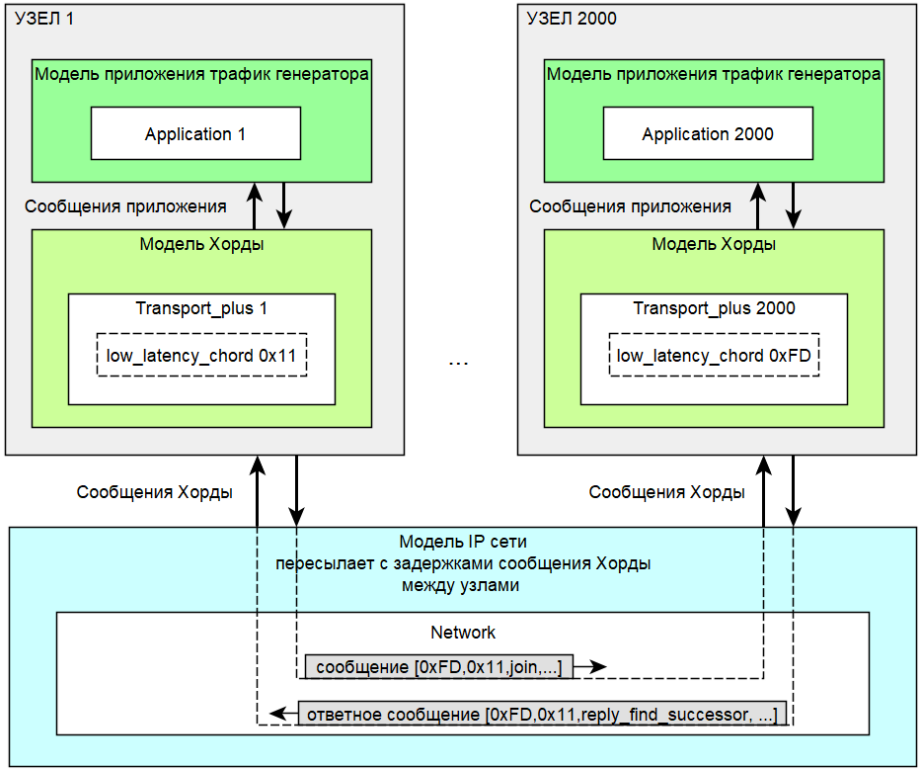
	Chord	Pastry	Kademlia
Geometry	Circular Node-ID space, logarithmic degree mesh	Plaxton-style mesh network, prefix routing	XOR metric for distance between points in the key space.
Routing algorithm	Search query forwarded to "closer" node	Matching Key and prefix of Node-ID	(XOR) Matching Key and Node-ID based routing done parrallely.
Routing performance	$O(logN)$, where N is the number of peers	$O(log_B N)$, where N is number of peers, and $B=2b$, b is number of bits of NodeID	$O(log_B N)+c$, Where N is number of peers, $B=2b$, b is number of bits of Node-ID, and c is a small constant
Join/Leave performance	$(logN)^2$	$log_B N$	$log_B N+c$
Routing table maintenance	Periodic stabilization protocol at nodes to learn about newly joined nodes, update successor and predecessor, and fix finger tables.	Neighboring nodes periodically exchange keep-alive messages. The leaf sets of nodes with adjacent Node-Id overlap.	Failure of peers will not cause network-wide failure. Replicate data across multiple peers.
Bootstrapping	A new node knows an existing Chord	A new node knows a nearby Pastry	A new node knows an existing Kadem-

Проект Хорда. Разработка модели

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

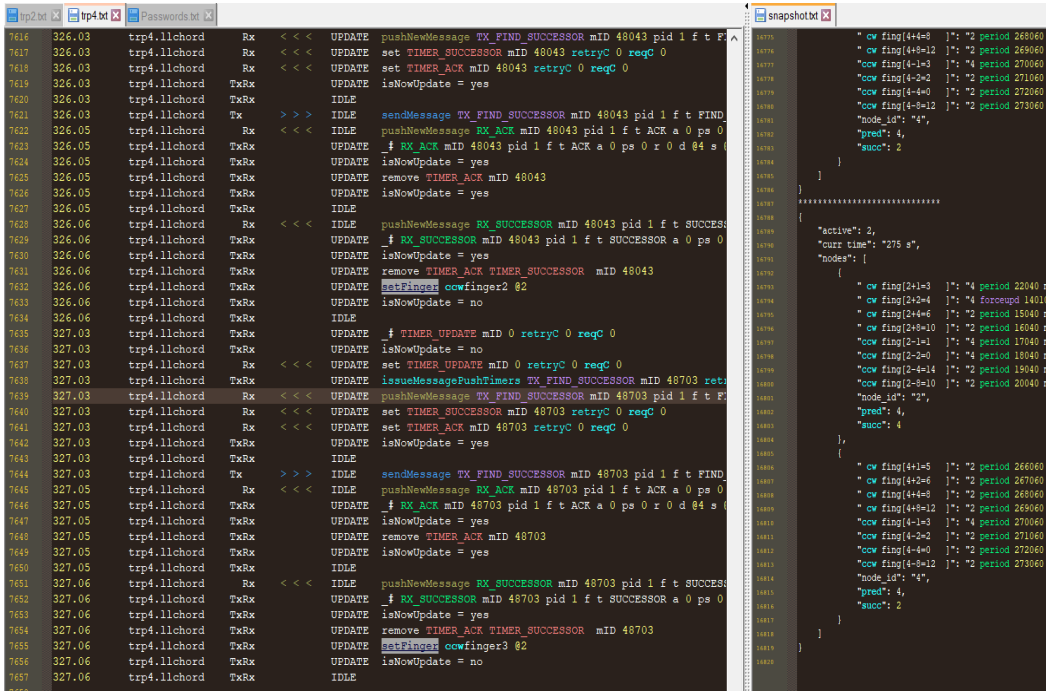


Проект Хорда. Разработка модели

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



Проект Хорда. Разработка модели

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

Проект Хорда. Разработка модели

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

Проект Хорда. Разработка модели

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

Проект Хорда. Разработка модели

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

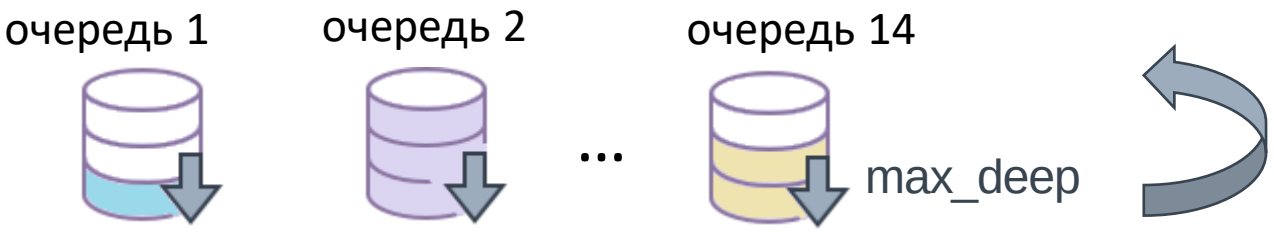
Проект Хорда. Разработка модели

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



Проект Хорда. Разработка модели

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

Проект Хорда. Разработка модели

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

Проект Хорда. Разработка модели

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}

Проект Хорда. Разработка модели

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 ∈ (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, 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 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
```

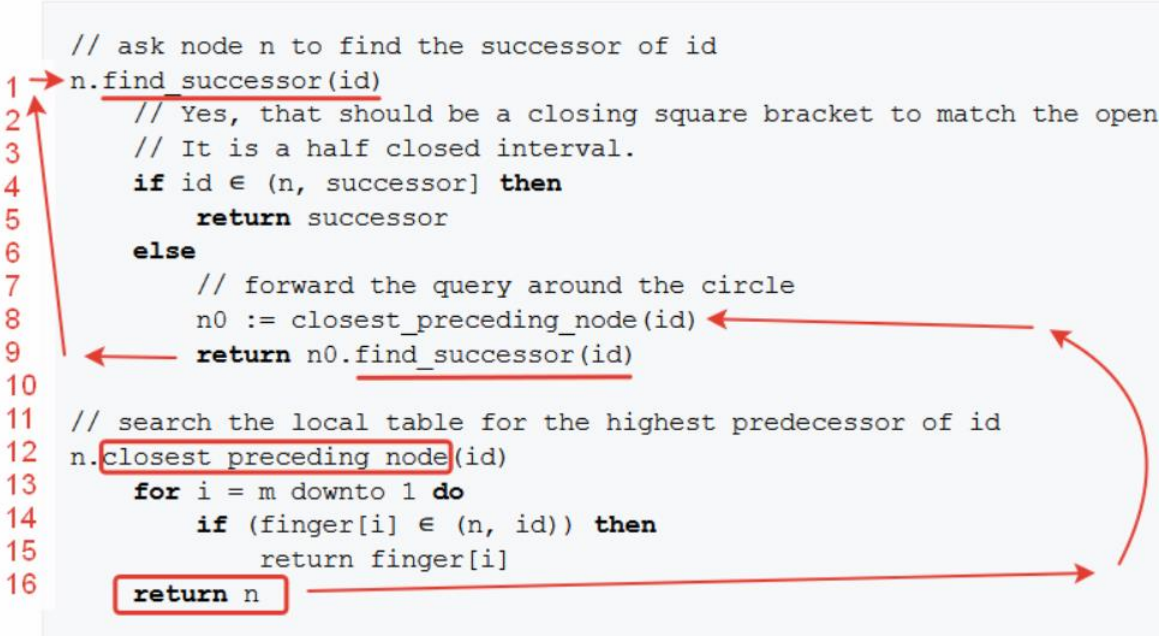
Проект Хорда. Разработка модели

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



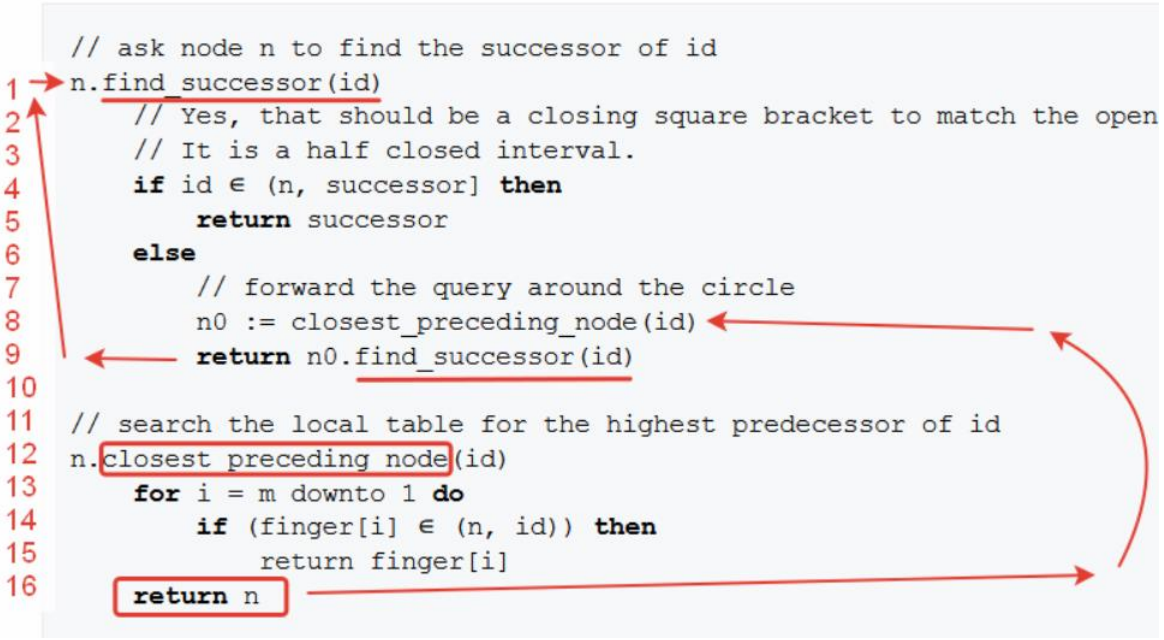
Проект Хорда. Разработка модели

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** ✓



Проект Хорда. Разработка модели

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

Проект Хорда. Разработка модели

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** 

Проект Хорда. Разработка модели

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
```

пример:
4 ∈ (1; 4) - не выполняется
будет возращен return n =1
failed

Проект Хорда. Разработка модели

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**

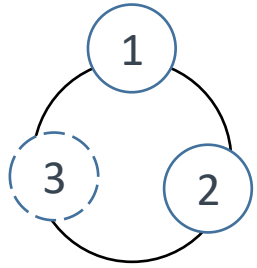
```
// search the local table for the highest predecessor of id
n.closest_preceding_node(id)
    for i = m downto 1 do
        if (finger[i] ∈ (n, id)) then
            return finger[i]
    return n
```

пример:
4 ∈ (1; 4) - не выполняется
будет возвращен return n =1
failed

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)
```

2|+1 = 3 -> null
2|+2 = 4 -> null
2|+4 = 6 -> null
2|+8 = 10 -> null



Проект Хорда. Разработка модели

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:

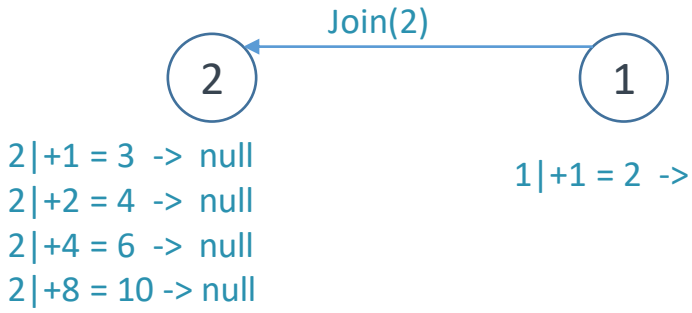
2. Нет запрета на обновление finger у самого себя **Исправлено**
3. Не видит finger, хотя он есть **Исправлено**

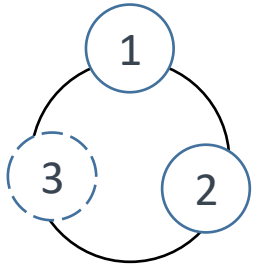
```
// search the local table for the highest predecessor of id
n.closest_preceding_node(id)
    for i = m downto 1 do
        if (finger[i] ∈ (n, id)) then
            return finger[i]
    return n
```

пример:
4 ∈ (1; 4) - не выполняется
будет возвращен return n =1
failed

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)
```





Проект Хорда. Разработка модели

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
```

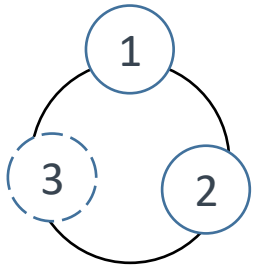
пример:
4 ∈ (1; 4) - не выполняется
будет возращен return n =1
failed

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

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

2|+1 = 3 -> null
2|+2 = 4 -> null
2|+4 = 6 -> null
2|+8 = 10 -> null

1|+1 = 2 -> 2



Проект Хорда. Разработка модели

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:

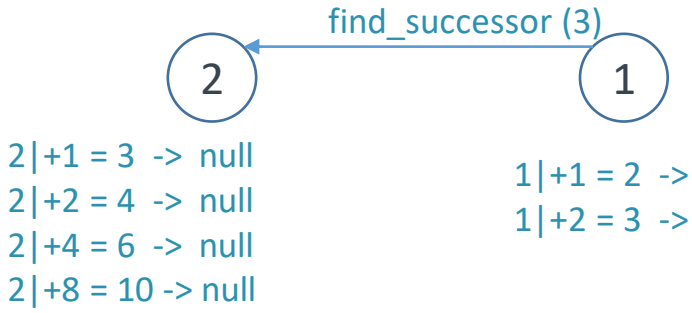
2. Нет запрета на обновление finger у самого себя **Исправлено**
3. Не видит finger, хотя он есть **Исправлено**

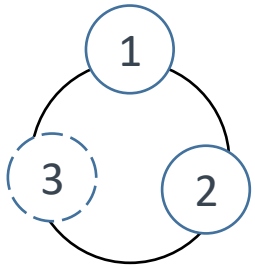
```
// search the local table for the highest predecessor of id
n.closest_preceding_node(id)
    for i = m downto 1 do
        if (finger[i] ∈ (n, id)) then
            return finger[i]
    return n
```

пример:
4 ∈ (1; 4) - не выполняется
будет возвращен return n =1
failed

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)
```





Проект Хорда. Разработка модели

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:

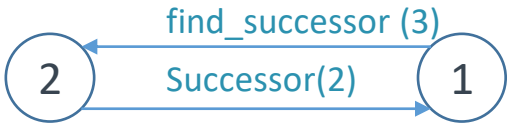
2. Нет запрета на обновление finger у самого себя **Исправлено**
3. Не видит finger, хотя он есть **Исправлено**

```
// 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
```

пример:
4 ∈ (1; 4) - не выполняется
будет возращен return n =1
failed

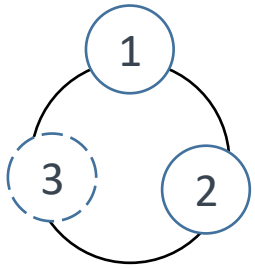
4. Механизм подключения узла к сети не позволяет в полной мере добавить узел в кольцо

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



2|+1 = 3 -> null
2|+2 = 4 -> null
2|+4 = 6 -> null
2|+8 = 10 -> null

1|+1 = 2 -> 2
1|+2 = 3 -> 2



Проект Хорда. Разработка модели

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:

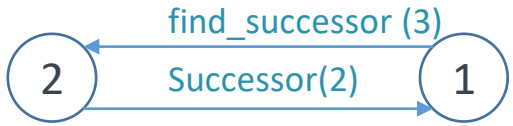
2. Нет запрета на обновление finger у самого себя **Исправлено**
3. Не видит finger, хотя он есть **Исправлено**

```
// 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
```

пример:
4 ∈ (1; 4) - не выполняется
будет возращен return n =1
failed

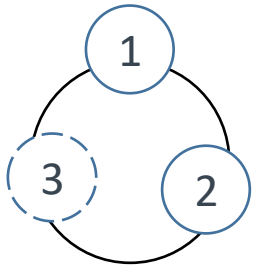
4. Механизм подключения узла к сети не позволяет в полной мере добавить узел в кольцо

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



2|+1 = 3 -> null
2|+2 = 4 -> null
2|+4 = 6 -> null
2|+8 = 10 -> null

1|+1 = 2 -> 2
1|+2 = 3 -> 2 failed!
правильно 1!
узел 2 не видит узла 1



Проект Хорда. Разработка модели

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**

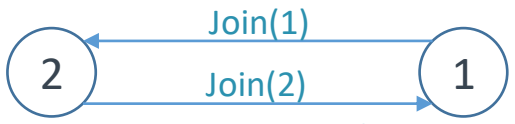
```
// search the local table for the highest predecessor of id
n.closest_preceding_node(id)
    for i = m downto 1 do
        if (finger[i] ∈ (n, id)) then
            return finger[i]
    return n
```

пример:
4 ∈ (1; 4) - не выполняется
будет возращен return n =1
failed

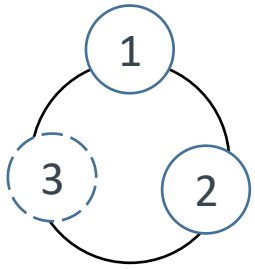
4. Механизм подключения узла к сети не позволяет в полной мере добавить узел в кольцо **Исправлено** Принудительно обновляю fingers узла 2 при добавлении узла 1, запускаю fix_fingers

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

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



1|+1 = 2 -> 2
1|+2 = 3 -> 1



Проект Хорда. Разработка модели

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**

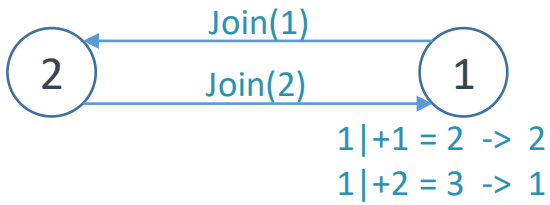
```
// search the local table for the highest predecessor of id
n.closest_preceding_node(id)
    for i = m downto 1 do
        if (finger[i] ∈ (n, id)) then
            return finger[i]
    return n
```

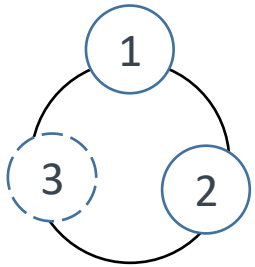
пример:
4 ∈ (1; 4) - не выполняется
будет возвращен return n =1
failed

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)
```

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





Проект Хорда. Разработка модели

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**

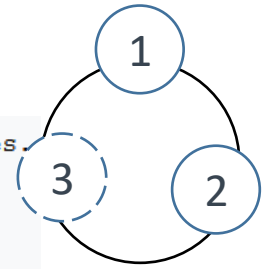
```
// search the local table for the highest predecessor of id
n.closest_preceding_node(id)
    for i = m downto 1 do
        if (finger[i] ∈ (n, id)) then
            return finger[i]
    return n
```

пример:
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 добавлении узла 1, запускаю fix_fingers

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



Проект Хорда. Разработка модели

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**

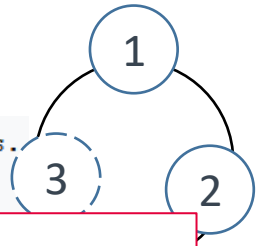
```
// search the local table for the highest predecessor of id
n.closest_preceding_node(id)
    for i = m downto 1 do
        if (finger[i] ∈ (n, id)) then
            return finger[i]
    return n
```

пример:
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 добавлении узла 1, запускаю fix_fingers

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



Проект Хорда. Проблема перехода через 0

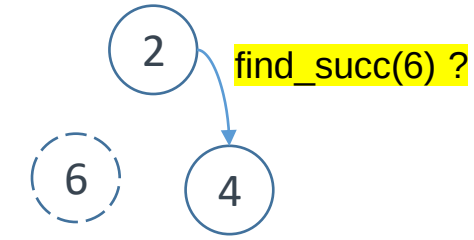
```
// ask node n to find the successor of id
n.find_successor(id)
    // Yes, that should be a closing square bra
    // It is a half closed interval.
    if id ∈ (n, successor] then
        return successor
    else
        // forward the query around the circle
        n0 := closest_preceding_node(id)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

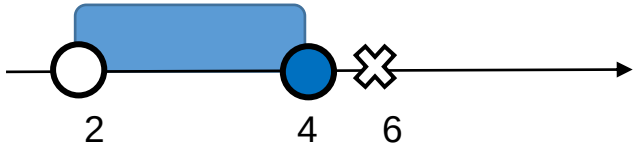
Пример 1:

succ = 2|+1 = 3 -> 4
2|+2 = 4 -> 4
2|+4 = 6 ->??

узел 2 спрашивает find_succ(6) у узла 4
id = 6, n = 4, succ = 2



вызов 4.find_successor(6)
6 ∈ (4; 2] false



Затем 4.closest_preceding_node(6)
4.finger[1] 2 ∈ (4; 6) false
return n **//не правильно**

Проект Хорда. Проблема перехода через 0

```
// 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
```

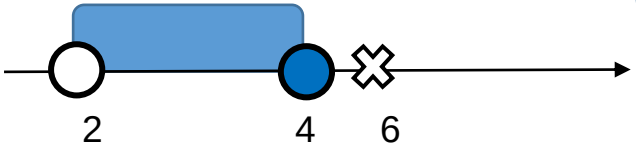
Пример 1:

узел 2 спрашивает find_succ(6) у узла 4

id = 6, n = 4, succ = 2

вызов 4.find_successor(6)

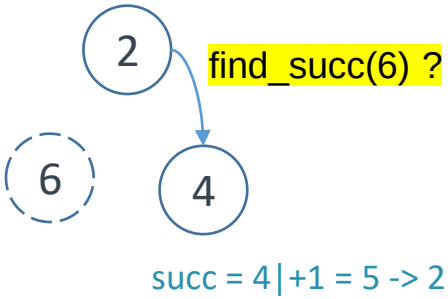
6 ∈ (4; 2] false



succ = 2|+1 = 3 -> 4

2|+2 = 4 -> 4

2|+4 = 6 ->??



succ = 4|+1 = 5 -> 2

Затем 4.closest_preceding_node

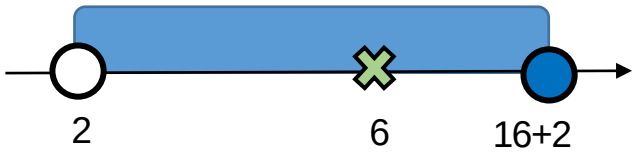
4.finger[1] 2 ∈ (4; 6) false

return n **//не правильно**

Сергей предложил:

6 ∈ (4; макс размер кольца + 2]

6 ∈ (4; 16 + 2] true

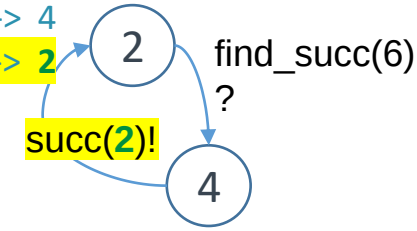


Затем return successor //**2 работает**

succ = 2|+1 = 3 -> 4

2|+2 = 4 -> 4

2|+4 = 6 -> **2**



Проект Хорда. Проблема перехода через 0

```
// 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 2: в обратную сторону

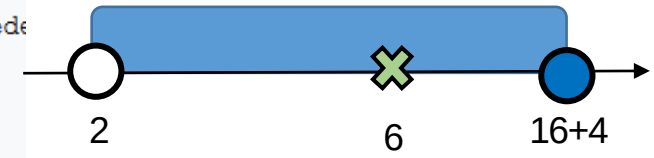
узел 4 спрашивает find_succ(6) у узла 2

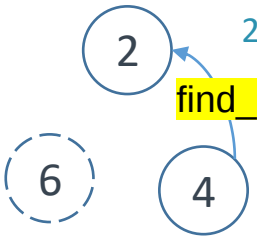
id = 6, n = 2, succ = 4

succ = 2|+1 = 3 -> 4
2|+2 = 4 -> 4
2|+4 = 6 -> 2

вызов 2.find_successor(6)

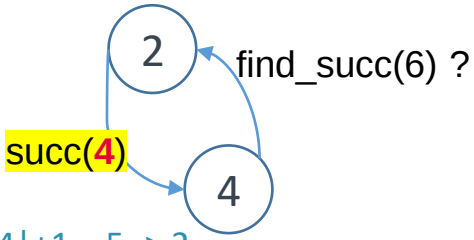
6 ∈ (2; 16+4] true





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

Затем return successor //4 **не правильно**



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

Проект Хорда. Проблема перехода через 0

```
// 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 2: в обратную сторону

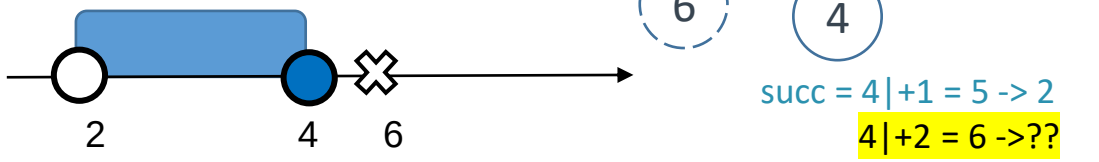
узел 4 спрашивает find_succ(6) у узла 2

id = 6, n = 2, succ = 4

succ = 2|+1 = 3 -> 4
2|+2 = 4 -> 4
2|+4 = 6 -> 2

вызов 2.find_successor(6)

6 ∈ (2; 4] false



Затем closest_preceding_node выполнит

finger[3] 2 ∈ (2; 6) false

finger[2] 4 ∈ (2; 6) true

return finger[2] //4 **не правильно**

Проект Хорда. Проблема перехода через 0

```
// 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 2: в обратную сторону

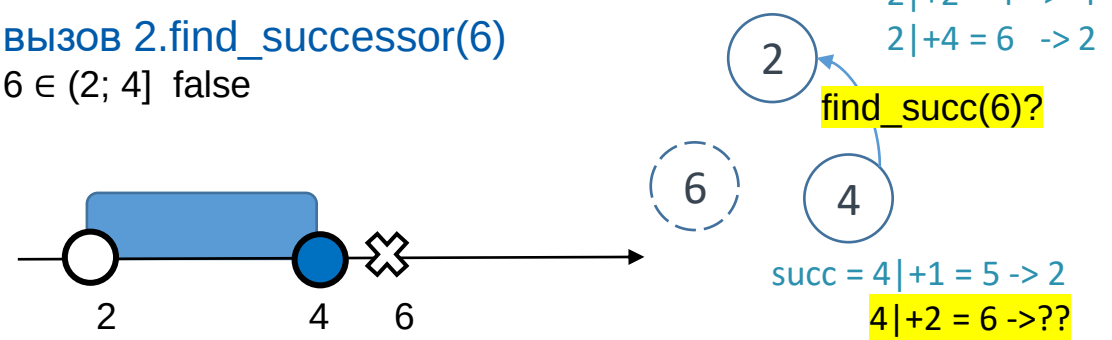
узел 4 спрашивает find_succ(6) у узла 2

id = 6, n = 2, succ = 4

succ = 2|+1 = 3 -> 4
2|+2 = 4 -> 4
2|+4 = 6 -> 2

вызов 2.find_successor(6)

6 ∈ (2; 4] false



Затем closest_preceding_node выполнит

finger[3] 2 ∈ (2; 6) false

finger[2] 4 ∈ (2; 6) true

return finger[2] //4 **не правильно**

добавлено: если проверять так:

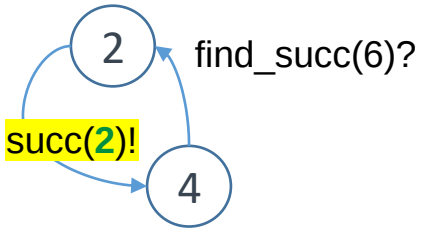
finger[3] 2 ∈ [2; 6) true,

if finger[3] == n

return REPLY finger[3] //**2 работает**

else

return n0.find_successor(finger[3])



succ = 4|+1 = 5 -> 2

4|+2 = 6 ->2

+

предыдущие исправления для fingers при подключении нового узла

Проект Хорда. Проблема перехода через 0

```
// 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]

Проект Хорда. Проблема перехода через 0

```
// 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

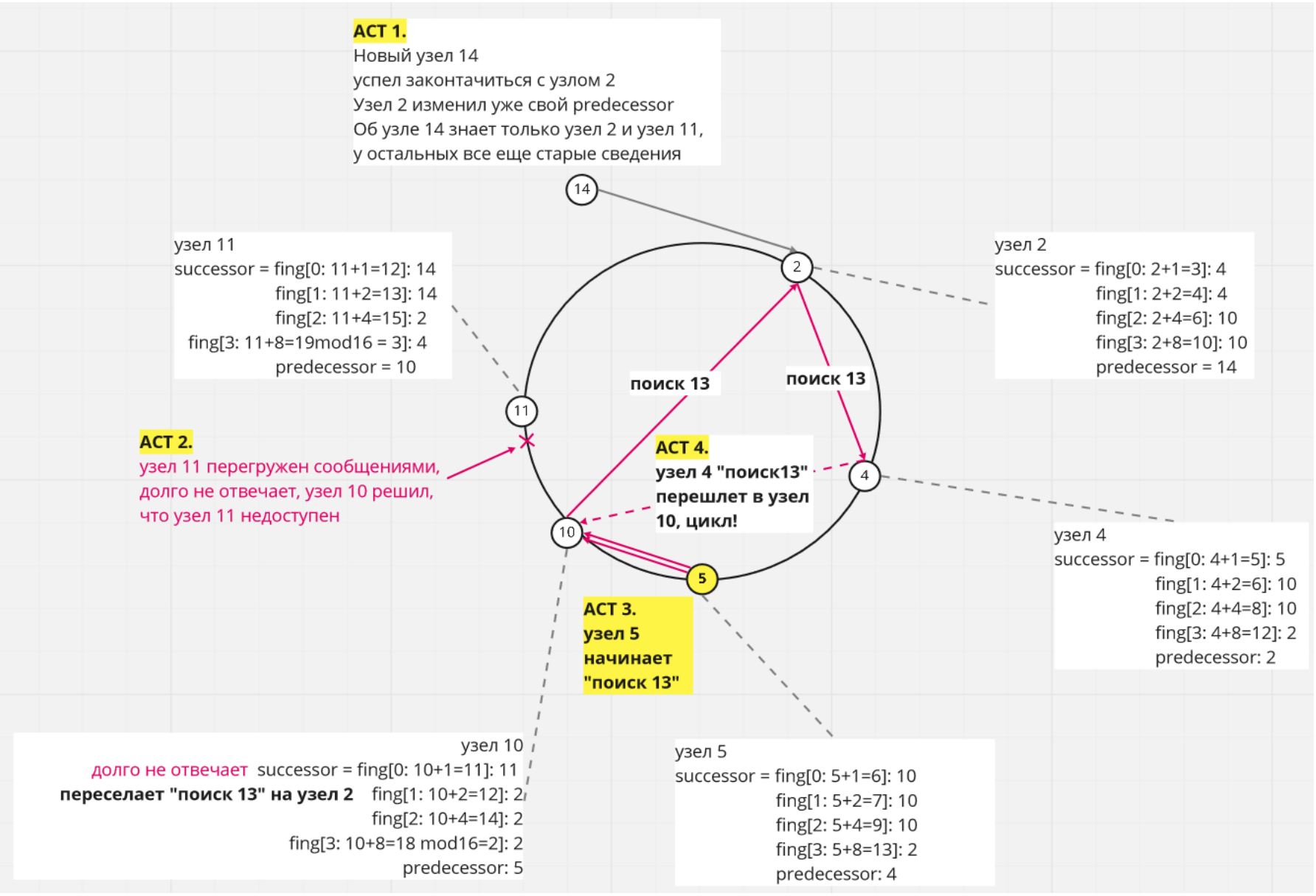
получается id=1 НЕ принадлежит интервалу (4; 18]

Исправлено 

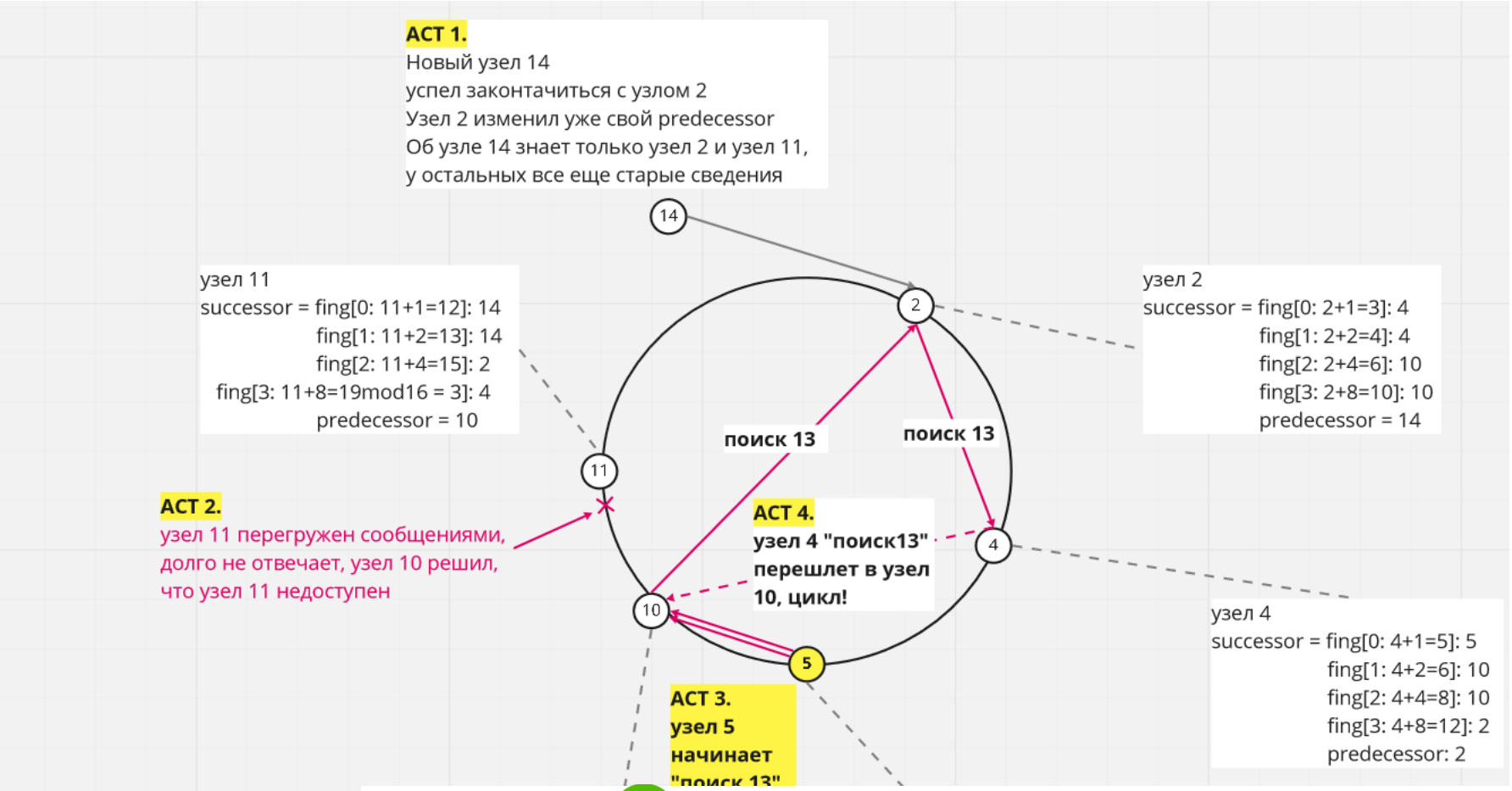
Придуман другой способ проверки (см. функцию isInRangeOverZero в inc.h).

В Хорде на JAVA также не правильно будет проверяться

Проект Хорда. Циклится по внутреннему кольцу



Проект Хорда. Циклится по внутреннему кольцу



Исправлено

1. Добавлен retransmit counter в сообщение. Если сообщение пересылается больше раз, чем размерность logN, то сообщение отбрасывается.
2. 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.

Спасибо за внимание!

Вопросы?