Распределенные системы и технологии.

Peer-to-peer системы

Дмитрий Юрьевич Чалый
декан факультета ИВТ,
зав. кафедрой информационных и сетевых технологий

22 мая 2016 г.

Постановка задачи

Необходимо построить распределенную систему хранения данных:

- масштабируемость;
- отказоустойчивость;
- узлы могут динамически становиться членами системы и выходить из нее.

Как показала практика, это актуально для облачных систем хранения данных (AWS S3), и для специализированных баз данных (AWS DynamoDB).

Napster

Архитектура:

- Серверы хранят только каталог файлов (т.е. где находится файл);
- Пользовательские машины хранят сами файлы;
- Серверы централизованно следят за хостами, которые сейчас онлайн.

Клиенты:

- загружают на серверы информацию о хранимых файлах;
- могут искать файлы по имени;
- загружать файлы с других клиентов с лучшей пропускной способностью.

Napster: достоинства и недостатки

Достоинства:

- простая архитектура.

Недостатки:

- неравномерное распределение нагрузки в системе — нагрузка на серверную часть;
- плохая надежность — любой сбой сервера приводит к проблемам;
- проблемы с безопасностью: все передается в открытом виде;
- Napster оказался ответственным за нарушение прав правообладателей контента.

Gnutella

- Идея: убрать очевидное слабое место — серверы.
- все машины являются равноправными;
- каждая машина хранит файлы...
- ...вместе с указателями на другие известные ей машины сети.

Gnutella: подключение к сети

- Найти адрес узла сети самостоятельно;
- **GnuCache** — база данных узлов сети, входила в поставку ПО;
- можно перехватить пакеты **ping** и **pong** чтобы узнать какие узлы есть в сети.

Gnutella: поиск файлов

- сообщение **Query**: описание файла;
- широковещательная рассылка (размножение запросов);
- ограничение TTL;
- каждый запрос имеет уникальный дескриптор: недопущение повторных широковещательных передач;
- если узел хранит искомые файлы: сообщение **QueryHit**;
- **QueryHit** возвращается по обратному пути.

Gnutella: передача файлов

- После получения ответов **QueryHit** (можно выбрать лучший источник);
- HTTP GET-запрос источнику;
- GET-запрос содержит поле *range*;
- если источник за фаерволлом:
  - запрос **Push**;
  - источник устанавливает TCP-соденение с приемником;
  - HTTP GET-запрос и т.д.

Gnutella: поддержка топологии сети

- широковещательные сообщения **Ping**;
- ответы **Pong** в обратном направлении;
- каждый хост сам решает сколько соседей ему хранить;
- распределение по количеству соседей является степенным:

$$P(\text{количество соседей} = L) = L^{-k}, k - \text{константа}.$$

Gnutella: достоинства и недостатки

Достоинства:

- децентрализованная архитектура;

Недостатки:

- значительную долю трафика составляют сообщения **Ping** и **Pong**;
- повторяющиеся популярные запросы;
- в файлообменных сетях много пользователей, которые только скачивают файлы;
- широковещательные сообщения создают нагрузку на сеть.

FastTrack

- Вводим понятие *репутации* узлов и те, у кого она высокая назначаем *суперузлами*;
- суперузел хранит каталог файлов своего окружения (суть как в Napster);
- когда узел ищет файлы сначала он делает запрос ближайшему суперузлу.

BitTorrent

.torrent-ссылка:

- находим в Интернете/на форуме ссылку на раздачу;
- содержит информацию о трекере и файле;
- устанавливаем связь с трекером.

Трекер:

- следит за входом и выходом узлов на раздачу;
- отдает список узлов, с которых мы можем скачать файл.

BitTorrent: особенности

- Файлы разбиваются на блоки (например, по 256Кб);
- сначала скачивается наименее реплицированный блок;
- предпочтение отдавать блоки узлам, которые лучше их отдают;
- ограничения по отдаче блока соседним узлам (например, не более 5 узлов, случайным образом обновлять это множество узлов).

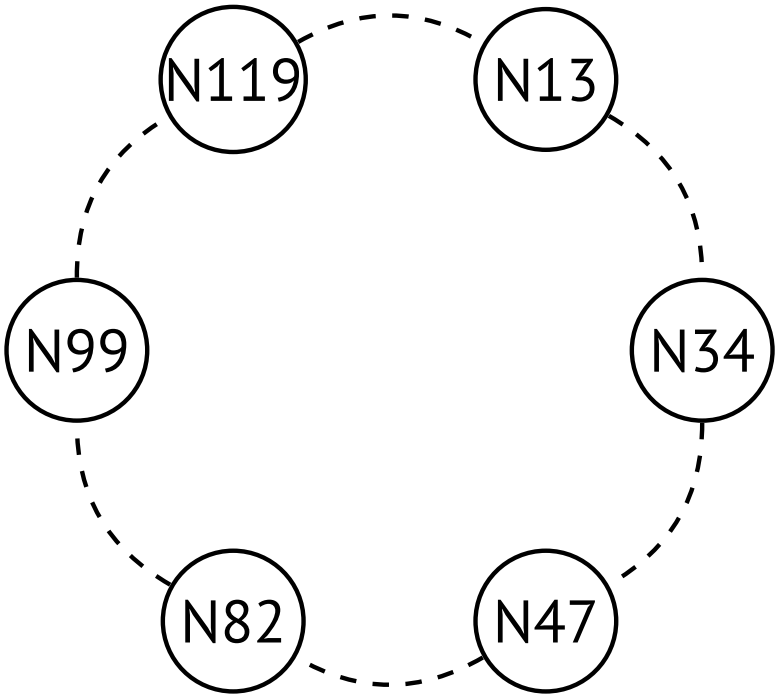
DHT: распределенная хеш-таблица

- Основной функционал: вставка, поиск и удаление ключей (и связанных с ними значений);
- балансировка нагрузки;
- устойчивость от сбоев;
- скорость поиска и вставки ключей;
- локальность.

Chord: адресация

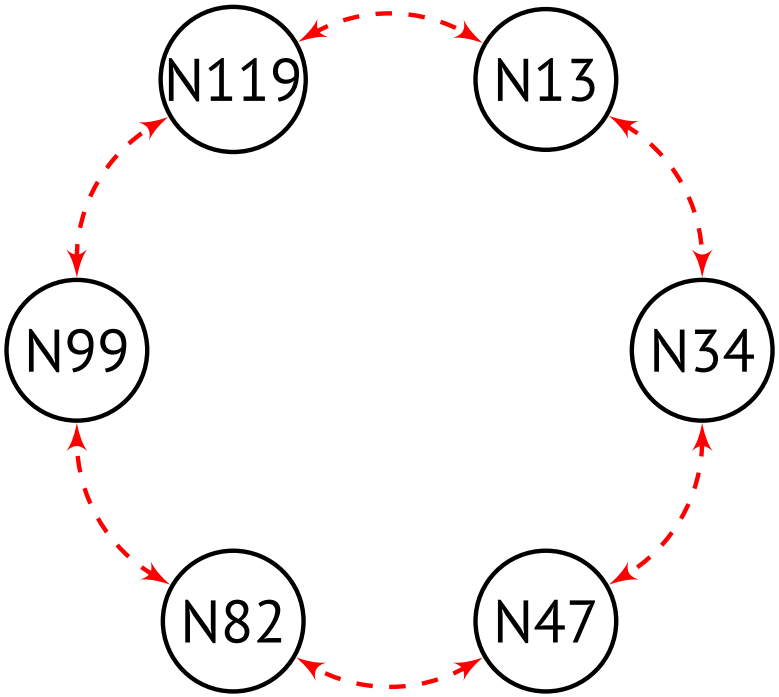
- Отображаем адрес узла при помощи криптографической хеш функции (SHA-1, MD-5) в число;
- младшие m бит этого числа и будут являться адресом узла в Chord;
- всего может быть 2^m узлов, коллизии могут быть, но вероятность полагается малой.

Chord: архитектура



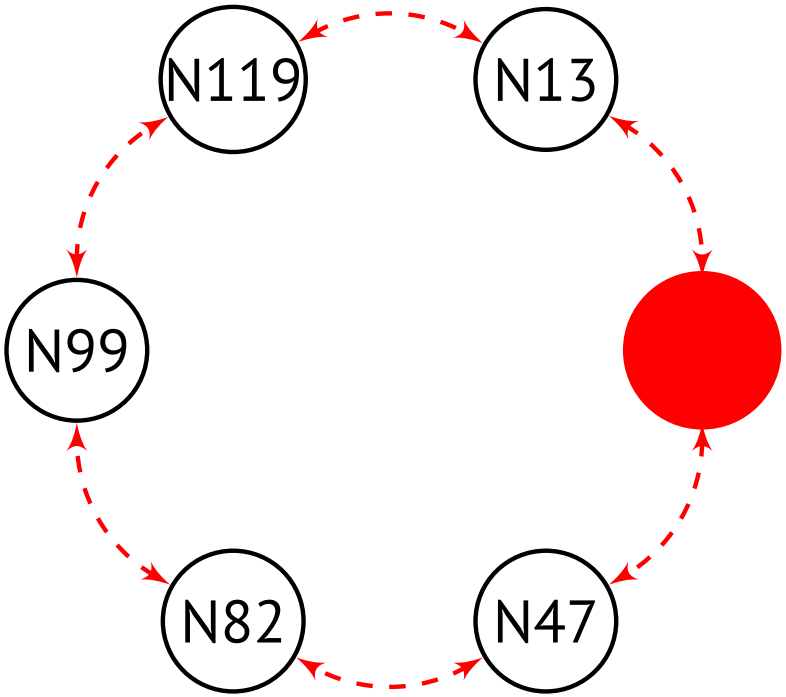
- все узлы располагаются по кругу;

Chord: архитектура



- все узлы располагаются по кругу;
- каждый узел знает о своих соседях;

Chord: архитектура



i	$ft[i]$
0	47
1	47
2	47
3	47
4	82
5	82
6	99

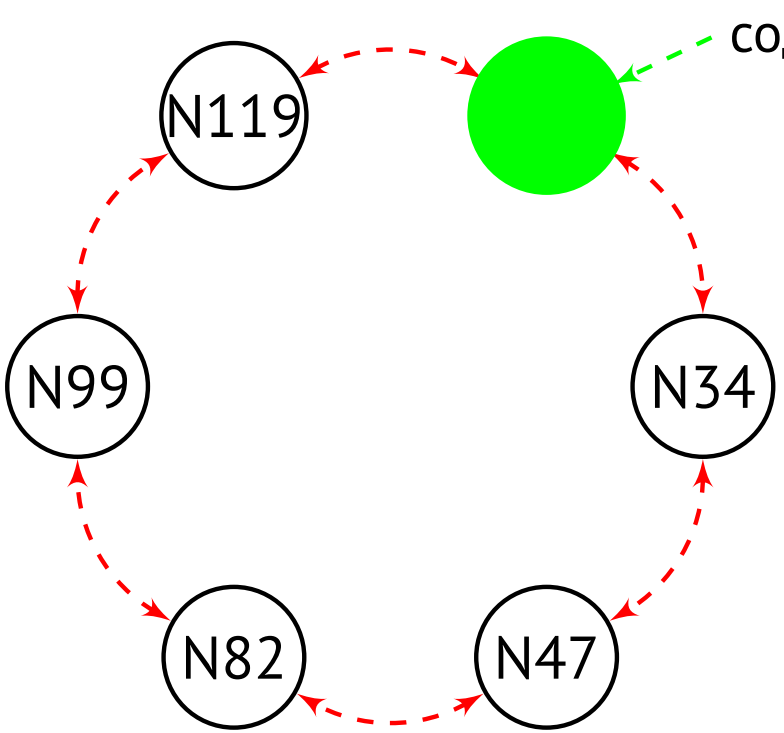
$$ft[i] = id \geqslant (n+2^i) \bmod 2^m$$

- все узлы располагаются по кругу;
- каждый узел знает о своих соседях;
- каждый узел поддерживает finger table.

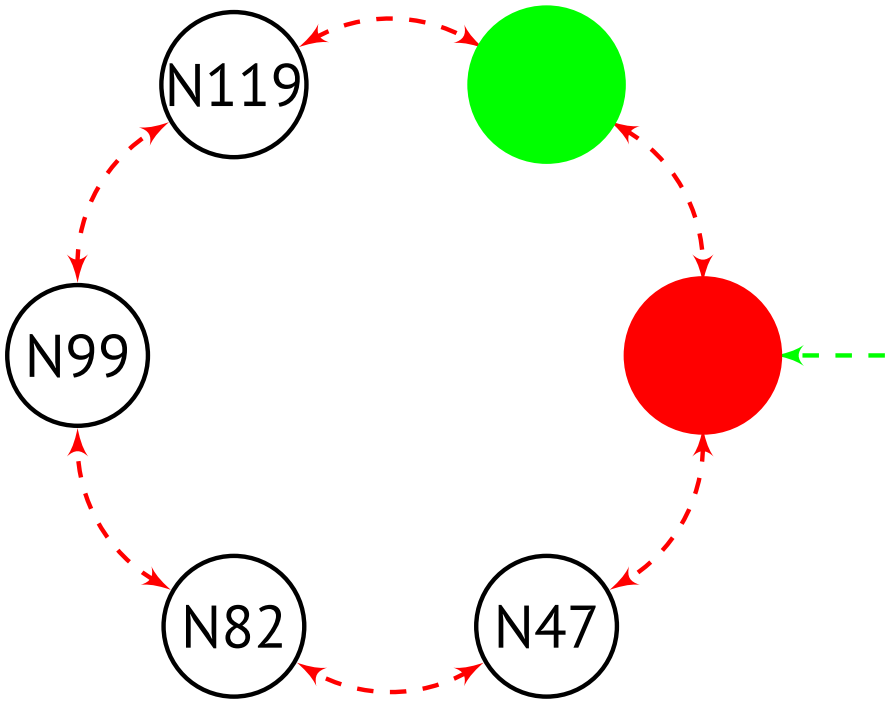
Chord: хранение файлов

- каждый файл отображается так же, как и адрес узла (получаем id файла);
- файл хранится на первом узле, id которого больше id файла;
- если узлов N, а файлов K и хеш функция «хорошая», то каждый узел хранит $O(K/N)$ файлов.

Chord: поиск файлов

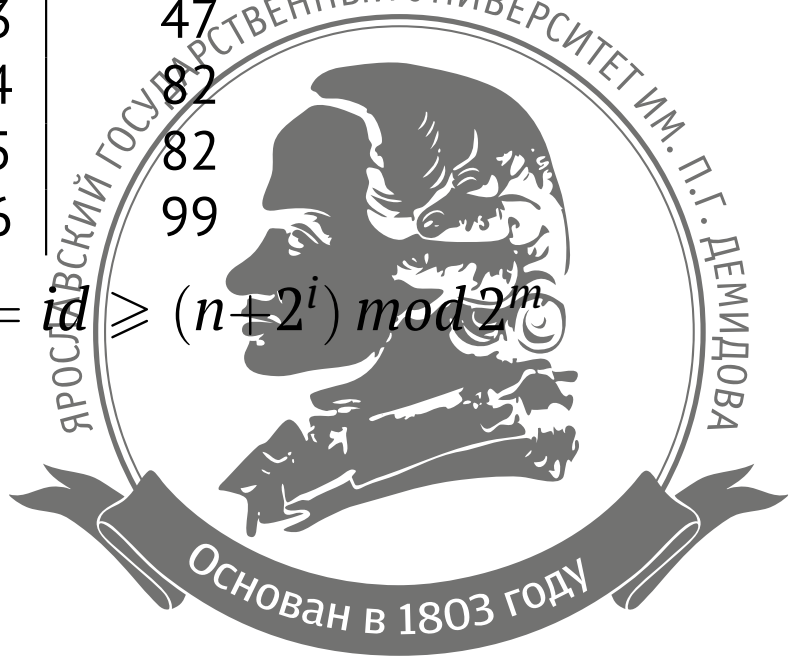


Chord: поиск файлов

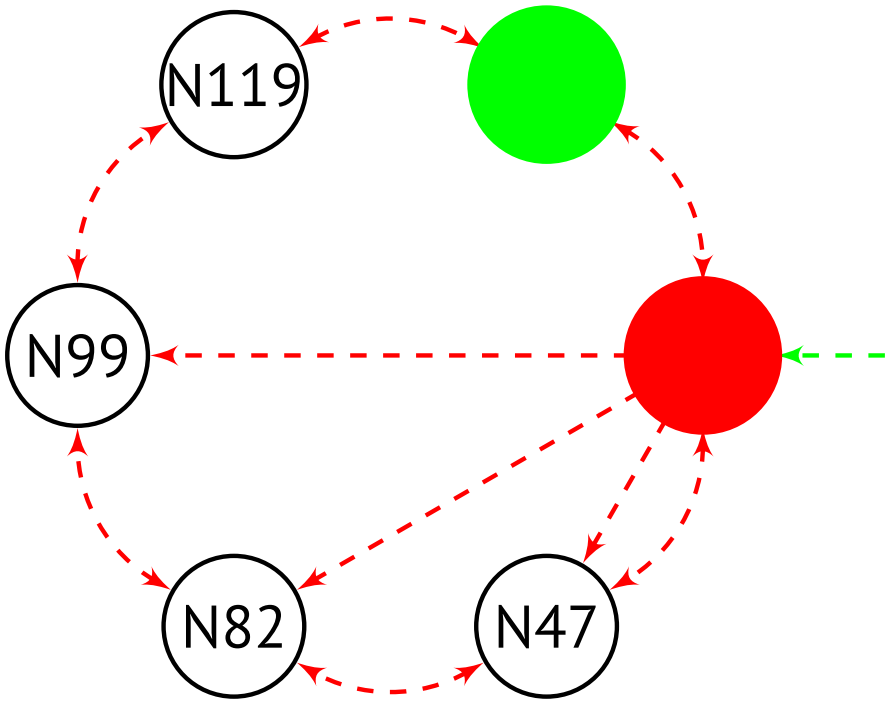


i	$ft[i]$
0	47
1	47
2	47
3	47
4	82
5	82
6	99

$$ft[i] = id \geqslant (n+2^i) \bmod 2^m$$

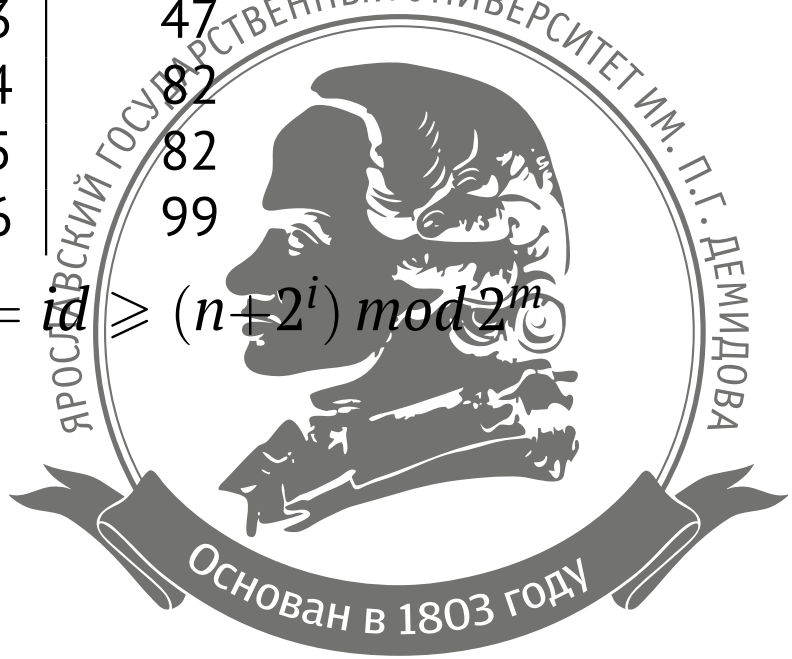


Chord: поиск файлов

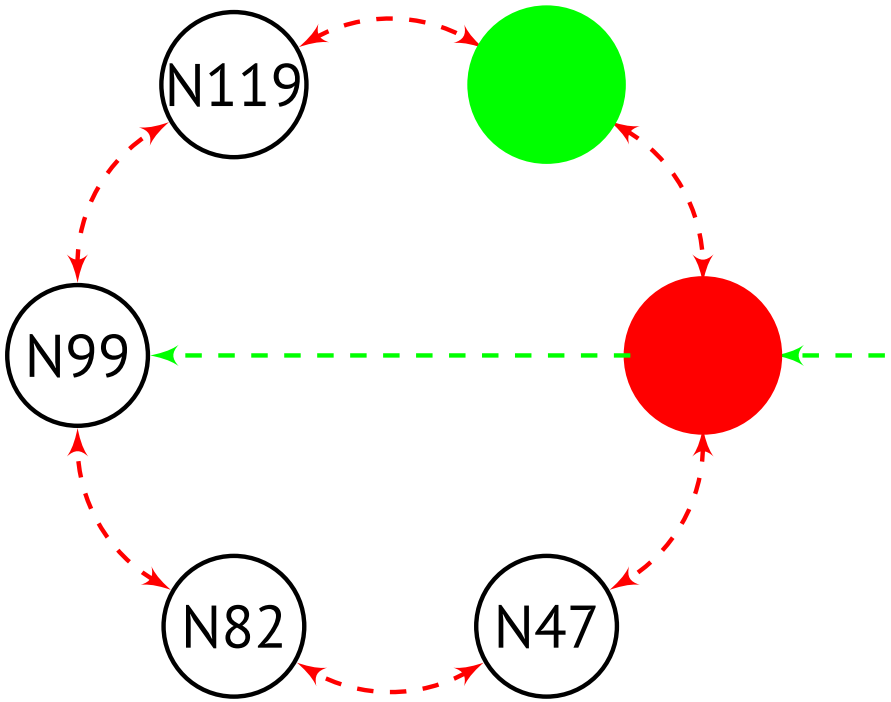


i	$ft[i]$
0	47
1	47
2	47
3	47
4	82
5	82
6	99

$$ft[i] = id \geqslant (n+2^i) \bmod 2^m$$

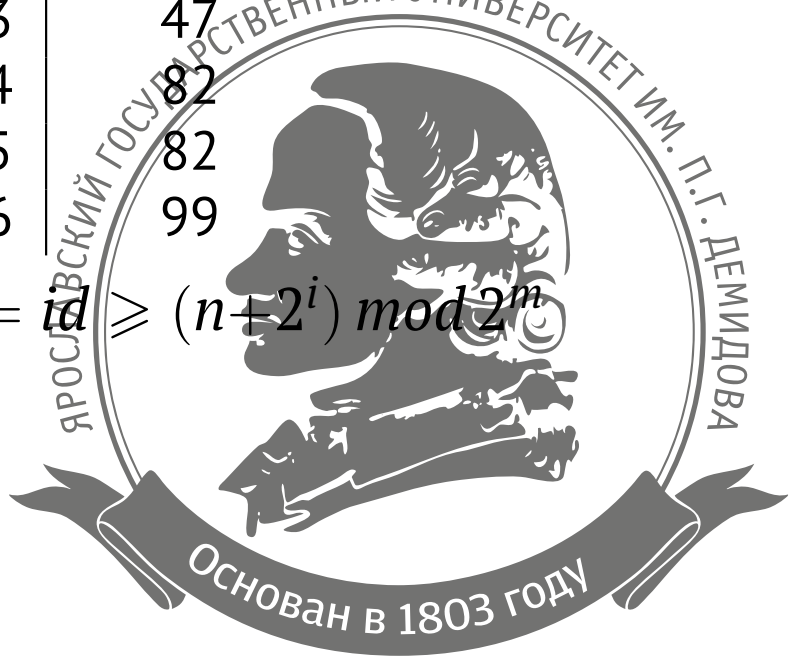


Chord: поиск файлов

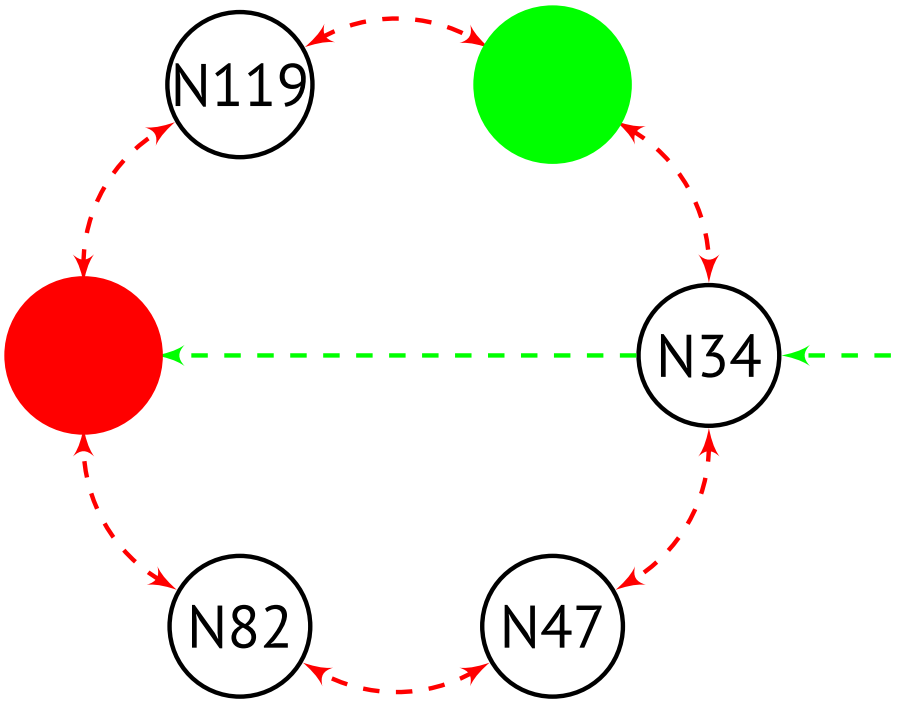


i	$ft[i]$
0	47
1	47
2	47
3	47
4	82
5	82
6	99

$$ft[i] = id \geqslant (n+2^i) \bmod 2^m$$

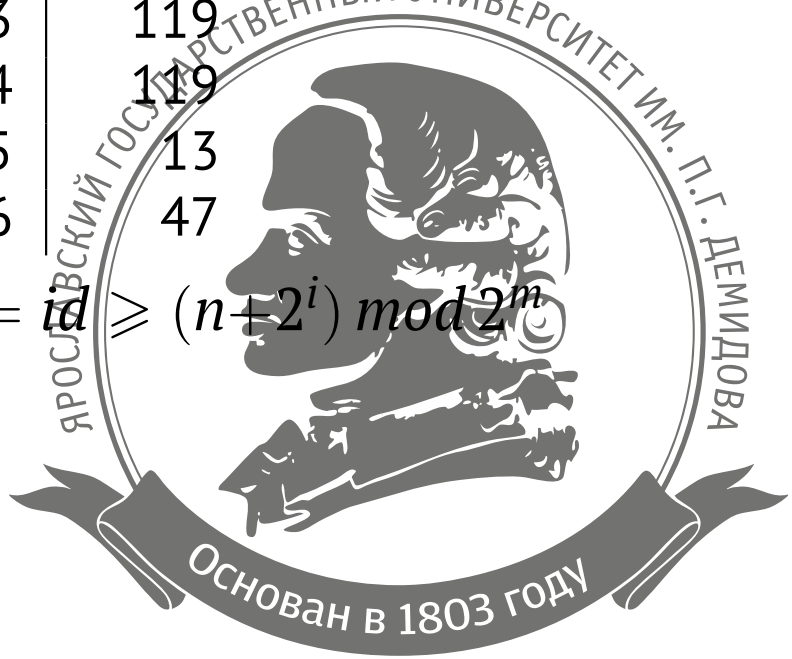


Chord: поиск файлов

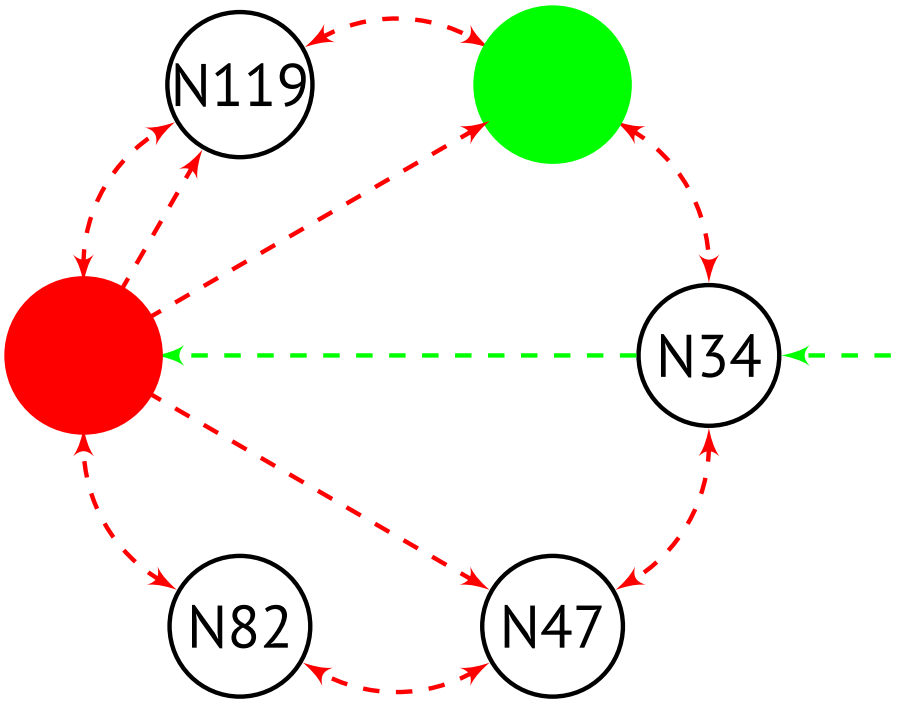


i	$ft[i]$
0	119
1	119
2	119
3	119
4	119
5	13
6	47

$$ft[i] = id \geqslant (n+2^i) \bmod 2^m$$

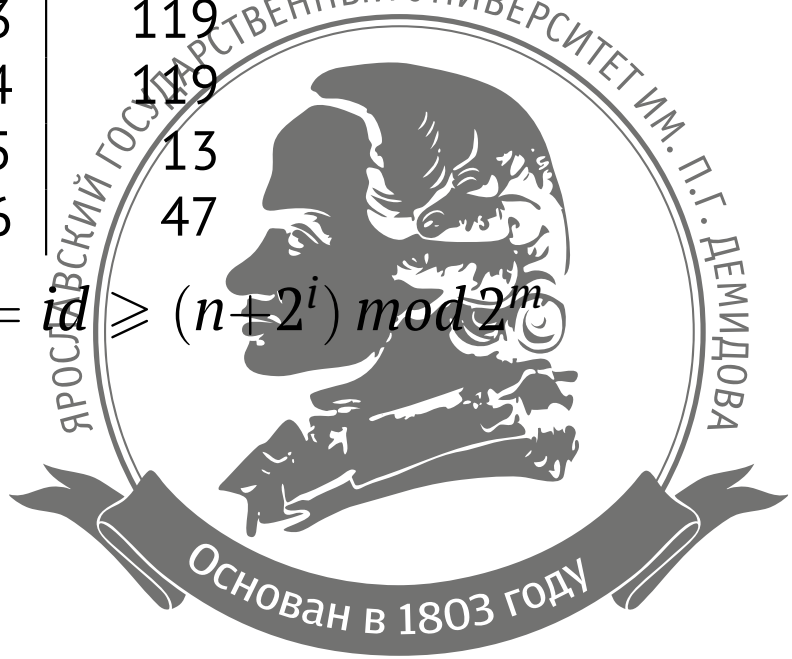


Chord: поиск файлов

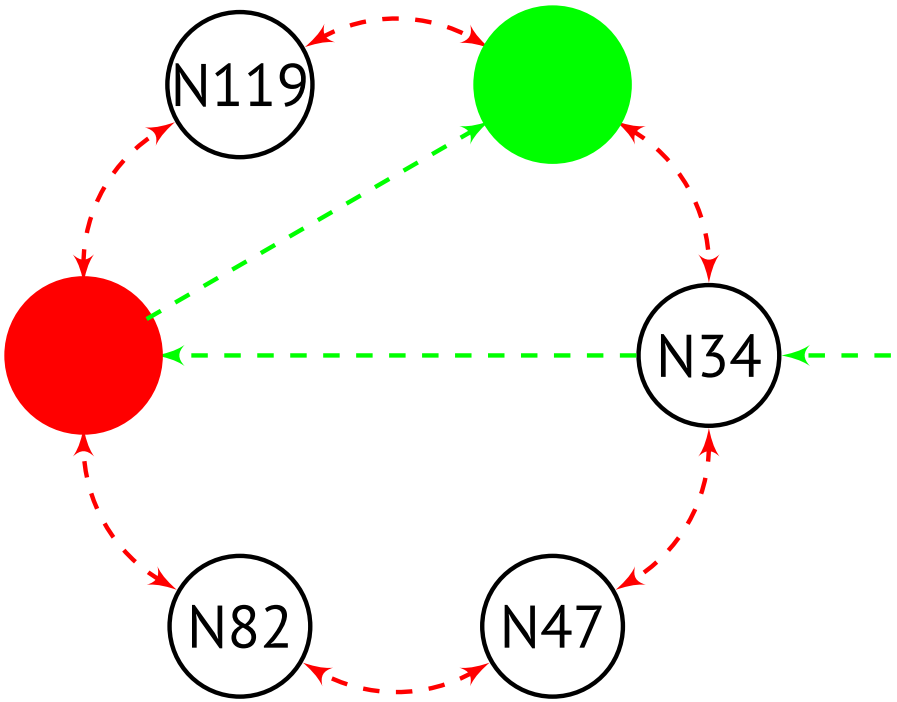


i	$ft[i]$
0	119
1	119
2	119
3	119
4	119
5	13
6	47

$$ft[i] = id \geqslant (n+2^i) \bmod 2^m$$

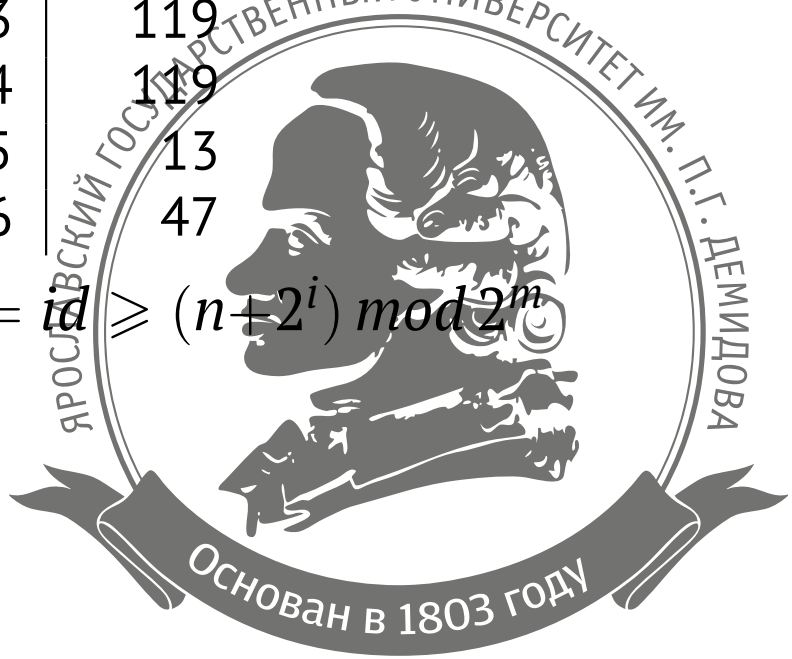


Chord: поиск файлов



i	$ft[i]$
0	119
1	119
2	119
3	119
4	119
5	13
6	47

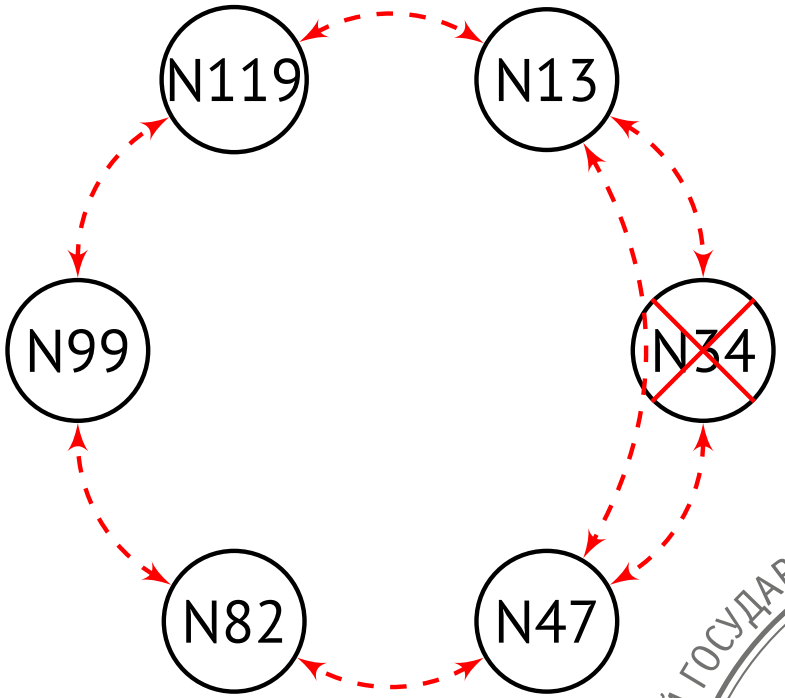
$$ft[i] = id \geqslant (n+2^i) \bmod 2^m$$



Chord: анализ

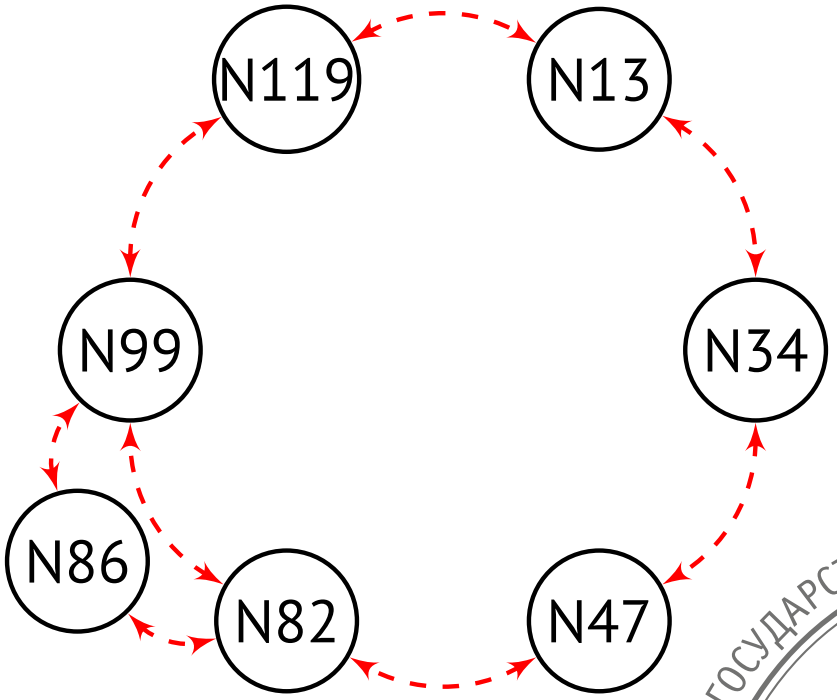
- поиск файла происходит за время $O(N)$;
- аналогична трудоемкость помещения/удаления файла.

Chord: сбои узлов



- хранить не одного последователя, а нескольких (например, $2 \log N$);
- динамически перестраивать кольцо;
- реплицировать файл на соседние узлы.

Chord: новые узлы



- Если появляется узел $N86$, то он представляется узлам $N82$ и $N99$;
- $N86$ обновляет finger table, используя информациюот узла $N99$;
- на $N86$ реплицируются файлы $K86 \ldots K98$.

Chord: динамика входа/выхода узлов

- В распределенных системах узлы входят и выходят из системы часто (25% в Overnet, до 100% в Gnutella за один час);
- большая нагрузка при репликации файлов;
- существенная часть ресурсов требуется для синхронизации finger table.