Methodology for Computer Science Research
Lecture 1: Introduction

Andrey Lukyanenko
Department of Computer Science and Engineering
Aalto University, School of Science
T-110.6130@aalto.fi

September 13, 2012
Course overview

Code: T-110.6130

Name: Methodology for Computer Science Research

Contact: T-110.6130@aalto.fi

Aim: Study of methods, tools, and development of reading and writing skills.

Structure: 6 Method lectures, 2 presentations, half lectures are removed for home study.
To pass the course...

... during the course:

1. Pick *one research topic* of your interest: select one of the provided by us or choose yourself *(be cautious!)*.
2. Attend the lectures: Thu/Tue, 16:15-18:00 at T5  
3. Write diaries after each lecture related to the methods and your topic.
4. Review diaries of others in a week after each lecture.
5. Write an assignment on the topic you chose *(here the diaries could help!)*.
6. Short presentations on your topic 11 Dec and 12 Dec.
7. Checkout English courses in Language Center if you need help?
To pass the course...

... during the course:

Lecture A
Method X

Write diary on your topic using Method X

Review 3 others diaries on their topics using Method X

Lecture B
Method Y

Write diary on your topic using Method Y

Lecture C
Method Z

timeline
Credits and grading

Credits: 5 credits

Grading:

- Diaries give 50% of the mark. \((g_d)\)
- Presentation gives 20% of the mark. \((g_p)\)
- Assignment gives 50% of the mark. \((g_a)\)

The final grade \(g\) will be calculated as

\[
g = \min \left\{ \frac{50 \cdot g_d + 20 \cdot g_p + 50 \cdot g_a}{100}, 5 \right\}.
\]
Assignment topics

Each student have to pick up one topic. During the course produce analysis of the topic it with studied methods.

Topics are...

- Congestion control in TCP.
- Fairness int TCP.
- Distributed Hash Tables (DHTs).
- Unstructured Peer-to-Peer (p2p).
- Cloud computing Systems.
- Mesh Networks.
- Sensor Networks.
- Ad-hoc Networks.
- Social media.
- Delay tolerant networks.
- ...
Assignment topics (cntd)

- Security in DHT.
- Internet of things.
- Datacenter architecture.
- BitTorrent protocol (tit-for-tat).
- Routing protocols in the Internet.
- Publish/Subscribe systems.
- P2P reputation systems.
- Energy consumptions in Wireless systems.
- Denial-of-Service attack.
- Multicast protocols.

OR you can choose your own topic.
It may be what you have as Master Thesis topic, or any topic you are interested in.
Structure of the Course

Course outline:

1) 13.09 Introduction (this lecture).
2) 20.09 Computer Simulation.
3) 04.10 Data analysis.
4) 11.10 Mathematical modeling.
5) 18.10 Academic programming.
6) 30.10 Experimental research.
7) 06.11 Network business models.
8) 10.12 Presentation 1.
9) 11.12 Presentation 2.

Assignment deadline is 01.12.
Studying process (1/3)

- After this lecture pick up own topic or 3 of our topics,
- Send a topic or the list to T-110.6130@aalto.fi with title "T-110.6130 assignment topic" (easier to find).
- Inside e-mail write your own topic; also say few words about the reason why did you choose it,
- or list 3 topics by priority from the provided ones, e.g.

```
“My priority list of topics is:
1. TCP.
2. DHT.
3. DoS attack”
or “Own topic: <Title> (I choose it because it’s my MSc topic)”.
```

- Before lecture 2 you will be provided with a unique topic.
After each methodological lecture (lectures 2-7) you

- write a short diary note (1 page IEEE double column format, no need to write too much, no need for introduction, title or conclusion; see Diary Instructions in Noppa.)
- upload (pdf and tex) the diary to optima.aalto.fi

When logged in to optima, you will find T-110.6130 workspace with diary subsection. The diary on previously studied method should be uploaded before next lecture (or during one week)

- review 3 other’s diaries from previous lectures (Diary grade is a combination of your diary quality and your review quality; Reviews are given as comments in optima.)

In this short diary you write how to use exactly given research method for your topic.

Warning: Avoid unnecessary information in diaries. Abstract, Introduction and Conclusion will be only in final assignment.
Studying process (3/3)

- Last 2 lectures are presentations. All students will have short presentations (~5min) on what they have studied during the course on their selected topic.

- One week before the presentations is an assignment deadline. The assignment is to cover the topic you choose with methodological view.

Your paper should:

- contain a short introduction to the topic,
- clearly state all methods used to study the topic in literature,
- compare them (pro and con),
- present own thoughts: what in the study is missing and why?

Remember: Your diaries on the same topic will help you with the final assignment!
What is this course about?

This course is about Scientific Research in the field of Computer Science (more precisely, in the field of Data Communications).

The course tries to answer on the questions:

- How to do the Scientific Research?
- How to do the Scientific Research efficiently?
- How to do what a Scientific Community needs, in the form which the Scientific Community demands?
- How to present your Scientific Research to the Community?

Although, the above in context of Scientific Research, the same skills are useful in any kind of IT related work.
What is Computer Science Research?

It is about studying an Idea: your Idea.

**Novelty** of the Idea.
Research is a study of new ideas in the field where the research belongs to.

**Significance** for the Community.
One of the most important questions of research is to understand what kind of idea is actually needed for the community “today”.

**Contribution** from the Researcher.
The amount of efforts made by a researcher to study the idea.

But before...
But before...

... understanding *Novelty* and *Significance* you have to know the state-of-the-art of knowledge in Scientific Community.

**How to be up-to-date?**

1. **Read** recent journal articles, and conference papers. Almost all of them has “History”, “Introduction” and “Future work” parts. (they correspond to “Past”, “Current” and “Possible Future” of the research.)

2. **Talk** to colleagues and scientific advisers :) (they may suggest ideas and explain the field development, without studying).

3. **Observe** the business tendency and technology levels (news from industry).

4. **Look through** the visions of the future (Sometimes knowledgeable people publish their visions of the future).
Literature sources

The search engines (and sources) for scientific publications.

- Google Scholar: http://scholar.google.com
- Academic Microsoft: http://academic.research.microsoft.com
- ACM Portal: http://portal.acm.org
- IEEE xplore: http://ieeexplore.ieee.org

Especially, papers published in famous conferences, e.g.,

- ACM SIGCOMM: http://www.sigcomm.org
- INFOCOM: http://www.ieee-infocom.org

Additionally, many famous publications appear in less famous, but still important conferences.

AR - acceptance rates for the conferences and IF - impact factor for the journals.
Accessing the publications

1. Traditional way: Go to the library and get an article or order one (an obsoleted way).

   Unfortunately, the articles and conference books in the library are quite old. Some journals are available in the coffee room.


3. Remotely: Outside Aalto University you can fetch them
   - directly from the Internet, some of them are publicly available
   - indirectly using the search site nelliportaali.fi or adding the proxy libproxy.aalto.fi, e.g. portal.acm.org.libproxy.aalto.fi
Reading as a part of Research

The reading refers to the studying of the field (remember *Significance* and *Novelty*?).

Reading:

▶ adds knowledge about the field.
▶ adds the confidence in own knowledge about the field.
▶ helps new Research Ideas to pop up in the mind.

Do not underestimate the Reading as a part of Research:

▶ Even if you have the full confidence in the new Idea, check the literature, search for it.
▶ If the Idea popped up after reading some paper, check who citing this paper. May be the Idea was already developed.

Remember: the previously mentioned paper search engines are able to search by criteria: “cited by”.
Writing as a part of Research

The writing refers to the production of own Research (recall Contribution!).

Writing:

▶ allows you to document your work for own needs.
▶ allows others to see your work, to see that you are actually working.
▶ putting Ideas on a paper allows to polish them and even invent new or extend existing Idea.

Writing is always hard in the middle of research, but it will greatly help you later if you put on the paper even small Ideas, points, thoughts.
Reeding ↔ Writing

Question: When should I switch from reading to writing?

Answer: Never.

- Starting the research you mainly read.
- Finishing the research you mainly write.
- In between, you write, but continue to keep abreast of the development of the Community.

Conferences happen all the time, papers appear. If you produce your research based on other authors’ paper, always check who is citing it.

Question: When to switch from mainly reading to mainly writing?

Answer: Whenever you have confidence in the field and double checked the Idea.
What is an outcome of the Research?

Accomplished research is determined by *written results*.

**Outcome** of the Research may be

- a survey of the field, if it is necessary overview, timely and shows new facets of the field,
- a new algorithm/protocol, if it gives some benefits compared to already existing ones,
- a mathematical model of a protocol/algorithm, if it is better predicts different features of the protocol/algorithm,
- a performance measurement of existing protocols, with additional analysis
- many more...

All these are “Scientific Findings”. Your *written* results should *address* it clearly.
How to develop a new “clever” idea?

There is no rule for Idea generation process, but when you have an Idea remember:

“There is nothing new except what is forgotten” (c) Rose Bertin

- Whenever you have a new idea, double check that it was not studied previously. Even in close/related fields.
- It can be an old idea from a different field, but was forgotten and the time for it has come.

The checking allows to skip waste of time to study something that already was studied and will concentrate on application of the idea to the field.
There is a big difference what you see when you look inside your paper, and what others see!

Good to know from the start: Peer Review a common practice in the Scientific Community.

An Example: Peer-review process in SIGCOMM:

Find and read the paper yourself, you know how and also read the one which is citing it!!
TPC Review System (An Example)

Traditional points of review process:

*** Contribution: Rate the evaluation of work and contribution.
*** Significance: Rate the significance to theory and practice.
*** Novelty: Rate the originality and novelty.
*** Relevance: How relevant is the paper to the call for papers?
*** Readability: Rate the readability and organization of the content.
*** Overall recommendation: Would you recommend this paper for conference?
*** Best paper award: Do you consider the paper a candidate for a best-paper award?
*** Detailed comments
When you are finally ready!

Ready?

- You have the Idea.
- You have the confidence in the idea (novelty and significance).
- You need Contribution!!

*This is what the course is about.*

The optimal process of the idea study is not unique and is fully dependent on the case, however, it has a set of known study methods:

- Mathematical Modeling.
- Computer Simulation.
- Experimental research.
- Data Analysis.
- Software Development (demo or product).
Mathematical modeling is a research method that performs the problem abstraction, when different properties of a system are defined using a set of parameters and interactions of these properties are defined with functions and inequalities over the parameters.

Mathematical modeling provides a set of features, it allows to

► investigate properties of the whole system, based on a subset of measured parameters.

► see the system’s asymptotic behavior.

► find optimal conditions for a system.
Methods: Computer Simulation

*Computer Simulation* is a research methods, when a small sample program representing the studying algorithm/protocol is created for an existing toolkit (or seldom from the scratch), which simulates the network, computer or system work.

Computer Simulation allows to

- produce “cheap” evaluation research.
- do research, when the development time is crucial.
- do research, when the sources (money, number of devices) are limited.
- do research even in the black-box architecture, real development is limited.
Experimental research is a research method, which is mainly based on an experimentation.

Experimental research allows to

- produce a research even in case if the modeling is difficult.
- acquire result, when a simulation may be a very slow process.
- see non-trivial dependencies between parameters.
Software development is a part of the research, when a product-like software (demo) is produced. Sometimes it is even in the form of commercial product, i.e., this methods allows to show that the research idea is fully feasible.

Software development allows to:

- create a proof-of-concept.
- see design errors/pitfalls in the idea.
- produce the research with most realistic environment.
Methods: Data Analysis

Data Analysis is a research method, that in a form of a bridge, connects together other methods, allowing to compare results, make the research consistent in different aspects and produce estimations for parameters.

Data Analysis allows to

- match mathematical models and experimental research.
- give estimations on the parameters for models and algorithms.
- prove some properties with high probability rates.
- provide reader with easily readable and understandable data.
Networking business research methods are the research methods examines telecommunications from the business point of view.

Networking business methods allows to

- study commercialization of the research.
- focus on the factors affecting the commercial success.
- predict future trends of the research and industry.
Iterative process: Big Cycles

1. Mathematical Modeling
2. Computer Simulation
3. Data Analysis
4. Experimental research
5. Research
Iterative process: Short Cycles

- Mathematical Modeling
- Research
- Computer Simulation
Iterative process: with paper outcomes

- Experimental research
- Mathematical Modeling
- Data Analysis
- Research
- Computer Simulation

Updates Experiments Section
Updates Model Section
Updates Analysis Section
Updates Simulation Section
On the paper.

<table>
<thead>
<tr>
<th>Section</th>
<th>Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>Title</td>
</tr>
<tr>
<td>-</td>
<td>Abstract</td>
</tr>
<tr>
<td>1</td>
<td>Introduction</td>
</tr>
<tr>
<td>2</td>
<td>History (Related work)</td>
</tr>
<tr>
<td>3</td>
<td>Idea, Algorithm</td>
</tr>
<tr>
<td>4</td>
<td>Model</td>
</tr>
<tr>
<td>5</td>
<td>Simulation, Measurements</td>
</tr>
<tr>
<td>6</td>
<td>Evaluation, Data Analysis</td>
</tr>
<tr>
<td>7</td>
<td>Implementation (Demo)</td>
</tr>
<tr>
<td>8</td>
<td>Discussion (Results), Future work</td>
</tr>
<tr>
<td>9</td>
<td>Conclusion</td>
</tr>
<tr>
<td>-</td>
<td>Reference</td>
</tr>
</tbody>
</table>
Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications

Ioan Stoica, Robert Morris, David Karger, M. Frans Kaashoek, Hari Balakrishnan
MIT Laboratory for Computer Science
chord@lcs.mit.edu
http://lcs.cs.mit.edu/chord/

Abstract

A fundamental problem that confronts peer-to-peer applications is to efficiently locate the node that stores a particular data item. This paper presents Chord, a distributed lookup protocol that addresses this problem. Chord provides support for just one operation: given a key, it maps the key onto a node. This location can be easily implemented on top of Chord by associating a key with each data item, and routing the key-data item pair at the node to which the key maps. Chord scales efficiently as nodes join and leave the system, and can survive quorums even if the system is continuously changing. Results from theoretical analysis, simulation, and experiments show that Chord is scalable, with communication cost and the state maintained by each node scaling logarithmically with the number of Chord nodes.

1. Introduction

Peer-to-peer systems and applications are distributed systems without any central control or hierarchical organization, where the software running on each node is equivalent in functionality. A review of the features of recent peer-to-peer applications yields a long list: redundant storage, permanence, reduction of server load, anonymity, search, authentication, and hierarchical naming. Despite this rich set of features, the core operation in most peer-to-peer systems is efficient location of data items. The contribution of this paper is a scalable protocol for looking up a key in a dynamic peer-to-peer system with frequent node arrivals and departures.

The Chord protocol supports just one operation: given a key, it maps the key onto a node. Depending on the application using Chord, that node might be responsible for storing a value associated with the key. Chord uses a variant of consistent hashing [11] to assign keys to Chord nodes. Consistent hashing tends to balance load, since each node receives roughly the same number of keys.

2. Related Work

While Chord maps keys onto nodes, traditional name and location services provide a direct mapping between keys and values. A value can be added, a document, or an arbitrary data item. Chord can easily implement this functionality by storing each key-value pair at the node to which that key maps. For this reason and to make the comparisons cleaner, the rest of this section assumes a Chord-based service that maps keys onto values.

DNS provides a hint to IP address mapping [12]. Chord can provide the same service with the same representing the key and the associated IP address representing the value. Chord requires no special servers, while DNS relies on a set of special root...
On the paper: Abstract.

Abstract:
When somebody finds out this article this is what is read first. Based on abstract a person should be able to preliminary know, should he/she read it further or not?

Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications

Ion Stoica; Robert Morris, David Karger, M. Frans Kaashoek, Hari Balakrishnan
MIT Laboratory for Computer Science
http://lcs.mit.edu/chord/

Abstract
A fundamental problem that confronts peer-to-peer applications is to efficiently locate the node that stores a particular data item. This paper presents Chord, a distributed lookup protocol that addresses this problem. Chord provides support for join and departure operations, issues a key to each node, maps the key to a node, and determines the location of data items. Chord's design is simple, scalable, and robust. A Chord node uses a consistent hash key to store a node's position in the system. The lookup operation, which is performed using a lookup protocol, consists of finding the node whose key is closest to the requested key. The protocol involves a small number of network messages and is robust to node failures.

1. Introduction

Chord is a distributed lookup service for peer-to-peer systems. It provides a simple and robust solution to the problem of locating data items in a distributed system. The key idea behind Chord is to use a consistent hash key to store a node's position in the system. This allows nodes to easily find the node responsible for storing a particular data item.

2. Related Work

While Chord maps keys onto nodes, traditional name and location services provide a direct mapping between keys and values. A value can be addressed, a document, or an arbitrary data item. Chord can easily implement this functionality by storing each key-value pair at the node to which it maps.
On the paper: Introduction.

Introduction describes more precisely current state of the research field, significance of the paper and relevance to it, gives some basic assumptions and limitations, as well as briefly outlines own achievements.
Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications

Ioannis Stocia, Robert Morris, David Karger, M. Frans Kaashoek, Hari Balakrishnan
MIT Laboratory for Computer Science
https://ocw.mit.edu/chord/

Abstract
A fundamental problem that confronts peer-to-peer applications is to efficiently locate the node that stores a particular data item. This paper presents Chord, a distributed lookup protocol that addresses this problem. Chord provides support for fast one-operation, given a key, it maps the key onto a node. Data location can be easily implemented on top of Chord by associating a key with each data item, and routing the key-data item pair at the node to which the key maps. Chord exhibits efficient node join and leave of systems, and can maintain queries even if the system is continuously changing. Results from theoretical analysis, simulation, and experiments show that Chord is scalable, with communication cost and the rate maintained by each node scaling logarithmically with the number of Chord nodes.

Introduction
Peer-to-peer systems and applications are distributed systems without any centralized control or hierarchical organizations, where the software residing on each node is equivalent in functionality. A review of the features of recent peer-to-peer applications yields a long list of redundant storage, persistence, replication of system, servers, anonymity, search, authentication, and hierarchical routing. Despite this rich set of features, the core operation in most peer-to-peer systems is efficient location of data items. The contribution of this paper is a scalable protocol for lookup in a dynamic peer-to-peer system with frequent node arrivals and departures.

The first part of Introduction is hardest part, it always is difficult to find what words/sentence to use in the beginning.
Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications

Ion Stoica; Robert Morris, David Karger, M. Frans Kaashoek, Hari Balakrishnan
MIT Laboratory for Computer Science
http://dbis.ics.mit.edu/chord/

Abstract
A fundamental problem that confronts peer-to-peer applications is to efficiently locate the node that stores a particular data item. This paper presents Chord, a distributed lookup protocol that addresses this problem. Chord provides support for fast and efficient operations, given a key, it maps the key onto a node. Data locations can be rapidly implemented on top of Chord by associating a key with each data item, and routing the key-data item pair to the node that stores the key. Chord solves efficiently in nodes just and keeps the system, and can survive queries even if the system is dynamically changing. Results from theoretical analysis, simulation, and experiments show that Chord is scalable, with communication cost and the time estimated by each node, and the routing table is distributed, a node receives the hash function communicating with a few other nodes. In the steady state, each node maintains information only about other nodes, and receives all lookups that pass through messages to other nodes. Chord maintains its routing information in space and leaves the system, with high probability each node receives relatively little movement of keys, and hence can

1. Introduction
Peer-to-peer systems and applications are distributed systems without any centralized control or hierarchical organization, where the software running at each node is equivalent in functionality. A survey of the features of recent peer-to-peer applications yields a long list: redundant storage, permanence, reduction of network traffic, anonymity, search, subscription, and hierarchal routing. Despite the rich set of features, core operation in most peer-to-peer systems is efficient location of data items. The contributions of this paper is a scalable protocol for looking up a dynamic data in a distributed system. The Chord protocol supports fast one-operation, given a key it maps the key onto a node. Depending on the applications used, Chord that node might be responsible for storing a set of data items. Chord uses a network of nodes, where each node is responsible for a range of keys, hence each node receives roughly the same number of keys.

2. Related Work
While Chord maps keys onto nodes, traditional name and location service provide a direct mapping between keys and values. A value can be an address, a document, or an arbitrary data item. Chord can easily implement this functionality by storing each key-value pair at the node to which that key maps. For this reason, and to make the comparison clear, the rest of this section assumes a Chord-based service that maps keys onto values.

Normally, the first sentence of each paragraph is general sentence, which includes all the knowledge, while the remaining parts are details, which provides the ground for the first sentence.

When you start writing an article, you can write down a set of general sentences and organize the article structure based on them, and later provide specific details about those.
On the paper: Related work.

Related work section contains the most significant knowledge a researcher have discovered concerning the topic.

It is the starting point of any article or any scientific work. You can start this section even during when you still study your field. It shows that you well aware of the topic.
On the paper: Finalizing the paper.

Figure 11: Looking latency on the Internet, percentage, as a function of the total number of nodes. Each of the ten physical sites runs multiple independent copies of the Chord node software.

copies of the Chord software at each site. This is different from running (N/10) 10 virtual nodes at each site to provide good load balance, either the intention is to measure how well our implementation scales even though we do not have more than a small number of deployed nodes.

For each number of nodes shown in Figure 11, each physical site runs 10 Chord lookups for randomly chosen keys one-by-one. The graph plots the median, the 50th, and the 95th percentile of looking latency. The median latency ranges from 190 to 285 ms, depending on number of nodes. For the case of 100 nodes, a typical lookup involves three two-way message exchanges: one for the Chord lookup, another to find the nearest node. Typical round-trip delays between sites are 50 milliseconds (as measured by ping). Thus the expected lookup time for 100 nodes is about 300 milliseconds, which is close to the measured mean of 285.

The low 25th percentile latencies are caused by lookups for keys close (in ID space) to the querying node and for query keys that remain local to the physical site. The high 95th percentile latencies are caused by lookups whose keys follow long delay paths.

The lesson from Figure 11 is that looking latency grows slowly with the total number of nodes, confirming the simulation results that demonstrates Chord’s scalability.

7. Future Work

Based on our experience with the prototype mentioned in Section 6.6, we would like to improve the Chord design in the following areas:

Chord currently has no specific mechanisms to heal partitioned rings; such rings could appear locally consistent to the stabilization procedure. One way to check global consistency is for each node to periodically ask other nodes to do a Chord lookup for a; if the lookup does not yield node a, then node a may be a partition. This will only detect partitions whereas nodes know of each other. One way to obtain knowledge is for every node to know of some small set of initial nodes. Another approach might be for nodes to maintain long-term memory of a random set of nodes they have encountered in the past; if a partition forms, the random sets in one partition are likely to include nodes from the other partition.

A malicious or buggy set of Chord participants could present an incorrect view of the Chord ring. Assuming that the data Chord is being used to locate is cryptographically authenticated, then it is a threat to availability of data rather than to authenticity. The same approach used above to detect partitions could help victim realize that the site is not using a globally consistent view of the Chord ring.

An attacker could target a particular data item by inserting a node into the Chord ring with an ID immediately following the item’s key, and having the node return errors when asked to retrieve the item. Encrypting and checking the node's ID 2D derived from the SHU-1 hash of the IP addresses makes this attack harder.

Even though messages per lookup may be too many for some applications of Chord, especially if each message must be sent to a random Internet host, instead of placing its fingers at distances that are all integer powers of 1/10, Chord could easily be changed to place its fingers at distances that are all integer powers of 1/10. Under such a scheme, a neighboring hop could be decoded from the distance to a query in (m+1)/10 of the original distance, meaning that (m+1)/10 logs would suffice. However, the number of fingers needed would increase to log N/(log 10 + 1/10) = log N / log 10.

A different approach to improving lookup latency might be to use server selection. Each finger table entry could point to the first 6 nodes in the entry’s interval; on the ID ring, and a node could measure the network delay to each of the 6 nodes. The 6 nodes are generally equivalent for purposes of lookup, so a node could forward lookups to one with lowest delay. This approach would be more effective with recursive Chord lookups, in which the node making the delay is also the node forwarding the lookup.

8. Conclusion

Many distributed peer-to-peer applications need to determine the node that stores a data item. The Chord protocol solves this challenging problem in decentralized manner. It offers a powerful primitives: given a key, it determines the node responsible for storing the key’s value and does so efficiently. In the steady state, an N-node network, each node maintains routing information for only about O(log 2N) other nodes, and resolves all lookups via just O(log 2N) messages to other nodes. Updates to the routing information for nodes leaving and joining require only O(N log 2N) messages.

Chord’s attractive features include its simplicity, provide connectivity, and provide performance even in the face of active node service and departures. It continues to function correctly, albeit in a degraded performance, when a node’s information is only partially correct. We have performed analysis, simulations, and experimental results confirm that Chord scales well with the number of nodes, reduces overheads of name independence node failures and joins, and avoids most lookups even during recovery.

We believe that Chord will be a valuable component for peer-to-peer, large-scale distributed applications such as cooperative file sharing, time-shared available storage systems, distributed indices for document and service discovery, and large-scale distributed computing platforms.

Acknowledgements:
We thank Frank Dehne for the measurements of the Chord prototype described in Section 6.6, and David Anderson for setting up the network used in those measurements.

9. References


On the paper: Discussion.

In Discussion or Future Work section you touch the important questions related to the work, which though stayed out of the scope of the paper are still important to mention explicitly.

Additionally, here you can discuss extensions, limitations and problems of the paper, and how to improve or solve them in the future.

You can strengthen own paper, if you add in this section solutions for possible flaws of your work. This will show to the reviewers that you are aware of these limitations, and at least know how to extend the work to overcome them, or why those scenarios are insignificant.

7. Future Work

Based on our experience with the prototype mentioned in Section 6.6, we would like to improve the Chord design in the following ways:

Client efficiency has no specific mechanisms to handle partitioned views; such maps could appear locally consistent to the stabilization process. One way to check global consistency is for each node to periodically ask other nodes to do a Chord lookup for c, if the lookup does not yield node c, there may be a partition. This will only detect partitions where nodes know of each other. One way to obtain knowledge of the entire node is to know of the same small set of initial nodes. Another approach might be to ask nodes to maintain long-term memory of a node set of nodes they have encountered in the past, if a partition forms, the random sets in one partition are likely to include nodes from the other partitions. A malicious or buggy set of Chord participants could present an incorrect view of the Chord ring. Assuming that the data Chord is being used to locate is cryptographically authenticated, this is a threat to availability of data from Slam to Slam. The same approach used above to detect partitions could help victims realize that they are not seeing a globally consistent view of the Chord ring.

A different approach to improving lookup latency might be to use server selection. Each finger can be used to point to the first k nodes in the entry's internal on 10 mg, and a node could measure the network delay to each of the k nodes. The k nodes are generally equivalent for purposes of lookup, so a node could forward lookup to the one with lowest delay. This approach would be more effective with recursive Chord lookups, in which the node assessing the delays is also the node forwarding the lookup.

8. Conclusion

Many distributed peer-to-peer applications need to determine the node that stores a data item. The Chord protocol solves this challenging problem in decentralized manner. It offers a powerful primitive: given a key, it determines the responsible node for storing the key's value and does so efficiently. In the steady state, in an N-node network, each node maintains routing information for only about $O(\log N)$ other nodes, and receives all lookups via $O(\log N)$ messages to other nodes. Updates to the routing information for nodes leaving and joining require only $O(\log N)$ messages.

Attractive features of Chord include its simplicity, provable correctness, and provably good performance even in the face of frequent node arrivals and departures. It continues to function correctly, albeit at degraded performance, when a node's information is only partially correct. Our theoretical analysis, simulations, and experimental results confirm that Chord scales well with the number of nodes, does not suffer from overpopulation, node failures and joins, and almost most lookups correctly even during incertitude.

We believe that Chord will be a valuable component for peer-to-peer, large-scale distributed applications such as cooperative file sharing, time-shared available storage systems, distributed indices for document and service discovery, and large-scale distributed computing platforms.

Acknowledgements

We thank Frank Deblo and the members of the Chord project described in Section 6.6, and David Anderson for setting up the network used in those measurements.

9. References

Conclusion.

Conclusion is the section for the final words about the work done. Here you almost repeat what was mentioned in Abstract and Introduction.

Naturally, if your paper reveals something new and important, you will try to put it as close as possible to the beginning of the paper: into Abstract and Introduction.

Here, however, you just list all those main findings of your paper and probably the impact of it onto the corresponding field.
On the paper: Acknowledgement.

You can mention here proof readers, people who helped you and sometimes guide.

You don't need (and have no obligation to put all those people into author list, though sometimes people mistakenly put them there) to put everyone as your co-author. This is the right place.

Acknowledgements:

We thank Frank Debek for the measurements of the Chord prototype described in Section 8.6, and David Anderson for setting up the network test in those measurements.

9. References

On the paper: References.

Figure 13: Lookup latency on the Internet prototype, as a function of the total number of nodes. Each of the ten physical sites runs multiple independent copies of the Chord node software.

7. Future Work

Based on our experience with the prototype described in Section 6.6, we would like to improve the Chord design in the following ways:

Client centrality has no specific mechanisms to load partitioned rings; such rings appear directly connected to the stabilization procedure. One way to check global consistency is for each node to periodically ask other nodes to do a Chord lookup for $\mathcal{V}$, if the lookup does not yield node $\mathcal{V}$, then there may be a partition. This will only detect partitions whereas nodes know of each other. One way to obtain this knowledge is for every node to know of the same small set of initial nodes. Another approach might be for nodes to maintain long-term memory of a random set of nodes; they may encounter in the past, if a partition forms, the random sets in one partition are likely to include nodes from the other partition.

A malicious or buggy set of Chord participants could present an incorrect view of the Chord ring. Assuming that the data Chord is being used to locate in cryptographically authenticated, then it is a threat to availability of data to other than to authenticate. The same approach used above to detect partitions could help victim realize that they are not seeing a globally consistent view of the Chord ring.

An attacker could try to partition a data item by inserting a node into the Chord ring with an ID immediately following the item’s key, and having the node return errors when asked to retrieve the item. Replicating and checking that nodes use the IDs derived from the SHA-1 hash of their IP addresses makes this attack harmless.

Even for $k_0$ messages per lookup, the lookup cost may be too heavy for some applications of Chord, especially if each message must be sent to a random Internet host. Instead of placing fingers at distances that are all integer powers of $\sqrt{k}$, Chord could try to place fingers at distances that are all integer powers of $\sqrt{k}/\sqrt{2}$. Under such a scheme, a simple routing hop could decrease the distance to a query to $(1/2)^{k/2}$ of the original distance, meaning that $k/\sqrt{2}$ hops would suffice. However, the number of fingers needed would increase to $\log_2 N/(\log_2 N + 1/2)$ to $\log_2 N$ for a routing table with $k$ entries.

A different approach to improving lookup latency might be to use server selection. Each finger table entry could point to the first $k$ nodes in that entry’s interval on the Chord ring, and a node could measure the network delay to each of the $k$ nodes. The $k$ nodes are generally equivalent for purposes of lookup, so a node could forward the lookup to the one with lowest delay. This approach would be more effective with recursive Chord lookups, in which a node measuring the delay is also the node forwarding the lookup.

8. Conclusion

Many distributed peer-to-peer applications need to determine the node that stores a data item. The Chord protocol solves this challenging problem in decentralized manner. It offers a powerful primitive: given a key, it determines the node responsible for storing the key’s value and does so efficiently. In the steady state, as an $N$-node network, each node maintains routing information for only about $\log_2 N$ other nodes, and involves all lookups via $\log_2 N$ messages to other nodes. Updates to the routing information for nodes leaving and joining require only $\log_2 N$ messages.

Attractive features of Chord include its simplicity, provable correctness, and provably performance even in the face of random node service and departures. It continues to function correctly, albeit at degraded performance, when a node’s information is only partially correct. Our theoretical analysis, simulations, and experimental results confirm that Chord scales well with the number of nodes, recovering from large numbers of unscheduled node failures and joins, and almost always correctly even during recovery.

We believe that Chord will be a valuable component for peer-to-peer, large-scale distributed applications such as cooperative file sharing, time-shared available storage systems, distributed indices for document and service discovery, and large-scale distributed computing platforms.

Acknowledgements

We thank Frank DeBak for the measurements of the Chord prototype described in Section 6.6, and David Anderson for setting up the testbed used in those measurements.

9. References


Reference section is for all relevant work you mention in the text. Most of the references goes from the Related work and Introduction section.

Never, include as references papers which you do not mention at all, or mention "not-enough". Some reviewers use reference list to understand how good you studied your own field. Try to be up-to-date here also.
Time usage

Try to optimize your time. You paper should be multi-sided, address different aspects, different methods. Find time for every side, even for self-criticism.

Remember:

► Conferences happens all the time, but try to aim at some conference with fixed date.
► Put as much deadlines for yourself as possible. This help you to organize your time.
► All documented small outcomes that are collected in one work produce big work.
► Having all pieces on the paper put your ideas in order and allow not to lose results.
► Find time for reading (and studying), find time for writing.
Tools to use

Researcher has a various set of tools to use for all the Research methods. They are not emerging as often as new papers, but still remember to keep track of the current tools. They are created to help you.

There are a lot of tools for Data communication researcher:

- **Simulators** (NS-2, NS-3, OverSim, OMNeT++).
- **Development tools** (Eclipse, Visual Studio, EMacs).
- **Document preparation systems (WYSIWYG)** ($\LaTeX$, MS Word, OpenOffice).
- **Networking tools** (BRITE, Wirshark).
- **Data Analysis tools** (R, GnuPlot).
- **Mathematical labs** (MatLab, Mathematica).
- **Networking labs** (PlanetLab, OneLab).
Other methods to increase own understanding

There is a set of other methods to increase the quality of a research:

▶ attend Computer Science courses and border Science courses.
▶ attend public presentations, especially with presenters from other groups and Universities.
▶ take part in the review process.

What else?

Non-formal collaboration!!!
Questions and Comments?

Thank you.