
Local Structures Determine Performance within Complex Networks

Zur Erlangung des akademischen Grades Doktor-Ingenieur (Dr.-Ing.)
genehmigte Dissertation von Dipl.-Math. Lachezar Aleksandrov Krumov aus Pernik, Bulgarien
November 2010 — Darmstadt — D 17



TECHNISCHE
UNIVERSITÄT
DARMSTADT

Fachbereich Informatik
Fachgebiet Algorithmen

Local Structures
Determine Performance
within Complex Networks

Genehmigte Dissertation von Dipl.-Math. Lachezar Aleksandrov Krumov aus Pernik, Bulgarien

1. Gutachten: Prof. Dr. Karsten Weihe
2. Gutachten: Prof. Dr. Thorsten Strufe
3. Gutachten: Prof. Dr. Jussi Kangasharju

Tag der Einreichung: Oktober 15, 2010

Tag der Prüfung: Oktober 29, 2010

Darmstadt – D 17

Erklärung zur Dissertation

Hiermit versichere ich die vorliegende Dissertation ohne Hilfe Dritter nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die aus Quellen entnommen wurden, sind als solche kenntlich gemacht. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Darmstadt, den November 9, 2010

(L. A. Krumov)



Abstract

Networks are ubiquitous. We as individuals are part of various social networks and each of us depends on multiple communication, traffic and supply networks in our everyday life. We are, however, still far from completely understanding and controlling those networks.

Network motifs, few nodes (un)directed subgraphs, are a well-defined intermediate scale for characterizing the local structure of networks beyond the scope of single nodes. Motifs have so far been used only as a statistical measure. Their over- or under-representation in various networks has been related to specific topological function within those networks.

This work investigates a so far unexplored perspective on complex networks. Namely, the relation between their motif content and the dynamic processes taking place on top of those networks. For this purpose, we project the dynamic output of a network back onto its topology and investigate weighted motifs while keeping the network topology intact. This approach is unique to this work and reveals a direct relation between the motif content and the output pattern of complex networks.

We engage network motifs in two distinct ways. On the one side, as an analytical tool to examine already emerged networks. On the other side, as an active mechanism to adapt and improve human-made systems. A strong interplay is found between the local structures, the motif content, within a network and the dynamic performance of that network. Those new insights are used to develop a series of novel distributed topology control mechanisms.

First, we use networks motifs as an analytical tool on a subclass of complex networks with large and easily accessible data sets: co-authorship networks. We address the question: Is there a relation between the citation frequencies of publications and the motifs they are involved in?

Our analysis reveals a collaboration pattern much more successful than other collaboration patterns: the box motif, a closed chain of four authors. The box motif has the highest success measured as the average citation frequency per motif edge.

Our findings are confirmed on two large data sets and on data snapshots over the past 20 years. Segregation seems to be the key to success: separation in time, rank and discipline are the major factors for the success of the box motif.

An analytical generative model for co-authorship networks is introduced that can reproduce our findings and sheds light into the social factors shaping co-authorship networks.

The revealed interplay between the motif content of complex networks and the dynamic processes taking place on those networks motivates a new perspective on distributed communication networks. We address the question: Instead to use network motifs just as a static analytical tool, is it possible to engage them in online decision rules to improve human-made systems?

As a result we develop a novel topology control approach for Peer-to-Peer (P2P) networks. Each peer steers its local motif content to a desired state. Consequently, the overall network properties of the P2P overlay shifts towards a desired property. Fair load balancing in the demonstrated cases. Our evaluation shows that the new approach highly improves the network while causing negligible messaging overhead.

Motivated by our results, we address a more general question: Are network motifs indeed also suitable for topology control within heterogeneous networks, where nodes play different roles?

We extend our distributed topology control mechanism to heterogeneous P2P overlays. A novel approach for constructing highly resilient P2P live streaming networks is introduced. The peers choose from a set of rules how to adapt their motif content. The new approach induces resilience comparable to the state of the art methods. However, the topology of the constructed networks is better balanced than those of existing methods, making the new approach better performant under normal circumstances. Most importantly, the new approach requires no network knowledge. Consequently, the new method is not only much faster, but it also provides much higher privacy to the participating peers. Attacks by malicious parties are practically impossible. No peer can determine neither its position, nor the position of any other peer within the network. In that sense, the new proposed approach clearly outperforms the state of the art.

Our findings so far clearly show that one can understand or even actively change the dynamic performance of networks by looking at their local topology. It is natural to investigate the opposite perspective: Is it possible to deploy suitable dynamic processes on a network with no global network knowledge, in order to reveal its topology?

Consequently, we impose a dynamic process on top of a network in order to determine critical topological constellations within the network. By deploying an extended gossiping protocol, we show how one can detect communication bottlenecks in distributed manner. Our novel approach clearly outperforms state of the art methods with respect to both, the precision of its results and its performance. Last but not least, it has a guarding mechanism against malicious peers trying to skew the network protocol.

So far we have shown that specific local structures lead to specific network properties and performance. Finally, we argue that random graphs and their random local structures also have unexploited potential. They have become notorious in the recent years for being poor null models of real world networks. Nevertheless, they have topological properties highly desirable within any P2P overlay. We introduce a novel P2P overlay based on random graphs. This new overlay is the first one to support exhaustive search queries and exact key-value lookups within the same overlay. Our overlay is both, highly scalable and efficient, and performs at least as good as already established P2P overlays.

Throughout this work we repeatedly show that analyzing networks on intermediate scale opens so far unexplored and very fruitful perspectives on complex networks. The introduced new methodology for distributed topology control, advocated through this work, is just one of those perspectives.

Zusammenfassung

Netzwerke sind allgegenwärtig. Wir nutzen tagtäglich diverse Kommunikations-, Transport- und Versorgungsnetzwerke und partizipieren an sozialen Netzwerken. Allerdings sind wir weit davon entfernt diese Netzwerke vollständig zu verstehen und zu beherrschen.

Motive in Graphen - dies sind aus wenigen Knoten bestehende (un)gerichtete Untergraphen - stellen ein wohldefiniertes intermediäres Maß zur Charakterisierung lokaler Strukturen eines Netzwerks dar, welches über die Bedeutung einzelner Knoten hinausgeht. Bislang wurden Motive nur als statistisches Maß verwendet. Ihre Unter- oder Überrepräsentanzen in zahlreichen Netzwerken konnte erfolgreich spezifischen topologischen Funktionen zugeordnet werden.

Diese Arbeit beschäftigt sich mit einer bislang unerforschten Perspektive komplexer Netzwerke: Sie betrachtet die Relation zwischen Motivgehalt und dynamischen Prozessen, die in diesen Netzwerken stattfinden. Um dies zu analysieren führen wir das Ausgabemuster eines Netzwerks zurück auf dessen Topologie und untersuchen dabei gewichtete Motive. Dieser bislang einzigartige Ansatz zeigt eine direkte Relation zwischen Motivgehalt und Ausgabemuster komplexer Netzwerke.

Wir setzen Motive in zwei unterschiedlichen Ansätzen ein: auf der einen Seite als analytisches Werkzeug zur Erforschung bereits vorhandener Netzwerke, auf der anderen Seite als aktiver Mechanismus zur Anpassung und Verbesserung von durch Menschenhand geschaffene Systeme. Wir weisen einen starken Zusammenhang zwischen den lokalen Strukturen, dem Motivgehalt eines Netzwerks und dessen Leistung nach. Diese neuen Einblicke werden später zur Entwicklung einer Reihe innovativer verteilter Ansätze zur Steuerung der Netzwerkstruktur eingesetzt.

Zu Beginn nutzen wir Motive als analytisches Werkzeug auf einer Unterklasse komplexer Netzwerke mit zahlreich vorhandenen und leicht zugänglichen Netzwerkdaten: Co-Autoren Netzwerke. Wir beschäftigen uns mit der Frage: Existiert ein Zusammenhang zwischen Zitathäufigkeit der Publikationen und der daran beteiligten Motive?

Unsere Analyse zeigt ein Muster der Zusammenarbeit deutlich erfolgreicher als alle anderen: das sogenannte *Box-Motiv*, eine geschlossene Kette aus vier Autoren. Das Box-Motiv hat den größten Erfolg gemessen an der durchschnittlichen Zitathäufigkeit pro Motivkante.

Unsere Ergebnisse lassen sich auf zwei großen Datensätzen und auf Daten-Schnappschüssen über den Zeitraum der letzten 20 Jahre bestätigen. Segregation scheint hier der Schlüssel zum Erfolg zu sein: eine Trennung in der Zeit, im Rang und in der Disziplin sind die stärksten Faktoren, die zum Erfolg des Box-Motivs führen.

Weiterhin stellen wir ein generatives Modell zur Erstellung gewichteter Co-Autoren Netzwerke vor, welches unsere Ergebnisse reproduzieren kann und einen Einblick in die sozialen Faktoren ermöglicht, die Co-Autoren Netzwerke beeinflussen.

Der entdeckte Zusammenhang zwischen Motivgehalt komplexer Netzwerke und dynamischer Prozesse, die in diesen Netzwerken stattfinden, hat eine neue Sicht auf verteilte Kommunikationsnetzwerke inspiriert. Wir haben uns folgende Frage gestellt: Ist es möglich, Motive

in Online-Entscheidungsprozesse so einzubinden, dass sie technische Netzwerke verbessern, anstatt sie nur als analytisches Werkzeug zu verwenden?

Dem zu Folge haben wir einen innovativen Ansatz zur Steuerung der Netzwerkstruktur von Peer-to-Peer (P2P) Netzwerken entwickelt. Jeder Peer steuert sein lokalen Motivgehalt in Richtung eines erwünschten Zustands. Infolge dessen konvergieren die allgemeinen Netzwerkeigenschaften gegen eine erwünschte Beschaffenheit, in den gezeigten Beispielen nämlich gegen eine gleichmässige Verteilung der Last im Netzwerk. Unsere Evaluation zeigt unumstritten, dass dieser neue Ansatz die betrachteten Netzwerke deutlich optimiert und zugleich keinen oder einen vernachlässigbar kleinen Kommunikationsmehraufwand verursacht.

Motiviert durch unsere Ergebnisse, haben wir uns eine allgemeinere Frage gestellt: Sind Netzwerk motive auch zur Steuerung der Netzwerkstruktur in heterogenen Netzwerken geeignet, in denen die Knoten unterschiedliche Rollen erfüllen?

Zu diesen Zweck haben wir unseren Ansatz für die verteilte Steuerung der Netzwerkstruktur auf verschiedenartige P2P-Overlays erweitert: wir begründen eine innovative Vorgehensweise zur Erzeugung stark elastischer P2P Live-Streaming Netzwerke. Die Peers wählen hierbei aus eine Menge vorgegebener Richtlinien aus. Diese Richtlinien geben genau vor, wie die Peers ihren Motivgehalt anpassen sollen. Dieser neue Ansatz induziert eine Elastizität vergleichbar der aktueller Methoden in diesem Bereich. Weiterhin ist die Struktur der generierten Netzwerke besser balanciert als die der vergleichbaren aktuellen Methoden, was dazu führt, dass unser Ansatz unter normalen Umständen besser funktioniert. Noch wichtiger ist, dass unserer Ansatz kein Wissen über das Netzwerk voraussetzt. Dadurch ist er nicht nur deutlich schneller, sondern auch garantiert sehr viel sicherer in Bezug auf die Daten. Folglich sind Angriffe durch böswillige Parteien praktisch unmöglich. Kein Peer kann weder seine eigene, noch die Positionen eines anderen Peers im Netzwerk feststellen. Dadurch ist unser neuer Ansatz vergleichbaren Methoden deutlich überlegen.

Wir haben nun deutlich gemacht, dass man Netzwerke besser verstehen kann oder sogar aktiv ihre dynamische Leistungsfähigkeit verändern kann, indem man ihre lokalen Strukturen näher betrachtet. Eine nahezu selbstverständliche Konsequenz ist es daher, sich auch mit der umgekehrten Sichtweise zu beschäftigen: Ist es möglich, durch gezieltes Einbringen geeigneter dynamischer Prozesse in ein Netzwerk seine Struktur zu erforschen, ohne globales Wissen über das Netzwerk zu besitzen?

Um diese Frage zu beantworten, führen wir einen dynamischen Prozess in einem gegebenen Netzwerk aus, um kritische topologische Konstellationen zu identifizieren. Wir zeigen wie man verteilt Kommunikationsengpässe ermitteln kann, indem man ein erweitertes Gossiping-Protokoll betreibt. Dieser neuartige Ansatz übertrifft aktuelle Methoden in diesem Bereich sowohl bezüglich der Präzision seiner Ergebnisse als auch bezüglich seiner Leistung. Nicht zuletzt hat unserer Verfahren Schutzmechanismen gegen böswillige Peers, die versuchen das Netzwerkprotokoll zu stören.

Spezifische lokale Strukturen führen also zu spezifischen Netzwerkeigenschaften und spezifischer Leistung und umgekehrt. Im letzten Teil der Arbeit zeigen wir, dass Zufallsgraphen und deren zufällige lokale Strukturen grosses Potenzial besitzen. Zufallsgraphen sind in den letzten Jahren als mangelhafte Nullmodelle realer Netzwerke zunehmend in Verruf geraten. Dennoch haben sie strukturelle Eigenschaften, die für P2P-Overlays hochgradig erwünscht sind.

Wir präsentieren einen innovativen und auf Zufallsgraphen basierenden P2P Overlay. Dieser neue Overlay unterstützt als erster Overlay überhaupt sowohl flächendeckende Suchanfragen als auch exakte Schlüsselwort-Suchläufe innerhalb ein und desselben Overlays. Unserer Overlay ist effizient und sehr gut skalierbar. Gleichzeitig ist er mindestens so leistungsfähig wie etablierte P2P Overlays.

Die vorliegende Arbeit zeigt in vielerlei Hinsicht, dass es bislang unerforschte und vielversprechende Perspektiven eröffnet, Netzwerke auf Basis eines intermediären Masses zu untersuchen. Die hier eingeführte Methodik zur verteilten Steuerung der Netzwerkstruktur, belegt durch diese Arbeit, ist nur eine dieser Perspektiven.



Acknowledgments

At this point, I would like to thank my colleagues, friends and family members. This work would not have been possible without their continuous support and encouragement over the last years.

First of and foremost, I would like to thank my supervisor, Karsten Weihe, for his faith in my work and me. His excellent advices, inspiring comments and the great working environment he has provided me play substantial role in putting this thesis together.

I am also deeply grateful to Thorsten Strufe and Jussi Kangasharju for guiding me through the process of conducting solid scientific work. Our discussions over the last years have improved my scientific attitude enormously.

Special thanks go to Marc-Thorsten Hütt for showing me what it takes to pursuit ambitious scientific goals. He has provided me with invaluable knowledge on how to formulate and substantiate scientific findings across disciplines.

The great inspiration power of my dear friend and colleague, Dirk Bradler, have involved me in numerous collaboration projects. Most of them have proved pure success and their various outcomes are essential parts of this work.

Very special thanks go to the Volkswagen Foundation. Without their funding support, this work would have never been possible, nor the numerous international and interdisciplinary collaborations that have emerged over the past years.

Finally, I am deeply grateful to my parents and my brother. I sincerely appreciate all the sacrifices you have made to give me the opportunity to stand where I am. My gratitude to you is endless.

Last but not least, my most whole-hearted thanks go to Adriana Andreeva. Her mental support and patience were invaluable to me. She helped me stay focused and gave me hope at the moments I needed it at most.

Darmstadt, September 2010,

Lachezar Krumov



Contents

1	Introduction	15
1.1	Motivation and Scope	20
1.1.1	Social Networks	20
1.1.2	Distributed Systems and Adaptive Networks	20
1.2	Outline and Results	21
1.2.1	Social Networks	21
1.2.2	Distributed Systems and Adaptive Networks	21
2	Separation Leads to High Citation Frequencies: the Box Motif	23
2.1	Introduction	23
2.1.1	Early History of Knowledge Production	24
2.1.2	Modern Perspective	24
2.1.3	Relevance to Complex Network Analysis	24
2.2	Background on Co-authorship Networks	25
2.2.1	Statistical Properties	25
2.2.2	Small World and Average Network Properties	25
2.2.3	Community Structure	25
2.3	Motivation and Related Work	26
2.4	Graph Representation	26
2.5	Findings: the Success of the Box Motif	27
2.5.1	Converting Publication Impact to Edge Weight	27
2.5.2	Main Result	28
2.6	Deeper Look: Separation	31
2.6.1	Separation in Rank: Established Authors and Newcomers	32
2.6.2	Separation in Time	33
2.6.3	Separation in Scientific Area: Interdisciplinary Collaborations	34
2.7	Supporting Experiments	34
2.7.1	Network Properties	35
2.7.2	Weight Distributions and Average Values	36
2.7.3	Most Successful Motif Instances	37
2.7.4	Eliminating Trivial Effects	37
2.8	Further Analysis: Generative Model	38
2.8.1	The Model	39
2.8.2	Evaluation	40
2.9	Technical Aspects	43
2.9.1	Publication Data	43
2.9.2	Citation Indices	44
2.9.3	Co-authorship Graph Representation	44

2.10	Summary and Outlook	45
2.10.1	Summary	45
2.10.2	Outlook	46
3	Motif Based Optimization of Structured P2P Networks: Fair Load	49
3.1	Introduction	49
3.2	Background on Load Balancing	50
3.3	Determining Target Motif Signatures	51
3.3.1	Target Motif Signature: CAN	53
3.3.2	Target Motif Signature: Kademlia	54
3.4	Motif Based Optimization	55
3.4.1	Motif Based Optimization: CAN	56
3.4.2	Motif Based Optimization: Kademlia	56
3.5	Evaluation	57
3.5.1	CAN	57
3.5.2	Kademlia	60
3.6	Summary and Outlook	62
3.6.1	Summary	62
3.6.2	Outlook	63
4	Resilient Peer-to-Peer Live-Streaming Using Motifs	65
4.1	Introduction	65
4.2	Background	67
4.3	Related Work	68
4.4	System Design	69
4.4.1	Network Motifs	69
4.4.2	Engaging Network Motifs in Topology Optimization	70
4.4.3	Implementation	71
4.5	Evaluation	72
4.5.1	Management Overhead	73
4.5.2	Topological Properties	73
4.5.3	Resilience to Attacks	78
4.6	Method Comparison	79
4.6.1	Topological Properties	79
4.6.2	Convergence and Complexity	79
4.6.3	Network Resilience	80
4.7	Summary and Outlook	81
4.7.1	Summary	81
4.7.2	Outlook	82
5	Finding Communication Bottlenecks in Distributed Environments	83
5.1	Introduction	83
5.1.1	Network Prerequisites	84
5.1.2	Application Domains	84

5.2	Properties of Critical Peers	84
5.2.1	Centrality Measures: Betweenness Centrality	85
5.2.2	Centrality Measures: Closeness Centrality	86
5.3	The BridgeFinder Algorithm	87
5.4	Evaluation	89
5.5	Gossiping Convergence	92
5.6	Related Work	94
5.7	Security Issues	95
5.8	Summary and Outlook	97
5.8.1	Summary	98
5.8.2	Outlook	98

6 Efficient Search and Lookups in Peer-to-Peer Networks 101

6.1	Introduction	101
6.2	Motivation	103
6.3	System Design: PathFinder	103
6.3.1	Challenges	103
6.3.2	System Model and Preliminaries	104
6.3.3	Storing Objects	107
6.3.4	Key Lookup	108
6.3.5	Searching with Complex Queries	110
6.3.6	Node Join and Leave	110
6.3.7	Node Crash	111
6.3.8	Network Size Adaptation	112
6.4	Comparison and Analysis	114
6.5	Resilience Against Failures	117
6.6	Security and Other Issues	119
6.7	Summary and Outlook	119
6.7.1	Summary	120
6.7.2	Outlook	120

7 Summary and Outlook 123

7.1	Summary	123
7.2	Outlook	124

8 Authors Publications 139

List of Figures 141

List of Tables 143

List of Algorithms 145

Index 147

1 Introduction

Networks are ubiquitous. We are all part of numerous social networks: at home in our family circle, at work with our colleagues, in the virtual space of online forums and many others. Moreover, each one of us relies on a variety of technological networks in our everyday life by using mobile phones, the Internet, the World Wide Web, public transport, just to list a few. Basically, any system where entities interact with each other can be represented as a network.

Understanding the basic functional principles and the factors governing the evolution of the networks surrounding us is a task of immense importance. Then, this will allow us to interact more efficiently within social networks, e.g. co-authorship networks, and open a new horizon for optimization of many already existing human-made systems like mobile networks, peer-to-peer systems, live streaming and many others.

A network is a set of entities called *vertices* or *nodes* and the connections among them are called *edges*. In most of the mathematical literature networks are called *graphs*. Examples include food webs, social networks, co-authorship networks, the Internet, the World Wide Web, peer-to-peer (P2P) systems, air transportation, etc. The analysis of networks goes back with centuries and was carried out mainly by mathematicians. The most famous example dates back at the 17th century, when Euler provided a solution of the Königsberg bridge problem. During the 20th century, graph theory has turned into a major field of mathematics and a substantial body of research has been developed.

In the recent few years, however, we have experienced new tremendous interest towards network research. It resulted from the fast increasing availability of computational power together with the improvement of communications networks. Both of which lead to the digitalization of network data, making it easy to access and evaluate. The research focus shifted from the analysis of single small graphs and the properties of single nodes or edges within those graphs towards large-scale statistical properties. Instead of investigating graphs with just 30 or 40 nodes, as it is common in social network analysis, researchers are now able to access networks of millions of nodes, which was considered science fiction just a few years back.

The new perspective on network research has lead to many unexpected discoveries about real world networks. In order to better place the scope of this work into the already existing body of research, we give a short summary of the different directions network research has taken.

Early History

One of the main discoveries on real world networks dates back to 1967 with the work of Milgram [1]. He carried out the following experiment: a bunch of letters were distributed among people resident on the east cost of the United States. The participants then had to pass their letters to people they knew on first name basis and in such a way that the letters came closer to their destinations on the west cost. Surprisingly, most of the letters traveling from person to person arrived at their destination in a very small number of steps - around six. Those experiments are the first confirmation of the *small-world effect* [2], which states that the average path length between any two individuals on the planet is very short. Nowadays, the small-world effect has

been verified in a large number of other networks and it has been shown that the average path length of those networks scales logarithmically with the network size.

Random graph models have as well always been in the center of network research as they provide a testbed for any analytical work. The most famous random graph model is the one of Erdős and Rényi [3]. Their model (ER) has a very simple construction principle: given N nodes, connect each pair of nodes with probability p . The set of all such graph instances is denoted by $G_{N,p}$. In fact, $G_{N,p}$ is the collection of all graphs with N nodes and m edges appearing with probability $p(1-p)^{M-m}$, where $M = \frac{1}{2}N(N-1)$ is the maximum possible number of edges. Many properties of those graphs are exactly solvable for large N [4]. Since the absence or presence of an edge is an independent event, for large n the degree distribution of those graphs is well approximated by the Poisson distribution and therefore called *Poisson random graphs*.

Recent Research Progress

The ER random graphs remained the standard testbed for network analysis for over 40 years. It was not short before the end of the last century, when it was discovered that many real world networks have properties not reflected by the ER graphs. One of those properties is *transitivity* or also called *clustering*. It represents the higher probability that two nodes are connected if they have a common neighbor. For example, two of your friends have higher probability of knowing each other than any other two individuals on the planet chosen at random.

It was the ground breaking work of Watts and Strogatz [5] that incorporated both, the logarithmic average shortest path typical for real world networks and ER graphs, and high clustering. Their model starts by ordering all nodes in a circle and connecting each node by a given number of nodes on the lefthand and on the righthand side from it on that circle. Then by *rewiring* only a very small portion of the links between randomly chosen nodes, they achieved both logarithmic path length and high clustering. Still, their model has one major limitation: it works only for fixed networks. That is, the number of nodes must be given and no nodes join or leave the network during the simulation time.

Motivated by the World Wide Web, Barabási and Albert [6] proposed a model incorporating network growth. They observed that not only nodes *join* and *leave* the network all the time, but that new nodes have affiliation towards connecting to older nodes, which are already part of the network for a long time. Their model induces the so called *preferential attachment*: it starts with a small number of connected nodes and then new nodes are continuously introduced in the network. Each new node connects to a number of other nodes, selecting them with probability proportional to their degree. That model not only reflected network growth while possessing logarithmic average shortest path length, but also revealed that real world networks are *scale free* [7, 8], meaning that some of the topological properties of the underlying network are independent from its size. Those include average shortest path, diameter, average degree, clustering, etc. and those properties remain almost constant no matter how large the network expands or shrinks. That was illustrated by investigating snapshots of the World Wide Web differing in size within several orders of magnitude, all of which showed almost identical topological characteristics [8].

The above model not only revealed the unsuspected scale-free property of real world networks, but it also confirmed a hypothesis circulating in the scientific community for a while now: real world networks are not Poisson distributed. Their degree distributions follow rather a *power law* [6, 9, 10, 11, 12], i.e. most of the nodes have small number of neighbors, but

there are a few nodes with very large degree. More precisely, the probability that a node v has degree k is given by $P_v(k) = k^{-\gamma}$ where $\gamma \in [2, 4]$. As a consequence, real world networks are significantly less resilient to attacks and spread of diseases than their ER random counterparts. Removing the few nodes with high degrees leads to drastic disruptions within the network.

After it was found that different nodes have different degrees, and with high probability carrying different function within their underlying network, another discovery was yet to be made. Newman investigated what he called the *assortativity* of networks [13]. It reflects the portion of high degree nodes connected to other high degree nodes and the portion of high degree nodes connected to low degree nodes. If in a network the better portion of high degree nodes are connected to similar nodes, it is called assortative, otherwise disassortative. Investigating assortativity on a range of social networks, Newman discovered that social networks are in deed assortative while biological and technological networks are on the other side disassortative.

The availability of network data and the shortly detected clustering and assortativity of real world networks motivated researchers to investigate another widely assumed hypothesis. Namely, that most social networks show *community structure*, meaning that there are groups of nodes highly interconnected among themselves, while the groups on the other side are only sparsely connected to each other [14]. Girvan and Newman introduced a new measure called *modularity* [15]. It is based on the adjacency matrix of the network and measures the deviation of connectivity among nodes in the investigated network from what one would expect uniformly at random. Then, by starting with the whole network and by consequently removing edges one can measure the modularity of the network at each step. The division of the network after the step producing the highest modularity is the community structure of the network.

Finding a division of a network that maximizes its modularity is believed to be *NP-hard* [16]. Therefore, many heuristic methods have been proposed. Those include greedy algorithms [17], spectral methods [18], extremal optimization [19], genetic algorithms [20], pass-the-parcel methods [21] and many others. The most successful algorithms so far are those removing edges from the network [15] based on their *edge betweenness* [22]. The betweenness of an edge reflects the number of shortest paths among nodes in the network running through that edge, for which Brandes have introduced a few very efficient algorithms [23, 24].

Independently of the approach at hand, it was undoubtedly shown that many real networks, aside from social networks, possess very characteristic community structures, revealing a completely different perspective on networks and their analysis.

Our contribution

So far we have barely scratched the surface of the immense body of research dedicated to network analysis in the last few years. We have shown a few of the research directions and their ground breaking publications. Still, there is a huge number of network specific publications, which investigate characteristic properties of the underlying real world networks at hand.

It is way beyond the focus of this work to present a complete chronology of all existing publications on network analysis. In this work we address one still pending question, that so far has not been addressed by the research community: can *local structures* reveal deeper understanding of the function of complex networks and what is their impact on the dynamic performance of the underlying networks?

The main contribution of this work is to reveal the interplay between dynamic data and local structures, also called *network motifs*, in complex networks. Then, to shift that knowledge on

dynamic self-organizing human-made networks and to deploy motif based live optimization of P2P overlays as well as to present a novel approach for constructing resilient P2P live streaming networks, competitive to the state of the art. We also involve a local decision rule for detecting communication bottlenecks in distributed environments and show that random graphs possess properties that make them suitable basis for a novel P2P overlay, supporting both: key based lookups and broad range searches.

In the following we shortly discuss network motifs, a perspective on local network structures introduced by the biological research community, as well as the functions they carry in biological networks. Network motifs are one of the major tools we use in this work to achieve our goals.

Network Motifs

Network motifs are small subgraphs with a specific interaction pattern, usually constituting of three or four nodes and are considered directed or undirected depending on the underlying network. Figure 1.1 displays the eight three- and four-node undirected motifs:

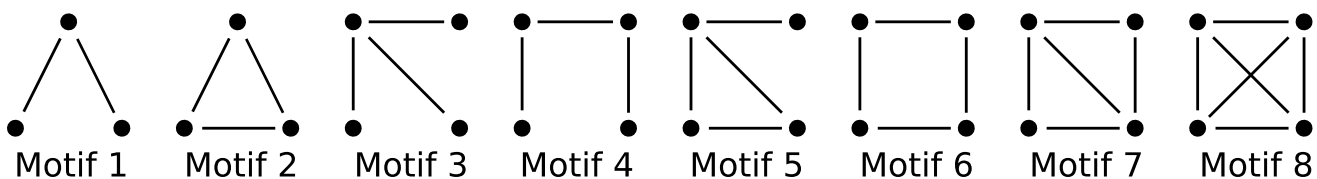


Figure 1.1: The eight possible undirected three- and four-node motifs.

They have been introduced by Milo et al. [25] and one usually counts the number of occurrences of the different motifs and compare them to a randomized null-model network.

Network motifs have been particularly successful in providing interesting, unexpected relations between network architecture and topological function. In particular in systems biology, an influential trend currently relates features of network performance to such small regulatory devices [26, 27], serving e.g. as a noise buffer or providing a suitable amount of redundancy for maintaining systemic function even under perturbations.

In particular such relations between the architecture of regulatory devices and topological functions have been worked out for circuits of negative feedback loops [28], for feedforward loops as noise filtering devices in gene regulation [29, 26], for interlinked feedback loops acting on different time scales [30], for a particular composition of regulatory units [31] and their relation to robustness [32, 33, 34, 35], and for the number of positive and negative feedback loops in regulatory circuits [36].

Before we continue with the exact scope and the challenges upon this work, we give a short introduction on one of our main application domains.

Peer-to-peer (P2P) is any distributed communication architecture where the participants connect directly to each other and not to dedicated servers. Every participant shares parts of her/his resources (bandwidth, disc space, even processing power) with the rest of the participants in the system. In that way all participants are both suppliers and consumers, which is exactly the opposite to traditional client-server models where only servers supply and clients consume. Figure 1.2 shows a small illustration of those two different architectures.

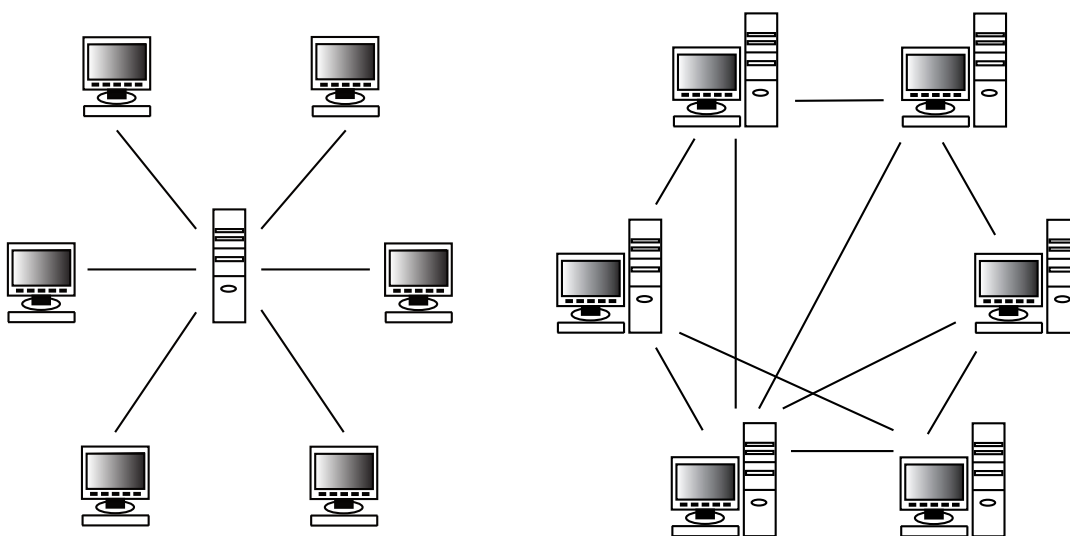


Figure 1.2: The traditional client-server architecture and a P2P architecture.

The resources in P2P networks grow with the network size as every new participant provides new resources to the network. In that sense P2P have *unlimited* resources. Furthermore, all participants are equally important for the network function, which eliminates the typical for client-server models single point of failure and in the same time also drastically increases the network robustness to perturbations.

Those are some of the reasons for the storm-like growth of P2P systems in the last few years. As of 2009, P2P networks constitute the largest portion of traffic on the network. Applications include file sharing networks like Gnutella and FastTrack; streaming media (P2PTV) like Cool-Streaming and LiveStation; research projects like Chord; voice over IP (VoIP); internet phone applications like Skype; and many others. Most of these networks have tens of millions of users.

The major drawback of P2P is that they are vulnerable to attacks by malicious peers. Furthermore, the usually deployed basic protocols only have availability and connectivity as major objectives. The resulting overlays are not optimized towards efficiency and performance and there is no central authority controlling and steering the network function.

Overcoming some of those drawbacks of P2P in a distributed manner is one of the major goals pursued in this work.

1.1 Motivation and Scope

In this work we inherit the simple perspective of network motifs to investigate social networks, more precisely co-authorship networks. We show how one can use them to reveal the success of collaboration patterns in terms of citation frequencies.

We then go one step further and propose a methodology for optimizing P2P overlays and creating resilient P2P live streaming topologies by deploying network motifs in changing and self-organizing environments.

In other words, we investigate social networks from the static perspective, as only fixed snapshots of the networks are available, and match local structures to dynamic performance. We then transfer that knowledge to networks adaptable in real time, in our case diverse P2P systems as communication infrastructures currently of great importance, to develop dynamic optimization strategies for a range of human-made systems.

We explore the reverse direction of the relation topology - dynamic performance. We show that deliberately inducing a dynamic process on top of a communication network can provide a reliable scheme to detect communication bottlenecks in a distributed manner.

To this end, only the relation between specific local structures and specific network performance is addressed. Finally, we argue that random graphs and their random local structures still have unexploited potential. We engage their numerous beneficial properties in a new P2P overlay, the first to support both broad-range queries and exact key-value lookups.

1.1.1 Social Networks

The search for fundamental relationships between network architecture and dynamical data is the guiding principle underlying our investigation. In order to identify such links between topology and dynamics for co-authorship networks, we explore the distribution of impact of publications across few-node subgraphs in the co-authorship networks.

For this purpose we investigate two large co-authorship networks, DBLP [37] and CiteSeerX [38], and use web crawlers to match citation frequencies onto publications by using CiteSeerX [38] and Google Scholar [39]. We map the co-authorship networks onto graphs, where two authors are connected if they have ever published together. The citation frequencies are then mapped to edge weights. The question we addresses is: Is there a particular collaboration pattern that is more successful than all others?

1.1.2 Distributed Systems and Adaptive Networks

To this point, motifs have been used only as a statistical measure for revealing the interaction between architecture and topological function within networks. In this work, we move one step further and engage network motifs in local decision rules.

Distributed and self-organizing system, such as P2P, allow for the network entities to alter their local environment. In consequence, we tackle the following question: How do the global network properties change if all network entities locally change their surrounding, reflected by the motifs they are involved in, towards a desired local state? In other words, how motifs can be used to create local decision rules for distributed network optimization.

Next we tackle the opposite perspective of the topology - dynamics relation. We investigate whether deliberately initiated dynamic processes on networks can be used to reveal their underlying topology. More precisely, whether one can use dynamic processes to construct a reliable scheme for detecting communication bottlenecks in distributed environments?

Network motifs reflect the specific local structure of a network, but specific topologies have their characteristic limitations in terms of robustness, communication flow, etc. Therefore, we address the following question as well: What if we stick to randomized local structures and random networks, such as the ER graphs? Will this allow us to profit from the numerous desirable for any P2P overlay properties of ER graphs, such as high robustness, short communication paths, etc.? More importantly, is it possible to combine those with a similar to key-based lookup function, such that we can unite the properties of structured and unstructured P2P systems within one single overlay?

1.2 Outline and Results

In the following we shortly discuss the results presented throughout this work and to what extent the goals listed above have been achieved, followed by a short outline.

1.2.1 Social Networks

Our results on co-authorship networks show that motif 6 (four nodes forming a closed chain, see Figure 1.1), which we call the *box motif*, is more successful than all other motifs, measured as the average citation frequency per motif edge. Furthermore, it has the highest ratio to its counterpart in a null-model network with identical topology, but shuffled citation frequencies. Our results are stable over two large databases, DBLP [37] and CiteSeerX [38] and over data snapshots for the past twenty years.

We successfully eliminate trivial effects, such as authors per paper or papers per edge, as possible explanation of our findings and show that all investigated distributions are monotone and all extracted average values justified.

We even go one step further and investigate separation effects, such as time, rank and discipline. We reveal that they to good extent explain the unexpected success of the box motif.

Finally, we introduce an analytical generative model for constructing co-authorship networks. This model incorporates dynamic processes on co-authorship networks, such as publishing and citing of scientific publications. Our model successfully reproduces the success of the box motif and sheds light into the social factors shaping co-authorship networks.

1.2.2 Distributed Systems and Adaptive Networks

We engage network motifs in local optimization rules for distributed optimization of P2P overlays. Each peer alters its local surrounding towards a desired one, extracted from an optimal topology with respect to a given network property. We investigate fair load balancing and uniform key space distribution in two different structured P2P overlays. Our results undoubtedly show a global shift of both overlays towards the desired global properties, while causing almost negligible overhead and operating in a fully distributed manner.

Furthermore, we extend that approach to construct resilient P2P live streaming systems. Our extensive simulations show that our novel approach is competitive to state of the art methods with respect to attacks and failures. More importantly, it provides even higher privacy for the network participants, making attacks by malicious peers almost impossible. Last but not least, our approach relies on local decision rules which are extremely fast and straightforward to calculate, making it applicable for devices with limited resources, such as mobile devices.

With respect to our next goal, we present a new approach, BridgeFinder, for detecting communication bottlenecks within distributed environments. It deploys a local decision rule based on gossiping to detect critical for the communication flow participants. Our results show that the peers detected by BridgeFinder with high accuracy overlap with the ones detected by combining centrality measures from graph theory. Furthermore, BridgeFinder has a guarding mechanism against malicious nodes and is more reliable and accurate than any existing approach. It can be implemented as a background process and integrated within the existing communication flow, causing no additional computational or communication overhead.

To address our last goal, we augmented random ER graphs with a deterministic lookup function. We deploy this graph on a top layer of virtual nodes, distributed among the actual peers to construct a novel P2P overlay called PathFinder. PathFinder is the first overlay to combine both structured and unstructured P2P systems within the same overlay by supporting broad-range searches as well as key-based lookups. Our extensive simulations, shows that PathFinder is extremely resilient to attacks and failures and scales to hundreds of millions of nodes, while performing lookups at least as fast as already established structured P2P overlays.

The rest of this work is structured as follows: In Chapter 2 we investigate the success of collaboration patterns in co-authorship networks. Based on our insights, we then present a distributed topology control scheme in Chapter 3. In Chapter 4 we extend this approach to construct resilient P2P live streaming topologies. Chapter 5 presents a scheme for detecting communication bottlenecks in distributed environments. A novel P2P overlay based on random graphs is introduced in Chapter 6. Chapter 7 summarizes our results and gives an extended outlook on future work of scientific interest.

2 Separation Leads to High Citation Frequencies: the Box Motif

In this Chapter we investigate the interplay between citation frequencies and local structure in co-authorship networks.

Co-authorship networks, where the nodes are authors and edges indicate joint publications, are very helpful representations for studying the processes that shape the scientific community. At the same time, they are networks with a large amount of available data and thus serve as vehicles for analyzing complex networks in general.

Our findings give any scientist, concerned about the impact of her/his publications, insights on the success of different collaboration patterns. They also deepen our understanding of the function and performance of complex networks in general. Our findings furthermore address a fundamental question in complex network analysis. Namely, how does the network topology shape the dynamical processes taking place on top of that network and vice versa.

2.1 Introduction

Previous work on co-authorship networks can be divided in three different branches. The first one focuses on the statistical properties of individual authors and individual publications, including citation distribution, degree distribution, etc. The second branch concentrates on the network as a whole, investigating its clustering, connectedness, etc. The last research direction is dedicated to the topological function of single authors, including average distance to other authors, number of shortest paths going through a given author, etc.

Here we show that the success of individual authors or publications in co-authorship networks depends unexpectedly strongly on an intermediate scale, i.e. beyond the scope of single authors, but still in their surrounding environment. For two large-scale data sets, CiteSeerX [25] and DBLP [31], we analyze the correlation of three/four node network motifs with citation frequencies. We find that the average citation frequency of a group of authors depends on the motifs these authors form. In particular, a box motif (four authors forming a closed chain without chords) has the highest average citation frequency per edge. This result is robust across the two databases, across different ways of mapping the citation frequencies of publications onto the corresponding graph representations, and over time.

We also relate this topological observation to the underlying social and socio-scientific processes that have been shaping the networks. We argue that the box motif may be an interesting category in a broad range of social and technical networks.

Despite the static nature of our analysis, it clearly indicates a close interplay on intermediate level between the structure and the dynamic processes taking place on complex networks. Later on in this work, we use that fact to develop novel and fully distributed topology control mechanisms for steering and improving various technological networks.

2.1.1 Early History of Knowledge Production

One of the classical debates in the history of science is, whether the production of knowledge can be viewed as an objective, content-driven process or is it rather dominated by the underlying social patterns formed by the involved actors.

This work contributes to that debate as well. We show that the collaboration patterns of authors indeed determine the success of their publications, measured as the number of citations by other publications.

Ever since the groundbreaking work of Thomas Kuhn in the 60-ties, “The Structure of Scientific Revolutions” [40], it is accepted that the social layer contributes heavily to scientific progress. Expectations of individuals and the adherence to agreed-upon terminologies all have a synchronizing effect, i.e. determine trends in scientific communities. These phenomena may be considered a socially generated inertia, leading to the characteristic discontinuous time course of scientific progress. The social layer plays an undeniable role in that process.

2.1.2 Modern Perspective

Nowadays, vast amounts of data on knowledge production are electronically available. Therefore, the study of complex networks provides a unique opportunity to quantitatively assess the contribution of the social layer to the production of knowledge.

From the network perspective, the strength of the contribution of the social layer can be rephrased as follows: Does the underlying interaction of authors and publications statistically explain parts of the output pattern of the scientific community? Hence, do specific collaboration patterns lead to higher citation frequencies. This question is at the core of our analysis.

2.1.3 Relevance to Complex Network Analysis

A fundamental topic of interest in complex systems theory and in the analysis of complex networks is currently, how network architecture shapes dynamical processes and vice versa.

Progress has been made over the last decade in identifying first ordering principles. One example is the synchronization of oscillators on hierarchical graphs [41]. The time course of the stepwise path towards a fully synchronized system seems to follow the pattern of gaps in the spectrum of the graph, or more precisely its associated Laplacian matrix.

Furthermore, using stylized minimal models has been helpful in revealing some other relationships between network topology and dynamics [42, 43, 44]. Stylized minimal models are null models which incorporate a given dynamic network process. Then through the model one varies the topology of the generated null networks and investigates their influence on the dynamic process at hand.

An interesting alternative to these simulation-driven studies is to explore the relationship between network architecture and dynamics from a data-analysis perspective. Namely, to extract this relationship from large-scale data sets. They can be expected to be produced, at least partly, by the dynamics of the network at hand.

Evidences for network architecture clearly contributing to the patterns observed in data, exist from a diverse range of fields: gene expression patterns, both on the level of whole transcriptional regulatory networks [45, 46, 47] and on the scale of small regulatory devices [25, 26], the epidemic spread of diseases [48] and attack tolerance related to broad degree distribution [49].

The case study investigated here is a special case of complex networks. The authors are nodes and edges represent joint publications, together defining the topology of the network. The dynamic process then taking place on that network is the citation of scientific publications.

2.2 Background on Co-authorship Networks

Co-authorship networks are probably the most extensively studied subclass of complex networks. On the one side they represent large, well-defined social networks, almost a luxury in the field of social science. On the other side, they are a prominent subclass of complex, self-organizing networks and a suitable testbed for many general theories on complex networks.

Motivated by the large amount of electronically available data, scientists across numerous fields have analyzed co-authorship networks. As a result, a substantial body of research dedicated to those networks has emerged.

To better place the scope of our own work, it is therefore essential to give a short overview of the already achieved main results.

2.2.1 Statistical Properties

Co-authorship networks are a snapshot of the knowledge production system. They are simultaneously shaped by the social aspects contributing to scientific activity and the topical organization of knowledge [11, 50, 51].

Early studies in the mid-1970s [52], in spite of the limited access to data, already extracted some surprising statistical properties within co-authorship and citation data, see [53, 54].

A giant leap towards analyzing the large-scale organizational features of the system came of course with the shift towards electronically available publications, see [50, 55, 56].

2.2.2 Small World and Average Network Properties

One of the first large-scale analyses was conducted in [11]. It confirmed that co-authorship networks indeed have the small world property, i.e. that any two scientists are separated only by a very small number of intermediate collaborators.

More importantly, the above study revealed that there are some scientific fields, like high energy physics, where the average network properties are dominated by a few individuals with many collaborators. Still, in most scientific fields the average network properties are indeed governed by the large number of scientists with just a few collaborators.

2.2.3 Community Structure

A very rich topic in the discussion of co-authorship networks is the centrality of authors and the network's community structure.

Repeated removal of the most central edges (sum of the betweenness values of the end nodes) is for example used in [15] to determine the community structures within the network.

Alternatively, [16] applies spectral theory to analyze the community structure. In fact, co-authorship networks have frequently served as an application example for module detecting approaches, including all ground breaking studies in this area.

2.3 Motivation and Related Work

The numerous studies described above clearly show that the topology of co-authorship networks is an extremely interesting object of investigation.

However, they all so far leave unexplored a certain aspect of co-authorship networks. Namely, the dynamic processes taking place on these networks. We believe that relating the topology to dynamical processes can yield outstanding insights into the functioning of the scientific system and some aspects of social dynamics.

The search for fundamental relationships between network architecture and dynamical data is the guiding principle underlying our investigation. In order to identify such relationships for co-authorship networks, we explore the distribution of impact of publications across few-node subgraphs in those networks.

The main conceptual idea of few-node subgraphs as a means of exploring complex networks is that one looks at network properties and network function at a well-defined intermediate scale between the whole network and the individual node.

To our knowledge, the only study that indirectly and only partially addresses network organization at the level of constellations of few collaborators is [57]. They explore the connection between team assembly mechanisms and the structure and performance of collaboration networks, including co-authorship networks.

The parameters of their team assembly model are the fraction of newcomers in a team and the probability of repeating previous collaborations. In this way they have been able to identify a phase transition towards a large connected component, as well as other structural network properties directly linked to the underlying process parameters.

The general relation between team properties and impact has also been addressed by [55], showing that teams produce more frequently cited research than individuals. This trend is increasing over time, is visible across many disciplines (from the natural sciences to the humanities), and includes the very high-impact research. A domain so far traditionally, but obviously falsely associated with the single-author “genius”.

Those two, most close to our work studies, still focus only on individual publications, rather than the intermediate network scale of few-node constellations.

2.4 Graph Representation

Here we adopt a specific definition of co-authorship networks. The nodes are authors and two authors are connected by an edge if, and only if, they have ever published together.

In this representation, one loses the separation of authors into distinct publication co-authorships. This information would be retained in a bi-partite graph representation. In a

hyper-graph representation, one also would be able to retain the grouping of authors beyond the two-author level, in terms of their institutions for example.

However, the uni-partite representation is particularly suited for our purposes, because of the enormous amount of graph-theoretical methods and empirical intuition available for exploring their statistical properties.

This very representation has already lead to remarkable successes in understanding systems of scientific collaborations [11, 12, 50, 53, 55, 58, 59, 60, 61, 62, 63, 64].

2.5 Findings: the Success of the Box Motif

We define the success of a motif as the average citation frequency per edge of all involved publications, i.e. all collaborations represented by this edge. By crawling Google Scholar [39] and CiteSeerX [38], we extracted a database of citation frequencies for a large subset of the publications entering our two co-authorship networks, DBLP [37] and CiteSeerX [38]. For more technical details, see Section 2.9

The extracted citation frequencies serve as our surrogate measure for the impact of publications. We measure the success of a publication by the number of citations by other publications.

2.5.1 Converting Publication Impact to Edge Weight

A crucial step is to convert the impact of publications into edge weights in the co-authorship network representation. This conversion can be done in several different ways.

For an edge e , let $P(e)$ denote the set of publications represented by e . For a publication p , let $c(p)$ denote the citation frequency of p and $A(p)$ the set of authors of p . The four possible edge weight w_e definitions are then as follows:

$$w_e := \sum_{p \in P(e)} c(p) \quad (2.1)$$

$$w_e := \frac{1}{|P(e)|} \sum_{p \in P(e)} c(p) \quad (2.2)$$

$$w_e := \sum_{p \in P(e)} \frac{c(p)}{|A(p)| - 1} \quad (2.3)$$

$$w_e := \frac{1}{|P(e)|} \sum_{p \in P(e)} \frac{c(p)}{|A(p)| - 1} \quad (2.4)$$

where $|S|$ denotes the number of elements in the set S .

The citation frequency of a publication can thus contribute to an edge weight either directly or normalized via the number of authors of that publication. Similarly, the frequencies of all publications contributing to an edge can either be summed up or averaged. These are the four variants of converting publication frequencies into edge weights given above.

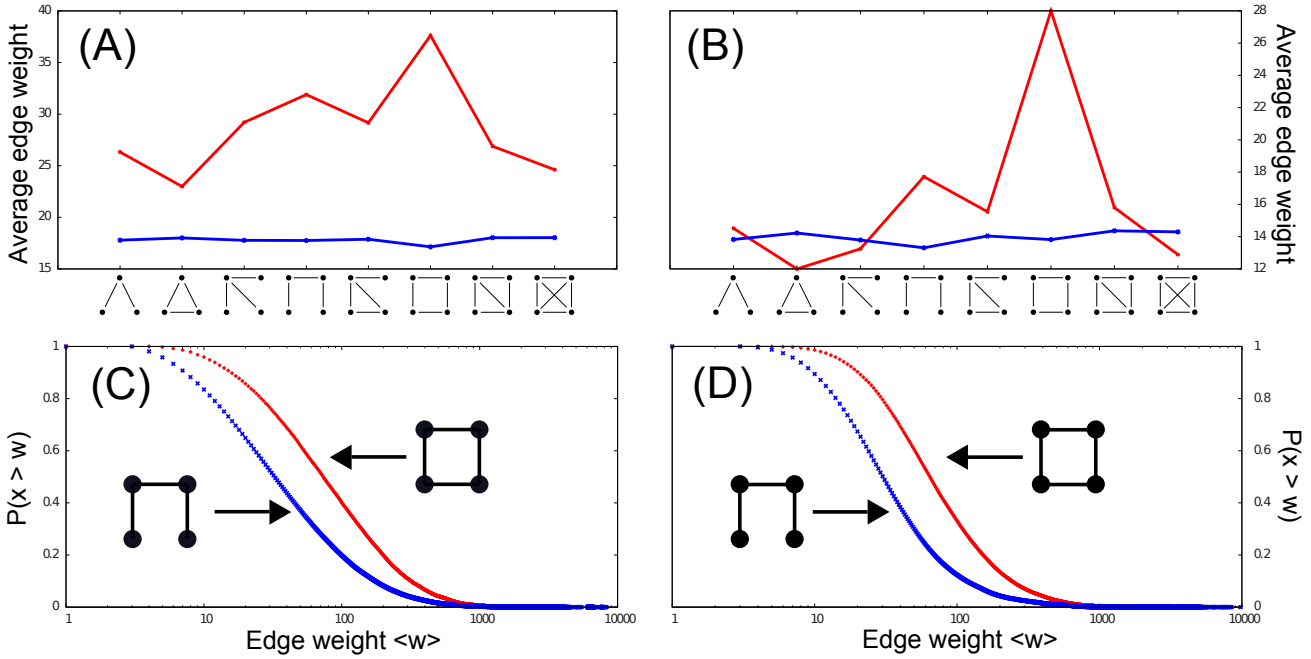


Figure 2.1: The average edge weight per motif compared to the null model for DBLP (A) and CiteSeerX (B), according to edge weight definition from eqs. (1) and (3), respectively. In order to resolve the data behind the averages from (A) and (B), the cumulative distributions of the edge weights for two of the motifs are shown, namely the box motif (motif 6) and motif 4, for DBLP (C) and CiteSeerX (D).

Note that it is not *a priori* clear which of the four normalizations is the most proper one for mapping the citation frequencies onto the co-authorship network.

When some normalizing quantity depends on a network property which varies across motifs, the normalization will affect the average edge weight per motif. Even when a motif has no direct shaping influence.

For example the number of authors of a publication or the number of publications per edge may depend on the degrees of the involved nodes. However, the degree of a node is a network property which also varies across motifs, see Figure 1.1.

2.5.2 Main Result

For our main result shown in Figure 2.1, the average edge weights for the different motifs from Figure 1.1, we select the normalization that most successfully eliminates the above described residual dependences.

We need a proper null model in order to estimate which edge weight normalization most effectively eliminates residual dependencies. For this purpose, we permute the citation frequencies of all publications. Then we convert them into edge weights again and re-compute the average edge weights of the motifs in this null-model scenario of shuffled citation frequencies. A uniform distribution of these null-model edge weights across the motifs indicates a successful elimination of the residual influences, as demonstrated in Figure 2.1.

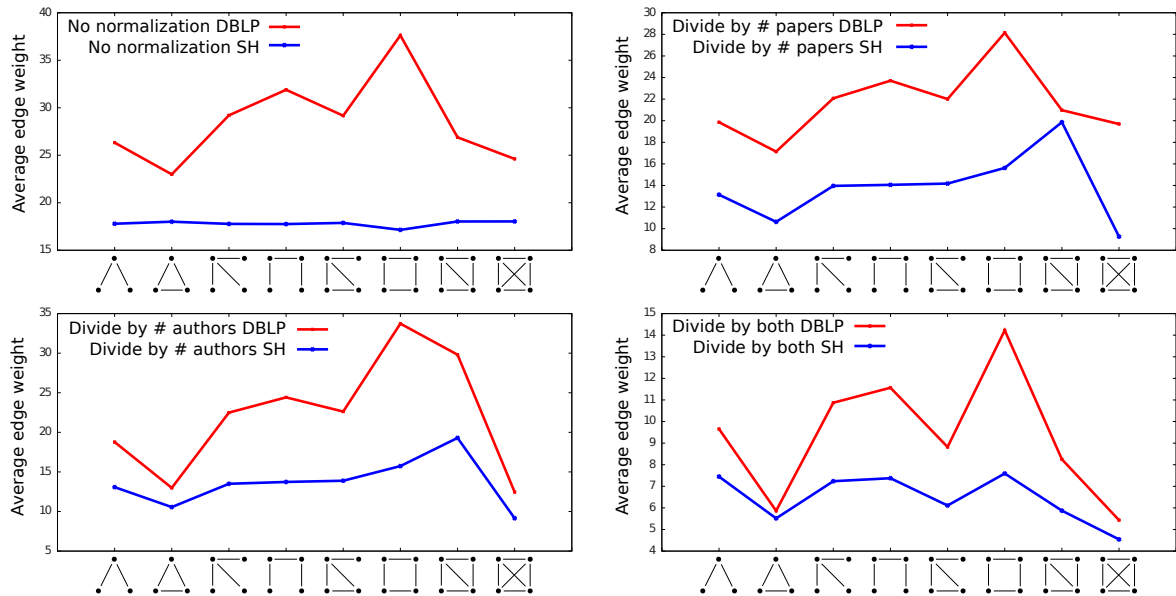


Figure 2.2: The average link weight per motif in DBLP for all four edge weight definitions compared to the shuffled null model denoted by SH.

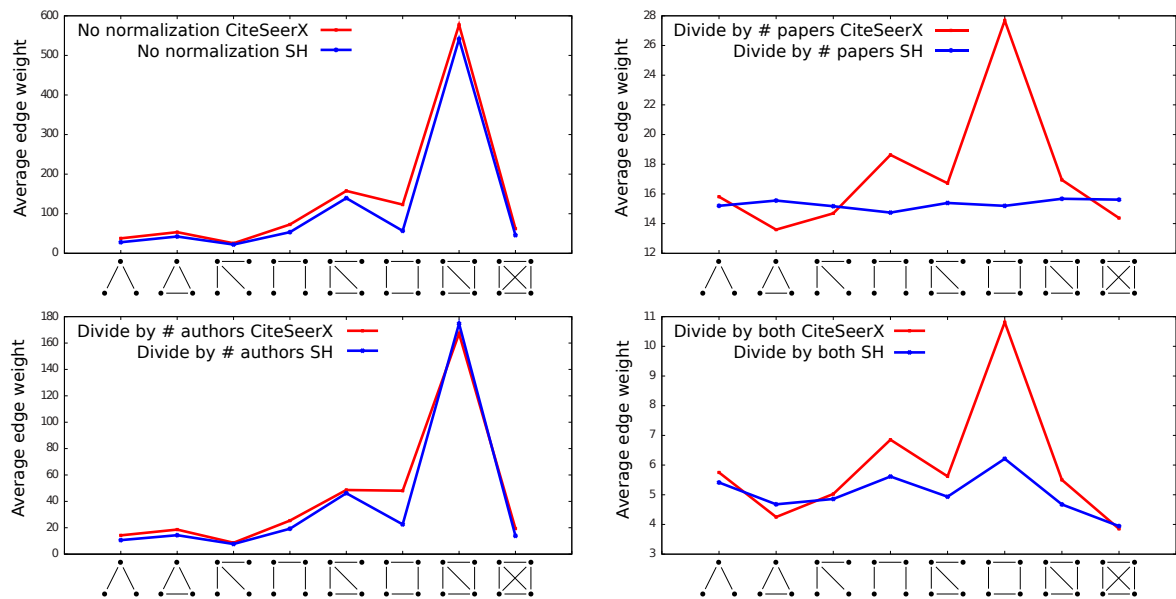


Figure 2.3: The average link weight per motif in CiteSeerX for all four edge weight definitions compared to the shuffled null model denoted by SH.

Note that in contrast to many network analyses, we do not randomize the network architecture, but rather shuffle the dynamical data on top of it. In this way we cannot discuss possible deviations of motif counts from randomness, but only the effect the motifs have in shaping the dynamical output of the network.

The distributions of average edge weights across the motifs for the remaining normalization schemes and both databases is shown in Figure 2.2 and Figure 2.3 respectively.

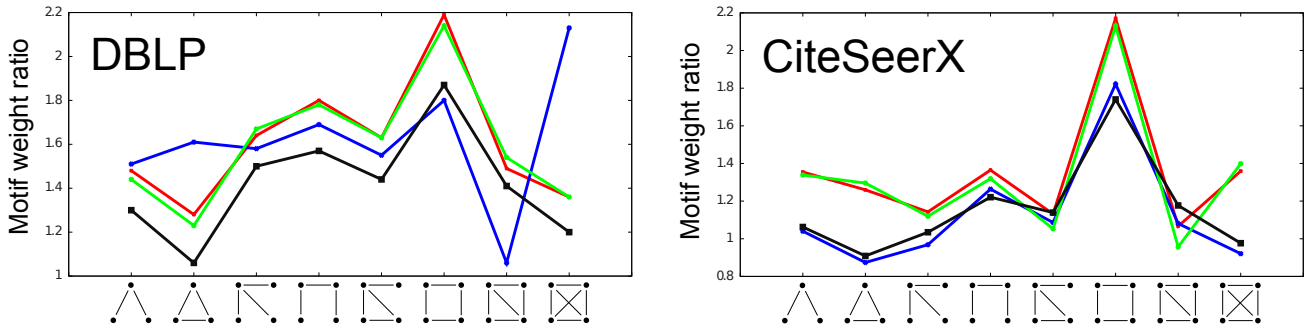


Figure 2.4: Ratio of average edge weights real data/null model for the edge weight definitions 1, 2, 3 and 4, DBLP on the left side and CiteSeerX on the right side.

It is clearly visible that not all normalizations yield flat distinction across the motifs for the shuffled citation frequencies. However, in all four normalizations and for both data sets, the **box motif** (motif 6) has the **highest ratio** to its null model counterparts. This is clearly shown in Figure 2.4.

Note that the average weight shown in Figure 2.1 is the weight *per edge* in a motif. Hence, differences in the number of edges between the different motifs do not affect this quantity directly. Furthermore, the unexpectedly high average weight observed for the box motif is not a trivial consequence of the fact that the box motif needs a minimum of four distinct publications for its construction. In fact, the box motif is no outlier with respect to the number of publications nor the number of authors per edge, see Section 2.8.

Robustness of Our Findings over Time

As a main test of robustness of our finding, we construct time-truncated versions of the co-authorship networks for the past 20 years. The network for year y includes all publications up to that year. For all the time-truncated networks the full, i.e. current-day, set of citations is used.

In Figure 2.5 the result from Figure 2.1A is thus shown for the time-truncated DBLP networks from 1990 up to 2008. The **box motif clearly stands out** as the motif with the highest average edge weight across all years. It should be noted that this time-resolved analysis of motif-related patterns in citation frequencies reveals some interesting additional features. For example the change in importance of motif 7 with respect to motif 4, probably associated with a trend towards denser collaborations and hence denser motifs.

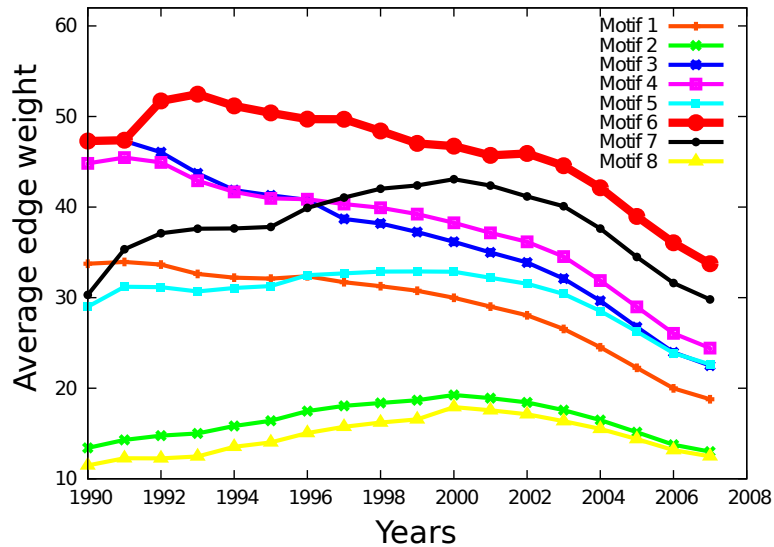


Figure 2.5: The average weight per motif link over the years for the DBLP database.

2.6 Deeper Look: Separation

A typical occurrence of the box motif in the co-authorship networks is shown in Figure 2.6. This example helps us to look deeper into the specific mechanisms behind the box motif.

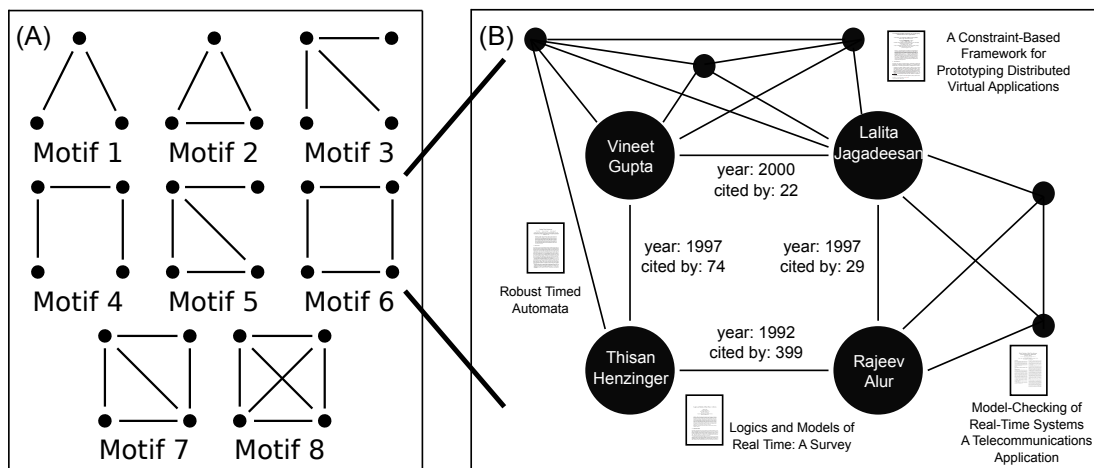


Figure 2.6: (A) The eight possible undirected three- and four-node motifs. (B) Example of a single occurrence of motif 6 (box motif) based on only four publications and embedded in the local network generated by these publications.

Topologically, the surprising feature of the box motif is the lack of the two cross links. The box motif is in this sense an “anti-clustered” motif. This “anti-clustering” is related to a segregation of the two pairs of involved authors, either geographically, temporally or with respect to their scientific disciplines.

In other words, we expect that strong segregation in space, time or discipline exist. In the following, we explore the nature of this separation from various angles.

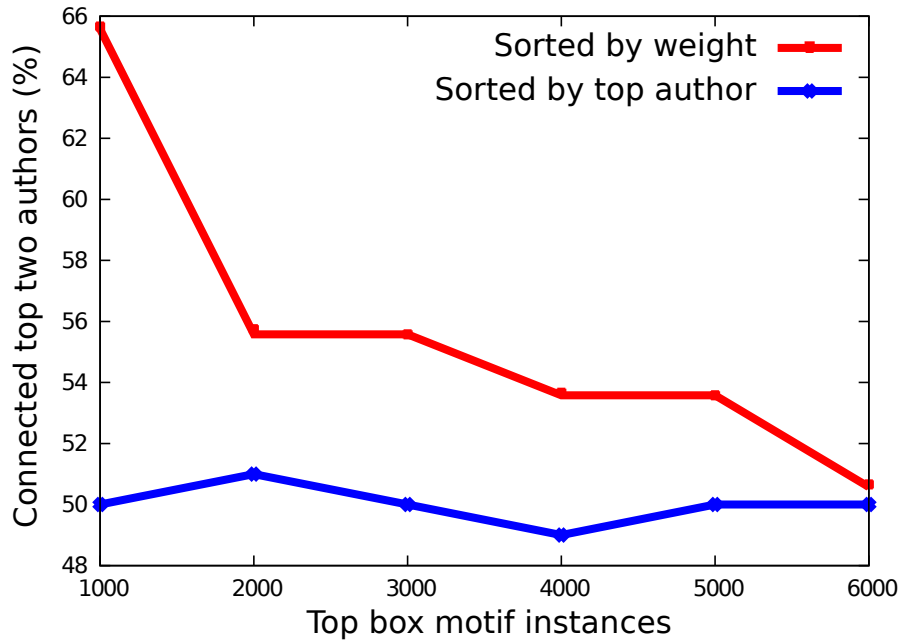


Figure 2.7: Percentage of box motif instances in DBLP where the top two authors are connected directly. The box motifs instances are divided in chunks of 1000 instances and sorted in descending order with respect to their weight.

We first address the question whether in the successful box motif cases the two established authors are directly linked or not.

2.6.1 Separation in Rank: Established Authors and Newcomers

One can use the number of citations of an author as a surrogate measure for how well this author is established. Despite deviations from that rule, experienced and prominent authors are expected to have more citations than newcomers to the scientific community.

In our co-authorship networks, we define the weight of an author as the total number of citation of that author. According to the third normalization scheme for edge weights, see equation 2.3 from Section 2.5, the author weight then corresponds to the sum of the weights of all edge linked to that author. The computed node weights can be used to sort the authors in box motif instances according to the number of their citations.

First, we partition all occurrences of the box motif into chunks of thousands. The first chunk comprises the 1,000 motif occurrences with the highest commutative weight, the second chunk contains the 1,000 next highest ones, and so on. Then we count the number of box motif instances where the two authors with highest weights are directly linked by an edge.

Next, we repeat the same procedure, but this time we sort the box motif instances not according to their weight, but rather according to the maximum weight of the involved authors. The computed results are displayed in Figure 2.7.

Our results clearly show that the higher the weight of a chunk, the more boxes can be found in this chunk such that the two strongest authors are adjacent. However, this is true only when one sorts the box motifs according to their commutative weight. This effect vanishes when the boxes are sorted according to the heaviest involved author. That is a clear indicator that it is the collaboration pattern that matters and not the individual authors involved in that pattern.

Our results also indicate that the success of the box motif partially comes from the separation of authors in rank. Well established authors publish together and eventually their students or remote collaborators also publish together.

Unfortunately, the data available to us does not allow to inspect geography or discipline structure directly. To substantiate our claim, we make two further computational studies to understand how important the segregative features are for the success of the box motif.

2.6.2 Separation in Time

We look next at the construction time of motifs. The edge initiation is given by the year of the first publication constituting this edge. For a motif occurrence, the construction time is the time between the earliest and the latest year of initiation of an edge within this occurrence.

For example, if the authors *A*, *B* and *C* all have published with each other, *A* and *B* in year 2000, *A* and *C* in year 2002, and *B* and *C* in year 2004. Then, the construction time of the clique *A*, *B* and *C* form is 4 years. Even if *A* and *B*, or any other pair combination publishes later on, it is the first time the two authors are connected that matters. That is, if *A* and *B* have also published together in 2004, the initiation time of their edge stays 2000 and so the construction time of their clique with *C* stays 4 years.

Figure 2.8 shows, for each motif and each construction time, the relative average weight of all occurrences of this motif that have the same construction time.

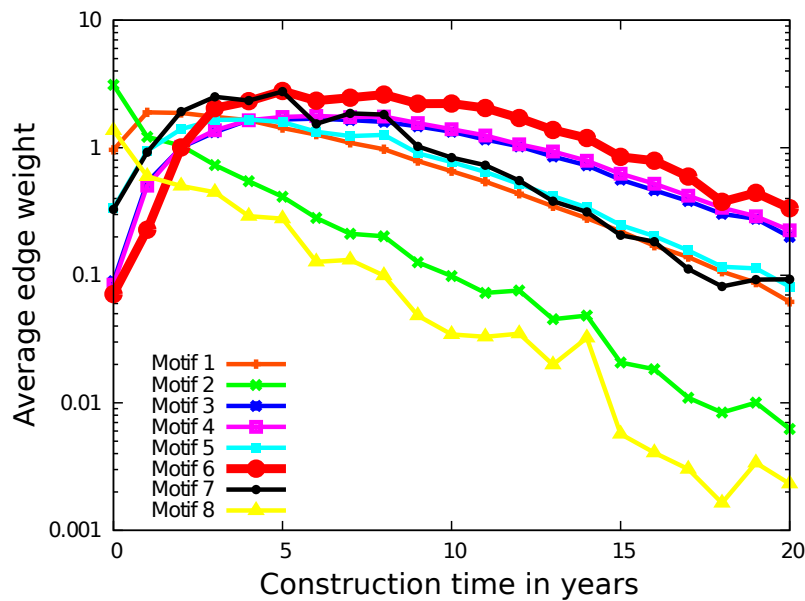


Figure 2.8: Relative average edge weight per motif. All motif instances are distributed in bins according to their creation time and the average weight per bin is displayed.

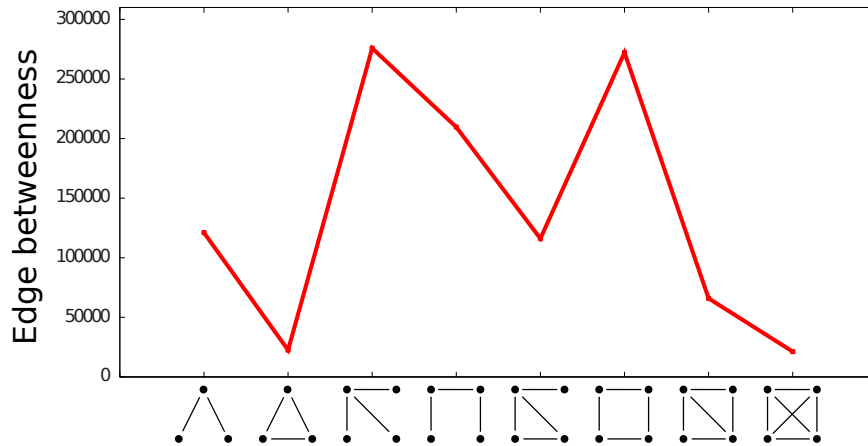


Figure 2.9: Average number of shortest paths passing through a motif edge for the 1990 snapshot of the DBLP (all publications dating before or from 1990).

It is clear that the box motif has a significantly stronger tendency than all other motifs for its heavy-weight occurrences to have long construction times. Thus, the heavy-weight occurrences of the box motif seem to span a bridge over time.

2.6.3 Separation in Scientific Area: Interdisciplinary Collaborations

Finally, we look at how the motifs, and in particular the box motif, are distributed across the co-authorship network. The aim is to investigate whether the box motifs lay dominantly within clusters of connected nodes, or rather among such clusters, indicating a certain degree of interdisciplinary collaborations.

Edge betweenness is a centrality measure that estimates whether an edge lays within a cluster of nodes, or connects two such clusters. The betweenness of an edge is the number of shortest paths between node pairs that go through that edge. Edges between clusters have very high betweenness, as all the shortest paths among both clusters go through those edges.

Obviously, edge betweenness is perfectly suitable for our analysis. Therefore, we compute the edge betweenness of all edges in the co-authorship network and use them as edge weights.

Figure 2.9 shows the average number of shortest paths that use edges of occurrences of a particular motif (normalized by the number of edges in this motif). Clearly, the box motif edges, together with those of motif 3, constitute high betweenness values and hence lay often on paths between larger communities within the network.

Our results are a strong indicator that the box motif is to a certain extent related to interdisciplinary collaborations.

2.7 Supporting Experiments

In the following, we carry out a series of experiments showing that the network properties of our two co-authorship networks comply with results from related work. We also show that the computed and presented in Section 2.5 average values are well defined and legitimate. We also

exclude trivial effects, such as the number of authors per publication or publications per edge, as responsible for the presented in Section 2.5 findings.

2.7.1 Network Properties

To assure that our two databases comply with already investigated co-authorship networks, we compute a set of network properties usually discussed in related work. These include degree distribution, citation distribution and average clustering coefficient.

None of the computed network measures shows a significant deviation from already published results on collaboration databases. The results are displayed in Figures 2.10, 2.11 and Table 2.1.

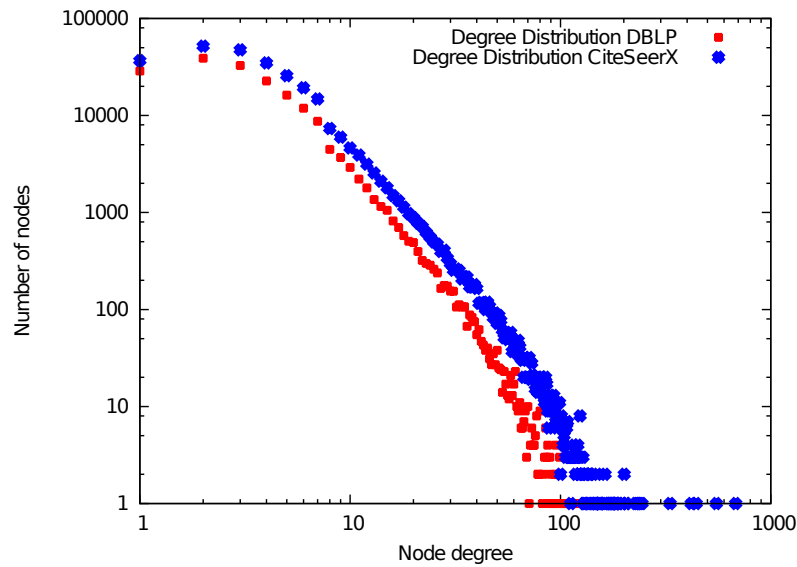


Figure 2.10: Degree distributions of DBLP and CiteSeerX.

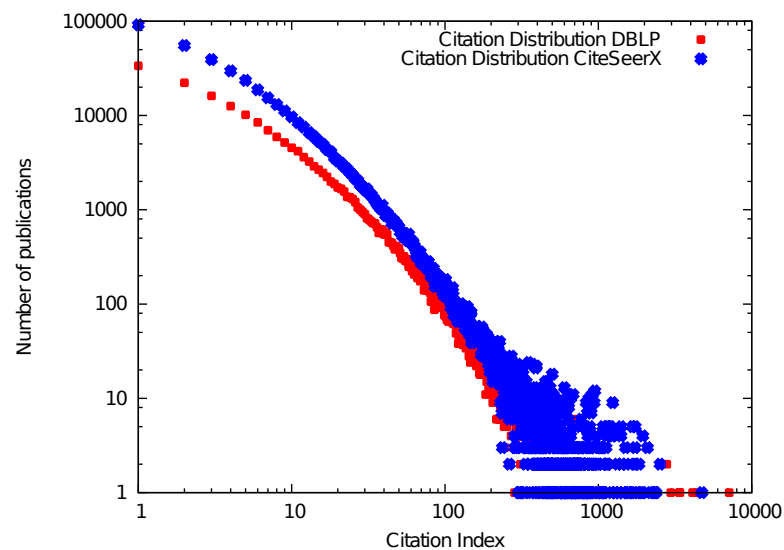


Figure 2.11: Citation distributions of DBLP and CiteSeerX.

Network	Authors per Paper	Papers per Author	Clustering Coefficient
DBLP	2.74	4.04	0.658
CiteSeerX	2.69	3.26	0.667

Table 2.1: Average number of authors per paper, papers per author and clustering coefficients for the DBLP and CiteSeerX databases. All values comply with results on co-authorship networks from related work.

The average number of authors per paper and papers per author are very similar to the same quantities computed on many other co-authorship networks investigated in related work [11, 58, 65, 66]. The same holds for the high average clustering coefficient, typical for social and co-authorship networks.

All four degree and citation distributions follow fat tail power law, characteristic for all so far investigated co-authorship networks.

2.7.2 Weight Distributions and Average Values

In Section 2.5 we have presented our main finding, namely the difference in average weight per edge across the various three- and four-node motifs. To assure that the computed average values are well defined and legitimate, we investigate the whole motif weight distributions instead of just looking at their average values. The results are displayed in Figure 2.12.

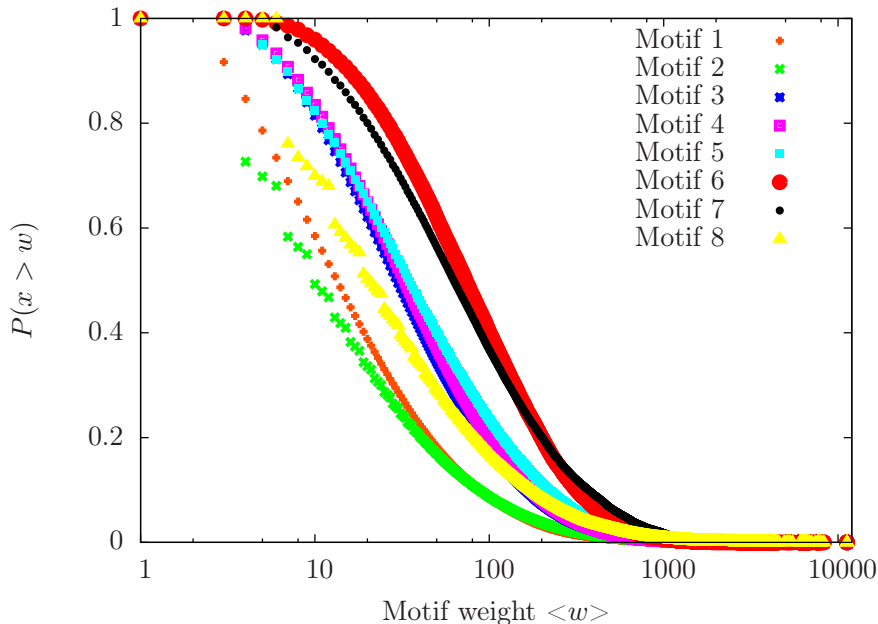


Figure 2.12: The motif edge weight distributions for all eight motifs within the DBLP database.

All eight distributions are monotone and governed by the box motif as can be seen from Figure 2.12. Therefore, the computed average value are also well defined. Furthermore, the

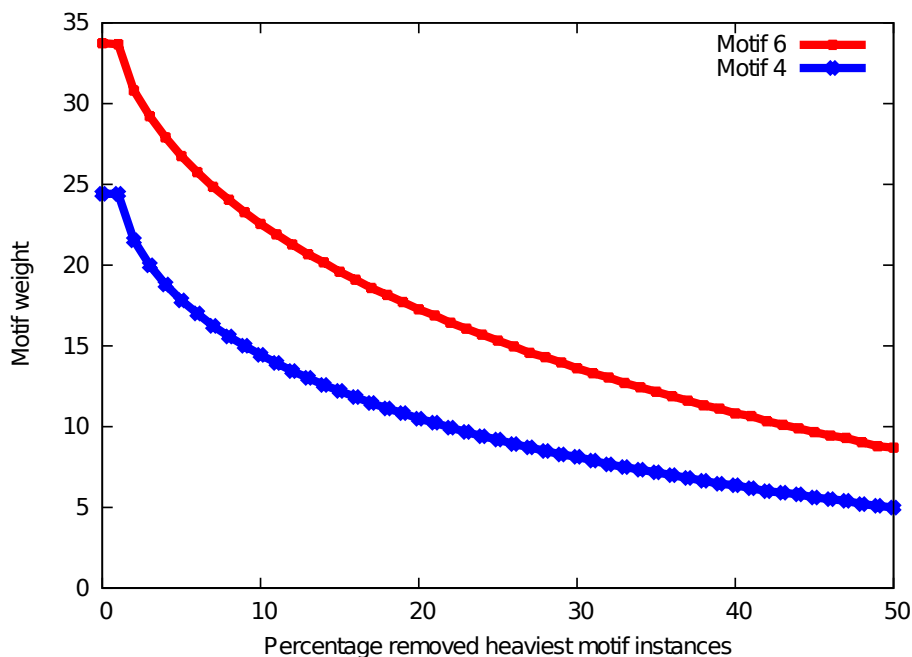


Figure 2.13: The effect on the average motif edge weight when one gradually removes the heaviest instances of that motif.

dominance of the box motif does not come from a few motif instances with extreme values, but is rather dictated by the whole weight distribution.

The motif weights are computed over the whole database and with respect to edge weight definition 2.3 from Section 2.5.

2.7.3 Most Successful Motif Instances

Our next step is to investigate how the average motif edge weight changes when one consistently disregards the heaviest motif instances when computing the mean values. Aim of our analysis is to show that it is not the top motif instances that make the box motif so successful, but rather all of them taken together.

Again we investigate the whole DBLP database under edge weight definition 2.3 and take motif 4 as a reference. The results are shown in Figure 2.13.

One observes that the average motif edge weight reduces gradually for the box motif as well as the reference motif 4. The high average value of the box motif is not a result of a few extremely heavy instances, but is rather dominated by the high number of intermediately heavy instances.

2.7.4 Eliminating Trivial Effects

Up to now we have shown that our two databases comply with related work on co-authorship networks. Furthermore, the presented in Section 2.5 mean values are justified and are not influenced by a few extreme values.

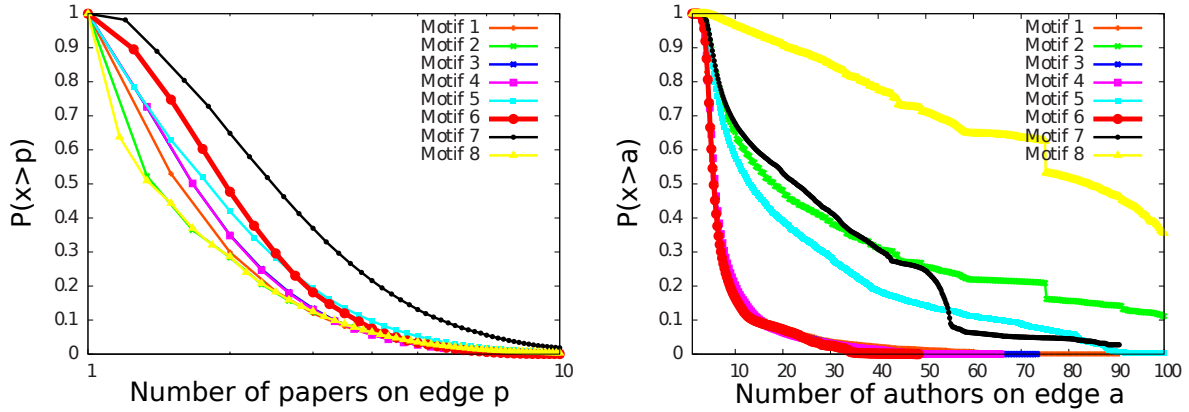


Figure 2.14: The number of papers respectively co-authors per motif edge for DBLP.

Finally, we want to tackle one last aspect. Namely, to exclude trivial effects as the number of papers and the number of their authors as possible causes for the success of the box motif.

Note that the four edge weight definitions introduced in Section 2.5 implicitly address the above issue. They integrate the number of papers between a pair of authors, the number of co-authors on those publications, or both effects simultaneously. Otherwise, one can assume that the high average value of the box motif comes from one of those two effects.

Recall from Section 2.5.2, that independently of the edge weight definition, the box motif was still the most *successful* one. To exclude any doubt, we have calculated the average number of publications between a pair of authors in all motifs, as well as the number of co-authors on those publications. The results are displayed in Figure 2.14.

One clearly sees that the box motif neither profits from a high number of papers running through its edges, nor have those publications significantly few authors. The box motif does not dominate any of the two distributions and in both cases there is at least one other motif with comparable values. The prevailing weight of the box motif is not a result of any trivial effects one may suspect.

To conclude, we carried out a set of supporting experiments on the analyzed data. We have observed that the properties of the investigated co-authorship networks comply with related work. Furthermore, we showed that the presented results are well defined and justified, as well as that they do not come from certain trivial effects.

Consequently, the success of the box motif revealed in Section 2.5 is apparent and undeniable.

2.8 Further Analysis: Generative Model

In this Section we briefly sketch a possible generative model. Generative models are null models which incorporate the network properties or dynamic process one is interested in. Then through the model one systematically varies the topology of the generated null networks, which reveals their influence on the network observable at hand.

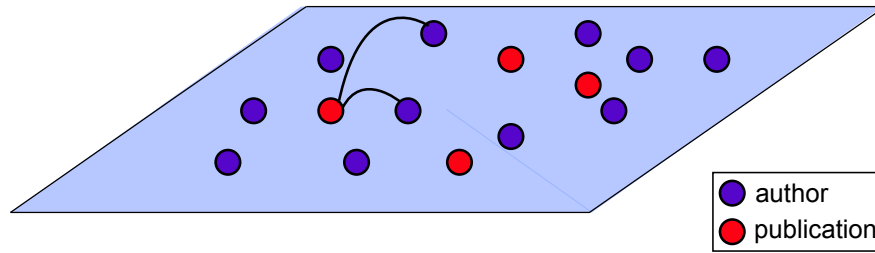


Figure 2.15: Schematic representation of the content proximity plane. Distance among authors and publications enters the computation of the scores, equations 2.5 and 2.6.

In that context, we use our generative model as a convenient analytical framework for exploring the relation between impact of scientific publications and topological properties of the underlying co-authorship network.

2.8.1 The Model

We assume the authors and publications to be distributed on a plane, which we call the “content proximity plane”. The two elementary processes in our model are the writing of a publication, paper production, and the citing of already existing publications in new ones, citing articles.

In the case of paper production, the content proximity plane is used to select authors from with a probability α_1 . With probability $(1 - \alpha_1)$ authors are selected at random. When the proximity plane is used another parameter β_1 regulates whether authors are selected according to impact or proximity.

For the second process, citing articles, the parameters α_2 and β_2 have the same function for selecting publications to be cited, analogically to selecting authors.

To simulate the two processes, paper production and paper citation, the number of authors N and the number of publications M has to be selected, as well as two distributions: authors per paper and citations per paper. Furthermore, to reflect the process of aging we introduce another parameter A as the maximal number of publications of an author.

Then, the workflow of the generative model is as follows: Choose the number of authors N and place them in the content proximity plane. Choose the number of publications M . For each publication choose the number of its authors, k , and place the publication into the content proximity plane. Then, choose k authors from the plane according to their proximity and impact, and *publish* the paper. Finally, choose the number l of existing publications the new publication should cite and choose those publications similarly according to their proximity and impact. This process is illustrated in Figure 2.15.

For a given publication p , we compute a score for each author in the plane. Then, we choose the k authors who should *write* p from the distribution of all author scores. The score of an author a is given by:

$$Score(a) := \alpha_1(Rank(a) + 1)^{\beta_1} e^{-\frac{\Delta_{ap}}{2}} + (1 - \alpha_1) \frac{1}{N} \quad (2.5)$$

where Δ_{ap} is the Euclidian distance between a and p in the proximity plane and $Rank(a)$ is the number of citations of all publications already published by a . In other words, α_1 balances between random assignment of authors to papers and between selecting the authors according to their impact, large β_1 , or their proximity in the plane to p , small β_1 .

For $\alpha_1 = 0$ all authors are chosen at random with probability $\frac{1}{N}$. For $\alpha_1 = 1$ the probability of an author a to be selected is governed solely by a 's rank and its distance to p . In that sense, α_1 linearly balances between these two extremes. For $\beta_1 = 0$ it is only the distance between a and p , measured as $e^{-\frac{\Delta_{ap}}{2}}$, that determines the probability of a to get selected for p . By choosing larger values for β_1 one assumes that established authors may get selected as co-authors, even if the paper is not in their concrete scientific area. I.e. an author a with a high rank still has good chance to be selected for large β_1 , even if the distance $e^{-\frac{\Delta_{ap}}{2}}$ of a to the paper p is significant.

After an author has published her/his first publication, it stays in the proximity plane for the next A publications. Afterwards, the author is marked retired and taken down from the plane and thus from the list of available authors for further publications.

In analogy to the paper production process, we select the papers each new publication should cite from the distribution of all paper scores. The score of a paper p is given by:

$$Score(p) := \alpha_2(Rank(p) + 1)^{\beta_2} e^{-\frac{\Delta_{pp_{new}}}{2}} + (1 - \alpha_2) \frac{1}{M} \quad (2.6)$$

where $\Delta_{pp_{new}}$ is the Euclidian distance between p and the new publication p_{new} , and $Rank(p)$ is the number of citations of p . In other words, α_2 balances between random citation of papers and between citing papers according to their impact, large β_2 , or their proximity in the plane, small β_2 .

Once all M papers have been *published*, we extract the collaboration network by connecting any two authors that have published together and assign the citation frequencies as edge weights according to definitions 2.1 through 2.4 from Section 2.5. Hence, our model produces weighted co-authorship networks.

Although our model naturally reflects the paper production and paper citation processes, it has a rather large and heterogeneous parameter space. One has to choose the lifetime of the authors A , all α_1 , β_1 , α_2 and β_2 , as well as the distribution of authors per paper and citations per paper.

2.8.2 Evaluation

Our aim is to check whether our empirical findings can in principle be reconstructed through our model. We take the DBLP snapshot from 1990 and approximate the network using simulated annealing with respect to degree distribution, citation distribution and motif content.

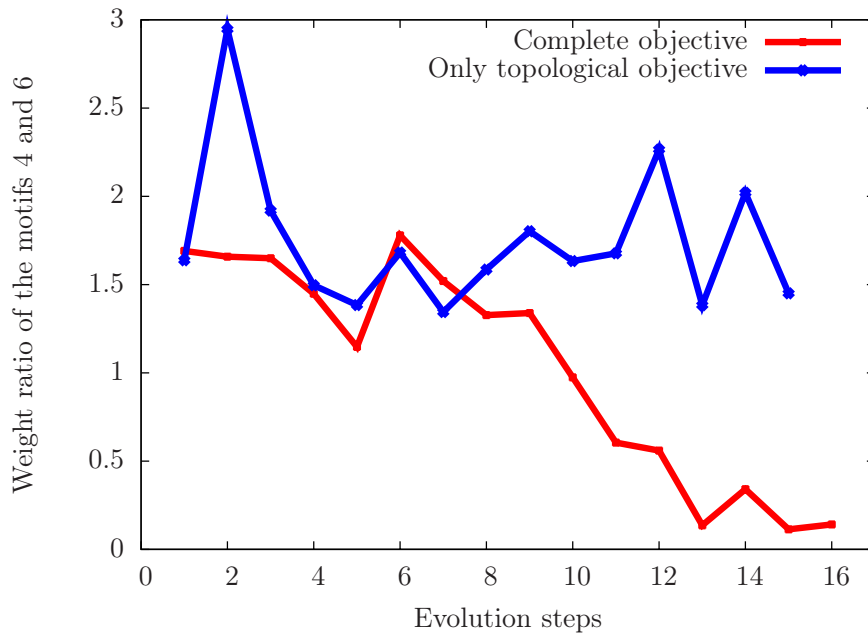


Figure 2.16: Approximating the DBLP snapshot from 1990. Once with respect to degree distribution, citation distribution and motif content only, and once augmented with the ratio in weight of motif 4 to motif 6.

We take the same number of authors and papers as the original network and the empirical distribution of authors per paper. The distribution of citations per paper cannot be reconstructed from our database. Therefore, each new paper in our model cites 10 already existing papers.

Indeed, the co-authorship networks generated by our model allow us to repeat the motif analysis presented in Section 2.5. Therefore, we perform two different evolutions based on simulated annealing.

In the first case we aim at the degree distribution, the citation distribution and motif content of the real world network. The objective function is composed of the differences with respect to those three measures between the real world and the generated networks.

In the second case the objective function is augmented with another term, which minizes the difference between the ratio of the weight of motif 4 (i.e. its average citation frequency) to the box motif in the real world and the generated networks. The results of both evolutions are shown in Figures 2.16, 2.17, 2.18 and 2.19.

It is easy to observe that our model very well approximates the real world network with respect to its topological properties. Most importantly, it is also capable of reconstructing the unexpected high edge weight of the box motif (it produces the same dominance of the box motif over the reference motif).

It is worthed to explore and determine the size and the form of the whole solution space. Nevertheless, the preliminary results of our generative model already show that the right combination of simple network processes like aging, paper production, paper citation, as well as social factors like proximity and impact, can reproduce the success of the box motif, revealed through our analysis.

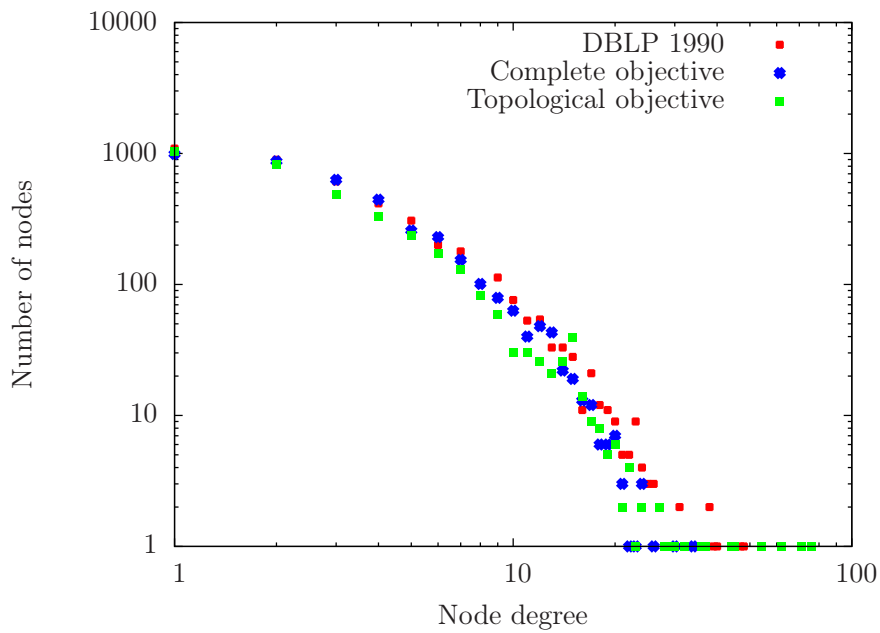


Figure 2.17: Approximating the degree distribution of the DBLP snapshot from 1990. Once with respect to topological properties only and once augmented with the ratio in weight of motif 4 to motif 6.

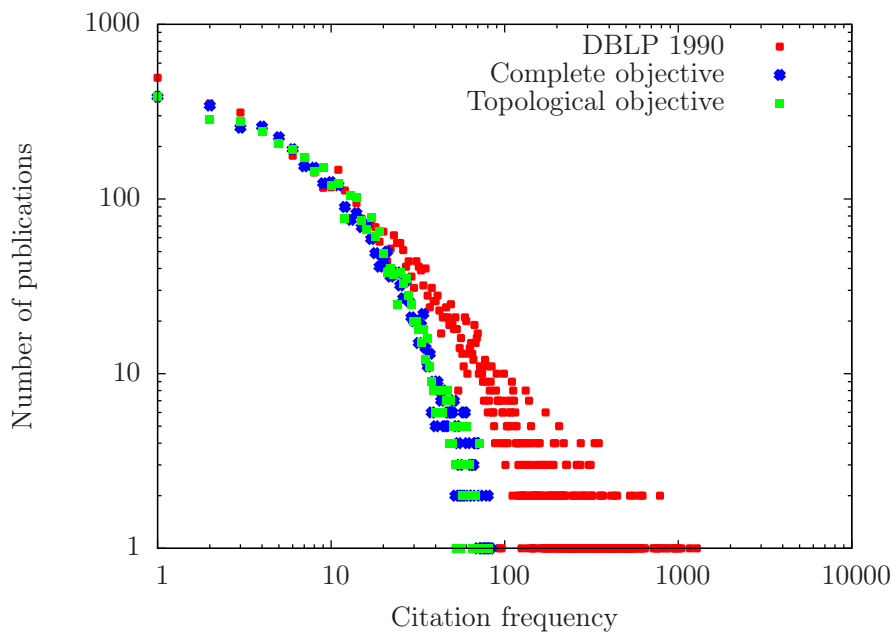


Figure 2.18: Approximating the citation distribution of the DBLP snapshot from 1990. Once with respect to topological properties only and once augmented with the ratio in weight of motif 4 to motif 6.

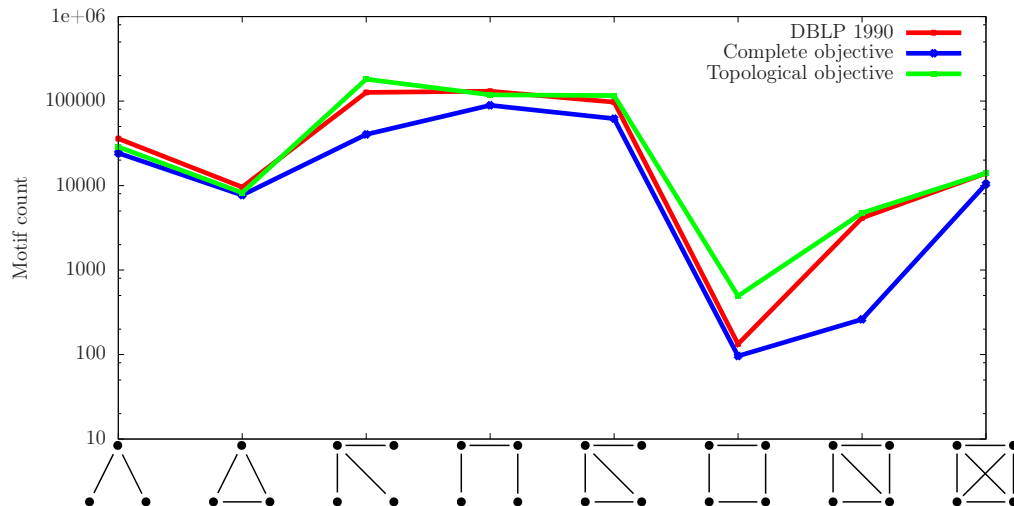


Figure 2.19: Approximating the motif content of the DBLP snapshot from 1990. Once with respect to topological properties only and once augmented with the ratio in weight of motif 4 to motif 6.

Remark: the citation distributions in both evaluations, see Figure 2.18, differ from the original one in their tails. A possible explanation is that we take a uniform distribution for the citations per paper as the real world distribution is not available. Citation networks are known to obey power laws, which very likely causes the deviations in our results.

2.9 Technical Aspects

In the following Section we provide the technical aspects of our study and the analyzed data.

2.9.1 Publication Data

In this work we have investigated two publication databases: CiteSeerX and DBLP

Dumps of the DBLP database in XML format are provided on regular bases by the DBLP official website: <http://dblp.uni-trier.de>. Each publication entry provided in the XML dump contains at least the publication title, the publication year and the list of authors who have co-authored the corresponding publication. The dump investigated in this work is as of May 2008, which contains 599,734 authors and 978,786 publications.

The CiteSeerX is available for download and synchronization through the CiteSeerX official website: <http://citeseerx.ist.psu.edu>. The data download and synchronization is available through Open Archive Initiative Harvesters. We developed our own harvester to acquire the publication data available in CiteSeerX as of October 2009. Each publication entry contains at least the publication title, the publication year and the list of authors. That snapshot of the online database contains 999,856 authors and 1,247,732 publications.

2.9.2 Citation Indices

In this work we investigate the *success* of collaboration patterns. We project the success of a given publication as the number of citations by other publications. Successful, innovative and ground breaking publications attract interest by other scientists, who then later on refer to those publication in their own work.

Hence, the next step for our analysis was to acquire citation indices for the the publications within the two investigated publication databases. For this purpose we deployed 107 web crawlers compatible with the online publication search engines CiteSeerX and GoogleScholar (<http://scholar.google.com>). All acquired data is publicly available through web interfaces of both search engines. The web crawlers obeyed the time-out policies and request frequencies provided by the search engines and ran as background processes, being even less intrusive than a casual human user.

We requested the title of each publication within the two acquired databases and stored the responses by both search engines (the responses are provided with citation indices by other publications). A response is considered a match, if the title (by trimming white spaces and special characters), the publication year and the list of authors (by trimming white spaces and special characters) were identical to a publication within one of the two acquired databases.

The title of each publication was requested on both search engines. If both of them returned a match, then the citation index for that publication was set to the maximum of both responses.

We were able to acquire non-empty citation indices for 192,688 of the papers within the DBLP database, which is around 19% of all publications. That number excludes publications which were found on the search engines, but still have not been cited by other papers. For the CiteSeerX databases we found 434,794 papers with citation index of at least one, which corresponds to 34% of all publication within the acquired database.

The lower match success for DBLP comes from the fact that it is actually a third party with respect to the search engines. On the other side, we requested the publications provided by CiteSeerX by using the Open Archive Initiative protocol directly from the CiteSeerX search engine itself, leading to a better match ratio.

Note that the citation indices considered in this work are as of their time of acquisition, which for both databases was short after acquiring the publication data.

2.9.3 Co-authorship Graph Representation

In this work we use the natural graph representation of co-authorship networks where the authors are the nodes and two nodes are connected if they have ever published together.

We parsed the publication lists in both databases with the following assumptions:

- A publication is considered unique through its title (trimmed from white spaces), publication year and list of authors. Publications without specified publication year are expelled from our analysis.
- Multiple publication entries with different publication years, but the same title and authors, are considered as distinct publications.
- An author is considered unique through her/his first and family names.

-
- Authors with identical first and family names (as they appear in the database) are considered the same author.

The above assumption may lead to considering two real world authors as the same author in our database if they have the same names. On the other side, if an author uses different signatures on her/his publications, the same real world author may be considered as two distinct authors in our database. There is no way around this problem and its impact was already investigated by related work [11]. The number of authors and publications within both databases presented in Section 2.9.1 are the result of the above assumptions.

Furthermore, both databases contain entries representing online reports and websites, listed with several hundreds of authors. To clean up the databases from such entries, we excluded from our analysis all publications with more than 8 authors. For DBLP those were less than 0.6% of all publications and 1.7% for CiteSeerX.

As we were interested in citation frequencies and their interplay with topology, we also excluded all publications with none or zero citations, i.e. publications that are not in the databases or have yet no citations respectively. Thus, our analysis was performed on 190,893 of all 978,786 publication entries within the DBLP snapshot and 430,233 of all 1,247,732 publication entries within CiteSeerX.

After acquiring both databases and available citation indices, we build a graph representation of each database based on the publication entries with citation index of at least one and less than 9 authors. Each distinct author is represented by a node and two nodes are connected if they have ever coauthored a publication together.

2.10 Summary and Outlook

In this Chapter we have analyzed co-authorship networks on a well-defined intermediate scale, their motif content. We investigated the relation between the underlying topology of the network and the dynamical processes taking place on top of that network. Those processes are the production of new articles and the citation of already existing publications.

Our analysis revealed that some collaboration patterns are much more successful than others, measured as the average number of citations. In particular, the box motif, four authors forming a closed chain without chords. The segregative power of the box motif seems to be crucial. It is the collaboration patterns that matters, rather than the involved collaborators.

2.10.1 Summary

We have analyzed two large publication data sets: CiteSeerX and DBLP. In order to measure the success rate of different collaboration patterns, we projected the citation frequencies of publications as edge weights on the graph representation of those two co-authorship networks.

We used four different mapping functions for the edge weights, eliminating various trivial effects. Independent of the mapping function and across both databases, there is one collaboration pattern more successful than all others: the box motif, a closed chain of four authors.

The box motif has the highest average citation frequency per motif edge. By constructing retrospective snapshots of the DBLP for the past 20 years, we showed that our findings are robust over time.

We then looked closer at the separation the box motif induces, the segregation its two missing cross edges indicate. It turned out that there are indeed three segregation factors leading partially to the success of the box motif: separation in rank, in time and in discipline.

We sorted all box motif instances according to their weight and found out that the two heaviest authors are often adjacent in heavy box motif instances. This effect vanishes when one sorts the boxes according to their heaviest authors. I.e. it is the collaboration pattern that matters and not the involved collaborators.

We then investigated the construction time of motifs, measured as the time needed for a motif to be constructed. The construction time of a motif is the difference between the initiation year of its first and its last edge. We found a clear tendency for box motifs with long construction times to be dominantly successful.

To measure whether the edges of the box motif span bridges among disciplines, we computed the edge betweenness of all edges in the network. Edges with high betweenness usually connect different clusters of densely connected nodes, rather than laying within such clusters. Investigating the average edge betweenness per motif edge revealed that it is the box motif together with motif 3 that have dominantly higher betweenness values than all other motifs. Hence, the box motifs indicate interdisciplinary collaborations.

To further substantiate our findings, we carried out a series of supporting experiments. They confirmed that the network properties of our databases fully comply with prior studies on co-authorship networks. Furthermore, all derived average values are well defined and justified. Last but not least, the box motif is not an outlier neither with respect to the number of publication per edge, nor the number of authors of those publications. The prevailing success of the box motif is not a result of any trivial effect one could suspect.

Finally, we introduced a slightly complicated generative model incorporating production and citation of publications. Despite its large solution space, preliminary results through simulated annealing show that the right combination of simple network processes can reproduce the success of the box motif. Those processes include aging, paper production, paper citation, as well as social factors like proximity and impact.

2.10.2 Outlook

The “anti-clustering” of the box motif seems comparable with the theory of weak links [67, 68]. Namely, high scientific success is on average associated with publications outside the densely clustered author collaborators. It would be worthwhile analyzing this from game-theoretical perspective, similar to the work of [69] on structural holes [70]. In fact, due to its “anti-clustering” feature, the box motif occurrences can be seen as small-scale versions of the structural holes distributed in the network.

Our generative model shed some light into the processes leading to the success of the box motif. Unexplored feature of our model is that it gives one the opportunity to observe the network evolution. The next level of data analysis to explore are the conversion rates of motifs as the network evolves. Thus, one could not just observe the outcome of the network evolution, but rather investigate how the success of the different collaboration patterns changes over time. One could also let the system evolve beyond the current state of the network and derive predictions which collaboration patterns will be successful in the future.

Another direction for continuing the line of research is to see the co-authorship networks as an example of social networks. And at the same time as a representative of a more generic class of production and distribution systems. In that way, the segregative capacity of the box motif may prove it outstanding in other systems as well. Table 2.2 puts forward several areas of application, where this hypothesis could be tested.

Network Type	Dynamical Observable	Potential Box Motif Role
Acquaintance networks	Gossip	Sites with maximal re-organization
Metabolic networks	Metabolic fluxes	New category of enzyme essentiality
Trust networks	Recommendations	Double reassuring of reliability
Peer-to-Peer	Data exchange	Alternative paths to target peer
Train Connections	Passenger flow	Alternative connections to destination
P2P Live Streaming	Video/Music/TV on demand	Concurrent frame exchange
Routing	Package delivery	Bandwidth separation along routing paths

Table 2.2: Expected applications of the box motif in diverse technological and social networks.

Later on in this work, we indeed show that motifs are important functional entities in technological and communication networks.



3 Motif Based Optimization of Structured P2P Networks: Fair Load

In the following Section we present a novel perspective on network motifs. Instead of using them as a pure statistical measure for investigating static networks, we deploy them within adaptive networks as a distributed approach for topology optimization. To the best of our knowledge, we are the first to use network motifs from a dynamic point of view. Our results reveal the great potential of this new perspective.

Topology adaptation is a vital operation in technological networks. It is frequently implemented as either an external process or a distributed online optimization that relies on gathering knowledge on the overall state of the system. In this Section we propose MBO (motif based optimization), a novel approach that uses network motifs for distributed topology optimization of arbitrary, adaptable networks. In order to give a proof of concept, we chose to optimize structured peer-to-peer overlays towards fair load balancing. MBO is parametrized using target motif signatures, which are derived from exemplary, generated topologies with the desired properties – fair load balancing in the demonstrated case. Our extensive simulations indicate that for CAN [71] and Kademia [72], two different types of P2P systems, MBO leads to well balanced load, while being minimally intrusive.

3.1 Introduction

The topology of complex networks significantly affects their functional and non-functional properties. This leads to the problem of *topology optimization*, which in general aims at adapting a complex network to achieve beneficial properties. The adaptation of these topologies, however, is impossible for a variety of static networks and for networks bounded by functional, spatial or other constraints.

Technological networks on the other hand, like logistic and communication networks, are usually characterized by the freedom to alter their nodes and links, even though with different levels of ease. Therefore topologies of communication networks [65], such as routing in infrastructure or wireless sensor networks, multicast trees [73] and all types of application overlays do benefit from proper topology optimization. Load distribution, resilience and energy efficiency for example are highly related to the network topology.

The optimization can either be achieved implicitly by systematic creation of the network, or explicitly through alteration of the set of nodes or the connections between them. Explicit adaptation requires either local or global knowledge about the network's state. In the case of large distributed networks like the World Wide Web (WWW), the Internet, or overlay networks, it causes significant effort to acquire a real time snapshot of the system, when feasible at all. A centralized topology optimization algorithm that works on the global state is therefore unsuited for large distributed networks. Distributed topology optimization on the other

hand scales well with the network size as it only uses local information. However, current distributed algorithms [74, 75] require application specific knowledge such as position or distance for geometric methods.

In contrast, our contribution MBO is a general approach for distributed topology optimization without the need of any application specific knowledge. The main idea is to engage network motifs, see Figure 3.1, in local decision rules. Using only local knowledge, each node detects the motifs it is part of and hence can determine its local environment. Then, if necessary, any node can take actions to improve its local environment.

A prerequisite of our approach is that given the underlying network and the properties towards which the network should be optimized, one can construct at least a theoretical optimal topology with respect to those properties. Then, a *target motif signature* is calculated based on the optimal topology by measuring its motif content. The target motif signature is in the kernel of the nodes' local decision rules. Comparing its local motif content to the target signature, each node determines whether it should adapt its local environment. In consequence of the local changes, the overall topology of the underlying network shifts towards the topology of the optimal network and so do its desired global properties as well.

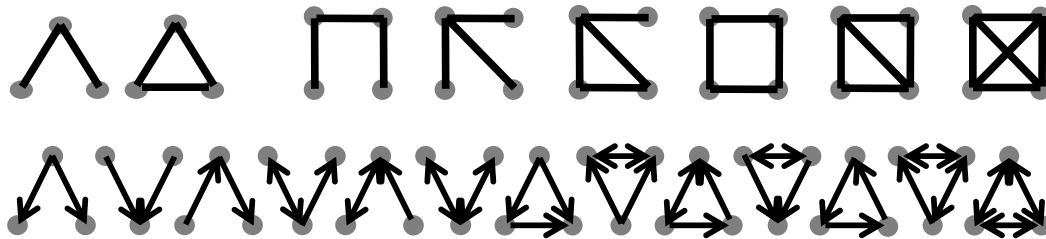


Figure 3.1: Network motifs: (un)directed subgraphs of 3/4 nodes.

MBO is suited for any network in which the nodes have a certain degree of freedom to choose their neighbors. A requirement which is fulfilled in most self-organizing and adaptive networks. We select CAN and Kademia, two different structured P2P systems, as target platforms and aim at more balanced overlay topologies.

Our results indicate that the overlays adapted using MBO indeed show highly improved topological properties. While optimizing Kademia, MBO causes only slight messaging overhead, optimizing CAN does not cause any overhead at all. MBO implies only simple local computations and requires only knowledge about the local neighborhood of the nodes.

Before we show how one can calculate the target motif signatures for our two case studies and thus construct the local decision rules for the MBO module, we give a short background on P2P systems and the challenges they are facing with respect to load balancing.

3.2 Background on Load Balancing

P2P overlays with improper underlying topology suffer severely from inefficient load balancing. This on the other side results in communication delay, poorly utilized network capacities and increased vulnerability to targeted attacks.

Load balancing in P2P systems consists of two challenges: address-space balancing and object-request balancing. The address-space of structured overlays, also known as Distributed Hash

Tables (DHT), is not evenly distributed among peers. Unlike regular hash-functions, a set of peers are responsible for larger parts of the key space. Some peers are responsible for key space areas in the size of $O(\log N)$ [76], where N is the number of peers in the overlay. Still, many approaches disregard those imbalances and assume that the system is static and that the node IDs are uniformly distributed [77] [78] [79]. The severe discrepancy between this assumption and reality explains to a high extent the very often observed difference between the *expected* and the actual performance of real world P2P systems.

On the other side, the object-request balancing refers to the user request popularity of objects. It closely follows the Zipf distribution. Only the most popular objects are less popular than the Zipf distribution predicts [80]. Both address-space distribution and object-request distribution are severely skewed. These effects interfere closely and intensify the unfair load distribution.

MBO does not actively perform object-request balancing. Nevertheless a fair address-space allocation reduces the adverse effect of the skewed object request distribution.

Furthermore, sophisticated load balancing protocols may still be applied on top of MBO. They usually collect and disseminate usage information, rearrange neighborhood relations and sometimes even deploy a structured P2P overlay network on their own [81]. A common approach is to introduce "virtual servers" [78] [82]. That is, overlay maintenance, routing and data storage happens at the virtual server level. A physical node within a P2P network may be responsible for one or several virtual servers [83], which are transparent to the underlying DHT overlay.

The crucial advantage of technological and especially P2P systems is that one can easily deploy protocols changing their topology. The basic principle of our approach is to construct a local optimization strategy for each node in the underlying network based on their local motif signature. That is, each node tries to optimize its surrounding motif content towards an optimal one, which significantly alters the overall network topology and thus the dynamic performance of the system.

In the following Section we show precisely how one can calculate the motif target signatures needed for the local decision rules in MBO for both our case studies.

3.3 Determining Target Motif Signatures

The main idea behind our distributed topology optimization approach is to use pre-calculated target motif signatures. Given a desired topology, the objective is to shift the topology of the underlying network towards the targeted one by shifting the local environment of each node. In this Section we describe how a target motif signature is derived and integrated in a local decision rule. This rule dictates when a node needs to make any changes to its local environment.

First of all, each node needs a language to read its surrounding environment. This is done by counting the number of instances of different k -node motifs the given node is involved in, called the *motif signature* of the node. Second, and more importantly, the node needs a target motif signature towards which to adapt its own motif signature.

Given an initial topology I , the first step is to measure the motif frequency F_I of I . F_I represents the number of all different k -node motifs (where k is usually 3 or 4) found in I . Different motif instances may share nodes or edges, but are considered induced subgraphs.

The second step involves the construction of the target topology T . This step has to be carried out manually once. Then the motif frequency of T is measured and denoted by F_T . F_T is then provided to all nodes as their target motif signature.

To measure the relative change between the initial and the target topology we define the Φ -Score for each motif m as:

$$\Phi(m) := F_T(m) - F_I(m) \quad (3.1)$$

The Φ -Score for each motif m is then normalized as:

$$SP_{\Phi}(m) := \frac{\Phi(m)}{\sqrt{\sum_{i=1}^n \Phi(m_i)^2}} \quad (3.2)$$

The vector SP_{Φ} is called the target significance profile.

Naturally, there are networks where different nodes play different roles within the topology. Nodes with different roles may need to be treated differently. Therefore, we distinguish two classes of target signatures: *common* and *multiple* target signatures. The second class is divided further into two subclasses *deterministic* and *probabilistic*.

In the case of common target signatures, all nodes in the network use the same target signature F_T . Multiple target signatures means that there is not only one, but a set of target signatures available. In the deterministic subclass each node, based only on local knowledge, decides which of those signatures to follow. In the probabilistic subclass each node picks its target motif signature Φ_i from the set of available target signatures with a probability p_{Φ_i} . The total target signature Φ_n is given by:

$$\Phi_n := \frac{1}{n} \sum_{i=1}^n p_{\Phi_i} \Phi_i \quad (3.3)$$

where Φ_i is the i -th of the n available target signatures and p_{Φ_i} the probability of Φ_i .

Throughout this work we use common target signatures to optimize P2P overlays towards load balancing and multiple deterministic signatures to construct resilient live-streaming topologies. In both cases, the target motif signatures just reflect the local content of a network optimal with respect to a desired network property and *suggest* necessary changes to the local environment of the nodes in the actual network.

For our study, as a proof of concept, we optimize two different structured P2P networks, CAN and Kademia, with respect to fair load balancing. In the following we illustrate how to derive target motif signatures for those two cases.

3.3.1 Target Motif Signature: CAN

CAN is a content addressable network where the key space is divided into a multi-dimensional torus Θ and each node is mapped onto some fraction of Θ . When a node wants to join CAN it picks a random point P in Θ and contacts the node ν that currently is responsible for P . The key space of ν is then evenly divided between ν and the new node.

Since this process is probabilistic, the key spaces of the nodes is typically unevenly distributed and the registration and lookup load thus unbalanced [78]. Hence, the desired property for CAN is an evenly distributed key space, leading to a better load balancing. For a CAN network with N nodes and key space with volume V one can write $N = 2^x + r$ where $0 \leq r < 2^x$. The key space is then evenly distributed when $2^x - r$ nodes are responsible for areas of volume $V' = \frac{V}{2^x}$ and $2r$ nodes are responsible for areas with volume equal to $V'' = \frac{V}{2^{(x+1)}}$.

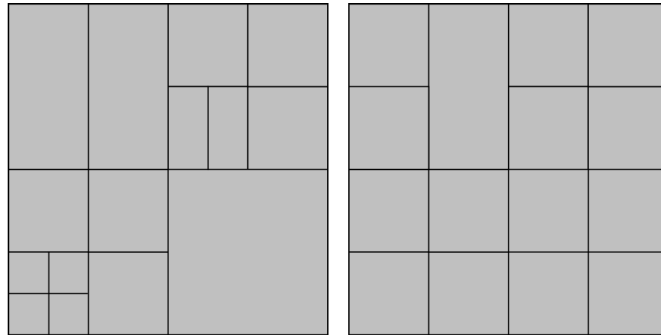


Figure 3.2: A suboptimal and optimal CAN topologies with 15 nodes.

Given the number of nodes N in the network, a topology satisfying the above conditions is very easy to build. Imagine the key space as a plane. Then, starting with the initial plane, one can divide its area into four equal quadrants. Consequently, each quadrant is divided in four equal quadrants and so on until N quadrants emerge. Finally, each network node is assigned a distinct quadrant and nodes which quadrants share a mutual quadrant facet are considered neighbors. Figure 3.2 displays a standard CAN topology on the lefthand side and on the righthand side is a topology constructed as just described.

It is straight forward that the constructed topology is optimal with respect to the above described space distribution conditions. Then, to construct the three-node motif target signature T_{CAN} for CAN one just need to count all instances of the directed three-node motifs in the generated topology. Since in CAN all neighbor connections are bidirectional, there are only two possible directed three-node motifs, see Figure 3.1.



Motif		
$F_I(m)$	0.92463	0.07537
$F_T(m)$	1.0	0.0
$\Phi(m)$	0.07537	-0.07537
$SP_{\Phi}(m)$	0.71	-0.71

Table 3.1: Initial and target motif signatures, Φ -Score and SP_{Φ} for CAN.

Investigating the constructed topology shows that one of the motifs disappears completely (when N is a power of two) and that T is a common target signature. The calculated results are displayed in Table 3.1.

Deriving the common target signature and target significance profile for CAN from Table 3.1 leads to:

$$T_{CAN} := \{m_0 = 1, m_1 = 0\} \quad (3.4)$$

$$SP_{\Phi_{CAN}} := \{m_0 = 0.71, m_1 = -0.71\} \quad (3.5)$$

Now we have T_{CAN} , the target motif signature for CAN, and in the following we show how to derive a target motif signature for our second test case: Kamdelia.

3.3.2 Target Motif Signature: Kademia

Kademlia is a distributed hash table that uses the XOR (exclusive or) metric as a distance measure between nodes. Every node v is a leaf in a virtual tree and has a maximum of k neighbors in every subbranch rooted at the path from v to the root of the tree. The neighbors for every subbranch are organized into buckets and the factor k is called the *bucket size*. When a new node joins the network it contacts its bootstrap node and starts a lookup on its own ID (each node is provided with an unique for the overlay ID). All nodes found by this procedure are added to the corresponding bucket, as long as it contains less than k contacts.

Kademlia creates topologies in which the out-degree of a node is approximately $k \cdot \log(N)$ for all N nodes in the network. The in-degree, however, is unevenly distributed since well connected, older nodes gain enormous in-degrees. The hidden preferential attachment built into the Kademia's join protocol, results from bootstrap nodes transferring neighbor information to new joining nodes. When a given bootstrap node has a node in a tree subbranch from which a new joining node also needs a neighbor, then the bootstrap node recommends that neighbor to the new joining node. In that way, old nodes land in the neighbor list of almost all nodes in the network. As a result, the few old nodes must process a very high portion of the requests traveling through the network. Hence, the hidden preferential attachment produces unnecessary waiting times, denial of service due to overloading and extremely unevenly distributed workflow among the network participants.

The desired property for Kademia is therefore defined as an uniform in-degree distribution. To construct such an optimal topology with N nodes evenly distributed in the key space, all nodes are connected according to the following scheme: a node v randomly picks another node w in a subbranch that it is not currently connected to. If w has no connection into v 's subbranch it allows the connection from v and an edge $v \rightarrow w$ is established. This leads to a topology with bucket size $k = 1$ and outdegree = indegree = $\log(N)$ for all nodes. If a larger bucket size k is required, then the procedure can be repeated k times for each tree subbranch. It is only crucial that nodes do not take recommendations from other nodes, but rather connect to random nodes within that subbranches. Note that due to the XOR metric, it is a priori known which node IDs lay in which tree subbranch and each node can pick k of those IDs uniformly at random.

The results from investigating a standard Kademia topology and an optimal topology of the same size, constructed as described above, are displayed in Table 3.2. The resulting target

Motif													
$F_I(m)$	0,015	0,023	0,025	0,785	0,008	0,007	0,014	0,001	0,000	0,110	0,006	0,003	0,003
$F_T(m)$	0,192	0,419	0,087	0,192	0,003	0,000	0,007	0,007	0,004	0,087	0,000	0,000	0,000
$\Phi(m)$	0,177	0,396	0,062	-0,592	-0,004	-0,007	-0,006	0,007	0,004	-0,023	-0,006	-0,003	-0,003
$SP_\Phi(m)$	0,240	0,537	0,084	-0,803	-0,006	-0,009	-0,009	0,009	0,005	-0,031	-0,008	-0,004	-0,004

Table 3.2: Initial and target motif signatures, Φ -Score and SP_Φ for Kademia.

signature T_{Kad} for Kademia is *common* and contains the 13 directed three-node motifs, see Figure 3.1. From Table 3.2 it is straightforward to derive $T_{Kad} := F_T$.

Now that we have derived the target motif signatures for both our case studies, in the following we show how they are used within the MBO module.

3.4 Motif Based Optimization

In this Section we show how our topology optimization approach can indeed be deployed. As a testbed we again consider the two structured P2P overlays CAN and Kademia, for which target motif signatures have already been derived in the previous Section.

The MBO is a topology control module that uses a target signature as an input. MBO changes the local signature of every node so that the global target signature is approximated. To achieve that goal it plugs into the protocol by intercepting all messages between the original implementation and the network. That is illustrated in Figure 3.3.

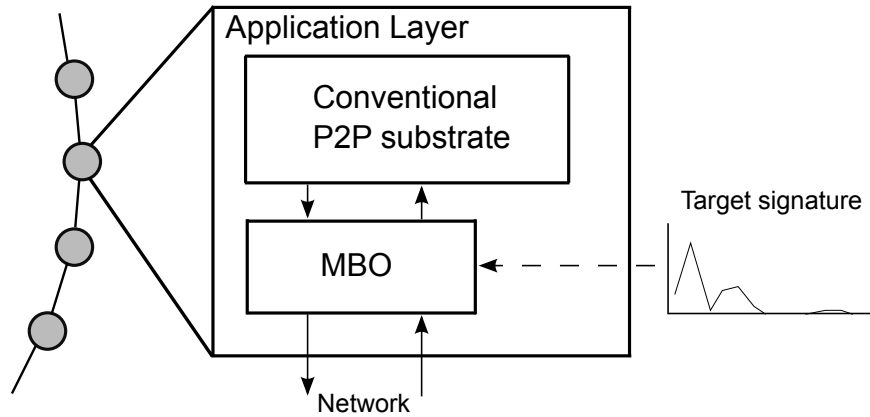


Figure 3.3: System Architecture of MBO

In order to use MBO with k -node motif signatures, the underlying network has to fulfill just two simple prerequisites:

1. Any node in the network must have a certain degree of freedom in the choice of its neighbors without destroying the network functionality.
2. MBO must be able to calculate the $(k-1)$ -hop neighborhood of its associated node.

In the following we show how exactly to apply MBO on our first test case: CAN.

3.4.1 Motif Based Optimization: CAN

To apply MBO the target signature T_{CAN} for CAN has to be computed first (see Section 3.3 for more details). Then, MBO is designed such that an instance of the 3-node clique motif (see Table 3.1) is destroyed whenever a new node joins the network. That de facto leads to a slight change of the join process. When a join message for a new node v is routed from the bootstrap node w towards the point P , the MBO module on every node on the way calculates the local motif signature. When the local signature at a node on the joining path requires optimization, v joins at this exact position. In case that no node on the path to P needs to optimize its neighborhood, v repeats the join process at a new random point. The join process for v is repeated up to r different times. That is, v tries to optimize a local motif signature along r different paths in the network. If all attempts are unsuccessful, v ultimately joins according to the original CAN protocol. In consequence, the undesired 3-node clique motif is subsequently suppressed, which leads to a global approximation of the target signature.

Setting $r = 1$ produces no additional messaging overhead. This not the case for $r \geq 2$. The trade-off between optimality and messaging overhead is investigated in details in Section 3.5. Our results show that one can indeed achieve significant topology optimization with very modest messaging overhead.

Before we proceed with the evaluation of MBO, we first show how it can be applied on our second test case: Kademia.

3.4.2 Motif Based Optimization: Kademia

Recall from Section 3.3 that the target signature of Kademia is slightly more complex than that of CAN and consists of the 13 directed 3-node motifs, see Table 3.2. Therefore, applying MBO on Kademia is a bit more complicated, but in the same time much more generic.

When a node v wants to add another node w to its own routing table, v constructs the 2-hop neighborhood for w and calculates its motif content. If the new edge $v \rightarrow w$ shifts the motif content of w towards the target signature, the edge $v \rightarrow w$ is established. Otherwise, the edge is rejected and v has to search for another partner within the subbranch of w .

This process inevitably produces computational and communication overhead. Still, in the next Section we show that there is a good balance between optimization overhead and topology improvement, leading to much better network performance on the cost of very modest overhead.

Remark: it is straightforward to generalize that approach to any other distributed system beyond the two cases investigated here. At the bottom line, an edge between two nodes is established if none of their local motif contents contradicts that decision.

Remark: note that MBO causes only local computations and decisions. Therefore, the larger is the underlying network, the higher is the benefit of applying MBO. The overhead per node caused by MBO is merely dependent on the network size, making it extremely scalable.

3.5 Evaluation

After we have presented our novel approach MBO, in this section we show that MBO indeed effectively changes the topology towards the desired properties.

For this purpose, MBO is implemented on the PlanetSim framework [84]. Networks of different sizes are generated, where churn is modeled as described in [85]. Lookups obey the Zipf distribution and the networks are kept running for 24h in simulation time. Three topology snapshots per network are taken for analysis: (i) as soon as the simulation leaves the transient phase and the network is stable, (ii) after the warmup phase and (iii) at the end of the 24 hours.

Our results show that MBO achieves significant topology improvement while producing only modest computational and messaging overhead. Naturally, the more precise and aggressive is the optimization strategy, the higher is the overhead produced by MBO. It depends on the exact application scenario to choose the right balance between these two competing aspects.

However, we once again emphasize that the overhead per node induced by MBO is practically **independent** of the network size. All nodes base their decision rules only on simple local computations, which complexity is exclusively affected only by the nodes connectivity. As a result, MBO is **highly scalable** and can directly be applied to networks of millions of nodes, leaving MBO as one of the few choices for topology adaptation in large scale application scenarios.

In the following we present the exact results and management overhead produced by MBO on our two test cases: CAN and Kademia.

3.5.1 CAN

MBO alters the join process in CAN. This may, but not necessarily does increase the messaging overhead during the join process.

Messaging Overhead

In order to measure the messaging overhead induced by MBO, we conducted simulations with different values of the retry parameter r in a network with 2,048 nodes. The results are displayed in Table 3.3.

Network	Average	Minimum	Maximum
Original CAN	13.8	13.6	14.1
CAN (MBO) $r = 1$	9.5	8.9	10.0
CAN (MBO) $r = 2$	13.2	12.8	13.8
CAN (MBO) $r = 3$	17.0	16.1	18.8
CAN (MBO) $r = 4$	20.0	18.7	21.2

Table 3.3: Join Process CAN: Number of messages for CAN with MBO compared to original CAN.

Setting the retry parameter to $r = 1$ actually decreases the cost for joining the network with MBO compared to the original CAN. Indeed, using MBO with $r = 1$ eventually terminates the join process before the final position of the initially planned path is reached. As a result, the

joining path becomes shorter and therefore causes smaller join costs. With $r = 1$ in MBO, the join path is at most as long as the one in the original CAN overlay.

Investigating the results for $r = 2$, one observes two effects on the number of join messages. MBO leads to a possible early termination of join messages, but on the other hand can also cause nodes to join a second time. For $r = 2$ both effects are roughly balanced. In consequence, choosing $r = 2$ produces the same messaging overhead, regardless whether the topologies are optimized using MBO or not. The messaging overhead increases roughly linearly with $r > 2$.

Since choosing $r = 2$ yields high degree of optimization without increasing the cost of the join process, we keep this parameter constant in our further experiments, aimed at estimating the quality of the results produced by MBO.

Key Space Distribution

In the following, we evaluate to what extent MBO improves the scope size distribution within CAN, our optimization goal, which inevitably leads to more fair load distribution.

We compare the original CAN to CAN with MBO. Both overlays are equipped with two CAN-dimensions and the retry parameter is fixed to $r = 2$. With that configuration we generate multiple networks with sizes ranging between 2^8 and 2^{15} nodes.

Since the desired property for CAN is a uniform distribution of the key space, we measure the size distribution of assigned name spaces in the topologies. That is, the distribution of how many nodes are responsible for areas of a given size. Figure 3.4 shows the area sizes relative to the smallest encountered area in the investigated networks with sizes ranging from 256 up to 32,768 nodes.

Independent of the network size, for the original CAN the area sizes differ from 2^7 up to 2^9 while the peak is covered by only about 40% of the nodes in the network. Hence, there are nodes responsible for very small key areas whereas there are nodes responsible for enormous parts of the key space.

On the other hand, the CAN augmented with MBO has a drastically different key space distribution. Independent of the network size, the peak is covered by over 80% of all nodes, while the remaining 20% nodes practically cover areas only within a multiplicative factor of two.

Thus, Figure 3.4 unambiguously shows that applying MBO on CAN produces much more closer to uniform key space distribution than the original CAN overlay and hence assures significantly more fair load balancing among the network participants.

Remark: Our simulation also revealed that churn has no impact on MBO. The damage caused by suddenly departing nodes, is then repaired by new joining nodes. All results presented above remained stable under churn.

In the following Section, we investigate the impact of MBO on our second test case: Kademlia.

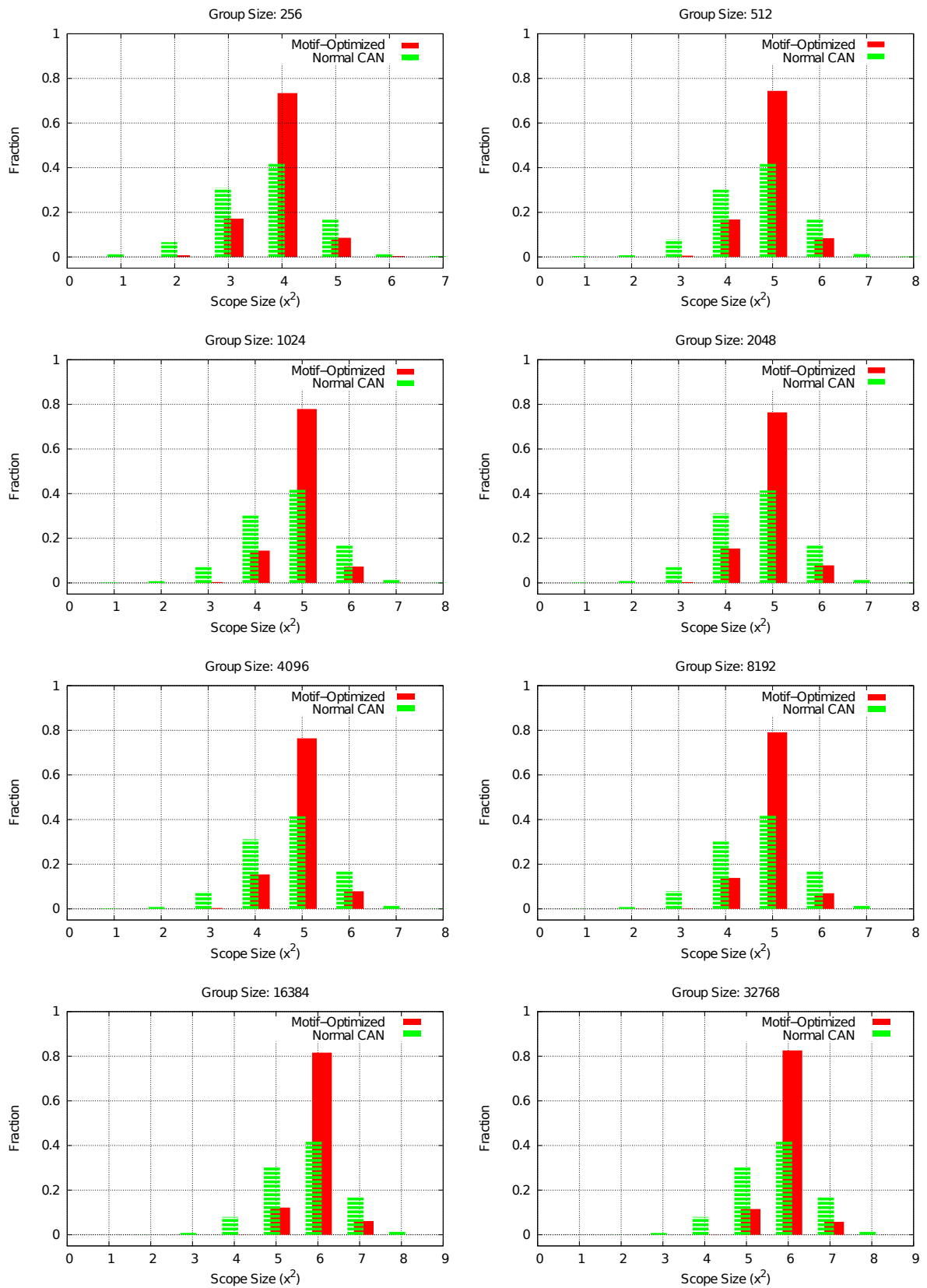


Figure 3.4: Probability of occurrence of scope in CAN with and without MBO.

3.5.2 Kademia

Optimizing Kademia using MBO implies the calculation of a motif signature whenever an edge between two nodes is established. Since MBO needs the 2-hop neighborhood information to calculate the motif content, the neighborhood information is piggy-backed on all messages routed in the network. This information is then used by MBO without the need to explicitly request 2-hop information from the nodes.

Messaging Overhead

To calculate the overhead induced by MBO, all messages within the network are counted. The messages directly related to lookups (request/reply) are counted separately from all other messages. They are denoted as maintenance messages. The results for a network with 2,048 nodes averaged over 100 simulation runs are displayed in Table 3.4.

Network	Maintenance	Lookup	Path Length
Original Kademia	629.6	131.0	6.48
Kademia (MBO)	654.0	128.1	6.85

Table 3.4: Kademia compared to Kademia with MBO with respect to maintenance messages, necessary lookups and average lookup length.

One observes that the maintenance and lookup messages per node remain almost constant. That is also the case for the average path length, measured as the number of lookup messages per lookup.

There is a straightforward explanation for the observed results. The neighborhood information needed for computing the local motif content required by MBO is piggy-backed through the already existing communication flow. I.e. no additional messaging is required for MBO to operate and the few simple local computations per node cause negligible computational overhead.

Remark: although the number of messages transported through the network is the same, their size increases due to the piggy-backing. This causes small additional network load.

Uniform Degree Distribution

We have just shown that MBO produces no additional messaging overhead, but how effective is it within Kademia? The original Kademia creates hub-nodes that are known by almost the entire population, leading to very skewed indegree distribution. Therefore, the goal for MBO is to produce an indegree distribution as close to uniform as possible. Figure 3.5 displays the indegree distribution of both the original Kademia and the one augmented with MBO.

The indegree distribution of the optimized network clearly possesses the desired exponential cut-off, which upper bound is around 75. Hence, in contrast to the original network, there are no nodes with extreme indegrees, neither too small nor too large. About 90% of the nodes

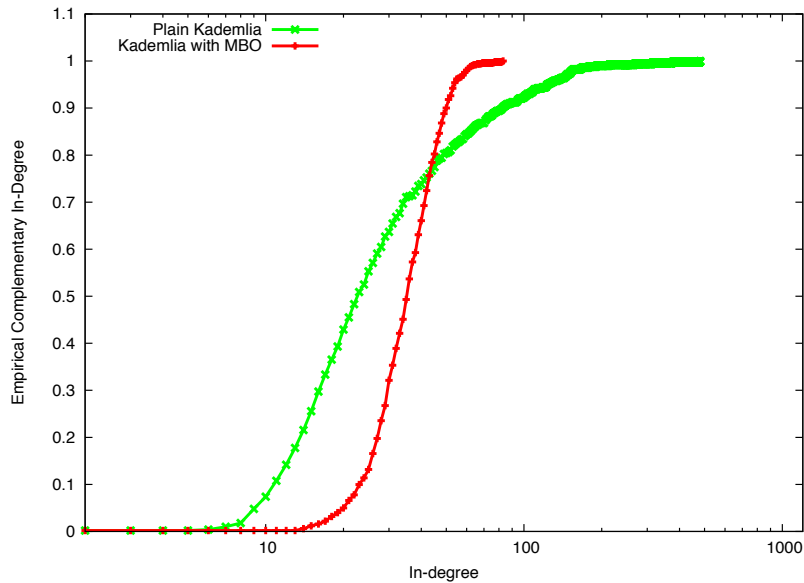


Figure 3.5: Indegree distribution Kademlia (ECD with 500 nodes)

have indegree between 30 and 50, making the network much more balanced. Most importantly, the typical for the original Kademlia hub-nodes, constituting around 10% of the nodes in the network, see Figure 3.5, are practically not present.

That has two effects on the communication flow within the underlying network. First, the communication flow does not go mainly through the few hub-nodes, avoiding node overloading, which can result in long waiting times or even node crashes. Second, the network is much more resilient to attacks by malicious parties aiming at disabling the few *most important* network nodes and hence damaging or even completely destroying the communication flow.

To evaluate the impact of attacks on both, the original Kademlia and Kademlia with MBO, we measure the average path length within the topologies while consequently removing the nodes with highest indegree. The results are shown in Figure 3.6. As one would expect, the optimized topology poses shorter characteristic path length, which is also less susceptible to the attack. This is a direct result of the more balanced topology.

At the bottom line, MBO can be integrated within the existing communication flow of Kademlia, causing no additional messaging overhead. Furthermore, the necessary simple local computations produce negligible computational overhead and basically only the size of the exchange messages is slightly increased.

The benefit of using MBO is however significant. The resulting topology has close to uniform degree distribution, assuring fair load distribution and much higher resilience to attacks by malicious parties.

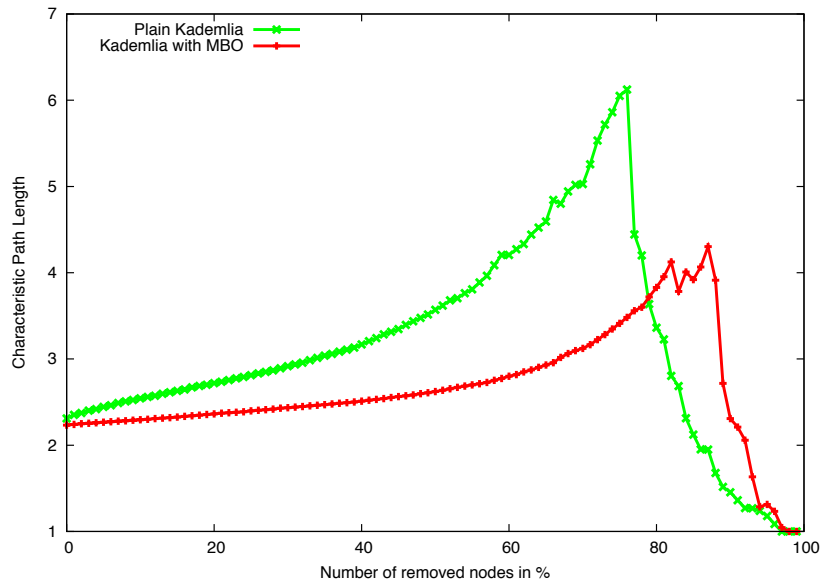


Figure 3.6: Characteristic path length under perfect attack (500 nodes).

3.6 Summary and Outlook

In this Chapter we have explored a completely different perspective on network motifs and local structures. We were motivated by the findings presented in Chapter 2, which clearly show close interplay between the local structures of complex, self-organizing networks and the dynamics taking place on top of these networks.

Instead of using network motifs as a purely statistical tool to analyze already emerged static networks, we engaged them in online local decision rules within distributed human-made systems. A prominent example of such systems are P2P, for which we have presented a novel motif based topology control approach.

3.6.1 Summary

In this Chapter we have introduced MBO, motif based optimization, a general approach for distributed topology optimization. MBO utilizes network motifs in order to optimize arbitrary, adaptable topologies towards any desired and well defined structural property. For the purpose of optimizing networks using MBO a topology, optimal to the desired objective, has to be synthesized first. Taking this network as a reference, a target motif signature, which represent the distribution of motifs in the network, is derived and subsequently used to parametrize the local optimization module at each node.

For our analysis we have selected two structured P2P overlays with inherently different protocols and characteristics: CAN and Kademlia. The uneven partition of the name space of CAN and the uneven in-degree distribution due to preferential attachment of Kademlia have been identified as the main adverse properties that lead to unfair load distribution in these systems.

We consequently have synthesized optimal sample topologies for both cases and derived the target motif signatures as parameters for the MBO module.

Subsequently, simulation models of MBO both for CAN and Kademlia have been implemented to examine the performance of our approach. The evaluation results show that optimizing topologies locally by engaging motif signatures significantly improves the topology balance in both investigated systems, expressed through even key space distribution for CAN and uniform indegree distribution for Kademlia.

MBO induces negligible messaging overhead in both cases. Even more importantly, MBO is extremely scalable as no information on the network state has to be exchanged at any time. It is based entirely on local information and computations. Therefore, with respect to complexity, MBO outperforms any current methods for distributed topology control.

3.6.2 Outlook

Optimizing topologies based on motifs so far has only been tested for common target motif signatures. However, topologies that are optimal with respect to other objectives may be characterized by a set of significance profiles which deviate for different nodes. Especially in networks where the nodes are not all *peers*, but rather play different roles within the underlying topology. One such example are optimally DoS-resistant overlay streaming topologies [86].

It is the objective of the next Chapter to explore whether network motifs still can successfully be engaged in local decision rules, despite the diversity of the system participants.



4 Resilient Peer-to-Peer Live-Streaming Using Motifs

In the following Chapter we show how one can engage network motifs not just as an online topology control mechanism, but also to provide substantial resilience to attacks and high level of privacy to the involved participants in P2P based live-streaming networks.

High robustness against churn and resilience towards adverse behavior are the key requirements for reliable P2P live-streaming systems. Their highly inter-dependent nature, based on the cooperative service delivery between all peers, necessitates a systematic and structural resilience. This is very challenging to assure because of the self-organization and decentralized control of such systems. Reliable services, and hence a structural resilience, still are a vital prerequisite for any commercial deployment of P2P live-streaming systems or IPTV infrastructures.

We propose an entirely distributed motif-based topology optimization to this end. Concise comparisons show that it creates resilient topologies comparable to state of the art, yet causing significantly less, almost negligible overhead with respect to both computation and messaging.

Our new approach also has another property of immense importance: it neither gathers nor forwards any information about the underlying network. Thus, it renders actions by malicious parties, aimed at disturbing the network operation, almost impossible as no peer can determine her/his position nor the position of any other peer in the network.

Such privacy is not guaranteed by any other state of the art method for P2P live-streaming networks, which makes our method outstanding in its area.

4.1 Introduction

Creating resilient topologies, and thus achieving high robustness of P2P streaming systems, is one of the main challenges when designing a new approach for this comparably new class of content distribution schemes.

P2P streaming promises to greatly relieve server load and aid in supplying large audiences with broadband multimedia content [87].

Two different services, video on demand and live streaming, create two fundamentally different problems in this field. While content can be distributed to some or all of the audience in advance in the first case [88, 89], it is delivered directly upon creation in the latter, making it much more difficult to guarantee satisfactory provision without deficits.

Introducing the cooperative delivery of P2P systems, in which the streaming packets are relayed between the nodes, further complicates the task. Forwarding peers are not only less performant than dedicated servers, but they additionally exhibit a much less reliable characteristic with high churn of frequently joining/leaving or even failing parties. Still, each node relies on the correct service of all preceding nodes on the packet path from the source, since failures and delays are propagated from peer to peer.

Moreover, using the system to deliver controversial content, or simply considering any commercial deployment, makes the system a worthwhile target for parties with malicious intent. However, service degradation or even disruption, be they caused by churn, failure, or DoS attack is unacceptable. Especially when the service is targeted at deployment in commercial scenarios.

Achieving robustness and resilience with P2P streaming systems hence is a challenging task. They represent interdependent large complex networks, composed of highly heterogeneous nodes with extremely dynamic behavior. Resilience in these systems consequently has to be achieved by decreasing the interdependence between the nodes while exploiting their resources to increase scalability. For reasons of complicated attacks by malicious parties, this task must be achieved with minimum disclosure of the system state.

In general, topology adaptation can either be implemented in a central oracle or by means of cooperative, distributed optimizations. Central instances represent single points of failure and potential bottlenecks, which makes them unfavorable when implementing large scale distributed systems. Distributed adaptation, on the other side, is traditionally based on broad information on the overall state of the system, which is costly to gather and may be misused by malicious parties.

An alternative approach for cooperatively adapting topologies is to use metrics based on local information only and by optimizing the local state, indirectly to approximate the overall desirable network characteristics.

In summary, we see the need to provide a scalable topology adaptation for live P2P streaming systems that successfully creates topologies of very low interdependency while considering only a bare minimum of local information.

The best distributed method (with respect to resilience) is currently a cost-based method. It creates highly resilient topologies by considering aggregated information on succeeding nodes [86]. However, the engaged cost metrics are computationally demanding and knowledge about parts of the topologies has to be gathered. That knowledge may potentially be exploited by malicious parties.

Network motifs on the other hand represent a metric much simpler to calculate. They also have the ability to reveal the complex interplay between topology and dynamics, see Chapter 2, and are a very promising application in topology adaptation, see Chapter 3.

In this Chapter, we propose to harness their ability to reproduce complex characteristics of networks for the purpose of adapting highly robust and resilient live streaming topologies.

We make the following contributions: we (i) present an innovative approach using network motifs to build local decision rules for the self-optimization of the network; (ii) improve the resilience of the topology to be close to the resilience of optimal topologies; (iii) evaluating our design in simulations, illustrate the benefits of our approach compared to the state of the art.

In order to better understand the challenges upon live-streaming applications, we first give a short background.

4.2 Background

It is important to understand the basic principles and the major problems upon live-streaming systems. Within such systems there is one source which provides the original streaming signal. All other participating peers are subscribing to that signal. The main goal is to build a topology providing each peer with the signal while being resilient to failures and misbehaving participants. To decrease the impact of failures as few dependencies among peers as possible should exist. This has the consequence, that the failure of a single or a subset of peers has the least possible impact on the service of the remaining peers.

Brinkmeier et al. [86] have defined a class of optimal topologies with proven resilience to DoS attacks. They also proposed a fully distributed approach for constructing nearly optimal streaming topologies, which is currently the most effective method in the field.

We use their approach as a reference throughout this Chapter and hence adopt their mathematical definition of the problem: Consider s being the source, or the originator of the streaming content, the system can be modeled as an undirected graph $G = (V, E)$ with finite set of N vertices $V = \{v_1, \dots, v_N\}$, the data source s and a set of edges $E = \{(u, v) : u, v \in V\}$ along which the content is forwarded.

The multimedia content can be modeled as a packet stream $S = \{p_1, \dots, p_p\}$ of p packets. All p packets can be replicated at each vertex and originate at the data source s . The bitrate of the stream is denoted as R_0 . To decrease the dependency between nodes, and hence vulnerability to attacks, the packet stream can alternatively be split into partial streams, with l sequences of k stripes: $S = \{\{p_{1_1}, \dots, p_{1_k}\}, \dots, \{p_{l_1}, \dots, p_{l_k}\}\}$. Each stripe in consequence has an average bitrate of $\frac{R_0}{k}$. The source s has a bandwidth capacity of C times the bitrate R_0 . That is, s can deliver the whole bitrate R_0 of the stream simultaneously C times, i.e. it can directly serve $C \cdot k$ stripes. We assume that all participating peers share at least $R_0 + c$ bandwidth in order to participate in the live streaming, and that in consequence they are able to receive the complete stream once and forward $c \cdot k$ stripes. In accordance with the state of the art, we assume c to be 2 and hence that each node is able to receive the stream once and to forward its bitrate at least twice. All joining nodes locate nodes already part of the topology, select them as initial parents and request the stream from them.

The benefit of the system is the number of stripes that are successfully and timely received at all nodes. An *attacker* is considered to aim at causing the highest possible damage to the system with the least necessary resources. The damage is measured as the fraction of stripes that are not successfully delivered in the system after the failure or removal of an arbitrary set of nodes. For this purpose the attacker is assumed to exploit knowledge used in the protocol to identify valuable targets: nodes on which depend a large set of other nodes. The attacker additionally is assumed to be able to either relocate itself to an identified position and subsequently stop the service, or to be able to launch attacks on identified nodes, thus disconnecting them from the system completely. Since disabling the source will result in immediate loss of the whole signal, the source is assumed to be hidden and cannot be attacked.

We have just reviewed the basics of live-streaming systems. We next shortly survey already existing approaches, in order to better place this work in the existing body of research.

4.3 Related Work

P2P streaming systems, commonly also referred to as overlay or application layer multicast (ALM) [90], are usually divided into two groups of *pull*- and *push*-based systems [91, 92].

In pull-based ALM the video is split into chunks and each peer requests all chunks using some P2P system (like different distributed hash tables, or more frequently BitTorrent-like approaches [93, 94]). The chunks then are delivered to all peers along the reverse path of their issued requests, which may vary quite notably depending on the P2P method used. Although pull-based approaches are naturally more resilient to churn and attacks, they induce significant delays to the content distribution. Thus, they are not viable for live-streaming scenarios [95].

Push-based approaches follow the strategy of creating longer lived distribution topologies over the participating nodes, along which all streaming packets are delivered. They are characterized by significantly lower delay penalties for two main reasons: (i) received packets are forwarded to succeeding nodes in the topology directly on reception; and (ii) since no per-packet management messages are needed, the stream can be split into smaller chunks, which then can be delivered in parallel. Since these approaches rely on the created topology, they generally are less resilient to churn and attacks. Node departures as well as node failures have to be detected first and then encountered by repairing and reconstructing the distribution topology.

The main strategy to increase the resilience of push-based ALM, besides using different encoding or other forward error correction mechanisms [96, 97, 98], is to decrease the dependency on potentially failing, preceding nodes in the topology [99]. This is achieved by two generally different strategies: feedback based predecessor selection or pro-active topology optimization.

The feedback based predecessor selection leverages on different types of reputation mechanisms in order to increase the dependency on highly reliable and available nodes. Respectively, to decrease the dependency on unreliable, weak-performant, or potentially failing nodes [100, 101, 102, 103]. Although these approaches lead to quite good results, they cause significant communication and computational overhead. Hence, they are of very limited practical use, especially in scenarios with large number of participating peers.

A slightly orthogonal concept is to split the stream into different partial streams (commonly named *stripes*). Then, to deliver these stripes along multiple paths, which are at best node and link disjoint. Split-Stream [104], DagStream [105] and BCBS [106] all follow this methodology. However, they all lead to reasonably high topologies with long paths of inter-dependencies among the nodes. Furthermore, they in practice fail to establish node disjoint paths [86].

Another way of decreasing the interdependency between nodes is to create topologies of very short paths, thus generating potentially very fat trees[107].

All so far described approaches are prone to massive damages in case of attacks: with knowledge on the protocol and through accumulating information on the topology, attackers are able to easily identify highly relevant nodes with large sets of successors. Consequently, malicious parties are given the opportunity to heavily disrupt the system by targeting such nodes.

Brinkmeier et al. [86] have successfully defined a class of balanced topologies having short, node disjoint paths and hence are optimally resilient to attacks. They also proposed an approach

that creates topologies resembling this class, locally optimizing each node's neighborhood, based only on local knowledge. That approach currently provides the highest resilience to attacks and failures in P2P live-streaming systems. However, it still gathers information about some properties of the succeeding subtrees and causes significant computational overhead.

After we have discussed the challenges upon P2P live-streaming systems and already existing methods in that field, we proceed with our new approach.

4.4 System Design

In this Section we describe the basic outline of our novel distributed approach and how it engages network motifs in local decision rules for constructing robust streaming topologies.

4.4.1 Network Motifs

In streaming networks, due to security implications, it is important that the peers know as little about the network topology as possible. Network motifs provide exactly the required non-intrusive way of looking at the surrounding environment of a given peer. Therefore, we engage them in a simple decision rule. When the participating peers obey that rule they improve their local environment. More importantly, those multiple local changes lead to significant improvement of the global properties of the underlying overlay network.

From the problem description, see Section 4.2, it can be derived that the best case, thus the theoretically optimal solution, is to create an optimally balanced spanning tree of minimum height for each of the k stripes. An example of such a topology is displayed in Figure 4.1, it consists of $N = 37$ nodes and maximum number of neighbors (outgoing and ingoing) $n_{max} = 4$.

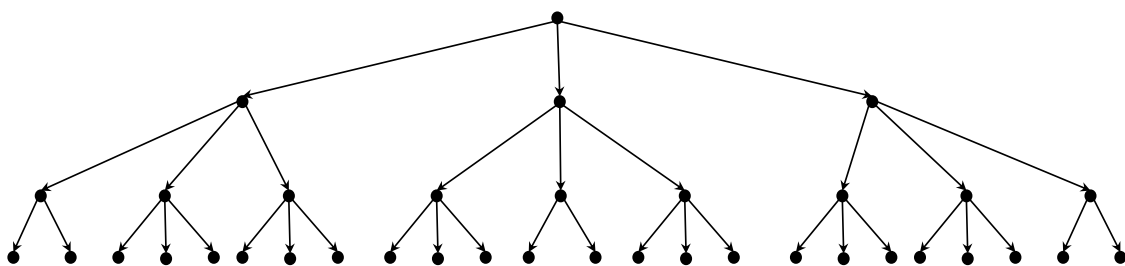


Figure 4.1: Example of an optimal topology, $N = 37$, $n_{max} = 4$ (cmp. [86]).

Our key observation is that the nodes in an optimal topology can be divided in three groups. In the first group are the *internal* nodes. They have the full number of direct successors: $d = n_{max} - 1 = c \cdot k$ and each of the successors has d direct successors on their turn. The group of internal nodes includes the root s . The second group of nodes are the *intermediate* nodes. Those are the nodes which all have d direct successors, but some/all of those successors are leaves.

The intermediate nodes are the nodes on the last but one layer in the topology. The last group of nodes are the *leaves* of the topology, which have no successors.

One observes that in the optimal topology only the first and third directed three-node motifs are present, see Figures 3.1 and 4.1. Throughout this Chapter we refer to these two motifs as motif 1 and motif 3 respectively.

Furthermore, the three groups of nodes have very distinct local motif signatures, i.e. the type and number of motifs a node is involved in. If one computes the local motif signature of each node in the optimal topology and sort them in ascending order with respect to the ratio of motif 1 to motif 3, one gets a clear three phase transition of local motif signatures, see Figure 4.2.

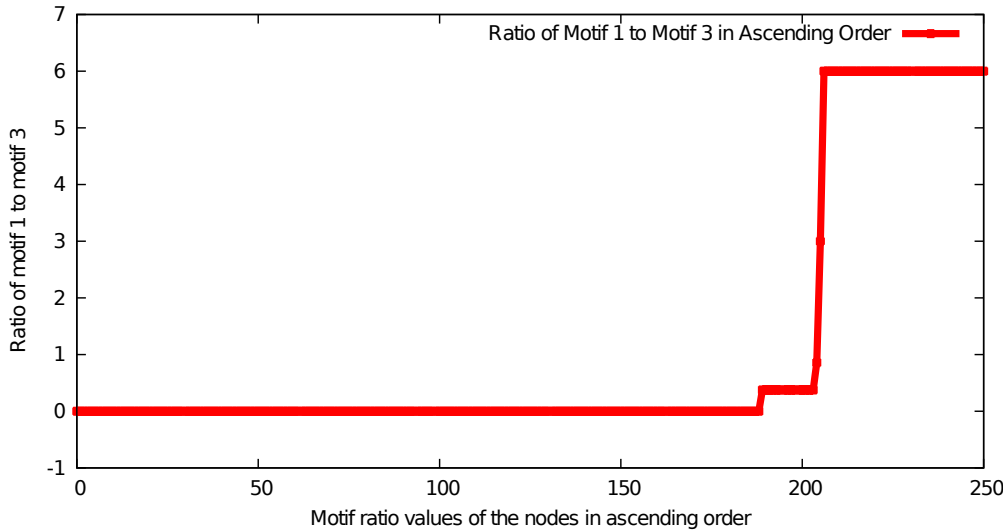


Figure 4.2: Motif ratio phase transition of an optimal topology.

Note that when a node is not involved in any motif 3 instances, then its ration is set to zero. When it is not involved in any motif 1 instances, its ratio is set to $d!$. In this way the motif ratio is always well defined.

From the three phase transition in Figure 4.2 one can easily distinguish the three classes of nodes. All leaves have ratio value of six, the intermediate nodes ratio value of zero and the internal nodes have ratios between these two values.

In the next Section we show how one can benefit from the above observed phase transition.

4.4.2 Engaging Network Motifs in Topology Optimization

To construct local decision rules for optimizing live-streaming topologies, we consider the phase transition shown in Figure 4.2. One observes that the few internal nodes have a specific motif ratio. Its value can be computed from the maximum number of successors d , representing the bandwidth capacities of the nodes. The number of motif 1 instances an internal node is involved in is equal to all pair combinations of its successors without repetition and is given by $\sum_{i=1}^{d-1} i$. The number of motif 3 instances is given by $\sum_{i=1}^d d_i$ because each successor has d successors on its turn and this results in d instances of motif 3 per successor. Thus, the ratio θ of motif 1 and motif 3 for each internal node, which we call the *threshold*, is then given by:

$$\theta = \frac{\sum_{i=1}^{d-1} i}{\sum_{i=1}^d d} = \frac{\frac{d(d-1)}{2}}{d^2} = \frac{d-1}{2d} = \frac{1}{2} - \frac{1}{2d} \quad (4.1)$$

This threshold θ is then used in the local decision rules of our approach.

4.4.3 Implementation

In this Section we present the workflow of our new motif-based approach.

The source node s is the initial bootstrap node. It estimates the minimum bandwidth d the participating peers should provide depending on their expected configuration. Then, it divides the original signal in k stripes and waits for the rest of the participants to join the network. The value of θ is derived from the estimation of d and provided to the joining peers.

Each node is provided with a *Node Manager* (NM). When a node joins the overlay, it randomly joins at all k stripes as a leaf. The NM monitors that the overall used bandwidth over all k stripes is less than the available bandwidth of its node.

The outline of the algorithm is as follows: for each stripe tree the NM computes the motif ratio of its node within that tree. When it is not equal to the pre-given threshold θ , the NM makes changes to the local environment of its node in that tree so that its motif ratio *improves*. For that purpose the NM make exchange operations with the node's predecessor/successors.

The following is a pseudocode outline of the balance operation carried out by the NM of each peer p . The balance operation is repeated for each stripe tree T_i in some predefined time step.

```

Input:  $p, T_i, \theta, \text{Childs}_{T_i}(p) \leftarrow \{w \text{ child of } p \text{ in } T_i\}$ 
1  $\text{Pred}(p) \leftarrow \{w \text{ predecessor of } p \text{ in } T_i\};$ 
2  $M_1 = \frac{|\text{Childs}_{T_i}(p)| + |\text{Childs}_{T_i}(p)+1|}{2};$ 
3  $M_2 \leftarrow 0;$ 
4  $R \leftarrow 1;$ 
5 foreach  $w \in \text{Childs}_{T_i}(p)$  do
6    $\text{NumChilds}_{T_i}(w) \leftarrow \{\text{number of childs of } w \text{ in } T_i\};$ 
7    $M_2 \leftarrow M_2 + \text{NumChilds}_{T_i}(w);$ 
8 end
9 if  $M_2 \neq 0$  then  $R \leftarrow \frac{M_1}{M_2};$ 
10 if  $R > \theta$  then
11    $\text{requestSuccessors}(\text{pickOne}(\text{Childs}_{T_i}(p)));$ 
12 else if  $R < \theta$  then
13    $\text{giveUpSuccessors}(\text{Pred}(p));$ 
14 end

```

Algorithm 1: Topology Control

All nodes only make requests (procedures *requestSucc* and *giveUpSucc*) to their predecessors/successors and all transfers are made from successors to predecessors. This constraint guarantees that the resulting structures are indeed spanning trees as they are invariant to the transfer operations. The nodes have the right to reject requested transfers. For example when their bandwidth does not permit serving another successor, or the requested exchange operation is unfavorable to their own motif content. Figure 4.3 illustrates the two transfer operations.

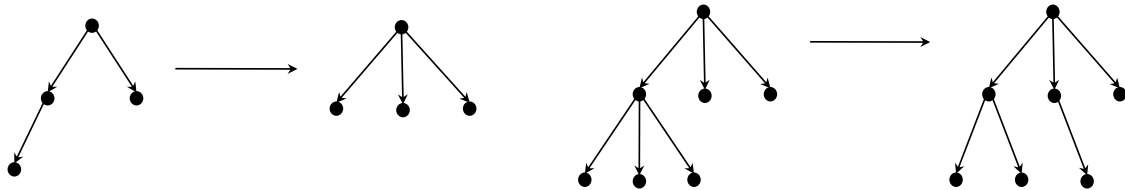


Figure 4.3: Successor exchange operations.

The only exception to the local decision rule is the root, which in general always denies giving up successors and pushes its bandwidth utilization to the maximum. It requests as many successors as bandwidth is available, and allocates it uniformly over all k stripes.

The algorithm has converged when each of the stripe trees has converged. The algorithm has converged in a stripe tree when all nodes in that tree either have motif ratios equal to the threshold or no more improvements are possible due to local deadlocks.

Our approach improves the topology in each step and does not need to converge explicitly. The trees are functional at any time, it is just their resilience that increases over time.

4.5 Evaluation

In this Section we present the results from extensive evaluations of our novel approach. They include the management overhead, the structural properties of the produced topologies and most importantly their resilience to attacks.

In all of our experiments we use the following scenario: There is one root node providing the system with the original streaming signal. It decides in how many stripes the signal should be divided and estimates d , the number of maximum successors per node. To enable comparison to our reference system we keep $c = 2$ fixed and divide the source signal into $k = 10$ stripes. Networks with 300 to 3,000 nodes are simulated for all experiments. All results are averaged over 100 repetitions of each experiment and all results are indicated with their standard deviation.

In our resilience simulations we evaluate worst case attacks, causing maximum damage to the topology. Since churn is a subset of that attack and no time measurements are taken, we refrain from packet level simulations and use a turn based simulator to generate the topologies.

We investigate our new approach from three different perspectives: management overhead, topological properties and resilience to attacks.

4.5.1 Management Overhead

Recall from Section 4.4.3 that each node is provided with a Node Manager optimizing its motif content within each stripe. Therefore, the overhead produced by our approach is given by the number of exchange operations per node and per stripe caused by the Node Managers. That is, how many exchanges each node has to perform on average for the approach to converge. The results are displayed in Figure 4.4, the error bars are smaller than the data points.

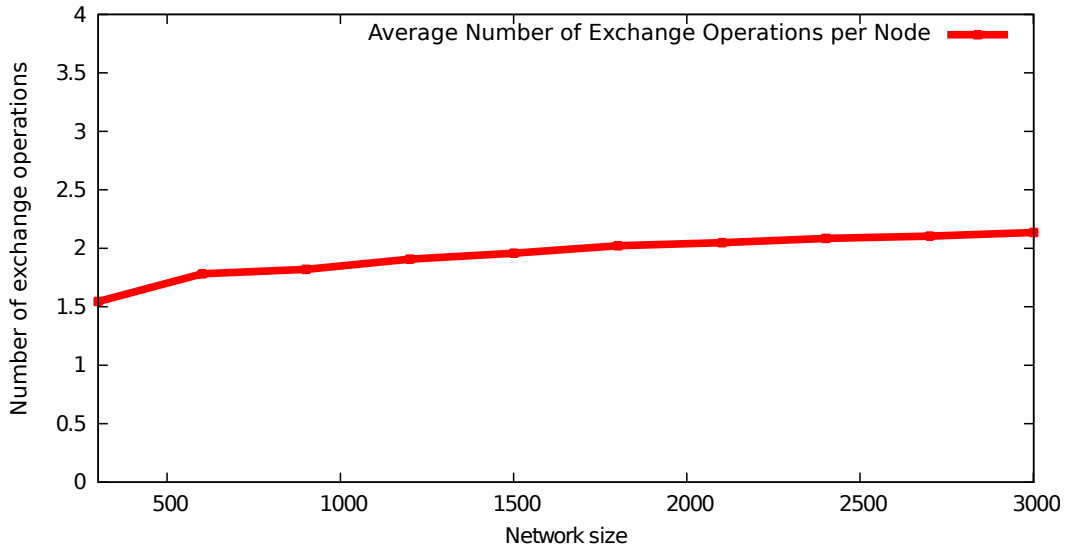


Figure 4.4: Number of exchange operations per node per stripe.

One observes that the average number of exchange operations per node per stripe depends only marginally on the network size and in total it grows sublinearly with the network size. It increases from 1.5 to just a bit over 2 while the network size grows with one order of magnitude. Even more important is the small number, just around two, of required exchange operations per node which represents a negligible overhead per node.

4.5.2 Topological Properties

Now we know that the overhead per node produced by our approach is very small and more importantly independent of the network size.

Our next step is to investigate the quality of the produced topologies. For this purpose we use a set of four topological metrics: *ToPo metric*, *tree height*, *Balance metric* and *vertex connectivity*. Additionally, we measure their resilience towards perfect attacks.

The first measure we apply to the generated topologies is a simple topological measure, which we call the *ToPo* metric. Given a tree, it reflects the balance in height of the subtree starting at each node. For each node the *ToPo* metric is defined as the difference between the longest and shortest branches of the succeeding subtree, starting at that particular node in the overall tree. Figure 4.5 shows a sample tree with the *ToPo* metric values of the nodes within the tree.

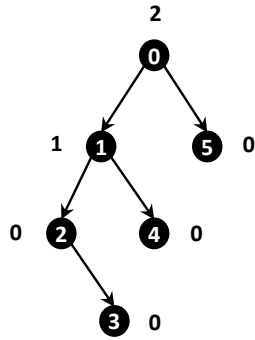


Figure 4.5: Sample tree with corresponding ToPo metric values.

The ToPo metric value of a tree is defined as the sum of the ToPo metric values of its nodes. In our example $0 + 0 + 0 + 0 + 1 + 2 = 3$. It follows, that in a tree with N nodes, which with respect to its height is perfectly balanced, the ToPo metric is zero. In the worst case where the whole tree is just a list, the ToPo metric value is equal to $\frac{N(N+1)}{2}$.

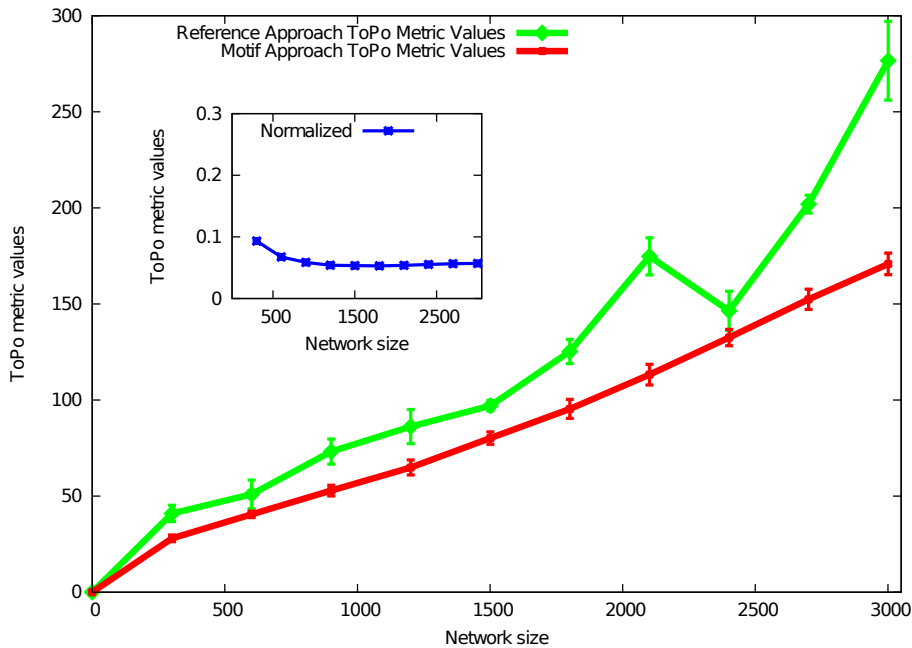


Figure 4.6: ToPo metric values of streaming topologies (with std. deviation). Inset: the ToPo metric results of the motif approach normalized with respect to the network size.

The ToPo metric is measured for each stripe tree and the value for the whole topology is the average over all its stripes. The results are displayed in Figure 4.6.

One observes that the normalized ToPo metric values of the generated topologies are independent of the network size. Furthermore, the motif based approach produces better balanced

trees than the reference approach. Recall from Section 4.2 that we use the most effective existing method of Brinkmeier et al. [86] as a comparison benchmark.

The smaller is the ToPo metric value of a given tree, the better are all nodes distributed among the tree branches (under the constraint of maximum number of successors per node). This on the other hand means shorter paths among the nodes. In the context of P2P live streaming networks this means shorter delivery paths and hence smaller delays as the signal travels through the network.

Recall from Section 4.2 that streaming topologies should have minimum predecessor/successor dependencies. In that way fewer peers are effected in case of a failure or overloading of a peer. Thus, the streaming topology should be as flat as possible. Therefore, we also investigate the height of the generated topologies. It is defined as the average height of all stripe trees within the topology. The results are displayed in Figure 4.7.

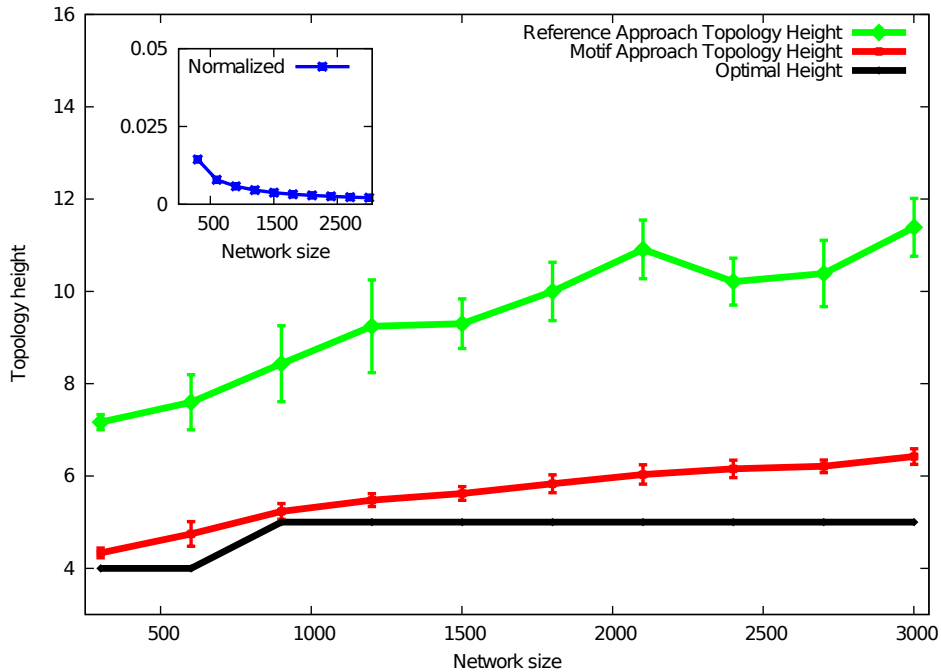


Figure 4.7: Height of streaming topologies (with standard deviation). Inset: topology height of the motif approach normalized with respect to the network size.

Once again our results are independent of the network size, confirmed by the inset in Figure 4.7. They are also clearly flatter than those of the reference approach and very close to optimal. For a perfectly balanced tree with N nodes and maximum allowed successors per node d , the height of the tree is given by $\lceil \log_d N \rceil$.

Note that the cost functions used in the reference approach is targeted at the higher levels of the topology. Therefore, it achieves almost uniform distribution of the nodes among the direct successors of the root and becomes suboptimal near the leaves of the topology.

On the other hand, the motif-based approach treats all nodes equivalent. That not only leads to better performance with respect to global topological properties, but also has further impor-

tant advantages as shown in Section 4.6. The drawback of the motif-based approach is that exhibits partially skewed distribution of the nodes among the direct neighbors of the root.

To investigate the difference between the two approaches with respect to the distribution of the nodes among the direct neighbors of the root, we measure their *Balance metric*. For a tree with N nodes and $|N_{root}|$ direct neighbors of the root, the optimal distribution is $\frac{N-1}{|N_{root}|}$ nodes per subtree starting at each direct neighbor of the root. The balance metric for a given tree is then defined as:

$$B(T) := \sum_{i \in N_{root}} \left| suc(i) - \frac{N-1}{|N_{root}|} \right| \quad (4.2)$$

where T is the given tree, N_{root} the set of direct neighbors of the root, $|N_{root}|$ the size of N_{root} , $suc(i)$ the number of nodes in the subtree starting at node i , including i , and N the number of all nodes in T . The balance metric values of both approaches are displayed in Figure 4.8.

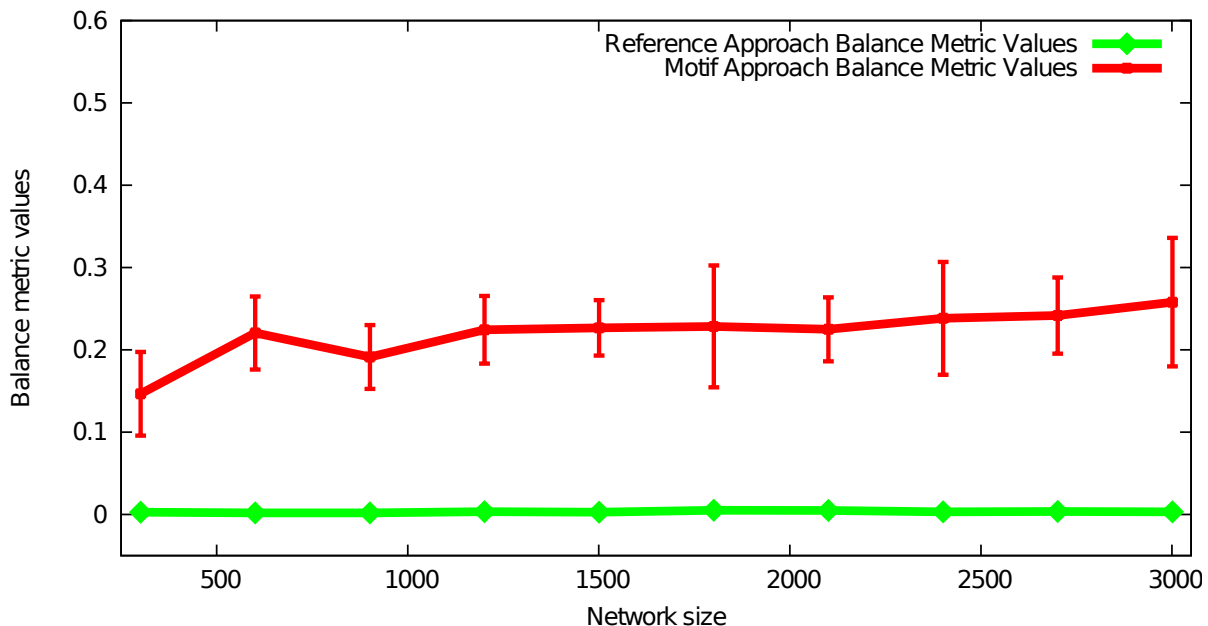


Figure 4.8: Balance metric values of streaming topologies normalized with respect to the network size (with standard deviation).

One observes that the Balance metric values of our approach are independent of the network size. As it was to be expected, they are not as stable and close to optimal as those of the reference approach. This is because the cost functions in the reference approach punish more severely imbalances closer to the root. The motif approach treats all nodes equally and therefore suffers from imbalances all over the topology and not only at the leaves. In fact our new approach does not rely on any other knowledge on the underlying topologies but the direct neighborhood of each node. That dramatically increases the privacy of the participating peers as we discuss in detail in Section 4.6.

The next topological measure we apply to our topologies is the *vertex connectivity*. It counts the number of nodes that have to be removed from a given graph, such that it disintegrates into two disjoint parts. In the context of P2P live streaming, the vertex connectivity reflects the minimum number of malfunctioning peers such that a subset of the remaining peers is cut off from the content stream.

Note that due to the division in 10 stripe trees, the theoretically optimal node connectivity lies at exactly 10 nodes. Figure 4.9 shows the evaluation results of both approaches.

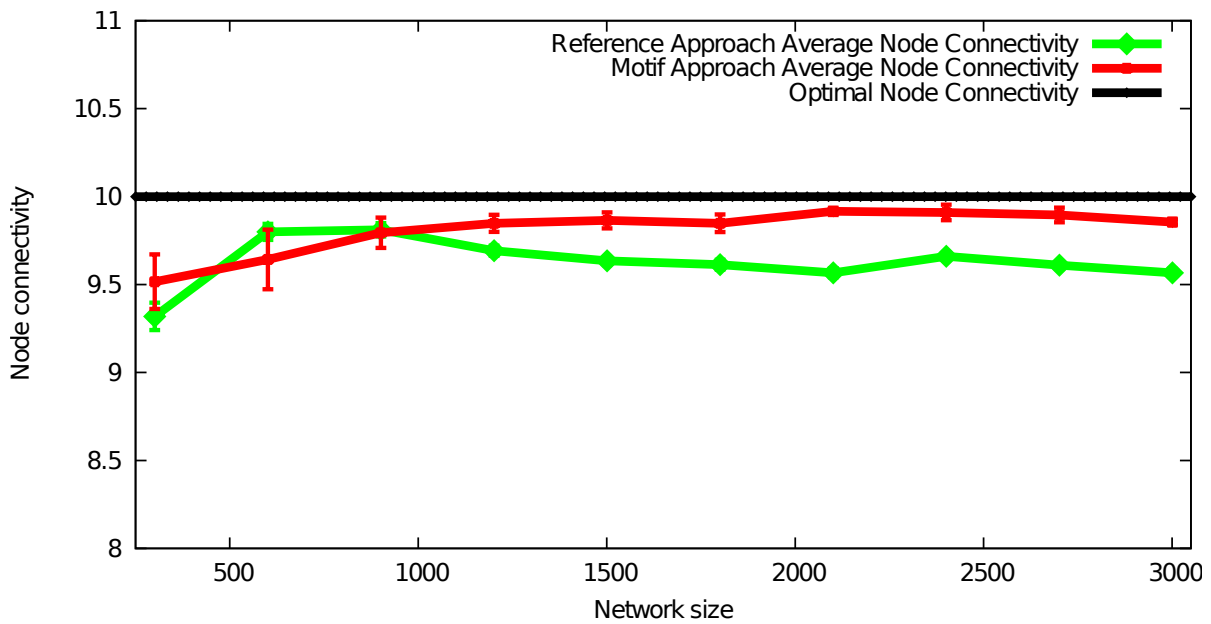


Figure 4.9: Average node connectivity of streaming topologies (with std. deviation).

One observes that the node connectivity even increases with the network size. For networks with more than 1000 nodes it is independent of the network size around 9.7. Our approach once again outperforms the reference one.

In summary, we have observed that the our motif based approach outperforms the reference approach on a set of crucial topological measures. The new method generates topologies that are flatter, more balanced and stronger connected. For P2P live-streaming topologies that means shorter delivery paths, i.e. smaller signal delays, and less node interdependencies, i.e. higher robustness to failures. The bottom line is: the new method promises better performance under normal circumstances.

For a commercial deployment of a P2P live-streaming systems a good resilience to attacks is indispensable as intervention by malicious parties is very likely, see Section 4.2. Therefore, in the next Section we investigate the resilience of our approach to perfect attacks.

4.5.3 Resilience to Attacks

We have so far investigated the topological measures that reflect the dependencies among the participating peers. Finally, we evaluate the resilience of our topologies toward perfect attacks with a metric called *stability* [86].

The stability of a topology is given by the number of still received stripes after removing the set of the *most important* nodes in the topology. The most important nodes are those nodes whose absence leads to the highest damage to the service of the overall system. To determine the set of the most important nodes for increasing set-sizes, we use exhaustive enumeration and branch and bound methods. That is a highly unrealistic scenario for streaming networks, but represents the worst case attack and thus an upper bound for possible damages through correlated failures, churn or any conceivable attack.

The BCBS [106] method has been shown to provide an upper bound with respect to resilience in the field of P2P live-streaming networks. Consequently, in addition to the reference approach, we also take BCBS as a comparison benchmark for the resilience to attacks of the existing methods in this field.

We divide the source signal into 10 stripes and allocate a source bandwidth of $C = 2$. It follows that even in the perfect case, where each direct successor of the root in one of the stripe trees is a leaf in all other trees, it is the 20 nodes directly connected to the source that need to be removed to completely disrupt the service. The evaluation results are shown in Figure 4.10.

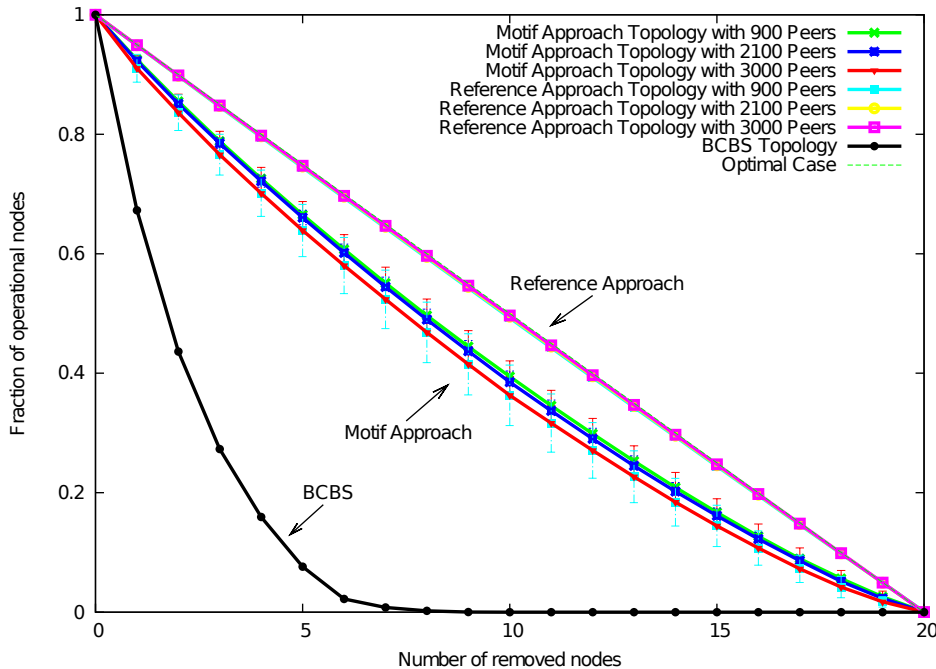


Figure 4.10: Stability of streaming topologies (with standard deviation).

One observes that our approach is practically independent of the system size. It produces similar results for network sizes varying by one order of magnitude. One also observes that

there is a clear gap between the new approach and the reference [86], which achieves almost perfect results. That is no surprise considering the differences with respect to the balance metric, see Figure 4.8. Still, our results are significantly better than any other current streaming topologies [86]. This is confirmed by the significantly worse performance of the BCBS method.

It is clear that our approach is not as optimal as the reference with respect to stability, but it still produces excellent results. More importantly, it has some other crucial advantages, which are discussed in detail in the next Section.

The bottom line is: We have tested the streaming topologies generated by our new approach for their topological properties as well as their resilience to attacks. In all of the test cases the achieved results are very close to optimal and independent of the network size.

Consequently, our new method assures outstanding performance under normal circumstances and excellent resilience to attacks, both on the cost of modest overhead, and thus clearly outperforming the state of the art methods.

4.6 Method Comparison

In this section we directly compare the motif based approach to the reference method [86]. They both share some similar features, but each one of them has its advantages and drawbacks. To point out the most important of them we compare the methods on three different criteria: optimality of the constructed topologies; convergence and complexity; and their resilience and vulnerability to attacks by malicious parties.

4.6.1 Topological Properties

We have shown in Section 4.5 that with respect to three different topological metrics (the ToPo metric, average three height and vertex connectivity, see Figures 4.6, 4.9 and 4.7) the motif approach produces results close to optimal and clearly outperforms the reference method. The motif based approach merely leaves space for further improvements and guarantees better performance under normal circumstances.

The complex cost functions used in the reference approach are targeted at the direct neighbors of the source. Therefore, with respect to the Balance metric it outperforms the new approach, see Figure 4.8, and has better prerequisites with respect to resilience under perfect attacks, which we discuss extensively in Section 4.6.3.

4.6.2 Convergence and Complexity

Both approaches provide streaming topologies functional throughout the whole network operation, i.e. they do not have to converge for the live streaming to take place. Still, the longer the algorithms run, the better the topologies become. Therefore, it is crucial to compare their convergence time as it is well known that P2P networks have a constantly changing nature.

Our new approach requires only a few exchange operations per node, see Figure 4.4. The local decision rule is based only on a few simple computations, in $O(1)$ computational complexity, and hence requires negligible effort from the participating peers. The messaging overhead over all stripes per peer is in $O(kd)$, since it requires one value from each neighbor in each stripe.

On the other side, the reference approach causes a computational complexity in $O(d^2)$ and produces the same messaging overhead $O(kd)$.

Therefore, the new approach is clearly better suited for devices with little or constrained resources, e.g. most mobile devices, or for application scenarios with highly fluctuating underlying network topologies.

4.6.3 Network Resilience

The resilience against attacks and failures is the outstanding feature of the two approaches we discuss and therefore our main point of comparison.

We first tested the node connectivity of the generated topologies, see Figure 4.9. That is, the minimal number of nodes that have to be removed, such that the underlying topology breaks down into two separate fragments. In this test the new approach outperformed the reference approach. However, they both produce close to optimal results, leaving almost no space for further improvements.

Subsequently, we have performed perfect attacks on the topologies generated by the two approaches. Both attacks rely on complete network knowledge. The possession of such knowledge is highly unrealistic for distributed systems, but is indeed the ultimate challenge for both approaches. This attack tests the stability of the generated topologies, see Figure 4.10). It measures the number of residually received stripes after a perfect attack on the network has been performed. The perfect attack considers global knowledge and is carried out through exhaustive enumeration and branch and bound methods. Being unrealistic, it definitely represents an upper bound of possible damage through correlated failure or attacks.

In this test the reference method outperformed the new approach, managing almost perfect, linear decay of the operational nodes. Nevertheless, the stability of the topologies generated by the motif approach is significantly higher than those of topologies generated by other ALM approaches.

The bottom line is that the reference approach provides higher resilience to attacks. Our new approach, on the other hand, performs very well too, while causing significantly less computational and messaging overhead.

A clear advantage of the new approach is the fact that no knowledge about the topologies is gathered nor forwarded. In contrast, the reference approach aggregates and forwards information about the underlying topologies. Furthermore, in the new approach the behavior of the root node with respect to its successors is no different than any other node. Thus, even when a malicious node is as high as being a direct successor of the root, there is no way for it to determine that.

Any miscreant capable of gathering knowledge about the topologies represents a threat. It might be allowed to infer facts on its current location in the topology and consequently estimate at least partly the current state of the system .

Consequently, our new approach provides much higher privacy to the participating peers, rendering attacks by malicious parties almost impossible. This is an advantage over the reference approach of immense importance for any commercial deployment.

4.7 Summary and Outlook

This Chapter studies the resilience of P2P live-streaming topologies. The considered scenario consists of a source peer, which provides the original signal, and a set of peers that are interested in the stream and provide parts of their available bandwidth to cooperatively distribute it among each other. Hence, these systems harness the resources at the end-hosts and thus greatly decrease the server load and aid scalability to large audiences.

With each peer relying on the correct service of all preceding peers on the packet path from the source, these systems are prone to experience service disruption due to delays or departing peers. Especially considering a commercial deployment, service degradation or disruption are unacceptable. Taking into account the existence of malicious parties complicates the problem even further due to the cooperative and open nature of these systems.

4.7.1 Summary

To this end, we propose a novel approach for constructing streaming topologies that are resilient to node failures and DoS attacks. It employs network motifs in online topology control and not just as a statistical measure, for what they have been exclusively used so far. Hence, our new approach relies on decision rules based on local knowledge of the nodes, only.

Our approach consequently does not provide the participating parties with any knowledge on their position within the network nor its overall state. Hence, attackers have no means of inferring the position of other nodes nor their importance and are therefore unable to identify valuable targets for attacks.

In extensive evaluation we compare our new approach to the currently most effective approach in this field. It has been shown to produce streaming topologies that are almost optimally resilient towards DoS attacks [86]. The comparison includes series of topological measures as well as their response to perfect attacks.

The evaluation results indicate that both approaches achieve comparable performance. Both approaches create topologies significantly more resilient to DoS attacks than other methods from related work.

The reference approach achieves a slightly better resilience to attacks. Yet it relies on gathering some knowledge on the succeeding topologies of each node and is characterized by much higher computational and messaging complexity.

The motif based approach on the other hand achieves better topological properties and is independent of any information on the topology other than the direct neighborhood of each node. Therefore, the new approach crucially guarantees better performance under normal circumstances and provides higher privacy to the participating peers, rendering intrusions by malicious parties almost impossible. Both these properties are of immense importance to enable possible commercial deployment.

4.7.2 Outlook

Our new approach currently only aims at creating resilient live-streaming topologies. However, common objectives in this scenario are to decrease the end-to-end delay and to achieve location awareness to efficiently use the infrastructure of the underlying network. We are currently in the process of extending our approach to incorporate location information in order to provide network efficient streaming topologies. We are additionally adapting protocol and local decision rules to decrease overlay path lengths and the observed delays. On a broader view we are pursuing the question, whether such simple local decision rules can be applied in other decentralized settings, such as routing and topology adaptation in wireless sensor networks.

5 Finding Communication Bottlenecks in Distributed Environments

Throughout the previous Chapters we have shown that by looking at the local structures of complex networks one can understand the dynamic processes taking place on those networks. In Chapters 3 and 4 we went one step further and showed that one can even control the dynamic performance of networks by intentionally steering their local structures in live time.

In this Chapter we investigate the reverse perspective: Is it possible to deliberately use dynamic processes to reveal the topological structure of distributed complex networks? Our answer to that question is yes. As we show shortly, one can indeed engage gossiping (a typical dynamic process in social networks) to find communications bottlenecks in a prominent subclass of distributed complex networks: multihop wireless networks.

The results of our extensive evaluation show that our novel approach is indeed very effective in detecting the few crucial for the network operation nodes and that it clearly outperforms existing state of the art methods.

Nodes in mobile networks are usually unevenly distributed over space. Several dense clusters of nodes are interconnected by a few nodes in sparsely occupied areas. Removing vital nodes along such *bridges* would partition the network and severely reduce the overall connectivity. Consequently, detecting and protecting those few vital nodes in live time is crucial for keeping the network operational.

In order to achieve this task, we present our novel approach: BridgeFinder. It is based on an extended gossiping protocol and is significantly faster and more precise than any existing mechanisms. On the one side, BridgeFinder allows us to calculate good estimates for global graph measures while operating as a fully distributed algorithm and causing only modest messaging overhead. On the other side, in contrast to conventional gossiping algorithms, it has an efficient guarding mechanism against malicious nodes trying to screw the protocol operation.

5.1 Introduction

Our main target environments are distributed wireless environments, such as wireless multihop networks. They impose severe challenges upon any distributed application. Established connections or even nodes may suddenly disappear due to many unpredictable factors. Because communication flow depends on the network connectivity, it is crucial to identify and protect the few critical peers within the network. Those are the articulation points which failure leads to malfunction or complete breakdown of the network, as they constitute the few *bridges* keeping the network connected.

Standard approaches for identifying such vital links require global network knowledge, which is unavailable in mobile multihop networks. Even if it were, one would require $O(N^3)$ running time, where N is the number of nodes within the network (by using Floyd-Warshall algorithm for example). Obviously, such approaches are of extremely limited practical use.

In order to overcome these challenges, we developed BridgeFinder. It is compliant with the Push-Sum gossiping algorithm [108]. BridgeFinder identifies critical paths between densely connected clusters based only on local knowledge. The basic idea is to let a floating value *diffuse* through the network and then detect the communications bottlenecks by examining the diffusion speed of the single nodes.

Our extensive evaluation shows that BridgeFinder very efficiently identifies the critical nodes. Furthermore, it can be integrated in the regular maintenance and application traffic and therefore produces almost no additional messaging overhead.

Last but not the least, BridgeFinder is augmented with efficient guarding mechanism, providing it with solid resilience against malicious nodes trying to screw the protocol operation. That makes our approach outstanding in the field of gossiping based methods.

5.1.1 Network Prerequisites

There are two requirements the underlying network has to fulfill in order for BridgeFinder to function. First, a node must be able to communicate with other nodes in the network, at least with its direct neighbors. Second, the links in the network must be undirected, i.e. communications must be bi-directional.

In general, BridgeFinder operates on any network that meets these two criteria. Both criteria are fulfilled by mobile multihop networks, at least to the extent required by BridgeFinder.

5.1.2 Application Domains

BridgeFinder detects critical peers, crucial for communication within the network. Therefore, the fewer the nodes on which communication depends, the higher the benefit of using our approach. Compared to algorithms based on global knowledge in static networks, BridgeFinder is less accurate than standard approaches for detecting critical nodes. *However*, its most significant advantage is that it requires no global knowledge and can operate on continuously changing underlying networks. That makes it very attractive for distributed environments, like wireless ad-hoc networks, P2P overlay networks and wireless sensor networks.

Once the critical peers are known, partitioning can be avoided by establishing additional links among susceptible clusters connected by those critical peers. In any complex network, maintaining information flow is equal to the critical nodes remaining operational. The higher the importance of a node, the higher is the impact of losing it and reversely the higher is the benefit of detecting and protecting that node a priori.

5.2 Properties of Critical Peers

There are two categories of critical peers: global and local [109]. If a global critical peer fails, the network disassembles into two or more components. If a locally critical peer fails, this peer along with its neighbors are isolated from the network. The rest of the network is operational.

There is a set of specialized algorithms for detecting the second type of peers [109, 110, 111]. They produce different computational overhead, depending on the accuracy of their results. Still, all of them require only the neighbor list of each peer and no further network knowledge.

Global critical peers are much more important. Their failure affects the function of the whole network. There are **no local** and hence **efficient** algorithms for detecting such peers.

The crucial observation on which our approach relies is that critical peers play a significant role in distributing information within the underlying network. Our results show that with very high probability a critical peer either lies on many communication paths among other nodes or has very short communication paths to almost all other nodes, and often even both. Detecting peers with such properties is equivalent to detecting global critical peers.

A straightforward question arises: How do we describe and detect peers playing such an important role in supporting communication flow? The intuitive answer is to use centrality measures from graph theory. Note that these measures require global network knowledge and are very expensive to compute. Overcoming these two issues in a distributed manner is the main contribution of BridgeFinder.

5.2.1 Centrality Measures: Betweenness Centrality

One of the important topological measures from graph theory for describing nodes in complex networks is *betweenness centrality* or simply *betweenness* [22, 23].

The betweenness of a node is proportional to the number of shortest paths going through this node. Nodes that occur on many shortest paths within the network have higher betweenness. Consider a graph $G = (V, E)$, where V is the set of nodes and E the set of edges connecting them. The betweenness $C_B(v)$ of a node $v \in V$ is given by:

$$C_B(v) := \sum_{s \neq v \neq t, s \neq t} \frac{\sigma_{st}(v)}{\sigma_{st}} \quad (5.1)$$

where $s, t \in V$ and $\sigma_{st}(v)$ is the number of shortest paths from s to t going through v and σ_{st} the number of all shortest paths between s and t . The higher the betweenness of a given node, the more important is this node for communication within the network.

Note that the betweenness of a node v could be zero if it is even one single edge away from a node with high betweenness. That is because no shortest path goes through v as this automatically increases the path length by one, see Figure 5.1. Still, that node plays an important role for the communication flow. It can be used as a backup in case that the direct connection between the two high betweenness nodes vanishes.

To overcome that problem we introduce the following measure:

Definition 1 *The average betweenness of a node v is equal to the average betweenness of its own and the betweenness of its neighbors:*

$$C_{AB}(v) := \frac{C_B(v) + \sum_{w \in N} C_B(w)}{|N| + 1} \quad (5.2)$$

where N is the set of neighbors of v and $|N|$ its size.

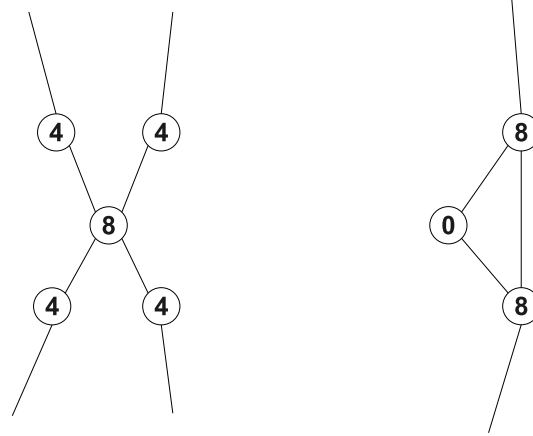


Figure 5.1: Betweenness does not always reflect the central role of a node. The displayed values represent the betweenness coefficients of the corresponding nodes.

In other words, not only nodes with high betweenness but also their neighbors have high average betweenness. This more accurately reflects their central position in the network for two reasons. First, these are the nodes which must overtake the information flow if the leader node fails. Second, they can serve as a backup for the leader node before it fails by interconnecting among each other and thus reducing its load. That also significantly reduces the impact of suddenly losing the leader node due to some unexpected reasons.

5.2.2 Centrality Measures: Closeness Centrality

Another important measure is *closeness centrality* [112, 113], here referred to just as closeness. Using closeness we can identify those nodes within a network which are responsible for fast communication flow. The closeness of a node v is defined as the mean geodesic distance (i.e. the shortest path) between the node v and all other nodes reachable from it:

$$C_c(v) := \frac{1}{n-1} \sum_{t \in M \setminus v} d_G(v, t) \quad (5.3)$$

where $d_G(v, t)$ is the geodesic distance between v and t in G and n is the size of the connected component M reachable from v . Closeness is a measure of how long it will take for a particular information to spread from a given node to all other reachable nodes.

Remark: when G is not connected, d_G becomes infinity. In that cases equation (5.3) is no more well defined. In such cases one usually takes the reciprocal distance measure $\frac{1}{d_G}$. Consequently a node has short communication paths to other nodes when it has high closeness coefficient. However, in our analysis we consider only connected networks, see Section 5.4, and therefore we stick to the definition of closeness as in equation (5.3).

Nodes with small closeness are fast in distributing information through the network. Still, closeness is a linear measure. In large networks there are whole regions of nodes with very similar closeness values. That makes the standard closeness measure in some cases imprecise in highlighting the nodes with significant closeness coefficients.

In order to sharpen the precision of the standard closeness measure, we introduce *square closeness*. It also computes all shortest distances among all nodes in the network, but squares them before the average is computed:

Definition 2 *The square closeness of a node v is the sum of the square distances from v to all other reachable nodes:*

$$C_{sqc}(v) := \frac{1}{n-1} \sum_{t \in M \setminus v} d_G(v, t)^2 \quad (5.4)$$

where d_G , M and n are defined as in equation (5.3).

Nodes with best closeness have significantly higher squared closeness than the rest of the nodes. That makes it easier to detect outstanding nodes within clusters of nodes with relatively similar closeness coefficients.

Although all these metrics are useful in identifying the global critical nodes, they are: (i) very expensive to compute and (ii) are not applicable without global network knowledge, for example in distributed environments.

To overcome that problem is the main contribution of BridgeFinder. Our results show that the nodes BridgeFinder identifies agree to large extent with the nodes one identifies by applying the analytical measures discussed above.

5.3 The BridgeFinder Algorithm

Gossiping works in the following manner: at an arbitrary position in the network a node starts a *gossip*. It sends the information of interest to all its neighbors. Then each node on its turn shares the information it has just received with its neighbors and so on. Eventually, all nodes become aware of the new *gossip*.

To detect global critical peers, BridgeFinder does extended gossiping. Instead just to spread a piece of information among the nodes, it rather lets a given value *diffuse* through the network. Some node A picks a random floating number d and divides it uniformly among its neighbors and itself. If A has N neighbors, then after the first iteration, all of them as well as A have value of $\frac{d}{N+1}$. In the next iteration, each node that just has received a value (including A) distributes its value among its neighbors as well and so on. Each node also keeps track of the number of iterations it has exchanged values. In Section 5.5 we show in detail how to avoid multiple instances of BridgeFinder running simultaneously.

To estimate when d has successfully diffused through the network, we introduce a tolerance parameter ϵ . Overall in our simulations we set $\epsilon = 0.02$. When the current exchanged value of a node v does not differ from its previous value with more than 2%, it sets its status to converged. The whole network has converged, when all of its nodes have converged.

The following is a pseudocode illustration of the exchange operation for a node v which just has received a value d_{in} from some of its neighbors:

```

Input:  $v, d_{in}, \text{Neighbors}(v) \leftarrow \{w : (v, w) \in G\}$ 
1 if  $\text{hasConverged}(v)$  is false then
2    $M \leftarrow |\text{Neighbors}(v)|;$ 
3    $d_{old} \leftarrow \text{getValue}(v);$ 
4    $d_{new} \leftarrow \frac{d_{old} + d_{in}}{M+1};$ 
5   if  $|d_{old} - d_{new}| > \epsilon$  then
6      $\text{setValue}(v, d_{new});$ 
7      $\text{iterationCount}(v) \leftarrow \text{iterationCount}(v) + 1;$ 
8     foreach  $w \in \text{Neighbors}(v)$  do
9        $\text{sendValue}(w, d_{new});$ 
10    end
11  else
12     $\text{markConverged}(v);$ 
13  end
14 end

```

Algorithm 2: BridgeFinder Exchange Operation

The workflow of the the central value exchange operation can be described as follows: If the node v still has not converged, it checks if its old value d_{old} is more than an ϵ away from its new value d_{new} . In the negative case, v marks itself as converged. Otherwise, it posts its new value to all its neighbors. The node v repeats exchanging values with its neighbors until it has converged. The algorithm stops when all nodes have converged.

Each node also keeps a list of the k fastest converging nodes, in our simulations $k = 10$, and the number of exchange steps they required to converge. When a node v has converged, it checks if it has not been faster than some of the nodes in its top list. If this is the case, v sends to its neighbors the information that it has converged and that it qualifies for the top k nodes list. The neighbors of v then respectively adapt their lists of top converging nodes, on their turn propagate the change to their neighbors and so on.

Assume that a node w receives a message that v has qualified for the top k list. Then, if not already done so, w adapts its top list (note that w can receive the update message along different paths). Next, w on its turn builds the update request in its next exchange messages to its neighbors and so on. When BridgeFinder is started for the first time, every node has only one entry in its list: the node itself. Distributing an update message to every node in the network is integrated in the exchange messages and in worst case needs as many exchange operation as the network diameter (a relatively small number for many communication and social networks).

After each run of the BridgeFinder algorithm, every node knows the fastest converging nodes. The results of the previous run are distributed in parallel to the value exchange operation of the next run and so on. Figure 5.2 illustrates the situation. As we show in Section 5.4, the fastest converging nodes are also the critical nodes. Once they have been identified, measures can be taken to protect them.

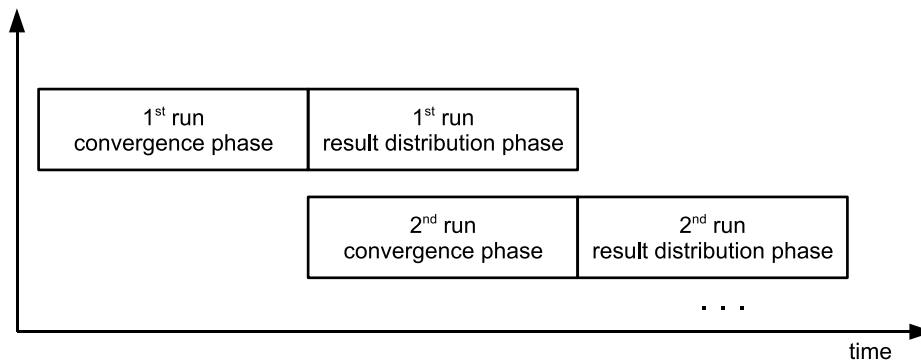


Figure 5.2: Phases of the BridgeFinder algorithm and interleaving among multiple runs.

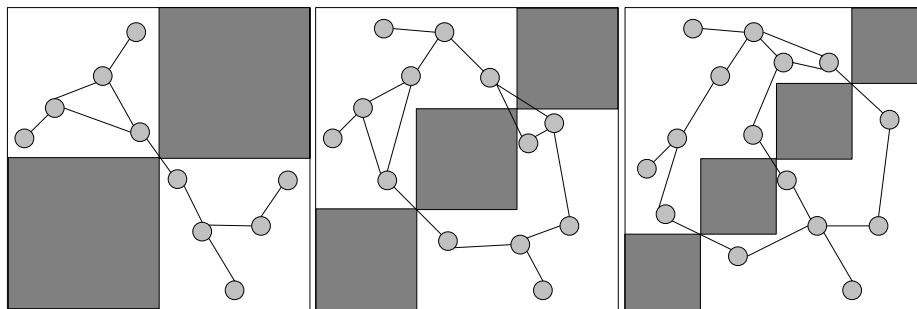


Figure 5.3: Three different obstacle scenarios. Examples show 1, 2, and 3 bridges respectively.

5.4 Evaluation

Wireless multihop networks are usually modeled with unit disk graphs (UDG) [114]. The construction principle is as follows: given N nodes and a distance threshold ϑ , distribute all N nodes uniformly at random within an area of a fixed size. Then, all nodes among which the geometric distance is less than ϑ are connected by edges. In other words, all nodes which lay close enough to each other are considered neighbors in the resulting wireless ad-hoc network. This model is valid for wireless networks within a perfect surrounding environment. In practice the entities usually move in groups, have favorite spots and there are areas that they avoid.

We extend the UDG model by inserting obstacles of different sizes within the area where we place the nodes. These obstacles represent unpopular or inaccessible areas. The generated graphs represent real world ad-hoc wireless networks more precisely. The approach for more realistic network models is based on the work of Jardosh [115]. We call the network model ODG (obstacle disk graph), see Figure 5.3 for an illustration.

We evaluate BridgeFinder using the ODG model for four different network types with none (equivalent to the standard UDG model), one, two and three *bridges*.

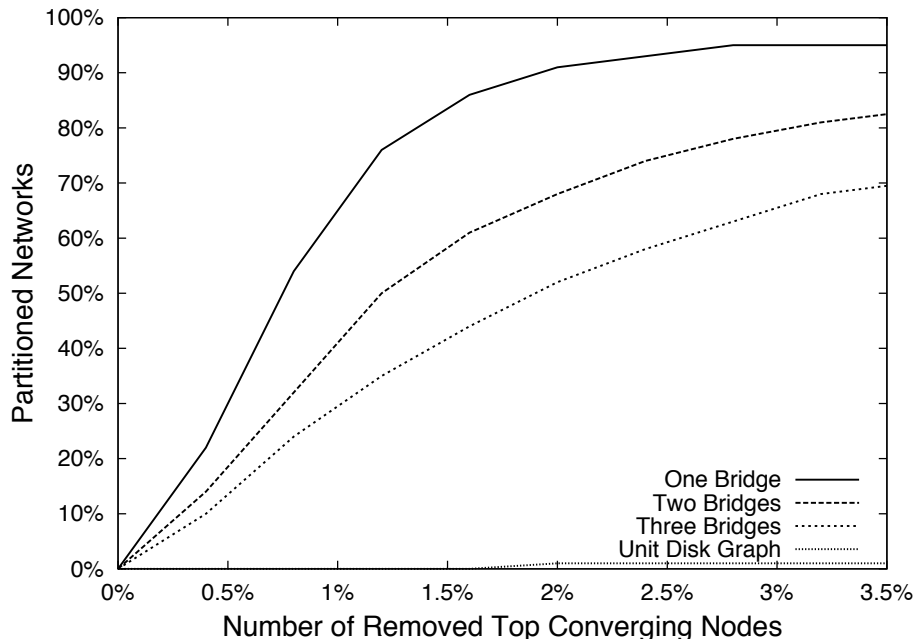


Figure 5.4: Destroying networks by removing the fastest converging nodes.

Note that nodes are distributed uniformly at random within the areas around the obstacles. Therefore, there is no guarantee how many of the existing bridges are actually used to connect the different clusters, nor how many different paths run through each bridge. However, networks generated in this manner are very vulnerable. The few nodes lying on the paths among clusters are exactly the few nodes keeping the network together.

There are two important questions that need to be addressed. What topological properties do these few nodes have? More importantly, how effective is indeed BridgeFinder in detecting those very nodes?

We evaluate our approach on 500 networks of each type. Each network consists of 250 nodes placed in a physical area of 100x100 units. The maximum edge length is set to 12 units, i.e., we place edges among all nodes within distance of 12 units from each other. We create multiple obstacles with dimensions of 25x25, 30x30 and 50x50 units (see Figure 5.3 for an example setup). Because of the obstacles and the random placement of nodes, the resulting networks are not always connected. Therefore, during the generating process we discard non-connected networks and generate new ones until 500 connected instances of each type were acquired.

Now that we have diverse test scenarios, we finally can address the central question of this Chapter: what role do the nodes identified by BridgeFinder play for keeping the network connected. For this purpose, one by one we remove the fastest 3.5% converging nodes. The results are displayed in Figure 5.4. The x-axis shows the percentage of removed nodes. The y-axis shows the fraction of partitioned networks from the 500 instances of each type. We consider a network partitioned if more than 25% of the remaining nodes are not able to communicate with the rest of the network.

Figure 5.4 shows averaged results for all four different network types. Removing only 2% (equivalent to just 5) of the fastest converging nodes breaks 90%, 70% and 50% of the one, two and three bridge networks respectively. BridgeFinder finds with very high probability exactly the few nodes that keep the networks together.

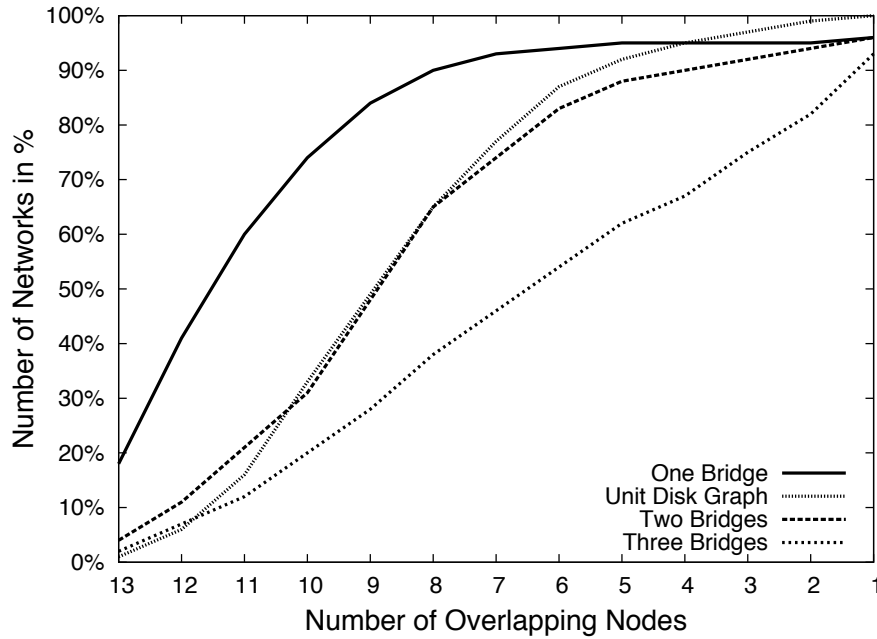


Figure 5.5: Intersecting best centrality measure nodes with 5% of the fastest converging nodes.

Note that the standard UDG graph does not partition. This is not surprising. In such graphs, there are no critical nodes at all. Depending on the density of the nodes, there are two possibilities. Either each node is very strongly interconnected with the rest of the network, or all links are so sparse that each node is critical. Thus, in UDG graphs there are no “special” nodes. They all have very similar convergence rates. Removing fastest converging nodes is equivalent to removing randomly chosen nodes. Figure 5.4 just confirms the well known fact that UDG graphs are resilient to random failures.

The next question we tackle is whether there is an analytical explanation for the above presented empirical results? Recall the two global measures we defined in Section 5.2: *average betweenness* and *square closeness*. They describe the importance of nodes for distributing information within a network. To answer the above question we test to what extent the top nodes identified by BridgeFinder overlap with the best nodes identified by the two global measures.

For each of our test networks we compute two sets: A and B . A consists of 5% (i.e. 13 of all 250 nodes) of the highest average betweenness nodes and B contains the highest 5% square closeness nodes. The nodes in the union of A and B , $A \cup B$, have either the highest average betweenness, or the highest square distance coefficients in the network, or in most of the cases both. Note that A and B are not necessarily disjoint. On the contrary, averaged over all generated networks, A and B overlapped to over 80%. The overlapping factor is almost identical for the four different network types.

The set $A \cup B$ contains the nodes with the “best” topological properties in the network. The question is how many of those nodes are within the nodes identified by BridgeFinder. We intersect $A \cup B$ with 5% (13 nodes) of the fastest converging nodes. Figure 5.5 displays the results. At least half of the fastest converging nodes lay on key topological positions in their networks. This is the case in more than 80% of the two-bridge networks and in over 90% of the one-bridge networks.

Our results clearly show that with very high probability BridgeFinder identifies the same nodes one would get by using global network measures. In both cases those are the few nodes keeping the network together, see Figure 5.4.

There is however a **huge** difference between the two approaches. We had to use global network knowledge in centralized algorithms to compute the above discussed topological properties of the nodes. Such global knowledge is unavailable in distributed environments, rendering that kind of algorithms useless in practice.

The striking advantage of BridgeFinder: It relies only on information directly exchanged among the participating peers in a fully distributed manner. That makes it not only applicable in distributed environments, while still being compatible to centralized approaches, but the results it produces are also simultaneously available to all participating peers.

All this together makes BridgeFinder outstanding in the area of distributed algorithms for detecting communication bottlenecks, as we discuss in detail in Section 5.6.

5.5 Gossiping Convergence

The convergence speed of the BridgeFinder algorithm is determined by the diffusion speed of the underlying network. The diffusion speed characterizes how fast a value starting from an arbitrary node diffuses through the whole network.

Another gossiping protocol based on diffusing values through the underlying network is Push-Sum [116]. We use it as a benchmark in our comparison as BridgeFinder similarly relies on a basic value exchange operation.

In graphs with high expansion the convergence speed of Push-Sum has been proven to be $O(\log N + \log \frac{1}{\epsilon})$ [116], where N is the number of nodes and ϵ the relative convergence error. For geometric networks with “roughly” uniform distribution of nodes in the Euclidian space the diffusion speed is shown to be $O(\log^{1+\epsilon} D)$ [117], where D is the diameter of the network.

Many complex networks have a different topology than the networks described above. They consist of densely connected clusters which are only sparsely interconnected. An O -notation solution for the running time of Push-Sum in such networks is only available for special metric spaces. A general proof is not yet available [117]. Both, their irregular structure and varying number of clusters, make it very unlikely that a general solution for mobile multihop networks could ever be derived.

Therefore, we investigate the convergence of Push-Sum through extensive simulations on all 500 instances of each ODG network type that we investigated in Section 5.4. The averaged results are displayed in Table 5.1. The result for a single network is measured as the required for the network to converge average number of exchange operations per node. We observe a large discrepancy in the required exchange operations between the ideal case, the UDG graph, and the most demanding case with one bridge.

Note that the gossiping protocol of Push-Sum is indeed information diffusion within the network. Therefore, the faster the node that starts the algorithm distributes information within the network, the better is the convergence time of the protocol. From Section 5.4 we know that those are with high probability exactly the fastest converging nodes of BridgeFinder.

Network Type	Push-Sum	BridgeFinder
3 Bridges	469	214
2 Bridges	597	282
1 Bridge	823	367
UDG	94	72

Table 5.1: Convergence speed on different network types measured in average number of exchange steps per node.

An intuitive solution for optimizing the original Push-Sum protocol arises. After running the algorithm once, the fastest converging node from the last run is responsible for starting the next run. If it is not able to do that, the second fastest converging node becomes responsible for starting the algorithm and so on.

This modification is integrated in BridgeFinder. Recall from Section 5.3 that the list of the fastest converging nodes is constantly exchanged among the participating nodes and that at the end of the algorithm each node possesses the list of the k fastest converging nodes. Thus, all nodes within the network are aware which node should start the next run.

We only have to overcome one last problem: starting the algorithm for the first time while avoiding different nodes from starting and running BridgeFinder simultaneously. To achieve that each node which decides to start the algorithm and has not already participated in it, picks a random number and sends it over the network together with its values.

When a node receives values with different random numbers, this means that two simultaneous instances of BridgeFinder are running. The node ignores the gossiping messages with the smaller random number and processes only the larger values. Thus, BridgeFinder instances started with larger random numbers take priority over those started with smaller numbers. This produces a slight overhead during the first run of the algorithm, as initially computed values are being thrown away, but still gives each node the possibility to start the algorithm. This resolves the problem of having to pick a starting node within a distributed environment. Any node can start the algorithm.

The benefit of starting BridgeFinder from the fastest converging nodes from previous runs of the algorithm can be seen in Table 5.1. We run the algorithm over the 500 instances of each network model and average the results. The values of BridgeFinder are always measured at the end of the second run as the first one is used to determine the top converging nodes within the network. Thanks to our modification BridgeFinder performs better than the original Push-Sum algorithm in all four network models. The acquired speed-up ranges from 20% to 220%.

In summary, even in hostile environments like one bridge network models, our algorithm requires just a few more exchange operations per node than the number of nodes in the network.

It is to be noted, that after BridgeFinder has converged, the computed results are known to all participating peers. That is a clear advantage of BridgeFinder over current distributed algorithms for detecting critical peers, i.e. communication bottlenecks. More details are provided in the next Section.

5.6 Related Work

Gossiping is used in many network applications. One of them is the Push-Sum algorithm [108]. Its original epidemic protocol is called anti-entropy and is used for database replication [118]. Another application of the Push-Sum algorithm is counting peers in P2P overlay networks [119]. An overview of actual epidemic protocols is given by Eugster [120].

Our contribution, BridgeFinder, is fully compliant with the Push-Sum based algorithm proposed in [119]. Our modification enables BridgeFinder to detect global critical peers in a distributed manner. Global critical peers have significant impact on the connectivity of the underlying network. The absence even of a small number of those nodes may easily render the network completely unusable [121].

There is a large body of recent research dedicated to identifying critical peers. The common approach is to send probe messages from a node, let's say C , to other nodes in the network. When a node receives a probe message, it must send it back to C . C contacts iteratively a chosen set of nodes [122] or just floods the network [110]. After all probe messages have returned back to C , it computes how fast the single messages have returned. Based on its results, C decides if it is a critical peer or not. Short return times mean C is critical, otherwise it is not. After the procedure is finished the computed results are known solely to C .

Another approach is the detection algorithm based on the Midpoint Coverage Circle [109]. It also has a simple workflow. Assume that the node C wants to estimate whether it is a critical peer. It chooses random pairs of nodes within the network and asks them to contact each other. If the message exchanged by the two nodes on its way goes through C , then C considers itself critical and not otherwise. The computational overhead for C is smaller than the one in the previous approach. However, in order to guarantee reliable results C has to test a very large number of node pairs. This produces a large messaging overhead for testing just a single node. The acquired information is still solely available to C .

BridgeFinder does not test just some nodes, but rather evaluates in parallel all nodes in the network. When the algorithm is finished, the results are known to all nodes, in sharp contrast to the current methods discussed above. Even with a small state per peer and based only on local computations the results of BridgeFinder are highly accurate.

In networks with expander topology the overhead for finding all critical nodes with BridgeFinder scales with $O(\log N)$, where N is the number of nodes in the network [119]. This is faster than all known approaches, even though they make calculations for only one single node instead of the whole network. Our results show that in wireless multihop networks with arbitrary topologies BridgeFinder is slower just within a factor between 2 and 3 compared to expander graphs, see Section 5.5.

In summary, BridgeFinder outperforms current distributed approaches for detecting critical peers, i.e. communication bottlenecks, on several criteria: it is faster, more reliable and the computed results are known to all participating peers.

The last but not the least, it has an efficient guarding mechanism against malicious nodes, which is presented in the next Section.

5.7 Security Issues

All gossiping-based approaches are unfortunately vulnerable to attacks and malicious behaving parties. BridgeFinder is also based on gossiping and is therefore similarly vulnerable.

Still, it has one major advantage: it does not depend on the exact values exchanged among nodes, but rather on the convergence speed of that process. Relying on the convergence speed instead on the values is sufficient for us to develop highly efficient guarding mechanism for BridgeFinder. That is another significant difference between BridgeFinder and other approaches in its field.

The main idea behind our solution is that no node converges on its own. A node can only claim it has successfully converged when the fluctuations in the values of the surrounding nodes become relatively small. Vice versa, a node can only claim it has still not converged, when the values maintained by its neighbors are still varying significantly.

Therefore, each node just needs a scheme for estimating the fluctuations in the neighbors values either as acceptable or not acceptable. Then, it can protect both itself and the whole network from the following two types of possible attacks.

The first attack is when a compromised node falsely claims it has converged (lies to its neighbors about its real value), in order to qualify for the list of fastest converging nodes. The second type of attack is when a node neglects the values sent to it by its neighbors and keeps flooding the network with higher/smaller values and thus hindering the algorithm from converging.

When a node receives suspiciously small or high values from one of its neighbors, let say X , it reacts by warning its other neighbors. If its suspicions are confirmed by any other node, they send a message through the network that X has been compromised. All nodes having X in their neighbor list exclude X from their future communication flow.

To estimate if the values received from its neighbors are legitimate, each node calculates their mean value. The local decision states: a value is legitimate when it differs from the mean value only by a given factor. Now all we need is a scheme to determine that factor.

We know that the fluctuations in the values are significant at the beginning of the algorithm and decrease progressively with each iteration. The speed of that progression for a node ν at time i can be expressed through the difference of its new and its old value:

$$s_\nu(i) := \frac{x_i - x_{i-1}}{x_i} \quad (5.5)$$

where x_i and x_{i-1} are the current and the last value of ν respectively. We call s_ν the speed coefficient of ν . As the algorithm converges, s_ν goes to zero for all nodes in the network.

Furthermore, each node knows the number of exchange operations it has already carried out, see Section 5.3. Based on that number and the speed coefficients of the values it receives, each node qualifies them as suspicious or not suspicious.

The local guarding mechanism for a node ν works in three steps. When at iteration l , ν receives a new value x from its neighbor w , then: (i) ν computes the speed coefficient of w at timestamp l , namely $s_w(l)$; (ii) ν computes the mean value of the speed coefficients of the rest of its neighbors MS_ν ; and (iii) ν checks if the following inequality holds:

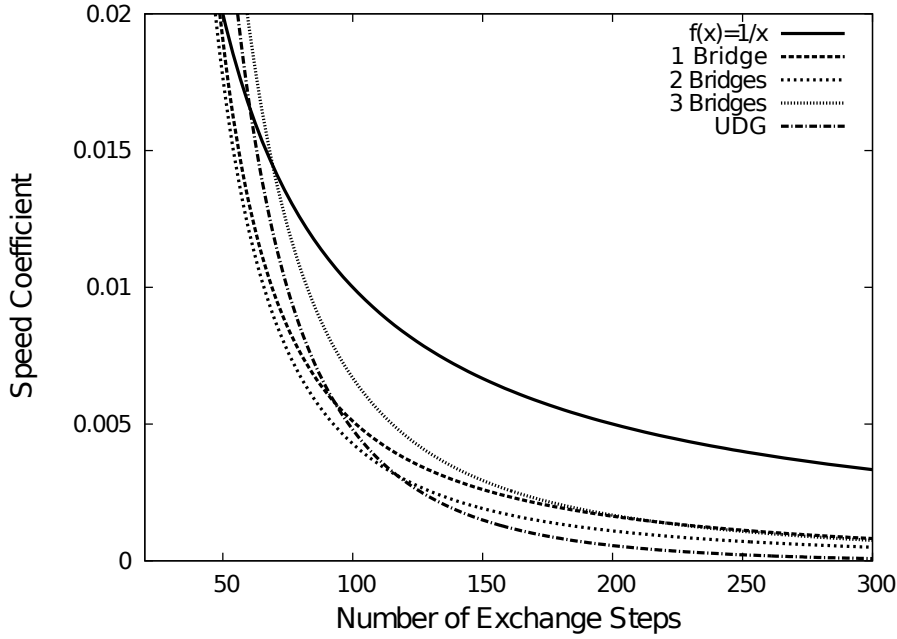


Figure 5.6: Average speed coefficient within 100 runs of BridgeFinder on all four network types

$$|s_w(l) - MS_v| \leq \frac{1}{l} \quad (5.6)$$

When the inequality (5.6) holds, ν accepts the new value x of w . Otherwise, it marks x as suspicious and reports it to its other neighbors. If enough nodes (one can vary the number of required votes from one to all neighbors) confirm the suspicions of ν , then w is excluded from further communication flow within the network.

Figure 5.6 shows the progression of the speed coefficients averaged over all nodes in the corresponding network and the function $1/x$. All plots are averaged results over 100 instances of each network type investigated in Section 5.4. The speed coefficients in all network types have similar gradient. They are all dominated by $1/x$ for $x > 70$.

One can easily conclude that the decision rule proposed in inequality 5.6 is inadequate within the first 70 iterations, but is very tight afterwards. That is, we allow compromised nodes to lie about their values at the beginning of the algorithm. The huge fluctuations in the beginning of the algorithm make it highly unlikely that a meaningful mechanism for that phase of the algorithm exists. At the beginning large values are exchanged with very small values and that makes every exchange operation suspicious.

Our security mechanism is still successful if it overcomes the following three issues: (i) detects nodes falsely claiming they have converged; (ii) detects nodes that manipulate their values after the network has stabilized; and (iii) malicious nodes changing their values always within the tolerance interval should not influence the convergence speed of the whole network.

We directly address the first issue. A node cannot possibly lie it has converged. Once the network has stabilized, its neighbor will detect that its speed coefficient is extremely close to zero in comparison to its surrounding nodes.

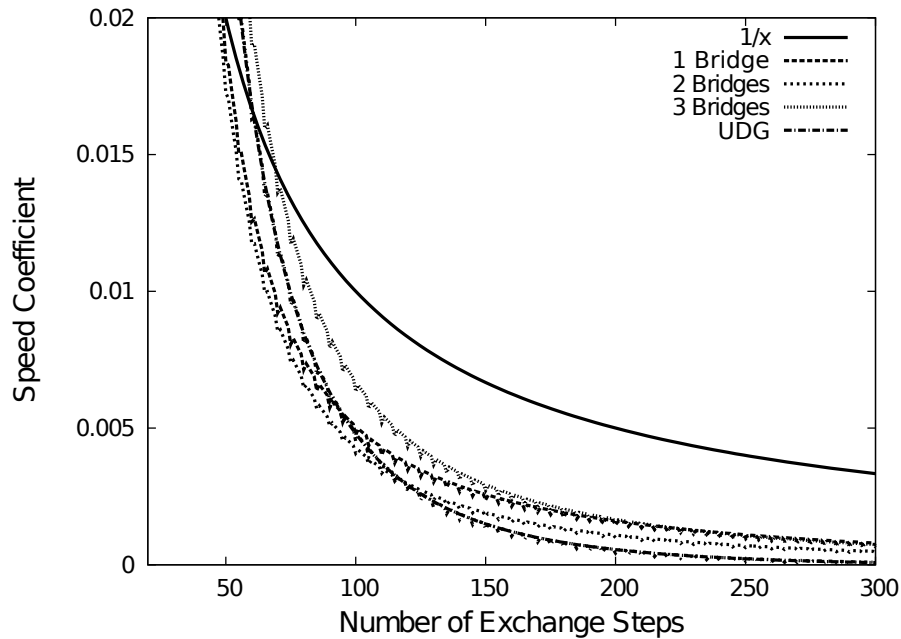


Figure 5.7: Perfect attack at every 5th iteration of BridgeFinder with 10% of malicious peers

Regarding the second issue, it is also possible to derive the magnitude to which a given value can be manipulated from inequality (5.6). One can easily calculate that after only 100 iterations even discrepancies within 5% of the real value will be detected as suspicious. The size of that buffer drastically decreases with each next exchange operation.

To address the final issue, we simulate the following scenario: 10% of the nodes behave as malicious parties. They all have an oracle that provides them with the mean value of the speed coefficients of their neighbors (these mean values are not known under realistic circumstances, but represent an upper bound for possible unsuspecting behavior by malicious nodes). Then we let all malicious nodes lie within the maximum buffer at each 5th iteration.

The results are displayed in Figure 5.7. Even under the above described *perfect attack*, the overall convergence time of BridgeFinder for all four network types stays intact. Compromised nodes affect the convergence speed only marginally. This guarantees the proper function of our approach even under substantial presence of compromised nodes.

In summary, the key observation that BridgeFinder does not rely on the actual exchanged values, but only on the convergence speed of that process, provides our approach with efficient guarding mechanism against malicious parties. In that sense, BridgeFinder clearly outperforms other approaches in its area.

5.8 Summary and Outlook

Throughout the previous Chapters of this work we have shown that by closely investigating the topology and more precisely the local structures of complex networks, one can acquire an inside into the nature of the dynamic processes taking place on those underlying networks.

We then have utilized that knowledge in online topology control mechanisms. We have shown that by carefully steering the local structures of complex networks, it is indeed possible to control the dynamic performance of those networks in a distributed manner.

Because many communication networks have distributed nature, that is actually the only available efficient and still not demanding type of topology control. Consequently, we were able to derive highly competitive to the state of the art approaches for load balancing in P2P and resilient P2P live-streaming.

In this Chapter, we have adopted the reverse perspective. Instead of steering the dynamic processes taking place on distributed networks by carefully adapting their topology, we have shown that one can deliberately deploy a dynamic process on its own, in order to reveal weak articulation points in the underlying network topology.

Consequently, we were able to derive a novel distributed approach for detecting critical peers, i.e. communication bottlenecks, in decentralized communication networks, multihop networks in the demonstrated case. As we have shown, our approach outperforms current state of the art methods with respect to several perspectives.

5.8.1 Summary

In this Chapter we have presented BridgeFinder, a distributed algorithm for identifying critical nodes in complex networks. Critical nodes are nodes whose disappearance would partition the network. Our extensive evaluation shows high consistency between the critical nodes identified with classic graph theory and the ones identified in a distributed manner by BridgeFinder. Detecting and protecting those few nodes is vital for keeping the network operational. The benefit of using BridgeFinder is significant in fast changing distributed environments such as wireless ad-hoc networks, sensor networks and P2P networks.

Furthermore, BridgeFinder relies only on simple exchange of information among the participating nodes. Being as generic as it is, it can be integrated in any maintenance or other existing communication flow already utilized by the network and hence cause no additional overhead.

Another advantage of our approach is that it does not rely on the actual information exchanged among the participating nodes, but rather only on the convergence speed of that process. This approach is completely transparent to the gossiping process and allows the deployment of efficient guarding mechanism against malicious nodes.

To the best of our knowledge, there is no equivalent distributed approach. BridgeFinder is fast, reliable, produces almost no additional messaging overhead and has high resilience to malicious behaving parties.

5.8.2 Outlook

Alike many novel approaches, BridgeFinder also has some minor issues to be resolved.

The first one is in the context of its first run. Although augmenting initial iterations with random identifiers resolve the problem of starting the algorithm in a distributed environments, it causes unnecessary overhead, see Section 5.3. In case of multiple instances of BridgeFinder

running simultaneously during the first run, already computed results of instances with smaller initial numbers are discarded. Instead of disregarding already available results, one could merge results between different instances and thus avoid unnecessary overhead during the first run of the algorithm.

The second issue of BridgeFinder are attacks by multiple malicious parties working together. If the overlay running on top of the underlying network allows peers to change their positions, a group of malicious peers may surround a targeted peer. Then, they could eventually claim that peer misbehaving and try to ban it from the network. Similar attacks are possible in many other distributed application. In such cases one relies on trusted authorities and reputation mechanism to detect false claims by third parties. Such reputation mechanism can be adopted in BridgeFinder as well, which will increase its resilience even further.

The last issue concerns the convergence of our algorithm. Independent of the solid empirical results, analytical estimate of its convergence speed will help indicate if any further speed-up techniques would be necessary for large application scenarios. Unfortunately, the diverse and improper topology of distributed communication networks leaves very little spare hope that such analytical proof does exist.

Despite the just listed open issues, it should be noted that BridgeFinder already outperforms current state of the art distributed methods for detecting communication bottlenecks with respect to several perspectives: speed, reliability, messaging overhead and resilience.



6 Efficient Search and Lookups in Peer-to-Peer Networks

Through the previous Chapters we have shown that local structures can reveal unexpected relations between dynamics and topology in complex networks, high citation frequencies in the demonstrated case of co-authorship networks, see Chapter 2. We also have investigated local structures from previously unexplored and very promising perspective: How to engage them in local decision rules for online topology control, see Chapters 3 and 4.

With the help of local structures we have been able to provide much better load distribution within existing P2P overlays with none or very modest overhead, see Chapter 3. We also have been able to provide a framework for very robust and highly resilient P2P live-streaming, outperforming the current state of the art methods, see Chapter 4.

However, the potential of local structures is not limited to the relation between specific local topology and specific network performance. In this Chapter we investigate a slightly orthogonal perspective and show that random local structures and their random graphs still have unexploited potential.

Recently, random graphs have become notorious for being poor null models and simulation testbeds of various real world networks. Nevertheless, random graphs have many outstanding properties, most of which are highly desirable in any technological network. Those include short average path length, several disjoint paths among nodes, high error tolerance, etc.

In this Chapter we show that random graphs, and their local structures can effectively be utilized in local decision rules, which provide a unique platform for the first P2P overlay that supports efficient exhaustive search and DHT alike key lookup within the same P2P overlay.

P2P networks are divided into two main classes: unstructured and structured. Overlays from the first class are better suited for exhaustive search, whereas those from the second class offer very efficient key-value lookups. In this Chapter we present a novel overlay, called PathFinder, which for the first time combines the advantages of both classes within one single overlay. Our evaluation shows that PathFinder is comparable or even better in terms of lookup and complex query performance than existing P2P overlays and scales to hundreds of millions of nodes. Peers in PathFinder are arranged as Erdős Renyi random graphs. Consequently, all overlay operations such as key-value lookup, complex queries and maintenance messages greatly benefit from the short average path length, the high number of alternative paths and the robustness of the underlying random graph topology.

6.1 Introduction

P2P overlay networks can be classified into *unstructured* and *structured* networks, depending on how they construct and manage the overlay [123].

In an unstructured network the peers are free to choose their overlay neighbors and what they offer to the network.¹ In order to discover if a certain piece of information is available a peer must somehow search through the overlay. There are several implementations of such search algorithms. The original Napster used a central index server, Kazaa relied on a hybrid network with supernodes and the original Gnutella used a decentralized flooding of queries [123]. The BubbleStorm network [119] is a fully decentralized network based on random graphs and is able to provide efficient exhaustive search over all peers. The query evaluation is performed locally by a peer holding the document and receiving the search query. In that sense every query evaluation method available (e.g. full text, XQuery, SQL, etc.) may be applied.

Structured networks, on the other hand, have strict rules about how the overlay is formed and where content should be placed within the network. Structured networks are also often called distributed hash tables (DHT) and the research world has seen several examples of DHTs [71, 77, 124, 125, 126, 127, 128, 129, 130]. DHTs are based on hashing peer and object identifiers and distributing the ID space among the peers. A DHT-specific routing algorithm defines how peers can route through the overlay when they want to retrieve a certain object. Typically, the number of messages needed to locate an object in a DHT grows logarithmically with the number of peers in the system. Thus, DHTs are very efficient for simple key-value lookups (for which they have been designed). Because objects are addressed with their unique names, searching in a DHT is hard to be made more efficient [131, 132, 133]. However, DHTs require the use of (globally) unique object identifiers, for example SHA-1 hashes, which are not very suitable for human users. In addition, wildcard searching and complex queries either impose extensive complexity and costs in terms of additional messages or are not supported at all.

Given the attractive properties of these two different network structures: (i) human-friendly keyword searches in unstructured networks and (ii) computer-friendly and efficient lookups in DHTs, a crucial question arises: *Is it possible to combine these two properties in one single network?*

Naturally, it would be possible to run two overlays in parallel, but that would require as much as twice the maintenance traffic and state keeping as well as space when objects are replicated. On the other side, a single overlay is much more desirable since it has considerably lower overhead, both in terms of overlay maintenance and replication effort.

Our answer to the above question is PathFinder, a P2P overlay which combines an unstructured and a structured network in a single overlay. PathFinder is based on a random graph which gives it short average path length, large number of alternative paths for fault tolerance, and highly robust and reliable overlay topology. Furthermore, the number of neighbors in PathFinder does not depend on the network size. Therefore, the load of individual peers in PathFinder remains constant even if the network grows up to 100 million or more peers. The main contribution of this Chapter is the efficient combination of exhaustive searching and key-value lookups in a single overlay.

We evaluate PathFinder analytically as well as empirically and investigate its resistance to churn and its robustness. Our results clearly show that PathFinder is highly scalable, fast, robust and requires only a small per-peer state. In terms of exhaustive search performance PathFinder

¹ In this Chapter we focus on networks where peers store and share content, e.g., files, database items, etc.

is comparable to BubbleStorm [119]. In terms of DHT-like lookup performance, our results show that PathFinder is at least as good as current DHTs and in most cases is able to retrieve objects with even less overlay hops than other DHTs.

6.2 Motivation

Unstructured networks are very suitable for human-friendly keyword searches. Note that the search is not limited to simple keywords, any query which can be evaluated locally on a peer is possible. However, unstructured networks offer little or no guarantee about finding the object neither it is possible to identify identical objects in an easy manner.

It is of course possible to compare two objects by downloading them both or comparing hashes of their contents. However, this assumes that both hashes have been calculated by trusted entities, which may not be the case in a P2P network.

Structured networks, on the other hand, offer very efficient means of looking up known objects, but their ability to perform widely spanning searches is limited and costly.

Being able to combine both efficient lookups and efficient search is extremely useful for distributed applications and human users. An efficient search allows users to discover what content is available. An efficient lookup allows them to retrieve the content later on or forward it to their friends, knowing that the friends will see the same content. Analogically on the Web is making a Google search and then bookmarking the resulting web page. The bookmark can then be retrieved later or sent to friends. The **key difference** of PathFinder is that there is no need for a centralized indexing service. The search and lookups are both **built into the same system architecture**.

6.3 System Design: PathFinder

In this Section we present the system design and preliminaries of PathFinder. We also describe how basic processes like key-value lookup and exhaustive search work as well as how our overlay manages nodes joining/leaving the network and handles crashed nodes. Finally, we discuss how PathFinder can be built in a practical scenario.

6.3.1 Challenges

We designed PathFinder to be fully compliant with the concept of BubbleStorm [119], namely an overlay structure based on random graphs. We augment the basic random graph with a deterministic lookup mechanism (see Section 6.3.4) to add efficient lookups to the exhaustive search provided by BubbleStorm.

As shown in [119] and discussed in [134], overlays based on random graphs are highly resilient even to large simultaneous crashes of peers. At the same time, they provide short average path lengths among the participating peers.

The challenge and one of the key contributions of this Chapter is to develop a deterministic mechanism for exploiting these short paths in order to implement DHT-like lookups.

PathFinder meets the following key requirements:

- **Scalability:** The average path length of an object lookup grows with $\ln(N)/\ln(c)$, where N is the number of peers and c the average number of neighbors per peer.
- **Constant per-peer state:** The list of neighbors maintained by each peer does not depend on the network size.
- **Flexible exhaustive search:** Thanks to its underlying random graph topology, PathFinder supports exhaustive search with tunable success probability [119]. Any type of queries, which can be processed locally are supported.
- **Key lookup:** Locating an object in the network is competitive or even faster (in terms of overlay hops) than in current DHT implementations.

6.3.2 System Model and Preliminaries

All processes in PathFinder benefit from the properties of its underlying random graph and the routing scheme built on top of it.

In the following, we summarize the properties of random graphs that PathFinder relies on. Then we show the main principle of the routing approach. Finally, we present an example of a small PathFinder overlay network.

Erdős-Rényi random graphs

Erdős-Rényi random graphs² have many attractive features. Those include short average distance between the nodes and small diameter (both increase only logarithmically with the network size), high resistance against node failures, and the existence of several disjoint paths between any two nodes in the network [4].

The average path length of a random graph can be estimated by $L = \frac{\log(N)}{\log(c)}$, where c is the average number of neighbors per node and N the number of nodes in the network. All these properties are highly desirable in any P2P overlay.

The challenge in building a P2P overlay on top of a random graph is that they have no characteristic structure, which implies that there is no rule stating which peer is a neighbor of which other peer. This is exactly the opposite of DHT overlays, which have construction principles allowing each node in the network to compute its neighbors in an unambiguous manner. This property enables DHTs to perform extremely efficient key lookups.

PathFinder's main contribution lies in defining a mechanism for reconstructing the neighbor list of another node in an Erdős-Rényi random graph. That provides a very robust network topology with straight-forward exhaustive search and exact key-value lookup mechanisms. Our solution allows for a **completely local reconstruction** of the neighbor lists, no additional network communication is required.

² In the remainder of this Chapter we use simply the term random graph

Construction Principle of PathFinder

The basic idea of PathFinder is to build a robust network of virtual nodes on top of the physical peers (i.e. actual physical nodes). Routing among peers is carried out in the virtual network. The actual data transfer still takes place directly among the physical peers. PathFinder builds a random graph of virtual nodes and then distributes them among the actual peers. At least one virtual node is assigned to each peer. From the routing point of view, the data in the network is stored on the virtual nodes.

When a peer *A* is looking for a particular piece of information it has to find a path from one of its virtual nodes to the virtual node containing the requested data. Then *A* contacts the underlying peer *B*, responsible for the targeted virtual node, and *A* retrieves the requested data directly from *B*. This process is described in detail in Section 6.3.4.

It is known that the degree sequence in a random graph is Poisson distributed. Therefore, we need two pseudo random number generators (PRNG), which initialized with the same ID always produce a deterministic sequence of numbers. Given a number c , the first generator returns Poisson distributed numbers with mean value c . The second PRNG, given a node ID produces a deterministic sequence of numbers which we use as IDs for the neighbors of the given node.

The construction principle of PathFinder is then as follows. First we fix a number c (see Section 6.3.8 on how to choose c according to the number of peers and how to adapt it once the network becomes too small/large). Then, for each virtual node we determine the number of neighbors with the first number generator. The actual nodes IDs to which the current virtual node should be connected are then chosen with the second number generator. The number generator is started with the ID of the virtual node. The process can be summarized in the following steps:

1. The underlying peer determines how many virtual nodes it should handle. Section 6.3.6 provides more details.
2. For every virtual node handled by the peer:
 - a) The peer uses the poisson number generator to determine the number of neighbors of the current virtual node.
 - b) The peer then draws as many pseudo random numbers according to the number drawn in the previous step.
 - c) The peer selects the virtual nodes with IDs matching to those numbers as neighbors for its current virtual node.

The following is a pseudo code implementation: The function `nextPoisson` is initialized with the current virtual node ID and returns a pseudorandom number from a Poisson distribution to determine the number of neighbors. The function `nextRandom` is initialized with the current virtual node ID as well and returns a deterministic random numbers uniformly distributed between 0 and N , where N is the number of virtual nodes in the network.

```

Input:  $c$ 
1 foreach  $vNode$  do
2    $numNeighbors = nextPoisson(c, vNode.getID());$ 
3    $random\_seed = init\_random\_seed(vNode.getID());$ 
4   while  $i < numNeighbors$  do
5      $neighborID = random\_seed.nextRandom();$ 
6      $vNode.store(neighborID);$ 
7      $i = i + 1;$ 
8   end
9 end

```

Algorithm 3: PathFinder Neighbor List Construction

The construction mechanism of PathFinder allows the peers to build a random graph out of their virtual nodes. It is of crucial importance that a peer only needs a PRNG to perform that operation. There is no need for network communication. Analogically, **any peer** can determine the neighbors of **any virtual node**, by simply seeding the pseudo random number generator with the ID corresponding to that virtual node.

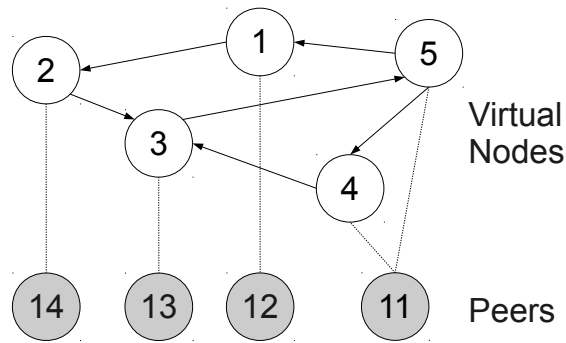
Now we have both, a random graph topology suited for exhaustive search and a mechanism for each node to compute the neighbor list of any other node. As we discuss in detail in Section 6.3.4, that is sufficient (disregarding the local computation the peer has to perform) for any peer to contact any other targeted peer in the network by traversing just one single path. Thus, we can guarantee an efficient DHT-similar behavior within the PathFinder overlay.

Note that neighbor links in the random graph are *directed*. The routing table of a peer is determined by the neighbors of its virtual nodes. It contains all the direct neighbors of all of its virtual nodes in the random graph. These tables are easy to maintain, because all peers hold only between one and two virtual nodes on average (i.e. c to $2c$ virtual neighbors). As our results show, value of $c = 20$ is sufficient for good performance and better performance can be obtained for even higher values of c . One entry in the routing table contains just the virtual node ID and its IP address. Hence, the value of c could possibly be set much higher. Routing tables with more than hundred entries are common in e.g. Kademia, Pastry and other P2P overlays.

PathFinder Routing Table Example

Figure 6.1 shows a small sample of PathFinder with a routing table for the peer with ID 11. The random graph has 5 virtual nodes (1 through 5) and there are 4 peers (with IDs from 11 through 14). Peer 11 handles two virtual nodes (4 and 5) and the rest of the peers have one virtual node each. The arrows between the virtual nodes show the directed neighbor links.

Each peer keeps track of its own outgoing links and of the incoming links from other virtual nodes. A peer notices incoming links from other peers when they initiate communication. Keeping track of the incoming links is strictly speaking not necessary, but makes key lookups much more efficient (see Section 6.3.4). The routing table of peer 11 consists of all outgoing links



	Outgoing Links		Incoming Links
Node ID	3	1	3
Peer	13	12	13

Figure 6.1: A small example of the PathFinder overlay.

from its virtual nodes 4 and 5 and the incoming link from virtual node number 3. In general, every peer is responsible for keeping its outgoing links alive. In contrast to established DHTs, the maintenance costs of PathFinder does not depend on the network size as the average number of neighbors within the random graph is fixed.

6.3.3 Storing Objects

Each object stored in PathFinder has a unique identifier. This identifier is derived by hashing the contents of the object with some hash function (e.g. SHA-1). That provides some degree of security against object modifications since any peer can verify that the content is what it is supposed to be. Objects such as news sites, which have changing content but would prefer to remain under a single identifier, are free to compute their hashes in any other manner.

An object is stored on the virtual node (i.e. on the peer responsible for the virtual node) which matches the object's identifier. If the hash space is larger than the number of virtual nodes, as with SHA-1, then the object is mapped to the virtual node whose identifier matches the prefix of the object hash.

There is no need for an additional lookup service since PathFinder provides exhaustive search over all the objects (see Section 6.3.5). When a peer is looking for an exact object, like a concrete file, it uses the hash function to compute the object's identifier. Then the peer performs an efficient key lookup to the corresponding virtual node (DHT similar behavior). When the peer is looking for a range of objects, like all files containing a given regular expression in their titles, the peer performs exhaustive search which returns the object and its identifier (unstructured overlay similar behavior). Subsequent retrievals as well as parallel requests to replicas of the same object can be done by using the identifier to perform a lookup (Section 6.3.4). This is a strong advantage of PathFinder. As soon as one copy of an object is found through exhaustive search, all remaining copies can easily be accessed in parallel through a subsequent key lookup.

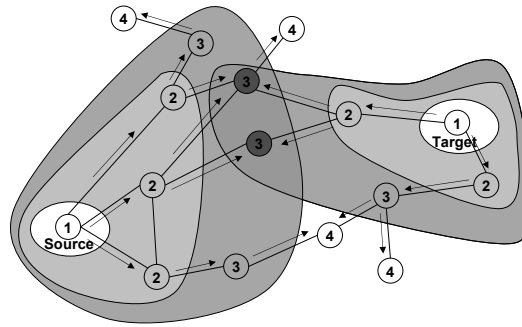


Figure 6.2: Key lookup with local expanding search rings from both the source and the target.

6.3.4 Key Lookup

Key lookup is the process when a peer contacts another peer possessing a given data of interest. Using the structure of the network, the requesting peer traverses only one single and usually short path from itself to the target peer.

Key lookup is the main function of a DHT. In order to perform quick lookups, the average number of hops between peers as well as the variance needs to be kept small. We now show how PathFinder achieves efficient lookups and thus behaves as any other DHT. Suppose that the peer A wants to retrieve an object O . A determines the virtual node w responsible for the object O by using the hash function described above. Now A has to route in the virtual network from one of its virtual nodes to w and directly retrieve O from the peer responsible for w .

Denote with V the set of virtual nodes managed by the peer A . For each virtual node in V , A calculates the neighbors of those nodes. A checks if any of those neighbors is the virtual node w . If yes, A contacts the underlying peer to retrieve O . If none of peer A 's virtual node neighbors is responsible for O , A calculates the neighbors of all of its neighbors, i.e. its second neighbors. Because the neighbors of each virtual node are pre-known (see Section 6.3.2), this is a simple local computation. Again, peer A checks if any of the new calculated neighbors is responsible for O . If yes, peer A sends its request to the virtual node whose neighbor is responsible for O . If still no match is found, peer A expands its search by calculating the neighbors of the nodes from the previous step and checks again. The process continues until a match is found. In the worst case, A has to calculate the neighbor list of each node in the network, but a match is guaranteed.

For an average degree of c per virtual node, the above process requires us to compute c^i nodes for each step i . This becomes unwieldy for large networks which may require several number of steps. For example, with $c = 20$ and 100 million nodes we need about 8 steps, i.e., $20^8 = 2.5 \cdot 10^{10}$ nodes. We mitigate this problem by **expanding the search rings from both A and w simultaneously**, as illustrated in Figure 6.2.

Because peer A is able to compute w 's neighboring virtual nodes, A can expand the search rings **locally** from both the source and target sides. This process is called forward/backward chaining. In every step the search depth of the source and target search ring is increased by one. In that way the number of rings around the source are divided between the source itself and the target. This leads to exponential decrease in the number of node IDs that have to be computed.

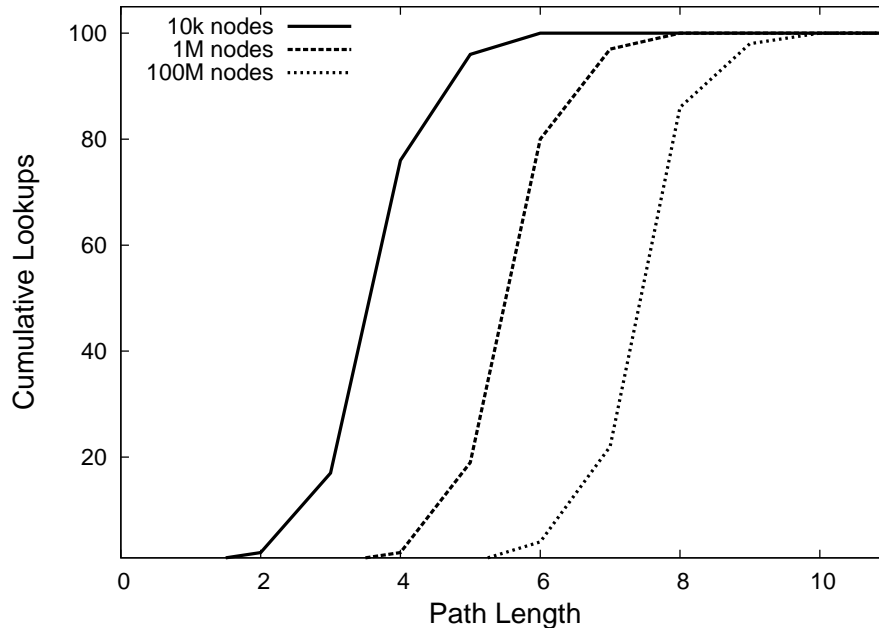


Figure 6.3: Distribution of complete path length for 5000 key lookups with $c = 20$.

Note that for efficiency reasons, peer A can keep the search rings on its own side pre-computed in memory. This way, peer A only needs to start expanding the ring on the target side.

Now assume that the virtual node v is the intersection among the search rings around the source and the target. Recall that the edges among virtual nodes are directed, but that the underlying peers also keep track of their incoming links. That is, we now have a path from one of the virtual nodes of A to v and a path from w to v . But how do we construct a path from A to w ? All the nodes between w and v keep track of their incoming links. Therefore, they can also traverse the path backwards from v to w and thus provide A with a routing path to w .

A passes the discovered path along with the lookup message. Thus, every peer on the path knows immediately to which of its neighbors it should forward the query. In essence, PathFinder uses source routing for key lookups. Note that the whole computation of the path happens **locally** on the source peer. **No additional messages have to be communicated.** All costs come in the form of memory usage and computation time on the source peer. Those are however negligible for any regular computer: around 3.2 Megabytes of memory storage and simple integer computations on hashtables with several thousand entries. That was the maximum computer power required for carrying out the experiments described below.

We generate various PathFinder networks from 10^3 up to 10^8 nodes with average degree 20. In all of them we perform 5000 arbitrary key lookups. It turns out that expanding rings of depth 3 or 4 (i.e. path length between 6 and 8) are sufficient for a successful key lookup, as shown in Figure 6.3. In the figure the x-axis shows the path length and the y-axis shows the cumulative fraction of observed paths. For example, for 1 million nodes the average path length is concentrated around 6. The theoretical average shortest path length for a random graph with 1 million nodes and average degree 20 is 4.6. The slight difference is caused by the forward/backward chaining.

Figure 6.3 also shows that increasing the network size by a factor of 100 leads to only two additional hops for key lookups. The key lookup performance depends mainly on the average number of neighbors c and only slightly on the number of virtual nodes N . It has been shown that the average path length scales with $O(\ln(N)/\ln(c))$ [4].

6.3.5 Searching with Complex Queries

PathFinder supports searching with complex queries with tunable success rate analogical to BubbleStorm [119]. In fact, since both PathFinder and BubbleStorm are based on random graphs, we implemented the search mechanism of BubbleStorm directly into PathFinder. In BubbleStorm both the data and the queries are sent to some number of nodes, where the exact number of messages depends on how we set the probability of finding a match. We use exactly the same algorithm in PathFinder for searching and the reader is referred to [119] for details.

The only difference is that the success probability in PathFinder is known a priori because it depends on the size of the random graph. The size of the random graph of virtual nodes is known to all peers.

6.3.6 Node Join and Leave

We now describe how peers join and leave the PathFinder overlay. The following invariants must hold at any time:

- All virtual nodes are assigned to peers.
- Outgoing links are maintained by the peer responsible for the corresponding virtual nodes.
- Incoming links are kept in the peer's state.

The purpose of the node join process is to integrate a new peer into the PathFinder overlay. The join process can also be used to automatically rebalance the network load, because virtual nodes may be unevenly distributed among the peers. When a new peer B wants to join the network it contacts a peer A that is already part of the network. The first virtual node assigned to B is calculated as the hash of its IP address. Peer A routes the join request using the key lookup procedure (Section 6.3.4) to the virtual node which matches B 's identifier. Let this node be handled by the peer C . C hands one or more of its virtual nodes over to B and informs its neighbors about the new peer B .

In Section 6.3.8 we consider the case where C has no excess virtual nodes so that B has to contact several other peers to find a free virtual node and how the network adapts to such cases.

A successful join means that: (i) a peer releases some of its assigned virtual nodes to the new peer, but keeps at least one virtual node for itself; and (ii) the new peer has successfully established connections to its neighbors. After the join process is completed, the new peer has at least one virtual node and an up-to-date neighbors table. It is straightforward to verify that the three invariants from above hold throughout the whole join procedure, assuming the absence of node failures. We handle them shortly.

When a peer leaves the network, it hands over all of its virtual nodes to its neighbors, which are then responsible for establishing connections to the underlying peers. Note that a peer is

allowed divide its virtual nodes evenly among its neighbors, as opposed to most existing DHTs where the responsibilities are transferred to a single node.

Observation: A peer joining/leaving the network causes on average $c + \ln(N) / \ln(c)$ messages.

When a peer joins the network, it is routed to an arbitrary position determined by the hash function of its IP address. This costs one key lookup, which on average takes $\ln(N) / \ln(c)$ messages. The outgoing neighbors are directly transferred from the issuing node. Then, on average c incoming links transferred from the issuing node need to be updated, which causes additional c update messages.

6.3.7 Node Crash

A node crash is the sudden departure of a peer from the network without correctly following the departure protocol from above. A crash violates the invariants from Section 6.3.6 and results in incorrect neighbor tables.

The absence of the failed peer is recognized by its neighbors when they stop receiving keep-alive messages from it. The time for detecting a failed peer depends on the interval used for keep-alive messages.

When a peer detects another failed peer it calculates locally all neighbors of its virtual node(s). The first peer in the sorted neighbor list has to take over the abandoned virtual node(s). Therefore the peer which has detected the failed peer sends the first virtual node in the list its own IP address and a message that it should start a recovery process.

If the first peer in the neighbor list is not responding, it is replaced by the next peer in the list. Each peer which has detected the missing peer and still has not received a recovery message, also follows the above protocol. In case of concurrent attempts for taking over an abandoned virtual node, the peer with the smallest position in the list continues the recovery process.

After a responsible peer has been determined a new routing table for the failed virtual node has to be generated. For this purpose c key lookups have to be performed. After this step, which needs $c \frac{\ln(N)}{\ln(c)}$ hops on average, a routing table containing all outgoing links of the abandoned virtual node is established.

The remaining incoming links are recovered automatically by the nodes pointing to the failed node. They notice timeouts of keep-alive pings sent to the failed node, because they still have the IP address of the crashed peer. They update their routing tables by performing a regular key lookup to the virtual node. This reveals the IP address of the new responsible peer in $c \frac{\ln(N)}{\ln(c)}$ steps on average.

Observation: A failed peer causes $2c \frac{\ln(N)}{\ln(c)}$ messages on average to repair the network overlay.

Figure 6.4 shows the volume of recovery messages for a simulated PathFinder network with 5,000 peers and different fractions of crashed peers. The x-axis shows the time in steps from the crash to full recovery. One step is at least one roundtrip time between two peers in real time. The y-axis shows the total amount of maintenance messages for each step in the whole system. The crash occurs at step 0 when the failed peers, 10–50% of all peers, disappear at once.

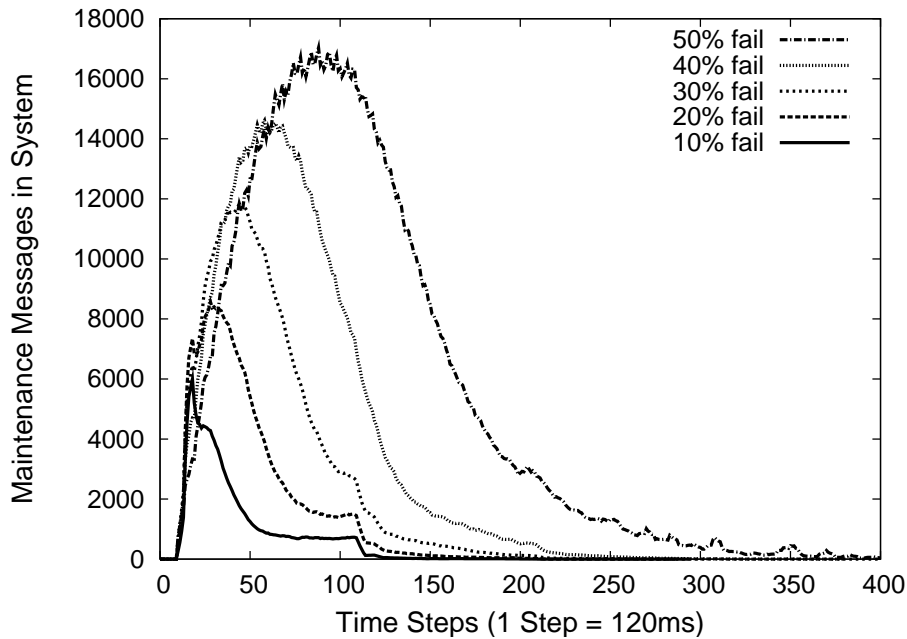


Figure 6.4: Repair costs for network with 5,000 peers

Moderate crashes (up to 30% crashed peers) are healed in about 100 simulated time units, and even a crash of half the peers is practically healed in 300 time units. The message load also remains reasonable, with about 36 messages per peer on average (half of 5,000 peers crashed in worst case and total number of messages in system is under 18,000). Considering the real-world time to heal the system, if 1 time unit is 120 milliseconds then the system would heal itself in 12 seconds for the smaller crashes and in 36 seconds for the 50% crash.

The main determining factor is the ability of peers to send all the required messages in one step, so the recovery could potentially take longer. However, recovery times are still very short, on the order of a few minutes at maximum. Similar performance was observed as well in BubbleStorm [119].

6.3.8 Network Size Adaptation

Because virtual nodes can easily be transferred between peers, PathFinder is able to adapt itself to the current workload. Weaker peers may give up virtual nodes to stronger peers. To keep a reasonable ratio between peers and virtual nodes we have to keep track of the number of peers in the network. We estimate the number of peers in PathFinder with the push-sum gossiping protocol [108]. In [116] this procedure was extended to make it applicable for P2P networks.

Recall the join process from Section 6.3.6. When a new peer joins the network, it receives a number of virtual nodes. However, when most of the peers in the network have only one virtual node, the joining node has to make several requests before finding a peer which still has spare virtual nodes.

The probability of finding a peer with more than one virtual nodes is described through a hypergeometric distribution $f(1; N, m, c)$. A peer can ask any of its c neighbors for virtual nodes and success depends on the number of peers (m) with more than 1 virtual nodes within the

network. When the ratio of peers in the network to average number of neighbors N/c is higher than 0.95, which is true in most of the cases, the process can also be approximated through the binomial distribution. A ratio of virtual nodes to peers of 1.15 establishes a successful join for 80% of all requests within 10 hops ($\mathcal{B}(10, 0.15)$) and over 96% after 20 hops.

We want to keep the cost for a node join reasonable. Therefore, when the threshold of 1.15 virtual nodes per peer is reached, the network starts a transition phase in which the amount of virtual nodes is doubled. Assume that a virtual node w has just run the gossiping protocol and notices that the above threshold is reached. Then w starts the transition phase by generating two new virtual nodes, w_1 and w_2 , by attaching a new bit to the left side of its own ID. The IDs of w_1 and w_2 are computed by attaching 0 respectively 1 to the old ID.

Now w_1 and w_2 have to calculate their neighbors. They proceed as in Section 6.3, but use the new ID space. The IDs of the calculated neighbors also have one extra bit. Let x_1 be one of the neighbors w_1 has calculated. At this moment w_1 still does not have a routing table and cannot route to x_1 . Therefore, w_1 disregards the left-most bit of x_1 's ID and uses the routing table of w to determine the node x corresponding to this ID.

The peer responsible for x will be responsible for x_1 when the transition phase is over, because the ID of x_1 is the ID of x with one bit added on the lefthand side. Therefore, w_1 adds this peer to its new routing table. If x has not yet started the transition phase, then it does so now.

When the above procedure is carried for all new neighbors of w_1 and w_2 , the transition phase for w is completed. When the pushsum protocol confirms that the number of virtual nodes has been indeed doubled, i.e. all virtual nodes have completed the transition, the old routing tables are abandoned.

We expect the transition phase to be a rare event in an operational network. Furthermore, it is not a time critical process. In a large network it can be artificially stretched out to hours or even days, since both the old and the new networks are operational during the transition phase. Actual measurements within the Skype network [135] show that the number of regular users almost doubled between 2006 and 2009, i.e. Skype would have needed only one transition phase within 3 years.

Shrinking the ID space is also possible and works analogically. If the virtual node to peer ratio is higher than 4 to 1, the amount of virtual nodes can be reduced in order to improve lookup performance. The procedure is the reverse of expanding the network, whereby peers strip one bit from left of their IDs.

In case two different peers map to the same stripped ID (quite likely), the one who has leading bit 0 takes over. The peer who has leading bit 1 has to find another virtual node to take over. If we set the threshold of shrinking high enough, like 4 to 1, then there are enough virtual nodes for all peers, but a peer might have to search for a free one.

Note that the average path length and thus routing performance does not substantially change if the network grows or shrinks by a factor of 2, which is a property of Erdős-Rényi random graphs. Still, having the right number of virtual nodes carefully balances between fast join process and modest computational effort per peer.

Protocol	Number of Neighbors	Latency
CAN	$2d$	$(d/2)N^{1/d}$
Chord	$2\ln_2(N)$	$\log_2(N)/2$
Viceroy	10	$\log_2(N)$
Pastry	$(2^b - 1)(\log_2 N)/b$	$\log_2(N)/b$
Symphony	$2k + 2$	$\log^2(n)/k$
PathFinder	c	$\log(N)/c$

Table 6.1: Comparison of various DHTs to PathFinder.

6.4 Comparison and Analysis

Most DHT overlays provide the similar functionality, since they all support the common interface for key based routing. The main differences between various DHT implementations are the average lookup path length, the resilience to failures, and the load balancing.

In this Section we compare PathFinder to other DHTs presented in the literature. We perform the comparisons both with simulations and analytically for networks which are too large to be simulated (over 1 million nodes). All simulations are performed on the P2P simulator Planet-Sim [84]. All overlays performed exactly as described in their corresponding publications.

Overlay Scalability

The lookup path length of Chord is well studied [136]. It is asymptotically given by: $L_{avg}(N) = \frac{1}{(1+d)\log(1+d)-d\log(d)} \log N$, where N is the number of peers in the network. The parameter d tunes the finger density. Usually Chord has finger density $d = 1$ and therefore $L_{avg} = \frac{\log(N)}{2}$. The maximum path length of Chord is $\frac{\log(N)}{\log(1+d)}$.

The average path length of PathFinder is $\frac{\log(N)}{\log(c)}$, where c is the average number of neighbors. In other words, even for relatively small c , PathFinder has much shorter path length than Chord.

The path length of the Pastry model can be estimated by $\lceil \log_{2^b}(N) \rceil$ [127], where b is a tunable parameter. The authors recommend $b = 4$. In this model, there are $\log_{2^b}(N)$ levels and $2^b - 1$ neighbors per level. This results in 96 neighbors for a network of 50 million peers. PathFinder achieves comparable results with only $c = 20$ neighbors on average. For $c = 50$ the average path length of PathFinder drops to 2/3 the path length of Pastry. Theoretically, PathFinder should achieve Pastry's performance for $c = 16$. Since our results show that PathFinder matches Pastry already for $c = 20$, we suspect that Pastry's real-world performance for large networks would not be quite as good as the theoretical model let one expects.

The Symphony overlay is based on a small-world graph. This leads to key lookups in $O(\frac{\log^2(N)}{k})$ hops [130]. The variable k refers only to long distance links. The actual amount of neighbors is indeed much higher [130].

The diameter of CAN is $\frac{1}{2}dN^{\frac{1}{d}}$ with a degree for each node $2d$, with a fixed d . For large d the path length distribution becomes gaussian, like Chord [137].

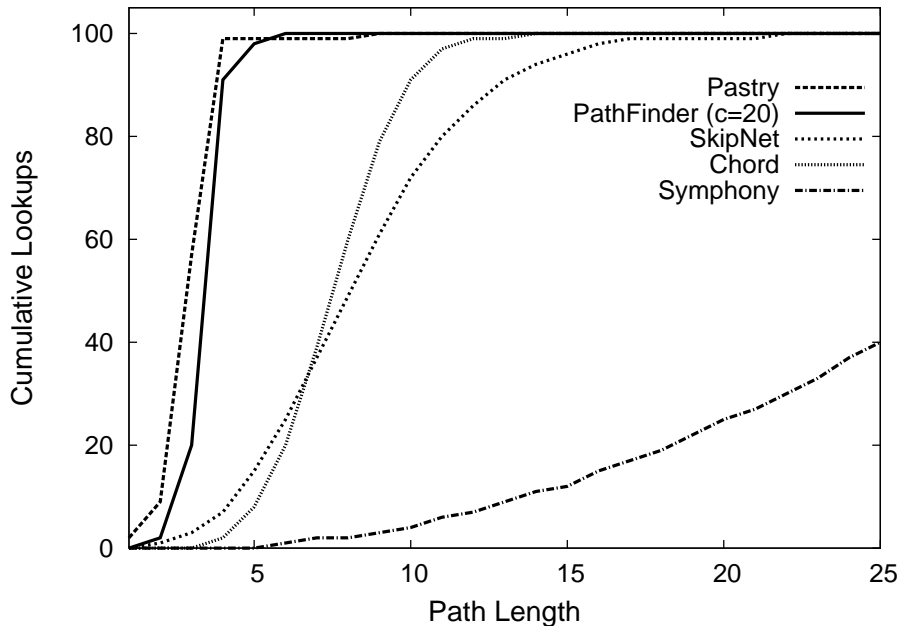


Figure 6.5: Average number of hops for 5,000 key lookups in different DHTs.

The butterfly network has close to optimal diameter and average path length. The average distance in a butterfly network is $\mu_d \approx \frac{3 \log_k(N)}{2}$ [138]. An implementation of the butterfly network, Viceroy [126], has an average path length of $3 \log_2(N)$. The theoretical average path length of PathFinder is $L = \frac{\log(N)}{\log(c)}$, a property of its underlying random graph, see Section 6.3.

Table 6.1 summarizes the characteristics of PathFinder and established P2P overlays.

In summary, most well known DHTs and PathFinder have a path length scaling (up to a multiplicative factor) as $\log(N)$. In this sense, PathFinder performs similar, but it also has a small and fixed number of neighbors, independent from the network size. This is a clear advantage of PathFinder over other DHTs.

Average Path Length

We use simulations to evaluate the practical effects of the various scaling factors described above. We compare PathFinder with Pastry, Chord, Symphony, and SkipNet [139]. Figure 6.5 shows the results for a 20,000 nodes network. We perform 5,000 lookups among random pairs of nodes and measure the number of hops it takes for each of the DHTs to find the object. We plot the number of hops on the x-axis and y-axis shows the fraction of requests which were successful within the corresponding number of hops.

Pastry and PathFinder have very similar performance, with the maximum number of hops being around 4. Chord and SkipNet perform worse, requiring on average 7 additional hops. Symphony's performance is extremely poor, some lookups requiring up to 40 hops (not shown in the figure). CAN and Viceroy have even worse performance and were thus dropped from further comparison.

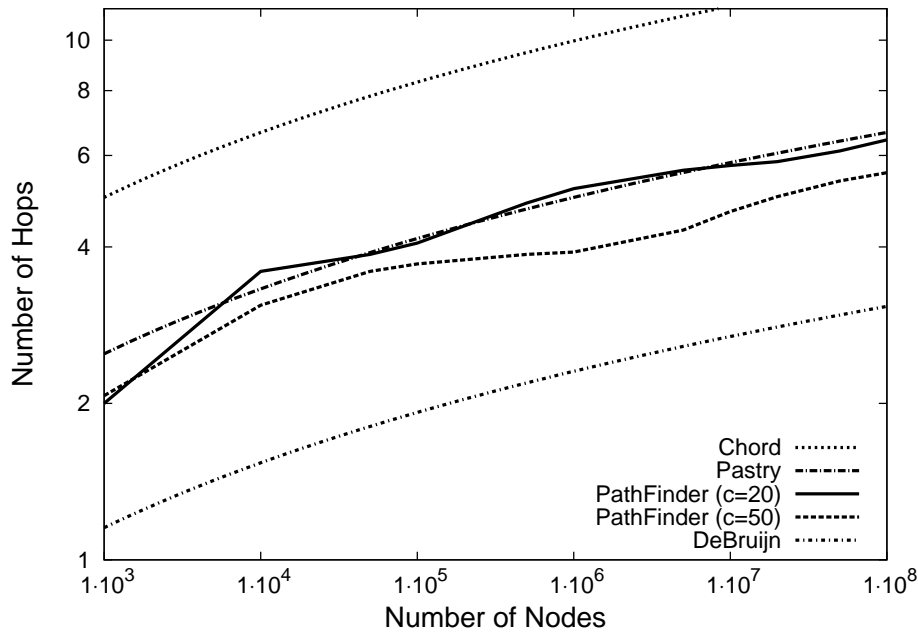


Figure 6.6: Average number of hops for different DHTs measured analytically. The values for PathFinder are from actual simulations.

We also perform an analytical comparison using the equations from the literature summarized in Table 6.1. Our goal is to gain some understanding about how well the different networks scale to hundreds of millions of peers. We compare PathFinder with Pastry and Chord. We ignore Symphony due to its poor performance in the previous experiment and SkipNet due to the lack of well-understood analytical model for its performance. We as well test a DeBruijn graph, because they are known to have optimal diameter.

Note that the PathFinder results come from **actual simulation**, not analytical calculations. For the other overlays we have to resort to analytical modeling in order to estimate scalability for network sizes over 10^6 peers. Figure 6.6 displays the results. The x-axis shows the system size and the y-axis shows the average path length. As expected, Chord's performance is clearly poorer than that of Pastry and PathFinder. Pastry and PathFinder are very similar in performance for $c = 20$. Rising c to 50 gives PathFinder a similar to Pastry neighbors tables and yields about 1 hop less in systems over 100 million nodes. The line for the DeBruijn graph shows the ultimate possible shortest path for the PathFinder network with $c = 20$. PathFinder needs only a bit over 1 hop longer.

To summarize, with respect to average path length PathFinder performs very similar and at least as good as other known DHTs. In terms of scalability it benefits from the small and fix number of neighbors per peer. Even networks of up to several millions peers do perform well with just 20 neighbors on average.

The major difference between PathFinder and other DHTs is that instead of following a routing protocol, the peers have to perform **only local computations** to acquire a path to other peers in the overlay. That is exactly what makes PathFinder a modern and highly competitive overlay. It keeps the communication flow to a minimum, but takes advantage of the computational power

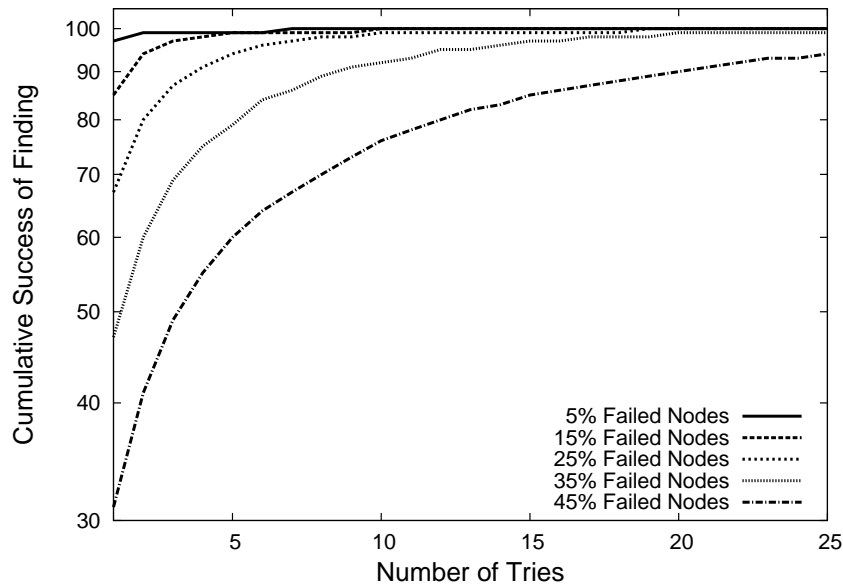


Figure 6.7: Required lookup retries between each couple of nodes under churn.

available in modern computers. The few Megabytes of storage media and hashtable lookups of a few hundreds of thousands of integers, see Section 6.3.4, is already negligible for any regular notebook.

Keep in mind that PathFinder is the first DHT that also supports exhaustive search queries.

Exhaustive Search

PathFinder also inherits the exhaustive search mechanism of BubleStorm. Hence, as an unstructured overlay it performs identical to BubleStorm and the reader is referred to [119] for thorough comparison to other unstructured systems.

6.5 Resilience Against Failures

High churn rates [140] are common in P2P networks. Therefore, alternative paths may be needed to find a particular node. This is where PathFinder benefits from its random network topology. There are always as many alternative disjoint routes between any two nodes as the minimum of their degrees [134].

The challenge then is: How difficult is it to find a valid alternative path? Note that a peer *A* is not aware if there is a failed node on its path to peer *B*. It is first when *A* tries to reach *B* when *A* notices that it has to search for an alternative path. Due to the high number of alternative independent paths (paths which have only common start and end node) between each two nodes, the number of the required retries is very small.

We evaluate the performance of PathFinder under churn by generating a network of 50,000 virtual nodes and then consequently failing different fraction of them. Then we perform a key lookup using the procedure from Section 6.3.4 between each pair of remained nodes. For each pair we count the number of retries we have to make to get from the one node to the other. The results are shown in Figure 6.7.

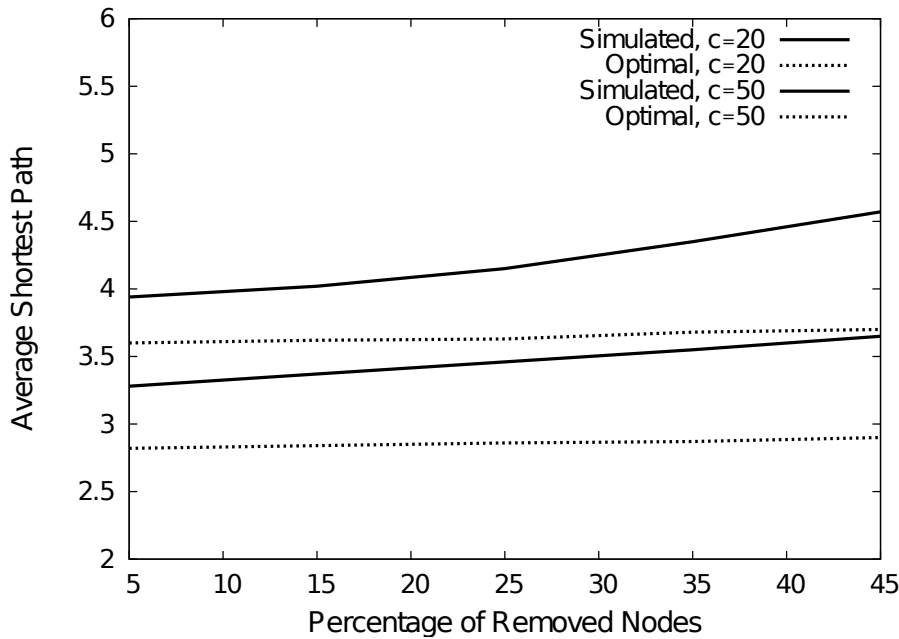


Figure 6.8: Increase of the average shortest path length under churn.

One observes that 5 retries are sufficient to connect over 90% of the remaining node pairs even when 25% of all nodes in the network have failed. In case that almost half of the nodes have failed, then 12 retries lead to successful lookup in 80% of the cases. In both cases the success rate is excellent. Note that in our tests we perform no maintenance in the overlay. After repairing all failed virtual nodes the number of retries drops back down to zero.

We have just shown that between each pair of nodes in our network there are enough alternative independent paths. Only a few attempts are necessary even under massive node crashes. However, there is still one crucial question remaining: How does the average path length change with the number of failed nodes? We know from Section 6.3.4 that the computational effort per peer is very sensitive to the average path length among nodes.

Figure 6.8 displays the average shortest path length for $N = 50,000$ and different values of c . The x-axis shows the fraction of failed nodes and the y-axis shows the average shortest path length. One observes that the average shortest path length increases only marginally, even when half of the nodes suddenly disappear.

From Section 6.3 we know that key lookups not always follow the shortest paths. Therefore, we also evaluated the average lookup length under 25% node failure for $N = 50,000$ and $c = 20$. The maximum number of required hops merely increased from 6 to almost 7.

In short, the average path length and the number of required retries for key lookups in Path-Finder stays stable even for severe fractions of failed nodes. Such a robust resilience is more than desirable in any P2P overlay.

6.6 Security and Other Issues

In terms of security, PathFinder faces the same challenges as most of the P2P overlays presented in the literature. The peers get their node IDs as a hash of their IP address, which is the same as in other DHTs. Note that an attacker with access to a large pool of IP addresses may place herself/himself in a strategic position and discard or alter messages on their way to the destination. This is a common weakness of all DHTs.

If a malicious peer drops messages that are routed through it, the sending peer will eventually notice that because it does not get a reply. Recall from Section 6.5 that there are as many node-disjoint paths between the sender and the receiver as the minimum of their degrees. Thus, the sender can simply try again using a different path. The attacker is unlikely to be present on all the alternative paths.

A simple approach for detecting malicious peers is always to send a message along two disjoint paths. Comparing the results will immediately indicate if the original message or the reply have been modified. Given the short average path length of PathFinder, a second or a third parallel request barely impose additional network load. The Kademia overlay for example sends parallel lookup requests by default.

One approach for handling malicious peers deleting information stored on them is to provide replicas, whereby an object is always handled by several independent peers. In PathFinder one can use an approach similar to Tapestry [128], namely a constant sequence of salt when hashing the object names.

Because PathFinder builds an overlay network with randomly selected neighbors, it is very likely that the virtual nodes do not match well with the underlying IP network. That is, first neighbors in the overlay may be very remotely placed actual peers and messages among them have to travel significant physical distances. Other DHTs suffer from the same problem as well, since their routing algorithms also require contacting arbitrary peers in the network.

DHT routing is typically optimized by selecting neighbors with small round trip times (RTT), with the goal of reducing the overall path latency. A similar approach is applicable in PathFinder as well. When a peer needs to route a message in PathFinder, it computes the shortest path as in Section 6.3.4. If the peer notices that another of its neighbors has *considerably* shorter RTT than the “correct” peer, it can send the message to that neighbor. This may increase the path length in terms of hops as well as the required computational effort per peer, but might reduce the latency. An evaluation of this scheme is part of ongoing work.

6.7 Summary and Outlook

Throughout the previous Chapters we have shown that local structures reveal a clear relation between dynamics and topology in complex networks. Furthermore, we successfully applied them in online topology control schemes, providing approaches for fair load distribution in P2P and resilient P2P live-streaming outperforming state of the art methods.

In this Chapter we have revealed that the potential of local structures is not limited to their specific architecture only, as in the above demonstrated case studies. Random graphs and their

random local structures have many vital for any technological network properties: short average path length, numerous disjoint paths among nodes, robustness, etc.

We have show that random graphs provide an unique platform for the first P2P overlay that supports both, efficient exhaustive search and DHT similar key lookup within the same overlay.

6.7.1 Summary

In this Chapter we have presented PathFinder, a novel overlay which combines efficient exhaustive search and efficient key-value lookups in the same overlay. Combining these two mechanisms in the same overlay is important, since it allows efficient and overhead-free implementation of natural usage patterns.

PathFinder is the first P2P overlay to combine exhaustive search and key-value lookups in an efficient manner.

Because PathFinder is based on a random graph, we directly benefit from the existing search mechanism of BubbleStorm to enable efficient exhaustive search. On the other side, PathFinder utilizes two PRNGs to provide DHT similar key-value lookups.

Our results show that PathFinder has at least comparable and often even better performance than established P2P overlays. Its key lookup performance in large networks outperforms existing DHTs. Furthermore, it scales easily to hundreds of millions of nodes, while keeping the state per peer independent of the network size. In contrast to other P2P overlays, the routing mechanism of the system is able to expand respectively shrinks according to the size of the physical network, allowing one to carefully balance between efficiency and reliability. Still, routing is based on local computations only and no additional communication overhead is required.

Furthermore, the randomized structure of the underlying network provides excellent robustness against failures, keeping the system efficient even under severe churn rates.

Last but not least, the numerous disjoint paths among the peers allow for effective mechanisms against malicious parties.

In short, randomized local structures and the random graphs they build have let us design PathFinder. A novel P2P overlay which clearly outperforms established P2P overlays with respect to several criteria, shown here through both, exhaustive simulations and analytical means.

6.7.2 Outlook

To this end, PathFinder faces several issues common for all existing P2P overlays.

Peer communication is not position aware, i.e. first neighbors in the overlay may be distributed over several continents. Resolving that problem, will shorten latency and minder unnecessary router load and bandwidth usage.

PathFinder has effective guarding mechanisms against malicious parties, determined to disturb the overall network operation. Still, massive attacks by multiple malicious parties towards the same peer, even though very hard to accomplish, are still theoretically possible. Additional guarding mechanism in such cases will improve the resilience of PathFinder even further.

Our novel overlay provides all the functionality required in a P2P overlay, but unlike other existing overlays, it does not consider secondary objectives. These include fair network load among the peers. Making the system aware of the peer load can reduce latency and as well increase efficiency. It is an interesting question, whether the topology control tools developed in the previous chapters can be applied here as well.

Despite the above described open issues, PathFinder already possesses many outstanding properties and advantages over established overlays. A natural next step is to implement an end-user application that takes PathFinder out from the research labs and introduces it in the field of widely spread real world P2P systems.



7 Summary and Outlook

In the following we shortly summarize the results presented throughout this work. We also point out various directions for future work of scientific interest.

7.1 Summary

A major scientific contribution of this work is the revealed, and so far unexplored, interplay between the performance of complex networks and their local structures. More precisely, the direct relation between the motif content and the output pattern of complex networks.

Network motifs are a well-defined intermediate scale for characterizing the local structure of networks beyond the scope of single nodes. Multiple times throughout this work we have shown that the dynamic performance of various complex networks directly depends on the number and types of motifs within these networks.

We have engaged networks motifs from two different aspects. First, as an analytical tool to better describe and understand already emerged real world networks. Second, to develop distributed topology optimization methods for technological systems.

We have applied network motifs as an analytical tool on two large co-authorship networks. Our analysis revealed that there is one collaboration pattern more successful than all others: the box motif. The box motif has the highest citation index per motif edge than all other motifs.

The structure of the box motif induces a certain degree of segregation. Through a series of experiments, we have been able to high extent to relate the success of the box motif to separation of its edges either in time, in rank or in scientific discipline.

Inspired by our findings, we have explored a novel perspective on network motifs. Instead of using them as a static analytical tool, we have engaged them in active topology control mechanisms. Our guiding principle was that there is a direct relation between the motif content of a given network and the dynamic processes taking place on top of that network. Then, by steering its motif content to a desired state, one should be able to control the dynamic performance of the network in a distributed manner.

The first distributed topology control mechanism that we have developed uses motifs to assure fair load balancing in structured P2P networks. It produces none or negligible messaging overhead, while successfully repairing skewed key space and degree distributions. Our novel mechanism is easy to deploy and does not alter the overlay layout nor its operation.

Next, we have tackled a more sophisticated subclass of complex networks. Namely, heterogeneous networks where nodes play different roles for the network operation. We have successfully extended our distributed topology control mechanism to heterogenous P2P overlays. Consequently, we have developed a novel approach for constructing resilient P2P live-streaming networks. Our new approach induces resilience competitive to state of the art methods. More importantly, our method requires no network knowledge, making it much faster than already

established methods. In the same time, it also provides much higher privacy to the participating peers, rendering attacks by malicious parties practically impossible. In that sense, our approach clearly outperforms the state of the art.

The so far presented results explore only one side of the relation between dynamic processes on top of complex networks and their local topology. Next, we have investigated the reverse perspective. Namely, whether it is possible to deploy a suitable dynamic process on a network with no global knowledge in order to reveal its topology. More precisely, to determine critical topological constellations within the network.

We have indeed successfully deployed extended gossiping protocol to detect communication bottlenecks in a distributed manner. Our novel approach clearly outperforms state of the art methods with respect to both, the precision of its results and its performance. Evenly important for distributed applications, our approach has an effective guarding mechanism against malicious parties trying to skew the protocol operation.

Up to this point we have shown that specific local structures lead to specific dynamic performance of the underlying network and vice versa. Finally, we have investigated a slightly orthogonal perspective and have shown that random graphs and their random local structures still have unexploited potential. Although they are poor null-models of real world networks, random graphs have many outstanding properties. Most of them are highly desirable in any technological network.

We have introduced a novel P2P overlay based on random graphs. It is extremely scalable and very efficient, and performs at least as good as already established P2P overlays. More importantly, the introduced overlay is the first overlay to support both, exhaustive search queries and key-value lookups within the same overlay.

To summarize, in this work we have repeatedly shown that exploring networks on intermediate scale opens a so far unexplored perspective on complex networks. We have transferred that perspective to various technological networks, resulting in numerous novel approaches for distributed topology control, competitive or even better than state of the art methods.

Nevertheless, this work has just barely scratched the surface of this new research direction, leaving behind many important scientific questions unexplored, yet.

7.2 Outlook

Despite the various findings and results presented in this work, it is far from fully exploring the potential of local structure analysis in complex networks. On the contrary. In the following we give at least a few further research directions, which are worth exploring with respect to the general methodology as well as in our particular case studies.

General Methodology

In this work we have revealed a relation between the dynamic performance of a network and its local structures, i.e. its motif content. To steer the performance of a given network we have deliberately altered its motif content. It is worth investigating how this strategy performs when

one tries to control two, three or even more network properties simultaneously. Especially in the case when some of those properties are competitive to each other. For example low node degree and low node inter-dependencies. Low node degrees means longer average shortest paths, as a path between two nodes usually takes several intermediate hops. In the same time, long shortest paths means higher node inter-dependencies, as the communication between two nodes depends on all intermediate nodes on the path between them.

Another open question is whether there is a strong correlation between particular motif content and specific network properties. In other words, if one is to quantify a set of networks into families according to a given network property, e.g. error tolerance, how similar are the motif contents of networks within the same families? If such relation indeed exists, one would be able to estimate global network properties by just looking at the motif content of the networks. Not only is that significantly less computationally demanding, but it also can be achieved in a distributed manner. Furthermore, one would be able to easily compare the properties of different networks by just comparing their motif contents. It will not be necessary to perform computationally demanding and time consuming experiments.

A general relation between particular network properties and specific motif content may open another research direction. Constructing a network with a desired property becomes equivalent to constructing a network with a particular motif content. It is a priori not clear, how computationally demanding such an algorithm will be, but it is undoubtedly worth investigating.

In that context, for a given number of nodes and edges, one would be able to construct the whole space of networks with the same number of nodes and edges but different motif contents. In that way, one would be able to estimate to what extent networks with similar motif contents differ in some other rationale, e.g. average shortest path, clustering, etc. Having that, one would be able to project real world networks into that network space and estimate how the properties of those real world networks could possibly change.

The above listed open questions are far from trivial. Confirming or rejecting the above hypothesis will in both cases improve our understanding of complex networks in general and in particular of the networks around us.

Concrete Application Scenarios

The methodology advocated throughout this work is based on a few different case studies. In the following we give a short overview of the remaining open questions within those concrete application scenarios.

We have revealed that the box motif is the most successful collaboration pattern in co-authorship networks, measured as the average citation frequency per motif edge. To better understand the social factors leading to this phenomenon we have introduced an analytical model for constructing co-authorship networks. Our model replicates the dynamic process of publishing and citing of scientific publications, while incorporating proximity, aging, impact etc.

However, one could use that model beyond the simple purpose of verification. Through our model one could observe the network evolution. More precisely, the changes in the motif content as the network evolves. In that sense, one would not only be able to see the outcome of

the network evolution, but also to investigate how the success of the different collaboration patterns changes over time. Ultimately, one could use our method to derive predictions which collaboration patterns are going to be successful in the future.

In the context of the box motif, one could also investigate its role in a more generic class of production and distribution systems. That is, whether the box motif edges comply with the theory of *weak links*. In social networks it has been shown that the most valuable information is transferred along only occasionally used acquaintances and not along everyday-based collaborations, as one would expect. Those occasional acquaintances are called the weak links and have been proven crucial for the communication flow within social networks. So far we have only preliminary, but already promising results concerning the box motif within trust networks.

In this work we also have introduced a new approach for constructing resilient live-streaming topologies. Resilience was the primary objective. However, for such applications it also crucial to decrease the end-to-end delay. To achieve that one could incorporate location awareness to efficiently explore the underlying infrastructure.

Another interesting direction for further development is to augment our approach with an upper bound for signal delivery. In other words, to assure that no participating party experiences delay in signal beyond some acceptable threshold. For that purpose the edges in the network, representing signal exchanges, have to be augmented with realistic delay values. Then, the method should be extended to weighted graphs. The motif based decisions rules should be adapted from binary (an edge is either there or not) to weighted, i.e. the edge weights have to be incorporated into the motif ratios.

Although the idea seems straightforward, only a thorough investigation will reveal if there are some unexpected complications.

In this work we also have presented BridgeFinder, a novel approach for detecting communication bottlenecks in a distributed manner. As we have shown earlier, it is augmented with a guarding mechanism against malicious parties, trying to screw the protocol operation.

Although they are very hard to deploy, attacks by multiple malicious parties working together still represent a threat to BridgeFinder. More precisely, to casual participating parties. A group of malicious parties may surround a targeted peer. Then they can claim that this peer is misbehaving and try to ban it from the network. One could engage trusted authorities and reputation mechanism to detect false claims by third parties. The challenging task here will be to keep intact the distributed nature of BridgeFinder.

Finally, we have presented a novel P2P overlay: PathFinder. It supports both, exhaustive search queries and exact key-value lookups. PathFinder performs at least as good as established overlays, but is the first overlay to combine these two operations within the same overlay.

For a commercial deployment there is still one open issue that needs to be addressed: Peer communication is not position aware. Direct neighbors in the overlay may be distributed over several continents. Under realistic circumstances that could lead to higher latency just because any piece of exchanged information has to travel significant physical distances. Resolving that problem will shorten latency and reduce unnecessary router load and bandwidth usage.

Many distributed real world applications face the same issue and a whole range of optimization techniques has already been developed. The challenge here will be to select the proper

technique to effectively relieve the communication traffic caused by PathFinder, while keeping the deterministic nature of its neighbor-selecting mechanism.

Final Words

At first sight this work seems to have raised more questions than it may have answered. However, one should keep in mind that it has presented unexpected findings on social networks, more precisely co-authorship networks, despite the immense body of research dedicated to them.

Furthermore, this work also advocates a new, and so far unexplored, perspective on complex networks: The motif content of a network is related to its output pattern. It is exactly this new perspective that has motivated a series of novel distributed approaches. Each one of them addresses known problems in an innovative manner and is highly competitive to state-of-the-art methods in their respective fields.

All this together simply confirms the fact that this work is just a first step in a new promising research direction, which is far from being fully explored.



Bibliography

- [1] S. Milgram, “The small world problem,” *Psychology Today*, vol. 2, pp. 60–67, 1967.
- [2] I. de S. Pool and M. Kochen, “Contacts and influence,” *Social Networks*, vol. 1, pp. 1–48, 1978.
- [3] P. Erdős and A. Rényi, “On random graphs,” *Publicationes Mathematicae*, vol. 6, pp. 290–297, 1959.
- [4] B. Bollobás, *Random Graphs*. New York: Cambridge University Press, 2 ed., 2001.
- [5] D. J. Watts and S. H. Strogatz, “Collective dynamics of ‘small-world’ networks,” *Nature*, vol. 393, pp. 440–442, 1998.
- [6] A.-L. Barabási and R. Albert, “Emergence of scaling in random networks,” *Science*, vol. 286, pp. 509–512, 1999.
- [7] A.-L. Barabási, R. Albert, and H. Jeong, “Mean-field theory of scale-free random networks,” *Physica A*, vol. 272, pp. 173–187, 1999.
- [8] A.-L. Barabási, R. Albert, and H. Jeong, “Scale-free characteristics of random networks: The topology of the world wide web,” *Physica A*, vol. 281, pp. 69–77, 2000.
- [9] L. A. N. Amaral, A. Scala, M. Barthélémy, and H. E. Stanley, “Classes of small-world networks,” *PNAS*, vol. 97, pp. 11149–11152, 2000.
- [10] A.-L. Barabási, R. Albert, H. Jeong, and G. Bianconi, “Power-law distribution of the world wide web,” *Science*, vol. 287, p. 2115, 2000.
- [11] M. E. J. Newman, “The structure of scientific collaboration networks,” *PNAS USA*, vol. 98, pp. 404–409, 2001.
- [12] S. Redner, “How popular is your paper? an empirical study of the citation distribution,” *Eur. Phys. J. B*, vol. 4, pp. 131–134, 1998.
- [13] M. E. J. Newman, “Assortative mixing in networks,” *Phys. Rev. Lett.*, vol. 89, no. 20, p. 208701, 2002.
- [14] C. P. Wasserman and K. Faust, *Social Network Analysis*. Cambridge: Cambridge University Press, 1 ed., 1994.
- [15] M. Girvan and M. E. J. Newman, “Community structure in social and biological networks,” *PNAS USA*, vol. 99, pp. 7821–7826, 2002.
- [16] M. E. J. Newman, “Modularity and community structure in networks,” *PNAS USA*, vol. 103, pp. 8577–8582, 2006.

-
- [17] A. Clauset, M. E. J. Newman, and C. Moore, “Finding community structure in very large networks,” *Phys. Rev. E*, vol. 70, p. 066111, 2004.
- [18] M. E. J. Newman, “Finding community structure in networks using the eigenvectors of matrices,” *Phys. Rev. E*, vol. 74, p. 036104, 2006.
- [19] J. Duuch and A. Arenas, “Community detection in complex networks using extremal optimization,” *Phys. Rev. E*, vol. 72, p. 027104, 2005.
- [20] M. Tasgin, A. Herdagdelen, and H. Bingol, “Community detection in complex networks using genetic algorithms,” *arXiv:cond-mat/0604419*, 2008.
- [21] C-Houghton, “Finding community structures in networks by playing pass-the-parcel,” <http://hdl.handle.net/2262/26359>, 2008.
- [22] L. Freeman, “A set of measures of centrality based upon betweenness,” *Sociometry*, vol. 40, pp. 35–41, 1977.
- [23] U. Brandes, “A faster algorithm for betweenness centrality,” *Journal of Mathematical Sociology*, vol. 25, pp. 163–177, 2001.
- [24] U. Brandes, “On variations of shortest-path betweenness centrality and their generic computation,” *Social Networks*, 2008.
- [25] R. Milo, S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii, and U. Alon, “Network motifs: Simple building blocks of complex networks,” *Science*, vol. 298, no. 5594, pp. 824–827, 2002.
- [26] U. Alon, “Network motifs: theory and experimental approaches,” *Nature Reviews Genetics*, vol. 8, pp. 450–461, 2007.
- [27] O. Brandman and T. Meyer, “Feedback loops shape cellular signals in space and time,” *Science*, vol. 322, pp. 390–395, 2008.
- [28] S. Pigolotti, S. Krishna, and M. Jensen, “Oscillation patterns in negative feedback loops,” *PNAS*, vol. 104, pp. 6533–6537, 2007.
- [29] S. Shen-Orr, R. Milo, S. Mangan, and U. Alon, “Network motifs in the transcriptional regulation network of *escherichia coli*,” *Nature Genetics*, vol. 31, p. 688, 2002.
- [30] O. Brandman, J. E. Ferrell, R. Li, and T. Meyer, “Interlinked fast and slow positive feedback loops drive reliable cell decisions,” *Science*, vol. 310, p. 496, 2005.
- [31] R. Milo, S. Itzkovitz, N. Kashtan, R. Levitt, S. Shen-Orr, I. Ayzenshtat, M. Sheffer, and U. Alon, “Superfamilies of evolved and designed networks,” *Science*, vol. 303, no. 5663, pp. 1538–1542, 2004.
- [32] K. Klemm and S. Bornholdt, “Topology of biological networks and reliability of information processing,” *PNAS*, vol. 102, pp. 18414–18419, 2005.
- [33] P. Kaluza, M. Ipsen, M. Vingron, and A. S. Mikhailov, “Design and statistical properties of robust functional networks: A model study of biological signal transduction,” *Phys. Rev. E*, vol. 75, p. 015101, 2007.

-
- [34] P. Kaluza, M. Vingron, and A. S. Mikhailov, “Self-correcting networks: Function, robustness, and motif distributions in biological signal processing,” *Chaos*, vol. 18, p. 026113, 2008.
- [35] P. Kaluza and A. Mikhailov, “Evolutionary design of functional networks robust against noise,” *Europhys. Lett.*, vol. 79, p. 48001, 2007.
- [36] Y.-K. Kwon and K.-H. Cho, “Quantitative analysis of robustness and fragility in biological networks based on feedback dynamics,” *Bioinformatics*, vol. 24, pp. 987–994, 2008.
- [37] DBLP, “Available online,” <http://dblp.uni-trier.de>, Online Search Engine.
- [38] CiteSeerX, “Available online,” <http://citeseerx.ist.psu.edu>, Online Search Engine.
- [39] Google Scholar, “Available online,” <http://scholar.google.com>, Online Search Engine.
- [40] T. S. Kuhn, *The Structure of Scientific Revolutions*. Chicago: University of Chicago Press, 1 ed., 1962.
- [41] A. Arenas, A. Diaz-Guilera, and C. J. Perez-Vicente, “Synchronization reveals topological scales in complex networks,” *Phys. Rev. Lett.*, vol. 96, p. 114102, 2006.
- [42] S. Bornholdt, “Less is more in modeling large genetic networks,” *Science*, vol. 310, p. 449, 2005.
- [43] M. Müller-Linow, C. Hilgetag, and M.-T. Hütt, “Organization of excitable dynamics in hierarchical biological networks,” *PLoS Comput. Biology*, vol. 4, p. e1000190, 2008.
- [44] C. Marr and M.-T. Hütt, “Outer-totalistic cellular automata on graphs,” *Phys. Lett. A*, vol. 373, pp. 546–549, 2009.
- [45] N. M. Luscombe, M. M. Babu, H. Yu, M. Snyder, S. A. Teichmann, and M. Gerstein, “Genomic analysis of regulatory network dynamics reveals large topological changes,” *Nature*, vol. 431, pp. 308–312, 2004.
- [46] M. J. Herrgard, M. W. Covert, and B. O. Palsson, “Reconciling gene expression data with known genome-scale regulatory network structures,” *Genome Research*, vol. 13, no. 11, pp. 2423–2434, 2003.
- [47] C. Marr, M. Geertz, M.-T. Hütt, and G. Muskhelishvili, “Dissecting the logical types of network control in gene expression profiles,” *BMC Systems Biology*, vol. 2, p. 18, 2008.
- [48] R. Pastor-Satorras and A. Vespignani, “Epidemic spreading in scale-free networks,” *Phys. Rev. Lett.*, vol. 86, pp. 3200–3203, 2001.
- [49] R. Albert, H. Jeong, and A.-L. Barabási, “Error and attack tolerance of complex networks,” *Nature*, vol. 406, p. 378, 2000.
- [50] M. E. J. Newman, “Coauthorship networks and patterns of scientific collaboration,” *PNAS*, vol. 101, no. 1, pp. 5200–5205, 2004.
- [51] M. Rosvall and C. T. Bergstrom, “Maps of random walks on complex networks reveal community structure,” *PNAS*, vol. 105, no. 4, pp. 1118–1123, 2008.

-
- [52] L. C. Freeman, “Centrality in social networks conceptual clarification,” *Social Networks*, vol. 1, pp. 215–239, 1978.
- [53] A. F. J. Vanraan, “Fractal dimension of co-citations,” *Nature*, vol. 347, no. 6294, p. 626, 1990.
- [54] P. O. Larsen and M. von Ins, “Lotka’s law, co-authorship and interdisciplinary publishing,” *WIS*, 2008.
- [55] S. Wuchty, B. F. Jones, and B. Uzzi, “The increasing dominance of teams in production of knowledge,” *Science*, vol. 316, no. 5827, pp. 1036–1039, 2007.
- [56] J. Bollen, H. V. de Sompel, A. Hagberg, L. Bettencourt, R. Chute, M. A. Rodriguez, and L. Balakireva, “Clickstream data yields high-resolution maps of science,” *PLoS ONE*, vol. 4, no. 3, p. 4803, 2009.
- [57] R. Guimerà, B. Uzzi, J. Spiro, and L. A. N. Amaral, “Team assembly mechanisms determine collaboration network structure and team performance,” *Science*, vol. 308, no. 5722, pp. 697–702, 2005.
- [58] A. L. Barabási, H. Jeong, Z. Néda, E. Ravasz, A. Schubert, and T. Vicsek, “Evolution of the social network of scientific collaborations,” *Physica A*, vol. 311, no. 3, pp. 590–614, 2002.
- [59] K. Börner, J. T. Maru, and R. L. Goldstone, “The simultaneous evolution of author and paper networks,” *PNAS*, vol. 101, no. 1, pp. 5266–5273, 2004.
- [60] J. J. Ramasco, S. N. Dorogovtsev, and R. Pastor-Satorras, “Self-organization of collaboration networks,” *Phys. Rev. E*, vol. 67, no. 3, p. 036106, 2004.
- [61] M. E. Newman, *Complex Networks*, vol. 650/2004, pp. 337–370. Springer Berlin / Heidelberg, 2004.
- [62] M. E. J. Newman, “Clustering and preferential attachment in growing networks,” *Phys. Rev. E*, vol. 64, no. 2, p. 025102, 2001.
- [63] A. Inzelt, A. Schubert, and M. Schubert, “Incremental citation impact due to international co-authorship in hungarian higher education institutions,” *Scientometrics*, vol. 78, no. 1, pp. 37–43, 2009.
- [64] T. Velden and C. Lagoze, “Patterns of collaboration in co-authorship networks in chemistry - mesoscopic analysis and interpretation,” *ISSI 2009*, 2009.
- [65] R. Albert and A.-L. Barabási, “Statistical mechanics of complex networks,” *Rev. Mod. Phys.*, vol. 74, pp. 47–97, 2002.
- [66] M. E. J. Newman, “The structure and function of complex networks,” *SIAM Review*, vol. 45, p. 167, 2003.
- [67] M. Granovetter, *Getting a job: A study of Contacts and Careers*. Chicago: The University of Chicago Press, 2 ed., 1995.

-
- [68] M. Granovetter, “The strength of weak ties,” *American Journal of Sociology*, vol. 78, no. 6, pp. 1360–1380, 1973.
- [69] S. Goyal and F. Vega-Redondo, “Structural holes in social networks,” *Journal of Economic Theory*, vol. 137, no. 1, pp. 460–492, 2007.
- [70] R. S. Burt, *Structural Holes: The Social Structure of Competition*. Cambridge, MA: Harvard University Press, 1 ed., 1992.
- [71] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, “A scalable content-addressable network,” *SIGCOMM*, pp. 161–172, 2001.
- [72] P. Maymounkov and D. Mazières, “Kademlia: A peer-to-peer information system based on the xor metric,” *LNCS*, pp. 251–260, 2002.
- [73] M. Brinkmeier, M. Fischer, S. Grau, G. Schäfer, and T. Strufe, “Methods for improving resilience in communication networks and p2p overlays,” *PIK*, vol. 32, pp. 64–78, 2009.
- [74] S. Ratnasamy, M. Handley, R. Karp, and S. Shenker, “Topologically-aware overlay construction and server selection,” *INFOCOM*, pp. 1190–1199, 2002.
- [75] M. Waldvogel and R. Rinaldi, “Efficient topology-aware overlay network,” *SIGCOMM*, vol. 33, no. 1, pp. 101–106, 2003.
- [76] D. R. Karger and M. Ruhl, “Simple efficient load-balancing algorithms for peer-to-peer systems,” *Theory of Computing Systems*, vol. 39, no. 6, pp. 787–804, 2006.
- [77] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, “Chord: A scalable peer-to-peer lookup service for internet applications,” *SIGCOMM*, vol. 31, no. 4, pp. 149–160, 2001.
- [78] A. Rao, K. Lakshminarayanan, S. Surana, R. Karp, and I. Stoica, “Load balancing in structured p2p systems,” *LNCS*, vol. 2735, pp. 68–79, 2003.
- [79] J. Byers, J. Considine, and M. Mitzenmacher, “Simple load balancing for distributed hash tables,” *LNCS*, vol. 2735, pp. 80–87, 2003.
- [80] K. P. Gummadi, R. J. Dunn, S. Saroiu, S. D. Gribble, H. M. Levy, and J. Zahorjan, “Measurement, modeling, and analysis of a peer-to-peer file-sharing workload,” *SOSP*, pp. 314–329, 2003.
- [81] K. Graffi, A. Kovacevic, S. Xiao, and R. Steinmetz, “Skyeye. kom: An information management over-overlay for getting the oracle view on structured p2p systems,” *ICPADS*, pp. 279–286, 2008.
- [82] B. Godfrey, K. Lakshminarayanan, S. Surana, R. Karp, and I. Stoica, “Load balancing in dynamic structured p2p systems,” *INFOCOM*, pp. 2253–2262, 2004.
- [83] F. Dabek, M. F. Kaashoek, D. Karger, R. Morris, and I. Stoica, “Wide-area cooperative storage with cfs,” *SOPS*, pp. 202–215, 2001.
- [84] P. García, G. Pairot, R. Mondéjar, J. Pujol, H. Tejedor, and R. Rallo, “Planetsim: A new overlay network simulation framework,” *SEM*, pp. 123–137, 2005.

-
- [85] D. Stutzbach and R. Rejaie, "Understanding churn in peer-to-peer networks," *SIGCOMM*, pp. 189–202, 2006.
- [86] M. Brinkmeier, G. Schäfer, and T. Strufe, "Optimally dos resistant p2p topologies for live multimedia streaming," *TPDS*, vol. 20, pp. 831–834, 2009.
- [87] T. Small, B. Liang, and B. Li, "Scaling laws and tradeoffs in peer-to-peer live multimedia streaming," *Proc. 14th annual ACM international conference on Multimedia*, pp. 539–548, 2006.
- [88] S. Annapureddy, C. Gkantsidis, and P. Rodriguez, "Providing video-on-demand using peer-to-peer networks," *IPTV Workshop, WWW*, 2006.
- [89] K. Graffi, S. Kaune, K. Pussep, A. Kovacevic, and R. Steinmetz, "Load balancing for multimedia streaming in heterogeneous peer-to-peer systems," *NOSSDAV*, 2008.
- [90] Y. Chu, S. Rao, S. Seshan, and H. Zhang, "A case for end system multicast," *JSAC*, vol. 20, no. 8, pp. 1456–1471, 2002.
- [91] V. Pai, K. Kumar, K. Tamilmani, V. Sambamurthy, and A. E. Mohr, "Chainsaw: Eliminating trees from overlay multicast," *IPTPS*, pp. 127–140, 2005.
- [92] D. Carra, R. L. Cigno, and E. W. Biersack, "Graph-based analysis of mesh overlay streaming systems," *JSAC*, vol. 25, no. 9, pp. 1667–1677, 2007.
- [93] J. A. Pouwelse, P. Garbacki, J. Wang, A. Bakker, J. Yang, A. Iosup, D. H. J. Epema, M. Reinders, M. R. van Steen, and H. J. Sips, "Tribler: a social-based peer-to-peer system," *Concurrency and Computation: Practice and Experience*, vol. 20, no. 2, pp. 127–138, 2008.
- [94] D. Carra, G. Neglia, and P. Michiardi, "On the impact of greedy strategies in bittorrent networks: the case of bittorrent," *8th IEEE International Conference on Peer-to-Peer Computing*, pp. 311–320, 2008.
- [95] A. Sentinelli, G. Marfia, M. Gerla, L. Kleinrock, and S. Tewari, "Will iptv ride the peer-to-peer stream?," *Communications Magazine*, pp. 86–92, 2007.
- [96] D. Nguyen, T. Tran, T. Pham, and V. Le, "Internet media streaming using network coding and path diversity," *IEEE Globecom*, 2008.
- [97] S. Banerjee, S. Lee, B. Bhattacharjee, and A. Srinivasan, "Resilient multicast using overlays," *ACM SIGMETRICS Performance Evaluation Review*, vol. 31, pp. 102–113, 2003.
- [98] V. N. Padmanabhan and K. Sripanidkulchai, "The case for cooperative networking," *IPTPS*, vol. 2429, pp. 178–190, 2002.
- [99] S. Grau, S. Fischer, M. Brinkmeier, and M. Schaefer, "On complexity and approximability of optimal dos attacks on multiple-tree p2p streaming topologies," *TDSC*, vol. 99, 2010.
- [100] W. Wang, Y. Xiong, Q. Zhang, and S. Jamin, "Ripple-stream: Safeguarding p2p streaming against dos attacks," *IEEE International Conference on Multimedia and Expo*, pp. 1417–1420, 2006.

-
- [101] J. Yang, Y. Li, B. Huang, and J. Ming, "Preventing dos attacks based on credit model for p2p streaming system," *Proc. 5th international conference on Autonomic and Trusted Computing*, pp. 13–20, 2008.
- [102] W. G. Conner, K. Nahrstedt, and I. Gupta, "Preventing dos attacks in peer-to-peer media streaming systems," *Annual SPIE/ACM Conference on Multimedia Computing and Networking (MMCN)*, 2006.
- [103] M. Rossberg, T. Strufe, and G. Schäfer, "Using recurring costs for reputation management in peer-to-peer streaming systems," *SecureComm*, 2007.
- [104] M. Castro, P. Druschel, A. Kermarrec, A. Nandi, A. Rowstron, and A. Singh, "Splitstream: High-bandwidth multicast in cooperative environments," *19th ACM. Symposium on Operating System Principles (SOSP)*, pp. 298–313, 2003.
- [105] J. Liang and K. Nahrstedt, "Dagstream: Locality aware and failure resilient peer-to-peer streaming," *Multimedia Computing and Networking*, vol. 6071, pp. 1–15, 2006.
- [106] T. Strufe, G. Schäfer, and A. Chang, "Bcbs: An efficient load balancing strategy for cooperative overlay live-streaming," *Proc. IEEE ICC*, 2006.
- [107] S. Birrer, D. Lu, F. E. Bustamante, Y. Qiao, and P. Dinda, "Fatnemo: Building a resilient multi-source multicast fat-tree," *Proc. 9th International Workshop on Web Content Caching and Distribution*, 2004.
- [108] D. Kempe, A. Dobra, and J. Gehrke, "Gossip-based computation of aggregate information," *44th Annual IEEE Symposium on Foundations of Computer Science (FOCS03)*, pp. 482–491, 2003.
- [109] M. Sheng, J. Li, and Y. Shi, "Critical nodes detection in mobile ad hoc network," *Advanced Information Networking and Applications*, vol. 2, pp. 336–340, 2006.
- [110] X. Liu, L. Xiao, A. Kreling, and Y. Liu, "Optimizing overlay topology by reducing cut vertices," *Internatoinal Workshop on Network and Operating Systems Support for Digital Audio and Video*, pp. 1–6, 2006.
- [111] R. Wattenhofer and A. Zollinger, "Xtc: A practical topology control algorithm for ad-hoc networks," *Proceedings of the 18th International Parallel and Distributed Processing Symposium*, 2004.
- [112] G. Sabidussi, "The centrality index of a graph," *Psychometrika*, vol. 31, pp. 581–603, 1966.
- [113] C. Dangalchev, "Residual closeness in networks," *Physica A*, vol. 365, no. 2, pp. 556–564, 2006.
- [114] F. Kuhn, T. Moscibroda, and R. Wattenhofer, "Unit disk graph approximation," *Proceedings of the 2004 Joint Workshop on Foundations of Mobile Computing*, pp. 17–23, 2004.
- [115] A. Jardosh, E. M. Belding-Royer, K. C. Almeroth, and S. Suri, "Towards realistic mobility models for mobile ad hoc networks," *Proceedings of the 9th annual international conference on Mobile computing and networking*, pp. 217–229, 2003.

-
- [116] W. Terpstra and C. L. and A. P. Buchmann, “Practical summation via gossip,” *Proc. 26th Annual ACM Symposium on Principles of Distributed Computing*, pp. 390–391, 2007.
- [117] D. Kempe, J. Kleinberg, and A. Demers, “Spatial gossip and resource location protocols,” *Journal of the ACM (JACM)*, vol. 51, no. 6, pp. 943–967, 2004.
- [118] A. Demers, D. Greene, C. Hauser, W. Irish, J. Larson, S. Shenker, H. Sturgis, D. Swinehart, and D. Terry, “Epidemic algorithms for replicated database maintenance,” *Proceedings of the Sixth Annual ACM Symposium on Principles of Distributed Computing*, pp. 1–12, 1987.
- [119] W. Terpstra, C. Leng, and A. P. Buchmann, “Bubblestorm: Resilient, probabilistic, and exhaustive peer-to-peer search,” *Proc. SIGCOMM*, pp. 49–60, 2007.
- [120] P. T. Eugster, R. Guerraoui, A. M. Kermarrec, and L. Massoulié, “From epidemics to distributed computing,” *IEEE Computer*, vol. 37, no. 5, pp. 60–67, 2004.
- [121] S. Saroiu, K. P. Gummadi, and S. D. Gribble, “Measuring and analyzing the characteristics of napster and gnutella hosts,” *Multimedia Systems*, vol. 9, no. 2, pp. 170–184, 2003.
- [122] T. Qiu, E. Chan, and G. Chen, “Overlay partition: Iterative detection and proactive recovery,” *IEEE International Conference on Communications 2007 (ICC)*, pp. 1854–1859, 2007.
- [123] R. Steinmetz and K. Wehrle, *Peer-To-Peer Systems and Applications*. Heidelberg: Springer, 1 ed., 2005.
- [124] M. F. Kaashoek and D. R. Karger, “Koorde: A simple degree-optimal distributed hash table,” *Proc. 2nd International Workshop on Peer-to-Peer Systems (IPTPS03)*, pp. 98–103, 2003.
- [125] F. Dabek, B. Y. Zhao, P. Druschel, J. Kubiawicz, and I. Stoica, “Towards a common api for structured peer-to-peer overlays,” *Proc. 2nd International Workshop on Peer-to-Peer Systems (IPTPS03)*, pp. 33–44, 2003.
- [126] D. Malkhi, M. Naor, and D. Ratajczak, “Viceroy: A scalable and dynamic emulation of the butterfly,” *Proceedings of the 21st ACM Symposium on Principles of Distributed Computing*, pp. 183–192, 2002.
- [127] A. Rowstron and P. Druschel, “Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems,” *IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, pp. 329–350, 2001.
- [128] B. Zhao, J. Kubiawicz, and A. Joseph, “Tapestry: An infrastructure for fault-tolerant wide-area location and routing,” *Computer*, vol. 74, pp. 11–20, 2001.
- [129] C. G. Plaxton, R. Rajaraman, and A. W. Rich, “Accessing nearby copies of replicated objects in a distributed environment,” *Theory of Computing Systems*, vol. 32, no. 3, pp. 241–280, 1999.
- [130] G. S. Manku, M. Bawa, and P. Raghavan, “Symphony: Distributed hashing in a small world,” *Proc. 4th USENIX Symposium on Internet Technologies and Systems*, pp. 127–140, 2003.

-
- [131] Y. Yang, R. Dunlap, M. Rexroad, and B. F. Cooper, "Performance of full text search in structured and unstructured peer-to-peer systems," *Proc. IEEE INFOCOM*, pp. 1–12, 2006.
- [132] J. Li, B. T. Loo, J. Hellerstein, F. Kaashoek, D. R. Karger, and R. Morris, "On the feasibility of peer-to-peer web indexing and search," *Proc. 2nd International Workshop on Peer-to-Peer Systems (IPTPS)*, pp. 20–21, 2003.
- [133] P. Reynolds and A. Vahdat, "Efficient peer-to-peer keyword searching," *Middleware*, pp. 21–40, 2003.
- [134] C. Greenhill, F. B. Holt, and N. Wormald, "Expansion properties of a random regular graph after random vertex deletions," *European Journal of Combinatorics*, vol. 29, pp. 1139–1150, 2008.
- [135] H. Barton, "Skype users online now," <http://idisk.mac.com/hhbv-Public/OnlineNow.htm>, 2009.
- [136] L. Zhuang and F. Zhou, "Understanding chord performance," *Technical Report CS268*, 2003.
- [137] D. Loguinov, A. Komar, V. Rai, and S. Ganesh, "Graph-theoretic analysis of structured peer-to-peer systems: routing distances and fault resilience," *Proc. Conference on Applications, technologies, architectures, and protocols for computer communications*, pp. 395–406, 2003.
- [138] M. G. Hluchyj and M. J. Karol, "Shuffle net: An application of generalized perfect shuffles to multihop lightwave networks," *Lightwave Technology*, vol. 9, no. 10, pp. 1386–1397, 1991.
- [139] N. J. A. Harvey, M. B. Jones, S. Saroiu, M. Theimer, and A. Wolman, "Skipnet: A scalable overlay network with practical locality properties," *Proc. of the 4th USENIX Symposium on Internet Technologies and Systems*, 2003.
- [140] S. Saroiu, K. Gummadi, and S. Gribble, "A measurement study of peer-to-peer file sharing systems," *Proceedings of Multimedia Computing and Networking (MMCN)*, 2002.



8 Authors Publications

[1] Lachezar Krumov, Christoph Fretter, Matthias Müller-Hannemann, Karsten Weihe, Marc-Thorsten Hütt, “Motifs in co-authorship networks and their relation to the impact of scientific publications”, *Submitted to E. P. J. B (the European Physical Journal B)*, 2011, see Chapter 2.

[2] Dirk Bradler, Lachezar Krumov, Max Mühlhäuser and Jussi Kangasharju, “BridgeFinder: Finding Communication Bottlenecks in Distributed Environments”, *Submitted to ICOIN 2011*, 2011, see Chapter 5.

[3] Dirk Bradler, Lachezar Krumov, Jussi Kangasharju and Max Mühlhäuser, “PathFinder: Efficient Lookups and Efficient Search in Peer-to-Peer Networks”, *ICDCN 2011*, 2011, see Chapter 6.

[4] Lachezar Krumov, Immanuel Schweizer, Dirk Bradler and Thorsten Strufe, “Leveraging Network Motifs for the Adaption of Structured Peer-to-Peer Networks”, *GlobeCom 2010*, 2010, see Chapter 3.

[5] Lachezar Krumov, Adriana Andreeva and Thorsten Stufe, “Resilient Peer-to-Peer Live-Steaming Using Motifs”, *WoWMoM 2010*, 2010, see Chapter 4.

[6] Christoph Fretter, Lachezar Krumov, Karsten Weihe, Matthias Müller-Hannemann, Marc-Thorsten Hütt, “Phase Synchronization in Railway Timetables”, *E. P. J. B (the European Physical Journal B)*, 2010.

[7] Dirk Bradler, Lachezar Krumov, Michael Wagner and Jussi Kangasharju, “Hierarchical Data Access in Structured P2P-Networks”, *SpringSim/CNS 2009*, 2009, see Chapter 6.

[8] Lachezar Krumov, “Degree and Diameter Bounded Minimum Spanning Trees”, Diploma Thesis, *Technical University Darmstadt*, October 2007.



List of Figures

1.1	The eight possible undirected three- and four-node motifs.	18
1.2	The traditional client-server architecture and a P2P architecture.	19
2.1	The average edge weight per motif compared to the null model for DBLP (A) and CiteSeerX (B), according to edge weight definition from eqs. (1) and (3), respectively. In order to resolve the data behind the averages from (A) and (B), the cumulative distributions of the edge weights for two of the motifs are shown, namely the box motif (motif 6) and motif 4, for DBLP (C) and CiteSeerX (D). . . .	28
2.2	The average link weight per motif in DBLP for all four edge weight definitions compared to the shuffled null model denoted by SH.	29
2.3	The average link weight per motif in CiteSeerX for all four edge weight definitions compared to the shuffled null model denoted by SH.	29
2.4	Ratio of average edge weights real data/null model for the edge weight definitions 1, 2, 3 and 4, DBLP on the left side and CiteSeerX on the right side.	30
2.5	The average weight per motif link over the years for the DBLP database.	31
2.6	(A) The eight possible undirected three- and four-node motifs. (B) Example of a single occurrence of motif 6 (box motif) based on only four publications and embedded in the local network generated by these publications.	31
2.7	Percentage of box motif instances in DBLP where the top two authors are connected directly. The box motifs instances are divided in chunks of 1000 instances and sorted in descending order with respect to their weight.	32
2.8	Relative average edge weight per motif. All motif instances are distributed in bins according to their creation time and the average weight per bin is displayed. . . .	33
2.9	Average number of shortest paths passing through a motif edge for the 1990 snapshot of the DBLP (all publications dating before or from 1990).	34
2.10	Degree distributions of DBLP and CiteSeerX.	35
2.11	Citation distributions of DBLP and CiteSeerX.	35
2.12	The motif edge weight distributions for all eight motifs within the DBLP database.	36
2.13	The effect on the average motif edge weight when one gradually removes the heaviest instances of that motif.	37
2.14	The number of papers respectively co-authors per motif edge for DBLP.	38
2.15	Schematic representation of the content proximity plane. Distance among authors and publications enters the computation of the scores, equations 2.5 and 2.6.	39
2.16	Approximating the DBLP snapshot from 1990. Once with respect to degree distribution, citation distribution and motif content only, and once augmented with the ratio in weight of motif 4 to motif 6.	41
2.17	Approximating the degree distribution of the DBLP snapshot from 1990. Once with respect to topological properties only and once augmented with the ratio in weight of motif 4 to motif 6.	42

2.18	Approximating the citation distribution of the DBLP snapshot from 1990. Once with respect to topological properties only and once augmented with the ratio in weight of motif 4 to motif 6.	42
2.19	Approximating the motif content of the DBLP snapshot from 1990. Once with respect to topological properties only and once augmented with the ratio in weight of motif 4 to motif 6.	43
3.1	Network motifs: (un)directed subgraphs of 3/4 nodes.	50
3.2	A suboptimal and optimal CAN topologies with 15 nodes.	53
3.3	System Architecture of MBO	55
3.4	Probability of occurrence of scope in CAN with and without MBO.	59
3.5	Indegree distribution Kademia (ECD with 500 nodes)	61
3.6	Characteristic path length under perfect attack (500 nodes).	62
4.1	Example of an optimal topology, $N = 37$, $n_{max} = 4$ (cmp. [86]).	69
4.2	Motif ratio phase transition of an optimal topology.	70
4.3	Successor exchange operations.	72
4.4	Number of exchange operations per node per stripe.	73
4.5	Sample tree with corresponding ToPo metric values.	74
4.6	ToPo metric values of streaming topologies (with std. deviation). Inset: the ToPo metric results of the motif approach normalized with respect to the network size.	74
4.7	Height of streaming topologies (with standard deviation). Inset: topology height of the motif approach normalized with respect to the network size.	75
4.8	Balance metric values of streaming topologies normalized with respect to the network size (with standard deviation).	76
4.9	Average node connectivity of streaming topologies (with std. deviation).	77
4.10	Stability of streaming topologies (with standard deviation).	78
5.1	Betweenness does not always reflect the central role of a node. The displayed values represent the betweenness coefficients of the corresponding nodes.	86
5.2	Phases of the BridgeFinder algorithm and interleaving among multiple runs.	89
5.3	Three different obstacle scenarios. Examples show 1, 2, and 3 bridges respectively.	89
5.4	Destroying networks by removing the fastest converging nodes.	90
5.5	Intersecting best centrality measure nodes with 5% of the fastest converging nodes.	91
5.6	Average speed coefficient within 100 runs of BridgeFinder on all four network types	96
5.7	Perfect attack at every 5th iteration of BridgeFinder with 10% of malicious peers	97
6.1	A small example of the PathFinder overlay.	107
6.2	Key lookup with local expanding search rings from both the source and the target.	108
6.3	Distribution of complete path length for 5000 key lookups with $c = 20$	109
6.4	Repair costs for network with 5,000 peers	112
6.5	Average number of hops for 5,000 key lookups in different DHTs.	115
6.6	Average number of hops for different DHTs measured analytically. The values for PathFinder are from actual simulations.	116
6.7	Required lookup retries between each couple of nodes under churn.	117
6.8	Increase of the average shortest path length under churn.	118

List of Tables

2.1	Average number of authors per paper, papers per author and clustering coefficients for the DBLP and CiteSeerX databases. All values comply with results on co-authorship networks from related work.	36
2.2	Expected applications of the box motif in diverse technological and social networks.	47
3.1	Initial and target motif signatures, Φ -Score and SP_{Φ} for CAN.	53
3.2	Initial and target motif signatures, Φ -Score and SP_{Φ} for Kademia.	55
3.3	Join Process CAN: Number of messages for CAN with MBO compared to original CAN.	57
3.4	Kademia compared to Kademia with MBO with respect to maintenance messages, necessary lookups and average lookup length.	60
5.1	Convergence speed on different network types measured in average number of exchange steps per node.	93
6.1	Comparison of various DHTs to PathFinder.	114



List of Algorithms

1	Topology Control	71
2	BridgeFinder Exchange Operation	88
3	PathFinder Neighbor List Construction	106



Index

- ALM, 68
- anti-clustering, 31
- application domain, 84
- application layer multicast, 68
- assortative, 17
- assortativity, 17
- attack, 61, 67, 73
- attacker, 67
- attacks, 95
- author score, 39
- average betweenness, 85, 91

- balance, 73
- Balance metric, 73, 76
- bandwidth, 71
- betweenness, 85
 - edge, 17
- betweenness centrality, 85
- bitrate, 67
- bootstrap node, 54
- box motif, 21, 31, 33, 34, 37
- bridge, 83
- BridgeFinder, 83
- bucket size, 54

- CAN, 49
- chunk, 68
- churn, 57
- citation frequency, 23, 27
- closeness, 87
- clustering, 16
- co-authorship networks, 23
- complex query, 101
- construction time, 33
- control, 71
- converge, 72, 73
- convergence, 79, 97
- correction mechanism, 68
- critical peers, 83, 84

- damage, 67
- data source, 67
- dependency, 68
- DHT, 101–104, 114
- disassortative, 17
- distributed algorithm, 83
- dynamic process, 83

- edge
 - betweenness, 34
 - initiation, 33
 - weight, 27
- encoding, 68
- error forward, 68
- exchange operation, 71, 73
- exhaustive search, 101, 106
- expansion, 92

- forward/backward chaining, 109

- gossiping, 83
 - algorithm, 84
- graph, 15
- guarding mechanism, 95

- hub-nodes, 61

- impact, 27
- interdependence, 66
- intermediate nodes, 69
- internal nodes, 69

- join, 110

- Kademlia, 49
- key lookup, 101
 - lookup, 108
- key space, 51

- leaves, 70
- live streaming, 65
- lookup, 101, 103, 108

- maintenance, 101
- malicious nodes, 83
- malicious parties, 95
- management overhead, 73
- mobile networks, 83
- modularity, 17
- motif, 49
 - based optimization, 49
 - frequency, 51
 - signature, 51
- motifs, 18
- multihop networks, 89
- multimedia content, 67

- network motifs, 18
- node crash, 111

Node Manager, 71

optimality, 79

overhead, 73

overlay, 101

- structured, 101
- unstructured, 101

P2P, 19

packet stream, 67

peer-to-peer (P2P), 19

perfect attack, 97

PlanetSim, 57

power law, 16

predecessor, 68, 72

- selection, 68

preferential attachment, 16

PRNG, 105

pro-active optimization, 68

pull-based system, 68

push-based system, 68

random graph, 16, 101

reputation mechanism, 68

resilience, 66, 68, 73, 79

robustness, 66, 101

scale free network, 16

score, 52

small-world effect, 15

speed coefficient, 95

square closeness, 87, 91

stability, 78, 80

streaming, 66

stripe, 67, 68

success, 27, 33, 38

successor, 72

target motif signature, 50

target significance profile, 52

target topology, 52

ToPo metric, 73

topology

- adaptation, 49
- control, 71
- optimization, 49
- quality, 73

torus, 53

transfer, 72

transitivity, 16

tree height, 73, 75

UDG, 89

unit disk graph, 89

vertex connectivity, 73, 77

weak links, 126

XOR metric, 54

zipf distribution, 51

Glossary

N	the number of nodes in a graph (network)
M	the number of edges in a graph (network)
p	the probability that two nodes are connected
$G_{N,p}$	the set of all graph instances of N nodes connected with probability p
ER	Erdős-Rényi random graph
$P_v(k)$	the probability that a node v has degree k
P2P	peer-to-peer
MBO	motif based optimization
F_I	the number of all different k -node motif instances within a topology I
SP_Φ	the target significance profile of a network
Θ	the multiple dimensional torus representing the key space of CAN
CAN	content addressable network (a structured P2P overlay)
PlanetSim	a framework for P2P overlay simulations
V	the set of nodes in a graph (network)
E	the set of edges in a graph (network)
S	the packet stream in a live streaming system
R_0	the bitrate in a live streaming system
θ	the motif ratio threshold (directed motif 1 to motif 3)
NM	the Node Manager of each peer within the overlay network
n_{max}	the maximum number of neighbors (outgoing plus ingoing in undirected graphs)
d	the maximum number of direct successors
BCBS	an efficient load balancing strategy for cooperative overlay live streaming
DHT	Distributed Hash Tables, used for key lookups in P2P overlays
PathFinder	a novel P2P overlay that combines both: exhaustive search and key lookups
PRNG	a pseudo random number generator
BridgeFinder	a novel distributed approach for detecting communication bottlenecks
$C_B(v)$	the betweenness centrality of a node v
$\sigma_{st}(v)$	the number of shortest paths from a node s to a node t going through the node v
$C_{AB}(v)$	the average betweenness of a node v

-
- $C_c(v)$ the closeness centrality of a node v
- $C_{sqc}(v)$ the square closeness of a node v
- UDG Unit Disk Graph
- ODG Obstacle Disk Graph
- ϑ neighborhood threshold in unit disk graphs
- s_v the speed coefficient of a node v in BridgeFinder
- $P(e)$ the set of publications represented by the edge e
- $c(p)$ the citation frequency of the publication p
- $A(p)$ the set of authors of the publication p
- w_e the weight of the edge e

Wissenschaftlicher Werdegang des Verfassers²

09/2001 – 4/2007	Studium Mathematics with Computer Science Technische Universität Darmstadt
09/2001 – 10/2007	Studium der Mathematik, Nebenfach Informatik Technische Universität Darmstadt
4/2007	Abschluss: Bachelor of Science Bachelorarbeitsthema: <i>Approximation Algorithm for the Minimum Degree Spanning Tree Problem</i>
10/2007	Abschluss: Diplom-Mathematiker Diplomarbeitsthema: <i>Degree and Diameter Bounded Minimum Spanning Trees</i>
seit 11/2007	Wissenschaftlicher Mitarbeiter am Fachbereich Informatik, Fachgebiet Algorithmik Technische Universität Darmstadt

² gemäß §20 Abs. 3 der Promotionsordnung der TU Darmstadt
